9. SEU Mitigation in Cyclone IV Devices

CYIV-51009-1.3

This chapter describes the cyclical redundancy check (CRC) error detection feature in user mode and how to recover from soft errors.

Configuration error detection is supported in all Cyclone® IV devices including Cyclone IV GX devices, Cyclone IV E devices with 1.0-V core voltage, and Cyclone IV E devices with 1.2-V core voltage. However, user mode error detection is only supported in Cyclone IV GX devices and Cyclone IV E devices with 1.2-V core voltage.

Dedicated circuitry built into Cyclone IV devices consists of a CRC error detection feature that can optionally check for a single-event upset (SEU) continuously and automatically.

In critical applications used in the fields of avionics, telecommunications, system control, medical, and military applications, it is important to be able to:

- Confirm the accuracy of the configuration data stored in an FPGA device
- Alert the system to an occurrence of a configuration error

Using the CRC error detection feature for Cyclone IV devices does not impact fitting or performance.

This chapter contains the following sections:

Configuration Error Detection

Configuration error detection is available in all Cyclone IV devices including Cyclone IV GX devices, Cyclone IV E devices with 1.0-V core voltage, and Cyclone IV E devices with 1.2-V core voltage.

Configuration error detection determines if the configuration data received through an external memory device is corrupted during configuration. To validate the configuration data, the Quartus® II software uses a function to calculate the CRC value for each configuration data frame and stores the frame-based CRC value in the configuration data as part of the configuration bit stream.

During configuration, Cyclone IV devices use the same methodology to calculate 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 either the device detects an error or all the values are calculated.

In addition to the frame-based CRC value, the Quartus II software generates a 32-bit CRC value for the whole configuration bit stream. This 32-bit CRC value is stored in the 32-bit storage register at the end of the configuration and is used for user mode error detection that is discussed in "User Mode Error Detection".

User Mode Error Detection

User mode error detection is available in Cyclone IV GX and Cyclone IV E devices with 1.2-V core voltage. Cyclone IV E devices with 1.0-V core voltage do not support user mode error detection.

Soft errors are changes in a configuration random-access memory (CRAM) bit state due to an ionizing particle. Cyclone IV devices have built-in error detection circuitry to detect data corruption by soft errors in the CRAM cells.

This error detection capability continuously computes the CRC of the configured CRAM bits based on the contents of the device and compares it with the pre-calculated CRC value obtained at the end of the configuration. If the CRCs 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 Cyclone IV device error detection feature does not check memory blocks and I/O buffers. These device memory blocks support parity bits that are used to check the contents of memory blocks for any error. The I/O buffers are not verified during error detection because the configuration data uses flip-flops as storage elements that are more resistant to soft errors. Similar flip-flops are used to store the pre-calculated CRC and other error detection circuitry option bits.

The error detection circuitry in Cyclone IV devices uses a 32-bit CRC IEEE 802 standard and a 32-bit polynomial as the CRC generator. Therefore, a single 32-bit CRC calculation is performed by the device. If a soft error does not occur, the resulting 32-bit signature value is `0x00000000`, that results in a `0` on the `CRC_ERROR` output signal. If a soft error occurs in the device, the resulting signature value is non-zero and the `CRC_ERROR` output signal is `1`.

You can inject a soft error by changing the 32-bit CRC storage register in the CRC circuitry. After verifying the induced failure, you can restore the 32-bit CRC value to the correct CRC value with the same instruction and inserting the correct value.

Before updating it with a known bad value, Altera recommends reading out the correct value.

In user mode, Cyclone IV devices support the CHANGE_EDREG JTAG instruction, that allows you to write to the 32-bit storage register. You can use Jam™ STAPL files (**.jam**) to automate the testing and verification process. You can only execute this instruction when the device is in user mode, and it is a powerful design feature that enables you to dynamically verify the CRC functionality in-system without having to reconfigure the device. You can then use the CRC circuit to check for real errors induced by an SEU.

Table 9–1 describes the CHANGE_EDREG JTAG instructions.

Table 9–1. CHANGE_EDREG JTAG Instruction

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.

After the test completes, Altera recommends that you power cycle the device.

Automated SEU Detection

Cyclone IV devices offer on-chip circuitry for automated checking of SEU detection. Applications that require the device to operate error-free at high elevations or in close proximity to earth's north or south pole require periodic checks to ensure continued data integrity. The error detection cyclic redundancy code feature controlled by the **Device and Pin Options** dialog box in the Quartus II software uses a 32-bit CRC circuit to ensure data reliability and is one of the best options for mitigating SEU.

