



Intel® MAX® 10 Device Handbook - Combined

This auto-generated document contains the following user guides. To download individual standalone documents, click on the respective PDF/HTML links.

- Intel® MAX® 10 FPGA Device Overview ([HTML](#) | [PDF](#))
- Intel® MAX® 10 FPGA Device Datasheet ([HTML](#) | [PDF](#))
- Intel® MAX 10 Device Errata ([HTML](#) | [PDF](#))
- Intel® MAX® 10 Device Family Pin Connection Guidelines ([HTML](#) | [PDF](#))
- Intel® MAX® 10 FPGA Design Guidelines ([HTML](#) | [PDF](#))
- Intel® MAX® 10 FPGA Device Architecture ([HTML](#) | [PDF](#))
- Intel® MAX® 10 Development Kit User Guide ([HTML](#) | [PDF](#))
- Intel® MAX® 10 FPGA Configuration User Guide ([HTML](#) | [PDF](#))
- Intel® MAX® 10 Power Management User Guide ([HTML](#) | [PDF](#))
- Intel® MAX® 10 Embedded Memory User Guide ([HTML](#) | [PDF](#))
- Intel® MAX® 10 Embedded Multipliers User Guide ([HTML](#) | [PDF](#))
- Intel® MAX® 10 Clocking and PLL User Guide ([HTML](#) | [PDF](#))
- Intel® MAX® 10 JTAG Boundary-Scan Testing User Guide ([HTML](#) | [PDF](#))
- Intel® MAX® 10 General Purpose I/O User Guide ([HTML](#) | [PDF](#))
- Intel® MAX® 10 Analog to Digital Converter User Guide ([HTML](#) | [PDF](#))

MAX[®] 10 FPGA Device Overview



Online Version

Send Feedback

M10-OVERVIEW

683658

2022.06.14

Contents

Intel® MAX® 10 FPGA Device Overview.....	3
Key Advantages of Intel MAX 10 Devices.....	3
Summary of Intel MAX 10 Device Features	4
Intel MAX 10 Device Ordering Information.....	5
Intel MAX 10 Device Feature Options.....	6
Intel MAX 10 Device Maximum Resources	6
Intel MAX 10 Devices I/O Resources Per Package	7
Intel MAX 10 Vertical Migration Support.....	7
Intel MAX 10 I/O Vertical Migration Support.....	8
Intel MAX 10 ADC Vertical Migration Support.....	8
Logic Elements and Logic Array Blocks.....	9
Analog-to-Digital Converter.....	9
User Flash Memory.....	10
Embedded Multipliers and Digital Signal Processing Support.....	10
Embedded Memory Blocks.....	11
Clocking and PLL.....	11
FPGA General Purpose I/O.....	12
External Memory Interface.....	12
Configuration.....	13
Power Management.....	13
Document Revision History for Intel MAX 10 FPGA Device Overview.....	13

Intel® MAX® 10 FPGA Device Overview

Intel® MAX® 10 devices are single-chip, non-volatile low-cost programmable logic devices (PLDs) to integrate the optimal set of system components.

The highlights of the Intel MAX 10 devices include:

- Internally stored dual configuration flash
- User flash memory
- Instant on support
- Integrated analog-to-digital converters (ADCs)
- Single-chip Nios II soft core processor support

Intel MAX 10 devices are the ideal solution for system management, I/O expansion, communication control planes, industrial, automotive, and consumer applications.

Related Information

[Intel MAX 10 FPGA Device Datasheet](#)

Key Advantages of Intel MAX 10 Devices

Table 1. Key Advantages of Intel MAX 10 Devices

Advantage	Supporting Feature
Simple and fast configuration	Secure on-die flash memory enables device configuration in less than 10 ms
Flexibility and integration	<ul style="list-style-type: none"> • Single device integrating PLD logic, RAM, flash memory, digital signal processing (DSP), ADC, phase-locked loop (PLL), and I/Os • Small packages available from 3 mm × 3 mm
Low power	<ul style="list-style-type: none"> • Sleep mode—significant standby power reduction and resumption in less than 1 ms • Longer battery life—resumption from full power-off in less than 10 ms
20-year-estimated life cycle	Built on TSMC's 55 nm embedded flash process technology
High productivity design tools	<ul style="list-style-type: none"> • Intel Quartus® Prime Lite edition (no cost license) • Platform Designer (Standard) system integration tool • DSP Builder for Intel FPGAs • Nios® II Embedded Design Suite (EDS)

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.

**ISO
9001:2015
Registered**

Summary of Intel MAX 10 Device Features

Table 2. Summary of Features for Intel MAX 10 Devices

Feature	Description
Technology	55 nm TSMC Embedded Flash (Flash + SRAM) process technology
Packaging	<ul style="list-style-type: none"> Low cost, small form factor packages—support multiple packaging technologies and pin pitches Multiple device densities with compatible package footprints for seamless migration between different device densities RoHS6-compliant
Core architecture	<ul style="list-style-type: none"> 4-input look-up table (LUT) and single register logic element (LE) LEs arranged in logic array block (LAB) Embedded RAM and user flash memory Clocks and PLLs Embedded multiplier blocks General purpose I/Os
Internal memory blocks	<ul style="list-style-type: none"> M9K—9 kilobits (Kb) memory blocks Cascadable blocks to create RAM, dual port, and FIFO functions
User flash memory (UFM)	<ul style="list-style-type: none"> User accessible non-volatile storage High speed operating frequency Large memory size High data retention Multiple interface option
Embedded multiplier blocks	<ul style="list-style-type: none"> One 18 × 18 or two 9 × 9 multiplier modes Cascadable blocks enabling creation of filters, arithmetic functions, and image processing pipelines
ADC	<ul style="list-style-type: none"> 12-bit successive approximation register (SAR) type Up to 17 analog inputs Cumulative speed up to 1 million samples per second (MSPS) Integrated temperature sensing capability
Clock networks	<ul style="list-style-type: none"> Global clocks support High speed frequency in clock network
Internal oscillator	Built-in internal ring oscillator
PLLs	<ul style="list-style-type: none"> Analog-based Low jitter High precision clock synthesis Clock delay compensation Zero delay buffering Multiple output taps
General-purpose I/Os (GPIOs)	<ul style="list-style-type: none"> Multiple I/O standards support On-chip termination (OCT) Up to 720 megabits per second (Mbps) LVDS receiver and transmitter
External memory interface (EMIF) ⁽¹⁾	Supports up to 600 Mbps external memory interfaces: <ul style="list-style-type: none"> DDR3, DDR3L, DDR2, LPDDR2 (on 10M16, 10M25, 10M40, and 10M50.) SRAM (Hardware support only)

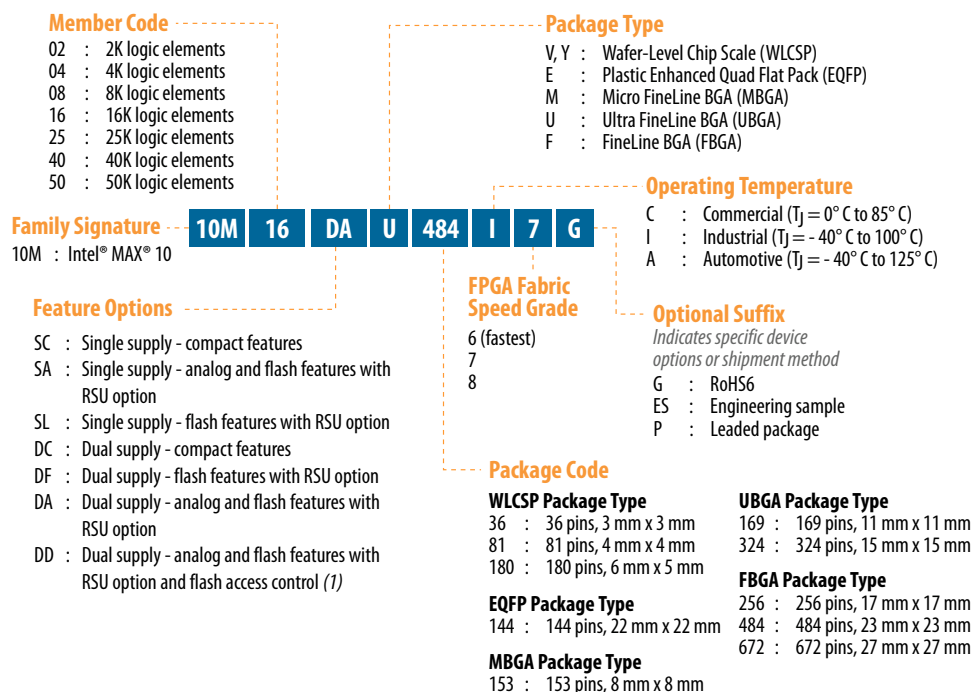
continued...

⁽¹⁾ EMIF is only supported in selected Intel MAX 10 device density and package combinations. Refer to the *External Memory Interface User Guide* for more information.

Feature	Description
	<i>Note:</i> For 600 Mbps performance, -6 device speed grade is required. Performance varies according to device grade (commercial, industrial, or automotive) and device speed grade (-6 or -7). Refer to the <i>Intel MAX 10 FPGA Device Datasheet</i> or <i>External Memory Interface Spec Estimator</i> for more details.
Configuration	<ul style="list-style-type: none"> Internal configuration JTAG Advanced Encryption Standard (AES) 128-bit encryption and compression options Flash memory data retention of 20 years at 85 °C
Flexible power supply schemes	<ul style="list-style-type: none"> Single- and dual-supply device options Dynamically controlled input buffer power down Sleep mode for dynamic power reduction

Intel MAX 10 Device Ordering Information

Figure 1. Sample Ordering Code and Available Options for Intel MAX 10 Devices



Note:

(1) DD OPN available only on 10M40 and 10M50 devices with F256, F484, and F762 packages.

Note: The -A6 speed grade of the Intel MAX 10 FPGA devices is not available by default in the Intel Quartus Prime software. Contact your local Intel sales representatives for support.

Related Information

Intel FPGA Product Selector

Provides the latest information about Intel MAX 10 FPGAs.

Intel MAX 10 Device Feature Options

Table 3. Feature Options for Intel MAX 10 Devices

Option	Feature
Compact	Devices with core architecture featuring single configuration image with self-configuration capability
Flash	Devices with core architecture featuring: <ul style="list-style-type: none"> Dual configuration image with self-configuration capability Remote system upgrade capability Memory initialization
Analog	Devices with core architecture featuring: <ul style="list-style-type: none"> Dual configuration image with self-configuration capability Remote system upgrade capability Memory initialization Integrated ADC

Intel MAX 10 Device Maximum Resources

Table 4. Maximum Resource Counts for Intel MAX 10 Devices

Resource		Device						
		10M02	10M04	10M08	10M16	10M25	10M40	10M50
Logic Elements (LE) (K)		2	4	8	16	25	40	50
M9K Memory (Kb)		108	189	378	549	675	1,260	1,638
User Flash Memory (Kb) ⁽²⁾		96	1,248	1,376	2,368	3,200	5,888	5,888
18 × 18 Multiplier		16	20	24	45	55	125	144
PLL		2	2	2	4	4	4	4
GPIO		246	246	250	320	360	500	500
LVDS	Dedicated Transmitter	15	15	15	22	24	30	30
	Emulated Transmitter	114	114	116	151	171	241	241
	Dedicated Receiver	114	114	116	151	171	241	241
Internal Configuration Image		1	2	2	2	2	2	2
ADC		—	1	1	1	2	2	2

⁽²⁾ The maximum possible value including user flash memory and configuration flash memory. For more information, refer to [Intel MAX 10 User Flash Memory User Guide](#).

Intel MAX 10 Devices I/O Resources Per Package

Table 5. Package Plan for Intel MAX 10 Single Power Supply Devices

Device	Package						
	Type	V81 81-pin WLCSP	Y180 180-pin WLCSP	M153 153-pin MBGA	U169 169-pin UBGA	U324 324-pin UBGA	E144 144-pin EQFP
	Size	4 mm × 4 mm	6 mm × 5 mm	8 mm × 8 mm	11 mm × 11 mm	15 mm × 15 mm	22 mm × 22 mm
	Ball Pitch	0.4 mm	0.35 mm	0.5 mm	0.8 mm	0.8 mm	0.5 mm
10M02		—	—	112	130	246	101
10M04		—	—	112	130	246	101
10M08		58	—	112	130	246	101
10M16		—	125	—	130	246	101
10M25		—	—	—	—	—	101
10M40		—	—	—	—	—	101
10M50		—	—	—	—	—	101

Table 6. Package Plan for Intel MAX 10 Dual Power Supply Devices

Device	Package						
	Type	V36 36-pin WLCSP	V81 81-pin WLCSP	U324 324-pin UBGA	F256 256-pin FBGA	F484 484-pin FBGA	F672 672-pin FBGA
	Size	3 mm × 3 mm	4 mm × 4 mm	15 mm × 15 mm	17 mm × 17 mm	23 mm × 23 mm	27 mm × 27 mm
	Ball Pitch	0.4 mm	0.4 mm	0.8 mm	1.0 mm	1.0 mm	1.0 mm
10M02		27	—	160	—	—	—
10M04		—	—	246	178	—	—
10M08		—	56	246	178	250	—
10M16		—	—	246	178	320	—
10M25		—	—	—	178	360	—
10M40		—	—	—	178	360	500
10M50		—	—	—	178	360	500

Related Information

- [Intel MAX 10 General Purpose I/O User Guide](#)
- [Intel MAX 10 High-Speed LVDS I/O User Guide](#)

Intel MAX 10 Vertical Migration Support



Vertical migration supports the migration of your design to other Intel MAX 10 devices of different densities in the same package with similar I/O and ADC resources.

Intel MAX 10 I/O Vertical Migration Support

Figure 2. Migration Capability Across Intel MAX 10 Devices

- The arrows indicate the migration paths. The devices included in each vertical migration path are shaded. Non-migratable devices are omitted. Some packages have several migration paths. Devices with lesser I/O resources in the same path have lighter shades.
- To achieve the full I/O migration across product lines in the same migration path, restrict I/Os usage to match the product line with the lowest I/O count.

Device	Package									
	V36	V81	Y180	M153	U169	U324	F256	E144	F484	F672
10M02				↑	↑	↑				
10M04				↓	↓	↓	↑	↑		
10M08							↑	↑	↑	
10M16					↓	↓	↑	↑	↑	
10M25							↑	↑	↑	
10M40							↑	↑	↑	↑
10M50							↑	↑	↑	↑

 Dual Power Supply Devices
  Single Power Supply Devices

Note: Before starting migration work, Intel recommends that you verify the pin migration compatibility through the **Pin Migration View** window in the Intel Quartus Prime software Pin Planner. For example, not all Intel MAX 10 devices support 1.0 V I/O.

Intel MAX 10 ADC Vertical Migration Support

Figure 3. ADC Vertical Migration Across Intel MAX 10 Devices

The arrows indicate the ADC migration paths. The devices included in each vertical migration path are shaded.

Device	Package						
	M153	U169	U324	F256	E144	F484	F672
10M04	↑	↑	↑	↑	↑		
10M08	↓	↓	↓	↑	↑	↑	
10M16		↓	↓	↑	↑	↑	
10M25				↑	↑	↑	
10M40				↑	↑	↑	↑
10M50				↑	↑	↑	↑




 **Dual ADC Device:** Each ADC (ADC1 and ADC2) supports 1 dedicated analog input pin and 8 dual function pins.
 **Single ADC Device:** Single ADC that supports 1 dedicated analog input pin and 16 dual function pins.
 **Single ADC Device:** Single ADC that supports 1 dedicated analog input pin and 8 dual function pins.

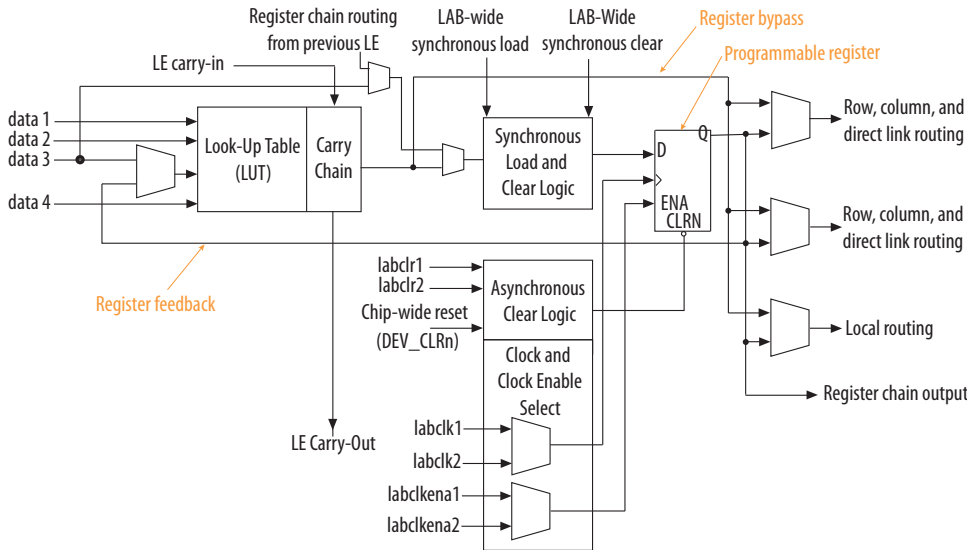
Table 7. Pin Migration Conditions for ADC Migration

Source	Target	Migratable Pins
Single ADC device	Single ADC device	You can migrate all ADC input pins
Dual ADC device	Dual ADC device	
Single ADC device	Dual ADC device	<ul style="list-style-type: none"> One dedicated analog input pin. Eight dual function pins from the ADC1 block of the source device to the ADC1 block of the target device.
Dual ADC device	Single ADC device	

Logic Elements and Logic Array Blocks

The LAB consists of 16 logic elements (LE) and a LAB-wide control block. An LE is the smallest unit of logic in the Intel MAX 10 device architecture. Each LE has four inputs, a four-input look-up table (LUT), a register, and output logic. The four-input LUT is a function generator that can implement any function with four variables.

Figure 4. Intel MAX 10 Device Family LEs



Analog-to-Digital Converter

Intel MAX 10 devices feature up to two ADCs. You can use the ADCs to monitor many different signals, including on-chip temperature.

Table 8. ADC Features

Feature	Description
12-bit resolution	<ul style="list-style-type: none"> Translates analog signal to digital data for information processing, computing, data transmission, and control systems Provides a 12-bit digital representation of the observed analog signal
Up to 1 MSPS sampling rate	Monitors single-ended external inputs with a cumulative sampling rate of 25 kilosamples per second to 1 MSPS in normal mode
<i>continued...</i>	

Feature	Description
Up to 17 single-ended external inputs for single ADC devices	One dedicated analog and 16 dual function input pins
Up to 18 single-ended external inputs for dual ADC devices	<ul style="list-style-type: none"> One dedicated analog and eight dual-function input pins in each ADC block Simultaneous measurement capability for dual ADC devices
On-chip temperature sensor	Monitors external temperature data input with a sampling rate of up to 50 kilosamples per second

User Flash Memory

The user flash memory (UFM) block in Intel MAX 10 devices stores non-volatile information.

UFM provides an ideal storage solution that you can access using Avalon Memory-Mapped (Avalon-MM) slave interface protocol.

Table 9. UFM Features

Features	Capacity
Endurance	Counts to at least 10,000 program/erase cycles
Data retention	<ul style="list-style-type: none"> 20 years at 85 °C 10 years at 100 °C
Operating frequency	Maximum 116 MHz for parallel interface and 7.25 MHz for serial interface
Data length	Stores data up to 32 bits length in parallel

Embedded Multipliers and Digital Signal Processing Support

Intel MAX 10 devices support up to 144 embedded multiplier blocks. Each block supports one individual 18 × 18-bit multiplier or two individual 9 × 9-bit multipliers.

With the combination of on-chip resources and external interfaces in Intel MAX 10 devices, you can build DSP systems with high performance, low system cost, and low power consumption.

You can use the Intel MAX 10 device on its own or as a DSP device co-processor to improve price-to-performance ratios of DSP systems.

You can control the operation of the embedded multiplier blocks using the following options:

- Parameterize the relevant IP cores with the Intel Quartus Prime parameter editor
- Infer the multipliers directly with VHDL or Verilog HDL

System design features provided for Intel MAX 10 devices:

- DSP IP cores:
 - Common DSP processing functions such as finite impulse response (FIR), fast Fourier transform (FFT), and numerically controlled oscillator (NCO) functions
 - Suites of common video and image processing functions
- Complete reference designs for end-market applications
- DSP Builder for Intel FPGAs interface tool between the Intel Quartus Prime software and the MathWorks Simulink and MATLAB design environments
- DSP development kits

Embedded Memory Blocks

The embedded memory structure consists of M9K memory blocks columns. Each M9K memory block of a Intel MAX 10 device provides 9 Kb of on-chip memory capable of operating at up to 284 MHz. The embedded memory structure consists of M9K memory blocks columns. Each M9K memory block of a Intel MAX 10 device provides 9 Kb of on-chip memory. You can cascade the memory blocks to form wider or deeper logic structures.

You can configure the M9K memory blocks as RAM, FIFO buffers, or ROM.

The Intel MAX 10 device memory blocks are optimized for applications such as high throughput packet processing, embedded processor program, and embedded data storage.

Table 10. M9K Operation Modes and Port Widths

Operation Modes	Port Widths
Single port	×1, ×2, ×4, ×8, ×9, ×16, ×18, ×32, and ×36
Simple dual port	×1, ×2, ×4, ×8, ×9, ×16, ×18, ×32, and ×36
True dual port	×1, ×2, ×4, ×8, ×9, ×16, and ×18

Clocking and PLL

Intel MAX 10 devices offer the following resources: global clock (GCLK) networks and phase-locked loops (PLLs) with a 116-MHz built-in oscillator.

Intel MAX 10 devices support up to 20 global clock (GCLK) networks with operating frequency up to 450 MHz. The GCLK networks have high drive strength and low skew.

The PLLs provide robust clock management and synthesis for device clock management, external system clock management, and I/O interface clocking. The high precision and low jitter PLLs offers the following features:

- Reduction in the number of oscillators required on the board
- Reduction in the device clock pins through multiple clock frequency synthesis from a single reference clock source
- Frequency synthesis
- On-chip clock de-skew
- Jitter attenuation
- Dynamic phase-shift

- Zero delay buffer
- Counter reconfiguration
- Bandwidth reconfiguration
- Programmable output duty cycle
- PLL cascading
- Reference clock switchover
- Driving of the ADC block

FPGA General Purpose I/O

The Intel MAX 10 I/O buffers support a range of programmable features.

These features increase the flexibility of I/O utilization and provide an alternative to reduce the usage of external discrete components such as a pull-up resistor and a PCI clamp diode.

External Memory Interface

Dual-supply Intel MAX 10 devices feature external memory interfaces solution that uses the I/O elements on the right side of the devices together with the UniPHY IP.

With this solution, you can create external memory interfaces to 16-bit SDRAM components with error correction coding (ECC).

Note: The external memory interface feature is available only for dual-supply Intel MAX 10 devices.

Table 11. External Memory Interface Performance

External Memory Interface ⁽³⁾	I/O Standard	Maximum Width	Maximum Frequency (MHz)
DDR3 SDRAM	SSTL-15	16 bit + 8 bit ECC	303
DDR3L SDRAM	SSTL-135	16 bit + 8 bit ECC	303
DDR2 SDRAM	SSTL-18	16 bit + 8 bit ECC	200
LPDDR2 SDRAM	HSUL-12	16 bit without ECC	200 ⁽⁴⁾

Related Information

External Memory Interface Spec Estimator

Provides a parametric tool that allows you to find and compare the performance of the supported external memory interfaces in Intel FPGAs.

⁽³⁾ The device hardware supports SRAM. Use your own design to interface with SRAM devices.

⁽⁴⁾ To achieve the specified performance, constrain the memory device I/O and core power supply variation to within $\pm 3\%$. By default, the frequency is 167 MHz.

Configuration

Table 12. Configuration Features

Feature	Description
Dual configuration	<ul style="list-style-type: none"> Stores two configuration images in the configuration flash memory (CFM) Selects the first configuration image to load using the CONFIG_SEL pin
Design security	<ul style="list-style-type: none"> Supports 128-bit key with non-volatile key programming Limits access of the JTAG instruction during power-up in the JTAG secure mode Unique device ID for each Intel MAX 10 device
SEU Mitigation	<ul style="list-style-type: none"> Auto-detects cyclic redundancy check (CRC) errors during configuration Provides optional CRC error detection and identification in user mode
Dual-purpose configuration pin	<ul style="list-style-type: none"> Functions as configuration pins prior to user mode Provides options to be used as configuration pin or user I/O pin in user mode
Configuration data compression	<ul style="list-style-type: none"> Decompresses the compressed configuration bitstream data in real-time during configuration Reduces the size of configuration image stored in the CFM
Instant-on	Provides the fastest power-up mode for Intel MAX 10 devices.

Table 13. Configuration Schemes for Intel MAX 10 Devices

Configuration Scheme	Compression	Encryption	Dual Image Configuration	Data Width
Internal Configuration	Yes	Yes	Yes	—
JTAG	—	—	—	1

Power Management

Table 14. Power Options

Power Options	Advantage
Single-supply device	Saves board space and costs.
Dual-supply device	<ul style="list-style-type: none"> Consumes less power Offers higher performance
Power management controller scheme	<ul style="list-style-type: none"> Reduces dynamic power consumption when certain applications are in standby mode Provides a fast wake-up time of less than 1 ms.

Document Revision History for Intel MAX 10 FPGA Device Overview

Document Version	Changes
2022.06.14	Updated the LVDS receiver and transmitter speeds from 830 Mbps and 800 Mbps, respectively, to 720 Mbps.
2021.11.01	<ul style="list-style-type: none"> Updated the <i>Sample Ordering Code and Available Options for Intel MAX 10 Devices</i> diagram. <ul style="list-style-type: none"> Added SL and DD feature options, Y package type, and 180 package code. Removed -I6 speed grade from contact information. All OPNs for -I6 speed grade are available in the Intel Quartus Prime Standard Edition software version 21.1 onwards. Added V81 and Y180 packages in the <i>Package Plan for Intel MAX 10 Single Power Supply Devices</i> table. Added Y180 package in the <i>Migration Capability Across Intel MAX 10 Devices</i> diagram.

Date	Version	Changes
December 2017	2017.12.15	<ul style="list-style-type: none"> Added the U324 package for the Intel MAX 10 single power supply devices. Updated the 10M02 GPIO and LVDS count in the <i>Maximum Resource Counts for Intel MAX 10 Devices</i> table. Updated the I/O vertical migration figure.
February 2017	2017.02.21	<ul style="list-style-type: none"> Rebranded as Intel.
December 2016	2016.12.20	<ul style="list-style-type: none"> Updated EMIF information in the <i>Summary of Features for Intel MAX 10 Devices</i> table. EMIF is only supported in selected Intel MAX 10 device density and package combinations, and for 600 Mbps performance, -6 device speed grade is required. Updated the device ordering information to include P for leaded package.
May 2016	2016.05.02	<ul style="list-style-type: none"> Removed all preliminary marks. Update the ADC sampling rate description. The ADC feature monitors single-ended external inputs with a cumulative sampling rate of 25 kilosamples per second to 1 MSPS in normal mode.
November 2015	2015.11.02	<ul style="list-style-type: none"> Removed SF feature from the device ordering information figure. Changed instances of <i>Quartus II</i> to <i>Intel Quartus Prime</i>.
May 2015	2015.05.04	<ul style="list-style-type: none"> Added clearer descriptions for the feature options listed in the device ordering information figure. Updated the maximum dedicated LVDS transmitter count of 10M02 device from 10 to 9. Removed the F672 package of the Intel MAX 10 10M25 device : <ul style="list-style-type: none"> Updated the devices I/O resources per package. Updated the I/O vertical migration support. Updated the ADC vertical migration support. Updated the maximum resources for 10M25 device: <ul style="list-style-type: none"> Maximum GPIO from 380 to 360. Maximum dedicated LVDS transmitter from 26 to 24. Maximum emulated LVDS transmitter from 181 to 171. Maximum dedicated LVDS receiver from 181 to 171. Added ADC information for the E144 package of the 10M04 device. Updated the ADC vertical migration diagram to clarify that there are single ADC devices with eight and 16 dual function pins. Removed the note about contacting Altera for DDR3, DDR3L, DDR2, and LPDDR2 external memory interface support. The Intel Quartus Prime software supports these external memory interfaces from version 15.0.
December 2014	2014.12.15	<ul style="list-style-type: none"> Changed terms: <ul style="list-style-type: none"> "dual image" to "dual configuration image" "dual-image configuration" to dual configuration" Added memory initialization feature for Flash and Analog devices. Added maximum data retention capacity of up to 20 years for UFM feature. Added maximum operating frequency of 7.25 MHz for serial interface for UFM feature.
September 2014	2014.09.22	Initial release.



Intel® MAX® 10 FPGA Device Datasheet



Online Version



Send Feedback

M10-DATASHEET

ID: **683794**

Version: **2022.10.31**

Contents

Intel® MAX® 10 FPGA Device Datasheet.....	3
Electrical Characteristics.....	3
Operating Conditions.....	4
Switching Characteristics.....	26
Core Performance Specifications.....	26
Periphery Performance Specifications.....	35
Configuration Specifications.....	59
JTAG Timing Parameters.....	60
Remote System Upgrade Circuitry Timing Specifications.....	61
User Watchdog Internal Circuitry Timing Specifications.....	61
Uncompressed Raw Binary File (.rbf) Sizes.....	61
Internal Configuration Time.....	62
Internal Configuration Timing Parameter.....	63
I/O Timing.....	63
Programmable IOE Delay.....	64
Programmable IOE Delay On Row Pins.....	64
Programmable IOE Delay for Column Pins.....	65
Glossary.....	66
Document Revision History for the Intel MAX 10 FPGA Device Datasheet.....	69

Intel® MAX® 10 FPGA Device Datasheet

This datasheet describes the electrical characteristics, switching characteristics, configuration specifications, and timing for Intel MAX® 10 devices.

Table 1. Intel MAX 10 Device Grades and Speed Grades Supported

Device Grade	Speed Grade Supported
Commercial	<ul style="list-style-type: none">–C7–C8 (slowest)
Industrial	<ul style="list-style-type: none">–I6 (fastest)–I7–I8
Automotive	<ul style="list-style-type: none">–A6–A7

Note: The –A6 speed grade of the Intel MAX 10 FPGA devices is not available by default in the Intel Quartus® Prime software. Contact your local Intel sales representatives for support.

Note: The –I8 speed grade is only applied to 10M04, 10M08, and 10M16 devices.

Related Information

[Device Ordering Information, Intel MAX 10 FPGA Device Overview](#)

Provides more information about the densities and packages of devices in the Intel MAX 10.

Electrical Characteristics

The following sections describe the operating conditions and power consumption of Intel MAX 10 devices.

Operating Conditions

Intel MAX 10 devices are rated according to a set of defined parameters. To maintain the highest possible performance and reliability of the Intel MAX 10 devices, you must consider the operating requirements described in this section.

Absolute Maximum Ratings

This section defines the maximum operating conditions for Intel MAX 10 devices. The values are based on experiments conducted with the devices and theoretical modeling of breakdown and damage mechanisms. The functional operation of the device is not implied for these conditions.

Caution: Conditions outside the range listed in the absolute maximum ratings tables may cause permanent damage to the device. Additionally, device operation at the absolute maximum ratings for extended periods of time may have adverse effects on the device.

Single Supply Devices Absolute Maximum Ratings

Table 2. Absolute Maximum Ratings for Intel MAX 10 Single Supply Devices

Symbol	Parameter	Min	Max	Unit
V _{CC_ONE}	Supply voltage for core and periphery through on-die voltage regulator	-0.5	3.9	V
V _{CCIO}	Supply voltage for input and output buffers	-0.5	3.9	V
V _{CCA}	Supply voltage for phase-locked loop (PLL) regulator and analog-to-digital converter (ADC) block (analog)	-0.5	3.9	V

Dual Supply Devices Absolute Maximum Ratings

Table 3. Absolute Maximum Ratings for Intel MAX 10 Dual Supply Devices

Symbol	Parameter	Min	Max	Unit
V _{CC}	Supply voltage for core and periphery	-0.5	1.63	V
V _{CCIO}	Supply voltage for input and output buffers	-0.5	3.9	V
V _{CCA}	Supply voltage for PLL regulator (analog)	-0.5	3.41	V

continued...

Symbol	Parameter	Min	Max	Unit
V _{CCD_PLL}	Supply voltage for PLL regulator (digital)	-0.5	1.63	V
V _{CCA_ADC}	Supply voltage for ADC analog block	-0.5	3.41	V
V _{CCINT}	Supply voltage for ADC digital block	-0.5	1.63	V

Absolute Maximum Ratings

Table 4. Absolute Maximum Ratings for Intel MAX 10 Devices

Symbol	Parameter	Min	Max	Unit
V _I	DC input voltage	-0.5	4.12	V
I _{OUT}	DC output current per pin	-25	25	mA
T _{STG}	Storage temperature	-65	150	°C
T _J	Operating junction temperature	-40	125	°C

Maximum Allowed Overshoot During Transitions over a 11.4-Year Time Frame

During transitions, input signals may overshoot to the voltage listed in the following table and undershoot to -2.0 V for input currents less than 100 mA and periods shorter than 20 ns.

The maximum allowed overshoot duration is specified as a percentage of high time over the lifetime of the device. A DC signal is equivalent to 100% duty cycle.

For example, a signal that overshoots to 4.17 V can only be at 4.17 V for ~11.7% over the lifetime of the device; for a device lifetime of 11.4 years, this amounts to 1.33 years.

Table 5. Maximum Allowed Overshoot During Transitions over a 11.4-Year Time Frame for Intel MAX 10 Devices

Condition (V)	Overshoot Duration as % of High Time	Unit
4.12	100.0	%
4.17	11.7	%
4.22	7.1	%
4.27	4.3	%
<i>continued...</i>		

Condition (V)	Overshoot Duration as % of High Time	Unit
4.32	2.6	%
4.37	1.6	%
4.42	1.0	%
4.47	0.6	%
4.52	0.3	%
4.57	0.2	%

Recommended Operating Conditions

This section lists the functional operation limits for the AC and DC parameters for Intel MAX 10 devices. The tables list the steady-state voltage values expected from Intel MAX 10 devices. Power supply ramps must all be strictly monotonic, without plateaus.

Single Supply Devices Power Supplies Recommended Operating Conditions

Table 6. Power Supplies Recommended Operating Conditions for Intel MAX 10 Single Supply Devices

Symbol	Parameter	Condition	Min	Typ	Max	Unit
$V_{CC_ONE}^{(1)}$	Supply voltage for core and periphery through on-die voltage regulator	—	2.85/3.135	3.0/3.3	3.15/3.465	V
$V_{CCIO}^{(2)}$	Supply voltage for input and output buffers	3.3 V	3.135	3.3	3.465	V
		3.0 V	2.85	3	3.15	V
		2.5 V	2.375	2.5	2.625	V
		1.8 V	1.71	1.8	1.89	V
		1.5 V	1.425	1.5	1.575	V
		1.35 V	1.2825	1.35	1.4175	V

continued...

(1) V_{CCA} must be connected to V_{CC_ONE} through a filter.

(2) V_{CCIO} for all I/O banks must be powered up during user mode because V_{CCIO} I/O banks are used for the ADC and I/O functionalities.

Symbol	Parameter	Condition	Min	Typ	Max	Unit
		1.2 V	1.14	1.2	1.26	V
		1.0 V	0.95	1.0	1.05	V
$V_{CCA}^{(1)}$	Supply voltage for PLL regulator and ADC block (analog)	—	2.85/3.135	3.0/3.3	3.15/3.465	V

Dual Supply Devices Power Supplies Recommended Operating Conditions

Table 7. Power Supplies Recommended Operating Conditions for Intel MAX 10 Dual Supply Devices

Symbol	Parameter	Condition	Min	Typ	Max	Unit
V_{CC}	Supply voltage for core and periphery	—	1.15	1.2	1.25	V
$V_{CCIO}^{(3)}$	Supply voltage for input and output buffers	3.3 V	3.135	3.3	3.465	V
		3.0 V	2.85	3	3.15	V
		2.5 V	2.375	2.5	2.625	V
		1.8 V	1.71	1.8	1.89	V
		1.5 V	1.425	1.5	1.575	V
		1.35 V	1.2825	1.35	1.4175	V
		1.2 V	1.14	1.2	1.26	V
		1.0 V	0.95	1.0	1.05	V
$V_{CCA}^{(4)}$	Supply voltage for PLL regulator (analog)	—	2.375	2.5	2.625	V
$V_{CCD_PLL}^{(5)}$	Supply voltage for PLL regulator (digital)	—	1.15	1.2	1.25	V
V_{CCA_ADC}	Supply voltage for ADC analog block	—	2.375	2.5	2.625	V
V_{CCINT}	Supply voltage for ADC digital block	—	1.15	1.2	1.25	V

(3) V_{CCIO} for all I/O banks must be powered up during user mode because V_{CCIO} I/O banks are used for the ADC and I/O functionalities.

(4) All V_{CCA} pins must be powered to 2.5 V (even when PLLs are not used), and must be powered up and powered down at the same time.

(5) V_{CCD_PLL} must always be connected to V_{CC} through a decoupling capacitor and ferrite bead.

Recommended Operating Conditions

Table 8. Recommended Operating Conditions for Intel MAX 10 Devices

Symbol	Parameter	Condition	Min	Max	Unit
V _I	DC input voltage	—	−0.5	3.6	V
V _O	Output voltage for I/O pins	—	0	V _{CCIO}	V
T _J	Operating junction temperature	Commercial	0	85	°C
		Industrial	−40 ⁽⁶⁾	100	°C
		Automotive	−40 ⁽⁶⁾	125	°C
t _{RAMP}	Power supply ramp time	—	(7)	10	ms
I _{Diode}	Magnitude of DC current across PCI* clamp diode when enabled	—	—	10	mA

Programming/Erasure Specifications

Table 9. Programming/Erasure Specifications for Intel MAX 10 Devices

This table shows the programming cycles and data retention duration of the user flash memory (UFM) and configuration flash memory (CFM) blocks.

For more information about data retention duration with 10,000 programming cycles for automotive temperature devices, contact your Intel quality representative and quote ID#14015216870.

Erase and reprogram cycles (E/P) ⁽⁸⁾ (Cycles/page)	Temperature (°C)	Data retention duration (Years)
10,000	85	20
10,000	100	10

⁽⁶⁾ −40°C is only applicable to Start of Test, when the device is powered-on. The device does not stay at the minimum junction temperature for a long time.

⁽⁷⁾ There is no absolute minimum value for the ramp time requirement. Intel characterized the minimum ramp time at 200 μs.

⁽⁸⁾ The number of E/P cycles applies to the smallest possible flash block that can be erased or programmed in each Intel MAX 10 device. Each Intel MAX 10 device has multiple flash pages per device.

DC Characteristics

Supply Current and Power Consumption

Intel offers two ways to estimate power for your design—the Excel-based Early Power Estimator (EPE) and the Intel Quartus Prime Power Analyzer feature.

Use the Excel-based EPE before you start your design to estimate the supply current for your design. The EPE provides a magnitude estimate of the device power because these currents vary greatly with the usage of the resources.

The Intel Quartus Prime Power Analyzer provides better quality estimates based on the specifics of the design after you complete place-and-route. The Power Analyzer can apply a combination of user-entered, simulation-derived, and estimated signal activities that, when combined with detailed circuit models, yield very accurate power estimates.

Related Information

- [Early Power Estimator User Guide](#)
Provides more information about power estimation tools.
- [Intel Quartus Prime Standard Edition User Guide: Power Analysis and Optimization](#)
Provides more information about power estimation tools.

I/O Pin Leakage Current

The values in the table are specified for normal device operation. The values vary during device power-up. This applies for all V_{CCIO} settings (3.3, 3.0, 2.5, 1.8, 1.5, 1.35, and 1.2 V).

10 μ A I/O leakage current limit is applicable when the internal clamping diode is off. A higher current can be the observed when the diode is on.

Input channel leakage of ADC I/O pins due to hot socket is up to maximum of 1.8 mA. The input channel leakage occurs when the ADC IP core is enabled or disabled. This is applicable to all Intel MAX 10 devices with ADC IP core, which are 10M04, 10M08, 10M16, 10M25, 10M40, and 10M50 devices. The ADC I/O pins are in Bank 1A.

Table 10. I/O Pin Leakage Current for Intel MAX 10 Devices

Symbol	Parameter	Condition	Min	Max	Unit
I_I	Input pin leakage current	$V_I = 0 \text{ V to } V_{CCIO\text{MAX}}$	-10	10	μA
I_{OZ}	Tristated I/O pin leakage current	$V_O = 0 \text{ V to } V_{CCIO\text{MAX}}$	-10	10	μA

Table 11. ADC_VREF Pin Leakage Current for Intel MAX 10 Devices

Symbol	Parameter	Condition	Min	Max	Unit
I _{adc_vref}	ADC_VREF pin leakage current	Single supply mode	—	10	μA
		Dual supply mode	—	20	μA

Bus Hold Parameters

Bus hold retains the last valid logic state after the source driving it either enters the high impedance state or is removed. Each I/O pin has an option to enable bus hold in user mode. Bus hold is always disabled in configuration mode.

Table 12. Bus Hold Parameters for Intel MAX 10 Devices

Parameter	Condition	V _{CCIO} (V)												Unit
		1.2		1.5		1.8		2.5		3.0		3.3		
		Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	
Bus-hold low, sustaining current	V _{IN} > V _{IL} (maximum)	8	—	12	—	30	—	50	—	70	—	70	—	μA
Bus-hold high, sustaining current	V _{IN} < V _{IH} (minimum)	–8	—	–12	—	–30	—	–50	—	–70	—	–70	—	μA
Bus-hold low, overdrive current	0 V < V _{IN} < V _{CCIO}	—	125	—	175	—	200	—	300	—	500	—	500	μA
Bus-hold high, overdrive current	0 V < V _{IN} < V _{CCIO}	—	–125	—	–175	—	–200	—	–300	—	–500	—	–500	μA
Bus-hold trip point	—	0.3	0.9	0.375	1.125	0.68	1.07	0.7	1.7	0.8	2	0.8	2	V

Series OCT without Calibration Specifications

Table 13. Series OCT without Calibration Specifications for Intel MAX 10 Devices

This table shows the variation of on-chip termination (OCT) without calibration across process, voltage, and temperature (PVT).

Description	V _{CCIO} (V)	Resistance Tolerance		Unit
		-C7, -I6, -I7, -A6, -A7	-C8, -I8	
Series OCT without calibration	3.00	±35	±30	%
	2.50	±35	±30	%
	1.80	±40	±35	%
	1.50	±40	±40	%
	1.35	±40	±50	%
	1.20	±45	±60	%

Series OCT with Calibration at Device Power-Up Specifications

Table 14. Series OCT with Calibration at Device Power-Up Specifications for Intel MAX 10 Devices

OCT calibration is automatically performed at device power-up for OCT enabled I/Os.

Description	V _{CCIO} (V)	Calibration Accuracy	Unit
Series OCT with calibration at device power-up	3.00	±12	%
	2.50	±12	%
	1.80	±12	%
	1.50	±12	%
	1.35	±12	%
	1.20	±12	%

OCT Variation after Calibration at Device Power-Up

The OCT resistance may vary with the variation of temperature and voltage after calibration at device power-up.

Use the following table and equation to determine the final OCT resistance considering the variations after calibration at device power-up.

Table 15. OCT Variation after Calibration at Device Power-Up for Intel MAX 10 Devices

This table lists the change percentage of the OCT resistance with voltage and temperature.

Description	Nominal Voltage	dR/dT (%/°C)	dR/dV (%/mV)
OCT variation after calibration at device power-up	3.00	0.25	-0.027
	2.50	0.245	-0.04
	1.80	0.242	-0.079
	1.50	0.235	-0.125
	1.35	0.229	-0.16
	1.20	0.197	-0.208

Figure 1. Equation for OCT Resistance after Calibration at Device Power-Up

$$\Delta R_V = (V_2 - V_1) \times 1000 \times dR/dV$$

$$\Delta R_T = (T_2 - T_1) \times dR/dT$$

$$\text{For } \Delta R_X < 0; MF_X = 1/(|\Delta R_X|/100 + 1)$$

$$\text{For } \Delta R_X > 0; MF_X = \Delta R_X/100 + 1$$

$$MF = MF_V \times MF_T$$

$$R_{final} = R_{initial} \times MF$$

The definitions for equation are as follows:

- T_1 is the initial temperature.
- T_2 is the final temperature.
- MF is multiplication factor.
- $R_{initial}$ is initial resistance.
- R_{final} is final resistance.

- Subscript x refers to both V and T.
- ΔR_V is variation of resistance with voltage.
- ΔR_T is variation of resistance with temperature.
- dR/dT is the change percentage of resistance with temperature after calibration at device power-up.
- dR/dV is the change percentage of resistance with voltage after calibration at device power-up.
- V_1 is the initial voltage.
- V_2 is final voltage.

The following figure shows the example to calculate the change of 50 Ω I/O impedance from 25°C at 3.0 V to 85°C at 3.15 V.

Figure 2. Example for OCT Resistance Calculation after Calibration at Device Power-Up

$$\Delta R_V = (3.15 - 3) \times 1000 \times -0.027 = -4.05$$

$$\Delta R_T = (85 - 25) \times 0.25 = 15$$

Because ΔR_V is negative,

$$MF_V = 1/(4.05/100 + 1) = 0.961$$

Because ΔR_T is positive,

$$MF_T = 15/100 + 1 = 1.15$$

$$MF = 0.961 \times 1.15 = 1.105$$

$$R_{final} = 50 \times 1.105 = 55.25\Omega$$

Pin Capacitance

Table 16. Pin Capacitance for Intel MAX 10 Devices

Symbol	Parameter	Maximum	Unit
C _{IOB}	Input capacitance on bottom I/O pins	8	pF
C _{IOLRT}	Input capacitance on left/right/top I/O pins	7	pF
C _{LVDSB}	Input capacitance on bottom I/O pins with dedicated LVDS output ⁽⁹⁾	8	pF
C _{ADCL}	Input capacitance on left I/O pins with ADC input ⁽¹⁰⁾	9	pF
C _{VREFLRT}	Input capacitance on left/right/top dual purpose V _{REF} pin when used as V _{REF} or user I/O pin ⁽¹¹⁾	48	pF
C _{VREFB}	Input capacitance on bottom dual purpose V _{REF} pin when used as V _{REF} or user I/O pin	50	pF
C _{CLKB}	Input capacitance on bottom dual purpose clock input pins ⁽¹²⁾	7	pF
C _{CLKLRT}	Input capacitance on left/right/top dual purpose clock input pins ⁽¹²⁾	6	pF

Internal Weak Pull-Up Resistor

All I/O pins, except configuration, test, and JTAG pins, have an option to enable weak pull-up.

⁽⁹⁾ Dedicated LVDS output buffer is only available at bottom I/O banks.

⁽¹⁰⁾ ADC pins are only available at left I/O banks.

⁽¹¹⁾ When V_{REF} pin is used as regular input or output, F_{max} performance is reduced due to higher pin capacitance. Using the V_{REF} pin capacitance specification from device datasheet, perform SI analysis on your board setup to determine the F_{max} of your system.

⁽¹²⁾ 10M40 and 10M50 devices have dual purpose clock input pins at top/bottom I/O banks.

Table 17. Internal Weak Pull-Up Resistor for Intel MAX 10 Devices

Pin pull-up resistance values may be lower if an external source drives the pin higher than V_{CCIO} .

Symbol	Parameter	Condition	Min	Typ	Max	Unit
R _{PU}	Value of I/O pin (dedicated and dual-purpose) pull-up resistor before and during configuration, as well as user mode if the programmable pull-up resistor option is enabled	$V_{CCIO} = 3.3 \text{ V} \pm 5\%$	7	12	34	k Ω
		$V_{CCIO} = 3.0 \text{ V} \pm 5\%$	8	13	37	k Ω
		$V_{CCIO} = 2.5 \text{ V} \pm 5\%$	10	15	46	k Ω
		$V_{CCIO} = 1.8 \text{ V} \pm 5\%$	16	25	75	k Ω
		$V_{CCIO} = 1.5 \text{ V} \pm 5\%$	20	36	106	k Ω
		$V_{CCIO} = 1.2 \text{ V} \pm 5\%$	33	82	179	k Ω

Hot-Socketing Specifications

Table 18. Hot-Socketing Specifications for Intel MAX 10 Devices

Symbol	Parameter	Maximum
I _{IOPIN(DC)}	DC current per I/O pin	300 μA
I _{IOPIN(AC)}	AC current per I/O pin	8 mA ⁽¹³⁾

Hysteresis Specifications for Schmitt Trigger Input

Intel MAX 10 devices support Schmitt trigger input on all I/O pins. A Schmitt trigger feature introduces hysteresis to the input signal for improved noise immunity, especially for signal with slow edge rate.

⁽¹³⁾ The I/O ramp rate is 10 ns or more. For ramp rates faster than 10 ns, $|I_{IOPIN}| = C \, dv/dt$, in which C is I/O pin capacitance and dv/dt is the slew rate.

Table 19. Hysteresis Specifications for Schmitt Trigger Input for Intel MAX 10 Devices

Symbol	Parameter	Condition	Minimum	Unit
V _{HYS}	Hysteresis for Schmitt trigger input	V _{CCIO} = 3.3 V	180	mV
		V _{CCIO} = 2.5 V	150	mV
		V _{CCIO} = 1.8 V	120	mV
		V _{CCIO} = 1.5 V	110	mV

Figure 3. LVTTTL/LVCMOS Input Standard Voltage Diagram

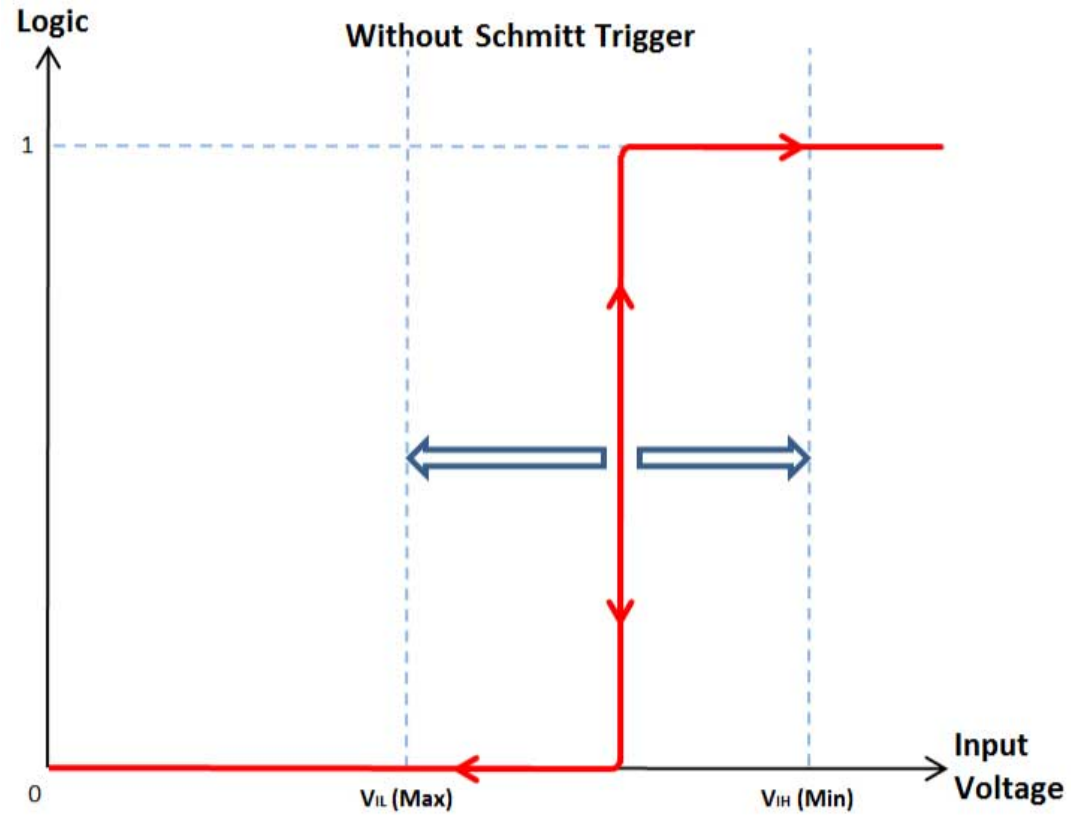
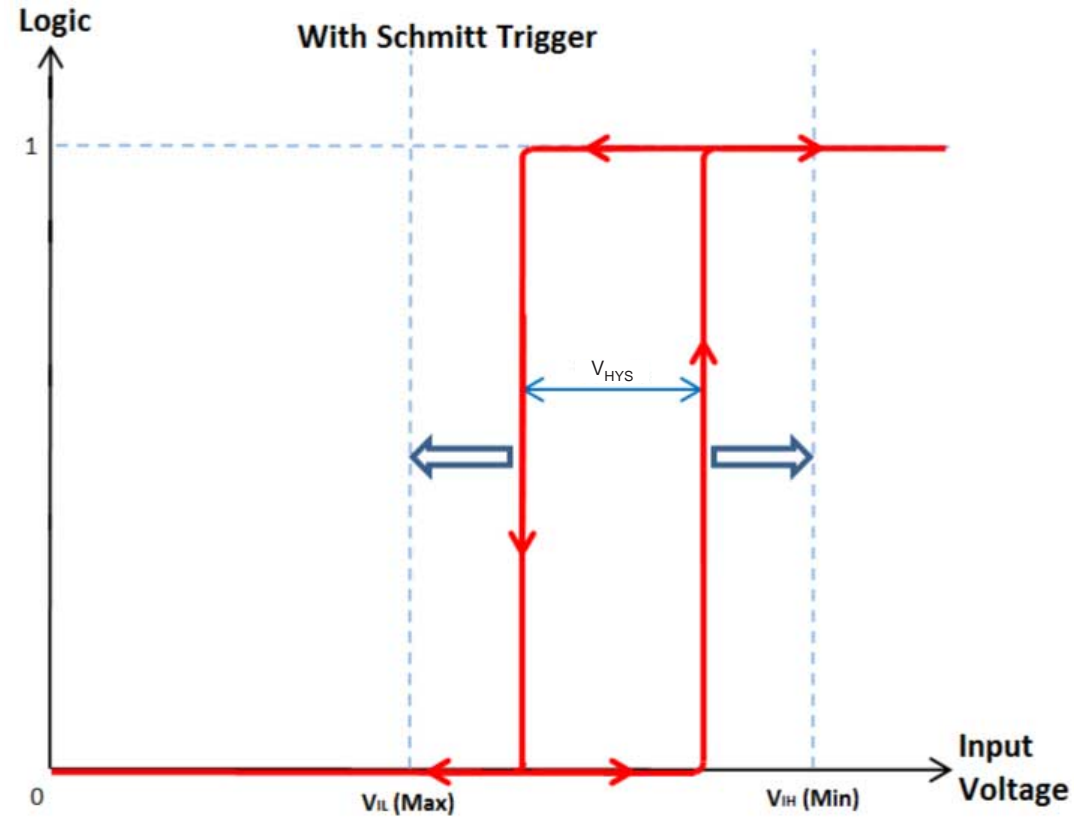


Figure 4. Schmitt Trigger Input Standard Voltage Diagram



I/O Standards Specifications

Tables in this section list input voltage (V_{IH} and V_{IL}), output voltage (V_{OH} and V_{OL}), and current drive characteristics (I_{OH} and I_{OL}) for various I/O standards supported by Intel MAX 10 devices.

For minimum voltage values, use the minimum V_{CCIO} values. For maximum voltage values, use the maximum V_{CCIO} values.

You must perform timing closure analysis to determine the maximum achievable frequency for general purpose I/O standards.

Single-Ended I/O Standards Specifications

Table 20. Single-Ended I/O Standards Specifications for Intel MAX 10 Devices

To meet the I_{OL} and I_{OH} specifications, you must set the current strength settings accordingly. For example, to meet the 3.3-V LVTTTL specification (4 mA), you should set the current strength settings to 4 mA. Setting at lower current strength may not meet the I_{OL} and I_{OH} specifications in the datasheet.

I/O Standard	V _{CCIO} (V)			V _{IL} (V)		V _{IH} (V)		V _{OL} (V)	V _{OH} (V)	I _{OL} (mA)	I _{OH} (mA)
	Min	Typ	Max	Min	Max	Min	Max	Max	Min		
3.3 V LVTTTL	3.135	3.3	3.465	-0.3	0.8	1.7	3.6	0.45	2.4	4	-4
3.3 V LVCMOS	3.135	3.3	3.465	-0.3	0.8	1.7	3.6	0.2	V _{CCIO} - 0.2	2	-2
3.0 V LVTTTL	2.85	3	3.15	-0.3	0.8	1.7	V _{CCIO} + 0.3	0.45	2.4	4	-4
3.0 V LVCMOS	2.85	3	3.15	-0.3	0.8	1.7	V _{CCIO} + 0.3	0.2	V _{CCIO} - 0.2	0.1	-0.1
2.5 V LVTTTL and LVCMOS	2.375	2.5	2.625	-0.3	0.7	1.7	V _{CCIO} + 0.3	0.4	2	1	-1
1.8 V LVTTTL and LVCMOS	1.71	1.8	1.89	-0.3	0.35 × V _{CCIO}	0.65 × V _{CCIO}	2.25	0.45	V _{CCIO} - 0.45	2	-2
1.5 V LVCMOS	1.425	1.5	1.575	-0.3	0.35 × V _{CCIO}	0.65 × V _{CCIO}	V _{CCIO} + 0.3	0.25 × V _{CCIO}	0.75 × V _{CCIO}	2	-2
1.2 V LVCMOS	1.14	1.2	1.26	-0.3	0.35 × V _{CCIO}	0.65 × V _{CCIO}	V _{CCIO} + 0.3	0.25 × V _{CCIO}	0.75 × V _{CCIO}	2	-2
1.0 V LVCMOS ⁽¹⁴⁾	0.95	1.0	1.05	-0.3	0.35 × V _{CCIO}	0.65 × V _{CCIO}	V _{CCIO} + 0.3	0.25 × V _{CCIO}	0.75 × V _{CCIO}	4	-4
3.3 V Schmitt Trigger	3.135	3.3	3.465	-0.3	0.8	1.7	V _{CCIO} + 0.3	—	—	—	—
2.5 V Schmitt Trigger	2.375	2.5	2.625	-0.3	0.7	1.7	V _{CCIO} + 0.3	—	—	—	—

continued...

⁽¹⁴⁾ The 1.0 V LVCMOS I/O standard is only supported in specific Intel MAX 10 devices. For the list of supported devices, refer to the *Intel MAX 10 General Purpose I/O User Guide*.

I/O Standard	V _{CCIO} (V)			V _{IL} (V)		V _{IH} (V)		V _{OL} (V)	V _{OH} (V)	I _{OL} (mA)	I _{OH} (mA)
	Min	Typ	Max	Min	Max	Min	Max	Max	Min		
1.8 V Schmitt Trigger	1.71	1.8	1.89	−0.3	0.35 × V _{CCIO}	0.65 × V _{CCIO}	V _{CCIO} + 0.3	—	—	—	—
1.5 V Schmitt Trigger	1.425	1.5	1.575	−0.3	0.35 × V _{CCIO}	0.65 × V _{CCIO}	V _{CCIO} + 0.3	—	—	—	—
3.0 V PCI	2.85	3	3.15	—	0.3 × V _{CCIO}	0.5 × V _{CCIO}	V _{CCIO} + 0.3	0.1 × V _{CCIO}	0.9 × V _{CCIO}	1.5	−0.5

Related Information

Intel MAX 10 I/O Standards Support, Intel MAX 10 General Purpose I/O User Guide

Provides the supported OPNs for 1.0 V LVCMOS I/O standard.

Single-Ended SSTL, HSTL, and HSUL I/O Reference Voltage Specifications

Table 21. Single-Ended SSTL, HSTL, and HSUL I/O Reference Voltage Specifications for Intel MAX 10 Devices

I/O Standard	V _{CCIO} (V)			V _{REF} (V)			V _{TT} (V) ⁽¹⁵⁾		
	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max
SSTL-2 Class I, II	2.375	2.5	2.625	1.19	1.25	1.31	V _{REF} − 0.04	V _{REF}	V _{REF} + 0.04
SSTL-18 Class I, II	1.7	1.8	1.9	0.833	0.9	0.969	V _{REF} − 0.04	V _{REF}	V _{REF} + 0.04
SSTL-15 Class I, II	1.425	1.5	1.575	0.49 × V _{CCIO}	0.5 × V _{CCIO}	0.51 × V _{CCIO}	0.49 × V _{CCIO}	0.5 × V _{CCIO}	0.51 × V _{CCIO}
SSTL-135 Class I, II	1.283	1.35	1.45	0.49 × V _{CCIO}	0.5 × V _{CCIO}	0.51 × V _{CCIO}	0.49 × V _{CCIO}	0.5 × V _{CCIO}	0.51 × V _{CCIO}
HSTL-18 Class I, II	1.71	1.8	1.89	0.85	0.9	0.95	0.85	0.9	0.95
HSTL-15 Class I, II	1.425	1.5	1.575	0.71	0.75	0.79	0.71	0.75	0.79
continued...									

⁽¹⁵⁾ V_{TT} of transmitting device must track V_{REF} of the receiving device.

I/O Standard	V _{CCIO} (V)			V _{REF} (V)			V _{TT} (V) ⁽¹⁵⁾		
	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max
HSTL-12 Class I, II	1.14	1.2	1.26	$0.48 \times V_{CCIO}^{(16)}$	$0.5 \times V_{CCIO}^{(16)}$	$0.52 \times V_{CCIO}^{(16)}$	—	$0.5 \times V_{CCIO}$	—
				$0.47 \times V_{CCIO}^{(17)}$	$0.5 \times V_{CCIO}^{(17)}$	$0.53 \times V_{CCIO}^{(17)}$			
HSUL-12	1.14	1.2	1.3	$0.49 \times V_{CCIO}$	$0.5 \times V_{CCIO}$	$0.51 \times V_{CCIO}$	—	—	—

Single-Ended SSTL, HSTL, and HSUL I/O Standards Signal Specifications

Table 22. Single-Ended SSTL, HSTL, and HSUL I/O Standards Signal Specifications for Intel MAX 10 Devices

To meet the I_{OL} and I_{OH} specifications, you must set the current strength settings accordingly. For example, to meet the SSTL-15 Class I specification (8 mA), you should set the current strength settings to 8 mA. Setting at lower current strength may not meet the I_{OL} and I_{OH} specifications in the datasheet.

I/O Standard	V _{IL(DC)} (V)		V _{IH(DC)} (V)		V _{IL(AC)} (V)		V _{IH(AC)} (V)		V _{OL} (V)	V _{OH} (V)	I _{OL} (mA)	I _{OH} (mA)
	Min	Max	Min	Max	Min	Max	Min	Max	Max	Min		
SSTL-2 Class I	—	$V_{REF} - 0.18$	$V_{REF} + 0.18$	—	—	$V_{REF} - 0.31$	$V_{REF} + 0.31$	—	$V_{TT} - 0.57$	$V_{TT} + 0.57$	8.1	–8.1
SSTL-2 Class II	—	$V_{REF} - 0.18$	$V_{REF} + 0.18$	—	—	$V_{REF} - 0.31$	$V_{REF} + 0.31$	—	$V_{TT} - 0.76$	$V_{TT} + 0.76$	16.4	–16.4
SSTL-18 Class I	—	$V_{REF} - 0.125$	$V_{REF} + 0.125$	—	—	$V_{REF} - 0.25$	$V_{REF} + 0.25$	—	$V_{TT} - 0.475$	$V_{TT} + 0.475$	6.7	–6.7
SSTL-18 Class II	—	$V_{REF} - 0.125$	$V_{REF} + 0.125$	—	—	$V_{REF} - 0.25$	$V_{REF} + 0.25$	—	0.28	$V_{CCIO} - 0.28$	13.4	–13.4
SSTL-15 Class I	—	$V_{REF} - 0.1$	$V_{REF} + 0.1$	—	—	$V_{REF} - 0.175$	$V_{REF} + 0.175$	—	$0.2 \times V_{CCIO}$	$0.8 \times V_{CCIO}$	8	–8
SSTL-15 Class II	—	$V_{REF} - 0.1$	$V_{REF} + 0.1$	—	—	$V_{REF} - 0.175$	$V_{REF} + 0.175$	—	$0.2 \times V_{CCIO}$	$0.8 \times V_{CCIO}$	16	–16

continued...

⁽¹⁶⁾ Value shown refers to DC input reference voltage, V_{REF(DC)}.

⁽¹⁷⁾ Value shown refers to AC input reference voltage, V_{REF(AC)}.

⁽¹⁵⁾ V_{TT} of transmitting device must track V_{REF} of the receiving device.

I/O Standard	V _{IL(DC)} (V)		V _{IH(DC)} (V)		V _{IL(AC)} (V)		V _{IH(AC)} (V)		V _{OL} (V)	V _{OH} (V)	I _{OL} (mA)	I _{OH} (mA)
	Min	Max	Min	Max	Min	Max	Min	Max	Max	Min		
SSTL-135	—	V _{REF} − 0.09	V _{REF} + 0.09	—	—	V _{REF} − 0.16	V _{REF} + 0.16	—	0.2 × V _{CCIO}	0.8 × V _{CCIO}	—	—
HSTL-18 Class I	—	V _{REF} − 0.1	V _{REF} + 0.1	—	—	V _{REF} − 0.2	V _{REF} + 0.2	—	0.4	V _{CCIO} − 0.4	8	−8
HSTL-18 Class II	—	V _{REF} − 0.1	V _{REF} + 0.1	—	—	V _{REF} − 0.2	V _{REF} + 0.2	—	0.4	V _{CCIO} − 0.4	16	−16
HSTL-15 Class I	—	V _{REF} − 0.1	V _{REF} + 0.1	—	—	V _{REF} − 0.2	V _{REF} + 0.2	—	0.4	V _{CCIO} − 0.4	8	−8
HSTL-15 Class II	—	V _{REF} − 0.1	V _{REF} + 0.1	—	—	V _{REF} − 0.2	V _{REF} + 0.2	—	0.4	V _{CCIO} − 0.4	16	−16
HSTL-12 Class I	−0.15	V _{REF} − 0.08	V _{REF} + 0.08	V _{CCIO} + 0.15	−0.24	V _{REF} − 0.15	V _{REF} + 0.15	V _{CCIO} + 0.24	0.25 × V _{CCIO}	0.75 × V _{CCIO}	8	−8
HSTL-12 Class II	−0.15	V _{REF} − 0.08	V _{REF} + 0.08	V _{CCIO} + 0.15	−0.24	V _{REF} − 0.15	V _{REF} + 0.15	V _{CCIO} + 0.24	0.25 × V _{CCIO}	0.75 × V _{CCIO}	14	−14
HSUL-12	—	V _{REF} − 0.13	V _{REF} + 0.13	—	—	V _{REF} − 0.22	V _{REF} + 0.22	—	0.1 × V _{CCIO}	0.9 × V _{CCIO}	—	—

Differential SSTL I/O Standards Specifications

Differential SSTL requires a V_{REF} input.

(15) V_{TT} of transmitting device must track V_{REF} of the receiving device.

Table 23. Differential SSTL I/O Standards Specifications for Intel MAX 10 Devices

I/O Standard	V_{CCIO} (V)			$V_{Swing(DC)}$ (V)		$V_{X(AC)}$ (V)			$V_{Swing(AC)}$ (V)	
	Min	Typ	Max	Min	Max ⁽¹⁸⁾	Min	Typ	Max	Min	Max
SSTL-2 Class I, II	2.375	2.5	2.625	0.36	V_{CCIO}	$V_{CCIO}/2 - 0.2$	—	$V_{CCIO}/2 + 0.2$	0.7	V_{CCIO}
SSTL-18 Class I, II	1.7	1.8	1.9	0.25	V_{CCIO}	$V_{CCIO}/2 - 0.175$	—	$V_{CCIO}/2 + 0.175$	0.5	V_{CCIO}
SSTL-15 Class I, II	1.425	1.5	1.575	0.2	—	$V_{CCIO}/2 - 0.15$	—	$V_{CCIO}/2 + 0.15$	$2(V_{IH(AC)} - V_{REF})$	$2(V_{IL(AC)} - V_{REF})$
SSTL-135	1.283	1.35	1.45	0.18	—	$V_{REF} - 0.135$	$0.5 \times V_{CCIO}$	$V_{REF} + 0.135$	$2(V_{IH(AC)} - V_{REF})$	$2(V_{IL(AC)} - V_{REF})$

Differential HSTL and HSUL I/O Standards Specifications

Differential HSTL requires a V_{REF} input.

Table 24. Differential HSTL and HSUL I/O Standards Specifications for Intel MAX 10 Devices

I/O Standard	V_{CCIO} (V)			$V_{DIF(DC)}$ (V)		$V_{X(AC)}$ (V)			$V_{CM(DC)}$ (V)			$V_{DIF(AC)}$ (V)
	Min	Typ	Max	Min	Max	Min	Typ	Max	Min	Typ	Max	Min
HSTL-18 Class I, II	1.71	1.8	1.89	0.2	—	0.85	—	0.95	0.85	—	0.95	0.4
HSTL-15 Class I, II	1.425	1.5	1.575	0.2	—	0.71	—	0.79	0.71	—	0.79	0.4
HSTL-12 Class I, II	1.14	1.2	1.26	0.16	V_{CCIO}	$0.48 \times V_{CCIO}$	$0.5 \times V_{CCIO}$	$0.52 \times V_{CCIO}$	$0.48 \times V_{CCIO}$	$0.5 \times V_{CCIO}$	$0.52 \times V_{CCIO}$	0.3
HSUL-12	1.14	1.2	1.3	0.26	—	$0.5 \times V_{CCIO} - 0.12$	$0.5 \times V_{CCIO}$	$0.5 \times V_{CCIO} + 0.12$	$0.4 \times V_{CCIO}$	$0.5 \times V_{CCIO}$	$0.6 \times V_{CCIO}$	0.44

⁽¹⁸⁾ The maximum value for $V_{SWING(DC)}$ is not defined. However, each single-ended signal needs to be within the respective single-ended limits ($V_{IH(DC)}$ and $V_{IL(DC)}$).

Differential I/O Standards Specifications

Table 25. Differential I/O Standards Specifications for Intel MAX 10 Devices

I/O Standard	V _{CCIO} (V)			V _{ID} (mV)		V _{ICM} (V) ⁽¹⁹⁾			V _{OD} (mV) ⁽²⁰⁾⁽²¹⁾			V _{OS} (V) ⁽²⁰⁾		
	Min	Typ	Max	Min	Max	Min	Condition	Max	Min	Typ	Max	Min	Typ	Max
LVPECL ⁽²²⁾	2.375	2.5	2.625	100	—	0.05	D _{MAX} ≤ 500 Mbps	1.8	—	—	—	—	—	—
						0.55	500 Mbps ≤ D _{MAX} ≤ 700 Mbps	1.8						
						1.05	D _{MAX} > 700 Mbps	1.55						
LVDS	2.375	2.5	2.625	100	—	0.05	D _{MAX} ≤ 500 Mbps	1.8	247	—	600	1.125	1.25	1.375
						0.55	500 Mbps ≤ D _{MAX} ≤ 700 Mbps	1.8						
						1.05	D _{MAX} > 700 Mbps	1.55						
1.8 V LVDS ⁽²³⁾	1.71	1.8	1.89	100	800	0.8	D _{MAX} ≤ 400 Mbps	1	100	—	800	0.81	0.9	0.99
BLVDS ⁽²⁴⁾	2.375	2.5	2.625	100	—	—	—	—	—	—	—	—	—	—
mini-LVDS ⁽²⁵⁾	2.375	2.5	2.625	—	—	—	—	—	300	—	600	1	1.2	1.4
continued...														

⁽¹⁹⁾ V_{IN} range: 0 V ≤ V_{IN} ≤ 1.85 V.

⁽²⁰⁾ R_L range: 90 ≤ R_L ≤ 110 Ω.

⁽²¹⁾ Low V_{OD} setting is only supported for RSDS standard.

⁽²²⁾ LVPECL input standard is only supported at clock input. Output standard is not supported.

⁽²³⁾ The 1.8 V LVDS buffers are supported as inputs on all high-speed I/O banks but as outputs only on the bottom banks. The low-speed and high-speed DDR3 I/O banks do not support 1.8 V LVDS. The 1.8 V LVDS I/O standard is supported in industrial- and commercial-grade Intel MAX 10 dual supply devices except in packages V36 and V81.

⁽²⁴⁾ No fixed V_{IN}, V_{OD}, and V_{OS} specifications for Bus LVDS (BLVDS). They are dependent on the system topology.

⁽²⁵⁾ Mini-LVDS, RSDS, and Point-to-Point Differential Signaling (PPDS) standards are only supported at the output pins for Intel MAX 10 devices.

I/O Standard	V _{CCIO} (V)			V _{ID} (mV)		V _{ICM} (V) ⁽¹⁹⁾			V _{OD} (mV) ⁽²⁰⁾⁽²¹⁾			V _{OS} (V) ⁽²⁰⁾		
	Min	Typ	Max	Min	Max	Min	Condition	Max	Min	Typ	Max	Min	Typ	Max
RSDS ⁽²⁵⁾	2.375	2.5	2.625	—	—	—	—	—	100	200	600	0.5	1.2	1.5
PPDS (Row I/Os) ⁽²⁵⁾	2.375	2.5	2.625	—	—	—	—	—	100	200	600	0.5	1.2	1.4
TMDS ⁽²⁶⁾	2.375	2.5	2.625	100	—	0.05	D _{MAX} ≤ 500 Mbps	1.8	—	—	—	—	—	—
						0.55	500 Mbps ≤ D _{MAX} ≤ 700 Mbps	1.8						
						1.05	D _{MAX} > 700 Mbps	1.55						
Sub-LVDS ⁽²⁷⁾	1.71	1.8	1.89	100	—	0.55	—	1.25	(28)			0.8	0.9	1
SLVS	2.375	2.5	2.625	100	—	0.05	—	1.1	(28)			(29)		
HiSpi	2.375	2.5	2.625	100	—	0.05	D _{MAX} ≤ 500 Mbps	1.8	—	—	—	—	—	—
						0.55	500 Mbps ≤ D _{MAX} ≤ 700 Mbps	1.8						
						1.05	D _{MAX} > 700 Mbps	1.55						

Related Information

[Intel MAX 10 LVDS SERDES I/O Standards Support, Intel MAX 10 High-Speed LVDS I/O User Guide](#)

Provides the list of I/O standards supported in single supply and dual supply devices.

⁽¹⁹⁾ V_{IN} range: 0 V ≤ V_{IN} ≤ 1.85 V.

⁽²⁰⁾ R_L range: 90 ≤ R_L ≤ 110 Ω.

⁽²¹⁾ Low V_{OD} setting is only supported for RSDS standard.

⁽²⁶⁾ Supported with requirement of an external level shift

⁽²⁷⁾ Sub-LVDS input buffer is using 2.5 V differential buffer.

⁽²⁸⁾ Differential output depends on the values of the external termination resistors.

⁽²⁹⁾ Differential output offset voltage depends on the values of the external termination resistors.

Switching Characteristics

This section provides the performance characteristics of Intel MAX 10 core and periphery blocks.

Core Performance Specifications

Clock Tree Specifications

Table 26. Clock Tree Specifications for Intel MAX 10 Devices

Device	Performance					Unit
	-I6	-A6, -C7	-I7	-A7	-C8, -I8	
10M02	450	416	416	382	402	MHz
10M04	450	416	416	382	402	MHz
10M08	450	416	416	382	402	MHz
10M16	450	416	416	382	402	MHz
10M25	450	416	416	382	402	MHz
10M40	450	416	416	382	402	MHz
10M50	450	416	416	382	402	MHz

PLL Specifications

Table 27. PLL Specifications for Intel MAX 10 Devices

V_{CCD_PLL} should always be connected to V_{CCINT} through decoupling capacitor and ferrite bead.

Symbol	Parameter	Condition	Min	Typ	Max	Unit
$f_{IN}^{(30)}$	Input clock frequency	—	5	—	472.5	MHz
f_{INPFD}	Phase frequency detector (PFD) input frequency	—	5	—	325	MHz
continued...						

⁽³⁰⁾ This parameter is limited in the Intel Quartus Prime software by the I/O maximum frequency. The maximum I/O frequency is different for each I/O standard.

Symbol	Parameter	Condition	Min	Typ	Max	Unit
$f_{VCO}^{(31)}$	PLL internal voltage-controlled oscillator (VCO) operating range	—	600	—	1300	MHz
f_{INDUTY}	Input clock duty cycle	—	40	—	60	%
$t_{INJITTER_CCJ}^{(32)}$	Input clock cycle-to-cycle jitter	$F_{INPFD} \geq 100$ MHz	—	—	0.15	UI
		$F_{INPFD} < 100$ MHz	—	—	± 750	ps
$f_{OUT_EXT}^{(30)}$	PLL output frequency for external clock output	—	—	—	472.5	MHz
f_{OUT}	PLL output frequency to global clock	–6 speed grade	—	—	472.5	MHz
		–7 speed grade	—	—	450	MHz
		–8 speed grade	—	—	402.5	MHz
$t_{OUTDUTY}$	Duty cycle for external clock output	Duty cycle set to 50%	45	50	55	%
t_{LOCK}	Time required to lock from end of device configuration	—	—	—	1	ms
t_{DLOCK}	Time required to lock dynamically	After switchover, reconfiguring any non-post-scale counters or delays, or when <code>areset</code> is deasserted	—	—	1	ms
$t_{OUTJITTER_PERIOD_IO}^{(33)}$	Regular I/O period jitter	$F_{OUT} \geq 100$ MHz	—	—	650	ps
		$F_{OUT} < 100$ MHz	—	—	75	mUI
$t_{OUTJITTER_CCJ_IO}^{(33)}$	Regular I/O cycle-to-cycle jitter	$F_{OUT} \geq 100$ MHz	—	—	650	ps
		$F_{OUT} < 100$ MHz	—	—	75	mUI

continued...

- (31) The VCO frequency reported by the Intel Quartus Prime software in the PLL summary section of the compilation report takes into consideration the VCO post-scale counter K value. Therefore, if the counter K has a value of 2, the frequency reported can be lower than the f_{VCO} specification.
- (32) A high input jitter directly affects the PLL output jitter. To have low PLL output clock jitter, you must provide a clean clock source, which is less than 200 ps.
- (33) Peak-to-peak jitter with a probability level of 10^{-12} (14 sigma, 99.9999999974404% confidence level). The output jitter specification applies to the intrinsic jitter of the PLL, when an input jitter of 30 ps is applied.

Symbol	Parameter	Condition	Min	Typ	Max	Unit
t_{PLL_PSERR}	Accuracy of PLL phase shift	—	—	—	±50	ps
t_{ARESET}	Minimum pulse width on areset signal.	—	10	—	—	ns
$t_{CONFIGPLL}$	Time required to reconfigure scan chains for PLLs	—	—	3.5 ⁽³⁴⁾	—	SCANCLK cycles
$f_{SCANCLK}$	scanclk frequency	—	—	—	100	MHz

Table 28. PLL Specifications for Intel MAX 10 Single Supply Devices

For V36 package, the PLL specification is based on single supply devices.

Symbol	Parameter	Condition	Max	Unit
$t_{OUTJITTER_PERIOD_DEDCLK}$ ⁽³³⁾	Dedicated clock output period jitter	$F_{OUT} \geq 100$ MHz	660	ps
		$F_{OUT} < 100$ MHz	66	mUI
$t_{OUTJITTER_CCJ_DEDCLK}$ ⁽³³⁾	Dedicated clock output cycle-to-cycle jitter	$F_{OUT} \geq 100$ MHz	660	ps
		$F_{OUT} < 100$ MHz	66	mUI

Table 29. PLL Specifications for Intel MAX 10 Dual Supply Devices

Symbol	Parameter	Condition	Max	Unit
$t_{OUTJITTER_PERIOD_DEDCLK}$ ⁽³³⁾	Dedicated clock output period jitter	$F_{OUT} \geq 100$ MHz	300	ps
		$F_{OUT} < 100$ MHz	30	mUI
$t_{OUTJITTER_CCJ_DEDCLK}$ ⁽³³⁾	Dedicated clock output cycle-to-cycle jitter	$F_{OUT} \geq 100$ MHz	300	ps
		$F_{OUT} < 100$ MHz	30	mUI

⁽³⁴⁾ With 100 MHz scanclk frequency.

Embedded Multiplier Specifications

Table 30. Embedded Multiplier Specifications for Intel MAX 10 Devices

Mode	Number of Multipliers	Power Supply Mode	Performance			Unit
			-I6	-A6, -C7, -I7, -A7	-C8, -I8	
9 × 9-bit multiplier	1	Single supply mode	198	183	160	MHz
		Dual supply mode	310	260	210	MHz
18 × 18-bit multiplier	1	Single supply mode	198	183	160	MHz
		Dual supply mode	265	240	190	MHz

Memory Block Performance Specifications

Table 31. Memory Block Performance Specifications for Intel MAX 10 Devices

Memory	Mode	Resources Used		Power Supply Mode	Performance			Unit
		LEs	M9K Memory		-I6	-A6, -C7, -I7, -A7	-C8, -I8	
M9K Block	FIFO 256 × 36	47	1	Single supply mode	232	219	204	MHz
				Dual supply mode	330	300	250	MHz
	Single-port 256 × 36	0	1	Single supply mode	232	219	204	MHz
				Dual supply mode	330	300	250	MHz
	Simple dual-port 256 × 36 CLK	0	1	Single supply mode	232	219	204	MHz
				Dual supply mode	330	300	250	MHz
	True dual port 512 × 18 single CLK	0	1	Single supply mode	232	219	204	MHz
				Dual supply mode	330	300	250	MHz

Internal Oscillator Specifications

Table 32. Internal Oscillator Frequencies for Intel MAX 10 Devices

You can access to the internal oscillator frequencies in this table. The duty cycle of internal oscillator is approximately 45%–55%.

Device	Frequency			Unit
	Minimum	Typical	Maximum	
10M02	55	82	116	MHz
10M04				
10M08				
10M16				
10M25				
10M40	35	52	77	MHz
10M50				

UFM Performance Specifications

Table 33. UFM Performance Specifications for Intel MAX 10 Devices

Block	Mode	Interface	Device	Frequency		Unit
				Minimum	Maximum	
UFM	Avalon®-MM slave	Parallel ⁽³⁵⁾	10M02 ⁽³⁶⁾	3.43	7.25	MHz
			10M04, 10M08, 10M16, 10M25, 10M40, 10M50	5	116	MHz
		Serial ⁽³⁶⁾	10M02, 10M04, 10M08, 10M16, 10M25	3.43	7.25	MHz
			10M40, 10M50	2.18	4.81	MHz

⁽³⁵⁾ Clock source is derived from user, except for 10M02 device.

⁽³⁶⁾ Clock source is derived from 1/16 of the frequency of the internal oscillator.

ADC Performance Specifications

Single Supply Devices ADC Performance Specifications

Table 34. ADC Performance Specifications for Intel MAX 10 Single Supply Devices

Parameter		Symbol	Condition	Min	Typ	Max	Unit
ADC resolution		—	—	—	—	12	bits
ADC supply voltage		V_{CC_ONE}	—	2.85	3.0/3.3	3.465	V
External reference voltage		V_{REF}	—	$V_{CC_ONE} - 0.5$	—	V_{CC_ONE}	V
Sampling rate		F_S	Accumulative sampling rate	—	—	1	MSPS
Operating junction temperature range		T_J	—	−40	25	125	°C
Analog input voltage		V_{IN}	Prescaler disabled	0	—	V_{REF}	V
			Prescaler enabled ⁽³⁷⁾	0	—	3.6	V
Input resistance		R_{IN}	—	—	⁽³⁸⁾	—	—
Input capacitance		C_{IN}	—	—	⁽³⁸⁾	—	—
DC Accuracy	Offset error and drift	E_{offset}	Prescaler disabled	−0.2	—	0.2	%FS
			Prescaler enabled	−0.5	—	0.5	%FS
	Gain error and drift	E_{gain}	Prescaler disabled	−0.5	—	0.5	%FS
			Prescaler enabled	−0.75	—	0.75	%FS
	Differential non linearity	DNL	External V_{REF} , no missing code	−0.9	—	0.9	LSB
			Internal V_{REF} , no missing code	−1	—	1.7	LSB

continued...

⁽³⁷⁾ Prescaler function divides the analog input voltage by half. The analog input handles up to 3.6 V for the Intel MAX 10 single supply devices.

⁽³⁸⁾ Download the SPICE models for simulation.

Parameter		Symbol	Condition	Min	Typ	Max	Unit
	Integral non linearity	INL	—	–2	—	2	LSB
AC Accuracy	Total harmonic distortion	THD	$F_{IN} = 50 \text{ kHz}$, $F_S = 1 \text{ MHz}$, PLL	–65 ⁽³⁹⁾	—	—	dB
	Signal-to-noise ratio	SNR	$F_{IN} = 50 \text{ kHz}$, $F_S = 1 \text{ MHz}$, PLL	54 ⁽⁴⁰⁾	—	—	dB
	Signal-to-noise and distortion	SINAD	$F_{IN} = 50 \text{ kHz}$, $F_S = 1 \text{ MHz}$, PLL	53 ⁽⁴¹⁾	—	—	dB
On-Chip Temperature Sensor	Temperature sampling rate	T_S	—	—	—	50	kSPS
	Absolute accuracy	—	–40 to 125°C, with 64 samples averaging ⁽⁴²⁾	—	—	±10	°C
Conversion Rate ⁽⁴³⁾	Conversion time	—	Single measurement	—	—	1	Cycle
			Continuous measurement	—	—	1	Cycle
			Temperature measurement	—	—	1	Cycle

Related Information

[SPICE Models for Intel FPGAs](#)

⁽³⁹⁾ THD with prescaler enabled is 6dB less than the specification.

⁽⁴⁰⁾ SNR with prescaler enabled is 6dB less than the specification.

⁽⁴¹⁾ SINAD with prescaler enabled is 6dB less than the specification.

⁽⁴²⁾ For the Intel Quartus Prime software version 15.0 and later, Modular ADC Core Intel FPGA IP and Modular Dual ADC Core Intel FPGA IP cores handle the 64 samples averaging. For the Intel Quartus Prime software versions prior to 14.1, you need to implement your own averaging calculation.

⁽⁴³⁾ For more detailed description, refer to the Timing section in the *Intel MAX 10 Analog-to-Digital Converter User Guide*.

Dual Supply Devices ADC Performance Specifications

Table 35. ADC Performance Specifications for Intel MAX 10 Dual Supply Devices

Parameter		Symbol	Condition	Min	Typ	Max	Unit
ADC resolution		—	—	—	—	12	bits
Analog supply voltage		V_{CCA_ADC}	—	2.375	2.5	2.625	V
Digital supply voltage		V_{CCINT}	—	1.15	1.2	1.25	V
External reference voltage		V_{REF}	—	$V_{CCA_ADC} - 0.5$	—	V_{CCA_ADC}	V
Sampling rate		F_S	Accumulative sampling rate	—	—	1	MSPS
Operating junction temperature range		T_J	—	−40	25	125	°C
Analog input voltage		V_{IN}	Prescaler disabled	0	—	V_{REF}	V
			Prescaler enabled ⁽⁴⁴⁾	0	—	3	V
Analog supply current (DC)		I_{ACC_ADC}	Average current	—	275	450	μA
Digital supply current (DC)		I_{CCINT}	Average current	—	65	150	μA
Input resistance		R_{IN}	—	—	⁽⁴⁵⁾	—	—
Input capacitance		C_{IN}	—	—	⁽⁴⁵⁾	—	—
DC Accuracy	Offset error and drift	E_{offset}	Prescaler disabled	−0.2	—	0.2	%FS
			Prescaler enabled	−0.5	—	0.5	%FS
	Gain error and drift	E_{gain}	Prescaler disabled	−0.5	—	0.5	%FS
			Prescaler enabled	−0.75	—	0.75	%FS
	Differential non linearity	DNL	External V_{REF} , no missing code	−0.9	—	0.9	LSB

continued...

⁽⁴⁴⁾ Prescaler function divides the analog input voltage by half. The analog input handles up to 3 V input for the Intel MAX 10 dual supply devices.

⁽⁴⁵⁾ Download the SPICE models for simulation.

Parameter	Symbol	Condition	Min	Typ	Max	Unit
		Internal V_{REF} , no missing code	-1	—	1.7	LSB
	Integral non linearity	INL	—	—	2	LSB
AC Accuracy	Total harmonic distortion	THD	$F_{IN} = 50 \text{ kHz}$, $F_S = 1 \text{ MHz}$, PLL	-70 ⁽⁴⁶⁾ (47) (48)	—	dB
	Signal-to-noise ratio	SNR	$F_{IN} = 50 \text{ kHz}$, $F_S = 1 \text{ MHz}$, PLL	62 ⁽⁴⁹⁾ (50)(48)	—	dB
	Signal-to-noise and distortion	SINAD	$F_{IN} = 50 \text{ kHz}$, $F_S = 1 \text{ MHz}$, PLL	61.5 ⁽⁵¹⁾ (52)(48)	—	dB
On-Chip Temperature Sensor	Temperature sampling rate	T_S	—	—	50	kSPS
	Absolute accuracy	—	-40 to 125°C, with 64 samples averaging (53)	—	±5	°C
continued...						

(46) Total harmonic distortion is -65 dB for dual function pin.

(47) THD with prescaler enabled is 6dB less than the specification.

(48) When using internal V_{REF} , THD = 66 dB, SNR = 58 dB and SINAD = 57.5 dB for dedicated ADC input channels.

(49) Signal-to-noise ratio is 54 dB for dual function pin.

(50) SNR with prescaler enabled is 6dB less than the specification.

(51) Signal-to-noise and distortion is 53 dB for dual function pin.

(52) SINAD with prescaler enabled is 6dB less than the specification.

(53) For the Intel Quartus Prime software version 15.0 and later, Modular ADC Core and Modular Dual ADC Core IP cores handle the 64 samples averaging. For the Intel Quartus Prime software versions prior to 14.1, you need to implement your own averaging calculation.

Parameter		Symbol	Condition	Min	Typ	Max	Unit
Conversion Rate ⁽⁵⁴⁾	Conversion time	—	Single measurement	—	—	1	Cycle
			Continuous measurement	—	—	1	Cycle
			Temperature measurement	—	—	1	Cycle

Related Information

[SPICE Models for Intel FPGAs](#)

Periphery Performance Specifications

This section describes the periphery performance, high-speed I/O, and external memory interface.

Actual achievable frequency depends on design and system specific factors. Ensure proper timing closure in your design and perform HSPICE/IBIS simulations based on your specific design and system setup to determine the maximum achievable frequency in your system.

High-Speed I/O Specifications

For more information about the high-speed and low-speed I/O performance pins, refer to the respective device pin-out files.

Related Information

[Documentation: Pin-Out Files for Intel FPGAs](#)

⁽⁵⁴⁾ For more detailed description, refer to the Timing section in the *Intel MAX 10 Analog-to-Digital Converter User Guide*.

True PPDS and Emulated PPDS_E_3R Transmitter Timing Specifications

Table 36. True PPDS and Emulated PPDS_E_3R Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices

True **PPDS** transmitter is only supported at bottom I/O banks. Emulated **PPDS** transmitter is supported at the output pin of all I/O banks.

Symbol	Parameter	Mode	-I6, -A6, -C7, -I7			-A7			-C8, -I8			Unit
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
f _{HCLK}	Input clock frequency (high-speed I/O performance pin)	×10	5	—	155	5	—	155	5	—	155	MHz
		×8	5	—	155	5	—	155	5	—	155	MHz
		×7	5	—	155	5	—	155	5	—	155	MHz
		×4	5	—	155	5	—	155	5	—	155	MHz
		×2	5	—	155	5	—	155	5	—	155	MHz
		×1	5	—	310	5	—	310	5	—	310	MHz
HSIODR	Data rate (high-speed I/O performance pin)	×10	100	—	310	100	—	310	100	—	310	Mbps
		×8	80	—	310	80	—	310	80	—	310	Mbps
		×7	70	—	310	70	—	310	70	—	310	Mbps
		×4	40	—	310	40	—	310	40	—	310	Mbps
		×2	20	—	310	20	—	310	20	—	310	Mbps
		×1	10	—	310	10	—	310	10	—	310	Mbps
f _{HCLK}	Input clock frequency (low-speed I/O performance pin)	×10	5	—	150	5	—	150	5	—	150	MHz
		×8	5	—	150	5	—	150	5	—	150	MHz
		×7	5	—	150	5	—	150	5	—	150	MHz
		×4	5	—	150	5	—	150	5	—	150	MHz
		×2	5	—	150	5	—	150	5	—	150	MHz
		×1	5	—	300	5	—	300	5	—	300	MHz
HSIODR	Data rate (low-speed I/O performance pin)	×10	100	—	300	100	—	300	100	—	300	Mbps
		×8	80	—	300	80	—	300	80	—	300	Mbps
		×7	70	—	300	70	—	300	70	—	300	Mbps

continued...

Symbol	Parameter	Mode	-I6, -A6, -C7, -I7			-A7			-C8, -I8			Unit
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
		×4	40	—	300	40	—	300	40	—	300	Mbps
		×2	20	—	300	20	—	300	20	—	300	Mbps
		×1	10	—	300	10	—	300	10	—	300	Mbps
t _{DUTY}	Duty cycle on transmitter output clock	—	45	—	55	45	—	55	45	—	55	%
TCCS ⁽⁵⁵⁾	Transmitter channel-to-channel skew	—	—	—	300	—	—	300	—	—	300	ps
t _{x jitter} ⁽⁵⁶⁾	Output jitter (high-speed I/O performance pin)	—	—	—	425	—	—	425	—	—	425	ps
	Output jitter (low-speed I/O performance pin)	—	—	—	470	—	—	470	—	—	470	ps
t _{RISE}	Rise time	20 – 80%, C _{LOAD} = 5 pF	—	500	—	—	500	—	—	500	—	ps
t _{FALL}	Fall time	20 – 80%, C _{LOAD} = 5 pF	—	500	—	—	500	—	—	500	—	ps
t _{LOCK}	Time required for the PLL to lock, after CONF_DONE signal goes high, indicating the completion of device configuration	—	—	—	1	—	—	1	—	—	1	ms

⁽⁵⁵⁾ TCCS specifications apply to I/O banks from the same side only.

⁽⁵⁶⁾ TX jitter is the jitter induced from core noise and I/O switching noise.

True RSDS and Emulated RSDS_E_3R Transmitter Timing Specifications

Single Supply Devices True RSDS and Emulated RSDS_E_3R Transmitter Timing Specifications

Table 37. True RSDS and Emulated RSDS_E_3R Transmitter Timing Specifications for Intel MAX 10 Single Supply Devices

True **RSDS** transmitter is only supported at bottom I/O banks. Emulated **RSDS** transmitter is supported at the output pin of all I/O banks.

Symbol	Parameter	Mode	-I6, -A6, -C7, -I7			-A7			-C8, -I8			Unit
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
f _{HCLK}	Input clock frequency (high-speed I/O performance pin)	×10	5	—	50	5	—	50	5	—	50	MHz
		×8	5	—	50	5	—	50	5	—	50	MHz
		×7	5	—	50	5	—	50	5	—	50	MHz
		×4	5	—	50	5	—	50	5	—	50	MHz
		×2	5	—	50	5	—	50	5	—	50	MHz
		×1	5	—	100	5	—	100	5	—	100	MHz
HSIODR	Data rate (high-speed I/O performance pin)	×10	100	—	100	100	—	100	100	—	100	Mbps
		×8	80	—	100	80	—	100	80	—	100	Mbps
		×7	70	—	100	70	—	100	70	—	100	Mbps
		×4	40	—	100	40	—	100	40	—	100	Mbps
		×2	20	—	100	20	—	100	20	—	100	Mbps
		×1	10	—	100	10	—	100	10	—	100	Mbps
f _{HCLK}	Input clock frequency (low-speed I/O performance pin)	×10	5	—	50	5	—	50	5	—	50	MHz
		×8	5	—	50	5	—	50	5	—	50	MHz
		×7	5	—	50	5	—	50	5	—	50	MHz
		×4	5	—	50	5	—	50	5	—	50	MHz
		×2	5	—	50	5	—	50	5	—	50	MHz
		×1	5	—	100	5	—	100	5	—	100	MHz
HSIODR	Data rate (low-speed I/O performance pin)	×10	100	—	100	100	—	100	100	—	100	Mbps
continued...												

Symbol	Parameter	Mode	-I6, -A6, -C7, -I7			-A7			-C8, -I8			Unit
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
		×8	80	—	100	80	—	100	80	—	100	Mbps
		×7	70	—	100	70	—	100	70	—	100	Mbps
		×4	40	—	100	40	—	100	40	—	100	Mbps
		×2	20	—	100	20	—	100	20	—	100	Mbps
		×1	10	—	100	10	—	100	10	—	100	Mbps
t _{DUTY}	Duty cycle on transmitter output clock	—	45	—	55	45	—	55	45	—	55	%
TCCS ⁽⁵⁷⁾	Transmitter channel-to-channel skew	—	—	—	300	—	—	300	—	—	300	ps
t _x Jitter ⁽⁵⁸⁾	Output jitter (high-speed I/O performance pin)	—	—	—	425	—	—	425	—	—	425	ps
	Output jitter (low-speed I/O performance pin)	—	—	—	470	—	—	470	—	—	470	ps
t _{RISE}	Rise time	20 – 80%, C _{LOAD} = 5 pF	—	500	—	—	500	—	—	500	—	ps
t _{FALL}	Fall time	20 – 80%, C _{LOAD} = 5 pF	—	500	—	—	500	—	—	500	—	ps
t _{LOCK}	Time required for the PLL to lock, after CONF_DONE signal goes high, indicating the completion of device configuration	—	—	—	1	—	—	1	—	—	1	ms

⁽⁵⁷⁾ TCCS specifications apply to I/O banks from the same side only.

⁽⁵⁸⁾ TX jitter is the jitter induced from core noise and I/O switching noise.

Dual Supply Devices True RSDS and Emulated RSDS_E_3R Transmitter Timing Specifications

Table 38. True RSDS and Emulated RSDS_E_3R Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices

True **RSDS** transmitter is only supported at bottom I/O banks. Emulated **RSDS** transmitter is supported at the output pin of all I/O banks.

Symbol	Parameter	Mode	-I6, -A6, -C7, -I7			-A7			-C8, -I8			Unit
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
f _{HCLK}	Input clock frequency (high-speed I/O performance pin)	×10	5	—	155	5	—	155	5	—	155	MHz
		×8	5	—	155	5	—	155	5	—	155	MHz
		×7	5	—	155	5	—	155	5	—	155	MHz
		×4	5	—	155	5	—	155	5	—	155	MHz
		×2	5	—	155	5	—	155	5	—	155	MHz
		×1	5	—	310	5	—	310	5	—	310	MHz
HSIODR	Data rate (high-speed I/O performance pin)	×10	100	—	310	100	—	310	100	—	310	Mbps
		×8	80	—	310	80	—	310	80	—	310	Mbps
		×7	70	—	310	70	—	310	70	—	310	Mbps
		×4	40	—	310	40	—	310	40	—	310	Mbps
		×2	20	—	310	20	—	310	20	—	310	Mbps
		×1	10	—	310	10	—	310	10	—	310	Mbps
f _{HCLK}	Input clock frequency (low-speed I/O performance pin)	×10	5	—	150	5	—	150	5	—	150	MHz
		×8	5	—	150	5	—	150	5	—	150	MHz
		×7	5	—	150	5	—	150	5	—	150	MHz
		×4	5	—	150	5	—	150	5	—	150	MHz
		×2	5	—	150	5	—	150	5	—	150	MHz
		×1	5	—	300	5	—	300	5	—	300	MHz
HSIODR	Data rate (low-speed I/O performance pin)	×10	100	—	300	100	—	300	100	—	300	Mbps
		×8	80	—	300	80	—	300	80	—	300	Mbps
		×7	70	—	300	70	—	300	70	—	300	Mbps

continued...

Symbol	Parameter	Mode	-I6, -A6, -C7, -I7			-A7			-C8, -I8			Unit
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
		×4	40	—	300	40	—	300	40	—	300	Mbps
		×2	20	—	300	20	—	300	20	—	300	Mbps
		×1	10	—	300	10	—	300	10	—	300	Mbps
t _{DUTY}	Duty cycle on transmitter output clock	—	45	—	55	45	—	55	45	—	55	%
TCCS ⁽⁵⁹⁾	Transmitter channel-to-channel skew	—	—	—	300	—	—	300	—	—	300	ps
t _{x jitter} ⁽⁶⁰⁾	Output jitter (high-speed I/O performance pin)	—	—	—	425	—	—	425	—	—	425	ps
	Output jitter (low-speed I/O performance pin)	—	—	—	470	—	—	470	—	—	470	ps
t _{RISE}	Rise time	20 – 80%, C _{LOAD} = 5 pF	—	500	—	—	500	—	—	500	—	ps
t _{FALL}	Fall time	20 – 80%, C _{LOAD} = 5 pF	—	500	—	—	500	—	—	500	—	ps
t _{LOCK}	Time required for the PLL to lock, after CONF_DONE signal goes high, indicating the completion of device configuration	—	—	—	1	—	—	1	—	—	1	ms

⁽⁵⁹⁾ TCCS specifications apply to I/O banks from the same side only.

⁽⁶⁰⁾ TX jitter is the jitter induced from core noise and I/O switching noise.

Emulated RSDS_E_1R Transmitter Timing Specifications

Table 39. Emulated RSDS_E_1R Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices

Emulated **RSDS_E_1R** transmitter is supported at the output pin of all I/O banks.

Symbol	Parameter	Mode	-I6, -A6, -C7, -I7			-A7			-C8, -I8			Unit
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
f _{HCLK}	Input clock frequency (high-speed I/O performance pin)	×10	5	—	85	5	—	85	5	—	85	MHz
		×8	5	—	85	5	—	85	5	—	85	MHz
		×7	5	—	85	5	—	85	5	—	85	MHz
		×4	5	—	85	5	—	85	5	—	85	MHz
		×2	5	—	85	5	—	85	5	—	85	MHz
		×1	5	—	170	5	—	170	5	—	170	MHz
HSIODR	Data rate (high-speed I/O performance pin)	×10	100	—	170	100	—	170	100	—	170	Mbps
		×8	80	—	170	80	—	170	80	—	170	Mbps
		×7	70	—	170	70	—	170	70	—	170	Mbps
		×4	40	—	170	40	—	170	40	—	170	Mbps
		×2	20	—	170	20	—	170	20	—	170	Mbps
		×1	10	—	170	10	—	170	10	—	170	Mbps
f _{HCLK}	Input clock frequency (low-speed I/O performance pin)	×10	5	—	85	5	—	85	5	—	85	MHz
		×8	5	—	85	5	—	85	5	—	85	MHz
		×7	5	—	85	5	—	85	5	—	85	MHz
		×4	5	—	85	5	—	85	5	—	85	MHz
		×2	5	—	85	5	—	85	5	—	85	MHz
		×1	5	—	170	5	—	170	5	—	170	MHz
HSIODR	Data rate (low-speed I/O performance pin)	×10	100	—	170	100	—	170	100	—	170	Mbps
		×8	80	—	170	80	—	170	80	—	170	Mbps
		×7	70	—	170	70	—	170	70	—	170	Mbps

continued...

Symbol	Parameter	Mode	-I6, -A6, -C7, -I7			-A7			-C8, -I8			Unit
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
		×4	40	—	170	40	—	170	40	—	170	Mbps
		×2	20	—	170	20	—	170	20	—	170	Mbps
		×1	10	—	170	10	—	170	10	—	170	Mbps
t _{DUTY}	Duty cycle on transmitter output clock	—	45	—	55	45	—	55	45	—	55	%
TCCS ⁽⁶¹⁾	Transmitter channel-to-channel skew	—	—	—	300	—	—	300	—	—	300	ps
t _{x jitter} ⁽⁶²⁾	Output jitter (high-speed I/O performance pin)	—	—	—	425	—	—	425	—	—	425	ps
	Output jitter (low-speed I/O performance pin)	—	—	—	470	—	—	470	—	—	470	ps
t _{RISE}	Rise time	20 – 80%, C _{LOAD} = 5 pF	—	500	—	—	500	—	—	500	—	ps
t _{FALL}	Fall time	20 – 80%, C _{LOAD} = 5 pF	—	500	—	—	500	—	—	500	—	ps
t _{LOCK}	Time required for the PLL to lock, after CONF_DONE signal goes high, indicating the completion of device configuration	—	—	—	1	—	—	1	—	—	1	ms

⁽⁶¹⁾ TCCS specifications apply to I/O banks from the same side only.

⁽⁶²⁾ TX jitter is the jitter induced from core noise and I/O switching noise.

True Mini-LVDS and Emulated Mini-LVDS_E_3R Transmitter Timing Specifications

Table 40. True Mini-LVDS and Emulated Mini-LVDS_E_3R Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices

True **mini-LVDS** transmitter is only supported at the bottom I/O banks. Emulated **mini-LVDS_E_3R** transmitter is supported at the output pin of all I/O banks.

Symbol	Parameter	Mode	-I6, -A6, -C7, -I7			-A7			-C8, -I8			Unit
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
f _{HCLK}	Input clock frequency (high-speed I/O performance pin)	×10	5	—	155	5	—	155	5	—	155	MHz
		×8	5	—	155	5	—	155	5	—	155	MHz
		×7	5	—	155	5	—	155	5	—	155	MHz
		×4	5	—	155	5	—	155	5	—	155	MHz
		×2	5	—	155	5	—	155	5	—	155	MHz
		×1	5	—	310	5	—	310	5	—	310	MHz
HSIODR	Data rate (high-speed I/O performance pin)	×10	100	—	310	100	—	310	100	—	310	Mbps
		×8	80	—	310	80	—	310	80	—	310	Mbps
		×7	70	—	310	70	—	310	70	—	310	Mbps
		×4	40	—	310	40	—	310	40	—	310	Mbps
		×2	20	—	310	20	—	310	20	—	310	Mbps
		×1	10	—	310	10	—	310	10	—	310	Mbps
f _{HCLK}	Input clock frequency (low-speed I/O performance pin)	×10	5	—	150	5	—	150	5	—	150	MHz
		×8	5	—	150	5	—	150	5	—	150	MHz
		×7	5	—	150	5	—	150	5	—	150	MHz
		×4	5	—	150	5	—	150	5	—	150	MHz
		×2	5	—	150	5	—	150	5	—	150	MHz
		×1	5	—	300	5	—	300	5	—	300	MHz
HSIODR	Data rate (low-speed I/O performance pin)	×10	100	—	300	100	—	300	100	—	300	Mbps
		×8	80	—	300	80	—	300	80	—	300	Mbps

continued...

Symbol	Parameter	Mode	-I6, -A6, -C7, -I7			-A7			-C8, -I8			Unit
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
		×7	70	—	300	70	—	300	70	—	300	Mbps
		×4	40	—	300	40	—	300	40	—	300	Mbps
		×2	20	—	300	20	—	300	20	—	300	Mbps
		×1	10	—	300	10	—	300	10	—	300	Mbps
t _{DUTY}	Duty cycle on transmitter output clock	—	45	—	55	45	—	55	45	—	55	%
TCCS ⁽⁶³⁾	Transmitter channel-to-channel skew	—	—	—	300	—	—	300	—	—	300	ps
t _{x jitter} ⁽⁶⁴⁾	Output jitter (high-speed I/O performance pin)	—	—	—	425	—	—	425	—	—	425	ps
	Output jitter (low-speed I/O performance pin)	—	—	—	470	—	—	470	—	—	470	ps
t _{RISE}	Rise time	20 – 80%, C _{LOAD} = 5 pF	—	500	—	—	500	—	—	500	—	ps
t _{FALL}	Fall time	20 – 80%, C _{LOAD} = 5 pF	—	500	—	—	500	—	—	500	—	ps
t _{LOCK}	Time required for the PLL to lock, after CONF_DONE signal goes high, indicating the completion of device configuration	—	—	—	1	—	—	1	—	—	1	ms

⁽⁶³⁾ TCCS specifications apply to I/O banks from the same side only.

⁽⁶⁴⁾ TX jitter is the jitter induced from core noise and I/O switching noise.

True LVDS Transmitter Timing

Single Supply Devices True LVDS Transmitter Timing Specifications

Table 41. True LVDS Transmitter Timing Specifications for Intel MAX 10 Single Supply Devices

True LVDS transmitter is only supported at the bottom I/O banks.

Symbol	Parameter	Mode	-C7, -I7			-A7			-C8, -I8			Unit
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
f _{HCLK}	Input clock frequency	×10	5	—	145	5	—	100	5	—	100	MHz
		×8	5	—	145	5	—	100	5	—	100	MHz
		×7	5	—	145	5	—	100	5	—	100	MHz
		×4	5	—	145	5	—	100	5	—	100	MHz
		×2	5	—	145	5	—	100	5	—	100	MHz
		×1	5	—	290	5	—	200	5	—	200	MHz
HSIODR	Data rate	×10	100	—	290	100	—	200	100	—	200	Mbps
		×8	80	—	290	80	—	200	80	—	200	Mbps
		×7	70	—	290	70	—	200	70	—	200	Mbps
		×4	40	—	290	40	—	200	40	—	200	Mbps
		×2	20	—	290	20	—	200	20	—	200	Mbps
		×1	10	—	290	10	—	200	10	—	200	Mbps
t _{DUTY}	Duty cycle on transmitter output clock	—	45	—	55	45	—	55	45	—	55	%
TCCS ⁽⁶⁵⁾	Transmitter channel-to-channel skew	—	—	—	300	—	—	300	—	—	300	ps
t _{x jitter} ⁽⁶⁶⁾	Output jitter	—	—	—	1,000	—	—	1,000	—	—	1,000	ps

continued...

⁽⁶⁵⁾ TCCS specifications apply to I/O banks from the same side only.

⁽⁶⁶⁾ TX jitter is the jitter induced from core noise and I/O switching noise.

Symbol	Parameter	Mode	-C7, -I7			-A7			-C8, -I8			Unit
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
t _{RISE}	Rise time	20 – 80%, C _{LOAD} = 5 pF	—	500	—	—	500	—	—	500	—	ps
t _{FALL}	Fall time	20 – 80%, C _{LOAD} = 5 pF	—	500	—	—	500	—	—	500	—	ps
t _{LOCK}	Time required for the PLL to lock, after CONF_DONE signal goes high, indicating the completion of device configuration	—	—	—	1	—	—	1	—	—	1	ms

Dual Supply Devices True LVDS Transmitter Timing Specifications

Table 42. True LVDS Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices

True LVDS transmitter is only supported at the bottom I/O banks.

Symbol	Parameter	Mode	-I6			-A6, -C7, -I7			-A7			-C8, -I8			Unit
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
f _{HSCLK}	Input clock frequency	×10	5	—	360	5	—	340	5	—	310	5	—	300	MHz
		×8	5	—	360	5	—	360	5	—	320	5	—	320	MHz
		×7	5	—	360	5	—	340	5	—	310	5	—	300	MHz
		×4	5	—	360	5	—	350	5	—	320	5	—	320	MHz
		×2	5	—	360	5	—	350	5	—	320	5	—	320	MHz
		×1	5	—	360	5	—	350	5	—	320	5	—	320	MHz
HSIODR	Data rate	×10	100	—	720	100	—	680	100	—	620	100	—	600	Mbps
		×8	80	—	720	80	—	720	80	—	640	80	—	640	Mbps
		×7	70	—	720	70	—	680	70	—	620	70	—	600	Mbps
		×4	40	—	720	40	—	700	40	—	640	40	—	640	Mbps
		×2	20	—	720	20	—	700	20	—	640	20	—	640	Mbps

continued...

Symbol	Parameter	Mode	-I6			-A6, -C7, -I7			-A7			-C8, -I8			Unit
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
		×1	10	—	360	10	—	350	10	—	320	10	—	320	Mbps
t _{DUTY}	Duty cycle on transmitter output clock	—	45	—	55	45	—	55	45	—	55	45	—	55	%
TCCS ⁽⁶⁷⁾	Transmitter channel-to-channel skew	—	—	—	300	—	—	300	—	—	300	—	—	300	ps
t _x Jitter ⁽⁶⁸⁾	Output jitter	—	—	—	380	—	—	380	—	—	380	—	—	380	ps
t _{RISE}	Rise time	20 – 80%, C _{LOAD} = 5 pF	—	500	—	—	500	—	—	500	—	—	500	—	ps
t _{FALL}	Fall time	20 – 80%, C _{LOAD} = 5 pF	—	500	—	—	500	—	—	500	—	—	500	—	ps
t _{LOCK}	Time required for the PLL to lock, after CONF_DONE signal goes high, indicating the completion of device configuration	—	—	—	1	—	—	1	—	—	1	—	—	1	ms

Table 43. True 1.8 V LVDS Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices

Symbol	Parameter	Mode	-I6			-C7, -I7			-C8, -I8			Unit
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
f _{HSCLK}	Input clock frequency	×10	5	—	200	5	—	200	5	—	200	MHz
		×8	5	—	200	5	—	200	5	—	200	MHz

continued...

⁽⁶⁷⁾ TCCS specifications apply to I/O banks from the same side only.

⁽⁶⁸⁾ TX jitter is the jitter induced from core noise and I/O switching noise.

Symbol	Parameter	Mode	-I6			-C7, -I7			-C8, -I8			Unit
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
		×7	5	—	200	5	—	200	5	—	200	MHz
		×4	5	—	200	5	—	200	5	—	200	MHz
		×2	5	—	200	5	—	200	5	—	200	MHz
		×1	5	—	100	5	—	100	5	—	100	MHz
HSIODR	Data rate	×10	100	—	400	100	—	400	100	—	400	Mbps
		×8	80	—	400	80	—	400	80	—	400	Mbps
		×7	70	—	400	70	—	400	70	—	400	Mbps
		×4	40	—	400	40	—	400	40	—	400	Mbps
		×2	20	—	400	20	—	400	20	—	400	Mbps
		×1	10	—	200	10	—	200	10	—	200	Mbps
t _{DUTY}	Duty cycle on transmitter output clock	—	45	—	55	45	—	55	45	—	55	%
TCCS ⁽⁶⁹⁾	Transmitter channel-to-channel skew	—	—	—	300	—	—	300	—	—	300	ps
t _{x jitter} ⁽⁷⁰⁾	Output jitter	—	—	—	380	—	—	380	—	—	380	ps
t _{RISE}	Rise time	20 – 80%, C _{LOAD} = 5 pF	—	500	—	—	500	—	—	500	—	ps
t _{FALL}	Fall time	20 – 80%, C _{LOAD} = 5 pF	—	500	—	—	500	—	—	500	—	ps
t _{LOCK}	Time required for the PLL to lock, after CONF_DONE signal goes high, indicating the completion of device configuration	—	—	—	1	—	—	1	—	—	1	ms

⁽⁶⁹⁾ TCCS specifications apply to I/O banks from the same side only.

⁽⁷⁰⁾ TX jitter is the jitter induced from core noise and I/O switching noise.

Emulated LVDS_E_3R, SLVS, and Sub-LVDS Transmitter Timing Specifications

Single Supply Devices Emulated LVDS_E_3R Transmitter Timing Specifications

Table 44. Emulated LVDS_E_3R Transmitter Timing Specifications for Intel MAX 10 Single Supply Devices

Emulated LVDS_E_3R transmitters are supported at the output pin of all I/O banks.

Symbol	Parameter	Mode	–C7, –I7			–A7			–C8, –I8			Unit
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
f _{HCLK}	Input clock frequency (high-speed I/O performance pin)	×10	5	—	142.5	5	—	100	5	—	100	MHz
		×8	5	—	142.5	5	—	100	5	—	100	MHz
		×7	5	—	142.5	5	—	100	5	—	100	MHz
		×4	5	—	142.5	5	—	100	5	—	100	MHz
		×2	5	—	142.5	5	—	100	5	—	100	MHz
		×1	5	—	285	5	—	200	5	—	200	MHz
HSIODR	Data rate (high-speed I/O performance pin)	×10	100	—	285	100	—	200	100	—	200	Mbps
		×8	80	—	285	80	—	200	80	—	200	Mbps
		×7	70	—	285	70	—	200	70	—	200	Mbps
		×4	40	—	285	40	—	200	40	—	200	Mbps
		×2	20	—	285	20	—	200	20	—	200	Mbps
		×1	10	—	285	10	—	200	10	—	200	Mbps
f _{HCLK}	Input clock frequency (low-speed I/O performance pin)	×10	5	—	100	5	—	100	5	—	100	MHz
		×8	5	—	100	5	—	100	5	—	100	MHz
		×7	5	—	100	5	—	100	5	—	100	MHz
		×4	5	—	100	5	—	100	5	—	100	MHz
		×2	5	—	100	5	—	100	5	—	100	MHz
		×1	5	—	200	5	—	200	5	—	200	MHz
HSIODR	Data rate (low-speed I/O performance pin)	×10	100	—	200	100	—	200	100	—	200	Mbps
continued...												

Symbol	Parameter	Mode	-C7, -I7			-A7			-C8, -I8			Unit
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
		×8	80	—	200	80	—	200	80	—	200	Mbps
		×7	70	—	200	70	—	200	70	—	200	Mbps
		×4	40	—	200	40	—	200	40	—	200	Mbps
		×2	20	—	200	20	—	200	20	—	200	Mbps
		×1	10	—	200	10	—	200	10	—	200	Mbps
t _{DUTY}	Duty cycle on transmitter output clock	—	45	—	55	45	—	55	45	—	55	%
TCCS ⁽⁷¹⁾	Transmitter channel-to-channel skew	—	—	—	300	—	—	300	—	—	300	ps
t _x Jitter ⁽⁷²⁾	Output jitter	—	—	—	1,000	—	—	1,000	—	—	1,000	ps
t _{RISE}	Rise time	20 – 80%, C _{LOAD} = 5 pF	—	500	—	—	500	—	—	500	—	ps
t _{FALL}	Fall time	20 – 80%, C _{LOAD} = 5 pF	—	500	—	—	500	—	—	500	—	ps
t _{LOCK}	Time required for the PLL to lock, after CONF_DONE signal goes high, indicating the completion of device configuration	—	—	—	1	—	—	1	—	—	1	ms

⁽⁷¹⁾ TCCS specifications apply to I/O banks from the same side only.

⁽⁷²⁾ TX jitter is the jitter induced from core noise and I/O switching noise.

Dual Supply Devices Emulated LVDS_E_3R, SLVS, and Sub-LVDS Transmitter Timing Specifications

Table 45. Emulated LVDS_E_3R, SLVS, and Sub-LVDS Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices

Emulated **LVDS_E_3R**, **SLVS**, and **Sub-LVDS** transmitters are supported at the output pin of all I/O banks.

Symbol	Parameter	Mode	-I6, -A6, -C7, -I7			-A7			-C8, -I8			Unit
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
f _{HSCLK}	Input clock frequency (high-speed I/O performance pin)	×10	5	—	300	5	—	275	5	—	275	MHz
		×8	5	—	300	5	—	275	5	—	275	MHz
		×7	5	—	300	5	—	275	5	—	275	MHz
		×4	5	—	300	5	—	275	5	—	275	MHz
		×2	5	—	300	5	—	275	5	—	275	MHz
		×1	5	—	300	5	—	275	5	—	275	MHz
HSIODR	Data rate (high-speed I/O performance pin)	×10	100	—	600	100	—	550	100	—	550	Mbps
		×8	80	—	600	80	—	550	80	—	550	Mbps
		×7	70	—	600	70	—	550	70	—	550	Mbps
		×4	40	—	600	40	—	550	40	—	550	Mbps
		×2	20	—	600	20	—	550	20	—	550	Mbps
		×1	10	—	300	10	—	275	10	—	275	Mbps
f _{HSCLK}	Input clock frequency (low-speed I/O performance pin)	×10	5	—	150	5	—	150	5	—	150	MHz
		×8	5	—	150	5	—	150	5	—	150	MHz
		×7	5	—	150	5	—	150	5	—	150	MHz
		×4	5	—	150	5	—	150	5	—	150	MHz
		×2	5	—	150	5	—	150	5	—	150	MHz
		×1	5	—	300	5	—	300	5	—	300	MHz
HSIODR	Data rate (low-speed I/O performance pin)	×10	100	—	300	100	—	300	100	—	300	Mbps
		×8	80	—	300	80	—	300	80	—	300	Mbps

continued...

Symbol	Parameter	Mode	-I6, -A6, -C7, -I7			-A7			-C8, -I8			Unit
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
		×7	70	—	300	70	—	300	70	—	300	Mbps
		×4	40	—	300	40	—	300	40	—	300	Mbps
		×2	20	—	300	20	—	300	20	—	300	Mbps
		×1	10	—	300	10	—	300	10	—	300	Mbps
t _{DUTY}	Duty cycle on transmitter output clock	—	45	—	55	45	—	55	45	—	55	%
TCCS ⁽⁷³⁾	Transmitter channel-to-channel skew	—	—	—	300	—	—	300	—	—	300	ps
t _{x jitter} ⁽⁷⁴⁾	Output jitter (high-speed I/O performance pin)	—	—	—	425	—	—	425	—	—	425	ps
	Output jitter (low-speed I/O performance pin)	—	—	—	470	—	—	470	—	—	470	ps
t _{RISE}	Rise time	20 – 80%, C _{LOAD} = 5 pF	—	500	—	—	500	—	—	500	—	ps
t _{FALL}	Fall time	20 – 80%, C _{LOAD} = 5 pF	—	500	—	—	500	—	—	500	—	ps
t _{LOCK}	Time required for the PLL to lock, after CONF_DONE signal goes high, indicating the completion of device configuration	—	—	—	1	—	—	1	—	—	1	ms

⁽⁷³⁾ TCCS specifications apply to I/O banks from the same side only.

⁽⁷⁴⁾ TX jitter is the jitter induced from core noise and I/O switching noise.

LVDS, TMDs, HiSpi, SLVS, and Sub-LVDS Receiver Timing Specifications

Single Supply Devices LVDS Receiver Timing Specifications

Table 46. LVDS Receiver Timing Specifications for Intel MAX 10 Single Supply Devices

LVDS receivers are supported at all banks.

Symbol	Parameter	Mode	-C7, -I7		-A7		-C8, -I8		Unit
			Min	Max	Min	Max	Min	Max	
f _{HSCLK}	Input clock frequency (high-speed I/O performance pin)	×10	5	145	5	100	5	100	MHz
		×8	5	145	5	100	5	100	MHz
		×7	5	145	5	100	5	100	MHz
		×4	5	145	5	100	5	100	MHz
		×2	5	145	5	100	5	100	MHz
		×1	5	290	5	200	5	200	MHz
HSIODR	Data rate (high-speed I/O performance pin)	×10	100	290	100	200	100	200	Mbps
		×8	80	290	80	200	80	200	Mbps
		×7	70	290	70	200	70	200	Mbps
		×4	40	290	40	200	40	200	Mbps
		×2	20	290	20	200	20	200	Mbps
		×1	10	290	10	200	10	200	Mbps
f _{HSCLK}	Input clock frequency (low-speed I/O performance pin)	×10	5	100	5	100	5	100	MHz
		×8	5	100	5	100	5	100	MHz
		×7	5	100	5	100	5	100	MHz
		×4	5	100	5	100	5	100	MHz
		×2	5	100	5	100	5	100	MHz
		×1	5	200	5	200	5	200	MHz
HSIODR	Data rate (low-speed I/O performance pin)	×10	100	200	100	200	100	200	Mbps

continued...

Symbol	Parameter	Mode	-C7, -I7		-A7		-C8, -I8		Unit
			Min	Max	Min	Max	Min	Max	
		×8	80	200	80	200	80	200	Mbps
		×7	70	200	70	200	70	200	Mbps
		×4	40	200	40	200	40	200	Mbps
		×2	20	200	20	200	20	200	Mbps
		×1	10	200	10	200	10	200	Mbps
SW	Sampling window (high-speed I/O performance pin)	—	—	910	—	910	—	910	ps
	Sampling window (low-speed I/O performance pin)	—	—	1,110	—	1,110	—	1,110	ps
$t_{x \text{ Jitter}}^{(75)}$	Input jitter	—	—	1,000	—	1,000	—	1,000	ps
t_{LOCK}	Time required for the PLL to lock, after CONF_DONE signal goes high, indicating the completion of device configuration	—	—	1	—	1	—	1	ms

Dual Supply Devices LVDS, TMDS, HiSpi, SLVS, and Sub-LVDS Receiver Timing Specifications

Table 47. LVDS, TMDS, HiSpi, SLVS, and Sub-LVDS Receiver Timing Specifications for Intel MAX 10 Dual Supply Devices

LVDS, TMDS, HiSpi, SLVS, and Sub-LVDS receivers are supported at all banks.

Symbol	Parameter	Mode	-I6, -A6, -C7, -I7		-A7		-C8, -I8		Unit
			Min	Max	Min	Max	Min	Max	
f_{HCLK}	Input clock frequency (high-speed I/O performance pin)	×10	5	350	5	320	5	320	MHz
		×8	5	360	5	320	5	320	MHz
		×7	5	350	5	320	5	320	MHz
		×4	5	360	5	320	5	320	MHz

continued...

⁽⁷⁵⁾ TX jitter is the jitter induced from core noise and I/O switching noise.

Symbol	Parameter	Mode	-I6, -A6, -C7, -I7		-A7		-C8, -I8		Unit
			Min	Max	Min	Max	Min	Max	
		×2	5	360	5	320	5	320	MHz
		×1	5	360	5	320	5	320	MHz
HSIODR	Data rate (high-speed I/O performance pin)	×10	100	700	100	640	100	640	Mbps
		×8	80	720	80	640	80	640	Mbps
		×7	70	700	70	640	70	640	Mbps
		×4	40	720	40	640	40	640	Mbps
		×2	20	720	20	640	20	640	Mbps
		×1	10	360	10	320	10	320	Mbps
f _{HCLK}	Input clock frequency (low-speed I/O performance pin)	×10	5	150	5	150	5	150	MHz
		×8	5	150	5	150	5	150	MHz
		×7	5	150	5	150	5	150	MHz
		×4	5	150	5	150	5	150	MHz
		×2	5	150	5	150	5	150	MHz
		×1	5	300	5	300	5	300	MHz
HSIODR	Data rate (low-speed I/O performance pin)	×10	100	300	100	300	100	300	Mbps
		×8	80	300	80	300	80	300	Mbps
		×7	70	300	70	300	70	300	Mbps
		×4	40	300	40	300	40	300	Mbps
		×2	20	300	20	300	20	300	Mbps
		×1	10	300	10	300	10	300	Mbps
SW	Sampling window (high-speed I/O performance pin)	—	—	510	—	510	—	510	ps
continued...									

Symbol	Parameter	Mode	-I6, -A6, -C7, -I7		-A7		-C8, -I8		Unit
			Min	Max	Min	Max	Min	Max	
	Sampling window (low-speed I/O performance pin)	—	—	910	—	910	—	910	ps
$t_{x \text{ jitter}}^{(76)}$	Input jitter	—	—	500	—	500	—	500	ps
t_{LOCK}	Time required for the PLL to lock, after CONF_DONE signal goes high, indicating the completion of device configuration	—	—	1	—	1	—	1	ms

Table 48. True 1.8 V LVDS Receiver Timing Specifications for Intel MAX 10 Dual Supply Devices

True 1.8 V LVDS receiver is only supported at the high-speed I/O banks, except high-speed DDR3 I/O banks.

Symbol	Parameter	Mode	-I6, -C7, -I7		-C8, -I8		Unit
			Min	Max	Min	Max	
f_{HSCLK}	Input clock frequency (bottom-bank I/O performance pin)	×10	5	200	5	200	MHz
		×8	5	200	5	200	MHz
		×7	5	200	5	200	MHz
		×4	5	200	5	200	MHz
		×2	5	200	5	200	MHz
		×1	5	100	5	100	MHz
HSIODR	Data rate (bottom-bank I/O performance pin)	×10	100	400	100	400	Mbps
		×8	80	400	80	400	Mbps
		×7	70	400	70	400	Mbps
		×4	40	400	40	400	Mbps
		×2	20	400	20	400	Mbps
		×1	10	200	10	200	Mbps

continued...

⁽⁷⁶⁾ TX jitter is the jitter induced from core noise and I/O switching noise.

Symbol	Parameter	Mode	-I6, -C7, -I7		-C8, -I8		Unit
			Min	Max	Min	Max	
SW	Sampling window (high-speed I/O performance pin)	—	—	510	—	510	ps
$t_{x \text{ Jitter}}^{(77)}$	Input jitter	—	—	500	—	500	ps
t_{LOCK}	Time required for the PLL to lock, after CONF_DONE signal goes high, indicating the completion of device configuration	—	—	1	—	1	ms

Memory Standards Supported by the Soft Memory Controller

Table 49. Memory Standards Supported by the Soft Memory Controller for Intel MAX 10 Devices

External Memory Interface Standard	Rate Support	Speed Grade	Voltage (V)	Max Frequency (MHz)
DDR3 SDRAM	Half	-I6	1.5	303
DDR3L SDRAM	Half	-I6	1.35	303
DDR2 SDRAM	Half	-I6	1.8	200
		-I7 and -C7		167
LPDDR2 ⁽⁷⁸⁾	Half	-I6	1.2	200 ⁽⁷⁹⁾

Note: Automotive-grade Intel MAX 10 devices do not support external memory interfaces.

Related Information

[External Memory Interface Spec Estimator](#)

Provides the specific details of the memory standards supported.

⁽⁷⁷⁾ TX jitter is the jitter induced from core noise and I/O switching noise.

⁽⁷⁸⁾ Intel MAX 10 devices support only single-die LPDDR2.

⁽⁷⁹⁾ To achieve the specified performance, constrain the memory device I/O and core power supply variation to within $\pm 3\%$. By default, the frequency is 167 MHz.

Memory Output Clock Jitter Specifications

Intel MAX 10 devices support external memory interfaces up to 303 MHz. The external memory interfaces for Intel MAX 10 devices calibrate automatically.

The memory output clock jitter measurements are for 200 consecutive clock cycles.

The clock jitter specification applies to memory output clock pins generated using DDIO circuits clocked by a PLL output routed on a PHY clock network.

DDR3 and LPDDR2 SDRAM memory interfaces are only supported on the fast speed grade device.

Table 50. Memory Output Clock Jitter Specifications for Intel MAX 10 Devices

Parameter	Symbol	–6 Speed Grade		–7 Speed Grade		Unit
		Min	Max	Min	Max	
Clock period jitter	$t_{JIT(per)}$	–127	127	–215	215	ps
Cycle-to-cycle period jitter	$t_{JIT(cc)}$	—	242	—	360	ps

Related Information

[Literature: External Memory Interfaces](#)

Provides more information about external memory system performance specifications, board design guidelines, timing analysis, simulation, and debugging information.

Configuration Specifications

This section provides configuration specifications and timing for Intel MAX 10 devices.

JTAG Timing Parameters

Table 51. JTAG Timing Parameters for Intel MAX 10 Devices

The values are based on $C_L = 10$ pF of TDO.

The affected Boundary Scan Test (BST) instructions are SAMPLE/PRELOAD, EXTEST, INTEST, and CHECK_STATUS.

Symbol	Parameter	Non-BST and non-CONFIG_IO Operation		BST and CONFIG_IO Operation		Unit
		Minimum	Maximum	Minimum	Maximum	
t_{JCP}	TCK clock period	40	—	50	—	ns
t_{JCH}	TCK clock high time	20	—	25	—	ns
t_{JCL}	TCK clock low time	20	—	25	—	ns
t_{JPSU_TDI}	JTAG port setup time	2	—	2	—	ns
t_{JPSU_TMS}	JTAG port setup time	3	—	3	—	ns
t_{JPH}	JTAG port hold time	10	—	10	—	ns
t_{JPCO}	JTAG port clock to output	—	<ul style="list-style-type: none"> 15 (for $V_{CCIO} = 3.3, 3.0,$ and 2.5 V) 17 (for $V_{CCIO} = 1.8$ and 1.5 V) 	—	<ul style="list-style-type: none"> 18 (for $V_{CCIO} = 3.3, 3.0,$ and 2.5 V) 20 (for $V_{CCIO} = 1.8$ and 1.5 V) 	ns
t_{JPZX}	JTAG port high impedance to valid output	—	<ul style="list-style-type: none"> 15 (for $V_{CCIO} = 3.3, 3.0,$ and 2.5 V) 17 (for $V_{CCIO} = 1.8$ and 1.5 V) 	—	<ul style="list-style-type: none"> 15 (for $V_{CCIO} = 3.3, 3.0,$ and 2.5 V) 17 (for $V_{CCIO} = 1.8$ and 1.5 V) 	ns
t_{JPXZ}	JTAG port valid output to high impedance	—	<ul style="list-style-type: none"> 15 (for $V_{CCIO} = 3.3, 3.0,$ and 2.5 V) 17 (for $V_{CCIO} = 1.8$ and 1.5 V) 	—	<ul style="list-style-type: none"> 15 (for $V_{CCIO} = 3.3, 3.0,$ and 2.5 V) 17 (for $V_{CCIO} = 1.8$ and 1.5 V) 	ns

Remote System Upgrade Circuitry Timing Specifications

Table 52. Remote System Upgrade Circuitry Timing Specifications for Intel MAX 10 Devices

Parameter	Device	Minimum	Maximum	Unit
t _{MAX_RU_CLK}	All	—	40	MHz
t _{RU_nCONFIG}	10M02, 10M04, 10M08, 10M16, 10M25	250	—	ns
	10M40, 10M50	350	—	ns
t _{RU_nRSTIMER}	10M02, 10M04, 10M08, 10M16, 10M25	300	—	ns
	10M40, 10M50	500	—	ns

User Watchdog Internal Circuitry Timing Specifications

Table 53. User Watchdog Timer Specifications for Intel MAX 10 Devices

The specifications are subject to PVT changes.

Parameter	Device	Minimum	Typical	Maximum	Unit
User watchdog frequency	10M02, 10M04, 10M08, 10M16, 10M25	3.4	5.1	7.3	MHz
	10M40, 10M50	2.2	3.3	4.8	MHz

Uncompressed Raw Binary File (.rbf) Sizes

Table 54. Uncompressed .rbf Sizes for Intel MAX 10 Devices

Device	CFM Data Size (bits)	
	Without Memory Initialization	With Memory Initialization
10M02/10M02SCU324	554,000/1,540,000	—
10M04	1,540,000	1,880,000
10M08	1,540,000	1,880,000
10M16	2,800,000	3,430,000
<i>continued...</i>		

Device	CFM Data Size (bits)	
	Without Memory Initialization	With Memory Initialization
10M25	4,140,000	4,780,000
10M40	7,840,000	9,670,000
10M50	7,840,000	9,670,000

Internal Configuration Time

The internal configuration time measurement is from the rising edge of nSTATUS signal to the rising edge of CONF_DONE signal.

Table 55. Internal Configuration Time for Intel MAX 10 Devices (Uncompressed .rbf)

Device	Internal Configuration Time (ms)							
	Unencrypted				Encrypted			
	Without Memory Initialization		With Memory Initialization		Without Memory Initialization		With Memory Initialization	
	Min	Max	Min	Max	Min	Max	Min	Max
10M02/ 10M02SCU324	0.3/0.6	1.7/2.7	—	—	1.7/5.0	5.4/15.0	—	—
10M04	0.6	2.7	1.0	3.4	5.0	15.0	6.8	19.6
10M08	0.6	2.7	1.0	3.4	5.0	15.0	6.8	19.6
10M16	1.1	3.7	1.4	4.5	9.3	25.3	11.7	31.5
10M25	1.0	3.7	1.3	4.4	14.0	38.1	16.9	45.7
10M40	2.6	6.9	3.2	9.8	41.5	112.1	51.7	139.6
10M50	2.6	6.9	3.2	9.8	41.5	112.1	51.7	139.6

Table 56. Internal Configuration Time for Intel MAX 10 Devices (Compressed .rbf)

Compression ratio depends on design complexity. The minimum value is based on the best case (25% of original .rbf sizes) and the maximum value is based on the typical case (70% of original .rbf sizes).

Device	Internal Configuration Time (ms)			
	Unencrypted/Encrypted			
	Without Memory Initialization		With Memory Initialization	
	Min	Max	Min	Max
10M02/10M02SCU324	0.3/0.6	5.2/10.7	—	—
10M04	0.6	10.7	1.0	13.9
10M08	0.6	10.7	1.0	13.9
10M16	1.1	17.9	1.4	22.3
10M25	1.1	26.9	1.4	32.2
10M40	2.6	66.1	3.2	82.2
10M50	2.6	66.1	3.2	82.2

Internal Configuration Timing Parameter

Table 57. Internal Configuration Timing Parameter for Intel MAX 10 Devices

Symbol	Parameter	Device	Minimum	Maximum	Unit
t _{CD2UM}	CONF_DONE high to user mode	10M02, 10M04, 10M08, 10M16, 10M25	182.8	385.5	μs
		10M40, 10M50	275.3	605.7	μs

I/O Timing

The data is typically used prior to designing the FPGA to get an estimate of the timing budget as part of the link timing analysis.

The Intel Quartus Prime Timing Analyzer provides a more accurate and precise I/O timing data based on the specific device and design after you complete place-and-route.

Table 58. I/O Timing for Intel MAX 10 Devices

These I/O timing parameters are for the 3.3-V LVTTTL I/O standard with the maximum drive strength and fast slew rate for 10M08DAF484 device.

Symbol	Parameter	–C7, –I7	–C8, –I8	Unit
T _{su}	Global clock setup time	–0.750	–0.808	ns
T _h	Global clock hold time	1.180	1.215	ns
T _{co}	Global clock to output delay	5.131	5.575	ns
T _{pd}	Best case pin-to-pin propagation delay through one LUT	4.907	5.467	ns

Programmable IOE Delay

Programmable IOE Delay On Row Pins

Table 59. IOE Programmable Delay on Row Pins for Intel MAX 10 Devices

The incremental values for the settings are generally linear. For exact values of each setting, refer to the **Assignment Name** column in the latest version of the Intel Quartus Prime software.

The minimum and maximum offset timing numbers are in reference to setting '0' as available in the Intel Quartus Prime software.

Parameter	Paths Affected	Number of Settings	Minimum Offset	Maximum Offset							Unit
				Fast Corner		Slow Corner					
				−I7	−C8, −I8	−A6	−C7	−C8, −I8	−I7	−A7	
Input delay from pin to internal cells	Pad to I/O dataout to core	7	0	0.815	0.873	1.831	1.811	1.874	1.871	1.922	ns
Input delay from pin to input register	Pad to I/O input register	8	0	0.924	0.992	2.081	2.055	2.125	2.127	2.185	ns
Delay from output register to output pin	I/O output register to pad	2	0	0.479	0.514	1.069	1.070	1.117	1.105	1.134	ns

Programmable IOE Delay for Column Pins

Table 60. IOE Programmable Delay on Column Pins for Intel MAX 10 Devices

The incremental values for the settings are generally linear. For exact values of each setting, refer to the **Assignment Name** column in the latest version of the Intel Quartus Prime software.

The minimum and maximum offset timing numbers are in reference to setting '0' as available in the Intel Quartus Prime software.

Parameter	Paths Affected	Number of Settings	Minimum Offset	Maximum Offset							Unit
				Fast Corner		Slow Corner					
				-I7	-C8, -I8	-A6	-C7	-C8, -I8	-I7	-A7	
Input delay from pin to internal cells	Pad to I/O dataout to core	7	0	0.81	0.868	1.823	1.802	1.864	1.862	1.912	ns
Input delay from pin to input register	Pad to I/O input register	8	0	0.914	0.981	2.06	2.032	2.101	2.102	2.161	ns
Delay from output register to output pin	I/O output register to pad	2	0	0.435	0.466	0.971	0.97	1.013	1.001	1.028	ns

Glossary

Table 61. Glossary

Term	Definition
JTAG Timing Specifications	<p>The diagram illustrates the JTAG timing specifications. It shows four signals: TMS, TDI, TCK, and TDO. TMS and TDI are shown as high-impedance inputs. TCK is the clock signal. TDO is the data output. The timing parameters are defined as follows: t_{JCP} (JTAG capture period), t_{JCH} (JTAG capture high), t_{JCL} (JTAG capture low), t_{JPSU} (JTAG period setup), t_{JPH} (JTAG period high), t_{JPZX} (JTAG period zero), t_{JPCO} (JTAG period clock output), and t_{JPXZ} (JTAG period zero).</p>
R_L	Receiver differential input discrete resistor (external to Intel MAX 10 devices).
RSKM (Receiver input skew margin)	HIGH-SPEED I/O block: The total margin left after accounting for the sampling window and TCCS. $RSKM = (TUI - SW - TCCS) / 2$.
Sampling window (SW)	HIGH-SPEED I/O Block: The period of time during which the data must be valid to capture it correctly. The setup and hold times determine the ideal strobe position in the sampling window.
Single-ended voltage referenced I/O standard	<p>The AC input signal values indicate the voltage levels at which the receiver must meet its timing specifications. The DC input signal values indicate the voltage levels at which the final logic state of the receiver is unambiguously defined. After the receiver input crosses the AC value, the receiver changes to the new logic state.</p> <p>The new logic state is then maintained as long as the input stays beyond the DC threshold. This approach is intended to provide predictable receiver timing in the presence of input waveform ringing.</p>
t_c	High-speed receiver/transmitter input and output clock period.
TCCS (Channel-to- channel-skew)	HIGH-SPEED I/O block: The timing difference between the fastest and slowest output edges, including t_{CO} variation and clock skew. The clock is included in the TCCS measurement.
t_{cin}	Delay from clock pad to I/O input register.
t_{CO}	Delay from clock pad to I/O output.
t_{cout}	Delay from clock pad to I/O output register.
continued...	

Term	Definition
t_{DUTY}	HIGH-SPEED I/O Block: Duty cycle on high-speed transmitter output clock.
t_{FALL}	Signal high-to-low transition time (80–20%).
t_H	Input register hold time.
Timing Unit Interval (TUI)	HIGH-SPEED I/O block: The timing budget allowed for skew, propagation delays, and data sampling window. (TUI = $1/(\text{Receiver Input Clock Frequency Multiplication Factor}) = t_C/w$).
$t_{INJITTER}$	Period jitter on PLL clock input.
$t_{OUTJITTER_DEDCLK}$	Period jitter on dedicated clock output driven by a PLL.
$t_{OUTJITTER_IO}$	Period jitter on general purpose I/O driven by a PLL.
t_{pllcin}	Delay from PLL inclk pad to I/O input register.
$t_{pllcout}$	Delay from PLL inclk pad to I/O output register.
t_{RISE}	Signal low-to-high transition time (20–80%).
t_{SU}	Input register setup time.
$V_{CM(DC)}$	DC common mode input voltage.
$V_{DIF(AC)}$	AC differential input voltage: The minimum AC input differential voltage required for switching.
$V_{DIF(DC)}$	DC differential input voltage: The minimum DC input differential voltage required for switching.
V_{HYS}	Hysteresis for Schmitt trigger input.
V_{ICM}	Input common mode voltage: The common mode of the differential signal at the receiver.
V_{ID}	Input differential Voltage Swing: The difference in voltage between the positive and complementary conductors of a differential transmission at the receiver.
V_{IH}	Voltage input high: The minimum positive voltage applied to the input which is accepted by the device as a logic high.
$V_{IH(AC)}$	High-level AC input voltage.
$V_{IH(DC)}$	High-level DC input voltage.
V_{IL}	Voltage input low: The maximum positive voltage applied to the input which is accepted by the device as a logic low.
$V_{IL(AC)}$	Low-level AC input voltage.
$V_{IL(DC)}$	Low-level DC input voltage.
V_{IN}	DC input voltage.
continued...	

Term	Definition
V _{OCM}	Output common mode voltage: The common mode of the differential signal at the transmitter.
V _{OD}	Output differential voltage swing: The difference in voltage between the positive and complementary conductors of a differential transmission line at the transmitter. $V_{OD} = V_{OH} - V_{OL}$.
V _{OH}	Voltage output high: The maximum positive voltage from an output which the device considers is accepted as the minimum positive high level.
V _{OL}	Voltage output low: The maximum positive voltage from an output which the device considers is accepted as the maximum positive low level.
V _{OS}	Output offset voltage: $V_{OS} = (V_{OH} + V_{OL}) / 2$.
V _{OX (AC)}	AC differential Output cross point voltage: The voltage at which the differential output signals must cross.
V _{REF}	Reference voltage for SSTL, HSTL, and HSUL I/O Standards.
V _{REF(AC)}	AC input reference voltage for SSTL, HSTL, and HSUL I/O Standards. $V_{REF(AC)} = V_{REF(DC)} + \text{noise}$. The peak-to-peak AC noise on V _{REF} should not exceed 2% of V _{REF(DC)} .
V _{REF(DC)}	DC input reference voltage for SSTL, HSTL, and HSUL I/O Standards.
V _{SWING (AC)}	AC differential input voltage: AC Input differential voltage required for switching.
V _{SWING (DC)}	DC differential input voltage: DC Input differential voltage required for switching.
V _{TT}	Termination voltage for SSTL, HSTL, and HSUL I/O Standards.
V _{X (AC)}	AC differential Input cross point voltage: The voltage at which the differential input signals must cross.

Document Revision History for the Intel MAX 10 FPGA Device Datasheet

Document Version	Changes
2022.10.31	<ul style="list-style-type: none"> Added support for –I8 speed grade. Updated footnote to 1.0 V LVCMOS I/O standard in the <i>Single-Ended I/O Standards Specifications</i> table. Added specifications for 1.8 V LVDS I/O standard in the <i>Differential I/O Standards Specifications for Intel MAX 10 Devices</i> table. Added the following tables: <ul style="list-style-type: none"> True 1.8 V LVDS Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices True 1.8 V LVDS Receiver Timing Specifications for Intel MAX 10 Dual Supply Devices Updated the <i>Memory Standards Supported by the Soft Memory Controller for Intel MAX 10 Devices</i> table. <ul style="list-style-type: none"> Removed contact information for –A6 speed grade devices. Added note on automotive-grade devices do not support external memory interfaces.
2021.11.01	<ul style="list-style-type: none"> Updated footnote for 1.0 V LVCMOS to include new devices in the <i>Single-Ended I/O Standards Specifications for Intel MAX 10 Devices</i> table. Removed –I6 speed grade from contact information in the following tables. All OPNs for –I6 speed grade are available in the Intel Quartus Prime Standard Edition software version 21.1 onwards. <ul style="list-style-type: none"> Intel MAX 10 Device Grades and Speed Grades Supported Memory Standards Supported by the Soft Memory Controller for Intel MAX 10 Devices
2020.06.30	<ul style="list-style-type: none"> Added V_{CCIO} specifications for 1.0 V in the following tables: <ul style="list-style-type: none"> Power Supplies Recommended Operating Conditions for Intel MAX 10 Single Supply Devices Power Supplies Recommended Operating Conditions for Intel MAX 10 Dual Supply Devices Added 1.0 V LVCMOS specifications in the <i>Single-Ended I/O Standards Specifications for Intel MAX 10 Devices</i> table. Added specifications for 10M02SCU324 device in the following tables: <ul style="list-style-type: none"> Uncompressed .rbf Sizes for Intel MAX 10 Devices Internal Configuration Time for Intel MAX 10 Devices (Uncompressed .rbf) Internal Configuration Time for Intel MAX 10 Devices (Compressed .rbf)
2018.06.29	<ul style="list-style-type: none"> Removed links on instant-on feature. Added JTAG timing specifications term in <i>Glossary</i>. Renamed the following IP cores as per Intel rebranding: <ul style="list-style-type: none"> Renamed Altera Modular ADC IP core to Modular ADC core Intel FPGA IP core. Renamed Altera Modular Dual ADC IP core to Modular Dual ADC core Intel FPGA IP core.

Date	Version	Changes
December 2017	2017.12.15	<ul style="list-style-type: none"> Removed the units for "Input resistance" and "Input capacitance" parameters in the following tables: <ul style="list-style-type: none"> ADC Performance Specifications for Intel MAX 10 Single Supply Devices ADC Performance Specifications for Intel MAX 10 Dual Supply Devices Removed the specification with memory initialization for 10M02 device in the <i>Uncompressed .rbf Sizes for Intel MAX 10 Devices</i> table.
June 2017	2017.06.16	<ul style="list-style-type: none"> Added notes for T_j for Industrial and Automotive devices in Recommended Operating Conditions for Intel MAX 10 Devices table. Updated the parameter in Internal Weak Pull-Up Resistor for Intel MAX 10 Devices table. Changed "Performance" to "Frequency" in UFM Performance Specifications for Intel MAX 10 Devices table. Removed PowerPlay text from tool name.
February 2017	2017.02.21	<ul style="list-style-type: none"> Rebranded as Intel.
October 2016	2016.10.31	<ul style="list-style-type: none"> Updated the note to the Intel MAX 10 Device Grades and Speed Grades Supported table. Updated the Memory Standards Supported by the Soft Memory Controller for Intel MAX 10 Devices table.
May 2016	2016.05.02	<ul style="list-style-type: none"> Updated t_{RAMP} specifications in Recommended Operating Conditions for Intel MAX 10 Devices table. <ul style="list-style-type: none"> Removed standard POR and fast POR specifications. Updated maximum value from 3 ms to 10 ms and added a not for the minimum value. Added Supply Current and Power Consumption section. Added the following tables: <ul style="list-style-type: none"> Memory Standards Supported by the Soft Memory Controller for Intel MAX 10 Devices Internal Configuration Timing Parameter for Intel MAX 10 Devices Removed POR Delay Specifications for Intel MAX 10 Devices table. Updated the description in the Internal Configuration Time section. Updated the following tables: <ul style="list-style-type: none"> Internal Configuration Time for Intel MAX 10 Devices (Uncompressed .rbf) Internal Configuration Time for Intel MAX 10 Devices (Compressed .rbf)
continued...		

Date	Version	Changes
January 2016	2016.01.22	<ul style="list-style-type: none"> Added description about automotive temperature devices in the Programming/Erase Specifications table. Changed the pin capacitance to maximum values. Updated maximum TCCS specifications from 410 ps to 300 ps in the following tables: <ul style="list-style-type: none"> True PPDS and Emulated PPDS_E_3R Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices True RSDS and Emulated RSDS_E_3R Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices Emulated RSDS_E_1R Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices True Mini-LVDS and Emulated Mini-LVDS_E_3R Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices True LVDS Transmitter Timing Specifications for Intel MAX 10 Single Supply Devices True LVDS Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices Emulated LVDS_E_3R Transmitter Timing Specifications for Intel MAX 10 Single Supply Devices Emulated LVDS_E_3R, SLVS, and Sub-LVDS Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices Added new table: True RSDS and Emulated RSDS_E_3R Transmitter Timing Specifications for Intel MAX 10 Single Supply Devices. Updated maximum f_{HCLK} and HSIO DR specifications for -A6, -C7, and -I7 speed grades in True LVDS Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices table. Updated SW specifications in the following tables: <ul style="list-style-type: none"> LVDS Receiver Timing Specifications for Intel MAX 10 Single Supply Devices LVDS, TMDS, HiSpi, SLVS, and Sub-LVDS Receiver Timing Specifications for Intel MAX 10 Dual Supply Devices Updated maximum f_{HCLK} and HSIO DR (high-speed I/O performance pin) specifications for -I6, -A6, -C7, -I7 speed grades in LVDS, TMDS, HiSpi, SLVS, and Sub-LVDS Receiver Timing Specifications for Intel MAX 10 Dual Supply Devices table. Removed Internal Configuration Time information in the Uncompressed .rbf Sizes for Intel MAX 10 Devices table. Added Internal Configuration Time tables for uncompressed .rbf files and compressed .rbf files. Removed Preliminary tags for all tables.
November 2015	2015.11.02	<ul style="list-style-type: none"> Added description to <i>Maximum Allowed Overshoot During Transitions over a 11.4-Year Time Frame</i> topic. Added ADC_VREF Pin Leakage Current for Intel MAX 10 Devices table. Updated the condition for "Bus-hold high, sustaining current" parameter from "$V_{IN} < V_{IL}$ (minimum)" to "$V_{IN} < V_{IH}$ (minimum)" in Bus Hold Parameters table.
continued...		

Date	Version	Changes
		<ul style="list-style-type: none"> Added –A6 speed grade in the following tables: <ul style="list-style-type: none"> Intel MAX 10 Device Grades and Speed Grades Supported Series OCT without Calibration Specifications for Intel MAX 10 Devices Clock Tree Specifications for Intel MAX 10 Devices Embedded Multiplier Specifications for Intel MAX 10 Devices Memory Block Performance Specifications for Intel MAX 10 Devices True PPDS and Emulated PPDS_E_3R Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices True RSDS and Emulated RSDS_E_3R Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices Emulated RSDS_E_1R Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices True Mini-LVDS and Emulated Mini-LVDS_E_3R Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices True LVDS Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices Emulated LVDS_E_3R, SLVS, and Sub-LVDS Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices LVDS, TMDS, HiSpi, SLVS, and Sub-LVDS Receiver Timing Specifications for Intel MAX 10 Dual Supply Devices IOE Programmable Delay on Row Pins for Intel MAX 10 Devices IOE Programmable Delay on Column Pins for Intel MAX 10 Devices Updated the maximum value for input clock cycle-to-cycle jitter ($t_{INJITTER_CCJ}$) with $F_{INPFD} < 100$ MHz condition from 750 ps to ± 750 ps in PLL Specifications for Intel MAX 10 Devices table. Updated the dual supply mode performance in Embedded Multiplier Specifications for Intel MAX 10 Devices table. Updated the dual supply mode performance in Memory Block Performance Specifications for Intel MAX 10 Devices table. Added typical specifications in Internal Oscillator Frequencies for Intel MAX 10 Devices table. Updated specifications in UFM Performance Specifications for Intel MAX 10 Devices table. Updated sampling window specifications in LVDS, TMDS, HiSpi, SLVS, and Sub-LVDS Receiver Timing Specifications for Intel MAX 10 Dual Supply Devices table. Updated IOE programmable delay for row and column pins. Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i>.
June 2015	2015.06.12	<ul style="list-style-type: none"> Updated the maximum values in Internal Weak Pull-Up Resistor for Intel MAX 10 Devices table. Removed Internal Weak Pull-Up Resistor equation. Updated the note for input resistance and input capacitance parameters in the ADC Performance Specifications table for both single supply and dual supply devices. Note: Download the SPICE models for simulation. Added a note to AC Accuracy - THD, SNR, and SINAD parameters in the ADC Performance Specifications for Intel MAX 10 Dual Supply Devices table. Note: When using internal V_{REF}, THD = 66 dB, SNR = 58 dB and SINAD = 57.5 dB for dedicated ADC input channels. Updated clock period jitter and cycle-to-cycle period jitter parameters in the Memory Output Clock Jitter Specifications for Intel MAX 10 Devices table.
continued...		

Date	Version	Changes
May 2015	2015.05.04	<ul style="list-style-type: none"> Updated a note to V_{CCIO} for both single supply and dual supply power supplies recommended operating conditions tables. Note updated: V_{CCIO} for all I/O banks must be powered up during user mode because V_{CCIO} I/O banks are used for the ADC and I/O functionalities. Updated Example for OCT Resistance Calculation after Calibration at Device Power-Up. Removed a note to BLVDS in Differential I/O Standards Specifications for Intel MAX 10 Devices table. BLVDS is now supported in Intel MAX 10 single supply devices. Note removed: BLVDS TX is not supported in single supply devices. Updated ADC Performance Specifications for both single supply and dual supply devices. <ul style="list-style-type: none"> Changed the symbol for Operating junction temperature range parameter from T_A to T_J. Edited sampling rate maximum value from 1000 kSPS to 1 MSPS. Added a note to analog input voltage parameter. Removed input frequency, f_{IN} specification. Updated the condition for DNL specification: External V_{REF}, no missing code. Added DNL specification for condition: Internal V_{REF}, no missing code. Added notes to AC accuracy specifications that the value with prescaler enabled is 6dB less than the specification. Added a note to On-Chip Temperature Sensor (absolute accuracy) parameter about the averaging calculation. Updated ADC Performance Specifications for Intel MAX 10 Single Supply Devices table. <ul style="list-style-type: none"> Added condition for On-Chip Temperature Sensor (absolute accuracy) parameter: with 64 samples averaging. Updated ADC Performance Specifications for Intel MAX 10 Dual Supply Devices table. <ul style="list-style-type: none"> Updated Digital Supply Voltage minimum value from 1.14 V to 1.15 V and maximum value from 1.26 V to 1.25 V. Updated f_{HCLK} and HSIODR specifications for -A7 speed grade in the following tables: <ul style="list-style-type: none"> True PPDS and Emulated PPDS_E_3R Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices True RSDS and Emulated RSDS_E_3R Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices True Mini-LVDS and Emulated Mini-LVDS_E_3R Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices True LVDS Transmitter Timing Specifications for Intel MAX 10 Single Supply Devices True LVDS Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices Emulated LVDS_E_3R Transmitter Timing Specifications for Intel MAX 10 Single Supply Devices Emulated LVDS_E_3R, SLVS, and Sub-LVDS Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices LVDS Receiver Timing Specifications for Intel MAX 10 Single Supply Devices LVDS, TMDS, HiSpi, SLVS, and Sub-LVDS Receiver Timing Specifications for Intel MAX 10 Dual Supply Devices

continued...

Date	Version	Changes
		<ul style="list-style-type: none"> Updated TCCS specifications in the following tables: <ul style="list-style-type: none"> True PPDS and Emulated PPDS_E_3R Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices True RSDS and Emulated RSDS_E_3R Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices Emulated RSDS_E_1R Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices True Mini-LVDS and Emulated Mini-LVDS_E_3R Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices True LVDS Transmitter Timing Specifications for Intel MAX 10 Single Supply Devices True LVDS Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices Emulated LVDS_E_3R Transmitter Timing Specifications for Intel MAX 10 Single Supply Devices Emulated LVDS_E_3R, SLVS, and Sub-LVDS Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices Updated t_{x_jitter} specifications in the following tables: <ul style="list-style-type: none"> True PPDS and Emulated PPDS_E_3R Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices True RSDS and Emulated RSDS_E_3R Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices Emulated RSDS_E_1R Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices True Mini-LVDS and Emulated Mini-LVDS_E_3R Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices True LVDS Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices Emulated LVDS_E_3R, SLVS, and Sub-LVDS Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices Updated SW specifications in LVDS Receiver Timing Specifications for Intel MAX 10 Single Supply Devices table. Added a note to t_{x_jitter} for all LVDS tables. Note: TX jitter is the jitter induced from core noise and I/O switching noise. Updated the description for t_{LOCK} for all LVDS tables: Time required for the PLL to lock, after CONF_DONE signal goes high, indicating the completion of device configuration. Updated Memory Output Clock Jitter Specifications section. <ul style="list-style-type: none"> Updated maximum external memory interfaces frequency from 300 MHz to 303 MHz. Updated PLL output routing from global clock network to PHY clock network. Added I/O Timing for Intel MAX 10 Devices table. Added V_{HYS} in the Glossary table.
January 2015	2015.01.23	<ul style="list-style-type: none"> Removed a note to V_{CCA} in Power Supplies Recommended Operating Conditions for Intel MAX 10 Dual Supply Devices table. This note is not valid: All V_{CCA} pins must be connected together for EQFP package. Corrected the maximum value for $t_{OUTJITTER_CCJ_IO}$ ($F_{OUT} \geq 100$ MHz) from 60 ps to 650 ps in PLL Specifications for Intel MAX 10 Devices table.
December 2014	2014.12.15	<ul style="list-style-type: none"> Restructured Programming/Erase Specifications for Intel MAX 10 Devices table to add temperature specifications that affect the data retention duration. Added statements in the I/O Pin Leakage Current section: Input channel leakage of ADC I/O pins due to hot socket is up to maximum of 1.8 mA. The input channel leakage occurs when the ADC IP core is enabled or disabled. This is applicable to all Intel MAX 10 devices with ADC IP core, which are 10M04, 10M08, 10M16, 10M25, 10M40, and 10M50 devices. The ADC I/O pins are in Bank 1A. Added a statement in the I/O Standards Specifications section: You must perform timing closure analysis to determine the maximum achievable frequency for general purpose I/O standards.

continued...

Date	Version	Changes
		<ul style="list-style-type: none"> Updated SSTL-2 Class I and II I/O standard specifications for JEDEC compliance as follows: <ul style="list-style-type: none"> VIL(AC) Max: Updated from $V_{REF} - 0.35$ to $V_{REF} - 0.31$ VIH(AC) Min: Updated from $V_{REF} + 0.35$ to $V_{REF} + 0.31$ Added a note to BLVDS in Differential I/O Standards Specifications for Intel MAX 10 Devices table: BLVDS TX is not supported in single supply devices. Added a link to MAX 10 High-Speed LVDS I/O User Guide for the list of I/O standards supported in single supply and dual supply devices. Added a statement in PLL Specifications for Intel MAX 10 Single Supply Device table: For V36 package, the PLL specification is based on single supply devices. Added Internal Oscillator Specifications from Intel MAX 10 Clocking and PLL User Guide. Added UFM specifications for serial interface. Updated total harmonic distortion (THD) specifications as follows: <ul style="list-style-type: none"> Single supply devices: Updated from 65 dB to -65 dB Dual supply devices: Updated from 70 dB to -70 dB (updated from 65 dB to -65 dB for dual function pin) Added condition for On-Chip Temperature Sensor—Absolute accuracy parameter in ADC Performance Specifications for Intel MAX 10 Dual Supply Devices table. The condition is: with 64 samples averaging. Updated the description in Periphery Performance Specifications to mention that proper timing closure is required in design. Updated HSIODR and f_{HCLK} specifications for x10 and x7 modes in True LVDS Transmitter Timing Specifications for Intel MAX 10 Dual Supply Devices. Added specifications for low-speed I/O performance pin sampling window in LVDS Receiver Timing Specifications for Intel MAX 10 Single Supply Devices table: Max = 900 ps for -C7, -I7, -A7, and -C8 speed grades. Added $t_{RU_nCONFIG}$ and $t_{RU_nRSTIMER}$ specifications for different devices in Remote System Upgrade Circuitry Timing Specifications for Intel MAX 10 Devices table. Removed the word "internal oscillator" in User Watchdog Timer Specifications for Intel MAX 10 Devices table to avoid confusion. Added IOE programmable delay specifications.
September 2014	2014.09.22	Initial release.



Intel® MAX® 10 Device Errata



Online Version

Send Feedback

ES-1054

683614

2016.05.16



Contents

1. Intel® MAX® 10 Device Errata..... 3

 1.1. Configuration Failure..... 3

 1.2. Revision History for Intel MAX 10 Device Errata..... 4

1. Intel® MAX® 10 Device Errata

This errata sheet provides information about known device issues affecting Intel® MAX® 10 production devices.

Table 1. Issues

Issue	Affected Devices	Planned Fix
Configuration Failure on page 3 Intermittent configuration failure when configured to start up in fast or slow POR delay mode	All Intel MAX 10 devices with date code prior to 1625	All Intel MAX 10 devices with date code 1625 or later

1.1. Configuration Failure

Description

Intel MAX 10 production devices shipped prior to date code 1625 may experience intermittent configuration failures. They may fail to enter user mode if the Power On Reset (POR) scheme is configured to operate in either **Fast POR delay** or **Slow POR delay** mode. When observed, devices may be power cycled to reconfigure or pulse low either the `nCONFIG` or the `nSTATUS` pins.

Designs configured to use the **Instant ON** mode of operation are not affected by this issue and will continue to operate without any problems.

Workaround

If your existing design POR scheme is configured to use either **Fast POR delay** or **Slow POR delay** modes, you need to modify it to **Instant ON** mode. Ensure your board power design meets the power up requirements listed below.

Table 2. Power Up Requirements for Intel MAX 10 Devices

Power Supply Device Options	Power Rails	Maximum Ramp Rate Requirement (t_{RAMP})
Single Supply Devices	V_{CC_ONE}/V_{CCA}	3 ms
Dual Supply Devices	$V_{CC}, V_{CCINT}, V_{CCD_PLL}, V_{CCA}, V_{CCA_ADC}$	3 ms

For both options, ensure that all IO banks are powered up to nominal operating voltage when configuration completes. Refer to the *Intel MAX 10 FPGA Configuration User Guide* for minimum configuration time specifications. If your board power design cannot meet the stated t_{RAMP} requirement, contact [Intel Premier Support](#).

Long Term Solution

Starting with shipments with date code 1625, Intel MAX 10 devices have a modified power on configuration scheme that enables a simplified power on scheme. This gives you an easier method of configuring Intel MAX 10 devices with the following benefits:

- Single power up scheme
- No power up sequencing requirements
- Maximum power supply rail ramp time increased from 3 ms to 10 ms
- Minimum power supply rail ramp time of 200 µs is now a recommendation rather than an absolute minimum
- No changes to either the die or the ordering part number (OPN)

Intel Quartus® Prime Update

From Intel Quartus® Prime software version 16.0 and onwards, the POR scheme option will be unavailable. This option will be removed in subsequent Intel Quartus Prime software versions when a single, simplified power on scheme is introduced.

Note: Ensure that you select the **Instant ON** POR scheme if you are using a Intel Quartus Prime software version prior to 16.0. If you select **Fast POR delay** or **Slow POR delay**, this may result in failure even if the device has a date code of 1625 or later.

Status

Affects: All Intel MAX 10 devices with date code prior to 1625

Status: Planned configuration setting update in all Intel MAX 10 devices with date code 1625 or later

Related Information

[Intel MAX 10 FPGA Configuration User Guide](#)

1.2. Revision History for Intel MAX 10 Device Errata

Date	Version	Changes
May 2016	2016.05.16	Clarified the workaround in the "Configuration Failure" section.
May 2016	2016.05.03	Initial release



Intel® MAX® 10 FPGA Device Family Pin Connection Guidelines



Online Version



Send Feedback

PCG-01018

ID: **683232**

Version: **2022.05.27**

Contents

Intel® MAX® 10 FPGA Device Family Pin Connection Guidelines.....	3
Intel® MAX® 10 FPGA Pin Connection Guidelines.....	4
Clock and PLL Pins.....	4
Configuration/JTAG Pins.....	5
Differential I/O Pins.....	8
External Memory Interface Pins.....	11
Reference Pins.....	11
Analog Input Pins.....	12
Intel MAX 10 (Single Supply) FPGA.....	14
Intel MAX 10 (Dual Supply) FPGA.....	15
Notes to the Intel MAX 10 FPGA Pin Connection Guidelines.....	18
Power Supply Sharing Guidelines for Intel MAX 10 FPGA Devices.....	20
Example 1—Intel MAX 10 (Dual Supply) FPGA.....	20
Example 2—Intel MAX 10 (Dual Supply) FPGA.....	21
Example 3—Intel MAX 10 (Dual Supply) FPGA.....	24
Example 4—Intel MAX 10 (Single Supply) FPGA.....	26
Example 5—Intel MAX 10 (Single Supply) FPGA.....	27
Example 6—Intel MAX 10 (Single Supply) FPGA.....	29
Document Revision History for Intel MAX 10 FPGA Device Family Pin Connection Guidelines.....	32

Intel® MAX® 10 FPGA Device Family Pin Connection Guidelines

Disclaimer

© 2022 Intel Corporation. All rights reserved. Intel, the Intel logo, Altera, Arria, Cyclone, MAX, NIOS, Quartus Prime and Stratix words and logos are trademarks of Intel Corporation in the US and/or other countries. Other marks and brands may be claimed as the property of others. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services. These pin connection guidelines, and your use thereof, are subject to and governed by Intel's terms and conditions below. By using these pin connection guidelines, you indicate your acceptance of all such terms and conditions. If you do not agree with such terms and conditions, you may not use the pin connection guidelines, and you are required to promptly and irrevocably destroy the pin connection guidelines and any copies or portions thereof in your possession or under your control.

