

Nios II Processor Booting from CFI Flash (Cyclone V GX FPGA Development Kit)

Date: 31 July 2017

Revision: 1.0

©2017 Intel Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, INTEL, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Intel Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Intel warrants performance of its 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.

Table of Contents

1. Introduction	3
2. Prerequisite.....	3
3. Requirements.....	3
4. Design Example Files.....	3
5. Block Diagram	4
6. Steps to Build a Bootable System	5
6.1 Quartus Setting	5
6.2 Design.....	5
6.2.1 Nios II Processor Qsys Setting.....	6
6.2.2 Generic Tri-State Controller Qsys Setting	7
6.3 BSP Editor Settings.....	11
6.4 Application	13
6.5 Generate programming files.....	14
6.6 Programming the CFI flash.....	16
6.7 Run the Design	19
7. Revision History	20

1. Introduction

This design example demonstrates step-by-step instructions on how to boot a simple Hello World Nios II application from CFI flash using [Cyclone V GX FPGA Development Kit](#). The boot option used in this design example is “boot copier” method, however, you may change the [BSP editor settings](#) to enable “execute-in-place” method.

2. Prerequisite

You are required to have knowledge in instantiating and developing a system with a Nios II processor. Intel recommends that you go through the online tutorials and training materials provided in [Intel FPGA Training](#) before using this design example.

Related information:

- [Generic Nios II Booting Methods User Guide](#)
- [Embedded Design Handbook](#)
- [Avalon Tri-State Conduit Components User Guide](#)
- [Parallel Flash Loader IP Core User Guide](#)

3. Requirements

The following are the hardware and software requirements for the design example:

- Cyclone V GX FPGA Development Kit & [Kit installation](#)
- Quartus Prime version 16.1
- USB cable for USB-Blaster II JTAG

4. Design Example Files

The master files are provided with this design example. You may proceed to reading [Programming the CFI flash](#) section and [Run the Design](#) section to program the CFI flash and run the design without any recompilation or programming file generation.

File Name	Description
master_image\output_file.pof	Quartus Prime programming file for CFI flash
master_image\ext_flash.hex master_image\cvgx_cfi_booting.sof	Hardware image file (.sof) and Nios II application image file (.hex) that are used to generate CFI flash programming file
software\cvgx_cfi_hello_world.zip software\cvgx_cfi_hello_world_bsp.zip	Hello world software code and BSP for the Nios II processor

5. Block Diagram

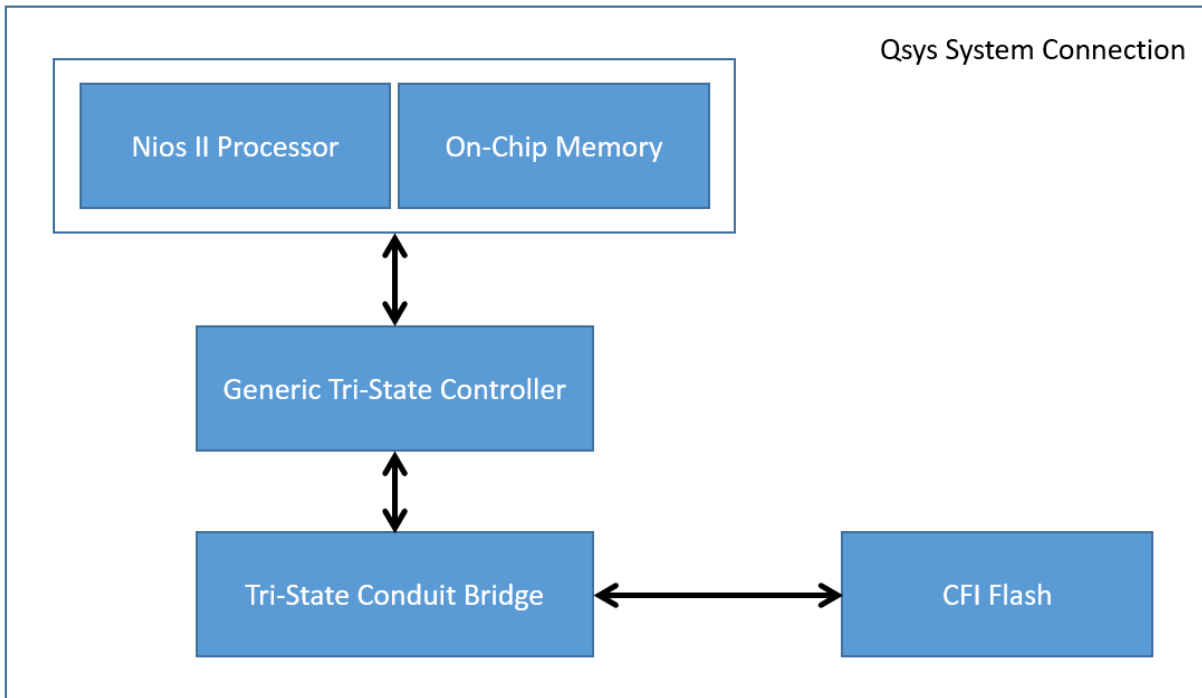


Figure 1: Qsys System Connection

Figure 1 shows the main IP components required for building a Nios II system booting from CFI flash. Details of the connection between each IP blocks can be found in the included Qsys file for this design example.

6. Steps to Build a Bootable System

The following sections describe the steps to build a bootable system for a:

- Nios II processor application executing in place from CFI flash
- Nios II processor application copied from CFI flash to RAM using a boot copier

6.1 Quartus Setting

The MSEL pins in Cyclone V GX development kit are defaulted to FPPx16 configuration scheme. In Quartus Prime software, go to **Assignments** → **Device** → **Device and Pin Options** → **Configuration**, ensure the *Configuration scheme* is set to **Passive Parallel x16**.

