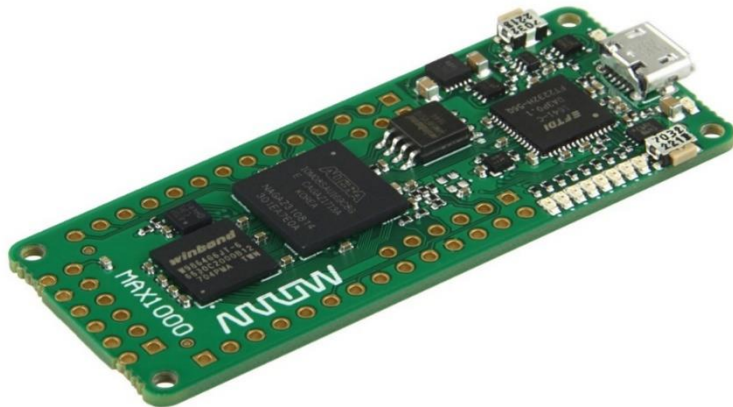


# MAX1000

## Simple Nios Lab



### Software and hardware requirements to complete all exercises

**Software Requirements:** Quartus® Prime Lite or Standard Edition version 18.0 or 18.1

**Hardware Requirements:** ARROW MAX1000 Board



## 1. Introduction

This tutorial provides comprehensive information to help you understand how to create a simple Nios soft processor system in an Intel FPGA and run it on your MAX1000 board. The Nios II processor is a soft intellectual property (IP) processor that you download (along with other hardware components that comprise the Nios II system) onto an Intel FPGA. At the end, you will understand basic concepts about Quartus Platform Designer (formerly Qsys), Software Build Tool and the basic development flow for the Nios II design flow.

**Lab Notes:** Many of the names that the lab asks you to choose for files, components, and other objects in this exercise must be spelled exactly as directed. This nomenclature is necessary because the pre-written software application includes variables that use the names of the hardware peripherals. Naming the components differently can cause the software application to fail. There are also other similar dependencies within the project that require you to enter the correct names.

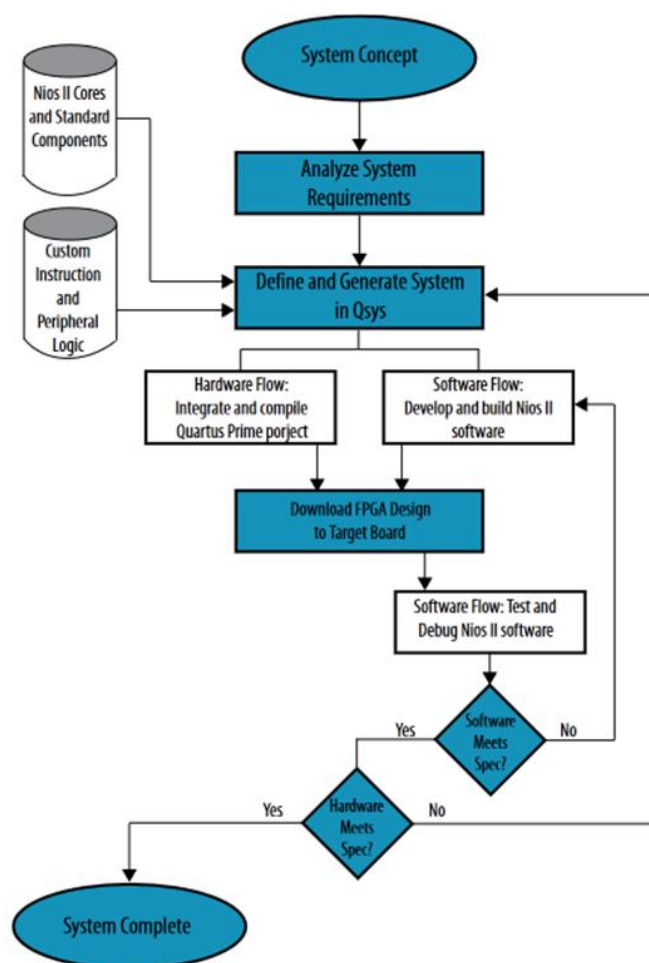
## 2. Getting Started

The first objective is to ensure that you have all the necessary hardware items and software installed so that the lab can be completed successfully. Below is a list of items required to complete this lab:

- MAX1000 Board (10M08SAU169C8G)
- USB Cable
- Quartus Prime 18.0 Lite was used for this lab. Previous/newer versions should work (If no Quartus Prime is installed, refer to MAX1000 User Guide for instructions)
- Installed Arrow USB Drivers (If not, refer to MAX1000 User Guide for instructions)
- Personal computer or laptop running 64-bit Linux / Windows 7 or later with at least an Intel i3 core (or equivalent), 4GB RAM and 12 GB of free hard disk space
- A desire to learn!

### 3. Design Flow

Developing software for an embedded system on a programmable chip requires an understanding of the design flow between the Platform Designer (formerly Qsys) system integration tool and the Nios II Embedded Development Suite (EDS). Typically, designs begin with requirements and become inputs to system definitions. System definition is the first step in the design flow process. For this workshop, the design will be built and then the FPGA image will be downloaded into the board. The objective of the module is to review the development tools that will be used.



The above diagram shows the typical design flow for the system design. The system definition is done with Platform Designer. The Nios II IDE uses the system description to create a new project for the software application. The output of the FPGA design is a FPGA image that is used to configure the FPGA. The output of the software flow is an executable which runs on the Nios II processor.

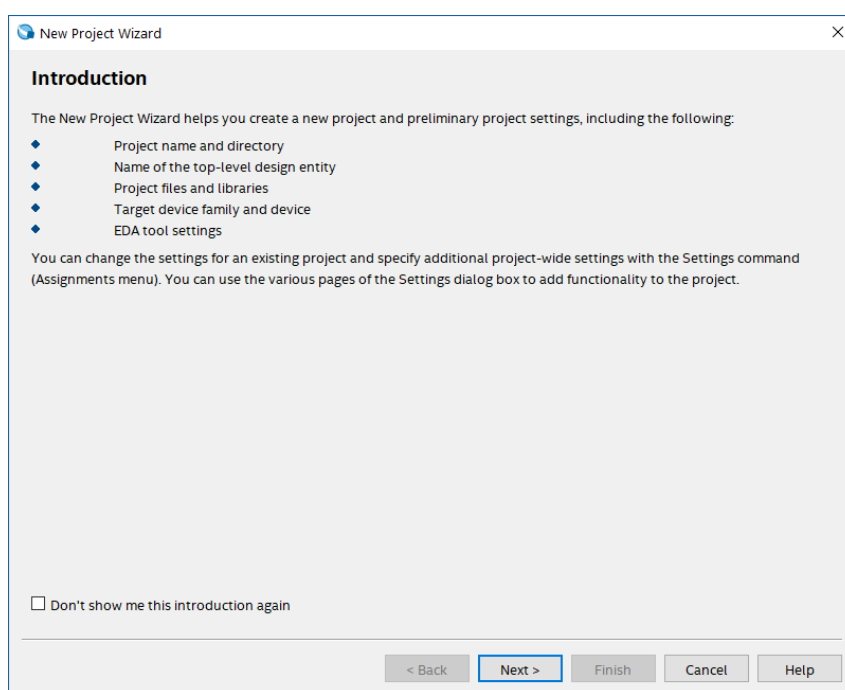
## 4. Project with MAX1000

### 4.1 Quartus Prime project

#### 4.1.1 Create a new Quartus Prime project

4.1.1.1 If not already open, from the Start menu or the Desktop, open the Quartus Prime 18.0 Lite software.

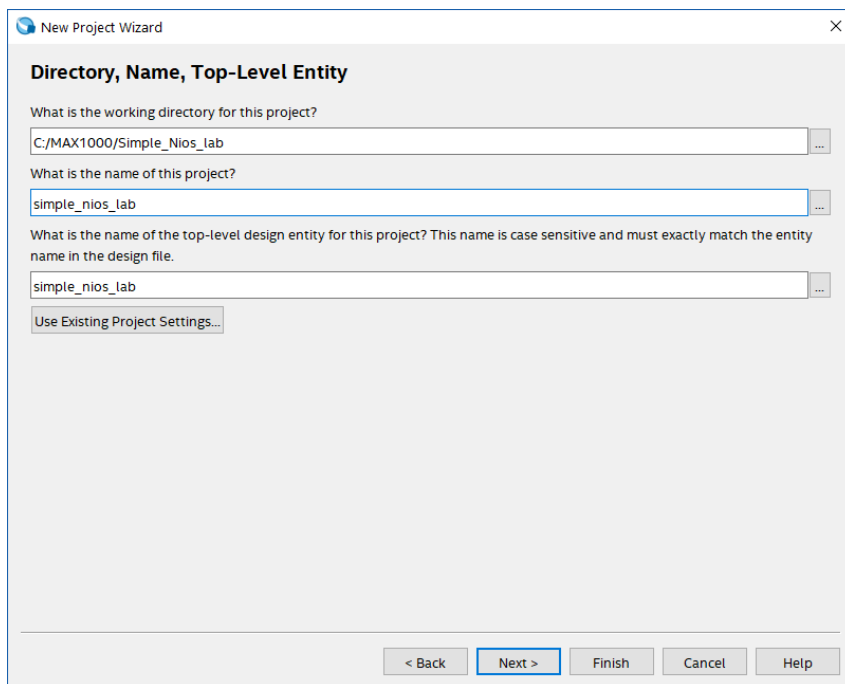
4.1.1.2 Create a new project using the New Project Wizard: **File → New Project Wizard**.



4.1.1.3 Click **Next**.

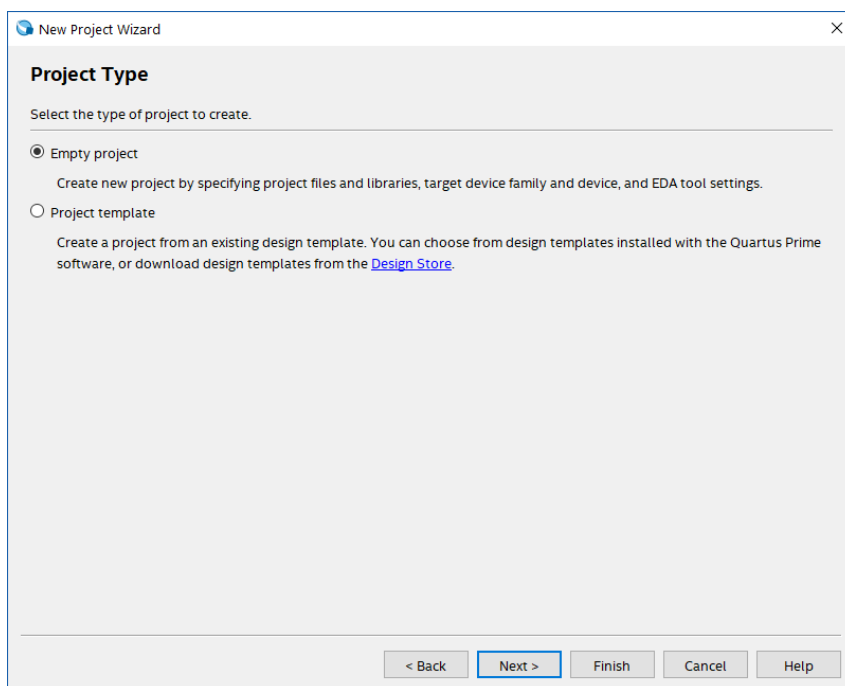
4.1.1.4 Configure the New Project Wizard directory, name and top-level entity information:

- Enter a directory in which you will store your Quartus project files for this design, for example, **C:\MAX1000\Simple\_Nios\_lab**
- Specify the name of the project: **simple\_nios\_lab**
- Specify the name of the top-level entity: **simple\_nios\_lab**

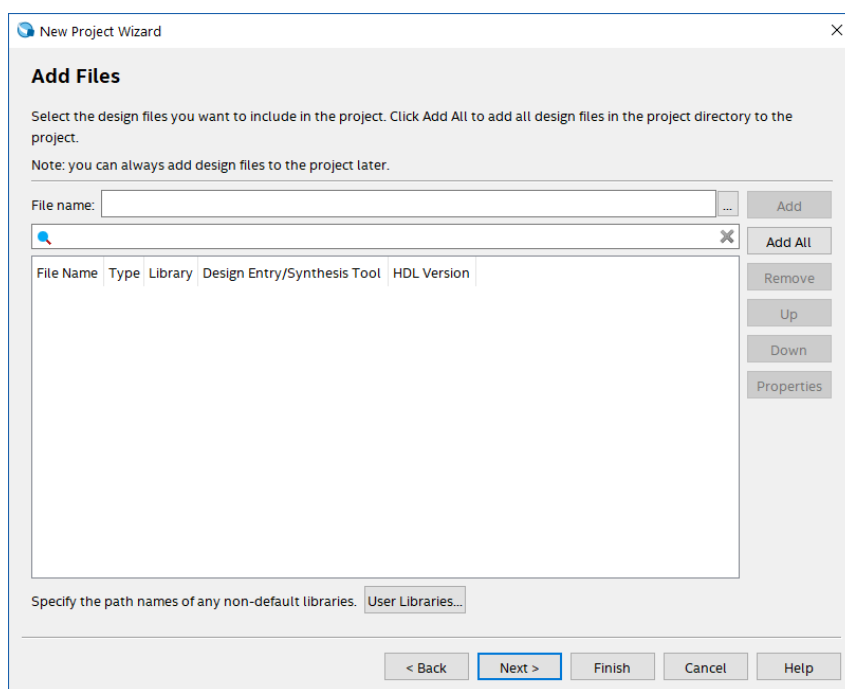


4.1.1.5 Click **Next**.

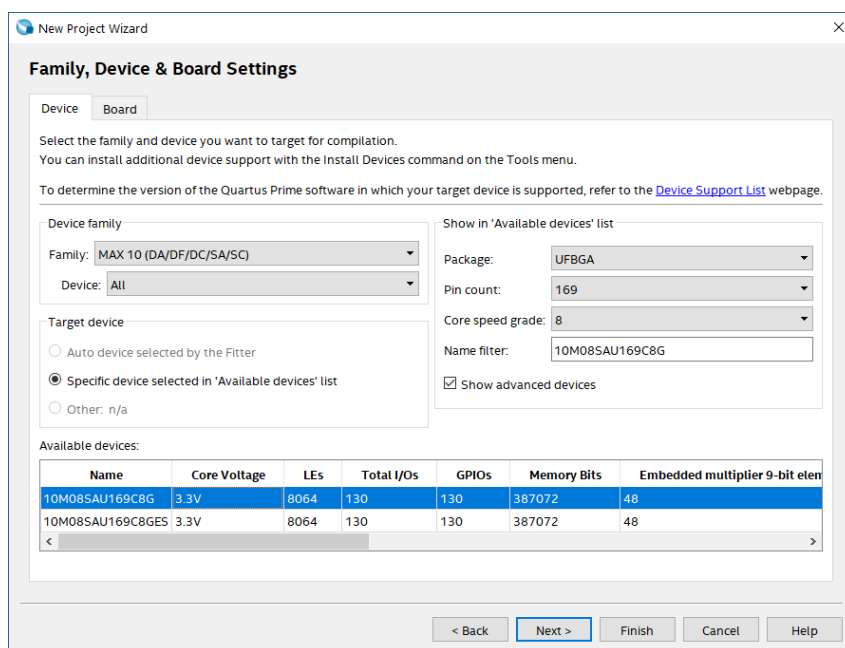
4.1.1.6 On the Project Type page, select “**Empty project**” and click **Next**.



#### 4.1.1.7 On the Add Files page, click **Next**.



#### 4.1.1.8 Specify Family and Device Settings. Use pull-down menus to select MAX10 family or enter the part number in the Name Filter text box. The part number is **10M08SAU169C8G**.



#### 4.1.1.9 Click **Finish**.


## 4.2 Design entry

**Overview:** In this module you will use Platform Designer to create hardware for the processor system. You will add standard and custom components, make interface connections and assign clocks.