Terms and Conditions:

1. These pin connection guidelines are provided as examples only, and should not be deemed to be technical specifications or recommendations. The use of the pin connection guidelines for any particular design should be verified for device operation with the applicable datasheet and Intel.
2. Subject to these terms and conditions, Intel grants to you the use of these pin connection guidelines as examples of possible pin connections of an Intel programmable logic device-based design. You may not use these pin connection guidelines for any other purpose except as expressly permitted in these terms and conditions. Intel does not recommend, suggest, or require that these pin connection guidelines be used in conjunction or combination with any other software or product, and makes no representations, warranties or guaranties, implied or express as well as any warranties arising from course of performance, course of dealing, or usage in trade including but not limited to the accuracy, completeness or genuineness thereof.
3. Intel will not be liable for any lost revenue, lost profits, or other consequential, indirect, or special damages caused by your use of these pin connection guidelines even if advised of the possibility of such damages occurring.
4. This agreement shall be governed in all respects by the laws of the State of Delaware, without regard to conflict of law or choice of law principles. You agree to submit to the exclusive jurisdiction of the federal and state courts in the State of Delaware for the resolution of any dispute or claim arising out of or relating to these terms of use.

Intel® MAX® 10 FPGA Pin Connection Guidelines

Clock and PLL Pins

Note: Intel® recommends that you create an Intel Quartus® Prime design, enter your device I/O assignments, and compile the design. The Intel Quartus Prime software will check your pin connections according to I/O assignment and placement rules. The rules differ from one device to another based on device density, package, I/O assignments, voltage assignments, and other factors that are not fully described in this document or the device handbook.

Table 1. Clock and PLL Pins

Pin Name	Pin Functions	Pin Description	Connection Guidelines
CLK [0..7]p	Clock, I/O	Dedicated global clock input pins that can also be used for the positive terminal inputs for differential global clock input or user input pins. When these clock input pins are used as single-ended pins, you can disregard the p notation. CLK[0..7]p pins can function as regular I/O pins.	Connect unused pins to the VCCIO of the bank in which the pins reside or GND. See Notes 2 and 3.
CLK[0..7]n	Clock, I/O	Dedicated global clock input pins that can also be used for the negative terminal inputs for differential global clock input or user input pins. When these clock input pins are used as single-ended pins, you can disregard the n notation. CLK[0..7]n pins can function as regular I/O pins.	Connect unused pins to the VCCIO of the bank in which the pins reside or GND. See Notes 2 and 3.
DPCLK[0..3]	I/O, Input	DPCLK pins can connect to the global clock network for high fan-out control signals such as clocks, asynchronous clears, presets, and clock enables. DPCLK pins cannot feed a PLL input.	Connect unused pins to the VCCIO of the bank they reside in or GND. These pins can function as regular I/O pins. See Note 3.
PLL_[L,R,B,T]_CLKOUTp	I/O, Output	Optional positive terminal for external clock outputs from PLL [1..4]. These pins can be assigned to single-ended or differential I/O standards if it is being fed by a PLL output. <ul style="list-style-type: none"> PLL_L_CLKOUTp is referring to PLL_1. PLL_R_CLKOUTp is referring to PLL_2. PLL_T_CLKOUTp is referring to PLL_3. PLL_B_CLKOUTp is referring to PLL_4. 	Connect unused pins to GND. These pins can function as regular I/O pins. See Note 3.

continued...

Pin Name	Pin Functions	Pin Description	Connection Guidelines
		The availability for PLL_[L,R,B,T]_CLKOUTp pins varies for each device density and package combination. For details, refer to the specific device pinout file.	
PLL_[L,R,B,T]_CLKOUTn	I/O, Output	<p>Optional negative terminal for external clock outputs from PLL [1..4]. These pins can be assigned to single-ended or differential I/O standards if it is being fed by a PLL output.</p> <ul style="list-style-type: none"> PLL_L_CLKOUTn is referring to PLL_1. PLL_R_CLKOUTn is referring to PLL_2. PLL_T_CLKOUTn is referring to PLL_3. PLL_B_CLKOUTn is referring to PLL_4. <p>The availability for PLL_[L,R,B,T]_CLKOUTn pins varies for each device density and package combination. For details, refer to the specific device pinout file.</p>	<p>Connect unused pins to GND.</p> <p>These pins can function as regular I/O pins. See Note 3.</p>

Configuration/JTAG Pins

Note: Intel recommends that you create an Intel Quartus Prime design, enter your device I/O assignments, and compile the design. The Intel Quartus Prime software will check your pin connections according to I/O assignment and placement rules. The rules differ from one device to another based on device density, package, I/O assignments, voltage assignments, and other factors that are not fully described in this document or the device handbook.

Table 2. Configuration/JTAG Pins

Pin Name	Pin Functions	Pin Description	Connection Guidelines
CONFIG_SEL	Input, I/O	<p>This is a dual-purpose pin. Use this pin to choose the configuration image in the dual-configuration images mode. If the CONFIG_SEL pin is set to low, the first configuration image is configuration image 0. If the CONFIG_SEL pin is set to high, the first configuration image is configuration image 1.</p> <p>This pin is read before user mode and before the nSTATUS pin is asserted.</p>	<p>To select the configuration image in the dual-configuration images mode, connect a weak 10-KΩ pull-up or weak 10-KΩ pull-down to this pin externally during the power-up phase. By default this pin is tri-stated.</p> <p>A weak 10-KΩ pull-up or weak 10-KΩ pull-down is not needed if you do not plan to use the dual configuration images mode. See Note 10.</p>
CONF_DONE	Bidirectional (open-drain), I/O	<p>This is a dual-purpose pin. The CONF_DONE pin drives low before and during configuration. After all configuration data is received without error and the initialization cycle starts, the CONF_DONE pin is released.</p>	<p>The CONF_DONE pin should be pulled high to VCCIO Bank 8 by an external 10-KΩ pull-up resistor. The Intel MAX® 10 device will not enter the initialization and user mode if the CONF_DONE pin is pulled low.</p>

continued...

Pin Name	Pin Functions	Pin Description	Connection Guidelines
			Hot socketing is disabled for the CONF_DONE pin. Due to this, a glitch maybe observed at the CONF_DONE pin. To monitor the status of the pin, Intel recommends to implement input buffer with hysteresis and digital filtering with the sampling duration larger than 5.5 ms in the external device to avoid false trip.
CRC_ERROR	Output (open-drain), I/O	This is a dual-purpose pin. Active high signal indicates that the error detection circuitry has detected errors in the configuration CRAM bits. The CRC_ERROR pin is an optional pin and is used when the cyclic redundancy check (CRC) error detection circuitry is enabled.	Intel recommends you to tie the CRC_ERROR pin to VCCIO, GND, or leave the pin unconnected when the CRC error detection circuitry is disabled or when you are not using the CRC_ERROR pin.
DEV_CLRn	Input, I/O	This is a dual-purpose pin. Optional chip-wide reset pin that allows you to override all clears on all device registers. When this pin is driven low, all registers are cleared. When this pin is driven high, all registers behave as programmed. The DEV_CLRn pin does not affect JTAG boundary-scan or programming operations. You can enable this pin by turning on the Enable device-wide reset (DEV_CLRn) option in the Intel Quartus Prime software.	Intel recommends you to tie the DEV_CLRn pin to GND when the Enable device-wide reset (DEV_CLRn) option is disabled and not used as a user I/O pin. You can also tie the DEV_CLRn pin to VCCIO or leave the DEV_CLRn pin unconnected provided that the Enable device-wide reset (DEV_CLRn) option is disabled and not used as a user I/O pin. When you leave the DEV_CLRn pin unconnected, Intel recommends you to set the DEV_CLRn pin to input tri-state with a weak pull-up.
DEV_OE	Input, I/O	This is a dual-purpose pin. Optional pin that allows you to override all tri-states on the device. When this pin is driven low, all I/O pins are tri-stated. When this pin is driven high, all I/O pins behave as programmed. You can enable this pin by turning on the Enable device-wide output enable (DEV_OE) option in the Intel Quartus Prime software.	Intel recommends you to tie the DEV_OE pin to GND when the Enable device-wide output enable (DEV_OE) option is disabled and not used as a user I/O pin. You can also tie the DEV_OE pin to VCCIO or leave the DEV_OE pin unconnected provided that the Enable device-wide output enable (DEV_OE) option is disabled and not used as a user I/O pin. When you leave the DEV_OE pin unconnected, Intel recommends you to set the DEV_OE pin to input tri-state with a weak pull-up.
JTAGEN	I/O	This is a dual-purpose pin. This pin functions according to the setting of the JTAG pin sharing option bit. If the JTAG pin sharing is not enabled, the JTAGEN pin is a regular I/O pin and JTAG pins function as JTAG dedicated pins. If the JTAG pin sharing is enabled and the JTAGEN pin is pulled low, JTAG pins function as dual-purpose pins. If the JTAG pin sharing is enabled and the JTAGEN pin is pulled high, JTAG pins function as JTAG dedicated pins.	This pin has an internal 25-kΩ pull up. In user mode, to use JTAG pins as regular I/O pins, tie the JTAGEN pin to a weak 1-kΩ pull-down. To use JTAG pin as dedicated pin, tie the JTAGEN pin to a weak 10-kΩ pull-up.
continued...			

Pin Name	Pin Functions	Pin Description	Connection Guidelines
nCONFIG	Input, I/O	<p>This is a dual-purpose pin, as an nCONFIG pin or a single-ended input pin in user mode. Before user mode, these pins function as configuration pins.</p> <p>During configuration mode, the pin name is nCONFIG. During user mode, the pin name is Input_only.</p> <p>If you pull this pin low during user mode the device will lose its configuration data, enter a reset state, and tri-state all I/O pins. Pulling this pin to a logic-high level initiates reconfiguration.</p>	<p>Upon power up, the nCONFIG pin must be pulled high. Connect this pin directly or through a 10-kΩ resistor to VCCIO.</p>
nSTATUS	Bidirectional (open-drain), I/O	<p>This is a dual-purpose pin, as an nSTATUS pin or a regular user I/O pin in user mode. By default, the nSTATUS pin is a dedicated configuration pin in user mode.</p> <p>The device drives the nSTATUS pin low immediately after power up and releases the pin after power-on reset (POR) time.</p> <p>As a status output, the nSTATUS pin is pulled low if an error occurs during configuration.</p> <p>As a status input, the device enters an error state when the nSTATUS pin is driven low by an external source during configuration or initialization.</p>	<p>Pull the nSTATUS pin high using an external 10-kΩ pull-up resistor. Pull the nSTATUS pin high using an external 10-kΩ pull-up resistor.</p> <p>Hot socketing is disabled for the nSTATUS pin. Due to this, a glitch maybe observed at the nSTATUS pin. To monitor the status of the pin, Intel recommends to implement input buffer with hysteresis and digital filtering with the sampling duration larger than 5.5 ms in the external device to avoid false trip.</p>
TCK	Input, I/O	JTAG test clock input pin. This is a dual-purpose pin.	<p>This TCK pin does not support internal weak pull-down. Connect this pin to an external 1-KΩ – 10-KΩ pull-down resistor.</p> <p>By default this pin is tri-stated.</p> <p>If the configuration voltage exceed 2.5 V (VCCIO Bank 1B), Intel recommends you to add an external capacitor and diode to reduce voltage overshoot.</p> <p>For more information about overshoot prevention circuitry, refer to the <i>Intel MAX 10 Configuration User Guide</i>.</p>
TDO	Output, I/O	This is a dual-purpose pin, as a JTAG TDO pin or a regular user I/O pin in user mode.	<p>Intel recommends you to leave this pin unconnected if not used.</p> <p>By default this pin is tri-stated.</p> <p>If the configuration voltage exceed 2.5 V (VCCIO Bank 1B), Intel recommends you to add an external capacitor and diode to reduce voltage overshoot.</p>
continued...			

Pin Name	Pin Functions	Pin Description	Connection Guidelines
			For more information about overshoot prevention circuitry, refer to the <i>Intel MAX 10 Configuration User Guide</i> .
TDI	Input, I/O	This is a dual-purpose pin, as a JTAG TDI pin or a regular user I/O pin in user mode. You can disable the JTAG circuitry by connecting the TDI pin to VCCIO Bank 1B.	This pin has a weak internal pull-up. For configuration voltage of 2.5 V, 3.0 V, or 3.3 V, connect this pin through a 10-kΩ resistor to 2.5 V (VCCIO Bank 1B) to prevent voltage overshoot. If power supplies exceed 2.5 V, Intel recommends you to add an external capacitor and diode to reduce voltage overshoot. For configuration voltage of 1.5 V and 1.8 V, connect this pin through a 10-kΩ resistor to 1.5 V or 1.8 V (VCCIO Bank 1B) supply, respectively. For more information about overshoot prevention circuitry, refer to the <i>Intel MAX 10 Configuration User Guide</i> .
TMS	Input, I/O	This is a dual-purpose pin, as a JTAG TMS pin or a regular user I/O pin in user mode. You can disable the JTAG circuitry by connecting the TMS pin to VCCIO Bank 1B.	This pin has a weak internal pull-up. For configuration voltage of 2.5 V, 3.0 V, or 3.3 V, connect this pin through a 10-kΩ resistor to 2.5 V (VCCIO Bank 1B) to prevent voltage overshoot. If power supplies exceed 2.5 V, Intel recommends you to add an external capacitor and diode to reduce voltage overshoot. For configuration voltage of 1.5 V and 1.8 V, connect this pin through a 10-kΩ resistor to 1.5 V or 1.8 V (VCCIO Bank 1B) supply, respectively. For more information about overshoot prevention circuitry, refer to the <i>Intel MAX 10 Configuration User Guide</i> .

Differential I/O Pins

Note: Intel recommends that you create an Intel Quartus Prime design, enter your device I/O assignments, and compile the design. The Intel Quartus Prime software will check your pin connections according to I/O assignment and placement rules. The rules differ from one device to another based on device density, package, I/O assignments, voltage assignments, and other factors that are not fully described in this document or the device handbook.

Table 3. Differential I/O Pins

Pin Name	Pin Functions	Pin Description	Connection Guidelines
DIFFIO_RX_L[#:#][n,p], DIFFOUT_L[#:#][n,p]	I/O, dedicated RX channel,	When used as differential inputs, these are true LVDS receiver channels on left I/O banks. Pins with a "p" suffix carry the positive signal for the differential channel. Pins with	Connect unused pins as defined in the Intel Quartus Prime software. For the number of LVDS pair count for each Intel MAX 10 device, refer to the respective device pinout file.
continued...			

Pin Name	Pin Functions	Pin Description	Connection Guidelines
	emulated LVDS output channel	<p>an "n" suffix carry the negative signal for the differential channel. If not used for differential signaling, these pins are available as user I/O pins.</p> <p>When used as differential outputs, these are emulated LVDS output channels on left I/O banks. External resistor network is needed for emulated LVDS output buffers. Pins with a "p" suffix carry the positive signal for the differential channel. Pins with an "n" suffix carry the negative signal for the differential channel. If not used for differential signaling, these pins are available as user I/O pins.</p>	
DIFFIO_RX_R[#:#][n,p], DIFFOUT_R[#:#][n,p]	I/O, dedicated RX channel, emulated LVDS output channel	<p>When used as differential inputs, these are true LVDS receiver channels on right I/O banks. Pins with a "p" suffix carry the positive signal for the differential channel. Pins with an "n" suffix carry the negative signal for the differential channel. If not used for differential signaling, these pins are available as user I/O pins.</p> <p>When used as differential outputs, these are emulated LVDS output channels on right I/O banks. External resistor network is needed for emulated LVDS output buffers. Pins with a "p" suffix carry the positive signal for the differential channel. Pins with an "n" suffix carry the negative signal for the differential channel. If not used for differential signaling, these pins are available as user I/O pins.</p>	<p>Connect unused pins as defined in the Intel Quartus Prime software.</p> <p>For the number of LVDS pair count for each Intel MAX 10 device, refer to the respective device pinout file.</p>
DIFFIO_RX_T[#:#][n,p], DIFFOUT_T[#:#][n,p]	I/O, dedicated RX channel, emulated LVDS output channel	<p>When used as differential inputs, these are true LVDS receiver channels on top I/O banks. Pins with a "p" suffix carry the positive signal for the differential channel. Pins with an "n" suffix carry the negative signal for the differential channel. If not used for differential signaling, these pins are available as user I/O pins.</p> <p>When used as differential outputs, these are emulated LVDS output channels on top I/O banks. External resistor network is needed for emulated LVDS output buffers. Pins with a "p" suffix carry the positive signal for the differential channel. Pins with an "n" suffix carry the negative signal for the differential channel. If not used for differential signaling, these pins are available as user I/O pins.</p>	<p>Connect unused pins as defined in the Intel Quartus Prime software.</p> <p>For the number of LVDS pair count for each Intel MAX 10 device, refer to the respective device pinout file.</p>
DIFFIO_RX_B[#:#][n,p], DIFFOUT_B[#:#][n,p]	I/O, dedicated RX channel, emulated LVDS output channel	<p>When used as differential inputs, these are true LVDS receiver channels on bottom I/O banks. Pins with a "p" suffix carry the positive signal for the differential channel. Pins with an "n" suffix carry the negative signal for the differential channel. If not used for differential signaling, these pins are available as user I/O pins.</p>	<p>Connect unused pins as defined in the Intel Quartus Prime software.</p> <p>For the number of LVDS pair count for each Intel MAX 10 device, refer to the respective device pinout file.</p>
continued...			

Pin Name	Pin Functions	Pin Description	Connection Guidelines
		When used as differential outputs, these are emulated LVDS output channels on bottom I/O banks. External resistor network is needed for emulated LVDS output buffers. Pins with a "p" suffix carry the positive signal for the differential channel. Pins with an "n" suffix carry the negative signal for the differential channel. If not used for differential signaling, these pins are available as user I/O pins.	
DIFFIO_TX_RX_B[#:#] [n,p]	I/O, dedicated TX/RX channel	These are true LVDS transmitter channels or true LVDS receiver channels on bottom I/O banks. Pins with a "p" suffix carry the positive signal for the differential channel. Pins with an "n" suffix carry the negative signal for the differential channel. If not used for differential signaling, these pins are available as user I/O pins.	Connect unused pins as defined in the Intel Quartus Prime software. For the number of LVDS pair count for each Intel MAX 10 device, refer to the respective device pinout file.
High_Speed	I/O	These are I/O pins. High_Speed I/O pins have higher performance compared to Low_Speed I/O pins. High_Speed I/O pins reside in Banks 2, 3, 4, 5, 6, and 7.	Connect unused pins as defined in the Intel Quartus Prime software.
Low_Speed	I/O	These are I/O pins. Low_Speed I/O pins have lower performance compared to High_Speed I/O pins. Low_Speed I/O pins reside in Banks 1A, 1B, and 8.	Connect unused pins as defined in the Intel Quartus Prime software.
RDN	I/O, Input	This pin is required for each OCT RS calibration block. OCT is only applicable for right I/O banks (banks 5 and 6) of 10M16, 10M25, and 10M50 devices. This pin is a dual-purpose pin, you can use the RDN pin as a regular I/O pin if the OCT calibration is not used. When you use OCT calibration, connect the RDN pin to GND through an external resistor.	When you use OCT, tie these pins to GND through either a 25-, 34-, 40-, 48-, or 50-Ω resistor depending on the desired I/O standard. When the device does not use this dedicated input pin for the external precision resistor or as an I/O pin, Intel recommends you to connect the RDN pin to GND.
RUP	I/O, Input	This pin is required for each OCT RS calibration block. OCT is only applicable for right I/O banks (banks 5 and 6) of 10M16, 10M25, and 10M50 devices. This pin is a dual-purpose pin, you can use the RUP pin as a regular I/O pin if the OCT calibration is not used. When you use OCT calibration, connect the RUP pin to VCCN through an external resistor.	When you use OCT, tie these pins to the required VCCIO banks through either a 25-, 34-, 40-, 48-, or 50-Ω resistor depending on the desired I/O standard. When the device does not use this dedicated input pin for the external precision resistor or as an I/O pin, Intel recommends you to connect the RUP pin to VCCIO of the bank in which the RUP pin resides or GND.
VREFB<#>N0	Power, I/O	These pins are dual-purpose pins. For Banks 1A and 1B, VREF pins are shared. Input reference voltage for each I/O bank. If a bank uses a voltage referenced I/O standard for input operation, then these pins are used as the voltage-reference pins for the bank.	If you are not using the VREF pins in banks or shared banks, connect unused pins as defined in the Intel Quartus Prime software. When VREF pins are used as I/O pins, they have higher capacitance than regular I/O pins which will slow the edge rates and affect I/O timing.

External Memory Interface Pins

Note: Intel recommends that you create an Intel Quartus Prime design, enter your device I/O assignments, and compile the design. The Intel Quartus Prime software will check your pin connections according to I/O assignment and placement rules. The rules differ from one device to another based on device density, package, I/O assignments, voltage assignments, and other factors that are not fully described in this document or the device handbook.

Table 4. External Memory Interface Pins

Pin Name	Pin Functions	Pin Description	Connection Guidelines
DQ[#]R	I/O, DQ	Optional data signal for use in external memory interfacing. The order of the DQ bits within a designated DQ bus is not important. However, use with caution when making pin assignments if you plan on migrating to a different memory interface that has a different DQ bus width.	Connect unused pins as defined in the Intel Quartus Prime software.
DQS[#]R	I/O, DQS	Optional data strobe signal for use in external memory interfacing.	Connect unused pins as defined in the Intel Quartus Prime software.
DQSn[#]R	I/O, DQSn	Optional complementary data strobe signal for use in external memory interfacing.	Connect unused pins as defined in the Intel Quartus Prime software.
DM[#]R	I/O, DM	A low signal on the DM pin indicates that the write is valid. Driving the DM pin high results in the memory masking of the DQ signals.	Connect unused pins as defined in the Intel Quartus Prime software.
CK_[#]	I/O, Output	Input clock for external memory devices.	Connect unused pins as defined in the Intel Quartus Prime software.
CK#_[#]	I/O, Output	Input clock for external memory devices, inverted CK.	Connect unused pins as defined in the Intel Quartus Prime software.

Reference Pins

Note: Intel recommends that you create an Intel Quartus Prime design, enter your device I/O assignments, and compile the design. The Intel Quartus Prime software will check your pin connections according to I/O assignment and placement rules. The rules differ from one device to another based on device density, package, I/O assignments, voltage assignments, and other factors that are not fully described in this document or the device handbook.

Table 5. Reference Pins

Pin Name	Pin Functions	Pin Description	Connection Guidelines
GND	Ground	Device ground pins.	Intel recommends you to tie REFGND to the GND pin with an isolating ferrite bead for the best ADC performance. Connect all GND pins to the board GND plane.
NC	No Connect	Do not drive signals into these pins.	When designing for device migration you may connect these pins to power, ground, or a signal trace depending on the pin assignment of the devices selected for migration. However, if device migration is not a concern, leave these pins floating.
DNU	Do Not Use	Do Not Use (DNU).	Do not connect to power, GND, or any other signal. These pins must be left floating.

Analog Input Pins

Note: Intel recommends that you create an Intel Quartus Prime design, enter your device I/O assignments, and compile the design. The Intel Quartus Prime software will check your pin connections according to I/O assignment and placement rules. The rules differ from one device to another based on device density, package, I/O assignments, voltage assignments, and other factors that are not fully described in this document or the device handbook.

Table 6. Analog Input Pins

Pin Name	Pin Functions	Pin Description	Connection Guidelines
ADC[1..2]IN[1..16]	I/O, Input	<p>These multi-purpose pins support single-ended analog input or, bank does not support both analog and digital signals simultaneously. When not used as analog input pins, these pins can be used as any other digital I/O pins.</p> <p>ADCIN[8] and ADCIN[16] pins support the prescaler feature.</p> <p>For 10M08 and 10M16 devices, ADC1IN[1..8] pins are available for the single power supply devices and ADC1IN[1..16] pins are available for 10M08U324 devices. ADC1IN[1..16] pins are available for the dual power supply devices.</p> <p>For 10M25 and 10M50 devices, ADC1IN[1..8] and ADC2IN[1..8] pins are available for both single and dual power supply devices.</p>	<p>All digital I/O pins will be tri-stated if any of these pins is configured as an analog input pin. For unused ADCIN pins, Intel recommends you to connect them to GND.</p> <p>No parallel routing between analog input signals and I/O traces. The crosstalk requirement for analog to digital signal is -100 dB up to 2 GHz. Route the analog input signal adjacent to the REFGND.</p> <p>Total RC value including package, trace, and driver parasitic values should be less than 42.4 ns. This is to ensure the input signal is fully settled during the sampling phase.</p> <p>Low pass filter is required for each analog input pin. The filter ground reference is REFGND.</p>

continued...

Pin Name	Pin Functions	Pin Description	Connection Guidelines
			For details about the board design guidelines, refer to the <i>Intel MAX 10 Analog to Digital Converter User Guide</i> .
ADC_VREF	Input	Analog-to-digital converter (ADC) voltage reference input.	<p>Tie the ADC_VREF pin to an external accurate voltage reference source. If you are not using the external reference, this pin is a no connect (NC).</p> <p>No parallel routing between analog input signals and I/O traces. The crosstalk requirement for analog to digital signal is -100 dB up to 2 GHz.</p> <p>For more information, refer to the Guidelines: Board Design for ADC Reference Voltage Pin section of the <i>Intel MAX 10 Analog to Digital Converter User Guide</i>.</p>
ANAIN[1]	Input	This is a dedicated single-ended analog input pin for ADC1.	<p>If this pin is not used, Intel recommends you to connect it to GND.</p> <p>No parallel routing between analog input signals and I/O traces. The crosstalk requirement for analog to digital signal is -100 dB up to 2 GHz. Route the analog input signal adjacent to the REFGND.</p> <p>Total RC value including package, trace, and driver parasitic values should be less than 42.4 ns. This is to ensure the input signal is fully settled during the sampling phase.</p> <p>Low pass filter is required for each analog input pin. The filter ground reference is REFGND.</p> <p>For details about the board design guidelines, refer to the <i>Intel MAX 10 Analog to Digital Converter User Guide</i>.</p>
ANAIN[2]	Input	This is a dedicated single-ended analog input pin for ADC2. This pin is not available in each device density and package combination. For details, refer to the specific device pinout file.	<p>If this pin is not used, Intel recommends you to connect it to GND. No parallel routing between analog input signals and I/O traces. The crosstalk requirement for analog to digital signal is -100 dB up to 2 GHz. The RC filter ground reference is REFGND.</p> <p>Total RC value including package, trace, and driver parasitic values should be less than 42.4 ns. This is to ensure the input signal is fully settled during the sampling phase.</p> <p>Low pass filter is required for each analog input pin. The filter ground reference is REFGND.</p> <p>For more information, refer to the <i>Intel MAX 10 Analog to Digital Converter User Guide</i>.</p>
REFGND	Input	This pin is the ADC ground reference pin for analog pins.	<p>Intel recommends you to tie REFGND to the GND pin with an isolating ferrite bead for the best ADC performance.</p> <p>If you are not using ADC, tie this pin directly to GND.</p>

Intel MAX 10 (Single Supply) FPGA

Note: Intel recommends that you create an Intel Quartus Prime design, enter your device I/O assignments, and compile the design. The Intel Quartus Prime software will check your pin connections according to I/O assignment and placement rules. The rules differ from one device to another based on device density, package, I/O assignments, voltage assignments, and other factors that are not fully described in this document or the device handbook.

Table 7. Intel MAX 10 (Single Supply) FPGA

Pin Name	Pin Functions	Pin Description	Connection Guidelines
VCC_ONE	Power	Power supply pin for core and periphery through an on-die regulator. The voltage is internally regulated to 1.2V to supply power to the core and periphery.	The VCC_ONE power supply pin supports E144, M153, U169, and U324 package-types only. Connect all VCC_ONE pins to either 3.0- or 3.3-V power supply. Tie VCC_ONE and VCCA with filter using the same power supply on board level.
VCCIO[#]	Power	I/O supply voltage pins for banks 1 through 8. Each bank supports different voltage level. The VCCIO pin supplies power to the input and output buffers for banks 1 through 8 for all I/O standards. The VCCIO pin powers up the JTAG and configuration pins.	Connect these pins to 1.0-, 1.2-, 1.35-, 1.5-, 1.8-, 2.5-, 3.0-, or 3.3-V power supplies, depending on the I/O standard assigned to each I/O bank. <i>Note:</i> The 1.0-V VCCIO is not supported on I/O banks 1B and 8. For single-supply device option, the 1.0-V VCCIO is only supported on specific Intel MAX 10 devices. For the list of supported devices, refer to the <i>Supported I/O Standards</i> section in the <i>Intel MAX 10 General Purpose I/O User Guide</i> . If you power-up a device from the power-down state, you need to power the VCCIO for bank 1B (bank 1 for 10M02 devices), bank 8, and the core to the appropriate level for the device to exit POR. The Intel MAX 10 device enters the configuration stage after exiting the power-up stage with a small POR delay. The VCCIO for bank 1B (bank 1 for 10M02 devices) and bank 8 must be powered up to a voltage between 1.5V – 3.3V during configuration. If you are migrating from other Intel MAX 10 devices to the 10M02 device, the VCCIO1A and VCCIO1B pins are shorted to the VCCIO1 pin of the 10M02 device. If you do not enable the ADC feature, you may connect VCCIO1A and VCCIO1B pins to different voltage levels, provided that the VREF pin is not used. If the VREF pin is used, you must connect the VCCIO1A and VCCIO1B pins to the same voltage level.

continued...

Pin Name	Pin Functions	Pin Description	Connection Guidelines
			<p>If you enable the ADC feature, connect VCCIO1A and VCCIO1B to either 3.0- or 3.3-V depending on the VCC_ONE pins used.</p> <p>The power supply sharing between VCCIO1A and VCCIO1B pins requires filtering to isolate the noise. The filter should be located near to VCCIO1A pins. Only 10M02 devices do not require filtering if VCCIO1A and VCCIO1B share the same power supply. When the ADC feature is enabled, filter is required.</p> <p>If you are migrating from the 10M08 or 10M16 device to the 10M02 device with ADC enabled, replace the filter with 0-Ω resistor in the 10M02 device.</p> <p>For details about the available VCCIO pins for each Intel MAX 10 device, refer to the respective device pinout file. See Note 4.</p> <p>Decoupling of these pins depends on the design decoupling requirements of the specific board.</p>
VCCA[1..6]	Power	Power supply pins for PLL and ADC block.	<p>Connect these pins to a 3.0- or 3.3-V power supplies even if the PLL and ADC are not used. These pins must be powered up and powered down at the same time. Connect all VCCA pins together.</p> <p>VCCA power supply to the FPGA should be isolated for better jitter performance. See Notes 5 and 6.</p> <p>VCCA[1..4] is available for M153, U169, and U324 packages while VCCA[1..6] is available for the E144 package.</p> <p>For more information about the UFM and CFM power-down requirement, refer to the <i>Intel MAX 10 User Flash Memory User Guide</i>.</p>

Related Information

- [Intel MAX 10 User Flash Memory User Guide](#)
- [Supported I/O Standards in Intel MAX 10 Devices](#)

Intel MAX 10 (Dual Supply) FPGA

Note: Intel recommends that you create an Intel Quartus Prime design, enter your device I/O assignments, and compile the design. The Intel Quartus Prime software will check your pin connections according to I/O assignment and placement rules. The rules differ from one device to another based on device density, package, I/O assignments, voltage assignments, and other factors that are not fully described in this document or the device handbook.

Table 8. Intel MAX 10 (Dual Supply) FPGA

Pin Name	Pin Functions	Pin Description	Connection Guidelines
VCC	Power	Power supply pin for core and periphery.	Connect all VCC pins to 1.2-V power supply. Decoupling of these pins depends on the design decoupling requirements of the specific board. See Note 4.
VCCIO[#]	Power	I/O supply voltage pins for banks 1 through 8. Each bank supports different voltage level. The VCCIO pin supplies power to the input and output buffers for all I/O standards. The VCCIO pin powers up the JTAG and configuration pins.	Connect these pins to 1.0-, 1.2-, 1.35-, 1.5-, 1.8-, 2.5-, 3.0-, or 3.3-V power supplies, depending on the I/O standard assigned to each I/O bank. <i>Note:</i> The 1.0-V VCCIO is not supported on I/O banks 1B and 8. For dual-supply device option, the 1.0-V VCCIO is only supported on specific Intel MAX 10 devices. For the list of supported devices, refer to the <i>Supported I/O Standards</i> section in the <i>Intel MAX 10 General Purpose I/O User Guide</i> . If you power-up a device from the power-down state, you need to power the VCCIO for bank 1B (bank 1 for 10M02 devices), bank 8, and the core to the appropriate level for the device to exit POR. The Intel MAX 10 device enters the configuration stage after exiting the power-up stage with a small POR delay. The VCCIO for bank 1B (bank 1 for 10M02 devices) and bank 8 must be powered up to a voltage between 1.5V – 3.3V during configuration. If you are migrating from other Intel MAX 10 devices to the 10M02 device, the VCCIO1A and VCCIO1B pins are shorted to the VCCIO1 pin of the 10M02 device. If you do not enable the ADC feature, you may connect VCCIO1A and VCCIO1B pins to different voltage levels, provided that the VREF pin is not used. If the VREF pin is used, you must connect the VCCIO1A and VCCIO1B pins to the same voltage level. If you enable the ADC feature, connect VCCIO1A and VCCIO1B to 2.5 V. The power supply sharing between VCCIO1A and VCCIO1B pins requires filtering to isolate the noise. The filter should be located near to VCCIO1A pins. Only 10M02 devices do not require filtering if VCCIO1A and VCCIO1B share the same power supply. When the ADC feature is enabled, filter is required. If you are migrating from the 10M08 or 10M16 device to the 10M02 device with ADC enabled, replace the filter with 0-Ω resistor in the 10M02 device.

continued...

Pin Name	Pin Functions	Pin Description	Connection Guidelines
			For details about the available VCCIO pins for each Intel MAX 10 device, refer to the respective device pinout file. Decoupling of these pins depends on the design decoupling requirements of the specific board. See Note 4.
VCCA[1..4]	Power	Power supply pins for PLL analog block.	Connect these pins to a 2.5 V power supply even if the PLL is not used. These pins must be powered up and powered down at the same time. Connect all VCCA pins together. VCCA power supply to the FPGA should be isolated for better jitter performance. See Notes 5 and 6. For more information about the UFM and CFM power-down requirement, refer to the <i>Intel MAX 10 User Flash Memory User Guide</i> .
VCCD_PLL[1..4]	Power	Power supply pins for PLL digital block.	Connect VCCD_PLL[1..4] pins to 1.2 V power supply even if the PLL is not used. Connect all VCCD_PLL[1..4] pins together. Intel recommends you to keep these pins isolated from other VCC pins for better jitter performance. See Notes 5 and 7.
VCCA_ADC	Power	Power supply pin for ADC analog block.	Connect the VCCA_ADC pin to the recommended power supply specification for the best ADC performance. Tie the VCCA_ADC pin to any 2.5 V power domain if you are not using ADC, and do not tie the VCCA_ADC pin to GND. Decoupling of these pins depends on the design decoupling requirements of the specific board. See Note 4.
VCCINT	Power	Power supply pin for ADC digital block.	Connect the VCCINT pin to the recommended power supply specification for the best ADC performance. Tie the VCCINT pin to any 1.2 V power domain if you are not using ADC. Decoupling of these pins depends on the design decoupling requirements of the specific board. See Note 4.

Related Information

- [Intel MAX 10 User Flash Memory User Guide](#)
- [Supported I/O Standards in Intel MAX 10 Devices](#)

Notes to the Intel MAX 10 FPGA Pin Connection Guidelines

Note: Intel recommends that you create an Intel Quartus Prime design, enter your device I/O assignments, and compile the design. The Intel Quartus Prime software will check your pin connections according to I/O assignment and placement rules. The rules differ from one device to another based on device density, package, I/O assignments, voltage assignments, and other factors that are not fully described in this document or the device handbook.

Intel provides these guidelines only as recommendations. It is the responsibility of the designer to apply simulation results to the design to verify proper device functionality.

1. These pin connection guidelines are created based on the Intel MAX 10 FPGA device family.
2. The number of dedicated global clocks for each device density is different.
3. The unused pins must be connected as specified in the Intel Quartus Prime software settings. The default Intel Quartus Prime setting for unused pins is 'As inputs tri-stated with weak pull-up resistors', unless for specific pins that the Intel Quartus Prime software connects them to GND automatically.
4. Capacitance values for the power supply decoupling capacitors should be selected after consideration of the amount of power needed to supply over the frequency of operation of the particular circuit being decoupled. A target impedance for the power plane should be calculated based on current draw and voltage drop requirements of the device/supply. The power plane should then be decoupled using the appropriate number of capacitors. On-board capacitors do not decouple higher than 100 MHz due to "Equivalent Series Inductance" of the mounting of the packages. Proper board design techniques such as interplane capacitance with low inductance should be considered for higher frequency decoupling. To assist in decoupling analysis, Intel's "Power Distribution Network (PDN) Design Tool" serves as an excellent decoupling analysis tool. The PDN design tool can be obtained at Power Distribution Network Design Tool.

Table 9. Transient Current and Voltage Ripple for Intel MAX 10 Devices

To calculate the target impedance of each Intel MAX 10 device supply, you should use the following transient current and voltage ripple percentages. Setting Ftarget to 70 MHz or higher should result in a robust PDN.

Intel MAX 10 Supply Rail	Transient Current (%)	Voltage Ripple (%)
VCC	50	5
VCCIO	100	5
VCCA	10	5
VCCD_PLL	10	3
VCCA_ADC	50	2
VCCINT	50	3

5. Use separate power islands for VCCA and VCCD_PLL. PLL power supply may originate from another plane on the board but must be isolated using a ferrite bead or other equivalent methods. If using a ferrite bead, choose an 0402 package with low DC resistance, higher current rating than the maximum steady state current for the supply it is connected to (VCCA or VCCD_PLL) and high impedance at 100 MHz.
6. The VCCA power island can be decoupled with a combination of decoupling capacitors. Please refer to the Power Distribution Network Design Tool to determine the decoupling capacitors value. Use 0402 package for 0.1 uF and smaller capacitors for lower mounting inductance. Place 0.1 uF and smaller capacitors as close to the device as possible. On-board capacitors do not decouple higher than 100 MHz due to "Equivalent Series Inductance" of the mounting of the packages. Proper board design techniques such as interplane capacitance with low inductance should be considered for higher frequency decoupling. To minimize impact on jitter, a 20 mV ripple voltage was used in the analysis for VCCA decoupling.
7. The VCCD_PLL power island can be decoupled with a combination of decoupling capacitors. Please refer to the "Power Distribution Network Design Tool" at Power Distribution Network Design Tool to determine the decoupling capacitors value. Place 0.1 uF and smaller capacitors as close to the device as possible. On-board capacitors do not decouple higher than 100 MHz due to "Equivalent Series Inductance" of the mounting of the packages. Proper board design techniques such as interplane capacitance with low inductance should be considered for higher frequency decoupling. To minimize impact on jitter, a 20 mV ripple voltage was used in the analysis for VCCD_PLL decoupling.
8. All configuration pins used in user mode are low-speed I/Os.
9. Low Noise Switching Regulator - defined as a switching regulator circuit encapsulated in a thin surface mount package containing the switch controller, power FETs, inductor, and other support components. The switching frequency is usually between 800kHz and 1MHz and has fast transient response. The switching frequency range is not an Intel requirement. However, Intel does require the Line Regulation and Load Regulation meet the following specifications:
 - Line Regulation < 0.4%
 - Load Regulation < 1.2%
10. If you enable the **Configure device from CFM0 only** option in the Intel Quartus Prime software when generating the POF file, the FPGA will always load the configuration image 0 without sampling the physical CONFIG_SEL pin during power up.

Power Supply Sharing Guidelines for Intel MAX 10 FPGA Devices

Example 1—Intel MAX 10 (Dual Supply) FPGA

Note: Intel recommends that you create an Intel Quartus Prime design, enter your device I/O assignments, and compile the design. The Intel Quartus Prime software will check your pin connections according to I/O assignment and placement rules. The rules differ from one device to another based on device density, package, I/O assignments, voltage assignments, and other factors that are not fully described in this document or the device handbook.

Table 10. Power Supply Sharing Guidelines for Intel MAX 10 (Dual Supply) FPGA – The ADC Feature is Not Used

Example Requiring 3 Power Regulators

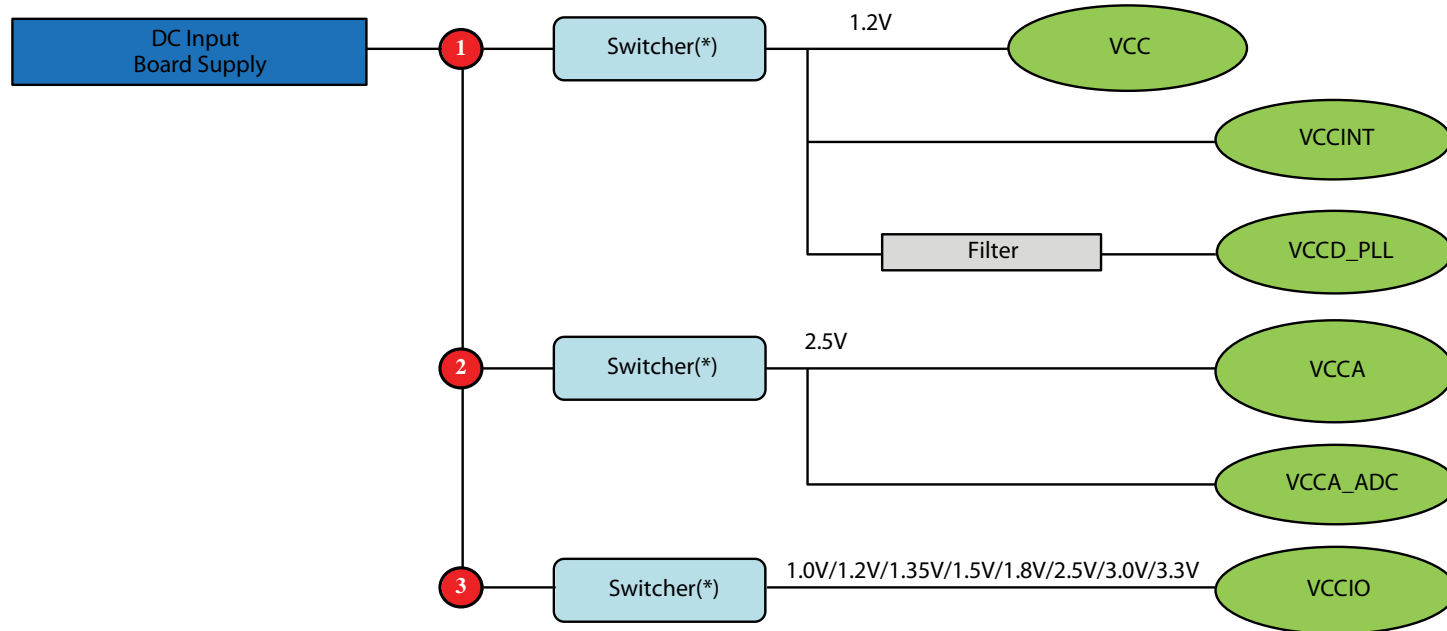
Power Pin Name	Regulator Group	Voltage Level (V)	Supply Tolerance	Power Source	Regulator Sharing	Notes
VCC	1	1.2	± 50mV	Switcher (*)	Share	You have the option to share VCCINT and VCCD_PLL with VCC using proper isolation filters.
VCCINT					Isolate	
VCCD_PLL					Isolate	
VCCA	2	2.5	± 5%	Switcher (*)	Share	You have the option to share VCCA_ADC with VCCA.
VCCA_ADC						
VCCIO	3	Varies	± 5%	Switcher (*)	Share	Individual power rail.

(*)When using a switcher to supply these voltages, the switcher must be a low noise switcher as defined in note 9 of the *Notes to the Intel MAX 10 FPGA Pin Connection Guidelines*.

Notes:

1. Use the EPE (Early Power Estimation) tool to assist in determining the power required for your specific design.
2. Each board design requires its own power analysis to determine the required power regulators needed to satisfy the specific board design requirements. An example block diagram using the Intel MAX 10 FPGA device is provided in Figure 1.
3. For LPDDR2 interface targeting 200MHz, you need to constraint the memory device I/O and core power supply to ± 3% variation.
4. Refer to the *Intel MAX 10 FPGA Configuration User Guide* for maximum ramp rate requirement.

Figure 1. Example Power Supply Sharing Guidelines for Intel MAX 10 (Dual Supply) FPGA – The ADC Feature is Not Used



The ADC power supply requires 0.1uF decoupling cap near the package and ferrite bead filter at power supply.

(*) When using a switcher to supply these voltages, the switcher must be a low noise switcher as defined in *note 9* of the Notes to the Intel MAX 10 FPGA Pin Connection Guidelines.

Example 2—Intel MAX 10 (Dual Supply) FPGA

Note: Intel recommends that you create an Intel Quartus Prime design, enter your device I/O assignments, and compile the design. The Intel Quartus Prime software will check your pin connections according to I/O assignment and placement rules. The rules differ from one device to another based on device density, package, I/O assignments, voltage assignments, and other factors that are not fully described in this document or the device handbook.

Table 11. Power Supply Sharing Guidelines for Intel MAX 10 (Dual Supply) FPGA – Using the ADC Feature and VCCIO[2..8] Pins are Powered Up at 2.5V

Example Requiring 3 Power Regulators

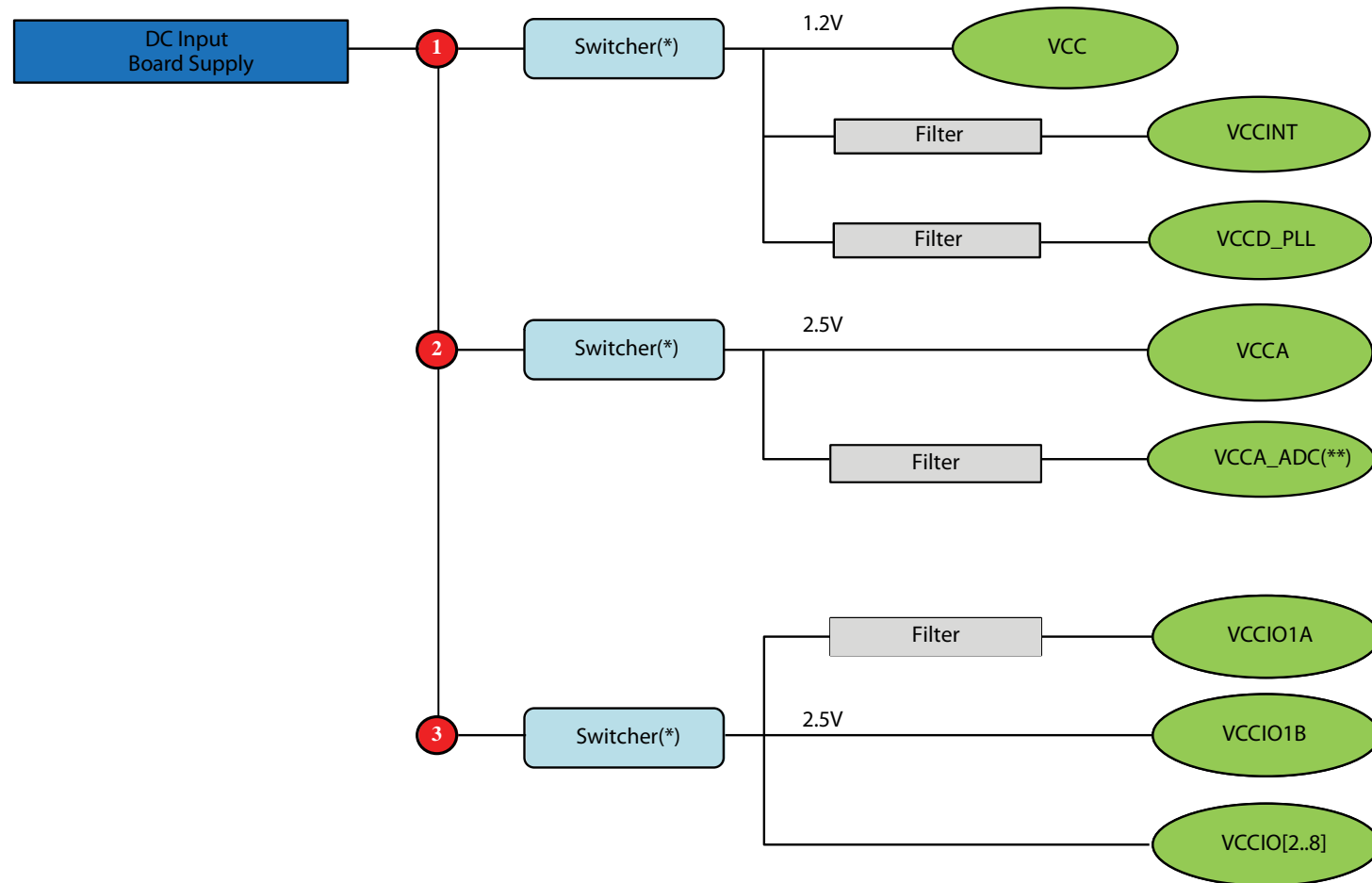
Power Pin Name	Regulator Count	Voltage Level (V)	Supply Tolerance	Power Source	Regulator Sharing	Notes
VCC	1	1.2	± 50mV	Switcher (*)	Share	You have the option to share VCCINT and VCCD_PLL with VCC with proper isolation filters.
VCCINT					Isolate	
VCCD_PLL					Isolate	
VCCA	2	2.5	± 5%	Switcher (*)	Share	You have the option to share VCCA_ADC with VCCA with proper isolation filters.
VCCA_ADC					Isolate	
VCCIO1B	3	2.5	± 5%	Switcher (*)	Share	You have the option to share VCCIO1B and VCCIO[2..8] when these pins are powered up at 2.5V.
VCCIO[2..8]					Share	
VCCIO1A					Isolate	You have the option to share VCCIO1A with VCCIO1B and VCCIO[2..8] using proper isolation filter.

(*)When using a switcher to supply these voltages, the switcher must be a low noise switcher as defined in note 9 of the *Notes to the Intel MAX 10 FPGA Pin Connection Guidelines*.

Notes:

1. Use the EPE (Early Power Estimation) tool to assist in determining the power required for your specific design.
2. Each board design requires its own power analysis to determine the required power regulators needed to satisfy the specific board design requirements. An example block diagram using the Intel MAX 10 FPGA device is provided in Figure 2.
3. For LPDDR2 interface targeting 200MHz, you need to constraint the memory device I/O and core power supply to ± 3% variation.
4. Refer to the *Intel MAX 10 FPGA Configuration User Guide* for maximum ramp rate requirement.

Figure 2. Example Power Supply Sharing Guidelines for Intel MAX 10 (Dual Supply) FPGA – Using the ADC Feature and VCCIO[2..8] Pins are Powered Up at 2.5V



The ADC power supply requires 0.1uF decoupling cap near the package and ferrite bead filter at power supply.

(*) When using a switcher to supply these voltages, the switcher must be a low noise switcher as defined in *note 9* of the Notes to the Intel MAX 10 FPGA Pin Connection Guidelines.

(**) Ferrite beads should be connected in series followed by a 10uF capacitor to ground. Place the decoupling of 0.1uF cap closer to the pin.

Example 3—Intel MAX 10 (Dual Supply) FPGA

Note: Intel recommends that you create an Intel Quartus Prime design, enter your device I/O assignments, and compile the design. The Intel Quartus Prime software will check your pin connections according to I/O assignment and placement rules. The rules differ from one device to another based on device density, package, I/O assignments, voltage assignments, and other factors that are not fully described in this document or the device handbook.

Table 12. Power Supply Sharing Guidelines for Intel MAX 10 (Dual Supply) FPGA – Using the ADC Feature and VCCIO[2..8] Pins are Powered Up at 1.0V/1.2V/1.35V/1.5V/1.8V/2.5V/3.0V/3.3V

Example Requiring 4 Power Regulators

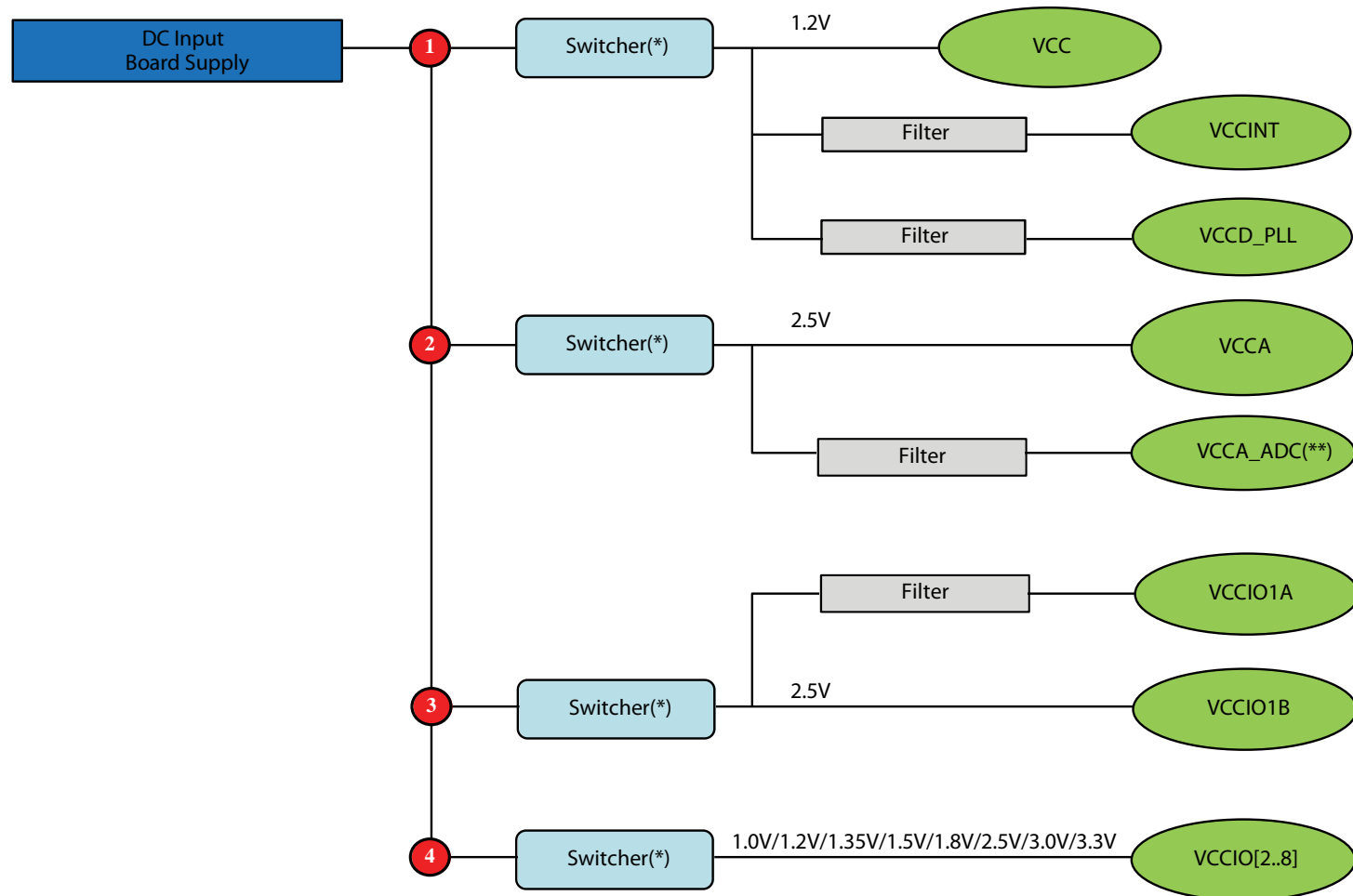
Power Pin Name	Regulator Count	Voltage Level (V)	Supply Tolerance	Power Source	Regulator Sharing	Notes
VCC	1	1.2	± 50mV	Switcher (*)	Share	You have the option to share VCCINT and VCCD_PLL with VCC using proper isolation filters.
VCCINT					Isolate	
VCCD_PLL					Isolate	
VCCA	2	2.5	± 5%	Switcher (*)	Share	You have the option to share VCCA_ADC with VCCA using proper isolation filters.
VCCA_ADC					Isolate	
VCCIO1B	3	2.5	± 5%	Switcher (*)	Share	You have the option to share VCCIO1A with VCCIO1B using proper isolation filter.
VCCIO1A					Isolate	
VCCIO[2..8]	4	Varies	± 5%	Switcher (*)	Share	Individual power rail.

(*)When using a switcher to supply these voltages, the switcher must be a low noise switcher as defined in note 9 of the Notes to the *Intel MAX 10 FPGA Pin Connection Guidelines*.

Notes:

1. Use the EPE (Early Power Estimation) tool to assist in determining the power required for your specific design.
2. Each board design requires its own power analysis to determine the required power regulators needed to satisfy the specific board design requirements. An example block diagram using the Intel MAX 10 FPGA device is provided in Figure 3.
3. For LPDDR2 interface targeting 200MHz, you need to constraint the memory device I/O and core power supply to ± 3% variation.
4. Refer to the *Intel MAX 10 FPGA Configuration User Guide* for maximum ramp rate requirement.

Figure 3. Example Power Supply Sharing Guidelines for Intel MAX 10 (Dual Supply) FPGA – Using the ADC Feature and VCCIO[2..8] Pins are Powered Up at 1.0V/1.2V/1.35V/1.5V/1.8V/2.5V/3.0V/3.3V



The ADC power supply requires 0.1uF decoupling cap near the package and ferrite bead filter at power supply.

(*) When using a switcher to supply these voltages, the switcher must be a low noise switcher as defined in *note 9* of the Notes to the Intel MAX 10 FPGA Pin Connection Guidelines.

(**) Ferrite beads should be connected in series followed by a 10uF capacitor to ground. Place the decoupling of 0.1uF cap closer to the pin.

Example 4—Intel MAX 10 (Single Supply) FPGA

Note: Intel recommends that you create an Intel Quartus Prime design, enter your device I/O assignments, and compile the design. The Intel Quartus Prime software will check your pin connections according to I/O assignment and placement rules. The rules differ from one device to another based on device density, package, I/O assignments, voltage assignments, and other factors that are not fully described in this document or the device handbook.

Table 13. Power Supply Sharing Guidelines for Intel MAX 10 (Single Supply) FPGA (E144, M153, U169, and U324 Packages) – The ADC Feature is Not Used

Example Requiring 2 Power Regulator

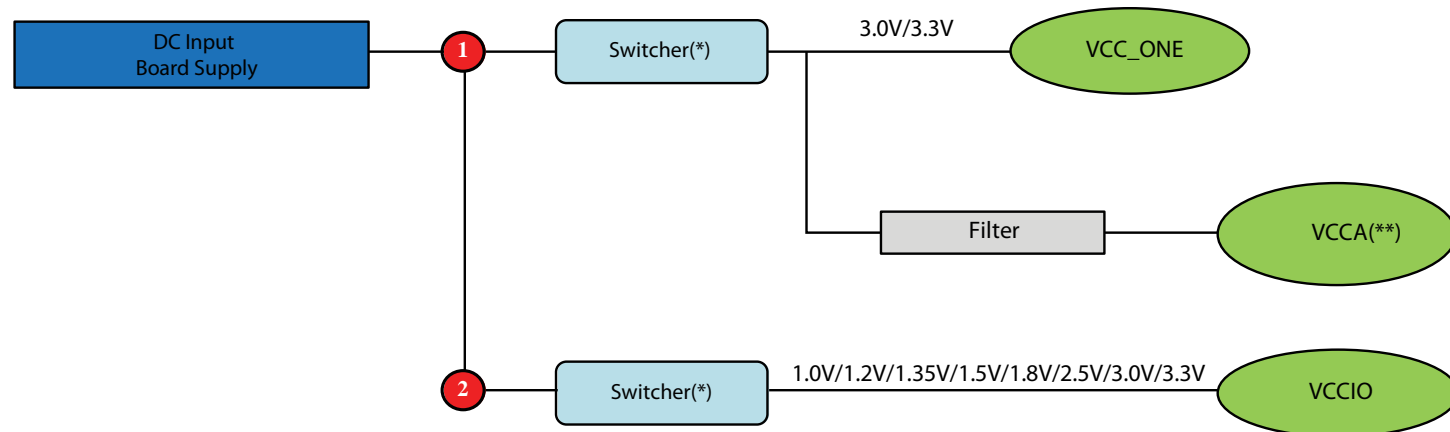
Power Pin Name	Regulator Count	Voltage Level (V)	Supply Tolerance	Power Source	Regulator Sharing	Notes
VCC_ONE	1	3.0/3.3	± 5%	Switcher (*)	Share	Both VCCA and VCC_ONE must share a single power source using proper isolation filter.
VCCA					Isolate	
VCCIO	2	Varies	± 5%	Switcher (*)	Share	Individual power rail.

(*)When using a switcher to supply these voltages, the switcher must be a low noise switcher as defined in note 9 of the *Notes to the Intel MAX 10 FPGA Pin Connection Guidelines*.

Notes:

1. Use the EPE (Early Power Estimation) tool to assist in determining the power required for your specific design.
2. Each board design requires its own power analysis to determine the required power regulators needed to satisfy the specific board design requirements. An example block diagram using the Intel MAX 10 FPGA device is provided in Figure 4.
3. Refer to the *Intel MAX 10 FPGA Configuration User Guide* for maximum ramp rate requirement.

Figure 4. Example Power Supply Sharing Guidelines for Intel MAX 10 (Single Supply) FPGA (E144, M153, U169, and U324 Packages) – The ADC Feature is Not Used



(*) When using a switcher to supply these voltages, the switcher must be a low noise switcher as defined in note 9 of the Notes to the Intel MAX 10 FPGA Pin Connection Guidelines.

(**) The VCCA power supply requires ferrite bead filter for noise isolation.

Example 5—Intel MAX 10 (Single Supply) FPGA

Note: Intel recommends that you create an Intel Quartus Prime design, enter your device I/O assignments, and compile the design. The Intel Quartus Prime software will check your pin connections according to I/O assignment and placement rules. The rules differ from one device to another based on device density, package, I/O assignments, voltage assignments, and other factors that are not fully described in this document or the device handbook.

Table 14. Power Supply Sharing Guidelines for Intel MAX 10 (Single Supply) FPGA – Using the ADC Feature and VCCIO[2..8] Pins are Powered Up 3.0V/3.3V (E144, M153, U169, and U324 Packages)

Example Requiring 2 Power Regulator

Power Pin Name	Regulator Count	Voltage Level (V)	Supply Tolerance	Power Source	Regulator Sharing	Notes
VCC_ONE	1	3.0/3.3	± 5%	Switcher (*)	Share	Both VCCA and VCC_ONE must share a single power source using proper isolation filter.
VCCA					Isolate	

continued...

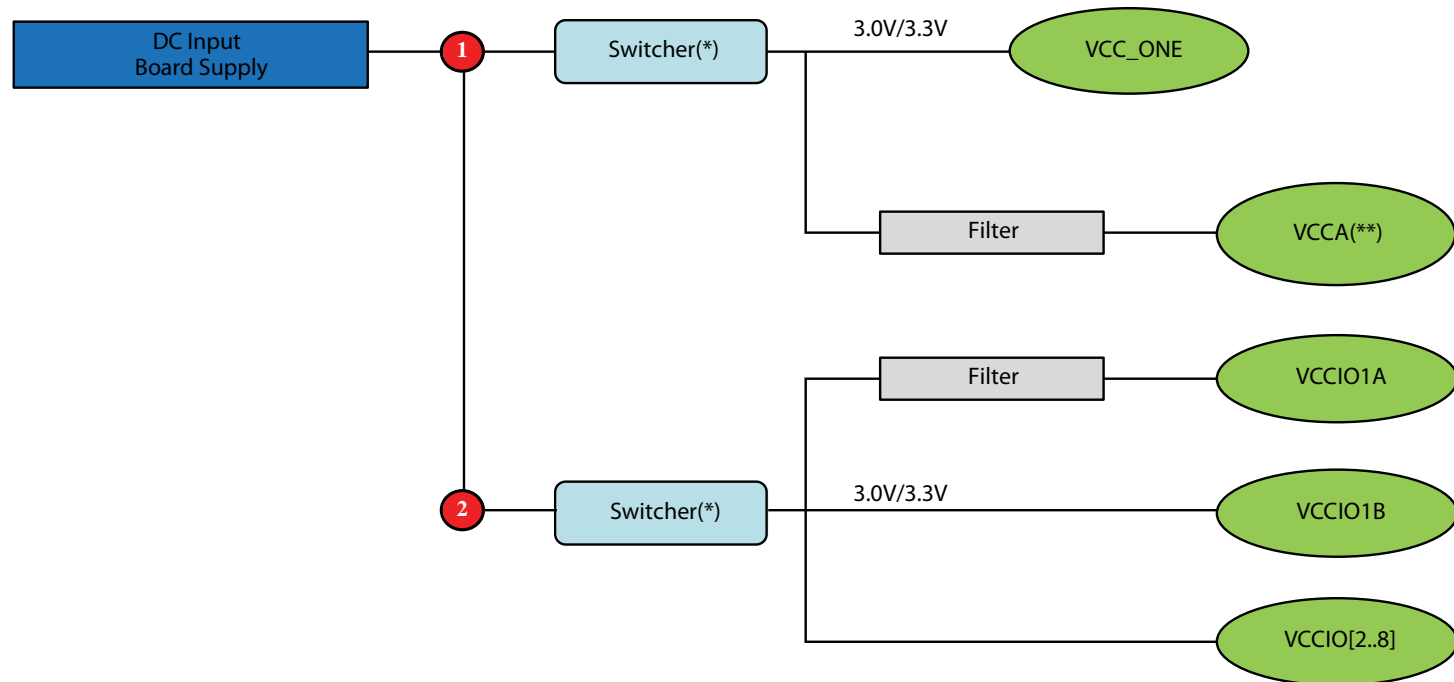
Power Pin Name	Regulator Count	Voltage Level (V)	Supply Tolerance	Power Source	Regulator Sharing	Notes
VCCIO1B	2	3.0/3.3	± 5%	Switcher (*)	Share	You have the option to share VCCIO1B and VCCIO[2..8] when these pins are powered up at 3.0V/3.3V.
VCCIO[2..8]						
VCCIO1A					Isolate	You have the option to share VCCIO1A with VCCIO1B and VCCIO[2..8] using proper isolation filter.

(*)When using a switcher to supply these voltages, the switcher must be a low noise switcher as defined in note 9 of the *Notes to the Intel MAX 10 FPGA Pin Connection Guidelines*.

Notes:

1. Use the EPE (Early Power Estimation) tool to assist in determining the power required for your specific design.
2. Each board design requires its own power analysis to determine the required power regulators needed to satisfy the specific board design requirements. An example block diagram using the Intel MAX 10 FPGA device is provided in Figure 5.
3. Refer to the *Intel MAX 10 FPGA Configuration User Guide* for maximum ramp rate requirement.

Figure 5. Example Power Supply Sharing Guidelines for Intel MAX 10 (Single Supply) FPGA – Using the ADC Feature and VCCIO[2..8] Pins are Powered Up 3.0V/3.3V (E144, M153, U169, and U324 Packages)



(*) When using a switcher to supply these voltages, the switcher must be a low noise switcher as defined in *note 9* of the Notes to the Intel MAX 10 FPGA Pin Connection Guidelines.

(**) The VCCA power supply requires ferrite bead filter for noise isolation.

Example 6—Intel MAX 10 (Single Supply) FPGA

Note: Intel recommends that you create an Intel Quartus Prime design, enter your device I/O assignments, and compile the design. The Intel Quartus Prime software will check your pin connections according to I/O assignment and placement rules. The rules differ from one device to another based on device density, package, I/O assignments, voltage assignments, and other factors that are not fully described in this document or the device handbook.

Table 15. Power Supply Sharing Guidelines for Intel MAX 10 (Single Supply) FPGA – Using the ADC Feature and VCCIO[2..8] Pins are Powered Up 1.0V/1.2V/1.35V/1.5V/1.8V/2.5V (E144, M153, U169, and U324 Packages)

Example Requiring 3 Power Regulator

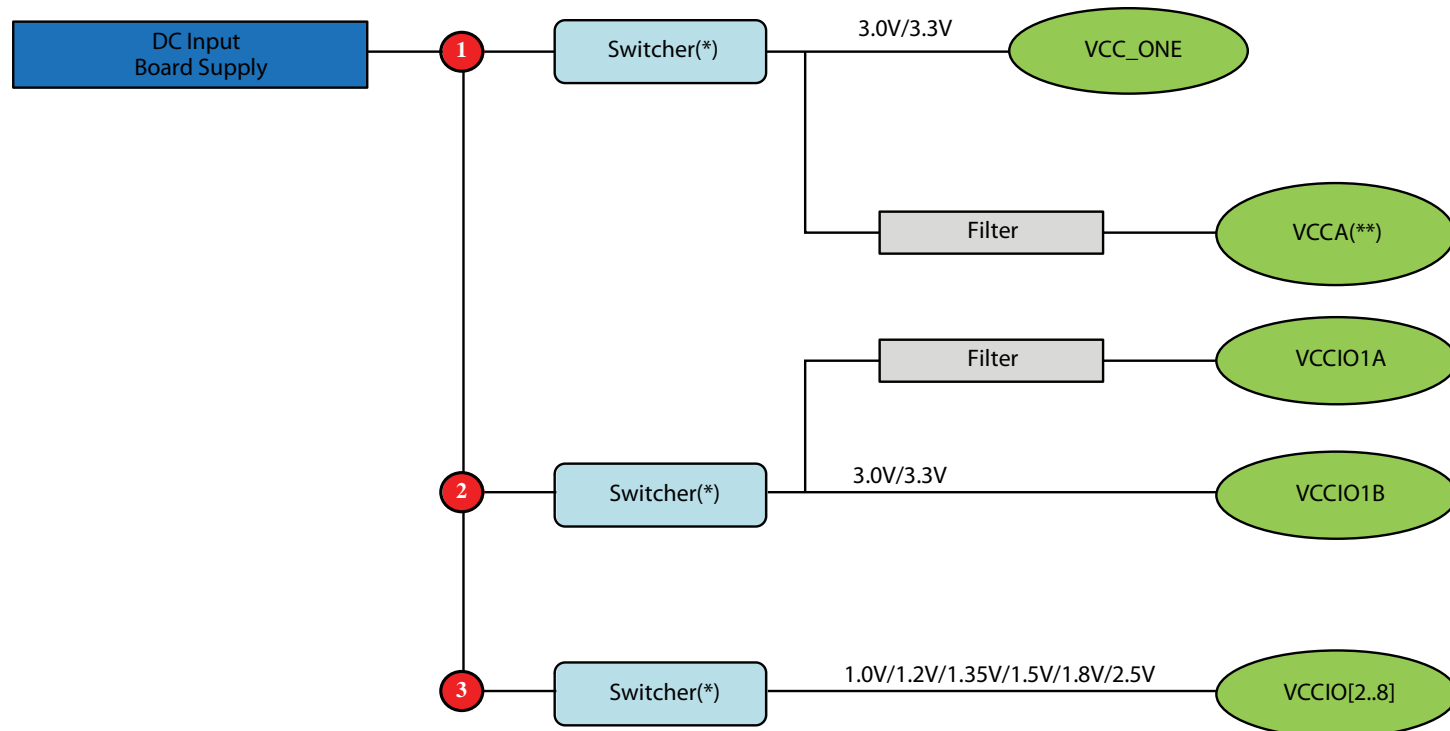
Power Pin Name	Regulator Count	Voltage Level (V)	Supply Tolerance	Power Source	Regulator Sharing	Notes
VCC_ONE	1	3.0/3.3	± 5%	Switcher (*)	Share	Both VCCA and VCC_ONE must share a single power source using proper isolation filter.
VCCA					Isolate	
VCCIO1B	2	3.0/3.3	± 5%	Switcher (*)	Share	You have the option to share VCCIO1A with VCCIO1B using proper isolation filter.
VCCIO1A					Isolate	
VCCIO[2..8]	3	Varies	± 5%	Switcher (*)	Share	Individual power rail.

(*)When using a switcher to supply these voltages, the switcher must be a low noise switcher as defined in note 9 of the *Notes to the Intel MAX 10 FPGA Pin Connection Guidelines*.

Notes:

1. Use the EPE (Early Power Estimation) tool to assist in determining the power required for your specific design.
2. Each board design requires its own power analysis to determine the required power regulators needed to satisfy the specific board design requirements. An example block diagram using the Intel MAX 10 FPGA device is provided in Figure 6.
3. Refer to the *Intel MAX 10 FPGA Configuration User Guide* for maximum ramp rate requirement.

Figure 6. Example Power Supply Sharing Guidelines for Intel MAX 10 (Single Supply) FPGA – Using the ADC Feature and VCCIO[2..8] Pins are Powered Up 1.0V/1.2V/1.35V/1.5V/1.8V/2.5V (E144, M153, U169, and U324 Packages)



(*) When using a switcher to supply these voltages, the switcher must be a low noise switcher as defined in *note 9* of the Notes to the Intel MAX 10 FPGA Pin Connection Guidelines.

(**) The VCCA power supply requires ferrite bead filter for noise isolation.

Document Revision History for Intel MAX 10 FPGA Device Family Pin Connection Guidelines

Document Version	Changes
2022.05.27	Removed all instances of Enpirion from <i>Example 1—Intel MAX 10 (Dual Supply) FPGA</i> , <i>Example 2—Intel MAX 10 (Dual Supply) FPGA</i> , <i>Example 3—Intel MAX 10 (Dual Supply) FPGA</i> , <i>Example 4—Intel MAX 10 (Single Supply) FPGA</i> , <i>Example 5—Intel MAX 10 (Single Supply) FPGA</i> and <i>Example 6—Intel MAX 10 (Single Supply) FPGA</i> sections.
2022.04.27	Updated the pin description of the ADC[1..2]IN[1..16] pins.
2021.11.26	Updated the connection guidelines of the VCCIO[#] pins for the <i>Intel MAX 10 (Single Supply) FPGA</i> and <i>Intel MAX 10 (Dual Supply) FPGA</i> .
2021.11.01	<ul style="list-style-type: none"> Updated the 1.0-V VCCIO note in the connection guidelines of the VCCIO[#] pins in the <i>Intel MAX 10 (Single Supply) FPGA</i> and <i>Intel MAX 10 (Dual Supply) FPGA</i>.
2020.06.30	<ul style="list-style-type: none"> Added the 1.0-V support to the VCCIO[#] pins for the <i>Intel MAX 10 (Single Supply) FPGA</i> and <i>Intel MAX 10 (Dual Supply) FPGA</i>. Added 1.0-V support to the VCCIO power supplies in the following power supply sharing guidelines: <ul style="list-style-type: none"> Example Power Supply Sharing Guidelines for Intel MAX 10 (Dual Supply) FPGA – The ADC Feature is Not Used Example Power Supply Sharing Guidelines for Intel MAX 10 (Dual Supply) FPGA – Using the ADC Feature and VCCIO[2..8] Pins are Powered Up at 1.0V/1.2V/1.35V/1.5V/1.8V/2.5V/3.0V/3.3V Example Power Supply Sharing Guidelines for Intel MAX 10 (Single Supply) FPGA (E144, M153, U169, and U324 Packages) – The ADC Feature is Not Used Example Power Supply Sharing Guidelines for Intel MAX 10 (Single Supply) FPGA – Using the ADC Feature and VCCIO[2..8] Pins are Powered Up 1.0V/1.2V/1.35V/1.5V/1.8V/2.5V (E144, M153, U169, and U324 Packages)
2019.07.01	<ul style="list-style-type: none"> Updated the connection guidelines about how to enable the ADC feature in the VCCIO[#] pin of the <i>Intel MAX 10 (Single Supply) FPGA</i> table. Updated the connection guidelines of the DEV_CLRN and DEV_OE pins.
2019.02.20	Updated the connection guidelines of the DEV_CLRN and DEV_OE pins.
2019.01.29	<ul style="list-style-type: none"> Updated the connection guidelines of the CONFIG_SEL pin. Updated note 10 in the <i>Notes to the Intel MAX 10 FPGA Pin Connection Guidelines</i> section.

Date	Version	Description of Changes
December 2017	2017.12.15	<ul style="list-style-type: none"> Added the support for the U324 package. Added a reference for the UFM and CFM power-down requirement to the VCCA[1..6] pins for the Intel MAX 10 (Single Supply) FPGA and VCCA[1..4] pins for the Intel MAX 10 (Dual Supply) FPGA. Added a note to provide references for each PLL clock output of the PLL_[L,R,B,T]_CLKOUTp and PLL_[L,R,B,T]_CLKOUTn pins.
continued...		

Date	Version	Description of Changes
		<ul style="list-style-type: none"> Updated the voltage overshoot connection guidelines for the TCK, TDO, TDI, and TMS pins. Updated the Voltage Sensor Pins section to Analog Input Pins. Removed PowerPlay text from tool name.
June 2017	2017.06.16	Updated the connection guidelines of the JTAGEN pin.
February 2017	2017.02.21	<ul style="list-style-type: none"> Rebranded as Intel.
December 2016	2016.12.09	<ul style="list-style-type: none"> Updated the connection guidelines of the TDI and TMS pins.
May 2016	2016.05.02	<ul style="list-style-type: none"> Added note (5) to the following power sharing guidelines: <ul style="list-style-type: none"> Example 1. Power Supply Sharing Guidelines for MAX 10D (Dual Supply) FPGA – The ADC Feature is Not Used Example 2. Power Supply Sharing Guidelines for MAX 10D (Dual Supply) FPGA – Using the ADC Feature and VCCIO[2..8] Pins are Powered Up at 2.5V Example 3. Power Supply Sharing Guidelines for MAX 10D (Dual Supply) FPGA – Using the ADC Feature and VCCIO[2..8] Pins are Powered Up at 1.2V/1.35V/1.5V/1.8V/3.0V/3.3V Added note (4) to the following power sharing guidelines: <ul style="list-style-type: none"> Example 4. Power Supply Sharing Guidelines for MAX 10S (Single Supply) FPGA (E144, M153, and U169 Packages) – The ADC Feature is Not Used Example 5. Power Supply Sharing Guidelines for MAX 10S (Single Supply) FPGA – Using the ADC Feature and VCCIO[2..8] Pins are Powered Up 3.0V/3.3V Example 6. Power Supply Sharing Guidelines for MAX 10S (Single Supply) FPGA – Using the ADC Feature and VCCIO[2..8] Pins are Powered Up 1.2V/1.35V/1.5V/1.8V/2.5V Updated CONF_DONE should pulled up to VCCIO Bank 8. Removed Note 3 “The voltage level for each power rail is preliminary.” from the following power sharing guidelines: <ul style="list-style-type: none"> Example 1. Power Supply Sharing Guidelines for MAX 10D (Dual Supply) FPGA – The ADC Feature is Not Used Example 2. Power Supply Sharing Guidelines for MAX 10D (Dual Supply) FPGA – Using the ADC Feature and VCCIO[2..8] Pins are Powered Up at 2.5V Example 3. Power Supply Sharing Guidelines for MAX 10D (Dual Supply) FPGA – Using the ADC Feature and VCCIO[2..8] Pins are Powered Up at 1.2V/1.35V/1.5V/1.8V/3.0V/3.3V Example 4. Power Supply Sharing Guidelines for MAX 10S (Single Supply) FPGA (E144, M153, and U169 Packages) – The ADC Feature is Not Used Example 5. Power Supply Sharing Guidelines for MAX 10S (Single Supply) FPGA – Using the ADC Feature and VCCIO[2..8] Pins are Powered Up 3.0V/3.3V (E144, M153, and U169 Packages) Example 6. Power Supply Sharing Guidelines for MAX 10S (Single Supply) FPGA – Using the ADC Feature and VCCIO[2..8] Pins are Powered Up 1.2V/1.35V/1.5V/1.8V/2.5V (E144, M153, and U169 Packages)
November 2015	2015.11.02	<ul style="list-style-type: none"> Changed instances of Quartus II to Quartus Prime. Updated the connection guidelines of the VCCIO[#] pins. Updated the connection guidelines of the TDI and TMS pins.
continued...		

Date	Version	Description of Changes
June 2015	2015.06.12	Added the DNU pin.
May 2015	2015.05.06	<ul style="list-style-type: none"> Added the following power sharing guidelines: <ul style="list-style-type: none"> Example 2. Power Supply Sharing Guidelines for MAX 10D (Dual Supply) FPGA – Using the ADC Feature and VCCIO[2..8] Pins are Powered Up at 2.5V Example 3. Power Supply Sharing Guidelines for MAX 10D (Dual Supply) FPGA – Using the ADC Feature and VCCIO[2..8] Pins are Powered Up at 1.2V/1.35V/1.5V/1.8V/3.0V/3.3V Example 5. Power Supply Sharing Guidelines for MAX 10S (Single Supply) FPGA – Using the ADC Feature and VCCIO[2..8] Pins are Powered Up 3.0V/3.3V (E144, M153, and U169 Packages) Example 6. Power Supply Sharing Guidelines for MAX 10S (Single Supply) FPGA – Using the ADC Feature and VCCIO[2..8] Pins are Powered Up 1.2V/1.35V/1.5V/1.8V/2.5V (E144, M153, and U169 Packages) Updated the following power sharing guidelines: <ul style="list-style-type: none"> Example 1. Power Supply Sharing Guidelines for MAX 10D (Dual Supply) FPGA – The ADC Feature is Not Used Updated the pin description of the DPCLK[0..3] pins. Updated the connection guidelines of the VCCIO[#] pins. Updated the connection guidelines of the ADC[1..2]IN[1..16] pins. Updated the connection guidelines of the ADC_VREF pin.
January 2015	2015.01.29	<ul style="list-style-type: none"> Updated the connection guidelines of the DPCLK[0..3] pins. Updated the connection guidelines of the PLL_[L,R,B,T]_CLKOUTp and PLL_[L,R,B,T]_CLKOUTn pins. Updated the connection guidelines of the VREFB<#>N0 pins. Updated the pin description of the ADC[1..2]IN[1..16] pins.
December 2014	2014.12.15	<ul style="list-style-type: none"> Added note 10 in the Notes to Pin Connection Guidelines. Added note (**) to Figure 2. Updated the pin name from BOOT_SEL to CONFIG_SEL. Updated the pin description of the CONFIG_SEL pin. Updated the connection guidelines of the VCC_ONE pin. Updated the connection guidelines of the nSTATUS pin. Updated the connection guidelines of the CONF_DONE pin. Updated note 4 in the Notes to Pin Connection Guidelines.
September 2014	2014.09.22	Initial release.



Intel® MAX® 10 FPGA Design Guidelines



Online Version



Send Feedback

M10-GUIDELINES

ID: **683196**

Version: **2020.10.19**

Contents

1. Intel® MAX® 10 FPGA Design Guidelines.....	4
1.1. Intel® MAX® 10 FPGA Design Guidelines.....	4
1.2. Before You Begin.....	4
1.2.1. Read through the Device Overview of the FPGA.....	5
1.2.2. Estimate design requirements.....	5
1.2.3. Review available design tools.....	5
1.2.4. Review available IP.....	5
1.3. Design Specifications.....	6
1.3.1. Create detailed design specifications.....	6
1.3.2. Create detailed functional verification or test plan.....	6
1.3.3. Select IP that affects system design, especially I/O interfaces.....	6
1.3.4. Ensure your board design supports the Intel FPGA IP Evaluation Mode tethered mode.....	6
1.3.5. Review available system development tools.....	7
1.4. Device Selection.....	7
1.4.1. Consider the available device variants.....	7
1.4.2. Estimate the required logic, memory, and multiplier density.....	7
1.4.3. Consider vertical device migration availability and requirements.....	8
1.4.4. Review resource utilization reports of similar designs.....	8
1.4.5. Reserve device resources for future development and debugging.....	8
1.4.6. Estimate the number of I/O pins that you require.....	8
1.4.7. Consider the I/O pins you need to reserve for debugging.....	9
1.4.8. Verify that the number of LVDS channels are enough.....	9
1.4.9. Verify the number of PLLs and clock routing resources.....	9
1.4.10. Determine the device speed grade that you require.....	9
1.4.11. Determine the number of images supported for the device.....	10
1.5. Board Design.....	10
1.5.1. Early Board Design.....	10
1.5.2. Power Pin Connections.....	13
1.5.3. Configuration Pin Connections.....	15
1.5.4. General I/O Pin Connections.....	17
1.6. I/O and Clock Planning.....	20
1.6.1. Early Pin Planning and I/O Assignment Analysis.....	20
1.6.2. I/O Features and Pin Connections.....	21
1.6.3. Clock Planning.....	27
1.6.4. I/O Simultaneous Switching Noise.....	28
1.7. Design Entry.....	29
1.7.1. Use synchronous design practices.....	29
1.7.2. Consider the following recommendations to avoid clock signals problems:.....	30
1.7.3. Use IP cores with the parameter editor.....	30
1.7.4. Review the information on dynamic reconfiguration feature.....	30
1.7.5. Consider the Intel's recommended coding styles to achieve optimal synthesis results.....	30
1.7.6. Enable the chip-wide reset to clear all registers if required.....	31
1.7.7. Use device architecture-specific register control signals.....	31
1.7.8. Review recommended reset architecture.....	31
1.7.9. Review the synthesis options available in your synthesis tool.....	32

1.7.10. Consider resources available for register power-up and control signals.....	32
1.7.11. Consider Intel's recommendations for creating design partitions.....	33
1.7.12. Perform timing budgeting and resource balancing between partitions.....	33
1.7.13. Create a design floorplan for incremental compilation partitions.....	34
1.8. Design Implementation.....	34
1.8.1. Synthesis and Compilation.....	34
1.8.2. Timing Optimization and Analysis.....	37
1.8.3. Formal Verification.....	40
1.8.4. Power Analysis and Optimization.....	40
1.9. Document Revision History for Intel MAX 10 Device Design Guidelines.....	43

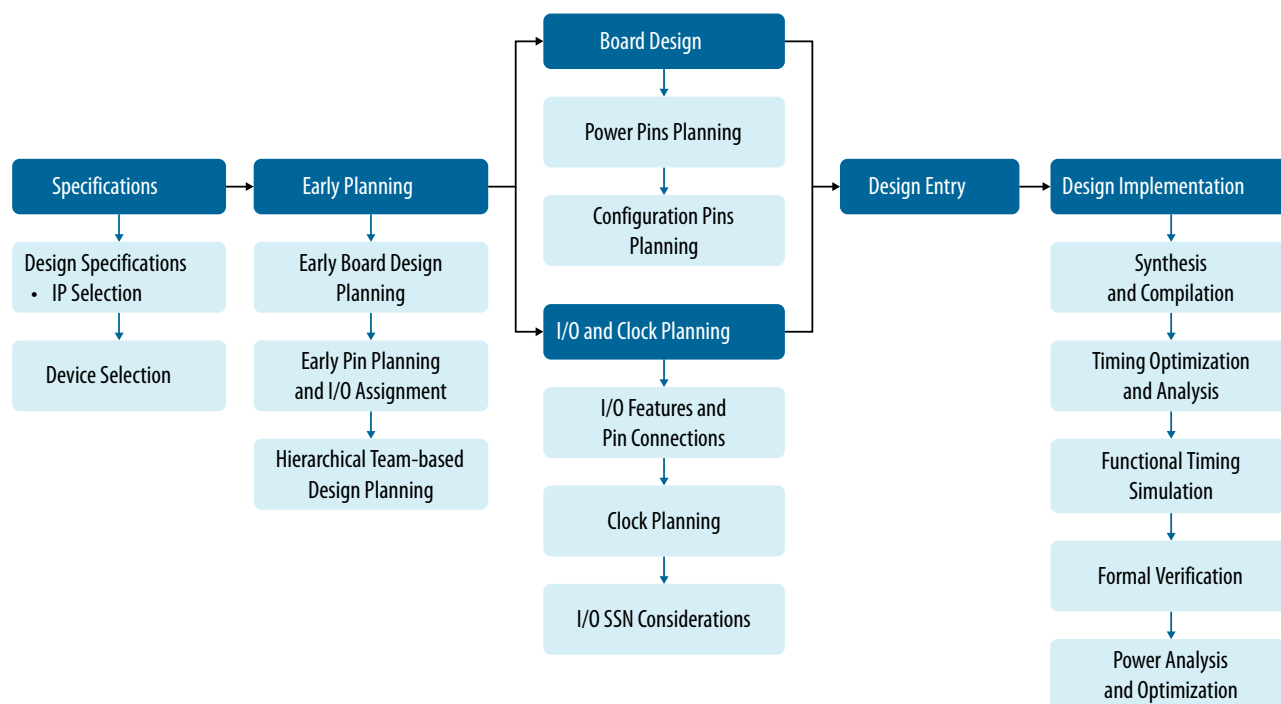
1. Intel® MAX® 10 FPGA Design Guidelines

1.1. Intel® MAX® 10 FPGA Design Guidelines

This application note provides a set of checklists that consist of design guidelines, recommendations, and factors to consider when you create designs using Intel® MAX® 10 FPGAs.

- Use this document to help you plan the FPGA and system early in the design process, which is crucial for a successful design.
- Follow Intel's recommendations throughout the design process to achieve good results, avoid common issues, and improve your design productivity.

Figure 1. Intel MAX 10 Design Flow



1.2. Before You Begin

Before you begin planning and designing your FPGA system, familiarize yourself with the FPGA device features, and the design tools and IP that are available for the Intel MAX 10 device family.

1.2.1. Read through the Device Overview of the FPGA

The Device Overview provides an overview of the capabilities and options available for a device family. Read through the document to familiarize yourself with the device family offerings and general features.

Related Information

[Intel MAX 10 FPGA Device Overview](#)

1.2.2. Estimate design requirements

Create a rough estimate of the design in the following terms:

- Basic functions of the product
- Similar previous designs
- General device requirements

1.2.3. Review available design tools

Consider the available design, estimators, system builders, and verification tools. The following items are some of the available tools provided by Intel:

- Intel Quartus® Prime software for design, synthesis, simulation, and programming; including integration with Platform Designer (Standard), simulation tools, and verification tools.
- Platform Designer (Standard) system integration tool—next-generation tool that automatically generates interconnect logic to connect intellectual property (IP) functions and subsystems.
- Mentor Graphics* ModelSim* - Intel FPGA Edition simulation software.
- Timing Analyzer for static timing analysis with support for Synopsys* Design Constraints (SDC) format.
- Power Analyzer for power analysis and optimization.
- Signal Probe and Signal Tap II Logic Analyzer debugging tools.
- External Memory Interface Toolkit available in the Intel Quartus Prime software.

Related Information

- [Intel Quartus Prime Software Suite and FPGA Development Tools](#)
For more information design tools
- [Design Software Support](#)
For more information on software support
- [AN 632: SOPC Builder to Qsys Migration Guidelines](#)
For a guideline to migrate from SOPC Builder to Platform Designer (Standard)

1.2.4. Review available IP

Intel and its FPGA IP partners offer a large selection of parameterized blocks of IP cores optimized for Intel FPGAs that you can implement to reduce your implementation and verification time.

Related Information[Intel FPGA Intellectual Property](#)

1.3. Design Specifications

Typically, the FPGA is an important part of the overall system and affects the rest of the system design. Use the following checklist to start your design process.

1.3.1. Create detailed design specifications

Before you create your logic design or complete your system design, perform the following:

- Specify the I/O interfaces for the FPGA
- Identify the different clock domains
- Include a block diagram of basic design functions
- Consider a common design directory structure—if your design includes multiple designers, a common design directory structure eases the design integration stages.
- When performing any UFM write or erase operation, make sure you provide stable power connection. Loss of power supply during a write or erase operation can cause damage to the device.

1.3.2. Create detailed functional verification or test plan

A functional verification plan ensures the team knows how to verify the system. Creating a test plan at an early stage helps you design for testability and manufacturability.

For example, if you plan to perform built-in-self test (BIST) functions to drive interfaces, you can plan to use a UART interface with a Nios® II processor inside the FPGA device.

Related Information

[Review available on-chip debugging tools](#) on page 11

1.3.3. Select IP that affects system design, especially I/O interfaces

Include intellectual property (IP) blocks in your detailed design specifications. Taking the time to create these specifications improves design efficiency.

Related Information

[Intel FPGA Intellectual Property](#)

For a list of available IP offered by Intel and Intel FPGA IP partners

1.3.4. Ensure your board design supports the Intel FPGA IP Evaluation Mode tethered mode

You can program your FPGA and verify your design in hardware before you purchase an IP license by using the Intel FPGA IP Evaluation Mode feature available for many IP cores. Intel FPGA IP Evaluation Mode supports the following modes:

- Untethered—your design runs for a limited time.
- Tethered—your design runs for the duration of the hardware evaluation period. This mode requires an Intel FPGA download cable connected to the JTAG port on your board and a host computer that runs the Intel Quartus Prime Programmer. If you plan to use this mode, ensure that your board design supports this mode.

1.3.5. Review available system development tools

Intel provides a complete suite of development tools for every stage of your design.

Whether you are creating a complex FPGA design as a hardware engineer, writing software for an embedded processor as a software developer, modeling a digital signal processing (DSP) algorithm, or focusing on system design, Intel has a tool that can help.

Related Information

- [Intel Quartus Prime Software Suite and FPGA Development Tools](#)
- [Design Software Support](#)

1.4. Device Selection

Use the following checklist to determine the device variant, density, and package combination that is suitable for your design.

1.4.1. Consider the available device variants

The Intel MAX 10 FPGA family consist of several device variants that are optimized for different application requirements.

Select a device based on I/O pin count, LVDS channels, package offering, logic/memory/multiplier density, single/dual supply device options, PLLs, clock routing, and speed grade.

Consider the following feature options:

- Compact
- Flash
- Analog-to-digital converter (ADC)

Related Information

[Intel MAX 10 FPGA Device Overview](#)

1.4.2. Estimate the required logic, memory, and multiplier density

Intel MAX 10 devices offer a range of densities that provide different amounts of device logic resources. Determining the required logic density can be a challenging part of the design planning process. Devices with more logic resources can implement larger and potentially more complex designs but generally have a higher cost. Smaller devices have lower static power utilization.

Related Information

[Intel MAX 10 Embedded Multipliers User Guide](#)

1.4.3. Consider vertical device migration availability and requirements

Determine whether you want the flexibility of migrating your design to another device density. Choose your device density and package to accommodate any possible future device migration to allow flexibility when the design nears completion.

To verify the pin migration compatibility, use the **Pin Migration View** window in the Intel Quartus Prime software Pin Planner. The **Pin Migration View** window helps you identify the difference in pins that can exist between migration devices:

- If one device has pins for connection to V_{CC} or GND but are I/O pins on a different device, the Intel Quartus Prime software ensures these pins are not used for I/O. For migration, ensure that these pins are connected to the correct PCB plane.
- If you are migrating between two devices in the same package, connect the pins that are not connected to the smaller die to V_{CC} or GND on the larger die in your original design.

Related Information

[I/O Management](#)

For more information about verifying the pin migration compatibility

1.4.4. Review resource utilization reports of similar designs

If you have other designs that target an Intel device, you can use their resource utilization as an estimate for your new design. Coding style, device architecture, and optimization options used in the Intel Quartus Prime software can significantly affect resource utilization and timing performance of a design.

To estimate resource utilization for certain configurations of Intel's FPGA IP designs, refer to the respective Intel MAX 10 user guides.

1.4.5. Reserve device resources for future development and debugging

Select a device that meets your design requirements with some safety margin in case you want to add more logic later in the design cycle, upgrade, or expand your design. You may also want additional space in the device to ease design floorplan creation for an incremental or team-based design.

Related Information

[Consider reserving resources for debugging](#) on page 11

1.4.6. Estimate the number of I/O pins that you require

Determine the required number of I/O pins for your application, considering the design's interface requirements with other system blocks. You can compile any existing designs in the Intel Quartus Prime software to determine how many I/O pins are used.

Other factors can also affect the number of I/O pins required for a design, including simultaneous switching noise (SSN) concerns, pin placement guidelines, pins used as dedicated inputs, I/O standard availability for each I/O bank, differences between I/O standards and speed for row and column I/O banks, and package migration options.

Related Information

- [I/O and Clock Planning](#) on page 20
- [Board Design](#) on page 10

1.4.7. Consider the I/O pins you need to reserve for debugging

Intel provides a complete design debugging environment that easily adapts to your specific design requirements. When planning to debugging, you should decide which I/O pins you need to reserve for debugging.

Related Information

[Consider the guidelines to plan for debugging tools](#) on page 11

1.4.8. Verify that the number of LVDS channels are enough

Larger densities and package pin counts offer more full-duplex LVDS channels for differential signaling. Ensure that your device density-package combination includes enough LVDS channels.

1.4.9. Verify the number of PLLs and clock routing resources

Verify that your chosen device density package combination includes enough PLLs and clock routing resources for your design. GCLK resources are shared between certain PLLs, which can affect the inputs that are available for use.

Related Information

[I/O and Clock Planning](#) on page 20

For more details and references regarding clock pins and global routing resources

1.4.10. Determine the device speed grade that you require

The device speed grade affects the device timing performance and timing closure, as well as power utilization. One way to determine which speed grade your design requires is to consider the supported clock rates for specific I/O interfaces.

You can use the fastest speed grade while prototyping to reduce compilation time because less time is spent optimizing the design to meet timing requirements. If the design meets the timing requirements, you can then move to a slower speed grade for production to reduce cost.

When migrating to a device of different speed grade, check the timing report from the timing analysis to ensure that there is no timing violation between different blocks within the Intel MAX 10 device, between Intel MAX 10 devices and other devices on the board.

Always design with a sufficient timing margin so that your design can work on devices of different speed grades.

Related Information

- [External Memory Interface Spec Estimator](#)
For information about supported clock rates for memory interfaces using I/O pins on different sides of the device in different device speed grades

- [Intel MAX 10 FPGA Device Datasheet](#)
For information about the available speed grades

1.4.11. Determine the number of images supported for the device

Select a device that support dual configuration images, two FPGA bitstreams, if dual configuration feature is needed in your design. All Intel MAX 10 devices support the dual configuration feature, except 10M02 device.

1.5. Board Design

1.5.1. Early Board Design

Early planning allows the FPGA team to provide early information to PCB board and system designers.

1.5.1.1. Select a configuration scheme

Intel offers a wide range of configuration solutions to configure Intel MAX 10 devices.

Related Information

[Intel MAX 10 FPGA Configuration User Guide](#)

For information about the internal configuration scheme, and the necessary and optional pin settings

1.5.1.2. Ensure board support the required features:

- Data decompression—if you enable data compression, the storage requirement and the programming time (writing to flash) are reduced. The configuration time (writing to CRAM) is increased.
- Design security—this feature utilizes a 128-bit security key to protect the designs from unauthorized copying, reverse engineering, and tampering. The devices can decrypt configuration bitstreams using the AES algorithm. Design security is not available for the JTAG configuration scheme.
- Dual configuration—this feature is supported only in self-download mode.
- SEU mitigation—dedicated circuitry in the devices perform cyclic redundancy check (CRC) error detection and check for SEU errors automatically. To detect SEU errors, use the `CRC_ERROR` pin to flag errors and design your system to take appropriate action. If you do not enable the CRC error detection feature, you can also use the `CRC_ERROR` pin as a design I/O pin.

Related Information

[Intel MAX 10 FPGA Configuration User Guide](#)

1.5.1.3. Plan for the Auto-restart after configuration error option

To reset the device internally by driving the `nSTATUS` pin low when a configuration error occurs, enable the **Auto-restart after configuration error** option. The device releases its `nSTATUS` pin after the reset time-out period. This behavior allows you to re-initiate the configuration cycle. The `nSTATUS` pin requires an external 10-kΩ pull-up resistor to V_{CCIO} .

1.5.1.4. Estimating configuration file size

To estimate the configuration file size, convert your configuration file in uncompressed Raw Binary File (.rbf). The .rbf file size provides the approximate uncompressed configuration file sizes.

Use uncompressed .rbf size only to estimate the file size before design compilation. Different configuration file formats, such as Hexadecimal (Intel-Format) File (.hex) or Tabular Text File (.tff) format, have different file sizes.

Related Information

[Intel MAX 10 FPGA Configuration User Guide](#)

For more information on the uncompressed .rbf sizes for Intel MAX 10 devices

1.5.1.5. Review available on-chip debugging tools

Take advantage of on-chip debugging features to analyze internal signals and perform advanced debugging techniques.

Different debugging tools work better for different systems and different designers. Early planning can reduce the time spent debugging, and eliminates design changes later to accommodate your preferred debugging methodologies. Adding debug pins may not be enough, because of internal signal and I/O pin accessibility on the device.

Related Information

- [System Debugging Tools Overview](#)
For more information about in-system debugging tools in the Quartus Prime software
- [Virtual JTAG \(sld_virtual_jtag\) IP Core User Guide](#)
For more information about virtual JTAG

1.5.1.6. Consider the guidelines to plan for debugging tools

- Select on-chip debugging schemes early to plan memory and logic requirements, I/O pin connections, and board connections.
- If you want to use Signal Probe incremental routing, the Signal Tap II Embedded Logic Analyzer, Logic Analyzer Interface, In-System Memory Content Editor, In-System Sources and Probes, or Virtual JTAG IP core, plan your system and board with JTAG connections that are available for debugging.
- Plan for the small amount of additional logic resources used to implement the JTAG hub logic for JTAG debugging features.
- For debugging with the Signal Tap II Embedded Logic Analyzer, reserve device memory resources to capture data during system operation.
- Reserve I/O pins for debugging with Signal Probe or the Logic Analyzer Interface so that you do not have to change the design or board to accommodate debugging signals later.
- Ensure the board supports a debugging mode where debugging signals do not affect system operation.
- Incorporate a pin header or micro connector as required for an external logic analyzer or mixed signal oscilloscope.

- To use debug tools incrementally and reduce compilation time, ensure incremental compilation is on so you do not have to recompile the design to modify the debug tool.
- To use the Virtual JTAG IP core for custom debugging applications, instantiate it in the HDL code as part of the design process.
- To use the In-System Sources and Probes feature, instantiate the IP core in the HDL code.
- To use the In-System Memory Content Editor for RAM or ROM blocks or the LPM_CONSTANT IP core, turn on the **Allow In-System Memory Content Editor** option to capture and update content independently of the system clock option for the memory block in the parameter editor.

Related Information

- [Quick Design Debugging Using Signal Probe](#)
For more information about debugging tools and methods
- [Design Debugging Using the Signal Tap II Logic Analyzer](#)
For more information about debugging with Signal Tap II
- [In-System Debugging Using External Logic Analyzers](#)
For more information about debugging with logic analyzers
- [In-System Updating of Memory and Constants](#)
For more information about debugging memory
- [Design Debugging Using In-System Sources and Probes](#)
For more information about debugging sources and probes
- [Virtual JTAG \(sld_virtual_jtag\) IP Core User Guide](#)
For more information about virtual JTAG

1.5.1.7. Use the Early Power Estimator (EPE) to estimate power supplies and cooling solution

FPGA power consumption depends on logic design and is challenging to estimate during early board specification and layout. However, it is an important design consideration and must be estimated accurately to develop an appropriate power budget to design the power supplies, voltage regulators, decoupling capacitors, heat sink, and cooling system.

Use the EPE spreadsheet to estimate power, current, and device junction temperature before you have a complete design. The EPE calculates the estimated information based on device information, planned device resources, operating frequency, toggle rates, ambient temperature, heat sinks information, air flow, board thermal model, and other environmental considerations.

- If you have an existing or partially-completed and compiled design—use the **Generate Early Power Estimator File** command in the Intel Quartus Prime software to provide input to the EPE spreadsheet.
- If you do not have an existing design—estimate manually the number of device resources used in your design and input into the EPE spreadsheet. If the device resources information changes during or after the design phase, your power estimation results will be less accurate.

Related Information

- [Early Power Estimators \(EPE\)](#)

- [Use power distribution network \(PDN\) tool to plan for power distribution and decoupling capacitor selection](#) on page 14

1.5.2. Power Pin Connections

The Intel MAX 10 devices require various voltage supplies depending on your design requirements. Use the following checklist to design the board for the FPGA power pin connections.

1.5.2.1. Design the board for power-up

Intel MAX 10 devices support hot socketing (hot plug-in/hot swap) and power sequencing without the use of external devices. Consider the following guidelines:

- During power-up, the output buffers are tri-stated and the internal weak pull-up resistors are disabled by default. You can enable the internal weak pull-up resistors through the Intel Quartus Prime software.
- with weak pull-up resistors enabled until the device is configured and configuration pins drive out.
- Design the voltage power supply ramps to be monotonic—ensure that the minimum current requirement for the power-on-reset (POR) supplies is available during device power up. The following are the POR monitored power supplies:
 V_{CC} or V_{CC_ONE} (after regulated down)
 V_{CCIO} of bank 1B and bank 8
 V_{CCA}
- You can extend the POR delay by using an external component to assert the `nSTATUS` pin low. To ensure the device configures properly and enters user mode, extend the POR delay if the board cannot meet the maximum power ramp time specifications.
- Design power sequencing and voltage regulators for the best device reliability—although power sequencing is not required for correct operation, consider the power-up timing of each rail to prevent problems with long-term device reliability if you are designing a multi-rail powered system.
- Take advantage of the power up sequence for instant-on feature. With instant-on, the device can directly enter user mode with the shortest time after power supplies reach the required level. During power up, the control block reads the POR delay value and instant-on setting bits. If the instant-on is set, the device directly enters the initialization phase. If the instant-on feature is not selected, the POR delay value delays the POR signal. Clear the DSM if you want to change the setting.
- Connect the GND between boards before connecting the power supplies—Intel uses GND as a reference for hot-socketing operations and I/O buffer designs. Connecting the GND between boards before connecting the power supplies prevents the GND on your board from being pulled up inadvertently by a path to power through other components on your board. A pulled up GND could otherwise cause an out-of-specification I/O voltage or current condition with the device.

Related Information

- [Intel MAX 10 Power Management User Guide](#)
- [Intel MAX 10 FPGA Device Family Pin Connection Guidelines](#)

- [Intel MAX 10 FPGA Configuration User Guide](#)

1.5.2.2. Review the list of required supply voltages and the power supply options

Intel MAX 10 devices offer single and dual supply device options.

Related Information

[Intel MAX 10 FPGA Device Datasheet](#)

For a list of the required supply voltages and the recommended operating conditions

1.5.2.3. Ensure I/O power pin compatibility with I/O standards

The output pins of the devices will not conform to the I/O standard specifications if the V_{CCIO} level is out of the recommended operating range of the I/O standard.

Related Information

[Intel MAX 10 General Purpose I/O User Guide](#)

For a complete list of the supported I/O standards and V_{CCIO} voltages

1.5.2.4. Ensure correct power pin connections

- Connect all power pins correctly.
- Connect V_{CCIO} pins and V_{REF} pins to support the I/O standards of each bank.
- For unused supplies, consider whether there is a need to ground, open, or retain the power connection.

Related Information

[Intel MAX 10 FPGA Device Family Pin Connection Guidelines](#)

For more information on connecting power pins

1.5.2.5. Determine power rail sharing

- Explore unique requirements for FPGA power pins or other power pins on your board, and determine which devices on your board can share a power rail. It is especially important for you to consider the power supply sharing ability of devices from different device families.
- Follow the suggested power supply sharing and isolation guidance, and the specific guidelines for each pin.

Related Information

- [Intel MAX 10 FPGA Device Family Pin Connection Guidelines](#)

For more information about power sharing

- [AN 583: Designing Power Isolation Filters with Ferrite Beads for Altera FPGAs](#)

For more information on isolation guidance

1.5.2.6. Use power distribution network (PDN) tool to plan for power distribution and decoupling capacitor selection

Intel MAX 10 devices include on-die decoupling capacitors to provide high-frequency decoupling.

To plan power distribution and return currents from the voltage regulating module to the FPGA power supplies, you can use the PDN design tool that optimizes the board-level PDN graphically. Although you can use SPICE simulation to simulate the circuit, the PDN design tool provides a fast, accurate, and interactive way to determine the right number of decoupling capacitors for optimal cost and performance trade-off.

Related Information

- [AN 574: Printed Circuit Board \(PCB\) Power Delivery Network \(PDN\) Design Methodology](#)
- [Power Delivery Network \(PDN\) Tool User Guide](#)

1.5.2.7. Review the following guidelines for PLL board design

- Connect all PLL power pins to power supplies to reduce noise even if the design does not use all the PLLs.
- Power supply nets should be provided by an isolated power plane, a power plane cut out, or a thick trace.

Related Information

[Intel MAX 10 FPGA Device Family Pin Connection Guidelines](#)
For pin voltage requirements

1.5.3. Configuration Pin Connections

Depending on your configuration scheme, different pull-up or pull-down resistor, signal integrity, and specific pin requirements apply. Connecting the configuration pins correctly is important. Use the following checklist to address common issues.

1.5.3.1. Verify configuration pin connections and pull-up or pull-down resistors are correct for your configuration schemes

Intel provides pin connection guidelines to help you plan your design. These guidelines describe each pin and give guidelines for their use.

Related Information

[Intel MAX 10 FPGA Device Family Pin Connection Guidelines](#)
For specifics about each configuration pin

1.5.3.2. Design configuration TCK pin using the same technique as in designing high-speed signal or system clock

- Noise on the TCK signal can affect JTAG configuration.
- For a chain of devices, noise on the TCK pin in the chain can cause JTAG programming or configuration to fail for the entire chain.
- For a chain of devices, ensure all devices in the JTAG chain are powered on during JTAG programming or configuration.

1.5.3.3. Verify the JTAG pins are connected to a stable voltage level if not in use

JTAG configuration takes precedence over all configuration methods. If you do not use the JTAG interface, do not leave the JTAG pins floating or toggling during configuration.

To disable the JTAG circuitry, connect TCK pin to GND through a 1-k Ω resistor. Connect TMS and TDI pins to V_{CCIO} through a 1-k Ω resistor. Leave TDO unconnected.

1.5.3.4. Verify the JTAG pin connections to the download cable header

A device operating in JTAG mode uses the required TDI, TDO, TMS, and TCK pins. The TCK pin does not support internal weak pull-down.

Connect the TCK pin to an external 1-k Ω to 10-k Ω pull-down resistor. The TDI and TMS pins have weak internal pull-up resistors. The JTAG output pin (TDO) and all JTAG input pins are powered by V_{CCIO}. The voltage range is 1.5 V to 3.3 V.

The download cable must be powered at 2.5 V when V_{CCIO} of the JTAG pins are powered at 2.5 V to 3.3 V to prevent voltage overshoot because JTAG pins do not have internal PCI* clamping diodes. The TCK pin must be pulled to ground. If the V_{CCIO} of the JTAG pins are powered at 1.5 V or 1.8 V, the download cable should be powered by the same V_{CCIO}.

1.5.3.5. Review the following JTAG pin connections guidelines:

- If you have multiple devices in the chain, connect the TDO pin of a device to the TDI pin of the next device in the chain.
- Noise on the JTAG pins during configuration, user mode, or power-up can cause the device to go into an undefined state or mode.
- To disable the JTAG state machine during power-up, pull the TCK pin low through a 1-k Ω resistor to ensure that an unexpected rising edge does not occur on TCK.
- Pull TMS and TDI high through a 1-k Ω to 10-k Ω resistor.

1.5.3.6. Ensure the download cable and JTAG pin voltages are compatible

The download cable interfaces with the JTAG pins of your device. The operating voltage supplied to the Intel FPGA download cable by the target board through the 10-pin header determines the operating voltage level of the download cable. The JTAG pins are powered by V_{CCIO}.

In a JTAG chain containing devices with different V_{CCIO} levels, the devices with a higher V_{CCIO} level should drive the devices with the same or lower V_{CCIO} level. A one-level shifter is required at the end of the chain with this device arrangement. If this arrangement is not possible, you have to add more level shifters into the chain.

Related Information

[JTAG Boundary-Scan Testing User Guide](#)

For recommendations about connecting a JTAG chain with multiple voltages across the devices in the chain

1.5.3.7. Buffer the JTAG signal according to the following guidelines:

- If a cable drives three or more devices, buffer the JTAG signal at the cable connector to prevent signal deterioration.
- Anything added to the board that affects the inductance or capacitance of the JTAG signals increases the likelihood that a buffer should be added to the chain.
- Each buffer should drive no more than eight loads for the TCK and TMS signals, which drive in parallel. If jumpers or switches are added to the path, decrease the number of loads.

Related Information

[Intel MAX 10 JTAG Boundary-Scan Testing User Guide](#)

For more information about JTAG pin sharing

1.5.3.8. Ensure all devices in the chain are connected properly

If your device is in a configuration chain, ensure all devices in the chain are connected properly and powered on.

1.5.3.9. Determine if you need to turn on device-wide output enable

The Intel MAX 10 device supports an optional chip-wide output enable that allows you to override all tri-states on the device I/Os. When the `DEV_OE` pin is driven low, all I/O pins are tri-stated; when this pin is driven high, all pins behave as programmed.

To use the chip-wide output enable feature:

- Turn on **Enable device-wide output enable (DEV_OE)** under the **General** category of the **Device and Pin Options** dialog box in the Intel Quartus Prime software before compiling your design
- Ensure that the `DEV_OE` pin is driven to a valid logic level on your board
- Do not leave the `DEV_OE` pin floating

1.5.4. General I/O Pin Connections

Use the following checklist to plan your general I/O pin connections and to improve signal integrity.

1.5.4.1. Specify the state of unused I/O pins

- To reduce power dissipation, set clock pins and other unused I/O pins **As inputs tri-stated**. By default, the Intel Quartus Prime software set the input pins tri-stated with weak pull-up resistor enabled.
- To improve signal integrity, in the **Reserve all unused pins** option under the **Unused Pins** category of the **Device and Pin Options** dialog box of the Intel Quartus Prime software, set the unused pins **As output driving ground**. This setting reduces inductance by creating a shorter return path and reduces noise on the neighboring I/Os. However, do not use this approach if it results in many via paths that causes congestion for signals under the device.
- Carefully check the pin connections in the Pin-Out File (.pin) generated by the Intel Quartus Prime software when you compile your design. The .pin file specifies how you should connect the device pins. I/O pins specified as GND can be left unconnected or connected to ground for improved noise immunity. Do not connect RESERVED pins.

1.5.4.2. Refer to the Board Design Resource Center

If your design has high-speed signals the board design has a major impact on the signal integrity in the system.

Related Information

- [Board Design Resource Center](#)
For more information about signal integrity and board design
- [AN 528: PCB Dielectric Material Selection and Fiber Weave Effect on High-Speed Channel Routing.](#)
For more information about high-speed board stack-up
- [AN 529: Via Optimization Techniques for High-Speed Channel Designs.](#)
For more information about high-speed designs
- [AN 530: Optimizing Impedance Discontinuity Caused by Surface Mount Pads for High-Speed Channel Designs.](#)
For more information about signal routing layers
- [I/O Management, Board Development Support, and Signal Integrity Analysis Resource Center](#)
For more information on board-level signal integrity information related to the Quartus Prime software

1.5.4.3. Design VREF pins to be noise free

Voltage deviation on a VREF pin can affect the threshold sensitivity for inputs.

Related Information

[I/O Features and Pin Connections](#) on page 21

For more information about VREF pins and I/O standards

1.5.4.4. Refer to the Board Design Guideline Solution Center

Noise generated by SSN—when too many pins in close proximity change voltage levels at the same time—can reduce the noise margin and cause incorrect switching. For example, consider these board layout recommendations:

- Break out large bus signals on board layers close to the device to reduce cross talk.
- If possible, route traces orthogonally if two signal layers are next to each other, and use a separation of two to three times the trace width.

Related Information

- [Board Design Guidelines Solution Center](#)
For more board layout recommendations that can help with noise reduction
- [I/O Simultaneous Switching Noise](#) on page 28
For a list of recommendations for I/O and clock connections

1.5.4.5. Verify I/O termination and impedance matching

Voltage-referenced I/O standards require both a VREF and a termination voltage (V_{TT}). The reference voltage of the receiving device tracks the termination voltage of the transmitting device. Consider the following items:

- Each voltage-referenced I/O standard requires a unique termination setup. For example, a proper resistive signal termination scheme is critical in **SSTL-2** standards to produce a reliable DDR memory system with superior noise margin.
- Although single-ended, non-voltage-referenced I/O standards do not require termination, impedance matching is necessary to reduce reflections and improve signal integrity.
- Differential I/O standards typically require a termination resistor between the two signals at the receiver. The termination resistor must match the differential load impedance of the signal line.

The Intel MAX 10 on-chip series termination provides the convenience of no external components. You can also use external pull-up resistors to terminate the voltage-referenced I/O standards such as SSTL and HSTL.

Related Information

[Intel MAX 10 General Purpose I/O User Guide](#)

For a complete list of on-chip termination (OCT) support for each I/O standard

1.5.4.6. Perform full board routing simulation using IBIS models

To ensure that the I/O signaling meets receiver threshold levels on your board setup, perform full board routing simulation with third-party board-level simulation tools using an IBIS model.

To select the IBIS output in the Quartus Prime software, on the Assignments menu, click **Settings**. Navigate to the **Board-Level** page of the **EDA Tool Settings** category. Under the **Board-level signal integrity analysis** section, in the **Format** option, select **IBIS**.

Related Information

[Signal Integrity Analysis with Third-Party Tools](#)

1.5.4.7. Configure board trace models for Intel Quartus Prime advanced timing analysis

For a system to operate properly, signal integrity and board routing propagation delays must be taken into consideration. If you use an FPGA with high-speed interfaces in your board design, analyze the board level timing as part of the I/O and board planning.

Differential I/Os at the top left corner are located in the low speed region.

To generate a more accurate I/O delays and extra reports to gain better insights into the signal behavior at the system level, turn on **Enable Advanced I/O Timing** under the **Timing Analyzer** category in the **Settings** dialog box of your Intel Quartus Prime project. With this option turned on, the Timing Analyzer uses simulation results for the I/O buffer, package, and board trace model to generate the I/O delays.

You can use the advanced timing reports as a guide to make changes to the I/O assignments and board design to improve timing and signal integrity.

Related Information

- [Intel MAX 10 Device Pin-Outs](#)
For more information about the performance of these I/O pins
- [Intel MAX 10 FPGA Device Datasheet](#)
For more information about I/O pins

1.5.4.8. Review your pin connections

Intel provides schematic review worksheets based on the device Pin Connection Guidelines and other board-level pin connections literature that you need to consider when you finalize your schematics.

Related Information

[Intel MAX 10 Schematic Review Worksheet](#)

For more information on finding mistakes in schematics and adhering to design guidelines

1.6. I/O and Clock Planning

1.6.1. Early Pin Planning and I/O Assignment Analysis

In many design environments, FPGA designers want to plan top-level FPGA I/O pins early so that board designers can start developing the PCB design and layout.

1.6.1.1. Verify pin locations early in the FPGA place-and-route software

The FPGA I/O capabilities and board layout guidelines influence pin locations and other types of assignments. Starting FPGA pin planning early improves the confidence in early board layouts, reduces the chance of error, and improves the overall time-to-market.

1.6.1.2. Use the Intel Quartus Prime Pin Planner for I/O pin planning, assignments, and validation

Early in the design process, the system architect typically has information about the standard I/O interfaces (such as memory and bus interfaces), IP cores to be used in the design, and any other I/O-related assignments defined by system requirements.

You can use the Intel Quartus Prime Pin Planner for I/O pin assignment planning, assignment, and validation:

- The Intel Quartus Prime **Start I/O Assignment Analysis** command checks that pin locations and assignments are supported in the target FPGA architecture. Checks include reference voltage pin usage, pin location assignments, and mixing of I/O standards.
- You can use I/O Assignment Analysis to validate I/O-related assignments that you make or modify throughout the design process.
- The Create/Import IP core feature of the Pin Planner interfaces with the parameter editor, and enables you to create or import custom IP cores that use I/O interfaces.
- Enter PLL and LVDS blocks. Then, use the **Create Top-Level Design File** command to generate a top-level design netlist file.
- You can use the I/O analysis results to change pin assignments or IP parameters and repeat the checking process until the I/O interface meets your design requirements and passes the pin checks in the Intel Quartus Prime software.

After planning is complete, you can pass the preliminary pin location information to PCB designers.

After the design is complete, you can use the reports and messages generated by the Intel Quartus Prime Fitter for the final sign-off of the pin assignments.

Related Information

- [Intel MAX 10 General Purpose I/O User Guide](#)
For more information about the I/O restrictions guidelines
- [Managing Device I/O Pins](#)
For more information about I/O assignment and analysis

1.6.1.3. Check I/O restrictions related to ADC usage

The Intel Quartus Prime software uses physics-based rules to define the number of I/O pins allowed in a particular bank based on the I/O's drive strength. These rules are based on noise calculation to analyze accurately the impact of I/O placement on the ADC performance. If you use the ADC block in your design, Intel recommends that you follow the guidelines.

Related Information

- [Intel MAX 10 General Purpose I/O User Guide](#)
For more information about ADC I/O restriction guidelines

1.6.2. I/O Features and Pin Connections

Use the checklist in this section for guidelines related to I/O features, I/O signal types, I/O standards, I/O banks memory interfaces, pad placements, and special pin connections.

Related Information

- [Intel MAX 10 FPGA Device Family Pin Connection Guidelines](#)
- [Intel MAX 10 Device Pin-Outs](#)

1.6.2.1. Determine if your system requires single-ended I/O signaling

- Single-ended I/O signaling provides a simple rail-to-rail interface.
- The speed is limited by the large voltage swing and noise.
- Single-ended I/Os do not require termination, unless reflection in the system causes undesirable effects.

Related Information

[Intel MAX 10 General Purpose I/O User Guide](#)

For more information about the I/O restrictions guidelines

1.6.2.2. Determine if your system requires voltage-referenced signaling

- Voltage-referenced signaling reduces the effects of simultaneous switching outputs (SSO) from pins changing voltage levels at the same time (for example, external memory interface data and address buses).
- Voltage-referenced signaling provides an improved logic transition rate with a reduced voltage swing, and minimizes noise caused by reflection with a termination requirement.
- Additional termination components are required for the reference voltage source (V_{TT}).

Related Information

[Intel MAX 10 General Purpose I/O User Guide](#)

For more information about the VREF restrictions guidelines

1.6.2.3. Determine if your system requires differential signaling

- Differential signaling eliminates the interface performance barrier of single-ended and voltage-referenced signaling, with superior speed using an additional inverted closely-coupled data pair.
- Differential signaling avoids the requirement for a clean reference voltage. This is possible because of lower swing voltage and noise immunity with a common mode noise rejection capability.
- Considerations for this implementation include the requirements for a dedicated PLL to generate a sampling clock, and matched trace lengths to eliminate the phase difference between an inverted and non-inverted pair.
- Allow the software to assign locations for the negative pin in differential pin pairs. You only need to assign the positive pins.

Related Information

[Intel MAX 10 General Purpose I/O User Guide](#)

For more information about the differential I/O restrictions guidelines

1.6.2.4. Select a suitable signaling type and I/O standard for each I/O pin

Ensure that the appropriate I/O standard support is supported in the targeted I/O bank.

Place I/O pins that share voltage levels in the same I/O bank

- Certain I/O banks can support different I/O standards and voltage levels.
- You can assign I/O standards and make other I/O-related settings in the Pin Planner.
- Use the correct dedicated pin inputs for signals such as clocks and global control signals.

Related Information

- [Intel MAX 10 General Purpose I/O User Guide.](#)
- [Intel MAX 10 High-Speed LVDS SERDES User Guide.](#)

1.6.2.5. Verify that all output signals in each I/O bank are intended to drive out at the bank's assigned VCCIO voltage level

- The board must supply each bank with one V_{CCIO} voltage level for every V_{CCIO} pin in a bank.
- Each I/O bank is powered by the V_{CCIO} pins of that particular bank and is independent of the V_{CCIO} of other I/O banks.
- A single I/O bank supports output signals that are driving at the same voltage as the V_{CCIO} .
- An I/O bank can simultaneously support any number of input signals with different I/O standards.

Related Information

[Intel MAX 10 General Purpose I/O User Guide](#)

1.6.2.6. Verify that all voltage-referenced signals in each I/O bank are intended to use the bank's V_{REF} voltage (for devices that support V_{REF} pins)

- To accommodate voltage-referenced I/O standards, each bank supports either a shared V_{REF} pin or an individual V_{REF} pin. Set the V_{REF} pins to the correct voltage for the I/O standards in the bank. For more information, refer to the *Intel MAX 10 General Purpose I/O User Guide* or *Intel MAX 10 Device Pin-Outs*.
- The voltage-referenced I/O standards are not supported in the following I/O banks of these device packages:
 - All I/O banks of V36 package of 10M02 device
 - All I/O banks of V81 package of 10M08 device
 - Banks 1A and 1B of E144 package of 10M50 device
- Each I/O bank can only have a single V_{CCIO} voltage level and a single V_{REF} voltage level at a given time. If you use a shared V_{REF} pin, you must supply a common V_{CCIO} voltage to the I/O banks which share the same V_{REF} pin. If the V_{REF} pins are not used as voltage references, the pins can be used as regular I/O pins.

- An I/O bank, including single-ended or differential standards, can support voltage-referenced standards as long as all voltage-referenced standards use the same V_{REF} setting.
- For performance reasons, voltage-referenced input standards use their own V_{CCIO} level as the power source. You can place voltage-referenced input signals in a bank with a V_{CCIO} of 2.5 V or below.
- Voltage-referenced bidirectional and output signals must drive out at the V_{CCIO} voltage level of the I/O bank.

Related Information

- [Intel MAX 10 FPGA Device Family Pin Connection Guidelines](#)
Provides more information about the V_{REF} pins support for voltage-referenced I/O standards.
- [Intel MAX 10 Device Pin-Outs](#)
For more information about the V_{REF} pins.

1.6.2.7. Check the I/O bank support for LVDS features

Different I/O banks include different support for LVDS signaling. Some banks have lower speed performance. Allocate the pins according to your data rate requirement.

Related Information

[Intel MAX 10 High-Speed LVDS SERDES User Guide](#)

1.6.2.8. Verify the usage of the V_{REF} pins that are used as regular I/Os

V_{REF} pins have higher pin capacitance that results in a different I/O timing:

- Do not use these pins in a grouped interface such as a bus.
- Do not use these pins for high edge rate signals such as clocks.

1.6.2.9. Test pin connections with boundary-scan test

A boundary-scan test allows you to test pin connections at board level without using physical test probes while the device is operating normally. To perform the boundary-scan test, you must have the boundary-scan description language (BSDL) file of the device.

Use the BSDL files from www.altera.com to perform boundary-scan test on pre-configured Intel MAX 10 devices. For post-configured devices, you must modify the BSDL file according to the design.

Related Information

BSDL Support

For information on performing boundary-scan test after configuration, BSDL generation tool, and guidelines

1.6.2.10. Use the UNIPHY IP core for each memory interface, and follow connection guidelines

The self-calibrating UNIPHY IP core is optimized to take advantage of the Intel MAX 10 structure. The UNIPHY IP core allows you to set external memory interface features and helps set up the physical interface (PHY) best suited for your system. When you use the Intel memory controller IP core functions, the UNIPHY IP core is instantiated automatically.

If you design multiple memory interfaces into the device using Intel FPGA IP, generate a unique interface for each instance to ensure good results instead of designing it once and instantiating it multiple times.

Related Information

[Planning Pin and FPGA Resources](#)

1.6.2.11. Use dedicated DQ/DQS pins and DQ groups for memory interfaces

The data strobe DQS and data DQ pin locations are fixed in Intel MAX 10 devices. Before you design your device pin-out, refer to the memory interface guidelines for details and important restrictions related to the connections for these and other memory related signals.

Related Information

- [Volume 2: Design Guidelines](#)
For more information about external memory guidelines
- [Intel MAX 10 External Memory Interface User Guide](#)
For more information about Intel MAX 10 external memory interfaces
- [External Memory Interface \(EMIF\) Spec Estimator](#)
For more information about the memory spec estimator

1.6.2.12. Make dual-purpose pin settings and check for any restrictions when using these pins as regular I/O

You can use dual-purpose configuration pins as general I/Os after device configuration is complete. Select the desired setting for each of the dual-purpose pins on the **Dual-Purpose Pins** category of the **Device and Pin Options** dialog box. Depending on the configuration scheme, these pins can be reserved as regular I/O pins, as inputs that are tri-stated, as outputs that drive ground, or as outputs that drive an unspecified signal.

For configuration pins that used as general purpose I/Os, take note of the limitations of the pins when operating in user mode.

You can also use dedicated clock inputs, which drive the GCLK networks, as general purpose input pins if they are not used as clock pins. If you use the clock inputs as general inputs, the I/O registers use arithmetic logic module (ALM)-based registers because the clock input pins do not include dedicated I/O registers.

The device-wide reset and clear pins are available as design I/Os if they are not enabled.

Related Information

- [Intel MAX 10 General Purpose I/O User Guide](#)

- [Intel MAX 10 FPGA Configuration User Guide](#)
- [Determine if you need to turn on device-wide output enable](#) on page 17
- [Enable the chip-wide reset to clear all registers if required](#) on page 31

1.6.2.13. Review available device I/O features that can help I/O interfaces

Check the available I/O features and consider the following guidelines:

- Programmable current strength—ensure that the output buffer current strength is sufficiently high, but does not cause excessive overshoot or undershoot that violates voltage threshold parameters for the I/O standard. Intel recommends performing an IBIS or SPICE simulations to determine the right current strength setting for your specific application.
- Programmable slew rate—confirm that your interface meets its performance requirements if you use slower slew rates. Intel recommends performing IBIS or SPICE simulations to determine the right slew rate setting for your specific application.
- Programmable input/output element (IOE) delays—helps read and time margins by minimizing the uncertainties between signals in the bus. For delay specifications, refer to the relevant device datasheet.
- Open-drain output—if configured as an open-drain, the logic value of the output is either high-Z or 0. This feature is used in system-level control signals that can be asserted by multiple devices in the system. Typically, an external pull-up resistor is required to provide logic high.
- Bus hold—If the bus-hold feature is enabled, you cannot use the programmable pull-up option. Disable the bus-hold feature if the I/O pin is configured for differential signals. For the specific sustaining current driven through this resistor and the overdrive current used to identify the next driven input and level for each V_{CCIO} voltage, refer to the relevant device datasheet.
- Programmable pull-up resistors—weakly holds the I/O to the V_{CCIO} level when in user mode. This feature can be used with the open-drain output to eliminate the need for an external pull-up resistor. If the programmable pull-up option is enabled, you cannot use the bus-hold feature.
- Programmable pre-emphasis—increases the amplitude of the high frequency component of the output signal, and thus helps to compensate for the frequency-dependent attenuation along the transmission line.

Related Information

[Intel MAX 10 General Purpose I/O User Guide](#)

1.6.2.14. Consider OCT features to save board space and verify that the required termination scheme is supported for all pin locations

Driver-impedance matching provides the I/O driver with controlled output impedance that closely matches the impedance of the transmission line to significantly reduce reflections. OCT maintains signal quality, saves board space, and reduces external component costs.

- OCT R_S are supported in the same I/O bank for different I/O standards if they use the same V_{CCIO} supply voltage
- Each I/O in an I/O bank can be independently configured to support OCT R_S or programmable current strength
- You cannot configure both OCT R_S and programmable current strength or slew rate control for the same I/O buffer

Related Information

- [Intel MAX 10 General Purpose I/O User Guide](#)
For more information about the support and implementation of OCT
- [Intel MAX 10 High-Speed LVDS SERDES User Guide](#)
For more information about high-speed design

1.6.3. Clock Planning

The first stage in planning your clocking scheme is to determine your system clock requirements:

- Understand your device's available clock resources and correspondingly plan the design clocking scheme. Consider your requirements for timing performance, and how much logic is driven by a particular clock.
- Based on your system requirements, define the required clock frequencies for your FPGA design and the input frequencies available to the FPGA. Use these specifications to determine your PLL scheme.
- Use the Intel Quartus Prime parameter editor to enter your settings in ALTPLL IP core, and check the results to verify whether particular features and input/output frequencies can be implemented in a particular PLL.

1.6.3.1. Use the device PLLs for clock management

Connect clock inputs to specific PLLs to drive specific low-skew routing networks. Analyze the global resource availability for each PLL and the PLL availability for each clock input pin. Use the following descriptions for the clock signals in your design:

- The GCLK networks can drive throughout the entire device, serving as low-skew clock sources for device logic.
- IOEs and internal logic can also drive GCLKs to create internally generated GCLKs and other high fan-out control signals; for example, synchronous or asynchronous clears and clock enables.
- PLLs cannot be driven by internally-generated GCLKs. The input clock to the PLL must come from dedicated clock input pins or from another pin/PLL-fed GCLK.

1.6.3.2. Ensure that you select the correct PLL feedback compensation mode

Intel MAX 10 PLLs support four different clock feedback modes.

Related Information

- [Intel MAX 10 Clocking and PLL User Guide](#)
For more information about clock feedback modes

1.6.3.3. Check that the PLL offers the required number of clock outputs and use dedicated clock output pins

You can connect clock outputs to dedicated clock output pins or clock networks.

Intel MAX 10 PLL only allows one clock output per PLL block. If your device have 4 PLLs, there are 4 clock outputs from the PLLs.

1.6.3.4. Use the clock control block for clock selection and power-down

Every GCLK network has its own clock control block. The control block provides the following features that you can use to select different clock input signals or power-down clock networks to reduce power consumption without using any combinational logic in your design:

- Clock source selection (with dynamic selection)
- GCLK multiplexing
- Clock power down (with static or dynamic clock enable or disable)

In Intel MAX 10 devices, the `clkena` signals are supported at the clock network level instead of at the PLL output counter level. This allows you to gate off the clock even when you are not using a PLL. You can also use the `clkena` signals to control the dedicated external clocks from the PLLs.

Related Information

[Intel MAX 10 Clocking and PLL User Guide](#)

1.6.3.5. Instantiate PLL with ADC

You can only use the PLL C0 counter to connect to the ADC reference clock pin.

1.6.4. I/O Simultaneous Switching Noise

SSN is a concern when too many I/Os (in close proximity) change voltage levels at the same time. Use the checklist in this section for recommendations to plan I/O and clock connections.

1.6.4.1. Consider the following recommendations to mitigate I/O simultaneous switching noise:

- Analyze the design for possible SSN problems.
- Reduce the number of pins that switch the voltage level at exactly the same time whenever possible.
- Use differential I/O standards and lower-voltage standards for high-switching I/Os.
- Use lower drive strengths for high-switching I/Os. The default drive strength setting might be higher than your design requires.
- Reduce the number of simultaneously switching output pins within each bank. Spread output pins across multiple banks if possible.
- Spread the switching I/Os evenly throughout the bank to reduce the number of aggressors in a given area to reduce SSN if bank usage is substantially below 100%.

- Separate simultaneously switching pins from input pins that are susceptible to SSN.
- Place important clock and asynchronous control signals near ground signals and away from large switching buses.
- Avoid using I/O pins one or two pins away from PLL power supply pins for high-switching or high-drive strength pins.
- Use staggered output delays to shift the output signals through time, or use adjustable slew rate settings.

Related Information

- [I/O Features and Pin Connections](#) on page 21
- [Signal and Power Integrity Support Center](#)

1.6.4.2. Check on the pin connection guidelines for the ADC pins

The Intel Quartus Prime software uses physics-based rules to define the number of I/O pins allowed in a particular bank based on the I/O's drive strength. These rules are based on noise calculation to analyze accurately the impact of I/O placement on the ADC performance. If you use the ADC block in your design, Intel recommends that you follow the guidelines.

Related Information

[Intel MAX 10 FPGA Device Family Pin Connection Guidelines](#)

For more information about the filter type requirement at the ADC power pin, analog input pins, and VREF pin

1.7. Design Entry

In complex FPGA design development, design practices, coding styles, and IP core usage have an enormous impact on your device's timing performance, logic utilization, and system reliability. In addition, while planning and creating the design, plan for a hierarchical or team-based design to improve design productivity.

1.7.1. Use synchronous design practices

In a synchronous design, a clock signal triggers all events. When all of the registers' timing requirements are met, a synchronous design behaves in a predictable and reliable manner for all process, voltage, and temperature (PVT) conditions. You can easily target synchronous designs to different device families or speed grades.

1.7.2. Consider the following recommendations to avoid clock signals problems:

- Use dedicated clock pins and clock routing for best results—dedicated clock pins drive the clock network directly, ensuring lower skew than other I/O pins. Use the dedicated routing network to have a predictable delay with less skew for high fan-out signals. You can also use the clock pins and clock network to drive control signals like asynchronous reset.
- For clock inversion, multiplication, and division use the device PLLs.
- For clock multiplexing and gating, use the dedicated clock control block or PLL clock switchover feature instead of combinational logic.
- If you must use internally generated clock signals, register the output of any combinational logic used as a clock signal to reduce glitches. For example, if you divide a clock using combinational logic, clock the final stage with the clock signal that was used to clock the divider circuit.
- In multi-clock designs, ensure that signals crossing clock domains are properly synchronized using the synchronizer, a handshake mechanism, or a FIFO.

Related Information

[Intel MAX 10 Clocking and PLL User Guide](#)

For more information about clock networks

1.7.3. Use IP cores with the parameter editor

Instead of coding your own logic, save your design time by using Intel FPGA IP cores—a library of parameterized modules and device-specific IP cores. The IP cores are optimized for Intel FPGA device architectures and can offer more efficient logic synthesis and device implementation.

To ensure that you set all ports and parameters correctly, use the Intel Quartus Prime parameter editor to build or change IP cores parameters.

For detailed information about a specific IP core, refer to the respective Intel MAX 10 user guides.

1.7.4. Review the information on dynamic reconfiguration feature

The Intel MAX 10 devices support dynamic reconfiguration—dynamically change the PMA settings or protocols of a channel affecting data transfer on adjacent channels.

1.7.5. Consider the Intel's recommended coding styles to achieve optimal synthesis results

HDL coding styles can have a significant impact on the quality of results for programmable logic designs. For example, when designing memory and digital system processing (DSP) functions, understanding the device architecture helps you to take advantage of the dedicated logic block sizes and configurations.

- You can use the HDL templates provided in the Intel Quartus Prime software as examples for your reference. To access the templates, right click the editing area in the Intel Quartus Prime text editor and click **Insert Template**.
- For additional tool-specific guidelines, refer to the documentation of your synthesis tool.

Related Information

[Recommended HDL Coding Styles](#)

For specific HDL coding examples and recommendations

1.7.6. Enable the chip-wide reset to clear all registers if required

Intel MAX 10 devices support an optional chip-wide reset that enables you to override all clears on all device registers, including the registers of the memory blocks (but not the memory contents).

- DEV_CLRn pin is driven low—all registers are cleared or reset to 0. The affected register behave as if they are preset to a high value when synthesis performs an optimization called NOT-gate-push back due to register control signals.
- DEV_CLRn pin is driven high—all registers behave as programmed.

To enable chip-wide reset, before compiling your design, turn on **Enable device-wide reset (DEV_CLRn)** under the **Options** list of the **General** category in the **Device and Pin Options** dialog box of the Quartus Prime software.

1.7.7. Use device architecture-specific register control signals

Each Intel MAX 10 logic array block (LAB) contains dedicated logic for driving register control signals to its ALMs. It is important that the control signals use the dedicated control signals in the device architecture. In some cases, you may be required to limit the number of different control signals in your design.

Related Information

[Intel MAX 10 FPGA Device Architecture](#)

For more information about LAB and ALM architecture

1.7.8. Review recommended reset architecture

- If the clock signal is not available when reset is asserted, an asynchronous reset is typically used to reset the logic.
- The recommended reset architecture allows the reset signal to be asserted asynchronously and deasserted synchronously.
- The source of the reset signal is connected to the asynchronous port of the registers, which can be directly connected to global routing resources.
- The synchronous deassertion allows all state machines and registers to start at the same time.
- Synchronous deassertion avoids an asynchronous reset signal from being released at, or near, the active clock edge of a flipflop that can cause the output of the flipflop to go to a metastable unknown state.

Related Information

www.sunburst-design.com/papers

For more information about good reset design

1.7.9. Review the synthesis options available in your synthesis tool

If you force a particular power-up condition for your design, use the synthesis options available in your synthesis tool:

- By default, the Intel Quartus Prime software Integrated Synthesis turns on the **Power-Up Don't Care** logic option that assumes your design does not depend on the power-up state of the device architecture. Other synthesis tools might use similar assumptions.
- Designers typically use an explicit reset signal for the design that forces all registers into their appropriate values after reset but not necessarily at power-up. You can create your design with asynchronous reset that allows you to power up the design safely with the reset active, regardless of the power-up conditions of the device.
- Some synthesis tools can also read the default or initial values for registered signals in your source code and implement the behavior in the device. For example, the Intel Quartus Prime software Integrated Synthesis converts HDL default and initial values for registered signals into **Power-Up Level** settings. The synthesized behavior matches the power-up conditions of the HDL code during a functional simulation.
- Registers in the device core always power up to a low (0) logic level in the physical device architecture. If you specify a high power-up level or a non-zero reset value (preset signal), synthesis tools typically use the clear signals available on the registers and perform the NOT-gate push back optimization technique. If you assign a high power-up level to a register that is reset low, or assign a low power-up value to a register that is preset high, synthesis tools cannot use the NOT-gate push back optimization technique and might ignore the power-up conditions.

Related Information

[Intel Quartus Prime Integrated Synthesis](#)

For more information about the Power-Up Level settings and the altera_attribute assignment that sets the power-up state

1.7.10. Consider resources available for register power-up and control signals

To implement a reset and preset signal on the same register, synthesis tools emulate the controls with logic and latches that can be prone to glitches because of the different delays between the different paths to the register. In addition, the power-up value is undefined for these registers.

Related Information

[Recommended HDL Coding Styles](#)

For more information about reset logic and power up conditions

1.7.11. Consider Intel's recommendations for creating design partitions

Partitioning a design for an FPGA requires planning to ensure optimal results when the partitions are integrated and ensures that each partition is well placed, relative to other partitions in the device.

Follow Intel's recommendations for creating design partitions to improve the overall quality of results. For example, registering partition I/O boundaries keeps critical timing paths inside one partition that can be optimized independently. Plan your source code so that each design block is defined in a separate file. The software can automatically detect changes to each block separately.

Use hierarchy in your design to provide more flexibility when partitioning. Keep your design logic in the leaves of the hierarchy trees; that is, the top level of the hierarchy should have very little logic, and the lower-level design blocks contain the logic.

Related Information

[Best Practices for Incremental Compilation Partitions and Floorplan Assignments](#)
For guidelines to help you create design partitions

1.7.12. Perform timing budgeting and resource balancing between partitions

If your design is created in multiple projects, it is important that the system architect provide guidance to designers of lower-level blocks to ensure that each partition uses the appropriate device resources:

- Because the designs are developed independently, each lower-level designer has no information about the overall design or how their partition connects with other partitions, which can lead to problems during system integration.
- The top-level project information, including pin locations, physical constraints, and timing requirements, should be communicated to the designers of lower-level partitions before they start their design.
- The system architect can plan design partitions at the top level and use the Intel Quartus Prime software **Generate Bottom-Up Design Partition Scripts** option on the Project menu to automate the process of transferring top-level project information to lower-level modules.

1.7.13. Create a design floorplan for incremental compilation partitions

- A design floorplan avoids conflicts between design partitions and ensure that each partition is well-placed relative to other partitions. When you create different location assignments for each partition, no location conflicts occur.
- A design floorplan helps avoid situations in which the Fitter is directed to place or replace a portion of the design in an area of the device where most resources have already been claimed.
- Floorplan assignments are recommended for timing-critical partitions in top-down flows. You can use the Intel Quartus Prime Chip Planner to create a design floorplan using Logic Lock (Standard) region assignments for each design partition.
- With a basic design framework for the top-level design, the floorplan editor enables you to view connections between regions, estimate physical timing delays on the chip, and move regions around the device floorplan.
- After you compiled the full design, you can also view logic placement and locate areas of routing congestion to improve the floorplan assignments.

Related Information

- [Best Practices for Incremental Compilation Partitions and Floorplan Assignments](#)
For more information and guidelines in creating a design floorplan and placement assignments in the floorplan
- [Analyzing and Optimizing the Design Floorplan with the Chip Planner](#)
For more information about the Floorplan Editor

1.8. Design Implementation

1.8.1. Synthesis and Compilation

1.8.1.1. Specify your synthesis tool and use correct supported version

The Intel Quartus Prime software includes integrated synthesis that fully supports Verilog HDL, VHDL, Intel hardware description language (AHDL), and schematic design entry. You can also use industry-leading third-party EDA synthesis tools to synthesize your Verilog HDL or VHDL design, and then compile the resulting output netlist file in the Intel Quartus Prime software:

- Specify a third-party synthesis tool in the New Project Wizard or the **EDA Tools Settings** page of the Settings dialog box to use the correct Library Mapping File (.lmf) for your synthesis netlist.
- Intel recommends that you use the most recent version of third-party synthesis tools because tool vendors are continuously adding new features, fixing tool issues, and enhancing performance for Intel devices.
- Different synthesis tools can give different results. If you want to select the best-performing tool for your application, you can experiment by synthesizing typical designs for your application and coding style, and comparing the results.
- Perform placement and routing in the Intel Quartus Prime software to get accurate timing analysis and logic utilization results.
- Your synthesis tool may offer the capability to create a Intel Quartus Prime project and pass constraints such as the EDA tool setting, device selection, and timing requirements that you specified in your synthesis project. You can use this capability to save time when setting up your Intel Quartus Prime project for placement and routing.

Related Information

- [Intel Quartus Prime Integrated Synthesis](#)
- [Synopsys Synplify Support](#)
- [Mentor Graphics Precision Synthesis Support](#)
- [Mentor Graphics LeonardoSpectrum Support](#)
- [Intel Quartus Prime Software Release Notes](#)
For information about the officially supported version of each synthesis tool in a Intel Quartus Prime software version

1.8.1.2. Review resource utilization reports after compilation

After compilation in the Intel Quartus Prime software, review the device resource utilization information:

- Use the information to determine whether the future addition of extra logic or other design changes introduce fitting difficulties.
- If your compilation results in a no-fit error, use the information to analyze fitting problems.
- To determine resource usage, refer to the **Flow Summary** section of the Compilation Report for a percentage representing the total logic utilization, which includes an estimation of resources that cannot be used due to existing connections or logic usage.
- For more detailed resource information, view the reports under **Resource Section** in the **Fitter** section of the Compilation Report. The Fitter **Resource Usage Summary** report breaks down the logic utilization information and indicates the number of fully and partially used ALMs, and provides other resource information including the number of bits in each type of memory block.

There are also reports that describe some of the optimizations that occurred during compilation. For example, if you use the Intel Quartus Prime Integrated Synthesis, the reports under the **Optimization Results** folder in the **Analysis & Synthesis** section provide information that includes registers that were removed during synthesis. Use this report to estimate device resource utilization for a partial design to ensure that registers were not removed due to missing connections with other parts of the design.

Low logic utilization does not mean the lowest possible ALM utilization. A design that is reported to be close to 100% may still have space for extra logic. The Fitter uses ALUTs in different ALMs, even when the logic can be placed within one ALM, so that it can achieve the best timing and routability results. Logic might be spread throughout the device when achieving these results. As the device fills up, the Fitter automatically searches for logic that can be placed together in one ALM.

1.8.1.3. Review all Intel Quartus Prime messages, especially warning or error messages

Each stage of the compilation flow generates messages, including informational notes, warnings, and critical warnings. Understand the significance of warning messages and make changes to the design or settings if required.

In the Intel Quartus Prime user interface, you can use the Message window tabs to look at only certain types of messages. You can suppress the messages if you have determined that your action is not required.

Related Information

[Managing Intel Quartus Prime Projects](#)

For more information about messages and message suppression

1.8.1.4. Consider using incremental compilation

Use the incremental compilation feature to preserve logic in unchanged parts of your design, preserve timing performance, and reach timing closure more efficiently. You can speed up design iteration time by an average of 60% when making changes to the design with the incremental compilation feature.

1.8.1.5. Ensure parallel compilation is enabled

The Intel Quartus Prime software can run some algorithms in parallel to take advantage of multiple processors and reduce compilation time when more than one processor is available to compile the design. Set the **Parallel compilation** option on the **Compilation Process Settings** page of the **Settings** dialog box, or change the default setting in the **Options** dialog box in the **Processing** page from the Tools menu.

1.8.1.6. Use the Compilation Time Advisor

The Compilation Time Advisor provides guidance in making settings that reduce your design compilation time. On the Tools menu, point to **Advisors** and click **Compilation Time Advisor**. Using some of these techniques to reduce compilation time can reduce the overall quality of results.

Related Information

[Area and Timing Optimization](#)

1.8.2. Timing Optimization and Analysis

Use the guidelines in the following checklist for analyzing your design timing and optimizing your timing performance.

1.8.2.1. Ensure timing constraints are complete and accurate

In an FPGA design flow, accurate timing constraints allow timing-driven synthesis software and place-and-route software to obtain optimal results. Timing constraints are critical to ensure designs meet their timing requirements, which represent actual design requirements that must be met for the device to operate correctly.

The Intel Quartus Prime software optimizes and analyzes your design using different timing models for each device speed grade, so you must perform timing analysis for the correct speed grade. The final programmed device might not operate as expected if the timing paths are not fully constrained, analyzed, and verified to meet requirements.

Related Information

[Timing Analysis Overview](#)

1.8.2.2. Review the Timing Analyzer reports after compilation

The Intel Quartus Prime software includes the Intel Quartus Prime Timing Analyzer, a powerful ASIC-style timing analysis tool that validates the timing performance of all logic in your design. It supports the industry standard Synopsys Design Constraints (SDC) format timing constraints, and has an easy-to-use GUI with interactive timing reports. It is ideal for constraining high-speed source-synchronous interfaces and clock multiplexing design structures.

The software also supports static timing analysis in the industry-standard Synopsys Primetime software. To generate the required timing netlist, specify the tool in the New Project Wizard or the **EDA Tools Settings** page of the **Settings** dialog box.

1.8.2.3. Ensure that the I/O timings are not violated when data is provided to the FPGA

A comprehensive static timing analysis includes analysis of register to register, I/O, and asynchronous reset paths. It is important to specify the frequencies and relationships for all clocks in your design.

Use input and output delay constraints to specify external device or board timing parameters. Specify accurate timing requirements for external interfacing components to reflect the exact system intent.

The Timing Analyzer performs static timing analysis on the entire system, using data required times, data arrival times, and clock arrival times to verify circuit performance and detect possible timing violations. It determines the timing relationships that must be met for the design to correctly function. You can use the `report_datasheet` command to generate a datasheet report that summarizes the I/O timing characteristics of the entire design.

1.8.2.4. Perform Early Timing Estimation before running a full compilation

If the timing analysis reports that your design requirements were not met, you must make changes to your design or settings and recompile the design to achieve timing closure. If your compilation results in no-fit messages, you must make changes to get successful placement and routing.

You can use the Early Timing Estimation feature in the Intel Quartus Prime software to estimate your design's timing results before the software performs full placement and routing. On the **Processing** menu, point to **Start** and click **Start Early Timing Estimate** to generate initial compilation results after you have run analysis and synthesis.

1.8.2.5. Consider the following recommendations for timing optimization and analysis assignment:

- Turn on **Optimize multi-corner timing** on the Fitter Settings page in the Settings dialog box.
- Use `create_clock` and `create_generated_clock` to specify the frequencies and relationships for all clocks in your design.
- Use `set_input_delay` and `set_output_delay` to specify the external device or board timing parameters
- Use `derive_pll_clocks` to create generated clocks for all PLL outputs, according to the settings in the PLL IP cores. Specify multicycle relationships for LVDS transmitters or receiver deserialization factors.
- Use `derive_clock_uncertainty` to automatically apply inter-clock, intra-clock, and I/O interface uncertainties.
- Use `check_timing` to generate a report on any problem with the design or applied constraints, including missing constraints
- Use the Intel Quartus Prime optimization features to achieve timing closure or improve the resource utilization.
- Use the Timing and Area Optimization Advisors to suggest optimization settings.

Related Information

[The Intel Quartus Prime Timing Analyzer](#)

For more guidelines about timing constraints

1.8.2.6. Perform functional simulation at the beginning of your design flow

Perform the simulation to check the design functionality or logical behavior of each design block. You do not have to fully compile your design; you can generate a functional simulation netlist that does not contain timing information.

1.8.2.7. Perform timing simulation to ensure your design works in targeted device

Timing simulation uses the timing netlist generated by the Timing Analyzer, which includes the delay of different device blocks and placement and routing information. You can perform timing simulation for the top-level design at the end of your design flow to ensure that your design works in the targeted device.

1.8.2.8. Specify your simulation tool and use correct supported version

- Intel provides the ModelSim - Intel FPGA Edition simulator Starter Edition and offers the higher-performance ModelSim - Intel FPGA Edition that enable you to take advantage of advanced testbench capabilities and other features.
- In addition, the Intel Quartus Prime EDA Netlist Writer can generate timing netlist files to support other third-party simulation tools such as Synopsys VCS*, Cadence NCSim, and Aldec Active-HDL.
- If you use a third-party simulation tool, use the software version that is supported with your Intel Quartus Prime software version.
- Specify your simulation tool in the **EDA Tools Settings** page of the **Settings** dialog box to generate the appropriate output simulation netlist. The software can also generate scripts to help you setup libraries in your tool with NativeLink integration.
- Use only the model libraries provided with your Intel Quartus Prime software version. Libraries may change between versions and this can cause a mismatch with your simulation netlist.
- To create a testbench in the Intel Quartus Prime software, on the **Processing** menu, point to **Start** and click **Start Testbench Template Writer**.

Related Information

- [Intel Quartus Prime Software Release Notes](#)
For information about the officially supported version of each simulation tool in a Intel Quartus Prime software version
- [Simulating Intel FPGA Designs](#)
- [Mentor Graphics ModelSim and QuestaSim Support](#)
- [Synopsys VCS and VCS MX Support](#)
- [Cadence Incisive Enterprise Simulator Support](#)
- [Aldec Active-HDL and Riviera-PRO* Support](#)

1.8.3. Formal Verification

Use the following guidelines if your design requires formal verification.

1.8.3.1. Determine if you require formal verification for your design

If formal verification is required for your design, it is easier to plan for limitations and restrictions in the beginning than to make changes later in the design flow.

1.8.3.2. Check for support and design limitations for formal verification

The Intel Quartus Prime software supports some formal verification flows. Using a formal verification flow can impact performance results because it requires that certain logic optimizations be turned off, such as register retiming, and forces hierarchy blocks to be preserved, which can restrict optimization.

Related Information

[Cadence Encounter Conformal Support](#)

1.8.3.3. Specify your formal verification tool and use correct supported version

Specify your formal verification tool in the **EDA Tools Settings** page of the **Settings** dialog box to generate the appropriate output netlist.

Related Information

[Intel Quartus Prime Software Release Notes](#)

For information about the officially supported version of each simulation tool in a Intel Quartus Prime software version

1.8.4. Power Analysis and Optimization

After compiling your design, analyze the power consumption and heat dissipation with the Intel Quartus Prime Power Analyzer to calculate the dynamic, static, and I/O thermal power consumption and ensure the design has not violated power supply and thermal budgets.

Power optimization in the Intel Quartus Prime software depends on accurate power analysis results. Use the following guidelines to ensure the software optimizes the power utilization correctly for the design's operating behavior and conditions.

1.8.4.1. Provide accurate typical signal activities to get accurate power analysis result

You need to provide accurate typical signal activities to Power Analyzer:

- Compile a design to derive the information about design resources, placement and routing, and I/O standards.
- Derive signal activity data (toggle rates and static probabilities) from simulation results or a user-defined default toggle rate and vectorless estimation. The signal activities used for analysis must be representative of the actual operating behavior.

For the most accurate power estimation, use gate-level simulation results with a Value Change Dump File (.vcd) output file from a third-party simulation tool. The simulation activity should include typical input vectors over a realistic time period and not the corner cases often used during functional verification. Use the recommended simulator settings, such as glitch filtering, to ensure good results.

1.8.4.2. Specify the correct operating conditions for power analysis

Specify the operating conditions, including the core voltage, device power characteristics, ambient and junction temperature, cooling solution, and the board thermal model.

In the Intel Quartus Prime software, select the appropriate settings on the **Operating Settings and Conditions** page in the **Settings** dialog box.

1.8.4.3. Analyze power consumption and heat dissipation in the Power Analyzer

In the Intel Quartus Prime software, on the Processing menu, click **Power Analyzer Tool**. The tool also provides a summary of the signal activities used for analysis and a confidence metric that reflects the overall quality of the data sources for signal activities.

The Power Analyzer report is a power estimate and is not a power specification. Always refer to the device datasheet for the power specification.

Related Information

Power Analysis

For more information about power analysis and recommendations for simulation settings for creating signal activity information

1.8.4.4. Review recommended design techniques and Intel Quartus Prime options to optimize power consumption

The Power Optimization Advisor provides specific power optimization advice and recommendations based on the current design project settings and assignments.

Related Information

Power Optimization

For information about design techniques to optimize power consumption

1.8.4.5. Consider using a faster speed grade device

If your design includes many critical timing paths that require the high-performance mode, you might be able to reduce power consumption by using a faster speed grade device if available. With a faster device, the software might be able to set more device tiles to use the low-power mode.

1.8.4.6. Optimize the clock power management

Clocks represent a significant portion of dynamic power consumption, because of their high switching activity and long paths. The Intel Quartus Prime software automatically optimizes clock routing power by enabling only the portions of a clock network that are required to feed downstream registers.

You can also use clock control blocks to dynamically enable or disable the clock network. When a clock network is powered down, all the logic fed by that clock network does not toggle, thereby reducing the overall power consumption of the device.

To reduce LAB-wide clock power consumption without disabling the entire clock tree, use the LAB-wide clock enable signal to gate the LAB wide clock. The Intel Quartus Prime software automatically promotes register-level clock enable signals to the LAB level.

Related Information

[Intel MAX 10 Clocking and PLL User Guide](#)

For more information about using clock control blocks

1.8.4.7. Reduce the number of memory clocking events

Reduce the number of memory clocking events to reduce memory power consumption. You can use clock gating or the clock enable signals in the memory ports.

1.8.4.8. Consider I/O power guidelines

- The dynamic power consumed in the I/O buffer is proportional to the total load capacitance—lower capacitance reduces power consumption.
- Dynamic power is proportional to the square of the voltage. Use lower voltage I/O standards to reduce dynamic power. Non-terminated I/O standards such as **LVTTTL** and **LVC MOS** have a rail-to-rail output swing equal to the V_{CCIO} supply voltage and consume little static power.
- Dynamic power is proportional to the output transition frequency. Use resistively-terminated I/O standards such as **SSTL** for high-frequency applications. The output load voltage swings by an amount smaller than the V_{CCIO} around a bias point. Because of this, the dynamic power is lower than for non-terminated I/O under similar conditions.
- Resistively-terminated I/O standards dissipate significant static power because current is constantly driven into the termination network. Use the lowest drive strength that meets your speed and waveform requirements to minimize static power when using resistively terminated I/O standards.
- The power used by external devices is not included in the Power Analyzer calculations. Ensure that you include the external devices power separately in your system power calculations.

1.8.4.9. Reduce design glitches through pipelining and retiming

A design that has many glitches consumes more power because of faster switching activity. Pipelining by inserting flip flops into long combinational paths can reduce design glitches.

However, if there are not many glitches in your design, pipelining may increase power consumption due to the addition of unnecessary registers.

1.8.4.10. Review the information on power-driven compilation and Power Optimization Advisor

The Intel Quartus Prime software offers power-driven compilation to fully optimize device power consumption. Power-driven compilation focuses on reducing your design's total power consumption using power-driven synthesis and power-driven place-and-route.

Related Information

[Power Optimization](#)

1.8.4.11. Reduce power consumption with architectural optimization

Use specific device architecture features to reduce power consumption.

For example, use the dedicated DSP block available in the Intel MAX 10 device in place of LEs to perform arithmetic-related functions; build large shift registers from RAM-based FIFO buffers instead of building the shift registers from the LE registers.

1.9. Document Revision History for Intel MAX 10 Device Design Guidelines

Document Version	Changes
2020.10.19	Updated VREF pin support for voltage-referenced I/O standards in the <i>Verify that all voltage-referenced signals in each I/O bank are intended to use the bank's VREF voltage (for devices that support VREF pins)</i> section.
2020.02.07	<ul style="list-style-type: none"> Updated the POR delay guideline in the <i>Design the board for power-up</i> section. Updated the following terms: <ul style="list-style-type: none"> Changed Qsys to Platform Designer (Standard). Changed TimeQuest Timing Analyzer to Timing Analyzer. Changed OpenCore Plus to Intel FPGA IP Evaluation Mode. Changed PowerPlay Early Power Estimator to Early Power Estimator. Changed LogicLock to Logic Lock (standard).

Date	Version	Changes
May 2017	2017.05.03	<ul style="list-style-type: none"> Rebranded as Intel.
December 2014	2014.12.15	<ul style="list-style-type: none"> Changed the following terms: <ul style="list-style-type: none"> Dual image to dual configuration image Dual image configuration to dual configuration Changed MAX 10 EMIF IP core to UNIPHY IP core.
September 2014	2014.09.22	Initial release.



Intel® MAX® 10 FPGA Device Architecture



Online Version



Send Feedback

**M10-
ARCHITECTURE**

ID: **683105**

Version: **2022.10.31**

Contents

1. Intel® MAX® 10 FPGA Device Architecture.....	3
1.1. Logic Array Block.....	4
1.1.1. LAB Interconnects.....	5
1.1.2. LAB Control Signals.....	6
1.1.3. Logic Elements.....	8
1.2. Embedded Memory.....	11
1.3. Embedded Multiplier.....	12
1.3.1. 18-Bit Multipliers.....	12
1.3.2. 9-Bit Multipliers.....	13
1.4. Clocking and PLL.....	14
1.4.1. Global Clock Networks.....	15
1.4.2. Internal Oscillator.....	16
1.4.3. PLL Block and Locations.....	16
1.5. General Purpose I/O.....	19
1.5.1. Intel MAX 10 I/O Banks Architecture.....	19
1.5.2. Intel MAX 10 I/O Banks Locations.....	19
1.6. High-Speed LVDS I/O.....	22
1.6.1. Intel MAX 10 High-Speed LVDS Circuitry.....	22
1.6.2. Intel MAX 10 High-Speed LVDS I/O Location.....	23
1.7. External Memory Interface.....	25
1.7.1. Intel MAX 10 I/O Banks for External Memory Interface.....	25
1.8. Analog to Digital Converter.....	26
1.8.1. ADC Block Locations.....	27
1.9. Configuration Schemes.....	30
1.9.1. JTAG Configuration.....	31
1.9.2. Internal Configuration.....	31
1.10. User Flash Memory.....	31
1.11. Power Management.....	32
1.11.1. Single-Supply Device.....	32
1.11.2. Dual-Supply Device.....	32
1.11.3. Power Management Controller Scheme.....	33
1.11.4. Hot Socketing.....	33
1.12. Document Revision History for Intel MAX 10 FPGA Device Architecture.....	33



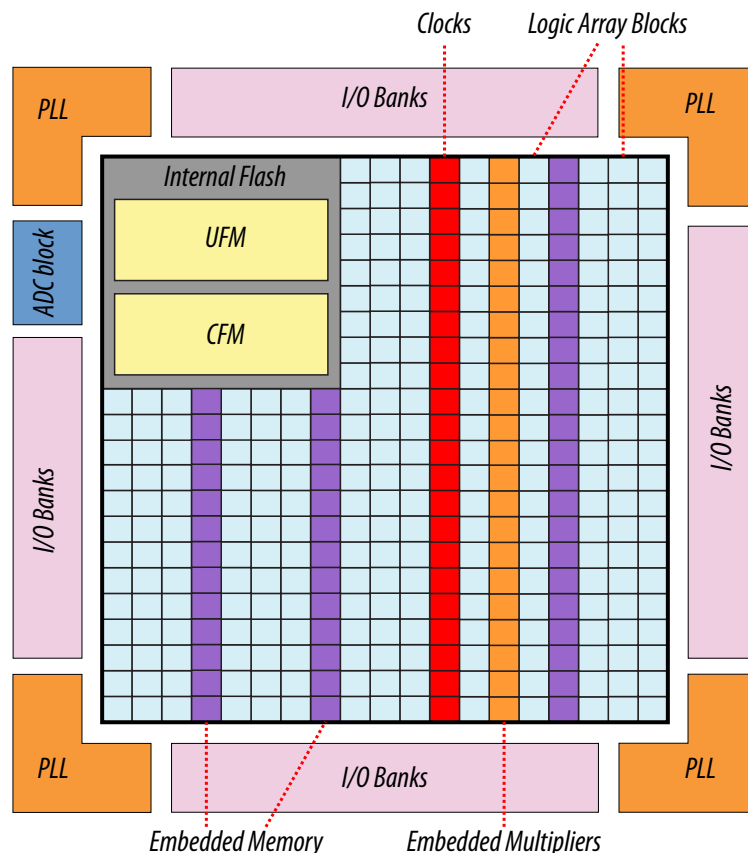
1. Intel® MAX® 10 FPGA Device Architecture

The Intel® MAX® 10 devices consist of the following:

- Logic array blocks (LABs)
- Analog-to-digital converter (ADC)
- User flash memory (UFM)
- Embedded multiplier blocks
- Embedded memory blocks (M9K)
- Clocks and phase-locked loops (PLL)
- General purpose I/O
- High-speed LVDS I/O
- External memory interfaces
- Configuration flash memory (CFM)

Figure 1. Typical Device Floorplan for Intel MAX 10 Devices

- The amount and location of each block varies in each Intel MAX 10 device.
- Certain Intel MAX 10 devices may not contain a specific block.