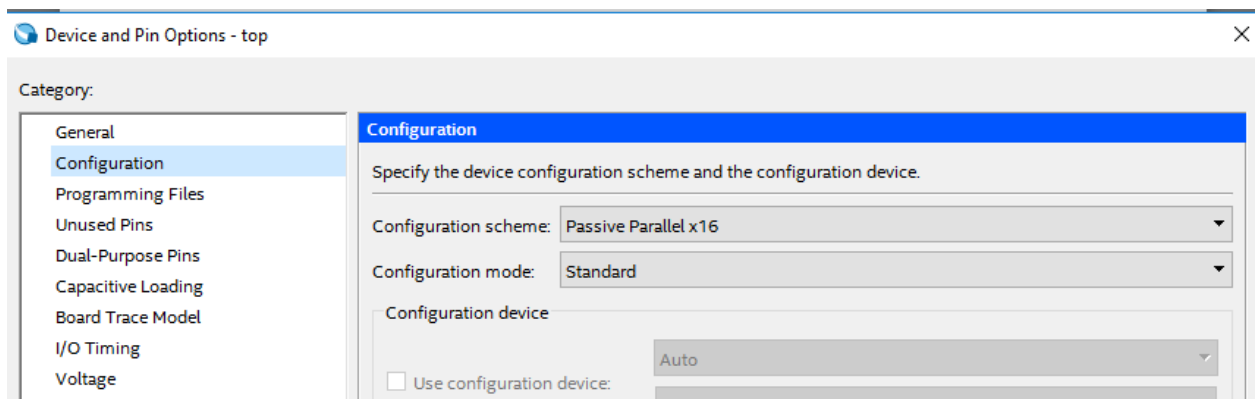


Figure 2: Configuration Scheme Setting

6.2 Design

The Qsys file in this design example includes all main components that are required for a Nios II system booting from CFI flash, essentially the *Generic Tri-State Controller* and the *Tri-State Conduit Bridge* that connects the CFI flash to the Nios II processor.

Connections	Name	Description	Export	Clock	Base	End	IRQ	
	<ul style="list-style-type: none"> clk_in_50 clk_in_reset clk clk_reset 	<ul style="list-style-type: none"> Clock Source Clock Input Reset Input Clock Output Reset Output 	<ul style="list-style-type: none"> clk_in_50 reset Double-click to export Double-click to export 	<ul style="list-style-type: none"> exported clk_in_50 				
	Nios II Processor							
	<ul style="list-style-type: none"> clk reset data_master instruction_master irq debug_reset_request debug_mem_slave custom_instruction_m... 	<ul style="list-style-type: none"> Clock Input Reset Input Avalon Memory Mapped Master Avalon Memory Mapped Master Interrupt Receiver Reset Output Avalon Memory Mapped Slave Custom Instruction Master 	<ul style="list-style-type: none"> Double-click to export Double-click to export Double-click to export Double-click to export Double-click to export Double-click to export Double-click to export 	<ul style="list-style-type: none"> clk_in_50 [clk] [clk] [clk] [clk] [clk] [clk] 			<ul style="list-style-type: none"> IRQ 0 IRQ 31 	
	On-Chip Memory (RAM or ROM)							
	<ul style="list-style-type: none"> clk1 s1 reset1 	<ul style="list-style-type: none"> Clock Input Avalon Memory Mapped Slave Reset Input 	<ul style="list-style-type: none"> Double-click to export Double-click to export Double-click to export 	<ul style="list-style-type: none"> clk_in_50 [clk1] [clk1] 	<ul style="list-style-type: none"> # 0x0408_0000 	<ul style="list-style-type: none"> 0x040f_ffff 		
	Tri-State Conduit Bridge							
	<ul style="list-style-type: none"> clk reset tcs out 	<ul style="list-style-type: none"> Tri-State Conduit Bridge Clock Input Reset Input Tristate Conduit Slave Conduit 	<ul style="list-style-type: none"> Double-click to export Double-click to export Double-click to export Double-click to export 	<ul style="list-style-type: none"> clk_in_50 [clk] [clk] 				
	Generic Tri-State Controller							
	<ul style="list-style-type: none"> clk reset uas tcm 	<ul style="list-style-type: none"> Clock Input Reset Input Avalon Memory Mapped Slave Tristate Conduit Master 	<ul style="list-style-type: none"> Double-click to export Double-click to export Double-click to export Double-click to export 	<ul style="list-style-type: none"> clk_in_50 [clk] [clk] [clk] 	<ul style="list-style-type: none"> # 0x0000_0000 	<ul style="list-style-type: none"> 0x03ff_ffff 		
	System ID Peripheral							
	<ul style="list-style-type: none"> clk reset control_slave 	<ul style="list-style-type: none"> Clock Input Reset Input Avalon Memory Mapped Slave 	<ul style="list-style-type: none"> Double-click to export Double-click to export Double-click to export 	<ul style="list-style-type: none"> clk_in_50 [clk] [clk] 	<ul style="list-style-type: none"> # 0x0410_1008 	<ul style="list-style-type: none"> 0x0410_100f 		
	JTAG UART							
	<ul style="list-style-type: none"> clk reset avalon_jtag_slave irq 	<ul style="list-style-type: none"> JTAG UART Clock Input Reset Input Avalon Memory Mapped Slave Interrupt Sender 	<ul style="list-style-type: none"> Double-click to export Double-click to export Double-click to export Double-click to export 	<ul style="list-style-type: none"> clk_in_50 [clk] [clk] [clk] 	<ul style="list-style-type: none"> # 0x0410_1000 	<ul style="list-style-type: none"> 0x0410_1007 		

Figure 3: Nios II Qsys System Connections

6.2.1 Nios II Processor Qsys Setting

1. In the Nios II Processor parameter editor, set the *Reset vector memory* to CFI flash (ext_flash.uas) and *Exception vector memory* to On-Chip Memory (onchip_ram.s1).
2. Set the *Reset vector offset* to 0x01bc0000. The reset vector must be the base address of your application. In this example, it is 0x01bc0000.

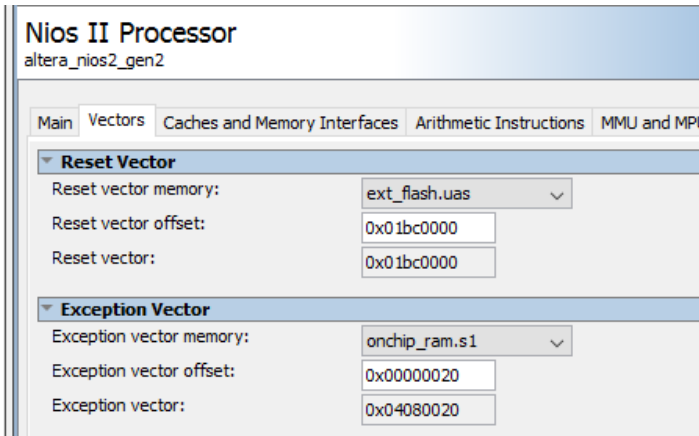


Figure 4: Nios II Processor Vectors Configuration

Note: Reset vector memory determines where the reset code (boot loader) resides, and Reset vector offset determines the location of the reset vector (reset address).

6.2.2 Generic Tri-State Controller Qsys Setting

The Generic Tri-State Controller provides preset configurations for many commonly used external devices. If you select one of the preset configurations, all the parameters are automatically assigned to the correct values. In case your CFI flash is not in the preset library, you can use the parameter editor to specify the required settings accordingly.