You can implement the error detection CRC feature with existing circuitry in Cyclone IV devices, eliminating the need for external logic. The CRC is computed by the device during configuration and checked against an automatically computed CRC during normal operation. The CRC_ERROR pin reports a soft error when configuration CRAM data is corrupted. You must decide whether to reconfigure the FPGA by strobing the nCONFIG pin low or ignore the error.

CRC_ERROR Pin

A specific CRC_ERROR error detection pin is required to monitor the results of the error detection circuitry during user mode. Table 9–2 describes the CRC_ERROR pin.

Table 9–2. Cyclone IV Device CRC_ERROR Pin Description

CRC_ERROR Pin Type	Description
I/O, Output (open-drain)	Active high signal indicates that the error detection circuit has detected errors in the configuration SRAM bits. This pin is optional and is used when the CRC error detection circuit is enabled in the Quartus II software from the **Error Detection CRC** tab of the **Device and Pin Options** dialog box. When using this pin, connect it to an external 10-kΩ pull-up resistor to an acceptable voltage that satisfies the input voltage of the receiving device.

The CRC_ERROR pin information for Cyclone IV devices is reported in the Cyclone IV Devices Pin-Outs on the Altera® website.

WYSIWYG is an optimization technique that performs optimization on a VQM (Verilog Quartus Mapping) netlist in the Quartus II software.

Error Detection Block

Table 9–3 lists the types of CRC detection to check the configuration bits.

Table 9–3. Types of CRC Detection to Check the Configuration Bits

First Type of CRC Detection	Second Type of CRC Detection
■ CRAM error checking ability (32-bit CRC) during user mode, for use by the CRC_ERROR pin. ■ There is only one 32-bit CRC value. This value covers all the CRAM data.	■ 16-bit CRC embedded in every configuration data frame. ■ During configuration, after a frame of data is loaded into the device, the pre-computed CRC is shifted into the CRC circuitry. ■ Simultaneously, the CRC value for the data frame shifted-in is calculated. If the pre-computed CRC and calculated CRC values do not match, nSTATUS is set low. ■ Every data frame has a 16-bit CRC. Therefore, there are many 16-bit CRC values for the whole configuration bit stream. ■ Every device has a different length of configuration data frame.

This section focuses on the first type—the 32-bit CRC when the device is in user mode.

Error Detection Registers

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 set high.

Figure 9–1 shows the block diagram of the error detection block and the two related 32-bit registers: the signature register and the storage register.

Figure 9–1. Error Detection Block Diagram

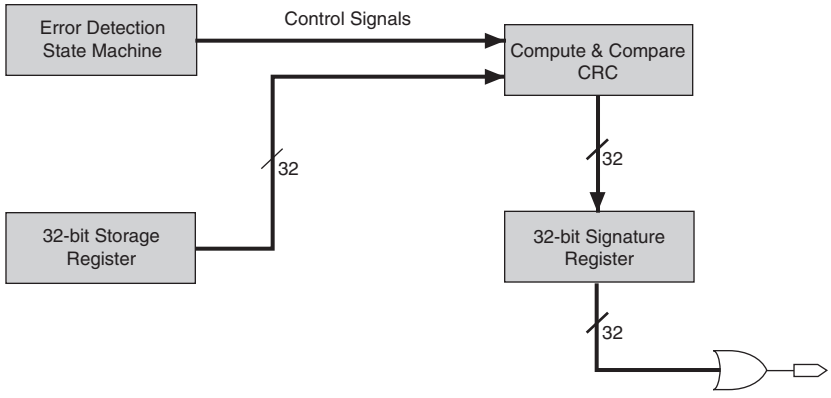


Table 9–4 defines the registers shown in Figure 9–1.

Table 9–4. Error Detection Registers

Register	Function
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 CRC circuit (called the Compute and Compare CRC block, as shown in Figure 9–1) 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 instruction.

Error Detection Timing

When the error detection CRC feature is enabled through the Quartus II software, the device automatically activates the CRC process upon entering user mode after configuration and initialization is complete.

The CRC_ERROR pin is driven low until the error detection circuitry detects 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 runs off an internal configuration oscillator with a divisor that sets the maximum frequency.

Table 9–5 lists the minimum and maximum error detection frequencies.

Table 9–5. Minimum and Maximum Error Detection Frequencies for Cyclone IV Devices

Error Detection Frequency	Maximum Error Detection Frequency	Minimum Error Detection Frequency	Valid Divisors (2^n)
80 MHz/2^n	80 MHz	312.5 kHz	0, 1, 2, 3, 4, 5, 6, 7, 8

You can set a lower clock frequency by specifying a division factor in the Quartus II software (for more information, refer to "Software Support"). The divisor is a power of two (2), where *n* is between 0 and 8. The divisor ranges from one through 256. Refer to Equation 9–1.

