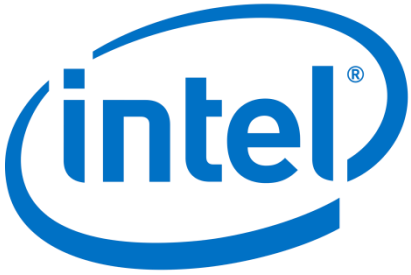

Software Design Example for Configuration via Avalon-ST (AVST) Scheme

Date: December 2019

Revision: 1.0

Contents

1 Introduction

The Avalon-ST configuration scheme is the fastest configuration scheme for Intel Stratix 10 FPGA, in which it replaces the FPP mode in previous device families. This scheme requires an external host to drive the configuration process. The external host can be a microprocessor or a hardware logic controller. The host is responsible for:

- Initiate the FPGA configuration
- Transfer the configuration data (or bitstream) from some external non-volatile memory (such as flash) to the FPGA
- Monitor the FPGA enter usermode successfully

Intel provides the hardware controller intellectual property (IP) for use with Avalon-ST scheme, namely the Intel FPGA Parallel Flash Loader II (PFL II). This IP is usually programmed to a CPLD device which connects with the FPGA. Such CPLD includes MAX II, MAX V or MAX 10 devices. To read more about the PFL II, refer to the Intel Stratix 10 Configuration User Guide.

You may use a general-purpose microprocessor as the Avalon-ST external host. The Avalon-ST signals on the FPGA end shall be connected to the general purpose I/O (GPIO) of the microprocessor. Since the microprocessor does not have hardware logic that can perform the configuration, the control and execution flow shall be implemented using software code.

In this design, we demonstrate how to configure the Stratix 10 FPGA using embedded software for microprocessor. To illustrate the example, we use the Nios II processor as the host to perform the Stratix 10 configuration. This design is based on the Intel Stratix 10 SoC FPGA development kit. The Nios II processor that we use is programmed onto the MAX10 device on the board, in which it serves as system controller for the development kit. Nios II reads the Stratix 10 configuration image from the flash and send to the Stratix 10 via the AVST x16 interface, as highlighted in the box of Figure 1.

This software example is written for Nios II processor; however, it should be portable to other microprocessor architecture by modifying the source codes.