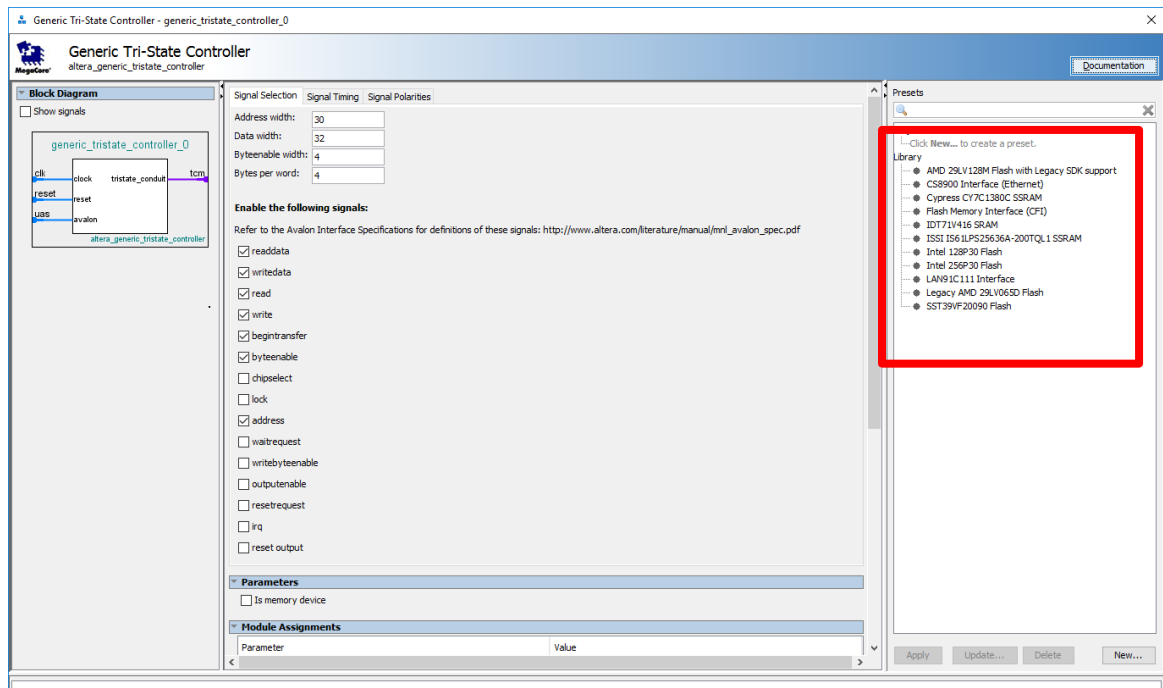


Figure 5: Generic Tri-State Controller Configuration Window

The following snapshots show the configuration setting for the CFI flash used in this development kit.

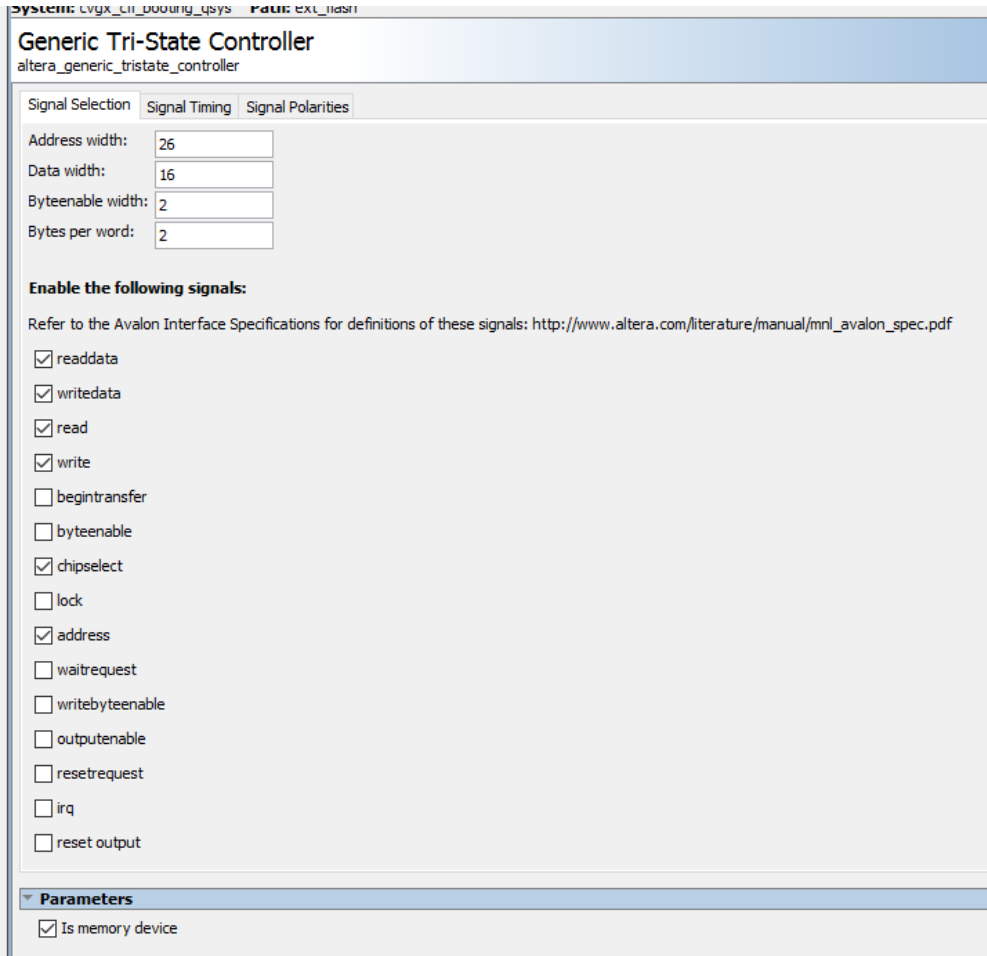


Figure 6: Signal Selection Part 1

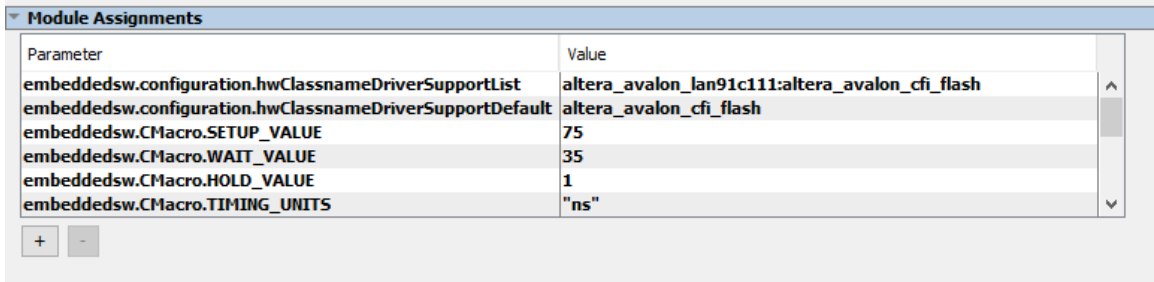


Figure 7: Signal Selection Part 2

Module Assignments	
Parameter	Value
embeddedsw.CMacro.SIZE	67108864u
embeddedsw.memoryInfo.MEM_INIT_DATA_WIDTH	16
embeddedsw.memoryInfo.HAS_BYTE_LANE	1
embeddedsw.memoryInfo.IS_FLASH	1
embeddedsw.memoryInfo.GENERATE_DAT_SYM	1
embeddedsw.memoryInfo.GENERATE_FLASH	1