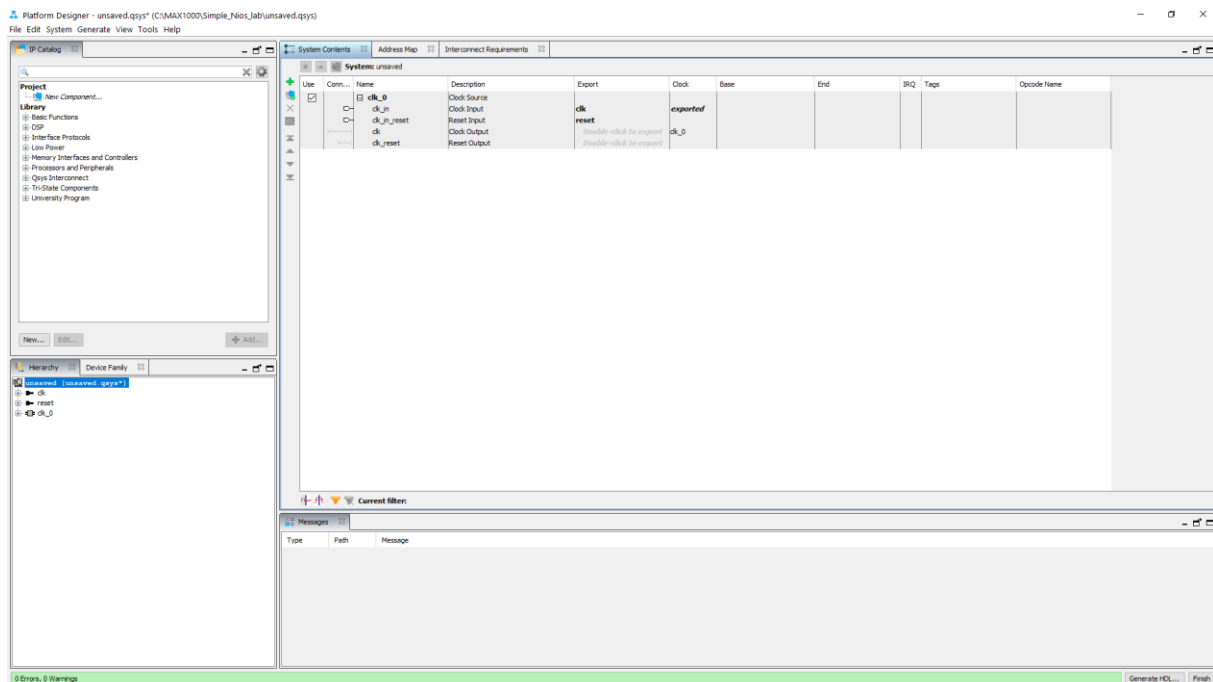
### 4.2.1 Add components to Platform Designer

Platform Designer is a high-level system integration tool that allows you to quickly build a system using Intel IP block as well as custom components. The tool automatically creates interconnect logic between the components for easy design use.

Platform Designer is made up of several components and automatically generates high performance interconnect between them. It allows you to connect components on an interface level, rather by signal by signal level. The tool understands the different types of interfaces and will only allow connections between interfaces of same type (i.e. a data master connects to a data slave, clock source to clock sink, etc.).

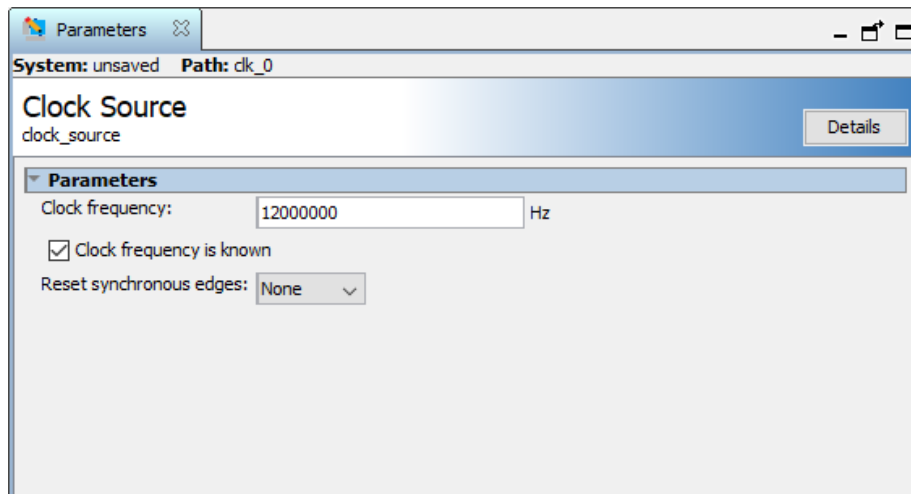
4.2.1.1 Open Platform Designer from the **Tools → Platform Designer** or clicking on  button on the toolbar.

4.2.1.2 In the new window, you should see a single clock source component, named `clk_0` in the System Components tab. This tab shows all the components currently in your system.

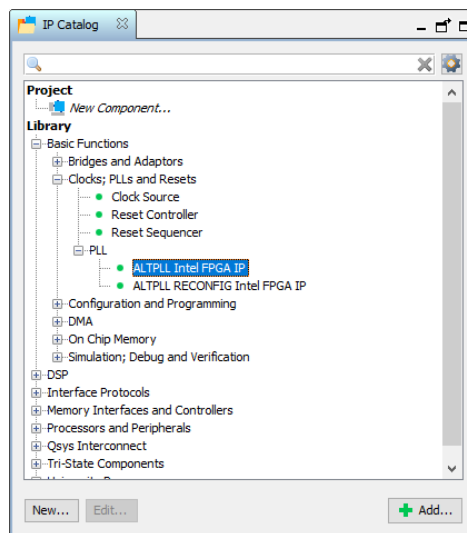


4.2.1.3 Double click on `clk_0`.

- 4.2.1.4 Set the Clock frequency to **12 MHz** (12000000 Hz).  
 Ensure that 'Clock frequency is known' parameter is enabled.

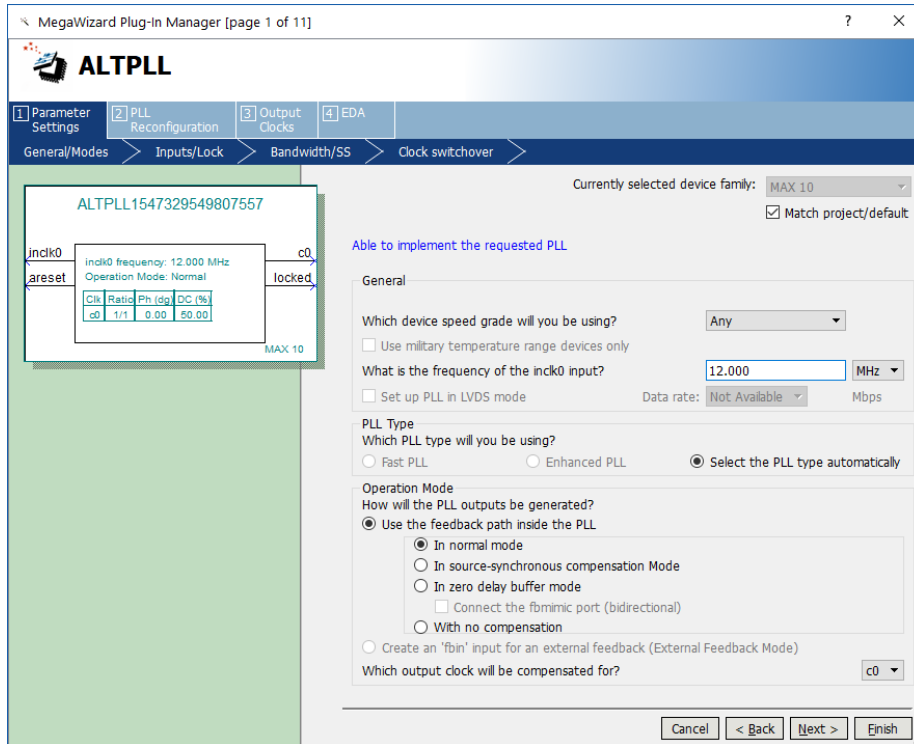


- 4.2.1.5 Click on **X** on the parameter tab to close the parameter window.
- 4.2.1.6 Right click on the clk\_0 and select **Rename**. Rename the clock source to **clk12mhz** and press Enter.
- 4.2.1.7 From the IP Catalog panel on the left side, expand the menus for the **Basic Functions** → **Clocks; PLLs and Resets** → **PLL** and double click on **ALTPLL Intel FPGA IP**.



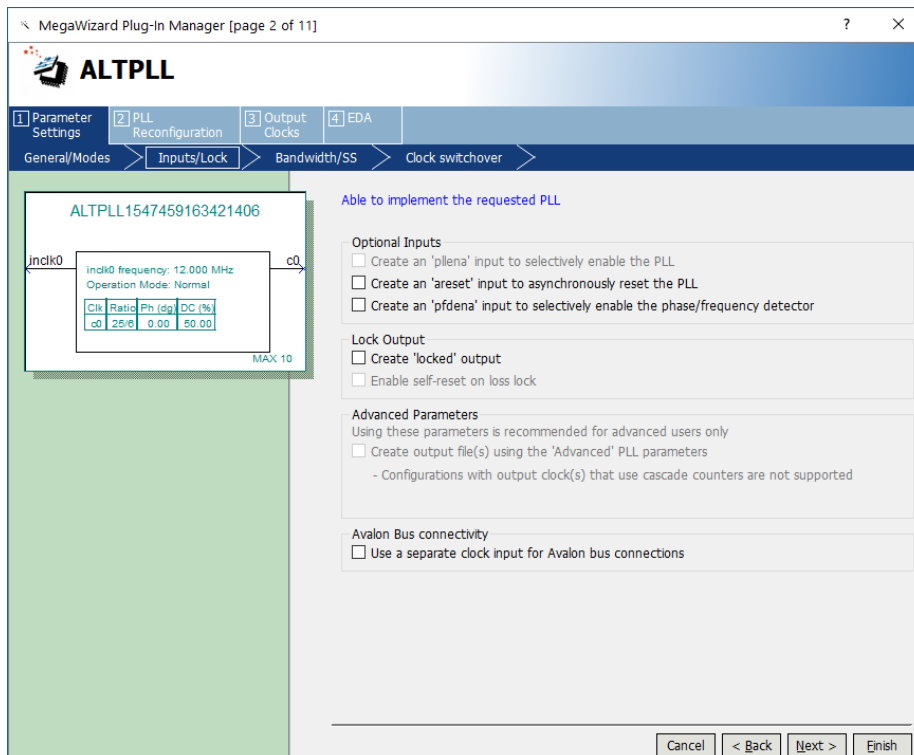


4.2.1.8 Under General/Modes tab (page 1 of 11) of PLL MegaWizard change the frequency of clock input to **12 MHz**. This source is provided by the internal oscillator in the MAX10 FPGA.



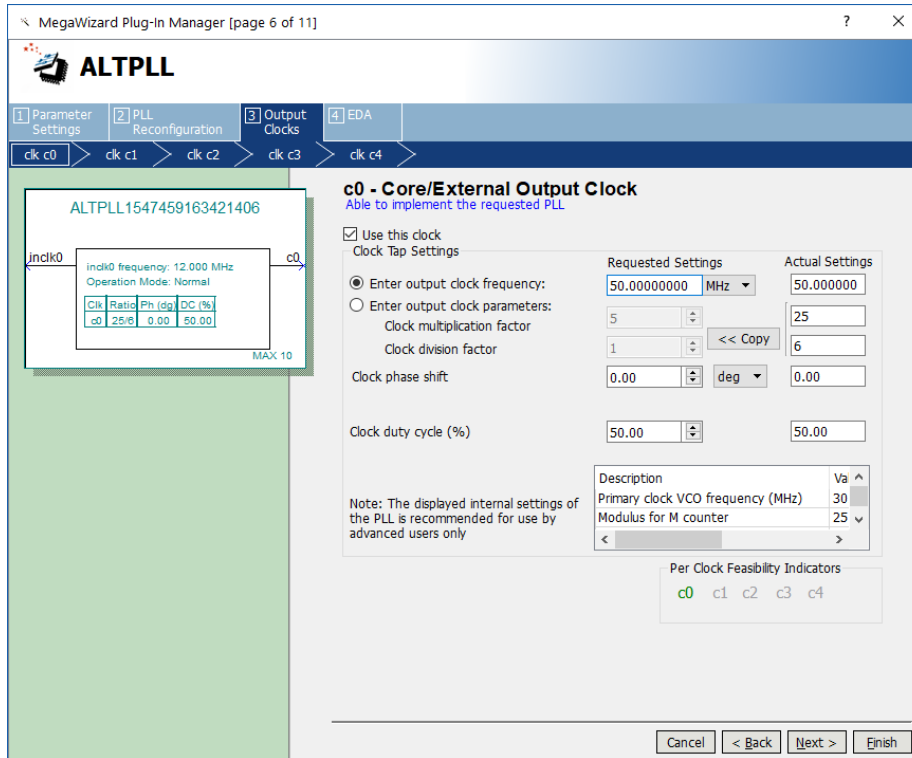
4.2.1.9 Click **Next**.

4.2.1.10 In Input/Lock tab (page 2 of 11) uncheck 'areset' input and locked output option.



4.2.1.11 Click **Next** until you reach the **Output Clocks** tab (page 6 of 11).

4.2.1.12 Under the clk c0 tab (page 6 of 11) select “Enter output clock frequency” and enter **50 MHz**.

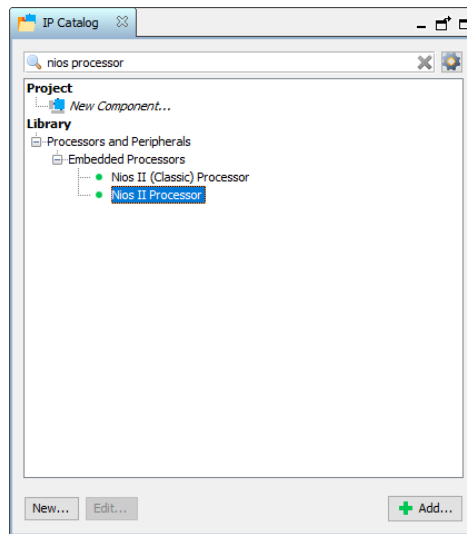


4.2.1.13 Click **Finish**. This will take you to the EDA tab (page 11 of 11). Click **Finish** again to close ALTPLL MegaWizard Manager.

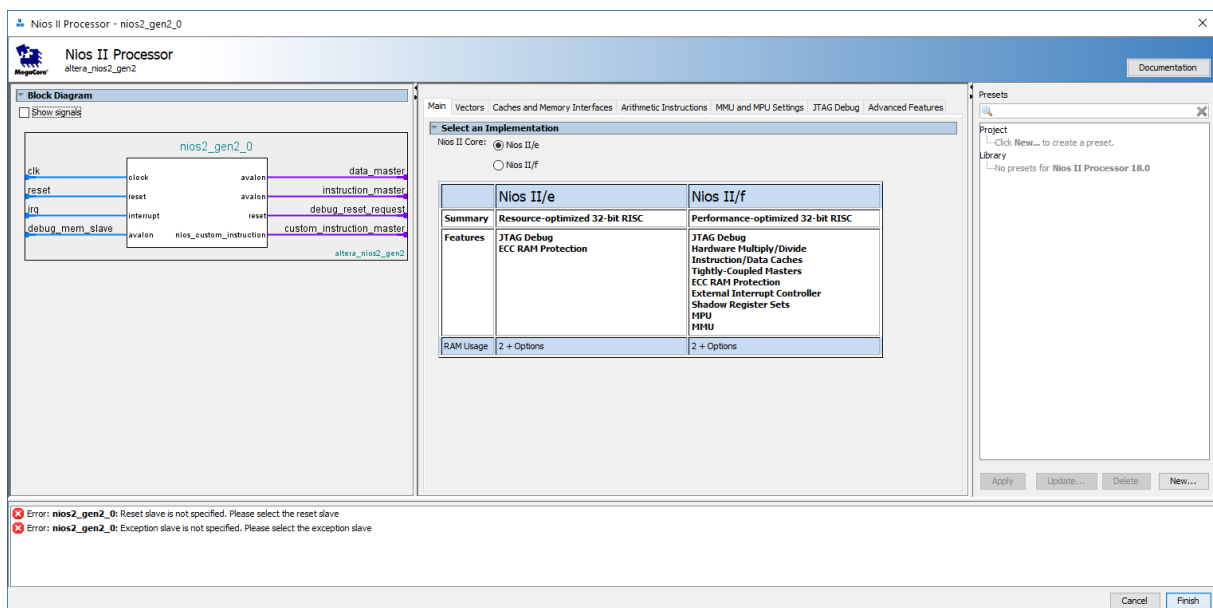
4.2.1.14 A component entitled altpll\_0 should appear under Module Name. Rename this component to **pll**.

