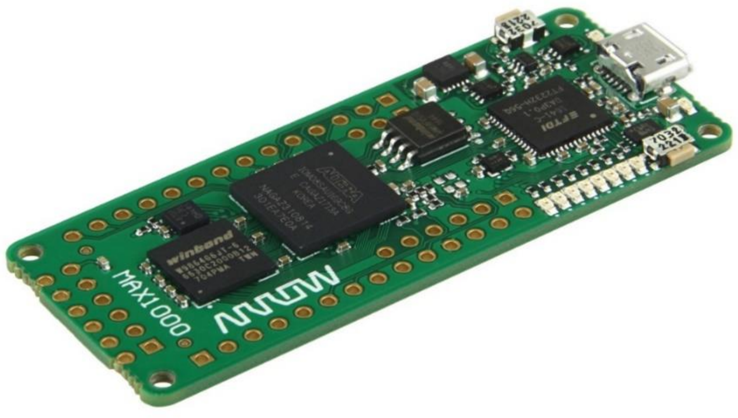
MAX1000 Embedded System Lab



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

Revision 3.0

1. Introduction

This tutorial provides comprehensive information to help you understand how to create a software project for a Nios II processor system in an Intel FPGA and run the software project on your MAX1000 board. The Nios II processor core 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. This tutorial introduces you to the basic software development flow for the Nios II processor.

Lab Overview:	This lab teaches you how to create an embedded system implemented in programmable logic. You will build a processor-based hardware system and run software on it. As the lab progresses, you will see how quick and easy it is to build entire systems using Quartus Platform Designer (formerly Qsys) tools to configure and integrate pre-verified IP blocks.
Project Details:	The lab will guide you through creating an embedded system using Platform Designer. This system will be able to retrieve data from internal ADC of MAX10 and the on-board accelerometer of the MAX1000. Depending on the data received by the Nios II processor, the LEDs will react to the Y-axis.
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.

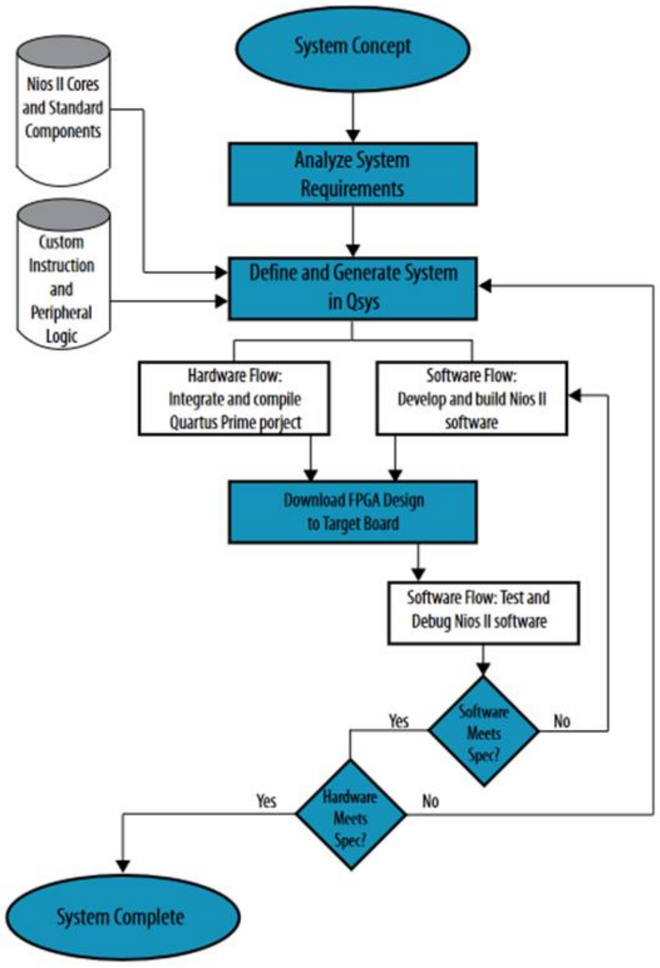
2. Getting Started

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
- Lab files: Embedded_System_lab_template: Template files are required to complete the lab. Includes: embedded_system_lab.vhd, embedded_system_lab.sdc, accelerometer.c, embedded_system_lab_pinout.csv
- 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.


System Concept
Nios II Cores and Standard Components
Analyze System Requirements
Custom Instruction and Peripheral Logic
Define and Generate System in Qsys
Hardware Flow: Integrate and compile Quartus Prime porject
Software Flow: Develop and build Nios II software
Download FPGA Design to Target Board
Software Flow: Test and Debug Nios II software
Yes
Software Meets Spec?
No
Yes
Hardware Meets Spec?
No
System Complete

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.

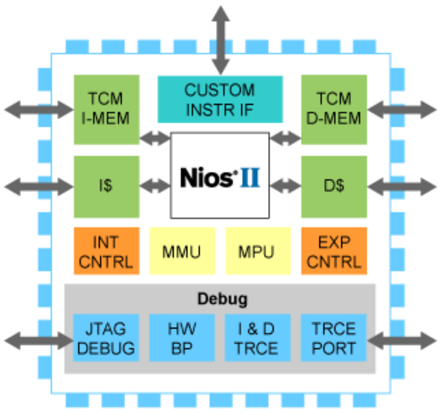
4. Nios II Soft Core

The Nios II processor delivers unprecedented flexibility for your cost-sensitive, real-time, safety-critical, ASIC-optimized, and applications processing needs. The Nios II processor supports all Intel® FPGA and SoC families.

Two different versions are available:

- NIOS II / f : License Fee, designed for high performance
- NIOS II / e : Royalty Free, designed for fewest FPGA logic and memory resources

There is a variety configuration options to choose from depending on the application's needs. More information on the Nios II Processor can be found at:
https://www.intel.com/content/www/us/en/products/programmable/processor/nios-ii.html



Nios II soft core can also support a variety of embedded operating systems, with more information found at:
https://www.intel.com/content/www/us/en/products/programmable/processor/nios-ii/ecosystem.html

5. Implementing Nios II soft core in MAX1000

In this module, you will create a Quartus Prime project for your embedded system design and create the software project to run on the Nios II processor.

We will be using Platform designer to add and interconnect different components. The following components will be included in our system:

- Clock source
- PLL
- Nios II Processor
- On-Chip memory
- SDRAM controller
- ADC for measuring internal temperature of MAX10
- SPI (3-Wire Serial for accelerometer interface)
- Parallel I/O for LED output

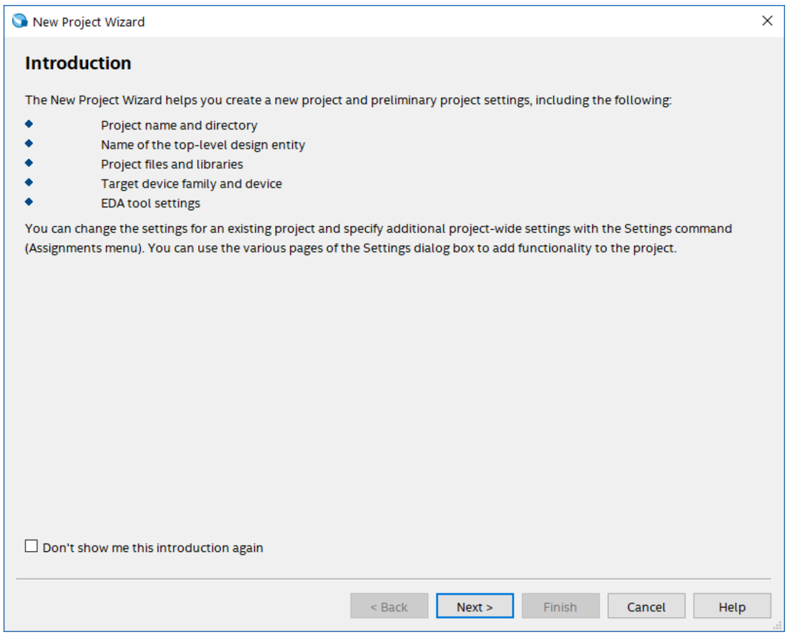
6. Project with MAX1000

6.1 Quartus Prime project

6.1.1 Create a new Quartus Prime project

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

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



New Project Wizard

Introduction

The New Project Wizard helps you create a new project and preliminary project settings, including the following:

- Project name and directory
- Name of the top-level design entity
- Project files and libraries
- Target device family and device
- EDA tool settings

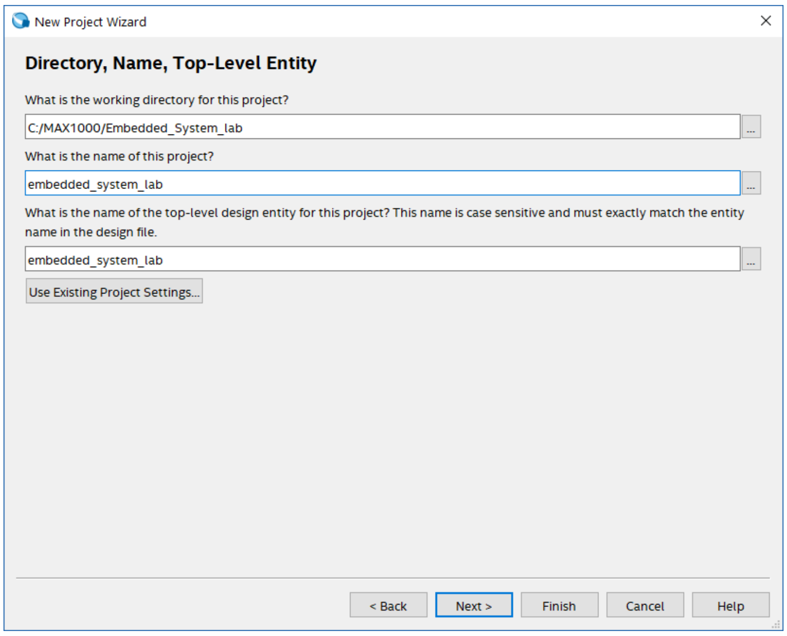
You can change the settings for an existing project and specify additional project-wide settings with the Settings command (Assignments menu). You can use the various pages of the Settings dialog box to add functionality to the project.

☐ Don't show me this introduction again

6.1.1.3 Click **Next**.

6.1.1.4 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/Embedded_System_lab**
- Specify the name of the project: **embedded_system_lab**
- Specify the name of the top-level entity: **embedded_system_lab**


New Project Wizard

Directory, Name, Top-Level Entity

What is the working directory for this project?

C:/MAX1000/Embedded_System_lab

What is the name of this project?

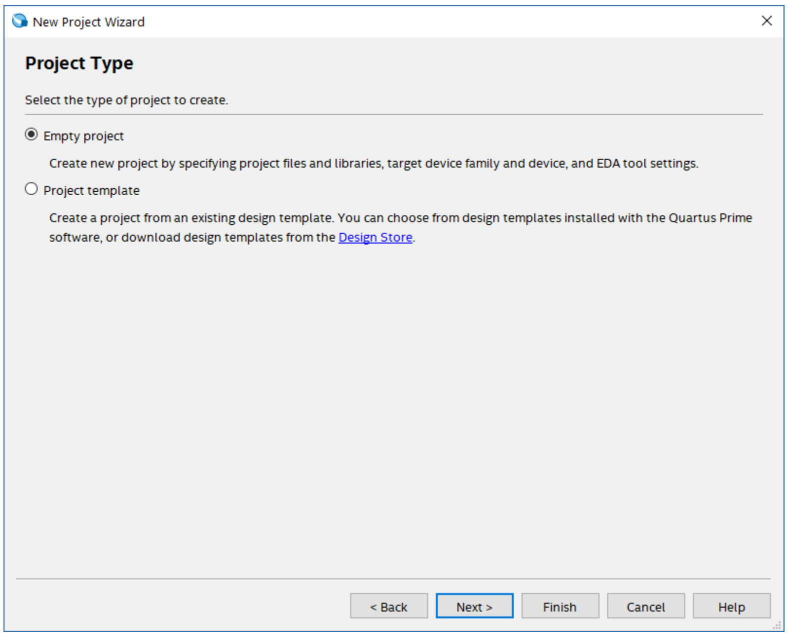
embedded_system_lab

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

embedded_system_lab

6.1.1.5 Click **Next**.

6.1.1.6 On the Project Type page, select **"Empty project"** and click **Next**.


New Project Wizard

Project Type

Select the type of project to create.

● Empty project

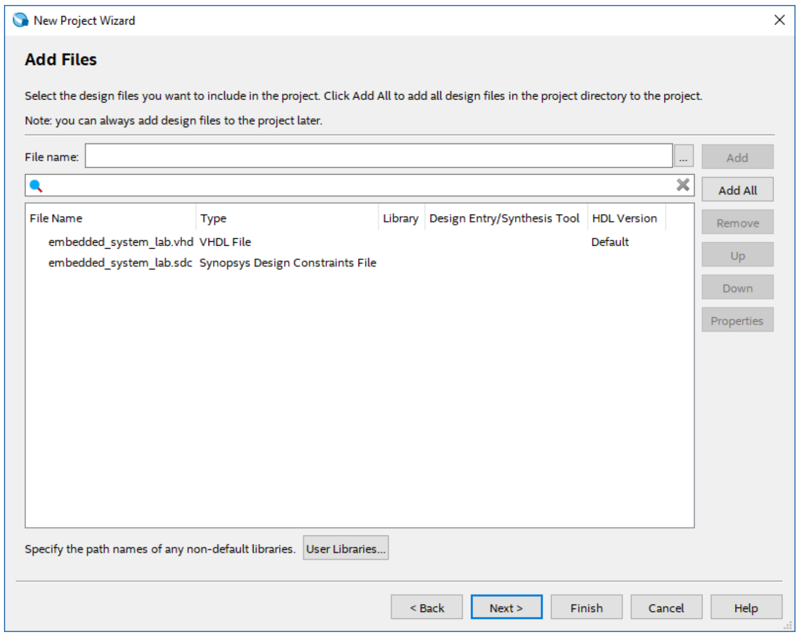
Create new project by specifying project files and libraries, target device family and device, and EDA tool settings.

○ Project template

Create a project from an existing design template. You can choose from design templates installed with the Quartus Prime software, or download design templates from the Design Store.

4.1.1.1 On the Add Files page, add source files to the project by clicking on the ... button and browse into the lab files folder where you will locate the provided design files and add:

- **embedded_system_lab.vhd**
- **embedded_system_lab.sdc**



New Project Wizard

Add Files

Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.

Note: you can always add design files to the project later.

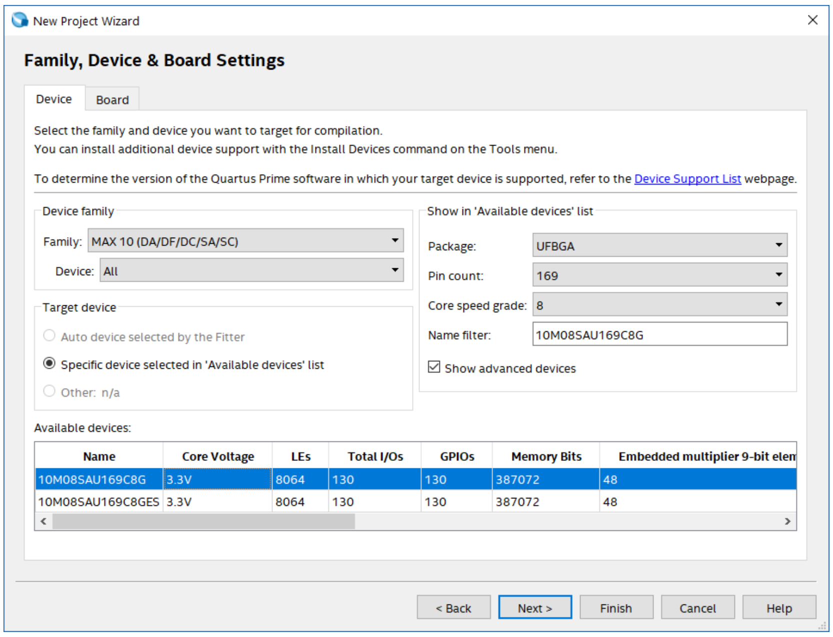
File name:

File Name	Type	Library	Design Entry/Synthesis Tool	HDL Version
embedded_system_lab.vhd	VHDL File			Default
embedded_system_lab.sdc	Synopsys Design Constraints File			

Specify the path names of any non-default libraries. User Libraries...

6.1.1.7 Click **Next**.

6.1.1.8 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**.



New Project Wizard

Family, Device & Board Settings

Device | Board

Select the family and device you want to target for compilation.
You can install additional device support with the Install Devices command on the Tools menu.

To determine the version of the Quartus Prime software in which your target device is supported, refer to the Device Support List webpage.

Device family
Family: MAX 10 (DA/DF/DC/SA/SC)
Device: All

Target device
Auto device selected by the Fitter
[✓ Specific device selected in 'Available devices' list]
Other: n/a

Show in 'Available devices' list
Package: UFBGA
Pin count: 169
Core speed grade: 8
Name filter: 10M08SAU169C8G
[✓] Show advanced devices

Available devices:

Name	Core Voltage	LEs	Total I/Os	GPIOs	Memory Bits	Embedded multiplier 9-bit elem
10M08SAU169C8G	3.3V	8064	130	130	387072	48
10M08SAU169C8GES	3.3V	8064	130	130	387072	48

6.1.1.9 Click **Finish**.

6.2 Design entry

Overview: In this module you will use Platform Designer system integration tool to design your hardware system. You will add standard and custom components, make interface connections, assign clocks and generate HDL for the system.