Figure 8: Signal Selection Part 3

Module Assignments	
Parameter	Value
embeddedsw.memoryInfo.HAS_BYTE_LANE	1
embeddedsw.memoryInfo.IS_FLASH	1
embeddedsw.memoryInfo.GENERATE_DAT_SYM	1
embeddedsw.memoryInfo.GENERATE_FLASH	1
embeddedsw.memoryInfo.DAT_SYM_INSTALL_DIR	SIM_DIR
embeddedsw.memoryInfo.FLASH_INSTALL_DIR	APP_DIR

Parameters

Use the module assignments to identify your components to downstream embedded software tools. A value of 1 identifies the parameter as true, a value of 0 identifies it as false.

Note: For memory devices, the module assignment **embeddedsw.CMacro.SIZE = Memory Size in Bytes** must be defined and it must equal the size of the memory. Value should be an unsigned integer.

Avalon Connection Point Assignments	
Parameter	Value
embeddedsw.configuration.isFlash	1
embeddedsw.configuration.isMemoryDevice	1
embeddedsw.configuration.isNonVolatileStorage	1

Figure 9: Signal Selection Part 4

Generic Tri-State Controller

altera_generic_tristate_controller

Signal Selection | **Signal Timing** | Signal Polarities

Read wait time:

Write wait time:

Setup time:

Data hold time:

Maximum pending read transactions:

Turnaround time:

Timing units: ▾

Read latency:

Chipselect through read latency

Figure 10: Signal Timing

Generic Tri-State Controller

altera_generic_tristate_controller

Signal Selection | Signal Timing | **Signal Polarities**

Enable active low polarity on the following signals:

read

lock

write

chipselect

byteenable

outputenable

writebyteenable

waitrequest

begintransfer

resetrequest

irq

reset output

Figure 11: Signal Polarities

6.3 BSP Editor Settings

There are two boot options for Nios II processor booting from CFI flash - Nios II application execute-in-place from CFI flash and Nios II application copied from CFI flash to RAM using boot copier. Refer to table below for BSP editor settings in Nios II SBT for each boot option:

Boot Option	BSP Editor Setting: Linker Script	BSP Editor Setting: Settings.Advanced.hal.linker
Nios II processor application execute-in-place from CFI flash	<ul style="list-style-type: none"> Set .text Linker Section to CFI flash Set other Linker Sections (.heap, .rwdata, .rodata, .bss, .stack) to OCRAM/ External RAM 	<p>If the exception vector memory is set to OCRAM/ External RAM, enable the following settings:</p> <ul style="list-style-type: none"> allow_code_at_reset enable_alt_load enable_alt_load_copy_rodata enable_alt_load_copy_rwdata enable_alt_load_copy_exceptions <p>If the exception vector memory is set to CFI flash, enable the following settings:</p> <ul style="list-style-type: none"> allow_code_at_reset enable_alt_load enable_alt_load_copy_rodata enable_alt_load_copy_rwdata
Nios II processor application copied from CFI flash to RAM using boot copier	All Linker Sections are set to OCRAM/ External RAM	All settings are left unchecked

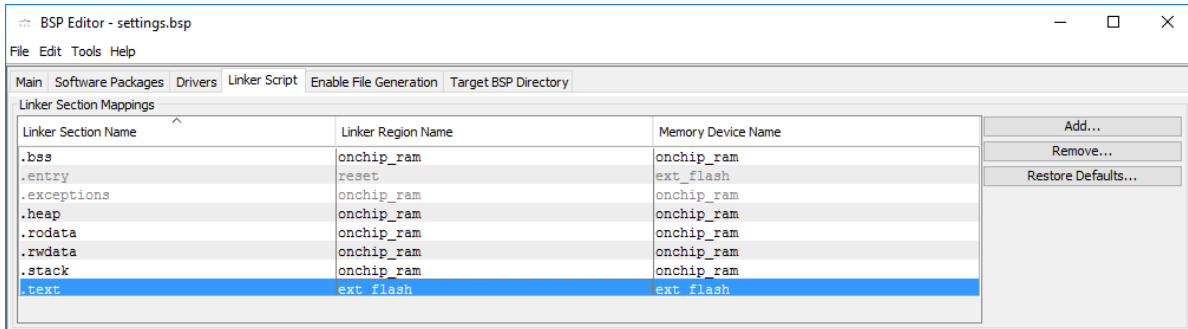


Figure 12: Linker Script for Execute-In-Place Method

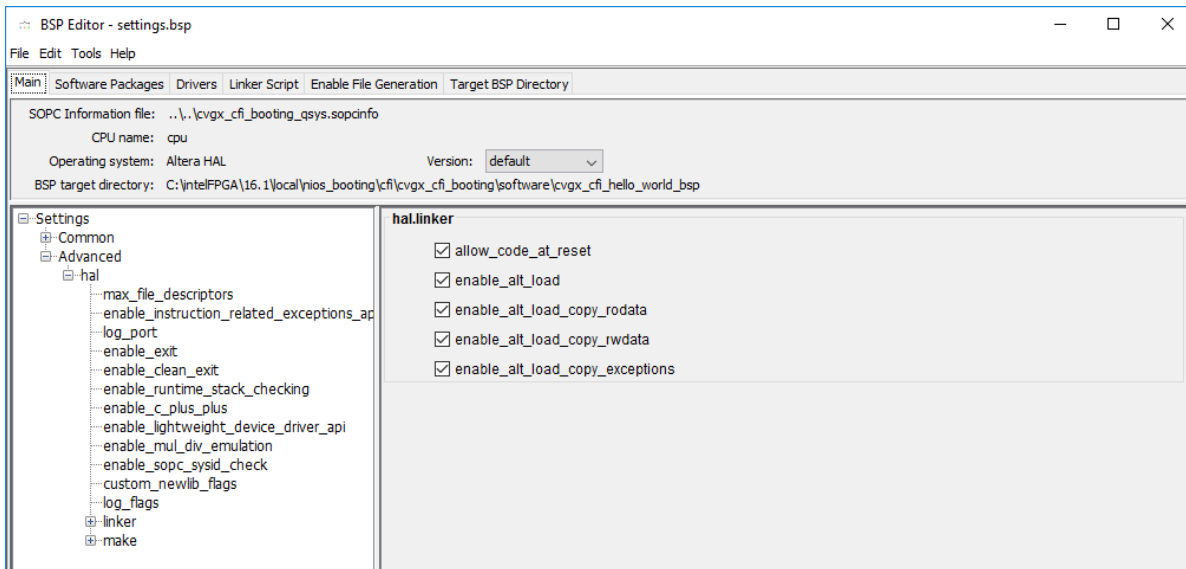


Figure 13: Settings.Advanced.hal.linker for Execute-In-Place Method if Exception vector is set to OCRAM/ External RAM