Use	Connect...	Name	Description	Export	Clock	Base	End	IRQ	Tags	Opcode Name
<input checked="" type="checkbox"/>		clk12mhz	Clock Source							
		clk_in	Clock Input	clk	exported					
		clk_in_reset	Reset Input	reset						
		clk	Clock Output	Double-click to export	clk12mhz					
		clk_reset	Reset Output	Double-click to export						
<input checked="" type="checkbox"/>		pll	ALTPLL Intel FPGA IP							
		indk_interface	Clock Input	Double-click to export	unconnected					
		indk_interface_reset	Reset Input	Double-click to export	[indk_interf...					
		pll_slave	Avalon Memory Mapped Slave	Double-click to export	[indk_interf... #					
		c0	Clock Output	Double-click to export	pll_c0					

4.2.1.15 In the search bar of the IP Catalog, type “nios processor”, and double click on **Nios II Processor**.

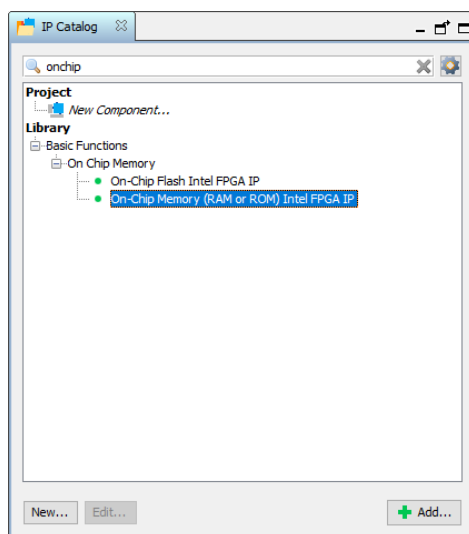


4.2.1.16 In the Main tab, ensure that the **Nios II /e** option is selected, and press **Finish**.

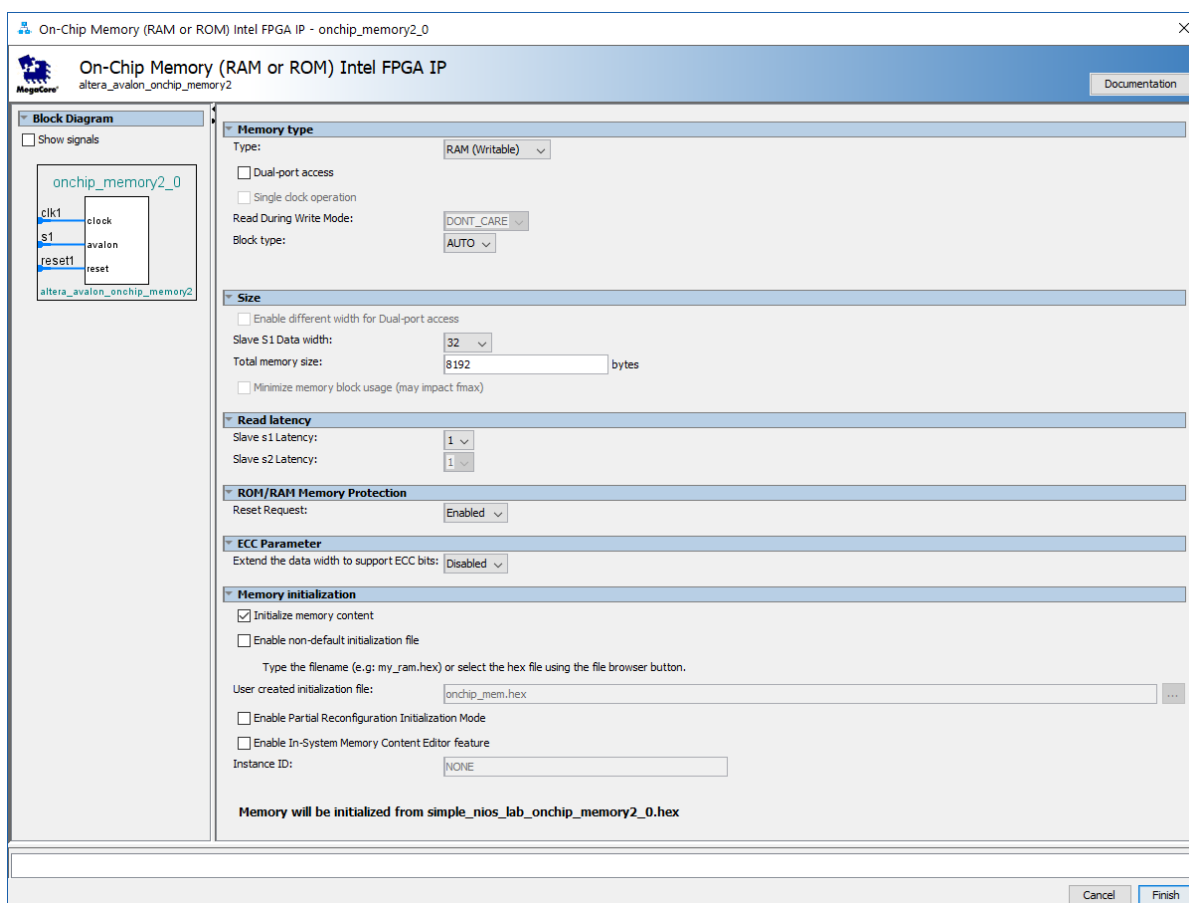


4.2.1.17 Rename nios2\_gen2\_0 to **nios**.

4.2.1.18 In the search bar of the IP Catalog, type “onchip”, and add **On-Chip Memory (RAM or ROM) Intel FPGA IP**.



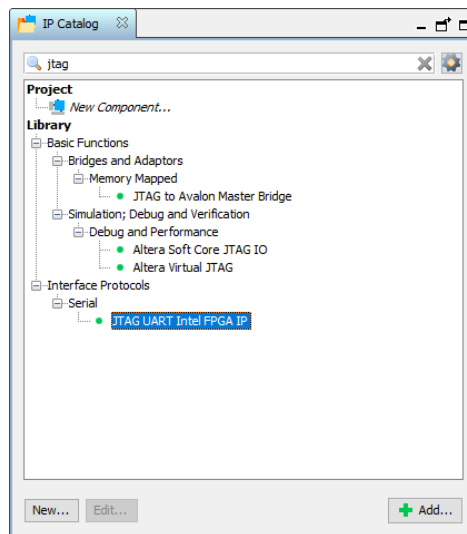
4.2.1.19 Change the Total memory size to **8192 bytes**.



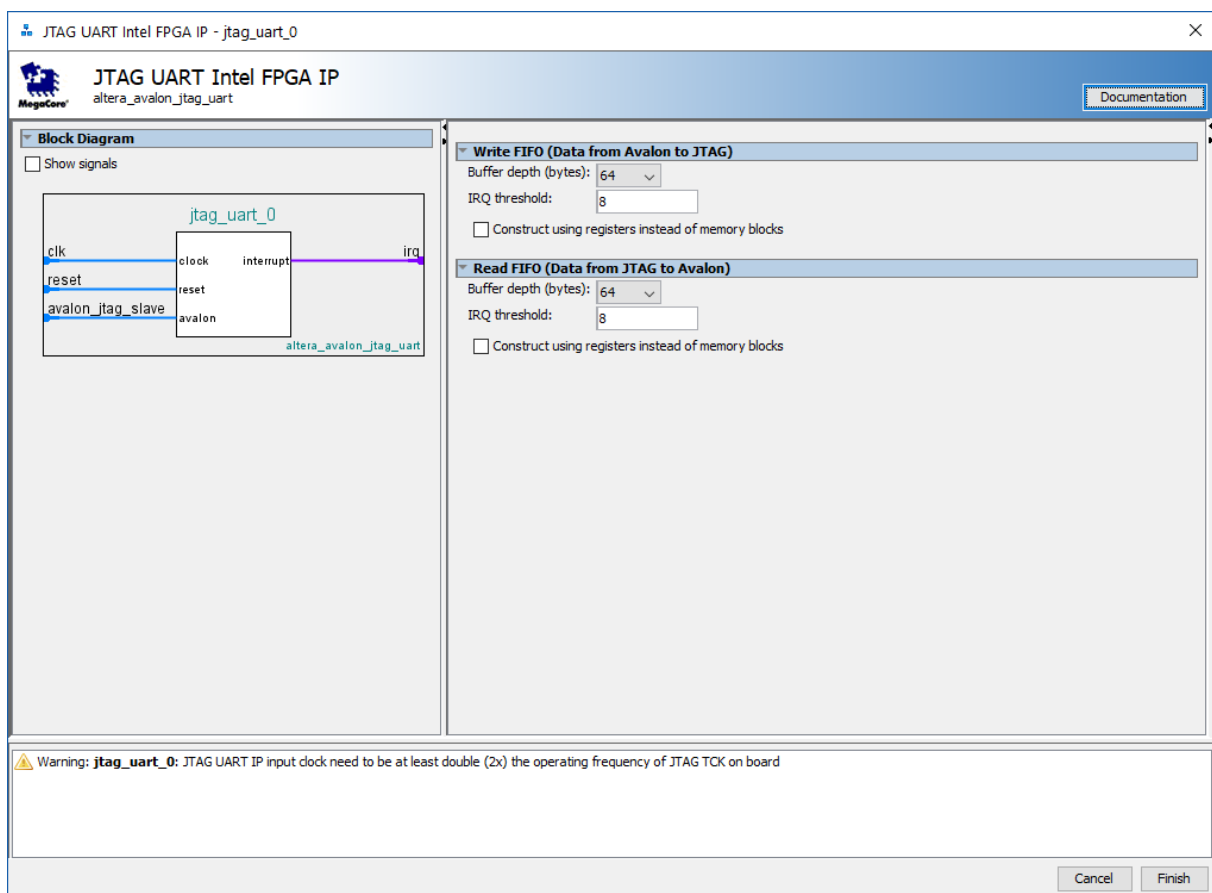
4.2.1.20 Accept the defaults for the remaining fields and press **Finish**.

4.2.1.21 Rename onchip\_memory2\_0 to **onchip**.

4.2.1.22 In the search bar of the IP Catalog, type “jtag”, and add **JTAG UART Intel FPGA IP**.

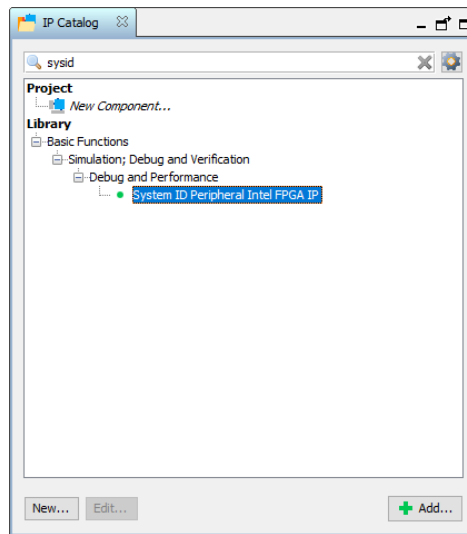


4.2.1.23 Accept all defaults and press **Finish**.

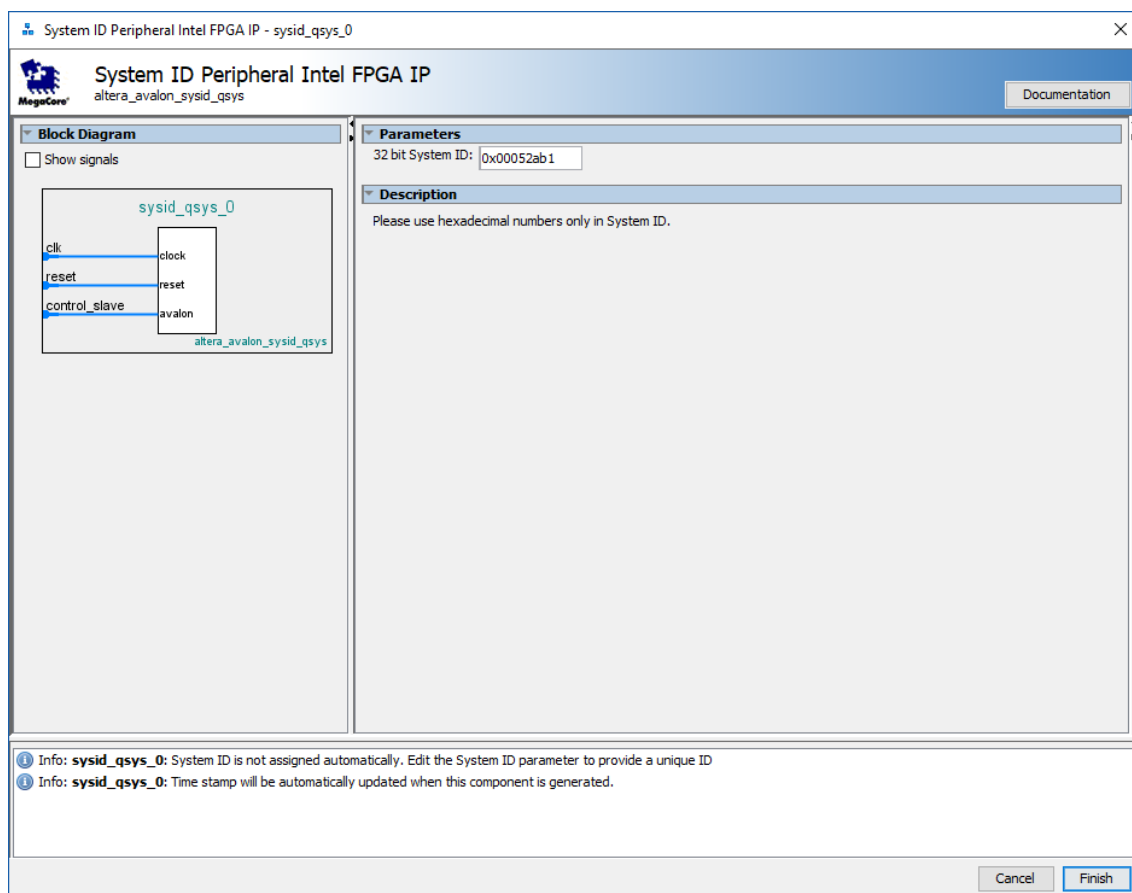


4.2.1.24 Rename `jtag_uart_0` to **jtag\_uart**.

4.2.1.25 In the search bar of the IP Catalog, type “sysid”, and add **System ID Peripheral Intel FPGA IP**.

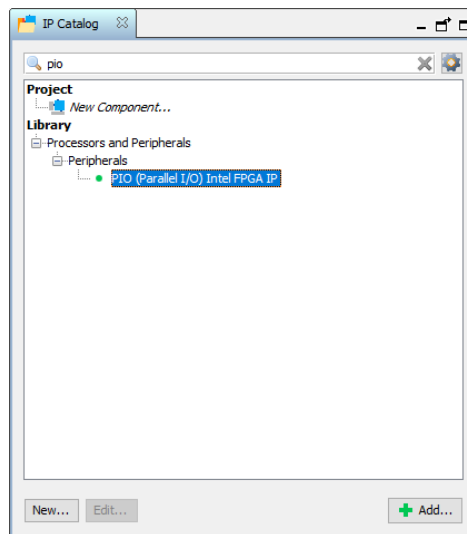


4.2.1.26 Edit the 32 bit System ID to any acceptable value you like, or just accept the default ID and press **Finish**.

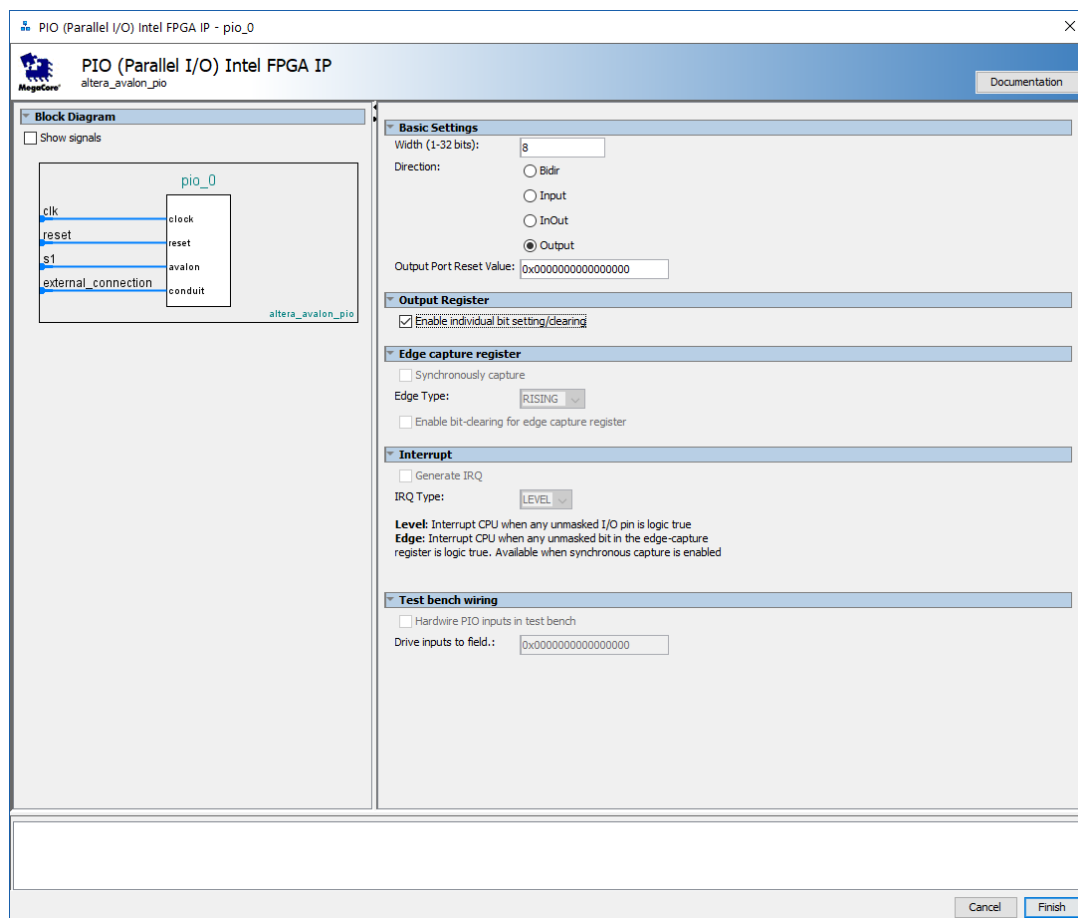


4.2.1.27 Rename sysid\_qsys\_0 to **sysid**.

4.2.1.28 In the search bar of the IP Catalog, type “pio”, and add **PIO (Parallel I/O) Intel FPGA IP**.



4.2.1.29 Verify that the width is **8 bits** and the direction is **output**. Check ‘Enable individual bit setting/clearing’ option.