Related Information

- [Intel MAX 10 Device Datasheet](#)
Provides more information about specification and performance for Intel MAX 10 devices.
- [Intel MAX 10 FPGA Device Overview](#)
Provides more information about maximum resources in Intel MAX 10 devices

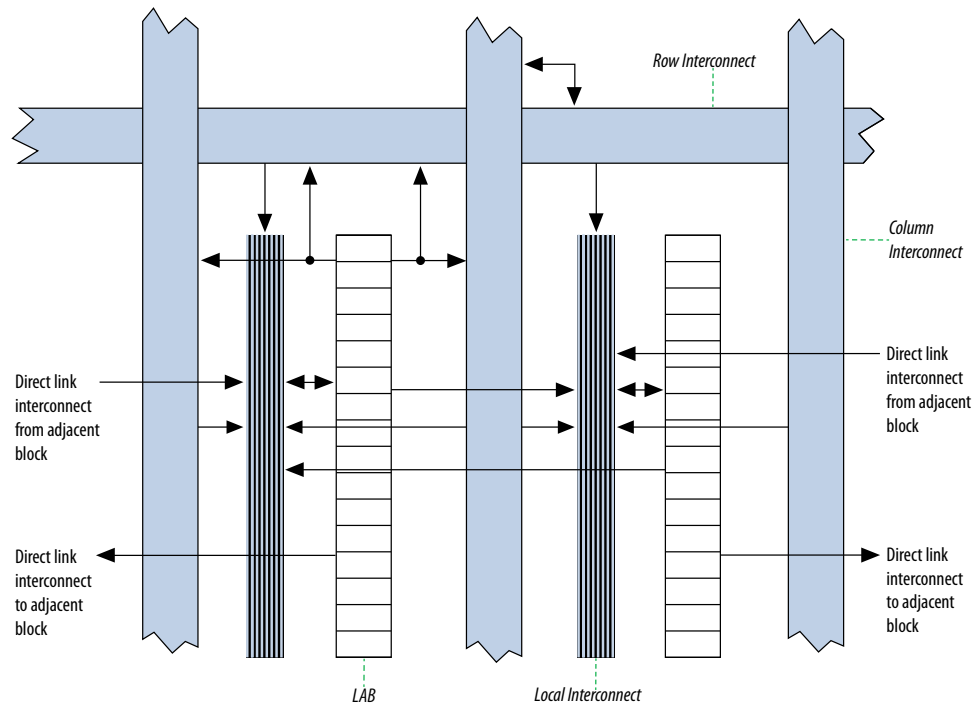
1.1. Logic Array Block

The LABs are configurable logic blocks that consist of a group of logic resources.

Each LAB consists of the following:

- 16 logic elements (LEs)—smallest logic unit in Intel MAX 10 devices
- LE carry chains—carry chains propagated serially through each LE within an LAB
- LAB control signals—dedicated logic for driving control signals to LEs within an LAB
- Local interconnect—transfers signals between LEs in the same LAB
- Register chains—transfers the output of one LE register to the adjacent LE register in an LAB

Figure 2. LAB Structure of Intel MAX 10 Devices



The Intel Quartus® Prime Compiler places associated logic in an LAB or adjacent LABs, allowing the use of local and register chain connections for performance and area efficiency.

1.1.1. LAB Interconnects

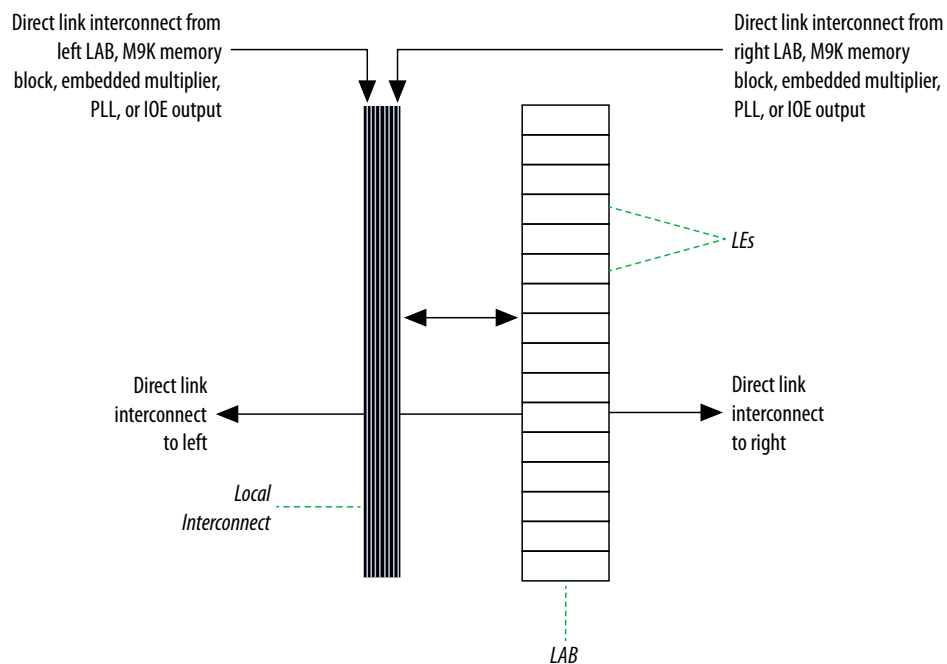
The LAB local interconnect is driven by column and row interconnects and LE outputs in the same LAB.

The direct link connection minimizes the use of row and column interconnects to provide higher performance and flexibility. The direct link connection enables the neighboring elements from left and right to drive the local interconnect of an LAB. The elements are:

- LABs
- PLLs
- M9K embedded memory blocks
- Embedded multipliers

Each LE can drive up to 48 LEs through local and direct link interconnects.

Figure 3. LAB Local and Direct Link Interconnects for Intel MAX 10 Devices



1.1.2. LAB Control Signals

Each LAB contains dedicated logic for driving control signals to its LEs.

The control signals include:

- Two clock signals
- Two clock enable signals
- Two asynchronous clear signals
- One synchronous clear signal
- One synchronous load signal

Figure 4. LAB-Wide Control Signals for Intel MAX 10 Devices

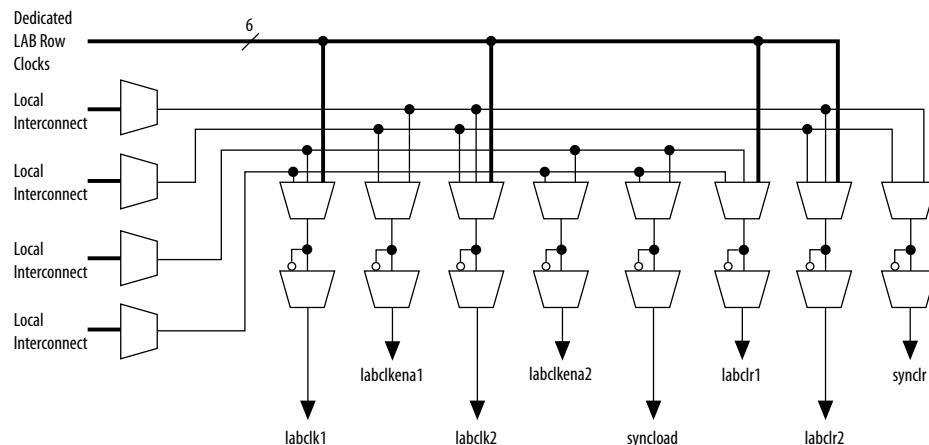


Table 1. Control Signal Descriptions for Intel MAX 10 Devices

Control Signal	Description
labclk1	<ul style="list-style-type: none"> Each LAB can use two clocks signals. The clock and clock enable signals of each LAB are linked. For example, any LE in a particular LAB using the labclk1 signal also uses the labclkena1 signal. If the LAB uses both the rising and falling edges of a clock, it also uses both LAB-wide clock signals. The LAB row clocks [5..0] and LAB local interconnect generate the LAB-wide clock signals. The MultiTrack interconnect inherent low skew allows clock and control signal distribution in addition to data distribution.
labclk2	
labclkena1	<ul style="list-style-type: none"> Each LAB can use two clock enable signals. The clock and clock enable signals of each LAB are linked. For example, any LE in a particular LAB using the labclk1 signal also uses the labclkena1 signal. Deasserting the clock enable signal turns off the LAB-wide clock signal.
labclkena2	
labclr1	Asynchronous clear signals: <ul style="list-style-type: none"> LAB-wide control signals that control the logic for the clear signal of the register. The LE directly supports an asynchronous clear function.
labclr2	
syncload	Synchronous load and synchronous clear signals: <ul style="list-style-type: none"> Can be used for implementing counters and other functions LAB-wide control signals that affect all registers in the LAB
syncclr	

You can use up to eight control signals at a time. Register packing and synchronous load cannot be used simultaneously.

Each LAB can have up to four non-global control signals. You can use additional LAB control signals as long as they are global signals.

An LAB-wide asynchronous load signal to control the logic for the preset signal of the register is not available. The register preset is achieved with a NOT gate push-back technique. Intel MAX 10 devices only support either a preset or asynchronous clear signal.

In addition to the clear port, Intel MAX 10 devices provide a chip-wide reset pin (DEV_CLRn) to reset all registers in the device. An option set before compilation in the Intel Quartus Prime software controls this pin. This chip-wide reset overrides all other control signals.

1.1.3. Logic Elements

LE is the smallest unit of logic in the Intel MAX 10 device family architecture. LEs are compact and provide advanced features with efficient logic usage.

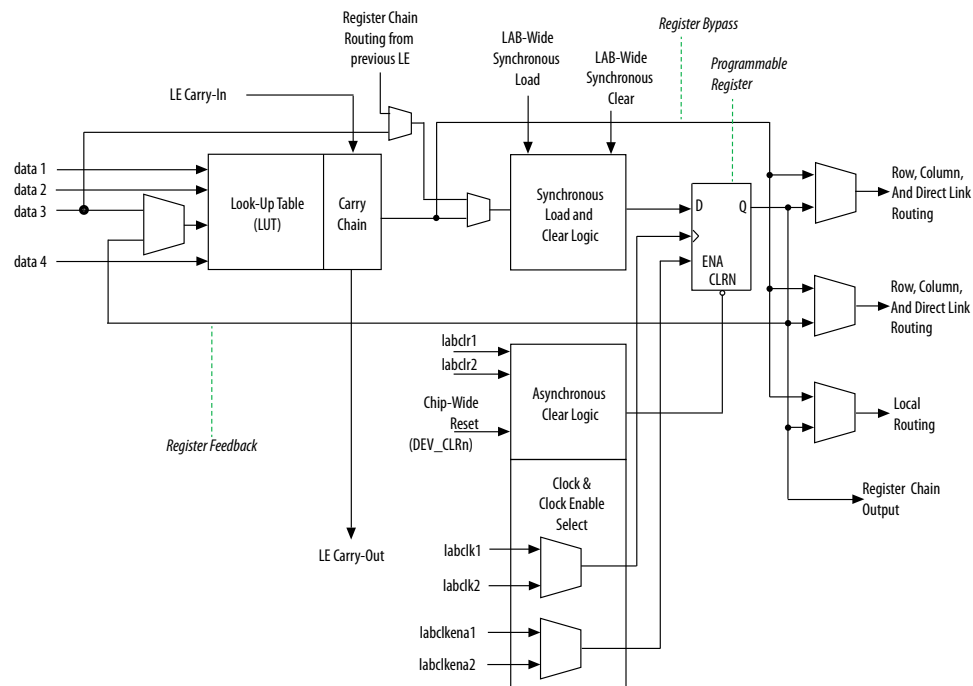
Each LE has the following features:

- A four-input look-up table (LUT) that can implement any function of four variables
- A programmable register
- A carry chain connection
- A register chain connection
- The ability to drive the following interconnects:
 - Local
 - Row
 - Column
 - Register chain
 - Direct link
- Register packing support
- Register feedback support

1.1.3.1. LE Features

LEs contain inputs, outputs, and registers to enable several features.

Figure 5. LE High-Level Block Diagram for Intel MAX 10 Devices.



LE Inputs

Each LE input is directed to different destinations to implement the desired logic function. In both the normal or arithmetic operating modes of the LE, there are six available inputs:

- Four data inputs from the LAB local interconnect
- One LE carry-in from the previous LE carry-chain
- One register chain connection

LE Outputs

Each LE has three general routing outputs:

- Two LE outputs drive the column or row and direct link routing connections
- One LE output drives the local interconnect resources

Intel MAX 10 devices support register packing. With register packing, the LUT or register output drives the three outputs independently. This feature improves device utilization by using the register and the LUT for unrelated functions.

The LAB-wide synchronous load control signal is not available if you use register packing.

Register Chain Output

Each LE has a register chain output that allows registers in the same LAB to cascade together. This feature speeds up connections between LABs and optimizes local interconnect resources:

- LUTs are used for combinational functions
- Registers are used for an unrelated shift register implementation

Programmable Register

You can configure the programmable register of each LE for D, T, JK, or SR flipflop operation. Each register has the following inputs:

- Clock—driven by signals that use the global clock network, general-purpose I/O pins, or internal logic
- Clear—driven by signals that use the global clock network, general-purpose I/O pins, or internal logic
- Clock enable—driven by the general-purpose I/O pins or internal logic

For combinational functions, the LUT output bypasses the register and drives directly to the LE outputs.

Register Feedback

The register feedback mode allows the register output to feed back into the LUT of the same LE. Register feedback ensures that the register is packed with its own fan-out LUT, providing another mechanism for improving fitting. The LE can also drive out registered and unregistered versions of the LUT output.

1.1.3.2. LE Operating Modes

The LEs in Intel MAX 10 devices operate in two modes.

- Normal mode
- Arithmetic mode

These operating modes use LE resources differently. Both LE modes have six available inputs and LAB-wide signals.

The Intel Quartus Prime software automatically chooses the appropriate mode for common functions, such as counters, adders, subtractors, and arithmetic functions, in conjunction with parameterized functions such as the library of parameterized modules (LPM) functions.

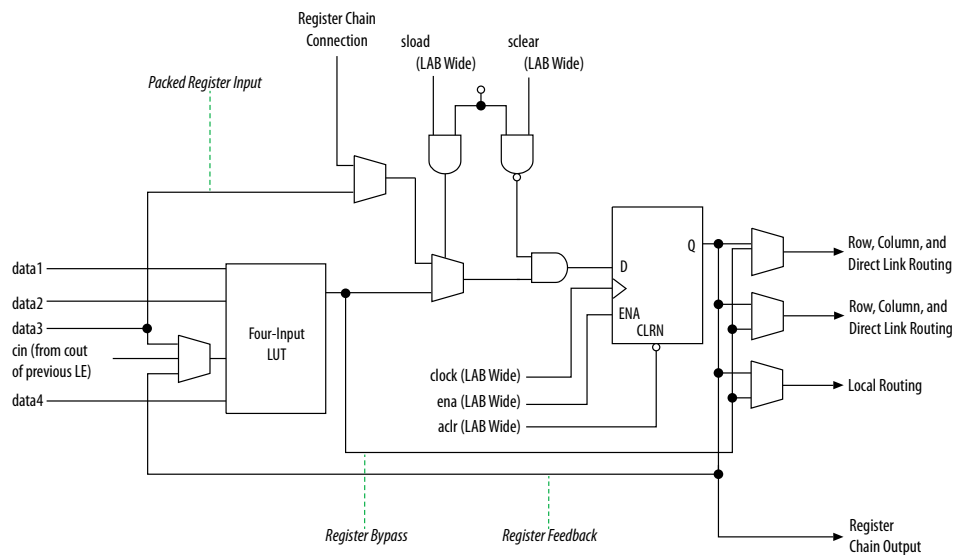
You can also create special-purpose functions that specify which LE operating mode to use for optimal performance.

1.1.3.2.1. Normal Mode

Normal mode is suitable for general logic applications and combinational functions.

In normal mode, four data inputs from the LAB local interconnect are inputs to a four-input LUT. The Intel Quartus Prime Compiler automatically selects the carry-in (cin) or the data3 signal as one of the inputs to the LUT. LEs in normal mode support packed registers and register feedback.

Figure 6. LE Operating in Normal Mode for Intel MAX 10 devices

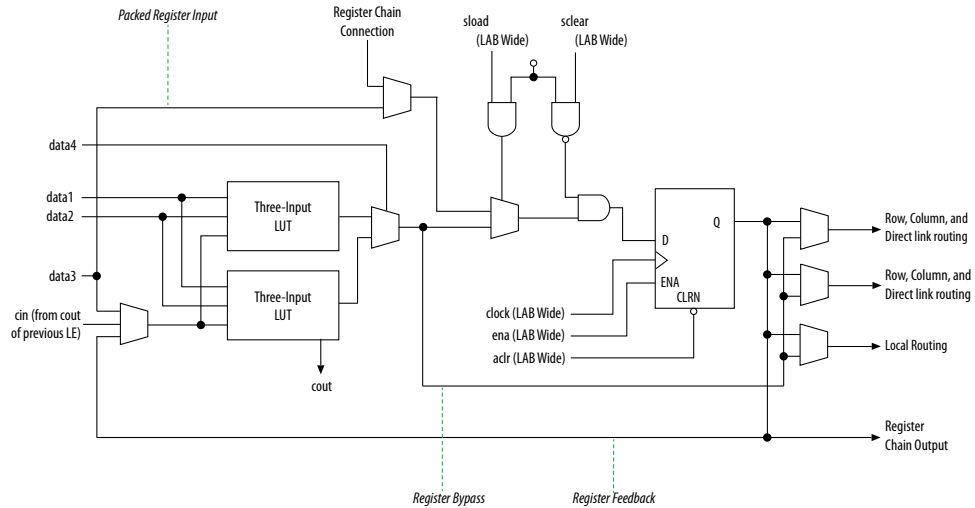


1.1.3.2.2. Arithmetic Mode

Arithmetic mode is ideal for implementing adders, counters, accumulators, and comparators.

The LE in arithmetic mode implements a two-bit full adder and basic carry chain. LEs in arithmetic mode can drive out registered and unregistered versions of the LUT output. Register feedback and register packing are supported when LEs are used in arithmetic mode.

Figure 7. LE Operating in Arithmetic Mode for Intel MAX 10 devices



Carry Chain

The Intel Quartus Prime Compiler automatically creates carry chain logic during design processing. You can also manually create the carry chain logic during design entry. Parameterized functions, such as LPM functions, automatically take advantage of carry chains for the appropriate functions. The Intel Quartus Prime Compiler creates carry chains longer than 16 LEs by automatically linking LABs in the same column.

To enhanced fitting, a long carry chain runs vertically, which allows fast horizontal connections to M9K memory blocks or embedded multipliers through direct link interconnects. For example, if a design has a long carry chain in an LAB column next to a column of M9K memory blocks, any LE output can feed an adjacent M9K memory block through the direct link interconnect.

If the carry chains run horizontally, any LAB which is not next to the column of M9K memory blocks uses other row or column interconnects to drive a M9K memory block.

A carry chain continues as far as a full column.

1.2. Embedded Memory

The Intel MAX 10 embedded memory block is optimized for applications such as high throughput packet processing, embedded processor program, and embedded data storage.

The Intel MAX 10 embedded memory structure consists of 9,216-bit (including parity bits) blocks. You can use each M9K block in different widths and configuration to provide various memory functions such as RAM, ROM, shift registers, and FIFO.

Intel MAX 10 embedded memory supports the following general features:

- 8,192 memory bits per block (9,216 bits per block including parity).
- Independent read-enable (*rden*) and write-enable (*wren*) signals for each port.
- Packed mode in which the M9K memory block is split into two 4.5 K single-port RAMs.

- Variable port configurations.
- Single-port and simple dual-port modes support for all port widths.
- True dual-port (one read and one write, two reads, or two writes) operation.
- Byte enables for data input masking during writes.
- Two clock-enable control signals for each port (port A and port B).
- Initialization file to preload memory content in RAM and ROM modes.

Related Information

[Intel MAX 10 Embedded Memory User Guide](#)

1.3. Embedded Multiplier

You can use an embedded multiplier block in one of two operational modes, depending on the application needs:

- One 18-bit x 18-bit multiplier
- Up to two 9-bit x 9-bit independent multipliers

You can also use embedded multipliers of the Intel MAX 10 devices to implement multiplier adder and multiplier accumulator functions. The multiplier portion of the function is implemented using embedded multipliers. The adder or accumulator function is implemented in logic elements (LEs).

Related Information

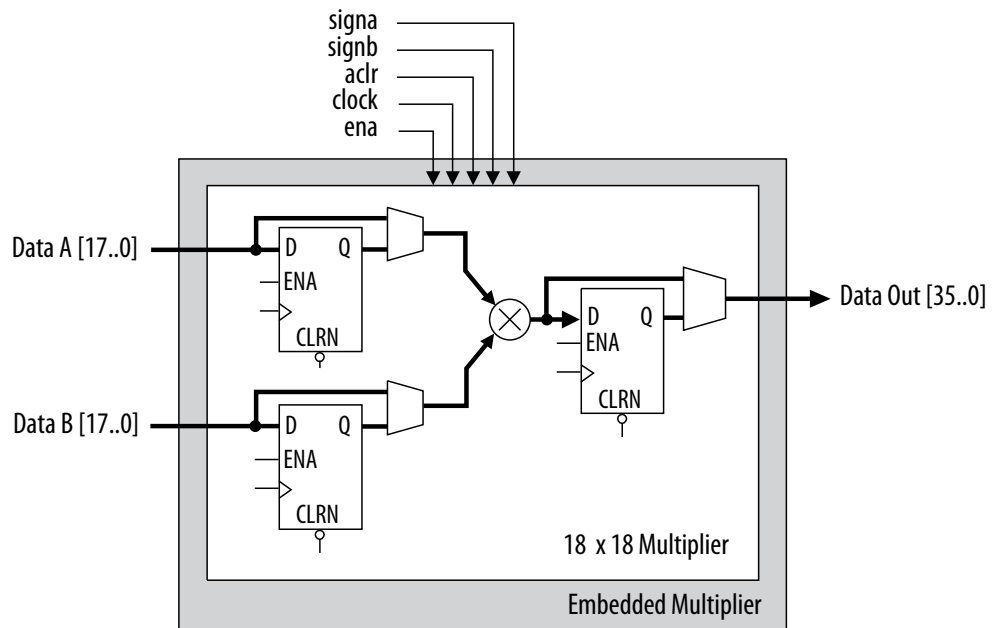
[Intel MAX 10 Embedded Multiplier User Guide](#)

1.3.1. 18-Bit Multipliers

You can configure each embedded multiplier to support a single 18 x 18 multiplier for input widths of 10 to 18 bits.

The following figure shows the embedded multiplier configured to support an 18-bit multiplier.

Figure 8. 18-Bit Multiplier Mode



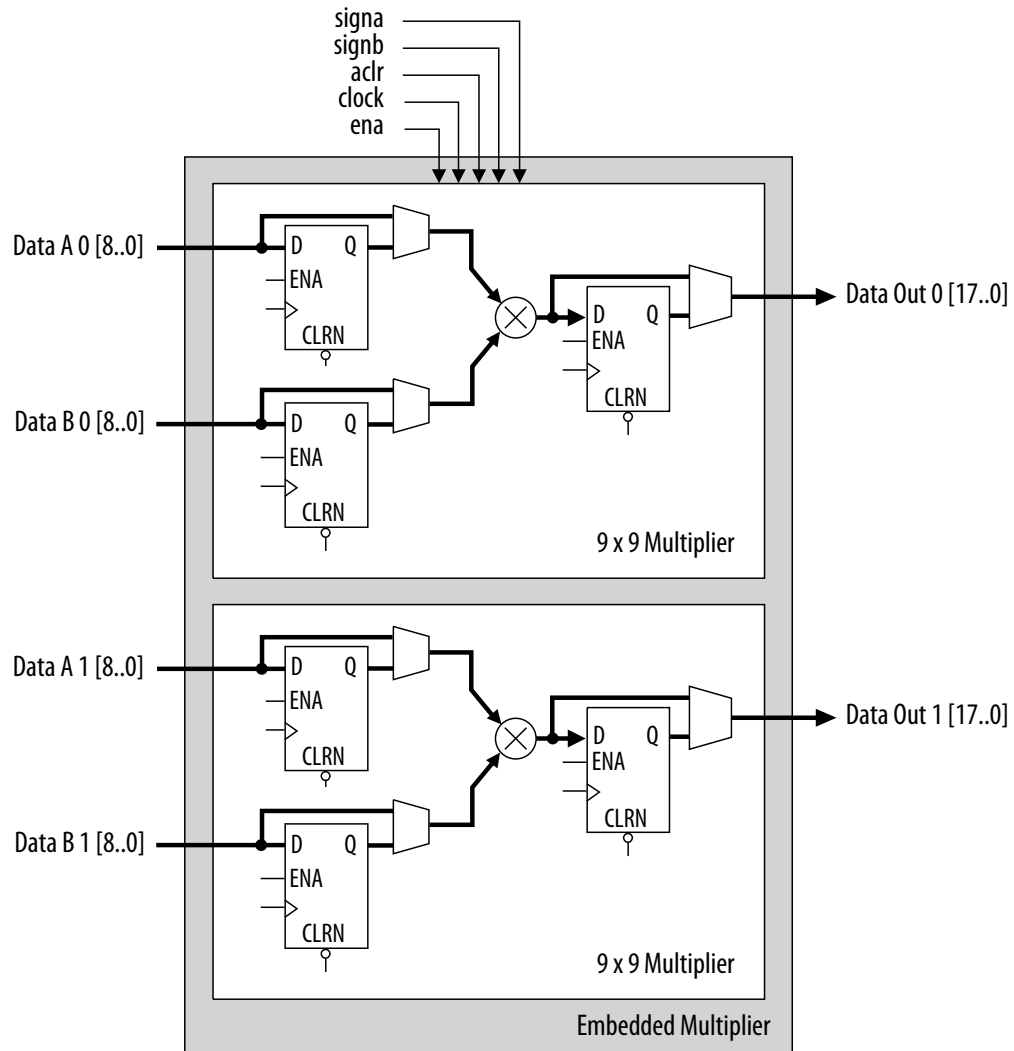
All 18-bit multiplier inputs and results are independently sent through registers. The multiplier inputs can accept signed integers, unsigned integers, or a combination of both. Also, you can dynamically change the `signa` and `signb` signals and send these signals through dedicated input registers.

1.3.2. 9-Bit Multipliers

You can configure each embedded multiplier to support two 9×9 independent multipliers for input widths of up to 9 bits.

The following figure shows the embedded multiplier configured to support two 9-bit multipliers.

Figure 9. 9-Bit Multiplier Mode



All 9-bit multiplier inputs and results are independently sent through registers. The multiplier inputs can accept signed integers, unsigned integers, or a combination of both.

Each embedded multiplier block has only one *signa* and one *signb* signal to control the sign representation of the input data to the block. If the embedded multiplier block has two 9×9 multipliers the following applies:

- The Data A input of both multipliers share the same *signa* signal
- The Data B input of both multipliers share the same *signb* signal

1.4. Clocking and PLL

Intel MAX 10 devices support global clock network (GCLK) and phase-locked loop (PLL).

Clock networks provide clock sources for the core. You can use clock networks in high fan out global signal network such as reset and clear.

PLLs provide robust clock management and synthesis for device clock management, external system clock management, and I/O interface clocking.

Related Information

[Intel MAX 10 Clock Networks and PLLs User Guide](#)

1.4.1. Global Clock Networks

GCLKs drive throughout the entire device, feeding all device quadrants. All resources in the device, such as the I/O elements, logic array blocks (LABs), dedicated multiplier blocks, and M9K memory blocks can use GCLKs as clock sources. Use these clock network resources for control signals, such as clock enables and clears fed by an external pin. Internal logic can also drive GCLKs for internally-generated GCLKs and asynchronous clears, clock enables, or other control signals with high fan-out.

Figure 10. GCLK Network Sources for 10M02, 10M04, and 10M08 Devices

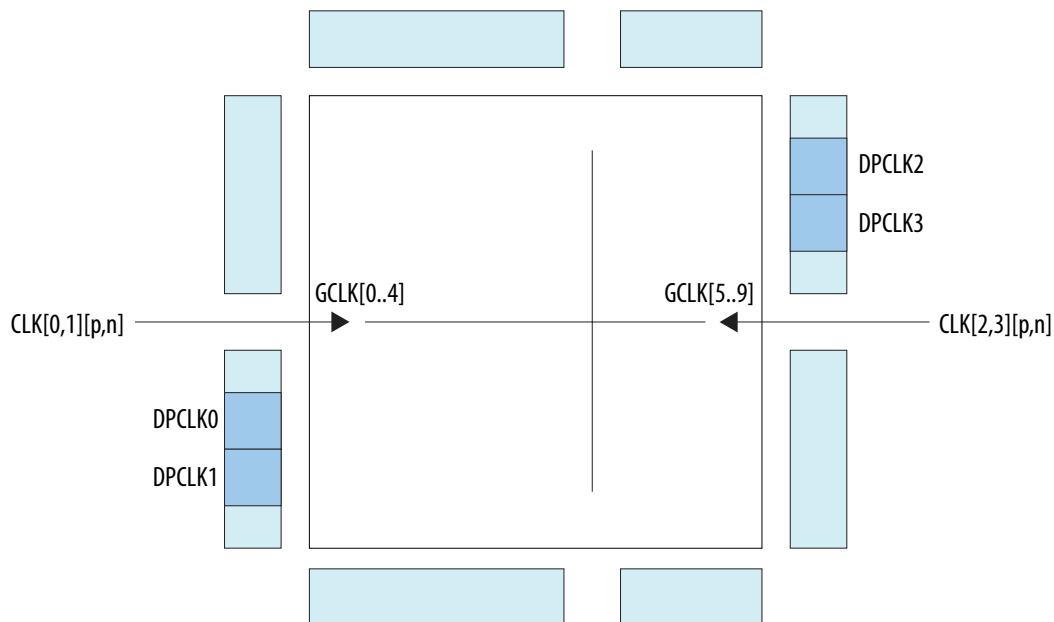
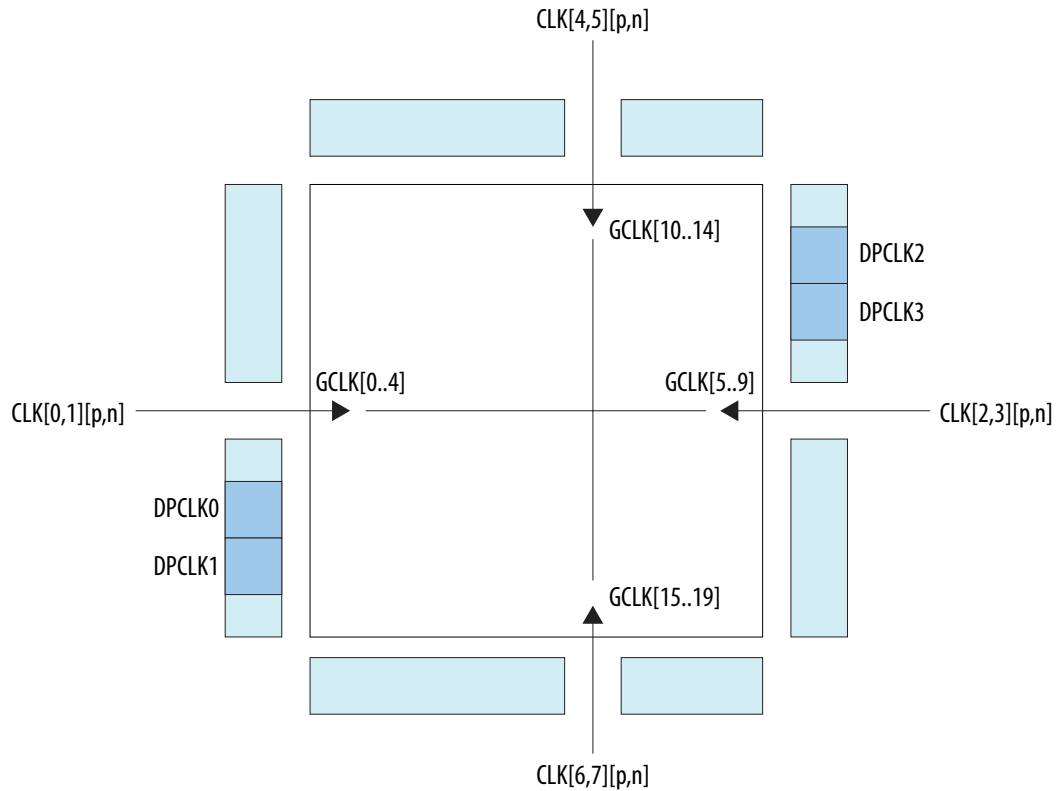


Figure 11. GCLK Network Sources for 10M16, 10M25, 10M40, and 10M50 Devices



1.4.2. Internal Oscillator

Intel MAX 10 devices have built-in internal ring oscillator with clock multiplexers and dividers. The internal ring oscillator operates up to 232 MHz which is not accessible. This operating frequency further divides down to slower frequencies.

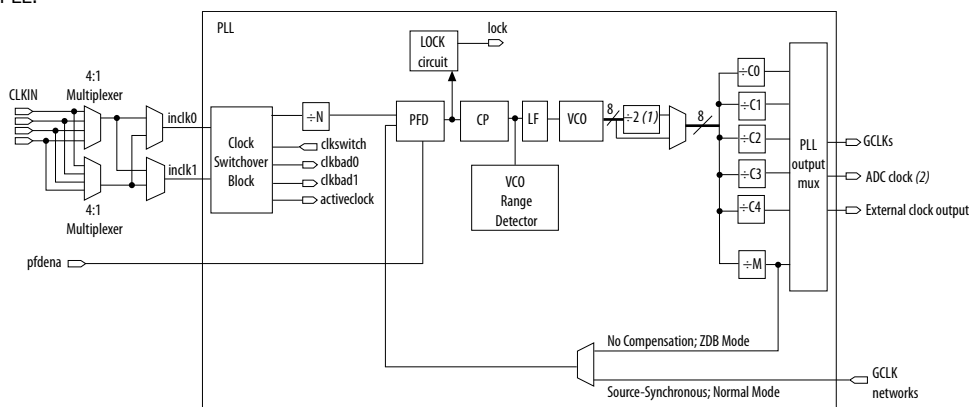
When the `oscena` input signal is asserted, the oscillator is enabled and the output can be routed to the logic array through the `clkout` output signal. When the `oscena` signal is set low, the `clkout` signal is constant high. You can analyze this delay using the Timing Analyzer.

1.4.3. PLL Block and Locations

The main purpose of a PLL is to synchronize the phase and frequency of the voltage-controlled oscillator (VCO) to an input reference clock.

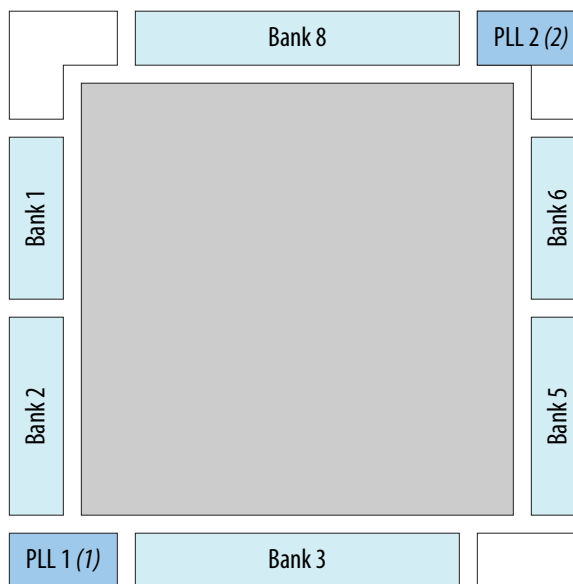
Figure 12. Intel MAX 10 PLL High-Level Block Diagram

Each clock source can come from any of the two or four clock pins located on the same side of the device as the PLL.

**Notes:**

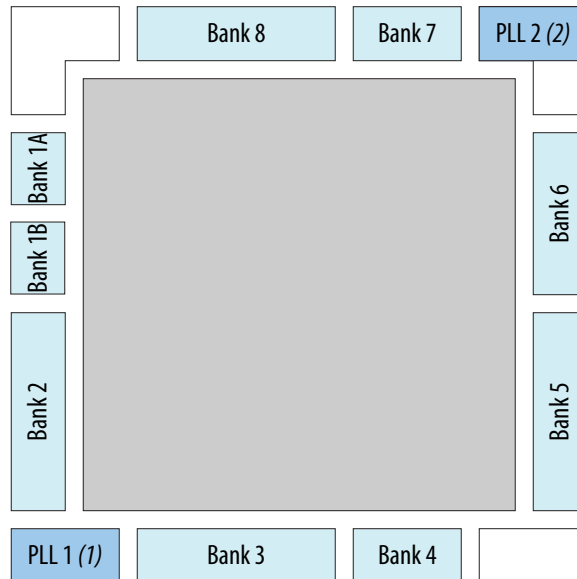
- (1) This is the VCO post-scale counter K.
- (2) Only counter C0 of PLL1 and PLL3 can drive the ADC clock.

The following figures show the physical locations of the PLLs. Every index represents one PLL in the device. The physical locations of the PLLs correspond to the coordinates in the Intel Quartus Prime Chip Planner.

Figure 13. PLL Locations for 10M02 Device**Notes:**

- (1) Available on all packages except V36 package.
- (2) Available on U324 and V36 packages only.

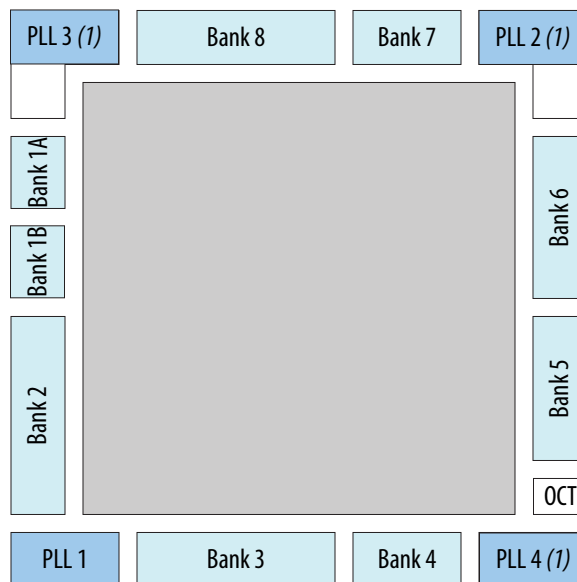
Figure 14. PLL Locations for 10M04 and 10M08 Devices



Notes:

- (1) Available on all packages except V81 package.
- (2) Available on F256, F484, U324 (dual power supply), and V81 packages only.

Figure 15. PLL Locations for 10M16, 10M25, 10M40 and 10M50 Devices



Note:

- (1) Available on all packages except E144, U169, Y180, and U324 (single power supply) packages.

1.5. General Purpose I/O

The I/O system of Intel MAX 10 devices support various I/O standards. In the Intel MAX 10 devices, the I/O pins are located in I/O banks at the periphery of the devices. The I/O pins and I/O buffers have several programmable features.

Related Information

[Intel MAX 10 General Purpose I/O User Guide](#)

1.5.1. Intel MAX 10 I/O Banks Architecture

The I/O elements are located in a group of four modules per I/O bank:

- High speed DDR3 I/O banks—supports various I/O standards and protocols including DDR3 but not the 1.8 V LVDS I/O standard. These I/O banks are available only on the right side of the device.
- High speed I/O banks—supports various I/O standards and protocols except DDR3. These I/O banks are available on the top, left, and bottom sides of the device.
- Low speed I/O banks—lower speeds I/O banks that are located at the top left side of the device.

For more information about I/O pins support, refer to the pinout files for your device.

1.5.2. Intel MAX 10 I/O Banks Locations

The I/O banks are located at the periphery of the device.

For more details about the modular I/O banks available in each device package, refer to the relevant device pin-out file.

Figure 16. I/O Banks for 10M02 Devices (Except Single Power Supply U324 Package)

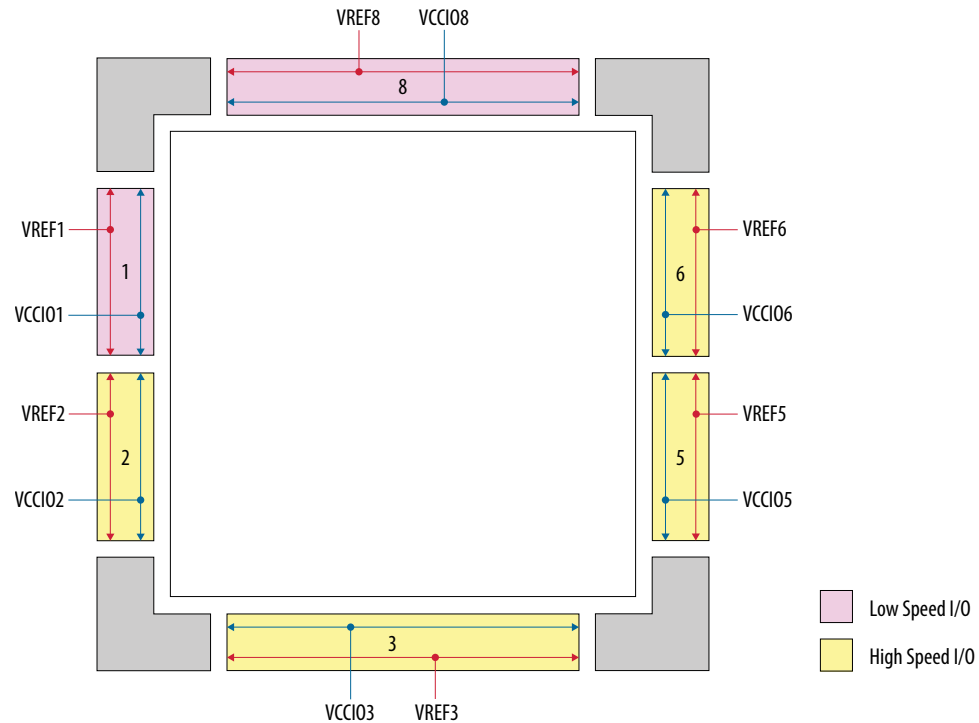


Figure 17. I/O Banks for 10M02 (Single Power Supply U324 Package), 10M04, and 10M08 (Except V81, M153, and U169 Packages) Devices

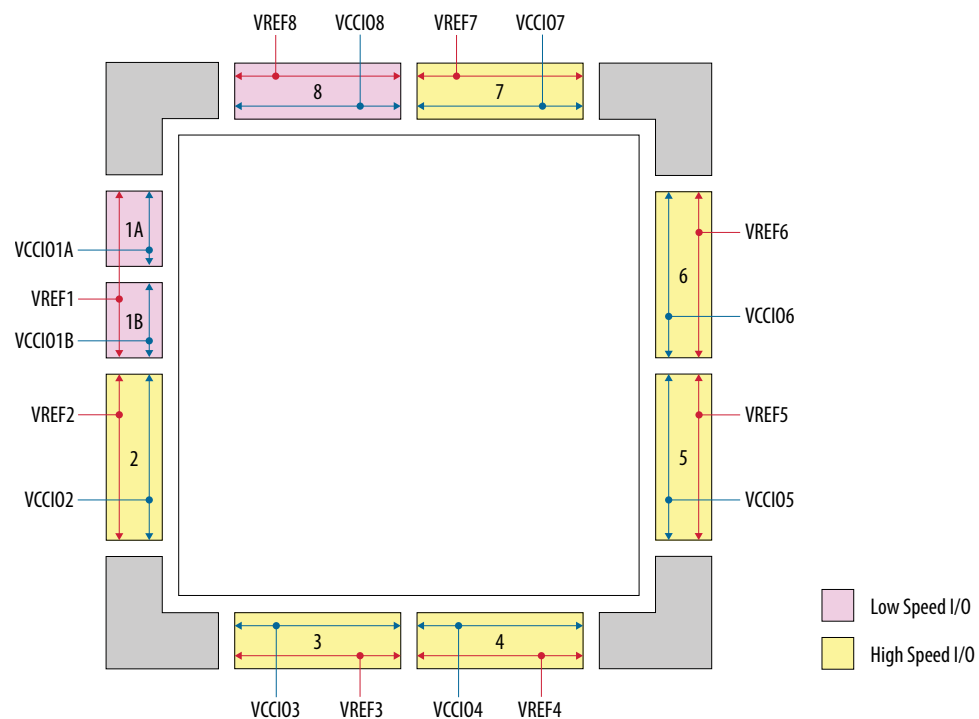


Figure 18. I/O Banks for 10M08 V81, M153, and U169 Packages Devices

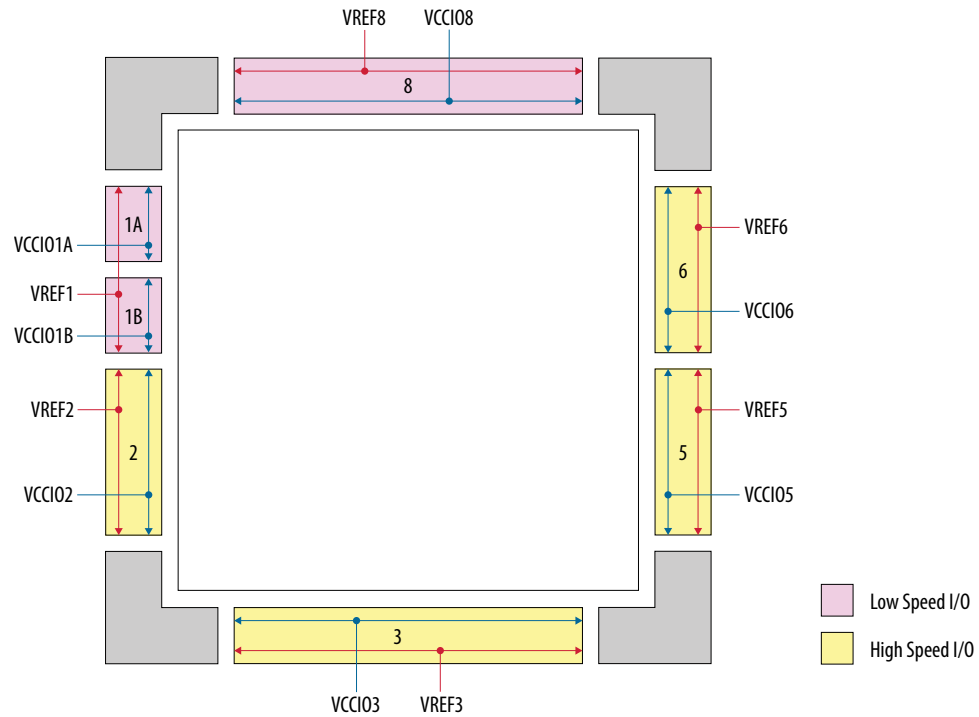
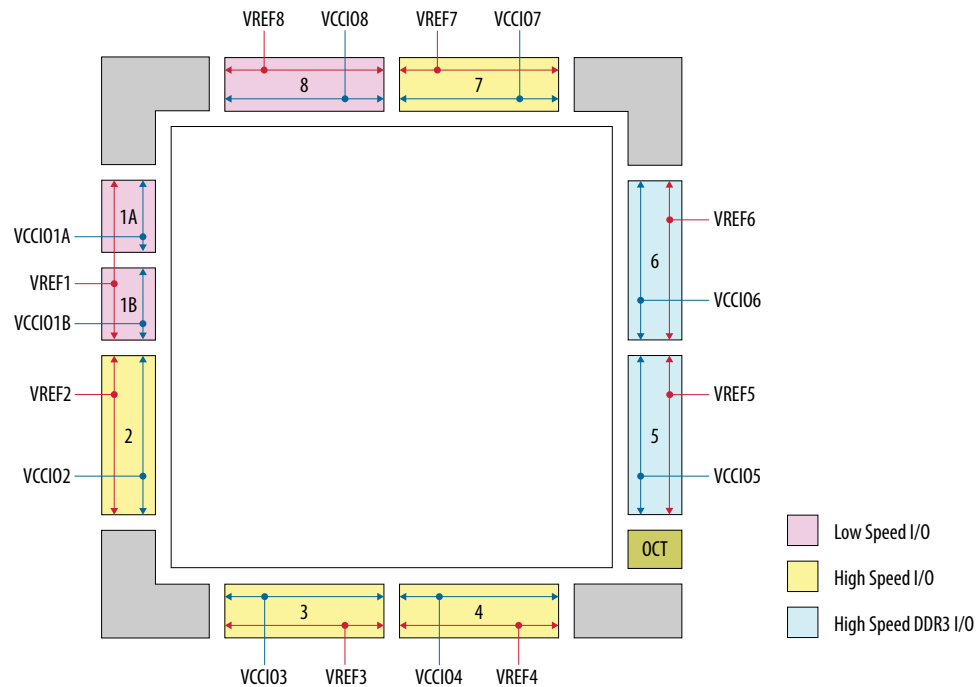


Figure 19. I/O Banks for 10M16, 10M25, 10M40, and 10M50 Devices



1.6. High-Speed LVDS I/O

The Intel MAX 10 device family supports high-speed LVDS protocols through the LVDS I/O banks and the Soft LVDS Intel FPGA IP.

The Intel MAX 10 devices use registers and logic in the core fabric to implement LVDS input and output interfaces.

- For LVDS transmitters and receivers, Intel MAX 10 devices use the double data rate I/O (DDIO) registers that reside in the I/O elements (IOE). This architecture improves performance with regards to the receiver input skew margin (RSKM) or transmitter channel-to-channel skew (TCCS).
- For the LVDS serializer/deserializer (SERDES), Intel MAX 10 devices use logic elements (LE) registers.

Related Information

[Intel MAX 10 High-Speed LVDS I/O User Guide](#)

1.6.1. Intel MAX 10 High-Speed LVDS Circuitry

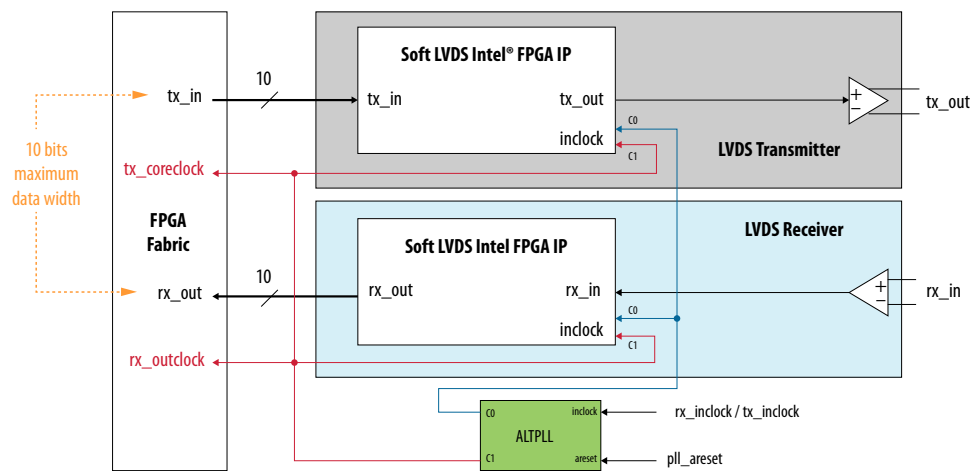
The LVDS solution uses the I/O elements and registers in the Intel MAX 10 devices. The Soft LVDS IP core implements the serializer and deserializer as soft SERDES blocks in the core logic.

The Intel MAX 10 devices do not contain dedicated serialization or deserialization circuitry:

- You can use I/O pins and core fabric to implement a high-speed differential interface in the device.
- The Intel MAX 10 solution uses shift registers, internal PLLs, and I/O elements to perform the serial-to-parallel and parallel-to-serial conversions of incoming and outgoing data.
- The Intel Quartus Prime software uses the parameter settings of the Soft LVDS IP core to automatically construct the differential SERDES in the core fabric.

Figure 20. Soft LVDS SERDES

This figure shows a transmitter and receiver block diagram for the soft LVDS SERDES circuitry with the interface signals of the transmitter and receiver data paths.



1.6.2. Intel MAX 10 High-Speed LVDS I/O Location

All I/O banks in Intel MAX 10 devices support 2.5 V true LVDS input and 2.5 V emulated LVDS output. The high-speed I/O banks also support 1.8 V true LVDS input. Only the bottom I/O banks support 2.5 V and 1.8 V true LVDS outputs.

Note: The 1.8 V LVDS buffers are supported as inputs on all high-speed I/O banks but as outputs only on the bottom banks. The low-speed and high-speed DDR3 I/O banks do not support 1.8 V LVDS. The 1.8 V LVDS I/O standard is supported in industrial- and commercial-grade Intel MAX 10 dual supply devices except in packages V36 and V81. Refer to the related information.

Figure 21. LVDS Support in I/O Banks of 10M02 Devices (Except Single Power Supply U324 Package)

This figure shows a top view of the silicon die. Each bank is labeled with the actual bank number. LVPECL support only in banks 2 and 6.

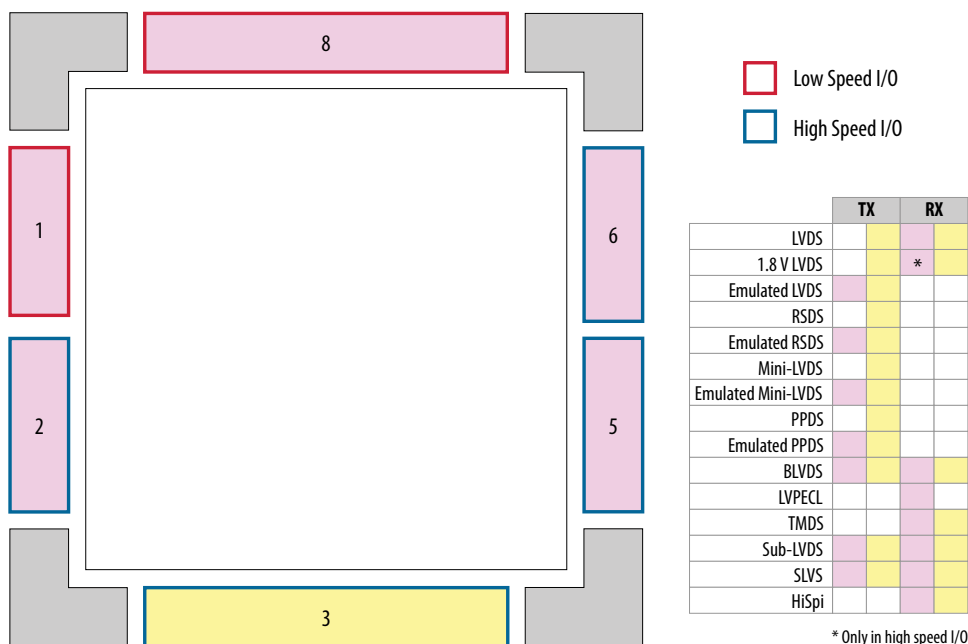


Figure 22. LVDS Support in I/O Banks of 10M02 (Single Power Supply U324 Package), 10M04, and 10M08 (Except V81, M153, and U169 Packages) Devices

This figure shows a top view of the silicon die. Each bank is labeled with the actual bank number. LVPECL support only in banks 2 and 6.

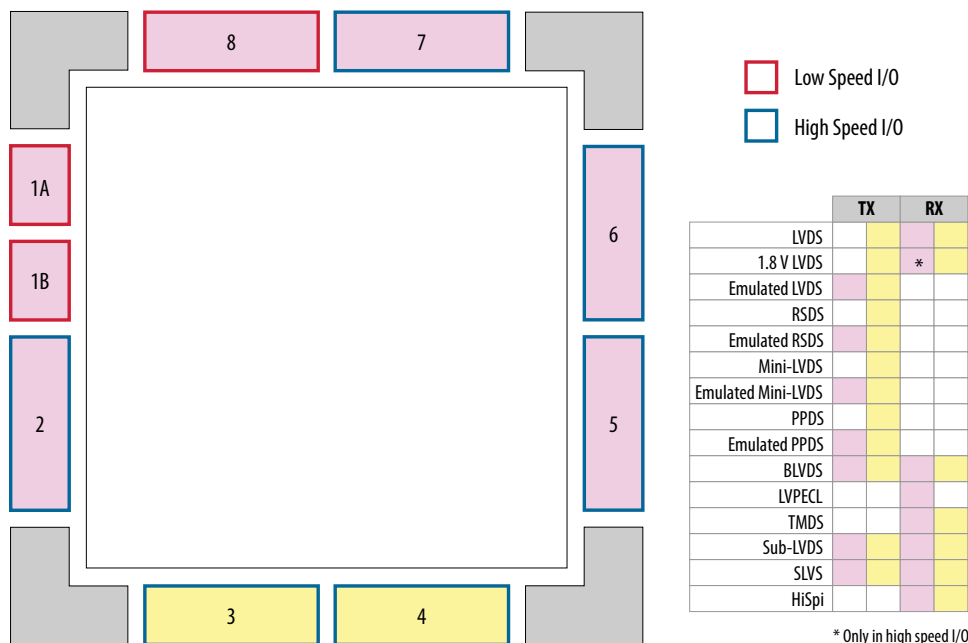


Figure 23. LVDS Support in I/O Banks of 10M08 V81, M153, and U169 Packages Devices

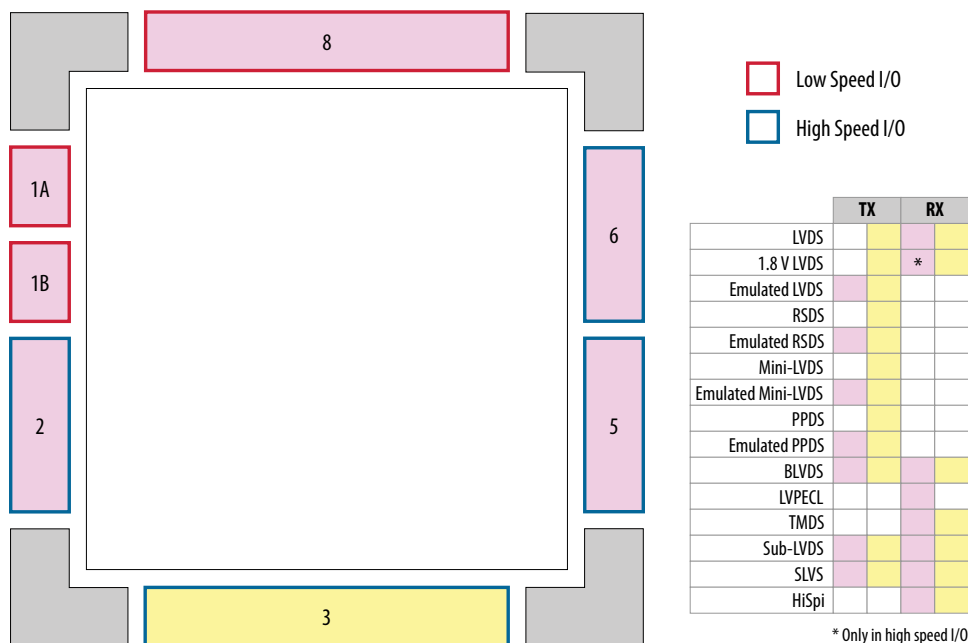
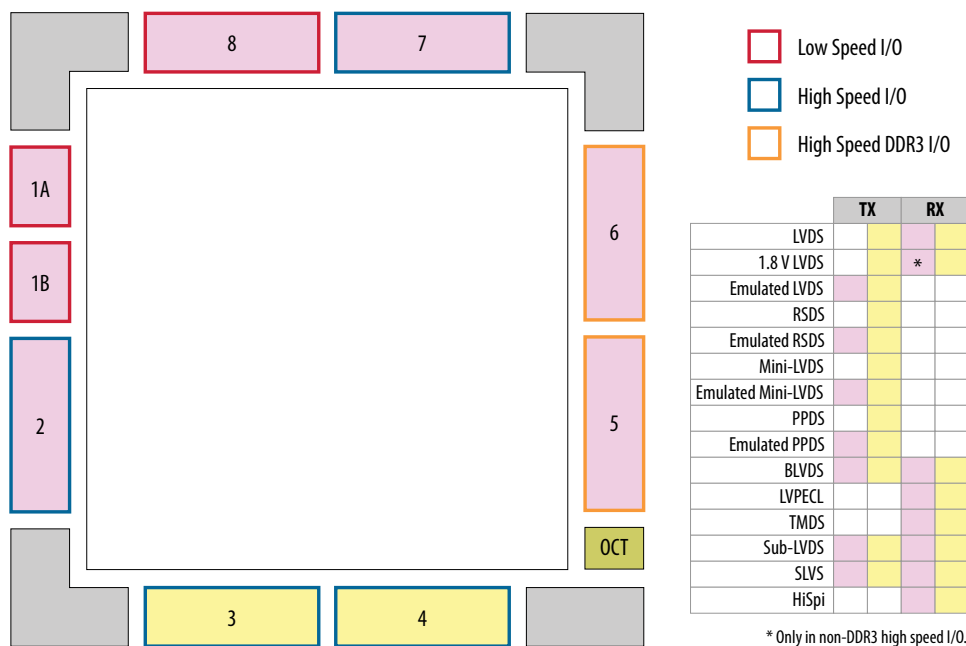


Figure 24. LVDS Support in I/O Banks of 10M16, 10M25, 10M40, and 10M50 Devices

This figure shows a top view of the silicon die. Each bank is labeled with the actual bank number. LVPECL support only in banks 2, 3, 6, and 8.



1.7. External Memory Interface

The Intel MAX 10 devices are capable of interfacing with a broad range of external memory standards.

This capability allows you to use the Intel MAX 10 devices in a wide range of applications such as image processing, storage, communications, and general embedded systems.

The external memory interface solution in Intel MAX 10 devices consist of:

- The I/O elements that support external memory interfaces.
- The UniPHY IP core that allows you to configure the memory interfaces to support different external memory interface standards.

Related Information

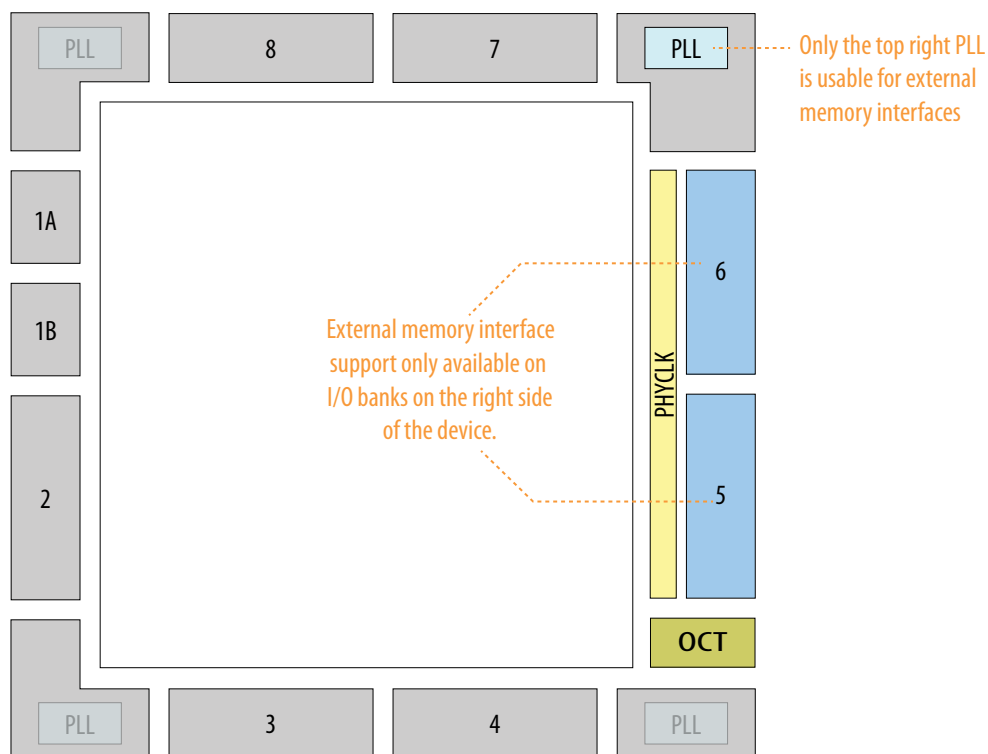
[Intel MAX 10 External Memory Interface User Guide](#)

1.7.1. Intel MAX 10 I/O Banks for External Memory Interface

In Intel MAX 10 devices, external memory interfaces are supported only on the I/O banks on the right side of the device. You must place all external memory I/O pins on the I/O banks on the right side of the device.

Figure 25. I/O Banks for External Memory Interfaces

This figure represents the top view of the silicon die that corresponds to a reverse view of the device package.



External memory interfaces support is available only for dual supply (DC, DF, and DA) variant on 10M16, 10M25, 10M40, and 10M50 devices.

1.8. Analog to Digital Converter

Intel MAX 10 devices feature up to two analog-to-digital converters (ADC). The ADCs provide the Intel MAX 10 devices with built-in capability for on-die temperature monitoring and external analog signal conversion.

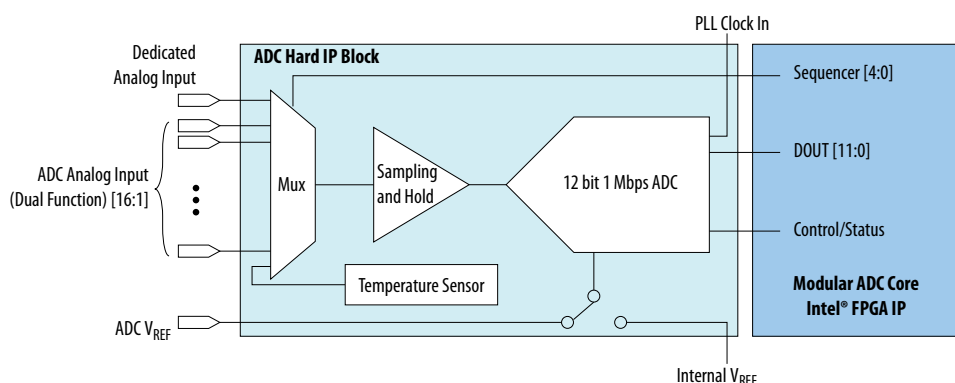
The ADC solution consists of hard IP blocks in the Intel MAX 10 device periphery and soft logic through the Modular ADC Core Intel FPGA IP and Modular Dual ADC Core Intel FPGA IP.

The ADC solution provides you with built-in capability to translate analog quantities to digital data for information processing, computing, data transmission, and control systems. The basic function is to provide a 12 bit digital representation of the analog signal being observed.

The ADC solution works in two modes:

- Normal mode—monitors single-ended external inputs with a cumulative sampling rate of up to 1 million samples per second (MSPS):
 - Single ADC devices—up to 17 single-ended external inputs (one dedicated analog and 16 dual function input pins)
 - Dual ADC devices—up to 18 single-ended external inputs (one dedicated analog and eight dual function input pins in each ADC block)
- Temperature sensing mode—monitors external temperature data input with a sampling rate of up to 50 kilosamples per second. In dual ADC devices, only the first ADC block supports this mode.

Figure 26. ADC Hard IP Block in Intel MAX 10 Devices



Related Information

[Intel MAX 10 Analog to Digital Converter User Guide](#)

1.8.1. ADC Block Locations

The ADC blocks are located at the top left corner of the Intel MAX 10 device periphery.

Figure 27. ADC Block Location in Intel MAX 10 04 and 08 Devices

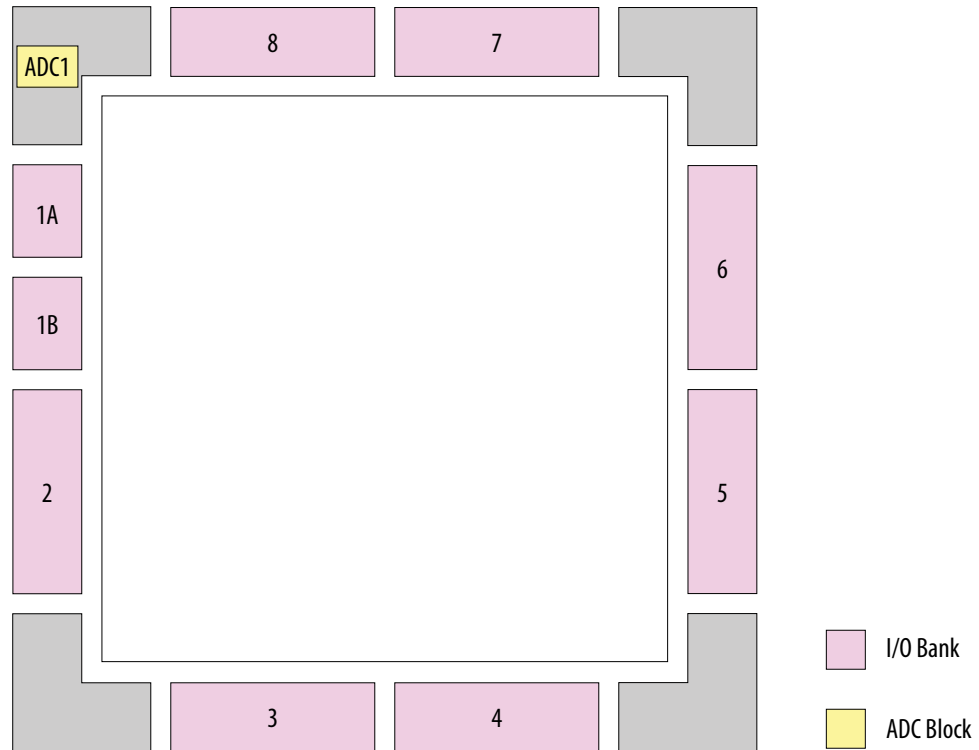


Figure 28. ADC Block Location in Intel MAX 10 16 Devices

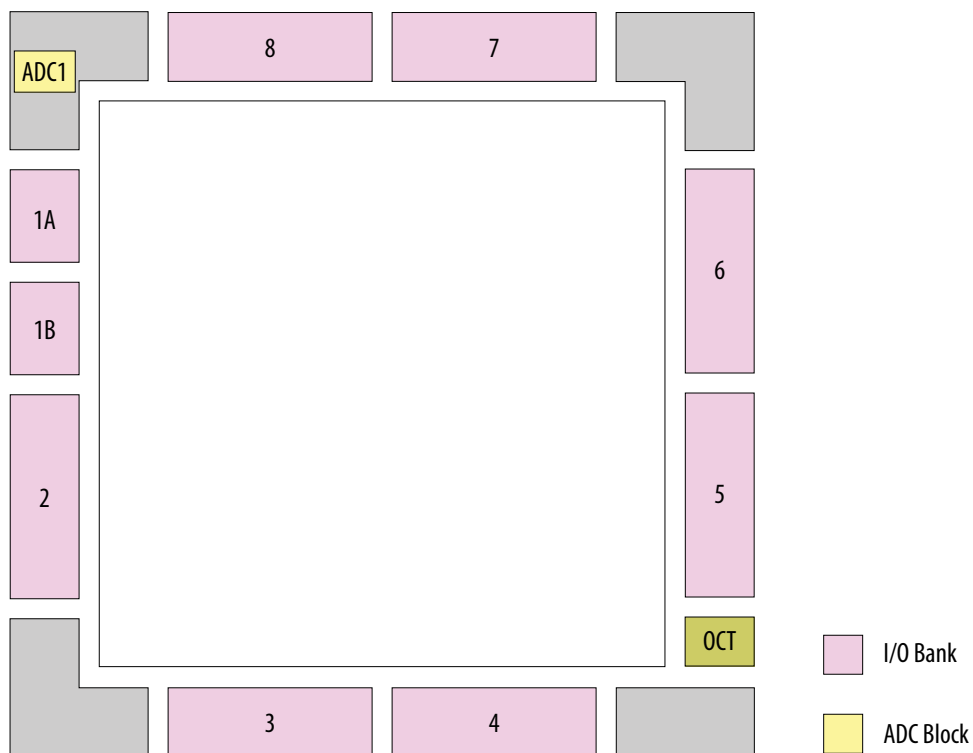
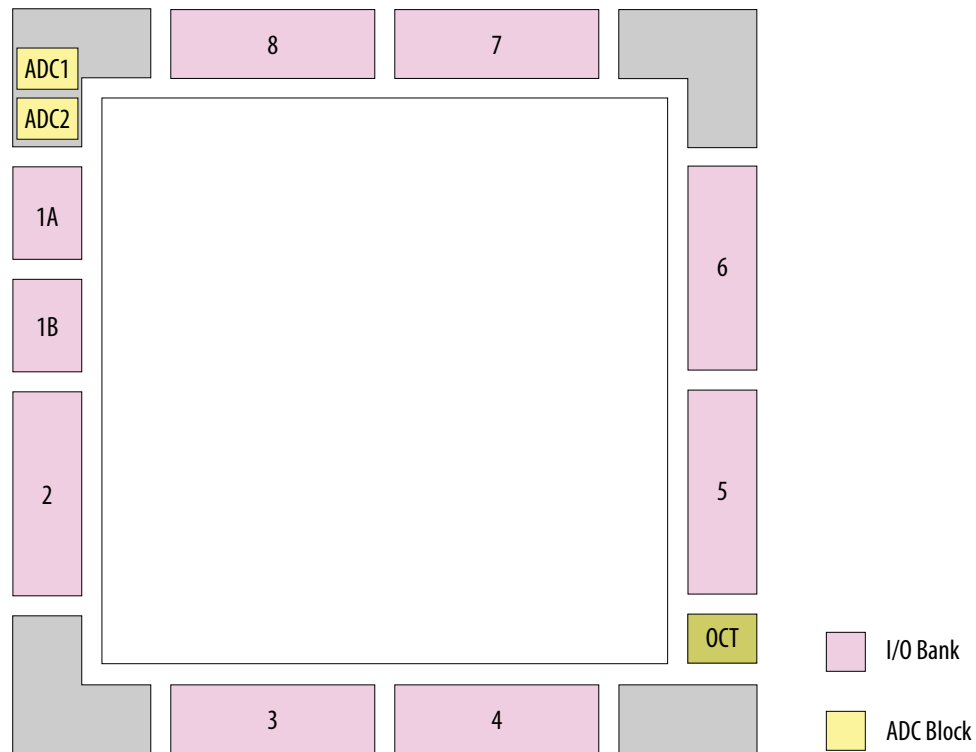


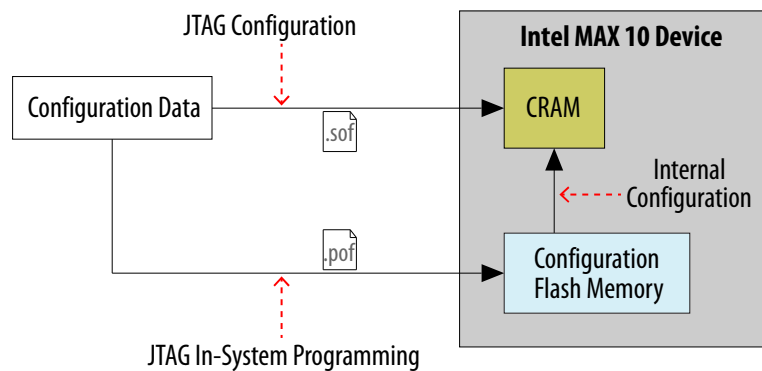
Figure 29. ADC Block Location in Intel MAX 10 25, 40, and 50 Devices

Package E144 of these devices have only one ADC block.



1.9. Configuration Schemes

Figure 30. High-Level Overview of JTAG Configuration and Internal Configuration for Intel MAX 10 Devices



Related Information

[Intel MAX 10 FPGA Configuration User Guide](#)

1.9.1. JTAG Configuration

In Intel MAX 10 devices, JTAG instructions take precedence over the internal configuration scheme.

Using the JTAG configuration scheme, you can directly configure the device CRAM through the JTAG interface—TDI, TDO, TMS, and TCK pins. The Intel Quartus Prime software automatically generates an SRAM Object File (**.sof**). You can program the **.sof** using a download cable with the Intel Quartus Prime software programmer.

1.9.2. Internal Configuration

You need to program the configuration data into the configuration flash memory (CFM) before internal configuration can take place. The configuration data to be written to CFM will be part of the programmer object file (**.pof**). Using JTAG In-System Programming (ISP), you can program the **.pof** into the internal flash.

During internal configuration, Intel MAX 10 devices load the CRAM with configuration data from the CFM.

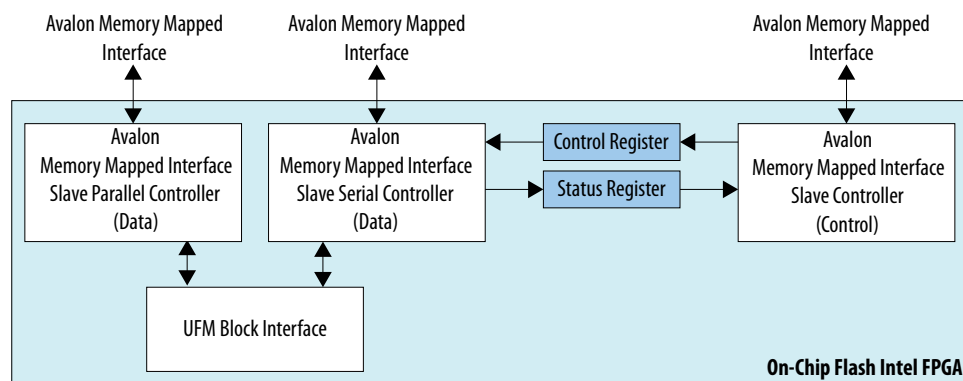
1.10. User Flash Memory

Intel MAX 10 devices feature a user flash memory (UFM) block that stores non-volatile information.

The UFM is part of the internal flash available in Intel MAX 10 devices.

The UFM architecture of Intel MAX 10 devices is a combination of soft and hard IPs. You can only access the UFM using the On-Chip Flash Intel FPGA IP in the Intel Quartus Prime software.

Figure 31. On-Chip Flash IP Block Diagram



This IP block has two Avalon® memory-mapped interface slave controllers:

- Data—a wrapper of the UFM block that provides read and write accesses to the flash.
- Control—the CSR and status register for the flash, that is required only for write operations.

Related Information

[Intel MAX 10 User Flash Memory \(UFM\) User Guide](#)

1.11. Power Management

Intel MAX 10 power optimization features are as follows:

- Single-supply or dual-supply device options
- Power-on reset (POR) circuitry
- Power management controller scheme
- Hot socketing

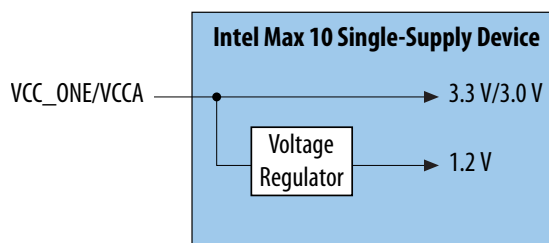
Related Information

[Intel MAX 10 Power Management User Guide](#)

1.11.1. Single-Supply Device

Intel MAX 10 single-supply devices only need either a 3.0- or 3.3-V external power supply. The external power supply serves as an input to the Intel MAX 10 device VCC_ONE and VCCA power pins. This external power supply is then regulated by an internal voltage regulator in the Intel MAX 10 single-supply device to 1.2 V. The 1.2-V voltage level is required by core logic operation.

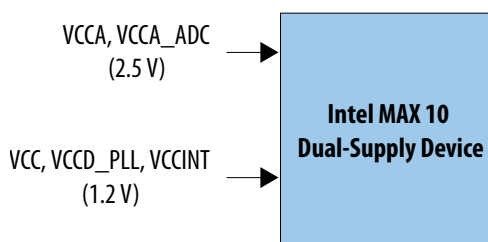
Figure 32. Intel MAX 10 Single-Supply Device



1.11.2. Dual-Supply Device

Intel MAX 10 dual-supply devices require 1.2 V and 2.5 V for the device core logic and periphery operation.

Figure 33. Intel MAX 10 Dual-Supply Device



1.11.3. Power Management Controller Scheme

The power management controller scheme allows you to allocate some applications in sleep mode during runtime. This enables you to turn off portions of the design, thus reducing dynamic power consumption. You can re-enable your application with a fast wake-up time of less than 1 ms.

1.11.4. Hot Socketing

The Intel MAX 10 device offers hot socketing, which is also known as hot plug-in or hot swap, and power sequencing support without the use of any external devices. You can insert or remove the Intel MAX 10 device on a board in a system during system operation. This does not affect the running system bus or the board that is inserted into the system.

The hot-socketing feature removes some encountered difficulties when using the Intel MAX 10 device on a PCB that contains a mixture of devices with different voltage levels.

With the Intel MAX 10 device hot-socketing feature, you no longer need to ensure a proper power-up sequence for each device on the board. Intel MAX 10 device hot-socketing feature provides:

- Board or device insertion and removal without external components or board manipulation
- Support for any power-up sequence
- Non-intrusive I/O buffers to system buses during hot insertion

1.12. Document Revision History for Intel MAX 10 FPGA Device Architecture

Document Version	Changes
2022.10.31	<ul style="list-style-type: none"> • Added 1.8 V LVDS I/O standard support.
2021.11.01	<ul style="list-style-type: none"> • Added Y180 package information and updated note in the <i>PLL Locations for 10M16, 10M25, 10M40, and 10M50 Devices</i> diagram. • Updated figure title for the following diagrams: <ul style="list-style-type: none"> — <i>I/O Banks for 10M02 (Single Power Supply U324 Package), 10M04, and 10M08 (Except V81, M153, and U169 Packages) Devices</i> — <i>LVDS Support in I/O Banks of 10M02 (Single Power Supply U324 Package), 10M04, and 10M08 (Except V81, M153, and U169 Packages) Devices</i> • Added the following diagrams: <ul style="list-style-type: none"> — <i>I/O Banks for 10M08 V81, M153, and U169 Packages Devices</i> — <i>LVDS Support in I/O Banks of 10M08 V81, M153, and U169 Packages Devices</i> • Updated the <i>Analog to Digital Converter</i> topic to add information for dual ADC devices.

Date	Version	Changes
February 2017	2017.02.21	Rebranded as Intel.
August 2016	2016.08.11	Removed content duplication in <i>Embedded Multiplier</i> .
continued...		

Date	Version	Changes
May 2016	2016.05.13	<ul style="list-style-type: none"> Added internal oscillator architectural information. Updated section name from <i>Clock Networks and PLL</i> to <i>Clocking and PLL</i>. Added high-speed LVDS circuitry information. Added power management controller scheme and hot socketing information.
May 2015	2015.05.04	<ul style="list-style-type: none"> Removed 'Internal Configuration' figure. Added 'Overview of JTAG Configuration and Internal Configuration for Intel MAX 10 Devices' figure in 'Configuration'.
December 2014	2014.12.15	<ul style="list-style-type: none"> Updated Altera On Chip Flash IP core block diagram for user flash memory. Updated links.
September 2014	2014.09.22	Initial release.

MAX 10 FPGA Development Kit User Guide



Subscribe



Send Feedback

UG-01169
2017.09.07

101 Innovation Drive
San Jose, CA 95134
www.altera.com

ALTERA
now part of Intel®

Contents

Overview.....	1-1
General Description.....	1-2
Handling the Board.....	1-4
Getting Started.....	2-1
Quartus II Web Edition Software.....	2-1
Installing the Development Kit.....	2-1
Installing the USB-Blaster Driver.....	2-2
Board Update Portal.....	2-2
Board Test System.....	3-1
Using the Configure Menu.....	3-3
The System Info Tab.....	3-5
The GPIO Tab.....	3-7
The Flash Tab.....	3-9
The HSMC Tab.....	3-11
The DDR3 Tab.....	3-13
The ADC Tab.....	3-15
The HDMI Tab.....	3-17
The Sleep Mode Tab.....	3-18
The Power Monitor.....	3-20
The Clock Control.....	3-22
Board Components.....	4-1
Board Overview.....	4-1
Featured Device.....	4-3
Configuration.....	4-4
Using the Quartus II Programmer.....	4-4
Selecting the Internal Configuration Scheme.....	4-4
Switch and Jumper Settings.....	4-5
Status Elements.....	4-7
Setup Elements.....	4-8
General User Input/Output.....	4-8
Clock Circuitry.....	4-9
On-Board Oscillators.....	4-10
Off-Board Clock Input/Output.....	4-11
Components and Interfaces.....	4-12
10/100/1000 Ethernet PHY.....	4-12
Digital-to-Analog Converter.....	4-15
HDMI Video Output.....	4-16

HSMC.....	4-17
Pmod Connectors.....	4-22
USB to UART.....	4-23
Memory.....	4-24
DDR3 Rev. B Board.....	4-24
DDR3 Rev. C Board.....	4-26
Flash.....	4-29
Power Distribution System.....	4-31
Additional Information.....	A-1
User Guide Revision History.....	A-1
Compliance and Conformity Statements.....	A-2
CE EMI Conformity Caution.....	A-2

2017.09.07

UG-01169



Subscribe



Send Feedback

The MAX[®] 10 FPGA development board provides a hardware platform for evaluating the performance and features of the Intel[®] MAX 10 device.

The development kit includes a RoHS- and CE-compliant MAX 10 FPGA Development board with the following components:

- Featured Devices:
 - MAX 10 FPGA (10M50D, dual supply, F484 package)
 - Enpirion[®] EN2342QI 4 A PowerSoC Voltage-Mode Synchronous Step-Down Converter with Integrated Inductor Enpirion
 - EN6337QI 3 A High-Efficiency PowerSoC DC-DC Step-Down Converters with Integrated Inductor
 - Enpirion EP5358xUI 600 mA PowerSoC DC-DC Step-Down Converters with Integrated Inductor
 - MAX II CPLD – EPM1270M256C4N (On-board USB-Blaster[™] II)
- Programming and Configuration:
 - Embedded USB-Blaster II (JTAG)
 - Optional JTAG direct via 10-pin header
- Memory Devices:
 - 64-Mx16 1 Gb DDR3 SDRAM with soft memory controller
 - 128-Mx8 1 Gb DDR3 SDRAM with soft memory controller
 - 512-Mb Quad serial peripheral interface (quad SPI) flash
- Communication Ports:
 - Two Gigabit Ethernet (GbE) RJ-45 ports
 - Ethernet Port A (Bottom)
 - Ethernet Port B (Top)
 - One mini-USB2.0 UART
 - One high-definition multimedia interface (HDMI) video output
 - One universal high-speed mezzanine card (HSMC) connector
 - Two 12-pin Digilent Pmod[™] compatible connectors

Intel Corporation. All rights reserved. Intel, the Intel logo, Altera, Arria, Cyclone, Enpirion, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.

ISO
9001:2015
Registered

ALTERA
now part of Intel

- Analog:
 - Two MAX 10 FPGA analog-to-digital converter (ADC) SMA inputs
 - 2x10 ADC header
 - Potentiometer input to ADC
 - One external 16 bit digital-to-analog converter (DAC) device with SMA output
- Clocking
 - 25 MHz single-ended, external oscillator clock source
 - Silicon labs clock generator with programmable frequency GUI
- Mini-USB cable for on-board USB-Blaster™ II
- 2A Power Supply and cord
- Free Quartus® II Web Edition design software (download software and license from website)
- Complete documentation
 - User manual, bill of materials, schematic, and board files

General Description

Figure 1-1: MAX 10 FPGA Board Components (Top)

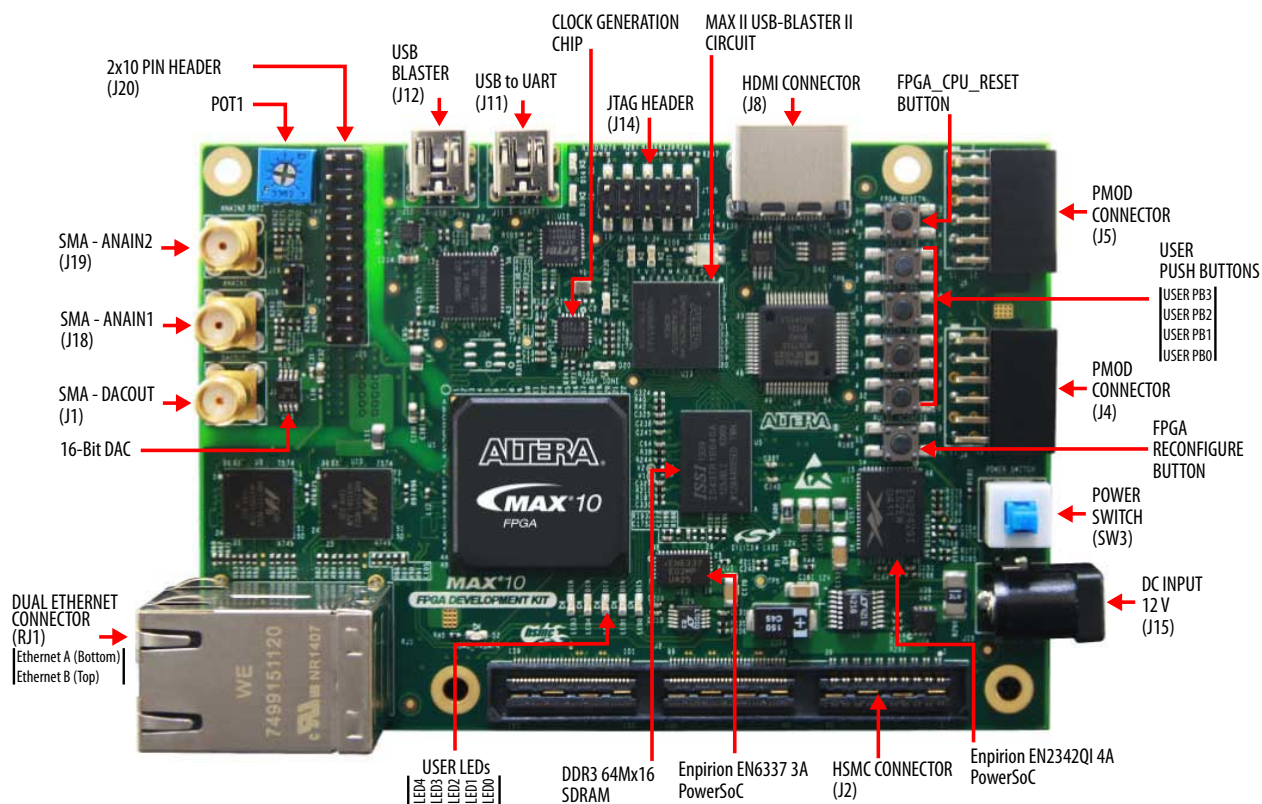


Figure 1-2: MAX 10 FPGA Board Components (Bottom)

Note: To determine the revision of your board, look for the serial number at the bottom of the board.

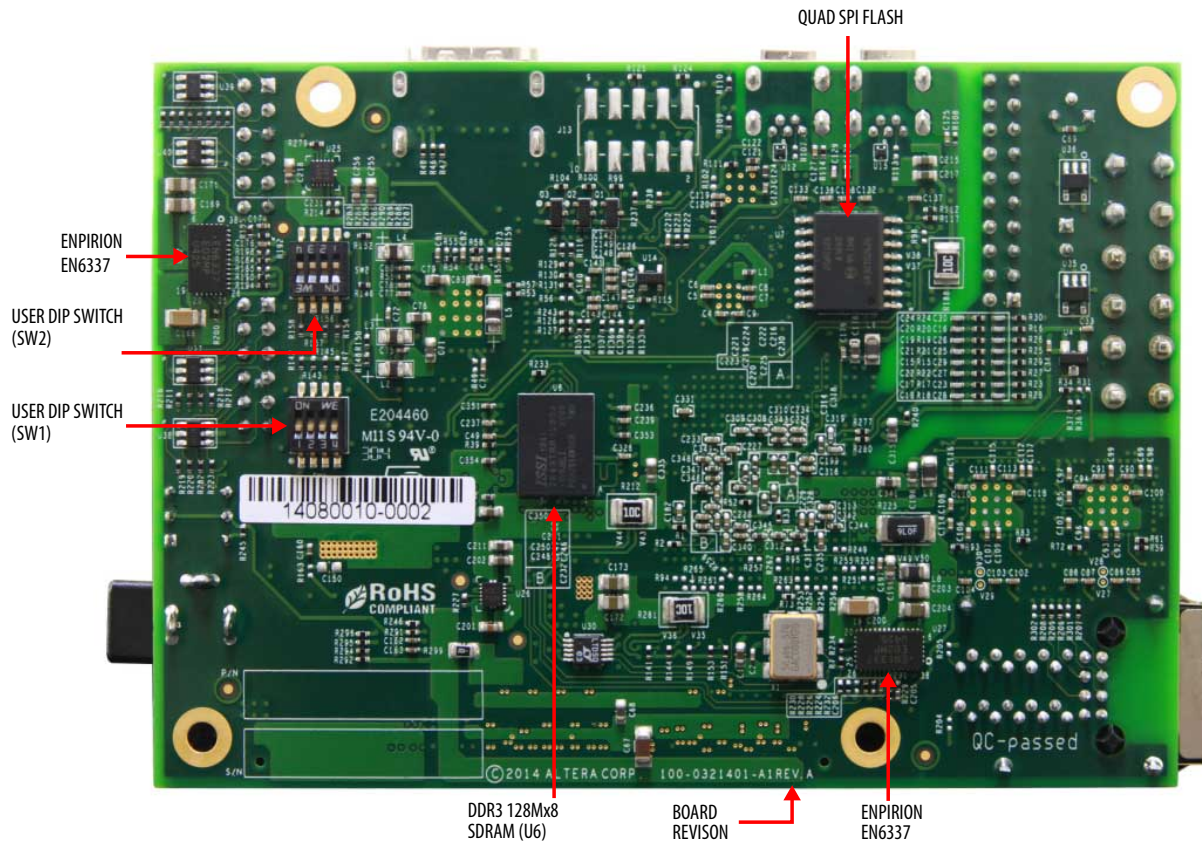
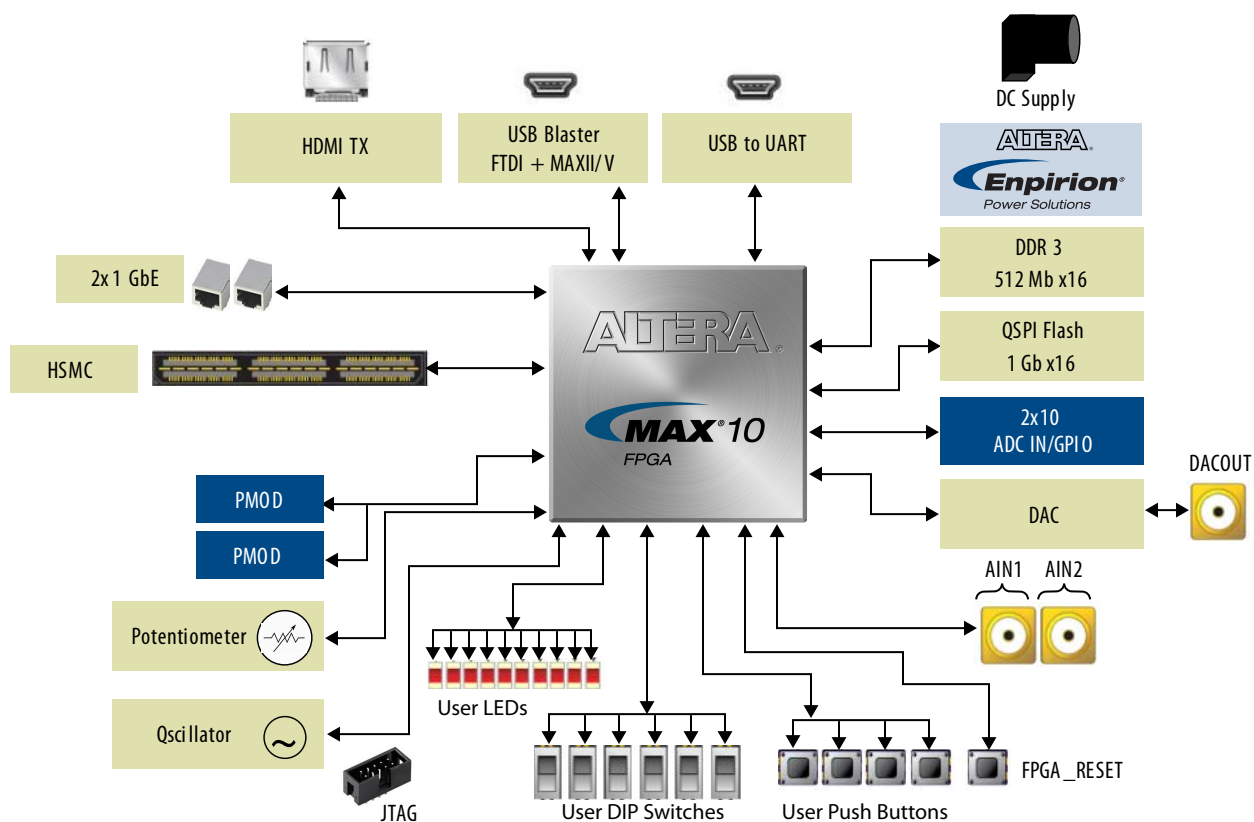


Figure 1-3: System Block Diagram



Handling the Board

When handling the board, it is important to observe static discharge precautions.

Caution: Without proper anti-static handling, the board can be damaged. Therefore, use anti-static handling precautions when touching the board.

Caution: This development kit should not be operated in a Vibration Environment.

2017.09.07

UG-01169



Subscribe



Send Feedback

Quartus II Web Edition Software

The Quartus II Web Edition Software is a free with no license required.

You can download the Web Edition software from the Altera website. Alternatively, you can request a DVD.

Related Information

- [Quartus II Web Edition Software](#)
- [Altera IP and Software DVD Request Form](#)

Installing the Development Kit

1. Download the MAX 10 Development Kit installer from the MAX 10 FPGA Development Kit page of the Altera website. Alternatively, you can request a development kit DVD from the Altera Kit Installations DVD Request Form page of the Altera website.
2. Run the MAX 10 FPGA Development Kit installer.
3. Follow the on-screen instructions to complete the installation process. Be sure that the installation directory you choose is in the same relative location to the Quartus II software installation.

The installation program creates the development kit directory structure shown in the following figure.

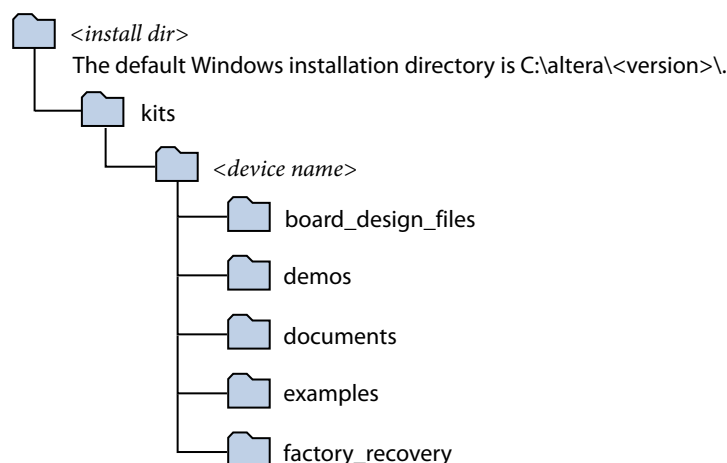
Attention: .sof files are used by BTS GUI to configure the MAX 10 device and start corresponding test. Therefore, do not to move the .sof files from the *\\examples\\board_test_system directory.

Intel Corporation. All rights reserved. Intel, the Intel logo, Altera, Arria, Cyclone, Enpirion, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.

ISO
9001:2015
Registered

ALTERA
now part of Intel

Figure 2-1: Installed Development Kit Directory Structure**Table 2-1: Installed Directory Contents**

Directory Name	Description of Contents
board_design_files	Contains schematic, layout, assembly, and bill of material board design files. Use these files as a starting point for a new prototype board design.
demos	Contains demonstration applications when available.
documents	Contains the following documentation: <ul style="list-style-type: none"> MAX 10 FPGA Development Kit User Guide Quick Start Guide Dear Customer Letter
examples	Contains the sample design files for this kit.
factory_recovery	Contains the original data programmed onto the board before shipment. Use this data to restore the board with its original factory contents.

Installing the USB-Blaster Driver

The development board includes integrated USB-Blaster circuitry for FPGA programming. However, for the host computer and board to communicate, you must install the On-Board USB-Blaster II driver on the host computer.

Installation instructions for the On-Board USB-Blaster II driver for your operating system are available on the Altera website. On the Altera Programming Cable Driver Information page of the Altera website, locate the table entry for your configuration and click the link to access the instructions.

Board Update Portal

You can keep your board current by accessing the Board Update Portal on www.altera.com.

This web site allows you access useful information and updated software and design examples for your board. For instructions on setting up your board to access the Board Update Portal, consult the printed *Quick Start Guide* that is included in the kit box.

2017.09.07

UG-01169



Subscribe



Send Feedback

This kit includes an application called the Board Test System (BTS).

The BTS provides an easy-to-use interface to alter functional settings and observe the results. You can use the BTS to test board components, modify functional parameters, observe performance, and measure power usage. While using the BTS, you reconfigure the FPGA several times with test designs specific to the functionality you are testing.

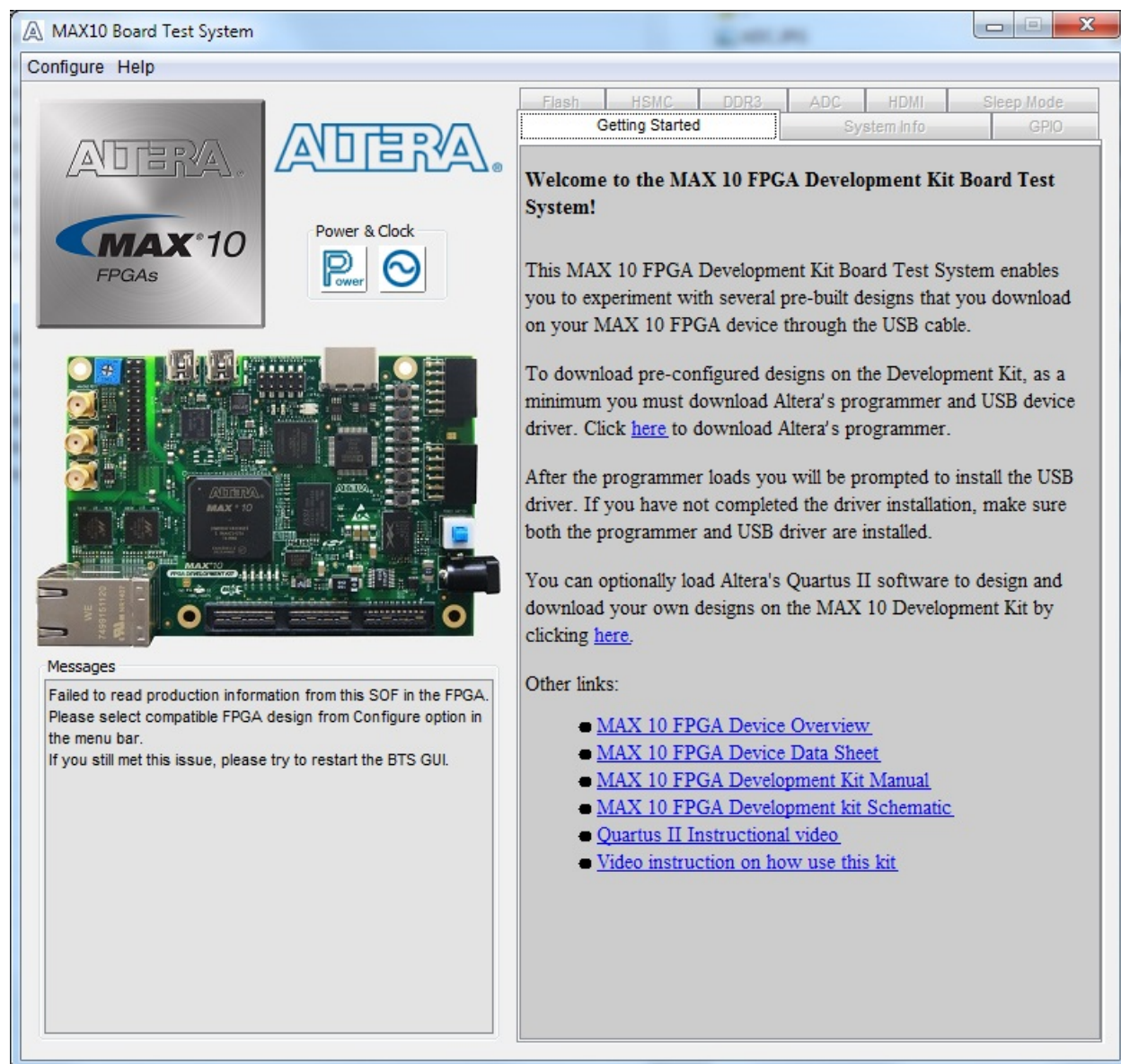
Intel Corporation. All rights reserved. Intel, the Intel logo, Altera, Arria, Cyclone, Enpirion, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.

ISO
9001:2015
Registered

ALTERA
now part of Intel

Figure 3-1: Board Test System GUI



Several designs are provided to test the major board features. Each design provides data for one or more tabs in the application. The Configure menu identifies the appropriate design to download to the FPGA for each tab.

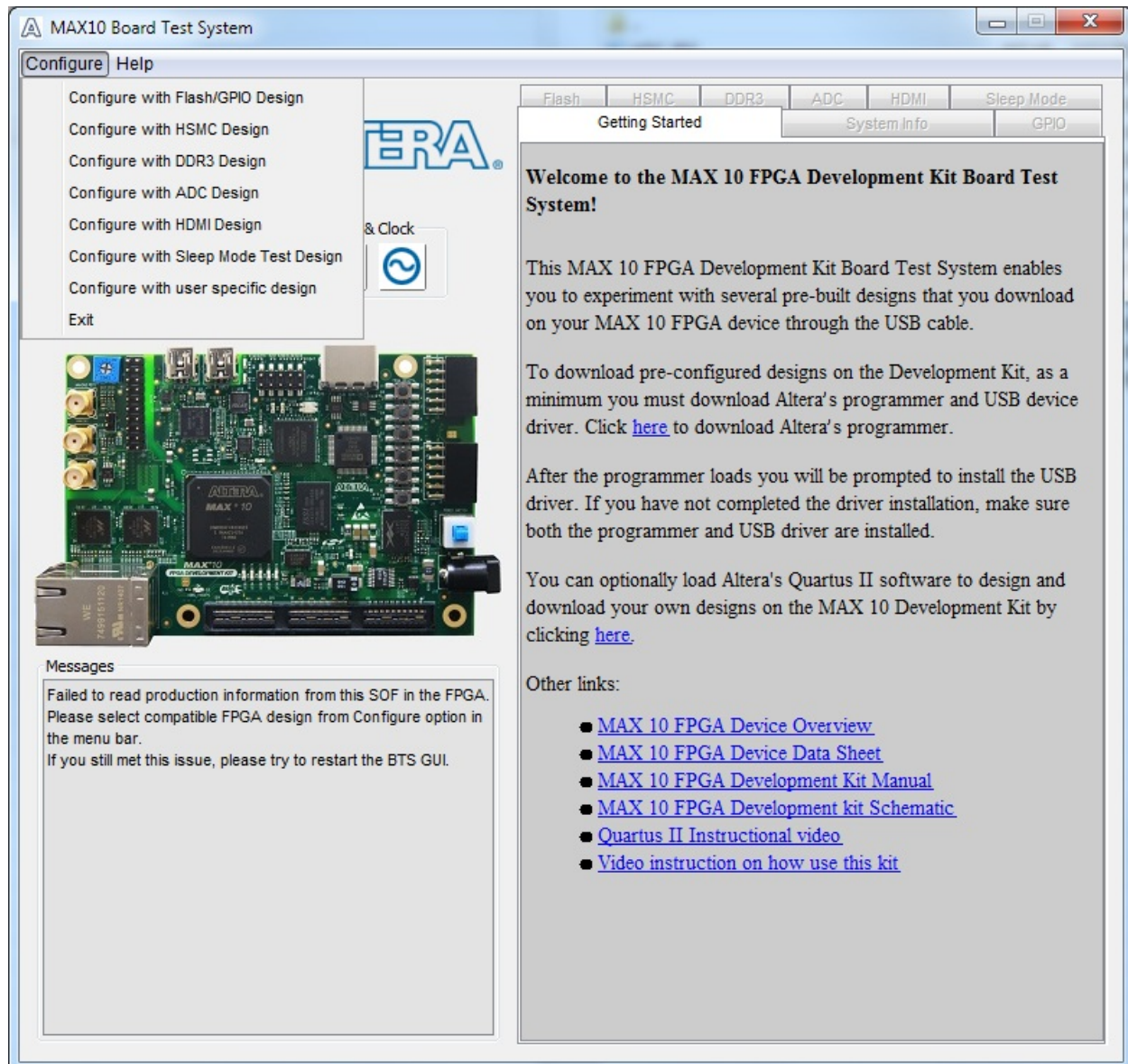
After successful FPGA configuration, the appropriate tab appears that allows you to exercise the related board features. Highlights appear in the board picture around the corresponding components

The BTS communicates over the JTAG bus to a test design running in the FPGA. The Board Test System and Power Monitor share the JTAG bus with other applications like the Nios II debugger and the SignalTap® II Embedded Logic Analyzer. Because the Quartus II programmer uses most of the bandwidth of the JTAG bus, other applications using the JTAG bus might time out. Be sure to close the other applications before attempting to reconfigure the FPGA using the Quartus II Programmer.

Using the Configure Menu

Use the Configure menu to select the design you want to use. Each design example tests different board features. Choose a design from this menu and the corresponding tabs become active for testing.

Figure 3-2: The Configure Menu



To configure the FPGA with a test system design, perform the following steps:

1. On the **Configure** menu, click the configure command that corresponds to the functionality you wish to test.
2. In the dialog box that appears, click **Configure** to download the corresponding design to the FPGA.
3. When configuration finishes, close the Quartus II Programmer if open. The design begins running in the FPGA. The corresponding GUI application tabs that interface with the design are now enabled.

If you use the Quartus II Programmer for configuration, rather than the Board Test System GUI, you may need to restart the GUI.

The System Info Tab

The System Info tab shows the board's current configuration. The tab displays the JTAG chain, the board's MAC address, the Qsys memory map, and other details stored on the board.

Figure 3-3: The System Info Tab

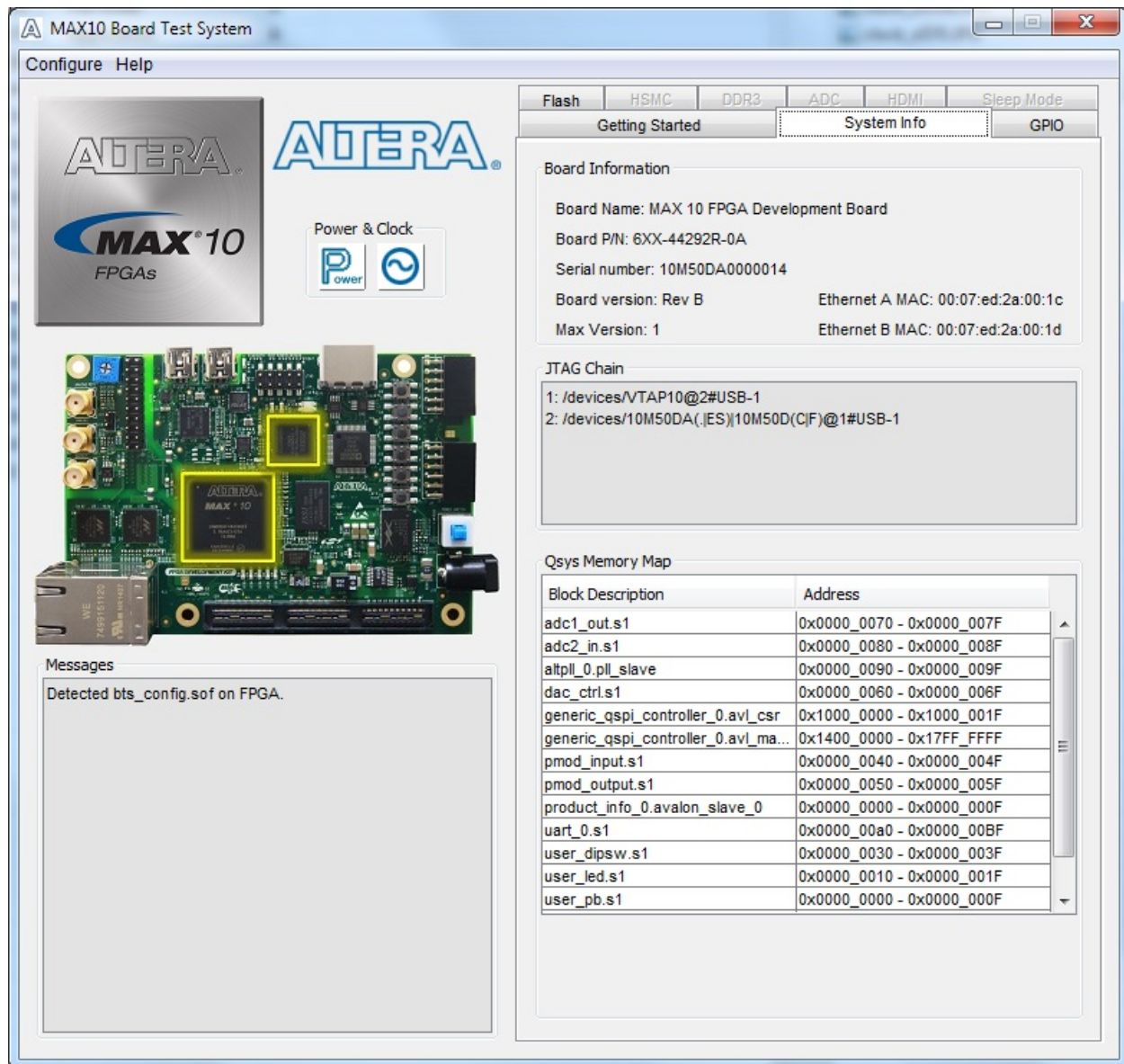


Table 3-1: Controls on the System Info Tab

Controls	Description
Board Information Controls	The board information is updated once the GPIO design is configured. Otherwise, this control displays the default static information about your board.
Board Name	Indicates the official name of the board, given by the Board Test System.
Board P/N	Indicates the part number of the board.
Serial Number	Indicates the serial number of the board.
Factory Test Version	Indicates the version of the Board Test System currently running on the board.
MAX Version	Indicates the version of MAX code currently running on the board.
Ethernet A MAC	Indicates the Ethernet A MAC address of the board.
Ethernet B MAC	Indicates the Ethernet B MAC address of the board.
JTAG Chain	Shows all the devices currently in the JTAG chain.
Qsys Memory Map	Shows the memory map of the Qsys system on your board.

The GPIO Tab

The GPIO tab allows you to interact with all the general purpose user I/O components on your board. You can read DIP switch settings, turn LEDs on or off, and detect push button presses.

Figure 3-4: The GPIO Tab



Table 3-2: Controls on the GPIO Tab

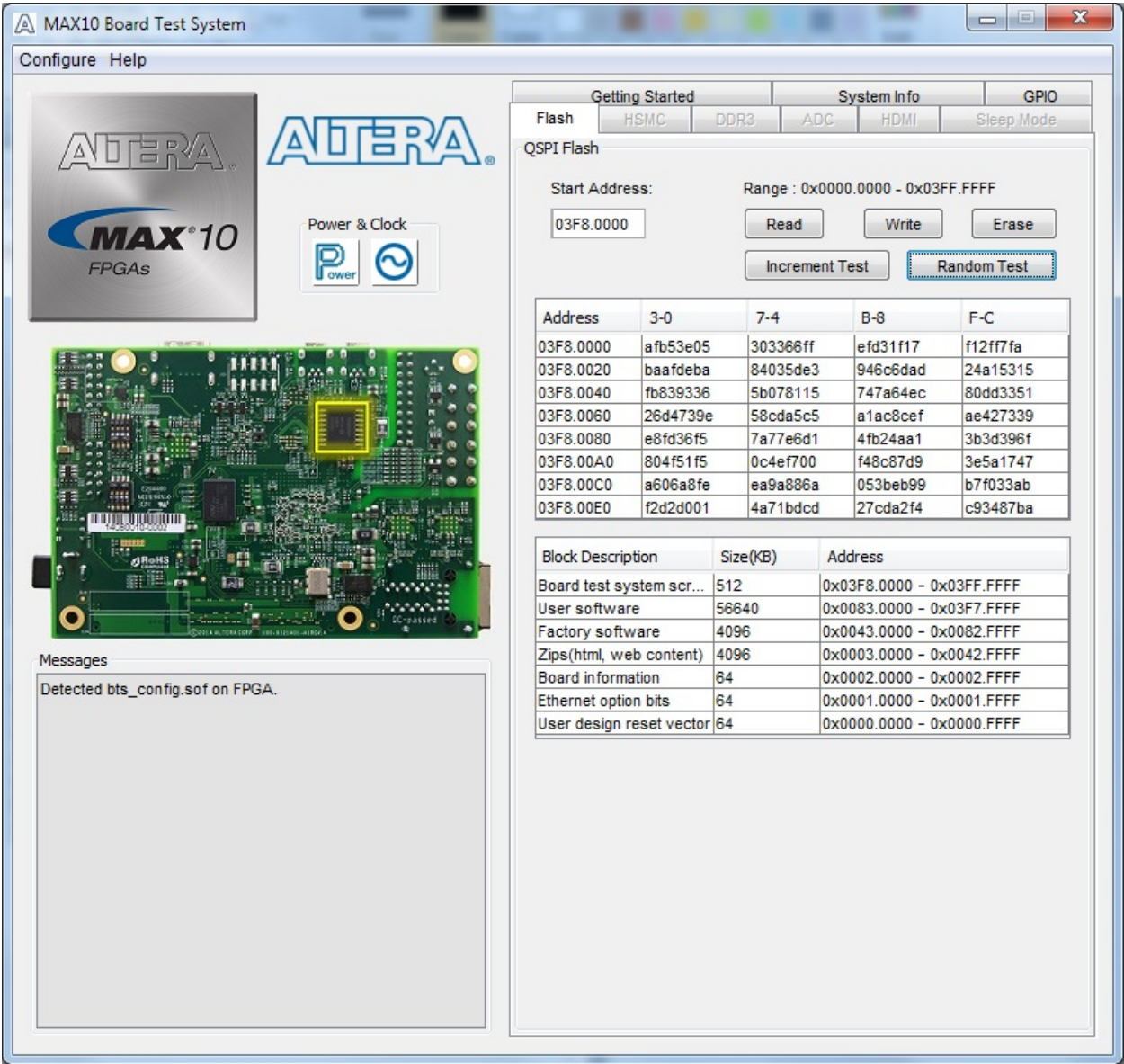
User DIP Switch	Displays the current positions of the switches in the user DIP switch banks. Change the switches on the board to see the graphical display change accordingly.
-----------------	--

User LEDs	Displays the current state of the user LEDs for the FPGA. To toggle the board LEDs, click the 0 to 4 buttons to toggle red or green LEDs, or click the All button.
Push Button Switches	Read-only control displays the current state of the board user push buttons. Press a push button on the board to see the graphical display change accordingly.

The Flash Tab

The **Flash Tab** allows you to read and write flash memory on your board.

Figure 3-5: The Flash Tab (Detail)



Control	Description
Read	Reads the flash memory on your board. To see the flash memory contents, type a starting address in the text box and click Read. Values starting at the specified address appear in the table.

Control	Description
Write	Writes the flash memory on your board. To update the flash memory contents, change values in the table and click Write. The application writes the new values to flash memory and then reads the values back to guarantee that the graphical display accurately reflects the memory contents.
Erase	Erases flash memory.
Increment Test	Starts an incrementing data pattern test to flash memory, limited to the 512 K test system scratch page.
Random Test	Starts a random data pattern test to flash memory, limited to the 512 K test system scratch page.
Flash Memory Map	Displays the flash memory map for the development board.

The HSMC Tab

The **HSMC Tab** allows you to test the CMOS port.

Figure 3-6: The HSMC Tab



Control	Description
Status	Pattern sync: Shows the pattern synced or not synced state. The pattern is considered synced when the start of the data sequence is detected.
Port	CMOS: The CMOS port is available for tests.

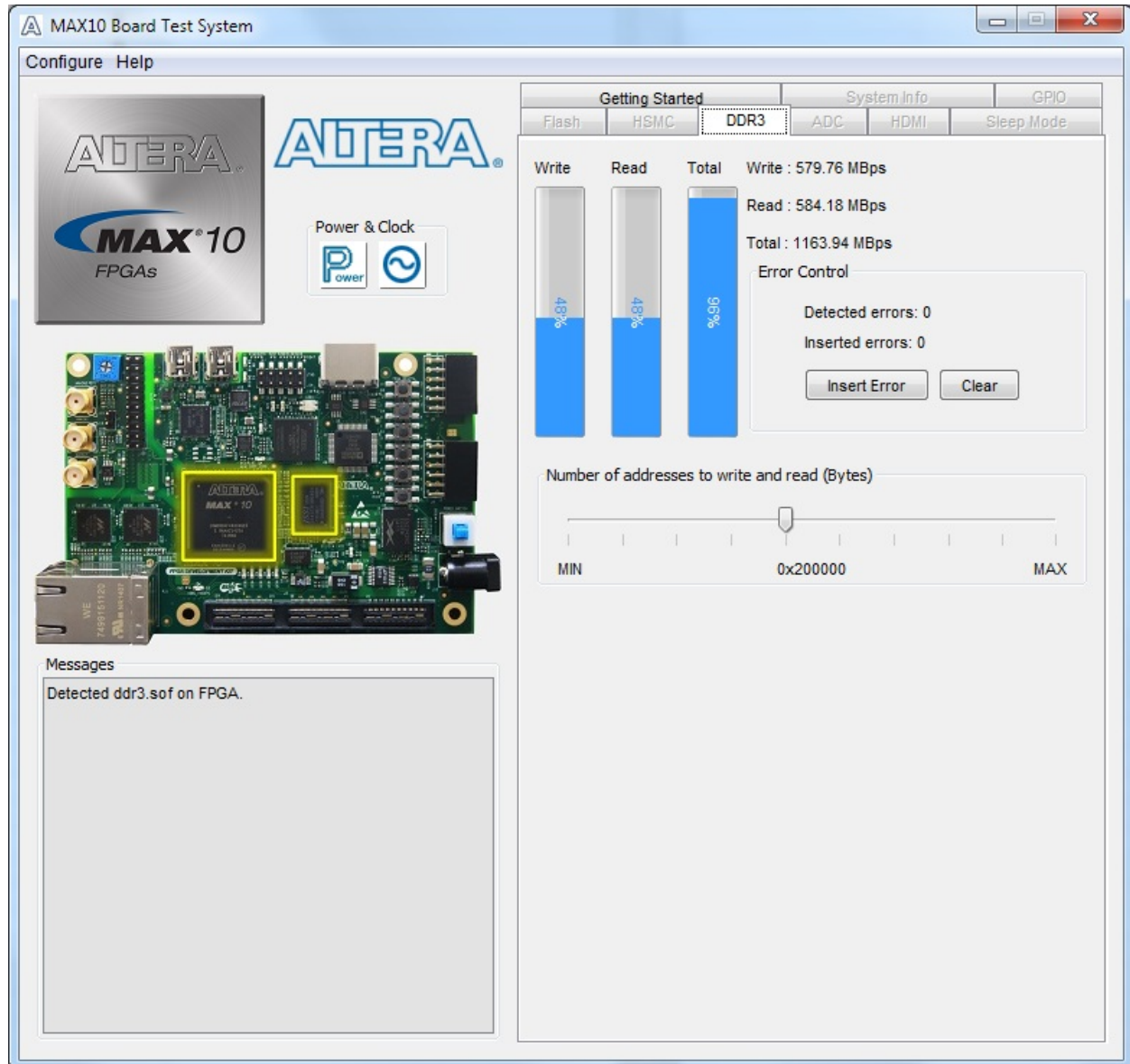
Control	Description
Data Type	<p>The following data types are available for analysis:</p> <ul style="list-style-type: none"> • prbs7: Selects pseudo-random 7-bit sequences. • prbs15: Selects pseudo-random 15-bit sequences. • prbs23: Selects pseudo-random 23-bit sequences. • prbs31: Selects pseudo-random 31-bit sequences. • high_frequency: Divide by data pattern. • low_frequency: Divide by data pattern.
Error Control	<ul style="list-style-type: none"> • Detected errors: Displays the number of data errors detected in the hardware. • Inserted errors: Displays the number of errors inserted into the transmit data stream. • Bit error rate (BER): Displays the bit error rate of the interface • Insert Error: Inserts a one-word error into the transmit data stream each time you click the button. • Clear: Resets the Detected errors and Inserted errors counters to zeroes.
Test Control	<ul style="list-style-type: none"> • Stop: Resets the test. • Number of bits tested: Displays the number of bits tested since the last reset.



The DDR3 Tab

The **DDR3 Tab** allows you to test the DDR3 by reading and writing to a selected amount of addresses.

Figure 3-7: The DDR3 Tab

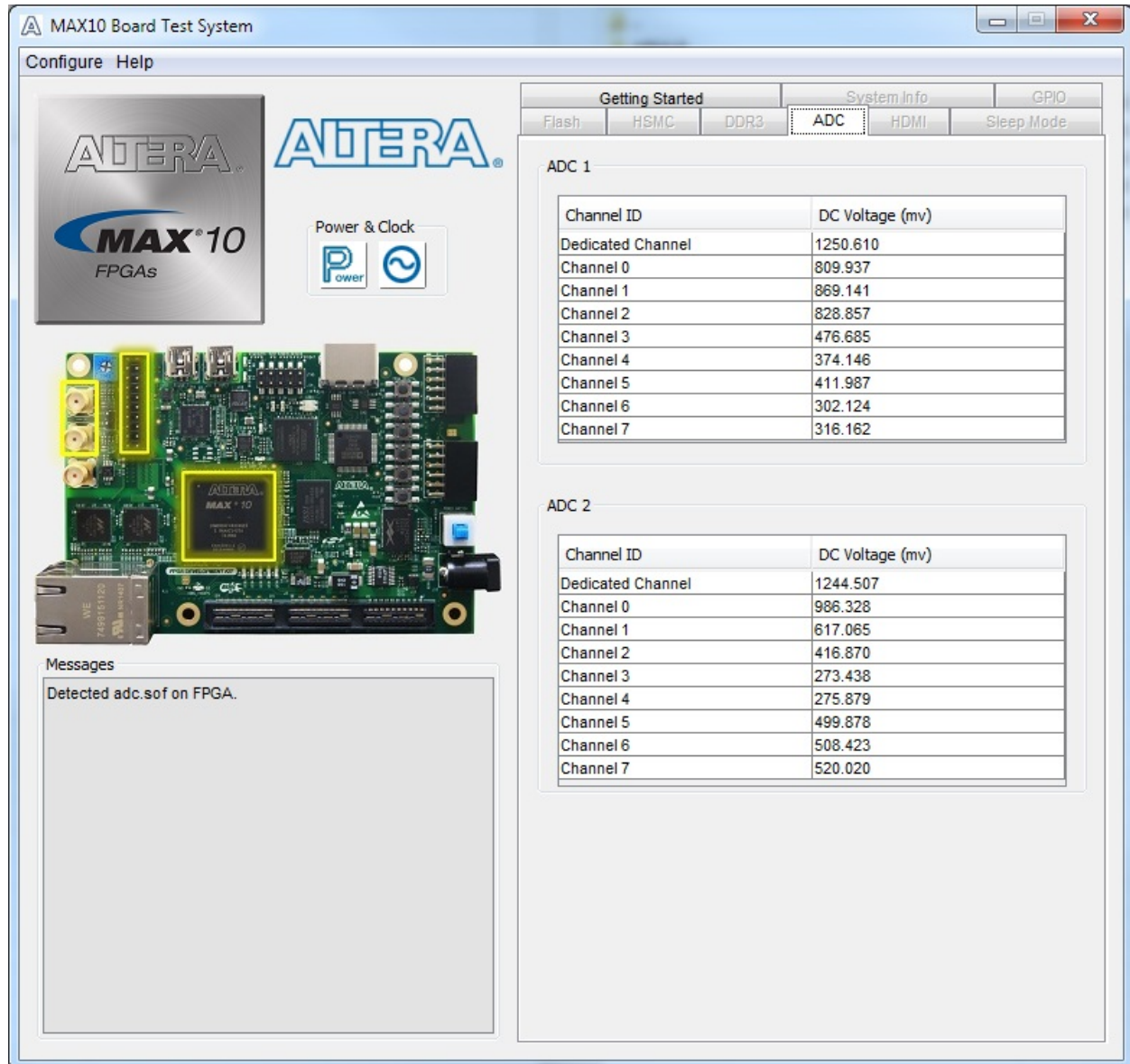


Control	Description
Performance Indicators	<p>These controls display current transaction performance analysis information collected since you last clicked Start:</p> <ul style="list-style-type: none"> • Write, Read, and Total performance bars—Show the percentage of maximum theoretical data rate that the requested transactions are able to achieve. • Write (MBps), Read (MBps), and Total (MBps)—Show the number of bytes of data analyzed per second. • Data bus: 72 bits (8 bits ECC) wide and the frequency is 1066 MHz double data rate. 2133 Megabits per second (Mbps) per pin. Equating to a theoretical maximum bandwidth of 136512 Mbps or 17064 MBps.
Error Control	<p>This control displays data errors detected during analysis and allows you to insert errors:</p> <ul style="list-style-type: none"> • Detected errors—Displays the number of data errors detected in the hardware. • Inserted errors—Displays the number of errors inserted into the transaction stream. • Insert Error—Inserts a one-word error into the transaction stream each time you click the button. Insert Error is only enabled during transaction performance analysis. • Clear—Resets the Detected errors and Inserted errors counters to zeroes.
Number of Addresses to Write and Read	Determines the number of addresses to use in each iteration of reads and writes.

The ADC Tab

The **ADC Tab** (analog-to-digital) shows the real-time voltage values of all of the ADC input channels.

Figure 3-8: The ADC Tab



The two tables displayed on this tab, **ADC 1** and **ADC 2** are not editable. The following table shows where the channels connect to.

Dedicated Channel	SMA Connector
ADC 1	ANAIN1_SMA(J18)
Channel0	ADC1_CH0(J20.1)
Channel1	ADC1_CH1(J20.3)
Channel2	ADC1_CH2(J20.5)
Channel3	ADC1_CH2(J20.7)
Channel4	ADC1_CH4(J20.11)
Channel5	ADC1_CH4(J20.13)
Channel6	ADC1_CH6(J20.15 or POT1)
Channel7	ADC1_CH7(J20.17)

Dedicated Channel	SMA Connector
ADC 2	ANAIN2_SMA(J19)
Channel0	ADC1_CH0(J20.2)
Channel1	ADC1_CH1(J20.4)
Channel2	ADC1_CH2(J20.6)
Channel3	ADC1_CH2(J20.8)
Channel4	ADC1_CH4(J20.12)
Channel5	ADC1_CH4(J20.14)
Channel6	ADC1_CH6(J20.16)
Channel7	ADC1_CH7(J20.18)

The HDMI Tab

This tab displays a transmitter color bar pattern from the high-definition multimedia interface (HDMI).

Figure 3-9: The HDMI Tab



Control	Description
TX Pattern	Color Bar: Use this control to choose TX patterns. The available choices are red, blue, green, white, and black. If you select the Start button, the TX pattern displays immediately.
Start	When you click this button, the selected TX pattern (from Color Bar) displays.

The Sleep Mode Tab

This tab allows you to test the sleep mode aspect of the power management controller.

Figure 3-10: The Sleep Mode Tab (Cropped View)



Control	Description
running (/sleeping)	This control displays the mode status as sleeping or running. It is not interactive.
Note	This control displays board LED events related to the sleep mode.

Related Information

[MAX 10 FPGA Configuration User Guide](#)

Provides detailed explanations and options for MAX 10 device configuration.

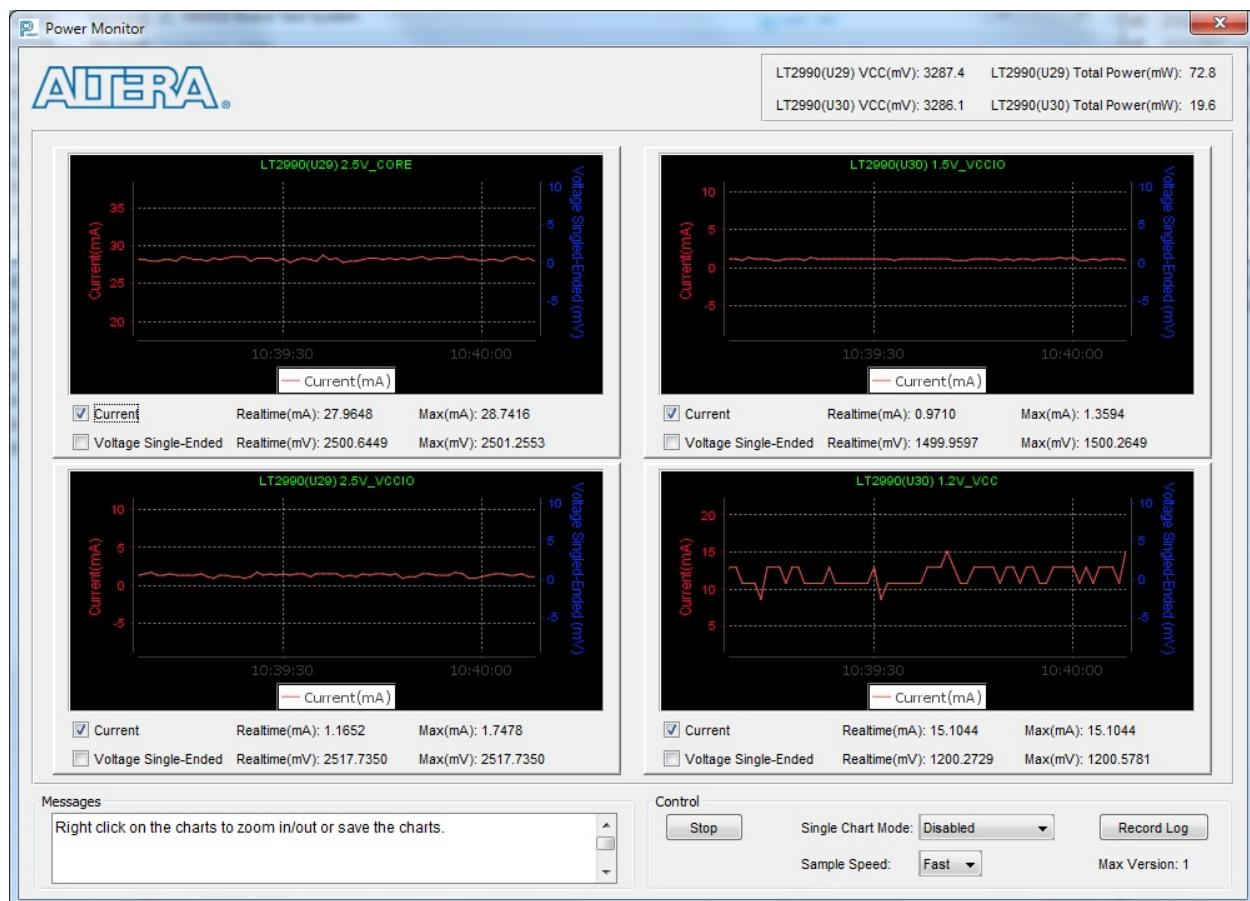
The Power Monitor

The Power Monitor measures and reports current power information and communicates with the MAX II device on the board through the JTAG bus. A power monitor circuit attached to the MAX II device allows you to measure the power that the FPGA is consuming.

To start the application, click the Power Monitor icon in the Board Test System application. You can also run the Power Monitor as a stand-alone application. The `PowerMonitor(32-bit.exe)` and `PowerMonitor(64-bit.exe)` reside in the `<install_dir>\kits\<device_name>\examples\board_test_system` directory.

Note: You cannot run the stand-alone power application and the BTS application at the same time.

Figure 3-11: The Power Monitor



This window displays both LTC2990 current and temperature monitors. The left side top and bottom quadrant shows U29 and the opposite side shows U30. Use the available controls to show **Current** or **Voltage Single-Ended**, or both.

Single Chart Mode allows you to choose how you want the panes to display. You can show only a single large pane, if needed.

Voltage Single-Ended shows the voltage value of each power rail:

- 2.5 V_{CORE}
- 2.5 V_{VCCIO}
- 1.5 V_{VCCIO}
- 1.2 V_{VCC}

Single-ended shows the voltage of SENSE_P only.

The LT2990 also shows a differential voltage value of the sampling resistor SENSE_P and SENSE_N.

Sample Speed allows you to select **Slow** at 5 seconds, or **Fast**: at 1 second (default).

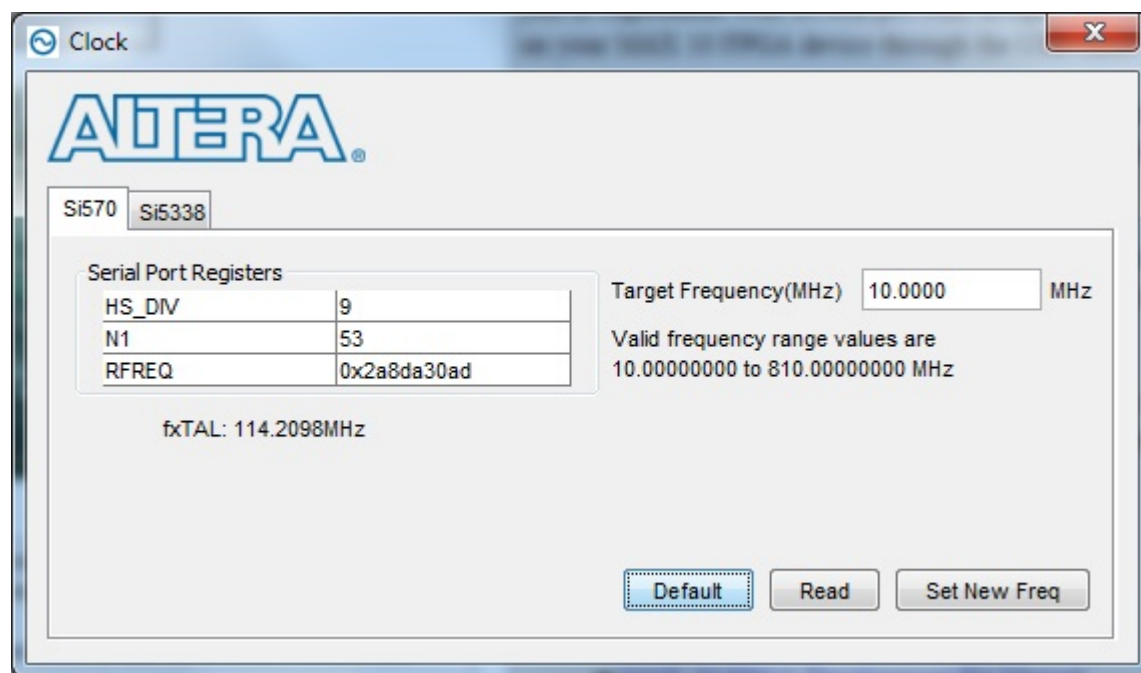
Record Log saves a comma-separated values (CSV) format file `ltc2990.csv` in the `*\examples\board_test_system` directory.

The Clock Control

The MAX 10 FPGA development board Clock Control application sets the programmable oscillators to any frequency between 10 MHz and 810 MHz. The frequencies support eight digits of precision to the right of the decimal point.

The Clock Control communicates with the MAX II device on the board through the JTAG bus. The programmable oscillators are connected to the MAX II device through a 2-wire serial bus.

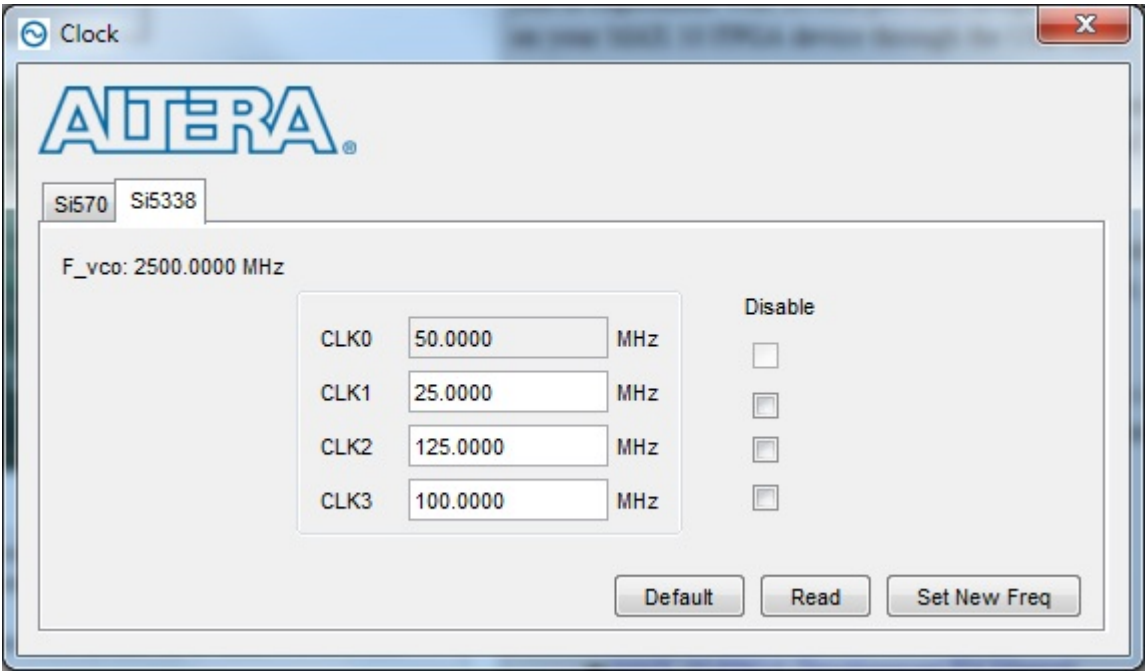
Figure 3-12: The Si570 Tab



Control	Description
Serial Port Registers	Shows the current values from the Si570 registers for frequency configuration.
Target frequency (MHZ)	Allows you to specify the frequency of the clock. Legal values are between 10 and 810 MHz with eight digits of precision to the right of the decimal point. For example, 421.31259873 is possible within 100 parts per million (ppm). The Target frequency control works in conjunction with the Set New Freq control.
fxTAL	Shows the calculated internal fixed-frequency crystal, based on the serial port register values.
Default	Sets the frequency for the oscillator associated with the active tab back to its default value. This can also be accomplished by power cycling the board.
Read	Reads the current frequency setting for the oscillator associated with the active tab.

Control	Description
Set New Freq	Sets the programmable oscillator frequency for the selected clock to the value in the Target frequency control for the programmable oscillators. Frequency changes might take several milliseconds to take effect. You might see glitches on the clock during this time. Altera recommends resetting the FPGA logic after changing frequencies.

Figure 3-13: The Si5338 Tab



Control	Description
F_vco	Displays the generating signal value of the voltage-controlled oscillator.
Registers	Display the current frequencies for each oscillator.
Frequency (MHz)	Allows you to specify the frequency of the clock.
Disable	Disable each oscillators as required.
Read	Reads the current frequency setting for the oscillator associated with the active tab.
Default	Sets the frequency for the oscillator associated with the active tab back to its default value. This can also be accomplished by power cycling the board.

Control	Description
Set New Freq	<p>Sets the programmable oscillator frequency for the selected clock to the value in the CLK0 to CLK3 controls. Frequency changes might take several milliseconds to take effect. You might see glitches on the clock during this time. Altera recommends resetting the FPGA logic after changing frequencies.</p> <p>Note: Changing CLK0 of Si5338 will affect the Clock/Power GUI. One clock from port CLK0 is used to drive the MAX II device which as a 2-wire serial bus interface connected to SI570, Si5338, and the power monitor.</p>

2017.09.07

UG-01169



Subscribe



Send Feedback

This chapter introduces all the important components on the development kit board.

A complete set of schematics, a physical layout database, and GERBER files for the development board reside in the development kit documents directory.

Board Overview

This topic provides a high-level list of the major components of the MAX 10 FPGA development board.

Table 4-1: MAX 10 FPGA Board Components

Board Reference	Type	Description
Featured Devices		
U1	FPGA	MAX MAX 10 FPGA 10M50DAF484C6GES, 50K LEs, F484 package.
U13	CPLD	MAX II EPM1270 256-MBGA, 2.5 V/3.3 V, VCCINT for On-Board USB-Blaster II.
U17	Power Regulator	Enpirion® EN2342QI, PowerSoC voltage-mode synchronous step-down converter with integrated inductor.
U22, U23, U27	Power Regulator	Enpirion EN6337QI, PowerSoC DC-DC step-down converters with integrated inductor.
U26	Power Regulator	Enpirion EP5358LUI, 600 mA PowerSoC DC-DC step-down converters with integrated inductor.
U24, U25	Power Regulator	Enpirion EP5358HUI, 600 mA PowerSoC DC-DC step-down converters with integrated inductor.
Configuration and Setup Elements		
J12	On-Board (Embedded) USB-BlasterBlaster II	Type-B USB connector for programming and debugging the FPGA.

Intel Corporation. All rights reserved. Intel, the Intel logo, Altera, Arria, Cyclone, Enpirion, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.

ISO
9001:2015
Registered

Board Reference	Type	Description
J14	10-pin header	Optional JTAG direct via 10-pin header for external download cables.
J20	2x10-pin header	16 dual-purpose ADC channels are connected to the 2x10 header.
SW2	DIP configuration and user switch	SW2 Includes switches to control boot images, JTAG bypass and HSMC bypass.
J7	Jumper for the MAX 10 ADC	Connects potentiometer for providing adjustable voltage to the ADC.
S5	Pulse_nconfig push button	Emulates pulsing the nCONFIG pin low to trigger reconfiguration even though the physical pin is unaffected.
S6	CPU reset push button	Default reset for the FPGA logic.
Status Elements		
D1	Blue power LED	Illuminates when 12-V power is present.
D2	Green high-speed mezzanine card (HSMC) LED	Illuminates when the HSMC is present.
D13, D14	Green USB-UART LEDs	Illuminates when the USB-UART transmitter and receiver are in use.
D20	Configuration done LED	Illuminates when the FPGA is configured.
D21, D22, D23	Power LEDs	Indicates that 3.3 V, 2.5 V, 1.2 V are powered up successfully.
Clock Circuitry		
X1	Programmable Clock for ADC	Programmable oscillator for ADC with default frequency of 10 MHz.
U2	Programmable Clock	Four channel programmable oscillator with default frequencies of 25, 50, 100, 125 MHz.
General User Input/Output		
S1, S2, S3, S4	General user push buttons	Four user push buttons. Driven low when pressed.
D15, D16, D17, D18, D19	User LEDs	Four user LEDs. Illuminates when driven low.
SW1, SW2.1	User DIP switches	Quad user DIP switches.
Memory Devices		
U5	DDR3 SDRAM A memory	64 Mx16.
U6	DDR3 SDRAM B memory	128 Mx8.
U7	Quad serial peripheral interface (quad SPI) flash	512 Mb.
Communication Ports		

Board Reference	Type	Description
J2	HSMC port	Provides 84 CMOS or 17 LVDS channels per HSMC specification.
U9, U10	Two Gigabit Ethernet ports <ul style="list-style-type: none"> Ethernet A (Bottom) Ethernet B (Top) 	RJ-45 connector which provides a 10/100/1000 Ethernet connection via a Marvell 88E1111 x 2 PHY and the FPGA-based Altera Triple Speed Ethernet MegaCore function in RGMII mode.
J4, J5	Two Diligent Pmod connectors	12-pin interface with 8 I/O signal pins used to connect low frequency, low I/O peripheral modules.
J11	Mini-USB 2.0 UART port	USB connector with USB-to-UART bridge for serial UART interface
J12	Mini-USB port	Embedded USB-Blaster II.
Analog		
J18, J19	SMA inputs	Two FPGA analog-to-digital converter (ADC).
J20	Header	2x10 ADC.
POT1	Potentiometer	Input to ADC.
J1	SMA output	External 16 bit digital-to-analog converter (DAC) device.
Video and Display Ports		
U8	HDMI video output	19-pin HDMI connector which provides a HDMI v1.4 video output of up to 1080p through an ADI (Analog Devices, Inc) PHY.
Power Supply		
J15	DC input jack	Accepts 12 V DC power supply.
SW3	Power switch	Switch to power on or off the board when power is supplied from the DC input jack.

Featured Device

The MAX 10 FPGA development board features the MAX 10 10M50DAF484C6GES device (U1) in a 484-pin FineLine BGA package.

Table 4-2: MAX 10 FPGA 10M50DAF484C6GES Features

ALMs	Equivalent LEs	M9K Memory (Kb)	Total RAM (Kb)	18-bit x 18-bit Multipliers	PLLs	Transceivers	Package Type
50,000	50	1,638	736	144	4	—	FineLine BGA 484 pins

Configuration

The MAX 10 FPGA development kit supports two configuration methods:

- Configuration by downloading a **.sof** file to the FPGA. Any power cycling of the FPGA or reconfiguration will power up the FPGA to a blank state.
- Programming of the on-die FPGA Configuration Flash Memory (CFM) via a **.pof** file. Any power cycling of the FPGA or reconfiguration will power up the FPGA in self-configuration mode, using the files stored in the CFM.

You can use two different USB-Blaster™ hardware components to program the **.sof** or **.pof** files:

- Embedded USB-Blaster II, type-B connector (J12).
- JTAG header (J14). Use an external USB-Blaster, USB-Blaster II, or Ethernet Blaster download cable. The external download cable connects to the board through the JTAG header (J14).

Using the Quartus II Programmer

You can use the Quartus II Programmer to configure the FPGA with a **.sof**.

Before configuring the FPGA:

- Ensure that the Quartus II Programmer and the USB-Blaster driver are installed on the host computer
- The USB cable is connected to the kit
- Power to the board is on, and no other applications that use the JTAG chain are running

To configure the MAX 10 FPGA:

1. Start the Quartus II Programmer.
2. Click **Add File** and select the path to the desired **.sof**.
3. Turn on the **Program/Configure** option for the added file.
4. Click **Start** to download the selected file to the FPGA. Configuration is complete when the progress bar reaches 100%.

The Quartus II Convert Programming File (CPF) GUI can be used to generate a **.sof** file that can use for internal configuration. You can directly program the MAX 10 device's flash which included Configuration Flash Memory (CFM) and User Flash Memory (UFM) by using a download cable with the Quartus II software programmer.

Selecting the Internal Configuration Scheme

For all MAX 10 devices, except 10M02 device, there are total of 5 different modes you can select internal configuration.

The internal configuration scheme needs to be selected before design compilation.

To select the configuration mode:

1. Open the Quartus II software and load a project using MAX 10 device family.
2. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.
3. In the Category list, select **Device**. The Device page appears.
4. Click **Device and Pin Options**.
5. In the **Device and Pin Options** dialog box, click the **Configuration** tab.

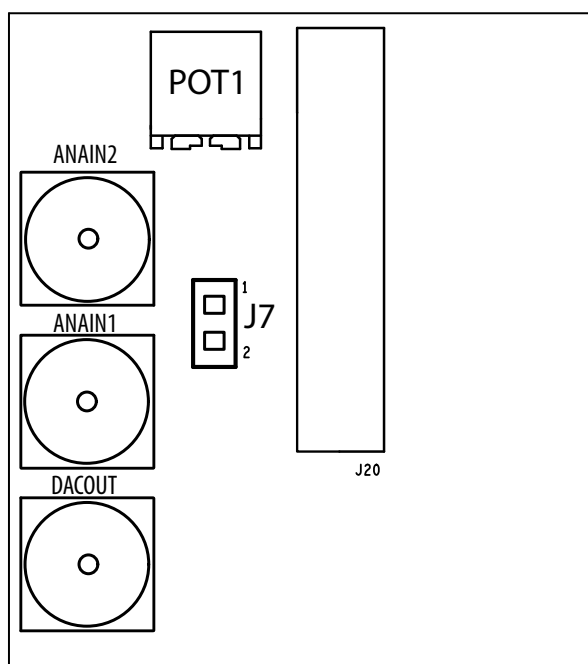
6. In the **Configuration Scheme** list, select **Internal Configuration**.
7. In the **Configuration Mode** list, select 1 out of 5 configuration modes. For the dual-boot feature:
 - a. Must have a Dual Boot IP core in the design, for example, in a Qsys component.
 - b. Choose **Dual Compressed Images (512 Kbits UFM)** for the **Configuration Mode**.
 - c. Generate two sof files above and convert them into one POF file for CFM programming.
8. Turn on Generate compressed bitstreams if needed, and click **OK**.

Switch and Jumper Settings

This topic is for the MAX 10 FPGA development kit. This topic shows you how to restore the default factory settings and explains their functions.

The J7 jumper connects the output of potentiometer (POT1.2) to ADC1_CH6. When J7 jumper is on, you can use the potentiometer to provide adjustable voltage (0~2.5 V) to the MAX 10 ADC through ADC1_CH6. When J7 jumper is off, ADC1_CH6 is connected to the 2x10 header as the other ADC channels.

Figure 4-1: Jumper J7 on the Top of the Board (Detail)

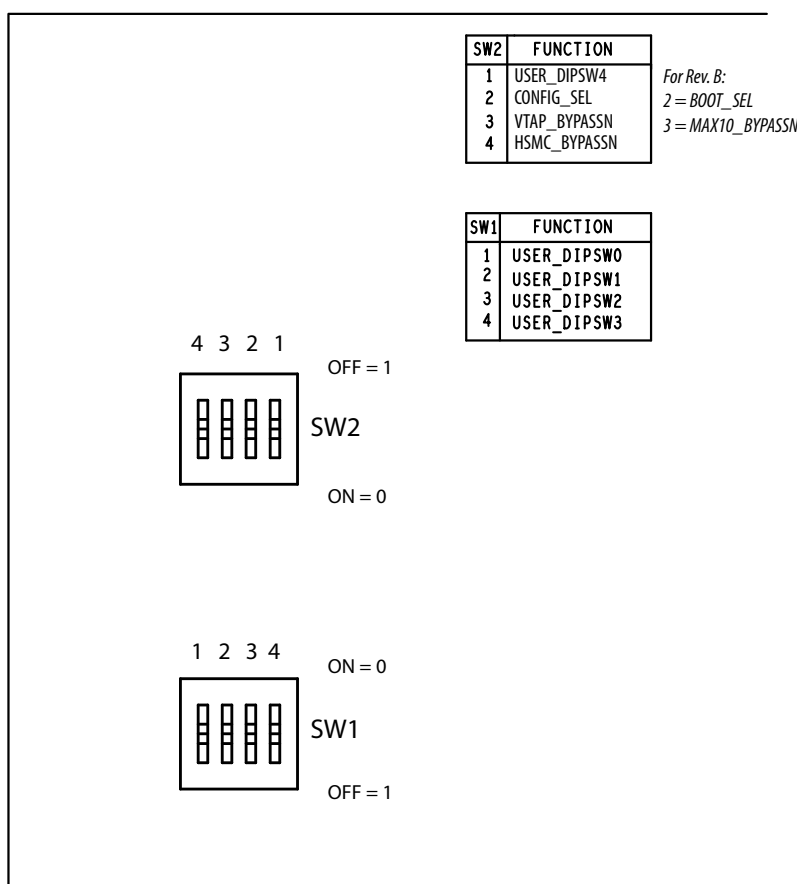


There are two switches on the bottom of the board. SW1 is for user functions, and SW2 allows for booting selection and bypassing some components.

Figure 4-2: Switches on the Bottom Board (Detail)

When a switch is ON, it means the FUNCTION SIGNAL is connected to GND. So it is a LOGIC LOW (0). When switch is OFF, it means the FUNCTION SIGNAL is disconnected from GND. So it is a LOGIC HIGH (1).

Note: The following figure shows the switch labels for the Rev. C board and a note for the Rev. B board. The change of name for SW2.3 is just a name change, not a functional change. Rev. B is labeled MAX10_BYPASSN, but it is actually a VTAP bypass.

**Table 4-3: SW2 DIP Switch Settings (Board Bottom)**

Switch	Board Label	Function	Default Position
1	USER_DIPSW4	User defined switch #4, #0/1/2/3 is on SW1. No default function.	—
2	BOOT_SEL (for Rev. B Board) CONFIG_SEL (for Rev. C board)	Use this pin to choose CFM0, CFM1 or CFM2 image as the first boot image in dual-image configuration. If the CONFIG_SEL is set to low, the first boot image is CFM0 image. If the CONFIG_SEL pin is set to high, the first boot image is CFM1 or CFM2 image. This pin is read before user mode and before the nSTATUS pin is asserted.	LOW

Switch	Board Label	Function	Default Position
3	VTAP_BYPASSN	A virtual JTAG device is provided within the On-Board USB-Blaster II, it provides access to diagnostic hardware and board identification information. The device shows up as an extra device on the JTAG chain with ID: 020D10DD. This switch removes the virtual JTAG device from the JTAG chain.	HIGH
4	HSMC_BYPASSN	Use this pin to bypass HSMC from JTAG chain. The default value of this signal is high so HSMC is in the JTAG chain. (However, there is no daughter cards connected to HSMC normally so it would not be detected by JTAG master). When it is set to low, HSMC is bypassed.	HIGH

Status Elements

This topic lists the non-user status elements for the MAX 10 FPGA development board.

Table 4-4: General LED Signal Names

Board Reference	Signal Name	Description
D1	—	Blue Power LED
D2	HSMC_PRSENTn	Green LED
D13	UART_TXLED	Green LED for USB to UART
D14	UART_RXLED	Green LED for USB to UART

Table 4-5: MAX II CPLD LED Signal Names

Board Reference	Signal Name	I/O Standard	MAX II CPLD Pin Number
D20	MAXII_CONF_DONE	3.3 V	W17
D21	3.3V_LED	3.3 V	U4
D22	2.5V_LED	3.3 V	U5
D23	1.2V_LED	3.3 V	U6

Setup Elements

Table 4-6: Board Settings DIP Switch and Jumper Schematic Signals

Board Reference	Signal Name	Device / Pin Number	I/O Standard
SW2.1	USER_DIPSW4	MAX 10 / H21	1.5 V
SW2.2	CONFIG_SEL	MAX 10 / H10	3.3 V
SW2.3	VTAP_BYPASSN	MAX II / P17	3.3 V
SW2.4	HSMC_BYPASSn	MAX II / P18	3.3 V
J7.1	—	POT1	2.5 V
J7.2	ADC1_CH6	2x10 Header / J20.15	2.5 V

Table 4-7: General Push Button Signal Names

Board Reference	Signal Name	MAX 10 FPGA Pin Number	I/O Standard
S5	PULSE_NCONFIG	H9	3.3 V
S6	CPU_RESETh	D9	3.3 V

General User Input/Output

User-defined I/O signal names, FPGA pin numbers, and I/O standards for the MAX 10 FPGA development board.

Table 4-8: User-Defined Push Button Signal Names

Board Reference	Signal Name	MAX 10 FPGA Pin Number	I/O Standard
S1	USER_PB0	L22	1.5 V
S2	USER_PB1	M21	1.5 V
S3	USER_PB2	M22	1.5 V
S4	USER_PB3	N21	1.5 V

Table 4-9: User-Defined DIP Switch Schematic Signal Names

Board Reference	Signal Name	MAX 10 FPGA Pin Number	I/O Standard
SW1.1	USER_DIPSW0	H21	1.5 V

Board Reference	Signal Name	MAX 10 FPGA Pin Number	I/O Standard
SW1.2	USER_DIPSW1	H22	1.5 V
SW1.3	USER_DIPSW2	J21	1.5 V
SW1.4	USER_DIPSW3	J22	1.5 V
SW2.1	USER_DIPSW4	G19	1.5 V

Table 4-10: User LED (Green) Schematic Signal Names

Board Reference	Signal Name	MAX 10 FPGA Pin Number	I/O Standard
D15	USER_LED0	T20	1.5 V
D16	USER_LED1	U22	1.5 V
D17	USER_LED2	U21	1.5 V
D18	USER_LED3	AA21	1.5 V
D19	USER_LED4	AA22	1.5 V

For a MAX 10 Development Kit Baseline Pinout design visit the Altera Design Store.

Related Information

[Altera Design Store \(MAX 10 Development Kit\)](#)

Clock Circuitry

The development board includes a four channel programmable oscillator with default frequency of 25-MHz, 50-MHz, 100-MHz, 125-MHz. The board also includes a 10-MHz programmable oscillator connected to the ADC.

On-Board Oscillators

Figure 4-3: MAX 10 FPGA Development Board Clocks

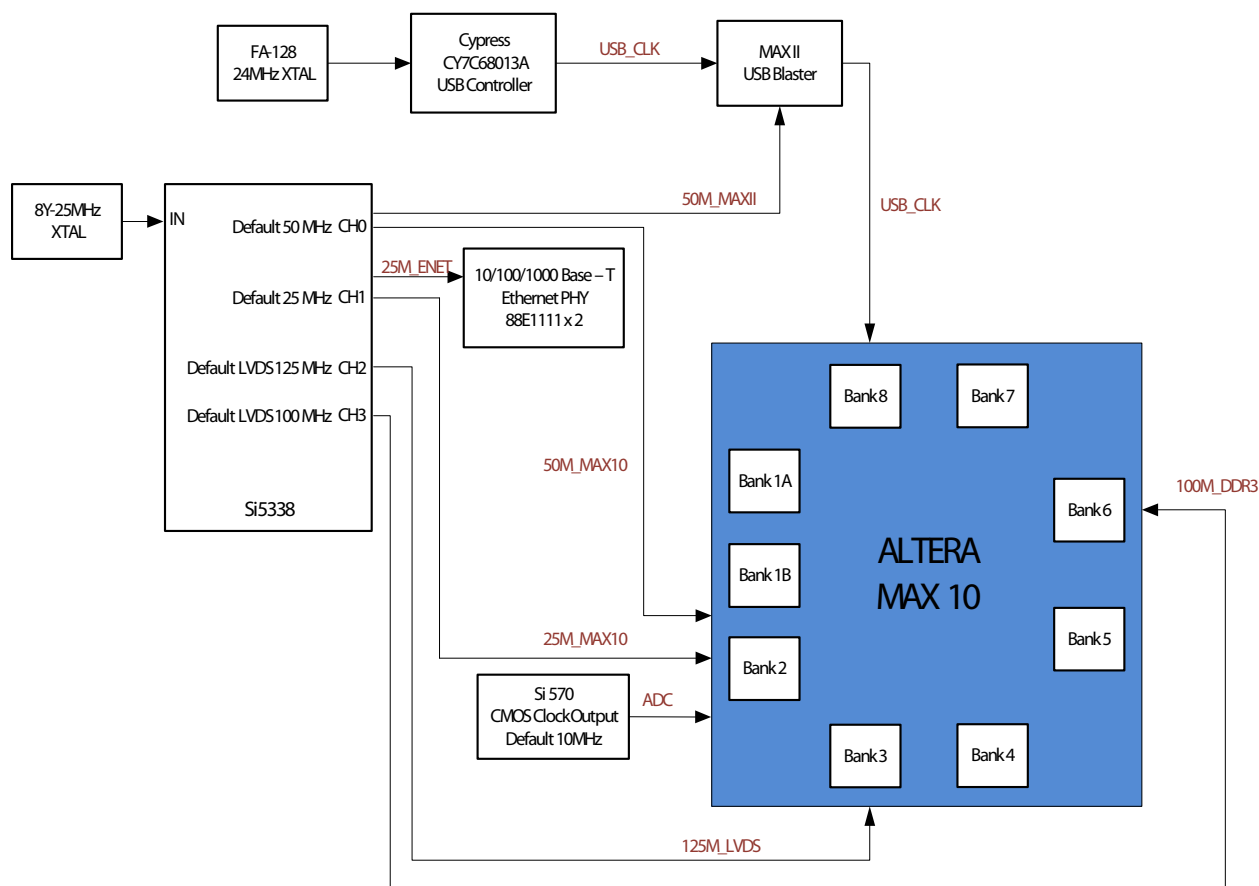


Table 4-11: On-Board Oscillators

Source	Schematic Signal Name	Frequency	I/O Standard	MAX 10 FPGA Pin Number	Application
X1	CLK_10_ADC	10.000 MHz	2.5 V CMOS	N5	Programmable default 10MHz clock for ADC
U2	CLK_25_ENET	25.000 MHz	2.5 V CMOS	-	Ethernet clock
U2	CLK_25_MAX10	25.000 MHz	2.5 V CMOS	M8	MAX 10 clock
U2	CLK_50_MAXII	25.000 MHz	2.5 V/3.3V CMOS	-	Clock for On-Board USB Blaster II
U2	CLK_50_MAX10	50.000 MHz	2.5 V CMOS	M9	MAX 10 clock

Source	Schematic Signal Name	Frequency	I/O Standard	MAX 10 FPGA Pin Number	Application
U2	CLK_DDR3_100_N	100.000 MHz	Differential SSTL-15	N15	DDR3 clocks
U2	CLK_DDR3_100_P	100.000 MHz	Differential SSTL-15	N14	DDR3 clocks
U2	CLK_LVDS_125_N	125.000 MHz	2.5 V LVDS	R11	LVDS clocks
U2	CLK_LVDS_125_P	125.000 MHz	2.5 V LVDS	P11	LVDS clocks

Note: For signal CLK_50_MAXII, the output side voltage is 2.5V and the input side voltage is 3.3V. However, they are compatible electrically.

Note: For signals CLK_DDR3_100_P and CLK_DDR3_100_N, at the MAX 10 input side, Differential SSTL-15 is used as I/O standard because this bank's VCCIO is 1.5V.

Off-Board Clock Input/Output

The development board has input and output clocks which can be driven onto the board. The output clocks can be programmed to different levels and I/O standards according to the FPGA device's specification.

Table 4-12: Off-Board Clock Inputs

Source	Schematic Signal Name	I/O Standard	MAX 10 FPGA Pin Number	Description
HSMC	HSMC_CLK_IN_N1	2.5 V	AB21	LVDS input from the installed HSMC cable or board.
HSMC	HSMC_CLK_IN_P1	2.5 V	AA20	LVDS input from the installed HSMC cable or board.
HSMC	HSMC_CLK_IN_N2	2.5 V	V9	LVDS input from the installed HSMC cable or board.
HSMC	HSMC_CLK_IN_P2	2.5 V	V10	LVDS input from the installed HSMC cable or board.
HSMC	HSMC_CLK_IN0	2.5 V	N4	Single-ended input from the installed HSMC cable or board.

Table 4-13: Off-Board Clock Outputs

Source	Schematic Signal Name	I/O Standard	MAX 10 FPGA Pin Number	Description
HSMC	HSMC_CLK_OUT_N1	2.5 V	R13	LVDS output.
HSMC	HSMC_CLK_OUT_P1	2.5 V	P13	LVDS output.
HSMC	HSMC_CLK_OUT_N2	2.5 V	V14	LVDS output.
HSMC	HSMC_CLK_OUT_P2	2.5 V	W15	LVDS output.
HSMC	HSMC_CLK_OUT0	2.5 V	AA13	FPGA CMOS output (or GPIO)

Components and Interfaces

This section describes the development board's communication ports and interface cards relative to the MAX 10 FPGA device.

10/100/1000 Ethernet PHY

The MAX 10 FFPGA development kit supports 10/100/1000 base-T Ethernet using an external Marvell 88E1111 PHY and Altera Triple-Speed Ethernet MegaCore MAC function.