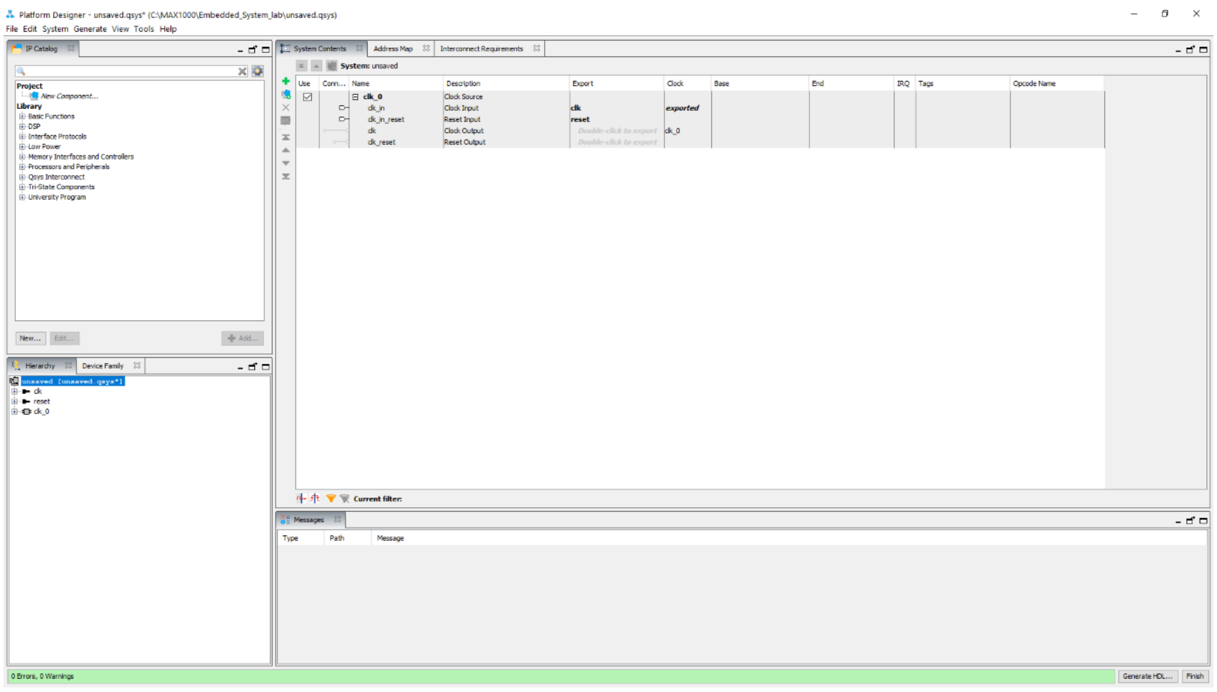
6.2.1 Add components to Platform Designer

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.).

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

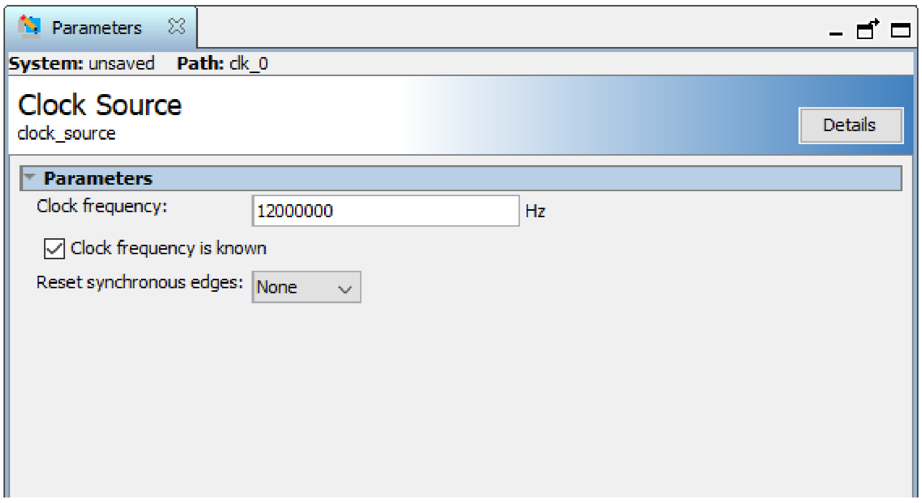
6.2.1.2 In the new window, you should see a single clock source component, named clk_0 in the System Components tab. This tab shows all the components currently in your system.



Use	Conn...	Name	Description	Export	Clock	Base	End	IRQ	Tags	Opcode Name
✓		clk_0	Clock Source							
		clk_in	Clock Input	clk	exported					
		clk_in_reset	Reset Input	reset						
		clk	Clock Output	Double-click to export	clk_0					
		clk_reset	Reset Output	Double-click to export						

6.2.1.3 Double click on clk_0.

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


Parameters

System: unsaved **Path:** clk_0

Clock Source
clock_source

Details

Parameters

Clock frequency: 12000000 Hz

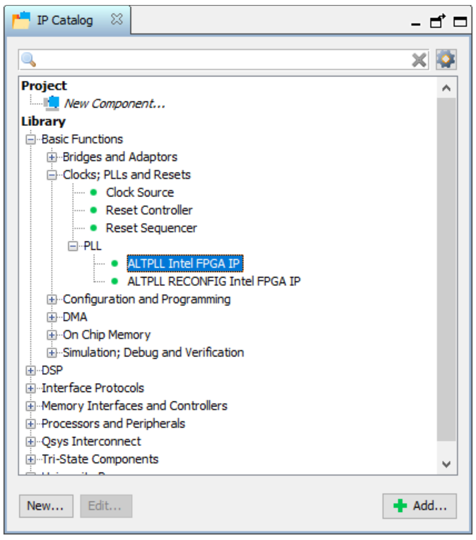
☑ Clock frequency is known

Reset synchronous edges: None

6.2.1.6 Click on **X** on the parameter tab to close the parameter window.

6.2.1.7 Right click on the clk_0 and select **Rename**. Rename the clock source to **clk12mhz** and press Enter.

6.2.1.8 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**.

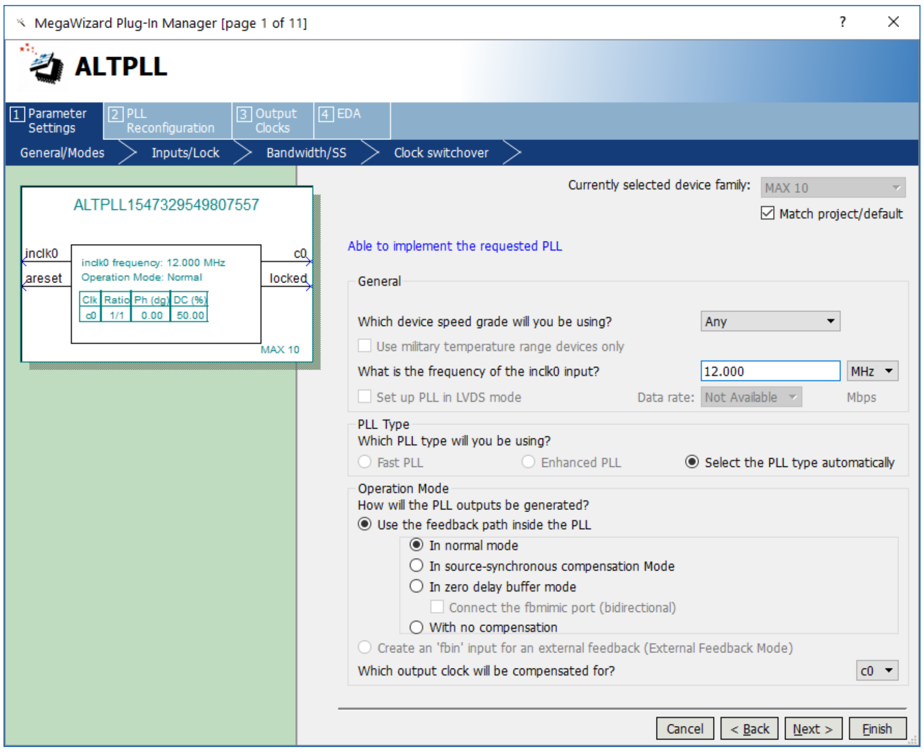

IP Catalog

Project
- *New Component...*

Library
- Basic Functions
  - Bridges and Adaptors
  - Clocks; PLLs and Resets
    - Clock Source
    - Reset Controller
    - Reset Sequencer
    - PLL
      - ALTPLL Intel FPGA IP
      - ALTPLL RECONFIG Intel FPGA IP
  - Configuration and Programming
  - DMA
  - On Chip Memory
  - Simulation; Debug and Verification
- DSP
- Interface Protocols
- Memory Interfaces and Controllers
- Processors and Peripherals
- Qsys Interconnect
- Tri-State Components

New... Edit... Add...

6.2.1.9 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.


MegaWizard Plug-In Manager [page 1 of 11]

ALTPLL

1 Parameter Settings | 2 PLL Reconfiguration | 3 Output Clocks | 4 EDA

General/Modes > Inputs/Lock > Bandwidth/SS > Clock switchover >

ALTPLL1547329549807557

inclk0 frequency: 12.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	1/1	0.00	50.00

Currently selected device family: MAX 10
☑ Match project/default

Able to implement the requested PLL

General

Which device speed grade will you be using? Any
☐ Use military temperature range devices only
What is the frequency of the inclk0 input? 12.000 MHz
☐ Set up PLL in LVDS mode Data rate: Not Available Mbps

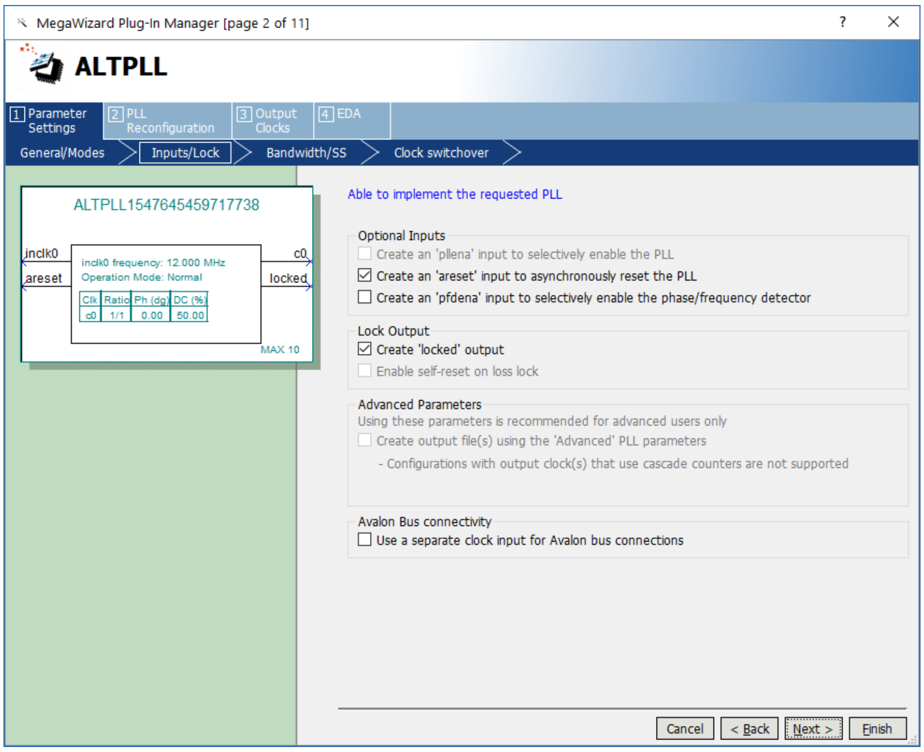
PLL Type
Which PLL type will you be using?
○ Fast PLL ○ Enhanced PLL ● Select the PLL type automatically

Operation Mode
How will the PLL outputs be generated?
● Use the feedback path inside the PLL
● In normal mode
○ In source-synchronous compensation Mode
○ In zero delay buffer mode
☐ Connect the fbmimic port (bidirectional)
○ With no compensation
○ Create an 'fbin' input for an external feedback (External Feedback Mode)
Which output clock will be compensated for? c0

Cancel | < Back | Next > | Finish

6.2.1.10 Click **Next**.

6.2.1.11 In Input/Lock tab (page 2 of 11) make sure that areset and locked output option are checked.


MegaWizard Plug-In Manager [page 2 of 11]

ALTPLL

1 Parameter Settings | 2 PLL Reconfiguration | 3 Output Clocks | 4 EDA

General/Modes > Inputs/Lock > Bandwidth/SS > Clock switchover >

ALTPLL1547645459717738

inclk0 frequency: 12.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	1/1	0.00	50.00

Able to implement the requested PLL

Optional Inputs
☐ Create an 'pllena' input to selectively enable the PLL
☑ Create an 'areset' input to asynchronously reset the PLL
☐ Create an 'pfdena' input to selectively enable the phase/frequency detector

Lock Output
☑ Create 'locked' output
☐ Enable self-reset on loss lock

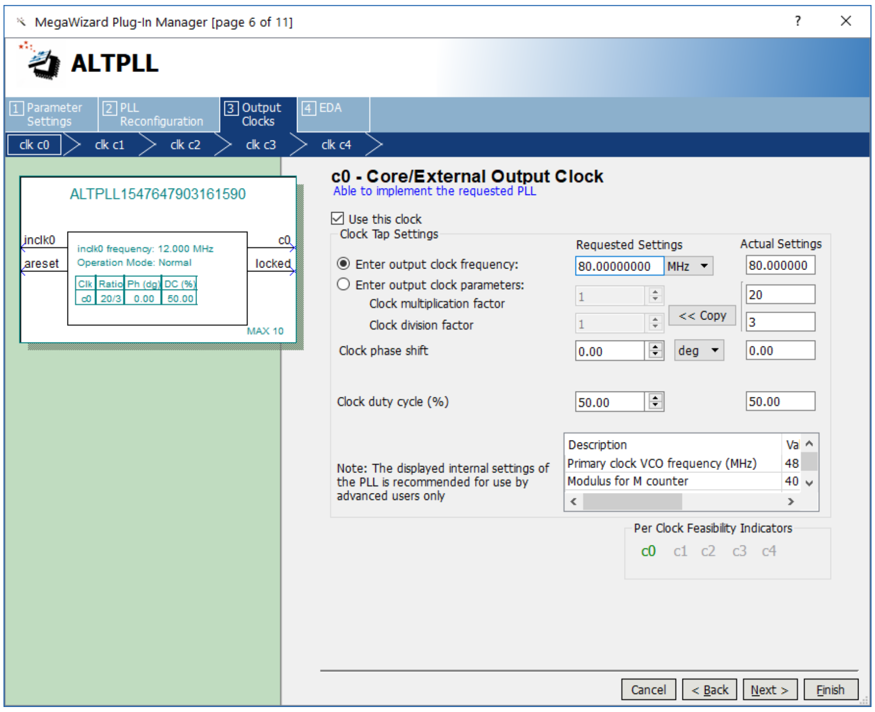
Advanced Parameters
Using these parameters is recommended for advanced users only
☐ Create output file(s) using the 'Advanced' PLL parameters
- Configurations with output clock(s) that use cascade counters are not supported

Avalon Bus connectivity
☐ Use a separate clock input for Avalon bus connections

Cancel | < Back | Next > | Finish

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

6.2.1.13 Under the clk c0 tab (page 6 of 11) select “Enter output clock frequency” and enter **80 MHz**.



MegaWizard Plug-In Manager [page 6 of 11]

ALTPLL

1 Parameter Settings | 2 PLL Reconfiguration | 3 Output Clocks | 4 EDA

clk c0 | clk c1 | clk c2 | clk c3 | clk c4

ALTPLL1547647903161590

inclk0 frequency: 12.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	20/3	0.00	50.00

c0 - Core/External Output Clock
Able to implement the requested PLL

☑ Use this clock

Clock Tap Settings

	Requested Settings		Actual Settings
● Enter output clock frequency:	80.00000000	MHz	80.000000
○ Enter output clock parameters:			
Clock multiplication factor	1	<< Copy	20
Clock division factor	1		3
Clock phase shift	0.00	deg	0.00
Clock duty cycle (%)	50.00		50.00

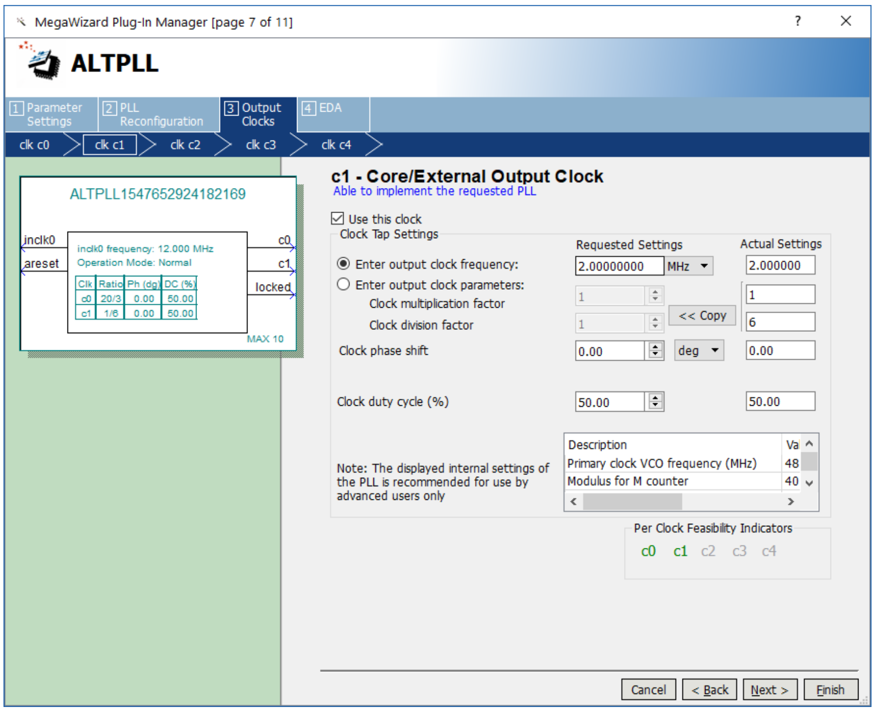
Note: The displayed internal settings of the PLL is recommended for use by advanced users only

Description	Va
Primary clock VCO frequency (MHz)	48
Modulus for M counter	40

Per Clock Feasibility Indicators: c0 c1 c2 c3 c4

6.2.1.14 Click **Next**.

6.2.1.15 Under the clk c1 tab (page 7 of 11) select **Use this clock** and enter **2 MHz** for the output clock frequency.