4.2.1.30 Click **Finish**.

4.2.1.31 Rename `pio_0` to **leds**.

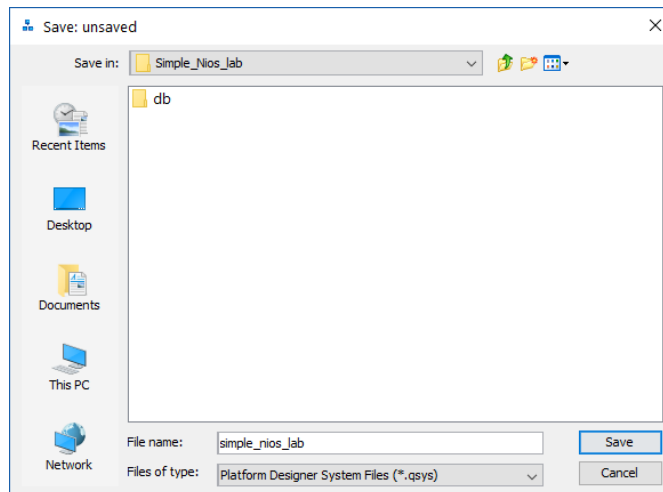
There are multiple errors and warning in the bottom console indication that various ports are not connected, and the memory addresses are not correct. Ignore these for now, as we will address these connections and setups in the following steps.

At this point, there are 7 components in the system and should look as follows:

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags	Opcode Name
<input checked="" type="checkbox"/>		<b>clk12mhz</b>	Clock Source	<b>clk</b>	<b>exported</b>					
		clk_in	Clock Input	clk						
		clk_in_reset	Reset Input	reset						
		clk	Clock Output	clk12mhz						
		clk_reset	Reset Output							
<input checked="" type="checkbox"/>		<b>pll</b>	ALTPLL Intel FPGA IP		<b>unconnected</b>					
		indk_interface	Clock Input							
		indk_interface_reset	Reset Input							
		pll_slave	Avalon Memory Mapped Slave							
		c0	Clock Output							
<input checked="" type="checkbox"/>		<b>nios</b>	Nios II Processor		<b>unconnected</b>					
		clk	Clock Input							
		reset	Reset Input							
		data_master	Avalon Memory Mapped Master							
		instruction_master	Interrupt Receiver					IRQ 0	IRQ S1	
		irq	Interrupt Receiver							
		debug_reset_request	Reset Output							
		debug_mem_slave	Avalon Memory Mapped Slave			0x0800	0x0fff			
		custom_instruction_m...	Custom Instruction Master							
<input checked="" type="checkbox"/>		<b>onchip</b>	On-Chip Memory (RAM or ROM) Intel ...		<b>unconnected</b>					
		clk1	Clock Input							
		s1	Avalon Memory Mapped Slave							
		reset1	Reset Input							
<input checked="" type="checkbox"/>		<b>jtag_uart</b>	JTAG UART Intel FPGA IP		<b>unconnected</b>					
		clk	Clock Input							
		reset	Reset Input							
		avalon_jtag_slave	Avalon Memory Mapped Slave							
		irq	Interrupt Sender							
<input checked="" type="checkbox"/>		<b>sysid</b>	System ID Peripheral Intel FPGA IP		<b>unconnected</b>					
		clk	Clock Input							
		reset	Reset Input							
		control_slave	Avalon Memory Mapped Slave							
<input checked="" type="checkbox"/>		<b>leds</b>	P10 (Parallel I/O) Intel FPGA IP		<b>unconnected</b>					
		clk	Clock Input							
		reset	Reset Input							
		s1	Avalon Memory Mapped Slave							
		external_connection	Conduit							

4.2.1.32 Save your design **File** → **Save** and enter the following information.

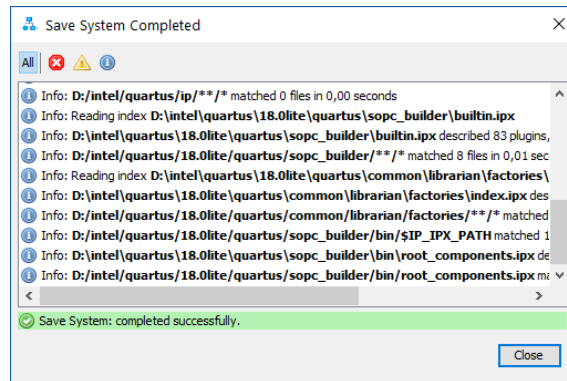
- File name: **simple\_nios\_lab**
- Files of type: **Platform Designer System Files (\*.qsys)**



4.2.1.33 Click **Save**.



#### 4.2.1.34 When the Save System Completed, then click **Close**.

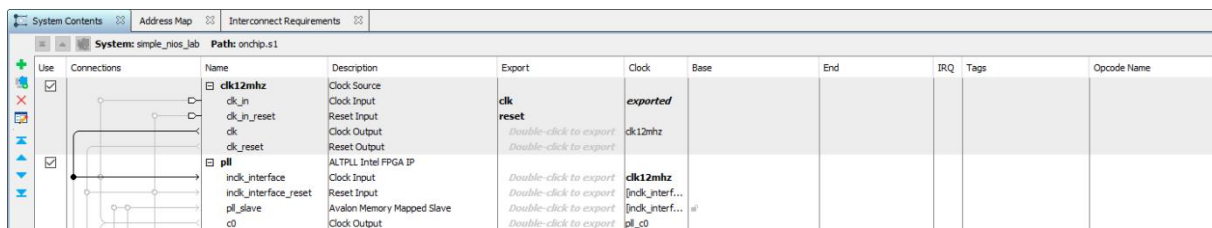


## 4.2.2 Connections

4.2.2.1 In the Connections column, hover over the connections and you will then be able to fill in dots to make the connections by clicking them.

4.2.2.2 Create the following connection: **clk12mhz | clk ↔ pll | inclk\_interface**

The connection should look as follow:



4.2.2.3 As in the previous step, make the following connections for the clock signal:

Component A	Component B
pll   c0	↔ nios   clk
pll   c0	↔ onchip   clk1
pll   c0	↔ jtag_uart   clk
pll   c0	↔ sysid   clk
pll   c0	↔ leds   clk

The connections should look as the follows:

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags	Opcode Name
<input checked="" type="checkbox"/>		clk12mhz	Clock Source	clk	exported					
<input checked="" type="checkbox"/>		clk_in	Clock Input	reset						
<input checked="" type="checkbox"/>		clk_in_reset	Reset Input							
<input checked="" type="checkbox"/>		clk	Clock Output							
<input checked="" type="checkbox"/>		clk_reset	Reset Output							
<input checked="" type="checkbox"/>		pll	ALTPLL Intel FPGA IP							
<input checked="" type="checkbox"/>		indk_interface	Clock Input		clk12mhz					
<input checked="" type="checkbox"/>		indk_interface_reset	Reset Input							
<input checked="" type="checkbox"/>		pll_slave	Avalon Memory Mapped Slave			#				
<input checked="" type="checkbox"/>		c0	Clock Output							
<input checked="" type="checkbox"/>		nios	Nios II Processor							
<input checked="" type="checkbox"/>		clk	Clock Input		pll_c0					
<input checked="" type="checkbox"/>		reset	Reset Input							
<input checked="" type="checkbox"/>		data_master	Avalon Memory Mapped Master							
<input checked="" type="checkbox"/>		instruction_master	Avalon Memory Mapped Master							
<input checked="" type="checkbox"/>		irq	Interrupt Receiver					IRQ 0	IRQ 31	
<input checked="" type="checkbox"/>		debug_reset_request	Reset Output							
<input checked="" type="checkbox"/>		debug_mem_slave	Avalon Memory Mapped Slave			#	0x0800	0x0fff		
<input checked="" type="checkbox"/>		custom_instruction_m...	Custom Instruction Master							
<input checked="" type="checkbox"/>		onchip	On-Chip Memory (RAM or ROM) Intel ...							
<input checked="" type="checkbox"/>		clk1	Clock Input		pll_c0					
<input checked="" type="checkbox"/>		s1	Avalon Memory Mapped Slave			#				
<input checked="" type="checkbox"/>		reset1	Reset Input							
<input checked="" type="checkbox"/>		jtag_uart	JTAG UART Intel FPGA IP							
<input checked="" type="checkbox"/>		clk	Clock Input		pll_c0					
<input checked="" type="checkbox"/>		reset	Reset Input							
<input checked="" type="checkbox"/>		avalon_jtag_slave	Avalon Memory Mapped Slave			#				
<input checked="" type="checkbox"/>		irq	Interrupt Sender							
<input checked="" type="checkbox"/>		sysid	System ID Peripheral Intel FPGA IP							
<input checked="" type="checkbox"/>		clk	Clock Input		pll_c0					
<input checked="" type="checkbox"/>		reset	Reset Input							
<input checked="" type="checkbox"/>		control_slave	Avalon Memory Mapped Slave			#				
<input checked="" type="checkbox"/>		leds	P10 (Parallel I/O) Intel FPGA IP							
<input checked="" type="checkbox"/>		clk	Clock Input		pll_c0					
<input checked="" type="checkbox"/>		reset	Reset Input							
<input checked="" type="checkbox"/>		s1	Avalon Memory Mapped Slave			#				
<input checked="" type="checkbox"/>		external_connection	Conduit							

4.2.2.4 Make the following connections for the data and instruction bus:

Component A			Component B	
nios   data_master	↔	pll   pll_slave		
nios   instruction_master	↔	onchip   s1		
nios   data_master	↔	onchip   s1		
nios   data_master	↔	jtag_uart   avalon_jtag_slave		
nios   data_master	↔	sysid   control_slave		
nios   data_master	↔	leds   s1		

The connections should look as the follows:

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags	Opcode Name
<input checked="" type="checkbox"/>		clk12mhz	Clock Source	clk	exported					
<input checked="" type="checkbox"/>		clk_in	Clock Input	reset						
<input checked="" type="checkbox"/>		clk_in_reset	Reset Input							
<input checked="" type="checkbox"/>		clk	Clock Output							
<input checked="" type="checkbox"/>		clk_reset	Reset Output							
<input checked="" type="checkbox"/>		pll	ALTPLL Intel FPGA IP							
<input checked="" type="checkbox"/>		indk_interface	Clock Input		clk12mhz					
<input checked="" type="checkbox"/>		indk_interface_reset	Reset Input							
<input checked="" type="checkbox"/>		pll_slave	Avalon Memory Mapped Slave			#	0x0000	0x000f		
<input checked="" type="checkbox"/>		c0	Clock Output							
<input checked="" type="checkbox"/>		nios	Nios II Processor							
<input checked="" type="checkbox"/>		clk	Clock Input		pll_c0					
<input checked="" type="checkbox"/>		reset	Reset Input							
<input checked="" type="checkbox"/>		data_master	Avalon Memory Mapped Master							
<input checked="" type="checkbox"/>		instruction_master	Avalon Memory Mapped Master							
<input checked="" type="checkbox"/>		irq	Interrupt Receiver					IRQ 0	IRQ 31	
<input checked="" type="checkbox"/>		debug_reset_request	Reset Output							
<input checked="" type="checkbox"/>		debug_mem_slave	Avalon Memory Mapped Slave			#	0x0800	0x0fff		
<input checked="" type="checkbox"/>		custom_instruction_m...	Custom Instruction Master							
<input checked="" type="checkbox"/>		onchip	On-Chip Memory (RAM or ROM) Intel ...							
<input checked="" type="checkbox"/>		clk1	Clock Input		pll_c0					
<input checked="" type="checkbox"/>		s1	Avalon Memory Mapped Slave			#	0x0000	0x1fff		
<input checked="" type="checkbox"/>		reset1	Reset Input							
<input checked="" type="checkbox"/>		jtag_uart	JTAG UART Intel FPGA IP							
<input checked="" type="checkbox"/>		clk	Clock Input		pll_c0					
<input checked="" type="checkbox"/>		reset	Reset Input							
<input checked="" type="checkbox"/>		avalon_jtag_slave	Avalon Memory Mapped Slave			#	0x0000	0x0007		
<input checked="" type="checkbox"/>		irq	Interrupt Sender							
<input checked="" type="checkbox"/>		sysid	System ID Peripheral Intel FPGA IP							
<input checked="" type="checkbox"/>		clk	Clock Input		pll_c0					
<input checked="" type="checkbox"/>		reset	Reset Input							
<input checked="" type="checkbox"/>		control_slave	Avalon Memory Mapped Slave			#	0x0000	0x0007		
<input checked="" type="checkbox"/>		leds	P10 (Parallel I/O) Intel FPGA IP							
<input checked="" type="checkbox"/>		clk	Clock Input		pll_c0					
<input checked="" type="checkbox"/>		reset	Reset Input							
<input checked="" type="checkbox"/>		s1	Avalon Memory Mapped Slave			#	0x0000	0x001f		
<input checked="" type="checkbox"/>		external_connection	Conduit							

4.2.2.5 Create the following connection for interrupt request:

**nios | irq ↔ jtag\_uart | irq**

4.2.2.6 Double click on the Export field next to the external\_connection of leds and name it **led**.

4.2.2.7 Automatically create global reset by selecting **System → Create Global Reset Network** from the menu.

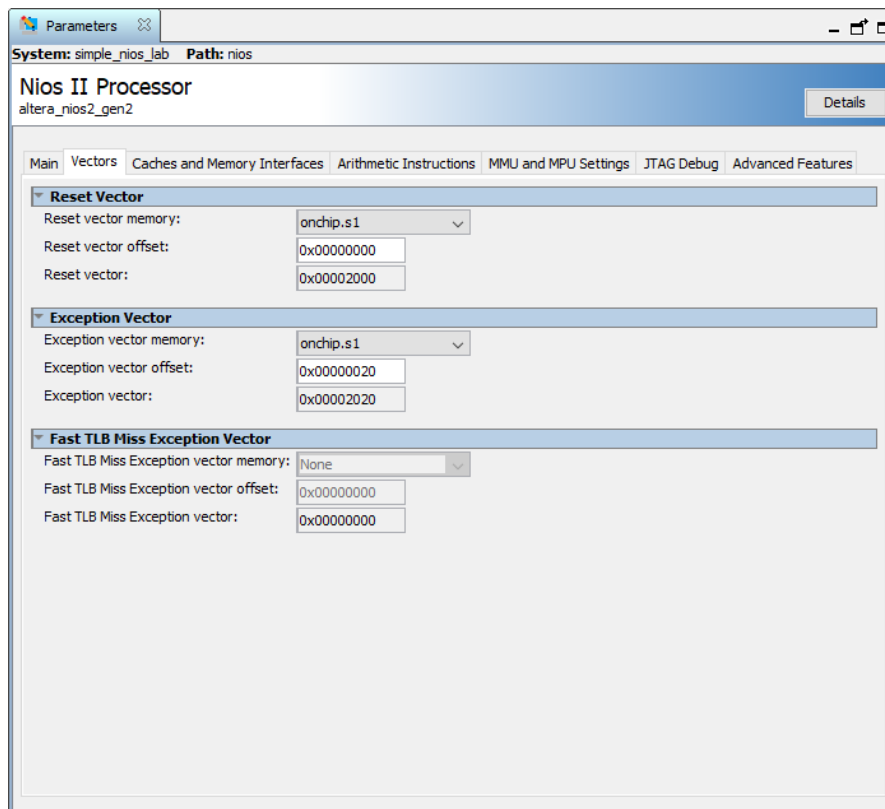
4.2.2.8 Automatically assign base addresses for the peripherals by selecting **System → Assign Base Addresses** from the menu.