Table 4-14: Ethernet PHY A Pin Assignments, Signal Names and Functions

Board Reference (U9)	Schematic Signal Name	Max 10 FPGA Pin Number	I/O Standard	Description
U9.8	ENETA_GTX_CLK	T5	2.5V CMOS	125 MHz RGMII TX clock
U9.4	ENETA_TX_CLK	E10	3.3V LVCMOS	25/2.5 MHz MII TX clock
U9.11	ENETA_TX_D0	R5	2.5V CMOS	RGMII TX data 0
U9.12	ENETA_TX_D1	P5	2.5V CMOS	RGMII TX data 1
U9.14	ENETA_TX_D2	W1	2.5V CMOS	RGMII TX data 2
U9.16	ENETA_TX_D3	W2	2.5V CMOS	RGMII TX data 3
U9.9	ENETA_TX_EN	R4	2.5V CMOS	RGMII TX enable
U9.7	ENETA_TX_ER	P4	2.5V CMOS	II TX error
U9.2	ENETA_RX_CLK	P3	2.5V CMOS	RGMII RX clock
U9.95	ENETA_RX_D0	N9	2.5V CMOS	RGMII RX data 0
U9.92	ENETA_RX_D1	T1	2.5V CMOS	RGMII RX data 1
U9.93	ENETA_RX_D2	N1	2.5V CMOS	RGMII RX data 2
U9.91	ENETA_RX_D3	T3	2.5V CMOS	RGMII RX data 3

Board Reference (U9)	Schematic Signal Name	Max 10 FPGA Pin Number	I/O Standard	Description
U9.94	ENETA_RX_DV	T2	2.5V CMOS	RGMII RX valid
U9.3	ENETA_RX_ER	U2	2.5V CMOS	MII RX error
U9.28	ENETA_RESETN	V8	2.5V CMOS	Device reset
U9.23	ENETA_INTn	V7	2.5V CMOS	Management bus interrupt
U9.25	ENET_MDC	Y6	2.5V CMOS	MDI clock
U9.24	ENET_MDIO	Y5	2.5V CMOS	MDI data
U9.84	ENETA_RX_CRS	N8	2.5V CMOS	MII Carrier Sense
U9.83	ENETA_RX_COL	P1	2.5V CMOS	MII Collision
U9.55	CLK_25_ENET	—	2.5V CMOS	25 MHz Reference clock
U9.70	ENETA_LED_DUPLEX	—	2.5 V CMOS	Duplex or collision LED
U9.76	ENETA_LED_LINK10	—	2.5 V CMOS	10 Mb link LED
U9.74	ENETA_LED_LINK100	R9	2.5V CMOS	100 Mb link LED
U9.73	ENETA_LED_LINK1000	—	2.5V CMOS	1000 Mb link LED
U9.58, 69	ENETA_LED_RX	—	2.5V CMOS	RX data active LED
U9.61, 68	ENETA_LED_TX	—	2.5V CMOS	TX data active LED
U9.29	ENETA_MDI_P0	—	2.5V CMOS	MDI
U9.31	ENETA_MDI_N0	—	2.5V CMOS	MDI
U9.33	ENETA_MDI_P1	—	2.5V CMOS	MDI
U9.34	ENETA_MDI_N1	—	2.5V CMOS	MDI
U9.39	ENETA_MDI_P2	—	2.5V CMOS	MDI
U9.41	ENETA_MDI_N2	—	2.5V CMOS	MDI
U9.42	ENETA_MDI_P3	—	2.5V CMOS	MDI
U9.43	ENETA_MDI_N3	—	2.5V CMOS	MDI

Table 4-15: Ethernet PHY B Pin Assignments, Signal Names and Functions

Board Reference (U10)	Schematic Signal Name	Max 10 FPGA Pin Number	I/O Standard	Description
U10.8	ENETB_GTX_CLK	T6	2.5V CMOS	125 MHz RGMII TX clock
U10.4	ENETB_TX_CLK	E11	3.3V LVC MOS	25/2.5 MHz MII TX clock
U10.11	ENETB_TX_D0	U1	2.5V CMOS	RGMII TX data 0
U10.12	ENETB_TX_D1	V1	2.5V CMOS	RGMII TX data 1
U10.14	ENETB_TX_D2	U3	2.5V CMOS	RGMII TX data 2
U10.16	ENETB_TX_D3	U4	2.5V CMOS	RGMII TX data 3
U10.9	ENETB_TX_EN	V3	2.5V CMOS	RGMII TX enable
U10.7	ENETB_TX_ER	U5	2.5V CMOS	II TX error
U10.2	ENETB_RX_CLK	R3	2.5V CMOS	RGMII RX clock
U10.95	ENETB_RX_D0	P8	2.5V CMOS	RGMII RX data 0
U10.92	ENETB_RX_D1	M1	2.5V CMOS	RGMII RX data 1
U10.93	ENETB_RX_D2	M2	2.5V CMOS	RGMII RX data 2
U10.91	ENETB_RX_D3	R7	2.5V CMOS	RGMII RX data 3
U10.94	ENETB_RX_DV	R1	2.5V CMOS	RGMII RX valid
U10.3	ENETB_RX_ER	R2	2.5V CMOS	II RX error
U10.28	ENETB_RESETh	AB4	2.5V CMOS	Device reset
U10.23	ENETB_INTn	AA3	2.5V CMOS	Management bus interrupt
U10.25	ENET_MDC	Y6	2.5V CMOS	MDI clock
U10.24	ENET_MDIO	Y5	2.5V CMOS	MDI data
U10.84	ENETB_RX_CR	N3	2.5V CMOS	II Carrier Sense
U10.83	ENETB_RX_COL	N2	2.5V CMOS	II Collision
U10.55	CLK_25_ENET	—	2.5V CMOS	25 MHz Reference clock
U10.70	ENETB_LED_DUPLEX	—	2.5V CMOS	Duplex or collision LED
U10.76	ENETB_LED_LINK10	—	2.5V CMOS	10 Mb link LED
U10.74	ENETB_LED_LINK100	P9	2.5V CMOS	100 Mb link LED

Board Reference (U10)	Schematic Signal Name	Max 10 FPGA Pin Number	I/O Standard	Description
U10.73	ENETB_LED_LINK1000	—	2.5V CMOS	1000 Mb link LED
U10.58, 69	ENETB_LED_RX	—	2.5V CMOS	RX data active LED
U10.61, 65, 68	ENETB_LED_TX	—	2.5V CMOS	TX data active LED
U10.29	ENETB_MDI_P0	—	2.5V CMOS	MDI
U10.31	ENETB_MDI_N0	—	2.5V CMOS	MDI
U10.33	ENETB_MDI_P1	—	2.5V CMOS	MDI
U10.34	ENETB_MDI_N1	—	2.5V CMOS	MDI
U10.39	ENETB_MDI_P2	—	2.5V CMOS	MDI
U10.41	ENETB_MDI_N2	—	2.5V CMOS	MDI
U10.42	ENETB_MDI_P3	—	2.5V CMOS	MDI
U10.43	ENETB_MDI_N3	—	2.5V CMOS	MDI

Digital-to-Analog Converter

The MAX 10 FPGA comes with one external 16 bit digital-to-analog converter (DAC) device with an SMA output.

The MAX 10 FPGA has two 12-bit successive approximation register (SAR) ADCs with sample rate of 1 MSps. One potentiometer is connected to ADC1_CH6 to function as a user-controlled DC, and it is connected to 2.5 V. To ensure performance evaluation of the ADCs, the MAX 10 development kit has separate analog supply and split partition for analog ground. An external 16-bit single channel DAC is connected to Bank 7 to enable closed loop evaluation. The DAC uses a 3-wire serial interface that operates at clock rates up to 30 MHz. It is compatible with standard serial peripheral interface (SPI), quad SPI, Microwire, and digital signal processor (DSP) interfaces.

Table 4-16: Digital-to-Analog Converter Signals

Board Reference (U33)	Signal Name	MAX 10 FPGA Pin Number	I/O Standard	Description
U33.5	DAC_SYNC	U1.B10	3.3 V	Level-triggered control input (active LOW). Frame synchronization signal for the input data.
U33.6	DAC_SCLK	A7	3.3 V	Serial clock input
U33.7	DAC_DIN	A8	3.3 V	Serial data input

HDMI Video Output

The MAX 10 FPGA development kit supports one HDMI transmitter and one HDMI receptacle.

The transmitter incorporates HDMI v1.4 features, and is capable of supporting an input data rate up to 165 MHz (1080p @60Hz, UXGA @60Hz). The connection between HDMI transmitter and MAX 10 is established in Bank 7, and the communication can be done via I²C interface.

Table 4-17: HDMI Pin Assignments, Signal Names and Functions

Board Reference (U8)	Schematic Signal Name	Max 10 FPGA Pin Number	I/O Standard	Description
U8.62	HDMI_TX_D0	A17	3.3 V	HDMI digital video data bus
U8.61	HDMI_TX_D1	A18	3.3 V	HDMI digital video data bus
U8.60	HDMI_TX_D2	A12	3.3 V	HDMI digital video data bus
U8.59	HDMI_TX_D3	F16	3.3 V	HDMI digital video data bus
U8.58	HDMI_TX_D4	A16	3.3 V	HDMI digital video data bus
U8.57	HDMI_TX_D5	B12	3.3 V	HDMI digital video data bus
U8.56	HDMI_TX_D6	F15	3.3 V	HDMI digital video data bus
U8.55	HDMI_TX_D7	B11	3.3 V	HDMI digital video data bus
U8.54	HDMI_TX_D8	A13	3.3 V	HDMI digital video data bus
U8.52	HDMI_TX_D9	C15	3.3 V	HDMI digital video data bus
U8.50	HDMI_TX_D10	C11	3.3 V	HDMI digital video data bus
U8.49	HDMI_TX_D11	A11	3.3 V	HDMI digital video data bus
U8.48	HDMI_TX_D12	A20	3.3 V	HDMI digital video data bus
U8.47	HDMI_TX_D13	H13	3.3 V	HDMI digital video data bus
U8.46	HDMI_TX_D14	E14	3.3 V	HDMI digital video data bus

Board Reference (U8)	Schematic Signal Name	Max 10 FPGA Pin Number	I/O Standard	Description
U8.45	HDMI_TX_D15	D12	3.3 V	HDMI digital video data bus
U8.44	HDMI_TX_D16	C12	3.3 V	HDMI digital video data bus
U8.43	HDMI_TX_D17	C19	3.3 V	HDMI digital video data bus
U8.42	HDMI_TX_D18	C18	3.3 V	HDMI digital video data bus
U8.41	HDMI_TX_D19	B19	3.3 V	HDMI digital video data bus
U8.40	HDMI_TX_D20	B17	3.3 V	HDMI digital video data bus
U8.39	HDMI_TX_D21	B16	3.3 V	HDMI digital video data bus
U8.38	HDMI_TX_D22	C16	3.3 V	HDMI digital video data bus
U8.37	HDMI_TX_D23	A15	3.3 V	HDMI digital video data bus
U8.53	HDMI_TX_CLK	D6	3.3 V	Video clock
U8.63	HDMI_TX_DE	C10	3.3 V	Video data enable
U8.64	HDMI_TX_HS	A19	3.3 V	Vertical Synchronization
U8.2	HDMI_TX_VS	J12	3.3 V	Horizontal Synchronization
U8.28	HDMI_TX_INT	D15	3.3 V	Interrupt Signal
U8.35	HDMI_SCL	A10	3.3 V	HDMI I2C clock
U8.36	HDMI_SDA	B15	3.3 V	HDMI I2C data

HSMC

The high-speed mezzanine card (HSMC) interface is based on the Samtec 0.5 mm pitch, surface-mount QTH/QSH family of connectors. It is designed to support a full SPI-4.2 interface (17 LVDS channels) and 3 input and output clocks as well as SMBus and JTAG signals.

Since MAX 10 does not have transceiver channels, the HSMC clock-data-recovery channels are left unconnected.

The HSMC interface has programmable bi-directional I/O pins that can be used as 2.5-V LVCMOS, which is 3.3-V LVTTL-compatible. These pins can also be used as various differential I/O standards including, but not limited to, LVDS, mini-LVDS, and RSDS with up to 17 full-duplex channels.

As noted in the *High Speed Mezzanine Card (HSMC) Specification manual*, LVDS and single-ended I/O standards are only guaranteed to function when mixed according to either the generic single-ended pin-out or generic differential pin-out.

For more information about the HSMC specification such as signaling standards, signal integrity, compatible connectors, and mechanical information, refer to the *High Speed Mezzanine Card (HSMC) Specification manual*.

Table 4-18: HSMC Schematic Signals

Board Reference (J2)	Schematic Signal Name	MAX 10 / MAX II Pin Number	I/O Standard	Description
33	HSMC_SDA	AA19	2.5V CMOS inout	Management serial data line
34	HSMC_SCL	Y18	2.5V CMOS out	Management serial clock line
35	HSMC_JTAG_TCK	A9 (MAX II)	part of chain	JTAG clock
36	HSMC_JTAG_TMS	A8 (MAX II)	part of chain	JTAG mode select
37	HSMC_JTAG_TDO	A7 (MAX II)	part of chain	JTAG data out
38	HSMC_JTAG_TDI	A6 (MAX II)	part of chain	JTAG data in
39	HSMC_CLK_OUT0	AA13	2.5V CMOS clock output	clock output 0
40	HSMC_CLK_IN0	N4	2.5V CMOS clock in	Clock input 0
41	HSMC_D0	Y7	2.5v CMOS inout	Data bus
42	HSMC_D1	Y8	2.5v CMOS inout	Data bus
43	HSMC_D2	AB2	2.5v CMOS inout	Data bus
44	HSMC_D3	AB3	2.5v CMOS inout	Data bus
47	HSMC_TX_D_P0	W3	2.5v CMOS inout or LVDS TX channels-p	Data bus
48	HSMC_RX_D_P0	V5	2.5V CMOS inout or LVDS RX channels-p	Data bus
49	HSMC_TX_D_N0	W4	2.5V CMOS inout or LVDS TX channels-n	Data bus
50	HSMC_RX_D_N0 ⁽¹⁾	V4	2.5V CMOS inout or LVDS RX channels-n	Data bus
53	HSMC_TX_D_P1	U7	2.5v CMOS inout or LVDS TX channels-p	Data bus
54	HSMC_RX_D_P1 ⁽¹⁾	Y2	2.5V CMOS inout or LVDS RX channels-p	Data bus

⁽¹⁾ MAX 10 doesn't have internal termination for LVDS RX. Install a 100-ohm resistor to support LVDS RX on HSMC.

Board Reference (J2)	Schematic Signal Name	MAX 10 / MAX II Pin Number	I/O Standard	Description
55	HSMC_TX_D_N1	U6	2.5V CMOS inout or LVDS TX channels-n	Data bus
56	HSMC_RX_D_N1 ⁽¹⁾	Y1	2.5V CMOS inout or LVDS RX channels-n	Data bus
59	HSMC_TX_D_P2	W6	2.5v CMOS inout or LVDS TX channels-p	Data bus
60	HSMC_RX_D_P2 ⁽¹⁾	AA20	2.5V CMOS inout or LVDS RX channels-p	Data bus
61	HSMC_TX_D_N2	W5	2.5V CMOS inout or LVDS TX channels-n	Data bus
62	HSMC_RX_D_N2 ⁽¹⁾	AA1	2.5V CMOS inout or LVDS RX channels-n	Data bus
65	HSMC_TX_D_P3	W8	2.5v CMOS inout or LVDS TX channels-p	Data bus
66	HSMC_RX_D_P3 ⁽¹⁾	AB8	2.5V CMOS inout or LVDS RX channels-p	Data bus
67	HSMC_TX_D_N3	W7	2.5V CMOS inout or LVDS TX channels-n	Data bus
68	HSMC_RX_D_N3 ⁽¹⁾	AA8	2.5V CMOS inout or LVDS RX channels-n	Data bus
71	HSMC_TX_D_P4	AA10	2.5v CMOS inout or LVDS TX channels-p	Data bus
72	HSMC_RX_D_P4 ⁽¹⁾	AB9	2.5V CMOS inout or LVDS RX channels-p	Data bus
73	HSMC_TX_D_N4	Y10	2.5V CMOS inout or LVDS TX channels-n	Data bus
74	HSMC_RX_D_N4 ⁽¹⁾	AA9	2.5V CMOS inout or LVDS RX channels-n	Data bus
77	HSMC_TX_D_P5	AA7	2.5v CMOS inout or LVDS TX channels-p	Data bus
78	HSMC_RX_D_P5 ⁽¹⁾	AB7	2.5V CMOS inout or LVDS RX channels-p	Data bus
79	HSMC_TX_D_N5	AA6	2.5V CMOS inout or LVDS TX channels-n	Data bus
80	HSMC_RX_D_N5 ⁽¹⁾	AB6	2.5V CMOS inout or LVDS RX channels-n	Data bus
83	HSMC_TX_D_P6	P10	2.5v CMOS inout or LVDS TX channels-p	Data bus
84	HSMC_RX_D_P6 ⁽¹⁾	Y4	2.5V CMOS inout or LVDS RX channels-p	Data bus

Board Reference (J2)	Schematic Signal Name	MAX 10 / MAX II Pin Number	I/O Standard	Description
85	HSMC_TX_D_N6	R10	2.5V CMOS inout or LVDS TX channels-n	Data bus
86	HSMC_RX_D_N6 ⁽¹⁾	Y3	2.5V CMOS inout or LVDS RX channels-n	Data bus
89	HSMC_TX_D_P7	W10	2.5v CMOS inout or LVDS TX channels-p	Data bus
90	HSMC_RX_D_P7 ⁽¹⁾	AB5	2.5V CMOS inout or LVDS RX channels-p	Data bus
91	HSMC_TX_D_N7	W9	2.5V CMOS inout or LVDS TX channels-n	Data bus
92	HSMC_RX_D_N7 ⁽¹⁾	AA5	2.5V CMOS inout or LVDS RX channels-n	Data bus
95	HSMC_CLK_OUT_P1	P13	2.5V CMOS inout or LVDS clock out	Clock output 1
96	HSMC_CLK_IN_P1	AA20	2.5V CMOS inout or LVDS clock in	Clock input 1
97	HSMC_CLK_OUT_N1	R13	2.5V CMOS inout or LVDS clock out	Clock output 1
98	HSMC_CLK_IN_N1	AB21	2.5V CMOS inout or LVDS clock in	Clock input 1
101	HSMC_TX_D_P8	W14	2.5v CMOS inout or LVDS TX channels-p	Data bus
102	HSMC_RX_D_P8 ⁽¹⁾	W13	2.5V CMOS inout or LVDS RX channels-p	Data bus
103	HSMC_TX_D_N8	V13	2.5V CMOS inout or LVDS TX channels-n	Data bus
104	HSMC_RX_D_N8 ⁽¹⁾	W12	2.5V CMOS inout or LVDS RX channels-n	Data bus
107	HSMC_TX_D_P9	Y14	2.5v CMOS inout or LVDS TX channels-p	Data bus
108	HSMC_RX_D_P9 ⁽¹⁾	AB15	2.5V CMOS inout or LVDS RX channels-p	Data bus
109	HSMC_TX_D_N9	Y13	2.5V CMOS inout or LVDS TX channels-n	Data bus
110	HSMC_RX_D_N9 ⁽¹⁾	AA14	2.5V CMOS inout or LVDS RX channels-n	Data bus
113	HSMC_TX_D_P10	V16	2.5v CMOS inout or LVDS TX channels-p	Data bus
114	HSMC_RX_D_P10 ⁽¹⁾	Y16	2.5V CMOS inout or LVDS RX channels-p	Data bus

Board Reference (J2)	Schematic Signal Name	MAX 10 / MAX II Pin Number	I/O Standard	Description
115	HSMC_TX_D_N10	U15	2.5V CMOS inout or LVDS TX channels-n	Data bus
116	HSMC_RX_D_N10 ⁽¹⁾	AA15	2.5V CMOS inout or LVDS RX channels-n	Data bus
119	HSMC_TX_D_P11	W16	2.5v CMOS inout or LVDS TX channels-p	Data bus
120	HSMC_RX_D_P11 ⁽¹⁾	AA16	2.5V CMOS inout or LVDS RX channels-p	Data bus
121	HSMC_TX_D_N11	V15	2.5V CMOS inout or LVDS TX channels-n	Data bus
122	HSMC_RX_D_N11 ⁽¹⁾	AB16	2.5V CMOS inout or LVDS RX channels-n	Data bus
125	HSMC_TX_D_P12	V17	2.5v CMOS inout or LVDS TX channels-p	Data bus
126	HSMC_RX_D_P12 ⁽¹⁾	AB18	2.5V CMOS inout or LVDS RX channels-p	Data bus
127	HSMC_TX_D_N12	W17	2.5V CMOS inout or LVDS TX channels-n	Data bus
128	HSMC_RX_D_N12 ⁽¹⁾	AB17	2.5V CMOS inout or LVDS RX channels-n	Data bus
131	HSMC_TX_D_P13	V12	2.5v CMOS inout or LVDS TX channels-p	Data bus
132	HSMC_RX_D_P13 ⁽¹⁾	Y11	2.5V CMOS inout or LVDS RX channels-p	Data bus
133	HSMC_TX_D_N13	V11	2.5V CMOS inout or LVDS TX channels-n	Data bus
134	HSMC_RX_D_N13 ⁽¹⁾	W11	2.5V CMOS inout or LVDS RX channels-n	Data bus
137	HSMC_TX_D_P14	P12	2.5v CMOS inout or LVDS TX channels-p	Data bus
138	HSMC_RX_D_P14 ⁽¹⁾	AB11	2.5V CMOS inout or LVDS RX channels-p	Data bus
139	HSMC_TX_D_N14	R12	2.5V CMOS inout or LVDS TX channels-n	Data bus
140	HSMC_RX_D_N14 ⁽¹⁾	AB10	2.5V CMOS inout or LVDS RX channels-n	Data bus
143	HSMC_TX_D_P15	AA12	2.5v CMOS inout or LVDS TX channels-p	Data bus
144	HSMC_RX_D_P15 ⁽¹⁾	AB13	2.5V CMOS inout or LVDS RX channels-p	Data bus

Board Reference (J2)	Schematic Signal Name	MAX 10 / MAX II Pin Number	I/O Standard	Description
145	HSMC_TX_D_N15	AA11	2.5V CMOS inout or LVDS TX channels-n	Data bus
146	HSMC_RX_D_N15 ⁽¹⁾	AB12	2.5V CMOS inout or LVDS RX channels-n	Data bus
149	HSMC_TX_D_P16	Y17	2.5v CMOS inout or LVDS TX channels-p	Data bus
150	HSMC_RX_D_P16 ⁽¹⁾	AB20	2.5V CMOS inout or LVDS RX channels-p	Data bus
151	HSMC_TX_D_N16	AA17	2.5V CMOS inout or LVDS TX channels-n	Data bus
152	HSMC_RX_D_N16 ⁽¹⁾	AB19	2.5V CMOS inout or LVDS RX channels-n	Data bus
155	HSMC_CLK_OUT_P2	W15	2.5V CMOS inout or LVDS clock out	Clock output 2
156	HSMC_CLK_IN_P2	V10	2.5V CMOS inout or LVDS clock in	Clock input 2
157	HSMC_CLK_OUT_N2	V14	2.5V CMOS inout or LVDS clock out	Clock output 2
158	HSMC_CLK_IN_N2	V9	2.5V CMOS inout or LVDS clock in	Clock input 2
160	HSMC_PRSENTn	AB14	2.5V	Present

Related Information**High Speed Mezzanine Card (HSMC) Specification****Pmod Connectors**

The MAX 10 FPGA development kit features two Digilent Pmod™ compatible headers, which are used to connect low frequency, low I/O pin count peripheral modules.

The 12-pin version Pmod connector used in this kit provides 8 I/O signal pins. The peripheral module interface also encompasses a variant using I²C interface, and two or four wire MTE cables. The Pmod signals are connected to Bank 8.

Table 4-19: Pmod A Pin Assignments, Signal Names and Functions

Schematic Signal Name	Schematic Share Bus Signal Name	MAX 10 FPGA Pin Number	I/O Standard	Description
PMODA_D0	PMODA_IO0	C7	3.3V	In/Out
PMODA_D1	PMODA_IO1	C8	3.3V	In/Out
PMODA_D2	PMODA_IO2	A6	3.3V	In/Out
PMODA_D3	PMODA_IO3	B7	3.3V	In/Out

Schematic Signal Name	Schematic Share Bus Signal Name	MAX 10 FPGA Pin Number	I/O Standard	Description
PMODA_D4	PMODA_IO4	D8	3.3V	In/Out
PMODA_D5	PMODA_IO5	A4	3.3V	In/Out
PMODA_D6	PMODA_IO6	A5	3.3V	In/Out
PMODA_D7	PMODA_IO7	E9	3.3V	In/Out
—	VCC	—	3.3V	Power
—	GND	—	—	GND

Table 4-20: Pmod B Pin Assignments, Signal Names and Functions

Schematic Signal Name	Schematic Share Bus Signal Name	MAX 10 FPGA Pin Number	I/O Standard	Description
PMODB_D0	PMODB_IO0	E8	3.3V	In/Out
PMODB_D1	PMODB_IO1	D5	3.3V	In/Out
PMODB_D2	PMODB_IO2	B5	3.3V	In/Out
PMODB_D3	PMODB_IO3	C4	3.3V	In/Out
PMODB_D4	PMODB_IO4	A2	3.3V	In/Out
PMODB_D5	PMODB_IO5	A3	3.3V	In/Out
PMODB_D6	PMODB_IO6	B4	3.3V	In/Out
PMODB_D7	PMODB_IO7	B3	3.3V	In/Out
—	VCC	—	3.3V	Power
—	GND	—	—	GND

USB to UART

The board uses a USB based UART bridge chip (FT232R) to bridge communication to a host for general software debug for Nios and non-Nios systems. This chip uses TXD and RXD for transmission and reception of data. A mini B plug receptacle is used to minimize board space. The related I/O utilization is implemented in Bank 4.

Table 4-21: USB-UART Pin Assignments, Signal Names and Functions

Board Reference (U11)	Schematic Signal Name	MAX 10 FPGA Pin Number	I/O Standard	Description
U11.2	UART_TX	W18	2.5 V	Transmit asynchronous data output

Board Reference (U11)	Schematic Signal Name	MAX 10 FPGA Pin Number	I/O Standard	Description
U11.30	UART_RX	Y19	2.5 V	Receive asynchronous data input

Memory

This section describes the development board's memory interface support and also their signal names, types, and connectivity relative to the FPGA.

DDR3 Rev. B Board

Note: For your board's revision, look for the board serial number on the back of the board at the bottom. Refer to the *General Description* section for an image of the back board.

The MAX 10 FPGA provides full-speed support to a x16 DDR3 300-MHz interface by using a 1 Gbit x16 memory. Additionally, the MAX 10 supports the error correction code (ECC) feature.

Caution: The DDR3 address signals at F18, E19, F20 and F21 on rev. B boards violate MAX 10 external memory guidelines when implementing DDR3 on the 10M50 F484 device. Altera recommends you follow the MAX 10 guidelines for your own board designs and utilize Quartus II software to verify pin location compliance. Contact Altera support if you received DDR3 pin location errors for your Rev. B kit designs.

Table 4-22: DDR3 Pin Assignments, Signal Names, and Functions

Board Reference (U5 & U6)	Schematic Signal Name	MAX 10 FPGA Pin Number	I/O Standard	Description
U5.N3 - U6.K3	DDR3_A0	V20	1.5V SSTL	Address bus
U5.P7 - U6.L7	DDR3_A1	F20	1.5V SSTL	Address bus Refer to Caution statement above.
U5.P3 - U6.L3	DDR3_A2	F18	1.5V SSTL	Address bus Refer to Caution statement above.
U5.N2 - U6.K2	DDR3_A3	U20	1.5V SSTL	Address bus
U5.P8 - U6.L8	DDR3_A4	F21	1.5V SSTL	Address bus Refer to Caution statement above.
U5.P2 - U6.L2	DDR3_A5	F19	1.5V SSTL	Address bus
U5.R8 - U6.M8	DDR3_A6	E21	1.5V SSTL	Address bus

Board Reference (U5 & U6)	Schematic Signal Name	MAX 10 FPGA Pin Number	I/O Standard	Description
U5.R2 - U6.M2	DDR3_A7	E19	1.5V SSTL	Address bus Refer to Caution statement above.
U5.T8 - U6.N8	DDR3_A8	D22	1.5V SSTL	Address bus
U5.R3 - U6.M3	DDR3_A9	E22	1.5V SSTL	Address bus
U5.L7 - U6.H7	DDR3_A10	Y20	1.5V SSTL	Address bus
U5.R7 - U6.M7	DDR3_A11	E20	1.5V SSTL	Address bus
U5.N7 - U6.K7	DDR3_A12	J14	1.5V SSTL	Address bus
U5.T3 - U6.N3	DDR3_A13	C22	1.5V SSTL	Address bus
U5.M2 - U6.J2	DDR3_BA0	V22	1.5V SSTL	Bank address bus
U5.N8 - U6.K8	DDR3_BA1	N18	1.5V SSTL	Bank address bus
U5.M3 - U6.J3	DDR3_BA2	W22	1.5V SSTL	Bank address bus
U5.K3 - U6.G3	DDR3_CASn	U19	1.5V SSTL	Row address bus
U5.K9 - U6.G9	DDR3_CKE	W20	1.5V SSTL	Clock enable
U5.J7 - U6.F7	DDR3_CLK_P	D18	Differential 1.5V SSTL	Differential output clock
U5.K7 - U6.G7	DDR3_CLK_N	E18	Differential 1.5V SSTL	Differential output clock
U5.L2 - U6.H2	DDR3_CSn	Y22	1.5V SSTL	Chip select
U5.E7	DDR3_DM0	J15	1.5V SSTL	Write mask byte lane 0
U5.D3	DDR3_DM1	N19	1.5V SSTL	Write mask byte lane 1
U6.B7	DDR3_DM2	T18	1.5V SSTL	Write mask byte lane 2
U5.E3	DDR3_DQ0	J18	1.5V SSTL	Data bus byte lane 0
U5.F7	DDR3_DQ1	K20	1.5V SSTL	Data bus byte lane 0
U5.F2	DDR3_DQ2	H18	1.5V SSTL	Data bus byte lane 0
U5.F8	DDR3_DQ3	K18	1.5V SSTL	Data bus byte lane 0
U5.H3	DDR3_DQ4	H19	1.5V SSTL	Data bus byte lane 0
U5.H8	DDR3_DQ5	J20	1.5V SSTL	Data bus byte lane 0
U5.G2	DDR3_DQ6	H20	1.5V SSTL	Data bus byte lane 0
U5.H7	DDR3_DQ7	K19	1.5V SSTL	Data bus byte lane 0
U5.D7	DDR3_DQ8	L20	1.5V SSTL	Data bus byte lane 1
U5.C3	DDR3_DQ9	M18	1.5V SSTL	Data bus byte lane 1
U5.C8	DDR3_DQ10	M20	1.5V SSTL	Data bus byte lane 1
U5.C2	DDR3_DQ11	M14	1.5V SSTL	Data bus byte lane 1

Board Reference (U5 & U6)	Schematic Signal Name	MAX 10 FPGA Pin Number	I/O Standard	Description
U5.A7	DDR3_DQ12	L18	1.5V SSTL	Data bus byte lane 1
U5.A2	DDR3_DQ13	M15	1.5V SSTL	Data bus byte lane 1
U5.B8	DDR3_DQ14	L19	1.5V SSTL	Data bus byte lane 1
U5.A3	DDR3_DQ15	N20	1.5V SSTL	Data bus byte lane 1
U6.B3	DDR3_DQ16	R14	1.5V SSTL	Data bus byte lane 2
U6.C7	DDR3_DQ17	P19	1.5V SSTL	Data bus byte lane 2
U6.C2	DDR3_DQ18	P14	1.5V SSTL	Data bus byte lane 2
U6.C8	DDR3_DQ19	R20	1.5V SSTL	Data bus byte lane 2
U6.E3	DDR3_DQ20	R15	1.5V SSTL	Data bus byte lane 2
U6.E8	DDR3_DQ21	T19	1.5V SSTL	Data bus byte lane 2
U6.D2	DDR3_DQ22	P15	1.5V SSTL	Data bus byte lane 2
U6.E7	DDR3_DQ23	P20	1.5V SSTL	Data bus byte lane 2
U5.F3	DDR3_DQS_P0	K14	Differential 1.5V SSTL	Data strobe P byte lane 0
U5.G3	DDR3_DQS_N0	K15	Differential 1.5V SSTL	Data strobe N byte lane 0
U5.C7	DDR3_DQS_P1	L14	Differential 1.5V SSTL	Data strobe P byte lane 1
U5.B7	DDR3_DQS_N1	L15	Differential 1.5V SSTL	Data strobe N byte lane 1
U6.C3	DDR3_DQS_P2	R18	Differential 1.5V SSTL	Data strobe P byte lane 2
U6.D3	DDR3_DQS_N2	P18	Differential 1.5V SSTL	Data strobe N byte lane 2
U5.K1 - U6.G1	DDR3_ODT	W19	1.5V SSTL	On-die termination enable
U5.J3 - U6.F3	DDR3_RASn	V18	1.5V SSTL	Row address select
U5.T2 - U6.N2	DDR3_RESETh	B22	1.5V SSTL	Reset
U5.L3 - U6.H3	DDR3_WEn	Y21	1.5V SSTL	Write enable
U5.L8	DDR3_ZQ1	—	1.5V SSTL	ZQ impedance calibration
U6.H8	DDR3_ZQ2	—	1.5V SSTL	ZQ impedance calibration

Related Information

[General Description](#) on page 1-2

DDR3 Rev. C Board

Note: For your board's revision, look for the board serial number on the back the board at the bottom.

The MAX 10 FPGA provides full-speed support to a x16 DDR3 300-MHz interface by using a 1 Gbit x16 memory. Additionally, the MAX 10 supports the error correction code (ECC) feature.

Table 4-23: DDR3 Pin Assignments, Signal Names, and Functions

Board Reference (U5 & U6)	Schematic Signal Name	MAX 10 FPGA Pin Number	I/O Standard	Description
U5.N3 - U6.K3	DDR3_A0	V20	1.5V SSTL	Address bus
U5.P7 - U6.L7	DDR3_A1	D19	1.5V SSTL	Address bus
U5.P3 - U6.L3	DDR3_A2	A21	1.5V SSTL	Address bus
U5.N2 - U6.K2	DDR3_A3	U20	1.5V SSTL	Address bus
U5.P8 - U6.L8	DDR3_A4	C20	1.5V SSTL	Address bus
U5.P2 - U6.L2	DDR3_A5	F19	1.5V SSTL	Address bus
U5.R8 - U6.M8	DDR3_A6	E21	1.5V SSTL	Address bus
U5.R2 - U6.M2	DDR3_A7	B20	1.5V SSTL	Address bus
U5.T8 - U6.N8	DDR3_A8	D22	1.5V SSTL	Address bus
U5.R3 - U6.M3	DDR3_A9	E22	1.5V SSTL	Address bus
U5.L7 - U6.H7	DDR3_A10	Y20	1.5V SSTL	Address bus
U5.R7 - U6.M7	DDR3_A11	E20	1.5V SSTL	Address bus
U5.N7 - U6.K7	DDR3_A12	J14	1.5V SSTL	Address bus
U5.T3 - U6.N3	DDR3_A13	C22	1.5V SSTL	Address bus
U5.M2 - U6.J2	DDR3_BA0	V22	1.5V SSTL	Bank address bus
U5.N8 - U6.K8	DDR3_BA1	N18	1.5V SSTL	Bank address bus
U5.M3 - U6.J3	DDR3_BA2	W22	1.5V SSTL	Bank address bus
U5.K3 - U6.G3	DDR3_CASn	U19	1.5V SSTL	Row address bus
U5.K9 - U6.G9	DDR3_CKE	W20	1.5V SSTL	Clock enable
U5.J7 - U6.F7	DDR3_CLK_P	D18	Differential 1.5V SSTL	Differential output clock
U5.K7 - U6.G7	DDR3_CLK_N	E18	Differential 1.5V SSTL	Differential output clock
U5.L2 - U6.H2	DDR3_CSn	Y22	1.5V SSTL	Chip select
U5.E7	DDR3_DM0	J15	1.5V SSTL	Write mask byte lane 0
U5.D3	DDR3_DM1	N19	1.5V SSTL	Write mask byte lane 1
U6.B7	DDR3_DM2	T18	1.5V SSTL	Write mask byte lane 2
U5.E3	DDR3_DQ0	J18	1.5V SSTL	Data bus byte lane 0
U5.F7	DDR3_DQ1	K20	1.5V SSTL	Data bus byte lane 0
U5.F2	DDR3_DQ2	H18	1.5V SSTL	Data bus byte lane 0

Board Reference (U5 & U6)	Schematic Signal Name	MAX 10 FPGA Pin Number	I/O Standard	Description
U5.F8	DDR3_DQ3	K18	1.5V SSTL	Data bus byte lane 0
U5.H3	DDR3_DQ4	H19	1.5V SSTL	Data bus byte lane 0
U5.H8	DDR3_DQ5	J20	1.5V SSTL	Data bus byte lane 0
U5.G2	DDR3_DQ6	H20	1.5V SSTL	Data bus byte lane 0
U5.H7	DDR3_DQ7	K19	1.5V SSTL	Data bus byte lane 0
U5.D7	DDR3_DQ8	L20	1.5V SSTL	Data bus byte lane 1
U5.C3	DDR3_DQ9	M18	1.5V SSTL	Data bus byte lane 1
U5.C8	DDR3_DQ10	M20	1.5V SSTL	Data bus byte lane 1
U5.C2	DDR3_DQ11	M14	1.5V SSTL	Data bus byte lane 1
U5.A7	DDR3_DQ12	L18	1.5V SSTL	Data bus byte lane 1
U5.A2	DDR3_DQ13	M15	1.5V SSTL	Data bus byte lane 1
U5.B8	DDR3_DQ14	L19	1.5V SSTL	Data bus byte lane 1
U5.A3	DDR3_DQ15	N20	1.5V SSTL	Data bus byte lane 1
U6.B3	DDR3_DQ16	R14	1.5V SSTL	Data bus byte lane 2
U6.C7	DDR3_DQ17	P19	1.5V SSTL	Data bus byte lane 2
U6.C2	DDR3_DQ18	P14	1.5V SSTL	Data bus byte lane 2
U6.C8	DDR3_DQ19	R20	1.5V SSTL	Data bus byte lane 2
U6.E3	DDR3_DQ20	R15	1.5V SSTL	Data bus byte lane 2
U6.E8	DDR3_DQ21	T19	1.5V SSTL	Data bus byte lane 2
U6.D2	DDR3_DQ22	P15	1.5V SSTL	Data bus byte lane 2
U6.E7	DDR3_DQ23	P20	1.5V SSTL	Data bus byte lane 2
U5.F3	DDR3_DQS_P0	K14	Differential 1.5V SSTL	Data strobe P byte lane 0
U5.G3	DDR3_DQS_N0	K15	Differential 1.5V SSTL	Data strobe N byte lane 0
U5.C7	DDR3_DQS_P1	L14	Differential 1.5V SSTL	Data strobe P byte lane 1
U5.B7	DDR3_DQS_N1	L15	Differential 1.5V SSTL	Data strobe N byte lane 1
U6.C3	DDR3_DQS_P2	R18	Differential 1.5V SSTL	Data strobe P byte lane 2
U6.D3	DDR3_DQS_N2	P18	Differential 1.5V SSTL	Data strobe N byte lane 2
U5.K1 - U6.G1	DDR3_ODT	W19	1.5V SSTL	On-die termination enable
U5.J3 - U6.F3	DDR3_RASn	V18	1.5V SSTL	Row address select

Board Reference (U5 & U6)	Schematic Signal Name	MAX 10 FPGA Pin Number	I/O Standard	Description
U5.T2 - U6.N2	DDR3_RESETh	B22	1.5V SSTL	Reset
U5.L3 - U6.H3	DDR3_WEn	Y21	1.5V SSTL	Write enable
U5.L8	DDR3_ZQ1	—	1.5V SSTL	ZQ impedance calibration
U6.H8	DDR3_ZQ2	—	1.5V SSTL	ZQ impedance calibration

Flash

The MAX 10 FPGA development kit provides a 512-Mb (megabit) quad SPI flash memory. Altera Generic QUAD SPI controller core is used by default to erase, read, and write quad SPI flash in reference designs of the Board Test System (BTS) installer.

If you use the parallel flash loader (PFL) IP to program the quad SPI flash, you need to generate a **.pof** (Programmer Object file) to configure the device.

Perform the following steps to generate a **.pof** file:

1. Create a byte-order `Quartus.ini` file with the setting:

```
PGMIO_SWAP_HEX_BYTE_DATA=ON
```

2. Copy the **.ini** file to the project root directory and open the project with Quartus
3. Open **Convert Programming Files** tool to generate the **.pof** file

Table 4-24: Default Memory Map of the 512-Mb QSPI Flash

Block Description	Size (KB)	Address Range
Board test system scratch	512	0x03F8.0000 – 0x03FF.FFFF
User software	56640	0x0083.0000 – 0x03F7.FFFF
Factory software	4096	0x0043.0000 – 0x0082.FFFF
Zips(html, web content)	4096	0x0003.0000 – 0x0042.FFFF
Board information	64	0x0002.0000 – 0x0002.FFFF
Ethernet option bits	64	0x0001.0000 – 0x0001.FFFF
User design reset vector	64	0x0000.0000 – 0x0000.FFFF

Table 4-25: Flash Pin Assignments, Schematic Signal Names, and Functions

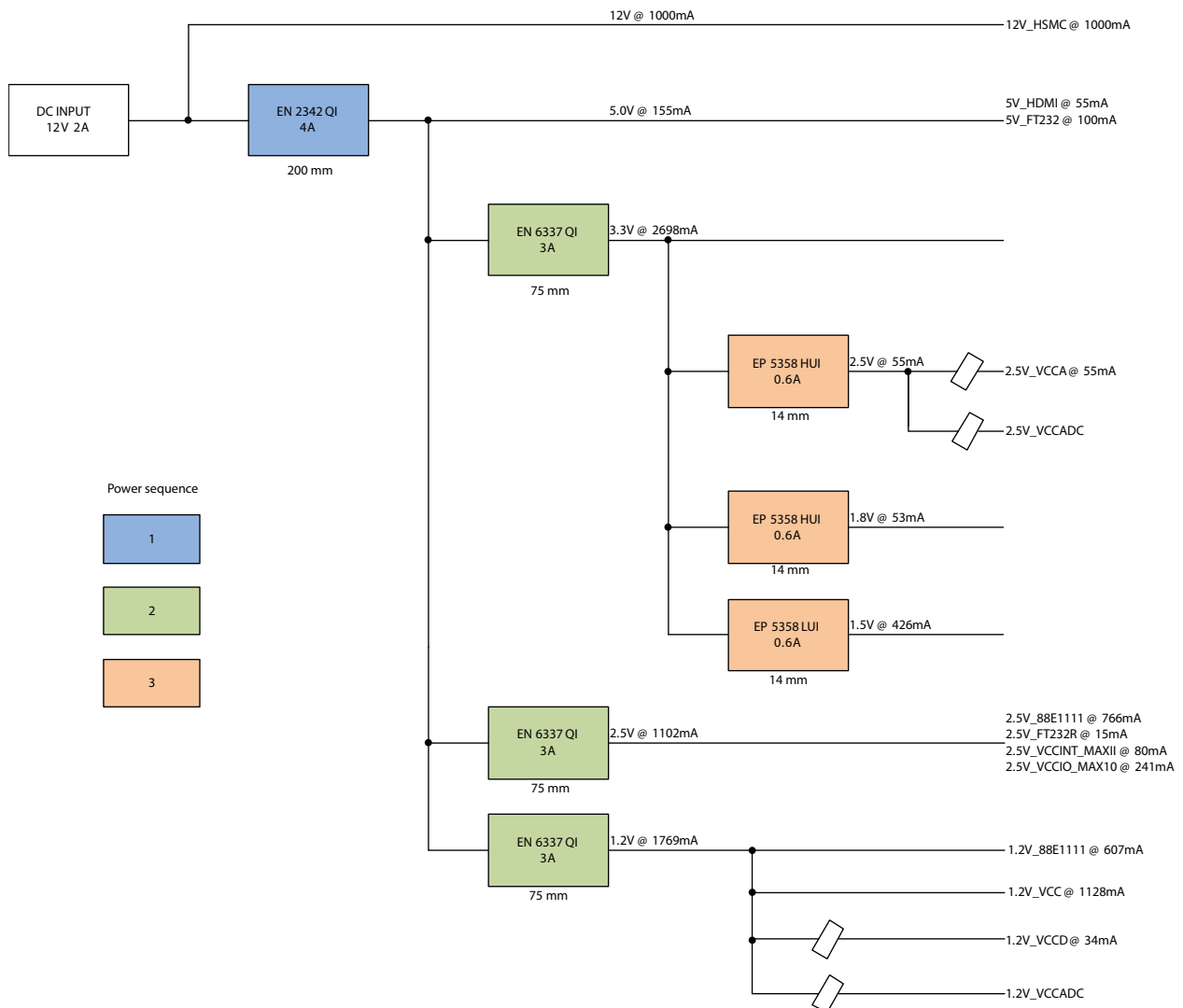
Board Reference (U7)	Schematic Signal Name	Max 10 FPGA Pin Number	I/O Standard	Description
U7.7	QSPI_CS _n	C2	3.3V	Chip select
U7.16	QSPI_CLK	B2	3.3V	Clock
U7.3	QSPI_RESETh	W12 (MAX II)	3.3V	Reset
U7.15	QSPI_IO0	C6	3.3V	Address bus
U7.8	QSPI_IO1	C3	3.3V	Address bus

Board Reference (U7)	Schematic Signal Name	Max 10 FPGA Pin Number	I/O Standard	Description
U7.9	QSPI_IO2	C5	3.3V	Address bus
U7.1	QSPI_IO3	B1	3.3V	Address bus

Power Distribution System

This topic shows the power tree drawing for the MAX 10 FPGA development board. Regulator inefficiencies and sharing are reflected in the currents shown, which are conservative absolute maximum levels.

Figure 4-4: Power Distribution System



2017.09.07

UG-01169



Subscribe



Send Feedback

User Guide Revision History

Table A-1: MAX 10 FPGA Development Kit User Guide Revision History

Date	Version	Changes
September 2017	2017.09.07	Updated I/O standard voltage values in the On-Board Oscillators table in On-Board Oscillators on page 4-10
January 2017	2017.01.04	Corrected the following pin assignments in "10/100/1000 Ethernet PHY": <ul style="list-style-type: none"> • ENETA_TX_D1 on pin P5 • ENETA_RX_ER on pin U2 • ENET_MDIO on pin Y5 • ENETB_TX_D2 on pin U3 • ENETB_RS_D3 on pin R7
November 2015	2015.11.06	<ul style="list-style-type: none"> • Updated "USB to UART" section. • Added note to "General User Input/Output section".
June 2015	2015.06.26	<ul style="list-style-type: none"> • Updated "DDR3 Rev. B Board" section.
May 2015	2015.05.21	<ul style="list-style-type: none"> • Added quad SPI content for Rev. B & C boards. • Corrected two PMOD pin signal names for Rev. B & C boards. • Changed four MAX 10 pins for DDR3 for Rev. C board only. • Changed two switch/signal names for SW2 for Rev. C board only. • Updated <i>Switch and Jumper Settings</i> section with VTAP description.
March 2015	2015.03.31	Initial release.

Intel Corporation. All rights reserved. Intel, the Intel logo, Altera, Arria, Cyclone, Enpirion, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.

ISO
9001:2015
Registered

ALTERA
now part of Intel

Compliance and Conformity Statements

CE EMI Conformity Caution

This board is delivered conforming to relevant standards mandated by Directive 2004/108/EC. Because of the nature of programmable logic devices, it is possible for the user to modify the kit in such a way as to generate electromagnetic interference (EMI) that exceeds the limits established for this equipment. Any EMI caused as the result of modifications to the delivered material is the responsibility of the user.





Intel® MAX® 10 FPGA Configuration User Guide



Online Version



Send Feedback

UG-M10CONFIG

ID: **683865**

Version: **2023.03.27**

Contents

1. Intel® MAX® 10 FPGA Configuration Overview.....	4
2. Intel MAX 10 FPGA Configuration Schemes and Features.....	5
2.1. Configuration Schemes.....	5
2.1.1. JTAG Configuration.....	5
2.1.2. Internal Configuration.....	6
2.2. Configuration Features.....	13
2.2.1. Remote System Upgrade.....	13
2.2.2. Configuration Design Security.....	20
2.2.3. SEU Mitigation and Configuration Error Detection.....	26
2.2.4. Configuration Data Compression.....	29
2.3. Configuration Details.....	30
2.3.1. Configuration Sequence.....	30
2.3.2. Intel MAX 10 Configuration Pins.....	33
3. Intel MAX 10 FPGA Configuration Design Guidelines.....	34
3.1. Dual-Purpose Configuration Pins.....	34
3.1.1. Guidelines: Dual-Purpose Configuration Pin.....	34
3.1.2. Enabling Dual-Purpose Pin.....	35
3.2. Configuring Intel MAX 10 Devices using JTAG Configuration.....	35
3.2.1. Auto-Generating Configuration Files for Third-Party Programming Tools.....	36
3.2.2. Generating Third-Party Programming Files using Intel Quartus Prime Programmer.....	36
3.2.3. JTAG Configuration Setup.....	37
3.2.4. ICB Settings in JTAG Configuration.....	39
3.3. Configuring Intel MAX 10 Devices using Internal Configuration.....	39
3.3.1. Selecting Internal Configuration Modes.....	40
3.3.2. .pof and ICB Settings.....	40
3.3.3. Programming .pof into Internal Flash.....	45
3.4. Implementing ISP Clamp in Intel Quartus Prime Software.....	46
3.4.1. Creating IPS File.....	46
3.4.2. Executing IPS File.....	46
3.5. Accessing Remote System Upgrade through User Logic.....	46
3.6. Error Detection.....	48
3.6.1. Verifying Error Detection Functionality.....	48
3.6.2. Enabling Error Detection.....	49
3.6.3. Accessing Error Detection Block Through User Logic.....	49
3.7. Enabling Data Compression.....	51
3.7.1. Enabling Compression Before Design Compilation.....	51
3.7.2. Enabling Compression After Design Compilation.....	52
3.8. AES Encryption.....	52
3.8.1. Generating .ekp File and Encrypt Configuration File.....	53
3.8.2. Generating .jam/.jbc/.svf file from .ekp file.....	54
3.8.3. Programming .ekp File and Encrypted POF File.....	54
3.8.4. Encryption in Internal Configuration.....	56
3.9. Intel MAX 10 JTAG Secure Design Example.....	58
3.9.1. Internal and External JTAG Interfaces.....	58

3.9.2. JTAG WYSIWYG Atom for JTAG Control Block Access Using Internal JTAG Interface.....	59
3.9.3. Executing LOCK and UNLOCK JTAG Instructions.....	60
3.9.4. Verifying the JTAG Secure Mode.....	61
4. Intel MAX 10 FPGA Configuration IP Core Implementation Guides.....	63
4.1. Unique Chip ID Intel FPGA IP Core.....	63
4.1.1. Instantiating the Unique Chip ID Intel FPGA IP Core.....	63
4.1.2. Resetting the Unique Chip ID Intel FPGA IP Core.....	63
4.2. Dual Configuration Intel FPGA IP Core.....	63
4.2.1. Instantiating the Dual Configuration Intel FPGA IP Core.....	64
5. Dual Configuration Intel FPGA IP Core References.....	65
5.1. Dual Configuration Intel FPGA IP Core Avalon Memory-Mapped Address Map.....	65
5.2. Dual Configuration Intel FPGA IP Core Parameters.....	67
6. Unique Chip ID Intel FPGA IP Core References.....	68
6.1. Unique Chip ID Intel FPGA IP Core Ports.....	68
7. Document Revision History for the Intel MAX 10 FPGA Configuration User Guide.....	69

1. Intel® MAX® 10 FPGA Configuration Overview

You can configure Intel® MAX® 10 configuration RAM (CRAM) using the following configuration schemes:

- JTAG configuration—using JTAG interface.
- Internal configuration—using internal flash.

Supported Configuration Features

Table 1. Configuration Schemes and Features Supported by Intel MAX 10 Devices

Configuration Scheme	Remote System Upgrade	Compression	Design Security	SEU Mitigation
JTAG configuration	—	—	—	Yes
Internal configuration	Yes	Yes	Yes	Yes

Related IP Cores

- Dual Configuration Intel FPGA IP—used in the remote system upgrade feature.
- Unique Chip ID Intel FPGA IP—retrieves the chip ID of Intel MAX 10 devices.

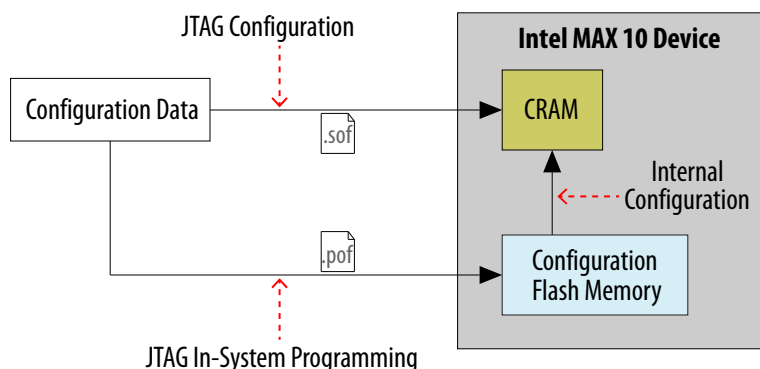
Related Information

- [Intel MAX 10 FPGA Configuration Schemes and Features](#) on page 5
Provides information about the configuration schemes and features.
- [Intel MAX 10 FPGA Configuration Design Guidelines](#) on page 34
Provides information about using the configuration schemes and features.
- [Unique Chip ID Intel FPGA IP Core](#) on page 21
- [Dual Configuration Intel FPGA IP Core](#) on page 19

2. Intel MAX 10 FPGA Configuration Schemes and Features

2.1. Configuration Schemes

Figure 1. High-Level Overview of JTAG Configuration and Internal Configuration for Intel MAX 10 Devices



2.1.1. JTAG Configuration

In Intel MAX 10 devices, JTAG instructions take precedence over the internal configuration scheme.

Using the JTAG configuration scheme, you can directly configure the device CRAM through the JTAG interface—TDI, TDO, TMS, and TCK pins. The Intel Quartus® Prime software automatically generates an SRAM Object File (.sof). You can program the .sof using a download cable with the Intel Quartus Prime software programmer.

Related Information

[Configuring Intel MAX 10 Devices using JTAG Configuration](#) on page 35

Provides more information about JTAG configuration using download cable with Intel Quartus Prime software programmer.

2.1.1.1. JTAG Pins

Table 2. JTAG Pin

Pin	Function	Description
TDI	Serial input pin for:	<ul style="list-style-type: none"> TDI is sampled on the rising edge of TCK TDI pins have internal weak pull-up resistors.
continued...		

Pin	Function	Description
	<ul style="list-style-type: none"> instructions boundary-scan test (BST) data programming data 	
TDO	Serial output pin for: <ul style="list-style-type: none"> instructions boundary-scan test data programming data 	<ul style="list-style-type: none"> TDO is sampled on the falling edge of TCK The pin is tri-stated if data is not shifted out of the device.
TMS	Input pin that provides the control signal to determine the transitions of the TAP controller state machine.	<ul style="list-style-type: none"> TMS is sampled on the rising edge of TCK TMS pins have internal weak pull-up resistors.
TCK	Clock input to the BST circuitry.	—

All the JTAG pins are powered by the V_{CCIO} 1B. In JTAG mode, the I/O pins support the LVTTTL/LVCMOS 3.3-1.5V standards.

Related Information

- [Intel MAX 10 FPGA Device Datasheet](#)
Provides more information about supported I/O standards in Intel MAX 10 devices.
- [Guidelines: Dual-Purpose Configuration Pin](#) on page 34
- [Enabling Dual-Purpose Pin](#) on page 35

2.1.2. Internal Configuration

You need to program the configuration data into the configuration flash memory (CFM) before internal configuration can take place. The configuration data to be written to CFM will be part of the programmer object file (.pof). Using JTAG In-System Programming (ISP), you can program the .pof into the internal flash.

During internal configuration, Intel MAX 10 devices load the CRAM with configuration data from the CFM.

2.1.2.1. Internal Configuration Modes

Table 3. Supported Internal Configuration Modes Based on Intel MAX 10 Feature Options

Intel MAX 10 Feature Options	Supported Internal Configuration Mode
Compact	<ul style="list-style-type: none"> Single Compressed Image Single Uncompressed Image
Flash and Analog	<ul style="list-style-type: none"> Dual Compressed Images Single Compressed Image Single Compressed Image with Memory Initialization Single Uncompressed Image Single Uncompressed Image with Memory Initialization

Note: In dual compressed images mode, you can use the CONFIG_SEL pin to select the configuration image.

Note: From Quartus Standard Edition version 19.1 onwards, the `Enable CONFIG_SEL` pin option is removed from **Device and Pin Options** for Intel MAX 10 devices with Compact feature option. The pin instead functions as user I/O.

Related Information

- [Configuring Intel MAX 10 Devices using Internal Configuration](#) on page 39
- [Remote System Upgrade](#) on page 13

2.1.2.2. Configuration Flash Memory

The CFM is a non-volatile internal flash that is used to store configuration images. The CFM may store up to two compressed configuration images, depending on the compression and the Intel MAX 10 devices. The compression ratio for the configuration image should be at least 30% for the device to be able store two configuration images.

Related Information

[Configuration Flash Memory Permissions](#) on page 23

2.1.2.2.1. Configuration Flash Memory Sectors

All CFM in Intel MAX 10 devices consist of three sectors, CFM0, CFM1, and CFM2 except for the 10M02. The sectors are programmed differently depending on the internal configuration mode you select.

The 10M02 device consists of only CFM0. The CFM0 sector in 10M02 devices is programmed similarly when you select single compressed image or single uncompressed image.

Figure 2. Configuration Flash Memory Sectors Utilization for all Intel MAX 10 with Analog and Flash Feature Options

Unutilized CFM1 and CFM2 sectors can be used for additional user flash memory (UFM).

Internal Configuration Mode	User Flash Memory Sectors		Configuration Flash Memory Sectors		
	UFM1	UFM0	CFM2	CFM1	CFM0
Dual Compressed Image	UFM		Compressed Image 1		Compressed Image 0
Single Uncompressed Image	UFM		Additional UFM	Uncompressed Image 0	
Single Uncompressed Image with Memory Initialization	UFM		Uncompressed Image 0 with Memory Initialization		
Single Compressed Image with Memory Initialization	UFM		Compressed Image 0 with Memory Initialization		
Single Compressed Image	UFM		Additional UFM		Compressed Image 0

Related Information

CFM and UFM Array Size

Provides more information about UFM and CFM sector sizes.

2.1.2.2.2. Configuration Flash Memory Programming Time

Table 4. Configuration Flash Memory Programming Time for Sectors in Intel MAX 10 Devices

Note: The programming time reflects JTAG interface programming time only without any system overhead. It does not reflect the actual programming time that you face. To compensate the system overhead, Intel Quartus Prime Programmer is enhanced to utilize flash parallel mode during device programming for Intel MAX 10 10M04/08/16/25/40/50 devices. The 10M02 device does not support flash parallel mode, you may experience a relatively slow programming time if compare to other device.

Device	In-System Programming Time (s)		
	CFM2	CFM1	CFM0
10M02 ⁽¹⁾	—	—	5.4
10M04 and 10M08	6.5	4.6	11.1
10M16	12.0	8.9	20.8
10M25	16.4	12.6	29.0
10M40 and 10M50	30.2	22.7	52.9

2.1.2.3. In-System Programming

You can program the internal flash including the CFM of Intel MAX 10 devices with ISP through industry standard IEEE 1149.1 JTAG interface. ISP offers the capability to program, erase, and verify the CFM. The JTAG circuitry and ISP instructions for Intel MAX 10 devices are compliant to the IEEE-1532-2002 programming specification.

During ISP, the Intel MAX 10 receives the IEEE Std. 1532 instructions, addresses, and data through the TDI input pin. Data is shifted out through the TDO output pin and compared with the expected data.

The following are the generic flow of an ISP operation:

1. Check ID—the JTAG ID is checked before any program or verify process. The time required to read this JTAG ID is relatively small compared to the overall programming time.
2. Enter ISP—ensures the I/O pins transition smoothly from user mode to the ISP mode.
3. Sector Erase—shifting in the address and instruction to erase the device and applying erase pulses.

⁽¹⁾ The CFM0 programming time for the 10M02SCU324 device is 11.1 s.

4. Program—shifting in the address, data, and program instructions and generating the program pulse to program the flash cells. This process is repeated for each address in the internal flash sector.
5. Verify—shifting in addresses, applying the verify instruction to generate the read pulse, and shifting out the data for comparison. This process is repeated for each internal flash address.
6. Exit ISP—ensures that the I/O pins transition smoothly from the ISP mode to the user mode.

You can also use the Intel Quartus Prime Programmer to program the CFM.

Related Information

[Programming .pof into Internal Flash](#) on page 45

Provides the steps to program the .pof using Intel Quartus Prime Programmer.

2.1.2.3.1. ISP Clamp

When a normal ISP operation begins, all I/O pins are tri-stated. For situations when the I/O pins of the device should not be tri-stated when the device is in ISP operation, you can use the ISP clamp feature.

When the ISP clamp feature is used, you can set the I/O pins to tri-state, high, low, or sample and sustain. The Intel Quartus Prime software determines the values to be scanned into the boundary-scan registers of each I/O pin, based on your settings. This will determine the state of the pins to be clamped to when the device programming is in progress.

Before clamping the I/O pins, the SAMPLE/PRELOAD JTAG instruction is first executed to load the appropriate values to the boundary-scan registers. After loading the boundary-scan registers with the appropriate values, the EXTEST instruction is executed to clamp the I/O pins to the specific values loaded into the boundary-scan registers during SAMPLE/PRELOAD.

If you choose to sample the existing state of a pin and hold the pin to that state when the device enters ISP clamp mode, you must ensure that the signal is in steady state. A steady state signal is needed because you cannot control the sample set-up time as it depends on the TCK frequency as well as the download cable and software. You might not capture the correct value when sampling a signal that toggles or is not static for long periods of time.

Related Information

[Implementing ISP Clamp in Intel Quartus Prime Software](#) on page 46

2.1.2.3.2. Real-Time ISP

In a normal ISP operation, to update the internal flash with a new design image, the device exits from user mode and all I/O pins remain tri-stated. After the device completes programming the new design image, it resets and enters user mode.

The real-time ISP feature updates the internal flash with a new design image while operating in user mode. During the internal flash programming, the device continues to operate using the existing design. After the new design image programming process completes, the device will not reset. The new design image update only takes effect in the next reconfiguration cycle.

2.1.2.3.3. ISP and Real-Time ISP Instructions

Table 5. ISP and Real-Time ISP Instructions for Intel MAX 10 Devices

Instruction	Instruction Code	Description
CONFIG_IO	00 0000 1101	<ul style="list-style-type: none"> Allows I/O reconfiguration through JTAG ports using the IOCSR for JTAG testing. This is executed after or during configurations. nSTATUS pin must go high before you can issue the CONFIG_IO instruction.
PULSE_NCONFIG	00 0000 0001	Emulates pulsing the nCONFIG pin low to trigger reconfiguration even though the physical pin is unaffected.
ISC_ENABLE_HIZ ⁽²⁾	10 1100 1100	<ul style="list-style-type: none"> Puts the device in ISP mode, tri-states all I/O pins, and drives all core drivers, logic, and registers. Device remains in the ISP mode until the ISC_DISABLE instruction is loaded and updated. The ISC_ENABLE instruction is a mandatory instruction. This requirement is met by the ISC_ENABLE_CLAMP or ISC_ENABLE_HIZ instruction.
ISC_ENABLE_CLAMP ⁽²⁾	10 0011 0011	<ul style="list-style-type: none"> Puts the device in ISP mode and forces all I/O pins to follow the contents of the JTAG boundary-scan register. When this instruction is activated, all core drivers, logics, and registers are frozen. The I/O pins remain clamped until the device exits ISP mode successfully.
ISC_DISABLE	10 0000 0001	<ul style="list-style-type: none"> Brings the device out of ISP mode. Successful completion of the ISC_DISABLE instruction happens immediately after waiting 200 µs in the Run-Test/Idle state.
ISC_PROGRAM ⁽³⁾	10 1111 0100	Sets the device up for in-system programming. Programming occurs in the run-test or idle state.
ISC_NOOP ⁽³⁾	10 0001 0000	<ul style="list-style-type: none"> Sets the device to a no-operation mode without leaving the ISP mode and targets the ISC_Default register. Use when: <ul style="list-style-type: none"> two or more ISP-compliant devices are being accessed in ISP mode and; a subset of the devices perform some instructions while other more complex devices are completing extra steps in a given process.
ISC_ADDRESS_SHIFT ⁽³⁾	10 0000 0011	Sets the device up to load the flash address. It targets the ISC_Address register, which is the flash address register.
ISC_ERASE ⁽³⁾	10 1111 0010	<ul style="list-style-type: none"> Sets the device up to erase the internal flash. Issue after ISC_ADDRESS_SHIFT instruction.
continued...		

⁽²⁾ Do not issue the ISC_ENABLE_HIZ and ISC_ENABLE_CLAMP instructions from the core logic.

⁽³⁾ All ISP and real-time ISP instructions are disabled when the device is not in the ISP or real-time ISP mode, except for the enabling and disabling instructions.

Instruction	Instruction Code	Description
ISC_READ ⁽³⁾	10 0000 0101	<ul style="list-style-type: none"> Sets the device up for verifying the internal flash under normal user bias conditions. The ISC_READ instruction supports explicit addressing and auto-increment, also known as the Burst mode.
BGP_ENABLE	01 1001 1001	<ul style="list-style-type: none"> Sets the device to the real-time ISP mode. Allows access to the internal flash configuration sector while the device is still in user mode.
BGP_DISABLE	01 0110 0110	<ul style="list-style-type: none"> Brings the device out of the real-time ISP mode. The device has to exit the real-time ISP mode using the BGP_DISABLE instruction after it is interrupted by reconfiguration.

Caution: Do not use unsupported JTAG instructions. Unsupported JTAG instructions put the device into an unknown state and requires a power cycle to recover the operation.

2.1.2.4. Initialization Configuration Bits

Initialization Configuration Bits (ICB) stores the configuration feature settings of the Intel MAX 10 device. You can set the ICB settings in the **Convert Programming File** tool.

Table 6. ICB Values and Descriptions for Intel MAX 10 Devices

Configuration Settings	Description	Default State/Value
Set I/O to weak pull-up prior usermode	<ul style="list-style-type: none"> Enable: Sets I/O to weak pull-up during device configuration. Disable: Tri-states I/O 	Enable
Configure device from CFM0 only.	Enable: <ul style="list-style-type: none"> CONFIG_SEL pin setting is disabled. Device automatically loads image 0. Device does not load image 1 if image 0 fails. Disable: <ul style="list-style-type: none"> Device automatically loads secondary image if initial image fails. 	Disable
Use secondary image ISP data as default setting when available.	Select ISP data from initial or secondary image to include in the POF. <ul style="list-style-type: none"> Disable: Use ISP data from initial image Enable: Use ISP data from secondary image ISP data contains the information about state of the pin during ISP. This can be either tri-state with weak pull-up or clamp the I/O state. You can set the ISP clamp through Device and Pin Option , or Pin Assignment tool.	Disable
Verify Protect	To disable or enable the Verify Protect feature.	Disable
Allow encrypted POF only	If enabled, a configuration error occurs if an unencrypted .pof is used.	Disable
continued...		

Configuration Settings	Description	Default State/Value
JTAG Secure ⁽⁴⁾	To disable or enable the JTAG Secure feature.	Disable
Enable Watchdog	To disable or enable the watchdog timer for remote system upgrade.	Enable
Watchdog value	To set the watchdog timer value for remote system upgrade.	0xFFFF ⁽⁵⁾

Related Information

- [.pof and ICB Settings](#) on page 40
- [Verify Protect](#) on page 22
- [JTAG Secure Mode](#) on page 22
- [ISP and Real-Time ISP Instructions](#) on page 10
- [User Watchdog Timer](#) on page 18
- [Generating .pof using Convert Programming Files](#) on page 41
Provides more information about setting the ICB during .pof generation using Convert Programming File.

2.1.2.5. Internal Configuration Time

The internal configuration time measurement is from the rising edge of nSTATUS signal to the rising edge of CONF_DONE signal.

Table 7. Internal Configuration Time for Intel MAX 10 Devices (Uncompressed .rbf)

Device	Internal Configuration Time (ms)							
	Unencrypted				Encrypted			
	Without Memory Initialization		With Memory Initialization		Without Memory Initialization		With Memory Initialization	
	Min	Max	Min	Max	Min	Max	Min	Max
10M02/ 10M02SCU324	0.3/0.6	1.7/2.7	—	—	1.7/5.0	5.4/15.0	—	—
10M04	0.6	2.7	1.0	3.4	5.0	15.0	6.8	19.6
10M08	0.6	2.7	1.0	3.4	5.0	15.0	6.8	19.6
10M16	1.1	3.7	1.4	4.5	9.3	25.3	11.7	31.5
10M25	1.0	3.7	1.3	4.4	14.0	38.1	16.9	45.7
10M40	2.6	6.9	3.2	9.8	41.5	112.1	51.7	139.6
10M50	2.6	6.9	3.2	9.8	41.5	112.1	51.7	139.6

⁽⁴⁾ The JTAG Secure feature is disabled by default in Intel Quartus Prime software. To make this option visible, refer to [Generating .pof using Convert Programming Files](#) on page 41 for more information.

⁽⁵⁾ The watchdog timer value depends on the Intel MAX 10 you are using. Refer to the Watchdog Timer section for more information.

Table 8. Internal Configuration Time for Intel MAX 10 Devices (Compressed .rbf)

Compression ratio depends on design complexity. The minimum value is based on the best case (25% of original .rbf sizes) and the maximum value is based on the typical case (70% of original .rbf sizes).

Device	Internal Configuration Time (ms)			
	Unencrypted/Encrypted			
	Without Memory Initialization		With Memory Initialization	
	Min	Max	Min	Max
10M02/10M02SCU324	0.3/0.6	5.2/10.7	—	—
10M04	0.6	10.7	1.0	13.9
10M08	0.6	10.7	1.0	13.9
10M16	1.1	17.9	1.4	22.3
10M25	1.1	26.9	1.4	32.2
10M40	2.6	66.1	3.2	82.2
10M50	2.6	66.1	3.2	82.2

2.2. Configuration Features

2.2.1. Remote System Upgrade

Intel MAX 10 devices support the remote system upgrade feature. By default, the remote system upgrade feature is enabled when you select the dual compressed image internal configuration mode.

The remote system upgrade feature in Intel MAX 10 devices offers the following capabilities:

- Manages remote configuration
- Provides error detection, recovery, and information
- Supports direct-to-application configuration image
- Supports compressed and encrypted .pof

There are two methods to access remote system upgrade in Intel MAX 10 devices:

- Dual Configuration Intel FPGA IP core
- User interface

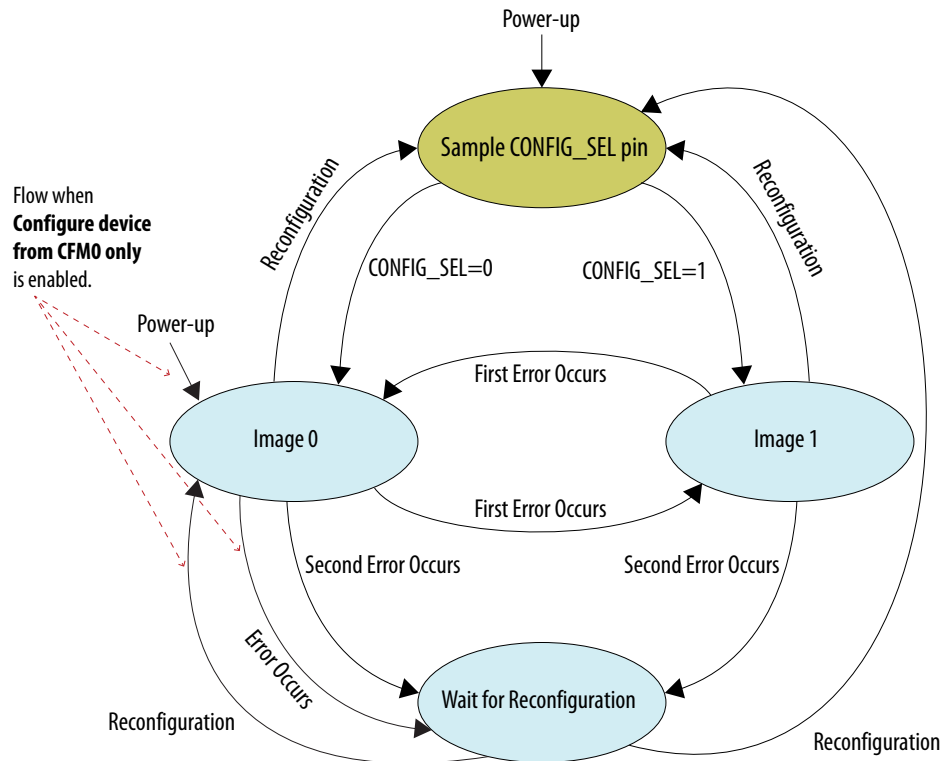
Related Information

- [Dual Configuration Intel FPGA IP Core](#) on page 19
- [Accessing Remote System Upgrade through User Logic](#) on page 46
- [AN 741: Remote System Upgrade for MAX 10 FPGA Devices over UART with the Nios II Processor](#)
Provides reference design for remote system upgrade in Intel MAX 10 FPGA devices.
- [I2C Remote System Update Example](#)
This example demonstrates a remote system upgrade using the I2C protocol.

2.2.1.1. Remote System Upgrade Flow

Both the application configuration images, image 0 and image 1, are stored in the CFM. The Intel MAX 10 device loads either one of the application configuration image from the CFM.

Figure 3. Remote System Upgrade Flow for Intel MAX 10 Devices



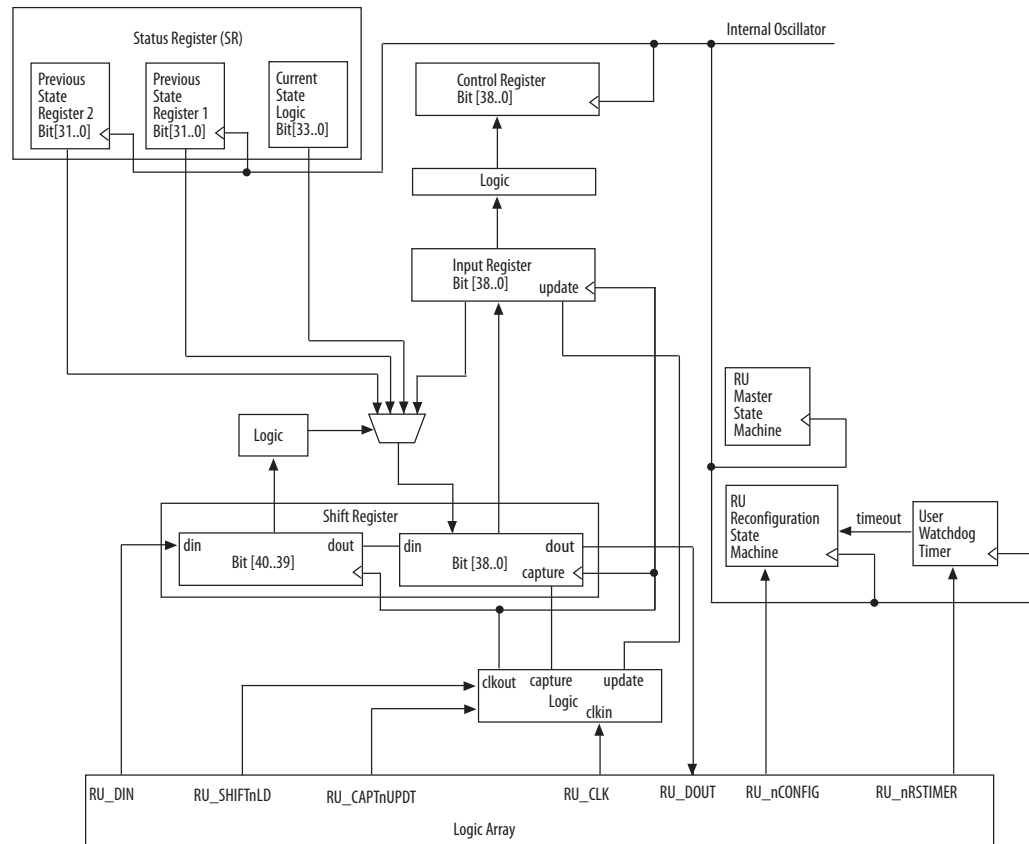
The remote system upgrade feature detects errors in the following sequence:

1. After power-up, the device samples the CONFIG_SEL pin to determine which application configuration image to load. The CONFIG_SEL pin setting can be overwritten by the input register of the remote system upgrade circuitry for the subsequent reconfiguration.
2. If an error occurs, the remote system upgrade feature reverts by loading the other application configuration image. These errors cause the remote system upgrade feature to load another application configuration image:
 - Internal CRC error
 - User watchdog timer time-out
3. Once the revert configuration completes and the device is in user mode, you can use the remote system upgrade circuitry to query the cause of error and which application image failed.
4. If a second error occurs, the device waits for a reconfiguration source. If the **Auto-restart configuration after error** is enabled, the device will reconfigure without waiting for any reconfiguration source.
5. Reconfiguration is triggered by the following actions:

- Driving the nSTATUS low externally.
- Driving the nCONFIG low externally.
- Driving RU_nCONFIG low.

2.2.1.2. Remote System Upgrade Circuitry

Figure 4. Remote System Upgrade Circuitry



The remote system upgrade circuitry does the following functions:

- Tracks the current state of configuration
- Monitors all reconfiguration sources
- Provides access to set up the application configuration image
- Returns the device to fallback configuration if an error occurs
- Provides access to the information on the failed application configuration image

2.2.1.2.1. Remote System Upgrade Circuitry Signals

Table 9. Remote System Upgrade Circuitry Signals for Intel MAX 10 Devices

Core Signal Name	Logical Signal Name	Input/Output	Description
RU_DIN	regin	Input	Use this signal to write data to the shift register on the rising edge of RU_CLK. To load data to the shift register, assert RU_SHIFThLD.
RU_DOUT	regout	Output	Use this signal to get output data from the shift register. Data is clocked out on each rising edge of RU_CLK if RU_SHIFThLD is asserted.
RU_nRSTIMER	rsttimer	Input	<ul style="list-style-type: none"> Use this signal to reset the user watchdog timer. A falling edge of this signal triggers a reset of the user watchdog timer. To reset the timer, pulse the RU_nRSTIMER signal for a minimum of 250 ns.
RU_nCONFIG	rconfig	Input	Use this signal to reconfigure the device. Driving this signal low triggers the device to reconfigure if you enable the remote system upgrade feature.
RU_CLK	clk	Input	The clock to the remote system upgrade circuitry. All registers in this clock domain are enabled in user mode if you enable the remote system upgrade. Shift register and input register are positive edge flip-flops.
RU_SHIFThLD	shiftnld	Input	Control signals that determine the mode of remote system upgrade circuitry.
RU_CAPThUPDT	captnupdt	Input	<ul style="list-style-type: none"> When RU_SHIFThLD is driven low and RU_CAPThUPDT is driven low, the input register is loaded with the contents of the shift register on the rising edge of RU_CLK. When RU_SHIFThLD is driven low and RU_CAPThUPDT is driven high, the shift register captures values from the input_cs_ps module on the rising edge of RU_CLK. When RU_SHIFThLD is driven high, the RU_CAPThUPDT is ignored and the shift register shifts data on each rising edge of RU_CLK.

Related Information

[Intel MAX 10 FPGA Device Datasheet](#)

Provides more information about Remote System Upgrade timing specifications.

2.2.1.2.2. Remote System Upgrade Circuitry Input Control

The remote system upgrade circuitry has three modes of operation.

- Update—loads the values in the shift register into the input register.
- Capture—loads the shift register with data to be shifted out.
- Shift—shifts out data to the user logic.

Table 10. Control Inputs to the Remote System Upgrade Circuitry

Remote System Upgrade Circuitry Control Inputs				Operation Mode	Input Settings for Registers	
RU_SHIFThLD	RU_CAPThUPDT	Shift register [40]	Shift register [39]		Shift Register[38:0]	Input Register[38:0]
0	0	Don't Care	Don't Care	Update	Shift Register [38:0]	Shift Register [38:0]
0	1	0	0	Capture	Current State	Input Register[38:0]
0	1	0	1	Capture	{8'b0, Previous State Application1}	Input Register[38:0]
0	1	1	0	Capture	{8'b0, Previous State Application2}	Input Register[38:0]
0	1	1	1	Capture	Input Register[38:0]	Input Register[38:0]
1	Don't Care	Don't Care	Don't Care	Shift	{ru_din, Shift Register [38:1]}	Input Register[38:0]

The following shows examples of driving the control inputs in the remote system upgrade circuitry:

- When you drive RU_SHIFThLD high to 1'b1, the shift register shifts data on each rising edge of RU_CLK and RU_CAPThUPDT has no function.
- When you drive both RU_SHIFThLD and RU_CAPThUPDT low to 1'b0, the input register is loaded with the contents of the shift register on the rising edge of RU_CLK.
- When you drive RU_SHIFThLD low to 1'b0 and RU_CAPThUPDT high to 1'b1, the shift register captures values on the rising edge of RU_DCLK.

2.2.1.2.3. Remote System Upgrade Input Register

Table 11. Remote System Upgrade Input Register for Intel MAX 10 Devices

The details below are referring to the hardware.

Bits	Name	Description
38:14	Reserved	Reserved—set to 0.
13	ru_config_sel	<ul style="list-style-type: none"> • 0: Load configuration image 0 • 1: Load configuration image 1 This bit only works when the ru_config_sel_overwrite bit is set to 1.
12	ru_config_sel_overwrite	<ul style="list-style-type: none"> • 0: Disable overwrite CONFIG_SEL pin • 1: Enable overwrite CONFIG_SEL pin
11:0	Reserved	Reserved—set to 0.

2.2.1.2.4. Remote System Upgrade Status Registers

Table 12. Remote System Upgrade Status Register—Current State Logic Bit for Intel MAX 10 Devices

The details below are referring to the hardware.

Bits	Name	Description
33:30	msm_cs	The current state of the master state machine (MSM).
29	ru_wd_en	The current state of the enabled user watchdog timer. The default state is active high.
28:0	wd_timeout_value	The current, entire 29-bit watchdog time-out value.

Table 13. Remote System Upgrade Status Register—Previous State Bit for Intel MAX 10 Devices

The details below are referring to the hardware.

Bits	Name	Description
31	nconfig	An active high field that describes the reconfiguration sources which caused the Intel MAX 10 device to leave the previous application configuration. In the event of a tie, the higher bit order takes precedence. For example, if the nconfig and the ru_nconfig triggered at the same time, the nconfig takes precedence over the ru_nconfig.
30	crcerror	
29	nstatus	
28	wdtimer	
27:26	Reserved	Reserved—set to 0.
25:22	msm_cs	The state of the MSM when a reconfiguration event occurred. The reconfiguration will cause the device to leave the previous application configuration.
21:0	Reserved	Reserved—set to 0.

Related Information

[Dual Configuration Intel FPGA IP Core Avalon Memory-Mapped Address Map](#) on page 65

2.2.1.2.5. Master State Machine

The master state machine (MSM) tracks current configuration mode and enables the user watchdog timer.

Table 14. Remote System Upgrade Master State Machine Current State Descriptions for Intel MAX 10 Devices

msm_cs Values	State Description
0010	Image 0 is being loaded.
0011	Image 1 is being loaded after a revert in application image happens.
0100	Image 1 is being loaded.
0101	Image 0 is being loaded after a revert in application image happens.

2.2.1.3. User Watchdog Timer

The user watchdog timer prevents a faulty application configuration from stalling the device indefinitely. You can use the timer to detect functional errors when an application configuration is successfully loaded into the device.

The counter is 29 bits wide and has a maximum count value of 2^{29} . When specifying the user watchdog timer value, specify only the most significant 12 bits. The granularity of the timer setting is 2^{17} cycles. The cycle time is based on the frequency of the user watchdog timer internal oscillator. Depending on the counter and the internal oscillator of the device, you can set the cycle time from 9 ms to 244 s.

Figure 5. Watchdog Timer Formula for Intel MAX 10 Devices

$$\text{Watchdog timer time-out (seconds)} = \frac{\text{Watchdog timer value (decimal)}}{\text{Watchdog timer frequency}}$$

The timer begins counting as soon as the application configuration enters user mode. When the timer expires, the remote system upgrade circuitry generates a time-out signal, updates the status register, and triggers the loading of the revert configuration image. To reset the timer and ensure that the application configuration is valid, pulse the RU_NRSTIMER continuously for a minimum of 250 ns per reset pulse.

When you enable the watchdog timer, the setting applies to all images, all images should contain the soft logic configuration to reset the timer. Application configuration will reset the control block registers.

Related Information

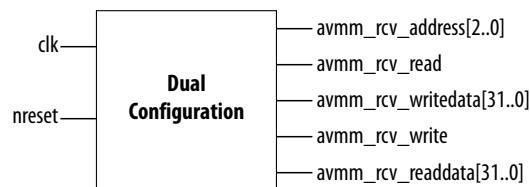
- [User Watchdog Internal Circuitry Timing Specifications](#)
Provides more information about the user watchdog frequency.
- [Initialization Configuration Bits](#) on page 11

2.2.1.4. Dual Configuration Intel FPGA IP Core

The Dual Configuration Intel FPGA IP core offers the following capabilities through Avalon® memory-mapped interface:

- Asserts RU_nCONFIG to trigger reconfiguration.
- Asserts RU_nRSTIMER to reset watchdog timer if the watchdog timer is enabled.
- Writes configuration setting to the input register of the remote system upgrade circuitry.
- Reads information from the remote system upgrade circuitry.

Figure 6. Dual Configuration Intel FPGA IP Core Block Diagram



Related Information

- [Dual Configuration Intel FPGA IP Core Avalon Memory-Mapped Address Map](#) on page 65

- [Avalon Interface Specifications](#)
Provides more information about the Avalon memory-mapped interface specifications applied in Dual Configuration Intel FPGA IP core.
- [Instantiating the Dual Configuration Intel FPGA IP Core](#) on page 64
- [Dual Configuration Intel FPGA IP Core References](#) on page 65
- [Remote System Upgrade](#) on page 13
- [AN 741: Remote System Upgrade for MAX 10 FPGA Devices over UART with the Nios II Processor](#)
Provides reference design for remote system upgrade in Intel MAX 10 FPGA devices.
- [I2C Remote System Update Example](#)
This example demonstrates a remote system upgrade using the I2C protocol.

2.2.2. Configuration Design Security

Security Notice:

Intel MAX 10 devices do not include an internal mechanism to authenticate bitstreams. Intel recommends implementing system-level controls external to the Intel MAX 10 devices via additional components or via soft IP within the Intel MAX 10 devices to incorporate bitstream authentication. Alternatively, you may select a device that includes internal bitstream authentication such as Intel Stratix® 10 or Intel Agilex® 7 devices.

The Intel MAX 10 design security feature supports the following capabilities:

- Encryption—Built-in encryption standard (AES) to support 128-bit key industry-standard design security algorithm
- Chip ID—Unique device identification
- JTAG secure mode—limits access to JTAG instructions
- Verify Protect—allows optional disabling of CFM content read-back
- Flash region access control and immutability—enable immutability of the flash region CFM0 when configured from CFM1. You can leverage this feature to implement a Device Identifier Composition Engine (DICE) compliant design.

Note: The flash region access control and immutability feature is only available in Intel MAX 10 devices with DD feature options (10M40DD and 10M50DD).

2.2.2.1. AES Encryption Protection

The Intel MAX 10 design security feature provides the following security protection for your designs:

- Helps protect against copying—the non-volatile key is securely stored in the Intel MAX 10 devices and cannot be read through any interface. Without this key, attackers are unable to decrypt the encrypted configuration image.
- Helps resist reverse engineering—reverse engineering from an encrypted configuration file is very difficult and time consuming because the file requires decryption.
- Helps defend against tampering—after you enable the JTAG Secure and Encrypted POF (EPOF) only, the Intel MAX 10 device can only accept configuration files encrypted with the same key. Additionally, configuration through the JTAG interface is blocked.

Related Information

[Generating .pof using Convert Programming Files](#) on page 41

2.2.2.1.1. Encryption and Decryption

Intel MAX 10 supports AES encryption. Programming bitstream is encrypted based on the encryption key that is specified by you. In Intel MAX 10 devices, the key is part of the ICB settings stored in the internal flash, making it a non-volatile key. However, you can clear/delete the key by performing a full chip erase on the device.

When you use compression with encryption, the configuration file is first compressed, and then encrypted using the Intel Quartus Prime software. During configuration, the device first decrypts, and then decompresses the configuration file.

The header and I/O configuration shift register (IOCSR) data will not be encrypted. The decryption block is activated after the IOCSR chain is programmed. The decryption block only decrypts core data and postamble.

Related Information

[JTAG Instruction Availability](#) on page 23

2.2.2.2. Unique Chip ID

Unique chip ID provides the following features:

- Identifies your device in your design as part of a security feature to protect your design from an unauthorized device.
- Provides non-volatile 64-bits unique ID for each Intel MAX 10 device with write protection.

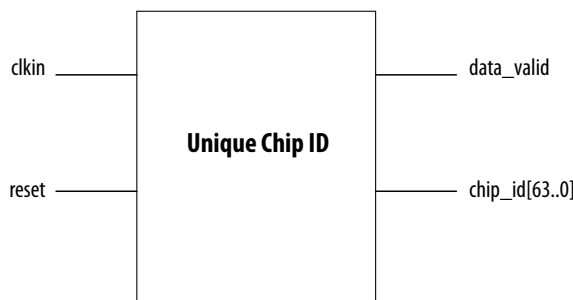
You can use the Unique Chip ID Intel FPGA IP core to acquire the chip ID of your Intel MAX 10 device.

Related Information

- [Unique Chip ID Intel FPGA IP Core](#) on page 63
- [Unique Chip ID Intel FPGA IP Core Ports](#) on page 68

2.2.2.2.1. Unique Chip ID Intel FPGA IP Core

Figure 7. Unique Chip ID Intel FPGA IP Core Block Diagram



At the initial state, the `data_valid` signal is low because no data is read from the unique chip ID block. After feeding a clock signal to the `clkin` input port, the Unique Chip ID Intel FPGA IP core begins to acquire the chip ID of your device through the unique chip ID block. After acquiring the chip ID of your device, the Unique Chip ID Intel FPGA IP core asserts the `data_valid` signal to indicate that the chip ID value at the output port is ready for retrieval.

The operation repeats only when you provide another clock signal when the `data_valid` signal is low. If the `data_valid` signal is high when you provide another clock signal, the operation stops because the `chip_id[63:0]` output holds the chip ID of your device.

A minimum of 67 clock cycles are required for the `data_valid` signal to go high.

The `chip_id[63:0]` output port holds the value of chip ID of your device until you reconfigure the device or reset the Unique Chip ID Intel FPGA IP core.

2.2.2.3. JTAG Secure Mode

In JTAG Secure mode, the device only allows mandatory IEEE 1149.1 JTAG instructions to be exercised.

You can enable the JTAG secure when generating the `.pof` in the Convert Programming Files. To exit JTAG secure mode, issue the `UNLOCK` JTAG instruction. The `LOCK` JTAG instruction puts the device in the JTAG secure mode again. The `LOCK` and `UNLOCK` JTAG instructions can only be issued through the JTAG core access. Refer to [JTAG Instruction Availability](#) for list of available instructions.

Related Information

- [JTAG Instruction Availability](#) on page 23
- [Configuration Flash Memory Permissions](#) on page 23
- [JTAG Secure Design Example](#)
- [Generating .pof using Convert Programming Files](#) on page 41

2.2.2.3.1. JTAG Secure Mode Instructions

Table 15. JTAG Secure Mode Instructions for Intel MAX 10 Devices

JTAG Instruction	Instruction Code	Description
LOCK	10 0000 0010	<ul style="list-style-type: none"> • Activates the JTAG secure mode. • Blocks access from both external pins and core to JTAG.
UNLOCK	10 0000 1000	Deactivates the JTAG secure mode.

2.2.2.4. Verify Protect

Verify Protect is a security feature to enhance CFM security. When you enable the **Verify Protect**, only program and erase operation are allowed on the CFM. This capability protects the CFM contents from being copied.

You can turn on the **Verify Protect** feature when converting the `.sof` file to `.pof` file in the Intel Quartus Prime Convert Programming File tool.

Related Information

- [Configuration Flash Memory Permissions](#) on page 23
- [Generating .pof using Convert Programming Files](#) on page 41

2.2.2.5. JTAG Instruction Availability

Table 16. JTAG Instruction Availability Based on JTAG Secure Mode and Encryption Settings

JTAG Secure Mode	Encryption	Description
Disabled	Disabled	All JTAG Instructions enabled
	Enabled	All JTAG Instructions are enabled except: <ul style="list-style-type: none"> • CONFIGURE
Enabled	Disabled	All non-mandatory IEEE 1149.1 JTAG instructions are disabled except: <ul style="list-style-type: none"> • SAMPLE/PRELOAD • BYPASS • EXTEST • IDCODE • UNLOCK • LOCK
	Enabled	

Related Information

- [JTAG Secure Mode](#) on page 22
- [Intel MAX 10 JTAG Secure Design Example](#) on page 58
- [JTAG Secure Design Example](#)
- [Encryption and Decryption](#) on page 21

2.2.2.6. Configuration Flash Memory Permissions

The JTAG secure mode and verify protect features determines the CFM operation permission. The table list the operations permitted based on the security settings.

Table 17. CFM Permissions for Intel MAX 10 Devices

Operation	JTAG Secure Mode Disabled		JTAG Secure Mode Enabled	
	Verify Protect Disabled	Verify Protect Enabled	Verify Protect Disabled	Verify Protect Enabled
ISP through core	Illegal operation	Illegal operation	Illegal operation	Illegal operation
ISP through JTAG pins	Full access	Program and erase only	No access	No access
Real-time ISP through core	Full access	Program and erase only	No access	No access
Real-time ISP through JTAG pins	Full access	Program and erase only	No access	No access
UFM interface through core ⁽⁶⁾	Full access	Full access	Full access	Full access

⁽⁶⁾ The UFM interface through core is available if you select the dual compressed image mode.

Related Information

- [JTAG Secure Mode](#) on page 22
- [Intel MAX 10 JTAG Secure Design Example](#) on page 58
- [JTAG Secure Design Example](#)
- [Verify Protect](#) on page 22
- [Generating .pof using Convert Programming Files](#) on page 41
- [Intel MAX 10 User Flash Memory User Guide](#)

2.2.2.6.1. Flash Region Access Control and Immutability

The Intel MAX 10 DD (10M40DD, 10M50DD) devices enable immutability of the flash region CFM0 when configured from CFM1.

The access permission of CFM0 is updated while the other flash region access permission remains the same as the standard Intel MAX 10 devices.

The JTAG secure mode and verify protect features determine the CFM operation permission. The table list the operations permitted based on the security settings.

Table 18. CFM Permissions for Intel MAX 10 DD Devices

Operation	JTAG Secure Mode Disabled		JTAG Secure Mode Enabled	
	Verify Protect Disabled	Verify Protect Enabled	Verify Protect Disabled	Verify Protect Enabled
ISP through core	Illegal operation	Illegal operation	Illegal operation	Illegal operation
ISP through JTAG pins	Full access with condition ⁽⁷⁾	Program and erase only ⁽⁸⁾	No access	No access
Real-time ISP through core	Full access with condition ⁽⁷⁾	Program and erase only ⁽⁸⁾	No access	No access
Real-time ISP through JTAG pins	Full access with condition ⁽⁷⁾	Program and erase only ⁽⁸⁾	No access	No access
UFM interface through core ⁽⁹⁾	Full access with condition ⁽⁷⁾	Full access with condition ⁽⁷⁾	Full access with condition ⁽⁷⁾	Full access with condition ⁽⁷⁾

⁽⁷⁾ Full access with condition:

- Dual image configuration—full access to CFM0 is applicable when the device is configured with CFM0 image. Otherwise, no access.
- Single image configuration—full access.

⁽⁸⁾ Program and erase only with condition:

- Dual image configuration—program and erase only to CFM0 is applicable when the device is configured with CFM0 image. Otherwise, no access.
- Single image configuration—program and erase only.

⁽⁹⁾ The UFM interface through core is available if you select the dual compressed image mode.

2.2.3. SEU Mitigation and Configuration Error Detection

The dedicated circuitry built in Intel MAX 10 devices consists of an error detection cyclic redundancy check (EDCRC) feature. You can use this feature to mitigate single-event upset (SEU) or soft errors.

The hardened on-chip EDCRC circuitry allows you to perform the following operations without any impact on the fitting of the device:

- Auto-detection of cyclic redundancy check (CRC) errors during configuration.
- Identification of SEU in user mode with the optional CRC error detection.
- Testing of error detection by error detection verification through the JTAG interface.

Related Information

- [Verifying Error Detection Functionality](#) on page 48
- [Enabling Error Detection](#) on page 49
- [Accessing Error Detection Block Through User Logic](#) on page 49

2.2.3.1. Configuration Error Detection

In configuration mode, a frame-based CRC is stored in the configuration data and contains the CRC value for each data frame.

During configuration, the Intel MAX 10 device calculates the CRC value based on the frame of data that is received and compares it against the frame CRC value in the data stream. Configuration continues until the device detects an error or when all the values are calculated.

For Intel MAX 10 devices, the CRC is computed by the Intel Quartus Prime software and downloaded into the device as part of the configuration bit stream. These devices store the CRC in the 32-bit storage register at the end of the configuration mode.

2.2.3.2. User Mode Error Detection

SEUs are changes in a CRAM bit state due to an ionizing particle. Intel MAX 10 devices have built-in error detection circuitry to detect data corruption in the CRAM cells.

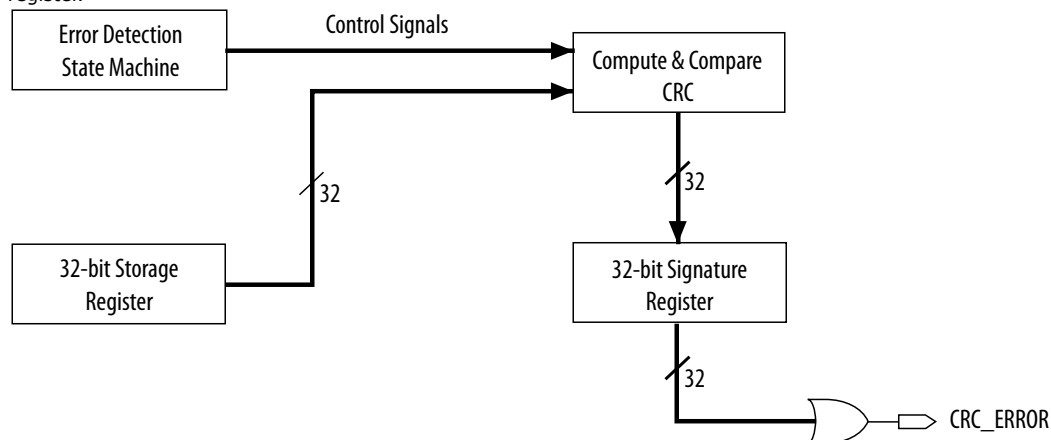
This error detection capability continuously computes the CRC of the configured CRAM bits. The CRC of the contents of the device are compared with the pre-calculated CRC value obtained at the end of the configuration. If the CRC values match, there is no error in the current configuration CRAM bits. The process of error detection continues until the device is reset—by setting `nCONFIG` to low.

The error detection circuitry in Intel MAX 10 device uses a 32-bit CRC IEEE Std. 802 and a 32-bit polynomial as the CRC generator. Therefore, the device performs a single 32-bit CRC calculation. If an SEU does not occur, the resulting 32-bit signature value is `0x000000`, which results in a 0 on the output signal `CRC_ERROR`. If an SEU occurs in the device, the resulting signature value is non-zero and the `CRC_ERROR` output signal is 1. You must decide whether to reconfigure the FPGA by strobing the `nCONFIG` pin low or ignore the error.

2.2.3.2.1. Error Detection Block

Figure 8. Error Detection Block Diagram

Error detection block diagram including the two related 32-bit registers—the signature register and the storage register.



There are two sets of 32-bit registers in the error detection circuitry that store the computed CRC signature and pre-calculated CRC value. A non-zero value on the signature register causes the CRC_ERROR pin to go high.

Table 19. Error Detection Registers for Intel MAX 10 Devices

Register	Description
32-bit signature register	This register contains the CRC signature. The signature register contains the result of the user mode calculated CRC value compared against the pre-calculated CRC value. If no errors are detected, the signature register is all zeros. A non-zero signature register indicates an error in the configuration CRAM contents. The CRC_ERROR signal is derived from the contents of this register.
32-bit storage register	This register is loaded with the 32-bit pre-computed CRC signature at the end of the configuration stage. The signature is then loaded into the 32-bit Compute and Compare CRC block during user mode to calculate the CRC error. This register forms a 32-bit scan chain during execution of the CHANGE_EDREG JTAG instruction. The CHANGE_EDREG JTAG instruction can change the content of the storage register. Therefore, the functionality of the error detection CRC circuitry is checked in-system by executing the instruction to inject an error during the operation. The operation of the device is not halted when issuing the CHANGE_EDREG JTAG instruction.

2.2.3.2.2. CHANGE_EDREG JTAG Instruction

Table 20. CHANGE_EDREG JTAG Instruction Description

JTAG Instruction	Instruction Code	Description
CHANGE_EDREG	00 0001 0101	This instruction connects the 32-bit CRC storage register between TDI and TDO. Any precomputed CRC is loaded into the CRC storage register to test the operation of the error detection CRC circuitry at the CRC_ERROR pin.

2.2.3.3. Error Detection Timing

When the error detection CRC feature is enabled through the Intel Quartus Prime software, the device automatically activates the CRC process upon entering user mode, after configuration and initialization is complete.

The CRC_ERROR pin remains low until the error detection circuitry has detected a corrupted bit in the previous CRC calculation. After the pin goes high, it remains high during the next CRC calculation. This pin does not log the previous CRC calculation. If the new CRC calculation does not contain any corrupted bits, the CRC_ERROR pin is driven low. The error detection runs until the device is reset.

The error detection circuitry is clocked by an internal configuration oscillator with a divisor that sets the maximum frequency. The CRC calculation time depends on the device and the error detection clock frequency.

Related Information

[Enabling Error Detection](#) on page 49

2.2.3.3.1. Error Detection Frequency

You can set a lower clock frequency by specifying a division factor in the Intel Quartus Prime software.

Table 21. Minimum and Maximum Error Detection Frequencies for Intel MAX 10 Devices

Device	Error Detection Frequency	Maximum Error Detection Frequency (MHz)	Minimum Error Detection Frequency (kHz)	Valid Values for n
10M02	55 MHz/2 ⁿ to 116 MHz/2 ⁿ	58	214.8	1, 2, 3, 4, 5, 6, 7, 8
10M04				
10M08				
10M16				
10M25				
10M40	35 MHz/2 ⁿ to 77 MHz/2 ⁿ	38.5	136.7	
10M50				

2.2.3.3.2. Cyclic Redundancy Check Calculation Timing

Table 22. Cyclic Redundancy Check Calculation Time for Intel MAX 10 Devices

Device	Divisor Value (n = 2)	
	Minimum Time (ms)	Maximum Time (ms)
10M02/10M02SCU324	2/6	6.6/15.7
10M04	6	15.7
10M08	6	15.7
10M16	10	25.5
10M25	14	34.7
10M40	43	106.7
10M50	43	106.7

Figure 9. CRC Calculation Formula

You can use this formula to calculate the CRC calculation time for divisor other than 2.

$$CRC\ Calculation\ Time_{Divisor\ n} = CRC\ Calculation\ Time_{Divisor\ 2} \times \frac{n}{2}$$

Example 1. CRC Calculation Example

For 10M16 device with divisor value of 256:

Minimum CRC calculation time for divisor 256 = $10 \times (256/2) = 1280$ ms

2.2.3.4. Recovering from CRC Errors

The system that Intel MAX 10 resides in must control device reconfiguration. After detecting an error on the `CRC_ERROR` pin, strobing the `nCONFIG` pin low directs the system to perform reconfiguration at a time when it is safe for the system to reconfigure the Intel MAX 10 device.

When the data bit is rewritten with the correct value by reconfiguring the device, the device functions correctly.

While SEUs are uncommon in Intel FPGA devices, certain high-reliability applications might require a design to account for these errors.

2.2.4. Configuration Data Compression

Intel MAX 10 devices can receive compressed configuration bitstream and decompress the data in real-time during configuration. This feature helps to reduce the configuration image size stored in the CFM. Data indicates that compression typically reduces the configuration file size by at least 30% depending on the design.

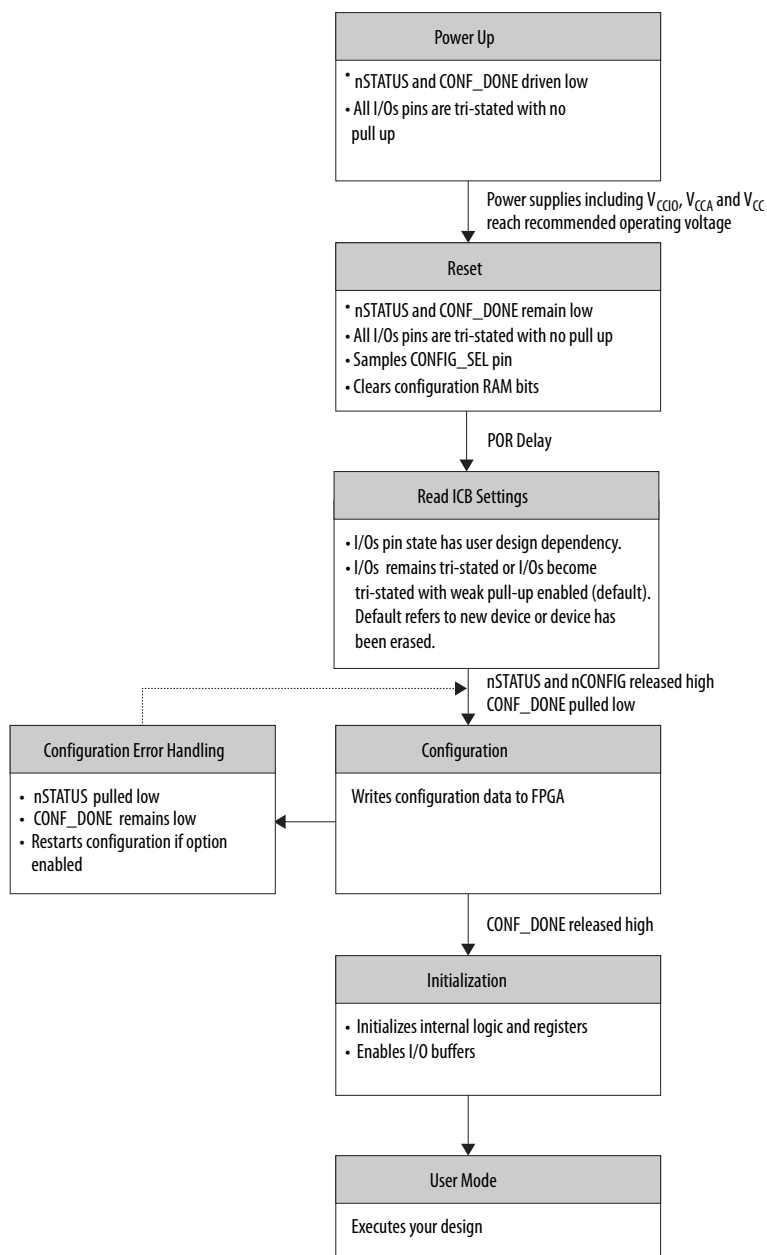
Related Information

- [Enabling Compression Before Design Compilation](#) on page 51
- [Enabling Compression After Design Compilation](#) on page 52

2.3. Configuration Details

2.3.1. Configuration Sequence

Figure 10. Configuration Sequence for Intel MAX 10 Devices



You can initiate reconfiguration by pulling the nCONFIG pin low to at least the minimum $t_{RU_nCONFIG}$ low-pulse width. When this pin is pulled low, the nSTATUS and CONF_DONE pins are pulled low and all I/O pins are either tied to an internal weak pull-up or tri-stated based on the ICB settings.

Related Information

[Generating .pof using Convert Programming Files](#) on page 41

Provides more information about how to set the weak pull-up during configuration.

2.3.1.1. Power-up

If you power-up a device from the power-down state, you need to power the V_{CCIO} for bank 1B (bank 1 for 10M02 devices), bank 8 and the core to the appropriate level for the device to exit POR. The Intel MAX 10 device enters the configuration stage after exiting the power-up stage with a small POR delay.

Related Information

- [Intel MAX 10 Power Management User Guide](#)
Provides more information about power supply modes in Intel MAX 10 devices
- [Intel MAX 10 FPGA Device Datasheet](#)
Provides more information about the ramp-up time specifications.
- [Intel MAX 10 FPGA Device Family Pin Connection Guideline](#)
Provides more information about configuration pin connections.

2.3.1.1.1. POR Monitored Voltage Rails for Single-supply and Dual-supply Intel MAX 10 Devices

To begin configuration, the required voltages must be powered up to the appropriate voltage levels as shown in the following table. The V_{CCIO} for bank 1B (bank 1 for 10M02 devices) and bank 8 must be powered up to a voltage between 1.5V – 3.3V during configuration.

Table 23. POR Monitored Voltage Rails for Single-supply and Dual-supply Intel MAX 10 Devices

There is no power-up sequence required when powering-up the voltages.

Power Supply Device Options	Power Supply Monitored by POR
Single-supply	Regulated V_{CC_ONE}
	V_{CCA}
	V_{CCIO} bank 1B ⁽¹⁰⁾ and bank 8
Dual-supply	V_{CC}
	V_{CCA}
	V_{CCIO} bank 1B ⁽¹⁰⁾ and bank 8

⁽¹⁰⁾ Bank 1 for 10M02 devices

2.3.1.1.2. Monitored Power Supplies Ramp Time Requirement for Intel MAX 10 Devices

Figure 11. Monitored Power Supplies Ramp Time Requirement Diagram for Intel MAX 10 Devices

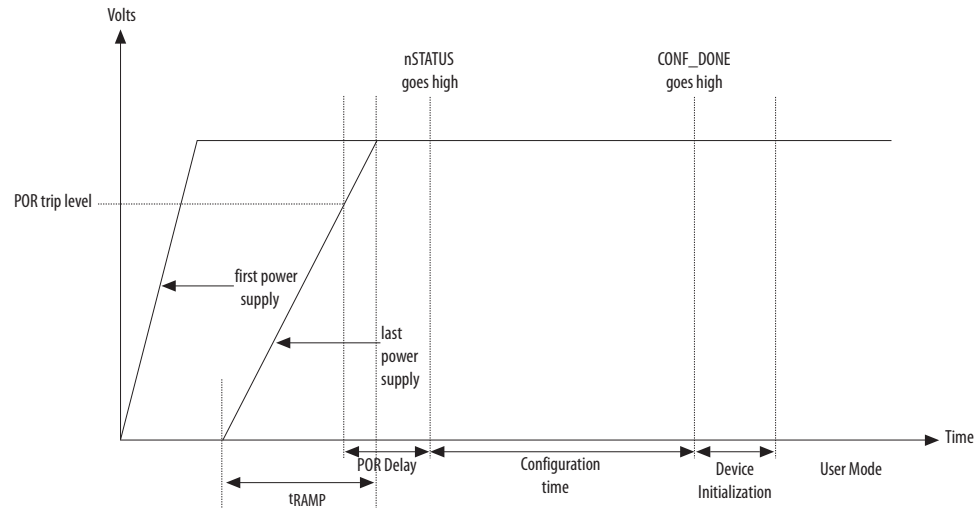


Table 24. Monitored Power Supplies Ramp Time Requirement for Intel MAX 10 Devices

Symbol	Parameter	Minimum	Maximum	Unit
t_{RAMP}	Power Supply Ramp Time ⁽¹¹⁾	— ⁽¹²⁾	10	ms

2.3.1.2. Configuration

During configuration, configuration data is read from the internal flash and written to the CRAM.

2.3.1.3. Configuration Error Handling

To restart configuration automatically, turn on the **Auto-restart configuration after error** option in the **General** page of the **Device and Pin Options** dialog box in the Intel Quartus Prime software.

If you do not turn on this option, you can monitor the `nSTATUS` pin to detect errors. To restart configuration, pull the `nCONFIG` pin low for at least the duration of $t_{RU_nCONFIG}$.

2.3.1.4. Initialization

The initialization sequence begins after the `CONF_DONE` pin goes high. The initialization clock source is from the internal oscillator and the Intel MAX 10 device receives enough clock cycles for proper initialization.

⁽¹¹⁾ Ensure that all V_{CCIO} power supply reaches full rail before configuration completes. See [Internal Configuration Time](#) on page 12.

⁽¹²⁾ There is no absolute minimum value for the ramp rate requirement. Intel characterized the minimum t_{RAMP} of 200 μ s.

2.3.1.5. User Mode

After the initialization completes, your design starts executing. The user I/O pins then function as specified by your design.

2.3.2. Intel MAX 10 Configuration Pins

All configuration pins and JTAG pins in Intel MAX 10 devices are dual-purpose pins. The configuration pins function as configuration pins prior to user mode. When the device is in user mode, they function as user I/O pins or remain as configuration pins.

Table 25. Configuration Pin Summary for Intel MAX 10 Devices

All pins are powered by V_{CCIO} Bank 1B (bank 1 for 10M02 devices) and 8.

Configuration Pin	Input/Output	Configuration Scheme
CRC_ERROR	Output only, open-drain	Optional, JTAG and internal configurations
CONFIG_SEL	Input only	Internal configuration
DEV_CLRn	Input only	Optional, JTAG and internal configurations
DEV_OE	Input only	Optional, JTAG and internal configurations
CONF_DONE	Bidirectional, open-drain	JTAG and internal configurations
nCONFIG	Input only	JTAG and internal configurations
nSTATUS	Bidirectional, open-drain	JTAG and internal configurations
JTAGEN	Input only	Optional, JTAG configuration
TCK	Input only	JTAG configuration
TDO	Output only	JTAG configuration
TMS	Input only	JTAG configuration
TDI	Input only	JTAG configuration

Related Information

- [Guidelines: Dual-Purpose Configuration Pin](#) on page 34
- [Enabling Dual-Purpose Pin](#) on page 35

3. Intel MAX 10 FPGA Configuration Design Guidelines

3.1. Dual-Purpose Configuration Pins

3.1.1. Guidelines: Dual-Purpose Configuration Pin

To use configuration pins as user I/O pins in user mode, you have to adhere to the following guidelines.

Table 26. Dual-Purpose Configuration Pin Guidelines for Intel MAX 10 Devices

Guidelines	Pins
Configuration pins during initialization: <ul style="list-style-type: none"> • Tri-state the external I/O driver and drive an external pull-up resistor⁽¹³⁾ or • Use the external I/O driver to drive the pins to the state same as the external weak pull-up resistor 	<ul style="list-style-type: none"> • nCONFIG • nSTATUS • CONF_DONE
JTAG pins: <ul style="list-style-type: none"> • If you intend to switch back and forth between user I/O pins and JTAG pin functions using the JTAGEN pin, all JTAG pins must be assigned as single-ended I/O pins or voltage-referenced I/O pins. Schmitt trigger input is the recommended input buffer. • JTAG pins cannot perform as JTAG pins in user mode if you assign any of the JTAG pin as a differential I/O pin. • You must use the JTAG pins as dedicated pins and not as user I/O pins during JTAG programming. • Do not toggle JTAG pin during the initialization stage. • Put the test access port (TAP) controller in reset state by driving the TDI and TMS pins high and toggle the TCK pin for at least 5 clock cycles before the initialization. • The Signal Tap logic analyzer IP, JTAG-to-Avalon master bridge IP, and other JTAG-related IPs cannot be used if you enable the JTAG pin sharing feature in your design. 	<ul style="list-style-type: none"> • TDO • TMS • TCK • TDI

Attention: Assign all JTAG pins as single-ended I/O pins or voltage-referenced I/O pins if you enable JTAG pin sharing feature.

Related Information

- [Intel MAX 10 FPGA Device Family Pin Connection Guidelines](#)
Provides more information about recommended resistor values.
- [Intel MAX 10 General Purpose I/O User Guide](#)
Provides more information about Schmitt trigger input.
- [Intel MAX 10 Configuration Pins](#) on page 33
- [JTAG Pins](#) on page 5

⁽¹³⁾ If you intend to remove the external weak pull-up resistor, Intel recommends that you remove it after the device enters user mode.

3.1.1.1. JTAG Pin Sharing Behavior

Table 27. JTAG Pin Sharing Behavior for Intel MAX 10 Devices

Configuration Stage	JTAG Pin Sharing	JTAGEN Pin	JTAG Pins (TDO, TDI, TCK, TMS)
User mode	Disabled	User I/O pin	Dedicated JTAG pins.
	Enabled	Driven low	User I/O pins.
		Driven high	Dedicated JTAG pins.
Configuration	Don't Care	Not used	Dedicated JTAG pins.

Note: You have to set the pins according to [Dual-Purpose Configuration Pin Guidelines for Intel MAX 10 Devices](#) and with correct pin direction (input, output or bidirectional) for the JTAG pins work correctly.

3.1.2. Enabling Dual-Purpose Pin

To use the configuration and JTAG pins as user I/O in user mode, you must do the following in the Intel Quartus Prime software:

1. On the **Assignments** menu, click **Device**. The **Device** dialog box appears.
2. In the **Device** dialog box, click **Device and Pin Options**. The **Device and Pin Options** dialog box appears.
3. In the **Device and Pin Option** dialog box, select **General** from the category pane.
4. In the Options list, do the following:
 - Check the **Enable JTAG pin sharing**.
 - Uncheck the **Enable nCONFIG, nSTATUS, and CONF_DONE pins**.

Note: Unchecking this option allows the nCONFIG, nSTATUS, and CONF_DONE pins to turn into user I/Os in user mode.

5. Click **OK**.

Related Information

- [Intel MAX 10 Configuration Pins](#) on page 33
- [JTAG Pins](#) on page 5

3.2. Configuring Intel MAX 10 Devices using JTAG Configuration

The Intel Quartus Prime software generates a .sof that you can use for JTAG configuration. You can directly configure the Intel MAX 10 device by using a download cable with the Intel Quartus Prime software programmer.

Alternatively, you can use the JAM Standard Test and Programming Language (STAPL) Format File (.jam), JAM Byte Code File (.jbc), or Serial Vector Format (.svf) with other third-party programming tools. You can either:

- Auto-generate these files
- Manually convert them using Intel Quartus Prime Programmer

Related Information

AN 425: Using the Command-Line Jam STAPL Solution for Device Programming

3.2.1. Auto-Generating Configuration Files for Third-Party Programming Tools

To generate the third-party programming tool files, perform the following steps:

1. On the **Assignments** menu, click **Settings**. The **Settings** dialog box appears.
2. In the **Category** list, select **Device**. The **Device** page appears.
3. Click **Device and Pin Options**.
4. In the **Device and Pin Options** dialog box, select the **Configuration Files** from the category pane.
5. Select the programming file you want to generate.

Note: The Intel Quartus Prime software generates two files for each optional programming file you selected. For example:

- `<project_name>.jbc`—This is the `.sof` equivalent file. Use this file to perform JTAG configuration.
- `<project_name>_pof.jbc`—This is the `.pof` equivalent file. Use this file to perform Internal configuration.

6. Click OK once setting is completed.

3.2.2. Generating Third-Party Programming Files using Intel Quartus Prime Programmer

To convert a `.sof` or `.pof` file to `.jam`, `.jbc`, or `.svf` file, perform the following steps:

1. On the **Tools** menu, click **Programmer**.
2. Click **Add File** and select the programming file and click **Open**.
3. On the Intel Quartus Prime Programmer menu, select **File ► Create/Update ► Create Jam, SVF, or ISC File**.
4. In the **File Format** list, select the format you want to generate.

Note: The generated file name does not indicate whether it was converted from a `.sof` or a `.pof` file. You can rename the generated file to avoid future confusion.

Generating Third-Party Programming Files using Command Line

Alternatively, you can generate third-party programming files through command line. Perform the following steps:

1. Run the following command to generate `.svf` file with JTAG voltage of 3.3 V and JTAG frequency of 10 MHz from `.pof` file without real-time ISP mode turned on.

```
quartus_cpf -c -q 10MHz -g 3.3 -n p <input_pof_file> <output_svf_file>
```

Similarly, JAM and JBC can be generated through the following command line.

```
quartus_cpf -c <input_pof_file> <output_jam/jbc_file>
```


2. Run the following command to generate .svf file with voltage of 3.3 V and JTAG frequency of 10 MHz from .pof file with real-time ISP mode turned on.

```
quartus_cpf -c -q 10MHz -g 3.3 -n p <input_pof_file> <output_svf_file> -o  
background_programming=on
```

Similarly, JAM and JBC can be generated through the following command line.

```
quartus_cpf -c <input_pof_file> <output_jam/jbc_file> -o  
background_programming=on
```

For more information, run the following command to understand the details of each option.

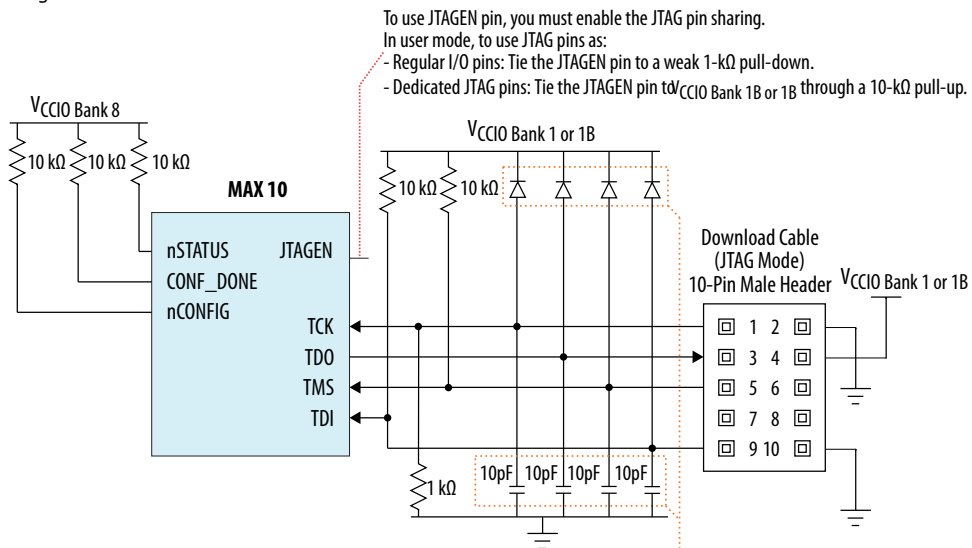
```
quartus_cpf --help=<option>
```

at which <option> can be jam, jbc, or svf.

3.2.3. JTAG Configuration Setup

Figure 12. Connection Setup for JTAG Single-Device Configuration using Download Cable

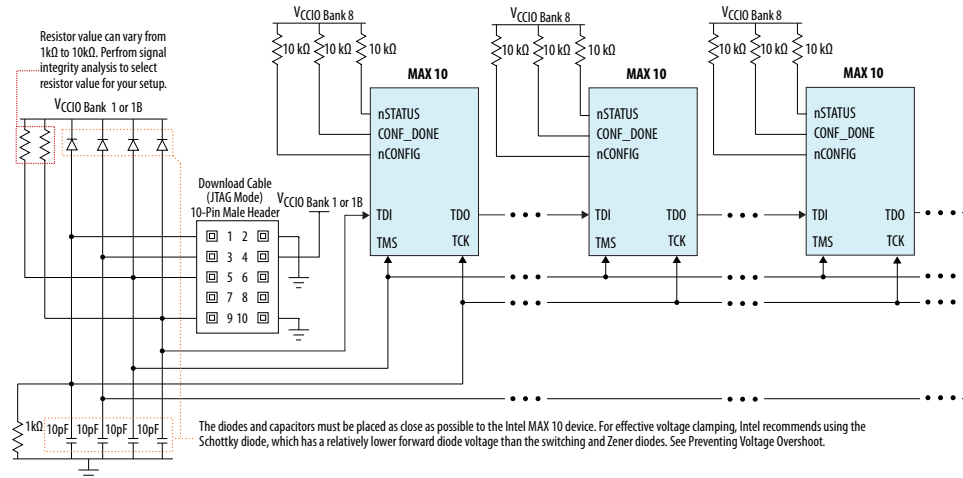
Connect to V_{CCIO} Bank 1 for 10M02 devices or V_{CCIO} Bank 1B for all other Intel MAX 10 devices. Since both multiple-supply and single-supply devices require external V_{CCIO} supplies, connect both type devices as shown in this figure.



The diodes and capacitors must be placed as close as possible to the Intel MAX 10 device. For effective voltage clamping, Intel recommends using Schottky diode, which has a relatively lower forward diode voltage than the switching and Zener diodes. See Preventing Voltage Overshoot.

Figure 13. Connection Setup for JTAG Multi-Device Configuration using Download Cable

Connect to V_{CCIO} Bank 1 for 10M02 devices or V_{CCIO} Bank 1B for all other Intel MAX 10 devices. Since both multiple-supply and single-supply devices require external VCCIO supplies, connect both type devices as shown in this figure.



To configure a device in a JTAG chain, the programming software sets the other devices to bypass mode. A device in bypass mode transfers the programming data from the TDI pin to the TDO pin through a single bypass register. The configuration data is available on the TDO pin one clock cycle later.

The Intel Quartus Prime software uses the CONF_DONE pin to verify the completion of the configuration process through the JTAG port:

- CONF_DONE pin is low—indicates that the configuration has failed.
- CONF_DONE pin is high—indicates that the configuration was successful.

After the configuration data is transmitted serially using the JTAG TDI port, the TCK port is clocked to perform device initialization.

Preventing Voltage Overshoot

To prevent voltage overshoot, you must use external diodes and capacitors if maximum AC voltage for both VCCIO and JTAG header exceed 3.9 V. However, Intel recommends that you use the external diodes and capacitors if the supplies exceed 2.5 V.

JTAGEN

If you use the JTAGEN pin, Intel recommends the following settings:

- Once you entered user mode and JTAG pins are regular I/O pins—connect the JTAGEN pin to a weak pull-down (1 kΩ).
- Once you entered user mode and JTAG pins are dedicated pins—connect the JTAGEN pin to a weak pull-up (10 kΩ).

Note: Intel recommends that you use three-pin header with a jumper or other switching mechanism to change the JTAG pins behavior.

3.2.4. ICB Settings in JTAG Configuration

The ICB settings are loaded into the device during `.pof` programming of the internal configuration scheme. The `.sof` used during JTAG configuration only programs the CRAM and does not contain ICB settings.

The Intel Quartus Prime Programmer behavior differs based on the following:

- Device without ICB settings—ICB settings cleared from the internal flash or new device
- Device with ICB settings—prior ICB settings programmed using `.pof`

Devices without ICB Settings

For devices without ICB settings, the default value is used. However, Intel Quartus Prime Programmer disables the user watchdog timer by setting the Watchdog Timer Enable bit to 0. This step is to avoid any unwanted reconfiguration from occurring due to user watchdog timeout.

If the default ICB setting is undesired, you can program the desirable ICB setting first by using `.pof` programming before doing the JTAG configuration.

Devices with ICB Settings

For device with ICB settings, the settings are preserved until the internal flash is erased. You can refer to the `.map` file to view the preserved ICB settings. JTAG configuration follows the preserved ICB setting and behave accordingly.

If the prior ICB setting is undesired, you can program the desirable ICB setting first by using `.pof` programming before doing the JTAG configuration.

Related Information

- [.pof and ICB Settings](#) on page 40
- [Verify Protect](#) on page 22
- [JTAG Secure Mode](#) on page 22
- [ISP and Real-Time ISP Instructions](#) on page 10
- [User Watchdog Timer](#) on page 18
- [Generating .pof using Convert Programming Files](#) on page 41
Provides more information about setting the ICB during `.pof` generation using Convert Programming File.

3.3. Configuring Intel MAX 10 Devices using Internal Configuration

There are three main steps for using internal configuration scheme for Intel MAX 10 devices:

1. Selecting the internal configuration scheme.
2. Generating the `.pof` with ICB settings
3. Programming the `.pof` into the internal flash

Related Information

- [Internal Configuration Modes](#) on page 6

- [Remote System Upgrade](#) on page 13

3.3.1. Selecting Internal Configuration Modes

To select the configuration mode, follow these steps:

1. Open the Intel Quartus Prime software and load a project using a Intel MAX 10 device.
2. On the **Assignments** menu, click **Device**. The **Device** dialog box appears.
3. In the **Device** dialog box, click **Device and Pin Options**. The **Device and Pin Options** dialog box appears.
4. In the **Device and Pin Option** dialog box, click the **Configuration** tab.
5. In the **Configuration Scheme** list, select **Internal Configuration**.
6. In the **Configuration Mode** list, select 1 out of 5 configuration modes available. The 10M02 devices has only 2 modes available.
7. Turn on **Generate compressed bitstreams** if needed.
8. Click **OK**.

3.3.2. .pof and ICB Settings

The internal configuration mode you select determines the corresponding method to generate the .pof and set the ICB.

Table 28. .pof Generation and ICB Setting Method for Internal Configuration Modes

Internal Configuration Mode	ICB Setting Method	.pof Generation Method
Single Compressed Image	Set the ICB in Device and Pin Options	Intel Quartus Prime software automatically generates the .pof ⁽¹⁴⁾ during project compilation.
Single Uncompressed Image		
Single Compressed Image with Memory Initialization.	Set the ICB during Convert Programming Files task.	You need to generate the .pof using Convert Programming Files .
Single Uncompressed Image with Memory Initialization		
Dual Compressed Images		

Each method is further elaborated in the following sections.

3.3.2.1. Auto-Generated .pof

To set the ICB for the auto-generated .pof, follow these steps:

1. On the **Assignments** menu, click **Device**. The **Device** dialog box appears.
2. In the **Device** dialog box, click **Device and Pin Options**. The **Device and Pin Options** dialog box appears.
3. In the **Device and Pin Option** dialog box, select **Configuration** from the category pane.

⁽¹⁴⁾ Auto-generated .pof does not allow encryption. To enable the encryption feature in Single Compressed and Single Uncompressed mode, use the Convert Programming Files method.

4. Click the **Device Options ...** button.
5. The **Max 10 Device Options** dialog box allows you to set the following:
 - a. User I/Os weak pull up during configuration.
 - b. Verify Protect.
6. To automatically generate configuration files for third-party programming tools, select the **Programming Files** from the category pane and select the format that you want to generate.

Note: The Intel Quartus Prime software generates two files for each optional programming file you selected. For example:

 - `<project_name>.jbc`—This is the `.sof` equivalent file. Use this file to perform JTAG configuration.
 - `<project_name>_pof.jbc`—This is the `.pof` equivalent file. Use this file to perform Internal configuration.
7. Click **OK** once setting is completed.

3.3.2.2. Generating .pof using Convert Programming Files

To convert `.sof` files to `.pof` files and to set the ICB using **Convert Programming Files**, follow these steps:

1. On the **File** menu, click **Convert Programming Files**.
2. Under **Output programming file**, select **Programmer Object File (.pof)** in the **Programming file type** list.
3. In the **Mode** list, select **Internal Configuration**.
4. To set the ICB settings, click **Option/Boot Info** and the **Max 10 Device Options** dialog box appears. The **Max 10 Device Options** dialog box allows you to set the following:
 - a. User I/Os weak pull up during configuration.
 - b. Configure device from CFM0 only.

Note: When you enable this feature, the device always loads configuration image 0 without sampling the physical CONFIG_SEL pin. After successfully loading configuration image 0, you can switch between configuration images using the `config_sel_overwrite` bit of the input register. Refer to related information for details about Dual Configuration Intel FPGA IP input register.
 - c. Use secondary image ISP data as default setting when available.
 - d. JTAG Secure.

Note: The JTAG Secure feature is disabled by default in Intel Quartus Prime software. To make this option visible in the GUI, you must create a `quartus.ini` file using the text editor, with the key-value pair: `PGM_ENABLE_MAX10_JTAG_SECURITY=ON` and save the file in one of the following folders:

- Project folder.
- Windows operating system: <Quartus installation folder> \bin64 folder.
- Linux operating system: <Quartus installation folder> / linux64 folder.

Caution: Intel MAX 10 FPGA device becomes permanently locked if you enabled JTAG secure mode in the POF file and POF is encrypted with the wrong key. You must instantiate the internal JTAG interface for you unlock the external JTAG when the device is in JTAG Secure mode.

- Verify Protect.
- Allow encrypted POF only.
- Watchdog timer for dual configuration and watchdog timer value (Enabled after adding 2 .sof page with two designs that compiled with Dual Compressed Internal Images).
- User Flash Memory settings.
- User Data in Configuration Flash Memory (CFM).

The function is to store user data in unused CFM0/1 space. It can be done by using Convert Programming File (CPF) GUI, command line (`quartus_cpf`), and the conversion setup file (`.cof`). CPF reports error messages for cases as shown in the following table:

Table 29. CPF Error Messages

Case	Error Message
if the start address overlaps with configuration data	Error (20646): The page CFM0 user data requested start address 0x0014BA00 overlaps with configuration data that end at address 0x0014BA6B.
if the HEX file cannot fit in unused CFM space	Error (20640): Memory file test.hex contains 256 bytes. It cannot fit in CFM0 section which has only 254 bytes unused memory space. Size of file(s) in CFM0 exceeds memory capacity.
if the start address exceeds the CFM block	Error (20648): The page CFM0 user data requested start address 0x00162000 exceeds page end address 0x00161FFF.

Convert Programming File GUI

You can specify a starting address and HEX file for insertion into CFM 0/1 in **CFM0/1 File path** and **CFM0/1 start address (32-bit hexadecimal)** respectively in the **Max 10 Device Options** dialog box. The tool automatically allocates an empty space for the HEX file if the start address remains as 0x0.

You can refer to the memory map file (`*.map`) to check the start address. The start address is after the end address of CFM0/1. For example, in the figure below, insert your data in CFM0 with start address 0x0014BA6C and end

address 0x00161FFF. The **.map** file is updated with CFM0_DATA start address and end address once you generate the POF file with **Create Memory Map** enabled.

Figure 14. Example of Memory Map File (*.map)

BLOCK	START ADDRESS	END ADDRESS
ICB	0x00000000	0x00001FFF
UFM	0x00002000	0x00071FFF
CFM0	0x00072000	0x00161FFF (0x0014BA6B)
CFM0_DATA	0x0014BA6C	0x0014BB6B

Command Line (quartus_cpf)

In the command line interface, run the `quartus_cpf` command with the following syntax:

```
quartus_cpf -c -o cfm0_source_file=<user>.hex -o
cfm0_start_address=<12345678> <user>.sof <user>.pof
```

Example of `quartus_cpf` command:

```
quartus_cpf -c -o cfm0_source_file=test.hex -o
cfm0_start_address=0014BA6C abc.sof abc.pof
```

Conversion Setup File (*.cof)

In the **Convert Programming File** dialog box, you may save the setting by click the **Save Conversion Setup...** and edit **cfm0_filepath** and **cfm_file_start_addr** in the `<user>.cof` file as shown below.

```
<MAX10_device_options>
  <por>0</por>
  <io_pullup>1</io_pullup>
  <config_from_cfm0_only>0</config_from_cfm0_only>
  <isp_source>0</isp_source>
  <verify_protect>0</verify_protect>
  <epof>0</epof>
  <ufm_source>2</ufm_source>
  <cfm0_filepath>test.hex</cfm0_filepath>
  <cfm0_file_start_addr>1358444</cfm0_file_start_addr>
</MAX10_device_options>
```

- j. RPD File Endianness.
5. In the **File name** box, specify the file name for the programming file you want to create.
6. To generate a Memory Map File (`.map`), turn on **Create Memory Map File** (Auto generate `output_file.map`). The **.map** contains the address of the CFM and UFM with the ICB setting that you set through the **Option/Boot Info** option.
7. To generate a Raw Programming Data (`.rpd`), turn on **Create config data RPD** (Generate `output_file_auto.rpd`).

Separate Raw Programming Data (.rpd) for each configuration flash memory and user flash memory (CFM0, CFM1, UFM) section are generated together for remote system upgrade purpose.

8. The .sof can be added through **Input files to convert** list and you can add up to two .sof files.

For remote system upgrade purpose, you can retain the original page 0 data in the .pof, and replace page 1 data with a new .sof file. To perform this, you must to add the .pof file in page 0, then add .sof page, then add the new .sof file to page 1.

9. After all settings are set, click **Generate** to generate related programming file.

Related Information

- [Intel MAX 10 User Flash Memory User Guide](#)
Provides more information about On-Chip Flash Intel FPGA IP core.
- [Encryption in Internal Configuration](#) on page 56
Provides more information about internal configuration image loaded based on various settings.

3.3.2.3. Generating Third-Party Programming Files using Intel Quartus Prime Programmer

To convert a .sof or .pof file to .jam, .jbc, or .svf file, perform the following steps:

1. On the **Tools** menu, click **Programmer**.
2. Click **Add File** and select the programming file and click **Open**.
3. On the Intel Quartus Prime Programmer menu, select **File > Create/Update > Create Jam, SVF, or ISC File**.
4. In the **File Format** list, select the format you want to generate.

Note: The generated file name does not indicate whether it was converted from a .sof or a .pof file. You can rename the generated file to avoid future confusion.

Generating Third-Party Programming Files using Command Line

Alternatively, you can generate third-party programming files through command line. Perform the following steps:

1. Run the following command to generate .svf file with JTAG voltage of 3.3 V and JTAG frequency of 10 MHz from .pof file without real-time ISP mode turned on.

```
quartus_cpf -c -q 10MHz -g 3.3 -n p <input_pof_file> <output_svf_file>
```

Similarly, JAM and JBC can be generated through the following command line.

```
quartus_cpf -c <input_pof_file> <output_jam/jbc_file>
```

2. Run the following command to generate .svf file with voltage of 3.3 V and JTAG frequency of 10 MHz from .pof file with real-time ISP mode turned on.

```
quartus_cpf -c -q 10MHz -g 3.3 -n p <input_pof_file> <output_svf_file> -o background_programming=on
```


Similarly, JAM and JBC can be generated through the following command line.

```
quartus_cpf -c <input_pof_file> <output_jam/jbc_file> -o  
background_programming=on
```

For more information, run the following command to understand the details of each option.

```
quartus_cpf --help=<option>
```

at which <option> can be jam, jbc, or svf.

3.3.3. Programming .pof into Internal Flash

You can use the Intel Quartus Prime Programmer to program the .pof into the CFM through JTAG interface. The Intel Quartus Prime Programmer also allows you to program the UFM part of the internal flash.

To program the .pof into the flash, follow these steps:

1. On the **Tools** menu, click **Programmer**.
2. In the **Programmer** window, click **Hardware Setup** and select **USB Blaster** in the currently selected hardware drop down list.
3. In the **Mode** list, select **JTAG**.
4. Click **Auto Detect** button on the left pane.
5. Select the device to be programmed, and click **Add File**.
6. Select the .pof to be programmed to the selected device.
7. There are several options in programming the internal flash:
 - To program any of the CFM0/CFM1/CFM2 only, select the corresponding CFM in the Program/Configure column.
 - To program the UFM only, select the UFM in the Program/Configure column.
 - To program the CFM and UFM only, select the CFM and UFM in the Program/Configure column.

Note: ICB setting is preserved in this option. However, before the programming starts, Intel Quartus Prime Programmer will make sure the ICB setting in the device and the ICB setting in the selected .pof are the same. If the ICB settings are different, Intel Quartus Prime Programmer will overwrite the ICB setting.

 - To program the whole internal flash including the ICB settings, select the <yourpoffile.pof> in the Program/Configure column.
8. To enable the real-time ISP mode, turn-on the **Enable real-time ISP to allow background programming**.
9. After all settings are set, click **Start** to start programming.

3.4. Implementing ISP Clamp in Intel Quartus Prime Software

To implement ISP clamp, you have to:

1. Create a pin state information (.ips) file. The .ips file defines the state for all the pins of the device when the device is in ISP clamp operation. You can use an existing .ips file.
2. Execute the .ips file.

Note: You can use the .ips file created to program the device with any designs, provided that it targets the same device and package. You must use the .ips file together with a POF file.

Related Information

ISP Clamp on page 9

3.4.1. Creating IPS File

To create an .ips file, perform the following steps:

1. Click **Programmer** on the toolbar, or on the **Tools** menu, click **Programmer** to open the **Programmer**.
2. Click **Add File** in the programmer to add the programming file (POF, Jam, or JBC).
3. Click on the programming file (highlights the entire row) and on the **Edit** menu, click **ISP CLAMP State Editor**.
4. Specify the states of the pins in your design in the **ISP Clamp State Editor**. By default, all pins are set to **tri-state**.
5. Click **Save** to save IPS file after making the modifications.

3.4.2. Executing IPS File

To execute ISP Clamp, perform the following steps:

1. In the Quartus Prime **Programmer**, select the .pof you want to program to the device.
2. Select the .pof, right click and select **Add IPS File** and turn-on **ISP CLAMP**.

Note: You can change the start-up delay of the I/O Clamp after configuration. To do this, select **Tools > Options**, turn-on the **Overwrite MAX10 configuration start up delay when using IO Clamp in Programmer** option, and change the delay value accordingly.

3. Select the .pof in the **Program/Configure** column.

Note: For third party programming, you can generate the .jam or .jbc file from the .pof file with .ips file.

4. After all settings are set, click **Start** to start programming.

3.5. Accessing Remote System Upgrade through User Logic

The following example shows how the input and output ports of a WYSIWYG atom are defined in the Intel MAX 10 device.

Note: WYSIWYG is a technique that performs optimization on the Verilog Quartus Mapping netlist within the Intel Quartus Prime software.

```
fiftyfivenm_rublock <rublock_name>
(
    .clk(<clock source>),
    .shiftnld(<shiftnld source>),
    .captnupdt(<captnupdt source>),
    .regin(<regin input source from the core>),
    .rsttimer(<input signal to reset the watchdog timer>),
    .rconfig(<input signal to initiate configuration>),
    .regout(<data output destination to core>)
);
defparam <rublock_name>.sim_init_config = <initial configuration for simulation only>;
defparam <rublock_name>.sim_init_watchdog_value = <initial watchdog value for simulation only>;
defparam <rublock_name>.sim_init_config = <initial status register value for simulation only>;
```

Table 30. Port Definitions

Port	Input/ Output	Definition
<rublock_name>	-	Unique identifier for the RSU Block. This is any identifier name which is legal for the given description language (e.g. Verilog, VHDL, AHDL, etc.). This field is required.
.clk(<clock source>)	Input	This signal designates the clock input of this cell. All operation of this cell are with respect to the rising edge of this clock. Whether it is the loading of the data into the cell or data out of the cell, it always occurs on the rising edge. This field is required.
.shiftnld(<shiftnld source>)	Input	This signal is an input into the remote system upgrade block. If shiftnld = 1, then data gets shifted from the internal shift registers to the regout at each rising edge of clk and it gets shifted into the internal shift registers from regin. This field is required.
.captnupdt(<captnupdt source>)	Input	This signal is an input into the remote system upgrade block. This controls the protocol of when to read the configuration mode or when to write into the registers that control the configuration. This field is required.
.regin(<regin input source from the core>)	Input	This signal is an input into the remote system upgrade block for all data being loaded into the core. The data is shifted into the internal registers at the rising edge of clk. This field is required.
.rsttimer(<input signal to reset the watchdog timer>)	Input	This signal is an input to reset the user watchdog timer. A falling edge of this signal triggers a reset of the user watchdog timer. To reset the timer, pulse the RU_nRSTIMER signal for a minimum of 250 ns.
.rconfig(<input signal to initiate configuration>)	Input	This signal is an input into the configuration section of the remote update block. When this signal goes high, it initiates a reconfiguration. This field is required.
.regout(<data output destination to core>)	Output	This is a 1 bit output which is the output of the internal shift register updated every rising edge of .clk. The data coming out depends on the control signals. This field is required.

Related Information

- [Dual Configuration Intel FPGA IP Core References](#) on page 65
- [Remote System Upgrade](#) on page 13

- [AN 741: Remote System Upgrade for MAX 10 FPGA Devices over UART with the Nios II Processor](#)
Provides reference design for remote system upgrade in Intel MAX 10 FPGA devices.
- [I2C Remote System Update Example](#)
This example demonstrates a remote system upgrade using the I2C protocol.

3.6. Error Detection

3.6.1. Verifying Error Detection Functionality

You can inject a soft error by changing the 32-bit CRC storage register in the CRC circuitry. After verifying the failure induced, you can restore the 32-bit CRC value to the correct CRC value using the same instruction and inserting the correct value. Be sure to read out the correct value before updating it with a known bad value.

In user mode, Intel MAX 10 devices support the `CHANGE_EDREG` JTAG instruction, which allows you to write to the 32-bit storage register. You can use **.jam** to automate the testing and verification process. You can only execute this instruction when the device is in user mode. This instruction enables you to dynamically verify the CRC functionality in-system without having to reconfigure the device. You can then switch to use the CRC circuit to check for real errors induced by an SEU.

After the test completes, you can clear the CRC error and restore the original CRC value using one of the following methods:

- Bring the TAP controller to the RESET state by holding TMS high for five TCK clocks
- Power cycle the device
- Perform these steps:
 1. After the configuration completes, use `CHANGE_EDREG` JTAG instruction to shift out the correct precomputed CRC value and load the wrong CRC value to the CRC storage register. When an error is detected, the `CRC_ERROR` pin is asserted.
 2. Use `CHANGE_EDREG` JTAG instruction to shift in the correct precomputed CRC value. The `CRC_ERROR` pin is de-asserted to show that the error detection CRC circuitry is working.

Example 2. JAM File

```
'EDCRC_ERROR_INJECT

ACTION ERROR_INJECT = EXECUTE;
DATA DEVICE_DATA;
BOOLEAN out[32];
BOOLEAN in[32] = $02040608;    'shift in any wrong CRC value
ENDDATA;
PROCEDURE EXECUTE USES DEVICE_DATA;
BOOLEAN X = 0;
DRSTOP IDLE;
IRSTOP IDLE;
STATE IDLE;
IRSCAN 10, $015;                'shift in CHANGE_EDREG instruction
WAIT IDLE, 10 CYCLES, 1 USEC, IDLE;
DRSCAN 32, in[31..0], CAPTURE out[31..0];
WAIT IDLE, 10 CYCLES, 50 USEC, IDLE;
PRINT " ";
PRINT "Data read out from the Storage Register: "out[31], out[30], out[29],
```

```
out[28], out[27],  
out[26], out[25], out[24], out[23], out[22], out[21], out[20], out[19],  
out[18], out[17], out[16], out[15], out[14], out[13], out[12], out[11],  
out[10], out[9], out[8], out[7], out[6], out[5], out[4], out[3],  
out[2], out[1], out[0];          'Read out correct precomputed CRC value  
PRINT " ";  
STATE IDLE;  
EXIT 0;  
ENDPROC;
```

You can run the .jam file using quartus_jli executable with the following command line:

```
quartus_jli -c<cable index> -a<action name> <filename>.jam
```

Related Information

- [SEU Mitigation and Configuration Error Detection](#) on page 26
- [AN 425: Using the Command-Line Jam STAPL Solution for Device Programming](#)
Provides more information about quartus_jli command line executable.

3.6.2. Enabling Error Detection

The CRC error detection feature in the Intel Quartus Prime software generates the CRC_ERROR output to the optional dual-purpose CRC_ERROR pin.

To enable the error detection feature using CRC, follow these steps:

1. Open the Intel Quartus Prime software and load a project using Intel MAX 10 device family.
2. On the **Assignments** menu, click **Device**. The **Device** dialog box appears.
3. In the **Device** dialog box, click **Device and Pin Options**. The **Device and Pin Options** dialog box appears.
4. Click **Device and Pin Option**. The **Device and Pin Option** dialog box appears.
5. In the **Device and Pin Option** dialog box, select **Error Detection CRC** from the category pane.
6. Turn on **Enable Error Detection CRC_ERROR pin**.
7. In the **Divide error check frequency by** field, enter a valid divisor.
The divisor value divides down the frequency of the configuration oscillator output clock. This output clock is used as the clock source for the error detection process.
8. Click **OK**.

Related Information

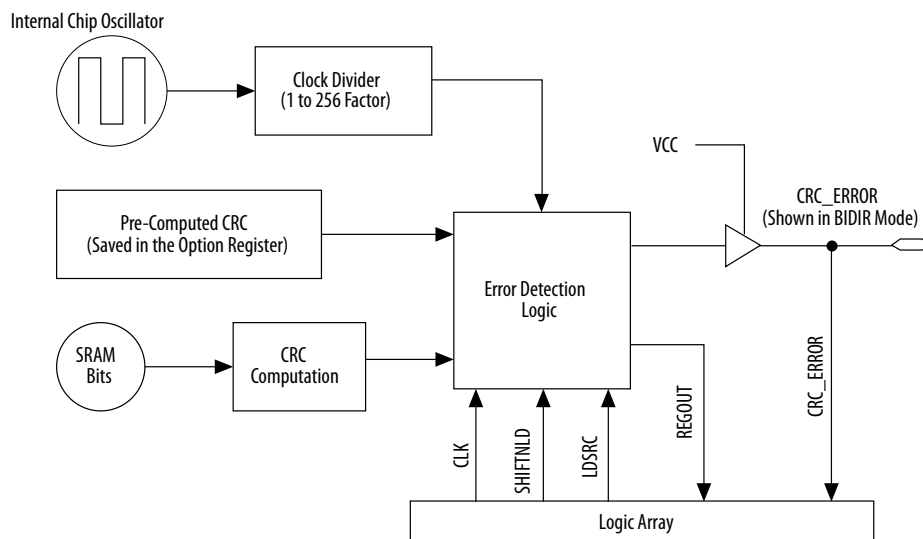
[SEU Mitigation and Configuration Error Detection](#) on page 26

3.6.3. Accessing Error Detection Block Through User Logic

The error detection circuit stores the computed 32-bit CRC signature in a 32-bit register. The user logic from the core reads out this signature. The fiftyfivenm_crcblock primitive is a WYSIWYG component used to establish the interface from the user logic to the error detection circuit. The fiftyfivenm_crcblock primitive atom contains the input and output ports that

must be included in the atom. To access the logic array, you must insert the `fiftyfivenm_crcblock` WYSIWYG atom into your design. The recommended clock frequency of `.clk` port is to follow the clock frequency of EDCRC block.

Figure 15. Error Detection Block Diagram with Interfaces for Intel MAX 10 Devices



The following example shows how the input and output ports of a WYSIWYG atom are defined in the Intel MAX 10 device.

```
fiftyfivenm_crcblock <name>
(
  .clk(<ED_CLK clock source>),
  .shiftnld(<ED_SHIFTNLD source>),
  .ldsrc (<LDSRC source>),
  .crcerror(<CRCERROR_CORE out destination>),
  .regout(<output destination>)
);
defparam <crblock_name>.oscillator_divider = <internal oscillator division (1
to 256)>;
```

Table 31. Port Definitions

Port	Input/ Output	Definition
<crblock_name>	—	Unique identifier for the CRC block and represents any identifier name that is legal for the given description language such as Verilog HDL, VHDL, AHDL. This field is required.
.clk(<clock source>	Input	This signal designates the clock input of this cell. All operations of this cell are with respect to the rising edge of the clock. Whether it is the loading of the data into the cell or data out of the cell, it always occurs on the rising edge. This port is required.
.shiftnld (<shiftnld source>)	Input	This signal is an input into the error detection block. If shiftnld=1, the data is shifted from the internal shift register to the regout at each rising edge of clk. If shiftnld=0, the shift register parallel loads either the pre-calculated CRC value or the update register contents depending on the ldsrc port input. This port is required.

continued...

Port	Input/ Output	Definition
.ldsrc (<ldsrc source>)	Input	This signal is an input into the error detection block. If <code>ldsrc=0</code> , the pre-computed CRC register is selected for loading into the 32-bit shift register at the rising edge of <code>clk</code> when <code>shiftnld=0</code> . If <code>ldsrc=1</code> , the signature register (result of the CRC calculation) is selected for loading into the shift register at the rising edge of <code>clk</code> when <code>shiftnld=0</code> . This port is ignored when <code>shiftnld=1</code> . This port is required.
.crcerror (<crcerror out destination>)	Output	This signal is the output of the cell that is synchronized to the internal oscillator of the device (100-MHz or 80-MHz internal oscillator) and not to the <code>clk</code> port. It asserts automatically high if the error block detects that a SRAM bit has flipped and the internal CRC computation has shown a difference with respect to the pre-computed value. This signal must be connected either to an output pin or a bidirectional pin. If it is connected to an output pin, you can only monitor the <code>CRC_ERROR</code> pin (the core cannot access this output). If the <code>CRC_ERROR</code> signal is used by core logic to read error detection logic, this signal must be connected to a <code>BIDIR</code> pin. The signal is fed to the core indirectly by feeding a <code>BIDIR</code> pin that has its <code>oe</code> port connected to <code>V_{CC}</code> .
.regout (<output destination>)	Output	This signal is the output of the error detection shift register synchronized to the <code>clk</code> port, to be read by core logic. It shifts one bit at each cycle. User should clock the <code>clk</code> signal 31 cycles to read out the 32 bits of the shift register. The values at the <code>.regout</code> port are an inversion of the actual values.

Related Information

- [SEU Mitigation and Configuration Error Detection](#) on page 26
- [Error Detection Timing](#) on page 27

3.7. Enabling Data Compression

When you enable compression, the Intel Quartus Prime software generates configuration files with compressed configuration data.

A compressed configuration file is needed to use the dual configuration mode in the internal configuration scheme. This compressed file reduces the storage requirements in internal flash memory, and decreases the time needed to send the bitstream to the Intel MAX 10 device family. There are two methods to enable compression for the Intel MAX 10 device family bitstreams in the Intel Quartus Prime software:

- Before design compilation—using the **Compiler Settings** menu.
- After design compilation—using the **Convert Programming Files** option.

3.7.1. Enabling Compression Before Design Compilation

To enable compression before design compilation, follow these steps:

1. On the **Assignments** menu, click **Device**. The **Device** dialog box appears.
2. Click **Device and Pin Options**. The **Device and Pin Options** dialog box appears.
3. In the **Device and Pin Option** dialog box, select **Configuration** from the category pane.
4. Turn on **Generate compressed bitstreams**.
5. Click **OK**.
6. In the **Settings** dialog box, click **OK**.

Related Information

[Configuration Data Compression](#) on page 29

3.7.2. Enabling Compression After Design Compilation

To enable compression after design compilation, follow these steps:

1. On the **File** menu, click **Convert Programming Files**.
2. Under **Output programming file**, from the Programming file type pull-down menu, select your desired file type.
3. If you select the Programmer Object File (**.pof**), you must specify a configuration device, directly under the file type.
4. In the **Input files to convert** box, select **SOF Data**.
5. Click **Add File** to browse to the Intel MAX 10 device family **.sof**.
6. In the **Convert Programming Files** dialog box, select the **.pof** you added to **SOF Data** and click **Properties**.
7. In the **SOF Properties** dialog box, turn on the **Compression** option.

Related Information

[Configuration Data Compression](#) on page 29

3.8. AES Encryption

Security Notice:

The Intel MAX 10 devices use 128-bit encryption keys. For higher levels of security, Intel recommends that you select devices using longer key lengths such as Intel Stratix 10 or Intel Agilex 7 devices, which use 256-bit encryption keys.

This section covers detailed guidelines on applying AES Encryption for design security.

There are two main steps in applying design security in Intel MAX 10 devices. First is to generate the encryption key programming (**.ekp**) file and second is to program the **.ekp** file into the device.

The **.ekp** file has other different formats, depending on the hardware and system used for programming. There are three file formats supported by the Intel Quartus Prime software:

- JAM Byte Code (**.jbc**) file
- JAM™ Standard Test and Programming Language (STAPL) Format (**.jam**) file
- Serial Vector Format (**.svf**) file

Only the **.ekp** file type generated automatically from the Intel Quartus Prime software. You must create the **.jbc**, **.jam** and **.svf** files using the Intel Quartus Prime software if these files are required in the key programming.

Note:

Intel recommends that you keep the **.ekp** file confidential.

3.8.1. Generating .ekp File and Encrypt Configuration File

To generate the **.ekp** file and encrypt your configuration file, follow these steps:

1. On the **File** menu, click **Convert Programming Files**.
2. Under **Output programming file**, select **Programmer Object File (.pof)** in the **Programming file type** list.
3. In the **Mode** list, select **Internal Configuration**.
4. Click **Option/Boot Info** and the **ICB setting** dialog box will appear.
5. You can enable the **Allow encrypted POF** only option. Click **OK** once ICB setting is set.

The device will only accept encrypted bitstream during internal configuration if this option is enabled. If you encrypt one of CFM0, CFM1 or CFM2 only, the Programmer will post a warning.

6. Type the file name in the **File name** field, or **browse** to and select the file.
7. Under the **Input files to convert** section, click **SOF Data**.
8. Click **Add File** to open the **Select Input File** dialog box.
9. Browse to the unencrypted **.sof** and click **Open**.
10. Under the **Input files to convert** section, click on the added **.sof**.
11. Click **Properties** and the **SOF Files Properties: Bitstream Encryption** dialog box will appear.
12. Turn on **Generate encrypted bitstream**.
13. Turn on **Generate key programming file** and type the **.ekp** file path and file name in the text area, or browse to and select **<filename>.ekp**.
14. You can the key with either a **.key** file or entering the key manually.

Note: Intel MAX 10 devices require the entry of 128-bit keys.

- Adding key with a **.key** file.

The **.key** file is a plain text file in which each line represents a key unless the line starts with "#". The "#" symbol is used to denote comments. Each valid key line has the following format:

```
<key identity><white space><128-bit hexadecimal key>
# This is an example key file
key1 0123456789ABCDEF0123456789ABCDEF
```

- a. Enable the **Use key file** checkbox.
 - b. Click **Open** and add the desired **.key** file and click **Open** again.
 - c. Under **Key entry** part, the key contained in the **.key** file will be selected in the drop-down list.
 - d. Click **OK**.
- Entering your key manually.

- a. Under **Key entry** part, click the **Add** button.
 - b. Select the **Key Entry Method** to enter the encryption key either with the **On-screen Keypad** or **Keyboard**.
 - c. Enter a key name in the **Key Name (alphanumeric)** field.
 - d. Key in the desired key in the **Key (128-bit hexadecimal)** field and repeat in the **Confirm Key** field below it.
 - e. Click **OK**.
15. Read the design security feature disclaimer. If you agree, turn on the **acknowledgment** box and click **OK**.
16. In the **Convert Programming Files** dialog box, click **OK**. The <filename>.ekp and encrypted configuration file will be generated in the same project directory.
- Note:* For dual configuration .pof file, both .sof file need to be encrypted with the same key. The generation of key file and encrypted configuration file will not be successful if different keys are used.

3.8.2. Generating .jam/.jbc/.svf file from .ekp file

To generate .jam/.jbc/.svf file from .ekp file, follow these steps:

1. On the **Tools** menu, click **Programmer** and the **Programmer** dialog box will appear.
2. In the **Mode** list, select **JTAG** as the programming mode.
3. Click **Hardware Setup**. The **Hardware Setup** dialog box will appear.
4. Select **USBBlaster** as the programming hardware in the **currently selected hardware list** and click **Done**.
5. Click **Add File** and the **Select Programmer File** dialog box will appear.
6. Type <filename>.ekp in the **File name** field and click **Open**.
7. Select the .ekp file you added and click **Program/Configure**.
8. On the **File** menu, point to **Create/Update** and click **Create JAM, SVF, or ISC File**. The **Create JAM, SVF, or ISC File** dialog box will appear.
9. Select the file format required for the .ekp file in the **File format** field.
 - JEDEC STAPL Format (.jam)
 - Jam STAPL Byte Code (.jbc)
 - Serial Vector Format (.svf)
10. Type the file name in the **File name** field, or browse to and select the file.
11. Click **OK** to generate the .jam, .jbc or .svf file.

3.8.3. Programming .ekp File and Encrypted POF File

There are two methods to program the encrypted .pof and .ekp files:

- Program the .ekp and .pof separately.

Note: You only can program the .ekp and .pof separately when **Allow encrypted POF only** option is disabled.

- Integrate the .ekp into .pof and program both altogether.

3.8.3.1. Programming .ekp File and Encrypted .pof Separately

To program the .ekp and encrypted .pof separately using the Intel Quartus Prime software, follow these steps:

1. In the Intel Quartus Prime Programmer, under the **Mode** list, select **JTAG** as the programming mode.
2. Click **Hardware Setup** and the **Hardware Setup** dialog box will appear.
3. Select **USBBlaster** as the programming hardware in the **Currently selected hardware** list and click **Done**.
4. Click **Add File** and the **Select Programmer File** dialog box will appear.
5. Type <filename>.ekp in the **File name** field and click **Open**.
6. Select the .ekp file you added and click **Program/Configure**.
7. Click **Start** to program the key.

Note: The Intel Quartus Prime software message window provides information about the success or failure of the key programming operation. Once the .ekp is programmed, .pof can be programmed separately. To retain the security key in the internal flash that had been programmed through the .ekp, continue with the following steps.

8. Select the .pof to be programmed to the selected device.
9. Check only the functional block that need to be updated at child level for CFM and UFM. Do not check operation at the parent level when using Programmer GUI.
10. After all settings are set, click **Start** to start programming.

3.8.3.2. Integrate the .ekp into .pof Programming

To integrate the .ekp into .pof and program both altogether using the Intel Quartus Prime software, follow these steps:

1. In the Intel Quartus Prime Programmer, under the **Mode** list, select **JTAG** as the programming mode.
2. Click **Hardware Setup** and the **Hardware Setup** dialog box will appear.
3. Select **USBBlaster** as the programming hardware in the **Currently selected hardware** list and click **Done**.
4. Click the **Auto Detect** button on the left pane.
5. Select the .pof you want to program to the device.
6. Select the <yourpoffile.pof>, right click and select **Add EKP File** to integrate .ekp file with the .pof file.

Once the .ekp is integrated into the .pof, you can to save the integrated .pof into a new .pof. This newly saved file will have original .pof integrated with .ekp information.

7. Select the <yourpoffile.pof> in the **Program/Configure** column.
8. After all settings are set, click **Start** to start programming

3.8.4. Encryption in Internal Configuration

During internal configuration, the FPGA decrypts the .pof with the stored key and uses the decrypted data to configure itself. The configuration image loaded during configuration is also affected by the encryption settings and the **Configure device from CFM0 only** setting.

Table 32. Configuration Image Outcome Based on Encryption Settings, Encryption Key and CONFIG_SEL Pin Settings

Table shows the scenario when you disable the **Configure device from CFM0 only**. Key X and Key Y are security keys included in your device and configuration image.

Configuration Image Mode	CFM0 (image 0) Encryption Key	CFM1 (image 1) Encryption Key	Key Stored in the Device	Allow Encrypted POF Only	CONFIG_SEL pin	Design Loaded After Power-up
Single	Not Encrypted	Not Available	No key	Disabled	0	image 0
Single	Not Encrypted	Not Available	No key	Disabled	1	image 0
Single	Not Encrypted	Not Available	Key X	Disabled	0	image 0
Single	Not Encrypted	Not Available	Key X	Disabled	1	image 0
Single	Not Encrypted	Not Available	Key X	Enabled	0	Configuration Fail
Single	Not Encrypted	Not Available	Key X	Enabled	1	Configuration Fail
Single	Key X	Not Available	No key	Enabled	0	Configuration Fail
Single	Key X	Not Available	No key	Enabled	1	Configuration Fail
Single	Key X	Not Available	Key X	Enabled	0	image 0
Single	Key X	Not Available	Key X	Enabled	1	image 0
Single	Key X	Not Available	Key Y	Enabled	0	Configuration Fail
Single	Key X	Not Available	Key Y	Enabled	1	Configuration Fail
Dual	Not Encrypted	Not Encrypted	No key	Disabled	0	image 0
Dual	Not Encrypted	Not Encrypted	No key	Disabled	1	image 1
Dual	Key X	Not Encrypted	No key	Disabled	0	image 1 ⁽¹⁵⁾
Dual	Key X	Not Encrypted	No key	Disabled	1	image 1
Dual	Key X	Not Encrypted	Key X	Disabled	0	image 0
Dual	Key X	Not Encrypted	Key X	Disabled	1	image 1
Dual	Key X	Not Encrypted	Key X	Enabled	0	image 0
Dual	Key X	Not Encrypted	Key X	Enabled	1	image 0
Dual	Key X	Not Encrypted	Key Y	Enabled	0	Configuration Fail
Dual	Key X	Not Encrypted	Key Y	Enabled	1	Configuration Fail
Dual	Key X	Key X	No key	Enabled	0	Configuration Fail
Dual	Key X	Key X	No key	Enabled	1	Configuration Fail
Dual	Key X	Key X	Key X	Enabled	0	image 0

continued...

⁽¹⁵⁾ After image 0 configuration failed, device will automatically load image 1.

Configuration Image Mode	CFM0 (image 0) Encryption Key	CFM1 (image 1) Encryption Key	Key Stored in the Device	Allow Encrypted POF Only	CONFIG_SEL pin	Design Loaded After Power-up
Dual	Key X	Key X	Key X	Enabled	1	image 1
Dual	Key X	Key Y	Key X	Enabled	0	image 0
Dual	Key X	Key Y	Key X	Enabled	1	image 0 ⁽¹⁶⁾
Dual	Key Y	Key Y	Key Y	Enabled	0	image 0
Dual	Key Y	Key Y	Key Y	Enabled	1	image 1
Dual	Key X	Key Y	Key Y	Enabled	0	image 1 ⁽¹⁵⁾
Dual	Key X	Key Y	Key Y	Enabled	1	image 1

Table 33. Configuration Image Outcome Based on Encryption Settings and Encryption Key

Table shows the scenario when you enable the **Configure device from CFM0 only**.