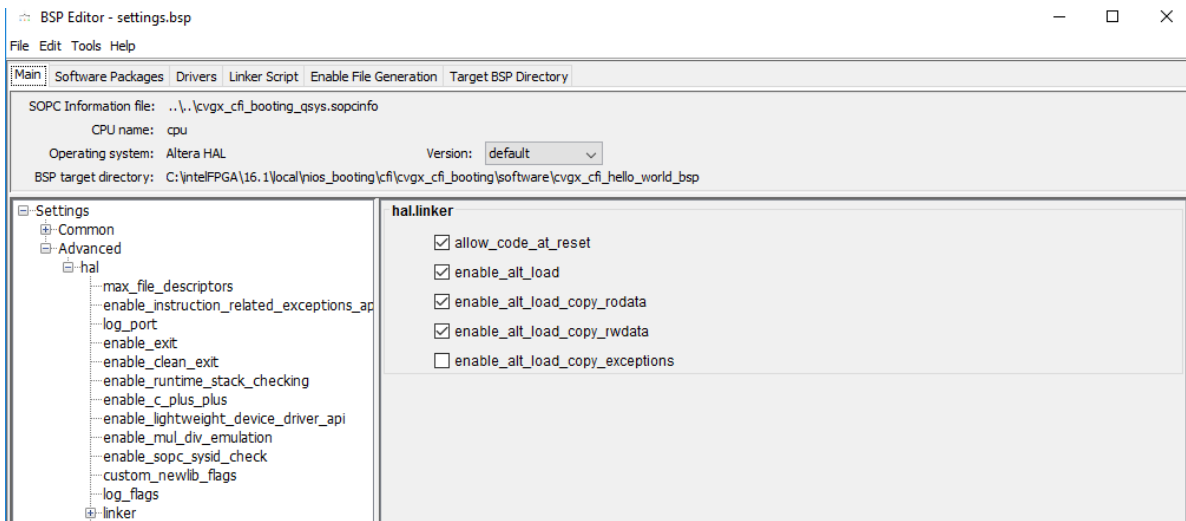


Figure 14: Settings.Advanced.hal.linker for Execute-In-Place Method if Exception vector is set to CFI flash

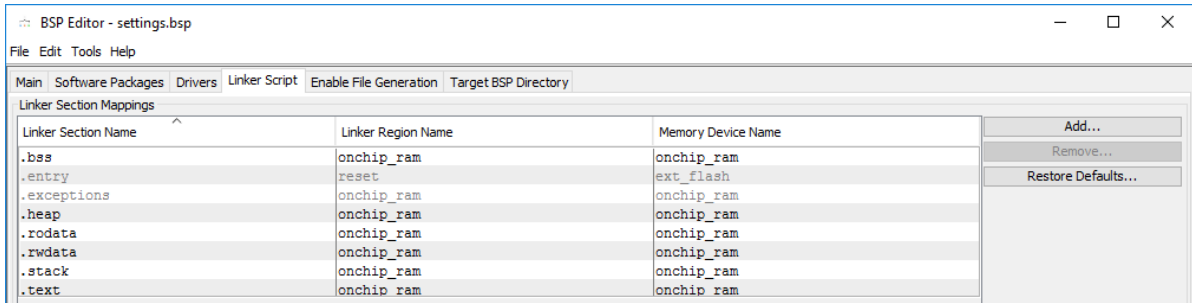


Figure 15: Linker Script for Boot Copier Method

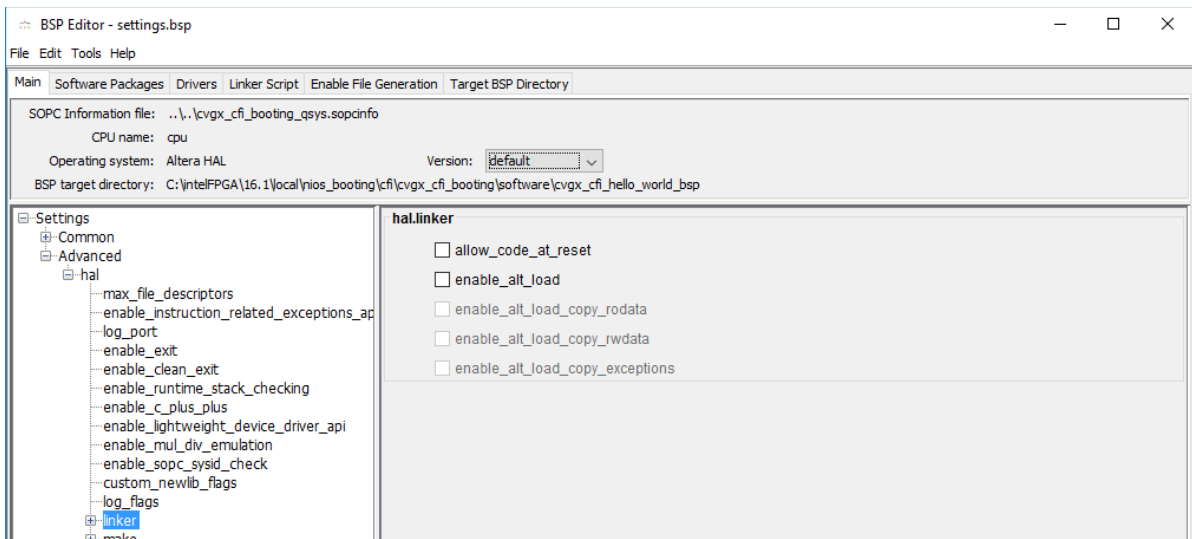


Figure 16: Settings.ADVANCED.hal.linker for Boot Copier Method

6.4 Application

This design example includes the Hello World software code and the BSP. You may also create your own Nios II application using the Nios II SBT.