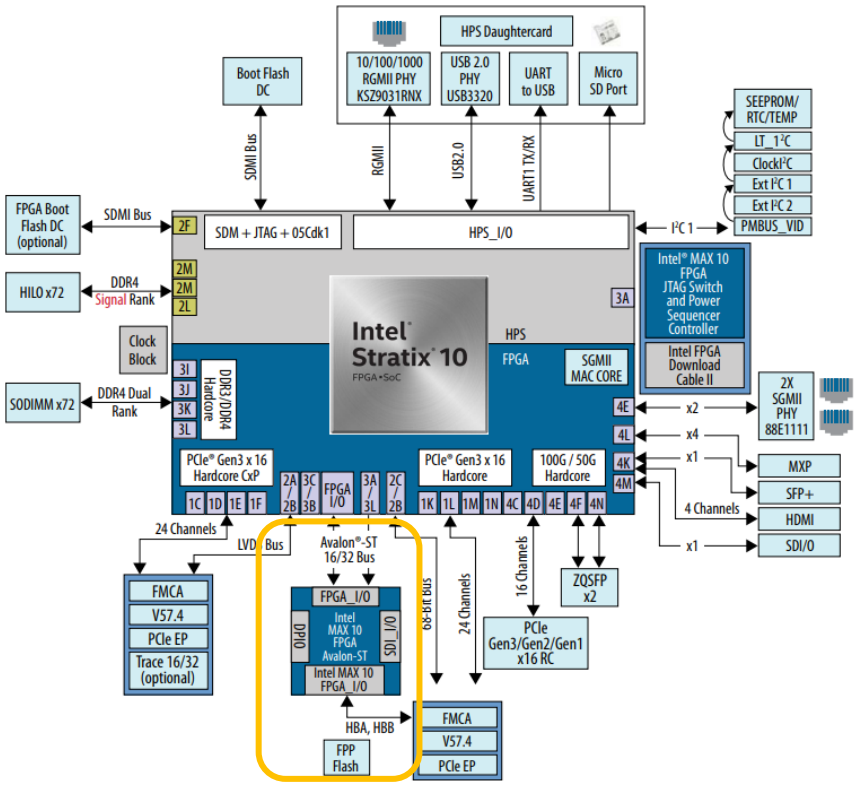

Figure 1. Stratix 10 SoC development kit block diagram

2 Prerequisite

You need to have knowledge in Avalon-ST configuration scheme as outline in Intel Stratix 10 Configuration User Guider, to be familiar with this design example. As this example is developed with Nios II processor, you may find more related information in the Nios II Software Developer Handbook.

Related information:

- Intel Stratix 10 Configuration User Guide
- Nios II Software Developer Handbook

3 Hardware and Software Requirements

3.1 Hardware Requirements

This design requires the following hardware:

- Stratix 10 SoC FPGA Development Kit Revision A (which has the CFI flash being mounted on board at U44 and U45)
- Micro USB cable for programming the device

3.2 Software Requirements

To program the CFI flash, download Nios II design to MAX10, and compile Nios II software code, use

- Quartus Prime version 19.1.0 Standard Edition and above

4 Block Diagram

Figure 2 shows the design example hardware block diagram. The Nios II processor subsystem consists of the on-chip memory and general purpose I/O. The software code resides on the memory and the I/O are used to interface with the flash memory and Stratix 10 Avalon-ST interface. This complete processor subsystem is programmed onto the MAX10.

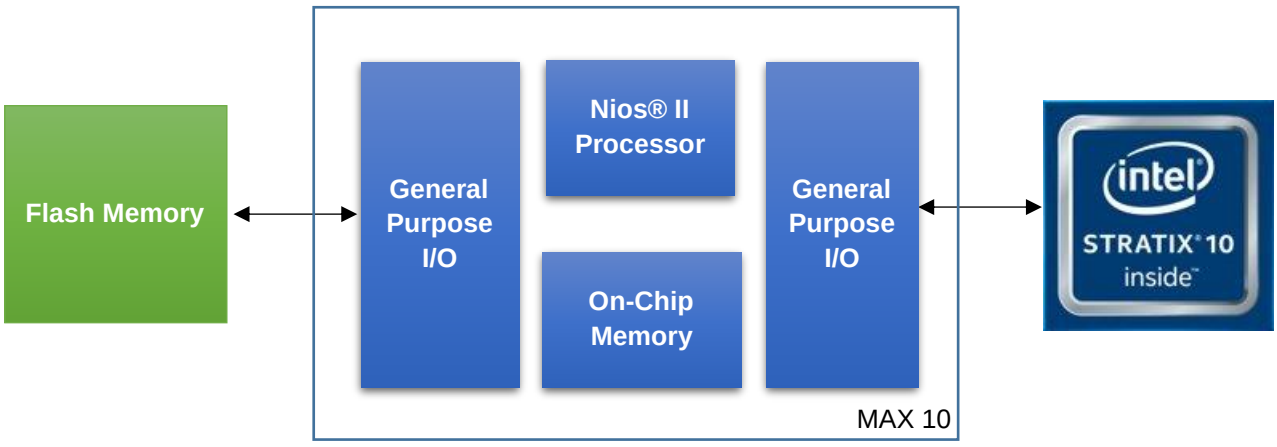

Figure 2: Design example block diagram

If you are migrating this example to another hardware platform, here are the requirements on the system:

1. Input-Output (I/O) pins to interface with the FPGA Configuration and Avalon-ST interface signals as shown in Table 1.

Signal Name	Direction (with reference to Stratix 10)	Description
nSTATUS	Output	Assert when configuration error happened
nCONFIG	Input	Input to start configuration
CONF_DONE	Output	Assert when configuration is completed
MSEL[2:0][1]	Input	Configuration mode selection
AVST_READY	Output	Assert when the FPGA Avalon-ST interface is ready to receive data
AVST_VALID	Input	Assert by the host to FPGA for indicating valid data cycle
AVST_DATA[31:0] or [15:0] or [7:0][2]	Input	Avalon-ST data bus
AVST_CLK	Input	Clock which is provided by the host to FPGA Avalon-ST interface. Require to clock all the Avalon-ST signals.

Table 1: FPGA Configuration and Avalon-ST Signals

Note to the table:
1 MSEL should be set to 000, 101 or 110 for Avalon-ST x32, x16 or x8 correspondingly.
2 AVST_DATA bus width equals to the Avalon-ST mode being chosen namely Avalon-ST x32, x16 or x8.

For more information, refer to Chapter 3.1 Avalon-ST Configuration of Intel Stratix 10 Configuration User Guide

2. Flash memory which is used to store the configuration bitstream file and is accessible by the microprocessor. You need flash programming tool to store the bitstream file into the flash memory.

5 FPGA Configuration Bitstream

You should use the raw binary file (RBF) with the Avalon-ST configuration scheme via external host such as microprocessor. You can program the RBF file into flash memory via the programmer that supports it. The external host retrieves the configuration data from the flash during the Avalon-ST configuration process.

You obtain the RBF file by converting your Stratix 10 project from SOF file to RBF. In this example, we use the Quartus Programmer to program the RBF into the CFI flash. See section 7.5 of this document for instructions of RBF conversion and flash programming.

6 Software Flow Diagram

Figure 3 shows the software flow diagram of the design example. This section highlights the software flow in general to perform the Avalon-ST configuration.