CFM0 (image 0) Encryption Key	Key Stored in the Device	Allow Encrypted POF Only	Design Loaded After Power-up
Not Encrypted	No key	Disabled	image 0
Not Encrypted	Key X	Disabled	image 0
Not Encrypted	Key Y	Disabled	image 0
Not Encrypted	No key	Enabled	Configuration Fail
Not Encrypted	Key X	Enabled	Configuration Fail
Not Encrypted	Key Y	Enabled	Configuration Fail
Key X	No key	Disabled	Configuration Fail
Key X	Key X	Disabled	image 0
Key X	Key Y	Disabled	Configuration Fail
Key X	No key	Enabled	Configuration Fail
Key X	Key X	Enabled	image 0
Key X	Key Y	Enabled	Configuration Fail
Key Y	No key	Disabled	Configuration Fail
Key Y	Key X	Disabled	Configuration Fail
Key Y	Key Y	Disabled	image 0
Key Y	No key	Enabled	Configuration Fail
Key Y	Key X	Enabled	Configuration Fail
Key Y	Key Y	Enabled	image 0

Related Information

Generating .pof using Convert Programming Files on page 41

⁽¹⁶⁾ After image 1 configuration failed, device will automatically load image 0.

3.9. Intel MAX 10 JTAG Secure Design Example

This design example demonstrates the instantiation of the JTAG WYSIWYG atom and the example of user logic implementation in the Intel Quartus Prime software to execute the LOCK and UNLOCK JTAG instructions. This design example is targeted for Intel MAX 10 devices with the JTAG Secure Mode enabled.

Related Information

- [JTAG Instruction Availability](#) on page 23
- [Configuration Flash Memory Permissions](#) on page 23
- [JTAG Secure Design Example](#)

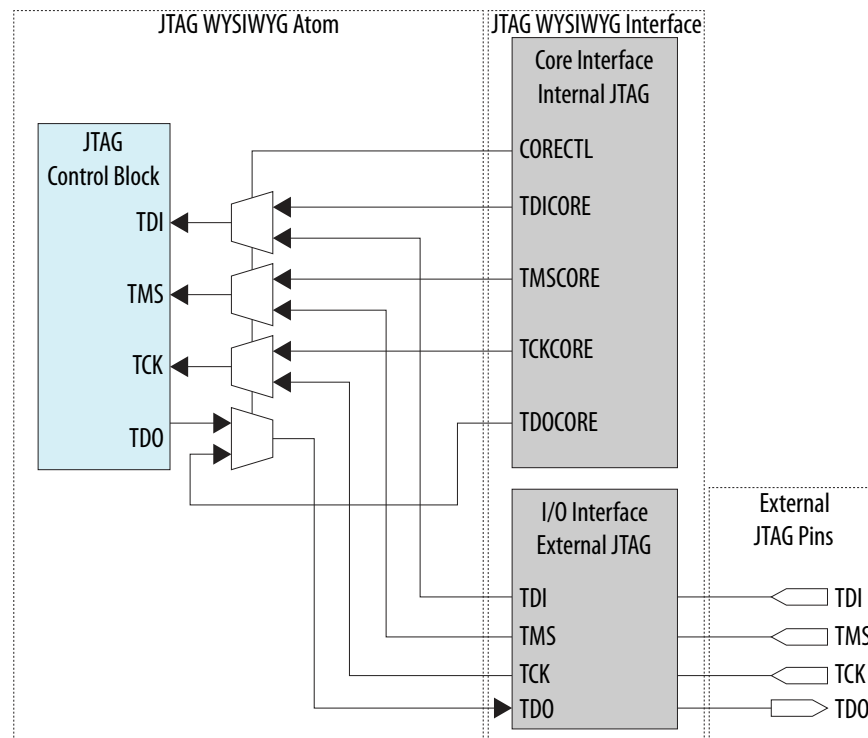
3.9.1. Internal and External JTAG Interfaces

There are two interfaces to access the JTAG control block in Intel MAX 10 devices:

- External JTAG interface—connection of the JTAG control block from the physical JTAG pins; TCK, TDI, TDO, and TMS.
- Internal JTAG interface—connection of the JTAG control block from the internal FPGA core fabric.

You can only access the JTAG control block using either external or internal JTAG interface one at a time. External JTAG interfaces are commonly used for JTAG configuration using programming cable. To access the internal JTAG interface, you must include the JTAG WYSIWYG atom in your Intel Quartus Prime software design.

Figure 16. Internal and External JTAG Interface Connections



Note: To ensure the internal JTAG interfaces of Intel MAX 10 devices function correctly, all four JTAG signals (TCK, TDI, TMS and TDO) in the JTAG WYSIWYG atom need to be routed out. The Intel Quartus Prime software will automatically assign the ports to their corresponding dedicated JTAG pins.

3.9.2. JTAG WYSIWYG Atom for JTAG Control Block Access Using Internal JTAG Interface

The following example shows how the input and output ports of a JTAG WYSIWYG atom are defined in the Intel MAX 10 device.

```
fiftyfivenm_jtag <name>
(
    .tms(),
    .tck(),
    .tdi(),
    .tdoutap(),
    .tdouser(),
    .tdicore(),
    .tmscore(),
    .tckcore(),
    .corectl(),
    .tdo(),
    .tmsutap(),
    .tckutap(),
    .tdiutap(),
    .shiftuser(),
    .clkdruser(),
    .updateuser(),
    .runidleuser(),
    .usruser(),
    .tdocore(),
    .ntdopinena()
);
```

Table 34. Port Description

Ports	Input/Output	Functions
<name>	—	Identifier for the Intel MAX 10 JTAG WYSIWYG atom and represents any identifier name that is legal for the given description language, such as Verilog HDL, VHDL, and AHDL.
.corectl()	Input	Active high input to the JTAG control block to enable the internal JTAG access from core interface. When the FPGA enters user mode after configuration, this port is low by default. Pulling this port to logic high will enable the internal JTAG interface (with external JTAG interface disabled at the same time) and pulling this port to logic low will disable the internal JTAG interface (with external JTAG interface enabled at the same time).
.tckcore()	Input	Core tck signal
.tdicore()	Input	Core tdi signal
.tmscore()	Input	Core tms signal
.tdocore()	Output	Core tdo signal
.tck()	Input	Pin tck signal
.tdi()	Input	Pin tdi signal
.tms()	Input	Pin tms signal
.tdo()	Output	Pin tdo signal
continued...		

Ports	Input/Output	Functions
.clkdruser()	Input/Output	These ports are not used for enabling the JTAG Secure mode using the internal JTAG interface, you can leave them unconnected.
.runidleuser()		
.shiftuser()		
.tckutap()		
.tdiutap()		
.tdouser()		
.tdoutap()		
.tmsutap()		
.updateuser()		
.usr1user()		
.ntdopinena()		

3.9.3. Executing LOCK and UNLOCK JTAG Instructions

When you configure this reference design into a Intel MAX 10 device with the JTAG Secure mode enabled, the device is in JTAG Secure mode after power-up and configuration.

To disable the JTAG Secure mode, trigger the `start_unlock` port of the user logic to issue the UNLOCK JTAG instruction. After the UNLOCK JTAG instruction is issued, the device exits from JTAG secure mode. When the JTAG Secure mode is disabled, you can choose to full-chip erase the internal flash of Intel MAX 10 device to disable the JTAG Secure mode permanently.

The `start_lock` port in the user logic triggers the execution of the LOCK JTAG instruction. Executing this instruction enables the JTAG Secure mode of the Intel MAX 10 device.

Figure 17. LOCK or UNLOCK JTAG Instruction Execution

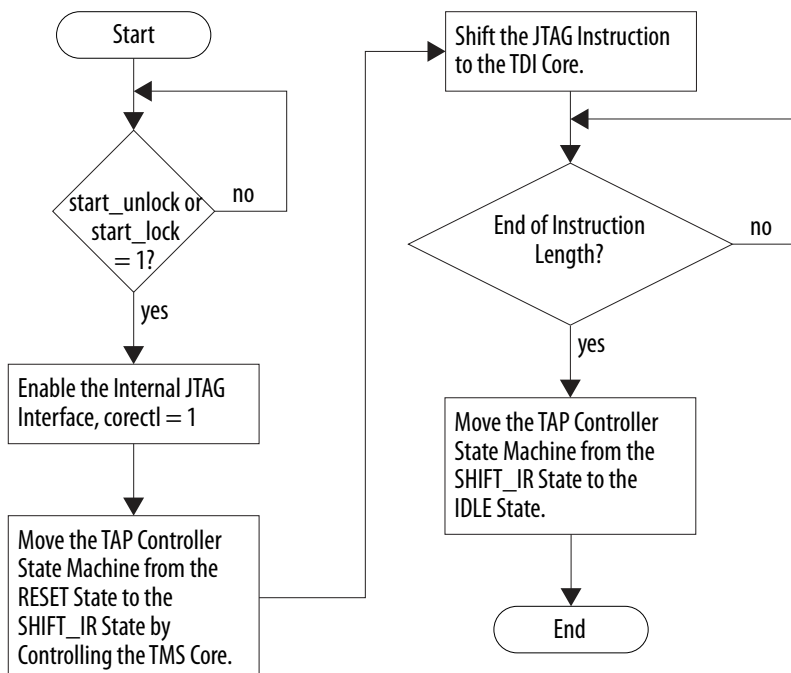


Table 35. Input and Output Port of the User Logic

Port	Input/ Output	Function
clk_in	Input	Clock source for the user logic. The f_{MAX} of the user logic depends on the timing closure analysis. You need to apply timing constraint and perform timing analysis on the path to determine the f_{MAX} .
start_lock	Input	Triggers the execution of the LOCK JTAG instruction to the internal JTAG interface. Pulse signal high for at least 1 clock cycle to trigger.
start_unlock	Input	Triggers the execution of the UNLOCK JTAG instruction to the internal JTAG interface. Pulse signal high for at least 1 clock cycle to trigger.
jtag_core_en_out	Output	Output to the JTAG WYSIWYG atom. This port is connected to the <code>corectl</code> port of the JTAG WYSIWYG atom to enable the internal JTAG interface.
tck_out	Output	Output to the JTAG WYSIWYG atom. This port is connected to the <code>tck_core</code> port of the JTAG WYSIWYG atom.
tdi_out	Output	Output to the JTAG WYSIWYG atom. This port is connected to the <code>tdi_core</code> port of the JTAG WYSIWYG atom.
tms_out	Output	Output to the JTAG WYSIWYG atom. This port is connected to the <code>tms_core</code> port of the JTAG WYSIWYG atom.
indicator	Output	Logic high of this output pin indicates the completion of the LOCK or UNLOCK JTAG instruction execution.

3.9.4. Verifying the JTAG Secure Mode

You can verify whether your device has successfully entered or exited JTAG secure mode by executing a non-mandatory JTAG instruction.

Note: You must instantiate the internal JTAG interface for you unlock the external JTAG when the device is in JTAG Secure mode.

When you enable the JTAG Secure option, the Intel MAX 10 device will be in the JTAG Secure mode after power-up. To validate the JTAG Secure feature in your design example, perform these steps:

1. Configure the reference design .pof file into the device with JTAG Secure mode enabled. After power cycle, the device should be in JTAG Secure mode.
2. You can ensure that the device enters user mode successfully by observing one of the following:
 - CONF_DONE pin goes high
 - counter_output pin starts toggling
3. Issue the PULSE_NCONFIG JTAG instruction using the external JTAG pins to reconfigure the device. You can use the pulse_ncfg.jam file attached in the design example. To execute the pulse_ncfg.jam file, you can use the quartus_jli or the JAM player. You can ensure that the device does not reconfigure by observing one of the following:

- CONF_DONE pin stays high
- counter_output pin continues toggling

Unsuccessful reconfiguration verifies that the device is currently in JTAG Secure mode.

4. Pull the start_unlock port of the user logic to logic high to execute the UNLOCK JTAG instruction.

The indicator port goes high after the UNLOCK JTAG instruction is complete.

5. Issue the PULSE_NCONFIG JTAG instruction using the external JTAG pins to reconfigure the device. You can ensure that the device reconfigures successfully by observing one of the following:

- CONF_DONE pin is low
- counter_output pin stops toggling

Successful reconfiguration verifies that the device is currently not in JTAG Secure mode.

4. Intel MAX 10 FPGA Configuration IP Core Implementation Guides

4.1. Unique Chip ID Intel FPGA IP Core

This section provides the guideline to implement the Unique Chip ID Intel FPGA IP core.

Related Information

- [Unique Chip ID](#) on page 21
- [Unique Chip ID Intel FPGA IP Core Ports](#) on page 68

4.1.1. Instantiating the Unique Chip ID Intel FPGA IP Core

To instantiate the Unique Chip ID Intel FPGA IP core, follow these steps:

1. On the Tools menu of the Intel Quartus Prime software, click **IP Catalog**.
2. Under the Library category, expand the Basic Functions and Configuration Programming.
3. Select **Unique Chip Intel FPGA IP** and click **Add**, and enter your desired output file name
4. In the Save IP Variation dialog box:
 - Set your IP variation filename and directory.
 - Select IP variation file type.
5. Click **Finish**.

4.1.2. Resetting the Unique Chip ID Intel FPGA IP Core

To reset the Unique Chip ID Intel FPGA IP core, you must assert high to the `reset` signal for at least one clock cycle. After you de-assert the `reset` signal, the Unique Chip ID Intel FPGA IP core re-reads the unique chip ID of your device from the fuse ID block. The Unique Chip ID Intel FPGA IP core asserts the `data_valid` signal after completing the operation.

4.2. Dual Configuration Intel FPGA IP Core

This section provides the guideline to implement the Dual Configuration Intel FPGA IP core.

4.2.1. Instantiating the Dual Configuration Intel FPGA IP Core

To instantiate the Dual Configuration Intel FPGA IP Core, follow these steps:

1. On the Tools menu of the Intel Quartus Prime software, click **IP Catalog**.
2. Under the Library category, expand the Basic Functions and Configuration Programming.
3. Select **Dual Configuration Intel FPGA IP** and after clicking **Add**, the IP Parameter Editor appears.
4. In the New IP Instance dialog box:
 - Set the top-level name of your IP.
 - Select the Device family.
 - Select the Device
5. Click **OK**.

5. Dual Configuration Intel FPGA IP Core References

Note: The Dual Configuration feature is not supported in Intel MAX 10 devices with Compact feature option.

Related Information

- [Dual Configuration Intel FPGA IP Core](#) on page 19
- [Accessing Remote System Upgrade through User Logic](#) on page 46
- [AN 741: Remote System Upgrade for MAX 10 FPGA Devices over UART with the Nios II Processor](#)
Provides reference design for remote system upgrade in Intel MAX 10 FPGA devices.
- [I2C Remote System Update Example](#)
This example demonstrates a remote system upgrade using the I2C protocol.

5.1. Dual Configuration Intel FPGA IP Core Avalon Memory-Mapped Address Map

Table 36. Dual Configuration Intel FPGA IP Core Avalon Memory-Mapped Address Map for Intel MAX 10 Devices

- Intel recommends you to set the reserve bits to 0 for write operations. For read operations, the IP core will always generate 0 as the output.
- Write 1 to trigger any operation stated in the description.
- You need to trigger the desired operation from offset 2 before any read operation of offset 4, 5, 6 and 7.

Offset	R/W	Width (Bits)	Description
0	W	32	<ul style="list-style-type: none"> • Bit 0—Trigger reconfiguration. • Bit 1—Reset the watchdog timer. • Bit 31:2—Reserved. <p>Signals are triggered at the same write cycle on Avalon.</p>
1	W	32	<ul style="list-style-type: none"> • Bit 0—Trigger <code>config_sel_overwrite</code> value to the input register. <ul style="list-style-type: none"> — 0: Disable overwrite <code>config_sel</code> pin. — 1: Enable overwrite <code>config_sel</code> pin. • Bit 1—Writes <code>config_sel</code> value to the input register. Set 0 or 1 to load from configuration image 0 or 1 respectively <ul style="list-style-type: none"> — 0: Load configuration image 0. — 1: Load configuration image 1. • Bit 31:2—Reserved. <p>The busy signal is generated right after the write cycle, while the configuration image information is registered. Once the busy signal is high, writing to this address is ignored until the process is completed and the busy signal is de-asserted.</p>

continued...

Offset	R/W	Width (Bits)	Description
2	W	32	<ul style="list-style-type: none"> Bit 0—Trigger read operation from the user watchdog. Bit 1—Trigger read operation from the previous state application 1 register. Bit 2—Trigger read operation from the previous state application 2 register. Bit 3—Trigger read operation from the input register. Bit 31:4—Reserved. <p>The busy signal is generated right after the write cycle. These bits are not one-hot. Multiple bits can be set to 1 at the same time to trigger the read operation from multiple registers.</p>
3	R	32	<ul style="list-style-type: none"> Bit 0—IP busy signal. Bit 31:1—Reserved. <p>The busy signal indicates that the Dual Configuration Intel FPGA IP core is in the writing or reading process. In this state, all write operation requests to the remote system upgrade block registers are ignored except for triggering the reset timer. Intel recommends you to poll this busy signal once you trigger any read or write process. The busy signal will not stay high for more than 531 clock cycles in each single operation triggered.</p>
4	R	32	<ul style="list-style-type: none"> Bit 11:0—User watchdog value. ⁽¹⁷⁾ Bit 12—The current state of the user watchdog enable. Bit 16:13—The msm_cs value of the current state. Bit 31:17—Reserved
5	R	32	<ul style="list-style-type: none"> Bit 3:0—Previous state application 1 reconfiguration source value from the Remote System Upgrade Status Register-Previous State Bit for Intel MAX 10 Devices table. Bit 7:4—The msm_cs value of the previous state application 1. Bit 31:8—Reserved.
6	R	32	<ul style="list-style-type: none"> Bit 3:0—Previous state application 2 reconfiguration source value from the Remote System Upgrade Status Register-Previous State Bit for Intel MAX 10 Devices table. Bit 7:4—The msm_cs value of the previous state application 2. Bit 31:8—Reserved.
7	R	32	<ul style="list-style-type: none"> Bit 0—config_sel_overwrite value from the input register. Bit 1—config_sel value of the input register.⁽¹⁸⁾ Bit 31:2—Reserved.

Related Information

- [Dual Configuration Intel FPGA IP Core](#) on page 19
- [Avalon Interface Specifications](#)
Provides more information about the Avalon memory-mapped interface specifications applied in Dual Configuration Intel FPGA IP core.
- [Instantiating the Dual Configuration Intel FPGA IP Core](#) on page 64
- [Remote System Upgrade Status Registers](#) on page 18
The Remote System Upgrade Status Register—Previous state bit for Intel MAX 10 Devices table provides more information about previous state applications reconfiguration sources.

⁽¹⁷⁾ You can only read the 12 most significant bit of the 29 bit user watchdog value using Dual Configuration IP Core.

⁽¹⁸⁾ Reads the config_sel of the input register only. It will not reflect the physical CONFIG_SEL pin setting.

5.2. Dual Configuration Intel FPGA IP Core Parameters

Table 37. Dual Configuration Intel FPGA IP Core Parameter for Intel MAX 10

Parameter	Value	Description
Clock frequency	Up to 80 MHz	Specifies the number of cycle to assert RU_nRSTIMER and RU_nCONFIG signals. Note that maximum RU_CLK is 40 MHz, the Dual Configuration Intel FPGA IP core has restriction to run at 80 MHz maximum, which is twice faster than hardware limitation. This is because the Dual Configuration Intel FPGA IP core generates RU_CLK at half rate of the input frequency.

6. Unique Chip ID Intel FPGA IP Core References

6.1. Unique Chip ID Intel FPGA IP Core Ports

Table 38. Unique Chip ID Intel FPGA IP Core Ports

Port	Input/Output	Width (Bits)	Description
clk_in	Input	1	<ul style="list-style-type: none"> Feeds clock signal to the unique chip ID block. The maximum supported frequency is 100 MHz. When you provide a clock signal, the IP core reads the value of the unique chip ID and sends the value to the <code>chip_id</code> output port.
reset	Input	1	<ul style="list-style-type: none"> Resets the IP core when you assert the <code>reset</code> signal to high for at least one clock cycle. The <code>chip_id [63:0]</code> output port holds the value of the unique chip ID until you reconfigure the device or reset the IP core.
data_valid	Output	1	<ul style="list-style-type: none"> Indicates that the unique chip ID is ready for retrieval. If the signal is low, the IP core is in initial state or in progress to load data from a fuse ID. After the IP core asserts the signal, the data is ready for retrieval at the <code>chip_id[63..0]</code> output port.
chip_id	Output	64	<ul style="list-style-type: none"> Indicates the unique chip ID according to its respective fuse ID location. The data is only valid after the IP core asserts the <code>data_valid</code> signal. The value at power-up resets to 0.

7. Document Revision History for the Intel MAX 10 FPGA Configuration User Guide

Document Version	Changes
2023.03.27	<ul style="list-style-type: none"> Updated links. Updated "Intel Agilex™" product family name to "Intel Agilex 7".
2022.08.08	<ul style="list-style-type: none"> Updated rows for bits 13 and 12 in table: <i>Remote System Upgrade Input Register for Intel MAX 10 Devices</i>. Updated rows for offset 1, 4, 5, and 6 in table: <i>Dual Configuration Intel FPGA IP Core Avalon Memory-Mapped Address Map for Intel MAX 10 Devices</i>.
2022.05.26	<ul style="list-style-type: none"> Updated table: <i>Remote System Upgrade Input Register for Intel MAX 10 Devices</i>. Updated table: <i>Dual Configuration Intel FPGA IP Core Avalon Memory-Mapped Address Map for Intel MAX 10 Devices</i>. Updated table: <i>Port Definitions</i>. Updated <i>Internal Configuration Modes</i> to note removal of the Enable CONFIG_SEL pin option.
2022.01.10	<ul style="list-style-type: none"> Added new topic—<i>Flash Region Access Control and Immutability</i>. Updated <i>Configuration Design Security</i>: <ul style="list-style-type: none"> Added a security notice. Added information about region access control and immutability feature. Added a security notice to <i>AES Encryption</i>. Updated <i>AES Encryption Protection</i>. Updated the descriptions for Figure: <i>Connection Setup for JTAG Single-Device Configuration using Download Cable</i> and Figure: <i>Connection Setup for JTAG Multi-Device Configuration using Download Cable</i>.
2021.07.02	Updated <i>Generating .pof using Convert Programming Files</i> to include information for the User Data in Configuration Flash Memory option.
2021.06.15	<ul style="list-style-type: none"> Updated Figure: <i>Configuration Sequence for Intel MAX 10 Devices</i>. Added a note to <i>Dual Configuration Intel FPGA IP Core References</i> to clarify that the Dual Configuration feature is not supported in Intel MAX 10 devices with Compact feature option.
2020.11.05	Updated the guidelines for JTAG pins in table <i>Dual-Purpose Configuration Pin Guidelines for Intel MAX 10 Devices</i> .
2020.06.30	<ul style="list-style-type: none"> Updated Table: <i>Configuration Flash Memory Programming Time for Sectors in Intel MAX 10 Devices</i> to include a footnote for 10M02. Added specifications for 10M02SCU324 device in the following tables: <ul style="list-style-type: none"> <i>Internal Configuration Time for Intel MAX 10 Devices (Uncompressed .rbf)</i> <i>Internal Configuration Time for Intel MAX 10 Devices (Compressed .rbf)</i> Updated Table: <i>Cyclic Redundancy Check Calculation Time for Intel® MAX 10 Devices</i> to include specification for device 10M02SCU324. Updated topic <i>Generating Third-Party Programming Files using Command Line</i> with commands to generate JAM and JBC.
2019.12.23	Updated Table: <i>ICB Values and Descriptions for Intel MAX 10 Devices</i> to correct the default watchdog timer value from 0x1FFF to 0xFFFF.
continued...	

Document Version	Changes
2019.10.07	Updated the description about generating third-party programming files using command line for <i>Generating Third-Party Programming Files using Intel Quartus Prime Programmer</i> in the <i>Configuring Intel MAX 10 Devices using JTAG Configuration</i> and <i>Configuring Intel MAX 10 Devices using Internal Configuration</i> sections.
2019.06.14	<ul style="list-style-type: none"> Added guideline for JTAG pin sharing feature in Table: <i>Dual-Purpose Configuration Pin Guidelines for Intel MAX 10 Devices</i>. Renamed sections to <i>Internal and External JTAG Interfaces</i> and <i>JTAG WYSIWYG Atom for JTAG Control Block Access Using Internal JTAG Interface</i> under the section <i>Intel MAX 10 JTAG Secure Design Example</i>. Updated Figure: <i>Internal and External JTAG Interface Connections</i> to correct external JTAG pin directions, remove ports from internal JTAG block, and add labels for JTAG WYSIWYG atom, JTAG WYSIWYG interface, and external JTAG pins. Added description that offset 2 bits are not one-hot and description for offset 3 on busy signal deassertion in Table: <i>Dual Configuration Intel FPGA IP Core Avalon-MM Address Map for Intel MAX 10 Devices</i>.
2019.04.30	Updated Table: <i>Dual Configuration Intel FPGA IP Core Avalon-MM Address Map for Intel MAX 10 Devices</i> to correct the offset 2 descriptions for bits 1 and 2.
2019.01.07	<ul style="list-style-type: none"> Updated the steps in following topics: <ul style="list-style-type: none"> Enabling Dual-purpose Pin Selecting Internal Configuration Modes Auto-Generated .pof Generating .pof using Convert Programming Files Programming .pof into Internal Flash Enabling Error Detection Enabling Compression Before Design Compilation Enabling Compression After Design Compilation Updated the note in step 4b in <i>Generating .pof using Convert Programming Files</i>. Added a note in <i>Accessing Remote System Upgrade through User Logic</i>. Renamed the following IP core names as per Intel rebranding: <ul style="list-style-type: none"> "Altera Dual Configuration IP core" to "Dual Configuration Intel FPGA IP" "Altera Unique Chip ID IP core" to "Unique Chip ID Intel FPGA IP"
2018.10.29	<ul style="list-style-type: none"> Updated Table: <i>ICB Values and Descriptions for Intel MAX 10 Devices</i> to update the footnote for JTAG Secure feature. Updated the description in <i>User Watchdog Timer</i>. Updated the note for JTAG Secure option in <i>Generating .pof using Convert Programming Files</i>. Updated the description of step 5 in <i>Generating .ekp File and Encrypt Configuration File</i>. Added a note in step 4 in <i>Enabling Dual-purpose Pin</i>. Updated Figure: <i>Configuration Sequence for Intel MAX 10 Devices</i> to add a Read ICB Settings block and a note for the Read ICB Settings block. Updated Table: <i>Dual-Purpose Configuration Pin Guidelines for Intel MAX 10 Devices</i> to update the guidelines for JTAG pins. Updated Figure: <i>Connection Setup for JTAG Single-Device Configuration using Download Cable</i>.
2018.06.01	Added 1 as valid value for n in <i>Minimum and Maximum Error Detection Frequencies for Intel MAX 10 Devices</i> table.
2018.02.12	Added steps to generate third-party programming tool files (.jbc, .jam, and .svf).

Date	Version	Changes
July 2017	2017.07.20	<ul style="list-style-type: none"> Updated CFM term to configuration flash memory in <i>High-Level Overview of JTAG Configuration and Internal Configuration for MAX 10 Devices</i> figure. Added BST definition that is boundary-scan test.
June 2017	2017.06.15	Updated methods to clear the CRC error and restore the original CRC value in <i>Verifying Error Detection Functionality</i> .
continued...		

Date	Version	Changes
April 2017	2017.04.06	Updated <i>Auto-reconfigure from secondary image when initial image fails (enabled by default)</i> option to <i>Configure device from CFM0 only</i> reflecting user interface update.
February 2017	2017.02.21	Rebranded as Intel.
October 2016	2016.10.31	<ul style="list-style-type: none"> Updated <i>Voltage Overshoot Prevention</i> description. Updated note in <i>Connection Setup for JTAG Single-Device Configuration using Download Cable</i> and <i>Connection Setup for JTAG Multi-Device Configuration using Download Cable</i> figures. Added steps to implement ISP clamp feature. Updated <i>Configuration Flash Memory Sectors Utilization for all Intel MAX 10 with Analog and Flash Feature Options</i> figure to include UFM sectors.
May 2016	2016.05.13	<ul style="list-style-type: none"> Changed instances of Standard POR to Slow POR to reflect Intel Quartus Prime GUI. Updated t_{CFG} to $t_{RU_nCONFIG}$. Corrected file type from .ekp to .pof in Step 8 of <i>Programming .ekp File and Encrypted .pof Separately</i>. Corrected Use secondary image ISP data as default setting when available description in <i>ICB Values and Descriptions for Intel MAX 10 Devices</i> table. Corrected CFM programming time. Added note on JTAG pin requirements when using JTAG pin sharing. Moved <i>JTAG Pin Sharing Behavior</i> under <i>Guidelines: Dual-Purpose Configuration Pin</i>. Updated configuration sequence diagram by moving 'Clears configuration RAM bits from Power-up state to Reset state. Corrected error detection port input and output for <crcblock_name> from input to none. Added example of remote system upgrade access through user interface and port definitions. Removed preliminary terms for <i>Error Detection Frequency</i> and <i>Cyclic Redundancy Check Calculation Timing</i>. Added <i>Connection Setup for JTAG Multi-Device Configuration using Download Cable</i> diagram. Updated <i>Connection Setup for JTAG Single-Device Configuration using Download Cable</i> diagram. Added new JTAG Secure design example. Edited Remote System Upgrade section title by removing in Dual Image Configuration. Updated <i>Monitored Power Supplies Ramp Time Requirement for MAX 10 Devices</i> table. Added <i>Internal Configuration Time</i>. Removed Instant ON feature. Updated User Flash Memory instances to additional UFM in <i>Configuration Flash Memory Sectors Utilization for all MAX 10 with Analog and Flash Feature Options</i> figure.
December	2015.12.14	<ul style="list-style-type: none"> Updated ICB setting description for <i>Set I/O to weak pull-up prior usermode</i> option to state the weak pull-up is enabled during configuration. Removed <i>Accessing the Remote System Upgrade Block Through User Interface</i>. Added input and output port definition for error detection WYSIWYG atom. Updated the I/O pin state to be dependent on ICB bit setting during reconfiguration.
continued...		

Date	Version	Changes
November 2015	2015.11.02	<ul style="list-style-type: none"> Removed JRunner support for JTAG configuration and link to AN 414. Updated differences in supported internal configuration mode supported based on device feature options in a table. Removed maximum number of compressed configuration image table do to redundancy. Updated Initialization Configuration Bits setting and description to reflect Quartus Prime 15.1 update. Updated Enable JTAG pin sharing and Enable nCONFIG, nSTATUS, and CONF_DONE pins to reflect Quartus II 15.1 update. Added information about ISP clamp feature. Updated information about steps to generate Raw Programming Data (.rpd). Renamed section title from <i>Configuration Total Flash Memory Programming Time</i> to <i>Configuration Flash Memory Programming Time</i>. Renamed table title from <i>Configuration Total Flash Memory Programming Time for Sectors in Intel MAX 10 Devices</i> to <i>Configuration Flash Memory Programming Time for Sectors in Intel MAX 10 Devices</i>. Added note to <i>Configuration Flash Memory Programming Time for Sectors in Intel MAX 10 Devices</i> table. Added information about internal JTAG interface and accessing internal JTAG block through user interface. Added Intel MAX 10 JTAG Secure design example.
June 2015	2015.06.15	<ul style="list-style-type: none"> Added related information link to AN 741: Remote System Upgrade for MAX 10 FPGA Devices over UART with the Nios II Processor in <i>Altera Dual Configuration IP Core References</i> and <i>Remote System Upgrade in Dual Compressed Images</i>. Added pulse holding requirement time for RU_nRSTIMER in <i>Remote System Upgrade Circuitry Signals for Intel MAX 10 Devices</i> table. Added link to <i>Remote System Upgrade Status Register—Previous State Bit for Intel MAX 10 Devices</i> table for related entries in <i>Altera Dual Configuration IP Core Avalon-MM Address Map for Intel MAX 10 Devices</i> table.
May 2015	2015.05.04	<ul style="list-style-type: none"> Rearranged and updated Configuration Setting names 'Initialization Configuration Bits for MAX 10 Devices' table. Updated 'High-Level Overview of Internal Configuration for MAX 10 Devices' figure with JTAG configuration and moved the figure to 'Configuration Schemes' section. Added link to corresponding description of configuration settings in 'Initialization Configuration Bits for MAX 10 Devices' table. Updated the default watchdog time value from hexadecimal to decimal value in 'Initialization Configuration Bits for MAX 10 Devices' table. Updated the ISP data description in 'Initialization Configuration Bits for MAX 10 Devices' table. Updated 'User Watchdog Timer' by adding time-out formula. Added link to 'User Watchdog Internal Circuitry Timing Specifications' in MAX 10 FPGA Device Datasheet. Added footnote to indicate that JTAG secure is disabled by default and require Altera support to enable in 'Initialization Configuration Bits for MAX 10 Devices' table. Updated minimum and maximum CRC calculation time for divisor 2. Updated remote system upgrade flow diagram. Updated 'Encryption in Internal Configuration' table by adding 'Key' terms and changed Image 1 and Image 2 to Image 0 and Image 1 respectively. Added footnote to 'Encryption in Internal Configuration' to indicate auto-reconfiguration when image fails. Added formula to calculate minimum and maximum CRC calculation time for other than divisor 2. Added caution when JTAG Secure is turned on.
continued...		

Date	Version	Changes
		<ul style="list-style-type: none">Added information about auto-generated .pof for certain type of internal configuration modes.Added .pof and ICB setting guide through Device and Pin Options and convert programming file.Added configuration RAM (CRAM) in 'Overview'Editorial changes.
December 2014	2014.12.15	<ul style="list-style-type: none">Rename BOOT_SEL pin to CONFIG_SEL pin.Update Altera IP Core name from Dual Boot IP Core to Altera Dual Configuration IP Core.Added information about the AES encryption key part of ICB.Added encryption feature guidelines.Updated ICB settings options available in 14.1 release.Updated Programmer options on CFM programming available in 14.1 release.
September 2014	2014.09.22	Initial release.



Intel® MAX® 10 Power Management User Guide

Updated for Intel® Quartus® Prime Design Suite: **18.0**



Online Version



Send Feedback

UG-M10PWR

ID: **683400**

Version: **2022.05.27**

Contents

1. Intel® MAX® 10 Power Management Overview.....	3
2. Intel MAX 10 Power Management Features and Architecture.....	4
2.1. Power Supply Device Options.....	4
2.1.1. Single-Supply Device.....	4
2.1.2. Dual-Supply Device.....	4
2.1.3. Comparison of the Intel MAX 10 Power Supply Device Options.....	5
2.1.4. Power Supply Design.....	6
2.2. Power-On Reset Circuitry.....	7
2.2.1. Power Supplies Monitored and Not Monitored by the POR Circuitry.....	8
2.2.2. Instant-On Support.....	9
2.3. Power Management Controller Scheme.....	9
2.3.1. Power Management Controller Architecture.....	9
2.4. Hot Socketing.....	11
2.4.1. Hot-Socketing Specifications.....	12
2.4.2. Hot-Socketing Feature Implementation.....	12
3. Power Management Controller Reference Design.....	14
3.1. Clock Control Block.....	15
3.2. I/O Buffer.....	15
3.3. Internal Oscillator.....	15
3.4. Power Management Controller.....	15
3.4.1. Entering State.....	16
3.4.2. Sleep State.....	16
3.4.3. Exiting State.....	16
3.4.4. Awake State.....	16
3.5. Entering or Exiting Sleep Mode.....	17
3.5.1. Entering Sleep Mode.....	17
3.5.2. Exiting Sleep Mode.....	18
3.5.3. Timing Parameters.....	18
3.6. Hardware Implementation and Current Measurement.....	19
4. Intel MAX 10 Power Management User Guide Archives.....	21
5. Document Revision History for the Intel MAX 10 Power Management User Guide.....	22

1. Intel® MAX® 10 Power Management Overview

Intel® MAX® 10 devices offer the following power supply device options:

- Single-supply device—requires 1 external power supply of 3.0 V or 3.3 V while offering maximum convenience and board simplicity.
- Dual-supply device—requires 2 external power supplies of 1.2 V and 2.5 V while offering the most features, highest performance, and the lowest power solution.

Related Information

- [Intel MAX 10 Power Management Features and Architecture](#) on page 4
Provides information about power management features and architecture.
- [Intel MAX 10 Power Management User Guide Archives](#) on page 21
Provides a list of user guides for previous versions.



2. Intel MAX 10 Power Management Features and Architecture

Intel MAX 10 power optimization features are as follows:

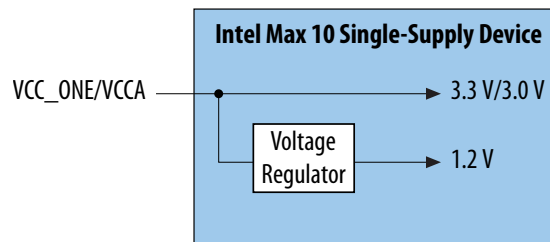
- Single-supply or dual-supply device options
- Power-on reset (POR) circuitry
- Power management controller scheme
- Hot socketing

2.1. Power Supply Device Options

2.1.1. Single-Supply Device

Intel MAX 10 single-supply devices only need either a 3.0- or 3.3-V external power supply. The external power supply serves as an input to the Intel MAX 10 device VCC_ONE and VCCA power pins. This external power supply is then regulated by an internal voltage regulator in the Intel MAX 10 single-supply device to 1.2 V. The 1.2-V voltage level is required by core logic operation.

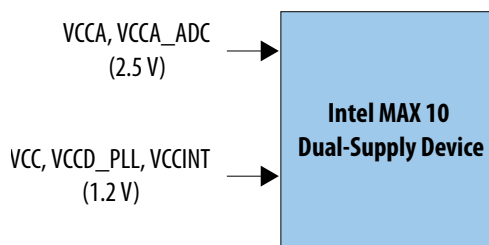
Figure 1. Intel MAX 10 Single-Supply Device



2.1.2. Dual-Supply Device

Intel MAX 10 dual-supply devices require 1.2 V and 2.5 V for the device core logic and periphery operation.

Figure 2. Intel MAX 10 Dual-Supply Device



2.1.3. Comparison of the Intel MAX 10 Power Supply Device Options

Table 1. Comparison of the Intel MAX 10 Power Supply Device Options

Characteristics	Single-Supply Device	Dual-Supply Device
Voltage regulator count ⁽¹⁾	1	2
Core and I/O performance	Low	High

For Intel MAX 10 single-supply devices, only one power supply is required—3.0 V or 3.3 V to power the core of the FPGA. The same power supply can be used to power the I/O if the same 3.0 V or 3.3 V voltage is required. If different I/O voltage is used, then additional voltage regulators will be needed.

For Intel MAX 10 dual-supply devices, two power supplies are required to supply power to the device core, periphery, phase-locked loop (PLL), and analog-to-digital converters (ADC) blocks—1.2 V and 2.5 V. Depending on the I/O standard voltage requirement, you may use two or more voltage regulators.

As the power rails for the FPGA core are supplied externally in the Intel MAX 10 dual-supply devices, the design can be optimized for power by using high efficiency switching power supplies on the board. The power savings will be equal to the increased efficiency of the regulators used compared to the internal linear regulators of the Intel MAX 10 single-supply devices. If linear regulators are used to power the Intel MAX 10 dual-supply devices, the power consumption of the Intel MAX 10 dual-supply devices will be approximately equal to the Intel MAX 10 single-supply devices.

The device performance of the single-supply device is lower than that of the dual-supply device. For the performance difference in terms of LVDS, pseudo-LVDS, digital signal processing (DSP), and internal memory performance, refer to the Intel MAX 10 FPGA device datasheet.

Related Information

Intel MAX 10 FPGA Device Datasheet

Provides details about the Intel MAX 10 performance difference in terms of LVDS, pseudo-LVDS, DSP, and internal memory performance.

⁽¹⁾ This shows the number of power supplies required by the core and periphery of the Intel MAX 10 devices. You may need additional voltage regulators to supply power to the VCCIO if the VCCIO does not have the same voltage level as the core and periphery supply.

2.1.4. Power Supply Design

Designing a power tree for a Intel MAX 10 single- or dual-supply device will vary depending on the static and dynamic power, as well as I/O and other feature utilization, for each specific use case.

The *Intel MAX 10 FPGA Device Family Pin Connection Guidelines* provides a more detailed recommendation about how to group inputs to power a Intel MAX 10 device. The Early Power Estimators (EPE) tool for Intel MAX 10 devices provides input rail power requirements and specific device recommendations based on each specific Intel MAX 10 use case.

Individual input rail voltage, current requirements, and input rail groupings are summarized on the **Report** tab.

Warning: Intel MAX 10 single-supply devices have maximum power consumption of V_{CC_ONE} , as listed in the following table. Running a design that goes beyond the maximum power consumption of V_{CC_ONE} of the Intel MAX 10 single-supply device may cause functional issue on the device. Therefore, ensure that your device does not exceed the maximum power consumption of V_{CC_ONE} when you analyze the power consumption of your design using the EPE spreadsheet.

Table 2. Maximum Power Consumption of V_{CC_ONE} for Intel MAX 10 Single-Supply Devices

Device	Maximum Power Consumption (W)
10M02S	0.778
10M04S	1.362
10M08S	1.362
10M16S	2.270
10M25S	2.943
10M40S	5.267
10M50S	5.267

Related Information

- [Intel MAX 10 FPGA Device Family Pin Connection Guidelines](#)
Provides a more detailed recommendation about how to group inputs in order to power an Intel MAX 10 device.
- [Early Power Estimators \(EPE\) and Power Analyzer](#)

2.1.4.1. Transient Current

You may observe a transient current at the V_{CCIO} power supply when powering up the Intel MAX 10 devices. The transient current of V_{CCIO} applies to all V_{CCIO} voltage levels supported by the Intel MAX 10 device.

Table 3. Maximum V_{CCIO} Power Supply Transient Current for Intel MAX 10 Devices

Device	Maximum Power Supply Transient Current (mA)	Duration (s)
10M02	220	25% of the ramp time
10M04	290	
10M08	300	
10M16	430	
10M25	510	
10M40	670	
10M50	680	

Note: The value of the transient current is based on the zero decoupling capacitance on the characterization board. The observed value will be less than the published value after adding the decoupling capacitance on your design board. Intel recommends using a soft start regulator that is able to reduce the transient current when the device is powered.

2.2. Power-On Reset Circuitry

The POR circuitry keeps the Intel MAX 10 device in the reset state until all power supplies reach the recommended operating range during device power up. The individual power supply must reach the recommended operating range within the maximum power supply ramp time, t_{RAMP} .

If the ramp time, t_{RAMP} , is not met, the Intel MAX 10 device I/O pins and programming registers remain tri-stated, during which device configuration could fail.

The Intel MAX 10 device POR circuit monitors the following power rails during power up regardless of the power supply device options:

- V_{CC} or regulated V_{CC_ONE}
- V_{CCIO} of banks 1B and 8 ⁽²⁾
- V_{CCA}

The POR circuitry also ensures V_{CCIO} level of I/O banks 1B and 8 ⁽²⁾ that contain configuration pins reach an acceptable level before configuration is triggered.

⁽²⁾ V_{CCIO} of banks 1 and 8 for the 10M02 device.

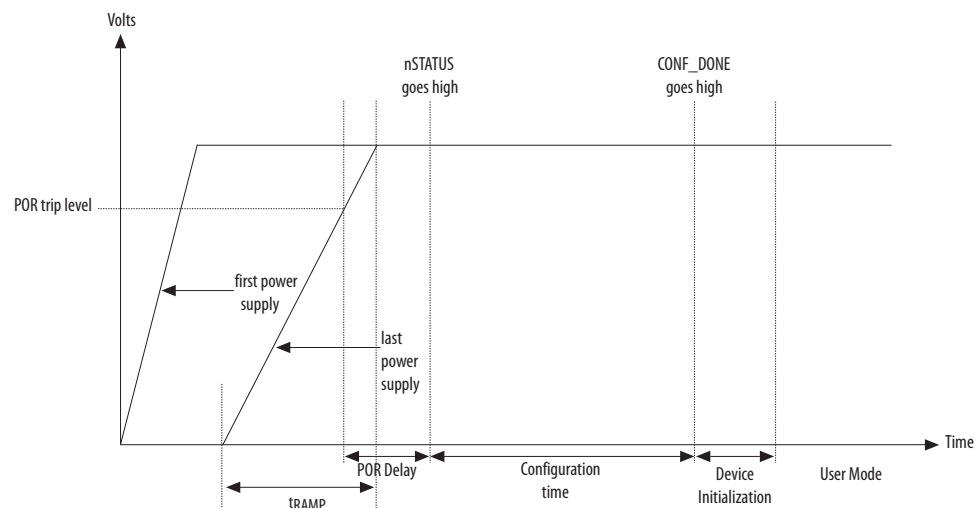
2.2.1. Power Supplies Monitored and Not Monitored by the POR Circuitry

Table 4. Power Supplies Monitored and Not Monitored by the POR Circuitry

Power Supply Device Options	Power Supplies Monitored	Power Supplies Not Monitored
Single-supply device	<ul style="list-style-type: none"> Regulated V_{CC_ONE} V_{CCA} V_{CCIO} ⁽³⁾ 	—
Dual-supply device	<ul style="list-style-type: none"> V_{CC} V_{CCA} V_{CCIO} ⁽³⁾ 	<ul style="list-style-type: none"> V_{CCD_PLL} V_{CCA_ADC} V_{CCINT}

The Intel MAX 10 POR circuitry uses an individual POR-detecting circuitry to monitor each of the configuration-related power supplies independently. The outputs of all the individual POR detectors gate the main POR circuitry. The main POR circuitry waits for all individual POR circuitries to release the POR signal before allowing the control block to start configuring the device. The main POR is released after the last ramp-up power reaches the POR trip level followed by a POR delay.

Figure 3. Monitored Power Supplies Ramp Up

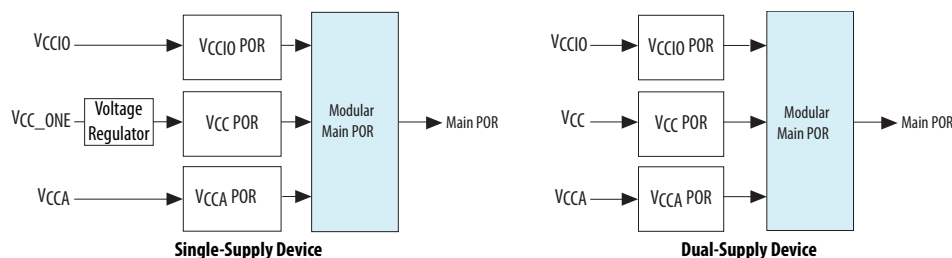


Note: Each individual power supply must reach the recommended operating range within the specified t_{RAMP} .

Note: All VCCIO banks must reach the recommended operating level before configuration completes.

Note: The typical value of POR delay is 2.5 ms for Intel MAX 10 devices.

⁽³⁾ For banks 1B and 8 for all Intel MAX 10 devices and banks 1 and 8 for the 10M02 device.

Figure 4. Simplified POR Diagram for Intel MAX 10 Devices

After the Intel MAX 10 device enters user mode, the POR circuit continues to monitor the V_{CCA} and V_{CC} power supplies. This is to detect a brown-out condition during user mode. If either the V_{CCA} or V_{CC} voltages go below the POR trip point during user mode, the main POR signal is asserted. When the main POR signal is asserted, the device is forced into reset state. $V_{CCIO}^{(3)}$ is monitored by the POR circuitry. In the event of the $V_{CCIO}^{(3)}$ voltage drops during user mode, the POR circuit does not reset the device. However, the POR circuit does monitor the V_{CCIO} voltage drop for up to 9 ms after the last power rail reaches its trip point.

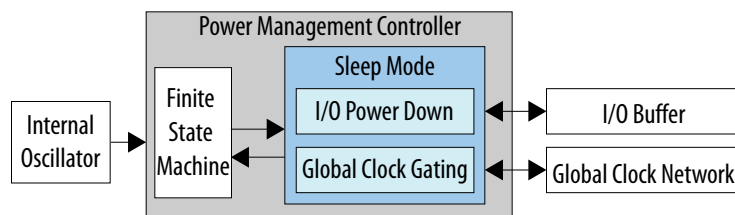
2.2.2. Instant-On Support

In some applications, it is necessary for a device to wake up very quickly to begin operation. The Intel MAX 10 device offers the instant-on feature to support fast wake-up time applications. With the instant-on feature, Intel MAX 10 devices can directly enter configuration mode with a short delay time after the POR trips for the monitored power supplies.

2.3. Power Management Controller Scheme

The power management controller scheme allows you to allocate some applications in sleep mode during runtime. This enables you to turn off portions of the design, thus reducing dynamic power consumption. You can re-enable your application with a fast wake-up time of less than 1 ms.

2.3.1. Power Management Controller Architecture

Figure 5. Power Management Controller Architecture

The Intel MAX 10 device contains hardware features that enable I/O power down and global clock (GCLK) gating to manage low-power state during sleep mode. You can power down the I/O buffer dynamically when your application is in idle or sleep mode. One example is the digital single lens reflex DSLR camera application where the LVDS I/O needs to be powered down during the idle condition. Without touching any buttons, the screen turns off while the camera is still powered on.

Intel provides a soft power management controller as reference design utilizing low-power features implemented in the Intel MAX 10 devices. You can modify the reference design based on your application. The soft power management controller includes a simple finite state machine (FSM) to manage the low-power state mode by powering down the I/O buffer and GCLK gating during sleep mode.

All Intel MAX 10 devices contain hardware features for clock gating. The 10M16, 10M25, 10M40, and 10M50 devices contain hardware features for I/O power down. With hardware features, you can manage the low-power state during sleep mode by using the soft power management controller that you define.

You can implement the power management controller in FPGA core fabric with a minimum of one I/O port reserved for sleep mode enter and exit signals.

2.3.1.1. Internal Oscillator

The internal oscillator clocks the power management controller operation. The internal oscillator is routed from flash to the core. The internal oscillator enables the power management controller to detect the wake-up event and the sleep mode event. In order to enable the internal oscillator clock when the power management controller is enabled, you have to set `oscena` to 1. For the clock frequency of the internal oscillator, refer to the *Intel MAX 10 FPGA Device Datasheet*.

Related Information

[Intel MAX 10 FPGA Device Datasheet](#)

Provides details about the Intel MAX 10 ramp time requirements, internal oscillator clock frequency, and hot-socketing specifications.

2.3.1.2. I/O Buffer Power Down

The Intel MAX 10 device has a dynamic power-down feature on some of the I/O buffers that have high-static power consumption. The dynamic power-down feature is only applicable for the I/O buffers that have been programmed for the I/O standards in the following table.

Table 5. I/O Buffer Power Down

I/O Buffer	I/O Standards	Control Port	Control Signal Capability
Input	SSTL, HSTL, HSUL, and LVDS	nsleep	1 per I/O bank ⁽⁴⁾
Output	All I/O standards	oe	1 per I/O buffer

During power-up and configuration modes, the soft power management controller is not yet configured and the control signals are forced to 1 (inactive). After configuration mode, when the power management controller is activated, the power management controller will default the control signals to 1. When control signals are 0, the power management controller powers down or tri-states the I/O buffers. Subsequently the I/O is put into the sleep mode.

The Intel MAX 10 device I/O buffers need to maintain the previous states during the sleep mode operation. The previous states in your core logics remain upon exiting the sleep mode.

⁽⁴⁾ I/O banks 1A and 1B share one control signal.

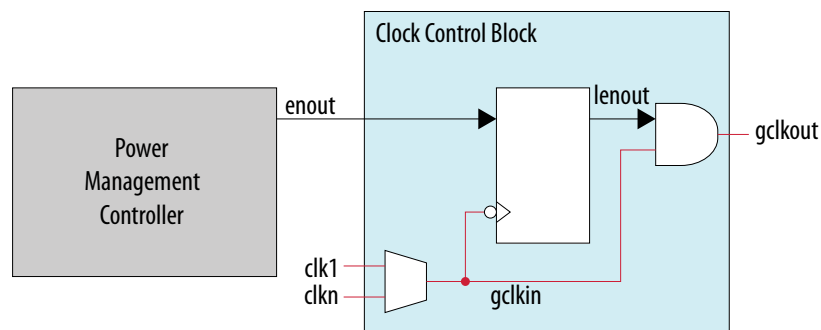
2.3.1.3. Global Clock Gating

The dynamic power-down feature is available in GCLK networks only. You can use the power management controller for the dynamic power-down of a GCLK network by controlling the active high `enout` signal. The GCLK networks serve as low-skew clock sources for functional blocks such as logic array blocks (LABs), DSP, embedded memory, and PLLs.

When a GCLK network is gated, all the logics fed by the GCLK network are in off-state. This reduces the overall power consumption of the device. The dynamic power-down feature allows core logics to control the following power-up and power-down conditions of the GCLK networks:

- Power down synchronously or asynchronously
- Power up asynchronously

Figure 6. GCLK Gating



2.4. Hot Socketing

The Intel MAX 10 device offers hot socketing, which is also known as hot plug-in or hot swap, and power sequencing support without the use of any external devices. You can insert or remove the Intel MAX 10 device on a board in a system during system operation. This does not affect the running system bus or the board that is inserted into the system.

The hot-socketing feature removes some encountered difficulties when using the Intel MAX 10 device on a PCB that contains a mixture of devices with different voltage levels.

With the Intel MAX 10 device hot-socketing feature, you no longer need to ensure a proper power-up sequence for each device on the board. Intel MAX 10 device hot-socketing feature provides:

- Board or device insertion and removal without external components or board manipulation
- Support for any power-up sequence
- Non-intrusive I/O buffers to system buses during hot insertion

2.4.1. Hot-Socketing Specifications

The Intel MAX 10 device is a hot-socketing compliant device that does not need any external components or special design requirements. Hot-socketing support in the Intel MAX 10 device has the following advantages:

- You can drive the devices before power up without damaging the device.
- I/O pins remain tri-stated during power up. The device does not drive out before or during power up, therefore not affecting other buses in operation.

2.4.1.1. Drive Intel MAX 10 Devices Before Power Up

Before or during power up or power down, you can drive signals into I/O pins, dedicated input pins, and dedicated clock pins without damaging the Intel MAX 10 devices.

The Intel MAX 10 device supports any power-up or power-down sequence to simplify system-level design.

2.4.1.2. I/O Pins Remain Tri-stated During Power up

The output buffers of the Intel MAX 10 device are turned off during system power up or power down. The Intel MAX 10 device family does not drive out until the device is configured and working in recommended operating conditions. The I/O pins are tri-stated during power up or power down.

A possible concern for semiconductor devices in general regarding hot-socketing is the potential for latch up. Latch up can occur when electrical subsystems are hot-socketed into an active system. During hot-socketing, the signal pins may be connected and driven by the active system. This occurs before the power supply can provide current to the V_{CC} of the device and ground planes. This condition can lead to latch up and cause a low-impedance path from V_{CC} to ground in the device. As a result, the device extends a large amount of current, possibly causing electrical damage.

The design of the I/O buffers and hot-socketing circuitry ensures that the Intel MAX 10 device family is immune to latch up during hot-socketing.

Related Information

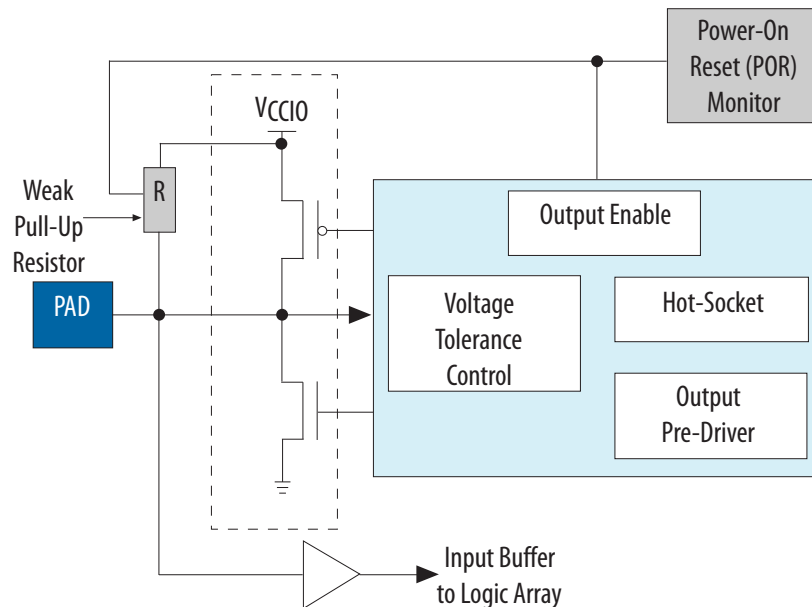
[Intel MAX 10 FPGA Device Datasheet](#)

Provides details about the Intel MAX 10 ramp time requirements, internal oscillator clock frequency, and hot-socketing specifications.

2.4.2. Hot-Socketing Feature Implementation

The hot-socketing feature tri-states the output buffer during the power-up (V_{CCIO} or V_{CC} power supplies) or power-down event. The hot-socketing circuitry generates an internal `HOTSKT` signal when V_{CCIO} or V_{CC} is below the threshold voltage during power up or power down. The `HOTSKT` signal cuts off the output buffer to ensure that no DC current leaks through the pin. Each I/O pin has the circuitry shown in the following figure. The hot-socketing circuit does not include `CONF_DONE` and `nSTATUS` pins to ensure that these pins are able to operate during configuration. Thus, it is an expected behavior for these pins to drive out during power-up and power-down sequences.

Figure 7. Hot-Socketing Circuitry for Intel MAX 10 Devices



The POR circuit monitors the voltage level of power supplies and keeps the I/O pins tri-stated during power up. The weak pull-up resistor in Intel MAX 10 device I/O elements (IOE) keeps the I/O pins from floating. The voltage tolerance control circuit protects the I/O pins from being driven before V_{CCIO} and V_{CC} supplies are powered up. This prevents the I/O pins from driving out when the device is not in user mode.

Intel uses GND as reference for hot-socketing operation and I/O buffer designs. To ensure proper operation, Intel recommends connecting the GND between boards before connecting the power supplies. This prevents the GND on your board from being pulled up inadvertently by a path to power through other components on your board. A pulled up GND can cause an out-of-specification I/O voltage or current condition with the Intel FPGA.

3. Power Management Controller Reference Design

This reference design utilizes the low-power feature supported in Intel MAX 10 devices. The following figure shows the related block diagrams in the power management controller reference design.

Figure 8. Power Management Controller Block Diagram

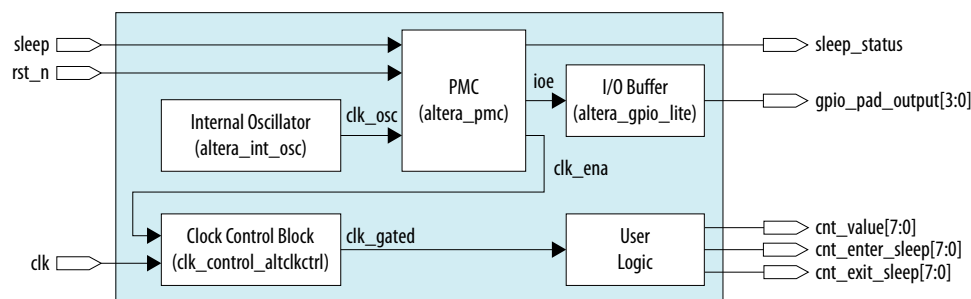


Table 6. Input and Output Ports of the Power Management Controller Reference Design

Port Name	Input/Output	Description
sleep	Input	Sleep control.
rst_n	Input	Active low reset signal.
clk	Input	Clock signal.
sleep_status	Output	Sleep status of the system. This signal is asserted high when the system is entering the sleep mode condition. This signal is de-asserted when the system exits the sleep mode condition completely.
gpio_pad_output[3:0]	Output	General-purpose I/O (GPIO) output ports.
cnt_value[7:0]	Output	Free-running counter value in user logic.
cnt_enter_sleep[7:0]	Output	Counter value when the system is entering sleep mode condition.
cnt_exit_sleep[7:0]	Output	Counter value when the system is exiting sleep mode condition.

The power management controller design is a FSM showing the state of powering down and powering up global clocks (GCLKs) and I/O buffers. The internal oscillator, clock control block, and I/O buffer are intellectual property (IP) that are supported by the Intel Quartus® Prime software and you can instantiate the IPs from the IP catalog. The user logic can be any logical circuitry that are implemented using logic element (LE) and an embedded component such as DSP and internal memory in your design. In this reference design, the user logic used is a free-running 8-bit counter. The cnt_enter_sleep and cnt_exit_sleep ports are used to ensure user logic can

enter and exit sleep mode without data corruption. It is expected for that `cnt_enter_sleep[7:0]` and `cnt_exit_sleep[7:0]` are at the same value after the user logic enter and exit sleep mode. `gpio_pad_output` ports demonstrate tri-stated state of the GPIO when the system is in sleep mode.

Related Information

[Power Management Controller Reference Design](#)

3.1. Clock Control Block

The ALTCLKCTRL Intel FPGA IP core (`clk_control_altclkctrl`) is an IP provided in the Intel Quartus Prime software. This IP is used to control the clock system in the device. The GCLKs that drive through the device can be dynamically powered down by controlling the active high `ena` signal. The `ena` port is an input to the clock control IP block. When this IP is instantiated, select the `ena` port to enable the controls of GCLKs.

Related Information

[ALTCLKCTRL Intel FPGA IP Core User Guide](#)

3.2. I/O Buffer

The GPIO Lite Intel FPGA IP core (`altera_gpio_lite`) is implemented as an input, output, or bidirectional I/O buffer. You can control the power down of these I/O buffers by enabling the `nsleep` port of the input buffer and the `oe` port of the output buffer. The `oe` and `nsleep` ports are pulled low by the power management controller design to power down the I/O buffers during sleep mode. Intel recommends using a separate GPIO Lite Intel FPGA IP core when some of the I/O buffer is not required to be powered down.

Related Information

[GPIO Lite Intel FPGA IP Core References](#)

3.3. Internal Oscillator

Internal Oscillator Intel FPGA IP core (`altera_in_osc`) is a free-running oscillator once you enable it. This oscillator runs throughout the entire power management controller design.

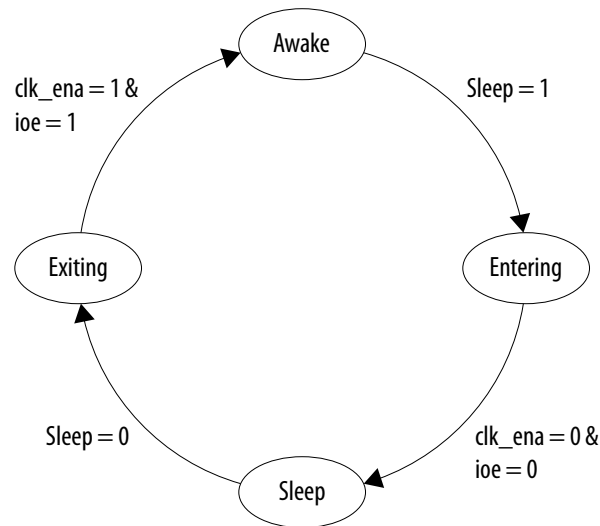
Related Information

[Internal Oscillator Intel FPGA IP Core](#)

3.4. Power Management Controller

The power management controller implements a simple FSM to control the power-up and power-down sequences of the GCLK networks and I/O buffer.

Figure 9. FSM of the Power Management Controller



3.4.1. Entering State

When the power management controller detects a sleep event, the FSM transitions to the Entering state and performs power-down operation on I/O buffers and GCLK networks. A sleep event is detected when the `sleep` signal is asserted. A sleep event could be triggered by an internal or external request.

3.4.2. Sleep State

After the power-down operation on I/O buffers and GCLK networks, the FSM transitions to the Sleep state and waits for the wake-up event. This state is the sleep mode state.

3.4.3. Exiting State

When the power management controller detects a wake-up event, the FSM transitions to the Exiting state and performs power-up operation on I/O buffers and GCLK networks. A wake-up event is detected when the `sleep` signal is de-asserted. A wake-up event could be triggered by an internal or external request such as interruption or time-out on some counters.

3.4.4. Awake State

After the power-up operation on I/O buffers and GCLK networks, the FSM transitions to the Awake state.

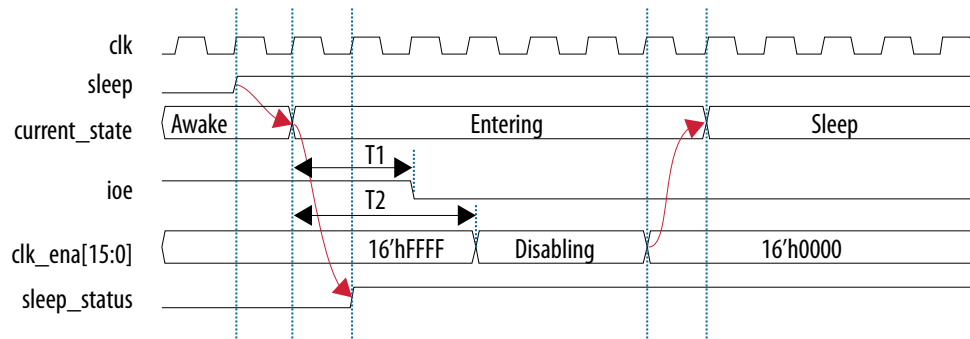
This process repeats when a sleep event is initiated again.

3.5. Entering or Exiting Sleep Mode

During power-up and configuration modes, the `sleep` signal must be low. When the `sleep` signal is asserted, the device immediately enters sleep mode. Upon entering sleep mode, the functionality of the device such as GCLK networks and I/O buffers are dynamically powered down—to minimize dynamic power dissipation. All configuration data is retained when the device is in the sleep mode.

3.5.1. Entering Sleep Mode

Figure 10. Entering Sleep Mode Timing Diagram

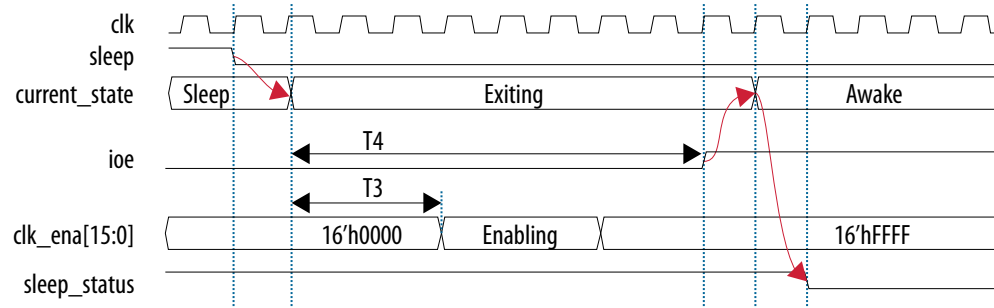


The following sequence occurs when the device enters sleep mode:

1. An internal or external request drives the `sleep` signal high, forcing the device to go into sleep mode.
2. After a delay of $T1$, the power management controller powers down all the I/O buffers by de-asserting `ioe` signal that connects to `oe` and `nsleep` ports of the I/O buffers.
3. After a delay of $T2$, the power management controller turns off all GCLK networks by disabling `clk_ena[15:0]` signal from LSB to MSB. After three clock cycles, the `clk_ena[15:0]` signal is fully disabled and transits into the sleep state.
4. The power management controller remains in sleep state until the `sleep` signal is de-asserted.
5. User logic will latch the running counter value before entering the sleep state and output to `cnt_sleep_enter` port. The running counter is then frozen.
6. `gpio_pad_output` (GPIO) is tri-stated when `ioe` is de-asserted.

3.5.2. Exiting Sleep Mode

Figure 11. Exiting Sleep Mode Timing Diagram



The following sequence occurs when the device exits sleep mode:

1. An internal or external request drives the `sleep` signal low, forcing the device to exit sleep mode.
2. After a delay of T3, the power management controller turns on all GCLK networks by enabling `clk_ena[15:0]` signal from LSB to MSB. After three clock cycles, the `clk_ena[15:0]` signal is fully enabled and all GCLK networks are turned on.
3. After a delay of T4, the power management controller powers up all the I/O buffers by asserting the `ioe` signal.
4. The power management controller remains in awake state until the `sleep` signal is asserted.
5. User logic will latch the running counter value before the awake state and output to `cnt_sleep_exit` port. The running counter is then release from freeze.
6. `gpio_pad_output` (GPIO) is driving its output value when `ioe` is asserted.

3.5.3. Timing Parameters

The following table lists the definition and minimum value of the T1, T2, T3, and T4 parameters in the entering sleep mode timing diagram and exiting sleep mode timing diagram, respectively.

Table 7. T1, T2, T3, and T4 Parameters Minimum Value and Definition

Parameter	Width (bits)	Minimum Value (Clock Cycle)	Description
T1	6	1	<code>ioe</code> disable timing.
T2	6	11	<code>clk_ena</code> disable timing.
T3	6	1	<code>clk_ena</code> enable timing.
T4	6	40	<code>ioe</code> enable timing.

T1, T2, T3, and T4 can be increased based on your system requirement.

3.6. Hardware Implementation and Current Measurement

This design is implemented using the 10M50DAF484C6 device. You can implement this design using any Intel MAX 10 device. This design runs on the Intel MAX 10 Development Kit Board to show current and power relative between user mode and sleep mode.

The resource utilization of this design is as follows:

- 42,000 LEs (84% of total LEs)—gray counter top module utilizes most of the LEs in the device
- 33 I/O pins (9% of total pins)—covering 3 input pins and 30 output pins

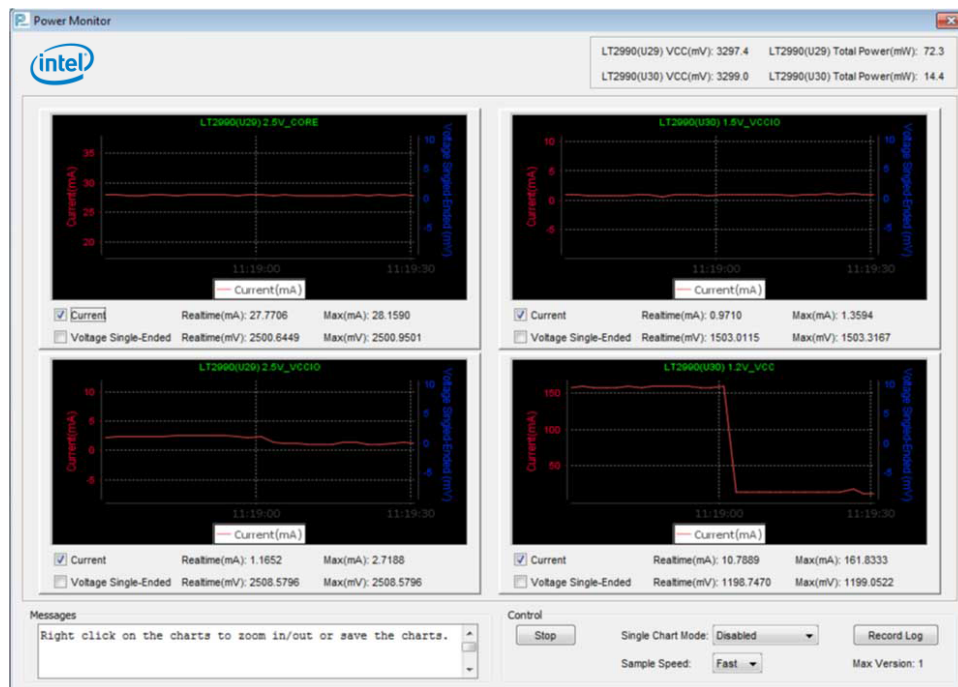
The current in this design is measured using a current monitor component (the Linear Technologies LTC 2990). The measured current is further processed by a pre-programmed design in a MAX II device. The measured current is shown on Intel FPGA power monitor GUI when the PowerMonitor.exe is launched. You will see a current monitor for each of the main supplies to the Intel MAX 10 device as follows:

- 2.5V_CORE⁽⁵⁾
- 2.5V_VCCIO
- 1.5V_VCCIO
- 1.2V_VCC

For design demonstration purpose, the push button is used for sleep control and the LEDs are used for sleep status. Thus, these signals have been inverted on the pin level. To enter sleep mode, press and hold the push button USER_PB0. To release the design to user mode, release the push button USER_PB0. LED0 indicates the sleep status of the device. LED0 is turned on when the device enters sleep mode and is turned off when the device is in user mode. During sleep mode, `gpio_pad_output` ports connecting to LED1–LED4 are tri-stated and then turned off.

⁽⁵⁾ This is 2.5V_VCCA.

Figure 12. Current Monitor for Each Supply



In sleep mode, all GCLK networks are gated and all output buffers are disabled.

Table 8. Comparison of Current and Power Consumption

Current and Power	User Mode	Sleep Mode
1.2V_ICC (mA)	160	11
2.5V_ICCA (mA)	28	28
1.5V_ICCIO (mA)	1.3	1.0
2.5V_ICCIO (mA)	2.7	1.2
Total power (mW)	270	88

The results show an approximate 93% reduction in the core current (1.2V_ICC) consumption and an approximate 56% reduction in I/O current (2.5V_ICCIO) consumption in sleep mode relative to user mode. The total power consumption reduction in this design in sleep mode is about 68%.

4. Intel MAX 10 Power Management User Guide Archives

If an IP core version is not listed, the user guide for the previous IP core version applies.

IP Core Version	User Guide
15.1	MAX 10 Power Management User Guide
15.0	MAX 10 Power Management User Guide
14.1	MAX 10 Power Management User Guide

5. Document Revision History for the Intel MAX 10 Power Management User Guide

Document Version	Intel Quartus Prime Version	Changes
2022.05.27	18.0	Removed instances of Enpirion from <i>Intel® MAX® 10 Power Management Overview</i> and <i>Power Supply Design</i> sections.
2018.07.04	18.0	<ul style="list-style-type: none"> Updated the <i>Power-On Reset Circuitry</i> section to include information that the individual power supply must reach the recommended operating range within the maximum power supply ramp time, t_{RAMP}. Beginning from the Intel Quartus Prime software version 18.0, the name of this IP core has been changed from Clock Control Block (ALTCLKCTRL) IP Core to ALTCLKCTRL Intel FPGA IP Core. Beginning from the Intel Quartus Prime software version 18.0, the name of this IP core has been changed from Altera GPIO Lite IP Core to GPIO Lite Intel FPGA IP Core. Beginning from the Intel Quartus Prime software version 18.0, the name of this IP core has been changed from Internal Oscillator IP Core to Internal Oscillator Intel FPGA IP Core. Updated Powerplay Early Power Estimator (EPE) to Early Power Estimator.

Date	Version	Changes
May 2017	2017.05.26	Updated the Hot-Socketing Feature Implementation section.
February 2017	2017.02.21	Rebranded as Intel.
May 2016	2016.05.02	<ul style="list-style-type: none"> Updated the I/O Pins Remain Tri-stated During Power Up section. Updated the Power Supplies Monitored and Not Monitored by the POR Circuitry section. Updated the information for the single-supply device in the Instant-On Power-Up Sequence Requirement table.
November 2015	2015.11.02	<ul style="list-style-type: none"> Added the Transient Current section. Changed instances of Quartus II to Quartus Prime.
February 2015	2015.02.09	Added the MAX 10 Power Management Controller Reference Design.
December 2014	2014.12.15	<ul style="list-style-type: none"> Updated the MAX 10 Power Management Overview section. Updated the Dual-Supply Device section to update details on power consumption for dual-supply devices. Updated the Power Supply Design section to include the maximum power consumption for each Intel MAX 10 single-supply device. Updated the Power Management Controller Scheme section to include updates on sleep mode.
September 2014	2014.09.22	Initial release.

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.



Intel® MAX™ 10 Embedded Memory User Guide



Online Version



Send Feedback

UG-M10MEMORY

683431

2023.05.05

Contents

1. Intel® MAX® 10 Embedded Memory Overview.....	4
2. Intel MAX 10 Embedded Memory Architecture and Features.....	5
2.1. Intel MAX 10 Embedded Memory General Features.....	5
2.1.1. Control Signals.....	5
2.1.2. Parity Bit.....	6
2.1.3. Read Enable.....	6
2.1.4. Read-During-Write.....	6
2.1.5. Byte Enable.....	7
2.1.6. Packed Mode Support.....	8
2.1.7. Address Clock Enable Support.....	8
2.1.8. Asynchronous Clear.....	10
2.2. Intel MAX 10 Embedded Memory Operation Modes.....	10
2.2.1. Supported Memory Operation Modes.....	10
2.3. Intel MAX 10 Embedded Memory Clock Modes.....	12
2.3.1. Asynchronous Clear in Clock Modes.....	12
2.3.2. Output Read Data in Simultaneous Read and Write.....	13
2.3.3. Independent Clock Enables in Clock Modes.....	13
2.4. Intel MAX 10 Embedded Memory Configurations.....	13
2.4.1. Port Width Configurations.....	13
2.4.2. Memory Configurations for Single-Port Modes.....	13
2.4.3. Memory Configurations for Dual-Port Modes.....	14
2.4.4. Maximum Block Depth Configuration.....	14
3. Intel MAX 10 Embedded Memory Design Consideration.....	16
3.1. Implement External Conflict Resolution.....	16
3.2. Customize Read-During-Write Behavior.....	16
3.2.1. Same-Port Read-During-Write Mode.....	16
3.2.2. Mixed-Port Read-During-Write Mode.....	18
3.3. Consider Power-Up State and Memory Initialization.....	18
3.4. Control Clocking to Reduce Power Consumption.....	19
3.5. Selecting Read-During-Write Output Choices.....	20
4. RAM: 1-Port IP Core References.....	21
4.1. RAM: 1-Port IP Core Signals For Intel MAX 10 Devices.....	21
4.2. RAM: 1-Port IP Core Parameters For Intel MAX 10 Devices.....	23
5. RAM: 2-PORT IP Core References.....	26
5.1. RAM: 2-Ports IP Core Signals (Simple Dual-Port RAM) For Intel MAX 10 Devices.....	29
5.2. RAM: 2-Port IP Core Signals (True Dual-Port RAM) for Intel MAX 10 Devices.....	30
5.3. RAM: 2-Port IP Core Parameters for Intel MAX 10 Devices.....	32
6. ROM: 1-PORT IP Core References.....	38
6.1. ROM: 1-PORT IP Core Signals For Intel MAX 10 Devices.....	39
6.2. ROM: 1-PORT IP Core Parameters for Intel MAX 10 Devices.....	40
7. ROM: 2-PORT IP Core References.....	43
7.1. ROM: 2-PORT IP Core Signals for Intel MAX 10 Devices.....	45
7.2. ROM: 2-Port IP Core Parameters For Intel MAX 10 Devices.....	46

8. FIFO IP Core References.....	49
8.1. FIFO IP Core Signals for Intel MAX 10 Devices.....	50
8.2. FIFO IP Core Parameters for Intel MAX 10 Devices.....	51
8.3. FIFO Functional Timing Requirements.....	53
8.4. SCFIFO ALMOST_EMPTY Functional Timing.....	54
9. Shift Register (RAM-based) IP Core References.....	56
9.1. Shift Register (RAM-based) IP Core Signals for Intel MAX 10 Devices.....	56
9.2. Shift Register (RAM-based) IP Core Parameters for Intel MAX 10 Devices.....	57
10. ALTMEMMULT IP Core References.....	58
10.1. ALTMEMMULT IP Core Signals for Intel MAX 10 Devices.....	58
10.2. ALTMEMMULT IP Core Parameters for Intel MAX 10 Devices.....	59
11. Document Revision History for the Intel MAX 10 Embedded Memory User Guide.....	60



1. Intel® MAX® 10 Embedded Memory Overview

Intel® MAX® 10 embedded memory block is optimized for applications such as high throughput packet processing, embedded processor program, and embedded data storage.



2. Intel MAX 10 Embedded Memory Architecture and Features

The Intel MAX 10 embedded memory structure consists of 9,216-bit (including parity bits) blocks. You can use each M9K block in different widths and configuration to provide various memory functions such as RAM, ROM, shift registers, and FIFO.

The following list summarizes the Intel MAX 10 embedded memory features:

- Embedded memory general features
- Embedded memory operation modes
- Embedded memory clock modes

Related Information

[Intel MAX 10 Device Overview](#)

For information about Intel MAX 10 devices embedded memory capacity and distribution.

2.1. Intel MAX 10 Embedded Memory General Features

Intel MAX 10 embedded memory supports the following general features:

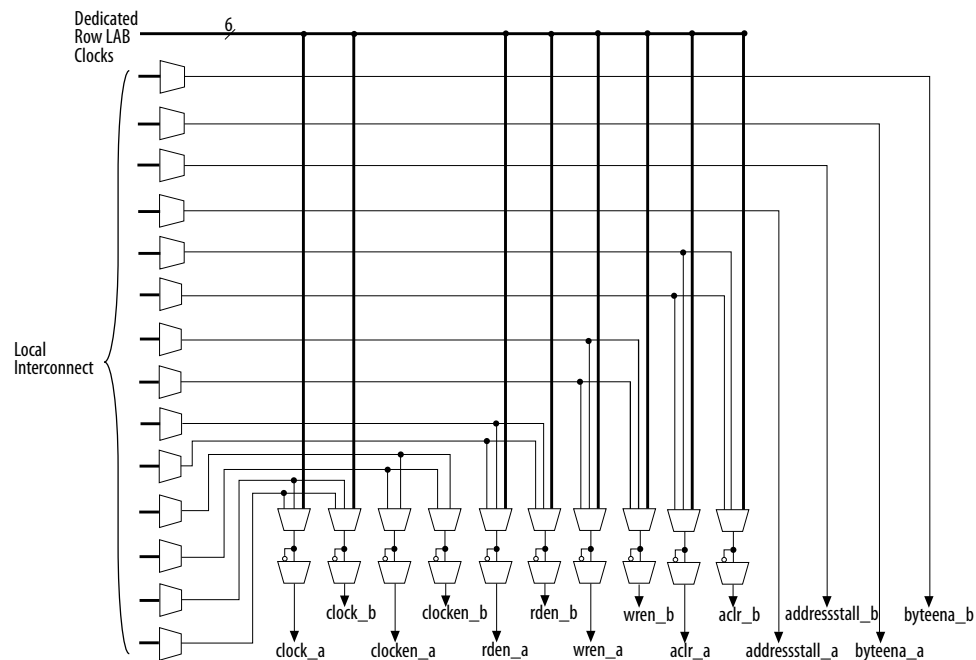
- 8,192 memory bits per block (9,216 bits per block including parity).
- Independent read-enable (`rden`) and write-enable (`wren`) signals for each port.
- Packed mode in which the M9K memory block is split into two 4.5 K single-port RAMs.
- Variable port configurations.
- Single-port and simple dual-port modes support for all port widths.
- True dual-port (one read and one write, two reads, or two writes) operation.
- Byte enables for data input masking during writes.
- Two clock-enable control signals for each port (port A and port B).
- Initialization file to preload memory content in RAM and ROM modes.

2.1.1. Control Signals

The clock-enable control signal controls the clock entering the input and output registers and the entire M9K memory block. This signal disables the clock so that the M9K memory block does not see any clock edges and does not perform any operations.

The `rden` and `wren` control signals control the read and write operations for each port of the M9K memory blocks. You can disable the `rden` or `wren` signals independently to save power whenever the operation is not required.

Figure 1. Register Clock, Clear, and Control Signals Implementation in M9K Embedded Memory Block



2.1.2. Parity Bit

You can perform parity checking for error detection with the parity bit along with internal logic resources. The M9K memory blocks support a parity bit for each storage byte. You can use this bit as either a parity bit or as an additional data bit. No parity function is actually performed on this bit. If error detection is not desired, you can use the parity bit as an additional data bit.

2.1.3. Read Enable

M9K memory blocks support the read enable feature for all memory modes.

Table 1. Effects of Read Enable on Data Output Port

If you...	...Then
Create the read-enable port and perform a write operation with the read enable port deasserted.	The data output port retains the previous values from the most recent active read enable.
Activate the read enable during a write operation or do not create a read-enable signal.	The output port shows either the new data being written and the old data at that address, or a "Don't Care" value when read-during-write occurs at the same address location.

2.1.4. Read-During-Write

The read-during-write operation occurs when a read operation and a write operation target the same memory location at the same time.

The read-during-write operation operates in the following ways:

- Same-port
- Mixed-port

Related Information

[Customize Read-During-Write Behavior](#) on page 16

2.1.5. Byte Enable

- Memory block that are implemented as RAMs support byte enables.
- The byte enable controls mask the input data, so that only specific bytes of data are written. The unwritten bytes retain the values written previously.
- The write enable (`wren`) signal, together with the byte enable (`byteena`) signal, control the write operations on the RAM blocks. By default, the `byteena` signal is high (enabled) and only the `wren` signal controls the writing.
- The byte enable registers do not have a `clear` port.
- M9K blocks support byte enables when the write port has a data width of $\times 16$, $\times 18$, $\times 32$, or $\times 36$ bits.
- Byte enables operate in a one-hot fashion. The Least Significant Bit (LSB) of the `byteena` signal corresponds to the LSB of the data bus. For example, if `byteena` = 01 and you are using a RAM block in $\times 18$ mode, `data[8:0]` is enabled and `data[17:9]` is disabled. Similarly, if `byteena` = 11, both `data[8:0]` and `data[17:9]` are enabled.
- Byte enables are active high.

2.1.5.1. Byte Enable Controls

Table 2. M9K Blocks Byte Enable Selections

byteena[3:0]	Affected Bytes. Any Combination of Byte Enables is Possible.			
	datain $\times 16$	datain $\times 18$	datain $\times 32$	datain $\times 36$
[0] = 1	[7:0]	[8:0]	[7:0]	[8:0]
[1] = 1	[15:8]	[17:9]	[15:8]	[17:9]
[2] = 1	—	—	[23:16]	[26:18]
[3] = 1	—	—	[31:24]	[35:27]

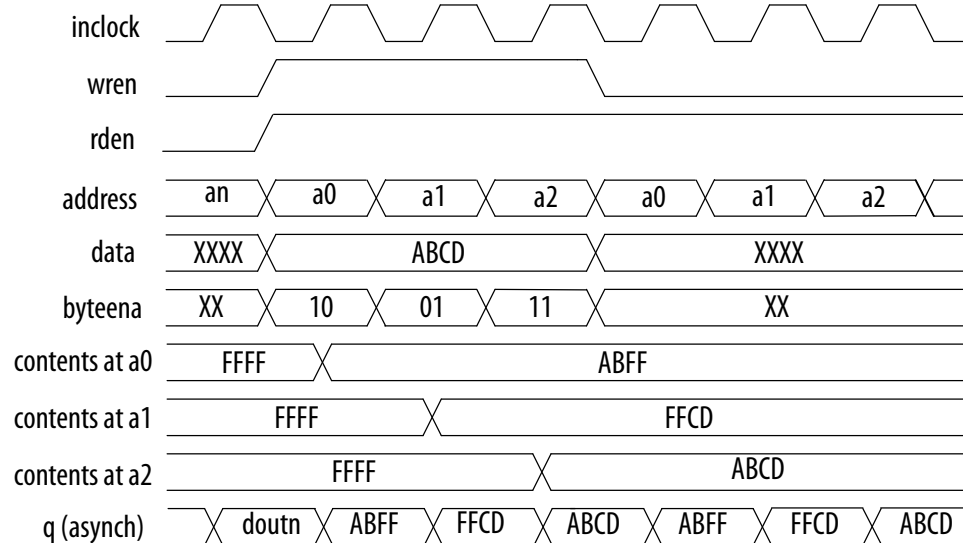
2.1.5.2. Data Byte Output

If you...	...Then
Deassert a byte-enable bit during a write cycle	The old data in the memory appears in the corresponding data-byte output.
Assert a byte-enable bit during a write cycle	The corresponding data-byte output depends on the Intel Quartus® Prime software setting. The setting can be either the newly written data or the old data at that location.

2.1.5.3. RAM Blocks Operations

Figure 2. Byte Enable Functional Waveform

This figure shows how the `wren` and `byteena` signals control the RAM operations.



For this functional waveform, New Data Mode is selected.

2.1.6. Packed Mode Support

You can implement two single-port memory blocks in a single block under the following conditions:

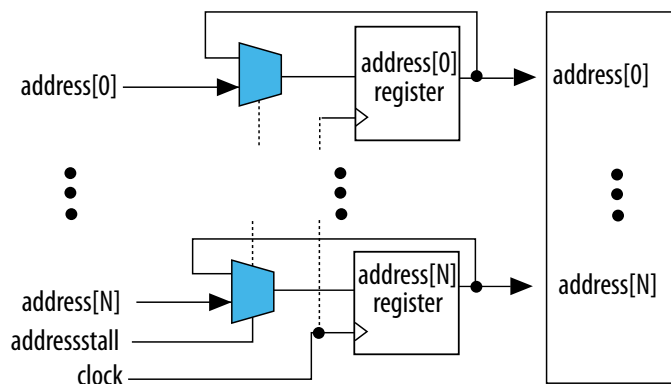
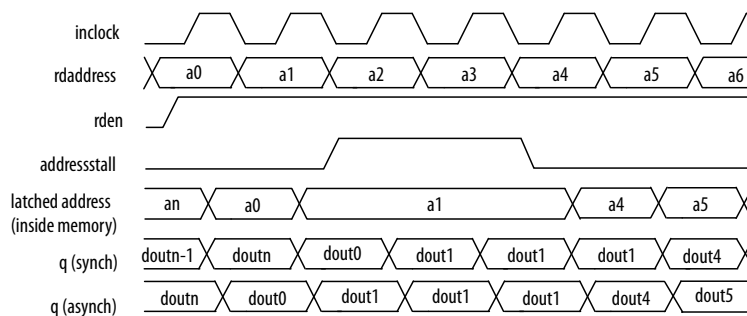
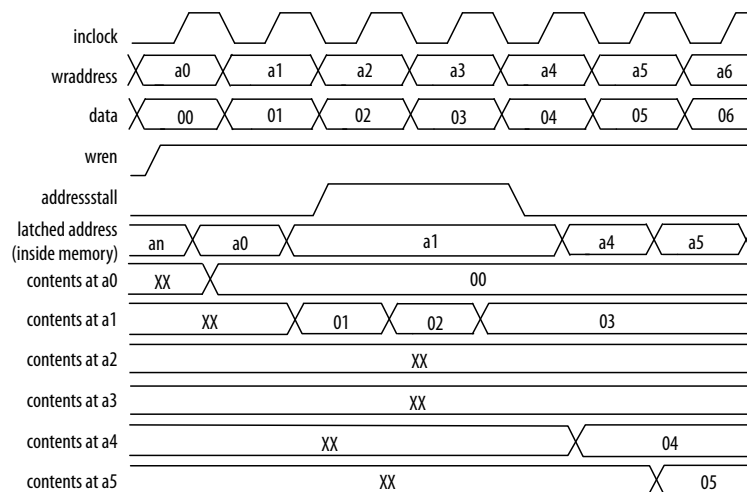
- Each of the two independent block sizes is less than or equal to half of the M9K block size. The maximum data width for each independent block is 18 bits wide.
- Each of the single-port memory blocks is configured in single-clock mode.

Related Information

[Intel MAX 10 Embedded Memory Clock Modes](#) on page 12

2.1.7. Address Clock Enable Support

- The address clock enable feature holds the previous address value for as long as the address clock enable signal (`addressstall`) is enabled (`addressstall = 1`).
- When you configure M9K memory blocks in dual-port mode, each port has its own independent address clock enable.
- Use the address clock enable feature to improve the effectiveness of cache memory applications during a cache-miss.
- The default value for the `addressstall` signal is low.
- The address register output feeds back to its input using a multiplexer. The `addressstall` signal selects the multiplexer output.

Figure 3. Address Clock Enable Block Diagram**2.1.7.1. Address Clock Enable During Read Cycle Waveform****Figure 4. Address Clock Enable Waveform During Read Cycle****2.1.7.2. Address Clock Enable During Write Cycle Waveform****Figure 5. Address Clock Enable Waveform During Write Cycle**

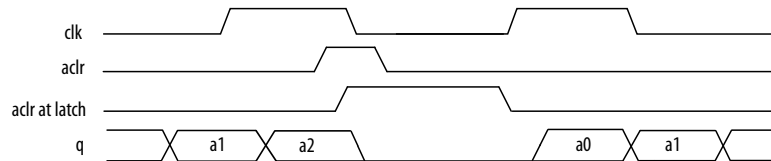
2.1.8. Asynchronous Clear

You can selectively enable asynchronous clear per logical memory using the RAM: 1-PORT and RAM: 2-PORT IP cores.

Support of asynchronous clear in the M9k memory block:

- Read address registers—input registers other than read address registers are not supported. Asserting asynchronous clear to the read address register during a read operation might corrupt the memory content.
- Output registers—if applied to output registers, the asynchronous clear signal clears the output registers and the effects are immediate. If your RAM does not use output registers, you can still clear the RAM outputs using the output latch asynchronous clear feature.
- Output latches

Figure 6. Output Latch Asynchronous Clear Waveform



2.1.8.1. Resetting Registers in M9K Blocks

There are three ways to reset registers in the M9K blocks:

- Power up the device
- Use the `aclr` signal for output register only
- Assert the device-wide reset signal using the **DEV_CLRn** option

2.2. Intel MAX 10 Embedded Memory Operation Modes

The M9K memory blocks allow you to implement fully-synchronous SRAM memory in multiple operation modes. The M9K memory blocks do not support asynchronous (unregistered) memory inputs.

Note: Violating the setup or hold time on the M9K memory block input registers may corrupt memory contents. This applies to both read and write operations.

2.2.1. Supported Memory Operation Modes

Table 3. Supported Memory Operation Modes in the M9K Embedded Memory Blocks

Memory Operation Mode	Related IP Core	Description
Single-port RAM	RAM: 1-PORT IP Core	Single-port mode supports non-simultaneous read and write operations from a single address.
continued...		

Memory Operation Mode	Related IP Core	Description
		<p>Use the read enable port to control the RAM output ports behavior during a write operation:</p> <ul style="list-style-type: none"> To show either the new data being written or the old data at that address, activate the read enable (<code>rden</code>) during a write operation. To retain the previous values that are held during the most recent active read enable, perform the write operation with the read enable port deasserted.
Simple dual-port RAM	RAM: 2-PORT IP Core	<p>You can simultaneously perform one read and one write operations to different locations where the write operation happens on Port A and the read operation happens on Port B.</p> <p>In this memory mode, the M9K memory blocks support separate <code>wren</code> and <code>rden</code> signals. To save power, keep <code>rden</code> signal low (inactive) when not reading.</p>
True dual-port RAM	RAM: 2-PORT IP Core	<p>You can perform any combination of two port operations:</p> <ul style="list-style-type: none"> Two reads, two writes, or; One read and one write at two different clock frequencies. <p>In this memory mode, the M9K memory blocks support separate <code>wren</code> and <code>rden</code> signals. To save power, keep <code>rden</code> signal low (inactive) when not reading.</p>
Single-port ROM	ROM: 1-PORT IP Core	<p>Only one address port is available for read operation. You can use the memory blocks as a ROM.</p> <ul style="list-style-type: none"> Initialize the ROM contents of the memory blocks using a <code>.mif</code> or <code>.hex</code> file. The address lines of the ROM are registered. The outputs can be registered or unregistered. The ROM read operation is identical to the read operation in the single-port RAM configuration.
Dual-port ROM	ROM: 2-PORT IP Core	<p>The dual-port ROM has almost similar functional ports as single-port ROM. The difference is dual-port ROM has an additional address port for read operation.</p> <p>You can use the memory blocks as a ROM.</p> <ul style="list-style-type: none"> Initialize the ROM contents of the memory blocks using a <code>.mif</code> or <code>.hex</code> file. The address lines of the ROM are registered. The outputs can be registered or unregistered. The ROM read operation is identical to the read operation in the single-port RAM configuration.
Shift-register	Shift Register (RAM-based) IP Core	<p>You can use the memory blocks as a shift-register block to save logic cells and routing resources.</p> <p>The input data width (w), the length of the taps (m), and the number of taps (n) determine the size of a shift register ($w \times m \times n$). The size of the shift register must be less than or equal to the maximum number of memory bits (9,216 bits). The size of ($w \times n$) must be less than or equal to the maximum of width of the blocks (36 bits).</p>

continued...

Memory Operation Mode	Related IP Core	Description
		You can cascade memory blocks to implement larger shift registers.
FIFO	FIFO IP Core	<p>You can use the memory blocks as FIFO buffers.</p> <ul style="list-style-type: none"> Use the FIFO IP core in single clock FIFO (SCFIFO) mode and dual clock FIFO (DCFIFO) mode to implement single- and dual-clock FIFO buffers in your design. Use dual clock FIFO buffers when transferring data from one clock domain to another clock domain. The M9K memory blocks do not support simultaneous read and write from an empty FIFO buffer.
Memory-based multiplier	ALTMEMMULT IP Core	You can use the memory blocks as a memory-based multiplier.

2.3. Intel MAX 10 Embedded Memory Clock Modes

Clock Mode	Description	Modes				
		True Dual-Port	Simple Dual-Port	Single-Port	ROM	FIFO
Independent Clock Mode	<p>A separate clock is available for the following ports:</p> <ul style="list-style-type: none"> Port A—Clock A controls all registers on the port A side. Port B—Clock B controls all registers on the port B side. 	Yes	—	—	Yes	—
Input/Output Clock Mode	<ul style="list-style-type: none"> M9K memory blocks can implement input or output clock mode for single-port, true dual-port, and simple dual-port memory modes. An input clock controls all input registers to the memory block, including data, address, <code>byteena</code>, <code>wren</code>, and <code>rden</code> registers. An output clock controls the data-output registers. 	Yes	Yes	Yes	Yes	—
Read or Write Clock Mode	<ul style="list-style-type: none"> M9K memory blocks support independent clock enables for both the read and write clocks. A read clock controls the data outputs, read address, and read enable registers. A write clock controls the data inputs, write address, and write enable registers. 	—	Yes	—	—	Yes
Single-Clock Mode	A single clock, together with a clock enable, controls all registers of the memory block.	Yes	Yes	Yes	Yes	Yes

Related Information

- [Packed Mode Support](#) on page 8
- [Control Clocking to Reduce Power Consumption](#) on page 19
- [Output Read Data in Simultaneous Read and Write](#) on page 13

2.3.1. Asynchronous Clear in Clock Modes

In all clock modes, asynchronous clear is available only for output latches and output registers. For independent clock mode, this is applicable on port A and port B.

2.3.2. Output Read Data in Simultaneous Read and Write

If you perform a simultaneous read/write to the same address location using the read or write clock mode, the output read data is unknown. If you want the output read data to be a known value, use single-clock or input/output clock mode and then select the appropriate read-during-write behavior in the RAM: 1-PORT and RAM: 2-PORT IP cores.

Related Information

Intel MAX 10 Embedded Memory Clock Modes on page 12

2.3.3. Independent Clock Enables in Clock Modes

Table 4. Supported Clock Modes for Independent Clock Enables

Clock Mode	Description
Read/write	Supported for both the read and write clocks.
Independent	Supported for the registers of both ports.

2.4. Intel MAX 10 Embedded Memory Configurations

2.4.1. Port Width Configurations

The following equation defines the port width configuration: Memory depth (number of words) \times Width of the data input bus.

- If your port width configuration (either the depth or the width) is more than the amount an internal memory block can support, additional memory blocks (of the same type) are used. For example, if you configure your M9K as 512×36 , which exceeds the supported port width, two 512×18 M9Ks are used to implement your RAM.
- In addition to the supported configuration provided, you can set the memory depth to a non-power of two, but the actual memory depth allocated can vary. The variation depends on the type of resource implemented.
- If the memory is implemented in dedicated memory blocks, setting a non-power of two for the memory depth reflects the actual memory depth.
- When you implement your memory using dedicated memory blocks, refer to the Fitter report to check the actual memory depth.

2.4.2. Memory Configurations for Single-Port Modes

Table 5. Single-Port Memory Configurations for M9K Blocks

This table lists the configuration supported for single-port memory configuration.

Configuration	M9K Block
Depth \times width	8192×1
	4096×2
	2048×4
continued...	

Configuration	M9K Block
	1024 × 8
	1024 × 9
	512 × 16
	512 × 18
	256 × 32
	256 × 36

2.4.3. Memory Configurations for Dual-Port Modes

Table 6. Simple Dual-Port Memory Configurations for M9K Blocks

This table lists the configuration supported simple dual-port memory configuration.

Read Port	Write Port								
	8192 × 1	4096 × 2	2048 × 4	1024 × 8	512 × 16	256 × 32	1024 × 9	512 × 18	256 × 36
8192 × 1	Yes	Yes	Yes	Yes	Yes	Yes	—	—	—
4096 × 2	Yes	Yes	Yes	Yes	Yes	Yes	—	—	—
2048 × 4	Yes	Yes	Yes	Yes	Yes	Yes	—	—	—
1024 × 8	Yes	Yes	Yes	Yes	Yes	Yes	—	—	—
512 × 16	Yes	Yes	Yes	Yes	Yes	Yes	—	—	—
256 × 32	Yes	Yes	Yes	Yes	Yes	Yes	—	—	—
1024 × 9	—	—	—	—	—	—	Yes	Yes	Yes
512 × 18	—	—	—	—	—	—	Yes	Yes	Yes
256 × 36	—	—	—	—	—	—	Yes	Yes	Yes

Table 7. True Dual-Port Memory Configurations for M9K Blocks

This table lists the configuration supported true dual-port memory configuration.

Read Port	Write Port						
	8192 × 1	4096 × 2	2048 × 4	1024 × 8	512 × 16	1024 × 9	512 × 18
8192 × 1	Yes	Yes	Yes	Yes	Yes	—	—
4096 × 2	Yes	Yes	Yes	Yes	Yes	—	—
2048 × 4	Yes	Yes	Yes	Yes	Yes	—	—
1024 × 8	Yes	Yes	Yes	Yes	Yes	—	—
512 × 16	Yes	Yes	Yes	Yes	Yes	—	—
1024 × 9	—	—	—	—	—	Yes	Yes
512 × 18	—	—	—	—	—	Yes	Yes

2.4.4. Maximum Block Depth Configuration

The **Set the maximum block depth** parameter allows you to set the maximum block depth of the dedicated memory block you use. You can slice the memory block to your desired maximum block depth. For example, the capacity of an M9K block is 9,216

bits, and the default memory depth is 8K, in which each address is capable of storing 1 bit ($8K \times 1$). If you set the maximum block depth to 512, the M9K block is sliced to a depth of 512 and each address is capable of storing up to 18 bits (512×18).

Use this parameter to save power usage in your devices and to reduce the total number of memory blocks used. However, this parameter might increase the number of LEs and affects the design performance.

When the RAM is sliced shallower, the dynamic power usage decreases. However, for a RAM block with a depth of 256, the power used by the extra LEs starts to outweigh the power gain achieved by shallower slices.

The maximum block depth must be in a power of two, and the valid values vary among different dedicated memory blocks.

This table lists the valid range of maximum block depth for M9K memory blocks.

Table 8. Valid Range of Maximum Block Depth for M9K Memory Blocks

Memory Block	Valid Range
M9K	256 - 8K. The maximum block depth must be in a power of two.

The IP parameter editor prompts an error message if you enter an invalid value for the maximum block depth. Intel recommends that you set the value of the **Set the maximum block depth** parameter to **Auto** if you are unsure of the appropriate maximum block depth to set or the setting is not important for your design. The **Auto** setting enables the Compiler to select the maximum block depth with the appropriate port width configuration for the type of internal memory block of your memory.

3. Intel MAX 10 Embedded Memory Design Consideration

There are several considerations that require your attention to ensure the success of your designs.

3.1. Implement External Conflict Resolution

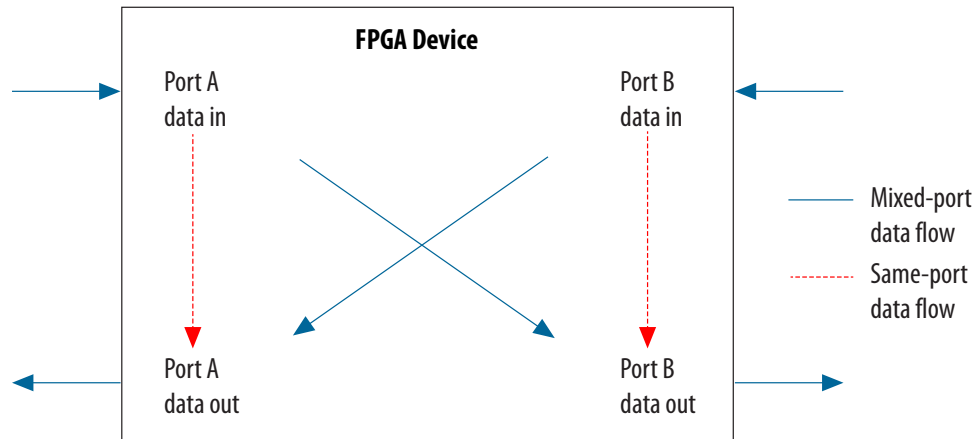
In the true dual-port RAM mode, you can perform two write operations to the same memory location. However, the memory blocks do not have internal conflict resolution circuitry.

To avoid unknown data being written to the address, implement external conflict resolution logic to the memory block.

3.2. Customize Read-During-Write Behavior

Customize the read-during-write behavior of the memory blocks to suit your design requirements.

Figure 7. Difference Between the Two Types of Read-during-Write Operations —Same Port and Mixed Port.



Related Information

[Read-During-Write](#) on page 6

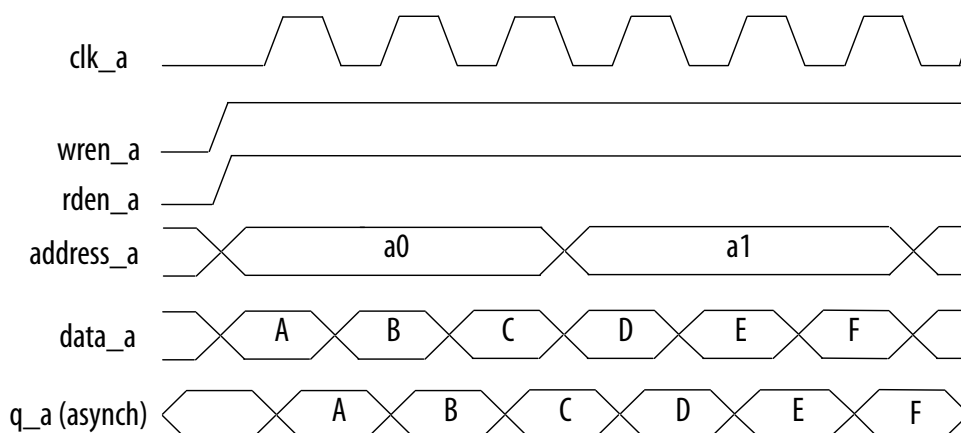
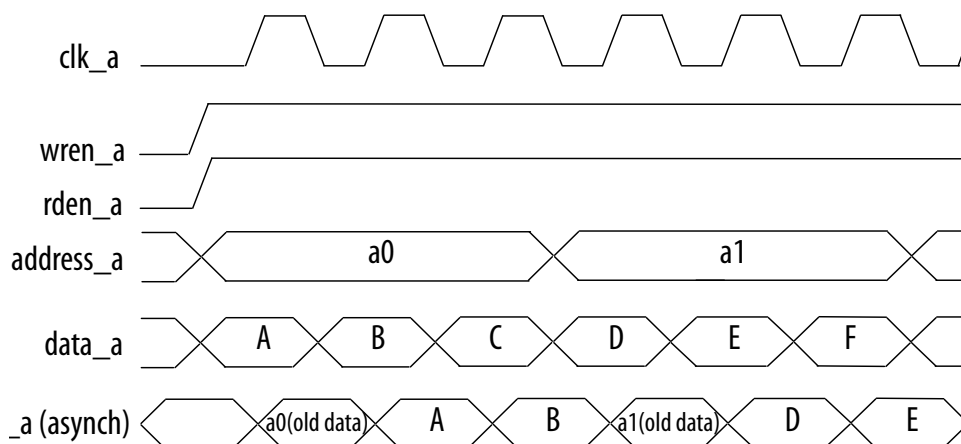
3.2.1. Same-Port Read-During-Write Mode

The same-port read-during-write mode applies to a single-port RAM or the same port of a true dual-port RAM.

Table 9. Output Modes for Embedded Memory Blocks in Same-Port Read-During-Write Mode

This table lists the available output modes if you select the embedded memory blocks in the same-port read-during-write mode.

Output Mode	Description
"new data" (flow-through)	<p>The new data is available on the rising edge of the same clock cycle on which the new data is written.</p> <p>When using New Data mode together with byte enable, you can control the output of the RAM:</p> <ul style="list-style-type: none"> When byte enable is high, the data written into the memory passes to the output (flow-through). When byte enable is low, the masked-off data is not written into the memory and the old data in the memory appears on the outputs. <p>Therefore, the output can be a combination of new and old data determined by <code>byteena</code>.</p>
"don't care"	The RAM outputs reflect the old data at that address before the write operation proceeds.

Figure 8. Same-Port Read-During-Write: New Data Mode**Figure 9. Same Port Read-During-Write: Old Data Mode**

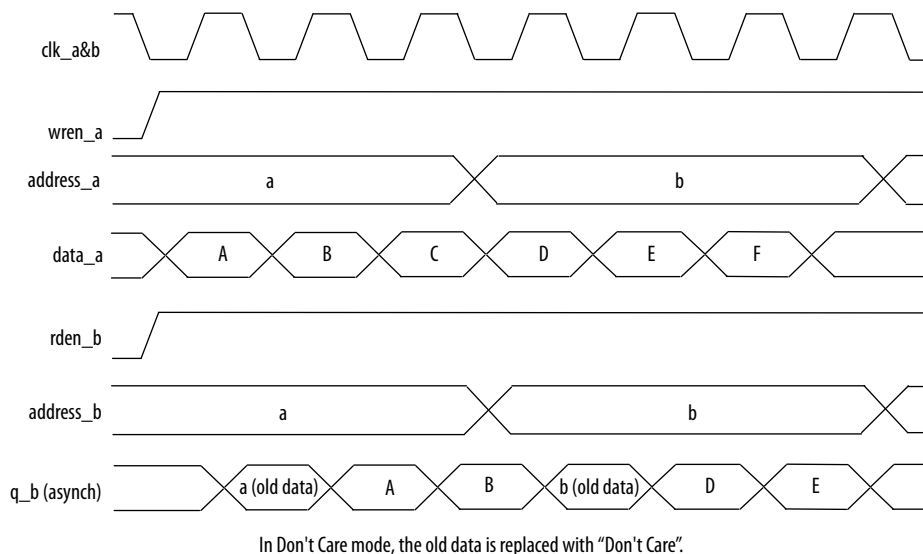
3.2.2. Mixed-Port Read-During-Write Mode

The mixed-port read-during-write mode applies to simple and true dual-port RAM modes where two ports perform read and write operations on the same memory address using the same clock—one port reading from the address, and the other port writing to it.

Table 10. Output Modes for RAM in Mixed-Port Read-During-Write Mode

Output Mode	Description
"old data"	A read-during-write operation to different ports causes the RAM output to reflect the "old data" value at the particular address.
"don't care"	The RAM outputs "don't care" or "unknown" value.

Figure 10. Mixed-Port Read-During-Write: Old Data Mode



3.2.2.1. Mixed-Port Read-During-Write Operation with Dual Clocks

For mixed-port read-during-write operation with dual clocks, the relationship between the clocks determines the output behavior of the memory.

If You...	...Then
Use the same clock for the two clocks	The output is the old data from the address location.
Use different clocks	The output is unknown during the mixed-port read-during-write operation. This unknown value may be the old or new data at the address location, depending on whether the read happens before or after the write.

3.3. Consider Power-Up State and Memory Initialization

Consider the power-up state of the different types of memory blocks if you are designing logic that evaluates the initial power-up values.

Table 11. Initial Power-Up Values of Embedded Memory Blocks

Memory Type	Output Registers	Power Up Value
M9K	Used	Zero (cleared)
	Bypassed	Zero (cleared)

By default, the Intel Quartus Prime software initializes the RAM cells to zero unless you specify a **.mif**.

All memory blocks support initialization with a **.mif**. You can create **.mif** files in the Intel Quartus Prime software and specify their use with the RAM IP when you instantiate a memory in your design. Even if a memory is preinitialized (for example, using a **.mif**), it still powers up with its output cleared. Only the subsequent read after power up outputs the preinitialized values.

Only the following Intel MAX 10 configuration modes support memory initialization:

- Single Compressed Image with Memory Initialization
- Single Uncompressed Image with Memory Initialization

Note: The memory initialization feature is supported in Intel MAX 10 Flash and Analog feature options only.

Related Information

- [Selecting Internal Configuration Modes](#)
Provides more information about selecting Intel MAX 10 internal configuration modes.
- [Intel MAX 10 Device Feature Options](#)
Provides information on devices that support memory initialization.

3.4. Control Clocking to Reduce Power Consumption

Reduce AC power consumption in your design by controlling the clocking of each memory block:

- Use the read-enable signal to ensure that read operations occur only when necessary. If your design does not require read-during-write, you can reduce your power consumption by deasserting the read-enable signal during write operations, or during the period when no memory operations occur.
- Use the Intel Quartus Prime software to automatically place any unused memory blocks in low-power mode to reduce static power.
- Create independent clock enable for different input and output registers to control the shut down of a particular register for power saving purposes. From the parameter editor, click **More Options** (beside the clock enable option) to set the available independent clock enable that you prefer.

Related Information

[Intel MAX 10 Embedded Memory Clock Modes](#) on page 12

3.5. Selecting Read-During-Write Output Choices

- Single-port RAM supports only same-port read-during-write. The clock mode must be either single clock mode or input/output clock mode.
- Simple dual-port RAM supports only mixed-port read-during-write. The clock mode must be either single clock mode, or input/output clock mode.
- True dual-port RAM supports same port read-during-write and mixed-port read-during-write:
 - For same port read-during-write, the clock mode must be either single clock mode, input/output clock mode, or independent clock mode.
 - For mixed port read-during-write, the clock mode must be either single clock mode, or input/output clock mode.

Note: If you are not concerned about the output when read-during-write occurs and want to improve performance, select **Don't Care**. Selecting **Don't Care** increases the flexibility in the type of memory block being used if you do not assign block type when you instantiate the memory block.

Table 12. Output Choices for the Same-Port and Mixed-Port Read-During-Write

Memory Block	Single-Port RAM	Simple Dual-Port RAM	True Dual-Port RAM	
	Same-Port Read-During-Write	Mixed-Port Read-During-Write	Same-Port Read-During-Write	Mixed-Port Read-During-Write
M9K	<ul style="list-style-type: none"> • Don't Care • New Data • Old Data 	<ul style="list-style-type: none"> • Old Data • Don't Care 	<ul style="list-style-type: none"> • New Data • Old Data 	<ul style="list-style-type: none"> • Old Data • Don't Care

4. RAM: 1-Port IP Core References

The RAM: 1-Port IP core implements the single-port RAM memory mode.

Figure 11. RAM: 1-Port IP Core Signals with the Single Clock Option Enabled

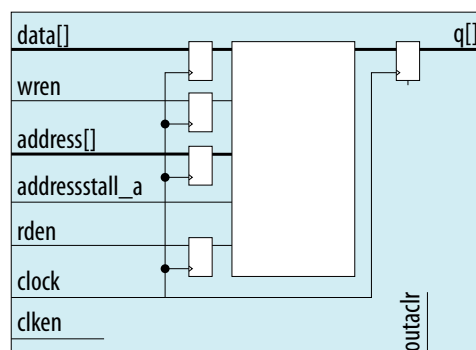
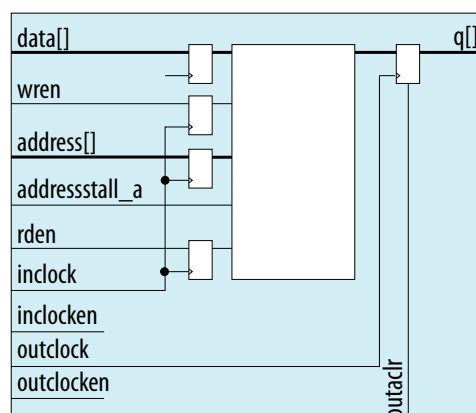


Figure 12. RAM: 1-Port IP Core Signals with the Dual Clock Option Enabled



4.1. RAM: 1-Port IP Core Signals For Intel MAX 10 Devices

Table 13. RAM: 1-Port IP Core Input Signals

Signal	Required	Description
data	Yes	Data input to the memory. The data port is required and the width must be equal to the width of the q port.
address	Yes	Address input to the memory.
wren	Yes	Write enable input for the wraddress port.
<i>continued...</i>		

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.

Signal	Required	Description
addressstall_a	Optional	Address clock enable input to hold the previous address of address_a port for as long as the addressstall_a port is high.
clock	Yes	<p>The following list describes which of your memory clock must be connected to the clock port, and port synchronization in different clocking modes:</p> <ul style="list-style-type: none"> Single clock—Connect your single source clock to clock port. All registered ports are synchronized by the same source clock. Read/Write—Connect your write clock to clock port. All registered ports related to write operation, such as data_a port, address_a port, wren_a port, and byteena_a port are synchronized by the write clock. Input/Output—Connect your input clock to clock port. All registered input ports are synchronized by the input clock. Independent clock—Connect your port A clock to clock port. All registered input and output ports of port A are synchronized by the port A clock.
clkena	Optional	Clock enable input for clock port.
rden	Optional	Read enable input for rdaddress port.
aclr	Optional	Asynchronously clear the registered input and output ports. The asynchronous clear effect on the registered ports can be controlled through their corresponding asynchronous clear parameter, such as indata_aclr, wraddress_aclr, and so on.
inclock	Optional	<p>The following list describes which of your memory clock must be connected to the inclock port, and port synchronization in different clock modes:</p> <ul style="list-style-type: none"> Single clock—Connect your single source clock to inclock port and outclock port. All registered ports are synchronized by the same source clock. Read/Write—Connect your write clock to inclock port. All registered ports related to write operation, such as data port, wraddress port, wren port, and byteena port are synchronized by the write clock. Input/Output—Connect your input clock to inclock port. All registered input ports are synchronized by the input clock.
inclocken	Optional	Clock enable input for inclock port.
outclock	Optional	<p>The following list describes which of your memory clock must be connected to the outclock port, and port synchronization in different clock modes:</p> <ul style="list-style-type: none"> Single clock—Connect your single source clock to inclock port and outclock port. All registered ports are synchronized by the same source clock. Read/Write—Connect your read clock to outclock port. All registered ports related to read operation, such as rdaddress port, rdren port, and q port are synchronized by the read clock. Input/Output—Connect your output clock to outclock port. The registered q port is synchronized by the output clock.
outclocken	Optional	Clock enable input for outclock port.

Table 14. RAM: 1-Port IP Core Output Signals

Signal	Required	Description
q	Yes	Data output from the memory. The q port must be equal in width to the data port.

4.2. RAM: 1-Port IP Core Parameters For Intel MAX 10 Devices

Table 15. RAM: 1-Port IP Core Parameters for Intel MAX 10 Devices

This table lists the IP core parameters applicable to Intel MAX 10 devices.

Parameter		Values	Description
Parameter Settings: Widths/Blk Type/Ciks			
How wide should the 'q' output bus be?		1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 36, 40, 48, 64, 72, 108, 128, 144, and 256.	Specifies the width of the 'q' output bus in bits.
How many <X>-bit words of memory?		32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, and 65536.	Specifies the number of <X>-bit words.
What should the memory block type be?			
Auto		On/Off	Specifies the memory block type. The types of memory block that are available for selection depends on your target device.
M9K		On/Off	
LC		On/Off	
Options	Use default logic cell style	On/Off	Specifies the logic cell implementation options. This option is enabled only when you choose LCs memory type.
	Use Stratix M512 emulation logic cell style	On/Off	
Set the maximum block depth to		Auto, 32, 64, 128, 256, 512, 1024, 2048, 4096, and 8192	Specifies the maximum block depth in words. This option is disabled when you choose LCs memory type.
What clocking method would you like to use?			
Single clock		On/Off	A single clock and a clock enable controls all registers of the memory block. This option is disabled when you choose LCs memory type.
Dual clock: use separate 'input' and 'output' clocks		On/Off	An input and an output clock controls all registers related to the data input and output to/from the memory block including data, address, byte enables, read enables, and write enables. This option is automatically enabled when you choose LCs memory type.
Parameter Settings: Regs/Cikens/Byte Enable/Aclrs			
Which ports should be registered?			
'data' and 'wren' input ports		—	This option is automatically enabled. Specifies whether to register the data and wren input ports.
continued...			

Parameter		Values	Description
'address' input port		—	This option is automatically enabled. Specifies whether to register the address input ports.
'q' output port		On/Off	Specifies whether to register the q output port.
Create one clock enable signal for each clock signal.		On/Off	Specifies whether to turn on the option to create one clock enable signal for each clock signal.
More Options	Use clock enable for port A input registers	On/Off	Specify whether to use clock enable for port A input registers.
	Use clock enable for port A output registers	On/Off	Specify whether to use clock enable for port A output registers.
	Create an 'addressstall_a' input port	On/Off	Specifies whether to create clock enables for address registers. You can create these ports to act as an extra active low clock enable input for the address registers.
Create an 'aclr' asynchronous clear for the registered ports.		On/Off	Specifies whether to create an asynchronous clear port for the registered ports.
More Options	'q' port	On/Off	Specifies whether the q port is cleared by the aclr port.
Create a 'rden' read enable signal		On/Off	Specifies whether to create a rden read enable signal.
Parameter Settings: Read During Write Option			
Single Port Read During Write Option			
What should the q output be when reading from a memory location being written to?		<ul style="list-style-type: none"> Don't Care New Data Old Data 	<p>Specifies the output behavior when read-during-write occurs.</p> <ul style="list-style-type: none"> Don't Care—The RAM outputs "don't care" or "unknown" values for read-during-write operation. New Data—New data is available on the rising edge of the same clock cycle on which it was written. Old Data— The RAM outputs reflect the old data at that address before the write operation proceeds.
Get x's for write masked bytes instead of old data when byte enable is used		On/Off	Turn on this option to obtain 'X' on the masked byte.
Parameter Settings: Mem Init			
Do you want to specify the initial content of the memory?			
No, leave it blank		On/Off	Specifies the initial content of the memory. Initialize the memory to zero.
Initialize memory content data to XX..X on power-up in simulation		On/Off	Specifies the initial content of the memory. Initialize the memory to "Don't Care".
Yes, use this file for the memory content data		On/Off	Allows you to specify a memory initialization file (.mif) or a hexadecimal (Intel-format) file (.hex).
continued...			

Parameter	Values	Description
		<i>Note:</i> The configuration scheme of your device is Internal Configuration. In order to use memory initialization, you must select a single image configuration mode with memory initialization, for example the Single Compressed Image with Memory Initialization option. You can set the configuration mode on the Configuration page of the Device and Pin Options dialog box.
Allow In-System Memory Content Editor to capture and update content independently of the system clock	On/Off	Specifies whether to allow In-System Memory Content Editor to capture and update content independently of the system clock.
The 'Instance ID' of this RAM is	—	Specifies the RAM ID.

5. RAM: 2-PORT IP Core References

The RAM: 2-PORT IP core implements the simple dual-port RAM and true dual-port RAM memory modes.

Figure 13. RAM: 2-Port IP Core Signals With the One Read Port and One Write Port, and Single Clock Options Enabled

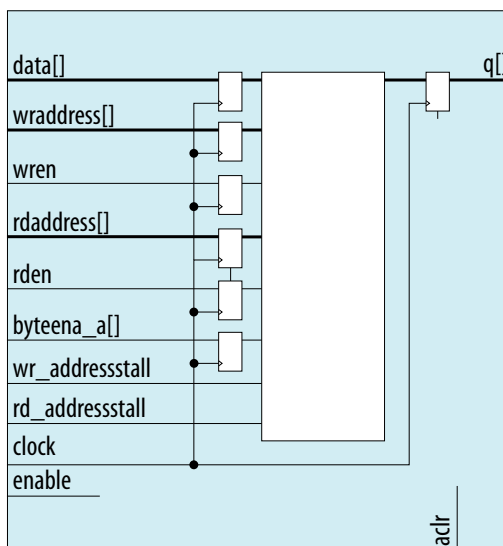


Figure 14. RAM: 2-Port IP Core Signals with the One Read Port and One Write Port, and Dual Clock: Use Separate 'Read' and 'Write' Clocks Options Enabled

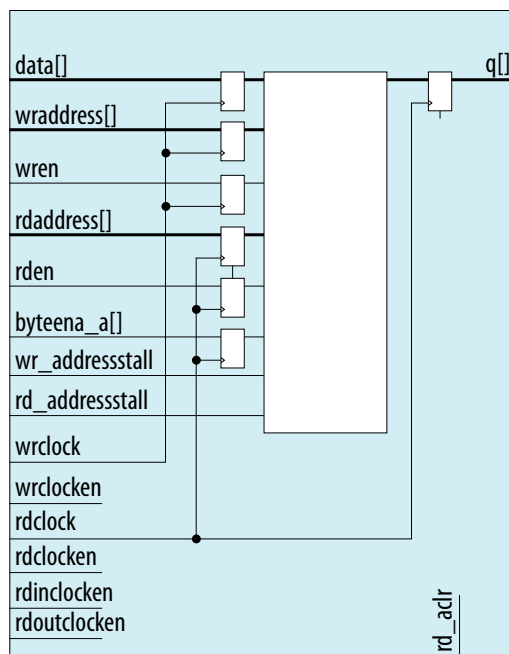


Figure 15. RAM: 2-Port IP Core Signals with the One Read Port and One Write Port, and Dual Clock: Use Separate 'Input' and 'Output' Clocks Options Enabled

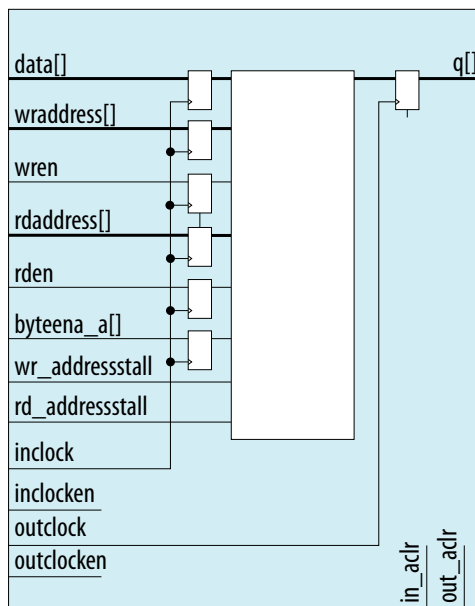


Figure 16. RAM: 2-Port IP Core Signals with the Two Read/Write Ports and Single Clock Options Enabled

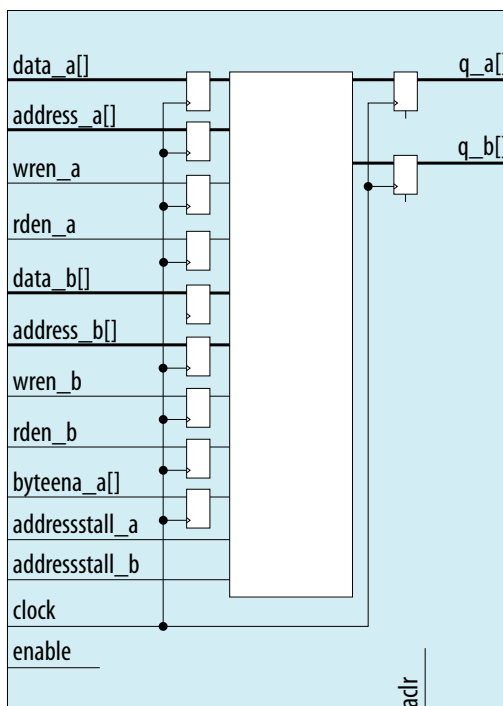


Figure 17. RAM: 2-Port IP Core Signals with the Two Read/Write Ports and Dual Clock: Use Separate 'Input' and 'Output' Clocks Options Enabled

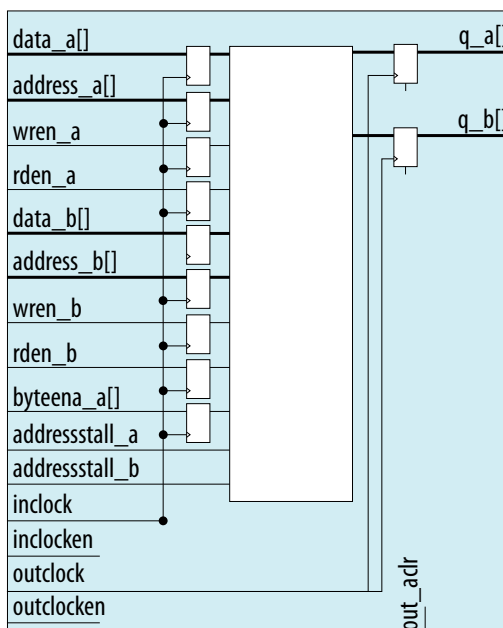
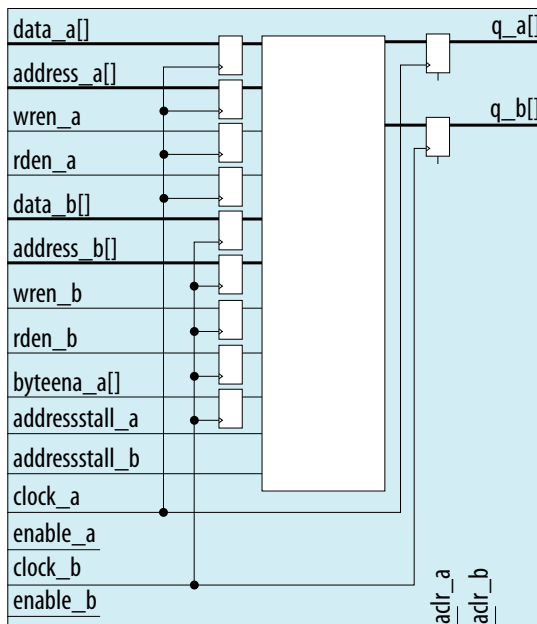


Figure 18. RAM: 2-Port IP Core Signals with the Two Read/Write Ports and Dual Clock: Use Separate for A and B Ports Options Enabled



5.1. RAM: 2-Ports IP Core Signals (Simple Dual-Port RAM) For Intel MAX 10 Devices

Table 16. RAM: 2-Ports IP Core Input Signals (Simple Dual-Port RAM)

Signal	Required	Description
data	Yes	Data input to the memory. The data port is required and the width must be equal to the width of the q port.
wraddress	Yes	Write address input to the memory. The wraddress port is required and must be equal to the width of the raddress port.
wren	Yes	Write enable input for wraddress port. The wren port is required.
rdaddress	Yes	Read address input to the memory. The rdaddress port is required and must be equal to the width of wraddress port.
clock	Yes	The following list describes which of your memory clock must be connected to the clock port, and port synchronization in different clock modes: <ul style="list-style-type: none"> Single clock—Connect your single source clock to clock port. All registered ports are synchronized by the same source clock. Read/Write—Connect your write clock to clock port. All registered ports related to write operation, such as data_a port, address_a port, wren_a port, and byteena_a port are synchronized by the write clock. Input/Output—Connect your input clock to clock port. All registered input ports are synchronized by the input clock. Independent clock—Connect your port A clock to clock port. All registered input and output ports of port A are synchronized by the port A clock.
inclock	Yes	The following list describes which of your memory clock must be connected to the inclock port, and port synchronization in different clock modes:

continued...

Signal	Required	Description
		<ul style="list-style-type: none"> Single clock—Connect your single source clock to <code>inclock</code> port and <code>outclock</code> port. All registered ports are synchronized by the same source clock. Read/Write—Connect your write clock to <code>inclock</code> port. All registered ports related to write operation, such as <code>data</code> port, <code>wraddress</code> port, <code>wren</code> port, and <code>byteena</code> port are synchronized by the write clock. Input/Output—Connect your input clock to <code>inclock</code> port. All registered input ports are synchronized by the input clock.
<code>outclock</code>	Yes	<p>The following list describes which of your memory clock must be connected to the <code>outclock</code> port, and port synchronization in different clock modes:</p> <ul style="list-style-type: none"> Single clock—Connect your single source clock to <code>inclock</code> port and <code>outclock</code> port. All registered ports are synchronized by the same source clock. Read/Write—Connect your read clock to <code>outclock</code> port. All registered ports related to read operation, such as <code>rdaddress</code> port, <code>rdren</code> port, and <code>q</code> port are synchronized by the read clock. Input/Output—Connect your output clock to <code>outclock</code> port. The registered <code>q</code> port is synchronized by the output clock.
<code>rdren</code>	Optional	Read enable input for <code>rdaddress</code> port. The <code>rdren</code> port is supported when the <code>use_eab</code> parameter is set to <code>OFF</code> . Instantiate the IP core if you want to use read enable feature with other memory blocks.
<code>byteena_a</code>	Optional	<p>Byte enable input to mask the <code>data_a</code> port so that only specific bytes, nibbles, or bits of the data are written. The <code>byteena_a</code> port is not supported in the following conditions:</p> <ul style="list-style-type: none"> If the <code>implement_in_les</code> parameter is set to <code>ON</code>. If the <code>operation_mode</code> parameter is set to <code>ROM</code>.
<code>outclocken</code>	Optional	Clock enable input for <code>outclock</code> port.
<code>inclocken</code>	Optional	Clock enable input for <code>inclock</code> port.

Table 17. RAM: 2-Ports IP Core Output Signals (Simple Dual-Port RAM)

Signal	Required	Description
<code>q</code>	Yes	Data output from the memory. The <code>q</code> port is required, and must be equal to the width <code>data</code> port.

5.2. RAM: 2-Port IP Core Signals (True Dual-Port RAM) for Intel MAX 10 Devices

Table 18. RAM: 2-Port IP Core Input Signals (True Dual-Port RAM)

Signal	Required	Description
<code>data_a</code>	Optional	<p>Data input to port A of the memory. The <code>data_a</code> port is required if the <code>operation_mode</code> parameter is set to any of the following values:</p> <ul style="list-style-type: none"> <code>SINGLE_PORT</code> <code>DUAL_PORT</code> <code>BIDIR_DUAL_PORT</code>
<code>address_a</code>	Yes	Address input to port A of the memory. The <code>address_a</code> port is required for all operation modes.
<code>wren_a</code>	Optional	Write enable input for <code>address_a</code> port. The <code>wren_a</code> port is required if you set the <code>operation_mode</code> parameter to any of the following values:
continued...		

Signal	Required	Description
		<ul style="list-style-type: none"> SINGLE_PORT DUAL_PORT BIDIR_DUAL_PORT
data_b	Optional	Data input to port B of the memory. The data_b port is required if the operation_mode parameter is set to BIDIR_DUAL_PORT.
address_b	Optional	Address input to port B of the memory. The address_b port is required if the operation_mode parameter is set to the following values: <ul style="list-style-type: none"> DUAL_PORT BIDIR_DUAL_PORT
wren_b	Yes	Write enable input for address_b port. The wren_b port is required if you set the operation_mode parameter to BIDIR_DUAL_PORT.
clock	Yes	The following list describes which of your memory clock must be connected to the clock port, and port synchronization in different clock modes: <ul style="list-style-type: none"> Single clock—Connect your single source clock to clock port. All registered ports are synchronized by the same source clock. Read/Write—Connect your write clock to clock port. All registered ports related to write operation, such as data_a port, address_a port, wren_a port, and byteena_a port are synchronized by the write clock. Input/Output—Connect your input clock to clock port. All registered input ports are synchronized by the input clock. Independent clock—Connect your port A clock to clock port. All registered input and output ports of port A are synchronized by the port A clock.
inclock	Yes	The following list describes which of your memory clock must be connected to the inclock port, and port synchronization in different clock modes: <ul style="list-style-type: none"> Single clock—Connect your single source clock to inclock port and outclock port. All registered ports are synchronized by the same source clock. Read/Write—Connect your write clock to inclock port. All registered ports related to write operation, such as data port, wraddress port, wren port, and byteena port are synchronized by the write clock. Input/Output—Connect your input clock to inclock port. All registered input ports are synchronized by the input clock.
outclock	Yes	The following list describes which of your memory clock must be connected to the outclock port, and port synchronization in different clock modes: <ul style="list-style-type: none"> Single clock—Connect your single source clock to inclock port and outclock port. All registered ports are synchronized by the same source clock. Read/Write—Connect your read clock to outclock port. All registered ports related to read operation, such as rdaddress port, rdren port, and q port are synchronized by the read clock. Input/Output—Connect your output clock to outclock port. The registered q port is synchronized by the output clock.
rden_a	Optional	Read enable input for address_a port. The rden_a port is supported depending on your selected memory mode and memory block.
rden_b	Optional	Read enable input for address_b port. The rden_b port is supported depending on your selected memory mode and memory block.
byteena_a		Byte enable input to mask the data_a port so that only specific bytes, nibbles, or bits of the data are written. The byteena_a port is not supported in the following conditions:
continued...		

Signal	Required	Description
		<ul style="list-style-type: none"> If the <code>implement_in_les</code> parameter is set to ON. If the <code>operation_mode</code> parameter is set to ROM.
<code>addressstall_a</code>	Optional	Address clock enable input to hold the previous address of <code>address_a</code> port for as long as the <code>addressstall_a</code> port is high.
<code>addressstall_b</code>	Optional	Address clock enable input to hold the previous address of <code>address_b</code> port for as long as the <code>addressstall_b</code> port is high.

Table 19. RAM: 2-Port IP Core Output Signals (True Dual-Port RAM)

Signal	Required	Description
<code>q_a</code>	Yes	<p>Data output from Port A of the memory. The <code>q_a</code> port is required if the <code>operation_mode</code> parameter is set to any of the following values:</p> <ul style="list-style-type: none"> <code>SINGLE_PORT</code> <code>BIDIR_DUAL_PORT</code> <code>ROM</code> <p>The width of <code>q_a</code> port must be equal to the width of <code>data_a</code> port.</p>
<code>q_b</code>	Yes	<p>Data output from Port B of the memory. The <code>q_b</code> port is required if you set the <code>operation_mode</code> to the following values:</p> <ul style="list-style-type: none"> <code>DUAL_PORT</code> <code>BIDIR_DUAL_PORT</code> <p>The width of <code>q_b</code> port must be equal to the width of <code>data_b</code> port.</p>

5.3. RAM: 2-Port IP Core Parameters for Intel MAX 10 Devices

Table 20. RAM: 2-Port IP Core Parameters for Intel MAX 10 Devices

This table lists the IP core parameters applicable to Intel MAX 10 devices.

Option		Legal Values	Description
Parameter Settings: General			
How will you be using the dual port RAM?		<ul style="list-style-type: none"> With one read port and one write port With two read/write ports 	Specifies how you use the dual port RAM.
How do you want to specify the memory size?		<ul style="list-style-type: none"> As a number of words As a number of bits 	Determines whether to specify the memory size in words or bits.
Parameter Settings: Widths/ Blk Type			
How many <X>-bit words of memory?		—	Specifies the number of <X>-bit words.
Use different data widths on different ports		On/Off	Specifies whether to use different data widths on different ports.
Read/Write Ports	<p>When you select With one read port and one write port, the following options are available:</p> <ul style="list-style-type: none"> How wide should the 'data_a' input bus be? How wide should the 'q' output bus be? 	1, 2, 3, 4, 5, 6, 7, 8, 9, 16, 18, 32, 36, 64, 72, 108, 128, 144, 256, and 288	<p>Specifies the width of the input and output ports.</p> <p>The How wide should the 'q' output bus be? and the How wide should the 'q_b' output bus be? options are only available when you turn on the Use different data widths on different ports parameter.</p>

continued...

Option		Legal Values	Description
	When you select With two read/write ports , the following options are available: <ul style="list-style-type: none"> How wide should the 'q_a' output bus be? How wide should the 'q_b' output bus be? 		
What should the memory block type be?		<ul style="list-style-type: none"> Auto M9K LCs 	<p>Specifies the memory block type. The types of memory block that are available for selection depends on your target device.</p> <p>The LCs value is only available under the following conditions:</p> <ul style="list-style-type: none"> Turn on the With one read port and one write port option Turn off Use different data widths on different ports option.
Option	How should the memory be implemented?	<ul style="list-style-type: none"> Use default logic cell style Use Stratix M512 emulation logic cell style 	Specifies the logic cell implementation options. This option is enabled only when you choose LCs memory type.
Set the maximum block depth to		<ul style="list-style-type: none"> Auto 128 256 512 1024 2048 4096 8192 	Specifies the maximum block depth in words.
Parameter Settings: Clks/Rd, Byte En			
What clocking method would you like to use?		<p>When you select With one read port and one write port, the following values are available:</p> <ul style="list-style-type: none"> Single clock Dual clock: use separate 'input' and 'output' clocks Dual clock: use separate 'read' and 'write' clocks <p>When you select With two read/write ports, the following options are available:</p> <ul style="list-style-type: none"> Single clock Dual clock: use separate 'input' and 'output' clocks Dual clock: use separate clocks for A and B ports 	Specifies the clocking method to use.

continued...

Option		Legal Values	Description
			<ul style="list-style-type: none"> • Single clock—A single clock and a clock enable controls all registers of the memory block. • Dual Clock: use separate 'input' and 'output' clocks—An input clock controls all registers related to the data input to the embedded memory block including data, address, byte enables, read enables, and write enables. An output clock controls the data output registers. • Dual clock: use separate 'read' and 'write' clocks—A write clock controls the data-input, write-address, and write-enable registers while the read clock controls the data-output, read-address, and read-enable registers. • Dual clock: use separate clocks for A and B ports—Clock A controls all registers on the port A side; clock B controls all registers on the port B side. Each port also supports independent clock enables for both port A and port B registers, respectively.
Create a 'rden' read enable signal		On/Off	<ul style="list-style-type: none"> • Available when you select With one read port and one write port option. • Specifies whether to create a read enable signal.
Create a 'rden_a' and 'rden_b' read enable signal		On/Off	<ul style="list-style-type: none"> • Available when you select With two read/write ports option. • Specifies whether to create a read enable signal for Port A and B.
Byte Enable Ports	Create byte enable for port A	On/Off	Specifies whether to create a byte enable for Port A and B. Turn on these options if you want to mask the input data so that only specific bytes, nibbles, or bits of data are written.
Parameter Settings: Regs/Clkens/Aclrs			
Which ports should be registered?	When you select With one read port and one write port , the following options are available: <ul style="list-style-type: none"> • Write input ports 'data_a', 'waddress_a', and 'wren_a' • Read input ports 'rdaddress' and 'rden' • Read output port(s) 'q_a' and 'q_b' 	On/Off	Specifies whether to register the read or write input and output ports.
			<i>continued...</i>

Option		Legal Values	Description
	<p>When you select With two read/write ports, the following options are available:</p> <ul style="list-style-type: none"> Write input ports 'data_a', 'waddress_a', and 'wren_a' write input ports Read output port(s) 'q_a' and 'q_b' 		
More Option	<p>When you select With one read port and one write port, the following options are available:</p> <ul style="list-style-type: none"> 'q_b' port <p>When you select With two read/write ports, the following options are available:</p> <ul style="list-style-type: none"> 'q_a' port 'q_b' port 	On/Off	The read and write input ports are turned on by default. You only need to specify whether to register the Q output ports.
Create one clock enable signal for each clock signal.		On/Off	Specifies whether to turn on the option to create one clock enable signal for each clock signal.
More Option	<p>When you select With one read port and one write port, the following option is available:</p> <ul style="list-style-type: none"> Clock enable options <ul style="list-style-type: none"> Clock enable options: Use clock enable for write input registers Address options <ul style="list-style-type: none"> Create an 'wr_addressstall' input port. Create an 'rd_addressstall' input port. <p>When you select With two read /write ports, the following options are available:</p> <ul style="list-style-type: none"> Clock enable options <ul style="list-style-type: none"> Use clock enable for port A input registers Use clock enable for port A output registers 	On/Off	<ul style="list-style-type: none"> Clock enable options—Clock enable for port B input and output registers are turned on by default. You only need to specify whether to use clock enable for port A input and output registers. Address options—Specifies whether to create clock enables for address registers. You can create these ports to act as an extra active low clock enable input for the address registers.

continued...

Option		Legal Values	Description
	<ul style="list-style-type: none"> Address options <ul style="list-style-type: none"> Create an 'addressstall_a' input port. Create an 'addressstall_b' input port. 		
Create an 'aclr' asynchronous clear for the registered ports.		On/Off	Specifies whether to create an asynchronous clear port for the registered ports.
More Option	<p>When you select With one read port and one write port, the following options are available:</p> <ul style="list-style-type: none"> 'rdaddress' port 'q_b' port <p>When you select With two read /write ports, the following options are available:</p> <ul style="list-style-type: none"> 'q_a' port 'q_b' port 	On/Off	Specifies whether the raddress, q_a, and q_b ports are cleared by the aclr port.
Parameter Settings: Output 1			
Mixed Port Read-During-Write for Single Input Clock RAM	<p>When you select With one read port and one write port, the following option is available:</p> <ul style="list-style-type: none"> How should the q output behave when reading a memory location that is being written from the other port? <p>When you select With two read /write ports, the following option is available:</p> <ul style="list-style-type: none"> How should the q_a and q_b outputs behave when reading a memory location that is being written from the other port? 	<ul style="list-style-type: none"> Old memory contents appear I do not care (the outputs will be undefined) 	<p>Specifies the output behavior when read-during-write occurs.</p> <ul style="list-style-type: none"> Old memory contents appear— The RAM outputs reflect the old data at that address before the write operation proceeds. I do not care—This option functions differently when you turn it on depending on the following memory block type you select: <ul style="list-style-type: none"> When you set the memory block type to Auto or M9K, the RAM outputs 'don't care' or "unknown" values for read-during-write operation without analyzing the timing path.
Do not analyze the timing between write and read operation. Metastability issues are prevented by never writing and reading at the same address at the same time.		On/Off	This option is automatically turned on when you turn on the I do not care (The outputs will be undefined) option. This option enables the RAM to output 'don't care' or 'unknown' values for read-during-write operation without analyzing the timing path.
Parameter Settings: Output 2 (This tab is only available when you select two read/write ports)			
Port A Read-During-Write Option	What should the 'q_a' output be when reading from a memory location being written to?	<ul style="list-style-type: none"> New data Old Data 	Specifies the output behavior when read-during-write occurs.
continued...			

Option		Legal Values	Description
Port B Read-During-Write Option	What should the 'q_b' output be when reading from a memory location being written to?		<ul style="list-style-type: none"> New Data—New data is available on the rising edge of the same clock cycle on which it was written. Old Data—The RAM outputs reflect the old data at that address before the write operation proceeds.
Get x's for write masked bytes instead of old data when byte enable is used		On/Off	This option is automatically turned on when you select the New Data value. This option obtains 'X' on the masked byte.
Parameter Settings: Mem Init			
Do you want to specify the initial content of the memory?		<ul style="list-style-type: none"> No, leave it blank Yes, use this file for the memory content data 	<p>Specifies the initial content of the memory.</p> <ul style="list-style-type: none"> To initialize the memory to zero, select No, leave it blank. To use a Memory Initialization File (.mif) or a Hexadecimal (Intel-format) File (.hex), select Yes, use this file for the memory content data. <p><i>Note:</i> The configuration scheme of your device is Internal Configuration. In order to use memory initialization, you must select a single image configuration mode with memory initialization, for example the Single Compressed Image with Memory Initialization option. You can set the configuration mode on the Configuration page of the Device and Pin Options dialog box.</p>
The initial content file should conform to which port's dimension?		<ul style="list-style-type: none"> PORT_A PORT_B 	Specifies which port's dimension that the initial content file should conform to.

6. ROM: 1-PORTR IP Core References

The ROM: 1-PORTR IP core implements the single-port ROM memory mode.

Figure 19. ROM: 1-PORTR IP Core Signals with the Single Clock Option Enabled

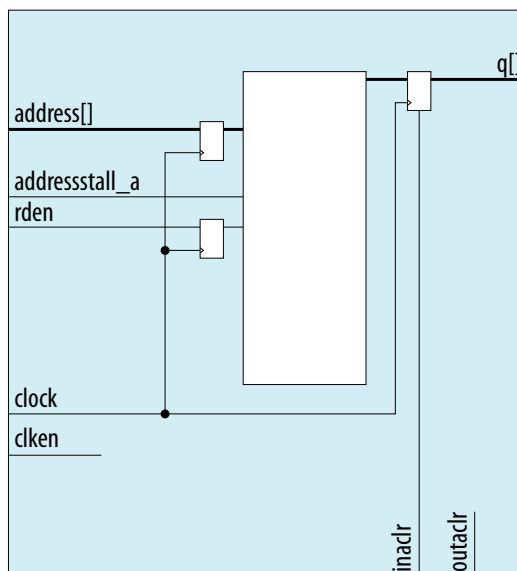
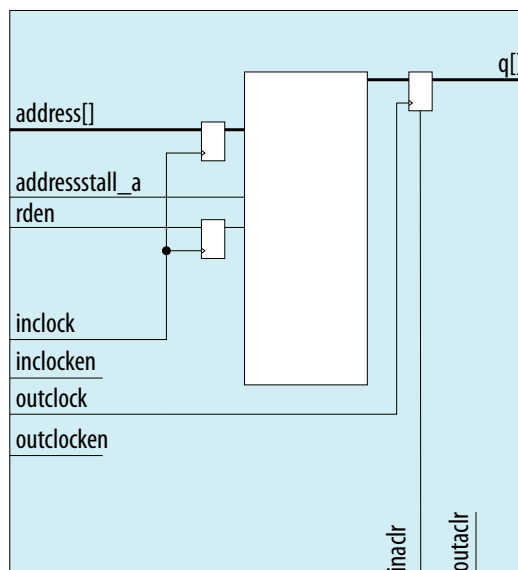


Figure 20. ROM: 1-PORT IP Core Signals with the Dual Clock: Use Separate 'Input' and 'Output' Clocks Option Enabled



6.1. ROM: 1-PORT IP Core Signals For Intel MAX 10 Devices

Table 21. ROM: 1-PORT IP Core Input Signals

Signal	Required	Description
address	Yes	Address input to the memory.
addressstall_a	Optional	Address clock enable input to hold the previous address of address_a port for as long as the addressstall_a port is high.
rden	Optional	Read enable input for rdaddress port. The rden port is supported when the use_eab parameter is set to OFF. Instantiate the IP if you want to use read enable feature with other memory blocks.
clock	Yes	The following list describes which of your memory clock must be connected to the clock port, and port synchronization in different clock modes: <ul style="list-style-type: none"> Single clock—Connect your single source clock to clock port. All registered ports are synchronized by the same source clock. Read/Write—Connect your write clock to clock port. All registered ports related to write operation, such as data_a port, address_a port, wren_a port, and byteena_a port are synchronized by the write clock. Input/Output—Connect your input clock to clock port. All registered input ports are synchronized by the input clock. Independent clock—Connect your port A clock to clock port. All registered input and output ports of port A are synchronized by the port A clock.
clken	Optional	Clock enable input for clock port.
inclock	Yes	The following list describes which of your memory clock must be connected to the inclock port, and port synchronization in different clock modes:

continued...

Signal	Required	Description
		<ul style="list-style-type: none"> Single clock—Connect your single source clock to <code>inclock</code> port and <code>outclock</code> port. All registered ports are synchronized by the same source clock. Read/Write—Connect your write clock to <code>inclock</code> port. All registered ports related to write operation, such as <code>data</code> port, <code>waddress</code> port, <code>wren</code> port, and <code>byteena</code> port are synchronized by the write clock. Input/Output—Connect your input clock to <code>inclock</code> port. All registered input ports are synchronized by the input clock.
<code>inclocken</code>	Optional	Clock enable input for <code>inclock</code> port.
<code>outclock</code>	Yes	<p>The following list describes which of your memory clock must be connected to the <code>outclock</code> port, and port synchronization in different clock modes:</p> <ul style="list-style-type: none"> Single clock—Connect your single source clock to <code>inclock</code> port and <code>outclock</code> port. All registered ports are synchronized by the same source clock. Read/Write—Connect your read clock to <code>outclock</code> port. All registered ports related to read operation, such as <code>rdaddress</code> port, <code>rdren</code> port, and <code>q</code> port are synchronized by the read clock. Input/Output—Connect your output clock to <code>outclock</code> port. The registered <code>q</code> port is synchronized by the output clock.
<code>outclocken</code>	Optional	Clock enable input for <code>outclock</code> port.

Table 22. ROM: 1-PORT IP Core Output Signals

Signal	Required	Description
<code>q</code>	Yes	Data output from the memory. The <code>q</code> port is required, and must be equal to the width data port.

6.2. ROM: 1-PORT IP Core Parameters for Intel MAX 10 Devices

Table 23. ROM: 1-Port IP Core Parameters for Intel MAX 10 Devices

This table lists the IP core parameters applicable to Intel MAX 10 devices.

Option	Legal Values	Description
Parameter Settings: General		
How wide should the 'q' output bus be?	1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 36, 40, 48, 64, 72, 108, 128, 144, and 256.	Specifies the width of the 'q' output bus in bits.
How many <X>-bit words of memory?	32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, and 65536.	Specifies the number of <X>-bit words.
What should the memory block type be?	<ul style="list-style-type: none"> Auto M9K 	Specifies the memory block type. The types of memory block that are available for selection depends on your target device.
Set the maximum block depth to	<ul style="list-style-type: none"> Auto 32 64 128 	Specifies the maximum block depth in words.
<i>continued...</i>		

Option		Legal Values	Description
		<ul style="list-style-type: none"> 256 512 1024 2048 4096 8192 	
What clocking method would you like to use?		<ul style="list-style-type: none"> Single clock Dual clock: use separate 'input' and 'output' clocks 	<p>Specifies the clocking method to use.</p> <ul style="list-style-type: none"> Single clock—A single clock and a clock enable controls all registers of the memory block. Dual clock: use separate 'input' and 'output' clocks—An input and an output clock controls all registers related to the data input and output to/from the memory block including data, address, byte enables, read enables, and write enables.
Parameter Settings: Regs/Clkens/Aclrs			
Which ports should be registered?	<ul style="list-style-type: none"> 'address' input port 'q' output port 	On/Off	Specifies whether to register the 'address' input port and 'q' output port.
Create one clock enable signal for each clock signal.		On/Off	Specifies whether to turn on the option to create one clock enable signal for each clock signal.
More Options	<ul style="list-style-type: none"> Clock enable options <ul style="list-style-type: none"> Use clock enable for port A input registers Use clock enable for port A output registers Address options <ul style="list-style-type: none"> Create an 'addressstall_a' input port 	On/Off	<ul style="list-style-type: none"> Clock enable options—Clock enable for port B input and output registers are turned on by default. You only need to specify whether to use clock enable for port A input and output registers. Address options—Specifies whether to create clock enables for address registers. You can create these ports to act as an extra active low clock enable input for the address registers.
Create an 'aclr' asynchronous clear for the registered ports.		On/Off	Specifies whether to create an asynchronous clear port for the registered ports.
More Options	<ul style="list-style-type: none"> 'address' port 'q' port 	On/Off	Specifies whether the address and q ports are cleared by the aclr port.
Create a 'rden' read enable signal		On/Off	Specifies whether to create a rden read enable signal.
Parameter Settings: Mem Init			
Do you want to specify the initial content of the memory?		Yes, use this file for the memory content data.	<p>Specifies the initial content of the memory. In ROM mode you must specify a Memory Initialization File (.mif) or a Hexadecimal (Intel-format) File (.hex). The configuration scheme of your</p> <p>continued...</p>

Option	Legal Values	Description
		device is Internal Configuration. In order to use memory initialization, you must select a single image configuration mode with memory initialization, for example the Single Compressed Image with Memory Initialization option. You can set the configuration mode on the Configuration page of the Device and Pin Options dialog box.
Allow In-System Memory Content Editor to capture and update content independently of the system clock	On/Off	Specifies whether to allow In-System Memory Content Editor to capture and update content independently of the system clock.
The 'Instance ID' of this RAM is	—	Specifies the RAM ID.

7. ROM: 2-PORT IP Core References

This IP core implements the dual-port ROM memory mode. The dual-port ROM has almost similar functional ports as single-port ROM. The difference is dual-port ROM has an additional address port for read operation.

Figure 21. ROM: 2-PORT IP Core Signals with the Single Clock Option Enabled

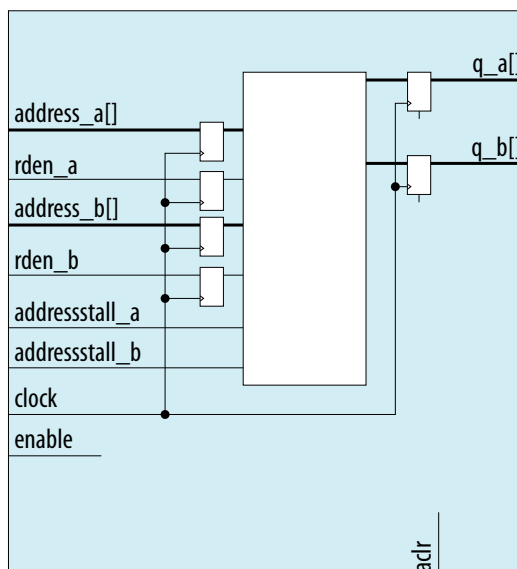


Figure 22. ROM: 2-PORT IP Core Signals with the Dual Clock: Use Separate 'Input' and 'Output' Clocks Option Enabled

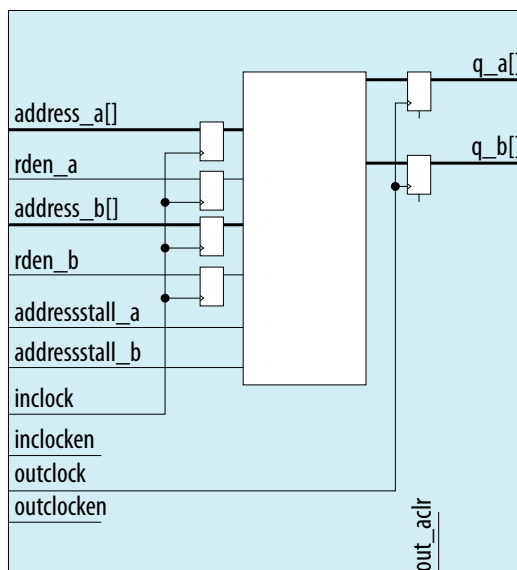
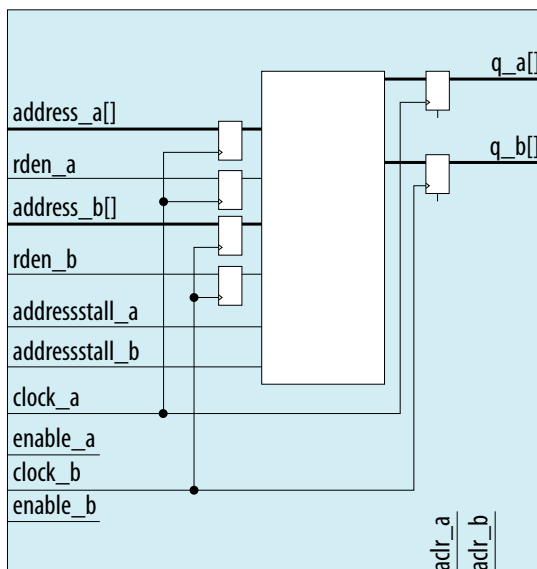


Figure 23. ROM: 2-PORT IP Core Signals with the Dual Clock: Use Separate Clocks for A and B Ports Option Enabled



7.1. ROM: 2-PORT IP Core Signals for Intel MAX 10 Devices

Table 24. ROM: 2-PORT IP Core Input Signals

Signal	Required	Description
address_a	Yes	Address input to port A of the memory. The address_a port is required for all operation modes.
rden_a	Optional	Read enable input for address_a port. The rden_a port is supported depending on your selected memory mode and memory block.
address_b	Optional	Address input to port B of the memory. The address_b port is required if the operation_mode parameter is set to the following values: <ul style="list-style-type: none"> DUAL_PORT BIDIR_DUAL_PORT
rden_b	Optional	Read enable input for address_b port. The rden_b port is supported depending on your selected memory mode and memory block.
clock	Yes	The following list describes which of your memory clock must be connected to the clock port, and port synchronization in different clock modes: <ul style="list-style-type: none"> Single clock—Connect your single source clock to clock port. All registered ports are synchronized by the same source clock. Read/Write—Connect your write clock to clock port. All registered ports related to write operation, such as data_a port, address_a port, wren_a port, and byteena_a port are synchronized by the write clock. Input/Output—Connect your input clock to clock port. All registered input ports are synchronized by the input clock. Independent clock—Connect your port A clock to clock port. All registered input and output ports of port A are synchronized by the port A clock.
addressstall_a	Optional	Address clock enable input to hold the previous address of address_a port for as long as the addressstall_a port is high.
addressstall_b	Optional	Address clock enable input to hold the previous address of address_b port for as long as the addressstall_b port is high.
inclock	Yes	The following list describes which of your memory clock must be connected to the inclock port, and port synchronization in different clock modes: <ul style="list-style-type: none"> Single clock—Connect your single source clock to inclock port and outclock port. All registered ports are synchronized by the same source clock. Read/Write—Connect your write clock to inclock port. The write clock synchronizes all registered ports related to write operation, such as data port, wraddress port, wren port, and byteena port. Input/Output—Connect your input clock to inclock port. The input clock synchronizes all registered input ports.
outclock	Yes	The following list describes which of your memory clock must be connected to the outclock port, and port synchronization in different clock modes: <ul style="list-style-type: none"> Single clock—Connect your single source clock to inclock port and outclock port. All registered ports are synchronized by the same source clock. Read/Write—Connect your read clock to outclock port. The read clock synchronizes all registered ports related to read operation, such as rdaddress port, rdren port, and q port. Input/Output—Connect your output clock to outclock port. The output clock synchronizes the registered q port.
continued...		

Signal	Required	Description
inclocken	Optional	Clock enable input for inclock port.
outclocken	Optional	Clock enable input for outclock port.
aclr	Optional	Asynchronously clear the registered input and output ports. The asynchronous clear effect on the registered ports can be controlled through their corresponding asynchronous clear parameter, such as <code>indata_aclr</code> and <code>wraddress_aclr</code> .

Table 25. ROM: 2-PORT IP Core Output Signals

Signal	Required	Description
q_a	Yes	Data output from port A of the memory. The <code>q_a</code> port is required if you set the <code>operation_mode</code> parameter to any of the following values: <ul style="list-style-type: none"> <code>SINGLE_PORT</code> <code>BIDIR_DUAL_PORT</code> <code>ROM</code> The width of the <code>q_a</code> port must be equal to the width of the <code>data_a</code> port.
q_b	Yes	Data output from port B of the memory. The <code>q_b</code> port is required if you set the <code>operation_mode</code> parameter to the following values: <ul style="list-style-type: none"> <code>DUAL_PORT</code> <code>BIDIR_DUAL_PORT</code> The width of <code>q_b</code> port must be equal to the width of <code>data_b</code> port.

7.2. ROM: 2-Port IP Core Parameters For Intel MAX 10 Devices

Table 26. ROM:2-Port IP Core Parameters for Intel MAX 10 Devices

This table lists the IP core parameters applicable to Intel MAX 10 devices.

Option		Legal Values	Description
Parameter Settings: Widths/Blk Type			
How do you want to specify the memory size?		<ul style="list-style-type: none"> As a number of words As a number of bits 	Determines whether to specify the memory size in words or bits.
How many <X>-bit words of memory?		—	Specifies the number of <X>-bit words.
Use different data widths on different ports		On/Off	Specifies whether to use different data widths on different ports.
Read Ports	How wide should the 'q_a' output bus be?	1, 2, 3, 4, 5, 6, 7, 8, 9, 16, 18, 32, 36, 64, 72, 108, 128, 144, 256, and 288	Specifies the width of the input and output ports. The How wide should the 'q_b' output bus be? option is only available when you turn on the Use different data widths on different ports parameter.
	How wide should the 'q_b' output bus be?		
What should the memory block type be?		Auto, M9K	Specifies the memory block type. The types of memory block that are available for selection depends on your target device.
Set the maximum block depth to		Auto, 128, 256, 512, 1024, 2048, 4096, 8192	Specifies the maximum block depth in words.
Parameter Settings: Clks/Rd, Byte En			
continued...			

Option		Legal Values	Description
What clocking method would you like to use?		<ul style="list-style-type: none"> Single clock Dual clock: use separate 'input' and 'output' clocks Dual clock: use separate clocks for A and B ports 	<p>Specifies the clocking method to use.</p> <ul style="list-style-type: none"> Single clock—A single clock and a clock enable controls all registers of the memory block. Dual Clock: use separate 'input' and 'output' clocks—An input and an output clock controls all registers related to the data input and output to/from the memory block including data, address, byte enables, read enables, and write enables. Dual clock: use separate clocks for A and B ports—Clock A controls all registers on the port A side; clock B controls all registers on the port B side. Each port also supports independent clock enables for both port A and port B registers, respectively.
Create a 'rden_a' and 'rden_b' read enable signal		On/Off	Specifies whether to create read enable signals.
Parameter Settings: Regs/Clkens/Aclrs			
Which ports should be registered?	<ul style="list-style-type: none"> Write input ports Read output port(s) 	On/Off	Specifies whether to register the write input ports and/or read output ports.
More Options	<ul style="list-style-type: none"> Input ports <ul style="list-style-type: none"> 'address_a' port 'address_b' port Q output ports <ul style="list-style-type: none"> 'q_a' port 'q_b' port 	On/Off	The read and write input ports are turned on by default. You only need to specify whether to register the Q output ports.
Create one clock enable signal for each clock signal.		On/Off	Specifies whether to turn on the option to create one clock enable signal for each clock signal.
More Options	<ul style="list-style-type: none"> Clock enable options <ul style="list-style-type: none"> Use clock enable for port A input registers Use clock enable for port A output registers Address options <ul style="list-style-type: none"> Create an 'addressstall_a' input port. Create an 'addressstall_b' input port. 	On/Off	<ul style="list-style-type: none"> Clock enable options—Clock enable for port B input and output registers are turned on by default. You only need to specify whether to use clock enable for port A input and output registers. Address options—Specifies whether to create clock enables for address registers. You can create these ports to act as an extra active low clock enable input for the address registers.
Create an 'aclr' asynchronous clear for the registered ports.		On/Off	Specifies whether to create an asynchronous clear port for the registered ports.
More Options	<ul style="list-style-type: none"> 'q_a' port 'q_b' port 	On/Off	Specifies whether the 'q_a', and 'q_b' ports are cleared by the aclr port.
Parameter Settings: Mem Init			
Do you want to specify the initial content of the memory?		Yes, use this file for the memory content data	Specifies the initial content of the memory.
continued...			