Equation 9–1.

$$\text{ror detection frequency} = \frac{80\ \text{MH}}{2^n}$$

CRC calculation time depends on the device and the error detection clock frequency.

Table 9–6 lists the estimated time for each CRC calculation with minimum and maximum clock frequencies for Cyclone IV devices.

Table 9–6. CRC Calculation Time

Device		Minimum Time (ms) (1)	Maximum Time (s) (2)
Cyclone IV E	EP4CE6 (3)	5	2.29
	EP4CE10 (3)	5	2.29
	EP4CE15 (3)	7	3.17
	EP4CE22 (3)	9	4.51
	EP4CE30 (3)	15	7.48
	EP4CE40 (3)	15	7.48
	EP4CE55 (3)	23	11.77
	EP4CE75 (3)	31	15.81
	EP4CE115 (3)	45	22.67
Cyclone IV GX	EP4CGX15	6	2.93
	EP4CGX22	12	5.95
	EP4CGX30	12	5.95
		34 (4)	17.34 (4)
	EP4CGX50	34	17.34
	EP4CGX75	34	17.34
	EP4CGX110	62	31.27
	EP4CGX150	62	31.27

Notes to Table 9–6:

(1) The minimum time corresponds to the maximum error detection clock frequency and may vary with different processes, voltages, and temperatures (PVT).

(2) The maximum time corresponds to the minimum error detection clock frequency and may vary with different PVT.

(3) Only applicable for device with 1.2-V core voltage

(4) Only applicable for the F484 device package.

Software Support

Enabling the CRC error detection feature in the Quartus II software generates the CRC_ERROR output to the optional dual purpose CRC_ERROR pin.

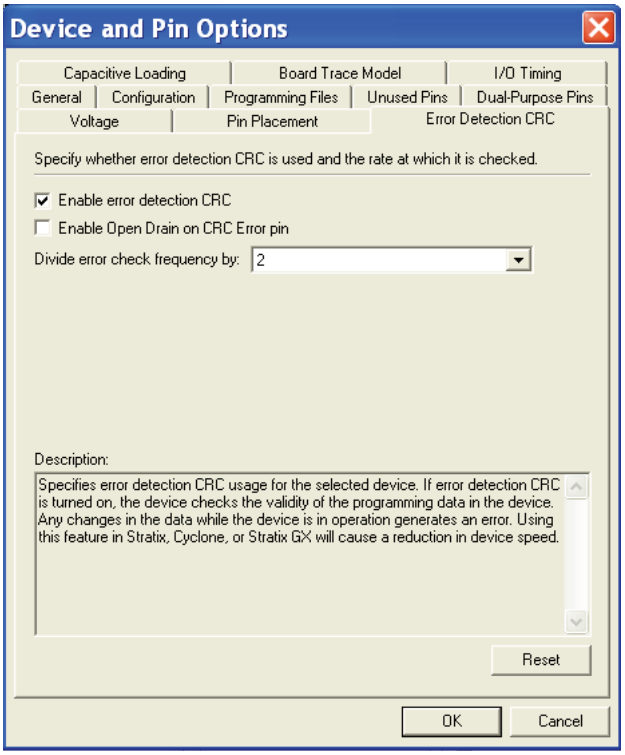
To enable the error detection feature using CRC, perform the following steps:

1. Open the Quartus II software and load a project using Cyclone IV devices.
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**. The **Device and Pin Options** dialog box appears as shown in Figure 9–2.
5. In the **Device and Pin Options** dialog box, click the **Error Detection CRC** tab.
6. Turn on **Enable error detection CRC**.
7. In the **Divide error check frequency by** box, enter a valid divisor as documented in Table 9–5 on page 9–5.

The divisor value divides 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**.