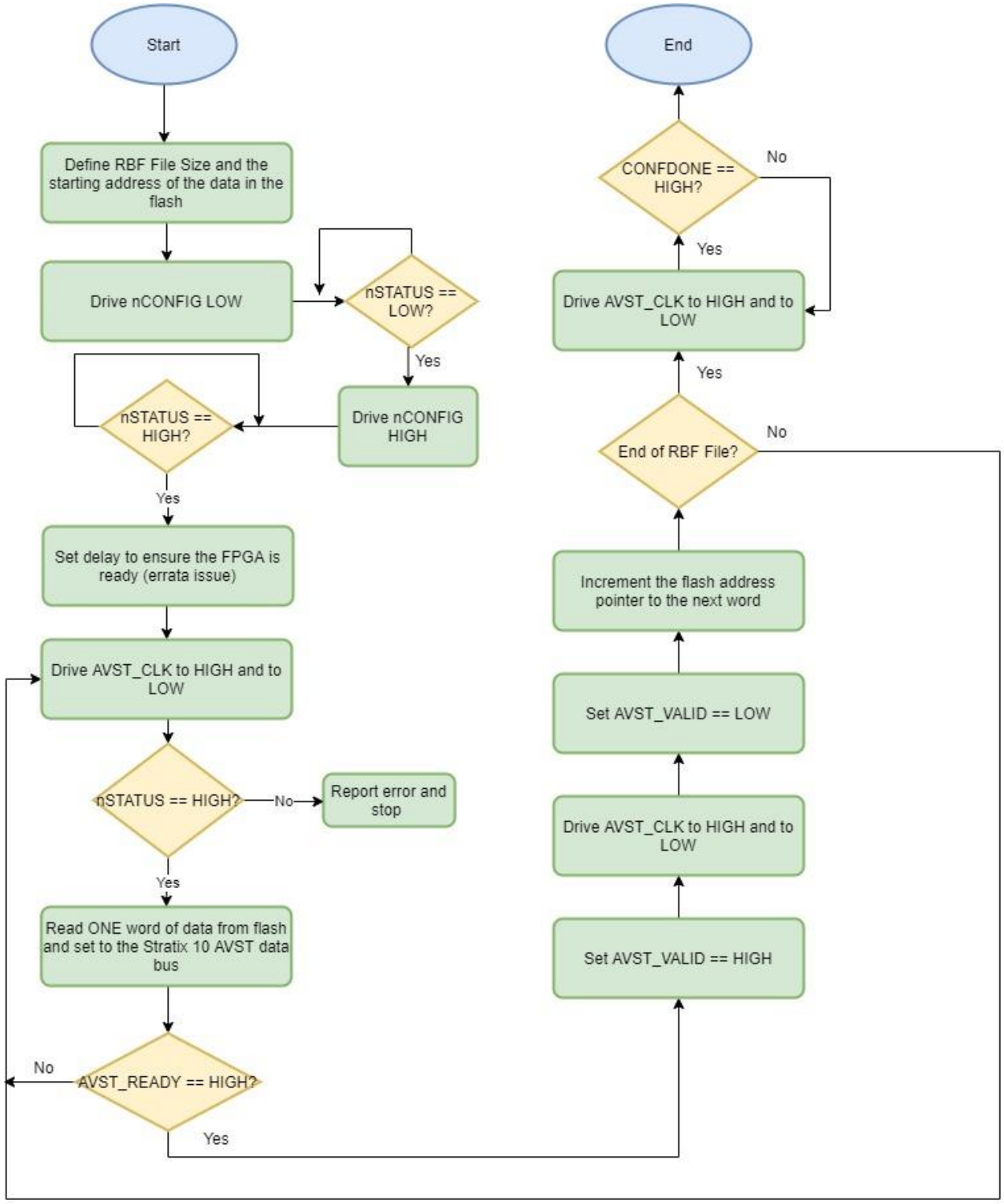

Figure 3: Avalon-ST Software Flow Diagram

The Avalon-ST configuration software example reads the whole bitstream in RBF from flash and send to the FPGA. Hence, the RBF file size in total number of bytes should be defined in the software. Likewise, the flash offset of the RBF first byte of data should be defined.

The configuration is initiated by driving the nCONFIG signal to LOW (0) and monitor the nSTATUS change to LOW. Next drive the nCONFIG to HIGH (1) and followed by monitoring the nSTATUS signal change to HIGH, otherwise keep waiting. There is an erratum for Stratix 10 on unexpected AVST_READY signal behavior after power-on-reset. After powering up the FPGA device power supplies in the proper sequence, the device asserts a Power-On-Reset (POR). When you drive the nCONFIG pin high, subsequently the nSTATUS pin goes high. If you use either of the Avalon-ST configuration scheme(s) (32/16/8 bits), you may notice an unexpected low pulse (20 to 100 µs) on AVST_READY pin. Hence, additional delay in software is used to mitigate this erratum and ensure the FPGA is truly ready. See the errata for more information.

The Avalon-ST clock is toggled (drive AVST_CLK HIGH and to LOW) and then check nSTATUS to ensure the FPGA is ready before we start. Next, retrieve the bitstream from flash and set it to the AVST_DATA bus. If the AVST_READY signal is LOW then the process is repeated.

Once the AVST_READY signal is HIGH, proceed to set the AVST_VALID to HIGH followed by toggling the AVST_CLK, and finally set the AVST_VALID to LOW. This whole process is to set the data on the AVST_DATA bus to be valid, so the FPGA will capture that. When this is done, the flash address pointer shall be incremented to continue with the next bitstream data. This entire process is repeated until the last data in the RBF file is sent to the FPGA.

Finally, the CONFDONE signal is monitored until it is HIGH which indicate the FPGA configuration is done and we can safely end the process. While the CONFDONE is LOW, you should continue toggle the AVST_CLK.

7 Using the Design Example

7.1 Install the Design Example Files

Before you start, download the design example from Design Store and restore the project to your <project_directory>.