Option	Legal Values	Description
		<ul style="list-style-type: none"> To initialize the memory to zero, select No, leave it blank. To use a Memory Initialization File (.mif) or a Hexadecimal (Intel-format) File (.hex), select Yes, use this file for the memory content data. <p><i>Note:</i> The configuration scheme of your device is Internal Configuration. In order to use memory initialization, you must select a single image configuration mode with memory initialization, for example the Single Compressed Image with Memory Initialization option. You can set the configuration mode on the Configuration page of the Device and Pin Options dialog box.</p>
The initial content file should conform to which port's dimension?	<ul style="list-style-type: none"> PORT_A PORT_B 	Specifies which port's dimension that the initial content file should conform to.

8. FIFO IP Core References

The FIFO IP core implements the FIFO mode, enabling you to use the memory blocks as FIFO buffers.

- Use the FIFO IP core in single clock FIFO (SCFIFO) and dual clock FIFO (DCFIFO) modes to implement single- and dual-clock FIFO buffers in your design.
- Dual clock FIFO buffers are useful when transferring data from one clock domain to another clock domain.
- The M9K memory blocks do not support simultaneous read and write from an empty FIFO buffer.

Figure 24. FIFO IP Core: SCFIFO Mode Signals

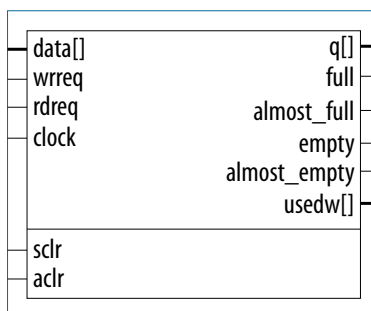
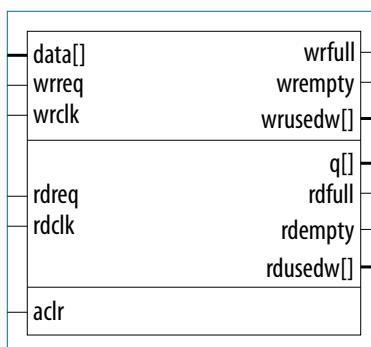


Figure 25. FIFO IP Core: DCFIFO Mode Signals



Related Information

FIFO Intel FPGA IP User Guide

Provides information on FIFO Intel® FPGA IP core through the parameterizable single-clock FIFO (SCFIFO) and dual-clock FIFO (DCFIFO) functions.

8.1. FIFO IP Core Signals for Intel MAX 10 Devices

Table 27. FIFO IP Core Input Signals

Signal	Required	Description
clock	Yes	Positive-edge-triggered clock.
wrclk	Yes	Positive-edge-triggered clock. Synchronizes the following ports: <ul style="list-style-type: none"> data wrreq wrfull wrempty wrusedw
rdclk	Yes	Positive-edge-triggered clock. Synchronizes the following ports: <ul style="list-style-type: none"> q rdreq rdfull rdempty rdusedw
data	Yes	Holds the data to be written in the FIFO IP core when the wrreq signal is asserted. If you manually instantiate the FIFO IP core, ensure that the port width is equal to the How wide should the FIFO be? parameter.
wrreq	Yes	Assert this signal to request for a write operation. Ensure that the following conditions are met: <ul style="list-style-type: none"> Do not assert the wrreq signal when the full (for the FIFO IP core in SCFIFO mode) or wrfull (for the FIFO IP core in DCFIFO mode) port is high. Enable the overflow protection circuitry or turn on the Disable overflow checking. Writing to a full FIFO will corrupt contents parameter so that the FIFO IP core can automatically disable the wrreq signal when it is full. The wrreq signal must meet the functional timing requirement based on the full or wrfull signal. Do not assert the wrreq signal during the deassertion of the aclr signal. Violating this requirement creates a race condition between the falling edge of the aclr signal and the rising edge of the write clock if the wrreq port is set to high.
rdreq	Yes	Assert this signal to request for a read operation. The rdreq signal acts differently in normal synchronous FIFO mode and show-ahead mode synchronous FIFO modes. Ensure that the following conditions are met: <ul style="list-style-type: none"> Do not assert the rdreq signal when the empty (for the FIFO IP core in SCFIFO mode) or rdempty (for the FIFO IP core in DCFIFO mode) port is high. Enable the underflow protection circuitry or turn on the Disable underflow checking. Reading from an empty FIFO will corrupt contents parameter so that the FIFO IP core can automatically disable the rdreq signal when it is empty. The rdreq signal must meet the functional timing requirement based on the empty or rdempty signal.
sclr	No	Assert this signal to clear all the output status ports, but the effect on the q output may vary for different FIFO configurations. There are no minimum number of clock cycles for aclr signals that must remain active.
aclr	No	

Table 28. FIFO IP Core Output Signals

Signal	Required	Description
q	Yes	Shows the data read from the read request operation. In SCFIFO and DCFIFO modes, the width of the q port must be equal to the width of the data port. If you manually instantiate the IPs, ensure that the port width is equal to the How wide should the FIFO be? parameter.
full	No	When asserted, the FIFO IP core is considered full. Do not perform write request operation when the FIFO IP core is full. In general, the rdfull signal is a delayed version of the wrfull signal. However, the rdfull signal functions as a combinational output instead of a derived version of the wrfull signal. Therefore, you must always refer to the wrfull port to ensure whether or not a valid write request operation can be performed, regardless of the target device.
wrfull		
rdfull		
empty	No	When asserted, the FIFO IP core is considered empty. Do not perform read request operation when the FIFO IP core is empty. In general, the wrempty signal is a delayed version of the rdempty signal. However, the wrempty signal functions as a combinational output instead of a derived version of the rdempty signal. Therefore, you must always refer to the rdempty port to ensure whether or not a valid read request operation can be performed, regardless of the target device.
wrempty		
rdempty		
almost_full	No	Asserted when the usedw signal is greater than or equal to the Almost full parameter. It is used as an early indication of the full signal.
almost_empty	No	Asserted when the usedw signal is less than the Almost empty parameter. It is used as an early indication of the empty signal.
usedw	No	Show the number of words stored in the FIFO. Ensure that the port width is equal to the usedw[] parameter if you manually instantiate the FIFO IP core in SCFIFO or DCFIFO modes.
wrusedw		
rdusedw		

8.2. FIFO IP Core Parameters for Intel MAX 10 Devices

Table 29. FIFO IP Core Parameters for Intel MAX 10 Devices

This table lists the IP core parameters applicable to Intel MAX 10 devices.

Parameter	HDL Parameter	Description
How wide should the FIFO be?	lpm_width	Specifies the width of the data and q ports for the FIFO IP core in SCFIFO mode and DCFIFO mode.
Usedw[]	lpm_widthu	Specifies the width of the usedw port for the FIFO IP core in SCFIFO mode, or the width of the rdusedw and wrusedw ports for the FIFO IP core in DCFIFO mode.
How deep should the FIFO be?	lpm_numwords	Specifies the depths of the FIFO you require. The value must be at least 4 . The value assigned must comply with the 2^{LPM_WIDTHU} equation.
Which kind of read access do you want with the rdreq signal?	lpm_showahead	Specifies whether the FIFO is in normal synchronous FIFO mode or show-ahead mode synchronous FIFO mode. For normal synchronous FIFO mode, the FIFO IP core treats the rdreq port as a normal read request that only performs read operation when the port is asserted. For show-ahead mode synchronous FIFO mode, the FIFO IP core treats the rdreq port as a read-acknowledge that automatically outputs the first word of valid data in the FIFO IP core (when the empty or rdempty port is low) without asserting the rdreq signal. Asserting the rdreq signal causes the FIFO IP core to output the next data word, if available. If you turn on this parameter, you may reduce performance.

continued...

Parameter	HDL Parameter	Description
Do you want a common clock for reading and writing the FIFO?	lpm_type	Identifies the library of parameterized modules (LPM) entity name. The values are SCFIFO and DCFIFO .
Disable overflow checking. Writing to a full FIFO will corrupt contents	overflow_checking	Specifies whether or not to enable the protection circuitry for overflow checking that disables the <code>wrreq</code> port when the FIFO IP core is full. This parameter is enabled by default.
Disable underflow checking. Reading from an empty FIFO will corrupt contents.	underflow_checking	Specifies whether or not to enable the protection circuitry for underflow checking that disables the <code>rdreq</code> port when the FIFO IP core is empty. This parameter is enabled by default. Note that reading from an empty SCFIFO mode gives unpredictable results.
Add an extra MSB to usedw⁽¹⁾	add_usedw_msb_bit	Increases the width of the <code>rdusedw</code> and <code>wrusedw</code> ports by one bit. By increasing the width, it prevents the FIFO IP core from rolling over to zero when it is full. This parameter is disabled by default.
How many sync stages?⁽¹⁾	rdsync_delaypipe	Specifies the number of synchronization stages in the cross clock domain. The value of the <code>rdsync_delaypipe</code> parameter relates the synchronization stages from the write control logic to the read control logic; the <code>wrsync_delaypipe</code> parameter relates the synchronization stages from the read control logic to the write control logic. Use these parameters to set the number of synchronization stages if the clocks are not synchronized, and set the <code>clocks_are_synchronized</code> parameter to FALSE. The actual synchronization stage implemented relates variously to the parameter value assigned and depends on the target device.
How many sync stages?⁽¹⁾	wrsync_delaypipe	Specifies the number of synchronization stages in the cross clock domain. The value of the <code>rdsync_delaypipe</code> parameter relates the synchronization stages from the write control logic to the read control logic; the <code>wrsync_delaypipe</code> parameter relates the synchronization stages from the read control logic to the write control logic. Use these parameters to set the number of synchronization stages if the clocks are not synchronized, and set the <code>clocks_are_synchronized</code> parameter to FALSE. The actual synchronization stage implemented relates variously to the parameter value assigned and depends on the target device.
Implement FIFO storage with logic cells only, even if the device contains memory blocks.	use_eab	Specifies whether or not the FIFO IP core is constructed using RAM blocks. This parameter is disabled by default. If you turn off this parameter, the FIFO IP core is implemented in logic elements, regardless of the memory block type assigned to the What should the memory block type be parameter.
Add circuit to synchronize 'aclr' input with 'wrclk'⁽¹⁾	write_aclr_sync	Specifies whether or not to add a circuit that causes the <code>aclr</code> port to be internally synchronized by the <code>wrclk</code> clock. Adding the circuit prevents the race condition between the <code>wrreq</code> and <code>aclr</code> ports that could corrupt the FIFO IP core. This parameter is disabled by default.
Add circuit to synchronize 'aclr' input with 'rdclk'	read_aclr_sync	Specifies whether or not to add a circuit that causes the <code>aclr</code> port to be internally synchronized by the <code>rdclk</code> clock. Adding the circuit prevents the race condition between the <code>rdreq</code> and <code>aclr</code> ports that could corrupt the FIFO IP core. This parameter is disabled by default.
Which type of optimization do you want?⁽¹⁾	clocks_are_synchronized	Specifies whether or not the write and read clocks are synchronized, which in turn determines the number of internal synchronization stages added for stable operation of the FIFO. The values are TRUE and FALSE. If omitted, the default value is FALSE. You must only set the parameter to TRUE if the write clock and the read clock are always synchronized and they are multiples of each
continued...		

⁽¹⁾ Applicable in DCFIFO mode only.

Parameter	HDL Parameter	Description
		other. Otherwise, set this to FALSE to avoid metastability problems. If the clocks are not synchronized, set the parameter to FALSE, and use the <code>rdsync_delaypipe</code> and <code>wrsync_delaypipe</code> parameters to determine the number of synchronization stages required.
What should the memory block type be	<code>ram_block_type</code>	Specifies the target device's memory block to be used. To get the proper implementation based on the RAM configuration that you set, allow the Intel Quartus Prime software to automatically choose the memory type by ignoring this parameter and turn on the Implement FIFO storage with logic cells only, even if the device contains memory blocks. parameter. This gives the Compiler the flexibility to place the memory function in any available memory resource based on the FIFO depth required.
Would you like to register the output to maximize the performance but use more area? ⁽²⁾	<code>add_ram_output_register</code>	Specifies whether to register the <code>q</code> output. The values are Yes (best speed) and No (smallest area) . The default value is No (smallest area) .
Becomes true when <code>usedw[]</code> is greater than or equal to: ⁽²⁾	<code>almost_full_value</code>	Sets the threshold value for the <code>almost_full</code> port. When the number of words stored in the FIFO IP core is greater than or equal to this value, the <code>almost_full</code> port is asserted.
Almost full ⁽²⁾		
Almost empty ⁽²⁾	<code>almost_empty_value</code>	Sets the threshold value for the <code>almost_empty</code> port. When the number of words stored in the FIFO IP core is less than this value, the <code>almost_empty</code> port is asserted.
Becomes true when <code>usedw[]</code> is less than: ⁽²⁾		
Currently selected device family	<code>intended_device_family</code>	Specifies the intended device that matches the device set in your Intel Quartus Prime project. Use this parameter only for functional simulation.

8.3. FIFO Functional Timing Requirements

The `wrreq` signal is ignored (when FIFO is full) if you enable the overflow protection circuitry in the FIFO Intel FPGA IP parameter editor, or set the `OVERFLOW_CHECKING` parameter to ON. The `rdreq` signal is ignored (when FIFO is empty) if you enable the underflow protection circuitry in the FIFO Intel FPGA IP core interface, or set the `UNDERFLOW_CHECKING` parameter to ON.

If the protection circuitry is not enabled, you must meet the following functional timing requirements:

Table 30. Functional Timing Requirements

DCFIFO	SCFIFO
Deassert the <code>wrreq</code> signal in the same clock cycle when the <code>wrfull</code> signal is asserted.	Deassert the <code>wrreq</code> signal in the same clock cycle when the <code>full</code> signal is asserted.
Deassert the <code>rdreq</code> signal in the same clock cycle when the <code>rdempty</code> signal is asserted. You must observe these requirements regardless of expected behavior based on <code>wrclk</code> and <code>rdclk</code> frequencies.	Deassert the <code>rdreq</code> signal in the same clock cycle when the <code>empty</code> signal is asserted.

⁽²⁾ Applicable in SCFIFO mode only.

Figure 26. Functional Timing for the wrreq Signal and the wrfull Signal

This figure shows the behavior for the wrreq and the wrfull signals.

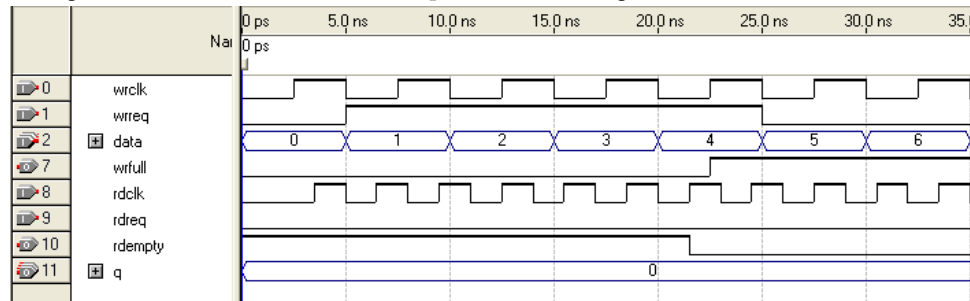
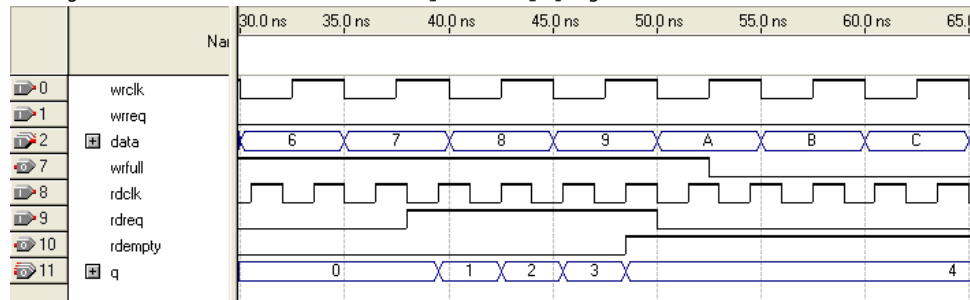


Figure 27. Functional Timing for the rdreq Signal and the rdempty Signal

This figure shows the behavior for the rdreq the rdempty signals.

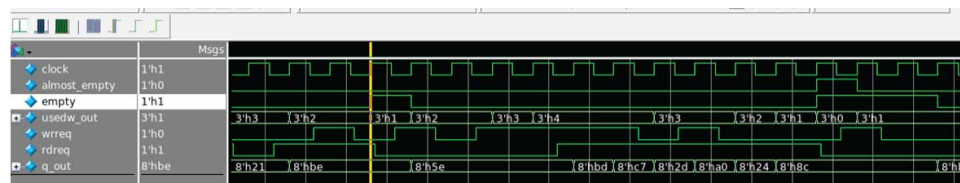


The required functional timing for the DCFIFO as described previously is also applied to the SCFIFO. The difference between the two modes is that for the SCFIFO, the wrreq signal must meet the functional timing requirement based on the full signal and the rdreq signal must meet the functional timing requirement based on the empty signal.

8.4. SCFIFO ALMOST_EMPTY Functional Timing

In SCFIFO, the almost_empty is asserted only when the usedw is less than the almost_empty_value that you set. The almost_empty signal does not consider the data readiness at the output. When the almost_empty_value is set too low, it is possible to observe that SCFIFO asserts the empty signal without asserting the almost_empty signal.

Figure 28. Example of empty Signal Assertion without Asserting almost_empty Signal



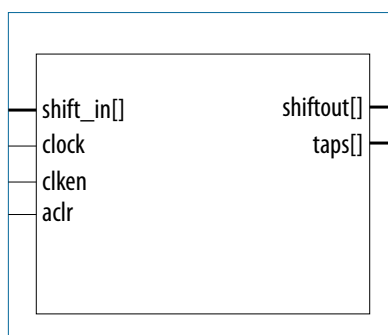
In this example, the almost_empty_value is 1 which means the almost_empty asserts when usedw is 0. There are three words in the FIFO before the read request is received. After the first read, the wrreq asserts and the rdreq signal remains high.

The `usedw` remains at 2. In the next cycle, the `wrreq` de-asserts but there is another `rdreq` going on. The `usedw` decrease to 1 and the `almost_empty` signal remains low. However, the write data has not been written into the FIFO due to the write latency. The `empty` signal asserts to indicate the FIFO is empty.

9. Shift Register (RAM-based) IP Core References

The Shift Register (RAM-based) IP core contains additional features not found in a conventional shift register. You can use the memory blocks as a shift-register block to save logic cells and routing resources. You can cascade memory blocks to implement larger shift registers.

Figure 29. Shift Register (RAM-based) IP Core Signals



9.1. Shift Register (RAM-based) IP Core Signals for Intel MAX 10 Devices

Table 31. Shift Register (RAM-based) IP Core Input Signals

Signal	Required	Description
<code>shiftin[]</code>	Yes	Data input to the shifter. Input port <code>WIDTH</code> bits wide.
<code>clock</code>	Yes	Positive-edge triggered clock.
<code>clken</code>	No	Clock enable for the <code>clock</code> port. <code>clken</code> defaults to <code>VCC</code> .
<code>aclr</code>	No	Asynchronously clears the contents of the shift register chain. The <code>shiftout</code> outputs are cleared immediately upon the assertion of the <code>aclr</code> signal.

Table 32. Shift Register (RAM-based) IP Core Output Signals

Signal	Required	Description
<code>shiftout[]</code>	Yes	Output from the end of the shift register. Output port <code>WIDTH</code> bits wide.
<code>taps[]</code>	Yes	Output from the regularly spaced taps along the shift register. Output port <code>WIDTH * NUMBER_OF_TAPS</code> wide. This port is an aggregate of all the regularly spaced taps (each <code>WIDTH</code> bits) along the shift register.

9.2. Shift Register (RAM-based) IP Core Parameters for Intel MAX 10 Devices

Table 33. Shift Register (RAM-based) IP Core Parameters for Intel MAX 10 Devices

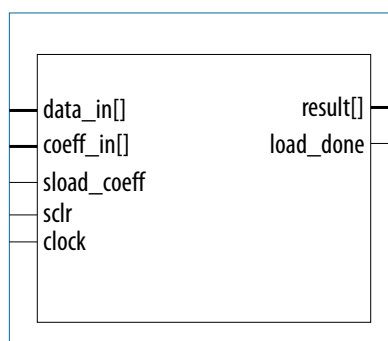
This table lists the IP core parameters applicable to Intel MAX 10 devices.

Option	Values	Description
How wide should the "shiftin" input and the "shiftout" output buses be?	1, 2, 3, 4, 5, 6, 7, 8, 12, 16, 24, 32, 48, 64, 96, 128, 192, and 256.	Specifies the width of the input pattern.
How many taps would you like?	1, 2, 3, 4, 5, 6, 7, 8, 12, 16, 24, 32, 48, 64, 96, and 128.	Specifies the number of regularly spaced taps along the shift register.
Create groups for each tap output	On/Off	Creates groups for each tap output.
How wide should the distance between taps be?	3, 4, 5, 6, 7, 8, 16, 32, 64, and 128	Specifies the distance between the regularly spaced taps in clock cycles. This number translates to the number of RAM words that will be used. The value must be at least 3 .
Create a clock enable port	On/Off	Creates the <code>clken</code> port.
Create an asynchronous clear port	On/Off	Creates the <code>aclr</code> port.
What should the RAM block type be?	Auto, M9K	Specifies the RAM block type.

10. ALTMEMMULT IP Core References

The ALTMEMMULT IP core creates only memory-based multipliers using on-chip memory blocks found in M9K memory blocks.

Figure 30. ALTMEMMULT IP Core Signals



10.1. ALTMEMMULT IP Core Signals for Intel MAX 10 Devices

Table 34. ALTMEMMULT IP Core Input Signals

Signal	Required	Description
clock	Yes	Clock input to the multiplier.
coeff_in[]	No	Coefficient input port for the multiplier. The size of the input port depends on the <code>WIDTH_C</code> parameter value.
data_in[]	Yes	Data input port to the multiplier. The size of the input port depends on the <code>WIDTH_D</code> parameter value.
sclr	No	Synchronous clear input. If unused, the default value is active high.
sel[]	No	Fixed coefficient selection. The size of the input port depends on the <code>WIDTH_S</code> parameter value.
sload_coeff	No	Synchronous load coefficient input port. Replaces the current selected coefficient value with the value specified in the <code>coeff_in</code> input.
sload_data	No	Synchronous load data input port. Signal that specifies new multiplication operation and cancels any existing multiplication operation. If the <code>MAX_CLOCK_CYCLES_PER_RESULT</code> parameter has a value of 1, the <code>sload_data</code> input port is ignored.

Table 35. ALTMEMMULT IP Core Output Signals

Signal	Required	Description
result[]	Yes	Multiplier output port. The size of the input port depends on the WIDTH_R parameter value.
result_valid	Yes	Indicates when the output is the valid result of a complete multiplication. If the MAX_CLOCK_CYCLES_PER_RESULT parameter has a value of 1, the result_valid output port is not used.
load_done	No	Indicates when the new coefficient has finished loading. The load_done signal asserts when a new coefficient has finished loading. Unless the load_done signal is high, no other coefficient value can be loaded into the memory.

10.2. ALTMEMMULT IP Core Parameters for Intel MAX 10 Devices

Table 36. ALTMEMMULT IP Core Parameters for Intel MAX 10 Devices

This table lists the IP core parameters applicable to Intel MAX 10 devices.

Option	Values	Description
How wide should the 'data_in' input bus be?	2, 3, 4, 5, 6, 7, 8, 16, 24, and 32	Specifies the width of the data_in port.
What is the representation of 'data_in'?	SIGNED, UNSIGNED	Specifies whether the data_in input port is signed or unsigned.
How wide should the coefficient be?	2, 3, 4, 5, 6, 7, 8, 16, 24	Specifies the width of the coeff_in port.
What is the representation of the coefficient?	SIGNED, UNSIGNED	Specifies whether the coeff_in input port and the pre-loaded coefficients are signed or unsigned.
What is the value of the initial coefficient?	0, 1, 2, 3, and 4	Specifies value of the first fixed coefficient.
Create ports to allow loading coefficients	On/Off	Creates the coeff_in and sload_coeff port.
Create a synchronous clear input	On/Off	Creates the sclr port.
What should the RAM block type be?	Auto, M9K	Specifies the RAM block type.

11. Document Revision History for the Intel MAX 10 Embedded Memory User Guide

Document Version	Changes
2023.05.05	<ul style="list-style-type: none"> Added new topics: <ul style="list-style-type: none"> FIFO Functional Timing Requirements SCFIFO ALMOST_EMPTY Functional Timing Removed DCFIFO_MIXED_WIDTH information as it is not supported in Intel MAX 10 devices in the following topics: <ul style="list-style-type: none"> FIFO IP Core Output Signals table in FIFO IP Core Signals for Intel MAX 10 Devices topic. FIFO IP Core Parameters for Intel MAX 10 Devices table in FIFO IP Core Parameters for Intel MAX 10 Devices topic.
2021.09.17	Updated the description for Dual clock: use separate 'input' and 'output' clocks in Table: RAM: 2-Port IP Core Parameters for Intel MAX 10 Devices.
2018.06.12	<ul style="list-style-type: none"> Renamed the document as <i>Intel MAX 10 Embedded Memory User Guide</i>. Added a new Topic: <i>Memory Configurations for Single-Port Modes</i>. Updated the following Topics: <ul style="list-style-type: none"> Byte Enable RAM Blocks Operations Port Width Configurations Updated the following Tables: <ul style="list-style-type: none"> Effects of Read Enable on Data Output Port Simple Dual-port Memory Configurations for M9K Blocks. RAM: 1-Port IP Core Output Signals RAM: 1-Port IP Core Parameters for Intel MAX 10 Devices RAM: 2-Port IP Core Parameters for Intel MAX 10 Devices ROM: 1-Port IP Core Parameters for Intel MAX 10 Devices ROM: 2-Port IP Core Parameters for Intel MAX 10 Devices FIFO IP Core Output Signals FIFO IP Core Parameters for Intel MAX 10 Devices Updated for latest Intel branding standards.

Date	Version	Changes
February 2017	2017.02.21	<ul style="list-style-type: none"> Rebranded as Intel.
October 2016	2016.10.31	<ul style="list-style-type: none"> Added note stating that the memory initialization feature is supported in MAX 10 Analog and Flash feature options only.
continued...		

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.

ISO
9001:2015
Registered

Date	Version	Changes
November 2015	2015.11.02	<ul style="list-style-type: none"> Revised the title for the tables in the Embedded Memory Configuration topic. Added a link to the MAX 10 FPGA Device Overview in the Consider Power-Up State and Memory Initialization topic. Changes instances of Quartus II to Quartus Prime.
May 2015	2015.05.04	<ul style="list-style-type: none"> Updated 'Yes, use this file for the memory content data' parameter note for RAM:1-Port, RAM:2-Port, ROM:1-Port, and ROM:2-Port. Added information about the internal configuration mode that supports memory initialization in 'Consider Power-Up State and Memory Initialization'
September 2014	2014.09.22	Initial release.

MAX[®] 10 Embedded Multipliers User Guide



Online Version

Send Feedback

UG-M10DSP

683467

2024.03.08

Contents

1. MAX® 10 Embedded Multiplier Block Overview.....	3
2. MAX® 10 Embedded Multipliers Features and Architecture.....	5
2.1. Embedded Multipliers Architecture.....	5
2.1.1. Input Register.....	5
2.1.2. Multiplier Stage.....	6
2.1.3. Output Register.....	6
2.2. Embedded Multipliers Operational Modes.....	7
2.2.1. 18-Bit Multipliers.....	7
2.2.2. 9-Bit Multipliers.....	8
3. MAX 10 Embedded Multipliers Implementation Guides.....	10
3.1. Files Generated by IP Cores.....	10
3.1.1. Verilog HDL Prototype Location.....	10
3.1.2. VHDL Component Declaration Location.....	11
4. LPM_MULT (Multiplier) IP Core References for MAX 10.....	12
4.1. LPM_MULT Parameter Settings.....	12
4.2. Signals.....	13
5. ALTMULT_ACCUM (Multiply-Accumulate) IP Core References for MAX 10.....	15
5.1. ALTMULT_ACCUM Parameter Settings.....	15
5.2. ALTMULT_ACCUM Ports.....	20
6. ALTMULT_ADD (Multiply-Adder) IP Core References for MAX 10.....	22
6.1. ALTMULT_ADD Parameter Settings.....	22
6.2. ALTMULT_ADD Ports.....	27
7. ALTMULT_COMPLEX (Complex Multiplier) IP Core References for MAX 10.....	28
7.1. ALTMULT_COMPLEX Parameter Settings.....	28
7.2. Signals.....	29
8. MAX 10 Embedded Multipliers User Guide Archives.....	30
9. Document Revision History for the MAX 10 Embedded Multipliers User Guide	31

1. MAX[®] 10 Embedded Multiplier Block Overview

The embedded multiplier is configured as either one 18 x 18 multiplier or two 9 x 9 multipliers. For multiplications greater than 18 x 18, the Quartus[®] Prime software cascades multiple embedded multiplier blocks together. There are no restrictions on the data width of the multiplier but the greater the data width, the slower the multiplication process.

Figure 1. Embedded Multipliers Arranged in Columns with Adjacent LABS

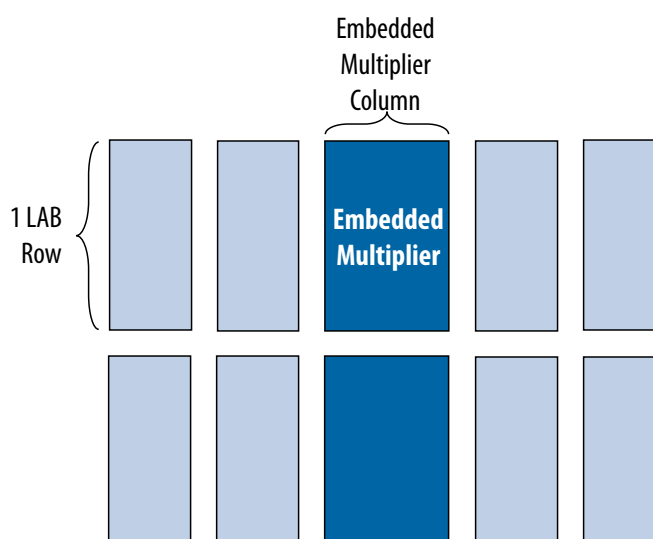


Table 1. Number of Embedded Multipliers in the MAX[®] 10 Devices

Device	Embedded Multipliers	9 x 9 Multipliers ⁽¹⁾	18 x 18 Multipliers ⁽¹⁾
10M02	16	32	16
10M04	20	40	20
10M08	24	48	24
10M16	45	90	45
10M25	55	110	55
10M40	125	250	125
10M50	144	288	144

⁽¹⁾ These columns show the number of 9 x 9 or 18 x 18 multipliers for each device. The total number of multipliers for each device is not the sum of all the multipliers.

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.

**ISO
9001:2015
Registered**

You can implement soft multipliers by using the M9K memory blocks as look-up tables (LUTs). The LUTs contain partial results from multiplying input data with coefficients implementing variable depth and width high-performance soft multipliers for low-cost, high-volume DSP applications. The availability of soft multipliers increases the number of available multipliers in the device.

Table 2. Number of Multipliers in the MAX[®] 10 Devices

Device	Embedded Multipliers	Soft Multipliers (16 x 16) (2)	Total Multipliers (3)
10M02	16	12	28
10M04	20	21	41
10M08	24	42	66
10M16	45	61	106
10M25	55	75	130
10M40	125	140	265
10M50	144	182	326

Related Information

[MAX 10 Embedded Multipliers User Guide Archives](#) on page 30

Provides a list of user guides for previous versions of the LPM_MULT, ALTMULT_ACCUM, ALTMULT_ADD, and ALTMULT_COMPLEX IP cores.

(2) Soft multipliers are implemented in sum of multiplication mode. M9K memory blocks are configured with 18-bit data widths to support 16-bit coefficients. The sum of the coefficients requires 18-bits of resolution to account for overflow.

(3) The total number of multipliers may vary, depending on the multiplier mode you use.

2. MAX[®] 10 Embedded Multipliers Features and Architecture

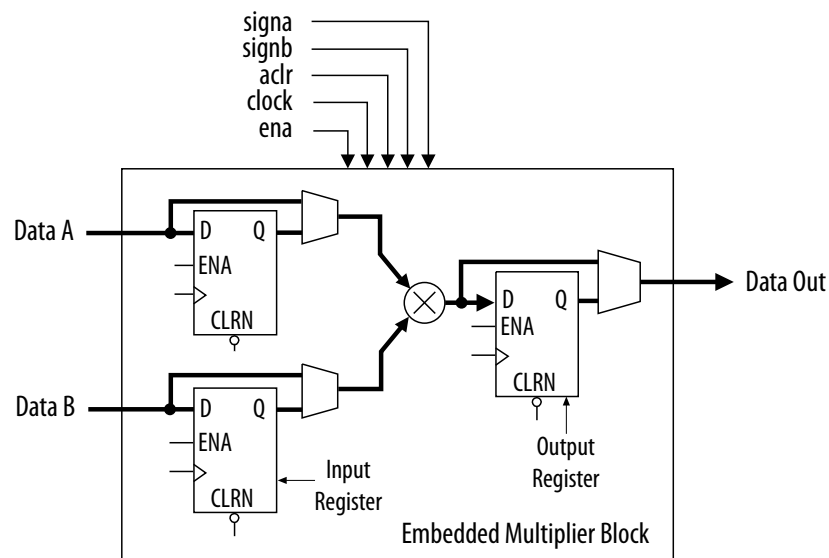
Each embedded multiplier consists of three elements. Depending on the application needs, you can use an embedded multiplier block in one of two operational modes.

2.1. Embedded Multipliers Architecture

Each embedded multiplier consists of the following elements:

- Multiplier stage
- Input and output registers
- Input and output interfaces

Figure 2. Multiplier Block Architecture



2.1.1. Input Register

Depending on the operational mode of the multiplier, you can send each multiplier input signal into either one of the following:

- An input register
- The multiplier in 9- or 18-bit sections

Each multiplier input signal can be sent through a register independently of other input signals. For example, you can send the multiplier `Data A` signal through a register and send the `Data B` signal directly to the multiplier.

The following control signals are available to each input register in the embedded multiplier:

- Clock
- Clock enable
- Asynchronous clear

All input and output registers in a single embedded multiplier are fed by the same clock, clock enable, and asynchronous clear signals.

2.1.2. Multiplier Stage

The multiplier stage of an embedded multiplier block supports 9×9 or 18×18 multipliers and other multipliers in between these configurations. Depending on the data width or operational mode of the multiplier, a single embedded multiplier can perform one or two multiplications in parallel.

Each multiplier operand is a unique signed or unsigned number. Two signals, `signa` and `signb`, control an input of a multiplier and determine if the value is signed or unsigned. If the `signa` signal is high, the `Data A` operand is a signed number. If the `signa` signal is low, the `Data A` operand is an unsigned number.

The following table lists the sign of the multiplication results for the various operand sign representations. The results of the multiplication are signed if any one of the operands is a signed value.

Data A		Data B		Result
<code>signa</code> Value	Logic Level	<code>signb</code> Value	Logic Level	
Unsigned	Low	Unsigned	Low	Unsigned
Unsigned	Low	Signed	High	Signed
Signed	High	Unsigned	Low	Signed
Signed	High	Signed	High	Signed

You can dynamically change the `signa` and `signb` signals to modify the sign representation of the input operands at run time. You can send the `signa` and `signb` signals through a dedicated input register. The multiplier offers full precision, regardless of the sign representation.

When the `signa` and `signb` signals are unused, the Quartus Prime software sets the multiplier to perform unsigned multiplication by default.

2.1.3. Output Register

You can register the embedded multiplier output using output registers in either 18- or 36-bit sections. This depends on the operational mode of the multiplier. The following control signals are available for each output register in the embedded multiplier:

- Clock
- Clock enable
- Asynchronous clear

All input and output registers in a single embedded multiplier are fed by the same clock, clock enable, and asynchronous clear signals.

2.2. Embedded Multipliers Operational Modes

You can use an embedded multiplier block in one of two operational modes, depending on the application needs:

- One 18-bit x 18-bit multiplier
- Up to two 9-bit x 9-bit independent multipliers

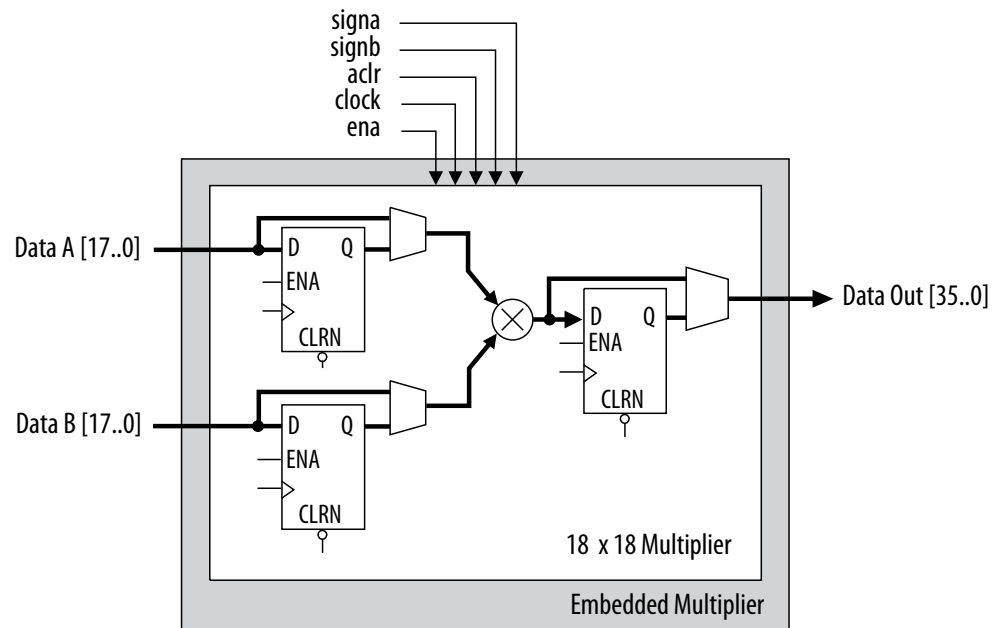
You can also use embedded multipliers of the MAX® 10 devices to implement multiplier adder and multiplier accumulator functions. The multiplier portion of the function is implemented using embedded multipliers. The adder or accumulator function is implemented in logic elements (LEs).

2.2.1. 18-Bit Multipliers

You can configure each embedded multiplier to support a single 18 x 18 multiplier for input widths of 10 to 18 bits.

The following figure shows the embedded multiplier configured to support an 18-bit multiplier.

Figure 3. 18-Bit Multiplier Mode



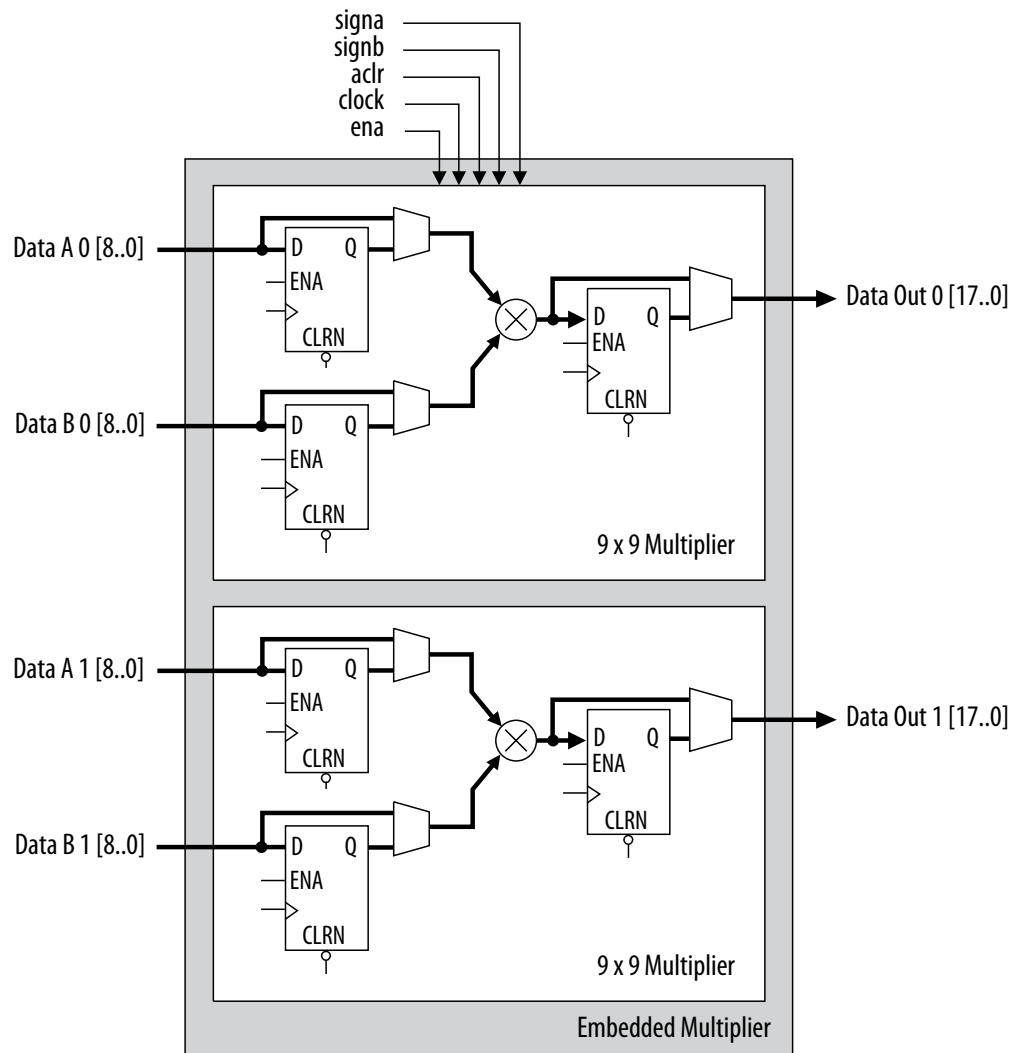
All 18-bit multiplier inputs and results are independently sent through registers. The multiplier inputs can accept signed integers, unsigned integers, or a combination of both. Also, you can dynamically change the *signa* and *signb* signals and send these signals through dedicated input registers.

2.2.2. 9-Bit Multipliers

You can configure each embedded multiplier to support two 9×9 independent multipliers for input widths of up to 9 bits.

The following figure shows the embedded multiplier configured to support two 9-bit multipliers.

Figure 4. 9-Bit Multiplier Mode



All 9-bit multiplier inputs and results are independently sent through registers. The multiplier inputs can accept signed integers, unsigned integers, or a combination of both.

Each embedded multiplier block has only one *signa* and one *signb* signal to control the sign representation of the input data to the block. If the embedded multiplier block has two 9 x 9 multipliers the following applies:

- The Data A input of both multipliers share the same *signa* signal
- The Data B input of both multipliers share the same *signb* signal

3. MAX 10 Embedded Multipliers Implementation Guides

The Quartus Prime software contains tools for you to create and compile your design, and configure your device.

You can prepare for device migration, set pin assignments, define placement restrictions, setup timing constraints, and customize IP cores using the Quartus Prime software.

Related Information

- [Introduction to Intel FPGA IP Cores](#)
Provides general information about all Intel FPGA IP cores, including parameterizing, generating, upgrading, and simulating IP cores.
- [Creating Version-Independent IP and Platform Designer Simulation Scripts](#)
Create simulation scripts that do not require manual updates for software or IP version upgrades.
- [Project File Best Practices](#)
Guidelines for efficient management and portability of your project and IP files.

3.1. Files Generated by IP Cores

The following integer arithmetic IP cores use the MAX 10 device embedded multipliers block:

- LPM_MULT
- ALTMULT_ACCUM (MAC)
- ALTMULT_ADD
- ALTMULT_COMPLEX

3.1.1. Verilog HDL Prototype Location

You can view the Verilog HDL prototype for the IP cores in the following Verilog Design Files (.v):

Table 3. Verilog HDL Prototype Location

Integer Arithmetic IP Core	Directory	Verilog Design File (.v)
LPM_MULT	<Quartus Prime installation directory>\eda \synthesis	lpm.v
<ul style="list-style-type: none"> • ALTMULT_ACCUM • ALTMULT_ADD • ALTMULT_COMPLEX 	<Quartus Prime installation directory>\eda \synthesis	altera_mf.v

3.1.2. VHDL Component Declaration Location

You can view the VHDL component declaration for the IP cores in the following VHDL Design Files (.vhd):

Integer Arithmetic IP Core	Directory	VHDL Design File (.vhd)
LPM_MULT	<Quartus Prime installation directory> \libraries\vhdl\lpm	LPM_PACK.vhd
<ul style="list-style-type: none">• ALTMULT_ACCUM• ALTMULT_ADD• ALTMULT_COMPLEX	<Quartus Prime installation directory> \libraries\vhdl\altera_mf	altera_mf_components.vhd

4. LPM_MULT (Multiplier) IP Core References for MAX 10

4.1. LPM_MULT Parameter Settings

There are three groups of options: **General**, **General2**, and **Pipelining**.

Table 4. LPM_MULT Parameters - General

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
Multiplier configuration	—	—	<ul style="list-style-type: none"> Multiply 'dataaa' input by 'datab' input Multiply 'dataaa' input by itself (squaring operation) 	Specifies the multiplier configuration.
How wide should the 'dataaa' input be?	LPM_WIDTHHA	—	1–256	Specifies the width of the dataaa[] port.
How wide should the 'datab' input be?	LPM_WIDTHHB	—	1–256	Specifies the width of the datab[] port.
How should the width of the 'result' output be determined?	LPM_WIDTHHP	—	<ul style="list-style-type: none"> Automatically calculate the width Restrict the width to [] bits 	Specifies how the result width is determined.
How should the width of the 'result' output be determined? > Restrict the width to [] bits	LPM_WIDTHHP	How should the width of the 'result' output be determined? > Restrict the width to [] bits = On	1–256	You can set the result width.

Table 5. LPM_MULT Parameters - General2

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
Does the 'datab' input bus have a constant value?	—	—	<ul style="list-style-type: none"> No Yes, the value is [] 	You can specify the constant value of the 'datab' input bus, if any.
Which type of multiplication do you want?	LPM_REPRESENTATION	—	<ul style="list-style-type: none"> Unsigned Signed 	Specifies the type of multiplication performed.
Which multiplier implementation should be used?	DEDICATED_MULTIPLIER_CIRCUITRY	—	<ul style="list-style-type: none"> Use default implementation Use the dedicated multiplier circuitry (Not available for all families) Use logic elements 	Specifies the multiplier implementation.

Table 6. LPM_MULT Parameters - Pipeling

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
Do you want to pipeline the function?	LPM_PIPELINE	—	<ul style="list-style-type: none"> No Yes, I want output latency of [] clock cycles 	You can add extra latency to the outputs, if any.
Create an 'aclr' asynchronous clear port	—	Do you want to pipeline the function? = Yes, I want output latency of [] clock cycles	On or off	Specifies asynchronous clear for the complex multiplier. Clears the function asynchronously when aclr port is asserted high.
Create a 'clken' clock enable clock	—	Do you want to pipeline the function? = Yes, I want output latency of [] clock cycles	On or off	Specifies active high clock enable for the clock port of the complex multiplier
What type of optimization do you want?	MAXIMIZE_SPEED	—	<ul style="list-style-type: none"> Default Speed Area 	You can specify if the type of optimization is determined by Quartus Prime, speed, or area.

4.2. Signals

Table 7. LPM_MULT Core Input Signals

Signal Name	Required	Description
dataa[]	Yes	Data input. The size of the input signal depends on the LPM_WIDTHA parameter value.
datab[]	Yes	Data input. The size of the input signal depends on the LPM_WIDTHB parameter value.
clock	No	Clock input for pipelined usage.
<i>continued...</i>		

Signal Name	Required	Description
		For LPM_PIPELINE values other than 0 (default), the clock signal must be enabled.
clken	No	Clock enable for pipelined usage. When the clken signal is asserted high, the adder/subtractor operation takes place. When the signal is low, no operation occurs. If omitted, the default value is 1.
aclr	No	Asynchronous clear signal used at any time to reset the pipeline to all 0s, asynchronously to the clock signal. The pipeline initializes to an undefined (X) logic level. The outputs are a consistent, but non-zero value.
sclr	No	Synchronous clear signal used at any time to reset the pipeline to all 0s, synchronously to the clock signal. The pipeline initializes to an undefined (X) logic level. The outputs are a consistent, but non-zero value.

Table 8. LPM_MULT Output signals

signal Name	Required	Description
result[]	Yes	Data output. For Stratix V, Arria V and Cyclone V, the size of the output signal depends on the LPM_WIDTHP parameter value. If LPM_WIDTHP < max (LPM_WIDTHA + LPM_WIDTHB, LPM_WIDTHS) or (LPM_WIDTHA + LPM_WIDTHS), only the LPM_WIDTHP MSBs are present.

5. ALTMULT_ACCUM (Multiply-Accumulate) IP Core References for MAX 10

5.1. ALTMULT_ACCUM Parameter Settings

There are four groups of options: **General**, **Extra Modes**, **Multipliers**, and **Accumulator**.

Table 9. ALTMULT_ACCUM Parameters - General

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
What is the number of multipliers?	NUMBER_OF_MULTIPLIERS	—	1	By default, only 1 multiplier is supported.
All multipliers have similar configurations	—	—	On	By default all multipliers have similar configurations
How wide should the A input buses be?	WIDTH_A	—	1–256	Specifies the width of A input buses.
How wide should the B input buses be?	WIDTH_B	—	1–256	Specifies the width of B input buses.
How wide should the 'result' output bus be?	WIDTH_RESULT	—	1–256	Specifies the width of 'result' output bus.
Create a 4 th asynchronous clear input option	—	—	On or Off	Turn on this option if you want to create a 4 th asynchronous clear input option.
Create an associated clock enable for each clock	—	—	On or Off	Turn on this option if you want to create an associated clock enable for each clock.
What is the representation format for A inputs?	REPRESENTATION_A	—	<ul style="list-style-type: none"> Signed Unsigned Variable 	Specifies the representation format for A inputs.
'signa' input controls the sign (1 signed/0 unsigned)	PORT_SIGNA	Input Representation > What is the representation format for A inputs? = Variable	More Options	High 'signa' input indicates signed and low 'signa' input indicates unsigned.
Register 'signa' input	—	Input Representation > More Options	On or Off	Turn on this option if you want to enable the register of 'signa' input
continued...				

GUI Parameter	Parameter	Condition	Value	Description
Add an extra pipeline register	—	Input Representation > More Options	On or Off	Turn on this option if you want to enable the extra pipeline register
Input Register > What is the source for clock input?	SIGN_REG_A	Input Representation > More Options	Clock0–Clock3	Specifies the source for clock input.
Input Register > What is the source for asynchronous clear input?	SIGN_ACLR_A	Input Representation > More Options	<ul style="list-style-type: none"> Aclr0–Aclr2 None 	Specifies the source for asynchronous clear input.
Pipeline Register > What is the source for clock input?	SIGN_PIPELINE_REG_A	Input Representation > More Options	Clock0–Clock3	Specifies the source for clock input.
Pipeline Register > What is the source for asynchronous clear input?	SIGN_PIPELINE_ACLR_A	Input Representation > More Options	<ul style="list-style-type: none"> Aclr0–Aclr2 None 	Specifies the source for asynchronous clear input.
What is the representation format for B inputs?	REPRESENTATIONS_B	—	<ul style="list-style-type: none"> Signed Unsigned Variable 	Specifies the representation format for B inputs.
'signb' input controls the sign (1 signed/0 unsigned)	PORT_SIGNB	Input Representation > What is the representation format for B inputs? = Variable	More Options	High 'signb' input indicates signed and low 'signb' input indicates unsigned.
Register 'signb' input	—	Input Representation > More Options	On or Off	Turn on this option if you want to enable the register of 'signb' input
Add an extra pipeline register	—	Input Representation > More Options	On or Off	Turn on this option if you want to enable the extra pipeline register
Input Register > What is the source for clock input?	SIGN_REG_B	Input Representation > More Options	Clock0–Clock3	Specifies the source for clock input.
Input Register > What is the source for asynchronous clear input?	SIGN_ACLR_B	Input Representation > More Options	<ul style="list-style-type: none"> Aclr0–Aclr2 None 	Specifies the source for asynchronous clear input.
Pipeline Register > What is the source for clock input?	SIGN_PIPELINE_REG_B	Input Representation > More Options	Clock0–Clock3	Specifies the source for clock input.
Pipeline Register > What is the source for asynchronous clear input?	SIGN_PIPELINE_ACLR_B	Input Representation > More Options	<ul style="list-style-type: none"> Aclr0–Aclr2 None 	Specifies the source for asynchronous clear input.

Table 10. ALTMULT_ACCUM Parameters - Extra Modes

GUI Parameter	Parameter	Condition	Value	Description
Create a shiftout output from A input of the last multiplier	—	—	On or Off	Turn on this option to create a shiftout output from A input of the last multiplier.
Create a shiftout output from B input of the last multiplier	—	—	On or Off	Turn on this option to create a shiftout output from B input of the last multiplier.
Add extra register(s) at the output	—	—	On	By default, output register must be enabled for accumulator.
What is the source for clock input?	OUTPUT_REG	Outputs Configuration > More Options	Clock0–Clock3	Specifies the clock signal for the registers on the outputs.
What is the source for asynchronous clear input?	OUTPUT_ACLR	Outputs Configuration > More Options	<ul style="list-style-type: none"> Aclr0–Aclr2 None 	Specifies the asynchronous clear signal for the registers on the outputs.
Add [] extra latency to the output	—	Outputs Configuration > More Options	0, 1, 2, 3, 4, 5, 6, 7, 8, or 12	Specifies the extra latency to add to the output.
Which multiplier-adder implementation should be used?	DEDICATED_MULTIPLIER_CIRCUITRY	—	<ul style="list-style-type: none"> Use the default implementation Use dedicated multiplier circuitry (Not available for all families) Use logic elements 	Specifies the multiplier-adder implementation.

Table 11. ALTMULT_ACCUM Parameters - Multipliers

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
Register input A of the multiplier	—	—	On or Off	Turn on to enable register input A of the multiplier.
What is the source for clock input?	INPUT_REG_A	<ul style="list-style-type: none"> Input Configuration > Register input A of the multiplier = On Input Configuration > More Options 	Clock0–Clock3	Specifies the clock port for the dataa[] port.
What is the source for asynchronous clear input?	INPUT_ACLR_A	<ul style="list-style-type: none"> Input Configuration > Register input A of the multiplier = On Input Configuration > More Options 	<ul style="list-style-type: none"> Aclr0–Aclr2 None 	Specifies the asynchronous clear port for the dataa[] port.
Register input B of the multiplier	—	—	On or Off	Turn on to enable register input B of the multiplier.
continued...				

GUI Parameter	Parameter	Condition	Value	Description
What is the source for clock input?	INPUT_REG_B	<ul style="list-style-type: none"> Input Configuration > Register input B of the multiplier = On Input Configuration > More Options 	Clock0–Clock3	Specifies the clock port for the datab[] port.
What is the source for asynchronous clear input?	INPUT_ACLR_B	<ul style="list-style-type: none"> Input Configuration > Register input B of the multiplier = On Input Configuration > More Options 	<ul style="list-style-type: none"> Aclr0–Aclr2 None 	Specifies the asynchronous clear port for the datab[] port.
What is the input A of the multiplier connected to?	—	—	Multiplier input	By default, input A of the multiplier is always connected to the multiplier's input.
What is the input B of the multiplier connected to?	—	—	Multiplier input	By default, input B of the multiplier is always connected to the multiplier's input.
Register output of the multiplier	—	—	On or Off	Turn on to enable register output of the multiplier.
What is the source for clock input?	MULTIPLIER_REG	<ul style="list-style-type: none"> Output Configuration > Register output of the multiplier = On Output Configuration > More Options 	Clock0–Clock3	Specifies the clock signal for the register that immediately follows the multiplier.
What is the source for asynchronous clear input?	MULTIPLIER_ACLR	<ul style="list-style-type: none"> Output Configuration > Register output of the multiplier = On Output Configuration > More Options 	<ul style="list-style-type: none"> Aclr0–Aclr2 None 	Specifies the asynchronous clear signal of the register that follows the corresponding multiplier.

Table 12. ALTMULT_ACCUM Parameters - Accumulator

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
Create an 'accum_sload' input port	—	—	On or off	Dynamically specifies whether the accumulator value is constant. If the accum_sload port is high, then the
continued...				

GUI Parameter	Parameter	Condition	Value	Description
				multiplier output is loaded into the accumulator.
Register 'accum_sload' input	—	<ul style="list-style-type: none"> Accumulator > Create an 'accum_sload' input port = On Accumulator > More Options 	On or off	Turn on to enable register 'accum_sload' input.
Add an extra pipeline register	—	<ul style="list-style-type: none"> Accumulator > Create an 'accum_sload' input port = On Accumulator > More Options 	On or off	Turn on this option if you want to enable the extra pipeline register
Input Register > What is the source for clock input?	ACCUM_SLOAD_REG	<ul style="list-style-type: none"> Accumulator > Create an 'accum_sload' input port = On Accumulator > More Options 	Clock0–Clock3	Specifies the clock signal for the accum_sload port.
Input Register > What is the source for asynchronous clear input?	ACCUM_SLOAD_ACLR	<ul style="list-style-type: none"> Accumulator > Create an 'accum_sload' input port = On Accumulator > More Options 	<ul style="list-style-type: none"> Aclr0–Aclr2 None 	Specifies the asynchronous clear source for the first register on the accum_sload input.
Pipeline Register > What is the source for clock input?	ACCUM_SLOAD_PIPELINE_REG	<ul style="list-style-type: none"> Accumulator > Create an 'accum_sload' input port = On Accumulator > More Options 	Clock0–Clock3	Specifies the source for clock input.
Pipeline Register > What is the source for asynchronous clear input?	ACCUM_SLOAD_PIPELINE_ACLR	<ul style="list-style-type: none"> Accumulator > Create an 'accum_sload' input port = On Accumulator > More Options 	<ul style="list-style-type: none"> Aclr0–Aclr2 None 	Specifies the source for asynchronous clear input.
Create an 'overflow' output port	—	—	On or Off	Overflow port for the accumulator
Add [] extra latency to the multiplier output	EXTRA_MULTIPLIER_LATENCY	—	0, 1, 2, 3, 4, 5, 6, 7, 8, or 12	Specifies the number of clock cycles of latency for the multiplier portion of the DSP block. If the MULTIPLIER_REG parameter is specified, then the specified clock port is used to add the latency.

5.2. ALTMULT_ACCUM Ports

Table 13. ALTMULT_ACCUM IP Core Input Ports

Port Name	Required	Description
accum_sload	No	Causes the value on the accumulator feedback path to go to zero (0) or to accum_sload_upper_data when concatenated with 0. If the accumulator is adding and the accum_sload port is high, then the multiplier output is loaded into the accumulator. If the accumulator is subtracting, then the opposite (negative value) of the multiplier output is loaded into the accumulator.
aclr0	No	The first asynchronous clear input. The aclr0 port is active high.
aclr1	No	The second asynchronous clear input. The aclr1 port is active high.
aclr2	No	The third asynchronous clear input. The aclr2 port is active high.
aclr3	No	The fourth asynchronous clear input. The aclr3 port is active high.
addnsub	No	Controls the functionality of the adder. If the addnsub port is high, the adder performs an add function; if the addnsub port is low, the adder performs a subtract function.
clock0	No	Specifies the first clock input, usable by any register in the IP core.
clock1	No	Specifies the second clock input, usable by any register in the IP core.
clock2	No	Specifies the third clock input, usable by any register in the IP core.
clock3	No	Specifies the fourth clock input, usable by any register in the IP core.
dataa[]	Yes	Data input to the multiplier. The size of the input port depends on the WIDTH_A parameter value.
datab[]	Yes	Data input to the multiplier. The size of the input port depends on the WIDTH_B parameter value.
ena0	No	Clock enable for the clock0 port.
ena1	No	Clock enable for the clock1 port.
ena2	No	Clock enable for the clock2 port.
ena3	No	Clock enable for the clock3 port.
signa	No	Specifies the numerical representation of the dataa[] port. If the signa port is high, the multiplier treats the dataa[] port as signed two's complement. If the signa port is low, the multiplier treats the dataa[] port as an unsigned number.
signb	No	Specifies the numerical representation of the datab[] port. If the signb port is high, the multiplier treats the datab[] port as signed two's complement. If the signb port is low, the multiplier treats the datab[] port as an unsigned number.

Table 14. ALTMULT_ACCUM IP Core Output Ports

Port Name	Required	Description
overflow	No	Overflow port for the accumulator.
result[]	Yes	Accumulator output port. The size of the output port depends on the WIDTH_RESULT parameter value.
scanouta[]	No	Output of the first shift register. The size of the output port depends on the WIDTH_A parameter value. When instantiating the ALTMULT_ACCUM IP core with the MegaWizard Plug-In Manager, the MegaWizard Plug-In Manager renames the scanouta[] port to shiftouta port.
scanoutb[]	No	Output of the second shift register. The size of the input port depends on the WIDTH_B parameter value. When instantiating the ALTMULT_ACCUM IP core with the MegaWizard Plug-In Manager, the MegaWizard Plug-In Manager renames the scanoutb[] port to shiftoutb port.

6. ALTMULT_ADD (Multiply-Adder) IP Core References for MAX 10

6.1. ALTMULT_ADD Parameter Settings

There are three groups of options: **General**, **Extra Modes**, and **Multipliers**.

Table 15. ALTMULT_ADD Parameters - General

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
What is the number of multipliers?	NUMBER_OF_MULTIPLIERS	—	1, 2, 3, or 4	Specifies the number of multipliers. You can specify up to four multipliers.
All multipliers have similar configurations	—	—	On or Off	Turn on this option if you want all multipliers to have similar configurations.
How wide should the A input buses be?	WIDTH_A	—	1-256	Specifies the width of A input buses.
How wide should the B input buses be?	WIDTH_B	—	1-256	Specifies the width of B input buses.
How wide should the 'result' output bus be?	WIDTH_RESULT	—	1-256	Specifies the width of 'result' output bus.
Create a 4 th asynchronous clear input option	—	—	On or Off	Turn on this option if you want to create a 4 th asynchronous clear input option.
Create an associated clock enable for each clock	—	—	On or Off	Turn on this option if you want to create an associated clock enable for each clock.
What is the representation format for A inputs?	REPRESENTATION_A	—	<ul style="list-style-type: none"> Signed Unsigned Variable 	Specifies the representation format for A inputs.
'signa' input controls the sign (1 signed/0 unsigned)	PORT_SIGNA	Input Representation > What is the representation format for A inputs? = Variable	More Options	High 'signa' input indicates signed and low 'signa' input indicates unsigned.
continued...				

GUI Parameter	Parameter	Condition	Value	Description
Register 'signa' input	—	Input Representation > More Options	On or Off	Turn on this option if you want to enable the register of 'signa' input
Add an extra pipeline register	—	Input Representation > More Options	On or Off	Turn on this option if you want to enable the extra pipeline register
Input Register > What is the source for clock input?	SIGNED_REGISTER_A	Input Representation > More Options	Clock0–Clock3	Specifies the source for clock input.
Input Register > What is the source for asynchronous clear input?	SIGNED_ACLR_A	Input Representation > More Options	<ul style="list-style-type: none"> Aclr0–Aclr2 None 	Specifies the source for asynchronous clear input.
Pipeline Register > What is the source for clock input?	SIGNED_PIPELINE_REGISTER_A	Input Representation > More Options	Clock0–Clock3	Specifies the source for clock input.
Pipeline Register > What is the source for asynchronous clear input?	SIGNED_PIPELINE_ACLR_A	Input Representation > More Options	<ul style="list-style-type: none"> Aclr0–Aclr2 None 	Specifies the source for asynchronous clear input.
What is the representation format for B inputs?	REPRESENTATIONS_B	—	<ul style="list-style-type: none"> Signed Unsigned Variable 	Specifies the representation format for B inputs.
'signb' input controls the sign (1 signed/0 unsigned)	PORT_SIGNB	Input Representation > What is the representation format for B inputs? = Variable	More Options	High 'signb' input indicates signed and low 'signb' input indicates unsigned.
Register 'signb' input	—	Input Representation > More Options	On or Off	Turn on this option if you want to enable the register of 'signb' input
Add an extra pipeline register	—	Input Representation > More Options	On or Off	Turn on this option if you want to enable the extra pipeline register
Input Register > What is the source for clock input?	SIGNED_REGISTER_B	Input Representation > More Options	Clock0–Clock3	Specifies the source for clock input.
Input Register > What is the source for asynchronous clear input?	SIGNED_ACLR_B	Input Representation > More Options	<ul style="list-style-type: none"> Aclr0–Aclr2 None 	Specifies the source for asynchronous clear input.
Pipeline Register > What is the source for clock input?	SIGNED_PIPELINE_REGISTER_B	Input Representation > More Options	Clock0–Clock3	Specifies the source for clock input.
Pipeline Register > What is the source for asynchronous clear input?	SIGNED_PIPELINE_ACLR_B	Input Representation > More Options	<ul style="list-style-type: none"> Aclr0–Aclr2 None 	Specifies the source for asynchronous clear input.

Table 16. ALTMULT_ADD Parameters - Extra Modes

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
Create a shiftout output from A input of the last multiplier	—	—	On or Off	Turn on to create a signal from A input.
Create a shiftout output from B input of the last multiplier	—	—	On or Off	Turn on to create a signal from B input.
Register output of the adder unit	—	—	On or Off	Turn on to create a register output of the adder unit.
What is the source for clock input?	OUTPUT_REGISTER	<ul style="list-style-type: none"> Outputs Configuration > Register output of the adder unit = On Outputs Configuration > More Options 	Clock0–Clock3	Specifies the clock signal for the output register.
What is the source for asynchronous clear input?	OUTPUT_ACLR	<ul style="list-style-type: none"> Outputs Configuration > Register output of the adder unit = On Outputs Configuration > More Options 	<ul style="list-style-type: none"> Aclr0–Aclr2 None 	Specifies the source for asynchronous clear input.
What operation should be performed on outputs of the first pair of multipliers?	MUTIPLIER1_DIRECTION	General > What is the number of multipliers? = 2, 3, or 4	<ul style="list-style-type: none"> Add Subtract Variable 	Specifies whether the second multiplier adds or subtracts its value from the sum. Values are add and subtract. If Variable is selected the addnsub1 port is used.
'addnsub1' input controls the operation (1 add/0 sub)	—	Adder Operation > What operation should be performed on outputs of the first pair of multipliers? = Variable	More Options	High 'addnsub1' input indicates add and low 'addnsub1' input indicates subtract.
Register 'addnsub1' input	—	—	On or Off	Turn on this option if you want to enable the register of 'addnsub1' input
Add an extra pipeline register	—	—	On or Off	Turn on this option if you want to enable the extra pipeline register
Input Register > What is the source for clock input?	ADDNSUB_MULTIPLIER_REGISTER[1]	Adder Operation > More Options	Clock0–Clock3	Specifies the source for clock input.
Input Register > What is the source for asynchronous clear input?	ADDNSUB_MULTIPLIER_ACLR[1]	Adder Operation > More Options	<ul style="list-style-type: none"> Aclr0–Aclr2 None 	Specifies the source for asynchronous clear input.

continued...

GUI Parameter	Parameter	Condition	Value	Description
Pipeline Register > What is the source for clock input?	ADDNSUB_MULTIPLIER_PIPELINE_REGISTER[1]	Adder Operation > More Options	Clock0–Clock3	Specifies the source for clock input.
Pipeline Register > What is the source for asynchronous clear input?	ADDNSUB_MULTIPLIER_PIPELINE_ACLR[1]	Adder Operation > More Options	<ul style="list-style-type: none"> Aclr0–Aclr2 None 	Specifies the source for asynchronous clear input.
What operation should be performed on outputs of the second pair of multipliers?	MUTIMPLIER3_DIRECTION	General > What is the number of multipliers? = 4	—	Specifies whether the fourth and all subsequent odd-numbered multipliers add or subtract their results from the total. Values are add and subtract. If Variable is selected, the addsub3 port is used.
'addsub3' input controls the sign (1 add/0 sub) - More Options	—	—	—	High 'addsub3' input indicates add and low 'addsub3' input indicates subtract.
Register 'addsub3' input	—	—	On or Off	Turn on this option if you want to enable the register of 'addsub3' input.
Add an extra pipeline register	—	—	On or Off	Turn on this option if you want to enable the extra pipeline register.
Input Register > What is the source for clock input?	ADDNSUB_MULTIPLIER_REGISTER[3]	Adder Operation > More Options	Clock0–Clock3	Specifies the source for clock input.
Input Register > What is the source for asynchronous clear input?	ADDNSUB_MULTIPLIER_ACLR[3]	Adder Operation > More Options	<ul style="list-style-type: none"> Aclr0–Aclr2 None 	Specifies the source for asynchronous clear input.
Pipeline Register > What is the source for clock input?	ADDNSUB_MULTIPLIER_PIPELINE_REGISTER[3]	Adder Operation > More Options	Clock0–Clock3	Specifies the source for clock input.
Pipeline Register > What is the source for asynchronous clear input?	ADDNSUB_MULTIPLIER_PIPELINE_ACLR[3]	Adder Operation > More Options	<ul style="list-style-type: none"> Aclr0–Aclr2 None 	Specifies the source for asynchronous clear input.
Which multiplier-adder implementation should be used?	DEDICATED_MULTIPLIER_CIRCUITRY	—	<ul style="list-style-type: none"> Use the default implementation Use dedicated multiplier circuitry (Not available for all families) Use logic elements 	Specifies the multiplier-adder implementation.

Table 17. ALTMULT_ADD Parameters - Multipliers

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
Register input A of the multiplier	—	—	On or Off	Turn on to enable register input A of the multiplier.
What is the source for clock input?	INPUT_REGISTER_A[0..3]	<ul style="list-style-type: none"> Input Configuration > Register input A of the multiplier = On • Input Configuration > More Options 	Clock0–Clock3	Specifies the source for clock input.
What is the source for asynchronous clear input?	INPUT_ACLR_A[0..3]	<ul style="list-style-type: none"> Input Configuration > Register input A of the multiplier = On • Input Configuration > More Options 	<ul style="list-style-type: none"> Aclr0–Aclr2 • None 	Specifies the source for asynchronous clear input.
Register input B of the multiplier	—	—	On or Off	Turn on to enable register input B of the multiplier.
What is the source for clock input?	INPUT_REGISTER_B[0..3]	<ul style="list-style-type: none"> Input Configuration > Register input B of the multiplier = On • Input Configuration > More Options 	Clock0–Clock3	Specifies the source for clock input.
What is the source for asynchronous clear input?	INPUT_ACLR_B[0..3]	<ul style="list-style-type: none"> Input Configuration > Register input B of the multiplier = On • Input Configuration > More Options 	<ul style="list-style-type: none"> Aclr0–Aclr2 • None 	Specifies the source for asynchronous clear input.
What is the input A of the multiplier connected to?	INPUT_SOURCE_A[0..3]	—	<ul style="list-style-type: none"> Multiplier input • Shiftin input 	Specifies the input A of the multiplier is connected to either multiplier input or shiftin input.
What is the input B of the multiplier connected to?	INPUT_SOURCE_B[0..3]	—	<ul style="list-style-type: none"> Multiplier input • Shiftin input 	Specifies the input B of the multiplier is connected to either multiplier input or shiftin input.
continued...				

GUI Parameter	Parameter	Condition	Value	Description
Register output of the multiplier	—	—	On or Off	Turn on to enable the register for output of the multiplier.
What is the source for clock input?	MULTIPLIER_REGISTER[]	<ul style="list-style-type: none"> Output Configuration > Register output of the multiplier = On Output Configuration > More Options 	Clock0–Clock3	Specifies the source for clock input.
What is the source for asynchronous clear input?	MULTIPLIER_ACLR[]	<ul style="list-style-type: none"> Output Configuration > Register output of the multiplier = On Output Configuration > More Options 	<ul style="list-style-type: none"> Aclr0–Aclr2 None 	Specifies the source for asynchronous clear input.

6.2. ALTMULT_ADD Ports

Table 18. ALTMULT_ADD IP Core Input Ports

Port Name	Required	Description
dataa[]	Yes	Data input to the multiplier. Input port [NUMBER_OF_MULTIPLIERS * WIDTH_A – 1..0] wide.
datab[]	Yes	Data input to the multiplier. Input port [NUMBER_OF_MULTIPLIERS * WIDTH_B – 1..0] wide.
clock[]	No	Clock input port [0..3] to the corresponding register. This port can be used by any register in the IP core.
aclr[]	No	Input port [0..3]. Asynchronous clear input to the corresponding register.
ena[]	No	Input port [0..3]. Clock enable for the corresponding clock[] port.
signa	No	Specifies the numerical representation of the dataa[] port. If the signa port is high, the multiplier treats the dataa[] port as a signed two's complement number. If the signa port is low, the multiplier treats the dataa[] port as an unsigned number.
signb	No	Specifies the numerical representation of the datab[] port. If the signb port is high, the multiplier treats the datab[] port as a signed two's complement number. If the signb port is low, the multiplier treats the datab[] port as an unsigned number.

Table 19. ALTMULT_ADD IP Core Output Ports

Port Name	Required	Description
result[]	Yes	Multiplier output port. Output port [WIDTH_RESULT – 1..0] wide.
overflow	No	Overflow flag. If output_saturation is enabled, overflow flag is set.
scanouta[]	No	Output of scan chain A. Output port [WIDTH_A – 1..0] wide.
scanoutb[]	No	Output of scan chain B. Output port [WIDTH_B – 1..0] wide.

7. ALTMULT_COMPLEX (Complex Multiplier) IP Core References for MAX 10

7.1. ALTMULT_COMPLEX Parameter Settings

There are two groups of options: **General** and **Implementation Style/Pipelining**.

Table 20. ALTMULT_COMPLEX Parameters - General

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
How wide should the A input buses be?	WIDTH_A	—	1–256	Specifies the width of A input buses.
How wide should the B input buses be?	WIDTH_B	—	1–256	Specifies the width of B input buses.
How wide should the 'result' output bus be?	WIDTH_RESULT	—	1–256	Specifies the width of 'result' output bus.
What is the representation format for A inputs?	REPRESENTATION_A	—	<ul style="list-style-type: none"> Signed Unsigned 	Specifies the representation format for A inputs.
What is the representation format for B inputs?	REPRESENTATIONS_B	—	<ul style="list-style-type: none"> Signed Unsigned 	Specifies the representation format for B inputs.

Table 21. ALTMULT_COMPLEX Parameters - Implementation Style/Pipelining

This table lists the IP core parameters applicable to MAX 10 devices.

GUI Parameter	Parameter	Condition	Value	Description
Which implementation style should be used?	IMPLEMENTATION_STYLE	—	Automatically select a style for best trade-off for the current settings	By default automatic selection for MAX 10 device is selected. Quartus Prime software will determine the best implementation based on the selected device family and input width.
Output latency [] clock cycles	PIPELINE	—	0–14	Specifies the number of clock cycles for output latency.
Create an asynchronous Clear input	—	—	On or off	Specifies synchronous clear for the complex multiplier. Clears the function

continued...

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.

GUI Parameter	Parameter	Condition	Value	Description
				asynchronously when the <code>aclr</code> port is asserted high.
Create clock enable input	—	—	On or off	Specifies active high clock enable for the clock port of the complex multiplier.

7.2. Signals

Table 22. ALTMULT_COMPLEX Input Signals

Signal	Required	Description
<code>aclr</code>	No	Asynchronous clear for the complex multiplier. When the <code>aclr</code> signal is asserted high, the function is asynchronously cleared.
<code>sclr</code>	No	Synchronous clear for the complex multiplier. When the <code>sclr</code> signal is asserted high, the function is asynchronously cleared.
<code>clock</code>	Yes	Clock input to the ALTMULT_COMPLEX function.
<code>dataa_imag[]</code>	Yes	Imaginary input value for the data A signal of the complex multiplier. The size of the input signal depends on the How wide should the A input buses be? parameter value.
<code>dataa_real[]</code>	Yes	Real input value for the data A signal of the complex multiplier. The size of the input signal depends on the How wide should the A input buses be? parameter value.
<code>datab_imag[]</code>	Yes	Imaginary input value for the data B signal of the complex multiplier. The size of the input signal depends on the How wide should the B input buses be? parameter value.
<code>datab_real[]</code>	Yes	Real input value for the data B signal of the complex multiplier. The size of the input signal depends on the How wide should the B input buses be? parameter value.
<code>ena</code>	No	Active high clock enable for the clock signal of the complex multiplier.

Table 23. ALTMULT_COMPLEX Output Signals

Signal	Required	Description
<code>result_imag</code>	Yes	Imaginary output value of the multiplier. The size of the output signal depends on the <code>WIDTH_RESULT</code> parameter value.
<code>result_real</code>	Yes	Real output value of the multiplier. The size of the output signal depends on the <code>WIDTH_RESULT</code> parameter value.



8. MAX 10 Embedded Multipliers User Guide Archives

For the latest and previous versions of this user guide, refer to [MAX 10 Embedded Multipliers User Guide](#). If an IP or software version is not listed, the user guide for the previous IP or software version applies.

9. Document Revision History for the MAX 10 Embedded Multipliers User Guide

Document Version	Changes
2024.03.08	Made minor editorial updates.

Date	Version	Changes
February 2017	2017.02.021	<ul style="list-style-type: none"> Rebranded as Intel.
May 2016	2016.05.02	<ul style="list-style-type: none"> Updated MAX 10 to each chapter in the user guide. Added MAX 10 Embedded Multipliers User Guide Archives chapter.
November 2015	2015.11.02	<ul style="list-style-type: none"> Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i>. Removed topics on generating IP cores and added links to Introduction to Altera IP Cores, Creating Version-Independent IP and Qsys Simulation Scripts, and Project Management Best Practices.
September 2014	2014.09.22	Initial release.

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.

ISO
9001:2015
Registered



Intel® MAX® 10 Clocking and PLL User Guide

Updated for Intel® Quartus® Prime Design Suite: **21.1**



Online Version

Send Feedback

UG-M10CLKPLL

ID: **683047**

Version: **2021.11.01**

Contents

1. Intel® MAX® 10 Clocking and PLL Overview.....	4
1.1. Clock Networks Overview.....	4
1.2. Internal Oscillator Overview.....	4
1.3. PLLs Overview.....	4
2. Intel MAX 10 Clocking and PLL Architecture and Features.....	6
2.1. Clock Networks Architecture and Features.....	6
2.1.1. Global Clock Networks.....	6
2.1.2. Clock Pins Introduction.....	6
2.1.3. Clock Resources.....	7
2.1.4. Global Clock Network Sources.....	7
2.1.5. Global Clock Control Block.....	9
2.1.6. Global Clock Network Power Down.....	11
2.1.7. Clock Enable Signals.....	12
2.2. Internal Oscillator Architecture and Features.....	13
2.3. PLLs Architecture and Features.....	13
2.3.1. PLL Architecture.....	13
2.3.2. PLL Features.....	15
2.3.3. PLL Locations.....	15
2.3.4. Clock Pin to PLL Connections.....	17
2.3.5. PLL Counter to GCLK Connections.....	17
2.3.6. PLL Control Signals.....	18
2.3.7. Clock Feedback Modes.....	19
2.3.8. PLL External Clock Output.....	23
2.3.9. ADC Clock Input from PLL.....	24
2.3.10. Spread-Spectrum Clocking.....	24
2.3.11. PLL Programmable Parameters.....	24
2.3.12. Clock Switchover.....	27
2.3.13. PLL Cascading.....	31
2.3.14. PLL Reconfiguration.....	32
3. Intel MAX 10 Clocking and PLL Design Considerations.....	34
3.1. Clock Networks Design Considerations.....	34
3.1.1. Guideline: Clock Enable Signals.....	34
3.1.2. Guideline: Connectivity Restrictions.....	34
3.2. Internal Oscillator Design Considerations.....	34
3.2.1. Guideline: Connectivity Restrictions.....	34
3.3. PLLs Design Considerations.....	35
3.3.1. Guideline: PLL Control Signals.....	35
3.3.2. Guideline: Connectivity Restrictions.....	35
3.3.3. Guideline: Self-Reset.....	35
3.3.4. Guideline: Output Clocks.....	36
3.3.5. Guideline: PLL Cascading.....	36
3.3.6. Guideline: Clock Switchover.....	37
3.3.7. Guideline: .mif Streaming in PLL Reconfiguration.....	38
3.3.8. Guideline: scandone Signal for PLL Reconfiguration.....	38

4. Intel MAX 10 Clocking and PLL Implementation Guides.....	39
4.1. ALTCLKCTRL Intel FPGA IP Core.....	39
4.2. ALTPLL Intel FPGA IP Core.....	39
4.2.1. Expanding the PLL Lock Range.....	40
4.2.2. Programmable Bandwidth with Advanced Parameters.....	41
4.2.3. PLL Dynamic Reconfiguration Implementation.....	42
4.2.4. Dynamic Phase Configuration Implementation.....	46
4.3. ALTPLL_RECONFIG Intel FPGA IP Core.....	48
4.3.1. Obtaining the Resource Utilization Report.....	49
4.4. Internal Oscillator Intel FPGA IP Core.....	49
5. ALTCLKCTRL Intel FPGA IP Core References.....	50
5.1. ALTCLKCTRL IP Core Parameters.....	50
5.2. ALTCLKCTRL IP Core Ports and Signals.....	51
6. ALTPLL Intel FPGA IP Core References.....	52
6.1. ALTPLL IP Core Parameters.....	52
6.1.1. Operation Modes Parameter Settings.....	52
6.1.2. PLL Control Signals Parameter Settings.....	52
6.1.3. Programmable Bandwidth Parameter Settings.....	53
6.1.4. Clock Switchover Parameter Settings.....	53
6.1.5. PLL Dynamic Reconfiguration Parameter Settings.....	54
6.1.6. Dynamic Phase Configuration Parameter Settings.....	54
6.1.7. Output Clocks Parameter Settings.....	55
6.2. ALTPLL IP Core Ports and Signals.....	56
7. ALTPLL_RECONFIG Intel FPGA IP Core References.....	59
7.1. ALTPLL_RECONFIG IP Core Parameters.....	59
7.2. ALTPLL_RECONFIG IP Core Ports and Signals.....	60
7.3. ALTPLL_RECONFIG IP Core Counter Settings.....	62
8. Internal Oscillator Intel FPGA IP Core References.....	65
8.1. Internal Oscillator IP Core Parameters.....	65
8.2. Internal Oscillator IP Core Ports and Signals.....	65
9. Intel MAX 10 Clocking and PLL User Guide Archives.....	66
10. Document Revision History for the Intel MAX 10 Clocking and PLL User Guide.....	67

1. Intel® MAX® 10 Clocking and PLL Overview

1.1. Clock Networks Overview

Intel® MAX® 10 devices support global clock (GCLK) networks.

Clock networks provide clock sources for the core. You can use clock networks in high fan-out global signal network such as reset and clear.

1.2. Internal Oscillator Overview

Internal oscillators enable implementing designs that require clocking, thereby saving on-board space and costs associated with external clocking circuitry.

Intel MAX 10 devices offer built-in internal oscillator up to 116 MHz.

You can enable or disable the internal oscillator.

Related Information

[AN 496: Using the Internal Oscillator IP Core](#)

Provides more information about the internal oscillator.

1.3. PLLs Overview

Phase-locked loops (PLLs) provide robust clock management and synthesis for device clock management, external system clock management, and I/O interface clocking.

You can use the PLLs as follows:

- Zero-delay buffer
- Jitter attenuator
- Low-skew fan-out buffer
- Frequency synthesizer
- Reduce the number of oscillators required on the board
- Reduce the clock pins used in the device by synthesizing multiple clock frequencies from a single reference clock source
- On-chip clock de-skew
- Dynamic phase shift
- Counters reconfiguration
- Bandwidth reconfiguration
- Programmable output duty cycle



- PLL cascading
- Reference clock switchover
- Drive the analog-to-digital converter (ADC) clock



2. Intel MAX 10 Clocking and PLL Architecture and Features

2.1. Clock Networks Architecture and Features

2.1.1. Global Clock Networks

GCLKs drive throughout the entire device, feeding all device quadrants. All resources in the device, such as the I/O elements, logic array blocks (LABs), dedicated multiplier blocks, and M9K memory blocks can use GCLKs as clock sources. Use these clock network resources for control signals, such as clock enables and clears fed by an external pin. Internal logic can also drive GCLKs for internally-generated GCLKs and asynchronous clears, clock enables, or other control signals with high fan-out.

2.1.2. Clock Pins Introduction

There are two types of external clock pins that can drive the GCLK networks.

Dedicated Clock Input Pins

You can use the dedicated clock input pins ($CLK\langle\# \rangle[p, n]$) to drive clock and global signals, such as asynchronous clears, presets, and clock enables for GCLK networks.

If you do not use the dedicated clock input pins for clock input, you can also use them as general-purpose input or output pins.

The CLK pins can be single-ended or differential inputs. When you use the CLK pins as single-ended clock inputs, both the $CLK\langle\# \rangle p$ and $CLK\langle\# \rangle n$ pins have dedicated connection to the GCLK networks. When you use the CLK pins as differential inputs, pair two clock pins of the same number to receive differential signaling.

Dual-Purpose Clock Pins

You can use the dual-purpose clock (DPCLK) pins for high fan-out control signals, such as protocol signals, TRDY and IRDY signals for PCI via GCLK networks.

The DPCLK pins are only available on the left and right of the I/O banks.

2.1.3. Clock Resources

Table 1. Intel MAX 10 Clock Resources

Clock Resource	Device	Number of Resources Available	Source of Clock Resource
Dedicated clock input pins	<ul style="list-style-type: none"> 10M02 10M04 10M08 	8 single-ended or 4 differential	CLK[3..0][p,n] pins on the left and right of the I/O banks
	<ul style="list-style-type: none"> 10M16 10M25 10M40 10M50 	16 single-ended or 8 differential	CLK[7..0][p,n] pins on the top, left, bottom, and right of the I/O banks
DPCLK pins	All	4	DPCLK[3..0] pins on the left and right of the I/O banks

For more information about the clock input pins connections, refer to the pin connection guidelines.

Related Information

[Intel MAX 10 FPGA Device Family Pin Connection Guidelines](#)

2.1.4. Global Clock Network Sources

Table 2. Intel MAX 10 Clock Pins Connectivity to the GCLK Networks

CLK Pin	GCLK
CLK0p	GCLK[0,2,4]
CLK0n	GCLK[1,2]
CLK1p	GCLK[1,3,4]
CLK1n	GCLK[0,3]
CLK2p	GCLK[5,7,9]
CLK2n	GCLK[6,7]
CLK3p	GCLK[6,8,9]
CLK3n	GCLK[5,8]
CLK4p ⁽¹⁾	GCLK[10,12,14]
CLK4n ⁽¹⁾	GCLK[11,12]
CLK5p ⁽¹⁾	GCLK[11,13,14]
CLK5n ⁽¹⁾	GCLK[10,13]
CLK6p ⁽¹⁾	GCLK[15,17,19]
CLK6n ⁽¹⁾	GCLK[16,17]
<i>continued...</i>	

⁽¹⁾ This only applies to 10M16, 10M25, 10M40, and 10M50 devices.

CLK Pin	GCLK
CLK7 _p ⁽¹⁾	GCLK[16,18,19]
CLK7 _n ⁽¹⁾	GCLK[15,18]
DPCLK0	GCLK[0,2]
DPCLK1	GCLK[1,3,4]
DPCLK2	GCLK[5,7]
DPCLK3	GCLK[6,8,9]

Figure 1. GCLK Network Sources for 10M02, 10M04, and 10M08 Devices

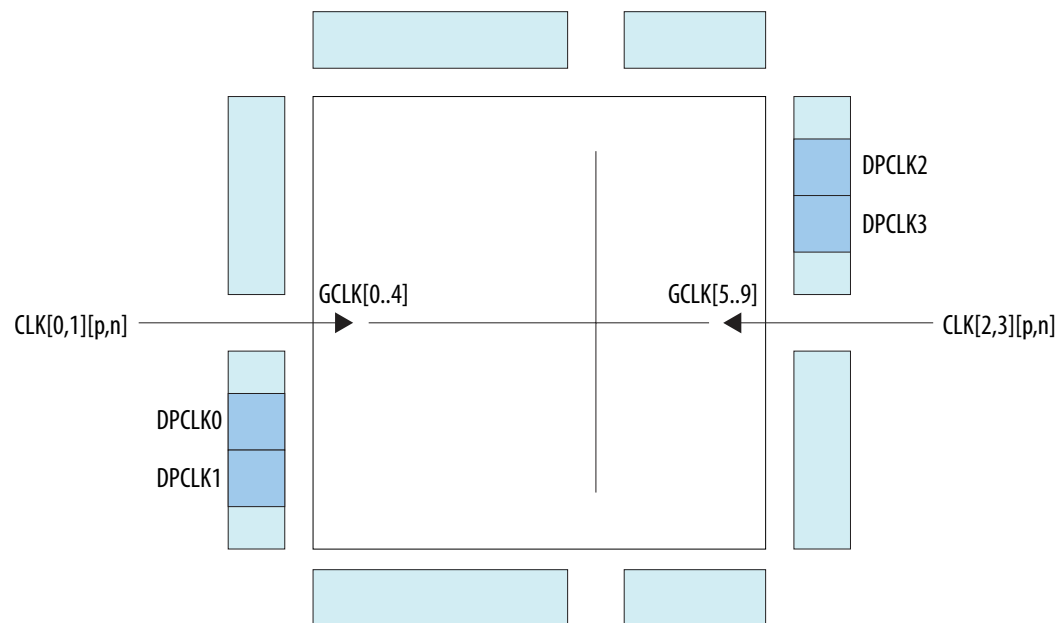
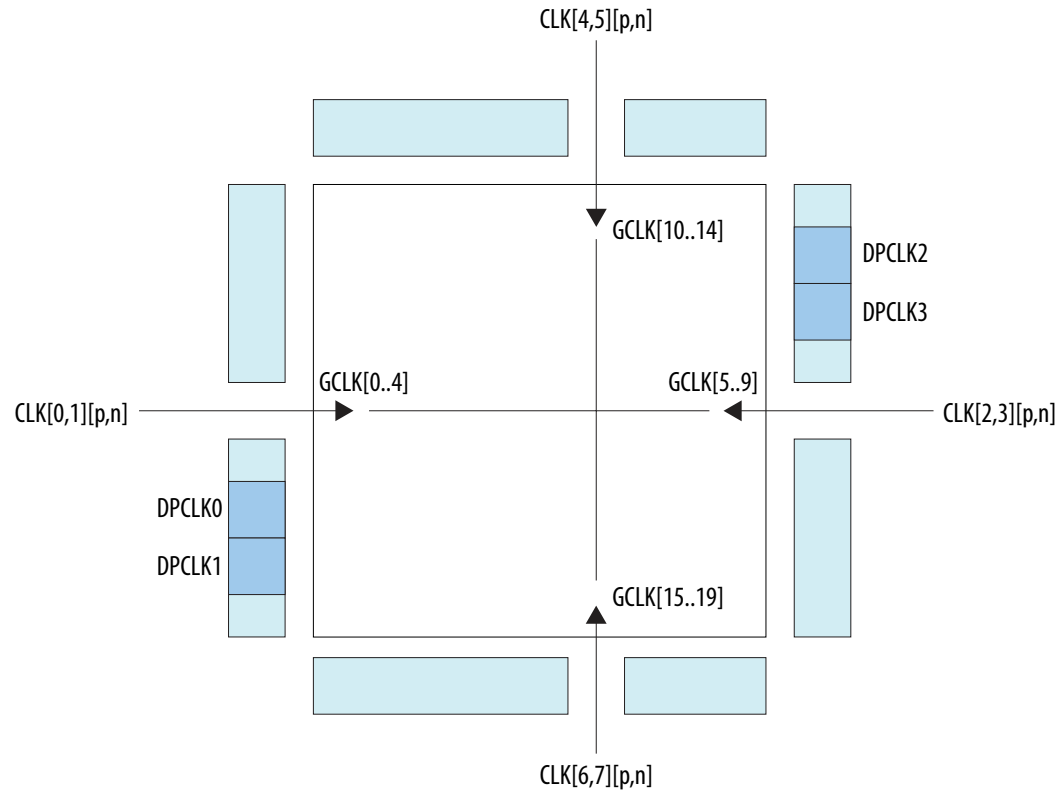


Figure 2. GCLK Network Sources for 10M16, 10M25, 10M40, and 10M50 Devices



2.1.5. Global Clock Control Block

The clock control block drives GCLKs. The clock control blocks are located on each side of the device, close to the dedicated clock input pins. GCLKs are optimized for minimum clock skew and delay.

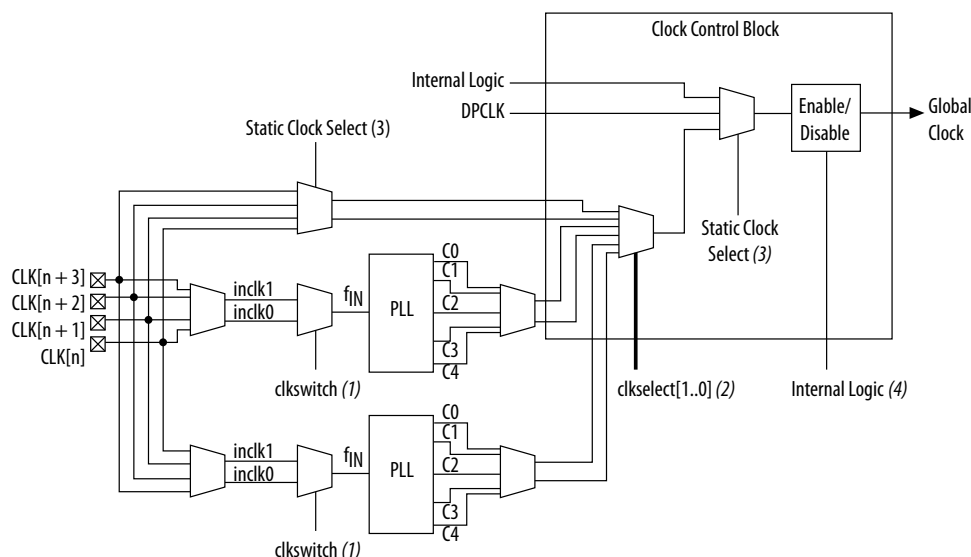
The clock control block has the following functions:

- Dynamic GCLK clock source selection (not applicable for DPCLK pins and internal logic input)
- GCLK multiplexing
- GCLK network power down (dynamic enable and disable)

Table 3. Clock Control Block Inputs

Input	Description
Dedicated clock input pins	Dedicated clock input pins can drive clocks or global signals, such as synchronous and asynchronous clears, presets, or clock enables onto given GCLKs.
DPCLK pins	DPCLK pins are bidirectional dual function pins that are used for high fan-out control signals, such as protocol signals, TRDY and IRDY signals for PCI via the GCLK. Clock control blocks that have inputs driven by DPCLK pins cannot drive PLL inputs.
PLL counter outputs	PLL counter outputs can drive the GCLK.
Internal logic	You can drive the GCLK through logic array routing to enable the internal logic elements (LEs) to drive a high fan-out, low-skew signal path. Clock control blocks that have inputs driven by internal logic cannot drive PLL inputs.

Figure 3. Clock Control Block



Notes:

- (1) The clkswitch signal can either be set through the configuration file or dynamically set when using the manual PLL switchover feature. The output of the multiplexer is the input clock (fIN) for the PLL.
- (2) The clkselect[1..0] signals are fed by internal logic. You can use the clkselect[1..0] signals to dynamically select the clock source for the GCLK when the device is in user mode. Only one PLL (applicable to PLLs on the same side) can be selected as the clock source to the GCLK.
- (3) The static clock select signals are set in the configuration file. Therefore, dynamic control when the device is in user mode is not feasible.
- (4) You can use internal logic to enable or disable the GCLK in user mode.

Each Intel MAX 10 device has a maximum of 20 clock control blocks. There are five clock control blocks on each side of the device.

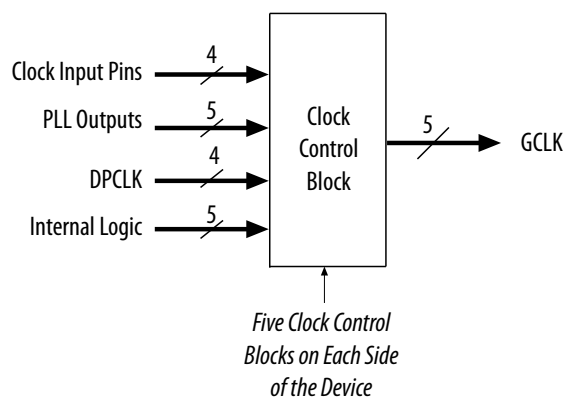
Each PLL generates five clock outputs through the $c[4..0]$ counters. Two of these clocks can drive the GCLK through a clock control block.

From the Clock Control Block Inputs table, only the following inputs can drive into any given clock control block:

- Two dedicated clock input pins
- Two PLL counter outputs
- One DPCLK pin
- One source from internal logic

The output from the clock control block in turn feeds the corresponding GCLK. The GCLK can drive the PLL input if the clock control block inputs are outputs of another PLL or dedicated clock input pins. Normal I/O pins cannot drive the PLL input clock port.

Figure 4. Clock Control Block on Each Side of the Device



Out of these five inputs to any clock control block, the two clock input pins and two PLL outputs are dynamically selected to feed a GCLK. The clock control block supports static selection of the signal from internal logic.

Related Information

- [ALTCLKCTRL IP Core Parameters](#) on page 50
- [ALTCLKCTRL IP Core Ports and Signals](#) on page 51

2.1.6. Global Clock Network Power Down

You can disable the Intel MAX 10 GCLK (power down) by using both static and dynamic approaches. In the static approach, configuration bits are set in the configuration file generated by the Intel Quartus® Prime software, which automatically disables unused GCLKs. The dynamic clock enable or disable feature allows internal logic to control clock enable or disable of the GCLKs.

When a clock network is disabled, all the logic fed by the clock network is in an off-state, reducing the overall power consumption of the device. This function is independent of the PLL and is applied directly on the clock network.

You can set the input clock sources and the clock enable (`clkena`) signals for the GCLK multiplexers through the ALTCLKCTRL Intel FPGA IP core parameter editor in the Intel Quartus Prime software.

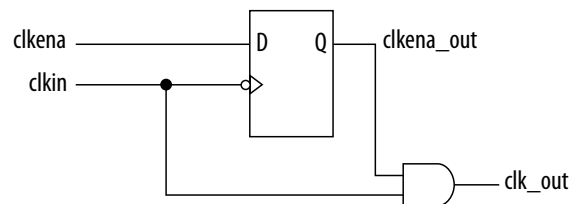
Related Information

- [ALTCLKCTRL IP Core Parameters](#) on page 50
- [ALTCLKCTRL IP Core Ports and Signals](#) on page 51

2.1.7. Clock Enable Signals

The Intel MAX 10 devices support `clkena` signals at the GCLK network level. This allows you to gate off the clock even when a PLL is used. After reenabling the output clock, the PLL does not need a resynchronization or relock period because the circuit gates off the clock at the clock network level. In addition, the PLL can remain locked independent of the `clkena` signals because the loop-related counters are not affected.

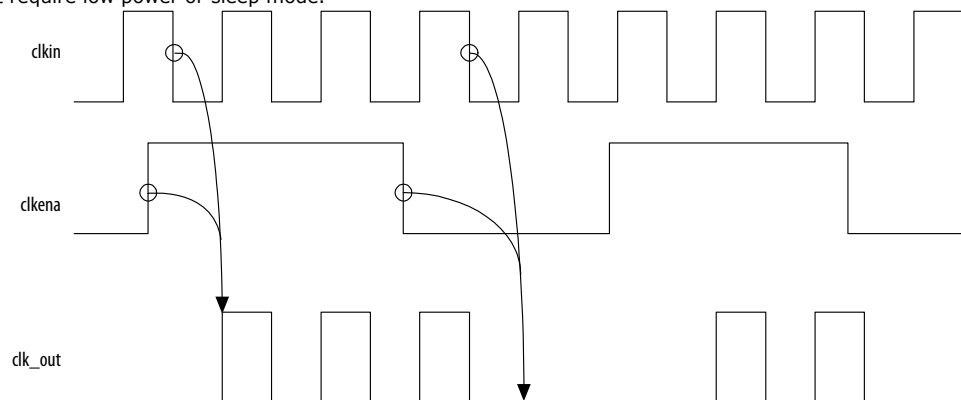
Figure 5. clkena Implementation



Note: The `clkena` circuitry controlling the C0 output of the PLL to an output pin is implemented with two registers instead of a single register.

Figure 6. Example Waveform of clkena Implementation with Output Enable

The `clkena` signal is sampled on the falling edge of the clock (`clkkin`). This feature is useful for applications that require low power or sleep mode.



The `clkena` signal can also disable clock outputs if the system is not tolerant to frequency overshoot during PLL resynchronization.

Related Information

- [Guideline: Clock Enable Signals](#) on page 34
- [ALTCLKCTRL IP Core Parameters](#) on page 50
- [ALTCLKCTRL IP Core Ports and Signals](#) on page 51

2.2. Internal Oscillator Architecture and Features

Intel MAX 10 devices have built-in internal ring oscillator with clock multiplexers and dividers. The internal ring oscillator operates up to 232 MHz which is not accessible. This operating frequency further divides down to slower frequencies.

By default internal oscillator is turned off in user mode. You can turn on the oscillator by asserting the `oscena` signal in the Internal Oscillator Intel FPGA IP core.

When the `oscena` input signal is asserted, the oscillator is enabled and the output can be routed to the logic array through the `clkout` output signal. When the `oscena` signal is set low, the `clkout` signal is constant high. You can analyze this delay using the Timing Analyzer.

Related Information

[AN 496: Using the Internal Oscillator IP Core](#)

Provides more information about the internal oscillator.

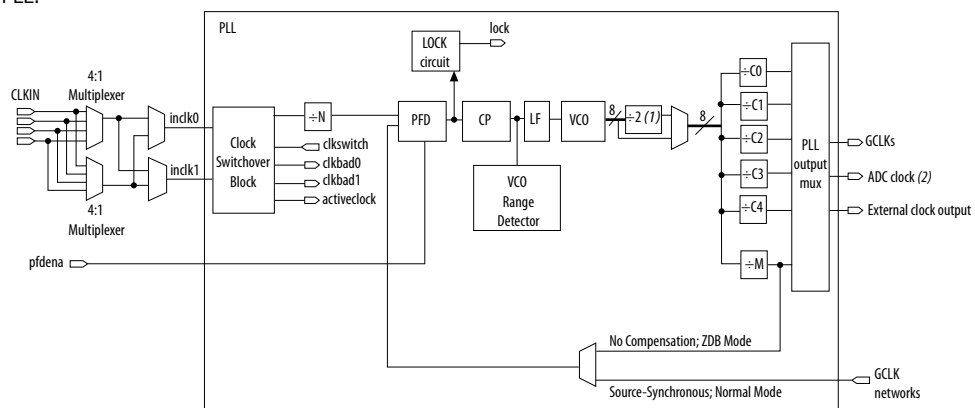
2.3. PLLs Architecture and Features

2.3.1. PLL Architecture

The main purpose of a PLL is to synchronize the phase and frequency of the voltage-controlled oscillator (VCO) to an input reference clock.

Figure 7. Intel MAX 10 PLL High-Level Block Diagram

Each clock source can come from any of the two or four clock pins located on the same side of the device as the PLL.



Notes:

- (1) This is the VCO post-scale counter K.
- (2) Only counter C0 of PLL1 and PLL3 can drive the ADC clock.

Phase-Frequency Detector (PFD)

The PFD has inputs from the feedback clock, f_{FB} , and the input reference clock, f_{REF} . The PLL compares the rising edge of the input reference clock to a feedback clock using a PFD. The PFD produces an up or down signal that determines whether the VCO needs to operate at a higher or lower frequency.

Charge Pump (CP)

If the charge pump receives a logic high on the up signal, current is driven into the loop filter. If the charge pump receives a logic high on the down signal, current is drawn from the loop filter.

Loop Filter (LF)

The loop filter converts the up and down signals from the PFD to a voltage that is used to bias the VCO. The loop filter filters out glitches from the charge pump and prevents voltage overshoot, which minimizes jitter on the VCO.

Voltage-Controlled Oscillator (VCO)

The voltage from the charge pump determines how fast the VCO operates. The VCO is implemented as a four-stage differential ring oscillator. A divide counter, M , is inserted in the feedback loop to increase the VCO frequency, f_{VCO} , above the input reference frequency, f_{REF} .

The VCO frequency is determined using the following equation:

$$f_{VCO} = f_{REF} \times M = f_{IN} \times M/N,$$

where f_{IN} is the input clock frequency to the PLL and N is the pre-scale counter.

The VCO frequency is a critical parameter that must be between 600 and 1,300 MHz to ensure proper operation of the PLL. The Intel Quartus Prime software automatically sets the VCO frequency within the recommended range based on the clock output and phase shift requirements in your design.

Post-Scale Counters (C)

The VCO output can feed up to five post-scale counters (C0, C1, C2, C3, and C4). These post-scale counters allow the PLL to produce a number of harmonically-related frequencies.

Internal Delay Elements

The Intel MAX 10 PLLs have internal delay elements to compensate for routing on the GCLK networks and I/O buffers. These internal delays are fixed.

PLL Outputs

The Intel MAX 10 PLL supports up to 5 GCLK outputs and 1 dedicated external clock output. The output frequency, f_{OUT} , to the GCLK network or dedicated external clock output is determined using the following equation:

$$f_{REF} = f_{IN}/N \text{ and}$$

$$f_{OUT} = f_{VCO}/C = (f_{REF} \times M)/C = (f_{IN} \times M)/(N \times C),$$

where C is the setting on the C0, C1, C2, C3, or C4 counter.

2.3.2. PLL Features

Table 4. Intel MAX 10 PLL Features

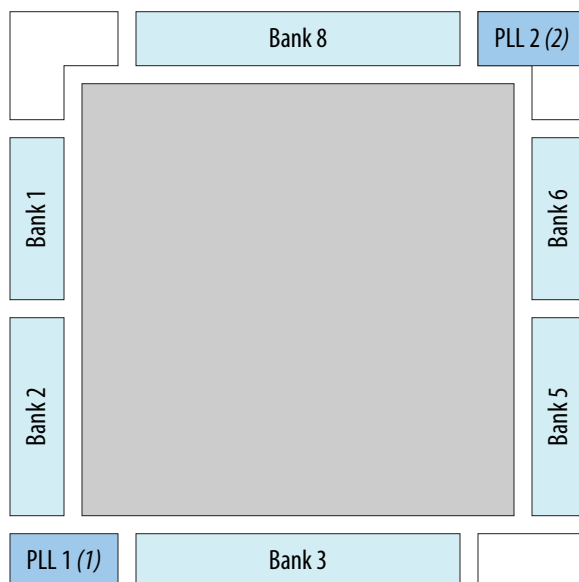
Feature	Support
C output counters	5
M, N, C counter sizes	1 to 512 ⁽²⁾
Dedicated clock outputs	1 single-ended or 1 differential
Dedicated clock input pins	4 single-ended or 2 differential
Spread-spectrum input clock tracking	Yes ⁽³⁾
PLL cascading	Through GCLK
Source synchronous compensation	Yes
No compensation mode	Yes
Normal compensation	Yes
Zero-delay buffer compensation	Yes
Phase shift resolution	Down to 96 ps increments ⁽⁴⁾
Programmable duty cycle	Yes
Output counter cascading	Yes
Input clock switchover	Yes
User mode reconfiguration	Yes
Loss of lock detection	Yes
4:1 multiplexer CLK input selection	Yes

2.3.3. PLL Locations

The following figures show the physical locations of the PLLs. Every index represents one PLL in the device. The physical locations of the PLLs correspond to the coordinates in the Intel Quartus Prime Chip Planner.

-
- ⁽²⁾ C counters range from 1 through 512 if the output clock uses a 50% duty cycle. For any output clocks using a non-50% duty cycle, the post-scale counters range from 1 through 256.
- ⁽³⁾ Only applicable if the input clock jitter is in the input jitter tolerance specifications.
- ⁽⁴⁾ The smallest phase shift is determined by the VCO period divided by eight. For degree increments, the Intel MAX 10 device family can shift all output frequencies in increments of at least 45°. Smaller degree increments are possible depending on the frequency and divide parameters.

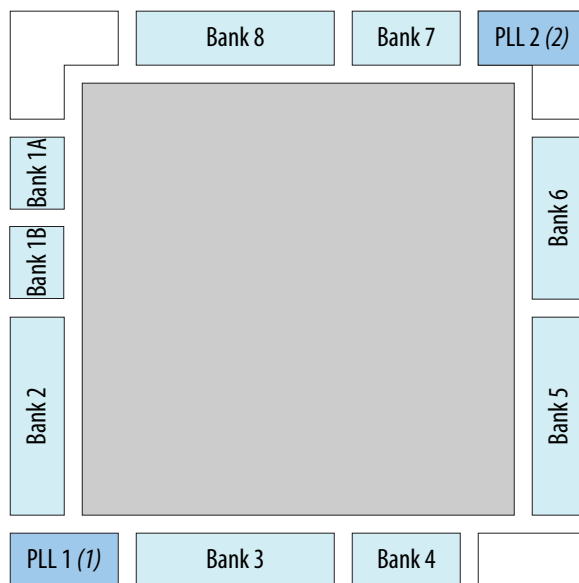
Figure 8. PLL Locations for 10M02 Device (Except Single Power Supply U324 Package)



Notes:

- (1) Available on all packages except V36 package.
- (2) Available on U324 and V36 packages only.

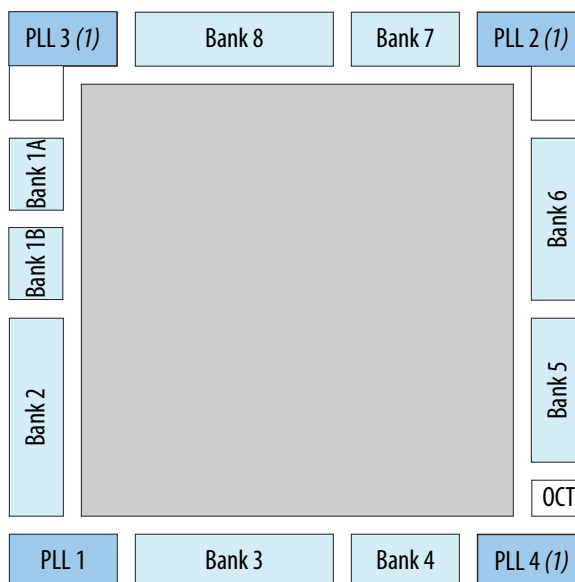
Figure 9. PLL Locations for 10M02 (Single Power Supply U324 Package), 10M04, and 10M08 Devices



Notes:

- (1) Available on all packages except V81 package.
- (2) Available on F256, F484, U324 (dual power supply), and V81 packages only.

Figure 10. PLL Locations for 10M16, 10M25, 10M40, and 10M50 Devices



Note:

(1) Available on all packages except E144, U169, Y180, and U324 (single power supply) packages.

2.3.4. Clock Pin to PLL Connections

Table 5. Intel MAX 10 Dedicated Clock Input Pin Connectivity to PLL

Dedicated Clock Pin	PLL
CLK[0,1][p,n]	PLL1, PLL3
CLK[2,3][p,n]	PLL2, PLL4
CLK[4,5][p,n]	PLL2, PLL3
CLK[6,7][p,n]	PLL1, PLL4

2.3.5. PLL Counter to GCLK Connections

Table 6. Intel MAX 10 PLL Counter Connectivity to the GCLK Networks

PLL Counter Output	GCLK
PLL1_C0	GCLK[0,3,15,18]
PLL1_C1	GCLK[1,4,16,19]
PLL1_C2	GCLK[0,2,15,17]
PLL1_C3	GCLK[1,3,16,18]
PLL1_C4	GCLK[2,4,17,19]
PLL2_C0	GCLK[5,8,10,13]

continued...

PLL Counter Output	GCLK
PLL2_C1	GCLK[6,9,11,14]
PLL2_C2	GCLK[5,7,10,12]
PLL2_C3	GCLK[6,8,11,13]
PLL2_C4	GCLK[7,9,12,14]
PLL3_C0 ⁽⁵⁾	GCLK[0,3,10,13]
PLL3_C1 ⁽⁵⁾	GCLK[1,4,11,14]
PLL3_C2 ⁽⁵⁾	GCLK[0,2,10,12]
PLL3_C3 ⁽⁵⁾	GCLK[1,3,11,13]
PLL3_C4 ⁽⁵⁾	GCLK[2,4,12,14]
PLL4_C0 ⁽⁵⁾	GCLK[5,8,15,18]
PLL4_C1 ⁽⁵⁾	GCLK[6,9,16,19]
PLL4_C2 ⁽⁵⁾	GCLK[5,7,15,17]
PLL4_C3 ⁽⁵⁾	GCLK[6,8,16,18]
PLL4_C4 ⁽⁵⁾	GCLK[7,9,17,19]

2.3.6. PLL Control Signals

You can use the following three signals to observe and control the PLL operation and resynchronization.

pfdena

Use the `pfdena` signal to maintain the last locked frequency so that your system has time to store its current settings before shutting down.

The `pfdena` signal controls the PFD output with a programmable gate. The PFD circuit is enabled by default. When the PFD circuit is disabled, the PLL output does not depend on the input clock, and tends to drift outside of the lock window.

areset

The `areset` signal is the reset or resynchronization input for each PLL. The device input pins or internal logic can drive these input signals.

When you assert the `areset` signal, the PLL counters reset, clearing the PLL output and placing the PLL out of lock. The VCO is then set back to its nominal setting. When the `areset` signal is deasserted, the PLL resynchronizes to its input as it rellocks.

The assertion of the `areset` signal does not disable the VCO, but instead resets the VCO to its nominal value. The only time that the VCO is completely disabled is when you do not have a PLL instantiated in your design.

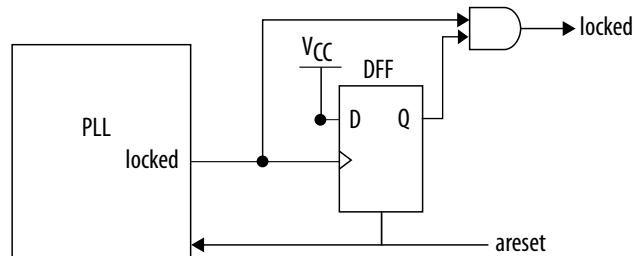
⁽⁵⁾ This only applies to 10M16, 10M25, 10M40, and 10M50 devices.

locked

The `locked` output indicates that the PLL has locked onto the reference clock and the PLL clock outputs are operating at the desired phase and frequency set in the ALTPLL Intel FPGA IP core parameter editor.

Intel recommends using the `areset` and `locked` signals in your designs to control and observe the status of your PLL. This implementation is illustrated in the following figure.

Figure 11. locked Signal Implementation



Note: If you use the Signal Tap II tool to probe the `locked` signal before the D flip-flop, the `locked` signal goes low only when `areset` is deasserted. If the `areset` signal is not enabled, the extra logic is not implemented in the ALTPLL IP core.

Related Information

- [Guideline: PLL Control Signals](#) on page 35
- [PLL Control Signals Parameter Settings](#) on page 52
- [ALTPLL IP Core Ports and Signals](#) on page 56

2.3.7. Clock Feedback Modes

The Intel MAX 10 PLLs support up to four different clock feedback modes. Each mode allows clock multiplication and division, phase shifting, and programmable duty cycle.

The PLL fully compensates input and output delays only when you use the dedicated clock input pins associated with a given PLL as the clock sources.

For example, when using `PLL1` in normal mode, the clock delays from one of the following clock input pins to the PLL and the PLL clock output-to-destination register are fully compensated:

- `CLK0`
- `CLK1`
- `CLK6`
- `CLK7`

When driving the PLL using the GCLK network, the input and output delays might not be fully compensated in the Intel Quartus Prime software.

Related Information

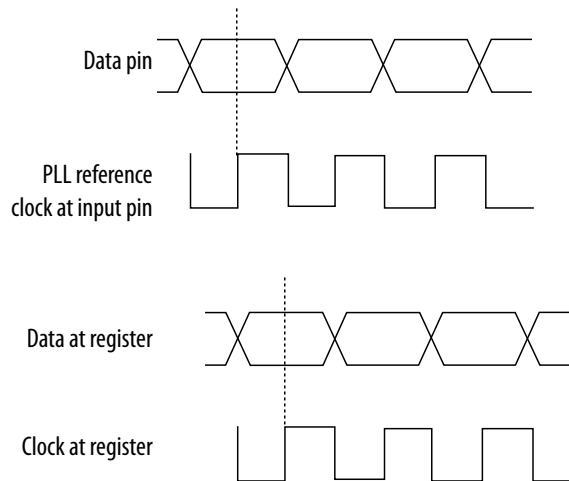
[Operation Modes Parameter Settings](#) on page 52

2.3.7.1. Source Synchronous Mode

If the data and clock arrive at the same time at the input pins, the phase relationship between the data and clock remains the same at the data and clock ports of any I/O element input register.

You can use this mode for source synchronous data transfers. Data and clock signals at the I/O element experience similar buffer delays as long as both signals use the same I/O standard.

Figure 12. Example of Phase Relationship Between Clock and Data in Source Synchronous Mode



Source synchronous mode compensates for clock network delay, including any difference in delay between the following two paths:

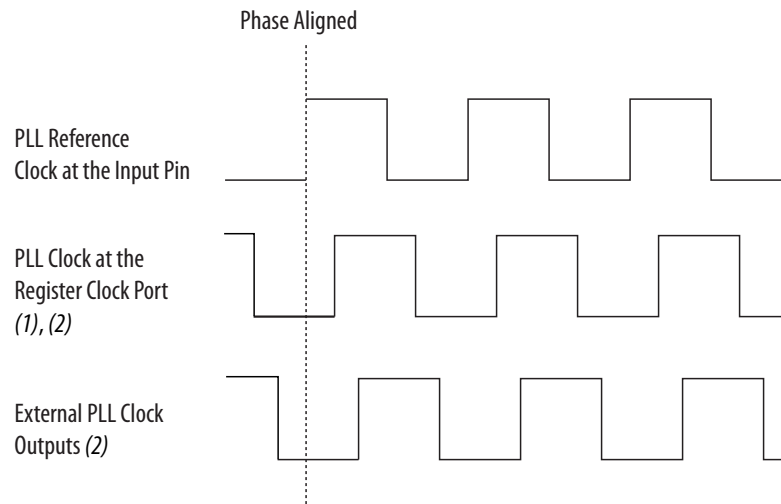
- Data pin to I/O element register input
- Clock input pin to the PLL PFD input

For all data pins clocked by a source synchronous mode PLL, set the input pin to the register delay chain in the I/O element to zero in the Intel Quartus Prime software. All data pins must use the **PLL COMPENSATED logic** option in the Intel Quartus Prime software.

2.3.7.2. No Compensation Mode

In no compensation mode, the PLL does not compensate for any clock networks. This mode provides better jitter performance because clock feedback into the PFD does not pass through as much circuitry. Both the PLL internal and external clock outputs are phase-shifted with respect to the PLL clock input.

Figure 13. Example of Phase Relationship Between the PLL Clocks in No Compensation Mode



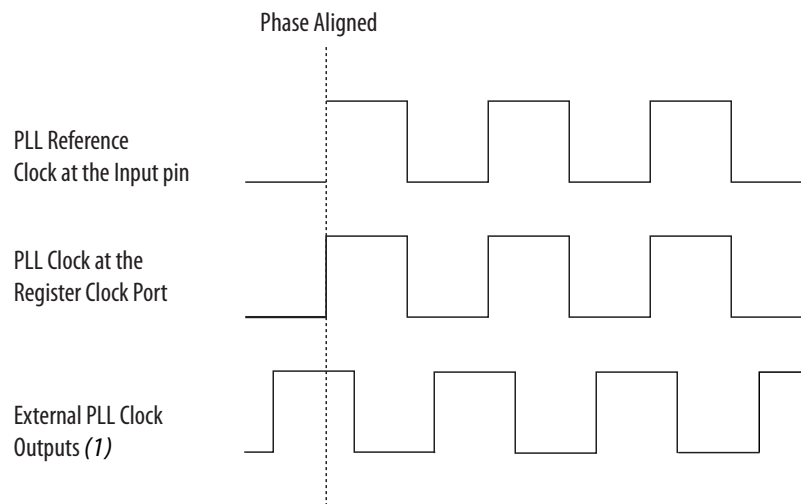
Notes:

- (1) Internal clocks fed by the PLL are phase-aligned to each other.
- (2) The PLL clock outputs can lead or lag the PLL input clocks. The PLL clock outputs lag the PLL input clocks depending on the routine delays.

2.3.7.3. Normal Mode

In normal mode, the PLL fully compensates the delay introduced by the GCLK network. An internal clock in normal mode is phase-aligned to the input clock pin. In this mode, the external clock output pin has a phase delay relative to the input clock pin. The Intel Quartus Prime software Timing Analyzer reports any phase difference between the two.

Figure 14. Example of Phase Relationship Between the PLL Clocks in Normal Compensation Mode



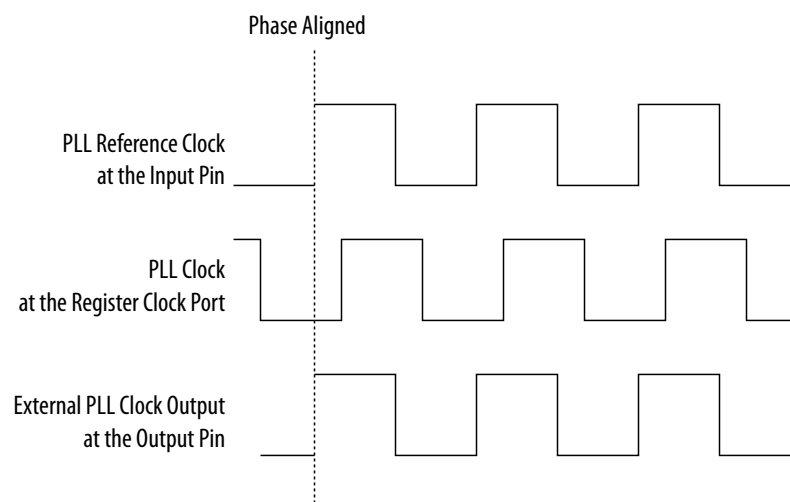
Note:

(1) The external clock output can lead or lag the PLL internal clock signals.

2.3.7.4. Zero-Delay Buffer Mode

In zero-delay buffer (ZDB) mode, the external clock output pin is phase-aligned with the clock input pin for zero delay through the device. When using this mode, use the same I/O standard for the input clock and output clocks to ensure clock alignment at the input and output pins. PLL_CLKOUTn pin is not supported for single-ended I/O standard in this mode.

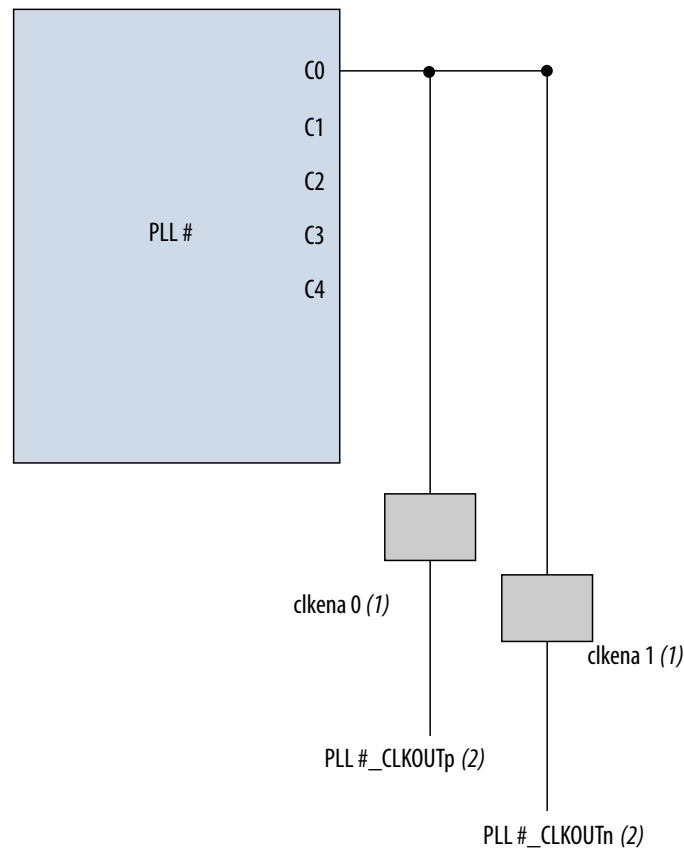
Figure 15. Example of Phase Relationship Between the PLL Clocks in ZDB Mode



2.3.8. PLL External Clock Output

Each PLL in the Intel MAX 10 devices supports one single-ended clock output or one differential clock output. Only the C0 output counter can feed the dedicated external clock outputs without going through the GCLK. C0 and all other output counters can feed other I/O pins through the GCLK.

Figure 16. PLL External Clock Output



Notes:

- (1) These external clock enable signals are available only when using the ALTCLKCTRL IP core.
- (2) PLL#_CLKOUTp and PLL#_CLKOUTn pins are dual-purpose I/O pins that you can use as one single-ended or one differential clock output.

Each pin of a differential output pair is 180° out of phase. To implement the 180° out-of-phase pin in a pin pair, the Intel Quartus Prime software places a NOT gate in the design into the I/O element.

The clock output pin pairs support the following I/O standards:

- Same I/O standard as the standard output pins (in the top and bottom banks)
- LVDS
- LVPECL
- Differential high-speed transceiver logic (HSTL)
- Differential SSTL

The Intel MAX 10 PLLs can drive out to any regular I/O pin through the GCLK. You can also use the external clock output pins as general-purpose I/O pins if you do not require any external PLL clocking.

Related Information

[Intel MAX 10 General Purpose I/O User Guide](#)

Provides more information about the I/O standards supported by the PLL clock output pins.

2.3.9. ADC Clock Input from PLL

Only the C0 output counter from PLL1 and PLL3 can drive the ADC clock.

Counter C0 has dedicated path to the ADC clock input.

2.3.10. Spread-Spectrum Clocking

The Intel MAX 10 devices allow a spread-spectrum input with typical modulation frequencies. However, the device cannot automatically detect that the input is a spread-spectrum signal. Instead, the input signal looks like deterministic jitter at the input of the PLL.

The Intel MAX 10 PLLs can track a spread-spectrum input clock if the input signal meets the following conditions:

- The input signal is within the input jitter tolerance specifications.
- The modulation frequency of the input clock is below the PLL bandwidth as specified in the Fitter report.

Intel MAX 10 devices cannot generate spread-spectrum signals internally. You must generate the spread-spectrum signals externally.

2.3.11. PLL Programmable Parameters

2.3.11.1. Programmable Duty Cycle

The programmable duty cycle allows PLLs to generate clock outputs with a variable duty cycle. This feature is supported on the PLL post-scale counters.

The duty cycle setting is achieved by a low and high time-count setting for the post-scale counters. To determine the duty cycle choices, the Intel Quartus Prime software uses the frequency input and the required multiply or divide rate.

The post-scale counter value determines the precision of the duty cycle. The precision is defined as 50% divided by the post-scale counter value. For example, if the C0 counter is 10, steps of 5% are possible for duty cycle choices between 5 to 90%.

Combining the programmable duty cycle with programmable phase shift allows the generation of precise nonoverlapping clocks.

Related Information

[Post-Scale Counters \(C0 to C4\)](#) on page 42

Provides more information about configuring the duty cycle of the post-scale counters in real time.

2.3.11.2. Programmable Bandwidth

The PLL bandwidth is the measure of the PLL's ability to track the input clock and its associated jitter. The Intel MAX 10 PLLs provide advanced control of the PLL bandwidth using the programmable characteristics of the PLL loop, including loop filter and charge pump. The 3-dB frequency of the closed-loop gain in the PLL determines the PLL bandwidth. The bandwidth is approximately the unity gain point for open loop PLL response.

Related Information

- [Programmable Bandwidth with Advanced Parameters](#) on page 41
- [Charge Pump and Loop Filter](#) on page 44
Provides more information about the PLL components to update PLL bandwidth in real time.
- [Programmable Bandwidth Parameter Settings](#) on page 53

2.3.11.3. Programmable Phase Shift

The Intel MAX 10 devices use phase shift to implement clock delays. You can phase shift the output clocks from the Intel MAX 10 PLLs using one of the following methods:

- Fine resolution using VCO phase taps
- Coarse resolution using counter starting time

The VCO phase output and counter starting time are the most accurate methods of inserting delays. These methods are purely based on counter settings, which are independent of process, voltage, and temperature.

The Intel MAX 10 devices support dynamic phase shifting of VCO phase taps only. The phase shift is configurable for any number of times. Each phase shift takes about one `scanclk` cycle, allowing you to implement large phase shifts quickly.

Fine Resolution Phase Shift

Fine resolution phase shifts are implemented by allowing any of the output counters (C[4..0]) or the M counter to use any of the eight phases of the VCO as the reference clock. This allows you to adjust the delay time with a fine resolution. The following equation shows the minimum delay time that you can insert using this method.

Figure 17. Fine Resolution Phase Shift Equation

f_{REF} in this equation is the input reference clock frequency

$$\Phi_{fine} = \frac{T_{VCO}}{8} = \frac{1}{8f_{VCO}} = \frac{1}{8} \times \frac{N}{M \times f_{REF}}$$

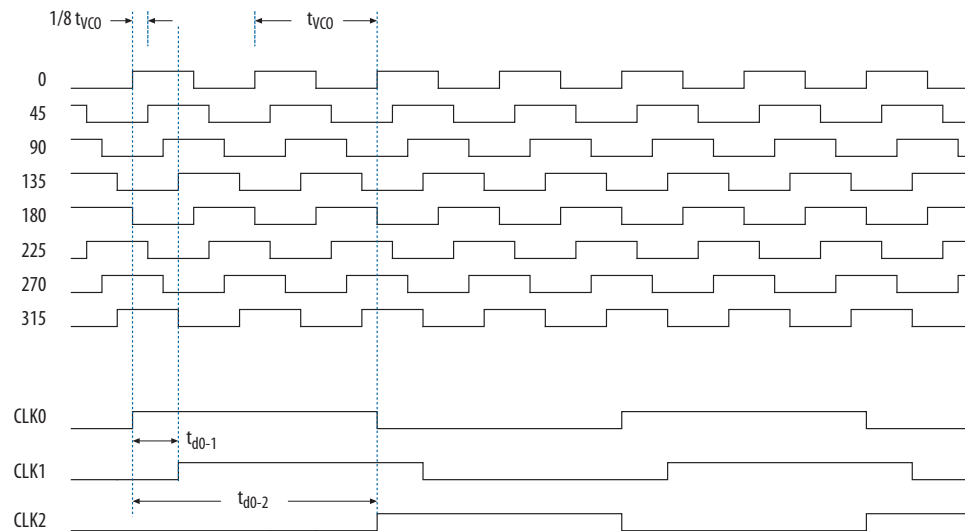
For example, if f_{REF} is 100 MHz, $N = 1$, and $M = 8$, then $f_{VCO} = 800$ MHz, and $\Phi_{fine} = 156.25$ ps. The PLL operating frequency defines this phase shift, a value that depends on the reference clock frequency and counter settings.

The following figure shows an example of phase shift insertion using the fine resolution through VCO phase taps method. The eight phases from the VCO are shown and labeled for reference.

Figure 18. Example of Delay Insertion Using VCO Phase Output and Counter Delay Time

The observations in this example are as follows:

- CLK0 is based on 0° phase from the VCO and has the C value for the counter set to one.
- CLK1 signal is divided by four, two VCO clocks for high time and two VCO clocks for low time. CLK1 is based on the 135° phase tap from the VCO and has the C value for the counter set to one.
- CLK2 signal is also divided by four. In this case, the two clocks are offset by $3\Phi_{fine}$. CLK2 is based on the 0° phase from the VCO but has the C value for the counter set to three. This creates a delay of two Φ_{coarse} (two complete VCO periods).



Coarse Resolution Phase Shift

Coarse resolution phase shifts are implemented by delaying the start of the counters for a predetermined number of counter clocks.

Figure 19. Coarse Resolution Phase Shift Equation

C in this equation is the count value set for the counter delay time—the initial setting in the PLL usage section of the compilation report in the Intel Quartus Prime software. If the initial value is 1, $C - 1 = 0^\circ$ phase shift.

$$\Phi_{coarse} = \frac{C - 1}{f_{VCO}} = \frac{(C - 1)N}{Mf_{REF}}$$

Related Information

- [Dynamic Phase Configuration Implementation](#) on page 46
- [Dynamic Phase Configuration Counter Selection](#) on page 47
- [Dynamic Phase Configuration with Advanced Parameters](#) on page 47

- [Dynamic Phase Configuration Parameter Settings](#) on page 54
Provides more information about the ALTPLL IP core parameter settings in the Intel Quartus Prime software.
- [ALTPLL_RECONFIG IP Core Parameters](#) on page 59
Provides more information about the ALTPLL_RECONFIG IP core parameter settings in the Intel Quartus Prime software.

2.3.12. Clock Switchover

The clock switchover feature allows the PLL to switch between two reference input clocks. Use this feature for clock redundancy or for a dual-clock domain application where a system turns on the redundant clock if the previous clock stops running. The design can perform clock switchover automatically when the clock is no longer toggling or based on a user-controlled signal, `clkswitch`.

The following clock switchover modes are supported in Intel MAX 10 PLLs:

- Automatic switchover—The clock sense circuit monitors the current reference clock. If the current reference clock stops toggling, the reference clock automatically switches to `inclk0` or `inclk1` clock.
- Manual clock switchover—The `clkswitch` signal controls the clock switchover. When the `clkswitch` signal goes from logic low to high, and stays high for at least three clock cycles, the reference clock to the PLL switches from `inclk0` to `inclk1`, or vice-versa.
- Automatic switchover with manual override—This mode combines automatic switchover and manual clock switchover. When the `clkswitch` signal goes high, it overrides the automatic clock switchover function. As long as the `clkswitch` signal is high, any further switchover action is blocked.

Related Information

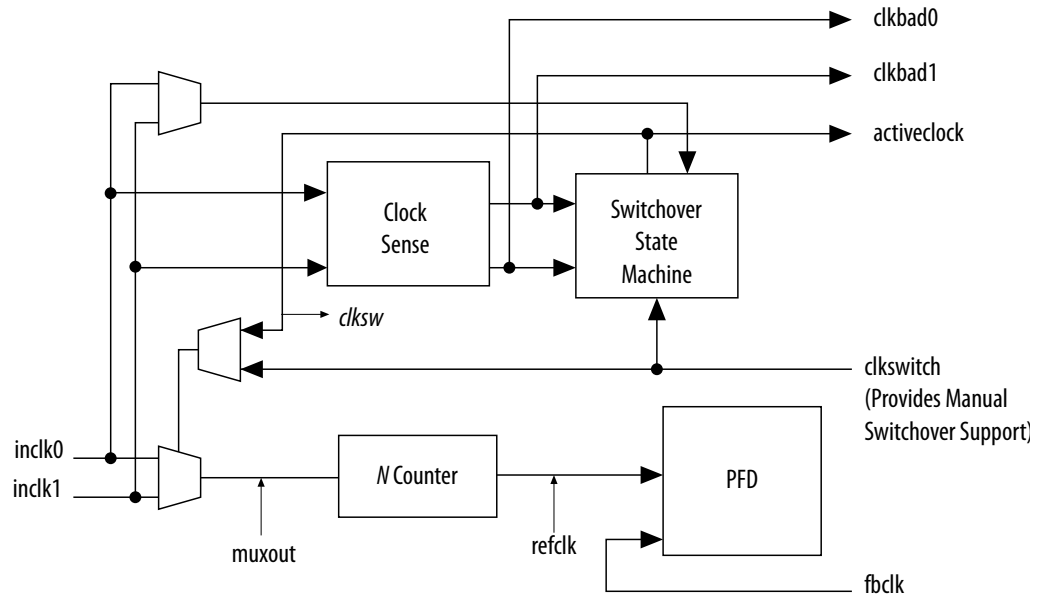
- [Guideline: Clock Switchover](#) on page 37
- [Clock Switchover Parameter Settings](#) on page 53

2.3.12.1. Automatic Clock Switchover

The Intel MAX 10 PLLs support a fully configurable clock switchover capability.

Figure 20. Automatic Clock Switchover Circuit Block Diagram

This figure shows a block diagram of the automatic switchover circuit built into the PLL.



When the current reference clock is not present, the clock sense block automatically switches to the backup clock for PLL reference. You can select a clock source at the backup clock by connecting it to the `inclk1` port of the PLL in your design.

The clock switchover circuit also sends out three status signals—`clkbad[0]`, `clkbad[1]`, and `activeclock`—from the PLL to implement a custom switchover circuit in the logic array.

In automatic switchover mode, the `clkbad[0]` and `clkbad[1]` signals indicate the status of the two clock inputs. When the `clkbad[0]` and `clkbad[1]` signals are asserted, the clock sense block detects that the corresponding clock input has stopped toggling. These two signals are not valid if the frequency difference between `inclk0` and `inclk1` is greater than 20%.

The `activeclock` signal indicates which of the two clock inputs (`inclk0` or `inclk1`) is selected as the reference clock to the PLL. When the frequency difference between the two clock inputs is more than 20%, the `activeclock` signal is the only valid status signal.

Note: Glitches in the input clock may cause the frequency difference between the input clocks to be more than 20%.

When the current reference clock to the PLL stops toggling, use the switchover circuitry to automatically switch from `inclk0` to `inclk1` that runs at the same frequency. This automatic switchover can switch back and forth between the `inclk0` and `inclk1` clocks any number of times when one of the two clocks fails and the other clock is available.

For example, in applications that require a redundant clock with the same frequency as the reference clock, the switchover state machine generates a signal (`clksw`) that controls the multiplexer select input. In this case, `inclk1` becomes the reference clock for the PLL.

When using automatic clock switchover mode, the following requirements must be satisfied:

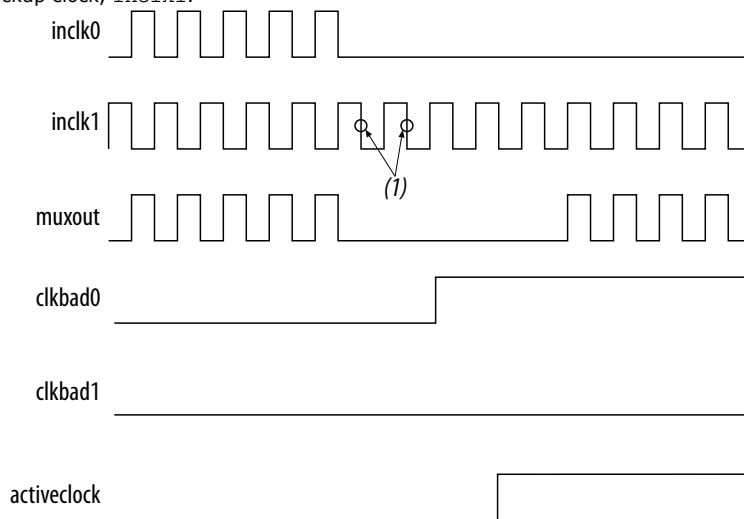
- Both clock inputs must be running when the FPGA is configured.
- The period of the two clock inputs differ by no more than 20%.

If the current clock input stops toggling while the other clock is also not toggling, switchover is not initiated and the `clkbad[0..1]` signals are not valid. If both clock inputs do not have the same frequency, but their period difference is within 20%, the clock sense block detects when a clock stops toggling. However, the PLL might lose lock after the switchover completes and needs time to relock.

Note: Intel recommends resetting the PLL using the `areset` signal to maintain the phase relationships between the PLL input and output clocks when using clock switchover.

Figure 21. Example of Automatic Switchover After Loss of Clock Detection

This figure shows an example waveform of the switchover feature in automatic switchover mode. In this example, the `inclk0` signal remains low. After the `inclk0` signal remains low for approximately two clock cycles, the clock sense circuitry drives the `clkbad[0]` signal high. Since the reference clock signal is not toggling, the switchover state machine controls the multiplexer through the `clksw` signal to switch to the backup clock, `inclk1`.



Note:

(1) Switchover is enabled on the falling edge of `inclk0` or `inclk1`, depending on which clock is available. In this figure, switchover is enabled on the falling edge of `inclk1`.

2.3.12.2. Automatic Switchover with Manual Override

In automatic switchover with manual override mode, you can use the `clkswitch` signal for user- or system-controlled switch conditions. You can use this mode for same-frequency switchover, or to switch between inputs of different frequencies.

For example, if `inclk0` is 66 MHz and `inclk1` is 200 MHz, you must control the switchover using the `clkswitch` signal. The automatic clock sense circuitry cannot monitor clock input (`inclk0` and `inclk1`) frequencies with a frequency difference of more than 20%.

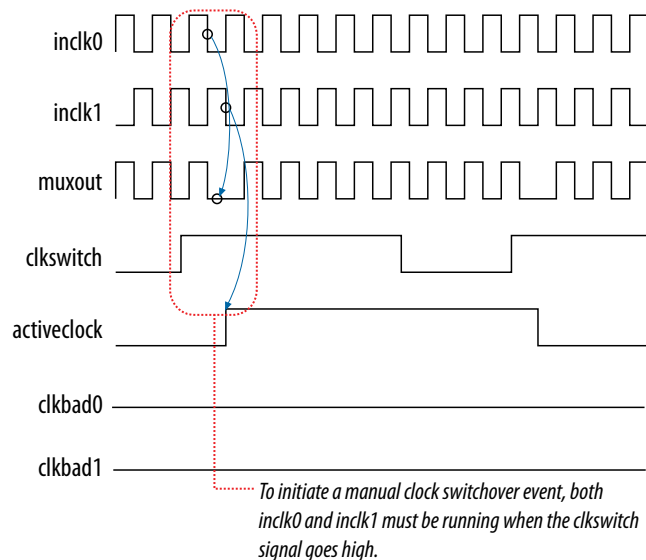
This feature is useful when clock sources originate from multiple cards on the backplane, requiring a system-controlled switchover between frequencies of operation.

You must choose the backup clock frequency and set the `M`, `N`, and `C` counters so that the VCO operates within the recommended frequency range.

The following figure shows a clock switchover waveform controlled by the `clkswitch` signal. In this case, both clock sources are functional and `inclk0` is selected as the reference clock. The `clkswitch` signal goes high, which starts the switchover sequence. On the falling edge of `inclk0`, the counter's reference clock, `muxout`, is gated off to prevent clock glitching. On the falling edge of `inclk1`, the reference clock multiplexer switches from `inclk0` to `inclk1` as the PLL reference. The `activeclock` signal is asserted to indicate the clock that is currently feeding the PLL, which is `inclk1`.

In automatic override with manual switchover mode, the `activeclock` signal mirrors the `clkswitch` signal. Since both clocks are still functional during the manual switch, neither `clkbad` signal goes high. Because the switchover circuit is positive-edge sensitive, the falling edge of the `clkswitch` signal does not cause the circuit to switch back from `inclk1` to `inclk0`. When the `clkswitch` signal goes high again, the process repeats.

Figure 22. Example of Clock Switchover Using the `clkswitch` (Manual) Control



The `clkswitch` signal and automatic switch work only if the clock being switched to is available. If the clock is not available, the state machine waits until the clock is available.

2.3.12.3. Manual Clock Switchover

In manual clock switchover mode, the `clkswitch` signal controls whether `inclk0` or `inclk1` is selected as the input clock to the PLL. By default, `inclk0` is selected.

A clock switchover event is initiated when the `clkswitch` signal transitions from logic low to logic high, and is being held high for at least three `inclk` cycles. You must bring the `clkswitch` signal back to low again to perform another switchover event. If you do not require another switchover event, you can leave the `clkswitch` signal in a logic high state after the initial switch. Pulsing the `clkswitch` signal high for at least three `inclk` cycles performs another switchover event.

If `inclk0` and `inclk1` have different frequencies and are always running, the minimum amount of time for which `clkswitch` signal is high must be greater than or equal to three of the slower-frequency `inclk0` and `inclk1` cycles.

2.3.13. PLL Cascading

Related Information

[Guideline: PLL Cascading](#) on page 36

2.3.13.1. PLL-to-PLL Cascading

Two PLLs are cascaded to each other through the clock network. If your design cascades PLLs, the source (upstream) PLL must have a low-bandwidth setting and the destination (downstream) PLL must have a high-bandwidth setting.

Related Information

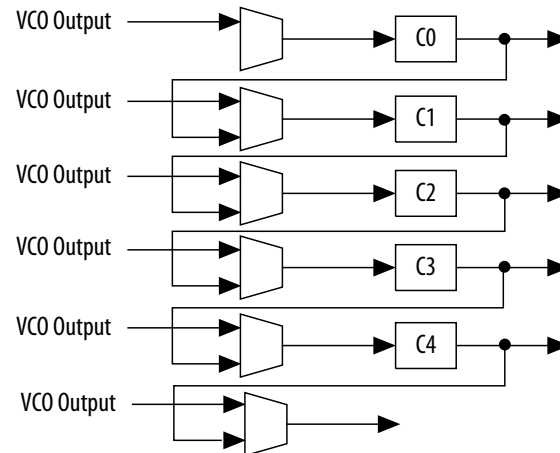
[Programmable Bandwidth Parameter Settings](#) on page 53

Provides more information about the bandwidth settings.

2.3.13.2. Counter-to-Counter Cascading

The Intel MAX 10 PLLs support post-scale counter cascading to create counters larger than 512. This is implemented by feeding the output of one `C` counter into the input of the next `C` counter.

Figure 23. Counter-to-Counter Cascading



When cascading counters to implement a larger division of the high-frequency VCO clock, the cascaded counters behave as one counter with the product of the individual counter settings.

For example, if $C0 = 4$ and $C1 = 2$, the cascaded value is $C0 \times C1 = 8$.

The Intel Quartus Prime software automatically sets all the post-scale counter values for cascading in the configuration file. Post-scale counter cascading cannot be performed using PLL reconfiguration.

2.3.14. PLL Reconfiguration

The PLLs use several divide counters and different VCO phase taps to perform frequency synthesis and phase shifts. In Intel MAX 10 PLLs, you can reconfigure both counter settings and phase shift the PLL output clock in real time. You can also change the charge pump and loop filter components, which dynamically affects the PLL bandwidth.

The following PLL components are configurable in real time:

- Pre-scale counter (N)
- Feedback counter (M)
- Post-scale output counters (C0-C4)
- Charge pump current (I_{CP})
- Loop filter components (R, C)

You can use these PLL components to update the following settings in real time without reconfiguring the entire FPGA:

- Output clock frequency
- PLL bandwidth
- Phase shift

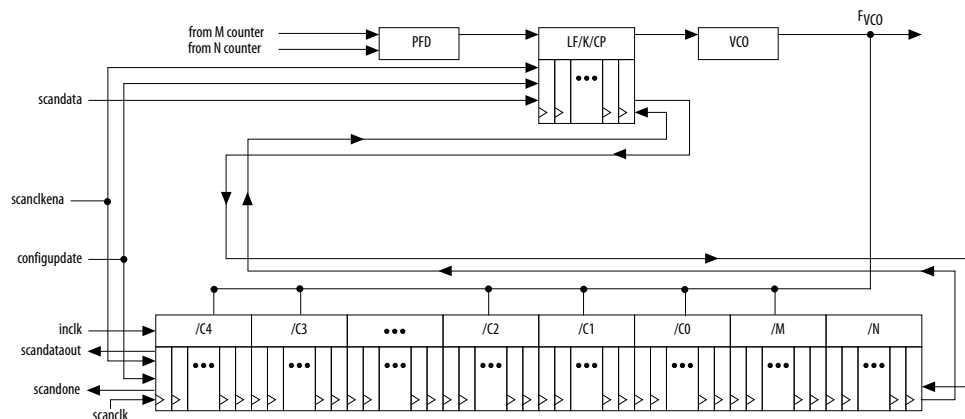
The ability to reconfigure the PLL in real time is useful in applications that may operate in multiple frequencies. It is also useful in prototyping environments, allowing you to sweep PLL output frequencies and dynamically adjust the output clock phase.

For instance, a system generating test patterns is required to generate and send patterns at 75 or 150 MHz, depending on the requirements of the device under test. Reconfiguring the PLL components in real time allows you to switch between two such output frequencies in a few microseconds.

You can also use this feature to adjust clock-to-out (t_{CO}) delays in real time by changing the PLL output clock phase shift. This approach eliminates the need to regenerate a configuration file with the new PLL settings.

Figure 24. PLL Reconfiguration Scan Chain

This figure shows the dynamic adjustment of the PLL counter settings by shifting their new settings into a serial shift register chain or scan chain. Serial data shifts to the scan chain via the `scandata` port, and shift registers are clocked by `scanclk`. The maximum `scanclk` frequency is 100 MHz. After shifting the last bit of data, asserting the `configupdate` signal for at least one `scanclk` clock cycle synchronously updates the PLL configuration bits with the data in the scan registers.



The counter settings are updated synchronously to the clock frequency of the individual counters. Therefore, not all counters update simultaneously.

The dynamic reconfiguration scheme uses configuration files, such as the Hexadecimal-format file (**.hex**) or the Memory Initialization file (**.mif**). These files are used together with the ALTPLL_RECONFIG Intel FPGA IP core to perform the dynamic reconfiguration.

Related Information

- [Guideline: .mif Streaming in PLL Reconfiguration](#) on page 38
- [PLL Dynamic Reconfiguration Implementation](#) on page 42
- [PLL Dynamic Reconfiguration Parameter Settings](#) on page 54
Provides more information about the ALTPLL IP core parameter settings in the Intel Quartus Prime software.
- [ALTPLL_RECONFIG IP Core Parameters](#) on page 59
Provides more information about the ALTPLL_RECONFIG IP core parameter settings in the Intel Quartus Prime software.

3. Intel MAX 10 Clocking and PLL Design Considerations

3.1. Clock Networks Design Considerations

3.1.1. Guideline: Clock Enable Signals

Intel recommends using the `clkena` signals when switching the clock source to the PLLs or GCLK. The recommended sequence is as follows:

1. Disable the primary output clock by deasserting the `clkena` signal.
2. Switch to the secondary clock using the dynamic select signals of the clock control block.
3. Allow some clock cycles of the secondary clock to pass before reasserting the `clkena` signal. The exact number of clock cycles to wait before enabling the secondary clock depends on your design. You can build a custom logic to ensure a glitch-free transition when switching between different clock sources.

Related Information

- [Clock Enable Signals](#) on page 12
- [ALTCLKCTRL IP Core Parameters](#) on page 50
- [ALTCLKCTRL IP Core Ports and Signals](#) on page 51

3.1.2. Guideline: Connectivity Restrictions

The following guidelines describe the restrictions associated with the signal sources that can drive the `inclk` input:

- You must use the `inclk` ports that are consistent with the `clkselect` ports.
- When you are using multiple input sources, the `inclk` ports can only be driven by the dedicated clock input pins and the PLL clock outputs.
- If the clock control block feeds any `inclk` port of another clock control block, both clock control blocks must be able to be reduced to a single clock control block of equivalent functionality.
- When you are using the glitch-free switchover feature, the clock you are switching from must be active. If the clock is not active, the switchover circuit cannot transition from the clock you originally selected.

3.2. Internal Oscillator Design Considerations

3.2.1. Guideline: Connectivity Restrictions

You cannot drive the PLLs with internal oscillator.

3.3. PLLs Design Considerations

3.3.1. Guideline: PLL Control Signals

You must include the `areset` signal in your designs if one of the following conditions is true:

- PLL reconfiguration or clock switchover is enabled in your design.
- Phase relationships between the PLL input clock and output clocks must be maintained after a loss-of-lock condition.
- The input clock to the PLL is toggling or unstable at power-up.
- The `areset` signal is asserted after the input clock is stable and within specifications.

Related Information

- [PLL Control Signals](#) on page 18
- [Guideline: Self-Reset](#) on page 35
Provides more details on loss-of-lock condition.

3.3.2. Guideline: Connectivity Restrictions

To comply with simultaneous switching noise (SSN) design guideline, Intel recommends that you do not use unterminated I/O in the same bank as the input clock signal to the PLL.

Related Information

- [Simultaneous Switching Noise \(SSN\) Analysis and Optimizations chapter, Intel Quartus Prime Standard Edition Handbook Volume 2 Design Implementation and Optimization](#)
Provides more information about SSN.
- [Guidelines: Clock and Asynchronous Control Input Signal, Intel MAX 10 FPGA Signal Integrity Design Guidelines](#)
Provides more information about using I/O connectivity restrictions.

3.3.3. Guideline: Self-Reset

The lock time of a PLL is the amount of time required by the PLL to attain the target frequency and phase relationship after device power-up, after a change in the PLL output frequency, or after resetting the PLL.

A PLL might lose lock for a number of reasons, such as the following causes:

- Excessive jitter on the input clock.
- Excessive switching noise on the clock inputs of the PLL.
- Excessive noise from the power supply, causing high output jitter and possible loss of lock.
- A glitch or stopping of the input clock to the PLL.
- Resetting the PLL by asserting the `areset` port of the PLL.

- An attempt to reconfigure the PLL might cause the `M` counter, `N` counter, or phase shift to change, causing the PLL to lose lock. However, changes to the post-scale counters do not affect the PLL `locked` signal.
- PLL input clock frequency drifts outside the lock range specification.
- The PFD is disabled using the `pfdena` port. When this happens, the PLL output phase and frequency tend to drift outside of the lock window.

The ALTPLL Intel FPGA IP core allows you to monitor the PLL locking process using a lock signal named `locked` and also allows you to set the PLL to self-reset on loss of lock.

3.3.4. Guideline: Output Clocks

Each Intel MAX 10 PLL supports up to five output clocks. You can use the output clock port as a core output clock or an external output clock port. The core output clock feeds the FPGA core and the external output clock feeds the dedicated pins on the FPGA.

The ALTPLL IP core does not have a dedicated output enable port. You can disable the PLL output using the `areset` signal to disable the PLL output counters.

3.3.5. Guideline: PLL Cascading

Consider the following guidelines when cascading PLLs:

- Set the primary PLL to low bandwidth to help filter jitter. Set the secondary PLL to high bandwidth to track the jitter from the primary PLL. You can view the Intel Quartus Prime software compilation report file to ensure the PLL bandwidth ranges do not overlap. If the bandwidth ranges overlap, jitter peaking can occur in the cascaded PLL scheme.

Note: You can get an estimate of the PLL deterministic jitter and static phase error (SPE) by using the Timing Analyzer in the Intel Quartus Prime software. Use the SDC command `derive_clock_uncertainty` to generate a report titled `PLLJ_PLLSPE_INFO.txt` in your project directory. Then, use `set_clock_uncertainty` command to add jitter and SPE values to your clock constraints.

- Keep the secondary PLL in a reset state until the primary PLL has locked to ensure the phase settings are correct on the secondary PLL.
- You cannot connect any of the `inclk` ports of any PLLs in a cascaded scheme to the clock outputs from PLLs in the cascaded scheme.

Related Information

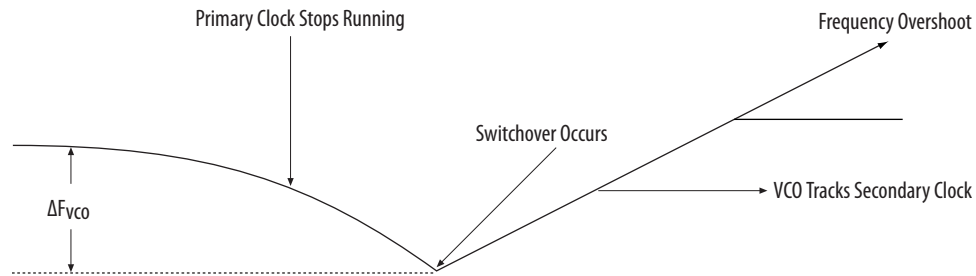
- [PLL Cascading](#) on page 31
- [The Intel Quartus Prime Timing Analyzer chapter, Intel Quartus Prime Standard Edition Handbook Volume 3 Verification](#)
Provides more information about Timing Analyzer.

3.3.6. Guideline: Clock Switchover

Use the following guidelines to design with clock switchover in PLLs:

- Clock loss detection and automatic clock switchover requires that the frequency difference between `inclk0` and `inclk1` is within 20% range. Failing to meet this requirement causes the `clkbad[0]` and `clkbad[1]` signals to function improperly.
- When using manual clock switchover, the frequency difference between `inclk0` and `inclk1` can be more than 20%. However, differences between the two clock sources (frequency, phase, or both) can cause the PLL to lose lock. Resetting the PLL ensures that the correct phase relationships are maintained between the input and output clocks.
- Both `inclk0` and `inclk1` must be running when the `clkswitch` signal goes high to start the manual clock switchover event. Failing to meet this requirement causes the clock switchover to malfunction.
- Applications that require a clock switchover feature and a small frequency drift must use a low-bandwidth PLL. When referencing input clock changes, the low-bandwidth PLL reacts slower than a high-bandwidth PLL. When the switchover happens, the low-bandwidth PLL propagates the stoppage of the clock to the output at a slower speed than the high-bandwidth PLL. The low-bandwidth PLL filters out jitter on the reference clock. However, be aware that the low-bandwidth PLL also increases lock time.
- After a switchover occurs, there might be a finite resynchronization period for the PLL to lock onto a new clock. The exact amount of time it takes for the PLL to relock depends on the PLL configuration.
- The phase relationship between the input clock to the PLL and output clock from the PLL is important in your design. Assert `areset` for 10 ns after performing a clock switchover. Wait for the locked signal (or gated lock) to go high before reenabling the output clocks from the PLL.
- Disable the system during switchover if the system is not tolerant of frequency variations during the PLL resynchronization period. You can use the `clkbad[0]` and `clkbad[1]` status signals to turn off the PFD (`pfdena = 0`) so that the VCO maintains its last frequency. You can also use the switchover state machine to switch over to the secondary clock. After enabling the PFD, the output clock enable signals (`clkena`) can disable clock outputs during the switchover and resynchronization period. After the lock indication is stable, the system can reenables the output clock or clocks.
- The VCO frequency gradually decreases when the primary clock is lost and then increases as the VCO locks onto the secondary clock, as shown in the following figure. After the VCO locks onto the secondary clock, some overshoot can occur (an over-frequency condition) in the VCO frequency.

Figure 25. VCO Switchover Operating Frequency



Related Information

- [Clock Switchover](#) on page 27
- [Clock Switchover Parameter Settings](#) on page 53

3.3.7. Guideline: .mif Streaming in PLL Reconfiguration

Consider the following guidelines when using .mif streaming in PLL reconfiguration:

- 10M02 devices do not support .mif streaming in PLL reconfiguration due to flash size limitation. Intel recommends using an external flash.
- 10M04, 10M08, 10M16, 10M25, 10M40, and 10M50 devices only support .mif streaming in single image mode. Intel recommends using an external flash for dual image mode. The Intel MAX 10 devices do not support using both dual image mode and PLL reconfiguration with .mif simultaneously.

Related Information

[PLL Reconfiguration](#) on page 32

3.3.8. Guideline: scandone Signal for PLL Reconfiguration

scandone signal must be low before the second PLL reconfiguration. For scandone signal to go low, PLL areset signal must be asserted.

4. Intel MAX 10 Clocking and PLL Implementation Guides

4.1. ALTCLKCTRL Intel FPGA IP Core

The ALTCLKCTRL IP core is a clock control function for configuring the clock control block.

The common applications of the ALTCLKCTRL IP core are as follows:

- Dynamic clock source selection—When using the clock control block, you can select the dynamic clock source that drives the global clock network.
- Dynamic power-down of a clock network—The dynamic clock enable or disable feature allows internal logic to power down the clock network. When a clock network is powered down, all the logic fed by that clock network is not toggling, thus reducing the overall power consumption of the device.

The ALTCLKCTRL IP core provides the following features:

- Supports clock control block operation mode specifications
- Supports specification of the number of input clock sources
- Provides an active high clock enable control input

Related Information

- [Introduction to Intel FPGA IP Cores](#)
Provides general information about all Intel FPGA IP cores, including parameterizing, generating, upgrading, and simulating IP cores.
- [Intel MAX 10 Clocking and PLL User Guide Archives](#) on page 66
Provides a list of user guides for previous versions of the ALTCLKCTRL IP core.

4.2. ALTPLL Intel FPGA IP Core

The ALTPLL IP core specifies the PLL circuitry. You can use this IP core to configure the PLL types, operation modes, and advanced features of the PLL.

Related Information

- [Introduction to Intel FPGA IP Cores](#)
Provides general information about all Intel FPGA IP cores, including parameterizing, generating, upgrading, and simulating IP cores.
- [Intel MAX 10 Clocking and PLL User Guide Archives](#) on page 66
Provides a list of user guides for previous versions of the ALTPLL IP core.

4.2.1. Expanding the PLL Lock Range

The PLL lock range is between the minimum (PLL Freq Min Lock parameter) and maximum (PLL Freq Max Lock parameter) input frequency values for which the PLL can achieve lock. Changing the input frequency might cause the PLL to lose lock, but if the input clock remains within the minimum and maximum frequency specifications, the PLL is able to achieve lock. The Intel Quartus Prime software shows these input frequency values in the PLL Summary report located under the Resource Section of the Fitter folder in the Compilation Report.

The Intel Quartus Prime software does not necessarily pick values for the PLL parameters to maximize the lock range. For example, when you specify a 75 MHz input clock in the ALTPLL parameter editor, the actual PLL lock range might be between 70 MHz to 90 MHz. If your application requires a lock range of 50 MHz to 100 MHz, the default lock range of this PLL is insufficient.

For devices that support clock switchover in PLLs, you can use the ALTPLL IP core parameter editor to maximize the lock range.

To extract valid parameter values to maximize your PLL lock range, perform the following steps:

1. In the schematic editor, double-click the ALTPLL instance in your design to open the ALTPLL parameter editor.
2. On the **General/Modes** page, for **What is the frequency of the inclk0 input?**, type the value of the low end of your desired PLL lock range.
For example, if your application requires a lock range of 50 MHz to 100 MHz, type 50 MHz.
3. On the **Inputs/Lock** page, turn on **Create output file(s) using the 'Advanced' PLL parameters**.
4. On the **Clock switchover** page, turn on **Create an 'inclk1' input for a second input clock** and enter the high end of your lock range as the frequency for `inclk1`.
For example, if your application requires a lock range of 50 MHz to 100 MHz, type 100 MHz.
5. Set the rest of the parameters in the remaining pages of the ALTPLL IP core parameter editor.
6. Compile your project and note the lock range shown in the PLL Usage Summary report. If it is satisfactory, note all of the values for the PLL from this report, such as the M value, N value, charge pump current, loop filter resistance, and loop filter capacitance.
7. In the schematic editor, double-click the ALTPLL instance in your design to open the ALTPLL parameter editor.
8. On the **Clock switchover** page, turn off **Create an 'inclk1' input for a second input clock**.
9. Click **Finish** to update the PLL wrapper file.
10. In a text editor, open the PLL wrapper file. Modify all of the values for the parameters listed in step 6. Save the changes.
— If the wrapper file is in Verilog format, go to the **defparam** section.

- If the wrapper file is in VHDL HDL, go to the **Generic Map** section.
- 11. Compile your project.
- 12. Check the PLL Summary report to confirm that the PLL lock range meets your requirements. The modified PLL should have the desired lock range.

If your input clock frequency is too close to the end of the desired PLL lock range—for example the low end of the desired lock range is 50 MHz and the input clock frequency is 50 MHz, the PLL might not maintain lock when the input clock has jitter or the frequency drifts below 50 MHz. You may choose to expand your PLL lock range to ensure your expected input clock frequency is further from the end of the range. For this example, you can enter 45 MHz and 105 MHz to ensure that your target lock range of 50 MHz to 100 MHz is within the PLL lock range.

The Intel Quartus Prime software prompts an error message if it is unable to implement your preferred lock range using this procedure. Therefore, you have to look into other options, such as PLL reconfiguration, to support your input frequency range.

4.2.2. Programmable Bandwidth with Advanced Parameters

An advanced level of control is also possible for precise control of the PLL loop filter characteristics. This level allows you to explicitly select the following advanced parameters:

- Charge pump current (`charge_pump_current`)
- Loop filter resistance (`loop_filter_r`)
- Loop filter capacitance (`loop_filter_c`)

This option is intended for advanced users who know the exact details of their PLL configuration. You can use this option if you understand the parameters well enough to set them optimally. The files generated are not intended to be reused by the ALTPLL IP core parameter editor. After the ALTPLL IP core output files are specified using the advanced parameters, the Intel Quartus Prime compiler cannot change them. For example, the compiler cannot perform optimization. Thus, your design cannot benefit from improved algorithms of the compiler. The Intel Quartus Prime compiler cannot select better settings or change some settings that the ALTPLL IP core parameter editor finds to be incompatible with your design.

The parameter settings to generate output files using advanced PLL parameters are located on the **Inputs/Lock** page of the ALTPLL IP core parameter editor.

Turn on **Create output file(s) using the 'Advanced' PLL parameters** to enable the feature.

When you turn on this option, the generated output files contain all of the initial counter values used in the PLL. You can use these values for functional simulation in a third-party simulator.

These parameter settings create no additional top-level ports.

Related Information

- [Programmable Bandwidth](#) on page 25
- [Charge Pump and Loop Filter](#) on page 44
 - Provides more information about the PLL components to update PLL bandwidth in real time.

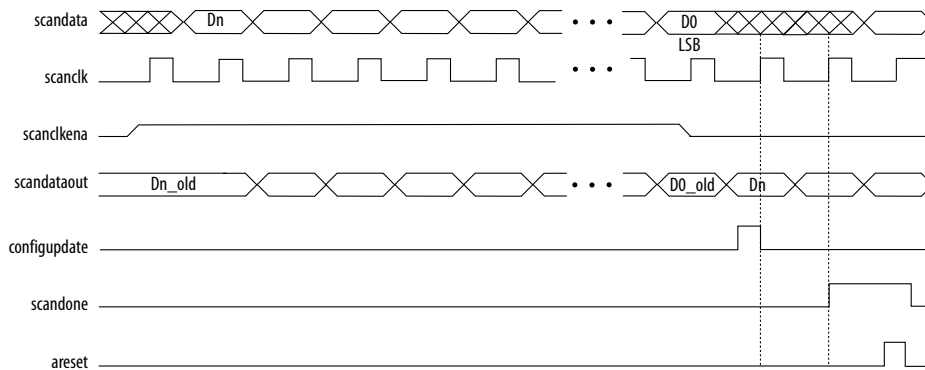
- [Programmable Bandwidth Parameter Settings](#) on page 53

4.2.3. PLL Dynamic Reconfiguration Implementation

To reconfigure the PLL counters, perform the following steps:

1. Assert the `scanclkena` signal at least one `scanclk` cycle prior to shifting in the first bit of `scandata` (`Dn`).
2. Shift the serial data (`scandata`) into the scan chain on the second rising edge of `scanclk`.
3. After all 144 bits have been scanned into the scan chain, deassert the `scanclkena` signal to prevent inadvertent shifting of bits in the scan chain.
4. Assert the `configupdate` signal for one `scanclk` cycle to update the PLL counters with the contents of the scan chain.
The `scandone` signal goes high indicating that the PLL is being reconfigured. A falling edge indicates that the PLL counters have been updated with new settings.
5. Reset the PLL using the `areset` signal if you make any changes to the `M`, `N`, post-scale output `C` counters, or the `ICP`, `R`, and `C` settings.
6. You can repeat steps 1 through 5 to reconfigure the PLL any number of times.