4.2.2.9 Verify that your system is the same as below:

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags	Opcode Name
<input checked="" type="checkbox"/>		clk12mhz	Clock Source	clk	exported					
		clk_in	Clock Input	clk						
		clk_in_reset	Reset Input	reset						
		clk	Clock Output		clk12mhz					
		clk_reset	Reset Output							
<input checked="" type="checkbox"/>		pll	ALTPLL Intel FPGA IP							
		indk_interface	Clock Input		clk12mhz					
		indk_interface_reset	Reset Input							
		pll_slave	Avalon Memory Mapped Slave			0x5020	0x502F			
		c0	Clock Output		pll_c0					
<input checked="" type="checkbox"/>		nios	Nios II Processor							
		clk	Clock Input		pll_c0					
		reset	Reset Input		[clk]					
		data_master	Avalon Memory Mapped Master		[clk]					
		instruction_master	Avalon Memory Mapped Master		[clk]					
		irq	Interrupt Receiver		[clk]			IRQ 0	IRQ 31	
		debug_reset_request	Reset Output		[clk]					
		debug_nem_slave	Avalon Memory Mapped Slave		[clk]	0x4800	0x4FFF			
		custom_instruction_m...	Custom Instruction Master		[clk]					
<input checked="" type="checkbox"/>		onchip	On-Chip Memory (RAM or ROM) Intel ...							
		clk1	Clock Input		pll_c0					
		s1	Avalon Memory Mapped Slave		[clk1]	0x2000	0x3FFF			
		reset1	Reset Input		[clk1]					
<input checked="" type="checkbox"/>		jtag_uart	JTAG UART Intel FPGA IP							
		clk	Clock Input		pll_c0					
		reset	Reset Input		[clk]					
		avalon_jtag_slave	Avalon Memory Mapped Slave		[clk]	0x5038	0x503F			
		irq	Interrupt Sender		[clk]					
<input checked="" type="checkbox"/>		sysid	System ID Peripheral Intel FPGA IP							
		clk	Clock Input		pll_c0					
		reset	Reset Input		[clk]					
		control_slave	Avalon Memory Mapped Slave		[clk]	0x5030	0x5037			
<input checked="" type="checkbox"/>		leds	P10 (Parallel I/O) Intel FPGA IP							
		clk	Clock Input		pll_c0					
		reset	Reset Input		[clk]					
		s1	Avalon Memory Mapped Slave		[clk]	0x5000	0x501F			
		external_connection	Conduit	led						

4.2.2.10 Until these settings are applied, only 2 errors should be in the Platform Designer window, because the reset and exception vector are not set previously. To set these vectors, double click on the Nios II component **nios**. The Nios II Processor parameter editor will reopen.

4.2.2.11 Click on Vectors tab and set Reset Vector and Exception Vector to **onchip.s1**.



4.2.2.12 Review message window for remains errors.

At this point should be no remaining errors in the message window. If there are, please refer again to the previous steps to resolve them.

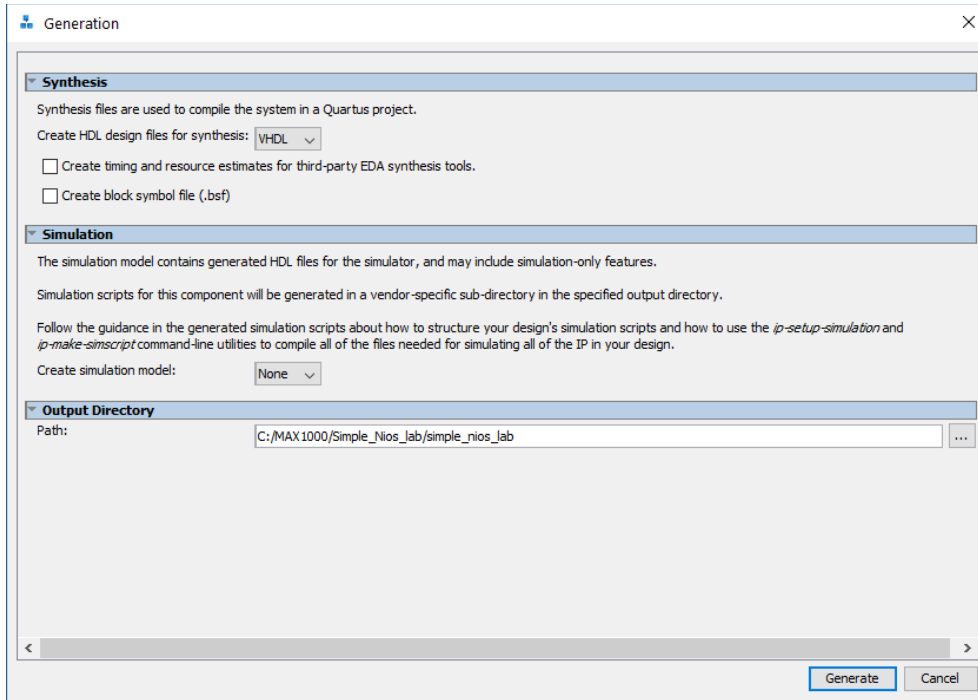
4.2.2.13 **Save** your design.

## 4.2.3 Generate the Platform Designer System

4.2.3.1 Select **Generate** → **Generate HDL...** from the menu or alternately click **Generate HDL...** button on the bottom right of the Platform Designer window.

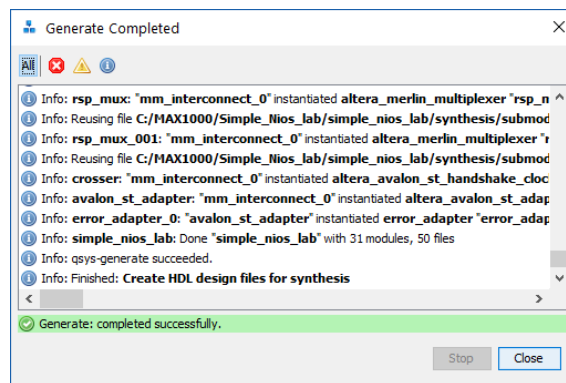
4.2.3.2 On the Generation window, enter the following information.

- Create HDL design files for synthesis: **VHDL**
- Uncheck Create timing and resource estimates for third-party EDA synthesis tools.
- Uncheck Create block symbol file (.bsf)
- Create simulation model: **None**



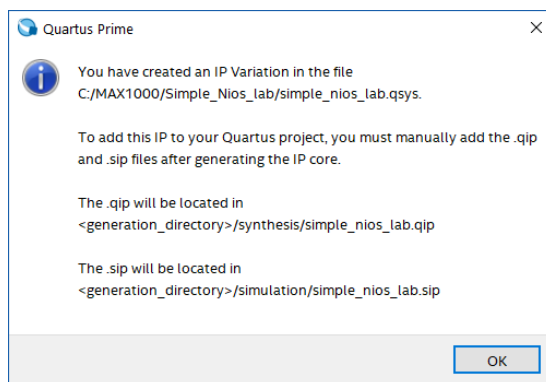
4.2.3.3 Click **Generate**.

4.2.3.4 When the generate process completed, click **Close**.



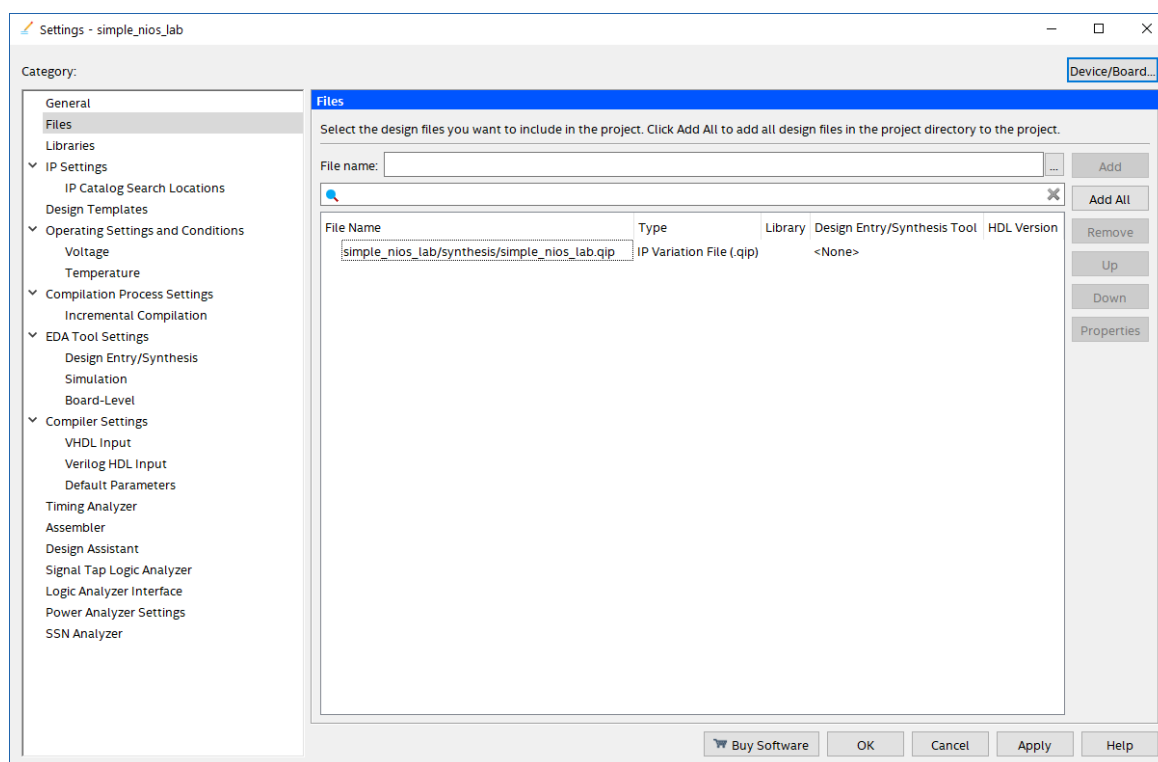
4.2.3.5 Click **Finish** button on the bottom right of the Platform Designer window.

4.2.3.6 It generates IP pointer files for both synthesis (.qip) and simulation (.sip) that will point Quartus to all the necessary design files needed to synthesize or simulate the Platform Designer system. Press **OK** to close as the .qip file will be added to the project in the following steps. Simulation will not be discussed in this lab, so no need to add the .sip file.



4.2.3.7 Choose **Project** → **Add/Remove Files in Project...** from the Quartus Prime menu.

4.2.3.8 Click on the button and browse through the synthesis directories:  
**<project\_directory>/simple\_nios\_lab/synthesis/** and open **simple\_nios\_lab.qip**.



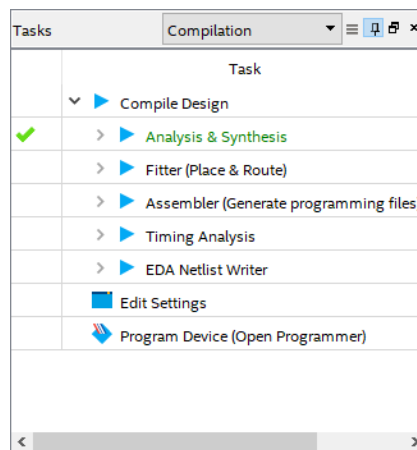
4.2.3.9 Click **Apply** and **OK**.

## 4.3 Compile design

### 4.3.1 Analysis and Synthesis

4.3.1.1 Run Analysis and Synthesis by clicking on button on the toolbars, or **Processing → Start → Analysis and Synthesis**.

There should be no errors. If there are errors, they should be fixed before continuing. If there are no errors the compilation task windows should look like this:



### 4.3.2 Pin Assignments

4.3.2.1 Open Pin Planner by clicking on button on the toolbars, or **Assignments → Pin Planner**.

**Top View - Wire Bond**  
MAX 10 - 10M08SAU169C8G

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	trict Preservatio
altera_reserved_tck	Input				2.5 V (default)		12mA (default)			
altera_reserved_tdi	Input				2.5 V (default)		12mA (default)			
altera_reserved_tdo	Output				2.5 V (default)		12mA (default)	2 (default)		
altera_reserved_tms	Input				2.5 V (default)		12mA (default)			
clk_clk	Input				2.5 V (default)		12mA (default)			
led_export[7]	Output				2.5 V (default)		12mA (default)	2 (default)		
led_export[9]	Output				2.5 V (default)		12mA (default)	2 (default)		
led_export[3]	Output				2.5 V (default)		12mA (default)	2 (default)		
led_export[4]	Output				2.5 V (default)		12mA (default)	2 (default)		
led_export[3]	Output				2.5 V (default)		12mA (default)	2 (default)		
led_export[2]	Output				2.5 V (default)		12mA (default)	2 (default)		
led_export[1]	Output				2.5 V (default)		12mA (default)	2 (default)		
led_export[0]	Output				2.5 V (default)		12mA (default)	2 (default)		
reset_reset_n	Input				2.5 V (default)		12mA (default)			



4.3.2.2 In the bottom table, type **PIN\_H6** in Location column of the clk\_clk.

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Tri-state Preservation
altera_reserved_tck	Input				2.5 V (default)		12mA (default)			
altera_reserved_tdi	Input				2.5 V (default)		12mA (default)			
altera_reserved_tdo	Output				2.5 V (default)		12mA (default)	2 (default)		
altera_reserved_tms	Input				2.5 V (default)		12mA (default)			
clk_clk	Input	PIN_H6	2	B2_NO	2.5 V (default)		12mA (default)			
led_export[7]	Output				2.5 V (default)		12mA (default)	2 (default)		
led_export[6]	Output				2.5 V (default)		12mA (default)	2 (default)		
led_export[5]	Output				2.5 V (default)		12mA (default)	2 (default)		
led_export[4]	Output				2.5 V (default)		12mA (default)	2 (default)		
led_export[3]	Output				2.5 V (default)		12mA (default)	2 (default)		
led_export[2]	Output				2.5 V (default)		12mA (default)	2 (default)		
led_export[1]	Output				2.5 V (default)		12mA (default)	2 (default)		
led_export[0]	Output				2.5 V (default)		12mA (default)	2 (default)		
reset_reset_n	Input				2.5 V (default)		12mA (default)			

4.3.2.3 Repeat the previous step with the following assignments:

Node Name	Pin Location
altera_reserved_tck	PIN_G2
altera_reserved_tdi	PIN_F5
altera_reserved_tdo	PIN_F6
altera_reserved_tms	PIN_G1
led_export[0]	PIN_A8
led_export[1]	PIN_A9
led_export[2]	PIN_A11
led_export[3]	PIN_A10
led_export[4]	PIN_B10
led_export[5]	PIN_C9
led_export[6]	PIN_C10
led_export[7]	PIN_D8
reset_reset_n	PIN_E6

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Tri-state Preservation
altera_reserved_tck	Input	PIN_g2	1B	B1_NO	2.5 V (default)		12mA (default)			
altera_reserved_tdi	Input	PIN_f5	1B	B1_NO	2.5 V (default)		12mA (default)			
altera_reserved_tdo	Output	PIN_f6	1B	B1_NO	2.5 V (default)		12mA (default)	2 (default)		
altera_reserved_tms	Input	PIN_g1	1B	B1_NO	2.5 V (default)		12mA (default)			
clk_clk	Input	PIN_H6	2	B2_NO	2.5 V (default)		12mA (default)			
led_export[0]	Output	PIN_a8	8	B8_NO	2.5 V (default)		12mA (default)	2 (default)		
led_export[1]	Output	PIN_a9	8	B8_NO	2.5 V (default)		12mA (default)	2 (default)		
led_export[2]	Output	PIN_a11	8	B8_NO	2.5 V (default)		12mA (default)	2 (default)		
led_export[3]	Output	PIN_a10	8	B8_NO	2.5 V (default)		12mA (default)	2 (default)		
led_export[4]	Output	PIN_b10	8	B8_NO	2.5 V (default)		12mA (default)	2 (default)		
led_export[5]	Output	PIN_c9	8	B8_NO	2.5 V (default)		12mA (default)	2 (default)		
led_export[6]	Output	PIN_c10	8	B8_NO	2.5 V (default)		12mA (default)	2 (default)		
led_export[7]	Output	PIN_d8	8	B8_NO	2.5 V (default)		12mA (default)	2 (default)		
reset_reset_n	Input	PIN_e6	8	B8_NO	2.5 V (default)		12mA (default)			



4.3.2.4 Double click in the I/O Standard column for any pins to open a drop-down list and change the 2.5V (default) to the specific I/O standard.