1. quartus_sh --platform_install –package <*your directory*>/avst_config_sw_de.par
2. quartus_sh --platform -name avst_config_sw_de -output <*project_directory*>

7.2 Running the Design

This section describes the instructions to run the design example using the prebuilt binary files. The next sections cover the steps to re-compile both the hardware and software designs.

This design example demonstrates the FPGA configuration process. The design being configured is an LED blinker application, hence at the end of these instructions you should see the FPGA configuration is successful and the LEDs on the board are blinking.

1. Setup the Stratix 10 SoC board:
    a. Switch SW1: Set the second bit (M10B) to OFF, and the remaining bits all to ON
    b. Switch SW2: Set MSEL to ON-OFF-ON-ON (AVSTx16 configuration mode)
    c. Connect micro USB cable to J57
    d. Turn on board SW7
2. Launch Quartus Prime version 19.1.0 Standard Edition
    a. Click Tools -> Programmer

b. Click Auto Detact, select 10M50DA if prompted

3. Program the CFI flash with Stratix 10 configuration image. This step is only needed for once, or when you need re-program new configuration image into flash

    a. Right click on CFI_1Gb, and select Change File

    b. Browse to <project_directory>/prebuilt_binary and choose s10_led_hex.pof

    c. Check Program/Configure box

    d. Click Start. See Figure 4 for the Programmer setup.

    e. Wait for the CFI flash programming to complete. This will program the Stratix 10 configuration image into the flash.

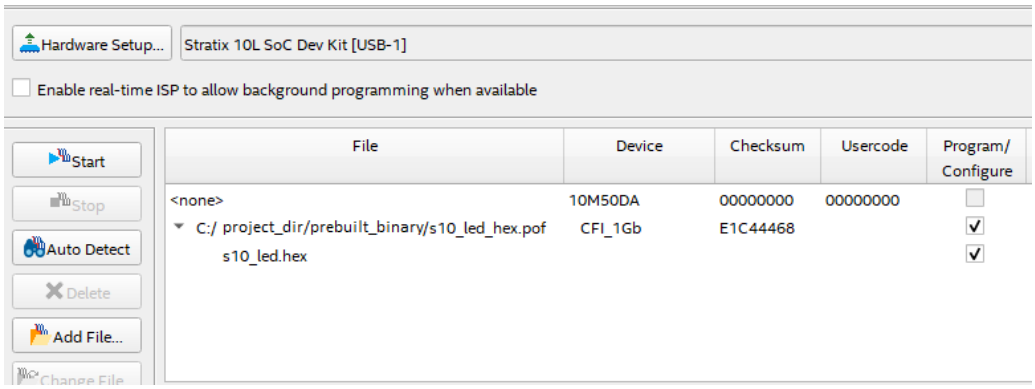

Figure 4: Programming CFI Flash with Stratix 10 Design Bitstream

4. Program the MAX10 with this design example

    a. Power off and on the board (if you continue from step 3)

    b. Make sure the Program/Configure box for CFI_1Gb is unchecked

    c. Right click on 10M50DA, and select Change File

    d. Browse to <project_directory>/prebuilt_binary and choose max10_system.sof

    e. Check Program/Configure box

    f. Click Start. Figure 5 shows the setup in Programmer.