Figure 26. PLL Reconfiguration Scan Chain Functional Simulation



When reconfiguring the counter clock frequency, you cannot reconfigure the corresponding counter phase shift settings using the same interface. You can reconfigure phase shifts in real time using the dynamic phase shift reconfiguration interface. If you want to keep the same nonzero phase shift setting (for example, 90°) on the clock output, you must reconfigure the phase shift after reconfiguring the counter clock frequency.

Related Information

[PLL Reconfiguration](#) on page 32

4.2.3.1. Post-Scale Counters (C0 to C4)

You can configure the multiply or divide values and duty cycle of the post-scale counters in real time. Each counter has an 8-bit high time setting and an 8-bit low time setting. The duty cycle is the ratio of output high or low time to the total cycle time, which is the sum of the two.

The post-scale counters have two control bits:

- `rbypass`—For bypassing the counter
- `rseledd`—For selecting the output clock duty cycle

When the `rbypass` bit is set to 1, it bypasses the counter, resulting in a division by one. When this bit is set to 0, the PLL computes the effective division of the VCO output frequency based on the high and low time counters. The PLL implements this duty cycle by transitioning the output clock from high-to-low on the rising edge of the VCO output clock.

For example, if the post-scale divide factor is 10, the high and low count values are set to 5 and 5 respectively, to achieve a 50–50% duty cycle. However, a 4 and 6 setting for the high and low count values, respectively, would produce an output clock with 40–60% duty cycle.

The `rseledd` bit indicates an odd divide factor for the VCO output frequency with a 50% duty cycle. The PLL implements this duty cycle by transitioning the output clock from high-to-low on a falling edge of the VCO output clock.

For example, if the post-scale divide factor is 3, the high and low time count values are 2 and 1 respectively, to achieve this division. This implies a 67%–33% duty cycle. If you need a 50%–50% duty cycle, you must set the `rseledd` control bit to 1 to achieve this duty cycle despite an odd division factor. When you set `rseledd` = 1, subtract 0.5 cycles from the high time and add 0.5 cycles to the low time.

The calculation for the example is shown as follows:

- High time count = 2 cycles
- Low time count = 1 cycle
- `rseledd` = 1 effectively equals:
 - High time count = 1.5 cycles
 - Low time count = 1.5 cycles
 - Duty cycle = (1.5/3)% high time count and (1.5/3)% low time count

Related Information

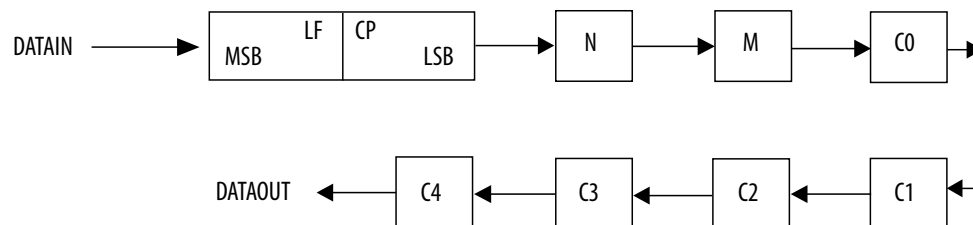
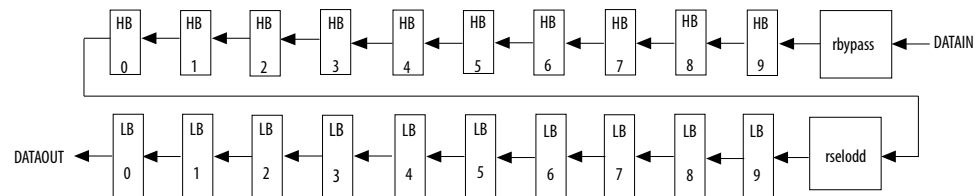
[Programmable Duty Cycle](#) on page 24

4.2.3.2. Scan Chain

The Intel MAX 10 PLLs have a 144-bit scan chain.

Table 7. PLL Component Reprogramming Bits

Block Name	Number of Bits		
	Counter	Control Bit	Total
C4 ⁽⁶⁾	16	2 ⁽⁷⁾	18
C3	16	2 ⁽⁷⁾	18
C2	16	2 ⁽⁷⁾	18
C1	16	2 ⁽⁷⁾	18
C0	16	2 ⁽⁷⁾	18
M	16	2 ⁽⁷⁾	18
N	16	2 ⁽⁷⁾	18
Charge Pump	9	0	9
Loop Filter ⁽⁸⁾	9	0	9
Total number of bits			144

Figure 27. PLL Component Scan Chain Order

Figure 28. PLL Post-Scale Counter Scan Chain Bit Order


4.2.3.3. Charge Pump and Loop Filter

You can reconfigure the following settings to update the PLL bandwidth in real time:

- Charge pump (I_{CP})
- Loop filter resistor (R)
- Loop filter capacitor (C)

⁽⁶⁾ LSB bit for C4 low-count value is the first bit shifted into the scan chain.

⁽⁷⁾ These two control bits include *rbypass*, for bypassing the counter, and *rseledd*, for selecting the output clock duty cycle.

⁽⁸⁾ MSB bit for loop filter is the last bit shifted into the scan chain.

Table 8. Charge Pump Bit Control

CP[2]	CP[1]	CP[0]	Setting (Decimal)
0	0	0	0
0	0	1	1
0	1	1	3
1	1	1	7

Table 9. Loop Filter Resistor Value Control

LFR[4]	LFR[3]	LFR[2]	LFR[1]	LFR[0]	Setting (Decimal)
0	0	0	0	0	0
0	0	0	1	1	3
0	0	1	0	0	4
0	1	0	0	0	8
1	0	0	0	0	16
1	0	0	1	1	19
1	0	1	0	0	20
1	1	0	0	0	24
1	1	0	1	1	27
1	1	1	0	0	28
1	1	1	1	0	30

Table 10. Loop Filter High Frequency Capacitor Control

LFC[1]	LFC[0]	Setting (Decimal)
0	0	0
0	1	1
1	1	3

Related Information

- [Programmable Bandwidth](#) on page 25
- [Programmable Bandwidth with Advanced Parameters](#) on page 41
- [Programmable Bandwidth Parameter Settings](#) on page 53

4.2.3.4. Bypassing PLL Counter

Bypassing a PLL counter results in a feedback counter (M), pre-scale counter (N), or post-scale output counters (C0–C4) factor of one.

Table 11. PLL Counter Settings

Description	PLL Scan Chain Bits [0..8] Settings								
	LSB								MSB
PLL counter bypassed	X	X	X	X	X	X	X	X	1 ⁽⁹⁾
PLL counter not bypassed	X	X	X	X	X	X	X	X	0 ⁽⁹⁾

To bypass any of the PLL counters, set the bypass bit to 1. The values on the other bits are ignored.

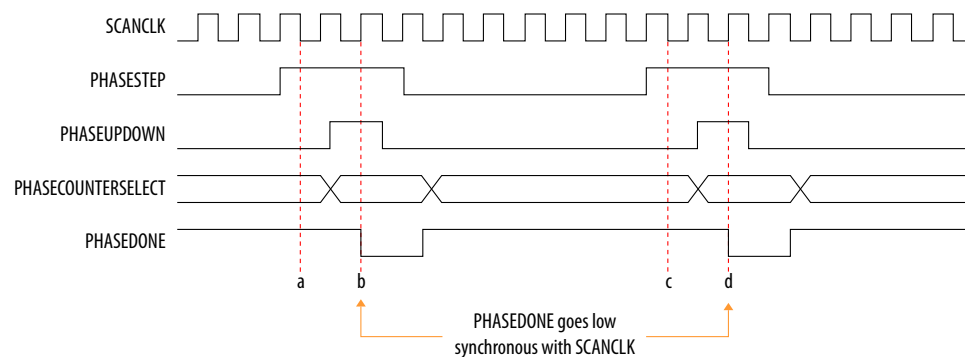
4.2.4. Dynamic Phase Configuration Implementation

To perform one dynamic phase shift step, follow these steps:

1. Set PHASEUPDOWN and PHASECOUNTERSELECT as required.
2. Assert PHASESTEP for at least two SCANCLK cycles. Each PHASESTEP pulse allows one phase shift.
3. Deassert PHASESTEP after PHASEDONE goes low.
4. Wait for PHASEDONE to go high.
5. Repeat steps 1 through 4 as many times as required to perform multiple phase shifts.

PHASEUPDOWN and PHASECOUNTERSELECT signals are synchronous to SCANCLK and must meet the set up time (t_{su}) and hold time (t_h) requirements with respect to the SCANCLK edges.

You can repeat dynamic phase-shifting indefinitely. For example, in a design where the VCO frequency is set to 1,000 MHz and the output clock frequency is set to 100 MHz, performing 40 dynamic phase shifts (each one yields 125 ps phase shift) results in shifting the output clock by 180°, which is a phase shift of 5 ns.

Figure 29. Dynamic Phase Shift Timing Diagram


The PHASESTEP signal is latched on the negative edge of SCANCLK (a,c) and must remain asserted for at least two SCANCLK cycles. Deassert PHASESTEP after PHASEDONE goes low.

⁽⁹⁾ Bypass bit

On the second SCANCLK rising edge (b,d) after PHASESTEP is latched, the values of PHASEUPDOWN and PHASECOUNTERSELECT are latched. The PLL starts dynamic phase-shifting for the specified counters and in the indicated direction.

The PHASEDONE signal is deasserted synchronous to SCANCLK at the second rising edge (b,d) and remains low until the PLL finishes dynamic phase-shifting. Depending on the VCO and SCANCLK frequencies, PHASEDONE low time may be greater than or less than one SCANCLK cycle.

You can perform another dynamic phase-shift after the PHASEDONE signal goes from low to high. Each PHASESTEP pulse enables one phase shift. The PHASESTEP pulses must be at least one SCANCLK cycle apart.

Related Information

- [Programmable Phase Shift](#) on page 25
- [Dynamic Phase Configuration Parameter Settings](#) on page 54
Provides more information about the ALTPLL IP core parameter settings in the Intel Quartus Prime software.
- [ALTPLL_RECONFIG IP Core Parameters](#) on page 59
Provides more information about the ALTPLL_RECONFIG IP core parameter settings in the Intel Quartus Prime software.
- [PLL Dynamic Reconfiguration Parameter Settings](#) on page 54
Provides more information about the ALTPLL IP core parameter settings in the Intel Quartus Prime software.
- [ALTPLL_RECONFIG IP Core Parameters](#) on page 59
Provides more information about the ALTPLL_RECONFIG IP core parameter settings in the Intel Quartus Prime software.

4.2.4.1. Dynamic Phase Configuration Counter Selection

Table 12. Phase Counter Select Mapping

PLL Counter Selection	PHASECOUNTERSELECT [2]	[1]	[0]
All output counters	0	0	0
M counter	0	0	1
C0 counter	0	1	0
C1 counter	0	1	1
C2 counter	1	0	0
C3 counter	1	0	1
C4 counter	1	1	0

Related Information

[Programmable Phase Shift](#) on page 25

4.2.4.2. Dynamic Phase Configuration with Advanced Parameters

The finest phase shift step resolution you can get in the ALTPLL IP core is 1/8 of the VCO period. If the VCO frequency is at the lower end of the supported VCO range, the phase shift step resolution might be larger than preferred for your design.

You can modify your phase shift resolution using the dynamic phase reconfiguration feature of the PLL. If you want to modify the phase shift resolution without the dynamic phase reconfiguration feature enabled, perform the following steps:

1. Create an ALTPLL instance. Make sure you specify the speed grade of your target device and the PLL type.
2. On the **PLL Reconfiguration** page, turn on **Create optional inputs for dynamic phase reconfiguration** and **Enable phase shift step resolution**.
3. On the **Output Clocks** page, set your desired phase shift for each required output clock. Note all the internal PLL settings shown.
4. On the **Bandwidth/SS** page, click **More Details** to see the internal PLL settings. Note all of the settings shown.
5. On the **Inputs/Lock** page, turn on **Create output file(s) using the 'Advanced' PLL Parameters**.
6. Return to the **PLL Reconfiguration** page and turn off **Create Optional Inputs for Dynamic Phase Reconfiguration**.
7. Click **Finish** to generate the PLL instantiation file(s).

When using Advanced Parameters, the PLL wrapper file (`<ALTPLL_instantiation_name>.v` or `<ALTPLL_instantiation_name>.vhd`) is written in a format that allows you to identify the PLL parameters. The parameters are listed in the **Generic Map** section of the VHDL file, or in the `defparam` section of the Verilog file.

8. Open your PLL instantiation wrapper file and locate either the **Generic Map** or the `defparam` section.
9. Modify the settings to match the settings that you noted in steps 3 and 4.
10. Save the PLL instantiation wrapper file and compile your design.
11. Verify that the output clock frequencies and phases are correct in the PLL Usage report located under the Resource section of the Fitter folder in the Compilation Report.

By using this technique, you can apply valid PLL parameters as provided by the ALTPLL IP core parameter editor to optimize the settings for your design.

Alternatively, you can leave the dynamic phase reconfiguration option enabled and tie the relevant input ports—`phasecounterselect[3..0]`, `phaseupdown`, `phasesstep`, and `scanclk`—to constants, if you prefer not to manually edit the PLL wrapper file using the Advanced PLL Parameters option.

Related Information

[Programmable Phase Shift](#) on page 25

4.3. ALTPLL_RECONFIG Intel FPGA IP Core

The ALTPLL_RECONFIG IP core implements reconfiguration logic to facilitate dynamic real-time reconfiguration of PLLs. You can use the IP core to update the output clock frequency, PLL bandwidth, and phase shifts in real time, without reconfiguring the entire FPGA.

Use the ALTPLL_RECONFIG IP core in designs that must support dynamic changes in the frequency and phase shift of clocks and other frequency signals. The IP core is also useful in prototyping environments because it allows you to sweep PLL output frequencies and dynamically adjust the output clock phase. You can also adjust the clock-to-output (t_{CO}) delays in real-time by changing the output clock phase shift. This approach eliminates the need to regenerate a configuration file with the new PLL settings. This operation requires dynamic phase-shifting.

Related Information

- [Introduction to Intel FPGA IP Cores](#)
Provides general information about all Intel FPGA IP cores, including parameterizing, generating, upgrading, and simulating IP cores.
- [Intel MAX 10 Clocking and PLL User Guide Archives](#) on page 66
Provides a list of user guides for previous versions of the ALTPLL_RECONFIG IP core.

4.3.1. Obtaining the Resource Utilization Report

For details about the resource usage and performance of the ALTPLL_RECONFIG IP core, refer to the compilation reports in the Intel Quartus Prime software.

To view the compilation reports for the ALTPLL_RECONFIG IP core in the Intel Quartus Prime software, follow these steps:

1. On the Processing menu, click **Start Compilation** to run a full compilation.
2. After compiling the design, on the Processing menu, click **Compilation Report**.
3. In the Table of Contents browser, expand the Fitter folder by clicking the "+" icon.
4. Under **Fitter**, expand **Resource section**, and select **Resource Usage Summary** to view the resource usage information.
5. Under **Fitter**, expand **Resource section**, and select **Resource Utilization by Entity** to view the resource utilization information.

4.4. Internal Oscillator Intel FPGA IP Core

The Internal Oscillator IP core specifies the internal oscillator frequencies for the devices.

Related Information

- [Introduction to Intel FPGA IP Cores](#)
Provides general information about all Intel FPGA IP cores, including parameterizing, generating, upgrading, and simulating IP cores.
- [Intel MAX 10 Clocking and PLL User Guide Archives](#) on page 66
Provides a list of user guides for previous versions of the Internal Oscillator IP core.

5. ALTCLKCTRL Intel FPGA IP Core References

5.1. ALTCLKCTRL IP Core Parameters

Table 13. ALTCLKCTRL IP Core Parameters for Intel MAX 10 Devices

This table lists the IP core parameters applicable to Intel MAX 10 devices.

Parameter	Value	Description
How do you want to use the ALTCLKCTRL	For global clock, or For external path	Specify the ALTCLKCTRL buffering mode. You can select from the following modes: <ul style="list-style-type: none"> For global clock—Allows a clock signal to reach all parts of the chip with the same amount of skew; you can select input port <code>clkselect</code> to switch between the four clock inputs. For external path—Represents the clock path from the outputs of the PLL to the dedicated clock output pins; only one clock output is accepted.
How many clock inputs would you like?	1, 2, 3, or 4	Specify the number of input clock sources for the clock control block. You can specify up to four clock inputs. You can change the number of clock inputs only if you choose For global clock option.
Create 'ena' port to enable or disable the clock network driven by this buffer	On or Off	Turn on this option if you want to create an active high clock enable signal to enable or disable the clock network.
Ensure glitch-free switchover implementation	On or Off	Turn on this option to implement a glitch-free switchover when you use multiple clock inputs. You must ensure the currently selected clock is running before switching to another source. If the selected clock is not running, the glitch-free switchover implementation will not be able to switch to the new clock source. By default, the <code>clkselect</code> port is set to 00. A clock must be applied to <code>inclk0x</code> for the values on the <code>clkselect</code> ports to be read.

Related Information

- [Global Clock Control Block](#) on page 9
- [Global Clock Network Power Down](#) on page 11
- [Clock Enable Signals](#) on page 12
- [Guideline: Clock Enable Signals](#) on page 34

5.2. ALTCLKCTRL IP Core Ports and Signals

Table 14. ALTCLKCTRL Input Ports for Intel MAX 10 Devices

Port Name	Condition	Description
clkselect[]	Optional	<p>Input that dynamically selects the clock source to drive the clock network that is driven by the clock buffer.</p> <p>Input port [1 DOWNTO 0] wide.</p> <p>If omitted, the default is GND.</p> <p>If this signal is connected, only the global clock network can be driven by this clock control block.</p> <p>The following list shows the signal selection for the binary value:</p> <ul style="list-style-type: none"> • 00—inclclk[0] • 01—inclclk[1] • 10—inclclk[2] • 11—inclclk[3]
ena	Optional	<p>Clock enable of the clock buffer.</p> <p>If omitted, the default value is V_{CC}.</p>
inclclk[]	Required	<p>Clock input of the clock buffer.</p> <p>Input port [3 DOWNTO 0] wide.</p> <p>You can specify up to four clock inputs, inclclk[3..0].</p> <p>Clock pins, clock outputs from the PLL, and core signals can drive the inclclk[] port.</p> <p>Multiple clock inputs are only supported for the global clock networks.</p>

Table 15. ALTCLKCTRL Output Ports for Intel MAX 10 Devices

Port Name	Condition	Description
outclk	Required	Output of the clock buffer.

Related Information

- [Global Clock Control Block](#) on page 9
- [Global Clock Network Power Down](#) on page 11
- [Clock Enable Signals](#) on page 12
- [Guideline: Clock Enable Signals](#) on page 34

6. ALTPLL Intel FPGA IP Core References

6.1. ALTPLL IP Core Parameters

The following tables list the IP core parameters applicable to Intel MAX 10 devices.

6.1.1. Operation Modes Parameter Settings

You can set the operation mode for PLL in the **General/Modes** page of the ALTPLL IP core parameter editor.

Table 16. Operation Mode Parameter Editor Settings

Parameter	Value	Description
Which device speed grade will you be using?	Any, 7, or 8	Specify the speed grade if you are not already using a device with the fastest speed. The lower the number, the faster the speed grade.
What is the frequency of the inclock0 input?	—	Specify the frequency of the input clock signal.
Use the feedback path inside the PLL	In normal mode, In source-synchronous compensation mode, In zero-delay buffer mode, or With no compensation	Specify which operation mode to use. For source-synchronous mode and zero-delay buffer mode, you must make PLL Compensation assignments using the Assignment Editor in addition to setting the appropriate mode in the IP core. The assignment allows you to specify an output pin as a compensation target for a PLL in zero-delay buffer mode, or to specify an input pin or group of input pins as compensation targets for a PLL in source-synchronous mode.
Which output clock will be compensated for?	C0, C1, C2, C3, or C4	Specify which PLL output port to compensate. The drop down list contains all output clock ports for the selected device. The correct output clock selection depends on the operation mode that you select. For example, for normal mode, select the core output clock. For zero-delay buffer mode, select the external output clock.

Related Information

[Clock Feedback Modes](#) on page 19

6.1.2. PLL Control Signals Parameter Settings

The parameter settings for the control signals are located on the **Inputs/Lock** page of the ALTPLL IP core parameter editor.

Turn on the control signal you want to create from the options available.

Related Information

[PLL Control Signals](#) on page 18

6.1.3. Programmable Bandwidth Parameter Settings

You can configure the bandwidth of the ALTPLL IP core on the **Bandwidth/SS** page of the ALTPLL IP core parameter editor.

Table 17. Bandwidth Configuration Parameter Editor Settings

Parameter	Value	Description
Auto	—	The ALTPLL parameter editor chooses the best possible bandwidth values to achieve the desired PLL settings. In some cases, you can get a bandwidth value outside the Low and High preset range. You can use the programmable bandwidth feature with the clock switchover feature to get the PLL output settings that you desire. You must set the bandwidth to Auto if you want to enable the spread-spectrum feature.
Preset	Low	PLL with a low bandwidth has better jitter rejection but a slower lock time.
	Medium	PLL with a medium bandwidth has a balance between lock time and jitter rejection.
	High	PLL with a high bandwidth has a faster lock time but tracks more jitter.

The table on the right in the **Bandwidth/SS** page shows the values of the following components:

- Charge pump current
- Loop filter resistance
- Loop filter capacitance
- M counter

These parameter settings create no additional top-level ports.

Related Information

- [Programmable Bandwidth](#) on page 25
- [Programmable Bandwidth with Advanced Parameters](#) on page 41
- [Charge Pump and Loop Filter](#) on page 44

6.1.4. Clock Switchover Parameter Settings

The parameter settings for clock switchover feature are located on the **Clock switchover** page of the ALTPLL IP core parameter editor.

Table 18. Clock Switchover Parameter Editor Settings

Parameter	Value	Description
Create an 'inclk1' input for a second input clock	On or Off	Turn on this option to enable the switchover feature. The <code>inclk0</code> signal is by default the primary input clock signal of the ALTPLL IP core.
Create a 'clkswitch' input to manually select between the input clocks	—	Select this option for manual clock switchover mode.
Allow PLL to automatically control the switching between input clocks	—	Select this option for automatic clock switchover mode. The automatic switchover is initiated during loss of lock or when the <code>inclk0</code> signal stops toggling.
<i>continued...</i>		

Parameter	Value	Description
Create a 'clkswitch' input to dynamically control the switching between input clocks	On or Off	Turn on this option for automatic clock switchover with manual override mode. The automatic switchover is initiated during loss of lock or when the <code>clkswitch</code> signal is asserted.
Perform the input clock switchover after (number) input clock cycles	On or Off	Turn on this option to specify the number of clock cycles to wait before the PLL performs the clock switchover. The allowed number of clock cycles to wait is device-dependent.
Create an 'activeclock' output to indicate the input clock being used	On or Off	Turn on this option to monitor which input clock signal is driving the PLL. When the current clock signal is <code>inclk0</code> , the <code>activeclock</code> signal is low. When the current clock signal is <code>inclk1</code> , the <code>activeclock</code> signal is high.
Create a 'clkbad' output for each input clock	On or Off	Turn on this option to monitor when the input clock signal has stopped toggling. The <code>clkbad0</code> signal monitors the <code>inclk0</code> signal. The <code>clkbad1</code> signal monitors the <code>inclk1</code> signal. The <code>clkbad0</code> signal goes high when the <code>inclk0</code> signal stops toggling. The <code>clkbad1</code> signal goes high when the <code>inclk1</code> signal stops toggling. The <code>clkbad</code> signals remain low when the input clock signals are toggling.

Related Information

- [Clock Switchover](#) on page 27
- [Guideline: Clock Switchover](#) on page 37

6.1.5. PLL Dynamic Reconfiguration Parameter Settings

The parameter settings for the normal dynamic reconfiguration scheme are located on the **PLL Reconfiguration** page of the ALTPLL IP core parameter editor.

Table 19. PLL Dynamic Reconfiguration Parameter Editor Settings

Parameter	Value	Description
Create optional inputs for dynamic reconfiguration	On or Off	Turn on this option to enable all the PLL reconfiguration ports for this instantiation— <code>scanclk</code> , <code>scanclkena</code> , <code>scandata</code> , <code>scandone</code> , <code>scandataout</code> , and <code>configupdate</code> .
Initial Configuration File	—	Specify the location of the configuration file that is used to initialize the ALTPLL_RECONFIG IP core.
Additional Configuration File(s)	—	Specify additional configuration file. This file might contain additional settings for the PLL, or might be used to initialize the ALTPLL_RECONFIG IP core.

Related Information

- [PLL Reconfiguration](#) on page 32
- [Dynamic Phase Configuration Implementation](#) on page 46

6.1.6. Dynamic Phase Configuration Parameter Settings

The parameter settings to enable the dynamic phase configuration feature are located on the **PLL Reconfiguration** page of the ALTPLL IP core parameter editor.

Table 20. Dynamic Phase Configuration Parameter Editor Settings

Parameter	Value	Description
Create optional inputs for dynamic phase reconfiguration	On or Off	Turn on this option to enable the dynamic phase configuration feature. The following ports are created: <ul style="list-style-type: none"> phasecounterselect[2..0] phaseupdown phasetest scanclk phasedone
Enable phase shift step resolution edit	On or Off	Turn on this option to modify the value for Phase shift step resolution(ps) for each individual PLL output clock on the Output Clocks page. By default, the finest phase shift resolution value is 1/8 of the VCO period. If the VCO frequency is at the lower end of the supported VCO range, the phase shift resolution might be larger than preferred for your design. Use this option to fine tune the phase shift step resolution.

Related Information

- [Programmable Phase Shift](#) on page 25
- [Dynamic Phase Configuration Implementation](#) on page 46

6.1.7. Output Clocks Parameter Settings

The **Output Clocks** page of the ALTPLL parameter editor contains the parameter settings of the clock output signals. You can configure the c0, c1, c2, c3, and c4 clock output signals of the ALTPLL IP core.

Each option has the following two columns:

- Requested settings—The settings that you want to implement.
- Actual settings—The settings closest values that can be implemented in the PLL circuit to best approximate the requested settings.

Use the values in the actual settings column as a guide to adjust the requested settings. If the requested settings for one of the output clocks cannot be approximated, the ALTPLL IP core parameter editor produces a warning message at the top of every page.

Table 21. Output Clocks Parameter Editor Settings

Parameter	Value	Description
Use this clock	On or Off	Turn on this option to generate an output clock port in your ALTPLL instance. The output clock port that is to be compensated for is enabled by default. It cannot be disabled, unless you select a different output clock port to be compensated for.
Enter output clock frequency	—	Specify the frequency of the output clock signal.
Enter output clock parameters	—	Specify the output clock parameters instead of the frequency.
Clock multiplication factor	—	Specify the clock multiplication factor of the signal.
<i>continued...</i>		

Parameter	Value	Description
Clock division factor	—	Specify the clock division factor of the signal.
Clock phase shift	—	Set the programmable phase shift for an output clock signals. The smallest phase shift is 1/8 of VCO period. For degree increments, the maximum step size is 45 degrees. You can set smaller steps using the Clock multiplication factor and Clock division factor options. For example, if the post-scale counter is 32, the smallest phase shift step is 0.1°. The up and down buttons let you cycle through phase shift values. Alternatively, you can enter a number in the phase shift field manually instead of using the buttons.
Clock duty cycle (%)	—	Set the duty cycle of the output clock signal.
Per Clock Feasibility Indicators	—	Indicate output clocks that contain unachievable settings. The output clock name in red is the name of the clock with unachievable settings. The clock listed in green has no settings issues, and the grayed-out names are the unselected output clocks. You must adjust the requested settings for the affected output clocks to resolve the warning messages.

The ALTPLL IP core parameter editor calculates the simplest fraction, and displays it in the actual settings column. You can use the copy button to copy values from the actual settings to the requested settings.

Figure 30. PLL Output Clock Frequency

$$\text{Output clock frequency} = \text{Input clock frequency} \times \frac{\text{Multiplication factor}}{\text{Division factor}}$$

For example, if the input clock frequency is 100 MHz, and the requested multiplication and division factors are 205 and 1025 respectively, the output clock frequency is calculated as $100 \times 205/1025=20$ MHz. The actual settings reflect the simplest fraction—the actual multiplication factor is 1, and the actual division factor is 5.

6.2. ALTPLL IP Core Ports and Signals

Table 22. ALTPLL Input Ports for Intel MAX 10 Devices

Port Name ⁽¹⁰⁾	Condition	Description
areset	Optional	Resets all counters to initial values, including the GATE_LOCK_COUNTER parameter.
clkswitch	Optional	The control input port to dynamically toggle between clock input ports (inclk0 and inclk1 ports), or to manually override the automatic clock switchover. You should create the clkswitch port if only the inclk1 port is created.
configupdate	Optional	Dynamic full PLL reconfiguration.
inclk[]	Required	The clock inputs that drive the clock network.

continued...

⁽¹⁰⁾ Replace brackets, [], in the port name with integer to get the exact name. For example, inclk0 and inclk1.

Port Name ⁽¹⁰⁾	Condition	Description
		If more than one <code>inclk[]</code> port is created, you must use the <code>clkselect</code> port to specify which clock is used. The <code>inclk0</code> port must always be connected; connect other clock inputs if switching is necessary. A dedicated clock pin or PLL output clock can drive this port.
<code>pfdena</code>	Optional	Enables the phase frequency detector (PFD). When the PFD is disabled, the PLL continues to operate regardless of the input clock. Because the PLL output clock frequency does not change for some time, you can use the <code>pfdena</code> port as a shutdown or cleanup function when a reliable input clock is no longer available.
<code>phasecounterselect[]</code>	Optional	Specifies counter select. You can use the <code>phasecounterselect[2..0]</code> bits to select either the <code>M</code> or one of the <code>C</code> counters for phase adjustment. One address map to select all <code>C</code> counters. This signal is registered in the PLL on the rising edge of <code>SCANCLK</code> .
<code>phasetest</code>	Optional	Specifies dynamic phase shifting. Logic high enables dynamic phase shifting.
<code>phaseupdown</code>	Optional	Specifies dynamic phase shift direction. 1 = UP, 0 = DOWN. Signal is registered in the PLL on the rising edge of <code>SCANCLK</code> .
<code>scanclk</code>	Optional	Input clock port for the serial scan chain. Free-running clock from core used in combination with <code>PHASESTEP</code> to enable or disable dynamic phase shifting. Shared with <code>SCANCLK</code> for dynamic reconfiguration.
<code>scanclkena</code>	Optional	Clock enable port for the serial scan chain.
<code>scandata</code>	Optional	Contains the data for the serial scan chain.

Table 23. ALTPLL Output Ports for Intel MAX 10 Devices

Port Name ⁽¹¹⁾	Condition	Description
<code>activeclock</code>	Optional	Specifies which clock is the primary reference clock when the clock switchover circuit initiates. If the <code>inclk0</code> is in use, the <code>activeclock</code> port goes low. If the <code>inclk1</code> is in use, the <code>activeclock</code> port goes high. You can set the PLL to automatically initiate the clock switchover when the primary reference clock is not toggling correctly, or you can manually initiate the clock switchover using the <code>clkswitch</code> input port.
<code>c[]</code>	Required	The clock output of the PLL.
<code>clkbad[]</code>	Optional	<code>clkbad1</code> and <code>clkbad0</code> ports check for input clock toggling. If the <code>inclk0</code> port stops toggling, the <code>clkbad0</code> port goes high. If the <code>inclk1</code> port stops toggling, the <code>clkbad1</code> port goes high.

continued...

⁽¹⁰⁾ Replace brackets, `[]`, in the port name with integer to get the exact name. For example, `inclk0` and `inclk1`.

⁽¹¹⁾ Replace the brackets, `[]`, in the port name with an integer to get the exact name (for example, `c0` and `c1`).

Port Name ⁽¹¹⁾	Condition	Description
locked	Optional	<p>This output port acts as an indicator when the PLL has reached phase-locked. The <code>locked</code> port stays high as long as the PLL is locked, and stays low when the PLL is out-of-lock.</p> <p>The number of cycles needed to gate the <code>locked</code> signal is based on the PLL input clock. The gated-lock circuitry is clocked by the PLL input clock. The maximum lock time for the PLL is provided in the <i>Intel MAX 10 FPGA Device Datasheet</i>.</p> <p>Take the maximum lock time of the PLL and divide it by the period of the PLL input clock. The result is the number of clock cycles needed to gate the <code>locked</code> signal.</p> <p>The lock signal is an asynchronous output of the PLL. The PLL lock signal is derived from the reference clock and feedback clock feeding the phase frequency detector (PFD) as follows:</p> <ul style="list-style-type: none"> Reference clock = Input Clock/N Feedback clock = VCO/M <p>The PLL asserts the <code>locked</code> port when the phases and frequencies of the reference clock and feedback clock are the same or within the lock circuit tolerance. When the difference between the two clock signals goes beyond the lock circuit tolerance, the PLL loses lock.</p>
phasedone	Optional	<p>This output port indicates that dynamic phase reconfiguration is completed.</p> <p>When <code>phasedone</code> signal is asserted, it indicates to core logic that the phase adjustment is complete and PLL is ready to act on a possible second adjustment pulse. This signal asserts based on internal PLL timing and deasserts on rising edge of <code>SCANCLK</code>.</p>
scandataout	Optional	<p>The data output for the serial scan chain.</p> <p>You can use the <code>scandataout</code> port to determine when PLL reconfiguration completes. The last output is cleared when reconfiguration completes.</p>
scandone	Optional	<p>This output port indicates that the scan chain write operation is initiated.</p> <p>The <code>scandone</code> port goes high when the scan chain write operation initiates, and goes low when the scan chain write operation completes.</p>

Related Information

[PLL Control Signals](#) on page 18

⁽¹¹⁾ Replace the brackets, [], in the port name with an integer to get the exact name (for example, `c0` and `c1`).

7. ALTPLL_RECONFIG Intel FPGA IP Core References

7.1. ALTPLL_RECONFIG IP Core Parameters

Table 24. ALTPLL_RECONFIG IP Core Parameters for Intel MAX 10 Devices

This table lists the IP core parameters applicable to Intel MAX 10 devices.

Page	Parameter	Value	Description
Parameter Settings	Currently Selected Device Family	—	Specifies the chosen device family.
	Which scan chain type will you be using?	—	The scan chain is a serial shift register chain that is used to store settings. It acts like a cache. When you assert the <code>reconfig</code> signal, the PLL is reconfigured with the values in the cache. The type of scan chain must follow the type of PLL to be reconfigured. The scan chain type has a default value of Top/Bottom .
	Do you want to specify the initial value of the scan chain?	No, leave it blank, Yes, use this file for the content data	Specifies the initial value of the scan chain. Select No, leave it blank to not specify a file or select Yes, use this file for the content data to browse for a <code>.hex</code> or <code>.mif</code> file. The option to initialize from a ROM is not available. However, you can choose to add ports to write to the scan chain from an external ROM during runtime by turning on Add ports to write to the scan chain from external ROM during run time .
	Add ports to write to the scan chain from external ROM during run time	On, Off	Turn on this option to take advantage of cycling multiple configuration files, which are stored in external ROMs during user mode.
EDA	Simulation Libraries	—	Specifies the libraries for functional simulation.
	Generate netlist	On, Off	Turn on this option to generate synthesis area and timing estimation netlist.
Summary	—	—	Specifies the types of files to be generated. A gray checkmark indicates a file that is automatically generated; an unchecked check box indicates an optional file. Choose from the following types of files: <ul style="list-style-type: none"> AHDL Include file (<code><function name>.inc</code>) VHDL component declaration file (<code><function name>.cmp</code>) Intel Quartus Prime symbol file (<code><function name>.bsf</code>) Instantiation template file (<code><function name>_inst.v</code> or <code><function name>_inst.vhd</code>) Verilog HDL black box file (<code><function name>_bb.v</code>) If the Generate netlist option is turned on, the file for that netlist is also available (<code><function name>_syn.v</code>).

Related Information

- Programmable Phase Shift on page 25

- [Dynamic Phase Configuration Implementation](#) on page 46
- [PLL Reconfiguration](#) on page 32
- [Dynamic Phase Configuration Implementation](#) on page 46

7.2. ALTPLL_RECONFIG IP Core Ports and Signals

Table 25. ALTPLL_RECONFIG Input Ports for Intel MAX 10 Devices

Port Name	Condition	Description
clock	Required	Clock input for loading individual parameters. This signal also clocks the PLL during reconfiguration. The clock input port must be connected to a valid clock. Refer to the <i>Intel MAX 10 FPGA Device Datasheet</i> for the clock f_{MAX} .
reset	Required	Asynchronous reset input to the IP core. Intel recommends that you reset this IP core before first use to guarantee that it is in a valid state. However, it does power up in the reset state. This port must be connected.
data_in[]	Optional	Data input that provides parameter value when writing parameters. This 9-bit input port provides the data to be written to the scan cache during a write operation. The bit width of the counter parameter to be written determines the number of bits of data_in[] that are read into the cache. For example, the low bit count of the C0 counter is 8-bit wide, so data_in[7..0] is read to the correct cache location. The bypass mode for the C0 counter is 1-bit wide, so data_in[0] is read for the value of this parameter. If omitted, the default value is 0.
counter_type[]	Optional	Specifies the counter type. An input port in the form of a 4-bit bus that selects which counter type should be selected for the corresponding operation (read, write, or reconfig). Refer to the counter_type[3..0] settings table for the mapping between the counter_type value and the physical counter to be set.
counter_param[]	Optional	Specifies the parameter for the value specified in the counter_type port. An input port in the form of a 3-bit bus that selects which parameter for the given counter type should be updated. The mapping to each parameter type and the corresponding parameter bit-width are defined in the counter_param[3..0] settings table.
read_param	Optional	Reads the parameter specified with the counter_type and counter_param ports from cache and fed to the data_out[] port. When asserted, the read_param signal indicates that the scan cache should be read and fed to data_out[]. The bit location of the scan cache and the number of bits read and sent to data_out[] depend on the counter_type and counter_param values. The read_param signal is sampled at the rising clock edge. If the read_param signal is asserted, the parameter value is read from the cache. Assert the read_param signal for 1 clock cycle only to prevent the parameter from being read twice. The busy signal is asserted on the rising clock edge following the assertion of the read_param signal. While the parameter is read, the busy signal remains asserted. After the busy signal is deasserted, the value on data_out[] is valid and the next parameter can be loaded. While the busy signal is asserted, the value on data_out[] is not valid. When the read_param signal is asserted, the busy signal is only asserted on the following rising edge of the clock and not on the same clock cycle as the read_param signal.

continued...

Port Name	Condition	Description
write_param	Optional	<p>Writes the parameter specified with the counter_type and counter_param ports to the cache with the value specified on the data_in[] port.</p> <p>When asserted, the write_param signal indicates that the value on data_in[] should be written to the parameter specified by counter_type[] and counter_param[]. The number of bits read from the data_in[] port depends on the parameter. The write_param signal is sampled at the rising clock edge. If the write_param signal is asserted, the parameter value is written to the cache. Assert the write_param signal for 1 clock cycle only to prevent the parameter from being written twice.</p> <p>The busy signal is asserted on the rising clock edge following the assertion of the write_param signal. While the parameter is being written, the busy signal remains asserted and input to data_in[] is ignored. After the busy signal is deasserted, the next parameter can be written.</p> <p>When the write_param signal is asserted, the busy signal is only asserted on the following rising edge of the clock. The busy signal is not asserted on the same clock cycle as the write_param signal.</p>
reconfig	Required	<p>Specifies that the PLL should be reconfigured with the PLL settings specified in the current cache.</p> <p>When asserted, the reconfig signal indicates that the PLL should be reconfigured with the values in the cache. The reconfig signal is sampled at the rising clock edge. If the reconfig signal is asserted, the cached settings are loaded in the PLL. Assert the reconfig signal for 1 clock cycle only to prevent reloading the PLL configuration. The busy signal is asserted on the rising clock edge following the assertion of the reconfig signal. While the PLL is being loaded, the busy signal remains asserted. After the busy signal is deasserted, the parameter values can be modified again.</p> <p>During and after reconfiguration, the scan chain data cache remains unchanged. This allows you to easily create a new set of reconfiguration settings using only one parameter.</p> <p>If write_param has not been asserted since the previous assertion of reconfig, the entire scan chain is shifted in to the PLL again.</p> <p>When the reconfig signal is asserted, the busy signal is only asserted on the following rising edge of the clock. The busy signal is not asserted on the same clock cycle as the reconfig signal.</p>
pll_areset_in	Optional	<p>Input signal indicating that the PLL should be reset.</p> <p>When asserted, the pll_areset_in signal indicates the PLL IP core should be reset. This port defaults to 0 if left unconnected. When using the ALTPLL_RECONFIG IP core in a design, you cannot reset the PLL in any other way. You must use this IP core port to manually reset the PLL.</p>
pll_scandone	Optional	<p>Input port for the ALTPLL_RECONFIG IP core. This port is driven by the PLL's scandone output signal and determines when the PLL is reconfigured.</p>
pll_scandataout	Required	<p>Input port driven by the scandataout signal from the ALTPLL IP core. Use this port to read the current configuration of the ALTPLL IP core. This input port holds the ALTPLL scan data output from the dynamically reconfigurable bits. The pll_scandataout port must be connected to the scandataout port of the PLL. The activity on this port can only be observed when the reconfig signal is asserted.</p>

Table 26. ALTPLL_RECONFIG Output Ports for Intel MAX 10 Devices

Port Name	Condition	Description
data_out[]	Optional	Data read from the cache when read_param is asserted.
continued...		

Port Name	Condition	Description
		This 9-bit output bus provides the parameter data to the user. When the <code>read_param</code> signal is asserted, the values on <code>counter_type[]</code> and <code>counter_param[]</code> determine the parameter value that is loaded from cache and driven on the <code>data_out[]</code> bus. When the IP core deasserts the <code>busy</code> signal, the appropriate bits of the bus (for example, [0] or [3..0]) hold a valid value.
<code>busy</code>	Optional	Indicates that the PLL is reading or writing a parameter to the cache, or is configuring the PLL. While the <code>busy</code> signal is asserted, no parameters can be read or written, and no reconfiguration can be initiated. Changes to the IP core can be made only when the <code>busy</code> signal is not asserted. The signal goes high when the <code>read_param</code> , <code>write_param</code> , or <code>reconfig</code> input port is asserted, and remains high until the specified operation is complete. In the case of a reconfiguration operation, the <code>busy</code> signal remains high until the <code>pll_areset</code> signal is asserted and then deasserted.
<code>pll_areset</code>	Required	Drives the <code>areset</code> port on the PLL to be reconfigured. The <code>pll_areset</code> port must be connected to the <code>areset</code> port of the ALTPLL IP core for the reconfiguration to function correctly. This signal is active high. The <code>pll_areset</code> is asserted when <code>pll_areset_in</code> is asserted, or, after reconfiguration, at the next rising clock edge after the <code>scandone</code> signal goes high. If you use the ALTPLL_RECONFIG IP core, use the <code>pll_areset</code> output port to drive the PLL <code>areset</code> port.
<code>pll_configupdate</code>	Optional	Drives the <code>configupdate</code> port on the PLL to be reconfigured. When asserted, the <code>pll_configupdate</code> port loads selected data to PLL configuration latches. The signal is asserted after the final data bit is sent out.
<code>pll_scanclk</code>	Required	Drives the <code>scanclk</code> port on the PLL to be reconfigured. For information about the maximum <code>scanclk</code> frequency for the various devices, refer to the respective device handbook.
<code>pll_scanclkena</code>	Optional	This port acts as a clock enable for the <code>scanclk</code> port on the PLL to be reconfigured. Reconfiguration begins on the first rising edge of <code>pll_scanclk</code> after <code>pll_scanclkena</code> assertion. On the first falling edge of <code>pll_scanclk</code> , after the deassertion of the <code>pll_scanclkena</code> signal, the IP core stops scanning data to the PLL.
<code>pll_scandata</code>	Required	Drives the <code>scandata</code> port on the PLL to be reconfigured. This output port from the IP core holds the scan data input to the PLL for the dynamically reconfigurable bits. The <code>pll_scandata</code> port sends <code>scandata</code> to the PLL. Any activity on this port can only be observed when the <code>reconfig</code> signal is asserted.

7.3. ALTPLL_RECONFIG IP Core Counter Settings

Table 27. `counter_type[3..0]` Settings for Intel MAX 10 Devices

Counter Selection	Binary	Decimal
N	0000	0
M	0001	1
CP/LF	0010	2
VCO	0011	3
C0	0100	4
continued...		

Counter Selection	Binary	Decimal
C1	0101	5
C2	0110	6
C3	0111	7
C4	1000	8
Illegal value	1001	9
Illegal value	1010	10
Illegal value	1011	11
Illegal value	1100	12
Illegal value	1101	13
Illegal value	1110	14
Illegal value	1111	15

Table 28. counter_param[2..0] Settings for Intel MAX 10 Devices

Counter Type	Counter Param	Binary	Decimal	Width (bits)
Regular counters (C0 - C4)	High count	000	0	8
	Low count	001	1	8
	Bypass	100	4	1
	Mode (odd/even division)	101	5	1
CP/LF	Charge pump unused	101	5	5
	Charge pump current	000	0	3
	Loop filter unused	100	4	1
	Loop filter resistor	001	1	5
	Loop filter capacitance	010	2	2
VCO	VCO post scale	000	0	1
M/N counters	High count	000	0	8
	Low count	001	1	8
	Bypass	100	4	1
	Mode (odd/even division)	101	5	1
	Nominal count	111	7	9

For even nominal count, the counter bits are automatically set as follows:

- $high_count = Nominalcount/2$
- $low_count = Nominalcount/2$

For odd nominal count, the counter bits are automatically set as follows:

- $high_count = (Nominalcount + 1)/2$
- $low_count = Nominalcount - high_count$
- odd/even division bit = 1

For nominal count = 1, bypass bit = 1.

8. Internal Oscillator Intel FPGA IP Core References

8.1. Internal Oscillator IP Core Parameters

Table 29. Internal Oscillator IP Core Parameters for Intel MAX 10 Devices

This table lists the IP core parameters applicable to Intel MAX 10 devices.

Parameter	Value	Description
Clock Frequency	<ul style="list-style-type: none"> 10M02, 10M04, 10M08, 10M16, and 10M25: 55 MHz, 116 MHz 10M40 and 10M50: 35 MHz, 77 MHz 	Specify the clock frequency for simulation.

8.2. Internal Oscillator IP Core Ports and Signals

Table 30. Internal Oscillator Input Port for Intel MAX 10 Devices

Port Name	Condition	Description
oscena	Required	Input control signal to turn on or turn off the internal oscillator.

Table 31. Internal Oscillator Output Port for Intel MAX 10 Devices

Port Name	Condition	Description
clkout	Optional	Output clock from the internal oscillator.

9. Intel MAX 10 Clocking and PLL User Guide Archives

If the table does not list a software version, the user guide for the previous software version applies.

Intel Quartus Prime Version	User Guide
18.0	Intel MAX 10 Clocking and PLL User Guide
17.1	Intel MAX 10 Clocking and PLL User Guide
16.0	MAX 10 Clocking and PLL User Guide
15.1	MAX 10 Clocking and PLL User Guide
15.0	MAX 10 Clocking and PLL User Guide
14.1	MAX 10 Clocking and PLL User Guide

10. Document Revision History for the Intel MAX 10 Clocking and PLL User Guide

Document Version	Intel Quartus Prime Version	Changes
2021.11.01	21.1	<ul style="list-style-type: none"> Added Y180 package information in the <i>PLL Locations for 10M16, 10M25, 10M40, and 10M50 Devices</i> diagram. Added description about PLL_CLKOUTn pin support in the <i>Zero-Delay Buffer Mode</i> section.
2021.02.09	18.0	<ul style="list-style-type: none"> Updated the note in the <i>PLL Locations for 10M02 (Single Power Supply U324 Package), 10M04, and 10M08 Devices</i> diagram. Updated the description for clkselect[] and inclk[] ports in the <i>ALTCLKCTRL Input Ports for Intel MAX 10 Devices</i> table.
2020.10.02	18.0	Updated the note to PLL 2, PLL 3, and PLL 4 in the <i>PLL Locations for 10M16, 10M25, 10M40, and 10M50 Devices</i> diagram.
2018.06.15	18.0	<ul style="list-style-type: none"> Added description in the <i>Internal Oscillator Overview</i> section. Defined clkena signal in the <i>Global Clock Network Power Down</i> section. Mentioned that the spread-spectrum signals must be generated externally in the <i>Spread-Spectrum Clocking</i> section. Updated the <i>Example of Automatic Switchover After Loss of Clock Detection</i> diagram. Corrected the parameter names in the <i>Expanding the PLL Lock Range</i> section. Updated the description in the <i>Bypassing PLL Counter</i> section. Defined t_{su} and t_h in the <i>Dynamic Phase Configuration Implementation</i> section. Renamed the following IP cores as per Intel rebranding: <ul style="list-style-type: none"> Renamed ALTCLKCTRL IP core to ALTCLKCTRL Intel FPGA IP core. Renamed ALTPLL IP core to ALTPLL Intel FPGA IP core. Renamed ALTPLL_RECONFIG IP core to ALTPLL_RECONFIG Intel FPGA IP core. Renamed Internal Oscillator IP core to Internal Oscillator Intel FPGA IP core. Updated SignalTap II to Signal Tap II.

Date	Version	Changes
December 2017	2017.12.29	<ul style="list-style-type: none"> Updated the title for the following PLL locations diagrams. <ul style="list-style-type: none"> PLL Locations for 10M02 Device (Except Single Power Supply U324 Package) PLL Locations for 10M02 (Single Power Supply U324 Package), 10M04, and 10M08 Devices Updated dedicated clock input pins for PLL1 in the <i>Clock Feedback Modes</i> section. Updated the description in the <i>PLL External Clock Output</i> section. Updated the value and description for the Clock Frequency parameter in the <i>Internal Oscillator IP Core Parameters for Intel MAX 10 Devices</i> table. Updated the following terms: <ul style="list-style-type: none"> Changed Qsys to Platform Designer (Standard) Changed TimeQuest Timing Analyzer to Timing Analyzer
February 2017	2017.02.21	Rebranded as Intel.
November 2015	2015.11.02	<ul style="list-style-type: none"> Removed the topics about the IP catalog and parameter editor, generating IP cores, and the files generated by the IP core, and added a link to <i>Introduction to Altera IP Cores</i>. Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i>.
June 2015	2015.06.12	Added connectivity restriction guideline to the PLL design considerations.
May 2015	2015.05.04	Rearranged the fine resolution phase shift equation.
December 2014	2014.12.15	<ul style="list-style-type: none"> Corrected the statement that if you do not use the dedicated clock input pins for clock input, you can also use them as general-purpose input or output pins. Added description in Internal Oscillator Architecture and Features to state that the internal ring oscillator operates up to 232 MHz and this frequency is not accessible. Added connectivity restrictions guideline for internal oscillator. Added Internal Oscillator IP Core parameter: Clock Frequency. Moved Internal Oscillator Frequencies table from Internal Oscillator Architecture and Features chapter to Intel MAX 10 FPGA Device Datasheet.
September 2014	2014.09.22	Initial release.



Intel® MAX® 10 JTAG Boundary-Scan Testing User Guide



Online Version



Send Feedback

UG-M10JTAG

ID: **683210**

Version: **2021.11.01**

Contents

1. Intel® MAX® 10 JTAG BST Overview.....	3
2. JTAG BST Architecture.....	4
2.1. JTAG Pins.....	4
2.2. JTAG Circuitry Functional Model.....	4
2.3. JTAG Boundary-Scan Register.....	5
2.3.1. Boundary-Scan Cells in Intel MAX 10 I/O Pin.....	5
3. BST Operation Control.....	7
3.1. JTAG IDCODE	7
3.2. JTAG Secure Mode.....	8
3.3. JTAG Private Instruction.....	8
3.4. JTAG Instructions.....	8
4. I/O Voltage Support in the JTAG Chain.....	10
5. Enabling and Disabling JTAG BST Circuitry.....	11
6. Guidelines for JTAG BST.....	12
7. Boundary-Scan Description Language Support.....	13
A. Document Revision History for the Intel MAX 10 JTAG Boundary-Scan Testing User Guide.....	14



1. Intel® MAX® 10 JTAG BST Overview

Intel® MAX® 10 devices support the IEEE Std.1149.1 (JTAG) boundary-scan testing (BST).

When you perform BST, you can test pin connections without using physical test probes and capture functional data during normal operation. The boundary-scan cells (BSCs) in a device can force signals onto pins, or capture data from pins or core logic signals. Forced test data is serially shifted in from the TDI pin to the BSCs. Captured data is serially shifted out to the TDO pin for external comparison with expected results.

Note: You can perform BST on Intel MAX 10 devices before, after, and during configuration.

Related Information

- [Intel MAX 10 FPGA Configuration User Guide](#)
Provides more information about JTAG in-system programming.
- [IEEE 1149.1 JTAG Boundary-Scan Testing in Altera Devices](#)
Provides more information on IEEE 1149.1 JTAG boundary-scan testing.
- [JTAG BST Architecture](#) on page 4
- [JTAG Boundary-Scan Register](#) on page 5
- [BST Operation Control](#) on page 7
- [I/O Voltage Support in the JTAG Chain](#) on page 10
- [Enabling and Disabling JTAG BST Circuitry](#) on page 11
- [Guidelines for JTAG BST](#) on page 12
- [Boundary-Scan Description Language Support](#) on page 13

2. JTAG BST Architecture

Intel MAX 10 JTAG interface uses four pins, TDI, TDO, TMS, and TCK.

2.1. JTAG Pins

Table 1. JTAG Pin Descriptions

Pin	Function	Description
TDI	Serial input pin for: <ul style="list-style-type: none"> Instructions Test data Programming data 	<ul style="list-style-type: none"> TDI is sampled on the rising edge of TCK TDI pins have internal weak pull-up resistors.
TDO	Serial output pin for: <ul style="list-style-type: none"> Instructions Test data Programming data 	<ul style="list-style-type: none"> TDO is sampled on the falling edge of TCK The pin is tri-stated if data is not being shifted out of the device.
TMS	Input pin that provides the control signal to determine the transitions of the TAP controller state machine.	<ul style="list-style-type: none"> TMS is sampled on the rising edge of TCK TMS pins have internal weak pull-up resistors.
TCK	The clock input to the BST circuitry.	—

All the JTAG pins are powered by the V_{CCIO} of I/O bank 1B. In JTAG mode, the I/O pins support the LVTTTL/LVCMOS 3.3-1.5V standards.

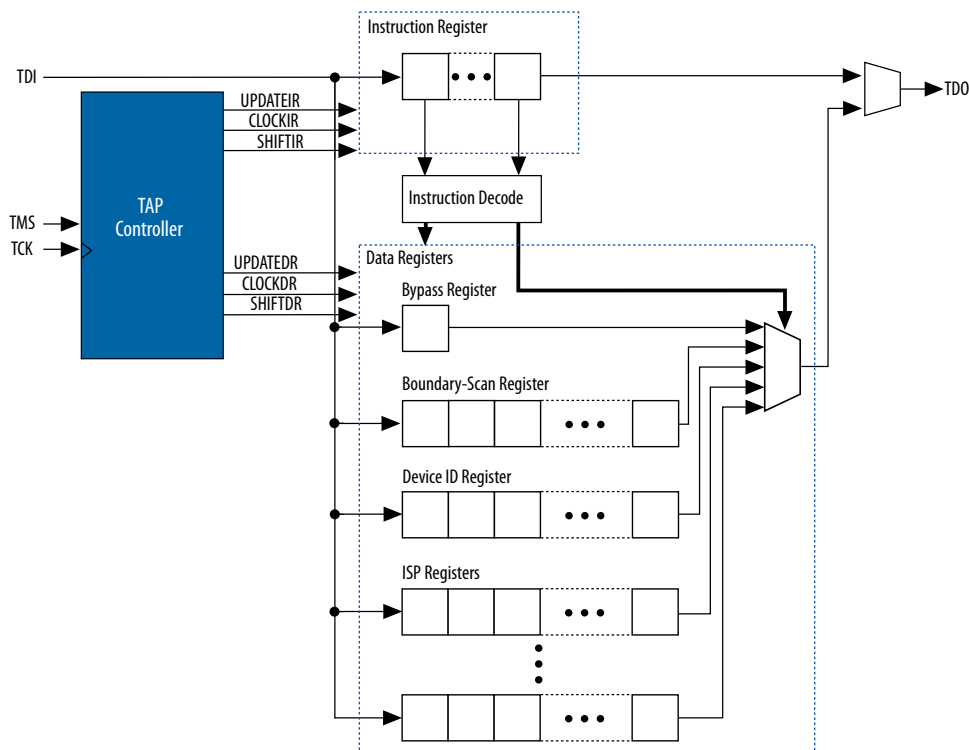
2.2. JTAG Circuitry Functional Model

The JTAG BST circuitry requires the following registers:

- Instruction register—determines which action to perform and which data register to access.
- Bypass register (1-bit long data register)—provides a minimum-length serial path between the TDI and TDO pins.
- Boundary-scan register—shift register composed of all the BSCs of the device.

Figure 1. JTAG Circuitry Functional Model

- Test access port (TAP) controller—controls the JTAG BST.
- TMS and TCK pins—operate the TAP controller.
- TDI and TDO pins—provide the serial path for the data registers.
- The TDI pin also provides data to the instruction register to generate the control logic for the data registers.



2.3. JTAG Boundary-Scan Register

You can use the boundary-scan register to test external pin connections or to capture internal data. The boundary-scan register is a large serial shift register that uses the TDI pin as an input and the TDO pin as an output. The boundary-scan register consists of 3-bit peripheral elements that are associated with Intel MAX 10 I/O pins.

2.3.1. Boundary-Scan Cells in Intel MAX 10 I/O Pin

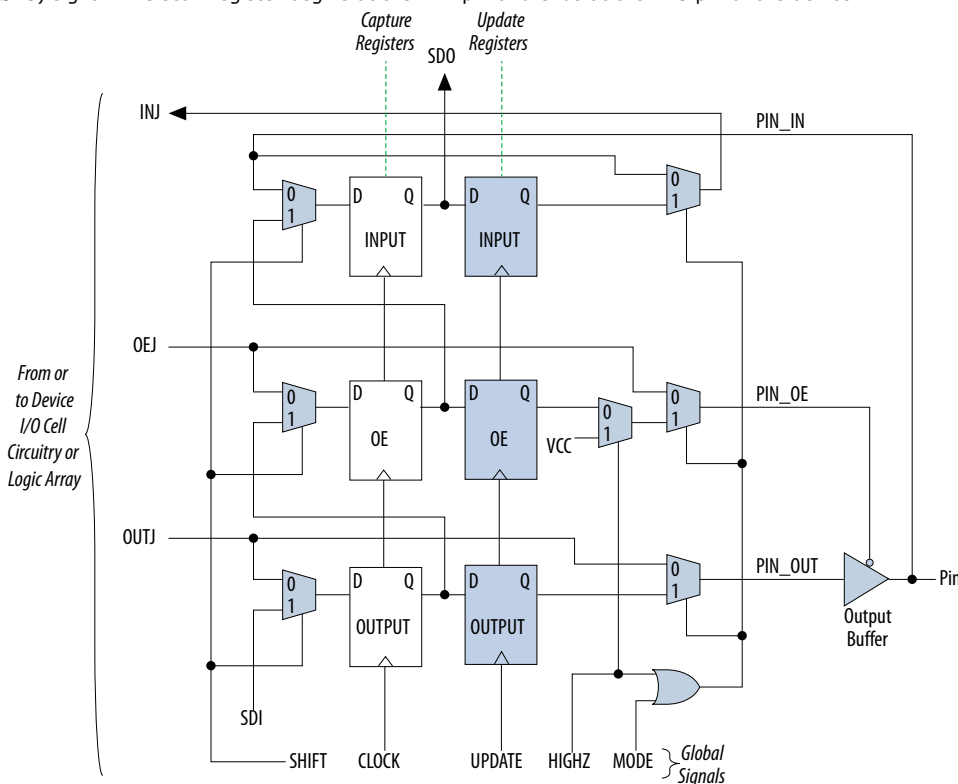
The Intel MAX 10 3-bit BSC contains the following registers:

- Capture registers—connect to internal device data through OUTJ, OEJ, and PIN_IN signals.
- Update registers—connect to external data through PIN_OUT and PIN_OE signals.

Figure 2. User I/O BSC with JTAG BST Circuitry for Intel MAX 10 Devices

The TAP controller generates the global control signals internally for the JTAG BST registers, *shift*, *clock*, and *update*. The instruction register generates the *MODE* signal.

The data signal path for the boundary-scan register runs from the serial data in (*SDI*) signal to the serial data out (*SDO*) signal. The scan register begins at the *TDI* pin and ends at the *TDO* pin of the device.


Table 2. BSC Capture and Update Register for Intel MAX 10 Devices

Pin Type	Captures			Drives		
	Output Capture Register	OE Capture Register	Input Capture Register	Output Update Register	OE Update Register	Input Update Register
User I/O	OUTJ	OEJ	PIN_IN	PIN_OUT	PIN_OE	INJ

Note: All VCC and GND pin types do not have BSCs.

3. BST Operation Control

Table 3. Boundary-Scan Register Length for Intel MAX 10 Devices

Device	Boundary-Scan Register Length
10M02	603
10M04	1080
10M08	732
10M16	1632
10M25	1164
10M40	1314
10M50	1620

3.1. JTAG IDCODE

The IDCODE is unique for each Intel MAX 10 device. Use this code to identify the devices in a JTAG chain.

Table 4. IDCODE Information for Intel MAX 10 Devices

Supply Option	Device	Device			
		Version (4 Bits)	Part Number (16 Bits)	Manufacturer Identity (11 Bits)	LSB (1 Bit)
Single-supply	10M02 (All except U324)	0000	0011 0001 1000 0001	000 0110 1110	1
	10M02 (U324)	0000	0011 0001 1001 1010	000 0110 1110	1
	10M04	0000	0011 0001 1000 1010	000 0110 1110	1
	10M08	0000	0011 0001 1000 0010	000 0110 1110	1
	10M16	0000	0011 0001 1000 0011	000 0110 1110	1
	10M25	0000	0011 0001 1000 0100	000 0110 1110	1
	10M40	0000	0011 0001 1000 1101	000 0110 1110	1
	10M50	0000	0011 0001 1000 0101	000 0110 1110	1
Dual-supply	10M02	0000	0011 0001 0000 0001	000 0110 1110	1
	10M04	0000	0011 0001 0000 1010	000 0110 1110	1
	10M08	0000	0011 0001 0000 0010	000 0110 1110	1
	10M16	0000	0011 0001 0000 0011	000 0110 1110	1

continued...

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.

**ISO
9001:2015
Registered**

Supply Option	Device	Device			
		Version (4 Bits)	Part Number (16 Bits)	Manufacturer Identity (11 Bits)	LSB (1 Bit)
	10M25	0000	0011 0001 0000 0100	000 0110 1110	1
	10M40	0000	0011 0001 0000 1101	000 0110 1110	1
	10M40DD	0001	0011 0001 0000 1101	000 0110 1110	1
	10M50	0000	0011 0001 0000 0101	000 0110 1110	1
	10M50DD	0001	0011 0001 0000 0101	000 0110 1110	1

3.2. JTAG Secure Mode

In JTAG secure mode, the device only allows SAMPLE/PRELOAD, BYPASS, EXTEST, and IDCODE JTAG instructions.

Related Information

[MAX 10 FPGA Configuration User Guide](#)

Provides more information about the JTAG Secure Mode.

3.3. JTAG Private Instruction

Caution: Never invoke the following instruction codes. These instructions can damage the device and render it unusable:

- 10 0100 0000
- 10 0011 0000
- 10 1110 0000
- 10 0011 0001

3.4. JTAG Instructions

Table 5. JTAG Instructions Supported by Intel MAX 10 Devices

Instruction Name	Instruction Binary	Description
SAMPLE/ PRELOAD	00 0000 0101	<ul style="list-style-type: none"> • Permits an initial data pattern to be an output at the device pins. • Allows you to capture and examine a snapshot of signals at the device pins if the device is operating in normal mode.
EXTEST ⁽¹⁾	00 0000 1111	<ul style="list-style-type: none"> • Forces test pattern at the output pins and capture the test results at the input pins. • Allows you to test the external circuitry and board-level interconnects.
BYPASS	11 1111 1111	<ul style="list-style-type: none"> • Places the 1-bit bypass register between the TDI and TDO pins. • Allows the BST data to pass synchronously through target devices to adjacent devices during normal device operation.

continued...

⁽¹⁾ HIGHZ, CLAMP, and EXTEST instructions do not disable weak pull-up resistors or bus hold features.

Instruction Name	Instruction Binary	Description
USERCODE	00 0000 0111	<ul style="list-style-type: none"> Places the 1-bit bypass register between the TDI and TDO pins. Allows you to shift the USERCODE register out of the TDO pin serially.
IDCODE	00 0000 0110	<ul style="list-style-type: none"> Selects the IDCODE register and places it between the TDI and TDO pins. Allows you to shift the IDCODE register out of the TDO pin serially.
HIGHZ ⁽¹⁾	00 0000 1011	<ul style="list-style-type: none"> Places the 1-bit bypass register between the TDI and TDO pins. The 1-bit bypass register tri-states all the I/O pins. Allow the BST data to pass synchronously through target devices to adjacent devices if device is operating in normal mode.
CLAMP ⁽¹⁾	00 0000 1010	<ul style="list-style-type: none"> Places the 1-bit bypass register between the TDI and TDO pins. The 1-bit bypass register holds I/O pins to a state defined by the data in the boundary-scan register. Allow the BST data to pass synchronously through target devices to adjacent devices if device is operating in normal mode.
USER0	00 0000 1100	<ul style="list-style-type: none"> Allows you to define the scan chain between the TDI and TDO pins in the Intel MAX 10 logic array. Use this instruction for custom logic and JTAG interfaces.
USER1	00 0000 1110	<ul style="list-style-type: none"> Allows you to define the scan chain between the TDI and TDO pins in the Intel MAX 10 logic array. Use this instruction for custom logic and JTAG interfaces.

4. I/O Voltage Support in the JTAG Chain

A JTAG chain can contain several Intel FPGA and non-Intel FPGA devices.

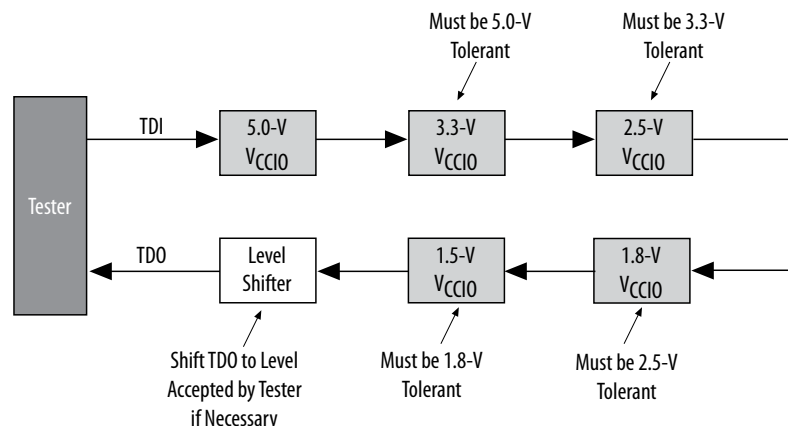
The TDO pin of a device drives out at the voltage level according to the V_{CCIO} of the device. The devices can interface with each other although the devices may have different V_{CCIO} levels.

For example, a device with 3.3-V V_{CCIO} can drive to a device with 5.0-V V_{CCIO} because 3.3 V meets the minimum V_{IH} on transistor-to-transistor logic (TTL)-level input for the 5.0-V V_{CCIO} device.

Intel MAX 10 devices can support 1.5-, 1.8-, 2.5-, or 3.3-V input levels, depending on the V_{CCIO} voltage of I/O Bank 1B.

To interface the TDI and TDO lines of the JTAG pins of devices that have different V_{CCIO} levels, insert a level shifter between the devices. If possible, construct the JTAG chain where device with a higher V_{CCIO} level drives to a device with an equal or lower V_{CCIO} level. In this setup, you only require a level shifter for shifting the TDO level to a level JTAG tester accept.

Figure 3. JTAG Chain of Mixed Voltages and Level Shifters



5. Enabling and Disabling JTAG BST Circuitry

The JTAG BST circuitry in Intel MAX 10 devices is automatically enabled after the power-up.

To ensure that you do not inadvertently enable the JTAG BST circuitry when it is not required, disable the circuitry permanently with pin connections as listed in the following table.

Table 6. Pin Connections to Permanently Disable the JTAG BST Circuitry in Intel MAX 10 Devices

JTAG Pins	Connection to Disable
TMS	V _{CCIO} supply of Bank 1B
TCK	GND
TDI	V _{CCIO} supply of Bank 1B
TDO	Leave open

You must enable this circuitry only if you use the BST or in-system programming (ISP) features.



6. Guidelines for JTAG BST

Consider the following guidelines when you perform BST with the device:

- If the "10..." pattern does not shift out of the instruction register through the TDO pin during the first clock cycle of the `SHIFT_IR` state, the TAP controller did not reach the proper state. To solve this problem, try one of the following procedures:
 - Verify that the TAP controller has reached the `SHIFT_IR` state correctly. To advance the TAP controller to the `SHIFT_IR` state, return TAP controller to the `RESET` state and send the 01100 code to the TMS pin.
 - Check the connections to the VCC, GND, JTAG, and dedicated configuration pins on the device.
- Perform a `SAMPLE/PRELOAD` test cycle before the first `EXTEST` test cycle to ensure that known data is present at the device pins when you enter `EXTEST` mode. If the `OEJ` update register contains 0, the data in the `OUTJ` update register is driven out. The state must be known and correct to avoid contention with other devices in the system.
- To perform testing before configuration, hold the `nCONFIG` pin low.



7. Boundary-Scan Description Language Support

The BSDL—a subset of VHDL—provides a syntax that allows you to describe the features of an IEEE Std. 1149.1 BST-capable device that can be tested. Test software development systems then use the BSDL files for test generation, analysis, failure diagnostics, and in-system programming.

Related Information

[IEEE 1149.1 BSDL Files](#)

Provides more information about BSC group definitions.

A. Document Revision History for the Intel MAX 10 JTAG Boundary-Scan Testing User Guide

Document Version	Changes
2021.11.01	Added 10M40DD and 10M50DD devices in the <i>IDCODE Information for Intel MAX 10 Devices</i> table.
2020.08.11	<ul style="list-style-type: none"> Removed table: <i>IDCODE Information for 32-Bit Intel MAX 10 Devices</i>. Removed 10M01 device from the <i>Boundary-Scan Register Length for Intel MAX 10 Devices</i> table.
2019.05.10	<ul style="list-style-type: none"> Renamed the document as <i>Intel MAX 10 JTAG Boundary-Scan Testing User Guide</i>. Added single-supply device "10M02 (U324)" in <i>IDCODE Information for Intel MAX 10 Devices</i> table. Updated single-supply device "10M02" to "10M02 (All except U324)" in <i>IDCODE Information for Intel MAX 10 Devices</i> table. Added references to TDI and TDO pins for description of forced test data and captured data in <i>Intel MAX 10 JTAG BST Overview</i> chapter.

Date	Version	Changes
February 2017	2017.02.21	Rebranded as Intel.
May 2015	2015.05.04	Added note on about performing the boundary-scan testing in 'Overview'.
September 2014	2014.09.22	Initial release.

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.

ISO
9001:2015
Registered



Intel® MAX® 10 General Purpose I/O User Guide

Updated for Intel® Quartus® Prime Design Suite: **22.1**



Online Version



Send Feedback

UG-M10GPIO

ID: **683751**

Version: **2022.10.31**

Contents

1. Intel® MAX® 10 I/O Overview.....	3
1.1. Intel MAX 10 Devices I/O Resources Per Package	4
1.2. Intel MAX 10 I/O Vertical Migration Support.....	5
2. Intel MAX 10 I/O Architecture and Features.....	6
2.1. Intel MAX 10 I/O Standards Support.....	6
2.1.1. Intel MAX 10 I/O Standards Voltage and Pin Support.....	9
2.2. Intel MAX 10 I/O Elements.....	12
2.2.1. Intel MAX 10 I/O Banks Architecture.....	13
2.2.2. Intel MAX 10 I/O Banks Performance.....	14
2.2.3. Intel MAX 10 I/O Banks Locations.....	14
2.3. Intel MAX 10 I/O Buffers.....	18
2.3.1. Schmitt-Trigger Input Buffer.....	18
2.3.2. Programmable I/O Buffer Features.....	18
2.4. I/O Standards Termination.....	26
2.4.1. Voltage-Referenced I/O Standards Termination.....	26
2.4.2. Differential I/O Standards Termination.....	27
2.4.3. Intel MAX 10 On-Chip I/O Termination.....	28
3. Intel MAX 10 I/O Design Considerations.....	31
3.1. Guidelines: V _{CCIO} Range Considerations.....	31
3.2. Guidelines: Voltage-Referenced I/O Standards Restriction.....	32
3.3. Guidelines: Enable Clamp Diode for LVTTTL/LVCMOS Input Buffers.....	32
3.4. Guidelines: Adhere to the LVDS I/O Restrictions Rules.....	33
3.5. Guidelines: I/O Restriction Rules.....	33
3.6. Guidelines: Placement Restrictions for 1.0 V I/O Pin.....	34
3.6.1. Calculating the Total Inductance for 1.0 V Pin Placement.....	34
3.7. Guidelines: Analog-to-Digital Converter I/O Restriction.....	36
3.8. Guidelines: External Memory Interface I/O Restrictions.....	39
3.9. Guidelines: Dual-Purpose Configuration Pin.....	39
3.10. Guidelines: Clock and Data Input Signal for Intel MAX 10 E144 Package.....	40
4. Intel MAX 10 I/O Implementation Guides.....	42
4.1. GPIO Lite Intel FPGA IP.....	42
4.1.1. GPIO Lite Intel FPGA IP Data Paths.....	43
4.2. Verifying Pin Migration Compatibility.....	45
5. GPIO Lite Intel FPGA IP References.....	47
5.1. GPIO Lite Intel FPGA IP Parameter Settings.....	47
5.2. GPIO Lite Intel FPGA IP Interface Signals.....	49
6. Intel MAX 10 General Purpose I/O User Guide Archives.....	51
7. Document Revision History for Intel MAX 10 General Purpose I/O User Guide.....	52



1. Intel® MAX® 10 I/O Overview

The Intel® MAX® 10 general purpose I/O (GPIO) system consists of the I/O elements (IOE) and the GPIO Lite Intel FPGA IP. You can use GPIOs in non-transceiver general applications, memory-like interfaces, or LVDS applications.

- The IOEs contain bidirectional I/O buffers and I/O registers located in I/O banks around the periphery of the device.
- The GPIO Lite IP core supports the GPIO components and features, including double data rate I/O (DDIO), delay chains, I/O buffers, control signals, and clocking.

Related Information

- [Intel MAX 10 I/O Architecture and Features](#) on page 6
Provides information about the architecture and features of the I/Os in Intel MAX 10 devices.
- [Intel MAX 10 I/O Design Considerations](#) on page 31
Provides I/O design guidelines for Intel MAX 10 Devices.
- [Intel MAX 10 I/O Implementation Guides](#) on page 42
Provides guides to implement I/Os in Intel MAX 10 Devices.
- [GPIO Lite Intel FPGA IP References](#) on page 47
Lists the parameters and signals of GPIO Lite IP core for Intel MAX 10 Devices.
- [Intel MAX 10 General Purpose I/O User Guide Archives](#) on page 51
Provides a list of user guides for previous versions of the GPIO Lite IP core.

1.1. Intel MAX 10 Devices I/O Resources Per Package

Table 1. Package Plan for Intel MAX 10 Single Power Supply Devices

Device	Package						
	Type	V81 81-pin WLCSP	Y180 180-pin WLCSP	M153 153-pin MBGA	U169 169-pin UBGA	U324 324-pin UBGA	E144 144-pin EQFP
	Size	4 mm × 4 mm	6 mm × 5 mm	8 mm × 8 mm	11 mm × 11 mm	15 mm × 15 mm	22 mm × 22 mm
	Ball Pitch	0.4 mm	0.35 mm	0.5 mm	0.8 mm	0.8 mm	0.5 mm
10M02		—	—	112	130	246	101
10M04		—	—	112	130	246	101
10M08		58	—	112	130	246	101
10M16		—	125	—	130	246	101
10M25		—	—	—	—	—	101
10M40		—	—	—	—	—	101
10M50		—	—	—	—	—	101

Table 2. Package Plan for Intel MAX 10 Dual Power Supply Devices

Device	Package						
	Type	V36 36-pin WLCSP (1)	V81 81-pin WLCSP (1)	U324 324-pin UBGA	F256 256-pin FBGA	F484 484-pin FBGA	F672 672-pin FBGA
	Size	3 mm × 3 mm	4 mm × 4 mm	15 mm × 15 mm	17 mm × 17 mm	23 mm × 23 mm	27 mm × 27 mm
	Ball Pitch	0.4 mm	0.4 mm	0.8 mm	1.0 mm	1.0 mm	1.0 mm
10M02		27	—	160	—	—	—
10M04		—	—	246	178	—	—
10M08		—	56	246	178	250	—
10M16		—	—	246	178	320	—
10M25		—	—	—	178	360	—
10M40		—	—	—	178	360	500
10M50		—	—	—	178	360	500

Related Information

[Intel FPGA Boards for Intel MAX 10 FPGAs](#)

⁽¹⁾ For the performance specifications of the V36 and V81 packages of Intel MAX 10 dual power supply devices, follow the data sheet specifications for single supply devices.

1.2. Intel MAX 10 I/O Vertical Migration Support

Figure 1. Migration Capability Across Intel MAX 10 Devices

- The arrows indicate the migration paths. The devices included in each vertical migration path are shaded. Non-migratable devices are omitted. Some packages have several migration paths. Devices with lesser I/O resources in the same path have lighter shades.
- To achieve the full I/O migration across product lines in the same migration path, restrict I/Os usage to match the product line with the lowest I/O count.

Device	Package									
	V36	V81	Y180	M153	U169	U324	F256	E144	F484	F672
10M02				↑	↑	↑				
10M04				↓	↓	↓	↑	↑		
10M08							↑	↑	↑	
10M16					↓	↓	↑	↑	↑	
10M25							↑	↑	↑	
10M40							↑	↑	↑	↑
10M50							↑	↑	↑	↑

Dual Power Supply Devices
 Single Power Supply Devices

Note: Before starting migration work, Intel recommends that you verify the pin migration compatibility through the **Pin Migration View** window in the Intel Quartus® Prime software Pin Planner. For example, not all Intel MAX 10 devices support 1.0 V I/O.

Related Information

[Verifying Pin Migration Compatibility](#) on page 45

2. Intel MAX 10 I/O Architecture and Features

The I/O system of Intel MAX 10 devices support various I/O standards. In the Intel MAX 10 devices, the I/O pins are located in I/O banks at the periphery of the devices. The I/O pins and I/O buffers have several programmable features.

Related Information

[Intel MAX 10 I/O Overview](#) on page 3

2.1. Intel MAX 10 I/O Standards Support

Intel MAX 10 devices support a wide range of I/O standards, including single-ended, voltage-referenced single-ended, and differential I/O standards.

Table 3. Supported I/O Standards in Intel MAX 10 Devices

The voltage-referenced I/O standards are not supported in the following I/O banks of these device packages:

- All I/O banks of V36 package of 10M02.
- All I/O banks of V81 package of 10M08.
- Banks 1A and 1B of E144 package of 10M50.

I/O Standard	Type	Device Support	Direction		Application	Standard Support
			Input	Output		
3.3 V LVTTTL/3.3 V LVCMOS	Single-ended	All	Yes	Yes	General purpose	JESD8-B
3.0 V LVTTTL/3.0 V LVCMOS	Single-ended	All	Yes	Yes	General purpose	JESD8-B
2.5 V LVCMOS	Single-ended	All	Yes	Yes	General purpose	JESD8-5
1.8 V LVCMOS	Single-ended	All	Yes	Yes	General purpose	JESD8-7
1.5 V LVCMOS	Single-ended	All	Yes	Yes	General purpose	JESD8-11
1.2 V LVCMOS	Single-ended	All	Yes	Yes	General purpose	JESD8-12
1.0 V LVCMOS ⁽²⁾	Single-ended	Specific devices ⁽²⁾	Yes	Yes	General purpose	—
3.0 V PCI	Single-ended	All	Yes	Yes	General purpose	PCI Rev. 2.2
3.3 V Schmitt Trigger	Single-ended	All	Yes	—	General purpose	—
2.5 V Schmitt Trigger	Single-ended	All	Yes	—	General purpose	—
continued...						

⁽²⁾ The 1.0 V LVCMOS I/O standard is available only for the following device combinations and subject to device OPN availability:
10M02/04/08/16/25/40/50 + SC/DC/SA/DA/DD + U324/U169/M153/F256/F484/F672 + I7/C7/C8

I/O Standard	Type	Device Support	Direction		Application	Standard Support
			Input	Output		
1.8 V Schmitt Trigger	Single-ended	All	Yes	—	General purpose	—
1.5 V Schmitt Trigger	Single-ended	All	Yes	—	General purpose	—
SSTL-2 Class I	Voltage-referenced	All	Yes	Yes	DDR1	JESD8-9B
SSTL-2 Class II	Voltage-referenced	All	Yes	Yes	DDR1	JESD8-9B
SSTL-18 Class I	Voltage-referenced	All	Yes	Yes	DDR2	JESD8-15
SSTL-18 Class II	Voltage-referenced	All	Yes	Yes	DDR2	JESD8-15
SSTL-15 Class I	Voltage-referenced	All	Yes	Yes	DDR3	—
SSTL-15 Class II	Voltage-referenced	All	Yes	Yes	DDR3	—
SSTL-15 ⁽³⁾	Voltage-referenced	All	Yes	Yes	DDR3	JESD79-3D
SSTL-135 ⁽³⁾	Voltage-referenced	All	Yes	Yes	DDR3L	—
1.8 V HSTL Class I	Voltage-referenced	All	Yes	Yes	DDR II+, QDR II+, and RLD RAM 2	JESD8-6
1.8 V HSTL Class II	Voltage-referenced	All	Yes	Yes	DDR II+, QDR II+, and RLD RAM 2	JESD8-6
1.5 V HSTL Class I	Voltage-referenced	All	Yes	Yes	DDR II+, QDR II+, QDR II, and RLD RAM 2	JESD8-6
1.5 V HSTL Class II	Voltage-referenced	All	Yes	Yes	DDR II+, QDR II+, QDR II, and RLD RAM 2	JESD8-6
1.2 V HSTL Class I	Voltage-referenced	All	Yes	Yes	General purpose	JESD8-16A
1.2 V HSTL Class II	Voltage-referenced	All	Yes	Yes	General purpose	JESD8-16A
HSUL-12 ⁽³⁾	Voltage-referenced	All	Yes	Yes	LPDDR2	—
Differential SSTL-2 Class I and II	Differential	All	Yes ⁽⁴⁾	Yes ⁽⁵⁾	DDR1	JESD8-9B
Differential SSTL-18 Class I and Class II	Differential	All	Yes ⁽⁴⁾	Yes ⁽⁵⁾	DDR2	JESD8-15

continued...

⁽³⁾ Available in Intel MAX 10 16, 25, 40, and 50 devices only.

⁽⁴⁾ The inputs treat differential inputs as two single-ended inputs and decode only one of them.

⁽⁵⁾ The outputs use two single-ended output buffers with the second output buffer programmed as inverted.

I/O Standard	Type	Device Support	Direction		Application	Standard Support
			Input	Output		
Differential SSTL-15 Class I and Class II	Differential	All	Yes ⁽⁴⁾	Yes ⁽⁵⁾	DDR3	—
Differential SSTL-15	Differential	All	Yes ⁽⁴⁾	Yes ⁽⁵⁾	DDR3	JESD79-3D
Differential SSTL-135	Differential	All	Yes ⁽⁴⁾	Yes ⁽⁵⁾	DDR3L	—
Differential 1.8 V HSTL Class I and Class II	Differential	All	Yes ⁽⁴⁾	Yes ⁽⁵⁾	DDR II+, QDR II+, and RLDRAM 2	JESD8-6
Differential 1.5 V HSTL Class I and Class II	Differential	All	Yes ⁽⁴⁾	Yes ⁽⁵⁾	DDR II+, QDR II+, QDR II, and RLDRAM 2	JESD8-6
Differential 1.2 V HSTL Class I and Class II	Differential	All	Yes ⁽⁴⁾	Yes ⁽⁵⁾	General purpose	JESD8-16A
Differential HSUL-12	Differential	All	Yes ⁽⁴⁾	Yes ⁽⁵⁾	LPDDR2	—
LVDS (dedicated) ⁽⁶⁾	Differential	All	Yes	Yes	—	ANSI/TIA/EIA-644
1.8 V LVDS (dedicated) ⁽⁷⁾	Differential	Dual supply devices ⁽⁷⁾	Yes	Yes	—	ANSI/TIA/EIA-644
LVDS (emulated, external resistors)	Differential	All	—	Yes	—	ANSI/TIA/EIA-644
Mini-LVDS (dedicated) ⁽⁶⁾	Differential	All	—	Yes	—	—
Mini-LVDS (emulated, external resistor)	Differential	Dual supply devices	—	Yes	—	—
RSDS (dedicated) ⁽⁶⁾	Differential	All	—	Yes	—	—
RSDS (emulated, external resistor, 1R)	Differential	Dual supply devices	—	Yes	—	—
RSDS (emulated, external resistors, 3R)	Differential	All	—	Yes	—	—
PPDS (dedicated) ⁽⁶⁾	Differential	Dual supply devices	—	Yes	—	—
PPDS (emulated, external resistor)	Differential	Dual supply devices	—	Yes	—	—
LVPECL	Differential	All	Yes	—	—	—
Bus LVDS	Differential	All	Yes	Yes ⁽⁸⁾	—	—

continued...

⁽⁶⁾ You can use dedicated LVDS transmitters only on the bottom I/O banks. You can use LVDS receivers on all I/O banks.

⁽⁷⁾ The 1.8 V LVDS buffers are supported as inputs on all high-speed I/O banks but as outputs only on the bottom banks. The low-speed and high-speed DDR3 I/O banks do not support 1.8 V LVDS. The 1.8 V LVDS I/O standard is supported in industrial- and commercial-grade Intel MAX 10 dual supply devices except in packages V36 and V81.

I/O Standard	Type	Device Support	Direction		Application	Standard Support
			Input	Output		
TMDS	Differential	Dual supply devices	Yes	—	—	—
Sub-LVDS	Differential	Dual supply devices	Yes	Yes ⁽⁹⁾	—	—
SLVS	Differential	Dual supply devices	Yes	Yes ⁽¹⁰⁾	—	—
HiSpi	Differential	Dual supply devices	Yes	—	—	—

Related Information

- [Intel MAX 10 I/O Buffers](#) on page 18
Provides more information about available I/O buffer types and supported I/O standards.
- [LVDS Transmitter I/O Termination Schemes, Intel MAX 10 High-Speed LVDS I/O User Guide](#)
Provides the required external termination schemes and resistor values for the emulated LVDS, Sub-LVDS, SLVS, emulated RSDS, emulated mini-LVDS, and emulated PPDS I/O standards.

2.1.1. Intel MAX 10 I/O Standards Voltage and Pin Support

Table 4. Intel MAX 10 I/O Standards Voltage Levels and Pin Support

Note: The I/O standards that each pin type supports depends on the I/O standards that the pin's I/O bank supports. For example, only the bottom I/O banks support the LVDS (dedicated) I/O standard. You can use the LVDS (dedicated) I/O standard for the PLL_CLKOUT pin only if the pin is available in your device's bottom I/O banks. To determine the pin's I/O bank locations for your device, check your device's pin out file.

I/O Standard	V _{CCIO} (V)		V _{REF} (V)	Pin Type Support				
	Input	Output		PLL_CLKOUT	MEM_CLK	CLK	DQS	User I/O
3.3 V LVTTL/3.3 V LVCMOS	3.3/3.0/2.5	3.3	—	Yes	Yes	Yes	Yes	Yes
3.0 V LVTTL/3.0 V LVCMOS	3.0/2.5	3.0	—	Yes	Yes	Yes	Yes	Yes
2.5 V LVCMOS	3.0/2.5	2.5	—	Yes	Yes	Yes	Yes	Yes

continued...

- ⁽⁸⁾ The outputs use two single-ended output buffers with the second output buffer programmed as inverted. A single series resistor is required.
- ⁽⁹⁾ Requires external termination resistors.
- ⁽¹⁰⁾ The outputs uses two single-ended output buffers as emulated differential outputs. Requires external termination resistors.

I/O Standard	V _{CCIO} (V)		V _{REF} (V)	Pin Type Support				
	Input	Output		PLL_CLKOUT	MEM_CLK	CLK	DQS	User I/O
1.8 V LVCMOS	1.8/1.5	1.8	—	Yes	Yes	Yes	Yes	Yes
1.5 V LVCMOS	1.8/1.5	1.5	—	Yes	Yes	Yes	Yes	Yes
1.2 V LVCMOS	1.2	1.2	—	Yes	Yes	Yes	Yes	Yes
1.0 V LVCMOS	1.0 ⁽¹¹⁾	1.0 ⁽¹¹⁾	—	—	—	Yes	—	Yes
3.0 V PCI	3.0	3.0	—	Yes	Yes	Yes	Yes	Yes
3.3 V Schmitt Trigger	3.3	—	—	—	—	Yes	Yes ⁽¹²⁾	Yes
2.5 V Schmitt Trigger	2.5	—	—	—	—	Yes	Yes ⁽¹²⁾	Yes
1.8 V Schmitt Trigger	1.8	—	—	—	—	Yes	Yes ⁽¹²⁾	Yes
1.5 V Schmitt Trigger	1.5	—	—	—	—	Yes	Yes ⁽¹²⁾	Yes
SSTL-2 Class I	2.5	2.5	1.25	Yes	Yes	Yes	Yes	Yes
SSTL-2 Class II	2.5	2.5	1.25	Yes	Yes	Yes	Yes	Yes
SSTL-18 Class I	1.8	1.8	0.9	Yes	Yes	Yes	Yes	Yes
SSTL-18 Class II	1.8	1.8	0.9	Yes	Yes	Yes	Yes	Yes
SSTL-15 Class I	1.5	1.5	0.75	Yes	Yes	Yes	Yes	Yes
SSTL-15 Class II	1.5	1.5	0.75	Yes	Yes	Yes	Yes	Yes
SSTL-15	1.5	1.5	0.75	Yes	Yes	Yes	Yes	Yes
SSTL-135	1.35	1.35	0.675	Yes	Yes	Yes	Yes	Yes
1.8 V HSTL Class I	1.8	1.8	0.9	Yes	Yes	Yes	Yes	Yes
1.8 V HSTL Class II	1.8	1.8	0.9	Yes	Yes	Yes	Yes	Yes
1.5 V HSTL Class I	1.5	1.5	0.75	Yes	Yes	Yes	Yes	Yes
1.5 V HSTL Class II	1.5	1.5	0.75	Yes	Yes	Yes	Yes	Yes
1.2 V HSTL Class I	1.2	1.2	0.6	Yes	Yes	Yes	Yes	Yes
1.2 V HSTL Class II	1.2	1.2	0.6	Yes	Yes	Yes	Yes	Yes
HSUL-12	1.2	1.2	0.6	Yes	Yes	Yes	Yes	Yes
Differential SSTL-2 Class I and II	—	2.5	—	Yes	Yes	—	Yes	—
	2.5	—	1.25	—	—	Yes	Yes	—
Differential SSTL-18 Class I and Class II	—	1.8	—	Yes	Yes	—	Yes	—
	1.8	—	0.9	—	—	Yes	Yes	—
Differential SSTL-15 Class I and Class II	—	1.5	—	Yes	Yes	—	Yes	—
	1.5	—	0.75	—	—	Yes	Yes	—
Differential SSTL-15	—	1.5	—	Yes	Yes	—	Yes	—

continued...

⁽¹¹⁾ Not supported on bank 1B and bank 8.

⁽¹²⁾ Bidirectional—use Schmitt Trigger input with LVTTTL output.

I/O Standard	V _{CCIO} (V)		V _{REF} (V)	Pin Type Support				
	Input	Output		PLL_CLKOUT	MEM_CLK	CLK	DQS	User I/O
	1.5	—	0.75	—	—	Yes	Yes	—
Differential SSTL-135	—	1.35	—	Yes	Yes	—	Yes	—
	1.35	—	0.675	—	—	Yes	Yes	—
Differential 1.8 V HSTL Class I and Class II	—	1.8	—	Yes	Yes	—	Yes	—
	1.8	—	0.9	—	—	Yes	Yes	—
Differential 1.5 V HSTL Class I and Class II	—	1.5	—	Yes	Yes	—	Yes	—
	1.5	—	0.75	—	—	Yes	Yes	—
Differential 1.2 V HSTL Class I and Class II	—	1.2	—	Yes	Yes	—	Yes	—
	1.2	—	0.6	—	—	Yes	Yes	—
Differential HSUL-12	—	1.2	—	Yes	Yes	—	Yes	—
	1.2	—	0.6	—	—	Yes	Yes	—
LVDS (dedicated)	2.5	2.5	—	Yes	Yes	Yes	—	Yes
1.8 V LVDS (dedicated)	1.8	1.8	—	Yes	Yes	Yes	—	Yes
LVDS (emulated, external resistors)	—	2.5	—	Yes	Yes	—	—	Yes
Mini-LVDS (dedicated)	—	2.5	—	Yes	Yes	—	—	Yes
Mini-LVDS (emulated, external resistor)	—	2.5	—	Yes	Yes	—	—	Yes
RSDS (dedicated)	—	2.5	—	Yes	Yes	—	—	Yes
RSDS (emulated, external resistor, 1R)	—	2.5	—	Yes	Yes	—	—	Yes
RSDS (emulated, external resistors, 3R)	—	2.5	—	Yes	Yes	—	—	Yes
PPDS (dedicated)	—	2.5	—	Yes	Yes	—	—	Yes
PPDS (emulated, external resistor)	—	2.5	—	Yes	Yes	—	—	Yes
LVPECL	2.5	—	—	—	—	Yes	—	—
Bus LVDS	2.5	2.5	—	—	—	—	—	Yes
TMDS	2.5	—	—	—	—	Yes	—	Yes
Sub-LVDS	2.5	1.8	—	Yes	Yes	Yes	—	Yes
SLVS	2.5	2.5	—	Yes	Yes	Yes	—	Yes
HiSpi	2.5	—	—	—	—	Yes	—	Yes

Related Information

- [Intel MAX 10 Device Pin-Out Files](#)
- [Intel MAX 10 I/O Standards Support](#) on page 6
- [Intel MAX 10 I/O Banks Locations](#) on page 14
- [Intel MAX 10 LVDS SERDES I/O Standards Support](#)

- [Intel MAX 10 High-Speed LVDS I/O Location](#)

2.2. Intel MAX 10 I/O Elements

The Intel MAX 10 I/O elements (IOEs) contain a bidirectional I/O buffer and five registers for registering input, output, output-enable signals, and complete embedded bidirectional single data rate (SDR) and double data rate (DDR) transfer.

The I/O buffers are grouped into groups of four I/O modules per I/O bank:

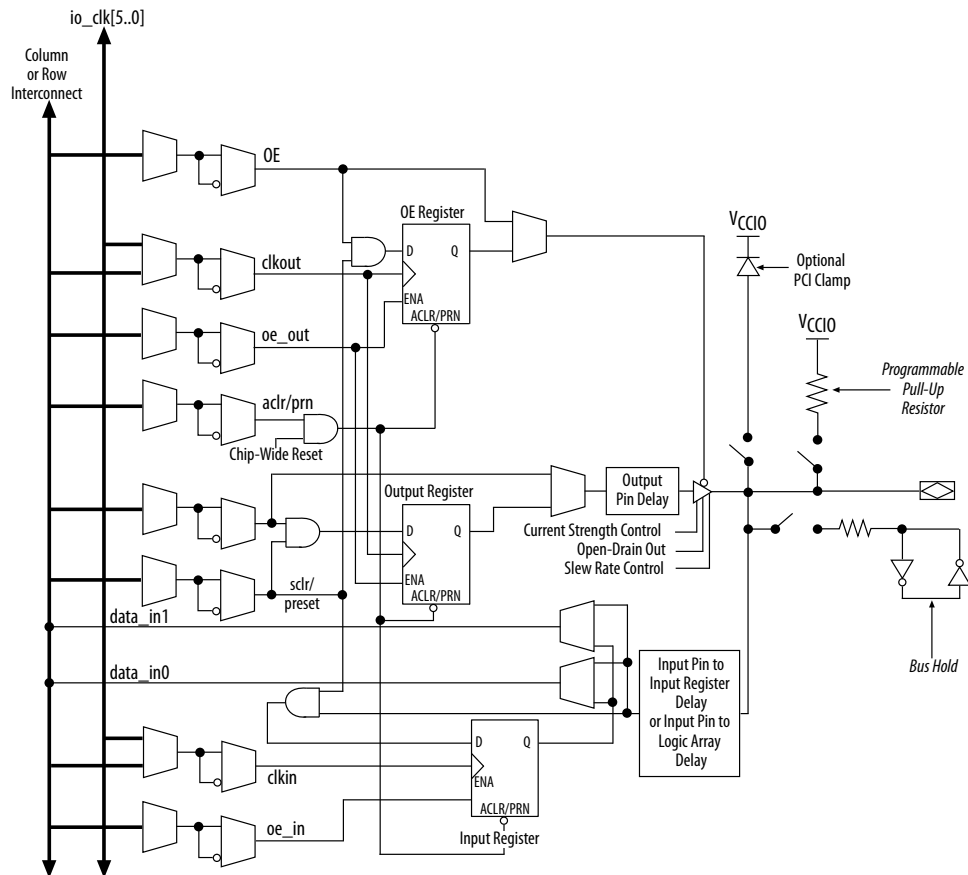
- The Intel MAX 10 devices share the user I/O pins with the VREF, RUP, RDN, CLKPIN, PLLCLKOUT, configuration, and test pins.
- Schmitt Trigger input buffer is available in all I/O buffers.
- When the Intel MAX 10 device is blank or erased, the I/Os are tri-stated.

Each IOE contains one input register, two output registers, and two output-enable (OE) registers:

- The two output registers and two OE registers are used for DDR applications.
- You can use the input registers for fast setup times and output registers for fast clock-to-output times.
- You can use the OE registers for fast clock-to-output enable times.

You can use the IOEs for input, output, or bidirectional data paths. The I/O pins support various single-ended and differential I/O standards.

Figure 2. IOE Structure in Bidirectional Configuration



Related Information

- [Intel MAX 10 Power Management User Guide](#)
Provides more information about the I/O buffers in different power cycles and hot socketing.
- [Schmitt-Trigger Input Buffer](#) on page 18

2.2.1. Intel MAX 10 I/O Banks Architecture

The I/O elements are located in a group of four modules per I/O bank:

- High speed DDR3 I/O banks—supports various I/O standards and protocols including DDR3 but not the 1.8 V LVDS I/O standard. These I/O banks are available only on the right side of the device.
- High speed I/O banks—supports various I/O standards and protocols except DDR3. These I/O banks are available on the top, left, and bottom sides of the device.
- Low speed I/O banks—lower speeds I/O banks that are located at the top left side of the device.

For more information about I/O pins support, refer to the pinout files for your device.

Related Information[Intel MAX 10 Device Pin-Out Files](#)**2.2.2. Intel MAX 10 I/O Banks Performance**

The performance of the I/O banks differs for different I/O standards and I/O bank types. You must ensure that the frequency you specified passes timing check in the Intel Quartus Prime software.

The low speed I/O banks have lower maximum frequency than other I/O banks because of longer propagation delays. However, the delays do not affect the timing parameters such as slew rate, rise time, and fall time.

For details about the location of the high speed and low speed I/O banks, refer to the device pinout files.

Related Information

- [High-Speed I/O Specifications](#)
Provides the performance information for different I/O standards in the low-speed and high-speed I/O banks.
- [IBIS Models for Intel Devices](#)
- [SPICE Models for Altera Devices](#)

2.2.3. Intel MAX 10 I/O Banks Locations

The I/O banks are located at the periphery of the device.

For more details about the modular I/O banks available in each device package, refer to the relevant device pin-out file.

Figure 3. I/O Banks for 10M02 Devices (Except Single Power Supply U324 Package)

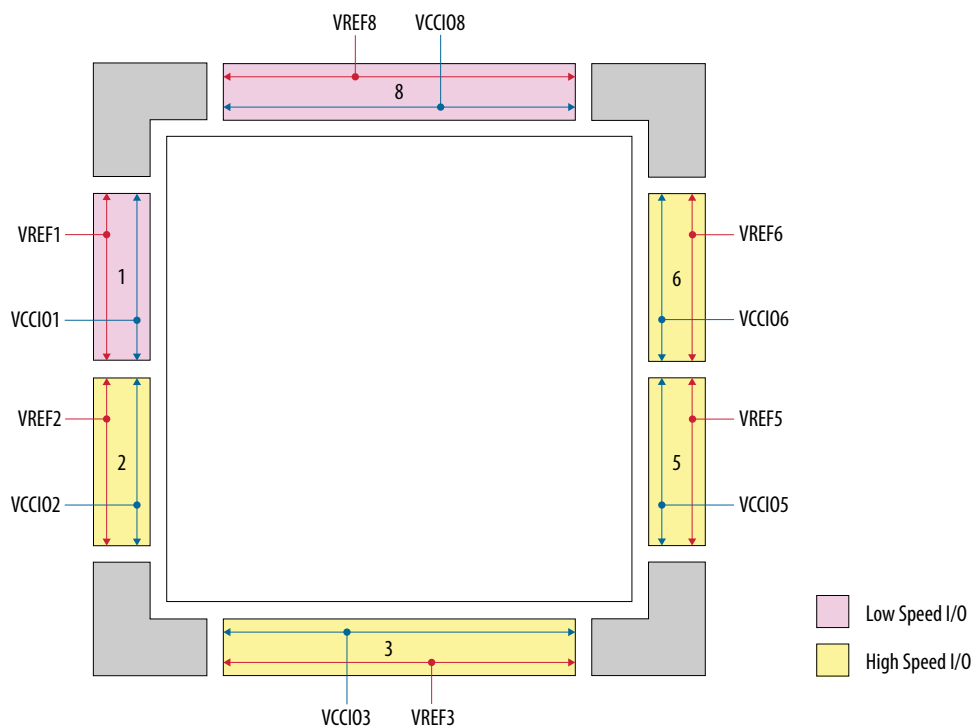


Figure 4. I/O Banks for 10M02 (Single Power Supply U324 Package), 10M04, and 10M08 (Except V81, M153, and U169 Packages) Devices

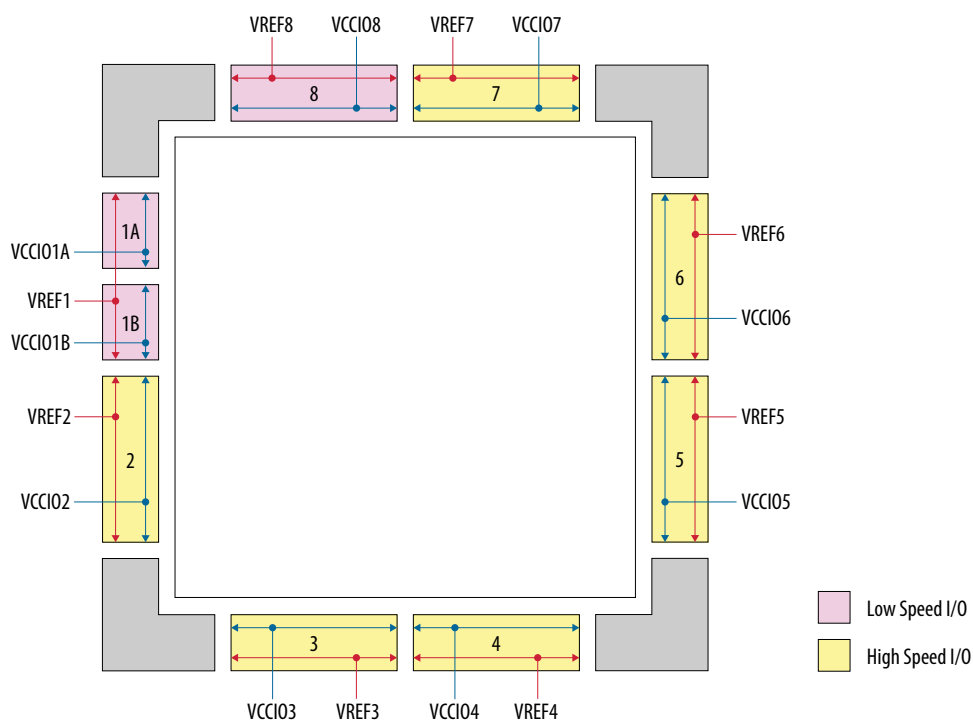


Figure 5. I/O Banks for 10M08 V81, M153, and U169 Packages Devices

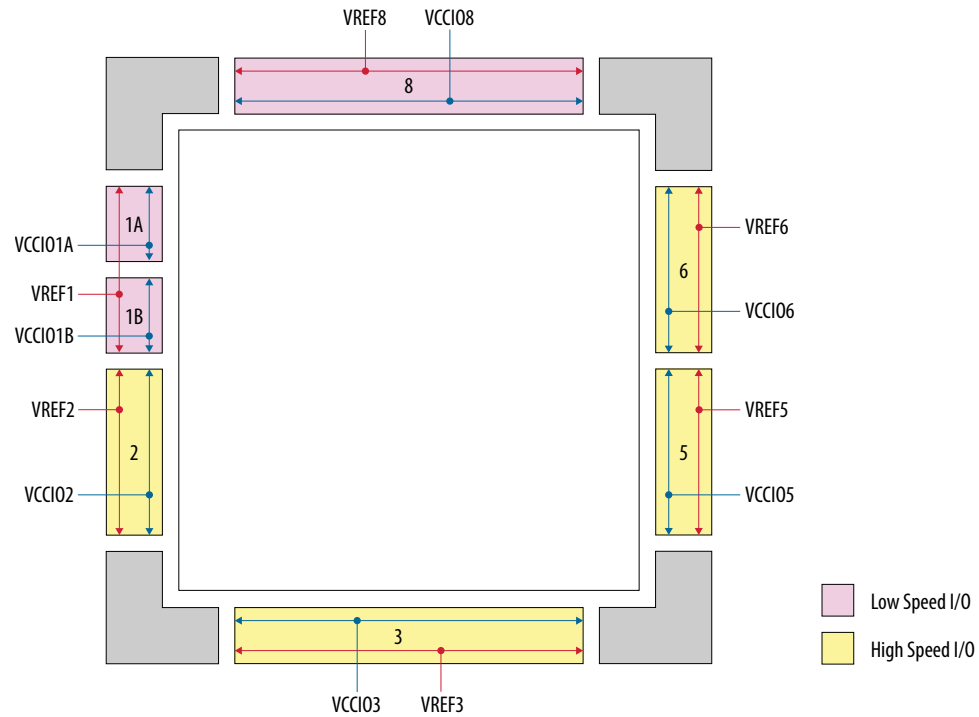
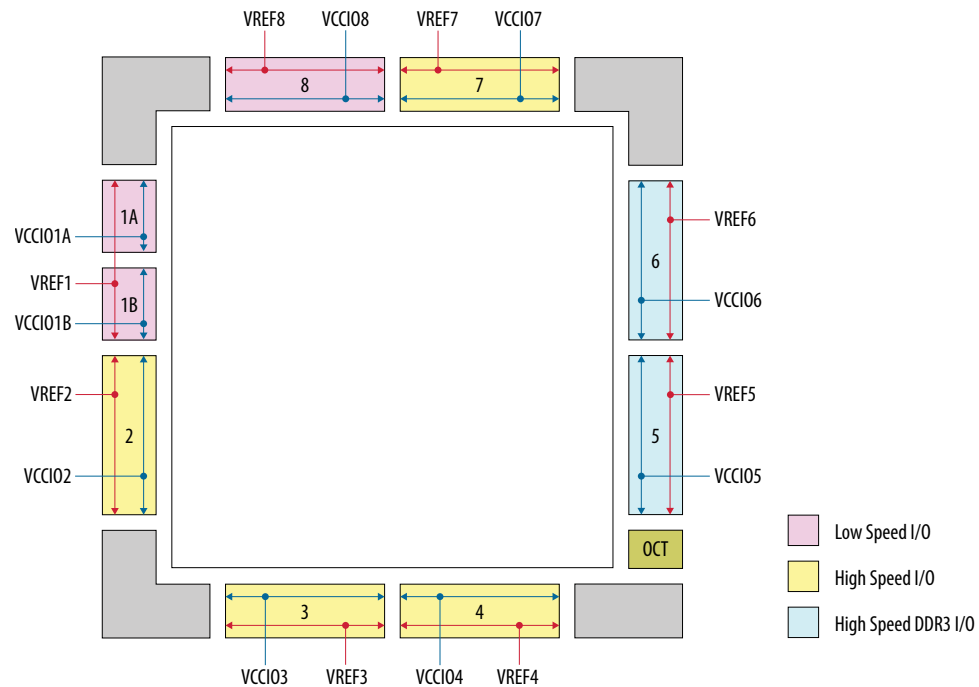


Figure 6. I/O Banks for 10M16, 10M25, 10M40, and 10M50 Devices





Related Information

- [Intel MAX 10 Device Pin-Out Files](#)
- [High-Speed I/O Specifications](#)
Provides the performance information for different I/O standards in the low-speed and high-speed I/O banks.

2.3. Intel MAX 10 I/O Buffers

The general purpose I/Os (GPIOs) in Intel MAX 10 devices consist of LVDS I/O and DDR I/O buffers.

Table 5. Types of GPIO Buffers in Intel MAX 10 Devices

LVDS I/O Buffers	DDR I/O Buffers
<ul style="list-style-type: none"> Support differential and single-ended I/O standards. Available only on I/O banks at the bottom side of the device. For LVDS, the bottom I/O banks support LVDS transmitter, emulated LVDS transmitter, and LVDS receiver buffers. 	<ul style="list-style-type: none"> Support differential and single-ended I/O standards. Available on I/O banks at the left, right, and top sides of the device. For LVDS, the DDR I/O buffers support only LVDS receiver and emulated LVDS transmitter buffers. For DDR, only the DDR I/O buffers on the right side of the device supports DDR3 external memory interfaces. DDR3 support is only available for Intel MAX 10 16, 25, 40, and 50 devices.

Related Information

- [Intel MAX 10 I/O Standards Support](#) on page 6
- [LVDS Transmitter I/O Termination Schemes, Intel MAX 10 High-Speed LVDS I/O User Guide](#)
Provides the required external termination schemes and resistor values for the emulated LVDS, Sub-LVDS, SLVS, emulated RSDS, emulated mini-LVDS, and emulated PPDS I/O standards.

2.3.1. Schmitt-Trigger Input Buffer

The Intel MAX 10 devices feature selectable Schmitt trigger input buffer on all I/O banks.

The Schmitt trigger input buffer has similar V_{IL} and V_{IH} as the LVTTTL I/O standard but with better noise immunity. The Schmitt trigger input buffers are used as default input buffers during configuration mode.

Related Information

[Intel MAX 10 Device Datasheet](#)

2.3.2. Programmable I/O Buffer Features

The Intel MAX 10 I/O buffers support a range of programmable features. These features increase the flexibility of I/O utilization and provide an alternative to reduce the usage of external discrete components such as a pull-up resistor and a diode.

Table 6. Summary of Supported Intel MAX 10 Programmable I/O Buffer Features and Settings

Feature	Setting	Condition	Assignment Name	Supported I/O Standards
Open Drain	On, Off (default)	To enable this feature, use the OPNDRN primitive.	—	<ul style="list-style-type: none"> 3.0 V and 3.3 V LVTTTL 1.0 V, 1.2 V, 1.5 V, 1.8 V, 2.5 V, 3.0 V, and 3.3 V LVCMOS SSTL-2, SSTL-18, SSTL-15, and SSTL-135
continued...				

Feature	Setting	Condition	Assignment Name	Supported I/O Standards
				<ul style="list-style-type: none"> 1.2 V, 1.5 V, and 1.8 V HSTL HSUL-12 3.0 V PCI
Bus-Hold	On, Off (default)	Disabled if you use the weak pull-up resistor feature.	Enable Bus-Hold Circuitry	<ul style="list-style-type: none"> 3.0 V and 3.3 V LVTTTL 1.2 V, 1.5 V, 1.8 V, 2.5 V, 3.0 V, and 3.3 V LVCMOS SSTL-2, SSTL-18, SSTL-15, and SSTL-135 1.2 V, 1.5 V, and 1.8 V HSTL HSUL-12 3.0 V PCI
Pull-up Resistor	On, Off (default)	Disabled if you use the bus-hold feature.	Weak Pull-Up Resistor	
Slew Rate Control	0 (Slow), 1 (Medium), 2 (Fast). Default is 2.	Disabled if you use OCT.	Slew Rate	<ul style="list-style-type: none"> 3.0 V LVTTTL 1.2 V, 1.5 V, 1.8 V, 2.5 V, and 3.0 V LVCMOS SSTL-2, SSTL-18, and SSTL-15 1.2 V, 1.5 V, and 1.8 V HSTL Differential SSTL-2, Differential SSTL-18, and Differential SSTL-15 Differential 1.2 V, 1.5 V, and 1.8 V HSTL
PCI Clamp Diode	On (default for input pins), Off (default for output pins, except 3.0 V PCI)	—	PCI I/O	<ul style="list-style-type: none"> 3.0 V and 3.3 V LVTTTL 2.5 V, 3.0 V, and 3.3 V LVCMOS 3.0 V PCI 2.5 V, 3.0 V, and 3.3 V Schmitt Trigger
Pre-Emphasis	0 (disabled), 1 (enabled). Default is 1.	—	Programmable Pre-emphasis	<ul style="list-style-type: none"> LVDS RSDS PPDS Mini-LVDS
Differential Output Voltage	0 (low), 1 (medium), 2 (high). Default is 2.	—	Programmable Differential Output Voltage (V_{OD})	

2.3.2.1. Programmable Open Drain

The optional open-drain output for each I/O pin is equivalent to an open collector output. If it is configured as an open drain, the logic value of the output is either high-Z or logic low.

Use an external resistor to pull the signal to a logic high.

2.3.2.2. Programmable Bus Hold

Each I/O pin provides an optional bus-hold feature that is active only after configuration. When the device enters user mode, the bus-hold circuit captures the value that is present on the pin by the end of the configuration.

The bus-hold circuitry holds the signal on an I/O pin at its last-driven state until the next input signal is present. Because of this, you do not require an external pull-up or pull-down resistor to hold a signal level when the bus is tri-stated.

User I/O pins can be in either the default weak pull-up state or tri-state during configuration. With the bus-hold feature, if you do not drive the I/O pin externally when it enters user mode from configuration mode:

- The I/O pin state is weak pull-up during configuration—the I/O pin retains the high value when the device enters user mode.
- The I/O pin is tri-stated during configuration—the I/O pin value can be high or low when the device enters user mode.

For each I/O pin, you can individually specify that the bus-hold circuitry pulls non-driven pins away from the input threshold voltage—where noise can cause unintended high-frequency switching. To prevent over-driving signals, the bus-hold circuitry drives the voltage level of the I/O pin lower than the V_{CCIO} level.

If you enable the bus-hold feature, you cannot use the programmable pull-up option. To configure the I/O pin for differential signals, disable the bus-hold feature.

2.3.2.3. Programmable Pull-Up Resistor

Each I/O pin provides an optional programmable pull-up resistor during user mode. The pull-up resistor weakly holds the I/O to the V_{CCIO} level.

If you enable the weak pull-up resistor, you cannot use the bus-hold feature.

2.3.2.4. Programmable Current Strength

You can use the programmable current strength to mitigate the effects of high signal attenuation that is caused by a long transmission line or a legacy backplane.

Table 7. Programmable Current Strength Settings for Intel MAX 10 Devices

The output buffer for each Intel MAX 10 device I/O pin has a programmable current strength control for the I/O standards listed in this table.

I/O Standard	I_{OH} / I_{OL} Current Strength Setting (mA) (Default setting in bold)
3.3 V LVCMOS	2
3.3 V LVTTTL	8 , 4
3.0 V LVTTTL/3.0 V LVCMOS	16, 12 , 8, 4
2.5 V LVTTTL/2.5 V LVCMOS	16, 12 , 8, 4
1.8 V LVTTTL/1.8 V LVCMOS	16, 12 , 10, 8, 6, 4, 2
1.5 V LVCMOS	16, 12 , 10, 8, 6, 4, 2
1.2 V LVCMOS	12, 10, 8 , 6, 4, 2
SSTL-2 Class I	12, 8
SSTL-2 Class II	16
SSTL-18 Class I	12, 10, 8
SSTL-18 Class II	16 , 12
SSTL-15 Class I	12, 10, 8
SSTL-15 Class II	16
1.8 V HSTL Class I	12, 10, 8
1.8 V HSTL Class II	16
1.5 V HSTL Class I	12, 10, 8

continued...

I/O Standard	I _{OH} / I _{OL} Current Strength Setting (mA) (Default setting in bold)
1.5 V HSTL Class II	16
1.2 V HSTL Class I	12, 10, 8
1.2 V HSTL Class II	14
BLVDS	16 , 12, 8
SLVS	16 , 12, 8
Sub-LVDS	12 , 8, 4

Note: Intel recommends that you perform IBIS or SPICE simulations to determine the best current strength setting for your specific application.

Related Information

- [IBIS Models for Intel Devices](#)
- [SPICE Models for Altera Devices](#)

2.3.2.5. Programmable Output Slew Rate Control

You have the option of three settings for programmable slew rate control—0, 1, and 2 with 2 as the default setting. Setting 0 is the slow slew rate and 2 is the fast slew rate.

- Fast slew rate—provides high-speed transitions for high-performance systems.
- Slow slew rate—reduces system noise and crosstalk but adds a nominal delay to the rising and falling edges.

Table 8. Programmable Output Slew Rate Control for Intel MAX 10 Devices

This table lists the single-ended I/O standards and current strength settings that support programmable output slew rate control. For I/O standards and current strength settings that do not support programmable slew rate control, the default slew rate setting is 2 (fast slew rate).

I/O Standard	I _{OH} / I _{OL} Current Strength Supporting Slew Rate Control
3.0 V LVTTL/3.0 V LVCMOS	16, 12, 8
2.5 V LVTTL/2.5 V LVCMOS	16, 12, 8
1.8 V LVTTL/1.8 V LVCMOS	16, 12, 8
1.5 V LVCMOS	16, 12, 10, 8
1.2 V LVCMOS	12, 10, 8
SSTL-2 Class I	12, 8
SSTL-2 Class II	16
SSTL-18 Class I	12, 10, 8
SSTL-18 Class II	16, 12
SSTL-15 Class I	12, 10, 8
SSTL-15 Class II	16
1.8 V HSTL Class I	12, 10, 8
1.8 V HSTL Class II	16
<i>continued...</i>	

I/O Standard	I _{OH} / I _{OL} Current Strength Supporting Slew Rate Control
1.5 V HSTL Class I	12, 10, 8
1.5 V HSTL Class II	16
1.2 V HSTL Class I	12, 10, 8
1.2 V HSTL Class II	14

You can specify the slew rate on a pin-by-pin basis because each I/O pin contains a slew rate control. The slew rate control affects both the rising and falling edges.

Note: Intel recommends that you perform IBIS or SPICE simulations to determine the best slew rate setting for your specific application.

2.3.2.6. Programmable IOE Delay

You can activate the programmable IOE delays to ensure zero hold times, minimize setup times, increase clock-to-output times, or delay the clock input signal. This feature helps read and write timing margins because it minimizes the uncertainties between signals in the bus.

Each pin can have a different delay value to ensure signals within a bus have the same delay going into or out of the device.

Table 9. Programmable Delay Chain

Programmable Delays	Intel Quartus Prime Logic Option
Input pin-to-logic array delay	Input delay from pin to internal cells
Input pin-to-input register delay	Input delay from pin to input register
Output pin delay	Delay from output register to output pin
Dual-purpose clock input pin delay	Input delay from dual-purpose clock pin to fan-out destinations

There are two paths in the IOE for an input to reach the logic array. Each of the two paths can have a different delay. This allows you to adjust delays from the pin to the internal logic element (LE) registers that reside in two different areas of the device. You must set the two combinational input delays with the input delay from pin to internal cells logic option in the Intel Quartus Prime software for each path. If the pin uses the input register, one of the delays is disregarded and the delay is set with the input delay from pin to input register logic option in the Intel Quartus Prime software.

The IOE registers in each I/O block share the same source for the preset or clear features. You can program preset or clear for each individual IOE, but you cannot use both features simultaneously. You can also program the registers to power-up high or low after configuration is complete. If programmed to power-up low, an asynchronous clear can control the registers. If programmed to power-up high, an asynchronous preset can control the registers. This feature prevents the inadvertent activation of the active-low input of another device upon power up. If one register in an IOE uses a preset or clear signal, all registers in the IOE must use that same signal if they require preset or clear. Additionally, a synchronous reset signal is available for the IOE registers.

Related Information

- [Intel MAX 10 Device Datasheet](#)

- [Timing Closure and Optimization, Intel Quartus Prime Standard Edition User Guide: Design Optimization](#)
Provides more information about the input and output pin delay settings.

2.3.2.7. PCI Clamp Diode

The Intel MAX 10 devices are equipped with optional PCI clamp diode that you can enable for the input and output of each I/O pin. You can use this diode to protect I/O pins during voltage overshoot.

The PCI clamp diode is available in the Intel Quartus Prime software for the following I/O standards:

- 3.3 V LVTTL/3.3 V LVCMOS
- 3.0 V LVTTL/3.0 V LVCMOS
- 2.5 V LVTTL/2.5 V LVCMOS
- 3.0 V PCI
- 3.3 V Schmitt Trigger
- 2.5 V Schmitt Trigger

Dual-purpose configuration pins support the diode in user mode if you do not use the pins as configuration pins for the selected configuration scheme. The dedicated configuration pins do not support the on-chip diode.

Related Information

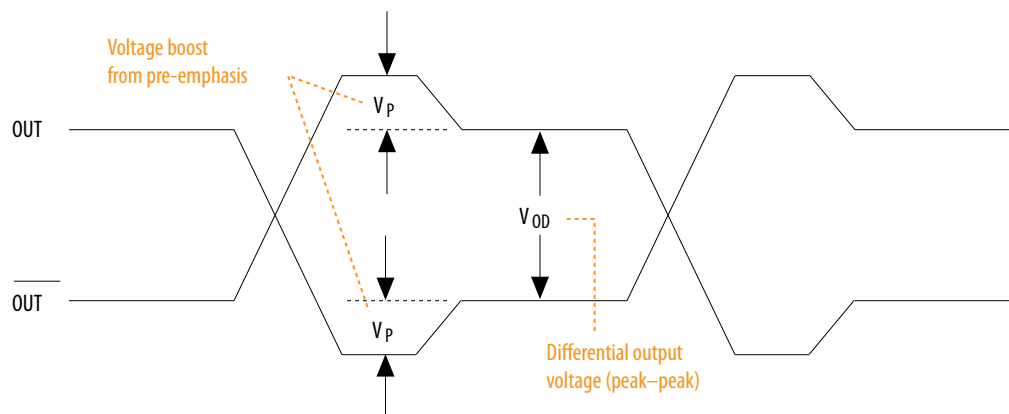
- [Guidelines: Enable Clamp Diode for LVTTL/LVCMOS Input Buffers](#) on page 32
- [Guideline: Use Internal PCI Clamp Diode on the Pin, AN 447: Interfacing Intel FPGA Devices with 3.3/3.0/2.5 V LVTTL/LVCMOS I/O Systems](#)

2.3.2.8. Programmable Pre-Emphasis

The differential output voltage (V_{OD}) setting and the output impedance of the driver set the output current limit of a high-speed transmission signal. At a high frequency, the slew rate may not be fast enough to reach the full V_{OD} level before the next edge, producing pattern-dependent jitter. Pre-emphasis momentarily boosts the output current during switching to increase the output slew rate.

Pre-emphasis increases the amplitude of the high-frequency component of the output signal. This increase compensates for the frequency-dependent attenuation along the transmission line.

The overshoot introduced by the extra current occurs only during change of state switching. This overshoot increases the output slew rate but does not ring, unlike the overshoot caused by signal reflection. The amount of pre-emphasis required depends on the attenuation of the high-frequency component along the transmission line.

Figure 7. LVDS Output with Programmable Pre-Emphasis

Table 10. Intel Quartus Prime Software Assignment for Programmable Pre-Emphasis

Field	Assignment
To	tx_out
Assignment name	Programmable Pre-emphasis
Allowed values	0 (disabled), 1 (enabled). Default is 1.

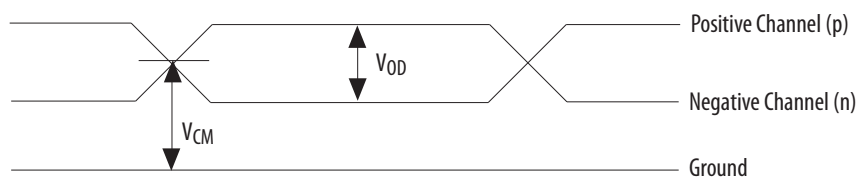
2.3.2.9. Programmable Differential Output Voltage

The programmable V_{OD} settings allow you to adjust the output eye opening to optimize the trace length and power consumption. A higher V_{OD} swing improves voltage margins at the receiver end, and a smaller V_{OD} swing reduces power consumption.

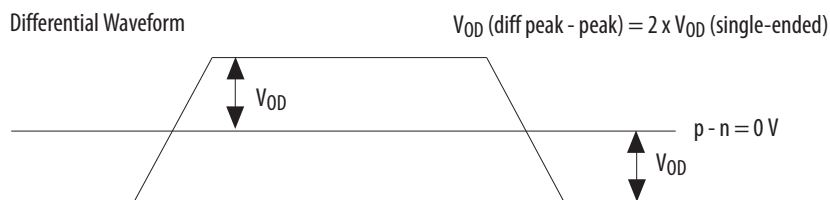
Figure 8. Differential V_{OD}

This figure shows the V_{OD} of the differential LVDS output.

Single-Ended Waveform



Differential Waveform



You can statically adjust the V_{OD} of the differential signal by changing the V_{OD} settings in the Intel Quartus Prime software Assignment Editor.

Table 11. Intel Quartus Prime Software Assignment Editor—Programmable V_{OD}

Field	Assignment
To	tx_out
Assignment name	Programmable Differential Output Voltage (V _{OD})
Allowed values	0 (low), 1 (medium), 2 (high). Default is 2.

2.3.2.10. Programmable Emulated Differential Output

The Intel MAX 10 devices support emulated differential output where a pair of single-ended output drives out a differential signal.

The emulated differential output feature is supported for the following I/O standards:

- Differential SSTL-2 Class I and II
- Differential SSTL-18 Class I and II
- Differential SSTL-15 Class I and II
- Differential SSTL-15
- Differential SSTL-135
- Differential 1.8 V HSTL Class I and II
- Differential 1.5 V HSTL Class I and II
- Differential 1.2 V HSTL Class I and II
- Differential HSUL-12
- LVDS 3R
- Mini-LVDS 3R
- PPDS 3R
- RSDS 1R and 3R
- BLVDS
- SLVS
- Sub-LVDS

2.3.2.11. Programmable Dynamic Power Down

The Intel MAX 10 16, 25, 40, and 50 devices feature programmable dynamic power down for several I/O standards to reduce the static power consumption.

In these devices, you can apply the programmable dynamic power down feature to the I/O buffers for the following I/O standards:

- Input buffer—SSTL, HSTL, HSUL, LVDS
- Output buffer—LVDS

Related Information

[Intel MAX 10 Power Management User Guide](#)

Provides more information about using the programmable dynamic power down feature.

2.4. I/O Standards Termination

Voltage-referenced and differential I/O standards requires different termination schemes.

According to JEDEC standards, the following I/O standards do not specify a recommended termination scheme:

- 3.3-V LVTTTL
- 3.0 V LVTTTL/3.0 V LVCMOS
- 2.5 V LVTTTL/2.5 V LVCMOS
- 1.8 V LVTTTL/1.8 V LVCMOS
- 1.5 V LVCMOS
- 1.2 V LVCMOS
- 1.0 V LVCMOS
- 3.0-V PCI

2.4.1. Voltage-Referenced I/O Standards Termination

Voltage-referenced I/O standards require an input reference voltage (V_{REF}) and a termination voltage (V_{TT}). The reference voltage of the receiving device tracks the termination voltage of the transmitting device.

Figure 9. HSTL I/O Standard Termination

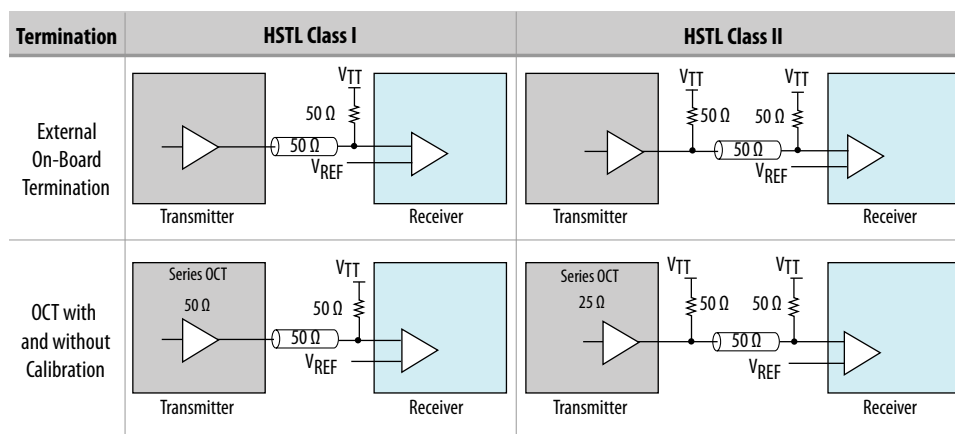
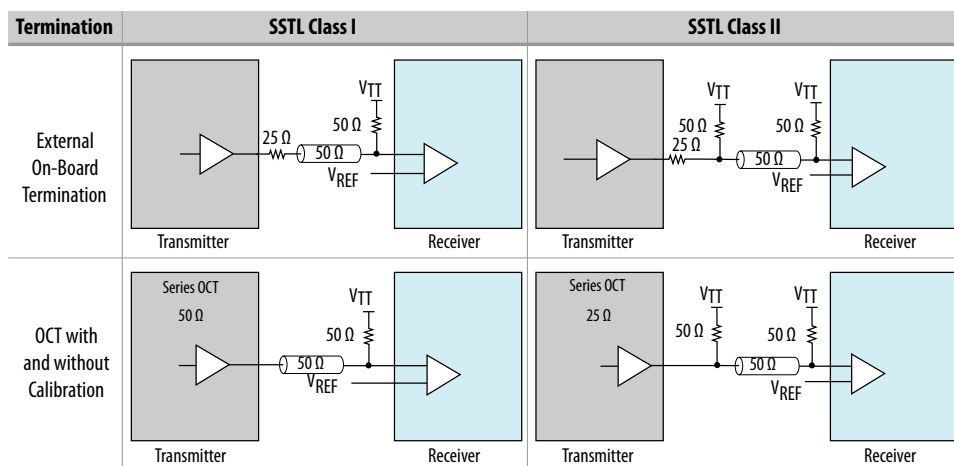


Figure 10. SSTL I/O Standard Termination



2.4.2. Differential I/O Standards Termination

Differential I/O standards typically require a termination resistor between the two signals at the receiver. The termination resistor must match the differential load impedance of the bus.

Figure 11. Differential HSTL I/O Standard Termination

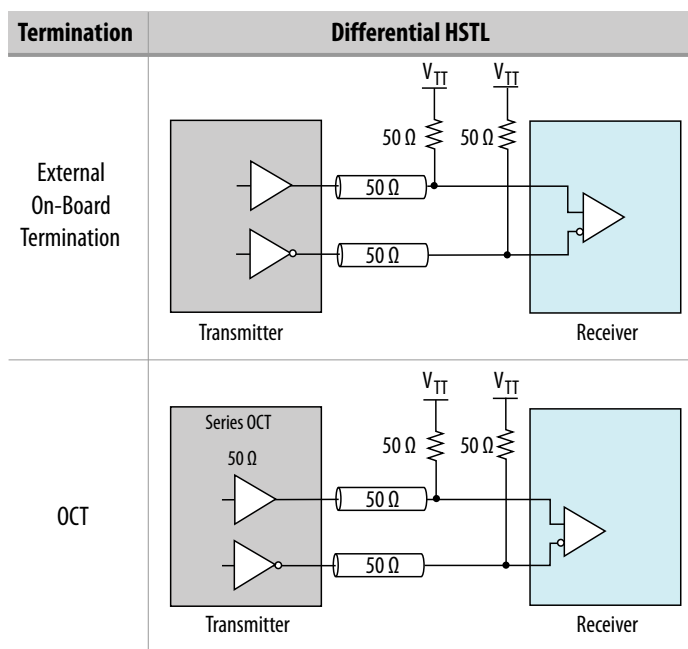
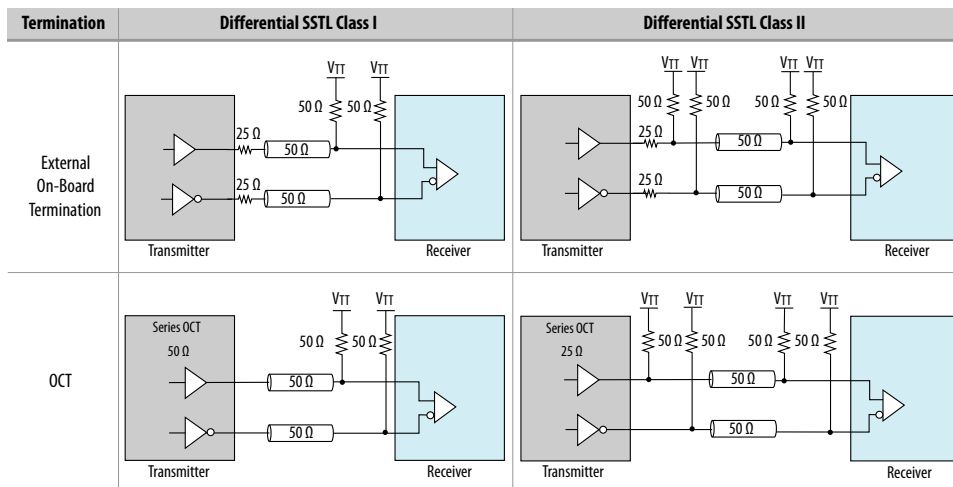


Figure 12. Differential SSTL I/O Standard Termination



Related Information

[Intel MAX 10 High-Speed LVDS I/O User Guide](#)

Provides more information about differential I/O external termination.

2.4.3. Intel MAX 10 On-Chip I/O Termination

The on-chip termination (OCT) block in Intel MAX 10 devices provides I/O impedance matching and termination capabilities. OCT maintains signal quality, saves board space, and reduces external component costs.

The Intel MAX 10 devices support serial (R_S) OCT for single-ended output pins and bidirectional pins. For bidirectional pins, OCT is active for output only.

Figure 13. Single-ended I/O Termination (R_S)

This figure shows the single-ended termination scheme supported in Intel MAX 10 device.

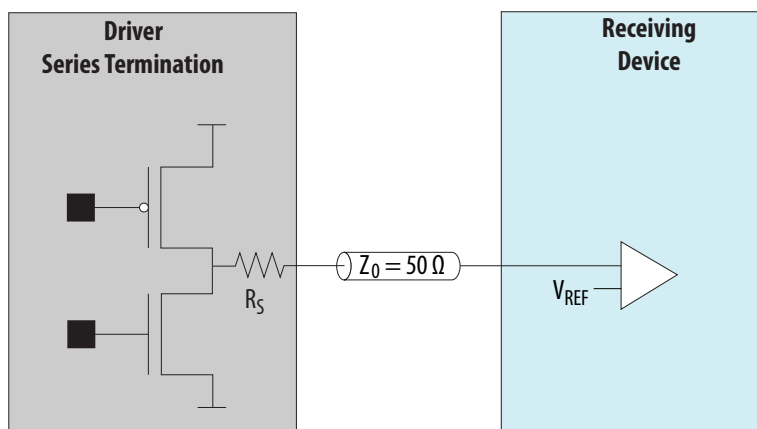


Table 12. OCT Schemes Supported in Intel MAX 10 Devices

Direction	OCT Schemes	Device Support	I/O Bank Support
Output	R_S OCT with calibration	Intel MAX 10 16, 25, 40, and 50 devices	Right bank only
	R_S OCT without calibration	All Intel MAX 10 devices	All I/O banks

2.4.3.1. OCT Calibration

The OCT calibration circuit compares the total impedance of the output buffer to the external resistors connected to the RUP and RDN pins. The circuit dynamically adjusts the output buffer impedance until it matches the external resistors.

Each calibration block comes with a pair of RUP and RDN pins.

During calibration, the RUP and RDN pins are each connected through an external 25 Ω , 34 Ω , 40 Ω , 48 Ω , or 50 Ω resistor for respective on-chip series termination value of 25 Ω , 34 Ω , 40 Ω , 48 Ω , and 50 Ω :

- RUP—connected to VCCIO.
- RDN—connected to GND.

The OCT calibration circuit compares the external resistors to the internal resistance using comparators. The OCT calibration block uses the comparators' output to dynamically adjust buffer impedance.

During calibration, the resistance of the RUP and RDN pins varies. To estimate of the maximum possible current through the external calibration resistors, assume a minimum resistance of 0 Ω on the RUP and RDN pins.

2.4.3.2. R_S OCT in Intel MAX 10 Devices**Table 13. Selectable I/O Standards for R_S OCT**

This table lists the output termination settings for R_S OCT with and without calibration on different I/O standards.

- R_S OCT with calibration—supported only on the right side I/O banks of the Intel MAX 10 16, 25, 40, and 50 devices.
- R_S OCT without calibration—supported on all I/O banks of all Intel MAX 10 devices.

I/O Standard	Calibrated OCT (Output)	Uncalibrated OCT (Output)
	R_S (Ω)	R_S (Ω)
3.0 V LVTTTL/3.0V LVCMOS	25, 50	25, 50
2.5 V LVTTTL/2.5 V LVCMOS	25, 50	25, 50
1.8 V LVTTTL/1.8 V LVCMOS	25, 50	25, 50
1.5 V LVCMOS	25, 50	25, 50
1.2 V LVCMOS	25, 50	25, 50
SSTL-2 Class I	50	50
SSTL-2 Class II	25	25
continued...		

I/O Standard	Calibrated OCT (Output)	Uncalibrated OCT (Output)
	R_S (Ω)	R_S (Ω)
SSTL-18 Class I	50	50
SSTL-18 Class II	25	25
SSTL-15 Class I	50	50
SSTL-15 Class II	25	25
SSTL-15	34, 40	34, 40
SSTL-135	34, 40	34, 40
1.8 V HSTL Class I	50	50
1.8 V HSTL Class II	25	25
1.5 V HSTL Class I	50	50
1.5 V HSTL Class II	25	25
1.2 V HSTL Class I	50	50
1.2 V HSTL Class II	25	25
HSUL-12	34, 40, 48	34, 40, 48
Differential SSTL-2 Class I	50	50
Differential SSTL-2 Class II	25	25
Differential SSTL-18 Class I	50	50
Differential SSTL-18 Class II	25	25
Differential SSTL-15 Class I	50	50
Differential SSTL-15 Class II	25	25
Differential SSTL-15	34, 40	34, 40
Differential SSTL-135	34, 40	34, 40
Differential 1.8 V HSTL Class I	50	50
Differential 1.8 V HSTL Class II	25	25
Differential 1.5 V HSTL Class I	50	50
Differential 1.5 V HSTL Class II	25	25
Differential 1.2 V HSTL Class I	50	50
Differential 1.2 V HSTL Class II	25	25
Differential HSUL-12	34, 40, 48	34, 40, 48

3. Intel MAX 10 I/O Design Considerations

There are several considerations that require your attention to ensure the success of your designs. Unless noted otherwise, these design guidelines apply to all variants of this device family.

Related Information

[Intel MAX 10 I/O Overview](#) on page 3

3.1. Guidelines: V_{CCIO} Range Considerations

There are several V_{CCIO} range considerations because of I/O pin configuration function and I/O bank location.

- Banks 1 and 8 have I/O pins with configuration function. The configuration function of these pins support only 1.5 V to 3.3 V. If you want to access the configuration function of these pins during user mode (run time), for example JTAG pins, the V_{CCIO} of the pin's bank is limited to a range of 1.5 V to 3.3 V. If you want to use I/O standards with 1.2 V to 1.35 V in bank 1 or 8 during user mode, do not use the configuration function of the bank's I/O pins.
- For devices with banks 1A and 1B:
 - If you use the V_{REF} pin or the ADC, you must supply a common V_{CCIO} voltage to banks 1A and 1B.
 - If you do not use the V_{REF} pin or the ADC, you can supply separate V_{CCIO} voltages to banks 1A and 1B.
- If you plan to migrate from devices that has banks 1A and 1B to devices that has only bank 1, ensure that the V_{CCIO} of bank 1A and 1B are the same.
- For the V36 package of the 10M02 device, the V_{CCIO} of these groups of I/O banks must be the same:
 - Group 1—banks 1, 2 and 8
 - Group 2—banks 3, 5, and 6
- For the V81 package of the 10M08 device, the V_{CCIO} of these groups of I/O banks must be the same:
 - Group 1—banks 1A, 1B, and 2
 - Group 2—banks 5 and 6

3.2. Guidelines: Voltage-Referenced I/O Standards Restriction

These restrictions apply if you use the V_{REF} pin.

- If you use a shared V_{REF} pin as an I/O, all voltage-reference input buffers (SSTL, HSTL, and HSUL) are disabled.
- If you use a shared V_{REF} pin as a voltage reference, you must enable the input buffer of specific I/O pin to use the voltage-reference I/O standards.
- The voltage-referenced I/O standards are not supported in the following I/O banks of these device packages:
 - All I/O banks of V36 package of 10M02.
 - All I/O banks of V81 package of 10M08.
 - Banks 1A and 1B of E144 package of 10M50.
- For devices with banks 1A and 1B, if you use the V_{REF} pin, you must supply a common V_{CCIO} to banks 1A and 1B.
- The maximum number of voltage-referenced inputs for each V_{REF} pin is 75% of the total number of I/O pads. The Intel Quartus Prime software warns you if you exceed the maximum number.
- Except for I/O pins that you used for static signals, all non-voltage-referenced output must be placed two pads away from a V_{REF} pin. The Intel Quartus Prime software will output an error message if this rule is violated.

Related Information

[Intel MAX 10 I/O Standards Support](#) on page 6

3.3. Guidelines: Enable Clamp Diode for LVTTL/LVCMOS Input Buffers

If the input voltage to the LVTTL/LVCMOS input buffers is higher than the V_{CCIO} of the I/O bank, Intel recommends that you enable the clamp diode.

- 3.3 V LVCMOS/LVTTL input buffers—enable clamp diode if V_{CCIO} of the I/O bank is 3.0 V.
- 3.3 V or 3.0 V LVCMOS/LVTTL input buffers—enable clamp diode if V_{CCIO} of the I/O bank is 2.5 V.

By enabling the clamp diode under these conditions, you limit overshoot. However, this does not comply with hot socket current specification.

If you do not enable the clamp diode under these conditions, the signal integrity for the I/O pin is impacted and overshoot occurs. In this situation, you must ensure that your board design conforms to the overshoot specifications.

Table 14. Voltage Tolerance Maximum Ratings for 3.3 V or 3.0 V

This table lists the voltage tolerance specifications. Ensure that your board design conforms to these specifications if you do not want to follow the clamp diode recommendation.

Voltage	Minimum (V)	Maximum (V)
$V_{CCIO} = 3.3 \text{ V}$	3.135	3.45
$V_{CCIO} = 3.0 \text{ V}$	2.85	3.15
$V_{IH} \text{ (AC)}$	—	4.1
$V_{IH} \text{ (DC)}$	—	3.6
$V_{IL} \text{ (DC)}$	-0.3	0.8

Related Information

- [PCI Clamp Diode](#) on page 23
- [Guideline: Use Internal PCI Clamp Diode on the Pin, AN 447: Interfacing Intel FPGA Devices with 3.3/3.0/2.5 V LVTTTL/LVCMOS I/O Systems](#)

3.4. Guidelines: Adhere to the LVDS I/O Restrictions Rules

For LVDS applications, adhere to the I/O restriction pin connection guidelines to avoid excessive jitter on the LVDS transmitter output pins. The Intel Quartus Prime software generates a critical warning if these rules are violated.

Related Information

[Intel MAX 10 FPGA Device Family Pin Connection Guidelines](#)

3.5. Guidelines: I/O Restriction Rules

For different I/O standards and conditions, you must limit the number of I/O pins. This I/O restriction rule is applicable if you use LVDS transmitters or receivers. Apply this restriction if one or more LVDS I/O standards reside in the I/O bank.

Table 15. Maximum Percentage of I/O Pins Allowed for Specific I/O Standards in an I/O Bank

This table lists the maximum number of general purpose output pins recommended in a bank in terms of percentage to the total number of I/O pins available in an I/O bank if you use these combinations of I/O standards and conditions.

I/O Standard	Condition	Max Output Pins Per Bank (%)
2.5 V LVTTTL/LVCMOS	16 mA current strength or 25 Ω OCT	25
	12 mA current strength	30
	8 mA current strength or 50 Ω OCT	45
	4 mA current strength	65
2.5 V SSTL	—	100

Related Information

[Guidelines: Placement Restrictions for 1.0 V I/O Pin](#) on page 34

3.6. Guidelines: Placement Restrictions for 1.0 V I/O Pin

To minimize the impact of simultaneous switching noise (SSN) on the I/O pins, ensure that the total mutual inductance (L_m) of the I/O pins in usage surrounding the 1.0 V I/O does not exceed the guidelines in the following table.

Table 16. Total L_m Guidelines for Pins Surrounding the 1.0 V Pin

I/O Standard of Surrounding Pins	Locations Relative to 1.0 V Pin	Total L_m of Surrounding Pins
1.0 V	Within the same bank	The total L_m of the surrounding pins in the bank must not exceed 7.41 nH.
	In an adjacent bank	The total L_m of the surrounding pins in the adjacent bank must not exceed 7.41 nH.
	Within the same bank and in an adjacent bank	The sum of the total L_m of the surrounding pins in both banks must not exceed 7.41 nH.
Other than 1.0 V	In an adjacent bank	The total L_m of the surrounding pins in the adjacent bank must not exceed 1 nH.

Example scenarios where the 1.0 V pin is in bank 3 and surrounding pins are in banks 3 and 4:

- Bank 3 and 4 are both 1.0 V—total L_m of all surrounding pins in both banks must not exceed 7.41 nH.
- Bank 3 is 1.0 V but bank 4 is 2.5 V—total L_m of surrounding pins in bank 3 must not exceed 7.41 nH and total L_m in bank 4 must not exceed 1 nH.

Related Information

- [Intel MAX 10 Mutual Coupling \(max10-1v-mutual-coupling.zip\)](#)
Provides spreadsheet files that list the mutual inductance values for 1.0 V I/O for the following Intel MAX 10 device combinations: 10M02/04/08/16/25/40/50 + SC/DC/SA/DA/DD + U324/U169/M153/F256/F484/F672 + I7/C7/C8
- [Calculating the Total Inductance for 1.0 V Pin Placement](#) on page 34

3.6.1. Calculating the Total Inductance for 1.0 V Pin Placement

You can calculate the total inductance of the surrounding pins by using mutual inductance values in the `max10-1v-mutual-coupling.zip` file.

- Download the `max10-1v-mutual-coupling.zip` file and extract the relevant mutual inductance spreadsheet for your device.
- In the mutual inductance spreadsheet, identify the pins in use.
- Calculate the total mutual inductance of the pin and surrounding pins in use to ensure that the placement adheres to the 1.0 V pin placement guideline.
- If the total inductance is above the guideline restriction, update your design to use other I/O pins that contribute less mutual inductance.

Example 1. Total Mutual Inductance Calculation

Table 17. Total Mutual Inductance Examples

The examples in this table refer to Table 18 on page 35.

Example Condition	Type	Example Result
Pin F5 is assigned with the 1 V I/O standard. The surrounding pins, F4, H3, and H4 are in the same I/O bank and are also assigned with the 1.0 V I/O standard.	Intrabank, all 1.0 V	Total L_m of F4, H3, and H4 does not exceed 7.4 nH. The placement does not violate the restriction.
Pin F5 is assigned with the 1 V I/O standard. The surrounding pins, F4, H3, and H4 are in an adjacent I/O bank and are assigned with the 1.0 V I/O standard.	Interbank, all 1.0 V	Total L_m of F4, H3, and H4 does not exceed 7.4 nH. The placement does not violate the restriction.
Pin F5 is assigned with the 1 V I/O standard. The surrounding pins, F4, H3, and H4 are in an adjacent I/O bank and are assigned with the 2.5 V I/O standard.	Interbank, mixed voltages	Total L_m of F4, H3, and H4 exceeds 1.0 nH. Update your design to use other I/O pins with smaller mutual inductance.

Table 18. Example Mutual Inductance Values

Pin Name	Mutual Coupling Pin	Mutual Inductance (nH)
F5	F5	3.496 ⁽¹³⁾
F5	F4	1.378
F5	H3	0.273
F5	J4	0.263
F5	K4	0.222
F5	E4	0.194
F5	F3	0.176
F5	H4	0.175
F5	E3	0.174
F5	G3	0.167
F5	G4	0.161

Related Information

- [Intel MAX 10 Mutual Coupling \(max10-1v-mutual-coupling.zip\)](#)
 Provides spreadsheet files that list the mutual inductance values for 1.0 V I/O for the following Intel MAX 10 device combinations: 10M02/04/08/16/25/40/50 + SC/DC/SA/DA/DD + U324/U169/M153/F256/F484/F672 + I7/C7/C8
- [Guidelines: Placement Restrictions for 1.0 V I/O Pin](#) on page 34

⁽¹³⁾ Self inductance for pin F5. Omit this value from the L_m calculation.

3.7. Guidelines: Analog-to-Digital Converter I/O Restriction

These restrictions are applicable if you use the analog-to-digital converter (ADC) block.

The Intel Quartus Prime software uses physics-based rules to define the number of I/Os allowed in a particular bank based on the I/O's drive strength. These rules are based on noise calculation to analyze accurately the impact of I/O placement on the ADC performance.

The physics-based rules are available for the following devices starting from these Intel Quartus Prime software versions:

- From Intel Quartus Prime version 14.1—Intel MAX 10 10M04, 10M08, 10M40, and 10M50 devices.
- From Intel Quartus Prime version 15.0.1—Intel MAX 10 10M02, 10M16, and 10M25 devices.

Geometry-Based Rules for Design Estimation

Intel highly recommends that you use the following geometry-based rules to ensure ADC performance. These guidelines help you to estimate the resources available and prevent additional critical warning from versions of the Intel Quartus Prime software that implements the physics-based rules.

Table 19. Geometry-Based I/O Restrictions Related to ADC Usage

This table lists the I/O restrictions by Intel MAX 10 device package if you use one of the following features in your design:

- You use the dedicated analog input (ANAIN1 or ANAIN2) or any dual function ADC I/O pins as ADC channel inputs.
- You use the built-in temperature sensing diode (TSD).

Package	Restriction/Guideline
All	Disable all JTAG operation during ADC sampling. The ADC signal-to-noise and distortion ratio (SINAD) is not guaranteed during JTAG operation.
M153 U169 U324 F256 F484 F672	<ul style="list-style-type: none"> Banks 1A and 1B—you cannot use GPIO pins in these banks. Banks 2, 3, 4, 5, 6, and 7—you can use GPIO pins located in these banks. Bank 8—you can use a percentage of the GPIO pins in this bank based on drive strength: <ul style="list-style-type: none"> For an example listing the percentage of GPIO pins allowed in bank 8 for the F484 package, refer to Table 20 on page 37⁽¹⁴⁾. Use low drive strength (8 mA and below) and differential I/O standards. You can use static pins such as RESET or CONTROL. <p><i>Note:</i> The GPIO pins in bank 8 are constrained by physics-based rules. The Intel Quartus Prime software issues a critical warning if the I/O settings violate any of the I/O physics-based rule. Table 20 on page 37 only provides an example for your reference.</p>
E144	<ul style="list-style-type: none"> Bank 1A, 1B, 2, and 8—you cannot use GPIO pins in these banks. Banks 4 and 6—you can use GPIO pins located in these banks. Banks 3, 5, and 7—you can use a percentage of the GPIO pins in this bank based on drive strength: <ul style="list-style-type: none"> For the percentage of GPIO pins allowed, refer to Table 21 on page 37. Use low drive strength (8 mA and below) and differential I/O standards.

continued...

⁽¹⁴⁾ For all device packages, the software displays a warning message if the number of GPIO pins in bank 8 is more than the allowed percentage.

Package	Restriction/Guideline
	<i>Note:</i> The GPIO pins in banks 3, 5, and 7 are constrained by physics-based rules. The Intel Quartus Prime software issues a critical warning if the I/O settings violate any of the I/O physics-based rule. Table 21 on page 37 only provides an example for your reference.

Table 20. I/O Usage Restriction for Bank 8 in Intel MAX 10 F484 Package

This table lists the percentage of I/O pins available in I/O bank 8 if you use the dedicated analog input (ANAIN1 or ANAIN2) or any dual function ADC I/O pins as ADC channel. Refer to [Table 22](#) on page 37 for the list of I/O standards in each group.

I/O Standards	TX	RX	Total	Availability (%)
Group 1	18	18	36	100
Group 2	16	16	32	89
Group 3	7	11	18	50
Group 4	5	7	12	33
Group 5	4	6	10	28
Group 6	4	4	8	22
Group 7	0	8	8	22

Table 21. I/O Usage Restriction for Banks 3, 5, and 7 in Intel MAX 10 E144 Package

This table lists the percentage of I/O pins available in banks 3, 5, and 7 if you use the dedicated analog input (ANAIN1 or ANAIN2) or any dual function ADC I/O pins as ADC channel inputs. Refer to [Table 22](#) on page 37 for the list of I/O standards in each group.

I/O Standards	Bank 3			Bank 5			Bank 7			Device I/O Availability (%)
	TX	RX	Availability (%)	TX	RX	Availability (%)	TX	RX	Availability (%)	
Group 1	7	8	88	6	6	100	4	3	100	54
Group 2	7	8	88	6	6	100	4	3	100	54
Group 3	4	5	50	6	6	100	2	0	29	45
Group 4	3	4	39	5	5	83	0	0	0	39
Group 5	2	3	28	5	5	83	0	0	0	37
Group 6	1	2	17	5	5	83	0	0	0	35
Group 7	0	0	0	5	5	83	0	0	0	32

Table 22. I/O Standards Groups Categorized According to Drive Strengths

I/O Standard Group	I/O Standards Name and Drive Strength
Group 1	<ul style="list-style-type: none"> 1.8 V LVDS 2.5 V LVDS 2.5 V RSDS BLVDS at 4 mA SLVS at 4 mA
Group 2	<ul style="list-style-type: none"> BLVDS at 8 mA SLVS at 8 mA Sub-LVDS at 8 mA 1.8 V, 1.5 V, and 1.2 V HSTL Class I at 8 mA

continued...

I/O Standard Group	I/O Standards Name and Drive Strength
	<ul style="list-style-type: none"> SSTL-15 at 34 Ω or 40 Ω SSTL-135 at 34 Ω or 40 Ω HSUL-12 at 34 Ω or 40 Ω SSTL-2 Class I at 8 mA SSTL-18 Class I at 8 mA SSTL-15 Class I at 8 mA 2.5 V and 1.8 V LVTTTL at 4 mA 2.5 V, 1.8 V, 1.5 V, and 1.2 V LVCMOS at 4 mA 1.8 V LVTTTL at 2 mA 1.8 V, 1.5 V, and 1.2 V LVCMOS at 2 mA
Group 3	<ul style="list-style-type: none"> BLVDS at 12 mA SLVS at 12 mA Sub-LVDS at 12 mA SSTL-2 Class I at 10 mA or 12 mA SSTL-18 Class I at 10 mA or 12 mA SSTL-15 Class I at 10 mA or 12 mA 1.8 V, 1.5 V, and 1.2 V HSTL Class I at 10 mA or 12 mA SSTL-2 at 50 Ω SSTL-18 at 50 Ω SSTL-15 at 50 Ω 1.8 V, 1.5 V and 1.2 V HSTL at 50 Ω HSUL-12 at 48 Ω 2.5 V and 1.8 V LVTTTL at 50 Ω 2.5 V, 1.8 V, 1.5 V, and 1.2 V LVCMOS at 50 Ω 1.8 V LVTTTL at 6 mA or 8 mA 1.8 V, 1.5 V, and 1.2 V LVCMOS at 6 mA or 8 mA 1.0 V LVCMOS 3.0 V LVTTTL at 4 mA 3.0 V LVCMOS at 4 mA
Group 4	<ul style="list-style-type: none"> SSTL-18 Class II at 12 mA 3.0 V LVTTTL at 50 Ω 3.0 V LVCMOS at 50 Ω 2.5 V LVTTTL at 8 mA 2.5 V LVCMOS at 8 mA 1.8 V LVTTTL at 10 mA or 12 mA 1.8 V, 1.5 V, and 1.2 V LVCMOS at 10 mA or 12 mA 3.3 V LVCMOS at 2 mA
Group 5	<ul style="list-style-type: none"> SSTL-2 Class II at 16 mA SSTL-18 Class II at 16 mA SSTL-15 Class II at 16 mA 1.8 V and 1.5 V HSTL Class II at 16 mA 1.2 V HSTL Class II at 14 mA SSTL-18 at 25 Ω SSTL-15 at 25 Ω SSTL-2 at 25 Ω 1.8 V, 1.5 V, and 1.2 V HSTL at 25 Ω 2.5 V and 1.8 V LVTTTL at 25 Ω 2.5 V, 1.8 V, 1.5 V, and 1.2 V LVCMOS at 25 Ω 1.8 V LVTTTL at 16 mA 1.8 V and 1.5 V LVCMOS at 16 mA 2.5 V LVCMOS at 12 mA 2.5 V LVTTTL at 12 mA
continued...	

I/O Standard Group	I/O Standards Name and Drive Strength
	<ul style="list-style-type: none"> 3.0 V LVCMOS at 8 mA 3.0 V LVTTTL at 8 mA 3.3 V LVTTTL at 4 mA or 8 mA
Group 6	<ul style="list-style-type: none"> 2.5 V LVTTTL at 16 mA 2.5 V LVCMOS at 16 mA 3.0 V LVTTTL at 12 mA 3.0 V LVCMOS at 12 mA 3.0 V LVTTTL at 25 Ω 3.0 V LVCMOS at 25 Ω
Group 7	<ul style="list-style-type: none"> 3.0 V LVTTTL at 16 mA 3.0 V LVCMOS at 16 mA

3.8. Guidelines: External Memory Interface I/O Restrictions

These I/O rules are applicable if you use external memory interfaces in your design.

Two GPIOs Adjacent to DQ Pin Is Disabled

This limitation is applicable to Intel MAX 10 10M16, 10M25, 10M40, and 10M50 devices, and only if you use DDR3 and LPDDR2 SDRAM memory standards.

Table 23. DDR3 and LPDDR2 Memory Interface Widths and Device Packages Where Two GPIOs Adjacent to DQ Pins Are Disabled

This table lists the combination of Intel MAX 10 10M16, 10M25, 10M40, and 10M50 device packages, and DDR3 and LPDDR2 memory interface widths where you cannot use two GPIO pins that are adjacent to the DQ pins.

Device Package	Memory Interface Width (DDR3 and LPPDR2 only)
F256	x8, x16
U324	x8, x16
F484	x8, x16, x24
F672	x8, x16, x24

Total I/O Utilization in Bank Must Be 75 Percent or Less in Some Devices

If you use DDR3 or LPDDR2 SDRAM memory interface standards, you can generally use a maximum of 75 percent of the total number of I/O pins available in a bank. This restriction differs from device to device. In some devices packages you can use all 100 percent of the I/Os. The Intel Quartus Prime software will output an error message if the I/O usage per bank of that device is affected by this rule.

If you use DDR2 memory interface standards, you can assign 25 percent of the I/O pins as input pins only.

3.9. Guidelines: Dual-Purpose Configuration Pin

To use configuration pins as user I/O pins in user mode, you have to adhere to the following guidelines.

Table 24. Dual-Purpose Configuration Pin Guidelines for Intel MAX 10 Devices

Guidelines	Pins
Configuration pins during initialization: <ul style="list-style-type: none"> • Tri-state the external I/O driver and drive an external pull-up resistor⁽¹⁵⁾ or • Use the external I/O driver to drive the pins to the state same as the external weak pull-up resistor 	<ul style="list-style-type: none"> • nCONFIG • nSTATUS • CONF_DONE
JTAG pins: <ul style="list-style-type: none"> • If you intend to switch back and forth between user I/O pins and JTAG pin functions using the JTAGEN pin, all JTAG pins must be assigned as single-ended I/O pins or voltage-referenced I/O pins. Schmitt trigger input is the recommended input buffer. • JTAG pins cannot perform as JTAG pins in user mode if you assign any of the JTAG pin as a differential I/O pin. • You must use the JTAG pins as dedicated pins and not as user I/O pins during JTAG programming. • Do not toggle JTAG pin during the initialization stage. • Put the test access port (TAP) controller in reset state by driving the TDI and TMS pins high and toggle the TCK pin for at least 5 clock cycles before the initialization. • The Signal Tap logic analyzer IP, JTAG-to-Avalon® master bridge IP, and other JTAG-related IPs cannot be used if you enable the JTAG pin sharing feature in your design. 	<ul style="list-style-type: none"> • TDO • TMS • TCK • TDI

Attention: Assign all JTAG pins as single-ended I/O pins or voltage-referenced I/O pins if you enable JTAG pin sharing feature.

Related Information

Intel MAX 10 FPGA Configuration User Guide

Provides more information about the dual-purpose I/O pins in configuration and user modes.

3.10. Guidelines: Clock and Data Input Signal for Intel MAX 10 E144 Package

There is strong inductive coupling on the Intel MAX 10 E144 lead frame package. Glitch may occur on an input pin when an aggressor pin with strong drive strength toggles directly adjacent to it.

PLL Clock Input Pins

The PLL clock input pins are sensitive to SSN jitter. To avoid the PLL from losing lock, do not use the output pins directly on the left and right of the PLL clock input pins.

Data Input Pins

Potential glitch on the data input pin, leading to input read signal failure, can occur in the following conditions:

- The output pin directly adjacent to the data input pin is assigned an unterminated I/O standard, such as LVTTTL and LVCMOS, with drive strength of 8 mA or higher.
- The output pin directly adjacent to the data input pin is assigned a terminated I/O standard, such as SSTL, with drive strength of 8 mA or higher.

⁽¹⁵⁾ If you intend to remove the external weak pull-up resistor, Intel recommends that you remove it after the device enters user mode.

Intel recommends that you implement these guidelines to reduce jitter on the data input pin:

- For unterminated I/O standards, implement one of these guidelines:
 - For the directly-adjacent output pin with these unterminated I/O standards, reduce the drive strength as follows:
 - 2.5 V, 3.0 V, and 3.3 V—reduce to 4 mA or below
 - 1.2 V, 1.5 V, and 1.8 V—reduce to 6 mA or below
 - Assign the pins directly on the left and right of the data input pin to a non-toggling signal.
 - Change the data input pin to a Schmitt Trigger input buffer for better noise immunity. If you are using Schmitt Trigger input buffer on the data input pin, you can use the directly-adjacent output pin with unterminated I/O standard at a maximum drive strength of 8 mA.
- For terminated I/O standard, you can use only one pin directly on the left or right of the data input pin as toggling signal, provided that you set the slew rate setting of this pin to "0" (slow slew rate). Otherwise, assign the pins directly on the left and right of the data input pin to a non-toggling signal.

4. Intel MAX 10 I/O Implementation Guides

You can implement your I/O design in the Intel Quartus Prime software. The software contains tools for you to create and compile your design, and configure your device.

The Intel Quartus Prime software allows you to prepare for device migration, set pin assignments, define placement restrictions, setup timing constraints, and customize IP cores. For more information about using the Intel Quartus Prime software, refer to the related information.

Related Information

[Intel MAX 10 I/O Overview](#) on page 3

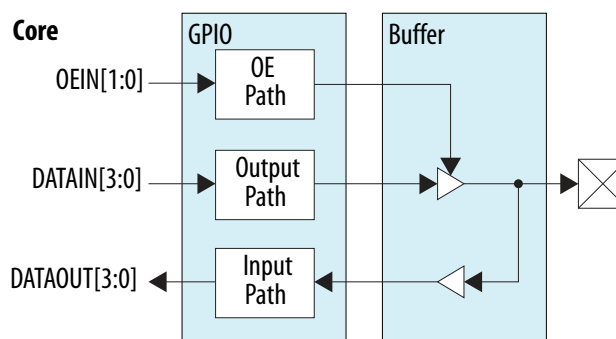
4.1. GPIO Lite Intel FPGA IP

The GPIO Lite IP core supports the Intel MAX 10 GPIO components. To implement the GPIOs in your design, you can customize the GPIO Lite IP core to suit your requirements and instantiate it in your design.

GPIOs are I/Os used in non-transceiver general applications, memory-like interfaces or LVDS applications. The GPIO Lite IP core features the following components:

- Double data rate input/output (DDIO)—A digital component that doubles the data-rate of a communication channel.
- I/O buffers—connect the pads to the FPGA.

Figure 14. High Level View of Single-Ended GPIO



Related Information

- [Introduction to Intel FPGA IP Cores](#)
Provides general information about all Intel FPGA IP cores, including parameterizing, generating, upgrading, and simulating IP cores.

- [Creating Version-Independent IP and Qsys Simulation Scripts](#)
Create simulation scripts that do not require manual updates for software or IP version upgrades.
- [Project Management Best Practices](#)
Guidelines for efficient management and portability of your project and IP files.