Node Name	Pin I/O Standard
altera_reserved_tck	3.3-V LVTTTL
altera_reserved_tdi	3.3-V LVTTTL
altera_reserved_tdo	3.3-V LVTTTL
altera_reserved_tms	3.3-V LVTTTL
clk_clk	3.3-V LVTTTL
led_export[0]	3.3-V LVTTTL
led_export[1]	3.3-V LVTTTL
led_export[2]	3.3-V LVTTTL
led_export[3]	3.3-V LVTTTL
led_export[4]	3.3-V LVTTTL
led_export[5]	3.3-V LVTTTL
led_export[6]	3.3-V LVTTTL
led_export[7]	3.3-V LVTTTL
reset_reset_n	3.3-V Schmitt Trigger

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Inert Preservatio
altera_reserved_tck	Input	PIN_G2	1B	B1_NO	3.3-V LVTTTL		8mA (default)			
altera_reserved_tdi	Input	PIN_F5	1B	B1_NO	3.3-V LVTTTL		8mA (default)			
altera_reserved_tdo	Output	PIN_F6	1B	B1_NO	3.3-V LVTTTL		8mA (default)	2 (default)		
altera_reserved_tms	Input	PIN_G1	1B	B1_NO	3.3-V LVTTTL		8mA (default)			
clk_clk	Input	PIN_H6	2	B2_NO	3.3-V LVTTTL		8mA (default)			
led_export[0]	Output	PIN_A8	8	B8_NO	3.3-V LVTTTL		8mA (default)	2 (default)		
led_export[1]	Output	PIN_A9	8	B8_NO	3.3-V LVTTTL		8mA (default)	2 (default)		
led_export[2]	Output	PIN_A11	8	B8_NO	3.3-V LVTTTL		8mA (default)	2 (default)		
led_export[3]	Output	PIN_A10	8	B8_NO	3.3-V LVTTTL		8mA (default)	2 (default)		
led_export[4]	Output	PIN_B10	8	B8_NO	3.3-V LVTTTL		8mA (default)	2 (default)		
led_export[5]	Output	PIN_C9	8	B8_NO	3.3-V LVTTTL		8mA (default)	2 (default)		
led_export[6]	Output	PIN_C10	8	B8_NO	3.3-V LVTTTL		8mA (default)	2 (default)		
led_export[7]	Output	PIN_D8	8	B8_NO	3.3-V LVTTTL		8mA (default)	2 (default)		
reset_reset_n	Input	PIN_E6	8	B8_NO	3.3-V Schmitt Trigger		8mA (default)			

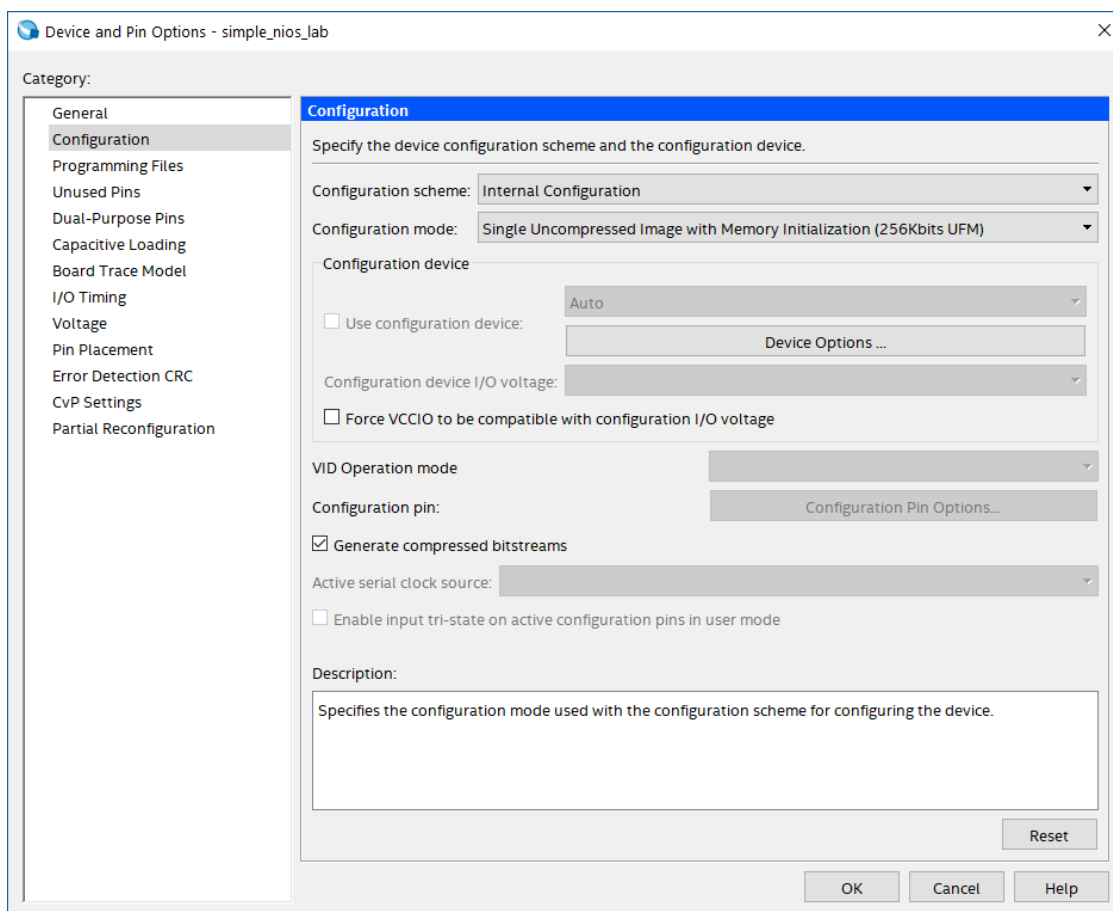
4.3.2.5 Close the Pin Planner, the settings are automatically saved.

### 4.3.3 Compiling the Design

4.3.3.1 Open the device settings window from **Assignments** → **Device...** and click **Device and Pin Options**.

4.3.3.2 Click to the **Configuration** category.

4.3.3.3 Set configuration mode to **Single Uncompressed Image with Memory Initialization (256kbits UFM)**.

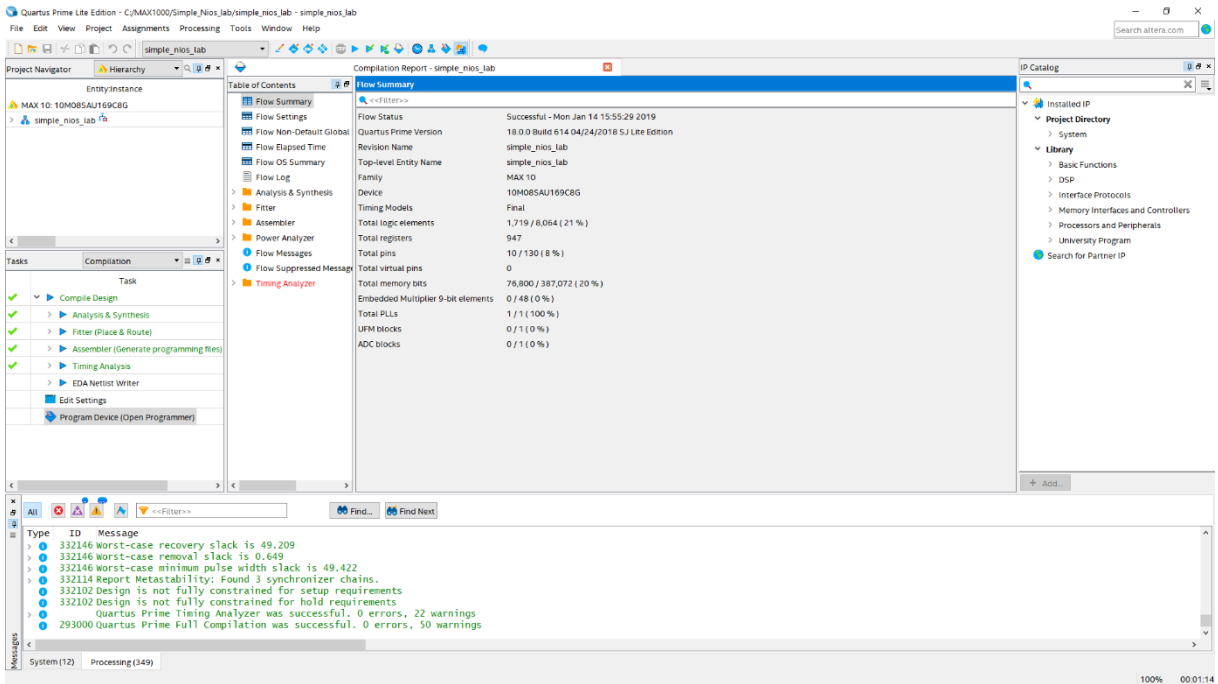


4.3.3.4 Press **OK** to close Device and Pin Options window. Press again **OK** to close Device window.

4.3.3.5 Start Compilation by clicking on  button on the toolbars, or **Processing** → **Start Compilation**.

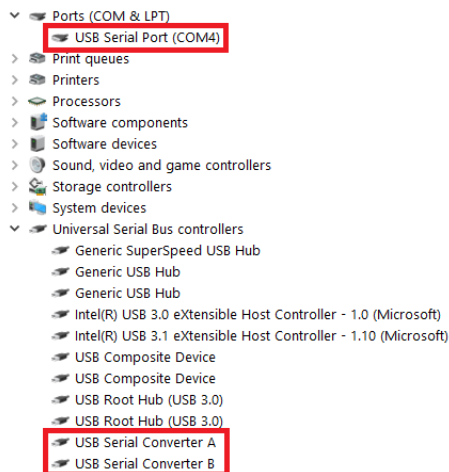


There should be no errors. If there are errors, they should be fixed before re-compiling. The 100% in the lower right corner or a green checkmark next to the Compile Design in the Compilation task window indicates that the compilation was successful.

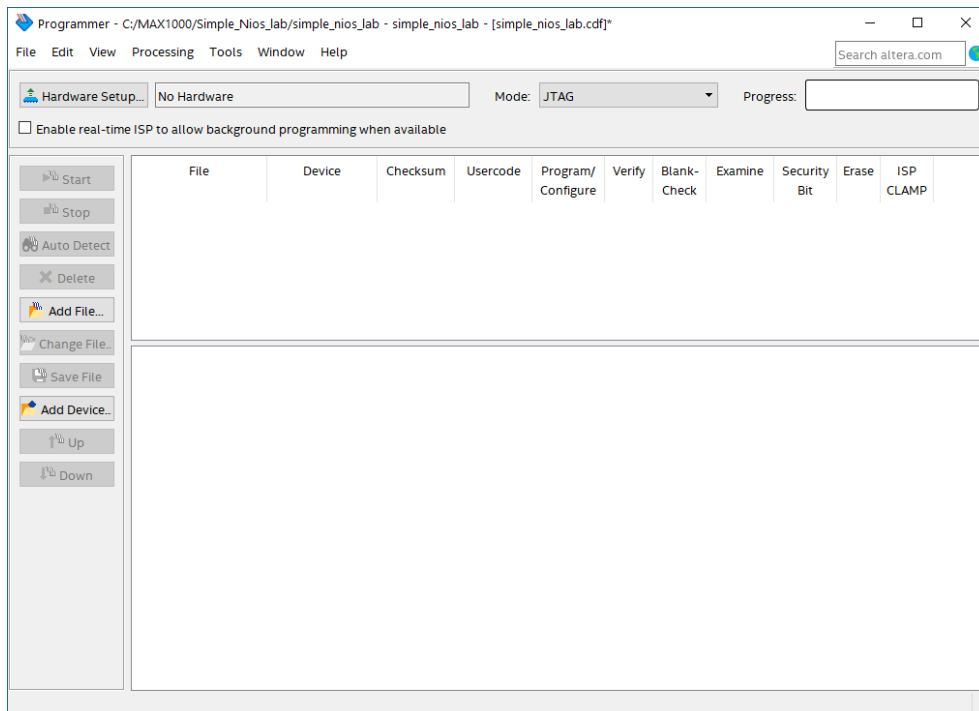


## 4.3.4 Configuration

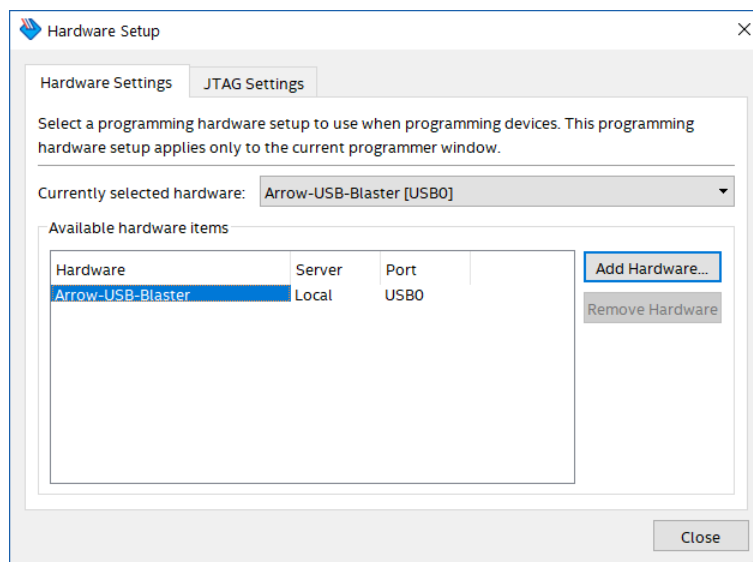
4.3.4.1 Connect your MAX1000 board to your PC using a USB cable. Since the Arrow USB Blaster should be already installed, the Window's Device Manager should display the following entries are highlighted in red (port number may differ depending on your PC):



4.3.4.2 Open the Quartus Prime Programmer from **Tools** → **Programmer** or double click on Program Device (Open Programmer) from the Task window.

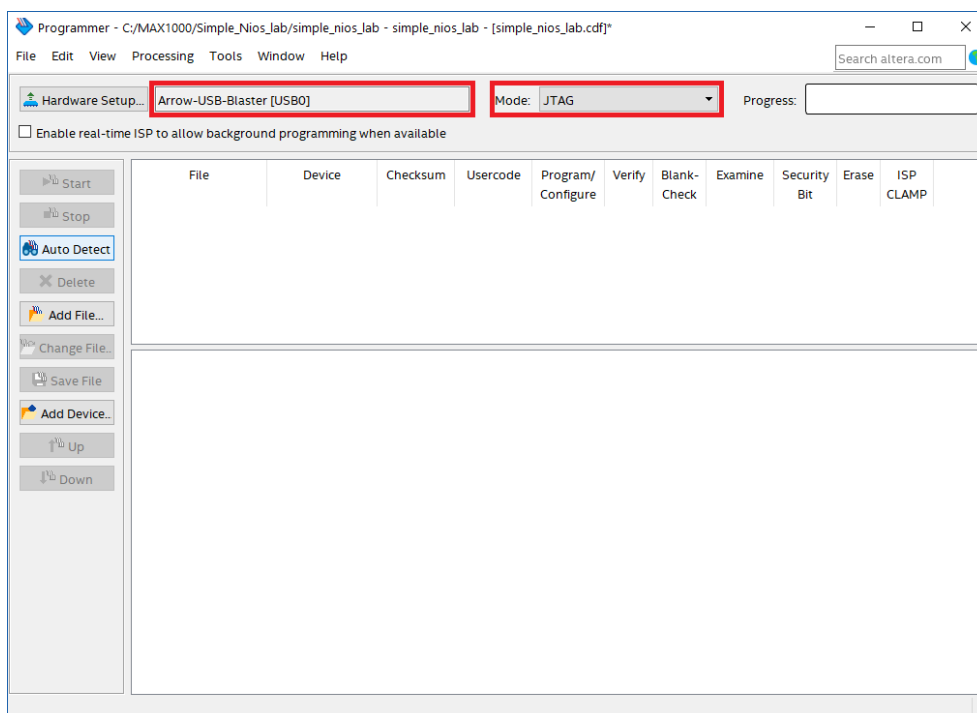


4.3.4.3 Click **Hardware Setup...** and double click **Arrow-USB-Blaster** entry in the Hardware Setup tab. The Currently selected hardware should now show Arrow-USB-Blaster [USB0] (depending on your PC, the USB port number may variant).