MegaWizard Plug-In Manager [page 7 of 11]

ALTPLL

1 Parameter Settings | 2 PLL Reconfiguration | 3 Output Clocks | 4 EDA

clk c0 | clk c1 | clk c2 | clk c3 | clk c4

ALTPLL1547652924182169

inclk0 frequency: 12.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	20/3	0.00	50.00
c1	1/6	0.00	50.00

c1 - Core/External Output Clock
Able to implement the requested PLL

☑ Use this clock

Clock Tap Settings

	Requested Settings		Actual Settings
● Enter output clock frequency:	2.00000000	MHz	2.000000
○ Enter output clock parameters:			
Clock multiplication factor	1	<< Copy	1
Clock division factor	1		6
Clock phase shift	0.00	deg	0.00
Clock duty cycle (%)	50.00		50.00

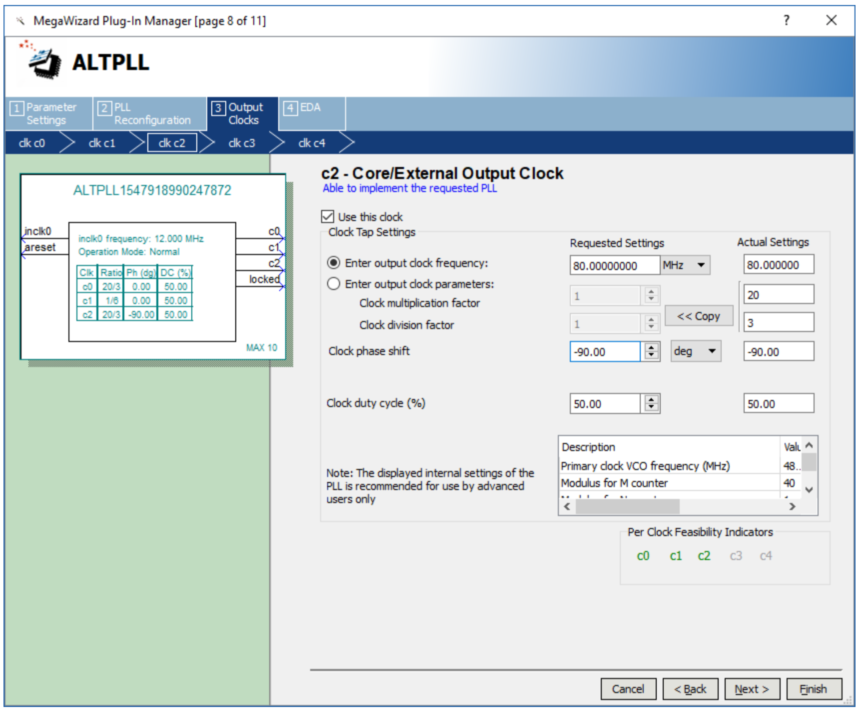
Note: The displayed internal settings of the PLL is recommended for use by advanced users only

Description	Va
Primary clock VCO frequency (MHz)	48
Modulus for M counter	40

Per Clock Feasibility Indicators: c0 c1 c2 c3 c4

6.2.1.16 Click **Next**.

6.2.1.17 Under the clk c2 tab (page 8 of 11) select **Use this clock** and enter **80 MHz** for the output clock frequency. Set the Clock phase shift to **-90 deg**.


MegaWizard Plug-In Manager [page 8 of 11]

ALTPLL

inclk0 frequency: 12.000 MHz
Operation Mode: Normal

Clk	Ratio	Ph (dg)	DC (%)
c0	20/3	0.00	50.00
c1	1/6	0.00	50.00
c2	20/3	-90.00	50.00

c2 - Core/External Output Clock
Able to implement the requested PLL

☑ Use this clock

Clock Tap Settings

	Requested Settings	Actual Settings
● Enter output clock frequency:	80.00000000 MHz	80.000000
○ Enter output clock parameters:		
Clock multiplication factor	1	20
Clock division factor	1	3
Clock phase shift	-90.00 deg	-90.00
Clock duty cycle (%)	50.00	50.00

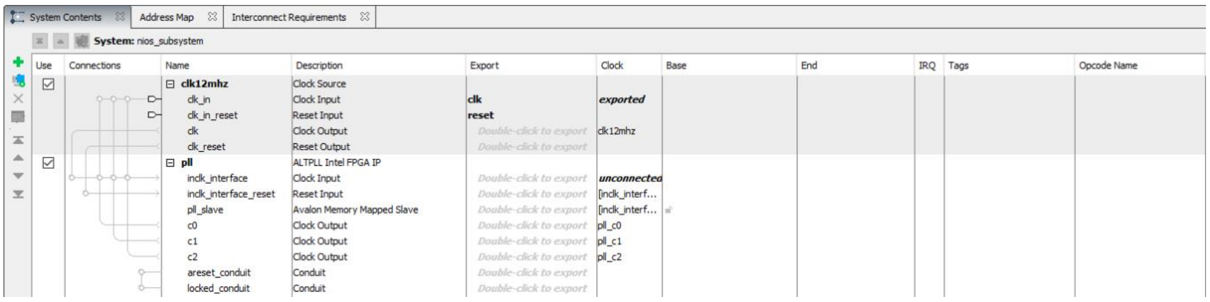
Note: The displayed internal settings of the PLL is recommended for use by advanced users only

Description	Valu
Primary clock VCO frequency (MHz)	48..
Modulus for M counter	40

Per Clock Feasibility Indicators
c0 c1 c2 c3 c4

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

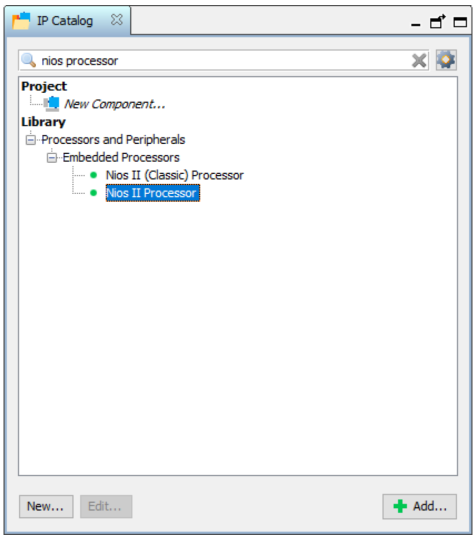
6.2.1.19 A component entitled altpll_0 should appear under Module Name. Rename this component to **pll**.



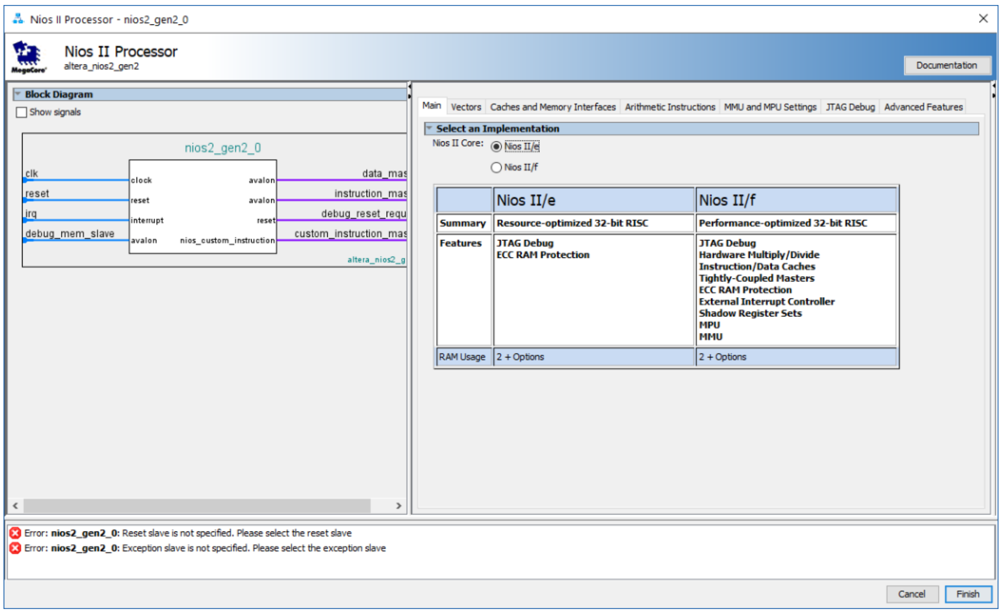
System: nios_subsystem

Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags	Opcode Name
☑		⊟ **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*						
☑		⊟ **pll**	ALTPLL Intel FPGA IP							
		inclk_interface	Clock Input	*Double-click to export*	***unconnected***					
		inclk_interface_reset	Reset Input	*Double-click to export*	[inclk_interf...					
		pll_slave	Avalon Memory Mapped Slave	*Double-click to export*	[inclk_interf...					
		c0	Clock Output	*Double-click to export*	pll_c0					
		c1	Clock Output	*Double-click to export*	pll_c1					
		c2	Clock Output	*Double-click to export*	pll_c2					
		areset_conduit	Conduit	*Double-click to export*						
		locked_conduit	Conduit	*Double-click to export*						

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



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



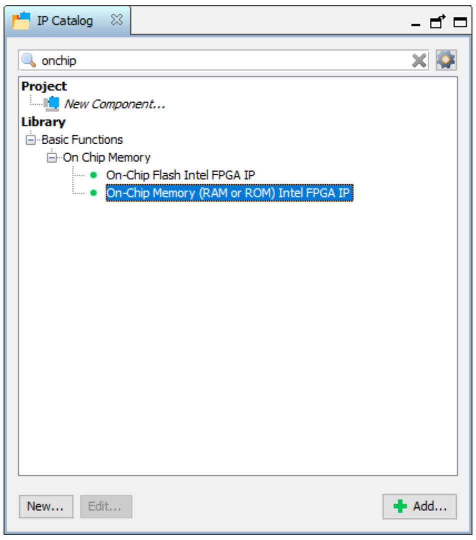
Nios II Core: [✓ Nios II/e] Nios II/f

	Nios II/e	Nios II/f
Summary	**Resource-optimized 32-bit RISC**	**Performance-optimized 32-bit RISC**
Features	**JTAG Debug** **ECC RAM Protection**	**JTAG Debug** **Hardware Multiply/Divide** **Instruction/Data Caches** **Tightly-Coupled Masters** **ECC RAM Protection** **External Interrupt Controller** **Shadow Register Sets** **MPU** **MMU**
RAM Usage	2 + Options	2 + Options

Error: **nios2_gen2_0:** Reset slave is not specified. Please select the reset slave
Error: **nios2_gen2_0:** Exception slave is not specified. Please select the exception slave

6.2.1.22 Rename nios2_gen2_0 to **nios**.

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

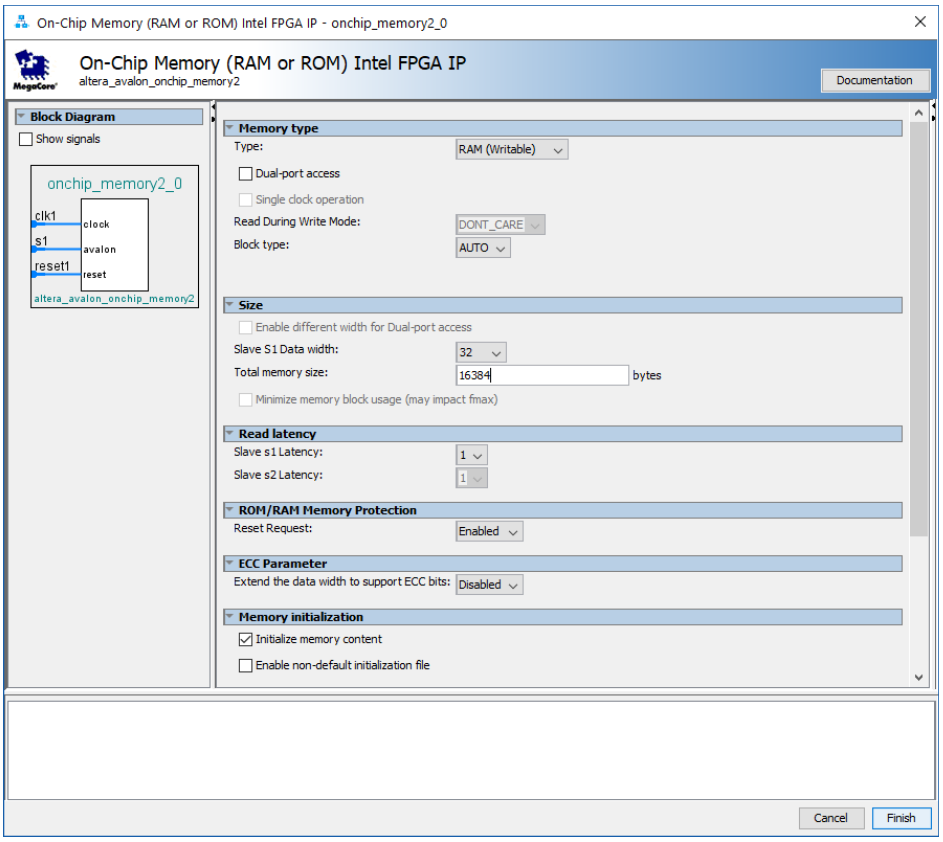

IP Catalog

onchip

Project
New Component...
Library
Basic Functions
On Chip Memory
On-Chip Flash Intel FPGA IP
On-Chip Memory (RAM or ROM) Intel FPGA IP

New... Edit... Add...

6.2.1.24 Change the Total memory size to **16384 bytes**.


On-Chip Memory (RAM or ROM) Intel FPGA IP - onchip_memory2_0

On-Chip Memory (RAM or ROM) Intel FPGA IP
altera_avalon_onchip_memory2

Memory type
Type: RAM (Writable)
Dual-port access
Single clock operation
Read During Write Mode: DONT_CARE
Block type: AUTO

Size
Enable different width for Dual-port access
Slave S1 Data width: 32
Total memory size: 16384 bytes
Minimize memory block usage (may impact fmax)

Read latency
Slave s1 Latency: 1
Slave s2 Latency: 1

ROM/RAM Memory Protection
Reset Request: Enabled

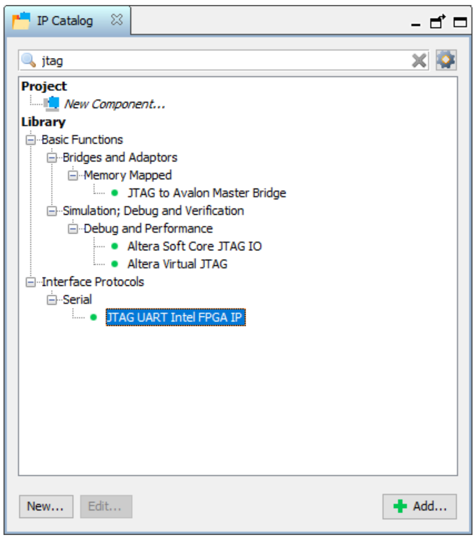
ECC Parameter
Extend the data width to support ECC bits: Disabled

Memory initialization
[✓] Initialize memory content
Enable non-default initialization file

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

6.2.1.26 Rename onchip_memory2_0 to **onchip**.

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



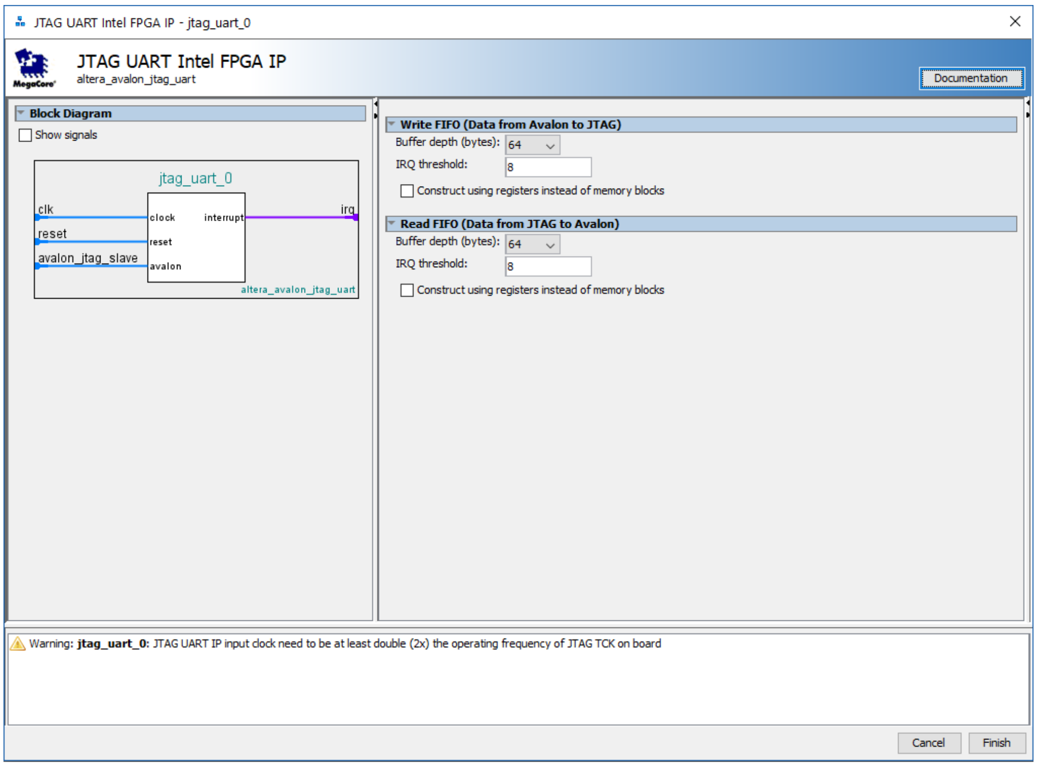
IP Catalog

jtag

Project
- New Component...