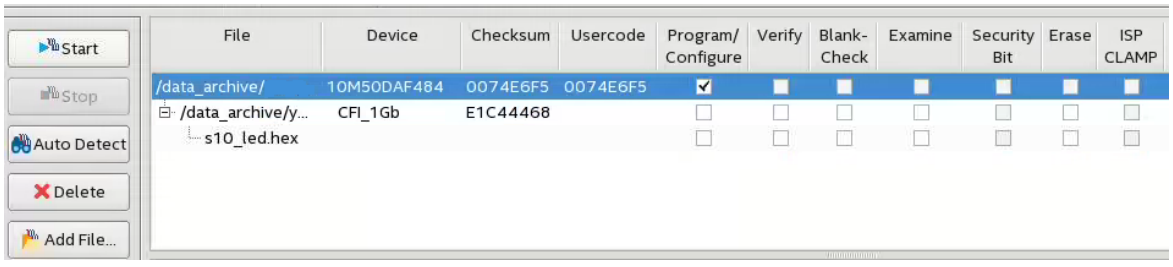

Figure 5: Program MAX10 with the Hardware Design

5. Launch Nios II software

    a. Launch Nios II Command Shell (use the Windows search to find the shell executable)

    b. Change directory to <project_directory>/prebuilt_binary

    c. Type nios2-download -g nios_software.elf && nios2-terminal

    This will download the NIos II software and open the JTAG UART terminal.

    You should see the following messages in the shell upon configuration successful.

```
$ nios2-download -g nios_software.elf && nios2-terminal
Using cable "Stratix 10L SoC Dev Kit [USB-1]", device 1, instance 0x00
Pausing target processor: OK
Initializing CPU cache (if present)
OK
Downloaded 30KB in 0.0s
Verified OK
Starting processor at address 0x01020238
nios2-terminal: connected to hardware target using JTAG UART on cable
nios2-terminal: "Stratix 10L SoC Dev Kit [USB-1]", device 1, instance 0
nios2-terminal: (Use the IDE stop button or Ctrl-C to terminate)

Start FPGA Configuration
CONF_DONE is high
AVST Configuration completed successfully
```

On the development kit, you should see LEDs (D21, D23, D25, D27) under USER_FPGA blinking.

7.3 Hardware Design Compilation

This section describes the instructions to re-compile the hardware design in Quartus Prime. The focus is on the Nios II processor design (which target the MAX10).

1. Launch Quartus Prime version 19.1.0 Standard Edition
2. Click File -> Open Project and navigate to <project_directory>/ and open top.qpf
3. Click Processing -> Start Compilation to start the compilation
4. You find the re-compile SOF file in <project_directory>/output_files directory
5. Optional step:

    a. To view the Nios II subsystem, click Tools -> Platform Designer

    b. In Platform Designer, browse to <project_directory>/ and open nios_avst_cfi.qsys

7.4 Software Design Compilation

This section describes the instructions to re-compile the Nios II software program into executable (.elf file).

1. Launch the Nios II Command Shell

2. In the shell, change directory to <project_directory>/software/
3. Change the script files to executable permission:
    a. chmod +x nios_software/create-this-app
    b. chmod +x nios_software_bsp/create-this-bsp
4. Change directory to <project_directory>/software/nios_software/
5. Run the create-this-app script by typing ./create-this-app
6. You find the executable file nios_software.elf at <project_directory>/software/nios_software/
7. Repeat the steps in section 7.2 to program the MAX10 and run the Nios II software

7.5 Bitstream Files Conversion

This section discusses how to convert your Stratix 10 design into the RBF file should you wish to replace the LED blinker image with your design.

It is important to ensure that your Stratix 10 Quartus project is set to AVST as your configuration scheme:

1. Click Assignments -> Device
2. Select Device and Pin Options
3. In the Configuration category, choose AVST x16 (or AVST x8/x32) under “Configuration Scheme” drop-down.

When you successfully compile your Quartus project, you should observe SOF file in your project/output_files directory. The following steps illustrate how to convert SOF to RBF using command line tools.

1. Launch the Nios II Command Shell
2. In the shell, navigate to your Stratix 10 project/output_files directory
3. In the shell, type quartus_pfg -c <your sof file name> <output rbf name>
4. You find the rbf file in the same directory

You use the obtained rbf file from above and program to the flash memory via the programming tool of your choice. In this example, we are using the Quartus Programmer tool to program the CFI flash on the Stratix 10 development kit.