4.3.4.4 Click **Close**.

4.3.4.5 Make sure the hardware setup is Arrow-USB-Blaster [USB0] and the mode is JTAG. Click **Auto Detect**.



4.3.4.6 If the configuration has been added by default, you can skip the following steps and continue with the 4.3.4.11 point.

4.3.4.7 Select **10M08SA** device and click **OK** in the pop-up window.

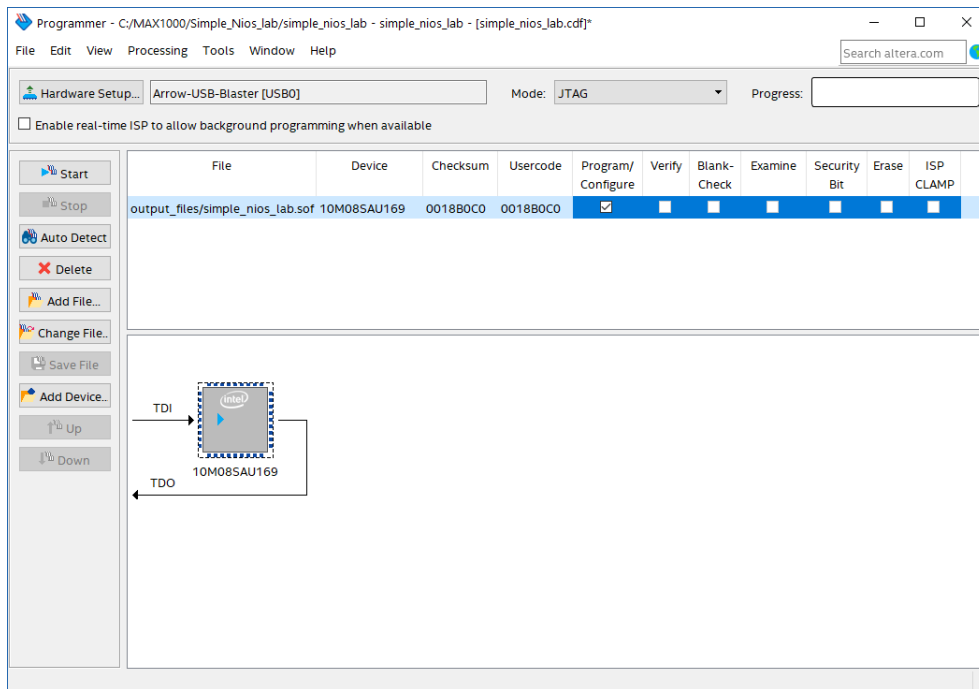


4.3.4.8 Click **Change File...** or double click <none> to choose the programming file.

4.3.4.9 Navigate to <project\_directory>/output\_files/ and select the **simple\_nios\_lab.sof** file.

4.3.4.10 Click **Open**.

4.3.4.11 Make sure the Programmer shows the correct file and the correct part in the JTAG chain and check the Program/Configure checkbox.



4.3.4.12 Click **Start** to program the board. When the configuration is complete, the Progress bar should show 100% (Successful).



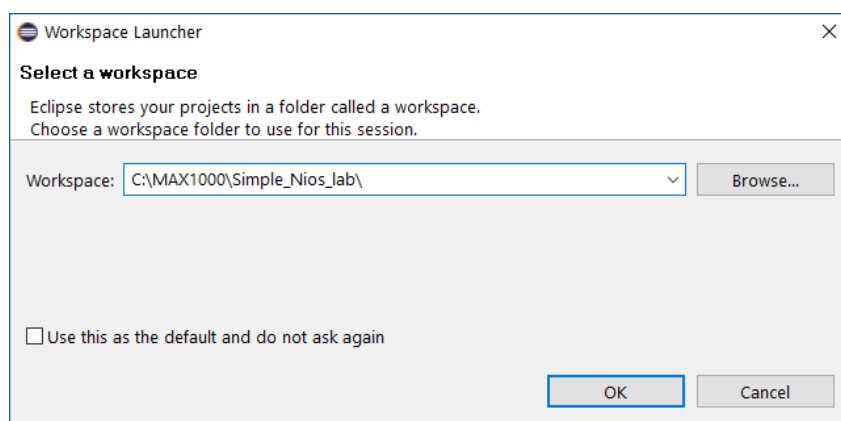
## 4.4 Software Design

**Overview:** In this section, you will use the Nios II Software Build Tools (SBT) for Eclipse to quickly create a board support package (BSP) and a C software application to run on the Nios II processor.

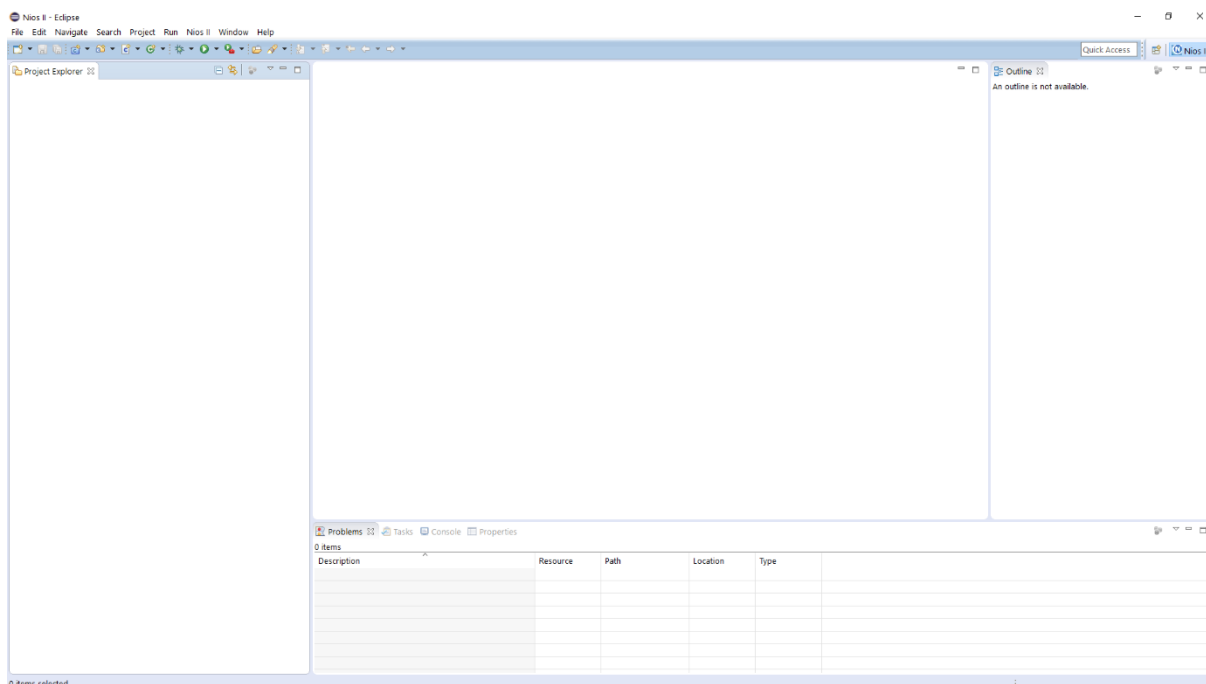
### 4.4.1 Create a new software project

4.4.1.1 From the main Quartus Prime window, start STB from **Tools → Nios II Software Build Tools for Eclipse**.


4.4.1.2 The Eclipse Workspace Launcher will open. Click **Browse...** and choose the directory of your project. In this case it was C:\MAX1000\Simple\_Nios\_lab\.

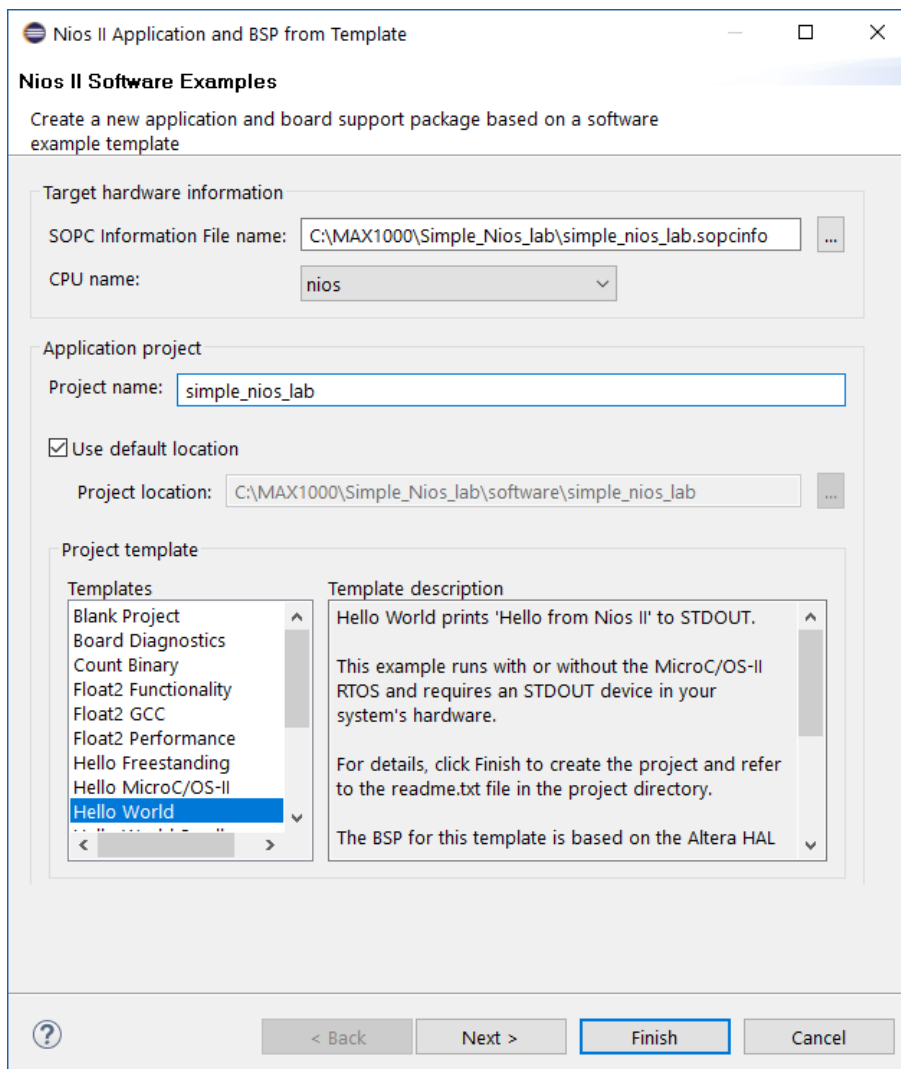


4.4.1.3 Click **OK** and the Eclipse will open.



4.4.1.4 Select **File** → **New** → **Nios II Application and BSP from Template**.

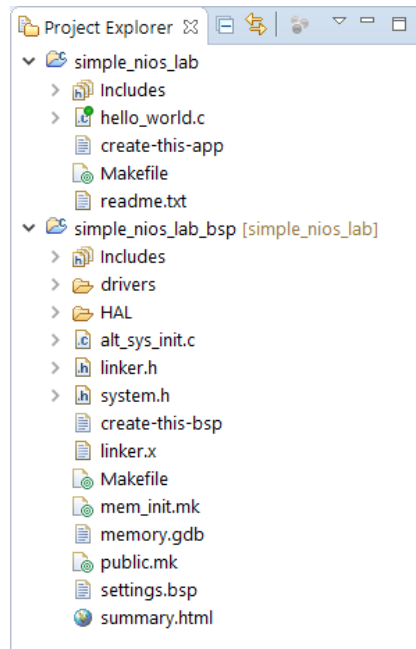
4.4.1.5 Click  to select the **simple\_nios\_lab.sopcinfo** from your project directory and name the project **simple\_nios\_lab**. Make sure you select **Hello World** from the Templates.



4.4.1.6 Click **Finish**.



4.4.1.7 Eclipse will create two directories in the workspace, one for the application project and one for the BSP. The application directory (simple\_nios\_lab) contains a hello\_world.c file while the BSP directory (simple\_nios\_lab\_bsp) contains software drivers, a system.h, header file and other software infrastructure.



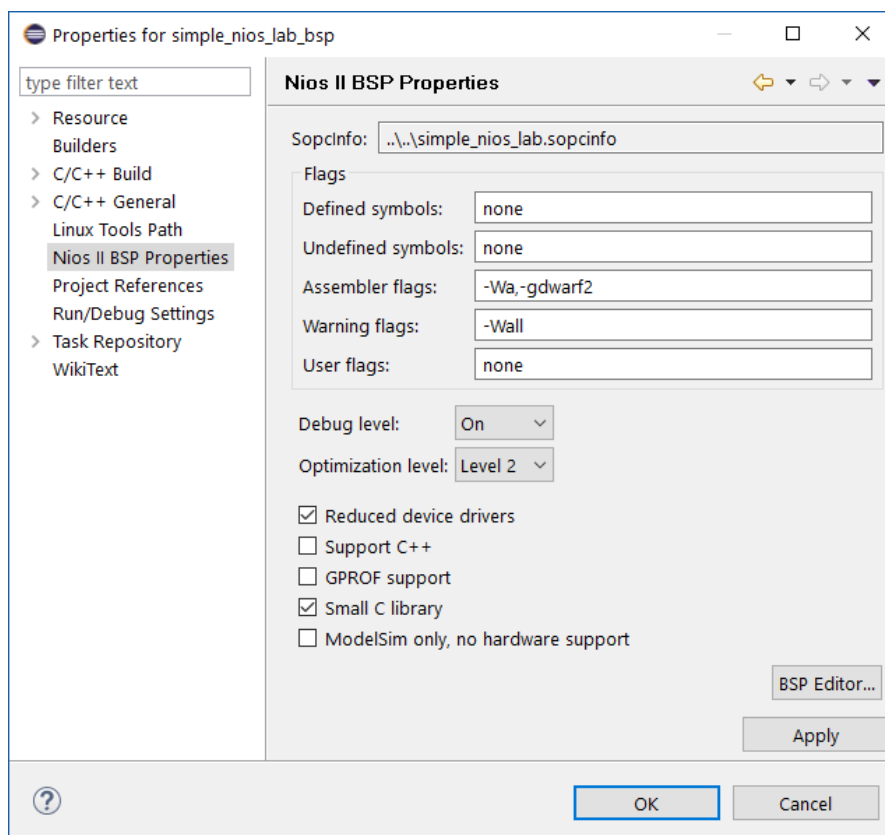
4.4.1.8 Open **hello\_world.c** file and read the code.  
This print 'Hello from Nios II!' to the display.

## 4.4.2 Build the software

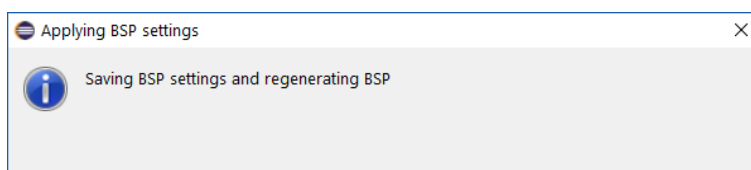
4.4.2.1 Right click on the simple\_nios\_lab\_bsp project and select **Properties** from the pop-up menu.

4.4.2.2 In the Properties window select the **Nios II BSP Properties** tab. It may take a moment to load the settings.

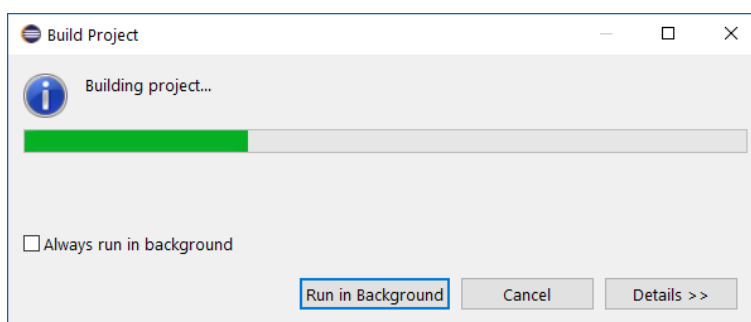
4.4.2.3 To keep the software footprint small so it fits our device, open the drop-down menu for Optimization level and change it to **Level 2**. Enable **Reduced device drivers** and **Small C library** options. As there is no C++ code, uncheck the Support C++ option.