4.1.1. GPIO Lite Intel FPGA IP Data Paths

Table 25. GPIO Lite IP Core Data Path Modes

Data Path	Mode		
	Bypass	Single Register	DDR
Input	Data goes from the delay element to the core, bypassing all double data rate I/Os (DDIOs).	The full-rate DDIO operates as a single register.	The full-rate DDIO operates as a regular DDIO.
Output	Data goes from the core straight to the delay element, bypassing all DDIOs.	The full-rate DDIO operates as a single register.	The full-rate DDIO operates as a regular DDIO.
Bidirectional	The output buffer drives both an output pin and an input buffer.	The full-rate DDIO operates as a single register. The output buffer drives both an output pin and an input buffer.	The full-rate DDIO operates as a regular DDIO. The output buffer drives both an output pin and an input buffer. The input buffer drives a set of three flip-flops.

If you use asynchronous clear and preset signals, all DDIOs share these same signals.

4.1.1.1. DDR Input Path

The pad sends data to the input buffer and the input buffer feeds the delay element. From the delay element, the data is fed to the DDIO stage, which consists of three registers:

- RegAi samples the data from pad_in at the positive clock edge.
- RegBi samples the data from pad_in at the negative clock edge.
- RegCi samples the data from RegAi at the negative clock edge.

Figure 15. Simplified View of GPIO Lite IP Core DDR Input Path

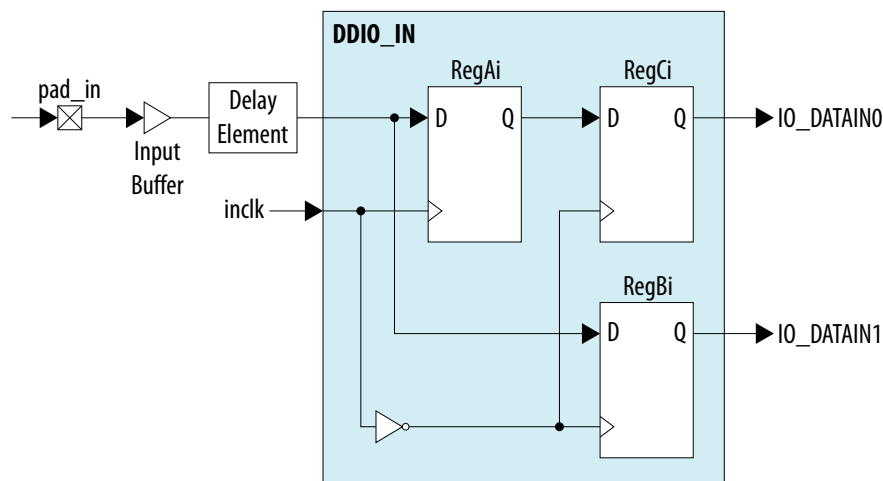
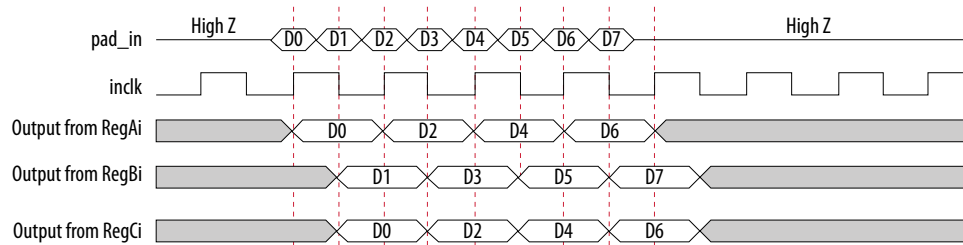
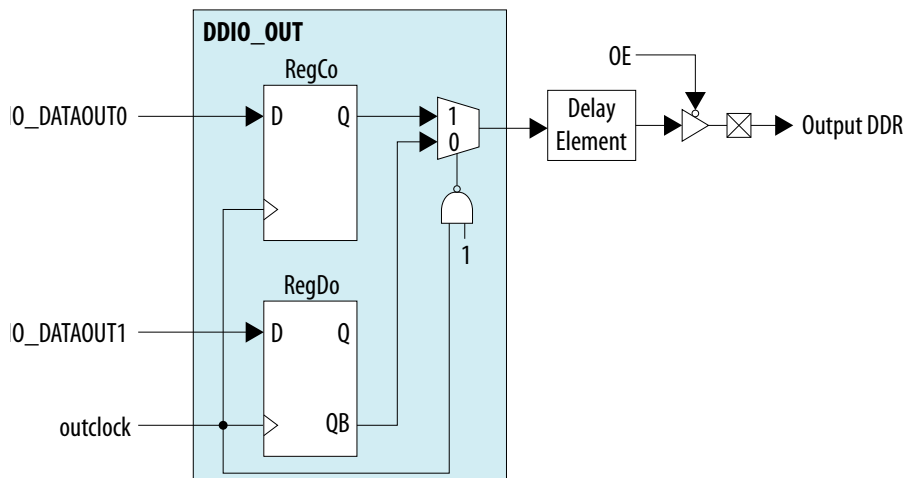


Figure 16. GPIO Lite IP Core Input Path Timing Diagram



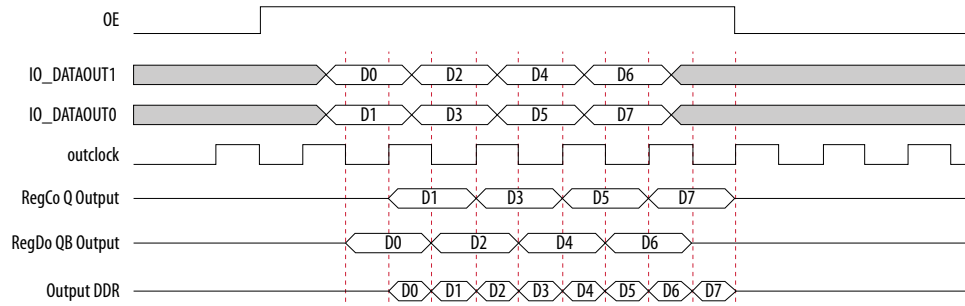
4.1.1.2. DDR Output Path with Output Enable

Figure 17. Simplified View of GPIO Lite IP Core DDR Output Path with Output Enable



- RegCo samples the data from IO_DATAOUT0 at the positive clock edge.
- RegDo samples the data from IO_DATAOUT1 when outclock value is 0.
- Output DDR samples the data from RegCo at the positive clock edge, and from RegDo at the negative clock edge.

Figure 18. GPIO Lite IP Core Output Path Timing Diagram



- The IP core feeds the first bit, D0, through IO_DATAOUT1 to RegDo. The IP core clocks out this bit at the RegDo QB port on a negative clock edge. At the next positive clock edge, the IP core produces the same bit at the multiplexer output.
- The IP core feeds the second bit, D1, through IO_DATAOUT0 to RegCo. The IP core clocks out this bit at the RegCo Q port on a positive clock edge. At the next negative clock edge, the IP core produces the same bit at the multiplexer output.

4.2. Verifying Pin Migration Compatibility

You can use the **Pin Migration View** window in the Intel Quartus Prime software Pin Planner to assist you in verifying whether your pin assignments migrate to a different device successfully.

You can vertically migrate to a device with a different density while using the same device package, or migrate between packages with different densities and ball counts.

1. Open **Assignments > Pin Planner** and create pin assignments.
2. If necessary, perform one of the following options to populate the Pin Planner with the node names in the design:
 - Analysis & Elaboration
 - Analysis & Synthesis
 - Fully compile the design
3. Then, on the menu in **Pin Planner**, click **View > Pin Migration Window**.
4. To select or change migration devices:
 - a. Click **Device** to open the **Device** dialog box.
 - b. Click **Migration Devices**.
5. To show more information about the pins:
 - a. Right-click anywhere in the **Pin Migration View** window and select **Show Columns**.
 - b. Then, click the pin feature you want to display.

6. If you want to view only the pins, in at least one migration device, that have a different feature than the corresponding pin in the migration result, turn on **Show migration differences**.
7. Click **Pin Finder** to open the **Pin Finder** dialog box and find and highlight pins with specific functionality.
If you want to view only the pins found and highlighted by the most recent query in the **Pin Finder** dialog box, turn on **Show only highlighted pins**.
8. To export the pin migration information to a Comma-Separated Value File (.csv), click **Export**.

Related Information

[Intel MAX 10 I/O Vertical Migration Support](#) on page 5

5. GPIO Lite Intel FPGA IP References

You can set various parameter settings for the GPIO Lite IP core to customize its behaviors, ports, and signals.

The Intel Quartus Prime software generates your customized GPIO Lite IP core according to the parameter options that you set in the parameter editor.

Related Information

[Intel MAX 10 I/O Overview](#) on page 3

5.1. GPIO Lite Intel FPGA IP Parameter Settings

You can set the parameter settings for the GPIO Lite IP core in the Intel Quartus Prime software. There are three groups of options: **General**, **Buffer**, and **Registers**.

Table 26. GPIO Lite Parameters - General

Parameter	Condition	Allowed Values	Description
Data direction	—	<ul style="list-style-type: none"> input output bidir 	Specifies the data direction for the GPIO.
Data width	—	1 to 128	Specifies the data width.

Table 27. GPIO Lite Parameters - Buffer

Parameter	Condition	Allowed Values	Description
Use true differential buffer	Data direction = input or output	<ul style="list-style-type: none"> On Off 	If turned on, enables true differential I/O buffers and disables pseudo differential I/O buffers.
Use pseudo differential buffer	Data direction = output or bidir	<ul style="list-style-type: none"> On Off 	<ul style="list-style-type: none"> If turned on in output mode—enables pseudo differential output buffers and disables true differential I/O buffers. If turned on in bidir mode—enables true differential input buffer and pseudo differential output buffer.
Use bus-hold circuitry	Data direction = input or output	<ul style="list-style-type: none"> On Off 	If turned on, the bus hold circuitry can weakly hold the signal on an I/O pin at its last-driven state where the output buffer state is 1 or 0 but not high-impedance.
<i>continued...</i>			

Parameter	Condition	Allowed Values	Description
Use open drain output	Data direction = output or bidir	<ul style="list-style-type: none"> On Off 	If turned on, the open drain output enables the device to provide system-level control signals such as interrupt and write enable signals that can be asserted by multiple devices in your system.
Enable oe port	Data direction = output	<ul style="list-style-type: none"> On Off 	If turned on, enables user input to the OE port. This option is automatically turned on for bidirectional mode.
Enable nsleep port (only available in selected devices)	Data direction = input or bidir	<ul style="list-style-type: none"> On Off 	If turned on, enables the nsleep port. This option is available for the 10M16, 10M25, 10M40, and 10M50 devices.

Table 28. GPIO Lite Parameters - Registers

Parameter	Condition	Allowed Values	Description
Register mode	—	<ul style="list-style-type: none"> bypass single-register ddr 	Specifies the register mode for the GPIO Lite IP core: <ul style="list-style-type: none"> bypass—specifies a simple wire connection from/to the buffer. single-register—specifies that the DDIO is used as a simple register in single data-rate mode (SDR). The Fitter may pack this register in the I/O. ddr— specifies that the IP core uses the DDIO.
Enable aclr port	<ul style="list-style-type: none"> Register mode = ddr 	<ul style="list-style-type: none"> On Off 	If turned on, enables the ACLR port for asynchronous clears.
Enable aset port	<ul style="list-style-type: none"> Data direction = output or bidir Register mode = ddr Set registers to power up high (when aclr and aset ports are not used) = off 	<ul style="list-style-type: none"> On Off 	If turned on, enables the ASET port for asynchronous preset.
Set registers to power up high (when aclr and aset ports are not used)	<ul style="list-style-type: none"> Register mode = ddr Enable aclr port = off Enable aset port = off Enable sclr port = off 	<ul style="list-style-type: none"> On Off 	If you are not using the ACLR and ASET ports: <ul style="list-style-type: none"> On—specifies that registers power up HIGH. Off—specifies that registers power up LOW.
Enable inclocken/outclocken ports	Register mode = ddr	<ul style="list-style-type: none"> On Off 	<ul style="list-style-type: none"> On—exposes the clock enable port to allow you to control when data is clocked in or out. This signal prevents data from being passed through without your control. Off—clock enable port is not exposed and data always pass through the register automatically.
Invert din	<ul style="list-style-type: none"> Data direction = output Register mode = ddr 	<ul style="list-style-type: none"> On Off 	If turned on, inverts the data out output port.
Invert DDIO inclock	<ul style="list-style-type: none"> Data direction = input or bidir Register mode = ddr 	<ul style="list-style-type: none"> On Off 	<ul style="list-style-type: none"> On—captures the first data bit on the falling edge of the input clock. Off—captures the first data bit on the rising edge of the input clock.
continued...			

Parameter	Condition	Allowed Values	Description
Use a single register to drive the output enable (oe) signal at the I/O buffer	<ul style="list-style-type: none"> Data direction = output or bidir Register mode = single-register or ddr Use DDIO registers to drive the output enable (oe) signal at the I/O buffer = off 	<ul style="list-style-type: none"> On Off 	If turned on, specifies that a single register drives the OE signal at the output buffer.
Use DDIO registers to drive the output enable (oe) signal at the I/O buffer	<ul style="list-style-type: none"> Data direction = output or bidir Register mode = ddr Use a single register to drive the output enable (oe) signal at the I/O buffer = off 	<ul style="list-style-type: none"> On Off 	If turned on, specifies that the DDR I/O registers drive the OE signal at the output buffer. The output pin is held at high impedance for an extra half clock cycle after the OE port goes high.
Implement DDIO input registers in hard implementation (Only available in certain devices)	<ul style="list-style-type: none"> Data direction = input or bidir Register mode = ddr 	<ul style="list-style-type: none"> On Off 	<ul style="list-style-type: none"> On—implements the DDIO input registers using hard block at the I/O edge. Off—implements the DDIO input registers as soft implementation using registers in the FPGA core fabric. <p>This option is applicable only for Intel MAX 10 16, 25, 40, and 50 devices because the DDIO input registers hard block is available only in these devices. To avoid Fitter error, turn this option off for other Intel MAX 10 devices.</p>

5.2. GPIO Lite Intel FPGA IP Interface Signals

Depending on parameter settings you specify, different interface signals are available for the GPIO Lite IP core.

Table 29. Pad Interface Signals

The pad interface connects the GPIO Lite IP core to the pads.

Signal Name	Direction	Description
pad_in	Input	Input pad port if you use the input path.
pad_in_b	Input	Input negative pad port if you use the input path and enable the true or pseudo differential buffers.
pad_out	Output	Output pad port if you use the output path.
pad_out_b	Output	Output negative pad port if you use the output path and enable the true or pseudo differential buffers.
pad_io	Bidirectional	Bidirectional pad port if you use bidirectional paths.
pad_io_b	Bidirectional	Bidirectional negative pad port if you use bidirectional paths and enable true or pseudo differential buffers.

Table 30. Data Interface Signals

The data interface is an input or output interface from the GPIO Lite IP core to the FPGA core.

Signal Name	Direction	Description
din	Input	Data received from the input pin.
continued...		

Signal Name	Direction	Description
		Signal width for each input pin: <ul style="list-style-type: none"> DDR mode—2 Other modes—1
dout	Output	Data to send out through the output pin. Signal width for each output pin: <ul style="list-style-type: none"> DDR mode—2 Other modes—1
oe	Input	Control signal that enables the output buffer. This signal is active HIGH.
nsleep	Input	Control signal that enables the input buffer. This signal is active LOW. This signal is available for the 10M16, 10M25, 10M40, and 10M50 devices.

Table 31. Clock Interface Signals

The clock interface is an input clock interface. It consists of different signals, depending on the configuration. The GPIO Lite IP core can have zero, one, two, or four clock inputs. Clock ports appear differently in different configurations to reflect the actual function performed by the clock signal.

Signal Name	Direction	Description
inclock	Input	Input clock that clocks the registers in the input path.
inclocken	Input	Control signal that controls when data is clocked in. This signal is active HIGH.
outclock	Input	Input clock that clocks the registers in the output path.
outclocken	Input	Control signal that controls when data is clocked out. This signal is active HIGH.

Table 32. Reset Interface Signals

The reset interface connects the GPIO Lite IP core to the DDIOs.

Signal Name	Direction	Description
aclr	Input	Control signal for asynchronous clear that sets the register output state to 0. This signal is active HIGH.
aset	Input	Control signal for asynchronous preset that sets the register output state to 1. This signal is active HIGH.
sclr	Input	Control signal for synchronous clear that sets the register output to 0. This signal is active HIGH.



6. Intel MAX 10 General Purpose I/O User Guide Archives

For the latest and previous versions of this user guide, refer to [Intel MAX 10 General Purpose I/O User Guide](#). If an IP or software version is not listed, the user guide for the previous IP or software version applies.

7. Document Revision History for Intel MAX 10 General Purpose I/O User Guide

Document Version	Intel Quartus Prime Version	Changes
2022.10.31	22.1	<ul style="list-style-type: none"> Updated the list of Intel MAX 10 device packages that support the 1.0 V LVCMOS I/O standard. Added 1.8 V LVDS I/O standard support. Updated the spreadsheet files (max10-1v-mutual-coupling.zip) that list the mutual inductance values for 1.0 V I/O.
2022.01.27	21.1	Updated the spreadsheet files (max10-1v-mutual-coupling.zip) that list the mutual inductance values for 1.0 V I/O.
2021.11.01	21.1	<ul style="list-style-type: none"> Added V81 and Y180 packages in the <i>Package Plan for Intel MAX 10 Single Power Supply Devices</i> table. Added Y180 package in the <i>Migration Capability Across Intel MAX 10 Devices</i> diagram. Updated footnote for 1.0 V LVCMOS to include new devices in the <i>Supported I/O Standards in Intel MAX 10 Devices</i> table. Updated figure title to <i>I/O Banks for 10M02 (Single Power Supply U324 Package), 10M04, and 10M08 (Except V81, M153, and U169 Packages) Devices</i>. Added diagram: <i>I/O Banks for 10M08 V81, M153, and U169 Packages Devices</i>. Updated max10-1v-mutual-coupling.zip file and the link description to include new devices. Added F256 device package and updated U324 device package in the <i>DDR3 and LPDDR2 Memory Interface Widths and Device Packages Where Two GPIOs Adjacent to DQ Pins Are Disabled</i> table.
2021.04.27	20.1	Updated the guidelines in the table listing the geometry-based I/O restrictions related to ADC usage.
2020.11.05	20.1	Updated the guidelines for JTAG pins in table <i>Dual-Purpose Configuration Pin Guidelines for Intel MAX 10 Devices</i> .
2020.09.22	20.1	Updated the clamp diode for LVTTTL/LVCMOS input buffers guidelines to remove references to "undershoot". The clamp diode manages overshoot voltages only.
2020.08.24	20.1	Updated the table in the I/O restriction rules guideline topic to improve clarity.
2020.06.30	20.1	<ul style="list-style-type: none"> Added support for 1.0 V LVCMOS I/O standard. Added placement restriction guideline for 1.0 V I/O pins.
2019.01.01	18.1	Removed support for 1.0 V LVCMOS I/O standard.
2018.12.20	18.1	<ul style="list-style-type: none"> Updated introductory statements about GPIO usage to improve clarity. Added support for 1.0 V LVCMOS I/O standard for commercial grade devices only. Added link to the list of Intel MAX 10 development kits and boards. Added statement to clarify that when the Intel MAX 10 device is blank or erased, the I/Os are tri-stated.
continued...		

Intel Corporation. All rights reserved. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

*Other names and brands may be claimed as the property of others.

Document Version	Intel Quartus Prime Version	Changes
		<ul style="list-style-type: none"> Updated the guideline for V_{CCIO} range to improve clarity. Updated the topic about the PCI clamp diode and added links to related information. Updated the topic about programmable emulated differential output to improve clarity.

Date	Version	Changes
December 2017	2017.12.15	<ul style="list-style-type: none"> Added the U324 package for the Intel MAX 10 single power supply devices. Updated the I/O vertical migration figure. Added a topic about the different I/O banks performance. Updated the GPIO Lite DDR output path figure, timing diagram, and added descriptions to improve clarity. Updated the description in the guideline topic about I/O restrictions to improve clarity. Updated the guideline topic about the clock and data input signal for the E144 package to improve clarity. Updated the guideline topic about the ADC I/O restriction to clarify that the guidelines are geometry-based rules for design estimation purpose. Removed all "Preliminary" markers. Updated the topic about the PCI clamp diode to remove the sentence that mention the active serial (AS) configuration scheme. Intel MAX 10 devices do not support the AS configuration scheme. Updated the guideline topic about enabling the clamp diode for the LVTTTL/LVCMOS input buffers to improve clarity.
February 2017	2017.02.21	Rebranded as Intel.
May 2016	2016.05.02	<ul style="list-style-type: none"> Updated the list of supported I/O standards to specify I/O standards that are supported only in dual power supply Intel MAX 10 devices. Updated the names of emulated differential I/O standards to improve clarity. Updated the topic about the I/O standards voltage and pin support to clarify that the I/O standards that a pin type supports depends on pin's I/O bank. Updated the setting information for PCI clamp diode: <ul style="list-style-type: none"> On by default for input pins for all supported I/O standards Off by default for output pins for all supported I/O standards, except 3.0 V PCI Updated the topic about the ADC I/O restriction: <ul style="list-style-type: none"> Added the list of devices with physics-based rules support from Intel Quartus Prime version 15.0.1. Clarified that the table listing the percentage of GPIOs allowed in bank 8 is an example for the F484 package. For all packages, the Intel Quartus Prime software displays a warning message if you exceed the allowed GPIO percentage.
November 2015	2015.11.02	<ul style="list-style-type: none"> Added PCI clamp diode support for the 3.3 V and 2.5 V Schmitt Trigger I/O standards. Added a table that summarizes the programmable I/O buffer features and settings. Updated the topics about V_{CCIO} range consideration and VREF I/O standards restriction with guidelines for using different V_{CCIO} supplies in bank 1A and bank 1B. Added guidelines topic about using the clock and input pins in the E144 package.
continued...		

Date	Version	Changes
		<ul style="list-style-type: none"> Added the Enable nsleep port parameter option. Removed the topics about the IP catalog and parameter editor, generating IP cores, and the files generated by the IP core, and added a link to <i>Introduction to Intel IP Cores</i>. Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i>.
June 2015	2015.06.10	<ul style="list-style-type: none"> Added related link to the Intel MAX 10 device pin-outs in topic about I/O banks locations. The device pin-out files provide more information about available I/O pins in each I/O bank. Updated the ADC I/O restriction guidelines topic.
May 2015	2015.05.04	<ul style="list-style-type: none"> Removed the F672 package of the Intel MAX 10 10M25 device. Updated footnote for LVDS (dedicated) in the table listing the supported I/O standards to clarify that you can use LVDS receivers on all I/O banks. Added missing footnote number for the DQS column of the 3.3 V Schmitt Trigger row in the table that lists the I/O standards voltage levels and pin support. Added a table listing the I/O standards and current strength settings that support programmable output slew rate control. Updated the topic about external memory interface I/O restrictions to add x24 memory interface width to the F484 package. Added topic about the programmable differential output voltage. Updated the guidelines for voltage-referenced I/O standards to add a list of device packages that do not support voltage-referenced I/O standards. Updated the topic about the I/O restriction rules to remove statements about the differential pad placement rules. Renamed the <code>input_ena</code> signal name to <code>nsleep</code> and updated the relevant description. Updated the description for the Invert DDIO inclock parameter of the GPIO Lite IP core.
December 2014	2014.12.15	Updated the topic about the ADC I/O restriction: <ul style="list-style-type: none"> Added information about implementation of physics-based rules in the Intel Quartus Prime software. Updated the list of I/O standards groups for the ADC I/O restriction.
September 2014	2014.09.22	Initial release.



Intel® MAX® 10 Analog to Digital Converter User Guide

Updated for Intel® Quartus® Prime Design Suite: **22.1**



Online Version



Send Feedback

UG-M10ADC

683596

2024.01.03

Contents

1. Intel® MAX® 10 Analog to Digital Converter Overview.....	4
1.1. ADC Block Counts in Intel MAX 10 Devices.....	5
1.2. ADC Channel Counts in Intel MAX 10 Devices.....	6
1.3. Intel MAX 10 ADC Vertical Migration Support.....	7
1.4. Intel MAX 10 Single or Dual Supply Devices.....	8
1.5. Intel MAX 10 ADC Conversion.....	8
1.5.1. Voltage Representation Conversion.....	9
2. Intel MAX 10 ADC Architecture and Features.....	11
2.1. Intel MAX 10 ADC Hard IP Block.....	11
2.1.1. ADC Block Locations.....	12
2.1.2. Single or Dual ADC Devices.....	14
2.1.3. ADC Analog Input Pins.....	15
2.1.4. ADC Prescaler.....	15
2.1.5. ADC Clock Sources.....	16
2.1.6. ADC Voltage Reference.....	16
2.1.7. ADC Temperature Sensing Diode.....	16
2.1.8. ADC Sequencer.....	19
2.1.9. ADC Timing.....	20
2.2. Modular ADC Core and Modular Dual ADC Core IP Cores.....	20
2.2.1. Modular ADC Core IP Core Configuration Variants.....	21
2.2.2. Modular ADC Core and Modular Dual ADC Core IP Cores Architecture.....	26
2.3. Intel FPGA ADC HAL Driver.....	30
2.4. ADC Toolkit for Testing ADC Performance.....	30
2.5. ADC Logic Simulation Output.....	31
2.5.1. Fixed ADC Logic Simulation Output.....	31
2.5.2. User-Specified ADC Logic Simulation Output.....	32
3. Intel MAX 10 ADC Design Considerations.....	34
3.1. Guidelines: ADC Ground Plane Connection.....	34
3.2. Guidelines: Board Design for Power Supply Pin and ADC Ground (REFGND).....	34
3.3. Guidelines: Board Design for Analog Input.....	35
3.4. Guidelines: Board Design for ADC Reference Voltage Pin.....	37
4. Intel MAX 10 ADC Implementation Guides.....	39
4.1. Creating Intel MAX 10 ADC Design.....	40
4.2. Customizing and Generating Modular ADC Core IP Core.....	40
4.3. Parameters Settings for Generating ALTPLL IP Core.....	41
4.4. Parameters Settings for Generating Modular ADC Core or Modular Dual ADC Core IP Core.....	42
4.5. Completing ADC Design.....	45
5. Modular ADC Core Intel FPGA IP and Modular Dual ADC Core Intel FPGA IP References.....	46
5.1. Modular ADC Core Parameters Settings.....	47
5.1.1. Modular ADC Core IP Core Channel Name to Intel MAX 10 Device Pin Name Mapping.....	49
5.2. Modular Dual ADC Core Parameters Settings.....	51

5.2.1. Modular Dual ADC Core IP Core Channel Name to Intel MAX 10 Device Pin Name Mapping.....	53
5.3. Valid ADC Sample Rate and Input Clock Combination.....	54
5.4. Modular ADC Core and Modular Dual ADC Core Interface Signals.....	55
5.4.1. Command Interface of Modular ADC Core and Modular Dual ADC Core.....	55
5.4.2. Response Interface of Modular ADC Core and Modular Dual ADC Core.....	56
5.4.3. Threshold Interface of Modular ADC Core and Modular Dual ADC Core.....	56
5.4.4. CSR Interface of Modular ADC Core and Modular Dual ADC Core.....	57
5.4.5. IRQ Interface of Modular ADC Core and Modular Dual ADC Core.....	57
5.4.6. Peripheral Clock Interface of Modular ADC Core and Modular Dual ADC Core....	58
5.4.7. Peripheral Reset Interface of Modular ADC Core and Modular Dual ADC Core....	58
5.4.8. ADC PLL Clock Interface of Modular ADC Core and Modular Dual ADC Core.....	58
5.4.9. ADC PLL Locked Interface of Modular ADC Core and Modular Dual ADC Core....	59
5.5. Modular ADC Core Register Definitions.....	59
5.5.1. Sequencer Core Registers.....	59
5.5.2. Sample Storage Core Registers.....	60
5.6. ADC HAL Device Driver for Nios Processor.....	61
5.6.1. Driver API.....	62
6. Intel MAX 10 Analog to Digital Converter User Guide Archives.....	65
7. Document Revision History for Intel MAX 10 Analog to Digital Converter User Guide...	66

1. Intel® MAX® 10 Analog to Digital Converter Overview

Intel® MAX® 10 devices feature up to two analog-to-digital converters (ADC). The ADCs provide the Intel MAX 10 devices with built-in capability for on-die temperature monitoring and external analog signal conversion.

The ADC solution consists of hard IP blocks in the Intel MAX 10 device periphery and soft logic through the Modular ADC Core Intel FPGA IP and Modular Dual ADC Core Intel FPGA IP.

The ADC solution provides you with built-in capability to translate analog quantities to digital data for information processing, computing, data transmission, and control systems. The basic function is to provide a 12 bit digital representation of the analog signal being observed.

The ADC solution works in two modes:

- Normal mode—monitors single-ended external inputs with a cumulative sampling rate of up to 1 million samples per second (MSPS):
 - Single ADC devices—up to 17 single-ended external inputs (one dedicated analog and 16 dual function input pins)
 - Dual ADC devices—up to 18 single-ended external inputs (one dedicated analog and eight dual function input pins in each ADC block)
- Temperature sensing mode—monitors external temperature data input with a sampling rate of up to 50 kilosamples per second. In dual ADC devices, only the first ADC block supports this mode.

Related Information

- [Intel MAX 10 ADC Architecture and Features](#) on page 11
- [Intel MAX 10 ADC Design Considerations](#) on page 34
- [Intel MAX 10 ADC Implementation Guides](#) on page 39
- [Modular ADC Core Intel FPGA IP and Modular Dual ADC Core Intel FPGA IP References](#) on page 46
- [Intel MAX 10 Getting Started](#)
- [Intel MAX 10 Online Training](#)
- [Intel MAX 10 How-to Videos](#)
- [How to Create ADC Design in Intel MAX 10 Device Using Platform Designer \(Standard\) Tool](#)
Provides video instruction that demonstrates how to create the ADC design in Intel MAX 10 devices using the Qsys system integration tool within the Intel Quartus® Prime software and how to use the ADC toolkit to view the measured analog signal.

- [How to Create Simultaneous Measurement with Intel MAX 10 ADC, Part 1](#)
Provides the first part of video instruction series that explains the differences between the Intel MAX 10 Modular ADC Core and Modular Dual ADC Core IP cores. The video also demonstrates how to create a simple simultaneous ADC measurement and how to place signal taps to measure the digital code output for analog signal.
- [How to Create Simultaneous Measurement with Intel MAX 10 ADC, Part 2](#)
Provides the second part of video instruction series that explains the differences between the Intel MAX 10 Modular ADC Core and Modular Dual ADC Core IP cores. The video also demonstrates how to create a simple simultaneous ADC measurement and how to place signal taps to measure the digital code output for analog signal.

1.1. ADC Block Counts in Intel MAX 10 Devices

The ADC block is available in single and dual supply Intel MAX 10 devices.

Table 1. Number of ADC Blocks in Intel MAX 10 Devices and Packages

For more information about the device part numbers that feature ADC blocks, refer to the device overview.

Package	Power Supply	Device					
		10M04	10M08	10M16	10M25	10M40	10M50
M153	Single	1	1	—	—	—	—
U169	Single	1	1	1	—	—	—
U324	Single	1	1	1	—	—	—
	Dual	1	1	1	—	—	—
F256	Dual	1	1	1	2	2	2
E144	Single	1	1	1	1	1	1
F484	Dual	—	1	1	2	2	2
F672	Dual	—	—	—	—	2	2

Related Information

[Intel MAX 10 FPGA Device Overview](#)

1.2. ADC Channel Counts in Intel MAX 10 Devices

Different Intel MAX 10 devices support different number of ADC channels.

Table 2. ADC Channel Counts in Intel MAX 10 Devices

- Devices with two ADC blocks have two dedicated analog inputs and each ADC block has 8 dual function pins. You can use the dual function pins in an ADC block as general purpose I/O (GPIO) pins if you do not use the ADC.
- For more information about the device part numbers that feature ADC blocks, refer to the device overview.

Package	Pin Type	ADC Channel Counts Per Device					
		10M04	10M08	10M16	10M25	10M40	10M50
M153	Dedicated	1	1	—	—	—	—
	Dual function	8	8	—	—	—	—
U169	Dedicated	1	1	1	—	—	—
	Dual function	8	8	8	—	—	—
U324	Dedicated	1	1	1	—	—	—
	Dual function	16	16	16	—	—	—
F256	Dedicated	1	1	1	2	2	2
	Dual function	16	16	16	16	16	16
E144	Dedicated	1	1	1	1	1	1
	Dual function	8	8	8	8	8	8
F484	Dedicated	—	1	1	2	2	2
	Dual function	—	16	16	16	16	16
F672	Dedicated	—	—	—	—	2	2
	Dual function	—	—	—	—	16	16

Related Information

- [Intel MAX 10 FPGA Device Overview](#)
- [Intel MAX 10 ADC Vertical Migration Support](#) on page 7

1.3. Intel MAX 10 ADC Vertical Migration Support

Figure 1. ADC Vertical Migration Across Intel MAX 10 Devices

The arrows indicate the ADC migration paths. The devices included in each vertical migration path are shaded.

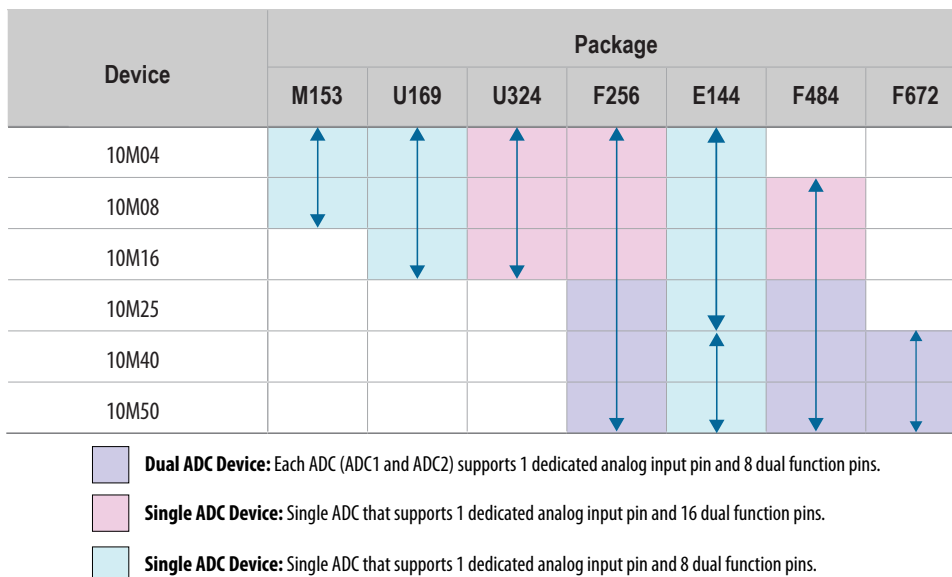


Table 3. Pin Migration Conditions for ADC Migration

Source	Target	Migratable Pins
Single ADC device	Single ADC device	You can migrate all ADC input pins
Dual ADC device	Dual ADC device	
Single ADC device	Dual ADC device	<ul style="list-style-type: none"> One dedicated analog input pin. Eight dual function pins from the ADC1 block of the source device to the ADC1 block of the target device.
Dual ADC device	Single ADC device	

Related Information

[ADC Channel Counts in Intel MAX 10 Devices](#) on page 6

1.4. Intel MAX 10 Single or Dual Supply Devices

Intel MAX 10 devices are available in single or dual supply packages.

- For devices with single power supply:
 - Use on chip regulator to power up the digital supply.
 - Use V_{CCA} to power up the ADC analog.
- For dual power supply devices, you must provide external power supplies of 1.2 V and 2.5 V to power up the ADC.

To choose the correct device, refer to the Intel MAX 10 device overview.

For more information about the ADC parameter, refer to the device datasheet.

Related Information

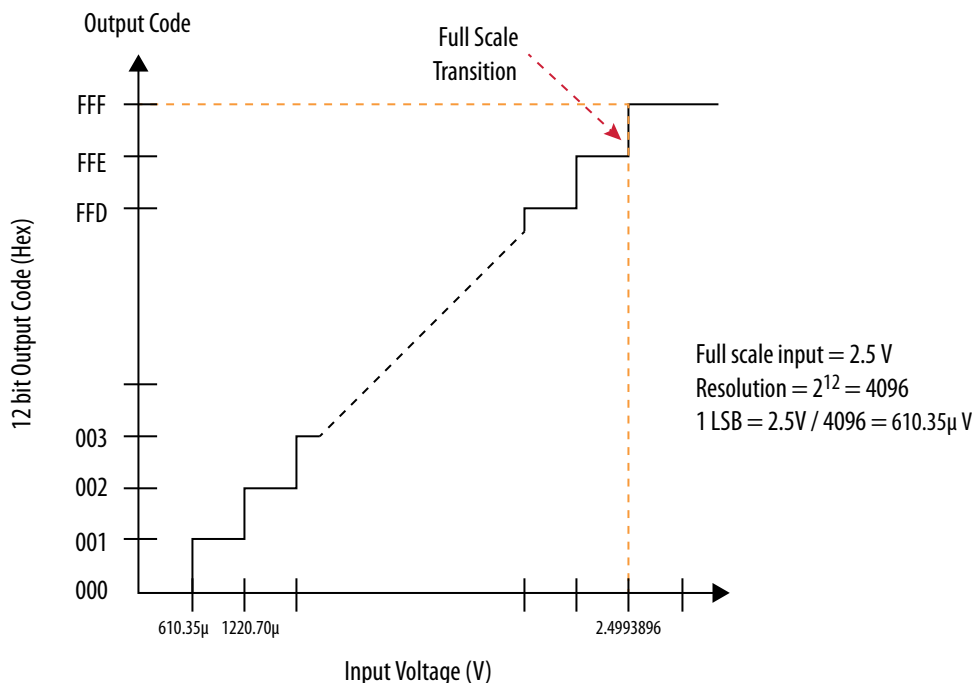
- [Intel MAX 10 Device Datasheet](#)
- [Intel MAX 10 FPGA Device Overview](#)

1.5. Intel MAX 10 ADC Conversion

The ADC in dual supply Intel MAX 10 devices can measure from 0 V to 2.5 V. In single supply Intel MAX 10 devices, it can measure up to 3.0 V or 3.3 V, depending on your power supply voltage.

- In prescaler mode, the analog input can measure up to 3.0 V in dual supply Intel MAX 10 devices and up to 3.6 V in single supply Intel MAX 10 devices.
- The analog input scale has full scale code from 000h to FFFh. However, the measurement can only display up to *full scale – 1 LSB*.
- For the 12 bits corresponding value calculation, use unipolar straight binary coding scheme.

Figure 2. ADC Measurement Display for 2.5 V



The Intel MAX 10 ADC is a 1 MHz successive approximation register (SAR) ADC. If you set up the PLL and Modular ADC Core IP core correctly, the ADC operates at up to 1 MHz during normal sampling and 50 kHz during temperature sensing.

Note: The analog value represented by the all-ones code is not full scale but *full scale – 1 LSB*. This is a common convention in data conversion notation and applies to ADCs.

Related Information

- [Creating Intel MAX 10 ADC Design](#) on page 40
- [Modular ADC Core Parameters Settings](#) on page 47
- [Modular Dual ADC Core Parameters Settings](#) on page 51

1.5.1. Voltage Representation Conversion

Use the following equations to convert the voltage between analog value and digital representation.

Equation 1. Conversion from Analog Value to Digital Code

$$\text{Digital Code} = \left(\frac{V_{IN}}{V_{REF}} \right) \times 2^{12}$$

Equation 2. Conversion from Digital Code to Analog Value

$$\text{Analog Value} = \text{Digital Code} \times \left(\frac{V_{REF}}{2^{12}} \right)$$

Example 1. Calculation Example for V_{REF} of 2.5 V

Analog voltage value to digital code (in decimal), where signal in is 2 V:

$$\text{Digital Code} = \left(\frac{2}{2.5} \right) \times 4096 = 3277$$

Digital code to analog voltage value, approximation to 4 decimal points:

$$\text{Analog Value} = 3277 \times \left(\frac{2.5}{4096} \right) = 2.0000$$



2. Intel MAX 10 ADC Architecture and Features

In Intel MAX 10 devices, the ADC is a 12-bit SAR ADC that provides the following features:

- Sampling rate of up to 1 MSPS
- Up to 18 channels for analog measurement: 16 dual function channels and two dedicated analog input channels in dual ADC devices
- Single-ended measurement capability
- Simultaneous measurement capability at the dedicated analog input pins for dual ADC devices
- Soft logic sequencer
- On-chip temperature sensor with sampling rate of 50 kilosamples per second
- Internal or external voltage references usage. The source of the internal voltage reference is the ADC analog supply; the ADC conversion result is ratiometric.

Related Information

- [Intel MAX 10 Analog to Digital Converter Overview](#) on page 4
- [Intel MAX 10 Analog to Digital Converter User Guide Archives](#) on page 65
Provides a list of user guides for previous versions of the Modular ADC Core and Modular Dual ADC Core IP cores.

2.1. Intel MAX 10 ADC Hard IP Block

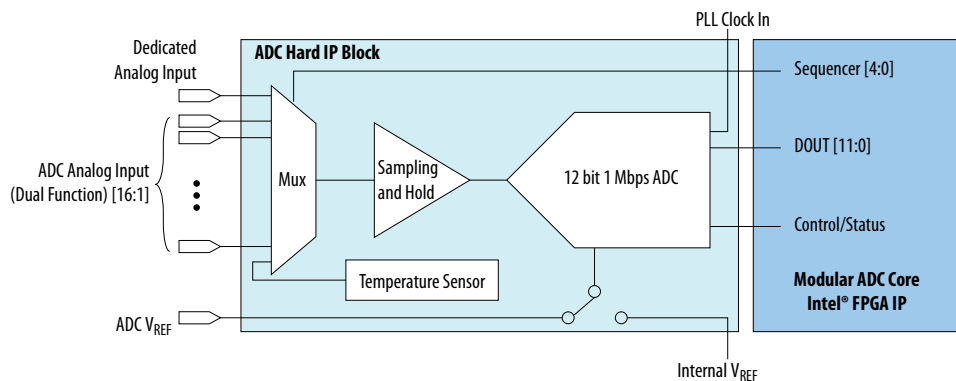
The Intel MAX 10 ADC is a successive approximation register (SAR) ADC that converts one analog sample in one clock cycle.

Each ADC block supports one dedicated analog input pin and up to 16 channels of dual function pins.

You can use the built-in temperature sensing diode (TSD) to perform on-chip temperature measurement.

Figure 3. ADC Hard IP Block in Intel MAX 10 Devices

Note: In dual ADC devices, the temperature sensor is available only in ADC1.



Related Information

[Sequencer Core](#) on page 27

Provides mode information about the sequencer conversion modes.

2.1.1. ADC Block Locations

The ADC blocks are located at the top left corner of the Intel MAX 10 device periphery.

Figure 4. ADC Block Location in Intel MAX 10 04 and 08 Devices

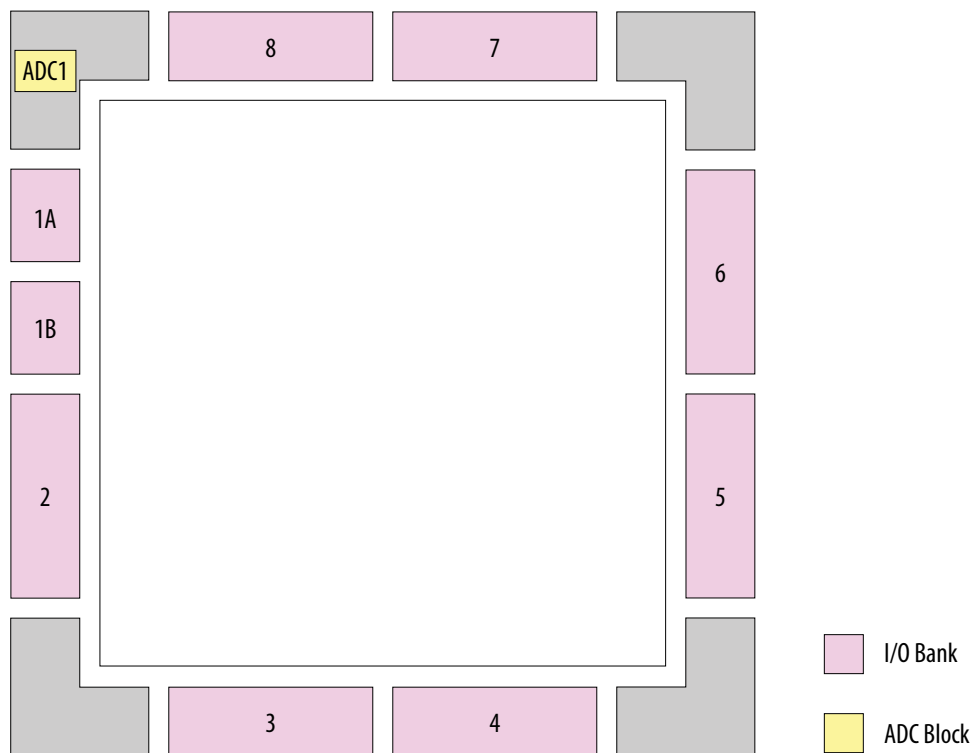


Figure 5. ADC Block Location in Intel MAX 10 16 Devices

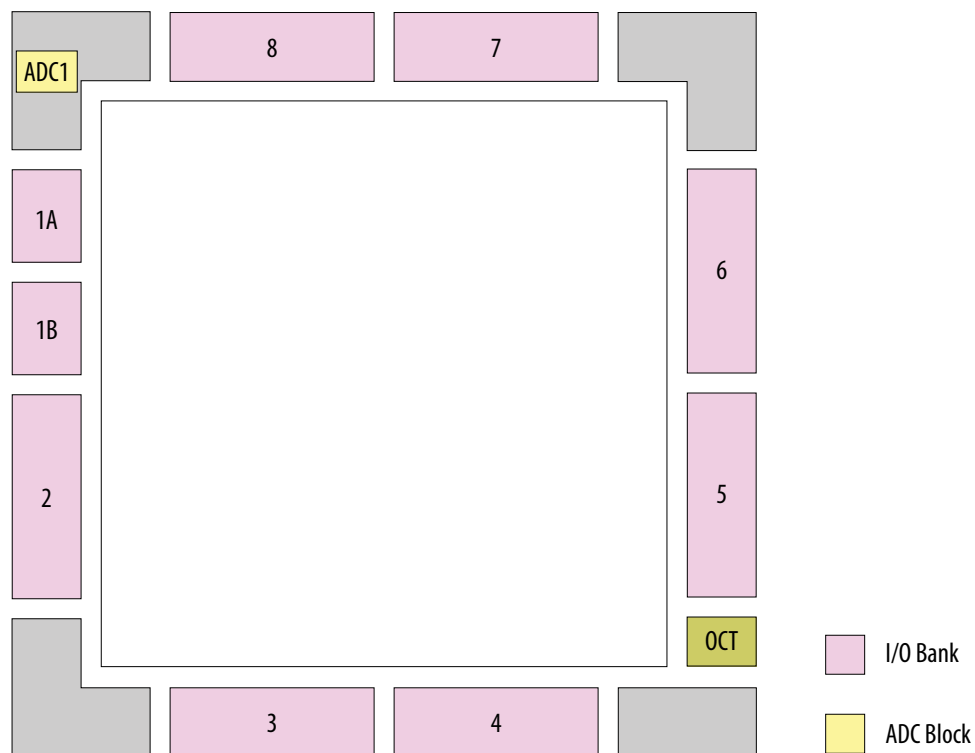
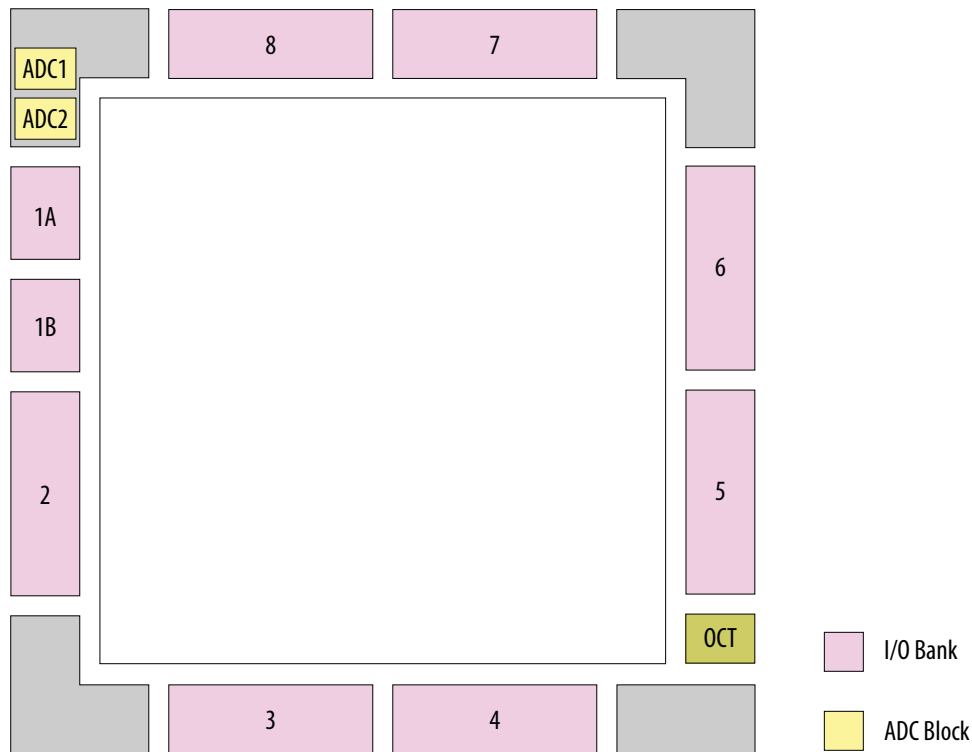


Figure 6. ADC Block Location in Intel MAX 10 25, 40, and 50 Devices

Package E144 of these devices have only one ADC block.



2.1.2. Single or Dual ADC Devices

Intel MAX 10 devices are available with single or dual ADC blocks.

For devices with one ADC block, you can use up to 17 ADC channels:

- These channels include one dedicated analog input and up to 16 dual function pins.
- You can use the dual function pins as GPIO pins when you do not use the ADC.

Note: Intel MAX 10 devices in the E144 package have only 8 dual function ADC pins.

For devices with two ADC blocks, you can use up to 18 ADC channels:

- For dual ADC devices, each ADC block can support one dedicated analog input pin and up to 8 dual function pins.
- If you use both ADC blocks in dual ADC devices, you can use up to two dedicated analog input pins and 16 dual function pins.
- For simultaneous measurement, you can use only dedicated analog input pins in both ADC blocks because the package routing of both dedicated analog pins are matched. For dual function pins, the routing latency between two ADC blocks may cause data mismatch in simultaneous measurement.
- For simultaneous measurement, use the Modular Dual ADC Core IP core.

To choose the correct device, refer to the Intel MAX 10 device overview.

Related Information

- [Intel MAX 10 FPGA Device Overview](#)
- [ADC Channel Counts in Intel MAX 10 Devices](#) on page 6

2.1.3. ADC Analog Input Pins

The analog input pins support single-ended and unipolar measurements.

The ADC block in Intel MAX 10 devices contains two types of ADC analog input pins:

- Dedicated ADC analog input pin—pins with dedicated routing that ensures both dedicated analog input pins in a dual ADC device has the same trace length.
- Dual function ADC analog input pin—pins that share the pad with GPIO pins.

If you use bank 1A for ADC, you cannot use the bank for GPIO.

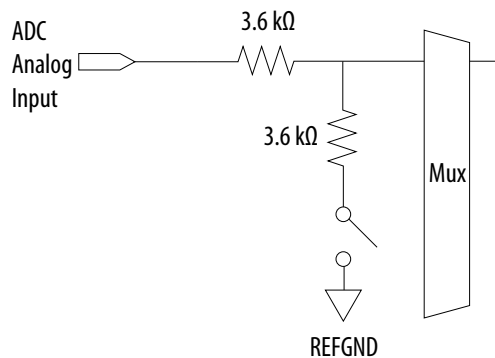
Each analog input pin in the ADC block is protected by electrostatic discharge (ESD) cell.

2.1.4. ADC Prescaler

The ADC block in Intel MAX 10 devices contains a prescaler function.

The prescaler function divides the analog input voltage by half. Using this function, you can measure analog input greater than 2.5 V. In prescaler mode, the analog input can handle up to 3 V input for the dual supply Intel MAX 10 devices and 3.6 V for the single supply Intel MAX 10 devices.

Figure 7. ADC Prescaler Block Diagram



The prescaler feature is available on these channels in each ADC block:

- Single ADC device—channels 8 and 16 (if available)
- Dual ADC device:
 - Using Modular ADC Core IP core—channel 8 of first or second ADC
 - Using Modular Dual ADC Core IP core—channel 8 of ADC1 and channel 17 of ADC2

2.1.5. ADC Clock Sources

The ADC block uses the device PLL as the clock source. The ADC clock path is a dedicated clock path. You cannot change this clock path.

Depending on the device package, the Intel MAX 10 devices support one or two PLLs—PLL1 only, or PLL1 and PLL3.

For devices that support two PLLs, you can select which PLL to connect to the ADC. You can configure the ADC blocks with one of the following schemes:

- Both ADC blocks share the same clock source for synchronization.
- Both ADC blocks use different PLLs for redundancy.

If each ADC block in your design uses its own PLL, the Intel Quartus® Prime Fitter automatically selects the clock source scheme based on the PLL clock input source:

- If each PLL that clocks its respective ADC block uses different PLL input clock source, the Intel Quartus Prime Fitter follows your design (two PLLs).
- If both PLLs that clock their respective ADC block uses the same PLL input clock source, the Intel Quartus Prime Fitter merges both PLLs as one.

In dual ADC mode, both ADC instance must share the same ADC clock setting.

Related Information

[PLL Locations, Intel MAX 10 Clocking and PLL User Guide](#)

Provides more information about the availability of PLL3 in different Intel MAX 10 devices and packages.

2.1.6. ADC Voltage Reference

Each ADC block in Intel MAX 10 devices can independently use an internal or external voltage reference. In dual ADC devices, you can assign an internal voltage reference to one ADC block and an external voltage reference to the other ADC block.

There is only one external V_{REF} pin in each Intel MAX 10 device. Therefore, if you want to assign external voltage reference for both ADC blocks in dual ADC devices, share the same external voltage reference for both ADC blocks.

Intel recommends that you use a clean external voltage reference with a maximum resistance of 100 Ω for the ADC blocks. If the ADC block uses an internal voltage reference, the ADC block is tied to its analog voltage and the conversion result is ratiometric.

2.1.7. ADC Temperature Sensing Diode

The ADC block in Intel MAX 10 devices has built-in TSD. You can use the built-in TSD to monitor the internal temperature of the Intel MAX 10 device.

- While using the temperature sensing mode, the ADC sampling rate is 50 kilosamples per second during temperature measurement.
- After the temperature measurement completes, if the next conversion in the sequence is normal sampling mode, the Modular ADC Core IP core automatically switches the ADC back to normal sampling mode. The maximum cumulative sampling rate in normal sampling mode is 1 MSPS.
- When the ADC switches from normal sensing mode to temperature sensing mode, and vice versa, calibration is run automatically for the changed clock frequency. The calibration incurs at least six clock calibration cycles from the new sampling rate.
- The ADC TSD measurement uses a 64-samples running average method. For example:
 - The first measured temperature value is the average of samples 1 to 64.
 - The second measured temperature value is the average of samples 2 to 65.
 - The third measured temperature value is the average of samples 3 to 66.
 - The subsequent temperature measurements follow the same method.

For dual ADC devices, the temperature sensor is available in ADC1 only.

2.1.7.1. Temperature Measurement Code Conversion

Use the temperature measurement code conversion table to convert the values measured by the ADC TSD to actual temperature.

Table 4. Temperature Code Conversion Table

Temp (C)	Code	Temp (C)	Code	Temp (C)	Code	Temp (C)	Code	Temp (C)	Code
-40	3798	-6	3738	28	3670	62	3593	96	3510
-39	3796	-5	3736	29	3667	63	3592	97	3507
-38	3795	-4	3733	30	3666	64	3591	98	3504
-37	3793	-3	3732	31	3664	65	3590	99	3501
-36	3792	-2	3731	32	3662	66	3589	100	3500
-35	3790	-1	3730	33	3660	67	3585	101	3498
-34	3788	0	3727	34	3658	68	3582	102	3496
-33	3786	1	3725	35	3656	69	3579	103	3494
-32	3785	2	3721	36	3654	70	3576	104	3492
-31	3782	3	3720	37	3651	71	3573	105	3490
-30	3781	4	3719	38	3648	72	3570	106	3489
-29	3780	5	3717	39	3645	73	3567	107	3486
-28	3779	6	3715	40	3643	74	3564	108	3483
-27	3777	7	3713	41	3642	75	3561	109	3480
-26	3775	8	3711	42	3641	76	3558	110	3477
-25	3773	9	3709	43	3640	77	3555	111	3474
-24	3771	10	3707	44	3638	78	3552	112	3471
-23	3770	11	3704	45	3636	79	3551	113	3468
-22	3768	12	3703	46	3634	80	3550	114	3465
-21	3766	13	3702	47	3632	81	3549	115	3461
-20	3765	14	3700	48	3630	82	3548	116	3460
-19	3764	15	3699	49	3628	83	3547	117	3459
-18	3762	16	3698	50	3625	84	3546	118	3456
-17	3759	17	3697	51	3622	85	3542	119	3451
-16	3756	18	3696	52	3619	86	3538	120	3450
-15	3754	19	3695	53	3616	87	3534	121	3449
-14	3752	20	3688	54	3613	88	3530	122	3445
-13	3751	21	3684	55	3610	89	3526	123	3440
-12	3750	22	3682	56	3607	90	3525	124	3432
-11	3748	23	3680	57	3604	91	3524	125	3431
-10	3746	24	3678	58	3601	92	3522	—	—

continued...

Temp (C)	Code	Temp (C)	Code	Temp (C)	Code	Temp (C)	Code	Temp (C)	Code
-9	3744	25	3677	59	3598	93	3519	—	—
-8	3742	26	3676	60	3595	94	3516	—	—
-7	3740	27	3673	61	3594	95	3513	—	—

2.1.7.2. Temperature Measurement Sampling Rate

In temperature sensing mode, the maximum ADC sampling rate is 50 kilosamples per second (50 KHz frequency). The sampling rate of the TSD depends on the **ADC Sample Rate** parameter you selected in the Modular ADC Core or Modular Dual ADC Core IP core.

Table 5. Intel MAX 10 TSD Sampling Rate Based on Selected ADC Sample Rate Parameter

ADC Sample Rate Selected	Actual TSD Sampling Rate
1 MHz	50 KHz
500 KHz	50 KHz
250 KHz	25 KHz
200 KHz	20 KHz
125 KHz	12.5 KHz
100 KHz	10 KHz
50 KHz	5 KHz
25 KHz	2.5 KHz

2.1.8. ADC Sequencer

The Modular ADC Core and Modular Dual ADC Core IP cores implement the sequencer. Use the Modular ADC Core or Modular Dual ADC Core parameter editor to define the ADC channel acquisition sequence and generate the HDL code.

The sequencer can support sequences of up to 64 ADC measurement slots. While configuring the Modular ADC Core or Modular Dual ADC Core IP core, you can select which channel, including the TSD channel, to sample in each sequencer slot. During runtime, you cannot change the channel sequence but you can configure the sequencer conversion mode using the Nios® II HAL driver API.

You can specify up to 64 slots and assign the channel for each slot. You can repeat the same channel number several times if required.

Related Information

Guidelines: [ADC Sequencer in Modular Dual ADC Core IP Core](#) on page 19

2.1.8.1. Guidelines: ADC Sequencer in Modular Dual ADC Core IP Core

Follow these sequencer guidelines if you use dual ADC blocks with the Modular Dual ADC Core IP core.

- The conversion sequence length of both ADC blocks must be the same.
- You can configure independent patterns for the conversion sequence of each ADC blocks.
- You can set a sequencer slot in ADC2 to NULL. If you set the slot to NULL, ADC2 performs a dummy conversion for the slot with output of "0". The NULL option is available only for ADC2.
- The temperature sensor is available only in ADC1. If you configure a sequencer slot in ADC1 for temperature sensing, you must set the same sequencer slot number in ADC2 to NULL.

Related Information

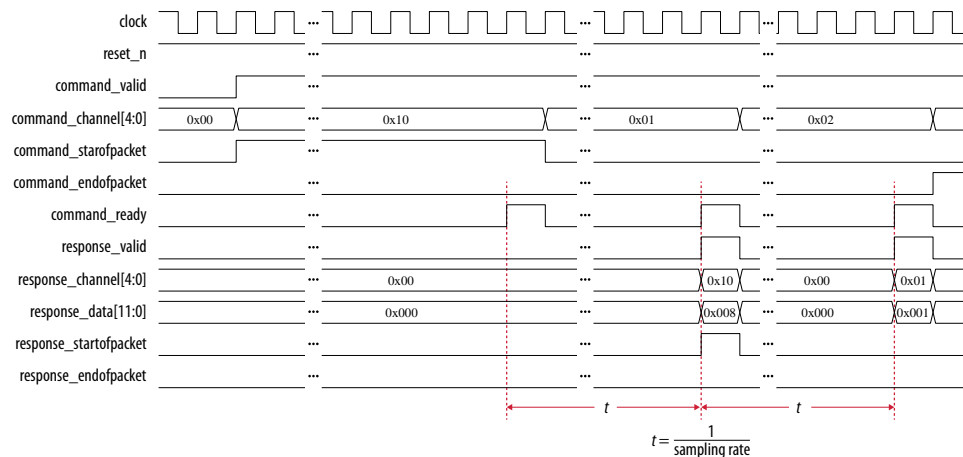
ADC Sequencer on page 19

2.1.9. ADC Timing

Figure 8. Intel MAX 10 ADC Control Core Timing Diagram

The following figure shows the timing diagram for the command and response interface of the Modular ADC Core control core. The timing diagram shows the latency of the first valid response data, and the latency between the first acknowledgment of the first command request and the back-to-back response data. The diagram shows an example where:

- The conversion sequence is from channel 16 to channel 1 to channel 2
- The response data for channel 16 is 8
- The response data for channel 1 is 1



2.2. Modular ADC Core and Modular Dual ADC Core IP Cores

You can use the Modular ADC Core and Modular Dual ADC Core IP cores to generate soft IP controllers for the ADC hard IP blocks in Intel MAX 10 devices.

There are two ADC IP cores:

- Modular ADC Core IP core—each instance can control one ADC hard IP block. In a dual ADC device, you can instantiate one Modular ADC Core IP core instance for each ADC block. However, both instances are asynchronous to each other.
- Modular Dual ADC Core IP core—you can control both ADC hard IP block with a single IP instance.
 - For the analog input pins (ANAIN1 and ANAIN2) in both ADC hard IP blocks, the measurement is synchronous.
 - For the dual function input pins, there are some measurement timing differences caused by the routing latency.

You can perform the following functions with the Modular ADC Core or Modular Dual ADC Core IP core parameter editor:

- Configure the ADC clock, sampling rate, and reference voltage.
- Select which analog input channels that the ADC block samples.
- Configure the threshold value to trigger a threshold violation notification.
- Set up a conversion sequence to determine which channel requires more frequent attention.

Related Information

- [Modular ADC Core Intel FPGA IP and Modular Dual ADC Core Intel FPGA IP References](#) on page 46
- [Introduction to Intel FPGA IP Cores](#)
Provides general information about all Intel FPGA IP cores, including parameterizing, generating, upgrading, and simulating IP cores.
- [Creating Version-Independent IP and Qsys Simulation Scripts](#)
Create simulation scripts that do not require manual updates for software or IP version upgrades.
- [Project Management Best Practices](#)
Guidelines for efficient management and portability of your project and IP files.

2.2.1. Modular ADC Core IP Core Configuration Variants

The Modular ADC Core IP core provides four configuration variants that target different ADC use cases. These configuration variants support usages from basic system monitoring to high performance ADC data streaming.

[Configuration 1: Standard Sequencer with Avalon-MM Sample Storage](#) on page 22

[Configuration 2: Standard Sequencer with Avalon-MM Sample Storage and Threshold Violation Detection](#) on page 22

[Configuration 3: Standard Sequencer with External Sample Storage](#) on page 23

[Configuration 4: ADC Control Core Only](#) on page 24

Related Information

[Modular ADC Core Intel FPGA IP and Modular Dual ADC Core Intel FPGA IP References](#) on page 46

2.2.1.1. Configuration 1: Standard Sequencer with Avalon-MM Sample Storage

In this configuration variant, you can use the standard sequencer micro core with internal on-chip RAM for storing ADC samples. This configuration is useful for basic system monitoring application.

In a system monitoring application, the ADC captures a block of samples data and stores them in the on-chip RAM. The host processor retrieves the data before triggering another block of ADC data sample request. The speed of the host processor in servicing the interrupt determines the interval between each block sample request.

Figure 9. Standard Sequencer with Avalon-MM Sample Storage (Modular ADC Core IP Core)

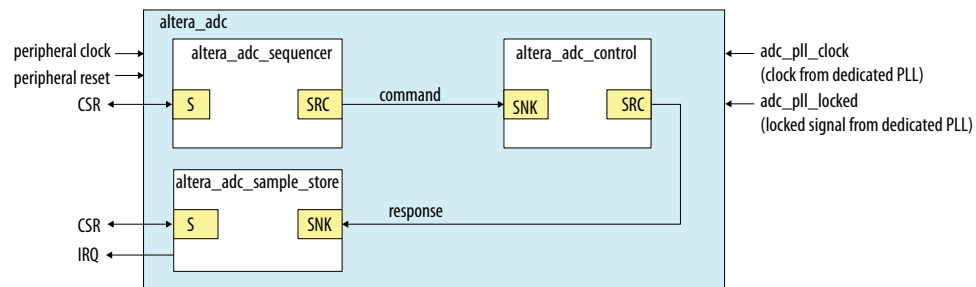
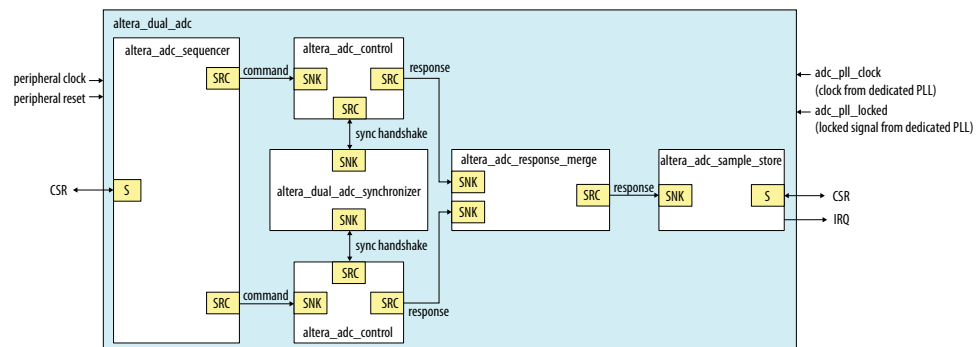


Figure 10. Standard Sequencer with Avalon-MM Sample Storage (Modular Dual ADC Core IP Core)



Related Information

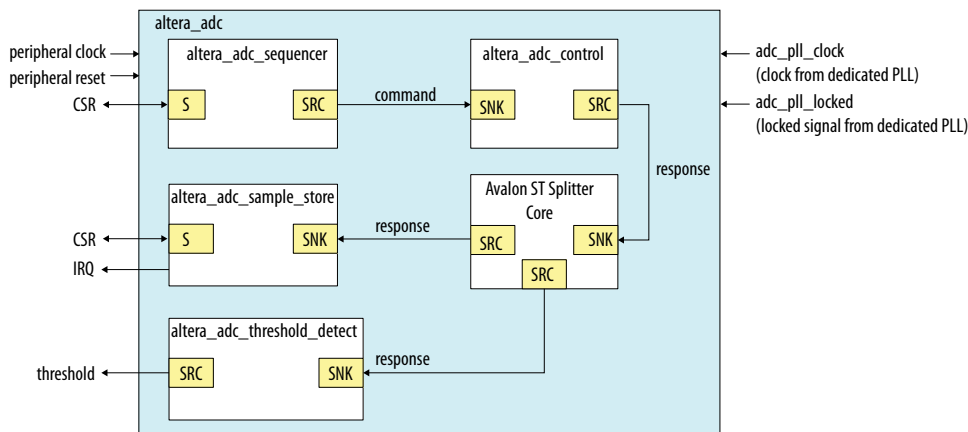
- [Customizing and Generating Modular ADC Core IP Core](#) on page 40
- [Completing ADC Design](#) on page 45

2.2.1.2. Configuration 2: Standard Sequencer with Avalon-MM Sample Storage and Threshold Violation Detection

In this configuration variant, you can use the standard sequencer micro core with internal on-chip RAM for storing ADC samples with the additional capability of detecting threshold violation. This configuration is useful for system monitoring application where you want to know whether the ADC samples value fall outside the maximum or minimum threshold value.

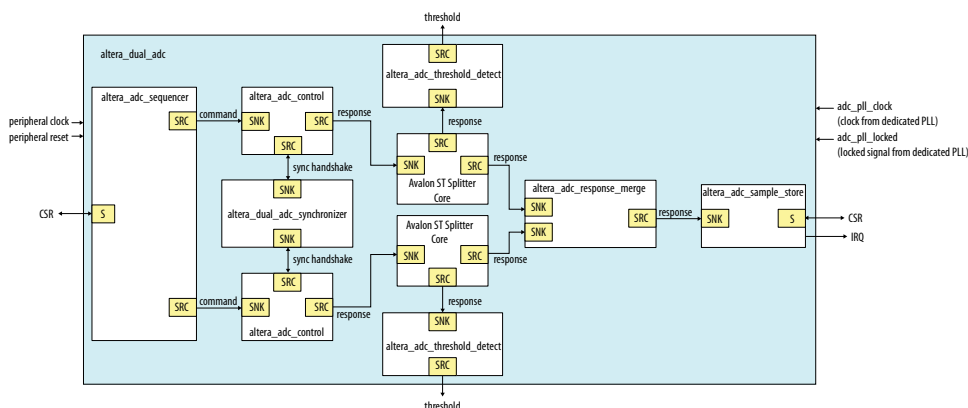
When the threshold value is violated, the Modular ADC Core or Modular Dual ADC Core IP core notifies the discrete logic component. The discrete component then triggers system recovery action. For example, the system can increase the fan speed in a temperature control system.

Figure 11. Standard Sequencer with Avalon-MM Sample Storage and Threshold Violation Detection (Modular ADC Core IP Core)



In dual ADC mode, you can configure the threshold detection of each ADC instance independently of each other. This capability is available because each ADC instance measures different analog metrics.

Figure 12. Standard Sequencer with Avalon-MM Sample Storage and Threshold Violation Detection (Modular Dual ADC Core IP Core)



Related Information

- [Customizing and Generating Modular ADC Core IP Core](#) on page 40
- [Completing ADC Design](#) on page 45

2.2.1.3. Configuration 3: Standard Sequencer with External Sample Storage

In this configuration variant, you can use the standard sequencer micro core and store the ADC samples in external storage.

You need to design your own logic to interface with the external storage.

Figure 13. Standard Sequencer with External Sample Storage (Modular ADC Core IP Core)

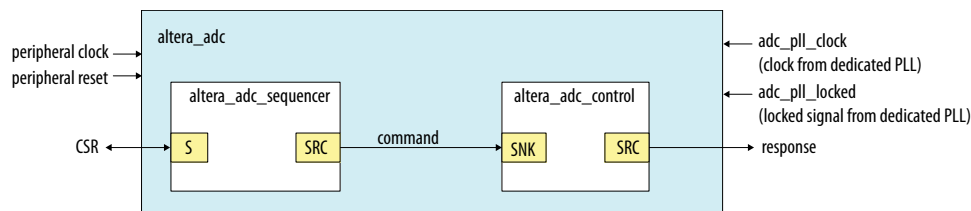
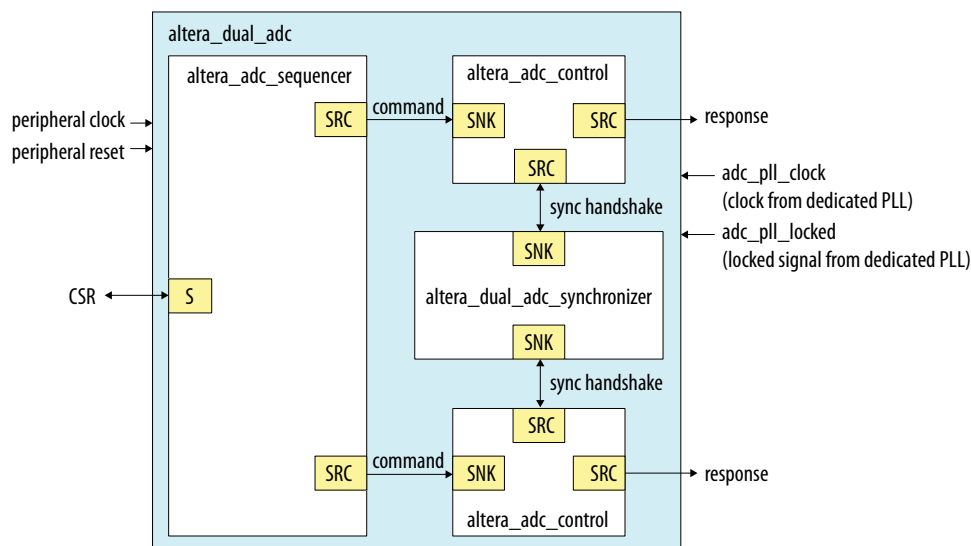


Figure 14. Standard Sequencer with External Sample Storage (Modular Dual ADC Core IP Core)



Related Information

- [Customizing and Generating Modular ADC Core IP Core](#) on page 40
- [Completing ADC Design](#) on page 45

2.2.1.4. Configuration 4: ADC Control Core Only

In this configuration variant, the Modular ADC Core generates only the ADC control core. You have full flexibility to design your own application-specific sequencer and use your own way to manage the ADC samples.

Figure 15. ADC Control Core Only (Modular ADC Core IP Core)

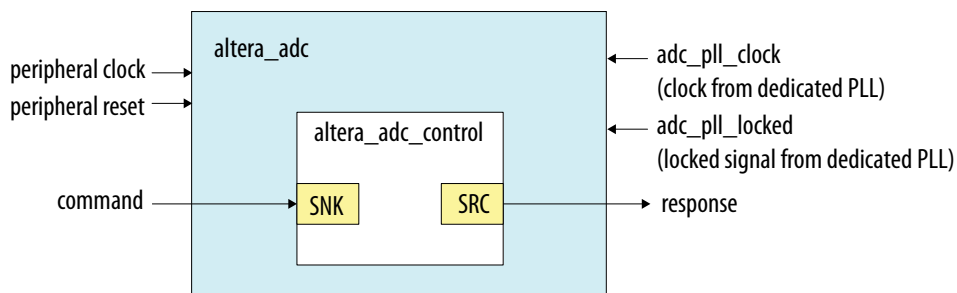
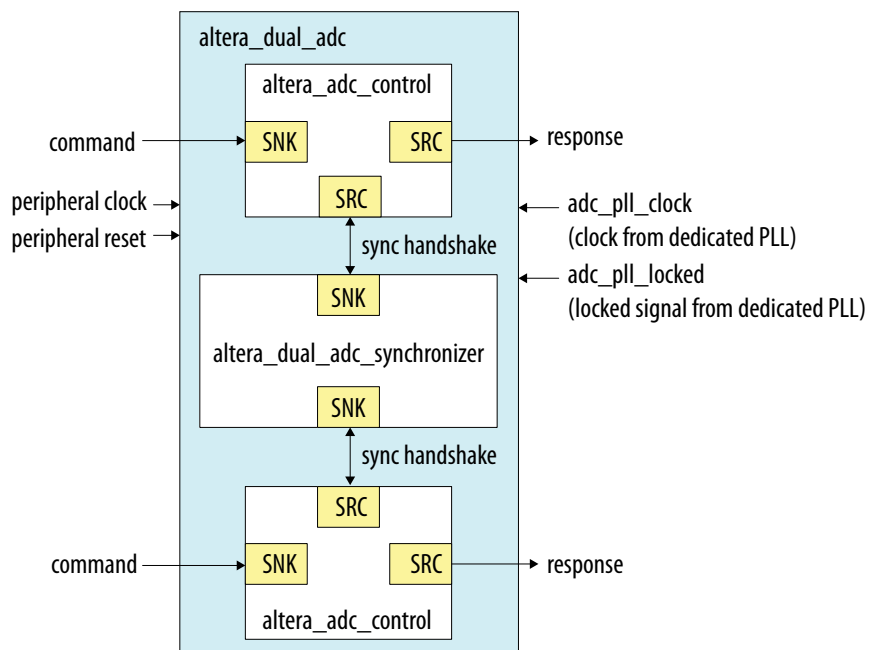


Figure 16. ADC Control Core Only (Modular Dual ADC Core IP Core)



Related Information

- [Customizing and Generating Modular ADC Core IP Core](#) on page 40
- [Completing ADC Design](#) on page 45

2.2.2. Modular ADC Core and Modular Dual ADC Core IP Cores Architecture

The Modular ADC Core IP core consists of six micro cores.

Table 6. Modular ADC Core Micro Cores

Micro Core	Description
ADC control	This core interacts with the ADC hard IP block. The ADC control core uses Avalon ST interface to receive commands from upstream cores, decodes, and drives the ADC hard IP block accordingly.
Sequencer	<p>This core contains command register and static conversion sequence data. The sequencer core issues commands for downstream cores to execute.</p> <ul style="list-style-type: none"> You can use the command register to configure the intended conversion mode. You can configure the length and content of the conversion sequence data only when generating the IP core. You can access the register of the sequencer core through the Avalon® memory-mapped slave interface. The command information to the downstream core goes through the Avalon ST interface.
Sample storage	<p>This core stores the ADC samples that are received through the Avalon ST interface.</p> <ul style="list-style-type: none"> The samples are stored in the on-chip RAM. You can retrieve the samples through the Avalon memory-mapped slave interface. With this core, you have the option to generate interrupt when the ADC receives a block of ADC samples (one full round of conversion sequence).
Response merge	<p>This core merges simultaneous responses from two ADC control cores into a single response packet to send to the sample storage core. This core is available only if you use the Modular Dual ADC Core IP core in the following configurations:</p> <ul style="list-style-type: none"> Standard Sequencer with Avalon-MM Sample Storage Standard Sequencer with Avalon-MM Sample Storage and Threshold Violation Detection
Dual ADC synchronizer core	This core performs synchronization handshakes between two ADC control cores. This core is available only if you use the Modular Dual ADC Core IP core.
Threshold detection	<ul style="list-style-type: none"> This core supports fault detection. The threshold detection core receives ADC samples through the Avalon ST interface and checks whether the samples value exceeds the maximum or falls below the minimum threshold value. The threshold detection core conveys threshold value violation information through the Avalon ST interface. You can configure which channel to enable for maximum and minimum threshold detection and the threshold values only during IP core generation.

2.2.2.1. ADC Control Core

The ADC control core drives the ADC hard IP according to the command it receives. The control core also maps the channels from the Modular ADC Core IP core to the channels in the ADC hard IP block.

The ADC control core of the Modular ADC Core IP core implements only the functions that are related to ADC hard IP block operations. For example:

- Power up
- Power down
- Analog to digital conversion on analog pins
- Analog to digital conversion on on-chip temperature sensor

The ADC control core has two clock domains:

- One clock domain for clocking the ADC control core soft logic
- Another clock domain for the ADC hard IP block

The ADC control core does not have run-time configurable options.

Figure 17. ADC Control Core High-Level Block Diagram

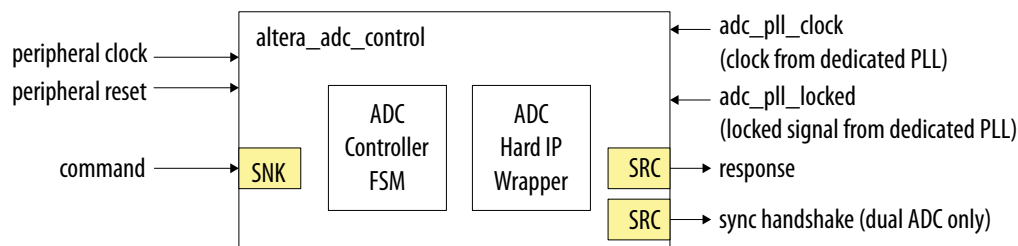


Table 7. ADC Control Core Backpressure Behavior

Interface	Backpressure Behavior
Command	The ADC control core asserts ready when it is ready to perform a sample conversion. The ADC control core only accepts one command at a time. The control core releases ready when it completes processing current command and prepares to perform the next command. Once the ADC control core asserts "cmd_ready=1" to acknowledge the current command, the Sequencer core provides the next valid request within two clock cycles. If the next valid request comes after two clock cycles, the ADC control core perform non-continuous sampling.
Response	The ADC control core does not support backpressure in the response interface. The fastest back-to-back assertion of valid request is 1 μ s.

2.2.2.2. Sequencer Core

The sequencer core controls the type of conversion sequence performed by the ADC hard IP. You can configure the conversion mode during run time using the sequencer core registers.

During Modular ADC Core or Modular Dual ADC Core IP core configuration, the sequencer core provides up to 64 configurable slots. You can define the sequence that the ADC channels are sampled by selecting the ADC channel for each sequencer slot.

The sequencer core has a single clock domain.

Figure 18. Sequencer Core High-Level Block Diagram

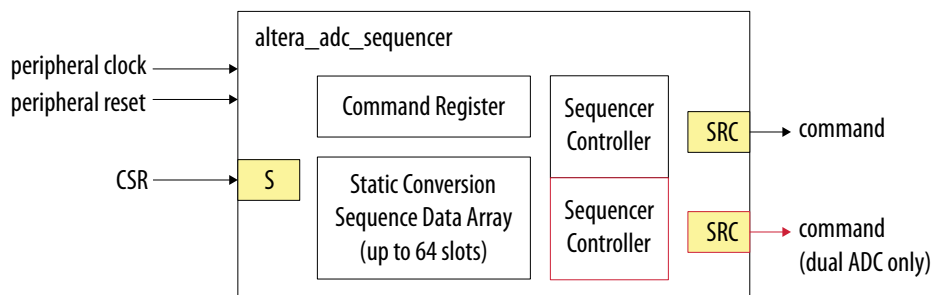


Table 8. Sequencer Core Conversion Modes

Conversion Mode	Description
Single cycle ADC conversion	<ul style="list-style-type: none"> In this mode, when the run bit is set, ADC conversion starts from the channel that you specify in the first slot. The conversion continues onwards with the channel that you specify in each sequencer slot. Once the conversion finishes with the last sequencer slot, the conversion cycle stops and the ADC hard IP block clears the run bit.
Continuous ADC conversion	<ul style="list-style-type: none"> In this mode, when the run bit is set, ADC conversion starts from the channel that you specify in the first slot. The conversion continues onwards with the channel that you specify in each sequencer slot. Once the conversion finishes with the last sequencer slot, the conversion begins again from the first slot of the sequence. To stop the continuous conversion, clear the run bit. The sequencer core continues the conversion sequence until it reaches the last slot and then stops the conversion cycle.

Related Information

- [Modular ADC Core Parameters Settings](#) on page 47
Lists the parameters available during Modular ADC Core IP configuration.
- [Modular Dual ADC Core Parameters Settings](#) on page 51
Lists the parameters available during Modular Dual ADC Core IP configuration.
- [Sequencer Core Registers](#) on page 59
Lists the registers for run-time control of the sequencer core.

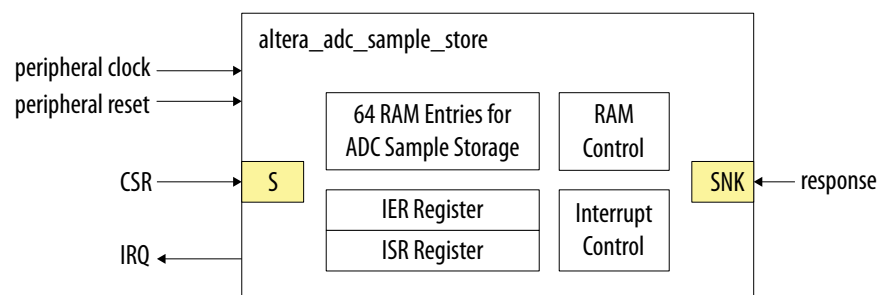
2.2.2.3. Sample Storage Core

The sample storage core stores the ADC sampling data in the on-chip RAM. The sample storage core stores the ADC samples data based on conversion sequence slots instead of ADC channels.

For example, if you sample a sequence of CH1, CH2, CH1, CH3, CH1, and then CH4, the ADC sample storage core stores the channel sample data in the same RAM entry sequence. This means that CH1 sample data is in the first, third, and fifth RAM entries; one for each sequence slot.

The sample storage core asserts IRQ when it completes receipt of a sample block. You can disable the IRQ assertion during run time using the interrupt enable register (IER) of the sample storage core. If you disable IRQ assertion, you must create polling methods in your design to determine the complete receipt of a sample block.

The sample storage core has a single clock domain.

Figure 19. Sample Storage Core High-Level Block Diagram


Related Information

[Sample Storage Core Registers](#) on page 60

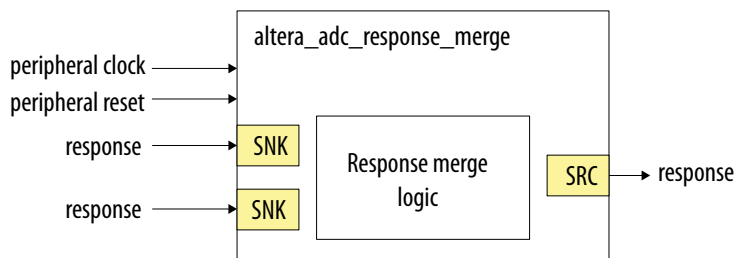
2.2.2.4. Response Merge Core

The response merge core merges simultaneous responses from two ADC control cores in the Modular Dual ADC Core IP core.

The Modular Dual ADC Core IP core uses the response merge core if you use the following configurations:

- Standard Sequencer with Avalon-MM Sample Storage
- Standard Sequencer with Avalon-MM Sample Storage and Threshold Violation Detection

Figure 20. Response Merge Core High-Level Block Diagram



2.2.2.5. Dual ADC Synchronizer Core

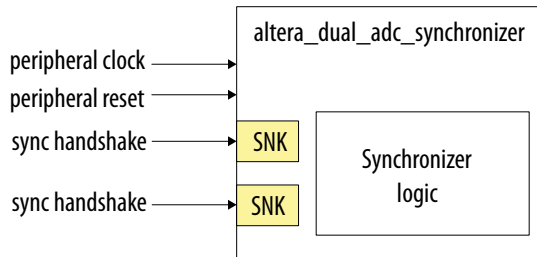
The dual ADC synchronizer core performs synchronization handshakes between two ADC control cores in the Modular Dual ADC Core IP core.

The peripheral clock domain is asynchronous to the ADC PLL clock domain in the ADC control core. Control event from the ADC hard IP block can appear at the peripheral clock domain at the same time, or by a difference of one peripheral clock between ADC1 and ADC2 control cores. Both ADC hard IP cores communicate with the dual ADC synchronizer core through the Avalon-ST interface.

For example, although a new command valid event from the sequencer arrives at both ADC control cores at the same peripheral clock cycle, the end of conversion signals arrive at one peripheral clock cycle difference between ADC1 and ADC2. To avoid the condition where ADC1 begins conversion earlier or later than ADC2, the ADC control core performs synchronization handshake using the dual ADC synchronizer core.

An ADC control core asserts a `sync_valid` signal when it detects an ADC PLL clock domain event. The dual ADC synchronizer core asserts the `sync_ready` signal after it receives `sync_valid` signals from both ADC control cores. After the `sync_ready` signal is asserted, both ADC control cores proceed to their next internal state.

Figure 21. Dual ADC Synchronizer Core High-Level Block Diagram



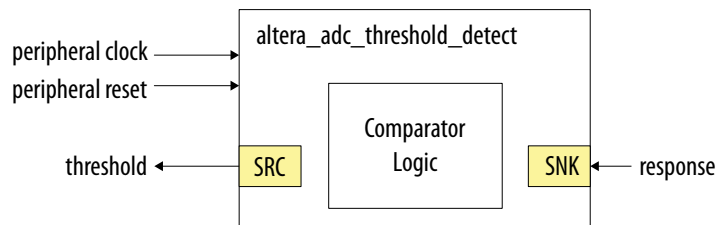
2.2.2.6. Threshold Detection Core

The threshold detection core compares the sample value that the ADC block receives to the threshold value that you define during Modular ADC Core IP core configuration. This core does not have run-time configurable options.

If the ADC sample value is beyond the maximum or minimum threshold limit, the threshold detection core issues a violation notification through the Avalon-ST interface.

The threshold detection core has a single clock domain.

Figure 22. Threshold Detection Core High-Level Block Diagram



2.3. Intel FPGA ADC HAL Driver

The Intel FPGA ADC HAL driver supports the following features:

- Read ADC channel data.
- Enable maximum or minimum threshold and return a user callback when the interrupt is triggered.
- Command the control of the ADC (run, stop, and recalibrate).

Related Information

[ADC HAL Device Driver for Nios Processor](#) on page 61

2.4. ADC Toolkit for Testing ADC Performance

You can use the ADC Toolkit provided with the Intel Quartus Prime software to understand the performance of the analog signal chain as seen by the Intel MAX 10 ADC blocks.

The ADC Toolkit supports monitoring the ADC whether you use the Modular ADC Core or Modular Dual ADC Core IP core. However, the ADC Toolkit can only monitor one ADC block at a time. If you are using the Modular Dual ADC Core IP core, configure the **Debug Path** parameter in the IP core to select which ADC block you want to hook up to the ADC Toolkit.

Related Information

ADC Toolkit

Provides more information about the ADC Toolkit.

2.5. ADC Logic Simulation Output

By default, the ADC logic simulation outputs a fixed unique value for each ADC channel. However, you can enable an option to specify your own output values for each ADC channel other than the TSD.

The ADC simulation model for Intel MAX 10 devices supports the standard digital logic simulators that the Intel Quartus Prime software supports.

Related Information

Intel Quartus Prime Simulator Support

2.5.1. Fixed ADC Logic Simulation Output

By default, the **Enable user created expected output file** option in the Modular ADC Core or Modular Dual ADC Core IP core is disabled. The ADC simulation always output a fixed value for each ADC channel, including the analog and TSD channels. The values are different for single and dual ADC devices.

Table 9. Fixed Expected Output Data for Single ADC Device Simulation

Channel	Expected Output Data (Decimal Value)
CH0	0
CH1	1
CH2	2
CH3	3
CH4	4
CH5	5
CH6	6
CH7	7
CH8	8
CH9	9
CH10	10
CH11	11
CH12	12
CH13	13
continued...	

Channel	Expected Output Data (Decimal Value)
CH14	14
CH15	15
CH16	16
TSD	3615

Table 10. Fixed Expected Output Data for Dual ADC Device Simulation

Channel	Expected Output Data (Decimal Value)	
	ADC1	ADC2
CH0	10	20
CH1	11	21
CH2	12	22
CH3	13	23
CH4	14	24
CH5	15	25
CH6	16	26
CH7	17	27
CH8	18	28
TSD	3615	— (No TSD in ADC2)

2.5.2. User-Specified ADC Logic Simulation Output

You can configure the Modular ADC Core or Modular Dual ADC Core IP core to output user-specified values in the logic simulation for each ADC channel except the TSD channel.

If you enable this feature, you must provide a simulation stimulus input file for each ADC channel that you enable. The logic simulation reads the input file for each channel and outputs the value of the current sequence. Once the simulation reaches the end of the file, it repeats from the beginning of the sequence.

The stimulus input file is a plain text file that contains two columns of numbers:

- The first column of numbers is ignored by the simulation model. You can use any values that you want such as time or sequence. The actual data sequencing is based on the text rows.
- The second column contains the voltage values.

The ADC IP core automatically converts each voltage value to a 12-bit digital value based on the reference voltage you specify in the IP core parameter settings.

Figure 23. Simulation Output Example, One Channel Enabled

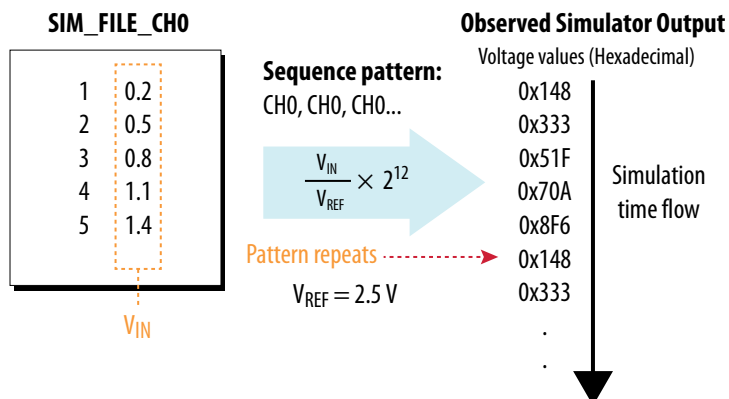
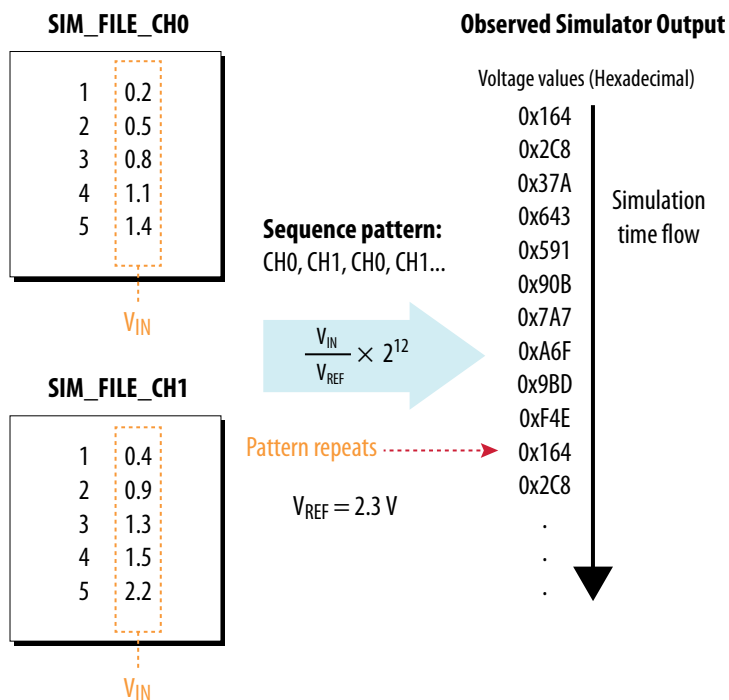


Figure 24. Simulation Output Example, Two Channels Enabled





3. Intel MAX 10 ADC Design Considerations

There are several considerations that require your attention to ensure the success of your designs. Unless noted otherwise, these design guidelines apply to all variants of this device family.

Related Information

[Intel MAX 10 Analog to Digital Converter Overview](#) on page 4

3.1. Guidelines: ADC Ground Plane Connection

For the ADC and V_{REF} pins, use the $REFGND$ pin as the analog ground plane connection.

Related Information

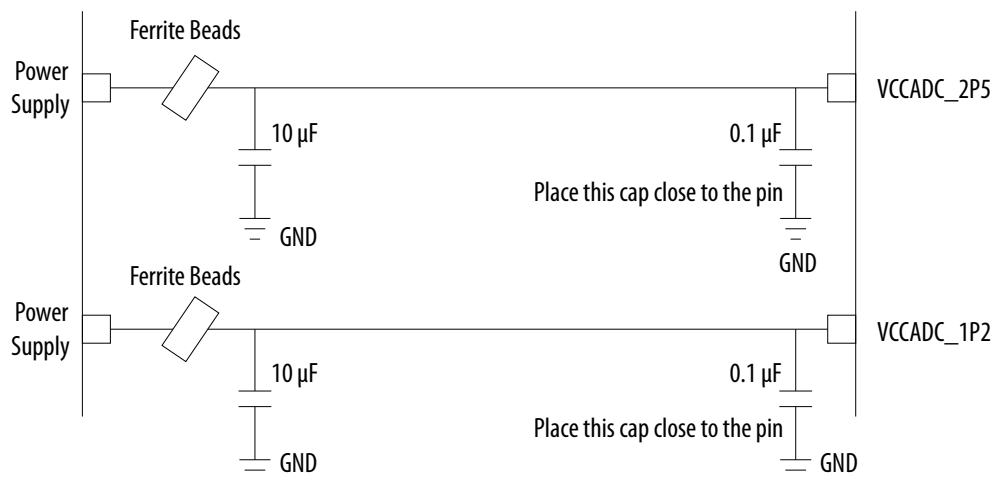
[Intel MAX 10 FPGA Device Family Pin Connection Guidelines](#)

Provides more information about pin connections including pin names and connection guidelines.

3.2. Guidelines: Board Design for Power Supply Pin and ADC Ground ($REFGND$)

The crosstalk requirement for analog to digital signal is -100 dB up to 2 GHz. There must be no parallel routing between power, ground, and surrounding general purpose I/O traces. If a power plane is not possible, route the power and ground traces as wide as possible.

- To reduce IR drop and switching noise, keep the impedance as low as possible for the ADC power and ground. The maximum DC resistance for power is 1.5 Ω .
- The power supplies connected to the ADC should have ferrite beads in series followed by a 10 μF capacitor to the ground. This setup ensures that no external noise goes into the device power supply pins.
- Decouple each of the device power supply pin with a 0.1 μF capacitor. Place the capacitor as close as possible to the device pin.

Figure 25. Recommended RC Filter for Power Traces

There is no impedance requirement for the REFGND. Intel recommends that you use the lowest impedance with the most minimum DC resistance possible. Typical resistance is less than 1 Ω .

Intel recommends that you set a REFGND plane that extends as close as possible to the corresponding decoupling capacitor and FPGA:

- If possible, define a complete REFGND plane in the layout.
- Otherwise, route the REFGND using a trace that is as wide as possible from the island to the FPGA pins and decoupling capacitor.
- The REFGND ground is the analog ground plane for the ADC V_{REF} and analog input.
- Connect REFGND ground to the system digital ground through ferrite beads. You can also evaluate the ferrite bead option by comparing the impedance with the frequency specifications.

3.3. Guidelines: Board Design for Analog Input

The crosstalk requirement for analog to digital signal is -100 dB up to 2 GHz. There must be no parallel routing between analog input signals and I/O traces, and between analog input signals and FPGA I/O signal traces.

- The ADC presents a switch capacitor load to the driving circuit. Therefore, the total RC constant, including package, trace, and parasitic driver must be less than 42.4 ns. This consideration is to ensure that the input signal is fully settled during the sampling phase.
- If you reduce the total sampling rate, you can calculate the required settling time as $0.45 \div F_S > 10.62 \times RC \text{ constant}$.
- To gain more total RC margin, Intel recommends that you make the driver source impedance as low as possible:
 - For non-prescaler channel—less than 1 k Ω
 - For prescaler channel—less than 11 Ω

Note: Not adhering to the source impedance recommendation may impact parameters such as total harmonic distortion (THD), signal-to-noise and distortion ratio (SINAD), differential non-linearity (DNL), and integral non-linearity (INL).

Trace Routing

- If possible, route the switching I/O traces on different layers.
- There is no specific requirement for input signal trace impedance. However, the DC resistance for the input trace must be as low as possible.
- Route the analog input signal traces as adjacent as possible to REFGND if there is no REFGND plane.
- Use REFGND as ground reference for the ADC input signal.
- For prescaler-enabled input signal, set the ground reference to REFGND. Performance degrades if the ground reference of prescaler-enabled input signal is set to common ground (GND).

Input Low Pass Filter Selection

- Intel recommends that you place a low pass filter to filter out high frequency noise being aliased back onto the input signal.
- Place the low pass filter as close as possible to the analog input signals.
- The cut off frequency depends on the analog input frequency. Intel recommends that the $F_{\text{cutoff}} @ -3\text{dB}$ is at least two times the input frequency.
- You can download the ADC input SPICE model for ADC front end board design simulation from the Intel website.

Table 11. RC Constant and Filter Value

This table is an example of the method to quantify the RC constant and identify the RC filter value.

$$\text{Total RC Constant} = (R_{\text{DRIVER}} + R_{\text{BOARD}} + R_{\text{PACKAGE}} + R_{\text{FILTER}}) \times (C_{\text{DRIVER}} + C_{\text{BOARD}} + C_{\text{PACKAGE}} + C_{\text{FILTER}} + C_{\text{PIN}})$$

Driver		Board		Package		Pin Capacitance (pF)	RC Filter		$F_{\text{cutoff}} @ -3\text{dB}$ (MHz)	Total RC Constant (ns)	Settling Time (ns)
R (Ω)	C (pF)	R (Ω)	C (pF)	R (Ω)	C (pF)		R (Ω)	C (pF)			
5	2	5	17	3	5	6	60	550	4.82	42.34	42.4
10	2	5	17	3	5	6	50	580	5.49	41.48	42.4

Figure 26. Passive Low Pass Filter Example

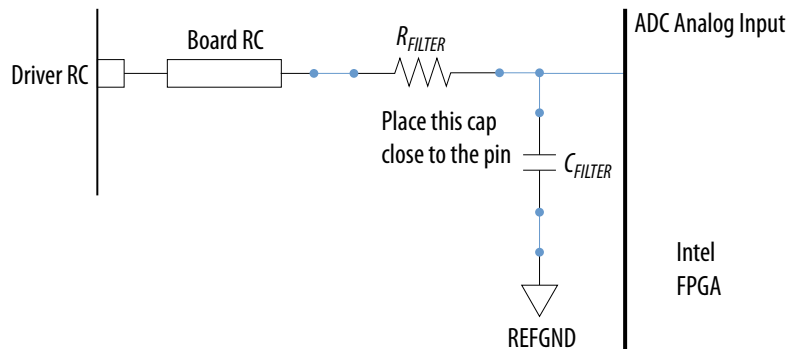
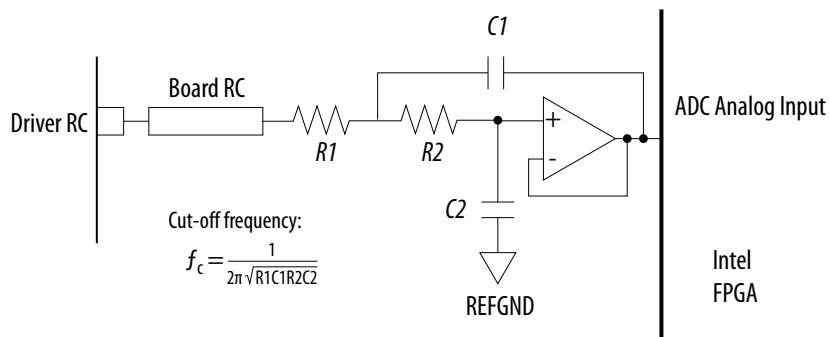


Figure 27. First Order Active Low Pass Filter Example

This figure is an example. You can design n th order active low pass filter.



Related Information

- [Parameters Settings for Generating Modular ADC Core or Modular Dual ADC Core IP Core](#) on page 42
- [Modular ADC Core Parameters Settings](#) on page 47
- [Modular Dual ADC Core Parameters Settings](#) on page 51
- [SPICE Models for Intel FPGAs](#)
Provides the MAX 10 ADC spice model download.

3.4. Guidelines: Board Design for ADC Reference Voltage Pin

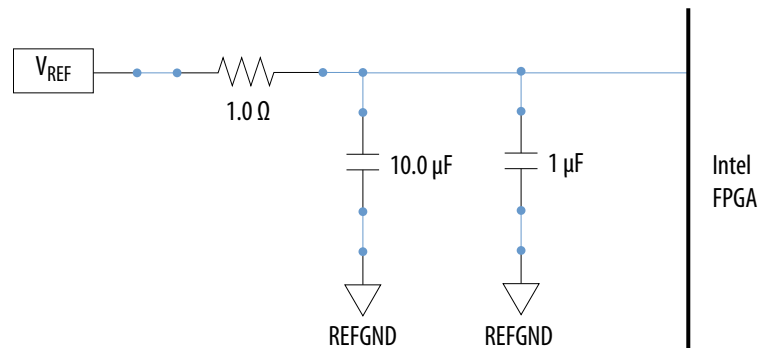
The crosstalk requirement for analog to digital signal is -100 dB up to 2 GHz. There is no parallel routing between analog input signals and I/O traces. Route the V_{REF} traces as adjacent as possible to REFGND.

If a REFGND plane is not possible, route the analog input signal as adjacent as possible to REFGND.

There is one ADC reference voltage pin in each Intel MAX 10 device. This pin uses REFGND as ground reference. Keep the trace resistance less than 0.8 Ω .

Figure 28. RC Filter Design Example for Reference Voltage Pin

Place the RC filter as close as possible to the analog input pin.

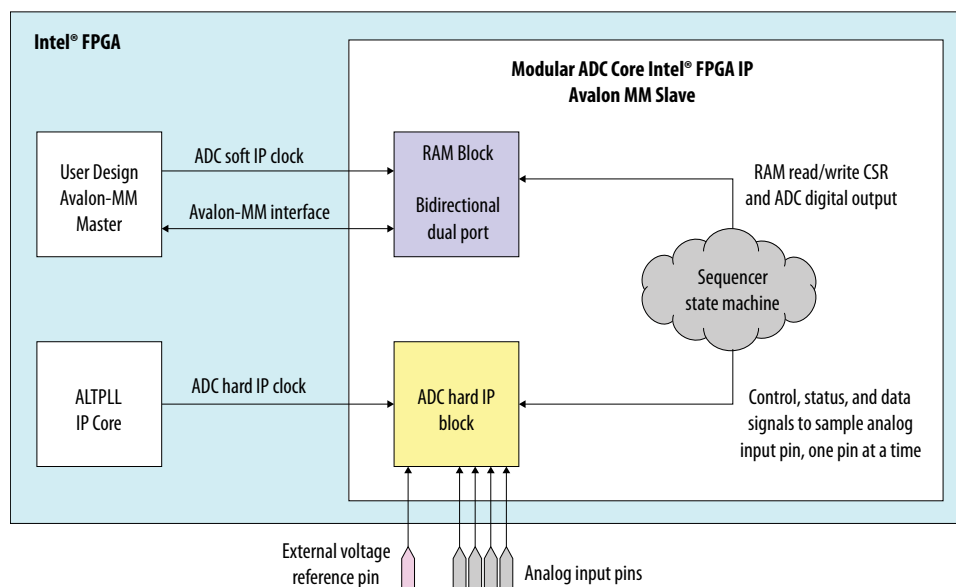


4. Intel MAX 10 ADC Implementation Guides

You can implement your ADC design in the Intel Quartus Prime software. The software contains tools for you to create and compile your design, and configure your device.

The Intel Quartus Prime software allows you to set up the parameters and generate your Modular ADC Core Intel FPGA IP or Modular Dual ADC Core Intel FPGA IP. To understand the ADC signal performance, you can use the Intel Quartus Prime ADC Toolkit. For more information about using the Intel Quartus Prime software and the ADC toolkit, refer to the related information.

Figure 29. High Level Block Diagram of the Intel MAX 10 ADC Solution



Related Information

- [Intel MAX 10 Analog to Digital Converter Overview](#) on page 4
- [Intel Quartus Prime Standard Edition Handbook, Volume 1: Design and Synthesis](#)
Provides more information about using IP cores in the Quartus II software.
- [Introduction to Intel FPGA IP Cores](#)
Provides general information about all Intel FPGA IP cores, including parameterizing, generating, upgrading, and simulating IP cores.
- [Creating Version-Independent IP and Qsys Simulation Scripts](#)
Create simulation scripts that do not require manual updates for software or IP version upgrades.
- [Project Management Best Practices](#)
Guidelines for efficient management and portability of your project and IP files.

- [ADC Toolkit](#)
Provides more information about the ADC Toolkit.
- [ADC Toolkit for Testing ADC Performance](#) on page 30

4.1. Creating Intel MAX 10 ADC Design

To create your ADC design, you must customize and generate the ALTPLL and Modular ADC Core IP cores.

The ALTPLL IP core provides the clock for the Modular ADC Core IP core.

1. Customize and generate the ALTPLL IP core.
2. Customize and generate the Modular ADC Core IP core.
3. Connect the ALTPLL IP core to the Modular ADC Core IP core.
4. Create ADC Avalon slave interface to start the ADC.

Related Information

- [Customizing and Generating Modular ADC Core IP Core](#) on page 40
- [Parameters Settings for Generating Modular ADC Core or Modular Dual ADC Core IP Core](#) on page 42
- [Parameters Settings for Generating ALTPLL IP Core](#) on page 41
- [Completing ADC Design](#) on page 45
- [Intel MAX 10 Getting Started](#)
- [Intel MAX 10 Online Training](#)
- [Intel MAX 10 How-to Videos](#)
- [How to Create ADC Design in Intel MAX 10 Device Using Platform Designer \(Standard\) Tool](#)
Provides video instruction that demonstrates how to create the ADC design in Intel MAX 10 devices using the Qsys system integration tool within the Intel Quartus Prime software and how to use the ADC toolkit to view the measured analog signal.
- [How to Create Simultaneous Measurement with Intel MAX 10 ADC, Part 1](#)
Provides the first part of video instruction series that explains the differences between the Intel MAX 10 Modular ADC Core and Modular Dual ADC Core IP cores. The video also demonstrates how to create a simple simultaneous ADC measurement and how to place signal taps to measure the digital code output for analog signal.
- [How to Create Simultaneous Measurement with Intel MAX 10 ADC, Part 2](#)
Provides the second part of video instruction series that explains the differences between the Intel MAX 10 Modular ADC Core and Modular Dual ADC Core IP cores. The video also demonstrates how to create a simple simultaneous ADC measurement and how to place signal taps to measure the digital code output for analog signal.

4.2. Customizing and Generating Modular ADC Core IP Core

Intel recommends that you use the Modular ADC Core IP core with a Nios II processor, which supports the ADC HAL driver.

1. Create a new project in the Intel Quartus Prime software.
While creating the project, select a device that has one or two ADC blocks.
2. In the Intel Quartus Prime software, select **Tools > Platform Designer (Standard)**.
3. In the **Platform Designer (Standard)** window, select **File > New System**.
A clock source block is automatically added under the **System Contents** tab.
4. In the **System Contents** tab, double click the clock name.
5. In the **Parameters** tab for the clock source, set the **Clock frequency**.
6. In the **IP Catalog** tab in the **Platform Designer (Standard)** window, double click **Processors and Peripherals > Peripherals > Modular ADC Core**.
The Modular ADC Core appears in the **System Contents** tab and the Modular ADC Core parameter editor opens.
7. In the Modular ADC Core parameter editor, specify the parameter settings and channel sampling sequence for your application.
8. In the **System Contents** tab in the **Platform Designer (Standard)** window, double click the **Export** column of the `adc_pll_clock` and `adc_pll_locked` interfaces to export them.
9. Connect the `clock`, `reset_sink`, `sample_store_csr`, and `sample_store_irq` signals. Optionally, you can use the Nios II Processor, On-Chip Memory, and JTAG UART IP cores to form a working ADC system that uses the Intel FPGA ADC HAL drivers.
10. In the **Platform Designer (Standard)** window, select **File > Save**.

You can copy an example HDL code to declare an instance of your ADC system. In the **Platform Designer (Standard)** window, select **Generate > HDL Example**.

Related Information

- [Creating Intel MAX 10 ADC Design](#) on page 40
- [Parameters Settings for Generating ALTPLL IP Core](#) on page 41
- [Parameters Settings for Generating Modular ADC Core or Modular Dual ADC Core IP Core](#) on page 42
- [Configuration 1: Standard Sequencer with Avalon-MM Sample Storage](#) on page 22
- [Configuration 2: Standard Sequencer with Avalon-MM Sample Storage and Threshold Violation Detection](#) on page 22
- [Configuration 3: Standard Sequencer with External Sample Storage](#) on page 23
- [Configuration 4: ADC Control Core Only](#) on page 24
- [ADC PLL Clock Interface of Modular ADC Core and Modular Dual ADC Core](#) on page 58
- [ADC PLL Locked Interface of Modular ADC Core and Modular Dual ADC Core](#) on page 59

4.3. Parameters Settings for Generating ALTPLL IP Core

Navigate through the ALTPLL IP core parameter editor and specify the settings required for your design. After you have specified all options as listed in the following table, you can generate the HDL files and the optional simulation files.

For more information about all ALTPLL parameters, refer to the related information.

Table 12. ALTPLL Parameters Settings

To generate the PLL for the ADC, use the following settings.

Tab	Parameter	Setting
Parameter Settings > General/Modes	What is the frequency of the inclk0 input?	Specify the input frequency to the PLL.
Parameter Settings > Inputs/Lock	Create an 'areset' input to asynchronously reset the PLL	Turn off this option.
	Create 'locked' output	Turn on this option. You need to connect this signal to the <code>adc_pll_locked</code> port of the Modular ADC Core or Modular Dual ADC Core IP core.
Output Clocks > clk c0	Use this clock	Turn on this option.
	Enter output clock frequency	Specify an output frequency of 2, 10, 20, 40, or 80 MHz. You can specify any of these frequencies. The ADC block runs at 1 MHz internally but it contains a clock divider that can further divide the clock by a factor of 2, 10, 20, 40, and 80. Use this same frequency value in your Modular ADC Core or Modular Dual ADC Core IP core. You need to connect this signal to the <code>adc_pll_clock</code> port of the Modular ADC Core or Modular Dual ADC Core IP core. Different ADC sampling rates support different clock frequencies. For a valid sampling rate and clock frequency combination, refer to the related information.

Related Information

- [Creating Intel MAX 10 ADC Design](#) on page 40
- [Customizing and Generating Modular ADC Core IP Core](#) on page 40
- [Completing ADC Design](#) on page 45
- [Intel MAX 10 Clock Networks and PLLs User Guide](#)
- [ADC PLL Clock Interface of Modular ADC Core and Modular Dual ADC Core](#) on page 58
- [ADC PLL Locked Interface of Modular ADC Core and Modular Dual ADC Core](#) on page 59
- [Valid ADC Sample Rate and Input Clock Combination](#) on page 54

4.4. Parameters Settings for Generating Modular ADC Core or Modular Dual ADC Core IP Core

Navigate through the Modular ADC Core IP core parameter editor and specify the settings required for your design. After you have specified all options as listed in the following tables, you can generate the HDL files and the optional simulation files.

Note: The Modular ADC Core and Modular Dual ADC Core IP cores support generating only Verilog* simulation scripts.

Intel recommends that you save the generated files in the design file directory (default setting).

For more information about each Modular ADC Core or Modular Dual ADC Core parameter, refer to the related information section.

Table 13. Parameter Settings in General Group

Parameter	Setting
Core Variant	There are four configuration variants of the Modular ADC Core IP core. Select the core variant that meets your requirement. For more information, refer to the related information.
Debug Path	Turn this on to enable the debug path for the selected core variant. You can use the ADC Toolkit to monitor the ADC performance.
Generate IP for which ADCs of this device?	For devices with two ADC blocks, select the ADC block for which you are generating the IP core. There are feature differences between the two ADC blocks. The temperature sensor is available only in the first ADC block. There are also different channel counts in both ADC blocks.
ADC Sample Rate	Select the predefined sampling rate for the ADC from 25 kHz to 1 MHz. A lower sampling rate allows you greater flexibility in designing your ADC front end driver circuit. For example, using a lower sampling rate gives you a wider settling time margin for your filter design. The sampling rate you select affects which ADC input clock frequencies are available. Refer to the related information for more details about the sampling rate and the required settling time.
ADC Input Clock	Select the same frequency that you set for the ALTPLL IP core that clocks the Modular ADC Core IP core. When configuring the ALTPLL IP core, specify a clock frequency that is supported by the ADC sampling rate. For more details, refer to the related information.
Reference Voltage Source	Select whether you want to use external or internal reference voltage. There is only one V_{REF} pin. For dual ADC blocks, you can use one external V_{REF} source for both ADC blocks, or external V_{REF} for one ADC block and internal V_{REF} for the other ADC block.
External Reference Voltage	If you use external V_{REF} source in your design, specify the V_{REF} level.
Enable user created expected output file	If you want to use your own stimulus input file to simulate the ADC output data, enable this function and specify the file for the specific ADC channel. For more information about user-specified ADC logic simulation output, refer to the related information.

Table 14. Parameters Settings in Channels Group

You can navigate through the tabs for all the available channels and turn on the channel you want to use. In each channel (and **TSD**) tab, you can specify the settings in this table.

Parameter	Setting
Use Channel 0 (Dedicated analog input pin - ANAIN)	This option is available in the CH0 tab. CH0 is the dedicated analog input channel. If you want to use the dedicated analog input, turn on this option.
User created expected output file	If you enabled the option to use your own stimulus input file to simulate the output data, click Browse and select the file for each enabled channel. This option is available in all channel tabs except the TSD tab.
Use Channel <i>N</i>	You can select which dual-function ADC channels to turn on or off. There are 16 channels (CH1 to CH16) for single ADC devices and 8 channels (CH1 to CH8) for each ADC block in dual ADC devices.
Use on-chip TSD	This option is available in the TSD tab. The TSD channel is the temperature sensing channel. Turn on this option if you want the IP core to read the built-in temperature sensor in the ADC block.
<i>continued...</i>	

Parameter	Setting
	The sampling rate of the ADC block reduces to 50 kHz when it reads the temperature measurement. After it completes the temperature reading, the ADC sampling rate returns to 1 MHz. For the Modular Dual ADC Core IP core, if you specify the TSD in a sequencer slot for ADC1, specify NULL in the same sequencer slot number for ADC2.
Enable Maximum threshold for Channel <i>N</i>	Turn on this option if you want to set a maximum threshold value for the channel.
Enter Maximum Threshold for Channel <i>N</i>	Enter the maximum threshold voltage for the channel. The IP core generates a threshold violation notification signal to indicate that the sampled data is over the threshold value that you specify.
Enable Maximum threshold for on-chip TSD (TSD tab)	Enter the maximum threshold temperature for the temperature sensor in Celsius. The IP core generates a threshold violation notification signal to indicate that the sampled temperature is over the temperature that you specify.
Enable Minimum threshold for Channel <i>N</i>	Turn on this option if you want to set a minimum threshold value for the channel.
Enter Minimum Threshold for Channel <i>N</i>	Enter the minimum threshold voltage for the channel. The IP core generates a threshold violation notification signal to indicate that the sampled data is below the threshold value that you specify.
Enter Minimum Threshold for on-chip TSD (TSD tab)	Enter the maximum threshold temperature for the temperature sensor in Celsius. The IP core generates a threshold violation notification signal to indicate that the sampled temperature is below the temperature that you specify.

Table 15. Parameters Settings in Sequencer Group

Parameter	Setting
Number of slot used	Select the number of channels to use for conversion. The parameter editor displays the number of slots available in the Conversion Sequence Channels based on your selection.
Slot <i>N</i>	For each available slot, select the channel to sample in the sequence. The available channels depend on the channels that you turned on in the Channels parameters group. If you turned on a channel but do not select the channel in any of the sequencer slots, the unselected channel is not measured during the ADC sampling sequence. The ADC block samples the measurements in the sequence you specify. After it reaches the last slot in the sequence, it repeats the sampling from the first slot.

Related Information

- [Creating Intel MAX 10 ADC Design](#) on page 40
- [Customizing and Generating Modular ADC Core IP Core](#) on page 40
- [Completing ADC Design](#) on page 45
- [Modular ADC Core Parameters Settings](#) on page 47
- [Modular Dual ADC Core Parameters Settings](#) on page 51
- [Modular ADC Core IP Core Channel Name to Intel MAX 10 Device Pin Name Mapping](#) on page 49
- [Modular Dual ADC Core IP Core Channel Name to Intel MAX 10 Device Pin Name Mapping](#) on page 53
- [Valid ADC Sample Rate and Input Clock Combination](#) on page 54

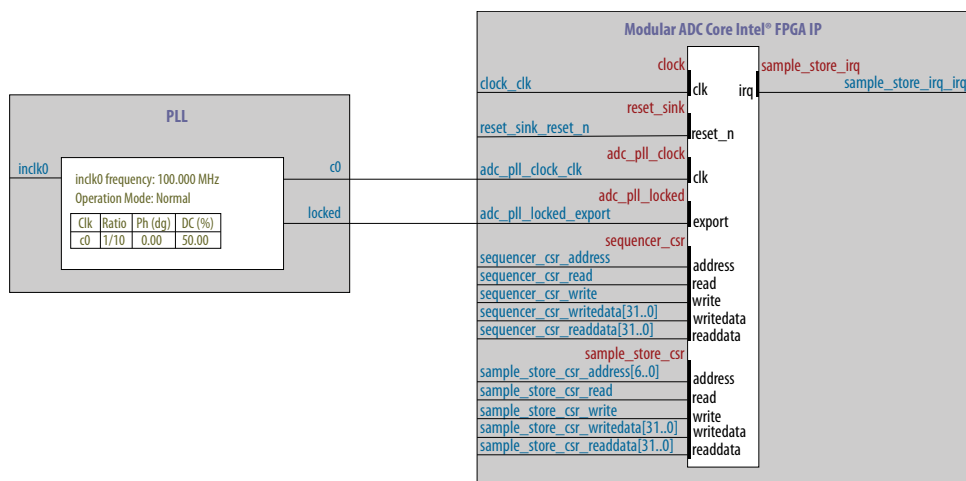
- [User-Specified ADC Logic Simulation Output](#) on page 32
Provides more information about using your own stimulus input file to simulate the ADC output data.
- [Guidelines: Board Design for Analog Input](#) on page 35
Provides more information about the sampling rate and settling time.

4.5. Completing ADC Design

The ADC design requires that the ALTPLL IP core clocks the Modular ADC Core IP core.

Generate the ALTPLL and Modular ADC Core IP cores with the settings in the related information.

Figure 30. Basic Intel MAX 10 ADC Design



1. Create the design as shown in the preceding figure.
2. Connect the `c0` signal from the ALTPLL IP core to the `adc_pll_clock_clk` port of the Modular ADC Core IP core.
3. Connect the `locked` signal from the ALTPLL IP core to the `adc_pll_locked_export` port of the Modular ADC Core IP core.
4. Create the ADC Avalon slave interface to start the ADC.

Related Information

- [Creating Intel MAX 10 ADC Design](#) on page 40
- [Parameters Settings for Generating ALTPLL IP Core](#) on page 41
- [Parameters Settings for Generating Modular ADC Core or Modular Dual ADC Core IP Core](#) on page 42
- [Configuration 1: Standard Sequencer with Avalon-MM Sample Storage](#) on page 22
- [Configuration 2: Standard Sequencer with Avalon-MM Sample Storage and Threshold Violation Detection](#) on page 22
- [Configuration 3: Standard Sequencer with External Sample Storage](#) on page 23
- [Configuration 4: ADC Control Core Only](#) on page 24



5. Modular ADC Core Intel FPGA IP and Modular Dual ADC Core Intel FPGA IP References

The Modular ADC Core or Modular Dual ADC Core IP core is a soft controller for the ADC hard IP blocks. You can generate soft IPs to instantiate the on-chip ADC blocks. With this IP core, you can configure the ADCs and abstract the low level handshake with the ADC hard IP blocks.

The Intel Quartus Prime software generates your customized Modular ADC Core or Modular Dual ADC Core IP core according to the parameter options that you set in the parameter editor.

Related Information

- [Intel MAX 10 Analog to Digital Converter Overview](#) on page 4
- [Modular ADC Core and Modular Dual ADC Core IP Cores](#) on page 20
- [Modular ADC Core IP Core Configuration Variants](#) on page 21

5.1. Modular ADC Core Parameters Settings

There are three groups of options: **General**, **Channels**, and **Sequencer**.

Table 16. Modular ADC Core Parameters - General

Parameter	Allowed Values	Description
Core Variant	<ul style="list-style-type: none"> Standard sequencer with Avalon-MM sample storage Standard sequencer with Avalon-MM sample storage and threshold violation detection Standard sequencer with external sample storage ADC control core only 	Selects the core configuration for the Modular ADC Core IP core.
Debug Path	<ul style="list-style-type: none"> Disabled Enabled 	Enables the debug path.
Generate IP for which ADCs of this device?	<ul style="list-style-type: none"> 1st ADC 2nd ADC 	For devices that have two ADC blocks, specifies which ADC block you want to instantiate using the IP core.
ADC Sample Rate	25 kHz, 50 kHz, 100 kHz, 200 kHz, 250 kHz, 500 kHz, and 1 MHz	Specifies the ADC sampling rate. The sampling rate you select affects which ADC input clock frequencies are available. Refer to the related information for more details about the sampling rate and the required settling time.
ADC Input Clock	2 MHz, 10 MHz, 20 MHz, 40 MHz, and 80 MHz	<p>Specifies the frequency of the PLL clock counter zero (c0) clock supply for the ADC core clock.</p> <ul style="list-style-type: none"> You must configure the c0 of the first ALTPLL IP core that you instantiate to output one of the frequencies in the allowed values list. Connect the ALTPLL c0 output signal to the Modular ADC Core clk_in_pll_c0 input signal. <p>For valid ADC sampling rate and input clock frequencies combinations, refer to the related information.</p>
Reference Voltage Source	<ul style="list-style-type: none"> External Internal 	<p>Specifies the source of voltage reference for the ADC:</p> <ul style="list-style-type: none"> External—uses ADC_VREF pin as the voltage reference source. Internal—uses the on-chip 2.5 V (3.0/3.3V on voltage-regulated devices) as the voltage reference source.
External Reference Voltage	<ul style="list-style-type: none"> Dual supply devices: up to 2.5 V Single supply devices: up to 3.63 V 	Specifies the voltage of ADC_VREF pin if you use it as reference voltage to the ADC.
Enable user created expected output file	<ul style="list-style-type: none"> Enabled Disabled 	<p>Specifies the source of output data for ADC logic simulation:</p> <ul style="list-style-type: none"> Enabled—uses the stimulus input file you provide for each ADC channel, except the TSD channel, to simulate the output data. Disabled—uses fixed expected output data for all ADC channels. This is the default setting.

continued...

Parameter	Allowed Values	Description
		For more information about user-specified ADC logic simulation output, refer to the related information.

Table 17. Modular ADC Core Parameters - Channels

This group of parameters is divided into several tabs—one for each channel, and one tab for the TSD.

Parameter	Allowed Values	Description
Use Channel 0 (Dedicated analog input pin - ANAIN) (CH0 tab)	<ul style="list-style-type: none"> On Off 	Enables the dedicated analog input pin.
User created expected output file	—	Specifies user-created stimulus input file to simulate the output data for the channel. This option is available for each enabled channel except the TSD if you select Enable user created expected output file .
Use Channel <i>N</i> (Each channel in its own tab)	<ul style="list-style-type: none"> On Off 	Enables the dual-function analog input, where <i>N</i> is: <ul style="list-style-type: none"> 1 to 16 channels for single ADC devices 1 to 8 channels for dual ADC devices
Use on-chip TSD (TSD tab)	<ul style="list-style-type: none"> On Off 	Specifies that the IP core reads the built-in temperature sensor in the ADC. If you turn on this option, the ADC sampling rate is up to 50 kHz when it reads the temperature measurement. After it completes the temperature reading, the ADC sampling rate is up to 1 MHz.
Enable Maximum threshold for Channel <i>N</i> (Each channel in its own tab)	<ul style="list-style-type: none"> On Off 	Enables the maximum threshold feature for the channel. This option is available only if you select the Standard sequencer with Avalon-MM sample storage and threshold violation detection core variant.
Enable Maximum threshold for on-chip TSD (TSD tab)	<ul style="list-style-type: none"> On Off 	Enables the maximum threshold feature for the TSD. This option is available only if you select the Standard sequencer with Avalon-MM sample storage and threshold violation detection core variant.
Enter Maximum Threshold for Channel <i>N</i> (Each channel in its own tab, including channel 0)	Depends on reference voltage	Specifies the maximum threshold value in Volts. This setting is available only if you select the Standard sequencer with Avalon-MM sample storage and threshold violation detection core variant.
Enter Maximum Threshold for on-chip TSD (TSD tab)	—	Specifies the maximum threshold value in Celsius. This setting is available only if you select the Standard sequencer with Avalon-MM sample storage and threshold violation detection core variant.
Enable Minimum threshold for Channel <i>N</i> (Each channel in its own tab, including channel 0)	<ul style="list-style-type: none"> On Off 	Enables the minimum threshold feature for the channel. This option is available only if you select the Standard sequencer with Avalon-MM sample storage and threshold violation detection core variant.
Enable Minimum threshold for on-chip TSD (TSD tab)	<ul style="list-style-type: none"> On Off 	Enables the minimum threshold feature for the TSD. This option is available only if you select the Standard sequencer with Avalon-MM sample storage and threshold violation detection core variant.
Enter Minimum Threshold for Channel <i>N</i> (Each channel in its own tab, including channel 0)	Depends on reference voltage	Specifies the minimum threshold value in Volts. This setting is available only if you select the Standard sequencer with Avalon-MM sample storage and threshold violation detection core variant.
Enter Minimum Threshold for on-chip TSD	—	Specifies the minimum threshold value in Celsius.

continued...

Parameter	Allowed Values	Description
(TSD tab)		This setting is available only if you select the Standard sequencer with Avalon-MM sample storage and threshold violation detection core variant.
Enable Prescaler for Channel <i>N</i>	<ul style="list-style-type: none"> On Off 	Enables the prescaler function, where <i>N</i> is: <ul style="list-style-type: none"> Channels 8 and 16 (if available) for single ADC devices. Channel 8 of ADC1 or ADC2 for dual ADC devices.

Table 18. Modular ADC Core Parameters - Sequencer

Parameter	Allowed Values	Description
Number of slot used	1 to 64	Specifies the number of conversion sequence slots to use. The Conversion Sequence Channels section displays the slots available according to the number of slots you select here.
Slot <i>N</i>	Enabled channel number (CH <i>N</i>)	Specifies which enabled ADC channel to use for the slot in the sequence. The selection option lists the ADC channels that you turned on in the Channels parameter group.

Related Information

- Sequencer Core on page 27
- Configuration 1: Standard Sequencer with Avalon-MM Sample Storage on page 22
- Configuration 2: Standard Sequencer with Avalon-MM Sample Storage and Threshold Violation Detection on page 22
- Configuration 3: Standard Sequencer with External Sample Storage on page 23
- Configuration 4: ADC Control Core Only on page 24
- Modular ADC Core IP Core Channel Name to Intel MAX 10 Device Pin Name Mapping on page 49
- Modular Dual ADC Core IP Core Channel Name to Intel MAX 10 Device Pin Name Mapping on page 53
- Valid ADC Sample Rate and Input Clock Combination on page 54
- User-Specified ADC Logic Simulation Output on page 32
Provides more information about using your own stimulus input file to simulate the ADC output data.
- Guidelines: Board Design for Analog Input on page 35
Provides more information about the sampling rate and settling time.

5.1.1. Modular ADC Core IP Core Channel Name to Intel MAX 10 Device Pin Name Mapping

Each ADC channel in the Modular ADC Core IP core corresponds to different device pin name for single and dual ADC devices.

Table 19. Modular ADC Core IP Core Channel to Pin Mapping for Single ADC Devices

Channel Name	Pin Name
CH0	ANAIN1
CH1	ADC1IN1
<i>continued...</i>	

Channel Name	Pin Name
CH2	ADC1IN2
CH3	ADC1IN3
CH4	ADC1IN4
CH5	ADC1IN5
CH6	ADC1IN6
CH7	ADC1IN7
CH8	ADC1IN8
CH9	ADC1IN9
CH10	ADC1IN10
CH11	ADC1IN11
CH12	ADC1IN12
CH13	ADC1IN13
CH14	ADC1IN14
CH15	ADC1IN15
CH16	ADC1IN16

Table 20. Modular ADC Core IP Core Channel to Pin Mapping for Dual ADC Devices

ADC Block	Channel Name	Pin Name
First ADC	CH0	ANAIN1
	CH1	ADC1IN1
	CH2	ADC1IN2
	CH3	ADC1IN3
	CH4	ADC1IN4
	CH5	ADC1IN5
	CH6	ADC1IN6
	CH7	ADC1IN7
	CH8	ADC1IN8
Second ADC	CH0	ANAIN2
	CH1	ADC2IN1
	CH2	ADC2IN2
	CH3	ADC2IN3
	CH4	ADC2IN4
	CH5	ADC2IN5
	CH6	ADC2IN6
	CH7	ADC2IN7
	CH8	ADC2IN8

5.2. Modular Dual ADC Core Parameters Settings

There are three groups of options: **General**, **Channels**, and **Sequencer**.

Table 21. Modular Dual ADC Core Parameters - General

Parameter	Allowed Values	Description
Core Variant	<ul style="list-style-type: none"> Standard sequencer with Avalon-MM sample storage Standard sequencer with Avalon-MM sample storage and threshold violation detection Standard sequencer with external sample storage ADC control core only 	Selects the core configuration for the Modular Dual ADC Core IP core.
ADC Sample Rate	25 kHz, 50 kHz, 100 kHz, 200 kHz, 250 kHz, 500 kHz, and 1 MHz	Specifies the ADC sampling rate. The sampling rate you select affects which ADC input clock frequencies are available. Refer to the related information for more details about the sampling rate and the required settling time.
ADC Input Clock	2 MHz, 10 MHz, 20 MHz, 40 MHz, and 80 MHz	<p>Specifies the frequency of the PLL clock counter zero (c0) clock supply for the ADC core clock.</p> <ul style="list-style-type: none"> You must configure the c0 of the first ALTPLL IP core that you instantiate to output one of the frequencies in the allowed values list. Connect the ALTPLL c0 output signal to the Modular Dual ADC Core clk_in_pll_c0 input signal. <p>For valid ADC sampling rate and input clock frequencies combinations, refer to the related information.</p>
Reference Voltage (ADC1 or ADC2)	<ul style="list-style-type: none"> External Internal 	<p>Specifies the source of voltage reference for the ADC:</p> <ul style="list-style-type: none"> External—uses ADC_VREF pin as the voltage reference source. Internal—uses the on-chip 2.5 V (3.0/3.3V on voltage-regulated devices) as the voltage reference source.
External Reference Voltage	<ul style="list-style-type: none"> Dual supply devices: up to 2.5 V Single supply devices: up to 3.63 V 	Specifies the voltage of ADC_VREF pin if you use it as reference voltage to the ADC.
Enable user created expected output file	<ul style="list-style-type: none"> Enabled Disabled 	<p>Specifies the source of output data for ADC logic simulation:</p> <ul style="list-style-type: none"> Enabled—uses the stimulus input file you provide for each ADC channel, except the TSD channel, to simulate the output data. Disabled—uses fixed expected output data for all ADC channels. This is the default setting. <p>For more information about user-specified ADC logic simulation output, refer to the related information.</p>

Table 22. Modular Dual ADC Core Parameters - Channels

This group of parameters is divided into two main tabs for ADC1 and ADC2. For each tab, there are several channel tabs—one for each channel, and one tab for the TSD in ADC1.

Parameter	Allowed Values	Description
Use Channel 0 or 9 (Dedicated analog input pin - ANAIN) (CH0 tab for ADC1 or CH9 tab for ADC2)	<ul style="list-style-type: none"> On Off 	Enables the dedicated analog input pin for ADC1 or ADC2.
User created expected output file	—	Specifies user-created stimulus input file to simulate the output data for the channel. This option is available for each enabled channel except the TSD if you select Enable user created expected output file .
Use Channel <i>N</i> (Each channel in its own tab)	<ul style="list-style-type: none"> On Off 	Enables the dual-function analog input, where <i>N</i> is: <ul style="list-style-type: none"> Channels 1 to 8 for ADC1 Channels 10 to 17 for ADC2
Use on-chip TSD (TSD tab in ADC1 only)	<ul style="list-style-type: none"> On Off 	Specifies that the IP core reads the built-in temperature sensor in ADC1. If you turn on this option, the ADC sampling rate is up to 50 kHz when it reads the temperature measurement. After it completes the temperature reading, the ADC sampling rate is up to 1 MHz. <i>Note:</i> If you select the TSD for a sequencer slot in ADC1, select NULL for the same sequencer slot number in ADC2.
Enable Maximum threshold for Channel <i>N</i> (Each channel in its own tab)	<ul style="list-style-type: none"> On Off 	Enables the maximum threshold feature for the channel. This option is available only if you select the Standard sequencer with Avalon-MM sample storage and threshold violation detection core variant.
Enable Maximum threshold for on-chip TSD (TSD tab)	<ul style="list-style-type: none"> On Off 	Enables the maximum threshold feature for the TSD. This option is available only if you select the Standard sequencer with Avalon-MM sample storage and threshold violation detection core variant.
Enter Maximum Threshold for Channel <i>N</i> (Each channel in its own tab, including channel 0)	Depends on reference voltage	Specifies the maximum threshold value in Volts. This setting is available only if you select the Standard sequencer with Avalon-MM sample storage and threshold violation detection core variant.
Enter Maximum Threshold for on-chip TSD (TSD tab)	—	Specifies the maximum threshold value in Celsius. This setting is available only if you select the Standard sequencer with Avalon-MM sample storage and threshold violation detection core variant.
Enable Minimum threshold for Channel <i>N</i> (Each channel in its own tab, including channel 0)	<ul style="list-style-type: none"> On Off 	Enables the minimum threshold feature for the channel. This option is available only if you select the Standard sequencer with Avalon-MM sample storage and threshold violation detection core variant.
Enable Minimum threshold for on-chip TSD (TSD tab)	<ul style="list-style-type: none"> On Off 	Enables the minimum threshold feature for the TSD. This option is available only if you select the Standard sequencer with Avalon-MM sample storage and threshold violation detection core variant.
Enter Minimum Threshold for Channel <i>N</i> (Each channel in its own tab, including channel 0)	Depends on reference voltage	Specifies the minimum threshold value in Volts. This setting is available only if you select the Standard sequencer with Avalon-MM sample storage and threshold violation detection core variant.
Enter Minimum Threshold for on-chip TSD (TSD tab)	—	Specifies the minimum threshold value in Celsius.

continued...

Parameter	Allowed Values	Description
		This setting is available only if you select the Standard sequencer with Avalon-MM sample storage and threshold violation detection core variant.
Enable Prescaler for Channel <i>N</i>	<ul style="list-style-type: none"> On Off 	Enables the prescaler function, where <i>N</i> is: <ul style="list-style-type: none"> Channel 8 in ADC1 Channel 17 in ADC2

Table 23. Modular Dual ADC Core Parameters - Sequencer

Parameter	Allowed Values	Description
Number of slot used	1 to 64	Specifies the number of conversion sequence slots to use for both ADC1 and ADC2. The Conversion Sequence Channels section displays the slots available for ADC1 and ADC2 according to the number of slots you select here.
Slot <i>N</i>	Enabled channel number (CH <i>N</i>)	Specifies which enabled ADC channel to use for the slot in the sequence. The selection option lists the ADC channels that you turned on in the Channels parameter group for ADC1 and ADC2. <i>Note:</i> If you select the TSD for a sequencer slot in ADC1, select NULL for the same sequencer slot number in ADC2.

Related Information

- [Sequencer Core](#) on page 27
- [Configuration 1: Standard Sequencer with Avalon-MM Sample Storage](#) on page 22
- [Configuration 2: Standard Sequencer with Avalon-MM Sample Storage and Threshold Violation Detection](#) on page 22
- [Configuration 3: Standard Sequencer with External Sample Storage](#) on page 23
- [Configuration 4: ADC Control Core Only](#) on page 24
- [Modular ADC Core IP Core Channel Name to Intel MAX 10 Device Pin Name Mapping](#) on page 49
- [Modular Dual ADC Core IP Core Channel Name to Intel MAX 10 Device Pin Name Mapping](#) on page 53
- [Valid ADC Sample Rate and Input Clock Combination](#) on page 54
- [User-Specified ADC Logic Simulation Output](#) on page 32
Provides more information about using your own stimulus input file to simulate the ADC output data.
- [Guidelines: Board Design for Analog Input](#) on page 35
Provides more information about the sampling rate and settling time.

5.2.1. Modular Dual ADC Core IP Core Channel Name to Intel MAX 10 Device Pin Name Mapping

Each ADC channel in the Modular Dual ADC Core IP core corresponds to different device pin name.

Table 24. Modular Dual ADC Core IP Core Channel to Pin Mapping

ADC Block	Channel Name	Pin Name
ADC1	CH0	ANAIN1
	CH1	ADC1IN1
	CH2	ADC1IN2
	CH3	ADC1IN3
	CH4	ADC1IN4
	CH5	ADC1IN5
	CH6	ADC1IN6
	CH7	ADC1IN7
	CH8	ADC1IN8
ADC2	CH0	ANAIN2
	CH1	ADC2IN1
	CH2	ADC2IN2
	CH3	ADC2IN3
	CH4	ADC2IN4
	CH5	ADC2IN5
	CH6	ADC2IN6
	CH7	ADC2IN7
	CH8	ADC2IN8

5.3. Valid ADC Sample Rate and Input Clock Combination

Each predefined ADC sampling rate supports a list of input clock frequencies. When you configure the ALTPLL IP core to clock the ADC, use an ADC input clock frequency supported by your ADC sampling rate.

The ability to specify the ADC sampling rate allows you more design flexibility. If you are not using the maximum Intel MAX 10 ADC sampling rate, you get a wider settling time margin.

Table 25. Valid Combination of ADC Sampling Rate and Input Clock

Total ADC Sampling Rate (kHz)	ADC Input Clock Frequency (MHz)				
	2	10	20	40	80
1000	Yes	Yes	Yes	Yes	Yes
500	—	Yes	Yes	Yes	—
250	—	Yes	Yes	—	—
200	Yes	—	—	—	—
125	—	Yes	—	—	—
continued...					

Total ADC Sampling Rate (kHz)	ADC Input Clock Frequency (MHz)				
	2	10	20	40	80
100	Yes	—	—	—	—
50	Yes	—	—	—	—
25	Yes	—	—	—	—

Related Information

- [Parameters Settings for Generating ALTPLL IP Core](#) on page 41
- [Parameters Settings for Generating Modular ADC Core or Modular Dual ADC Core IP Core](#) on page 42
- [Modular ADC Core Parameters Settings](#) on page 47
- [Modular Dual ADC Core Parameters Settings](#) on page 51

5.4. Modular ADC Core and Modular Dual ADC Core Interface Signals

Depending on parameter settings you specify, different signals are available for the Modular ADC Core or Modular Dual ADC Core IP core.

5.4.1. Command Interface of Modular ADC Core and Modular Dual ADC Core

The command interface is an Avalon-ST type interface that supports a ready latency of 0.

Table 26. Command Interface Signals

Signal	Width (Bit)	Description
valid	1	Indication from the source port that current transfer is valid.
ready	1	Indication from the sink port that it is ready for current transfer.
channel	5	Indicates the channel that the ADC hard block samples from for current command. <ul style="list-style-type: none"> • 31—recalibration request • 30:18—not used • 17—temperature sensor • 16:0—channels 16 to 0; where channel 0 is the dedicated analog input pin and channels 1 to 16 are the dual purpose analog input pins
startofpacket	1	Indication from the source port that current transfer is the start of packet. <ul style="list-style-type: none"> • For altera_adc_sequencer core implementation, the IP core asserts this signal during the first slot of conversion sequence data array. • For altera_adc_control core implementation, this signal is ignored. The IP core just passes the received information back to the corresponding response interface.
endofpacket	1	Indication from the source port that current transfer is the end of packet. <ul style="list-style-type: none"> • For altera_adc_sequencer core implementation, IP core asserts this signal during the final slot of conversion sequence data array. • For altera_adc_control core implementation, this signal is ignored. The IP core just passes the received information back to the corresponding response interface.

Related Information

- [Modular ADC Core IP Core Channel Name to Intel MAX 10 Device Pin Name Mapping](#) on page 49
- [Modular Dual ADC Core IP Core Channel Name to Intel MAX 10 Device Pin Name Mapping](#) on page 53

5.4.2. Response Interface of Modular ADC Core and Modular Dual ADC Core

The response interface is an Avalon-ST type interface that does not support backpressure. To avoid overflow condition at the source port, implement sink ports with response data process time that is fast enough, or with enough buffers storage.

Table 27. Response Interface Signals

Signal	Width (Bit)	Description
valid	1	Indication from the source port that current transfer is valid.
channel	5	Indicates the ADC channel to which the ADC sampling data corresponds for the current response. <ul style="list-style-type: none"> • 31:18—not used • 17—temperature sensor • 16:0—channels 16 to 0; where channel 0 is the dedicated analog input pin and channels 1 to 16 are the dual purpose analog input pins
data	12 or 24	ADC sampling data: <ul style="list-style-type: none"> • 12 bit width for Modular ADC Core • 24 bit width for Modular Dual ADC Core
startofpacket	1	Indication from the source port that current transfer is the start of packet. For altera_adc_control core implementation, the source of this signal is from the corresponding command interface.
endofpacket	1	Indication from the source port that current transfer is the end of packet. For altera_adc_control core implementation, the source of this signal is from the corresponding command interface.

5.4.3. Threshold Interface of Modular ADC Core and Modular Dual ADC Core

The threshold interface is an Avalon-ST type interface that does not support backpressure.

Table 28. Threshold Interface Signals

Signal	Width (Bit)	Description
valid	1	Indication from the source port that current transfer is valid.
channel	5	Indicates the ADC channel for which the threshold value has been violated. <ul style="list-style-type: none"> 31:18—not used 17—temperature sensor 16:0—channels 16 to 0; where channel 0 is the dedicated analog input pin and channels 1 to 16 are the dual purpose analog input pins
data	1	Indicates the type of threshold violation: <ul style="list-style-type: none"> 1—Exceeds maximum threshold value 0—Below minimum threshold value

Related Information

- [Modular ADC Core IP Core Channel Name to Intel MAX 10 Device Pin Name Mapping](#) on page 49
- [Modular Dual ADC Core IP Core Channel Name to Intel MAX 10 Device Pin Name Mapping](#) on page 53

5.4.4. CSR Interface of Modular ADC Core and Modular Dual ADC Core

The CSR interface is an Avalon memory-mapped slave interface.

Table 29. CSR Interface Signals

Signal	Width (Bit)	Description
address	1 or 7	Avalon memory-mapped address bus. The address bus width is in the unit of word addressing: <ul style="list-style-type: none"> altera_adc_sample_store core—address width is seven altera_adc_sequencer core—address width is one
read	1	Avalon memory-mapped read request.
write	1	Avalon memory-mapped write request.
writedata	32	Avalon memory-mapped write data bus.
readdata	32	Avalon memory-mapped read data bus.

5.4.5. IRQ Interface of Modular ADC Core and Modular Dual ADC Core

The IRQ interface is an interrupt interface type.

Table 30. IRQ Interface Signals

Signal	Width (Bit)	Description
irq	1	Interrupt request. Output signal. Active high.

5.4.6. Peripheral Clock Interface of Modular ADC Core and Modular Dual ADC Core

The peripheral clock interface is a clock sink interface type.

Table 31. Peripheral Clock Interface Signals

Signal	Width (Bit)	Description
clock	1	Single clock that clocks all Modular ADC Core or Modular Dual ADC Core micro cores. <i>Note:</i> To avoid functional failure, the required minimum peripheral clock frequency is 25 MHz.

5.4.7. Peripheral Reset Interface of Modular ADC Core and Modular Dual ADC Core

The peripheral reset interface is a reset sink interface type.

Table 32. Peripheral Reset Interface Signals

Signal	Width (Bit)	Description
reset_n	1	Single reset source that resets all Modular ADC Core or Modular Dual ADC Core micro cores. You must assert this reset signal asynchronously and deassert it synchronously. The IP cores do not synchronize the deassertion of the reset signal.

5.4.8. ADC PLL Clock Interface of Modular ADC Core and Modular Dual ADC Core

The ADC PLL clock interface is a clock sink interface type.

Table 33. ADC PLL Clock Interface Signals

Signal	Width (Bit)	Description
clock	1	ADC hard IP clock source from C0 output of dedicated PLL1 or PLL3. Export this interface from the Platform Designer (Standard) system.

Related Information

- [Customizing and Generating Modular ADC Core IP Core](#) on page 40
- [Parameters Settings for Generating ALTPLL IP Core](#) on page 41
- [ADC Clock Sources](#) on page 16
- [PLL Locations, Intel MAX 10 Clocking and PLL User Guide](#)
Provides more information about the availability of PLL3 in different Intel MAX 10 devices and packages.

5.4.9. ADC PLL Locked Interface of Modular ADC Core and Modular Dual ADC Core

The ADC PLL locked interface is a conduit end interface type.

Table 34. ADC PLL Locked Interface Signals

Signal	Width (Bit)	Description
conduit	1	ADC hard IP locked signal output of dedicated PLL1 or PLL3. Export this interface from the Platform Designer (Standard) system.

Related Information

- [Customizing and Generating Modular ADC Core IP Core](#) on page 40
- [Parameters Settings for Generating ALTPLL IP Core](#) on page 41
- [ADC Clock Sources](#) on page 16
- [PLL Locations, Intel MAX 10 Clocking and PLL User Guide](#)
Provides more information about the availability of PLL3 in different Intel MAX 10 devices and packages.

5.5. Modular ADC Core Register Definitions

The registers in the generated Modular ADC Core IP core provide the IP core with the control and settings during operation.

5.5.1. Sequencer Core Registers

Table 35. Command Register (CMD)

Address Offset: 0x0

Bit	Name	Attribute	Description	Value	Default
31:4	Reserved	Read	Reserved.	—	0
3:1	Mode	Read-Write	Indicates the operation mode of the sequencer core. These bits are ignored when the run bit (bit 0) is set. In continuous conversion, the data is overwritten after a complete sampling sequence.	<ul style="list-style-type: none"> • 7—Recalibrate the ADC • 6 to 2—Reserved • 1—Single cycle ADC conversion • 0—Continuous ADC conversion 	0
0	Run	Read-Write	Use this control bit to trigger the sequencer core operation. The Modular ADC Core IP core waits until the sequencer core completes its current operation before writing to this register bit.	<ul style="list-style-type: none"> • 1—Run • 0—Stop 	0

Related Information

[Sequencer Core](#) on page 27

5.5.2. Sample Storage Core Registers

Table 36. ADC Sample Register (ADC_SAMPLE) of Modular ADC Core

Address Offset: 0x3F (slot 64)—0x0 (slot 1)

Bit	Name	Attribute	Description	Value	Default
31:12	Reserved	Read	Reserved.	—	0
11:0	Sample	Read	The data sampled by the ADC for the corresponding slot.	Sampled data	0

Table 37. ADC Sample Register (ADC_SAMPLE) of Modular Dual ADC Core

Address Offset: 0x3F (slot 64)—0x0 (slot 1)

Bit	Name	Attribute	Description	Value	Default
31:28	Reserved	Read	Reserved.	—	0
27:16	Sample	Read	The data sampled by ADC2 for the corresponding slot.	Sampled data	0
15:12	Reserved	Read	Reserved.	—	0
11:0	Sample	Read	The data sampled by ADC1 for the corresponding slot.	Sampled data	0

Table 38. Interrupt Enable Register (IER)

Address Offset: 0x40

Clear the enable bit to prevent the corresponding interrupt status bit from causing interrupt output assertion (IRQ). The enable bit does not stop the interrupt status bit value from showing in the interrupt status register (ISR).

Bit	Name	Attribute	Description	Value	Default
31:1	Reserved	Read	Reserved.	—	0
0	M_EOP	Read-Write	The enable bit for the end of packet (EOP) interrupt.	<ul style="list-style-type: none"> 1—Enables the corresponding interrupt 0—Disables the corresponding interrupt 	1

Table 39. Interrupt Status Register (ISR)

Address Offset: 0x41

Bit	Name	Attribute	Description	Value	Default
31:1	Reserved	Read	Reserved.	—	0
0	EOP	Read-Write (one cycle)	EOP interrupt.	This bit is automatically set by the hardware. When "1", it indicates that a packet of samples is stored and ready to be read. You can retrieve the sample value from the	0

continued...

Bit	Name	Attribute	Description	Value	Default
				ADC_SAMPLE register. To clear this bit to "0" for the next interrupt, write "1".	

Related Information

[Sample Storage Core](#) on page 28

5.6. ADC HAL Device Driver for Nios Processor

The Modular ADC Core and Modular Dual ADC Core IP core provides a HAL device driver. You can integrate the device driver into the HAL system library for Nios processor systems.

The IP cores provide software files that define low-level access to the hardware. You can use the macros definition and functions in the software files to initialize the cores.

- `altera_modular_adc_sequencer_regs.h`—this file defines the register map for the sequencer core. It provides symbolic constants to access the low-level hardware.
- `altera_modular_adc_sample_store_regs.h`—this file defines the register for sample storage core. It provides symbolic constants to access the low-level hardware.
- `altera_modular_adc.h` or `altera_modular_dual_adc.h`—include this file into your application. It automatically includes the other header files and defines additional functions.
- `altera_modular_adc.c` or `altera_modular_dual_adc.c`—this file implements helper functions that are defined in the header file.

In the same design, there can only be a single type of IP core, either Modular ADC Core or Modular Dual ADC Core IP.

In Intel Quartus Prime Standard Edition software version 22.1, you need to apply the following software settings based on the IP core configuration (**Core Variant**).

1. All instantiated IP core(s) apply **Standard sequencer with external sample storage**.
 - a. Enable `external_sample_storage` in **BSP Editor > BSP Driver** tab.
2. Instantiated IP cores have different **Core Variant**.
 - a. Enable `different_sample_storage` in **BSP Editor > BSP Driver** tab.
 - b. Modify `alt_sys_init.c`
 - i. For Avalon-MM sample storage (`_CORE_VARIANT = 0` or `1`),


```
Change ALTERA_MODULAR_ADC_INSTANCE to
ALTERA_MODULAR_ADC_INSTANCE_AVL_MEM
Change ALTERA_MODULAR_ADC_INIT to
ALTERA_MODULAR_ADC_INIT_AVL_MEM
```
 - ii. For external sample storage (`_CORE_VARIANT = 2`),

Change ALTERA_MODULAR_ADC_INSTANCE to
ALTERA_MODULAR_ADC_INSTANCE_EXT

Change ALTERA_MODULAR_ADC_INIT to
ALTERA_MODULAR_ADC_INIT_EXT

Note: The value of _CORE_VARIANT can be found in system.h.

5.6.1. Driver API

Table 40. adc_stop

Prototype:	void adc_stop(int sequencer_base)
Include:	<ul style="list-style-type: none"> altera_modular_adc.h altera_modular_dual_adc.h
Parameter:	<ul style="list-style-type: none"> sequencer_base – Sequencer base value
Return:	-
Description:	Writes 0 to the Sequencer CMD register RUN bit, and polls the RUN bit until it is 0.

Table 41. adc_start

Prototype:	void adc_start(int sequencer_base)
Include:	<ul style="list-style-type: none"> altera_modular_adc.h altera_modular_dual_adc.h
Parameter:	<ul style="list-style-type: none"> sequencer_base – Sequencer base value
Return:	-
Description:	Writes 1 to the Sequencer CMD register RUN bit.

Table 42. adc_set_mode_run_once

Prototype:	void adc_set_mode_run_once(int sequencer_base)
Include:	<ul style="list-style-type: none"> altera_modular_adc.h altera_modular_dual_adc.h
Parameter:	<ul style="list-style-type: none"> sequencer_base – Sequencer base value
Return:	-
Description:	Writes 1 to the Sequencer CMD register MODE bit once. Note: Stop the ADC before calling this function. While RUN bit is set, changing ADC mode has no effect.

Table 43. adc_set_mode_run_continuously

Prototype:	void adc_set_mode_run_continuously(int sequencer_base)
Include:	<ul style="list-style-type: none"> altera_modular_adc.h altera_modular_dual_adc.h
Parameter:	<ul style="list-style-type: none"> sequencer_base – Sequencer base value
Return:	-
Description:	Writes 1 to the Sequencer CMD register MODE bit to continuous. Note: Stop the ADC before calling this function. While RUN bit is set, changing ADC mode has no effect.

Table 44. adc_recalibrate

Prototype:	void adc_recalibrate(int sequencer_base)
Include:	<ul style="list-style-type: none"> altera_modular_adc.h altera_modular_dual_adc.h
Parameter:	<ul style="list-style-type: none"> sequencer_base – Sequencer base value
Return:	-
Description:	<p>The function performs as follows,</p> <ol style="list-style-type: none"> 1. Backup Sequencer CMD register (because some values can be overwritten). 2. Stop the ADC Sequencer Core. 3. Set the recalibration request bits. 4. Start the ADC Sequencer Core. 5. Poll for RUN bit to be clear. 6. Restore Sequencer CMD register

Table 45. adc_interrupt_enable

Prototype:	void adc_interrupt_enable(int sample_store_base)
Include:	<ul style="list-style-type: none"> altera_modular_adc.h altera_modular_dual_adc.h
Parameter:	<ul style="list-style-type: none"> sample_store_base – Base address of sample store core
Return:	-
Description:	Sets the M_EOP bit in the ADC Sample Storage IER register.

Table 46. adc_interrupt_disable

Prototype:	void adc_interrupt_disable(int sample_store_base)
Include:	<ul style="list-style-type: none"> altera_modular_adc.h altera_modular_dual_adc.h
Parameter:	<ul style="list-style-type: none"> sample_store_base – Base address of sample store core
Return:	-
Description:	Clears the M_EOP bit in the ADC Sample Storage IER register.

Table 47. adc_clear_interrupt_status

Prototype:	void adc_clear_interrupt_status(int sample_store_base)
Include:	<ul style="list-style-type: none"> altera_modular_adc.h altera_modular_dual_adc.h
Parameter:	<ul style="list-style-type: none"> sample_store_base – Base address of sample store core
Return:	-
Description:	Clears the EOP bits in the Sample Storage ISR register.

Table 48. adc_wait_for_interrupt

Prototype:	void adc_wait_for_interrupt(int sample_store_base)
Include:	<ul style="list-style-type: none"> altera_modular_adc.h altera_modular_dual_adc.h
<i>continued...</i>	

Parameter:	<ul style="list-style-type: none"> sample_store_base – Base address of sample store core
Return:	-
Description:	Waits while the EOP bit of Sample Storage ISR register is 0.

Table 49. adc_interrupt_asserted

Prototype:	int adc_interrupt_asserted(int sample_store_base)
Include:	<ul style="list-style-type: none"> altera_modular_adc.h altera_modular_dual_adc.h
Parameter:	<ul style="list-style-type: none"> sample_store_base – Base address of sample store core
Return:	Return -1 if EOP bit is set, 0 if otherwise
Description:	Reads the EOP bit of Sample Storage IRQ ISR register.

Table 50. alt_adc_word_read

Prototype:	void alt_adc_word_read(alt_u32 sample_store_base, alt_u32* dest_ptr, alt_u32 len)
Include:	<ul style="list-style-type: none"> altera_modular_adc.h altera_modular_dual_adc.h
Parameter:	<ul style="list-style-type: none"> sample_store_base – Base address of sample store core dest_ptr – destination buffer len – number of 32-bit reads
Return:	-
Description:	Reads words from the sample store.

Related Information

- [Nios® V Processor Software Developer Handbook](#)
Provides more information about the HAL API in Nios® II processor.
- [Nios® II Software Developer Handbook](#)
Provides more information about the HAL API in Nios® II processor.



6. Intel MAX 10 Analog to Digital Converter User Guide Archives

For the latest and previous versions of this user guide, refer to [Intel MAX 10 Analog to Digital Converter User Guide](#). If an IP or software version is not listed, the user guide for the previous IP or software version applies.

IP versions are the same as the Intel Quartus Prime Design Suite software versions up to v19.1. From Intel Quartus Prime Design Suite software version 19.2 or later, IP cores have a new IP versioning scheme.

7. Document Revision History for Intel MAX 10 Analog to Digital Converter User Guide

Document Version	Intel Quartus Prime Version	Changes
2024.01.03	22.1	<ul style="list-style-type: none"> Changed the topic: <i>ADC HAL Device Driver for Nios II Gen 2</i> to <i>ADC HAL Device Driver for Nios Processor</i>. Added new topic: <i>Driver API</i>.
2022.10.31	22.1	Updated <i>ADC HAL Device Driver for Nios II Gen 2</i> section.
2021.05.04	20.1	Updated the description of the <code>irq</code> signal to specify that it is an output signal that is active high.
2021.01.12	20.1	Updated the table listing the Modular Dual ADC Core IP core channel to pin mapping.
2020.03.17	19.1	Updated the ADC control core timing diagram and removed the accompanying tables.
2019.01.14	18.1	<ul style="list-style-type: none"> Updated the description of the <code>reset_n</code> signal. Updated the IP core names: <ul style="list-style-type: none"> "Altera Modular ADC" to "Modular ADC Core Intel FPGA IP" "Altera Modular Dual ADC" to "Modular Dual ADC Core Intel FPGA IP"

Date	Version	Changes
December 2017	2017.12.15	<ul style="list-style-type: none"> Added single power supply U324 package. Added a topic that shows the equations and calculation example to convert between analog voltage values and digital representation. Updated the topic about ADC timing to add time calculation examples for the ADC control core based on the sampling rate. Added a note to specify that the Modular ADC Core and Modular Dual ADC Core IP cores support generating only Verilog simulation scripts. VHDL simulation is not supported. Added note to the topic about the peripheral clock interface signal to specify that the peripheral clock frequency must be at least 25 MHz. Updated the slot numbers of the ADC sample register address offset to correspond to the sequence slot numbers in Intel Quartus Prime.
July 2017	2017.07.06	Updated the description of the EOP bit "0" of the Interrupt Status Register (ISR) to improve clarity.
February 2017	2017.02.21	Rebranded as Intel.
January 2017	2017.01.25	Added a topic that lists the actual TSD sampling rate based on the ADC sampling rate selected in the IP core.
continued...		

Date	Version	Changes
October 2016	2016.10.31	<ul style="list-style-type: none"> Updated the topic about the ADC voltage reference to specify that you must use clean external voltage reference with a maximum resistance of 100 Ω. Updated the topic about the ADC sequencer to clarify that "conversion mode" refers to the sequencer conversion mode, namely the single-cycle and continuous ADC conversion modes. Added a related information link to a topic in the <i>Intel MAX 10 Clocking and PLL User Guide</i> that lists the availability of PLL1 and PLL3 in different Intel MAX 10 devices and packages. Updated various topics throughout the user guide to improve the clarity of descriptions related to the user-specified ADC logic simulation output feature. Updated the VCCVREF pin name to ADC_VREF. Edited the board design guidelines for analog input: <ul style="list-style-type: none"> Updated text to improve clarity. Updated the F_{cutoff} @ -3dB recommendation from "five times" to "at least two times" the input frequency. Updated the figure showing the first order active low pass filter example.
May 2016	2016.05.02	<ul style="list-style-type: none"> Removed all preliminary marks. Added new function to specify predefined ADC sampling rate up to 1 MSPS. Previously, the ADC always operate at the maximum sampling rate. Removed link to a workaround to reduce the sampling rate. Now you can set the sampling rate in the IP core parameter editor. Added the ADC Toolkit that supports the Modular ADC Core and Modular Dual ADC Core IP cores. Added feature to simulate ADC output using your own expected output files for each ADC channel except the TSD channel. Corrected the description for bits 11:0 and bits 27:16 of the ADC sample register for Modular ADC Core and Modular Dual ADC Core IP cores. Bits 11:0 and bits 27:16 hold the actual 12 bit sampled data for the storage slot instead of the slot number. Corrected the default value for bit 0 of the interrupt enable register (IER) and interrupt status register (ISR). The default value for M_EOP is 1 and for EOP is 0.
November 2015	2015.11.02	<ul style="list-style-type: none"> Added related information link to <i>Introduction to Altera IP Cores</i>. Added links to instructional videos that demonstrate how to create ADC designs in Intel MAX 10 devices. Changed instances of <i>Quartus II</i> to <i>Quartus Prime</i>.
June 2015	2015.06.11	Updated the board design guidelines for analog input.
May 2015	2015.05.04	<ul style="list-style-type: none"> Added the Modular Dual ADC Core IP core. Removed F672 from the 10M25 device and added ADC information for the E144 package of the 10M04 device: <ul style="list-style-type: none"> Updated the ADC block counts. Updated the ADC vertical migration support. Updated the ADC channel counts. Updated the table that lists the ADC channel count to list only 8 dual function pins (instead of 16) for the M153 and U169 packages. Updated the ADC vertical migration diagram to clarify that there are single ADC devices with eight and 16 dual function pins. Updated the topic about ADC conversion to specify that in prescaler mode, the analog input in dual and single supply devices can measure up to 3.0 V and 3.6 V, respectively. Updated the ADC IP core architecture figures to include features for the dual ADC IP core. Added information and topics about the response merge and dual ADC synchronizer micro cores.
continued...		

Date	Version	Changes
		<ul style="list-style-type: none"> Removed notes about contacting Altera for the ADC pin RLC filter design. Updated the ADC prescaler topic to change the ADC2 channel that supports prescaler from channel 16 to channel 17. Updated the diagram that shows the ADC timing: <ul style="list-style-type: none"> To clarify that the numbers are hexadecimal numbers. Relabeled the signals to match the command and response interface signal names. Updated the RC constant and filter value and the filter design example figure to clarify the source of the example values. Added guidelines for setting up the sequencer in dual ADC mode. Added topics that list the mapping of Modular ADC Core and Modular Dual ADC Core IP cores channel names to Intel MAX 10 device pin names. Corrected the address offset of the interrupt enable register (from 0x41 to 0x40) and interrupt status register (from 0x40 to 0x41) for the sample storage core. Updated the sample storage core registers table to include registers for Modular Dual ADC Core. Removed statements about availability of the threshold trigger feature in a future version of the Intel Quartus Prime software. The feature is now available from version 15.0 of the software.
December 2014	2014.12.15	<ul style="list-style-type: none"> Added ADC prescaler block diagram. Replaced the ADC continuous conversion timing diagram with the ADC timing diagram. Corrected a minor error in the example in the topic about the sample storage core. Added information that the ADC TSD measures the temperature using a 64-samples running average method. Updated majority of the temperature codes in the table that lists the temperature code conversion. Added chapter that provides the ADC design considerations. Removed mention of value "0" for values allowed for the number of sequencer slots used in Modular ADC Core IP core parameter editor. Only values 1 to 64 are allowed. Removed the statement about enabling and disabling additional ADC response interface or debugging in the topic about the Modular ADC Core IP core configuration variants. You can enable or disable the debug path in the parameter editor. Removed the debug paths diagrams for each ADC core configuration. Removed the statement about using the sequencer core to trigger recalibration. The ADC is automatically recalibrated when it switches from normal sensing mode to temperature sensing mode. Edited text to clarify about routing power or ground traces if power or ground plane is not possible. Updated the total RC constant values in the table that shows the RC constant and filter values calculation. Corrected spelling for "prescaler".
September 2014	2014.09.22	Initial release.