The Quartus Programmer tool does not program RBF file directly to CFI flash, and the following steps illustrate how to further convert the RBF file to POF file. **Note that this is not needed if your flash programming tool is able to program the raw binary (RBF) directly onto your flash.**

1. Convert the RBF to Intel HEX file. You may use software tools like srec_cat (from SRecord package) for this. The following is the actual command for srec_cat
    a. srec_cat.exe <rbf file> -binary -output <Intel hex file> -Intel
2. Once you have the Intel HEX file, go to Quartus File -> Programming File Generator

3. In the Output Files tab, choose your output directory and the output file name. Make sure to check Programmer Object File (.pof) box. The complete setup is shown in Figure 6.
4. Next, on to the Input Files tab, click Add Raw Data. Browse to the HEX file that you created in Step 1.
5. Next, on to the Configuration Device tab, click Add Device. Select your flash memory from the list of devices. In this example, we are using the CFI 1Gb flash. Click OK to finish.
6. Select the device that you added, click Add Partition. You may leave the partition name as default value. Select the Input file as the HEX file you are using. The complete setup is shown in Figure 7.
7. Click Generate when you are done. You observe the output POF file generated in your directory.
8. Repeat Step 2 in section 7.2 to program the generated POF onto the CFI Flash via Quartus Programmer.