4.4.2.4 Click **Apply**, and when it finished with Applying BSP Settings click **OK** to close Properties window.



4.4.2.5 Right click on the simple\_nios\_lab\_bsp project and select **Build Project** from the pop-up menu.



4.4.2.6 When it finished, repeat the previous step for the simple\_nios\_lab application project.

```

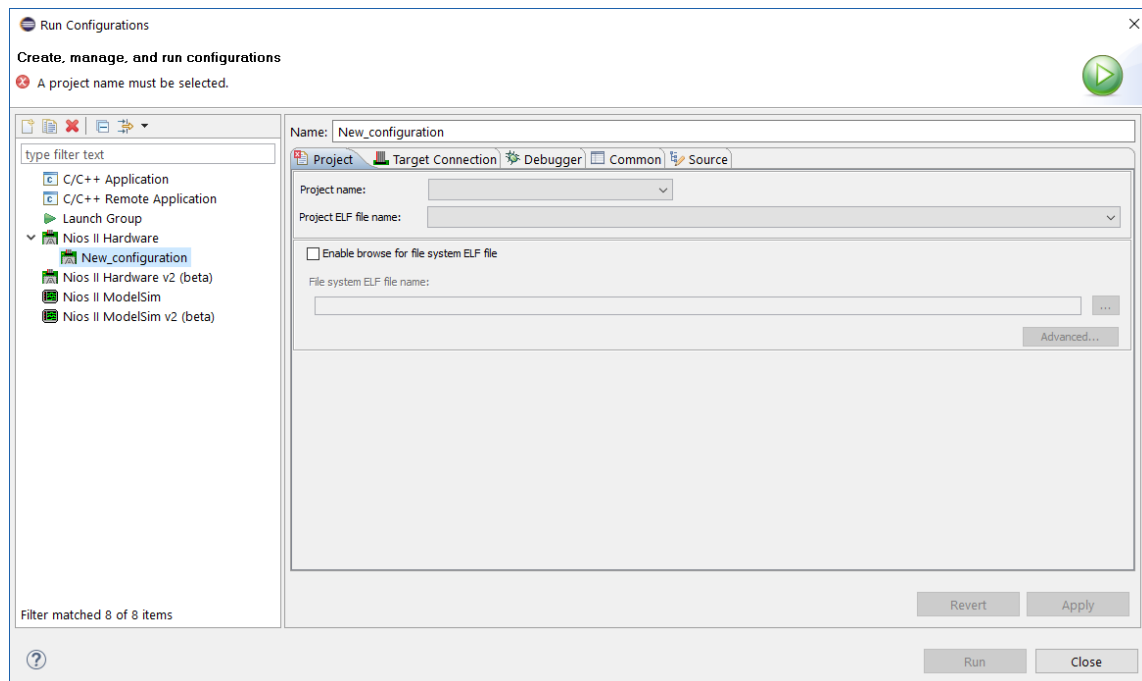
CDT Build Console [simple_nios_lab_bsp]
nios2-elf-gcc -xc -MP -MMD -c -I./HAL/inc -I. -I./drivers/inc -pipe -D_
Compiling altera_avalon_jtag_uart_write.c...
nios2-elf-gcc -xc -MP -MMD -c -I./HAL/inc -I. -I./drivers/inc -pipe -D_
Compiling altera_avalon_sysid_gsys.c...
nios2-elf-gcc -xc -MP -MMD -c -I./HAL/inc -I. -I./drivers/inc -pipe -D_
Creating libhal_bsp.a...
rm -f -f libhal_bsp.a
nios2-elf-ar -src libhal_bsp.a obj/HAL/src/alt_alarm_start.o obj/HAL/src/
[BSP build complete]

15:10:26 Build Finished (took 15s.967ms)

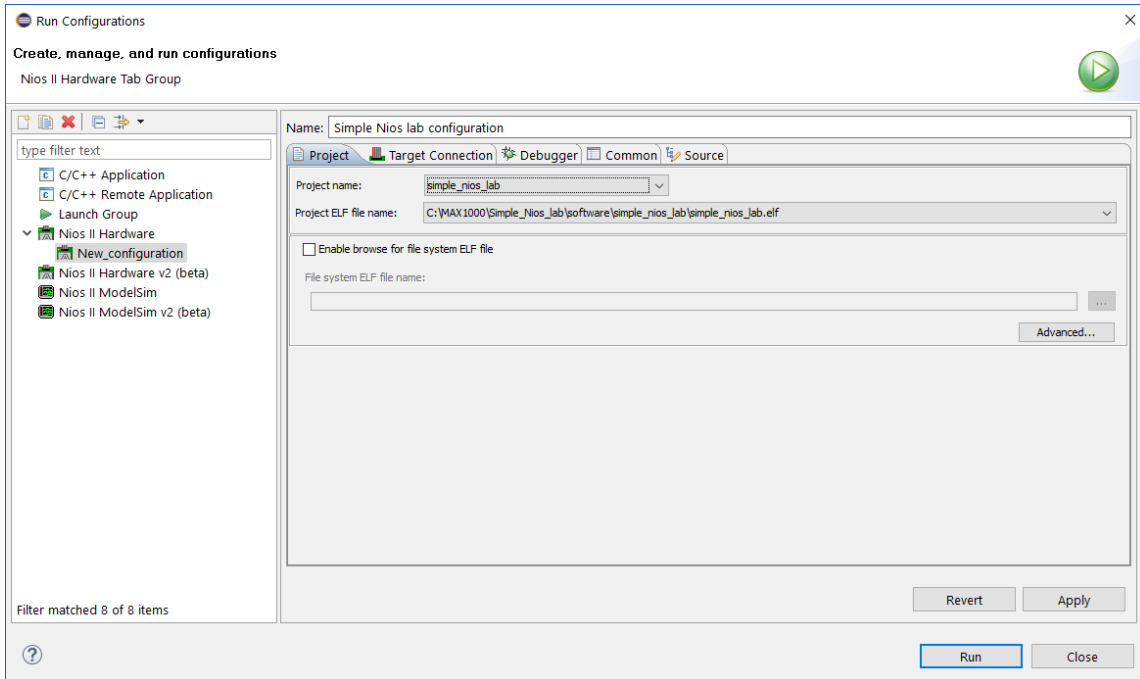
```

### 4.4.3 Run the application

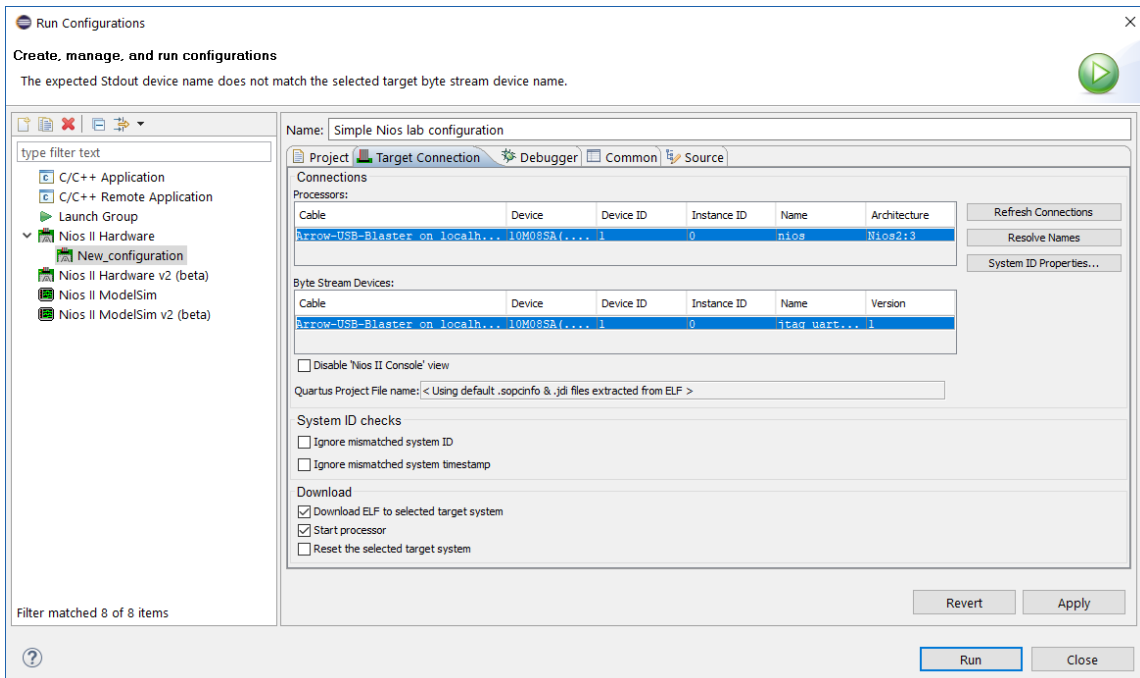
4.4.3.1 Select simple\_nios\_lab and go to **Run → Run Configurations...** and double click to **Nios II Hardware** to add a new configuration.



4.4.3.2 Rename it to **Simple Nios lab configuration** and on the Project tab select **simple\_nios\_lab** from the drop-down menu for the Project name.

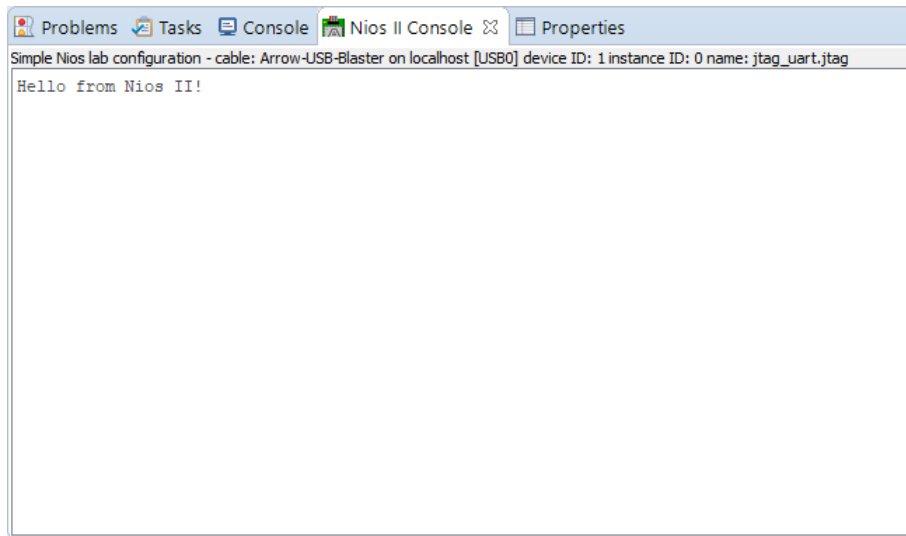


4.4.3.3 Click on the **Target Connection** tab and click **Refresh Connections** button. The configured MAX1000 board should appear.



4.4.3.4 Click **Apply** and **Run**.

4.4.3.5 After a few second, the Nios II Console should open at the bottom of the Eclipse



#### 4.4.4 Edit and Rerun the program

4.4.4.1 Stop the program running by clicking on  button on the top right corner of the Nios II Console window.





4.4.4.2 In the `hello_world.c` file replace the existing code with the following code:


```
#include <stdio.h>
#include "system.h"
#include "altera_avalon_pio_regs.h"

int main()
{
    int count = 1;
    int delay;
    printf("Hello from Nios II!\n");

    while (1)
    {
        IOWR_ALTERA_AVALON_PIO_DATA(LED0_BASE, count);
        for(delay=0; delay<65000; delay++);
        if(count<512)
        {
            count=count<<1;
        }
        else
        {
            count = 1;
        }
    }

    return 0;
}
```

4.4.4.3 **Save** the file.

4.4.4.4 Click on  button on the toolbar to run it on your board.

You do not need to build the project manually, the Nios II SBT for Eclipse automatically rebuilds the program before downloading it to the FPGA.

4.4.4.5 Observe the LEDs are running on your board.

**CONGRATULATIONS! YOU HAVE SUCCESSFULLY COMPLETED THE SIMPLE NIOS LAB!**



## 5 Revision History

Version	Change Log	Date of Change
V1.0	Initial Version	14/01/2019



## 6 Legal Disclaimer

### ARROW ELECTRONICS

#### EVALUATION BOARD LICENSE AGREEMENT

By using this evaluation board or kit (together with all related software, firmware, components, and documentation provided by Arrow, "Evaluation Board"), You ("You") are agreeing to be bound by the terms and conditions of this Evaluation Board License Agreement ("Agreement"). Do not use the Evaluation Board until You have read and agreed to this Agreement. Your use of the Evaluation Board constitutes Your acceptance of this Agreement.

#### PURPOSE

The purpose of this evaluation board is solely intended for evaluation purposes. Any use of the Board beyond these purposes is on your own risk. Furthermore, according the applicable law, the offering Arrow entity explicitly does not warrant, guarantee or provide any remedies to you with regard to the board.

#### LICENSE

Arrow grants You a non-exclusive, limited right to use the enclosed Evaluation Board offering limited features only for Your evaluation and testing purposes in a research and development setting. Usage in a live environment is prohibited. The Evaluation Board shall not be, in any case, directly or indirectly assembled as a part in any production of Yours as it is solely developed to serve evaluation purposes and has no direct function and is not a finished product.

#### EVALUATION BOARD STATUS

The Evaluation Board offers limited features allowing You only to evaluate and test purposes. The Evaluation Board is not intended for consumer or household use. You are not authorized to use the Evaluation Board in any production system, and it may not be offered for sale or lease, or sold, leased or otherwise distributed for commercial purposes.

#### OWNERSHIP AND COPYRIGHT

Title to the Evaluation Board remains with Arrow and/or its licensors. This Agreement does not involve any transfer of intellectual property rights ("IPR) for evaluation board. You may not remove any copyright or other proprietary rights notices without prior written authorization from Arrow or it licensors.

#### RESTRICTIONS AND WARNINGS

Before You handle or use the Evaluation Board, You shall comply with all such warnings and other instructions and employ reasonable safety precautions in using the Evaluation Board. Failure to do so may result in death, personal injury, or property damage.

You shall not use the Evaluation Board in any safety critical or functional safety testing, including but not limited to testing of life supporting, military or nuclear applications. Arrow expressly disclaims any responsibility for such usage which shall be made at Your sole risk.

#### WARRANTY

Arrow warrants that it has the right to provide the evaluation board to you. This warranty is provided by Arrow in lieu of all other warranties, written or oral, statutory, express or implied, including any warranty as to merchantability, non-infringement, fitness for any particular purpose, or uninterrupted or error-free operation, all of which are expressly disclaimed. The evaluation board is provided "as is" without any other rights or warranties, directly or indirectly.

You warrant to Arrow that the evaluation board is used only by electronics experts who understand the dangers of handling and using such items, you assume all responsibility and liability for any improper or unsafe handling or use of the evaluation board by you, your employees, affiliates, contractors, and designers.







### **LIMITATION OF LIABILITIES**

In no event shall Arrow be liable to you, whether in contract, tort (including negligence), strict liability, or any other legal theory, for any direct, indirect, special, consequential, incidental, punitive, or exemplary damages with respect to any matters relating to this agreement. In no event shall arrow's liability arising out of this agreement in the aggregate exceed the amount paid by you under this agreement for the purchase of the evaluation board.

### **IDENTIFICATION**

You shall, at Your expense, defend Arrow and its Affiliates and Licensors against a claim or action brought by a third party for infringement or misappropriation of any patent, copyright, trade secret or other intellectual property right of a third party to the extent resulting from (1) Your combination of the Evaluation Board with any other component, system, software, or firmware, (2) Your modification of the Evaluation Board, or (3) Your use of the Evaluation Board in a manner not permitted under this Agreement. You shall indemnify Arrow and its Affiliates and Licensors against and pay any resulting costs and damages finally awarded against Arrow and its Affiliates and Licensors or agreed to in any settlement, provided that You have sole control of the defense and settlement of the claim or action, and Arrow cooperates in the defense and furnishes all related evidence under its control at Your expense. Arrow will be entitled to participate in the defense of such claim or action and to employ counsel at its own expense.

### **RECYCLING**

The Evaluation Board is not to be disposed as an urban waste. At the end of its life cycle, differentiated waste collection must be followed, as stated in the directive 2002/96/EC. In all the countries belonging to the European Union (EU Dir. 2002/96/EC) and those following differentiated recycling, the Evaluation Board is subject to differentiated recycling at the end of its life cycle, therefore: It is forbidden to dispose the Evaluation Board as an undifferentiated waste or with other domestic wastes. Consult the local authorities for more information on the proper disposal channels. An incorrect Evaluation Board disposal may cause damage to the environment and is punishable by the law.