To build the Nios II application image file:

1. After creating the Nios II application, right click on the project in **Project Explorer** and select **Build Project**. A *.elf file is created under the project folder.
2. Right click on the project in Project Explorer and select **Make Targets** → **Build....** Select **mem_init_generate** and click **Build**. A *.flash file is created under the project folder.

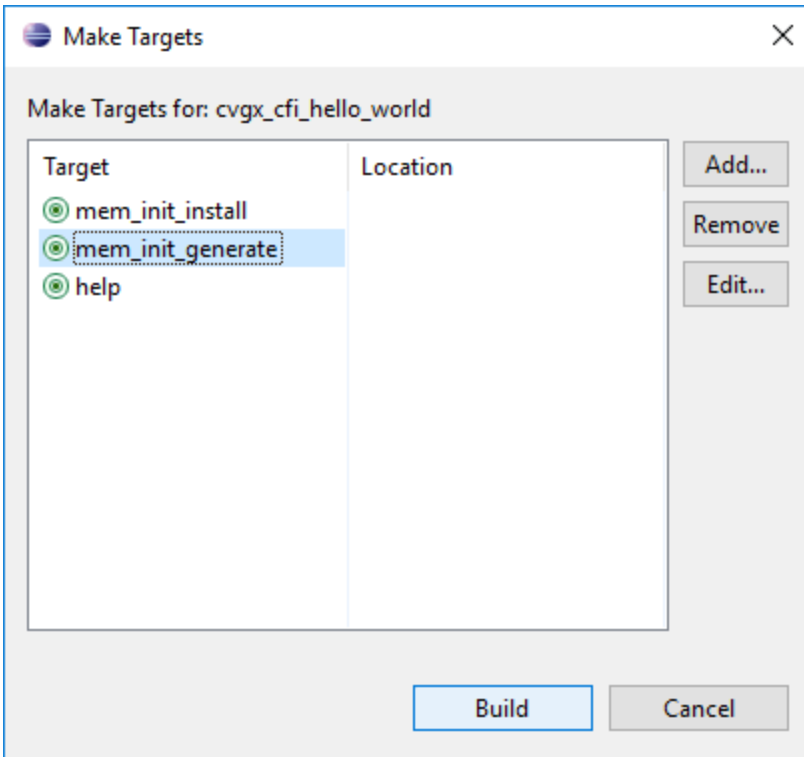


Figure 17: Make Targets Window

3. Launch **Nios II Command Shell**. Use *nios2-elf-objcopy* utility to convert **.flash** file to **HEX** file.

```

C:\cygdrive/c/intelFPGA/16.1/local/nios_booting/cfi/cvgx_cfi_booting/software/cvgx_cfi_hello_world
-----
Altera Nios2 Command Shell [GCC 4]
Version 16.1, Build 196
-----
ngchunhs@ngchunhs-MOBL /cygdrive/c/intelFPGA/16.1/local/nios_booting/cfi/cvgx_cfi_booting/software/cvgx_
$ nios2-elf-objcopy --input-target srec --output-target ihex ext_flash.flash ext_flash.hex --verbose
copy from `ext_flash.flash' [srec] to `ext_flash.hex' [ihex]

```

Figure 18: Nios II Command Shell Window

6.5 Generate programming files

Fast Passive Parallel (FPP) is the configuration scheme used in this design example. In Cyclone V GX development kit, the on board CPLD (MAX V) is used to control the FPGA configuration process. You may refer to the factory MAX V design in *<kit installation directory>/cycloneVGX_5cgxfc7df31es_fpga/examples/max5*, after completing the kit installation on your PC. This design example uses the factory MAX V design from development kit, thus the CFI flash

memory map must not be altered. You need to just focus on the factory software, factory hardware and PFL option bits.

Block Description	KB Size	Address Range
Unused	128	0x03FE.0000 - 03FF.FFFF
User software	28,800	0x023C.0000 - 03FD.FFFF
Factory software	8192	0x01BC.0000 - 023B.FFFF
zipfs (html, web content)	4096	0x017C.0000 - 01BB.FFFF
User hardware 2	8064	0x00FE.0000 - 017B.FFFF
User hardware 1	8064	0x0080.0000 - 00FD.FFFF
Factory hardware	8064	0x0002.0000 - 007F.FFFF
PFL option bits	32	0x0001.8000 - 0001.FFFF
Board information	32	0x0001.0000 - 0001.7FFF
Ethernet option bits	32	0x0000.8000 - 000.FFFF
User design reset vector	32	0x0000.0000 - 000.7FFF

Figure 19: Default CFI Flash Memory Map

To generate the programming file:

1. Open the **Convert Programming Files...** from **File** menu in Quartus Prime software.
2. Under **Output programming file** section, set the following items:
 - a. **Programming file type:** Programmer Object File (.pof)
 - b. **Configuration device:** CFI_512Mb
 - c. **Mode:** Passive Parallel x 16
 - d. **File name:** You may select your preferred path and file name for the output file (.pof).
By default, this is generated under your project root directory.
3. Under **Input files to convert** section, follow these steps:
 - a. For hardware image file:
 - i. Select **Page_0** of **SOF Data** and click **Properties**.
 - ii. Set the **Address mode for selected pages** to **Start**.
 - iii. Set the **Start address** to **0x020000** (this is to replace Factory hardware) and click **OK**.
 - iv. Click **Add File...** and open **master_image\cvgx_cfi_booting.sof**.
 - v. Select the added SOF file and click **Properties**.
 - vi. Check the **Compression** checkbox and click **OK**.
 - b. For Nios II application image file:
 - i. Click **Add Hex Data**.
 - ii. Select **Absolute addressing**.
 - iii. Open the HEX file **master_image\ext_flash.hex** and click **OK**.

4. Click **Options/Boot info...**, set the **Option bit address** to **0x018000** and click **OK**.
5. Click **Generate** to generate the POF programming file.