Library
- Basic Functions
  - Bridges and Adaptors
    - Memory Mapped
      - JTAG to Avalon Master Bridge
  - Simulation; Debug and Verification
    - Debug and Performance
      - Altera Soft Core JTAG IO
      - Altera Virtual JTAG
- Interface Protocols
  - Serial
    - JTAG UART Intel FPGA IP

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



JTAG UART Intel FPGA IP - jtag_uart_0

JTAG UART Intel FPGA IP
altera_avalon_jtag_uart

Block Diagram
- [ ] Show signals

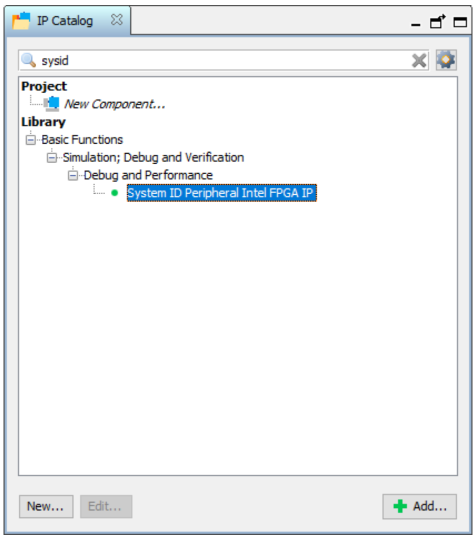
Write FIFO (Data from Avalon to JTAG)
Buffer depth (bytes): 64
IRQ threshold: 8
- [ ] Construct using registers instead of memory blocks

Read FIFO (Data from JTAG to Avalon)
Buffer depth (bytes): 64
IRQ threshold: 8
- [ ] Construct using registers instead of memory blocks

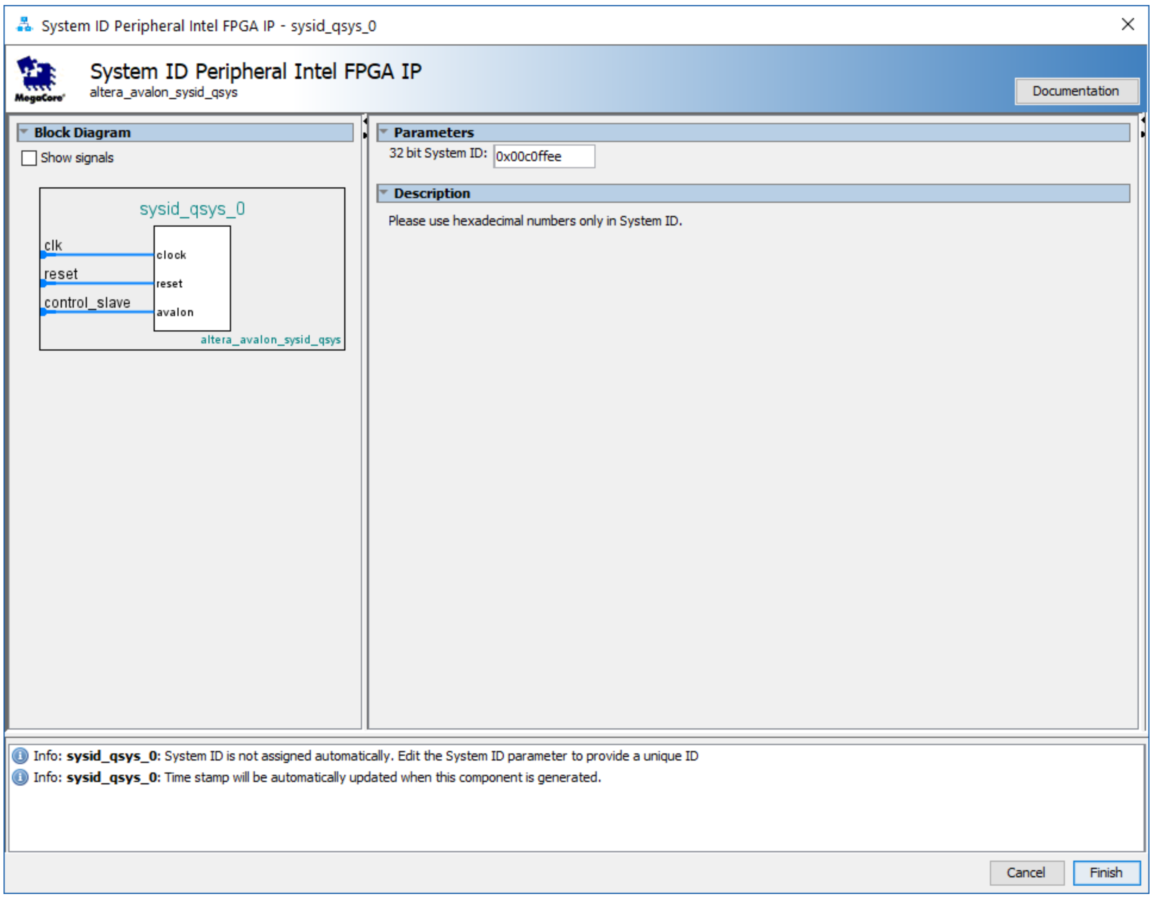
Warning: **jtag_uart_0:** JTAG UART IP input clock need to be at least double (2x) the operating frequency of JTAG TCK on board

6.2.1.29 Rename jtag_uart_0 to **jtag_uart**.

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



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



32 bit System ID: 0x00c0ffee

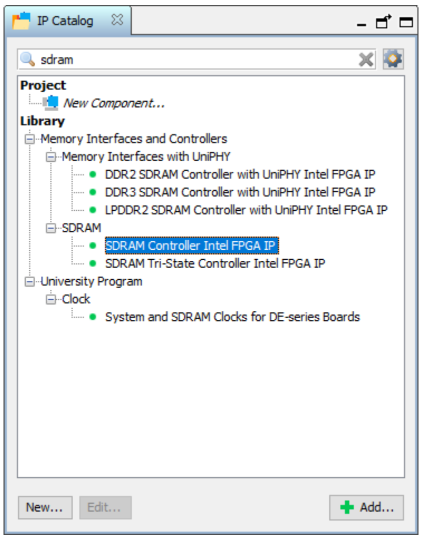
Please use hexadecimal numbers only in System ID.

Info: **sysid_qsys_0:** System ID is not assigned automatically. Edit the System ID parameter to provide a unique ID
Info: **sysid_qsys_0:** Time stamp will be automatically updated when this component is generated.

6.2.1.32 Rename sysid_qsys_0 to **sysid**.

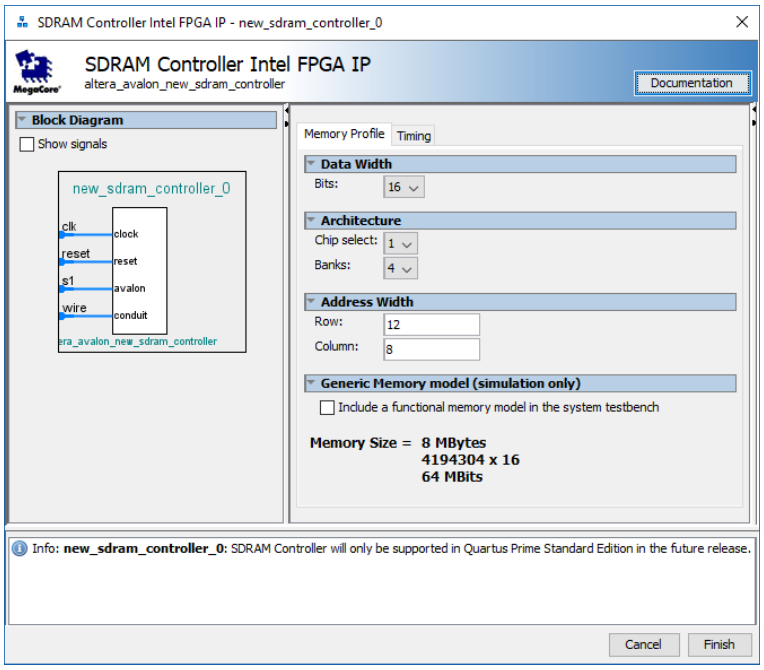
6.2.1.33 In the search bar of the IP Catalog, type "sdram" and add **SDRAM Controller Intel FPGA IP.**

Note: This lab was done by Quartus Prime Lite Edition version 18.0. However, SDRAM Controller will only be supported in Quartus Prime Standard Edition in the future. If you are using a newer Quartus Prime Lite version and this IP is not available, then skip the following steps and continue with 6.2.1.36.



IP Catalog
sdram
Project
New Component...
Library
Memory Interfaces and Controllers
Memory Interfaces with UniPHY
DDR2 SDRAM Controller with UniPHY Intel FPGA IP
DDR3 SDRAM Controller with UniPHY Intel FPGA IP
LPDDR2 SDRAM Controller with UniPHY Intel FPGA IP
SDRAM
SDRAM Controller Intel FPGA IP
SDRAM Tri-State Controller Intel FPGA IP
University Program
Clock
System and SDRAM Clocks for DE-series Boards
New...
Edit...
Add...

6.2.1.34 Under the Memory Profile tab set **16 bits** and accept the defaults for the remaining fields and click on the **Timing tab**.



SDRAM Controller Intel FPGA IP - new_sdram_controller_0
SDRAM Controller Intel FPGA IP
altera_avalon_new_sdram_controller
Documentation
Block Diagram
Show signals
new_sdram_controller_0
clk
reset
s1
wire
clock
reset
avalon
conduit
ra_avalon_new_sdram_controller
Memory Profile
Timing
Cancel
Finish

Data Width
Bits: 16

Architecture
Chip select: 1
Banks: 4

Address Width
Row: 12
Column: 8

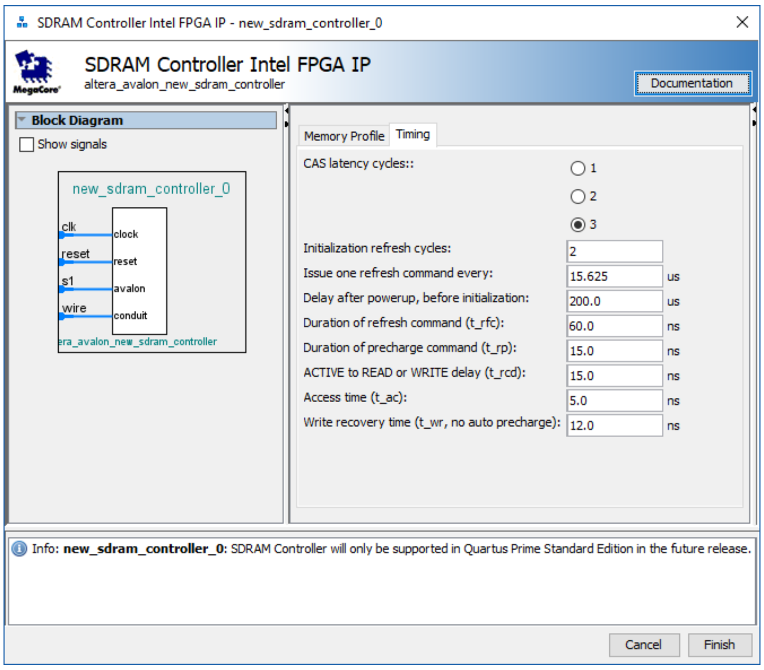
Generic Memory model (simulation only)
☐ Include a functional memory model in the system testbench

Memory Size = 8 MBytes
4194304 x 16
64 MBits

Info: **new_sdram_controller_0:** SDRAM Controller will only be supported in Quartus Prime Standard Edition in the future release.

6.2.1.35 On the Timing tab, set the following parameters:

- Delay after powerup, before initialization: **200 µs**
- Duration of refresh command: **60 ns**
- Duration of precharge command: **15 ns**
- ACTIVE to READ or WRITE delay: **15 ns**
- Access time: **5 ns**
- Write recovery time: **12 ns**


SDRAM Controller Intel FPGA IP - new_sdram_controller_0
SDRAM Controller Intel FPGA IP
altera_avalon_new_sdram_controller
Documentation
Block Diagram
Show signals
new_sdram_controller_0
clk
clock
reset
reset
s1
avalon
wire
conduit
era_avalon_new_sdram_controller
Memory Profile
Timing
Cancel
Finish

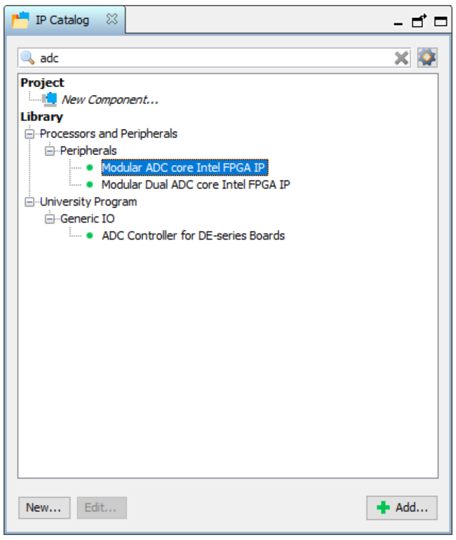
CAS latency cycles::	○ 1 ○ 2 ● 3	
Initialization refresh cycles:	2	
Issue one refresh command every:	15.625	us
Delay after powerup, before initialization:	200.0	us
Duration of refresh command (t_rfc):	60.0	ns
Duration of precharge command (t_rp):	15.0	ns
ACTIVE to READ or WRITE delay (t_rcd):	15.0	ns
Access time (t_ac):	5.0	ns
Write recovery time (t_wr, no auto precharge):	12.0	ns

Info: **new_sdram_controller_0:** SDRAM Controller will only be supported in Quartus Prime Standard Edition in the future release.

6.2.1.36 Click **Finish**.

6.2.1.37 Rename new_sdram_controller_0 to **sdram**.

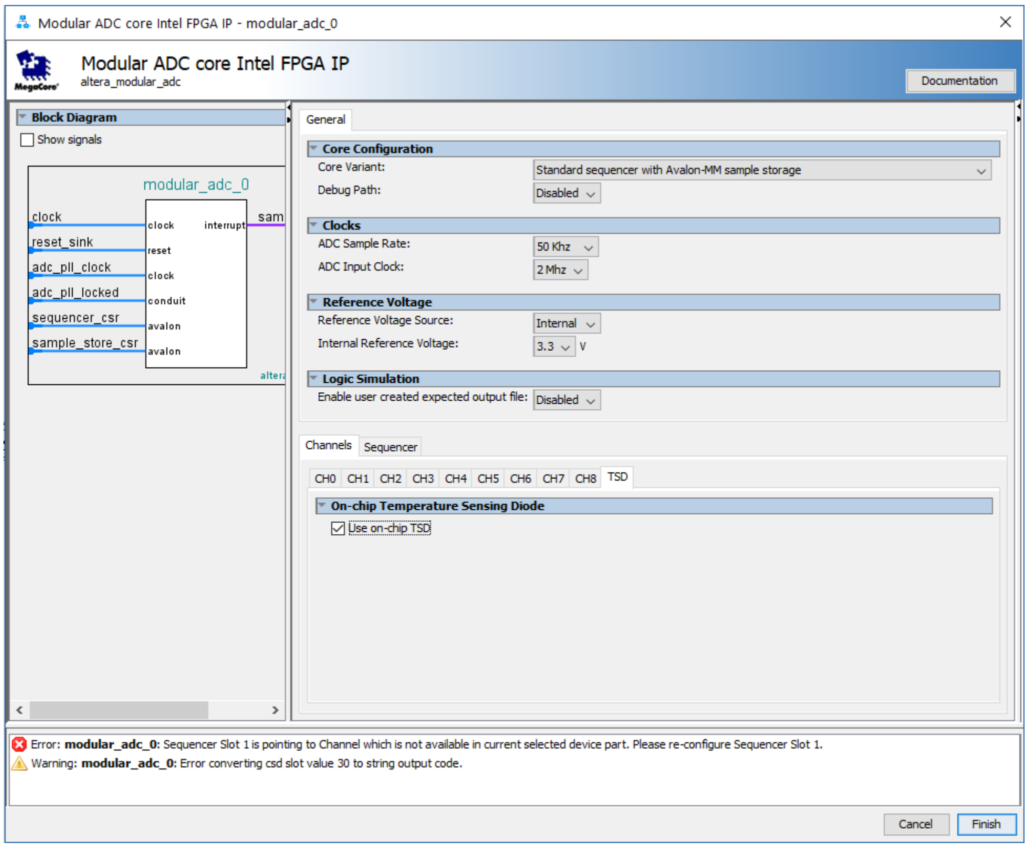
6.2.1.38 In the search bar of the IP Catalog, type “adc”, and add **Modular ADC core Intel FPGA IP**.