Figure 9–2. Enabling the Error Detection CRC Feature in the Quartus II Software

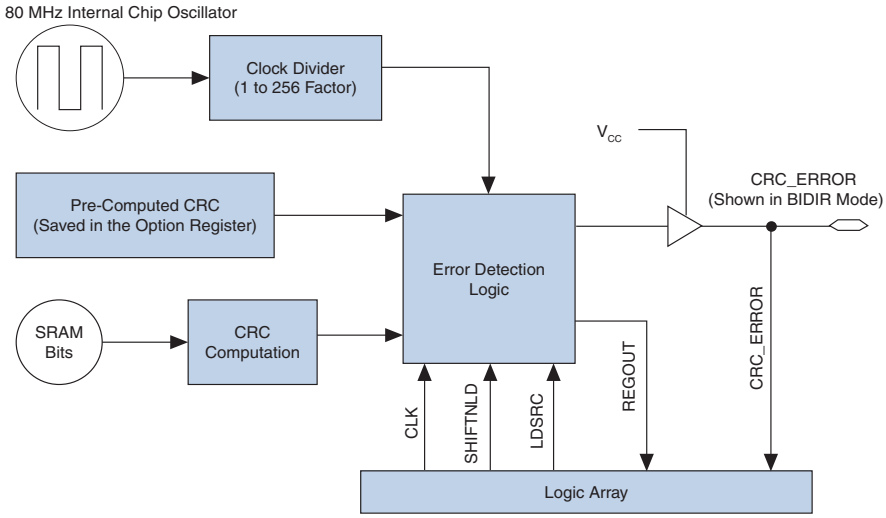


Accessing Error Detection Block Through User Logic

The error detection circuit stores the computed 32-bit CRC signature in a 32-bit register, which is read out by user logic from the core. The `cycloneiv_crcblock` primitive is a WYSIWYG component used to establish the interface from the user logic to the error detection circuit. The `cycloneiv_crcblock` primitive atom contains the input and output ports that must be included in the atom. To access the logic array, the `cycloneiv_crcblock` WYSIWYG atom must be inserted into your design.

Figure 9–3 shows the error detection block diagram in FPGA devices and shows the interface that the WYSIWYG atom enables in your design.

Figure 9–3. Error Detection Block Diagram



The user logic is affected by the soft error failure, so reading out the 32-bit CRC signature through the `regout` should not be relied upon to detect a soft error. You should rely on the `CRC_ERROR` output signal itself, because this `CRC_ERROR` output signal cannot be affected by a soft error.

To enable the `cycloneiv_crcblock` WYSIWYG atom, you must name the atom for each Cyclone IV device accordingly.

Example 9–1 shows an example of how to define the input and output ports of a WYSIWYG atom in a Cyclone IV device.

Example 9–1. Error Detection Block Diagram

```
cycloneiv_crcblock<crcblock_name>

(

.clk(<clock source>),

.shiftnld(<shiftnld source>),

.ldsrc(<ldsrc source>),

.crcerror(<crcerror out destination>),

.regout(<output destination>),

);
```

Table 9–7 lists the input and output ports that you must include in the atom.

Table 9–7. CRC Block Input and Output Ports

Port	Input/Output	Definition
<crcblock_name>	Input	Unique identifier for the CRC block, and represents any identifier name that is legal for the given description language (for example, Verilog HDL, VHDL, and 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. To do this, the `shiftnld` must be driven low for at least two clock cycles. This port is required.
.ldsrc (*<ldsrc source>*)	Input	This signal is an input into the error detection block. If `ldsrc=0`, the pre-computed CRC register is selected for loading into the 32-bit shift register at the rising edge of `clk` when `shiftnld=0`. If `ldsrc=1`, the signature register (result of the CRC calculation) is selected for loading into the shift register at the rising edge of `clk` when `shiftnld=0`. This port is ignored when `shiftnld=1`. This port is required.
.crcerror (*<crcerror indicator output>*)	Output	This signal is the output of the cell that is synchronized to the internal oscillator of the device (80-MHz internal oscillator) and not to the `clk` port. It asserts 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. You must connect this signal either to an output pin or a bidirectional pin. If it is connected to an output pin, you can only monitor the `CRC_ERROR` pin (the core cannot access this output). If the `CRC_ERROR` signal is used by core logic to read error detection logic, you must connect this signal to a `BIDIR` pin. The signal is fed to the core indirectly by feeding a `BIDIR` pin that has its output enable port connected to V_{CC} (see Figure 9–3 on page 9–8).
.regout (*<registered output>*)	Output	This signal is the output of the error detection shift register synchronized to the `clk` port to be read by core logic. It shifts one bit at each cycle, so you should clock the `clk` signal 31 cycles to read out the 32 bits of the shift register.

Recovering from CRC Errors

The system that the Altera FPGA resides in must control device reconfiguration. After detecting an error on the `CRC_ERROR` pin, strobing the `nCONFIG` low directs the system to perform the reconfiguration at a time when it is safe for the system to reconfigure the FPGA.

When the data bit is rewritten with the correct value by reconfiguring the device, the device functions correctly.

While soft errors are uncommon in Altera devices, certain high-reliability applications might require a design to account for these errors.

Document Revision History

Table 9–8 lists the revision history for this chapter.

Table 9–8. Document Revision History

Date	Version	Changes
May 2013	1.3	Updated “CRC_ERROR Pin Type” in Table 9–2.
October 2012	1.2	Updated Table 9–2.
February 2010	1.1	Updated for the Quartus II software version 9.1 SP1 release: ■ Updated “Configuration Error Detection” section. ■ Updated Table 9–6. ■ Added Cyclone IV E devices in Table 9–6.
November 2009	1.0	Initial release.