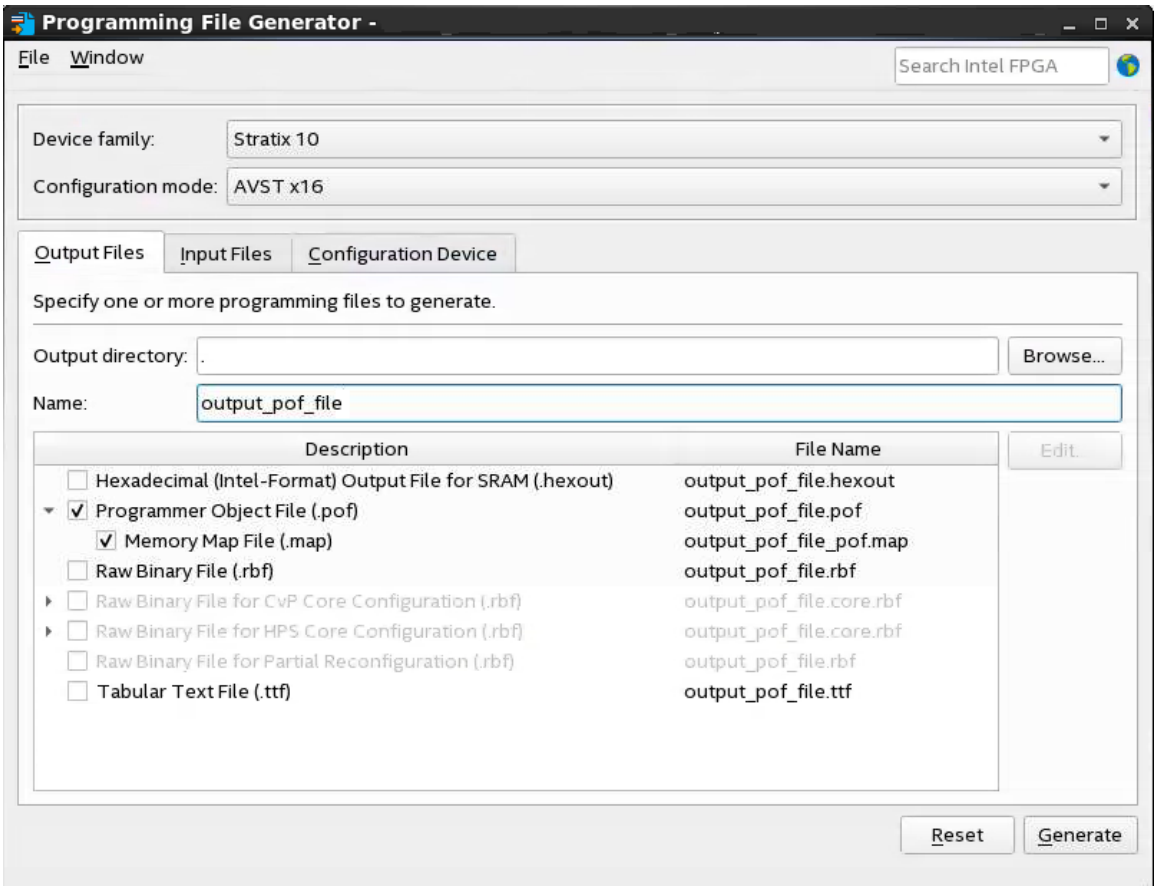

Figure 6: Output Files Setting in Programming File Generator

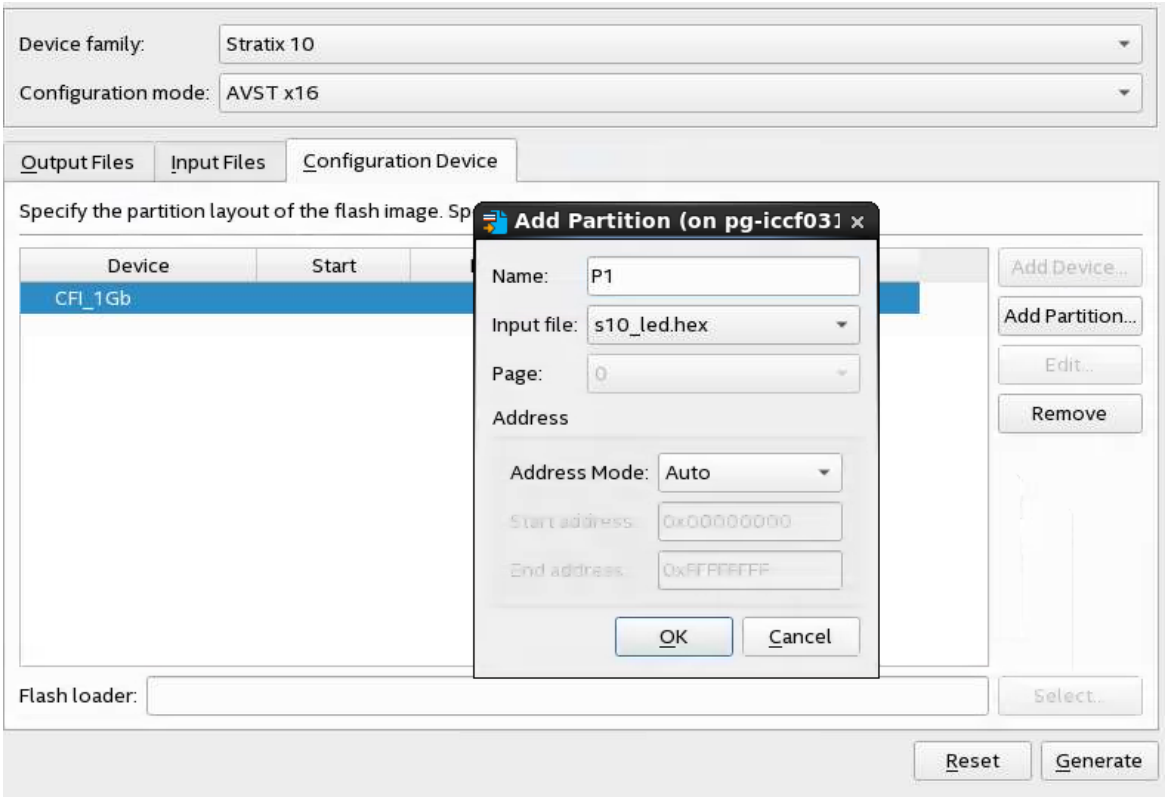

Figure 7: Create partition for flash device

8 Revision History

Revision	Description
1.0	Initial version