IP Catalog
adc
Project
New Component...
Library
Processors and Peripherals
Peripherals
Modular ADC core Intel FPGA IP
Modular Dual ADC core Intel FPGA IP
University Program
Generic IO
ADC Controller for DE-series Boards
New...
Edit...
Add...

6.2.1.39 Verify that the core variant is **Standard sequencer with Avalon-MM sample storage,** and set the followings from the drop-down menu:

- ADC Sample Rate: **50kHz**
- ADC Input Clock: **2 MHz**
- Reference Voltage Source: **Internal**
- Internal Reference Voltage: **3.3 V**

Under the Channels tab click on **TSD** and select **Use on-chip TSD**.


Modular ADC core Intel FPGA IP - modular_adc_0

Modular ADC core Intel FPGA IP
altera_modular_adc

Core Configuration
Core Variant: Standard sequencer with Avalon-MM sample storage
Debug Path: Disabled

Clocks
ADC Sample Rate: 50 Khz
ADC Input Clock: 2 Mhz

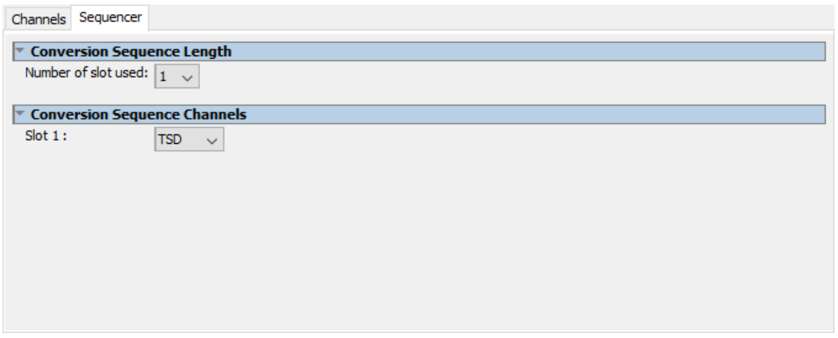
Reference Voltage
Reference Voltage Source: Internal
Internal Reference Voltage: 3.3 V

Logic Simulation
Enable user created expected output file: Disabled

On-chip Temperature Sensing Diode
☑ Use on-chip TSD

Error: **modular_adc_0:** Sequencer Slot 1 is pointing to Channel which is not available in current selected device part. Please re-configure Sequencer Slot 1.
Warning: **modular_adc_0:** Error converting csd slot value 30 to string output code.

6.2.1.40 Click on the Sequencer tab and under the Conversion Sequence Channels set **TSD** for Slot1.

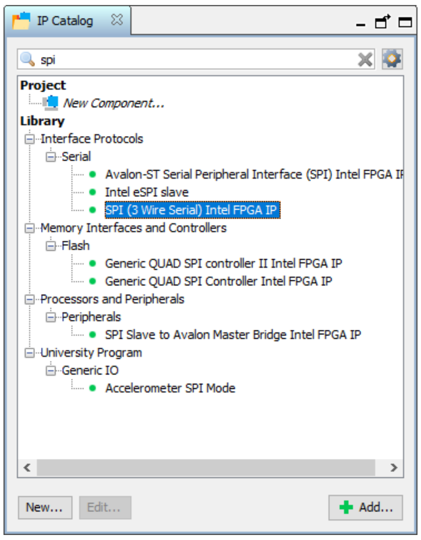

Conversion Sequence Length
Number of slot used: 1

Conversion Sequence Channels
Slot 1 : TSD

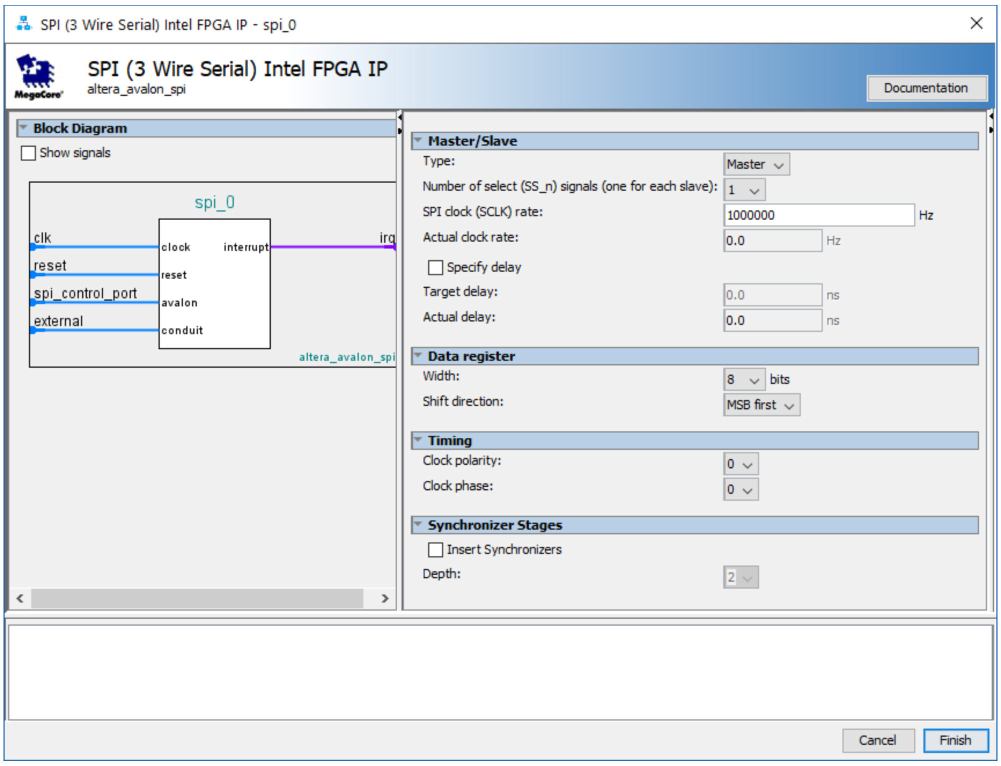
6.2.1.41 Click **Finish**.

6.2.1.42 Rename modular_adc_0 to **adc**.

6.2.1.43 In the search bar of the IP Catalog, type "spi", and add **SPI (3 Wire Serial) Intel FPGA IP** under **Interface Protocols → Serial**.


IP Catalog
spi
Project
New Component...
Library
Interface Protocols
Serial
Avalon-ST Serial Peripheral Interface (SPI) Intel FPGA IP
Intel eSPI slave
SPI (3 Wire Serial) Intel FPGA IP
Memory Interfaces and Controllers
Flash
Generic QUAD SPI controller II Intel FPGA IP
Generic QUAD SPI Controller Intel FPGA IP
Processors and Peripherals
Peripherals
SPI Slave to Avalon Master Bridge Intel FPGA IP
University Program
Generic IO
Accelerometer SPI Mode
New...
Edit...
Add...

6.2.1.44 Change the SPI Clock (SCLK) rate to **1 MHz** or type '1m' and the field will update.



Master/Slave
Type: Master
Number of select (SS_n) signals (one for each slave): 1
SPI clock (SCLK) rate: 1000000 Hz
Actual clock rate: 0.0 Hz
Specify delay
Target delay: 0.0 ns
Actual delay: 0.0 ns

Data register
Width: 8 bits
Shift direction: MSB first

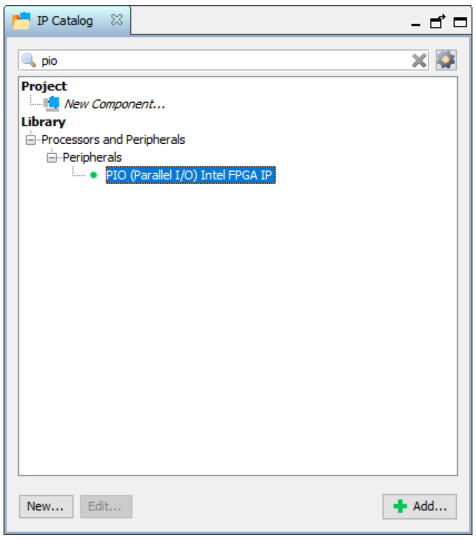
Timing
Clock polarity: 0
Clock phase: 0

Synchronizer Stages
Insert Synchronizers
Depth: 2

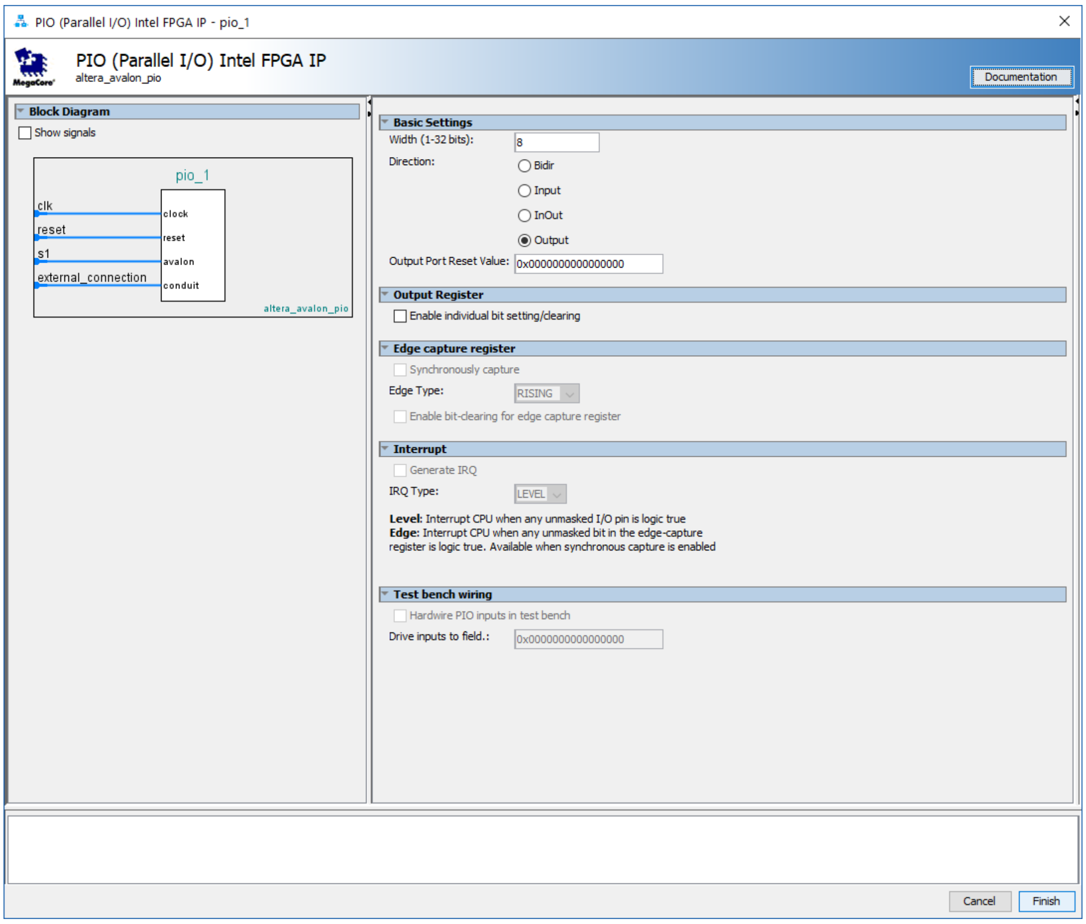
6.2.1.45 Accept the defaults for the remaining fields and click **Finish**.

6.2.1.46 Rename spi_0 to **spi**.

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



6.2.1.48 Verify that the width is **8 bits** and the direction is **output**.


PIO (Parallel I/O) Intel FPGA IP - pio_1

PIO (Parallel I/O) Intel FPGA IP
altera_avalon_pio

Basic Settings
Width (1-32 bits): 8
Direction: Bidir Input InOut [✓ Output]
Output Port Reset Value: 0x0000000000000000

Output Register
Enable individual bit setting/clearing

Edge capture register
Synchronously capture
Edge Type: RISING
Enable bit-clearing for edge capture register

Interrupt
Generate IRQ
IRQ Type: LEVEL
Level: Interrupt CPU when any unmasked I/O pin is logic true
Edge: Interrupt CPU when any unmasked bit in the edge-capture register is logic true. Available when synchronous capture is enabled

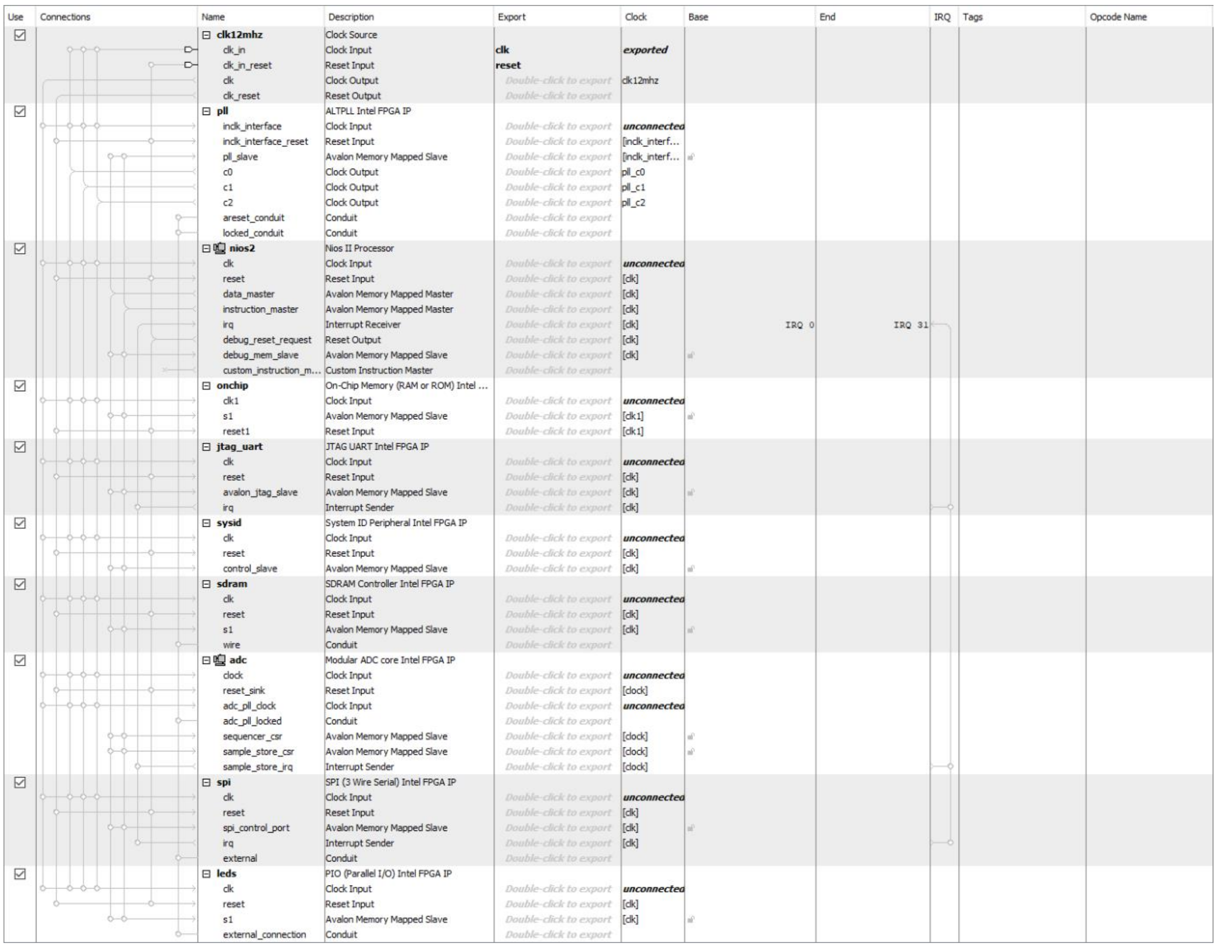
Test bench wiring
Hardwire PIO inputs in test bench
Drive inputs to field.: 0x0000000000000000

6.2.1.49 Accept the defaults for the remaining fields and click **Finish**.

6.2.1.50 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.