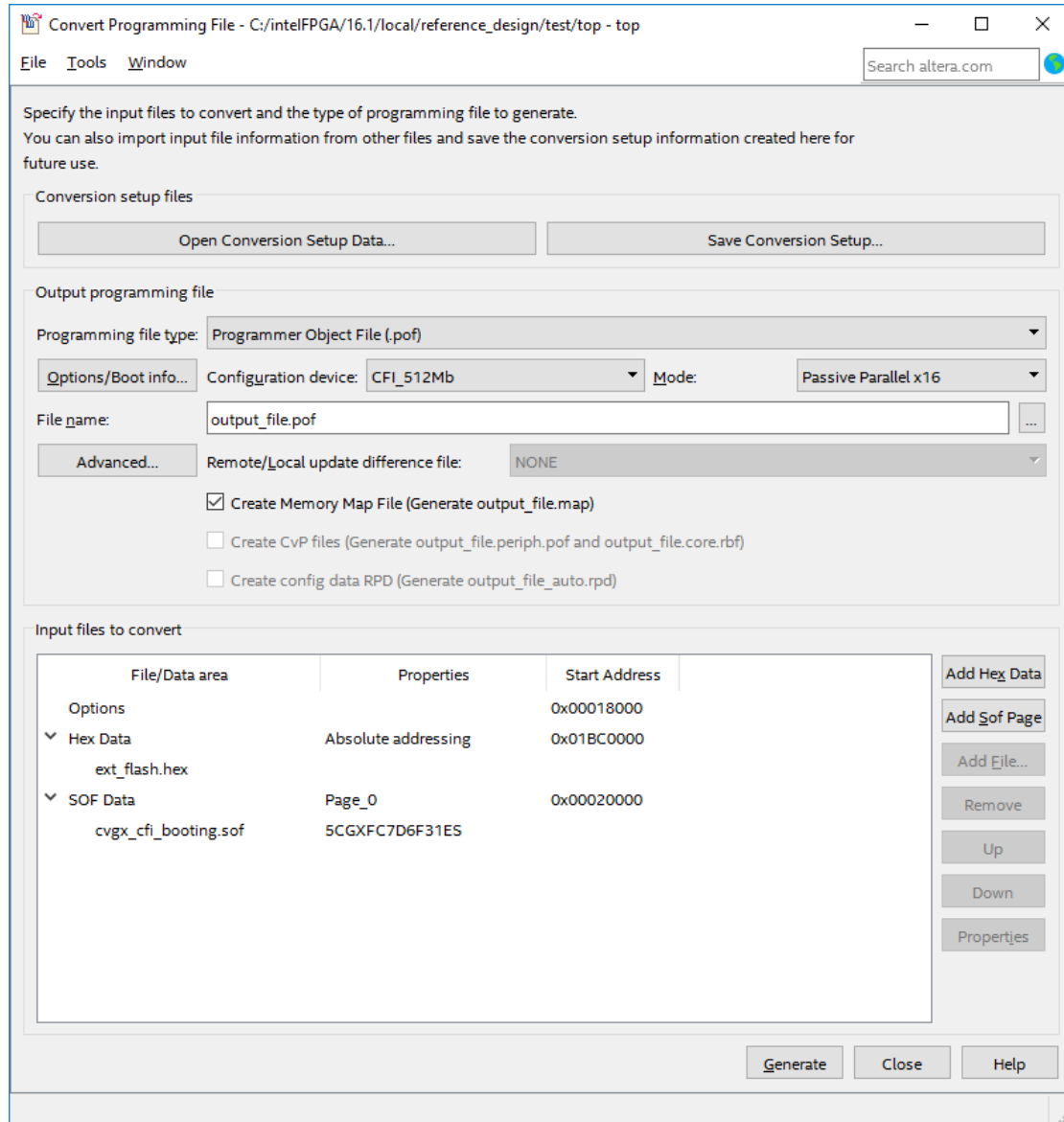


Figure 20: Convert Programming File Window

6.6 Programming the CFI flash

Intel provides Parallel Flash Loader (PFL) IP core that can program and control the FPGA configuration with data from external flash. A simple MAX V design that instantiates the PFL IP core to allow CFI flash programming is available [here](#).

To program the CFI flash:

1. Connect the USB Blaster cable from PC to the on-board USB Blaster II connector and power up the board.
2. Open the **Programmer** from **Tools** menu in Quartus Prime software, then click **Hardware Setup...** and select **USB-BlasterII**.
3. Click **Auto Detect**, then select **5CGXFC7D6ES** and click **OK**.
4. Right click on the MAX V device **5M2210Z** and click **Change File**.
5. Open the simple MAX V PFL design output file named **maxV_pfl.pof** from [here](#).
6. Tick the **Program/Configure** checkbox for MAX V **5M2210ZF256** device, including **CFM** and **UFM**.
7. Click **Start** to start programming the MAX V device.

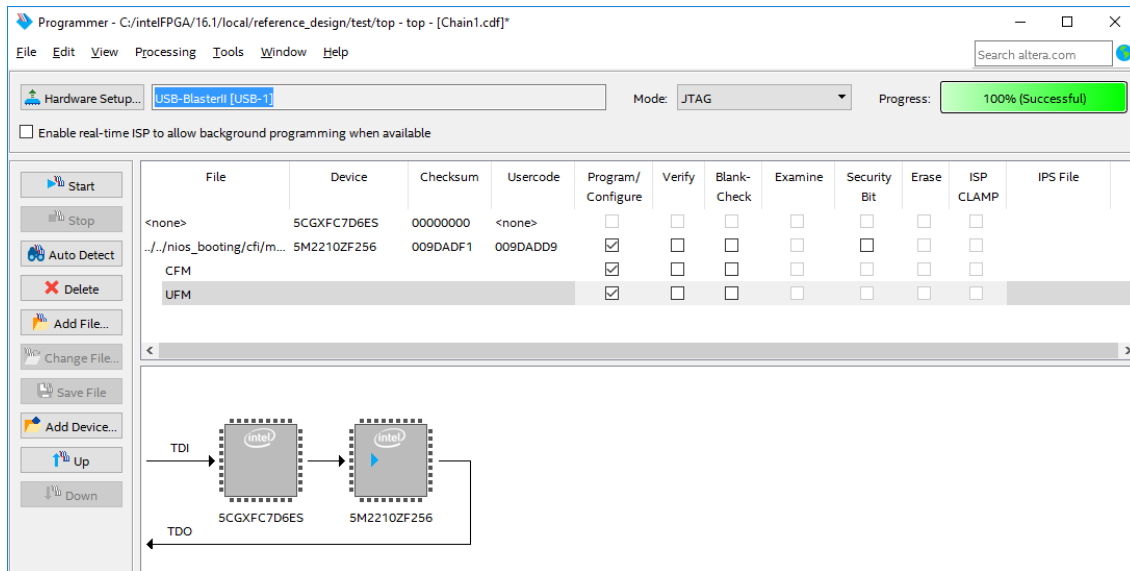


Figure 21: Programming MAX V Window

8. After the programming successful, click **Auto Detect**. If a warning dialog box pops up as below, click **Yes**.

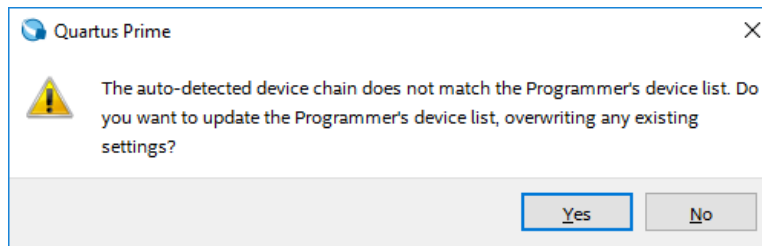


Figure 22: Warning Pop Up Box

9. You should see **CFI_512Mb** attached to the MAX V device.
10. Right click on the **CFI_512Mb** device and click **Change File**.