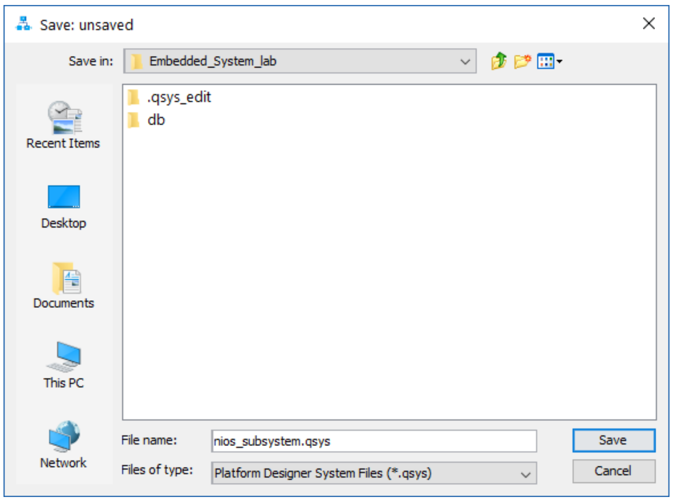
At this point, there are 10 components in the system and should look as follows:



Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags	Opcode Name
✓		**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*						
✓		**pll**	ALTPLL Intel FPGA IP							
		inclk_interface	Clock Input	*Double-click to export*	***unconnected***					
		inclk_interface_reset	Reset Input	*Double-click to export*	[inclk_interf...					
		pll_slave	Avalon Memory Mapped Slave	*Double-click to export*	[inclk_interf...					
		c0	Clock Output	*Double-click to export*	pll_c0					
		c1	Clock Output	*Double-click to export*	pll_c1					
		c2	Clock Output	*Double-click to export*	pll_c2					
		areset_conduit	Conduit	*Double-click to export*						
		locked_conduit	Conduit	*Double-click to export*						
✓		**nios2**	Nios II Processor							
		clk	Clock Input	*Double-click to export*	***unconnected***					
		reset	Reset Input	*Double-click to export*	[clk]					
		data_master	Avalon Memory Mapped Master	*Double-click to export*	[clk]					
		instruction_master	Avalon Memory Mapped Master	*Double-click to export*	[clk]					
		irq	Interrupt Receiver	*Double-click to export*	[clk]	IRQ 0	IRQ 31			
		debug_reset_request	Reset Output	*Double-click to export*	[clk]					
		debug_mem_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]					
		custom_instruction_m...	Custom Instruction Master	*Double-click to export*						
✓		**onchip**	On-Chip Memory (RAM or ROM) Intel ...							
		clk1	Clock Input	*Double-click to export*	***unconnected***					
		s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk1]					
		reset1	Reset Input	*Double-click to export*	[clk1]					
✓		**jtag_uart**	JTAG UART Intel FPGA IP							
		clk	Clock Input	*Double-click to export*	***unconnected***					
		reset	Reset Input	*Double-click to export*	[clk]					
		avalon_jtag_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]					
		irq	Interrupt Sender	*Double-click to export*	[clk]					
✓		**sysid**	System ID Peripheral Intel FPGA IP							
		clk	Clock Input	*Double-click to export*	***unconnected***					
		reset	Reset Input	*Double-click to export*	[clk]					
		control_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]					
✓		**sdram**	SDRAM Controller Intel FPGA IP							
		clk	Clock Input	*Double-click to export*	***unconnected***					
		reset	Reset Input	*Double-click to export*	[clk]					
		s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]					
		wire	Conduit	*Double-click to export*						
✓		**adc**	Modular ADC core Intel FPGA IP							
		clock	Clock Input	*Double-click to export*	***unconnected***					
		reset_sink	Reset Input	*Double-click to export*	[clock]					
		adc_pll_clock	Clock Input	*Double-click to export*	***unconnected***					
		adc_pll_locked	Conduit	*Double-click to export*						
		sequencer_csr	Avalon Memory Mapped Slave	*Double-click to export*	[clock]					
		sample_store_csr	Avalon Memory Mapped Slave	*Double-click to export*	[clock]					
		sample_store_irq	Interrupt Sender	*Double-click to export*	[clock]					
✓		**spi**	SPI (3 Wire Serial) Intel FPGA IP							
		clk	Clock Input	*Double-click to export*	***unconnected***					
		reset	Reset Input	*Double-click to export*	[clk]					
		spi_control_port	Avalon Memory Mapped Slave	*Double-click to export*	[clk]					
		irq	Interrupt Sender	*Double-click to export*	[clk]					
		external	Conduit	*Double-click to export*						
✓		**leds**	PIO (Parallel I/O) Intel FPGA IP							
		clk	Clock Input	*Double-click to export*	***unconnected***					
		reset	Reset Input	*Double-click to export*	[clk]					
		s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]					
		external_connection	Conduit	*Double-click to export*						

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

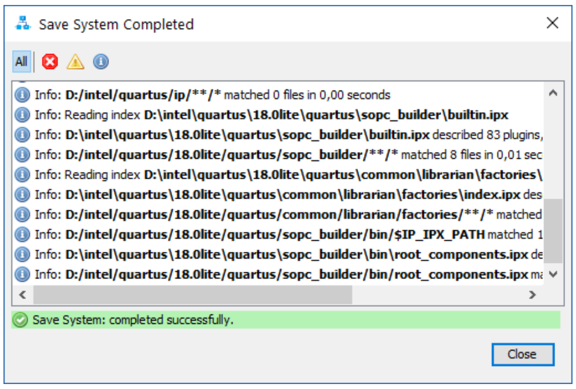
- File name: **nios_subsystem**
- Files of type: **Platform Designer System Files (*.qsys)**



Save: unsaved
Save in: Embedded_System_lab
.qsys_edit
db
Recent Items
Desktop
Documents
This PC
Network
File name: nios_subsystem.qsys
Files of type: Platform Designer System Files (*.qsys)
Save
Cancel

6.2.1.53 Click **Save**.

6.2.1.54 When the Save System Completed, then click **Close**.



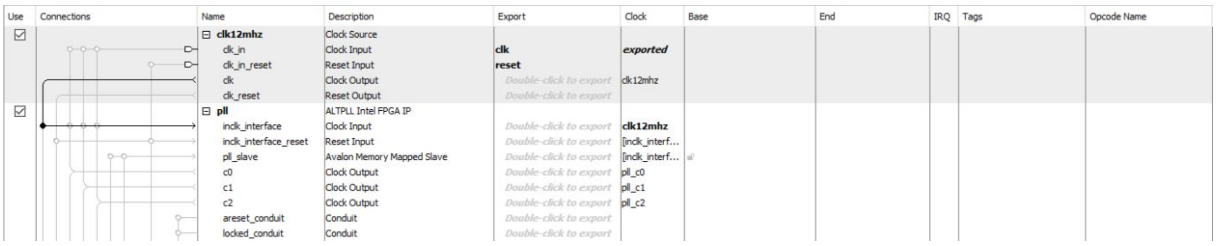
```
Info: D:/intel/quartus/ip/**/* matched 0 files in 0,00 seconds
Info: Reading index D:\intel\quartus\18.0lite\quartus\sopc_builder\builtin.ipx
Info: D:\intel\quartus\18.0lite\quartus\sopc_builder\builtin.ipx described 83 plugins,
Info: D:/intel/quartus/18.0lite/quartus/sopc_builder/**/* matched 8 files in 0,01 sec
Info: Reading index D:\intel\quartus\18.0lite\quartus\common\librarian\factories\
Info: D:\intel\quartus\18.0lite\quartus\common\librarian\factories\index.ipx des
Info: D:/intel/quartus/18.0lite/quartus/common/librarian/factories/**/* matched
Info: D:/intel/quartus/18.0lite/quartus/sopc_builder/bin/$IP_IPX_PATH matched 1
Info: D:\intel\quartus\18.0lite\quartus\sopc_builder\bin\root_components.ipx de
Info: D:/intel/quartus/18.0lite/quartus/sopc_builder/bin/root_components.ipx ma
Save System: completed successfully.
```

6.2.2 Connections

Overview: 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.

6.2.2.1 Create the following connection: **clk12mhz | clk ↔ pll | inclk_interface**

The connection should look as follow:

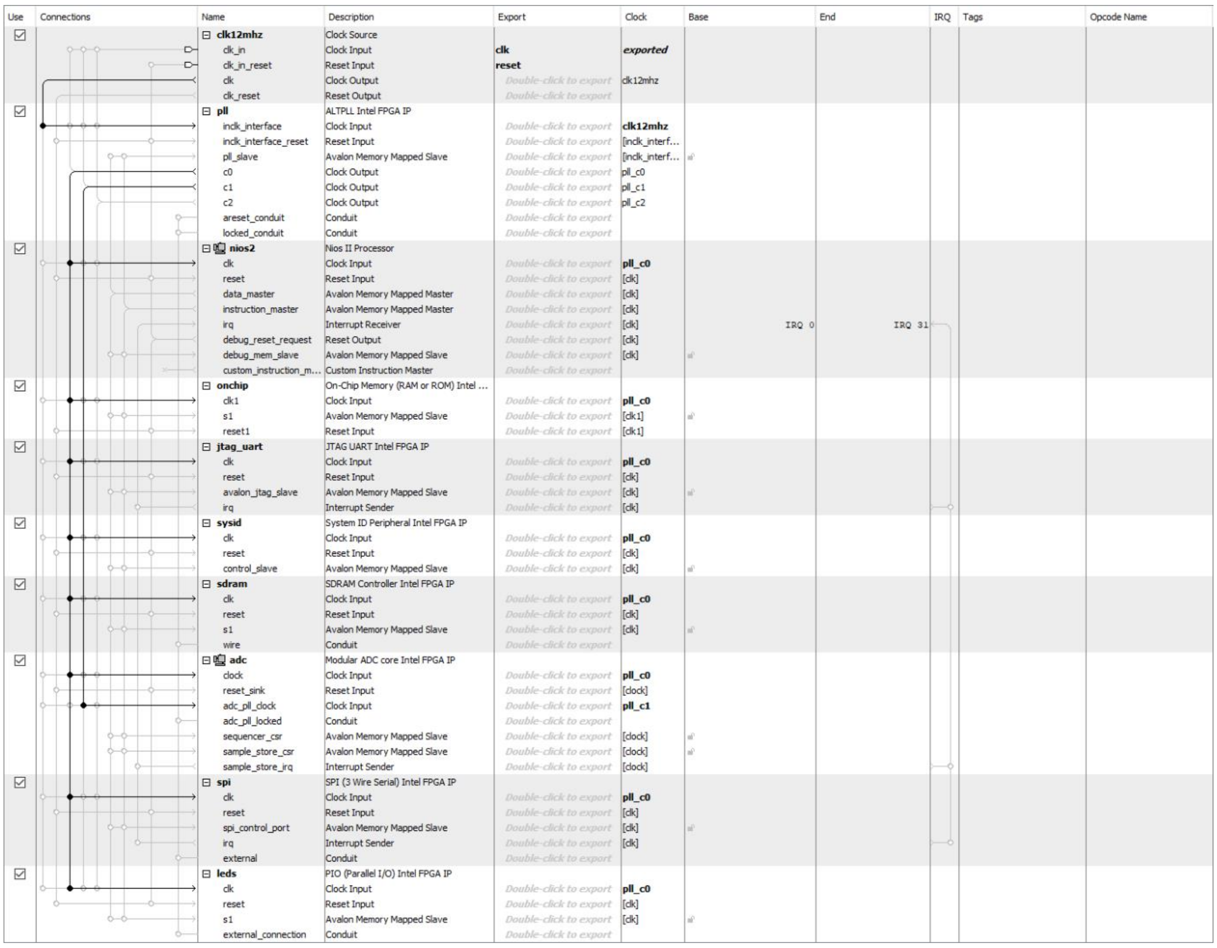


Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags	Opcode Name
☑		⊟ **clk12mhz**	Clock Source							
		clk_in	Clock Input	**clk**	***exported***					
		clk_in_reset	Reset Input	**reset**						
		clk	Clock Output	*Double-click to export*	clk12mhz					
		clk_reset	Reset Output	*Double-click to export*						
☑		⊟ **pll**	ALTPLL Intel FPGA IP							
		inclk_interface	Clock Input	*Double-click to export*	**clk12mhz**					
		inclk_interface_reset	Reset Input	*Double-click to export*	[inclk_interf...					
		pll_slave	Avalon Memory Mapped Slave	*Double-click to export*	[inclk_interf...					
		c0	Clock Output	*Double-click to export*	pll_c0					
		c1	Clock Output	*Double-click to export*	pll_c1					
		c2	Clock Output	*Double-click to export*	pll_c2					
		areset_conduit	Conduit	*Double-click to export*						
		locked_conduit	Conduit	*Double-click to export*						

6.2.2.2 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	↔	sdram | clk
pll | c0	↔	adc | clk
pll | c0	↔	spi | clk
pll | c0	↔	leds | clk
pll | c1	↔	adc | adc_pll_clock

The connections should look as the follows:

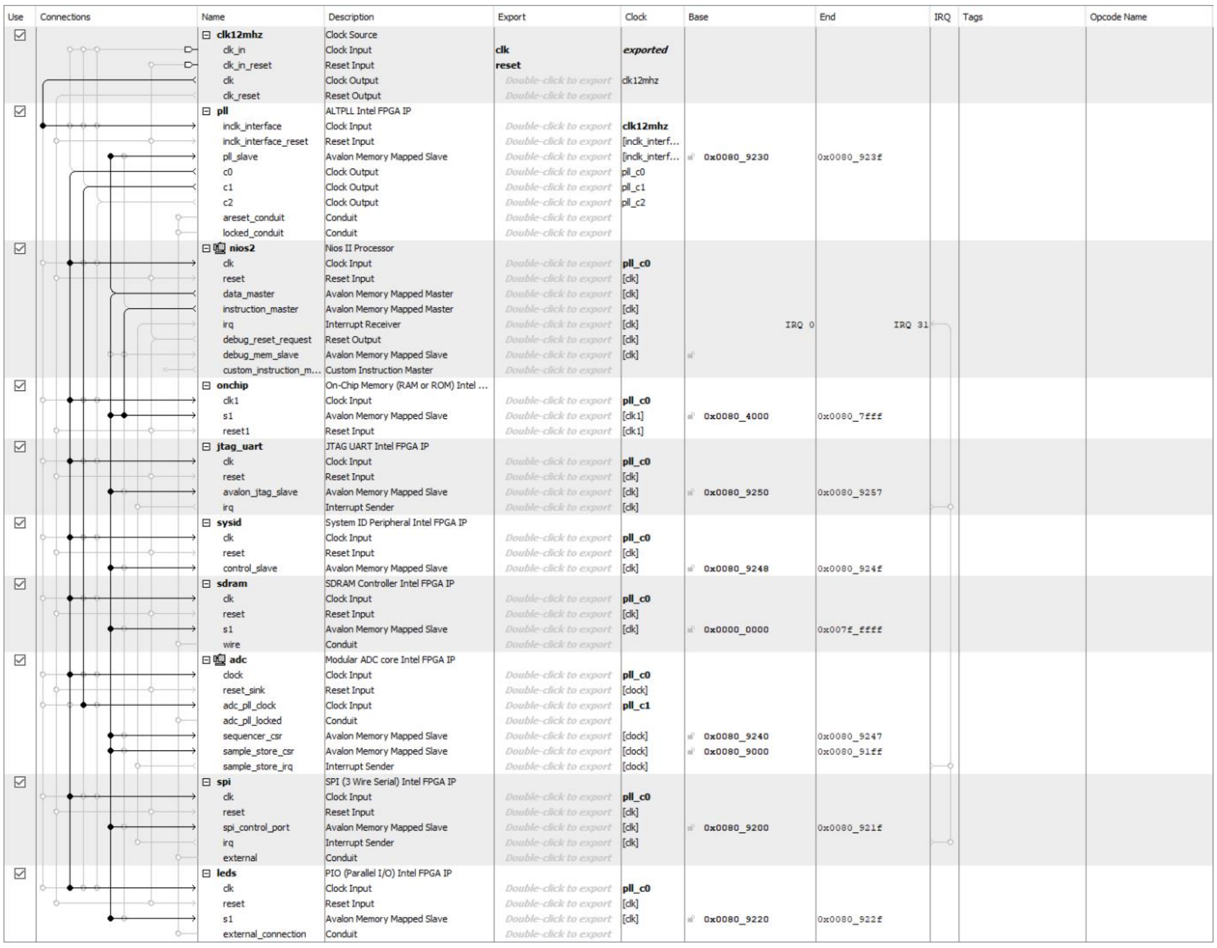


Use	Name	Description	Export	Clock	Base	End
✓	**clk12mhz**	Clock Source				
	clk_in	Clock Input	**clk**	***exported***		
	clk_in_reset	Reset Input	**reset**			
	clk	Clock Output	Double-click to export	clk12mhz		
	clk_reset	Reset Output	Double-click to export			
✓	**pll**	ALTPLL Intel FPGA IP				
	inclk_interface	Clock Input	Double-click to export	**clk12mhz**		
	inclk_interface_reset	Reset Input	Double-click to export	[inclk_interf...		
	pll_slave	Avalon Memory Mapped Slave	Double-click to export	[inclk_interf...		
	c0	Clock Output	Double-click to export	pll_c0		
	c1	Clock Output	Double-click to export	pll_c1		
	c2	Clock Output	Double-click to export	pll_c2		
	areset_conduit	Conduit	Double-click to export			
	locked_conduit	Conduit	Double-click to export			
✓	**nios2**	Nios II Processor				
	clk	Clock Input	Double-click to export	**pll_c0**		
	reset	Reset Input	Double-click to export	[clk]		
	data_master	Avalon Memory Mapped Master	Double-click to export	[clk]		
	instruction_master	Avalon Memory Mapped Master	Double-click to export	[clk]		
	irq	Interrupt Receiver	Double-click to export	[clk]	IRQ 0	IRQ 31
	debug_reset_request	Reset Output	Double-click to export	[clk]		
	debug_mem_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]		
	custom_instruction_m...	Custom Instruction Master	Double-click to export			
✓	**onchip**	On-Chip Memory (RAM or ROM) Intel ...				
	clk1	Clock Input	Double-click to export	**pll_c0**		
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk1]		
	reset1	Reset Input	Double-click to export	[clk1]		
✓	**jtag_uart**	JTAG UART Intel FPGA IP				
	clk	Clock Input	Double-click to export	**pll_c0**		
	reset	Reset Input	Double-click to export	[clk]		
	avalon_jtag_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]		
	irq	Interrupt Sender	Double-click to export	[clk]		
✓	**sysid**	System ID Peripheral Intel FPGA IP				
	clk	Clock Input	Double-click to export	**pll_c0**		
	reset	Reset Input	Double-click to export	[clk]		
	control_slave	Avalon Memory Mapped Slave	Double-click to export	[clk]		
✓	**sdram**	SDRAM Controller Intel FPGA IP				
	clk	Clock Input	Double-click to export	**pll_c0**		
	reset	Reset Input	Double-click to export	[clk]		
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]		
	wire	Conduit	Double-click to export			
✓	**adc**	Modular ADC core Intel FPGA IP				
	clock	Clock Input	Double-click to export	**pll_c0**		
	reset_sink	Reset Input	Double-click to export	[clock]		
	adc_pll_clock	Clock Input	Double-click to export	**pll_c1**		
	adc_pll_locked	Conduit	Double-click to export			
	sequencer_csr	Avalon Memory Mapped Slave	Double-click to export	[clock]		
	sample_store_csr	Avalon Memory Mapped Slave	Double-click to export	[clock]		
	sample_store_irq	Interrupt Sender	Double-click to export	[clock]		
✓	**spi**	SPI (3 Wire Serial) Intel FPGA IP				
	clk	Clock Input	Double-click to export	**pll_c0**		
	reset	Reset Input	Double-click to export	[clk]		
	spi_control_port	Avalon Memory Mapped Slave	Double-click to export	[clk]		
	irq	Interrupt Sender	Double-click to export	[clk]		
	external	Conduit	Double-click to export			
✓	**leds**	PIO (Parallel I/O) Intel FPGA IP				
	clk	Clock Input	Double-click to export	**pll_c0**		
	reset	Reset Input	Double-click to export	[clk]		
	s1	Avalon Memory Mapped Slave	Double-click to export	[clk]		
	external_connection	Conduit	Double-click to export			

6.2.2.3 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	↔	sdram | s1
nios | data_master	↔	adc | sequencer_csr
nios | data_master	↔	adc |sample_store_csr
nios | data_master	↔	spi | spi_control_port
nios | data_master	↔	leds | s1

The connections should look as the follows:



Use	Connections	Name	Description	Export	Clock	Base	End	IRQ	Tags	Opcode Name
✓		⊟ **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*						
✓		⊟ **pll**	ALTPLL Intel FPGA IP							
		inclk_interface	Clock Input	*Double-click to export*	**clk12mhz**					
		inclk_interface_reset	Reset Input	*Double-click to export*	[inclk_interf...					
		pll_slave	Avalon Memory Mapped Slave	*Double-click to export*	[inclk_interf...	0x0080_9230	0x0080_923f			
		c0	Clock Output	*Double-click to export*	pll_c0					
		c1	Clock Output	*Double-click to export*	pll_c1					
		c2	Clock Output	*Double-click to export*	pll_c2					
		areset_conduit	Conduit	*Double-click to export*						
		locked_conduit	Conduit	*Double-click to export*						
✓		⊟ **nios2**	Nios II Processor							
		clk	Clock Input	*Double-click to export*	**pll_c0**					
		reset	Reset Input	*Double-click to export*	[clk]					
		data_master	Avalon Memory Mapped Master	*Double-click to export*	[clk]					
		instruction_master	Avalon Memory Mapped Master	*Double-click to export*	[clk]					
		irq	Interrupt Receiver	*Double-click to export*	[clk]	IRQ 0	IRQ 31			
		debug_reset_request	Reset Output	*Double-click to export*	[clk]					
		debug_mem_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]					
		custom_instruction_m...	Custom Instruction Master	*Double-click to export*						
✓		⊟ **onchip**	On-Chip Memory (RAM or ROM) Intel ...							
		clk1	Clock Input	*Double-click to export*	**pll_c0**					
		s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk1]	0x0080_4000	0x0080_7fff			
		reset1	Reset Input	*Double-click to export*	[clk1]					
✓		⊟ **jtag_uart**	JTAG UART Intel FPGA IP							
		clk	Clock Input	*Double-click to export*	**pll_c0**					
		reset	Reset Input	*Double-click to export*	[clk]					
		avalon_jtag_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0080_9250	0x0080_9257			
		irq	Interrupt Sender	*Double-click to export*	[clk]					
✓		⊟ **sysid**	System ID Peripheral Intel FPGA IP							
		clk	Clock Input	*Double-click to export*	**pll_c0**					
		reset	Reset Input	*Double-click to export*	[clk]					
		control_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0080_9248	0x0080_924f			
✓		⊟ **sdram**	SDRAM Controller Intel FPGA IP							
		clk	Clock Input	*Double-click to export*	**pll_c0**					
		reset	Reset Input	*Double-click to export*	[clk]					
		s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0000_0000	0x007f_ffff			
		wire	Conduit	*Double-click to export*						
✓		⊟ **adc**	Modular ADC core Intel FPGA IP							
		clock	Clock Input	*Double-click to export*	**pll_c0**					
		reset_sink	Reset Input	*Double-click to export*	[clock]					
		adc_pll_clock	Clock Input	*Double-click to export*	**pll_c1**					
		adc_pll_locked	Conduit	*Double-click to export*						
		sequencer_csr	Avalon Memory Mapped Slave	*Double-click to export*	[clock]	0x0080_9240	0x0080_9247			
		sample_store_csr	Avalon Memory Mapped Slave	*Double-click to export*	[clock]	0x0080_9000	0x0080_91ff			
		sample_store_irq	Interrupt Sender	*Double-click to export*	[clock]					
✓		⊟ **spi**	SPI (3 Wire Serial) Intel FPGA IP							
		clk	Clock Input	*Double-click to export*	**pll_c0**					
		reset	Reset Input	*Double-click to export*	[clk]					
		spi_control_port	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0080_9200	0x0080_921f			
		irq	Interrupt Sender	*Double-click to export*	[clk]					
		external	Conduit	*Double-click to export*						
✓		⊟ **leds**	PIO (Parallel I/O) Intel FPGA IP							
		clk	Clock Input	*Double-click to export*	**pll_c0**					
		reset	Reset Input	*Double-click to export*	[clk]					
		s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0080_9220	0x0080_922f			
		external_connection	Conduit	*Double-click to export*						

6.2.2.4 Create the following connections for interrupt request:

Component A		Component B
nios | irq	↔	jtag_uart| irq
nios | irq	↔	adc | sample_store_irq
nios | irq	↔	spi | irq

6.2.2.5 Make the following connection:

pll | locked_conduit ↔ adc | adc_pll_locked

6.2.2.6 Double click on the Export field next to the c2 of pll and name it **dram_clk**.

6.2.2.7 Double click on the Export field next to the areset_conduit of pll and name it **pll_areset**.

6.2.2.8 Double click on the Export field next to the wire of sdram and name it **sdram**.

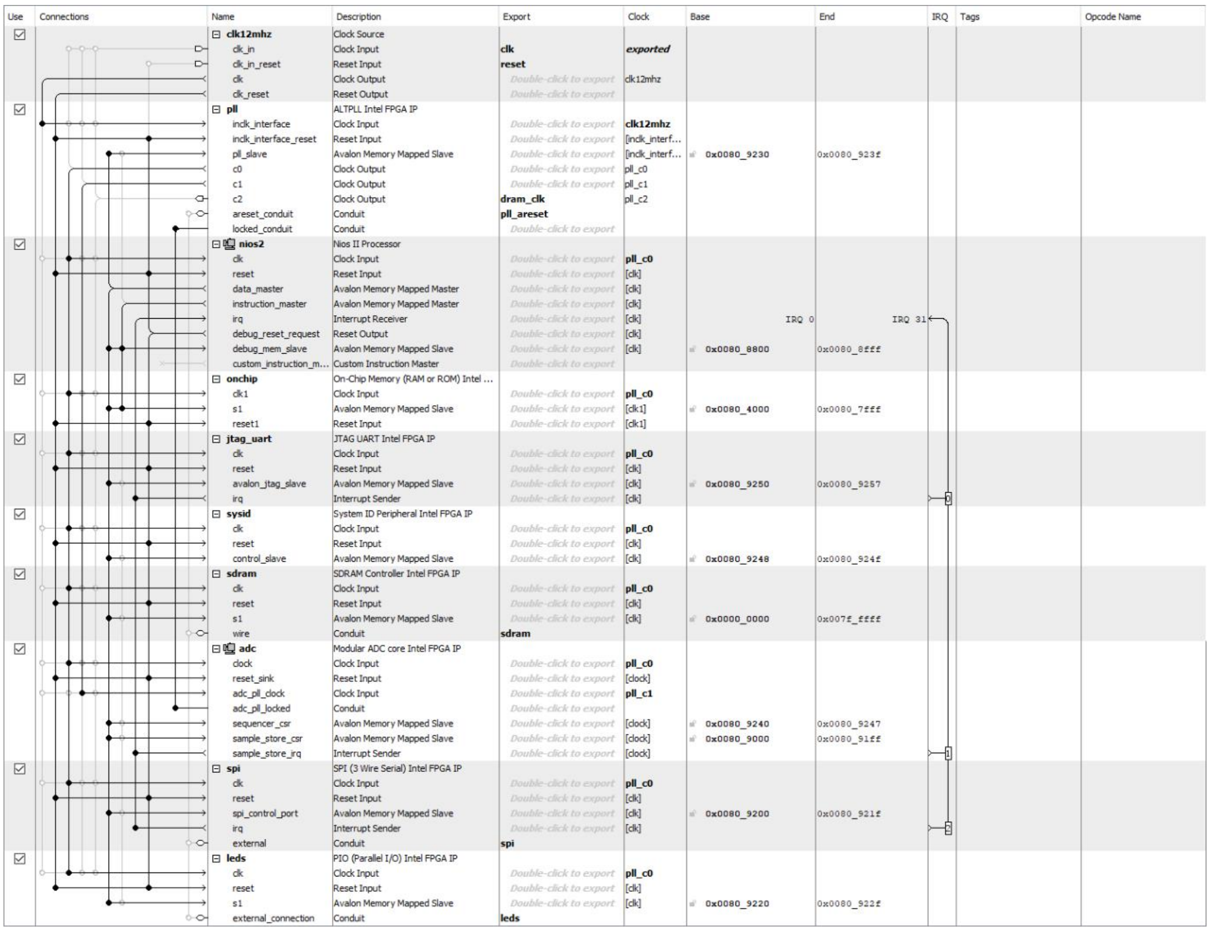
6.2.2.9 Double click on the Export field next to the external of spi and name it **spi**.

6.2.2.10 Double click on the Export field next to the external_connection of leds and name it **leds**.

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

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

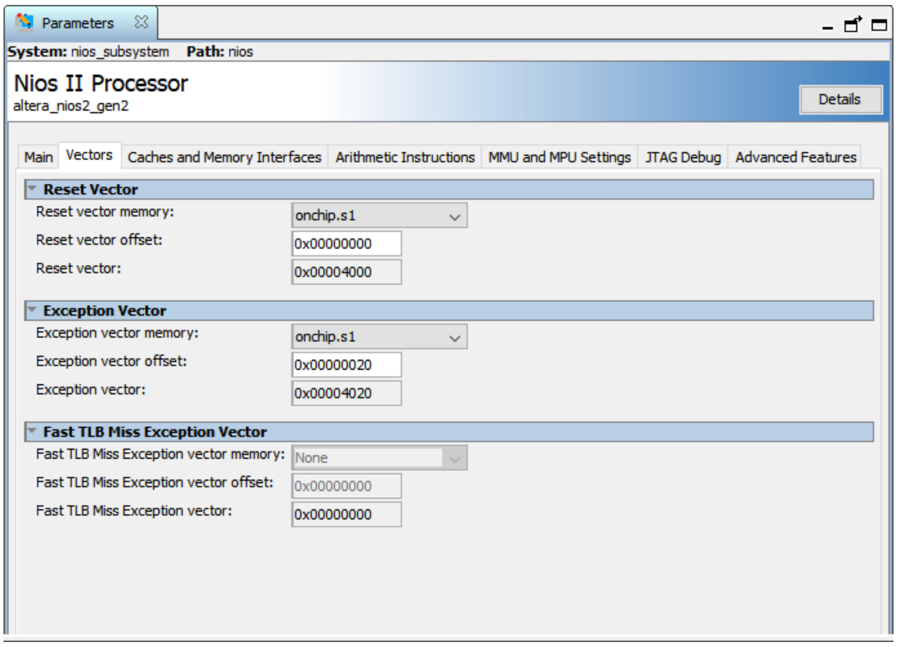
6.2.2.13 Verify that your system is the same as below:



Use	Name	Description	Export	Clock	Base	End	IRQ
✓	**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*				
✓	**pll**	ALTPLL Intel FPGA IP					
	inclk_interface	Clock Input	*Double-click to export*	**clk12mhz**			
	inclk_interface_reset	Reset Input	*Double-click to export*	[inclk_interf...			
	pll_slave	Avalon Memory Mapped Slave	*Double-click to export*	[inclk_interf...	0x0080_9230	0x0080_923f	
	c0	Clock Output	*Double-click to export*	pll_c0			
	c1	Clock Output	*Double-click to export*	pll_c1			
	c2	Clock Output	**dram_clk**	pll_c2			
	areset_conduit	Conduit	**pll_areset**				
	locked_conduit	Conduit	*Double-click to export*				
✓	**nios2**	Nios II Processor					
	clk	Clock Input	*Double-click to export*	**pll_c0**			
	reset	Reset Input	*Double-click to export*	[clk]			
	data_master	Avalon Memory Mapped Master	*Double-click to export*	[clk]			
	instruction_master	Avalon Memory Mapped Master	*Double-click to export*	[clk]			
	irq	Interrupt Receiver	*Double-click to export*	[clk]	IRQ 0	IRQ 31	
	debug_reset_request	Reset Output	*Double-click to export*	[clk]			
	debug_mem_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0080_8800	0x0080_8fff	
	custom_instruction_m...	Custom Instruction Master	*Double-click to export*				
✓	**onchip**	On-Chip Memory (RAM or ROM) Intel ...					
	clk1	Clock Input	*Double-click to export*	**pll_c0**			
	s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk1]	0x0080_4000	0x0080_7fff	
	reset1	Reset Input	*Double-click to export*	[clk1]			
✓	**jtag_uart**	JTAG UART Intel FPGA IP					
	clk	Clock Input	*Double-click to export*	**pll_c0**			
	reset	Reset Input	*Double-click to export*	[clk]			
	avalon_jtag_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0080_9250	0x0080_9257	
	irq	Interrupt Sender	*Double-click to export*	[clk]			0
✓	**sysid**	System ID Peripheral Intel FPGA IP					
	clk	Clock Input	*Double-click to export*	**pll_c0**			
	reset	Reset Input	*Double-click to export*	[clk]			
	control_slave	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0080_9248	0x0080_924f	
✓	**sdram**	SDRAM Controller Intel FPGA IP					
	clk	Clock Input	*Double-click to export*	**pll_c0**			
	reset	Reset Input	*Double-click to export*	[clk]			
	s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0000_0000	0x007f_ffff	
	wire	Conduit	**sdram**				
✓	**adc**	Modular ADC core Intel FPGA IP					
	clock	Clock Input	*Double-click to export*	**pll_c0**			
	reset_sink	Reset Input	*Double-click to export*	[clock]			
	adc_pll_clock	Clock Input	*Double-click to export*	**pll_c1**			
	adc_pll_locked	Conduit	*Double-click to export*				
	sequencer_csr	Avalon Memory Mapped Slave	*Double-click to export*	[clock]	0x0080_9240	0x0080_9247	
	sample_store_csr	Avalon Memory Mapped Slave	*Double-click to export*	[clock]	0x0080_9000	0x0080_91ff	
	sample_store_irq	Interrupt Sender	*Double-click to export*	[clock]			1
✓	**spi**	SPI (3 Wire Serial) Intel FPGA IP					
	clk	Clock Input	*Double-click to export*	**pll_c0**			
	reset	Reset Input	*Double-click to export*	[clk]			
	spi_control_port	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0080_9200	0x0080_921f	
	irq	Interrupt Sender	*Double-click to export*	[clk]			2
	external	Conduit	**spi**				
✓	**leds**	PIO (Parallel I/O) Intel FPGA IP					
	clk	Clock Input	*Double-click to export*	**pll_c0**			
	reset	Reset Input	*Double-click to export*	[clk]			
	s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0080_9220	0x0080_922f	
	external_connection	Conduit	**leds**				

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

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


Parameters

System: nios_subsystem Path: nios

Nios II Processor
altera_nios2_gen2

Reset Vector
Reset vector memory: onchip.s1
Reset vector offset: 0x00000000
Reset vector: 0x00004000

Exception Vector
Exception vector memory: onchip.s1
Exception vector offset: 0x00000020
Exception vector: 0x00004020

Fast TLB Miss Exception Vector
Fast TLB Miss Exception vector memory: None
Fast TLB Miss Exception vector offset: 0x00000000
Fast TLB Miss Exception vector: 0x00000000

6.2.2.16 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.