11. Open the **master_image\output_file.pof**.
12. Tick the **Program/Configure** checkbox for the **CFI_512Mb** device including **Page_0**, **ext_flash.hex** and **OPTION_BITS**.
13. Click **Start** to start programming the CFI flash.

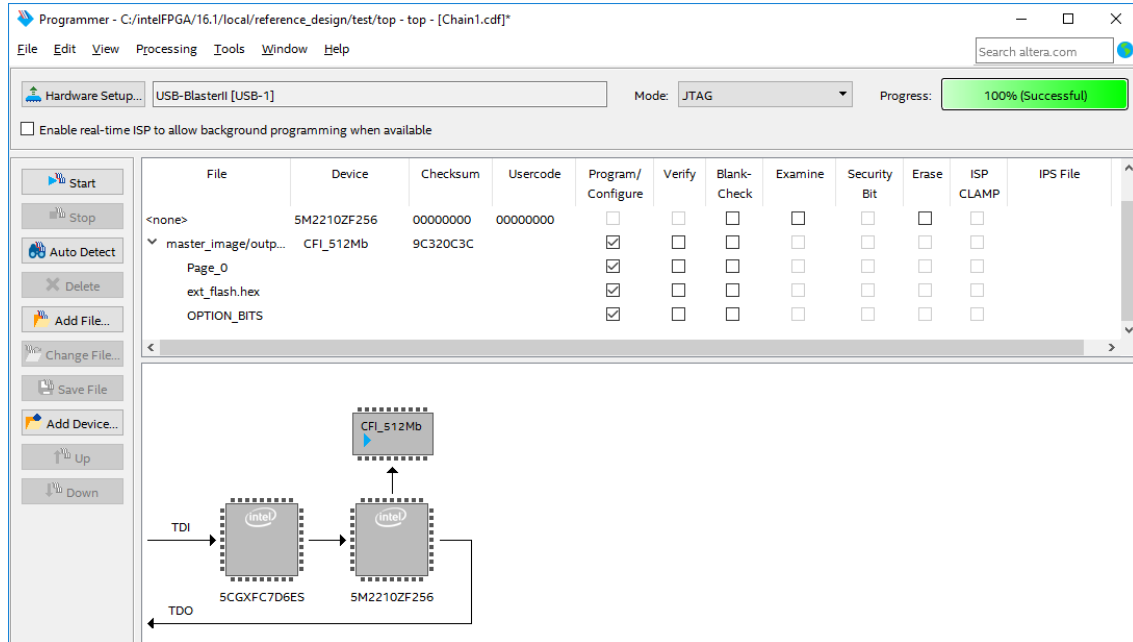


Figure 23: Programming CFI Flash Window

14. After the programming successful, right click on the MAX V device **5M2210ZF256** and click **Change File**.
15. Open the **max5.pof** from **<kit installation directory>/cycloneVGX_5cgxfc7df31es_fpga/examples/max5**.
16. Tick the **Program/Configure** checkbox for MAX V **5M2210ZF256** device, including **CFM** and **UFM**.
17. Click **Start** to start programming the MAX V device.

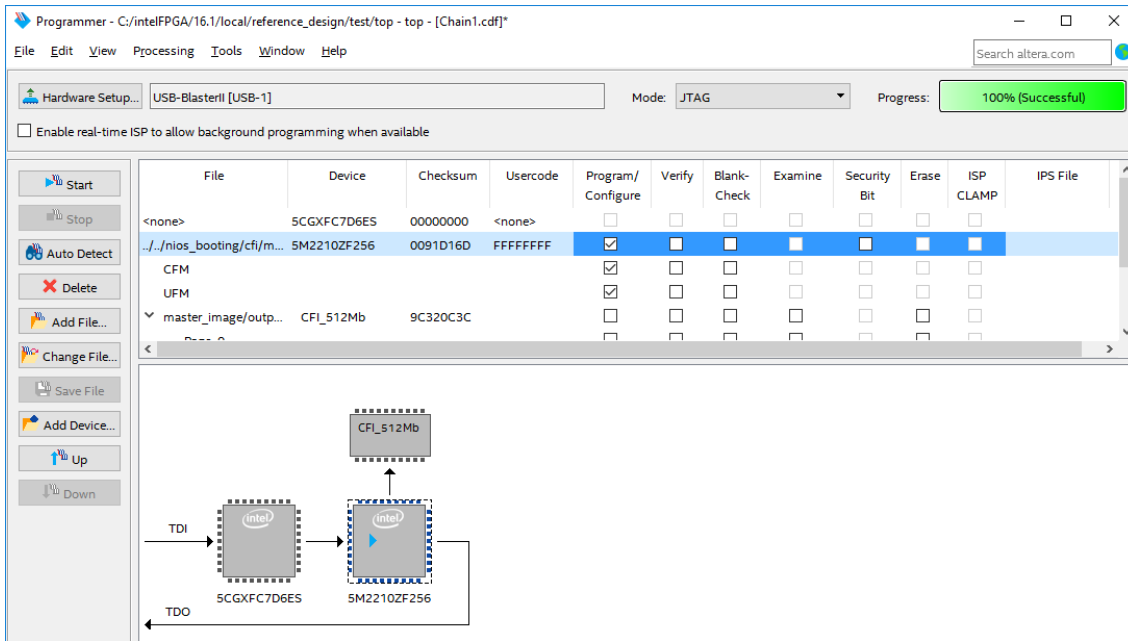


Figure 24: Programming MAX V Window

18. Power cycle the board.

6.7 Run the Design

Once you have programmed the CFI flash, the design is ready to be executed.

1. Launch **Nios II Command Shell**.
2. Run `nios2-terminal` and you should see **Hello from Cyclone V GX Nios II!**.

```

/cydrive/c/intelFPGA/16.1
-----
Altera Nios2 Command Shell [GCC 4]
Version 16.1, Build 196
-----
ngchunhs@ngchunhs-MOBL /cydrive/c/intelFPGA/16.1
$ nios2-terminal
nios2-terminal: connected to hardware target using JTAG UART on cable
nios2-terminal: "USB-BlasterII [USB-1]", device 1, instance 0
nios2-terminal: (Use the IDE stop button or Ctrl-C to terminate)

Hello from Cyclone V GX Nios II!

```

Figure 25: Nios II Command Shell

7. Revision History

Revision Date	Comments	Author
31 July 2017	First release.	Ng, Chun Hsien