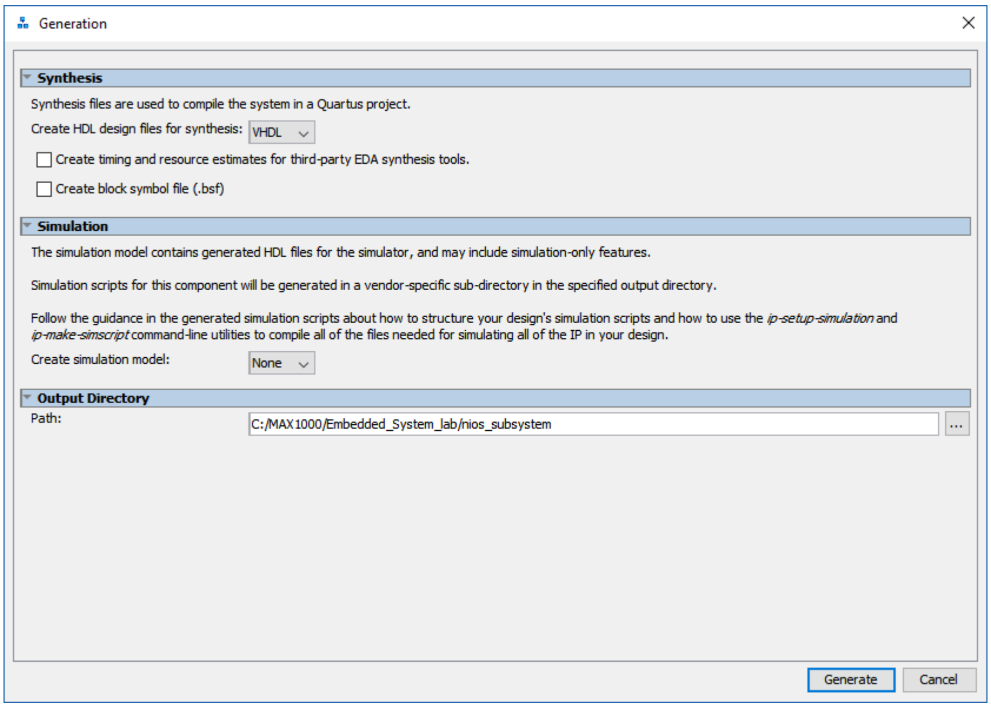
6.2.2.17 **Save** your design.

6.2.3 Generate the Platform Designer System

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

6.2.3.2 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**


Generation

Synthesis

Synthesis files are used to compile the system in a Quartus project.

Create HDL design files for synthesis: VHDL

☐ Create timing and resource estimates for third-party EDA synthesis tools.

☐ Create block symbol file (.bsf)

Simulation

The simulation model contains generated HDL files for the simulator, and may include simulation-only features.

Simulation scripts for this component will be generated in a vendor-specific sub-directory in the specified output directory.

Follow the guidance in the generated simulation scripts about how to structure your design's simulation scripts and how to use the *ip-setup-simulation* and *ip-make-simscript* command-line utilities to compile all of the files needed for simulating all of the IP in your design.

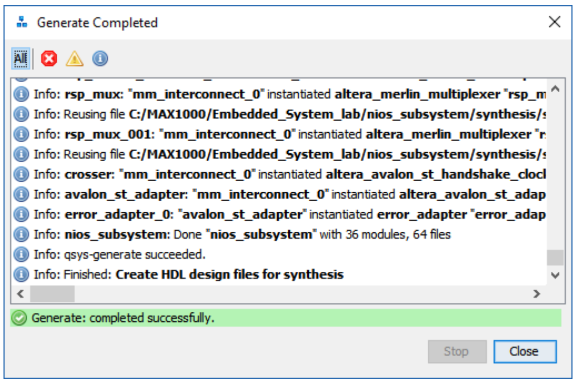
Create simulation model: None

Output Directory

Path: C:/MAX1000/Embedded_System_lab/nios_subsystem

6.2.3.3 Click **Generate**.

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

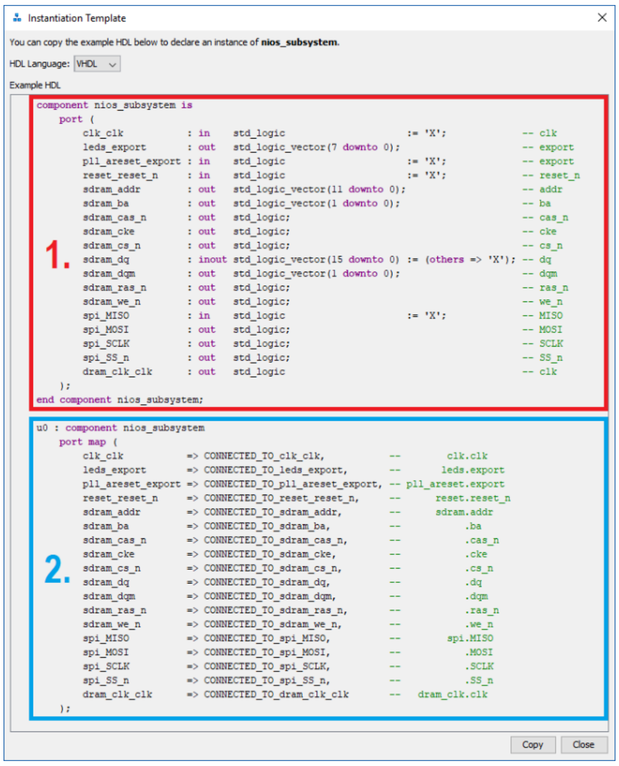

Generate Completed

```
Info: rsp_mux: "mm_interconnect_0" instantiated altera_merlin_multiplexer "rsp_m
Info: Reusing file C:/MAX1000/Embedded_System_lab/nios_subsystem/synthesis/s
Info: rsp_mux_001: "mm_interconnect_0" instantiated altera_merlin_multiplexer "r
Info: Reusing file C:/MAX1000/Embedded_System_lab/nios_subsystem/synthesis/s
Info: crosser: "mm_interconnect_0" instantiated altera_avalon_st_handshake_clocl
Info: avalon_st_adapter: "mm_interconnect_0" instantiated altera_avalon_st_adap
Info: error_adapter_0: "avalon_st_adapter" instantiated error_adapter "error_adap
Info: nios_subsystem: Done "nios_subsystem" with 36 modules, 64 files
Info: qsys-generate succeeded.
Info: Finished: Create HDL design files for synthesis
```

Generate: completed successfully.

6.2.3.5 In the Platform Designer window, select **Generate → Show Instantiation Template…**

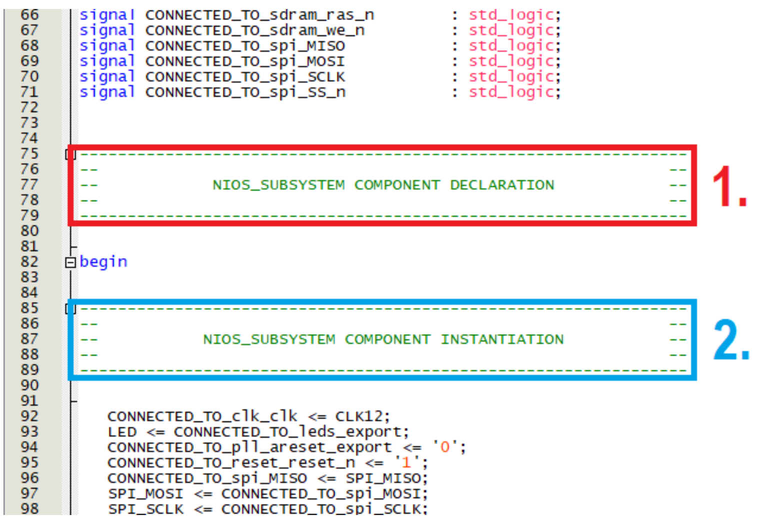
6.2.3.6 Select **VHDL** as the HDL Language. The following template will be shown which can be easily copied into your project, saving you valuable time.


Instantiation Template
You can copy the example HDL below to declare an instance of **nios_subsystem**.
HDL Language: VHDL
Example HDL
1.
2.
Copy
Close

```
component nios_subsystem is
    port (
        clk_clk            : in    std_logic                     := 'X';             -- clk
        leds_export        : out   std_logic_vector(7 downto 0);                     -- export
        pll_areset_export  : in    std_logic                     := 'X';             -- export
        reset_reset_n      : in    std_logic                     := 'X';             -- reset_n
        sdram_addr         : out   std_logic_vector(11 downto 0);                    -- addr
        sdram_ba           : out   std_logic_vector(1 downto 0);                     -- ba
        sdram_cas_n        : out   std_logic;                                        -- cas_n
        sdram_cke          : out   std_logic;                                        -- cke
        sdram_cs_n         : out   std_logic;                                        -- cs_n
        sdram_dq           : inout std_logic_vector(15 downto 0) := (others => 'X'); -- dq
        sdram_dqm          : out   std_logic_vector(1 downto 0);                     -- dqm
        sdram_ras_n        : out   std_logic;                                        -- ras_n
        sdram_we_n         : out   std_logic;                                        -- we_n
        spi_MISO           : in    std_logic                     := 'X';             -- MISO
        spi_MOSI           : out   std_logic;                                        -- MOSI
        spi_SCLK           : out   std_logic;                                        -- SCLK
        spi_SS_n           : out   std_logic;                                        -- SS_n
        dram_clk_clk       : out   std_logic                                         -- clk
    );
end component nios_subsystem;

u0 : component nios_subsystem
    port map (
        clk_clk           => CONNECTED_TO_clk_clk,           --           clk.clk
        leds_export       => CONNECTED_TO_leds_export,       --          leds.export
        pll_areset_export => CONNECTED_TO_pll_areset_export, -- pll_areset.export
        reset_reset_n     => CONNECTED_TO_reset_reset_n,     --         reset.reset_n
        sdram_addr        => CONNECTED_TO_sdram_addr,        --         sdram.addr
        sdram_ba          => CONNECTED_TO_sdram_ba,          --              .ba
        sdram_cas_n       => CONNECTED_TO_sdram_cas_n,       --              .cas_n
        sdram_cke         => CONNECTED_TO_sdram_cke,         --              .cke
        sdram_cs_n        => CONNECTED_TO_sdram_cs_n,        --              .cs_n
        sdram_dq          => CONNECTED_TO_sdram_dq,          --              .dq
        sdram_dqm         => CONNECTED_TO_sdram_dqm,         --              .dqm
        sdram_ras_n       => CONNECTED_TO_sdram_ras_n,       --              .ras_n
        sdram_we_n        => CONNECTED_TO_sdram_we_n,        --              .we_n
        spi_MISO          => CONNECTED_TO_spi_MISO,          --           spi.MISO
        spi_MOSI          => CONNECTED_TO_spi_MOSI,          --              .MOSI
        spi_SCLK          => CONNECTED_TO_spi_SCLK,          --              .SCLK
        spi_SS_n          => CONNECTED_TO_spi_SS_n,          --              .SS_n
        dram_clk_clk      => CONNECTED_TO_dram_clk_clk       --      dram_clk.clk
    );
```

6.2.3.7 There are two parts that would need be copied to the top-level entity embedded_system_lab in Quartus Prime.

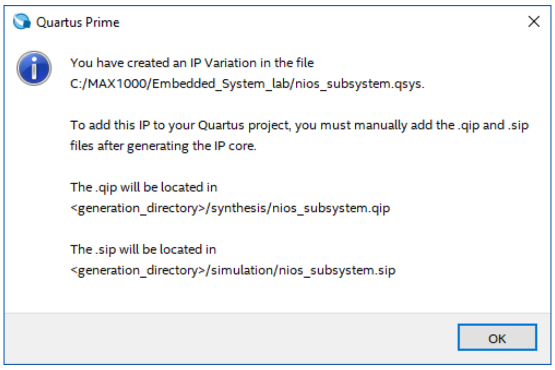
- nios_subsystem component declaration (1. Highlighted in red)
  Copy this section and paste in the architecture section of embedded_system_lab.vhd. There should be a commented area indicating where exactly.
- nios_subsystem component instantiation (2. Highlighted in blue)
  Copy this section and paste in the architecture section of embedded_system_lab.vhd after the word 'begin'. There should be a commented area indicating where exactly.



```
signal CONNECTED_TO_sdram_ras_n        : std_logic;
signal CONNECTED_TO_sdram_we_n         : std_logic;
signal CONNECTED_TO_spi_MISO           : std_logic;
signal CONNECTED_TO_spi_MOSI           : std_logic;
signal CONNECTED_TO_spi_SCLK           : std_logic;
signal CONNECTED_TO_spi_SS_n           : std_logic;



------------------------------------------------------------------
--                                                              --
--            NIOS_SUBSYSTEM COMPONENT DECLARATION              --
--                                                              --
------------------------------------------------------------------


begin


------------------------------------------------------------------
--                                                              --
--           NIOS_SUBSYSTEM COMPONENT INSTANTIATION              --
--                                                              --
------------------------------------------------------------------


    CONNECTED_TO_clk_clk <= CLK12;
    LED <= CONNECTED_TO_leds_export;
    CONNECTED_TO_pll_areset_export <= '0';
    CONNECTED_TO_reset_reset_n <= '1';
    CONNECTED_TO_spi_MISO <= SPI_MISO;
    SPI_MOSI <= CONNECTED_TO_spi_MOSI;
    SPI_SCLK <= CONNECTED_TO_spi_SCLK;
```

6.2.3.8 After copying close Instantiation Template window and click **Finish** button on the bottom right of the Platform Designer window.

6.2.3.9 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.


Quartus Prime

You have created an IP Variation in the file
C:/MAX1000/Embedded_System_lab/nios_subsystem.qsys.

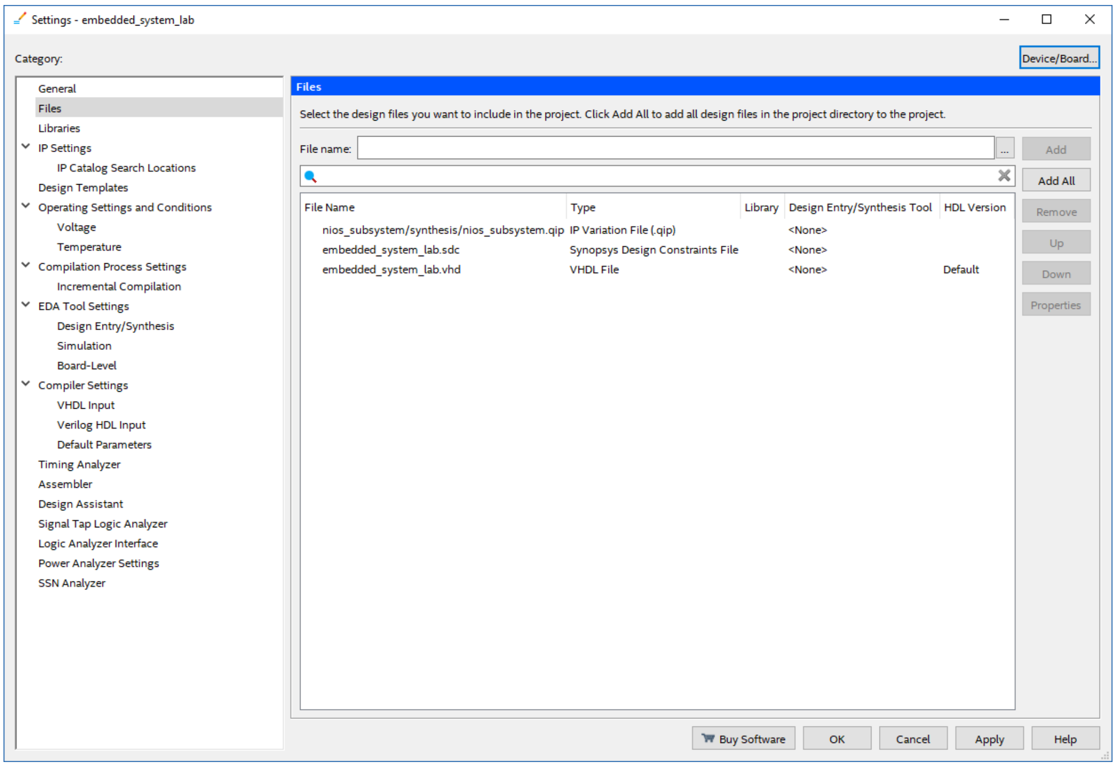
To add this IP to your Quartus project, you must manually add the .qip and .sip files after generating the IP core.

The .qip will be located in
<generation_directory>/synthesis/nios_subsystem.qip

The .sip will be located in
<generation_directory>/simulation/nios_subsystem.sip

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

6.2.3.11 Click on the ... button and browse through the synthesis directories: **<project_directory>/nios_subsystem/synthesis/** and open **nios_subsystem.qip**.


Settings - embedded_system_lab

Category: General, Files, Libraries, IP Settings, IP Catalog Search Locations, Design Templates, Operating Settings and Conditions, Voltage, Temperature, Compilation Process Settings, Incremental Compilation, EDA Tool Settings, Design Entry/Synthesis, Simulation, Board-Level, Compiler Settings, VHDL Input, Verilog HDL Input, Default Parameters, Timing Analyzer, Assembler, Design Assistant, Signal Tap Logic Analyzer, Logic Analyzer Interface, Power Analyzer Settings, SSN Analyzer

Files

Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.

File name:

File Name	Type	Library	Design Entry/Synthesis Tool	HDL Version
nios_subsystem/synthesis/nios_subsystem.qip	IP Variation File (.qip)		<None>	
embedded_system_lab.sdc	Synopsys Design Constraints File		<None>	
embedded_system_lab.vhd	VHDL File		<None>	Default

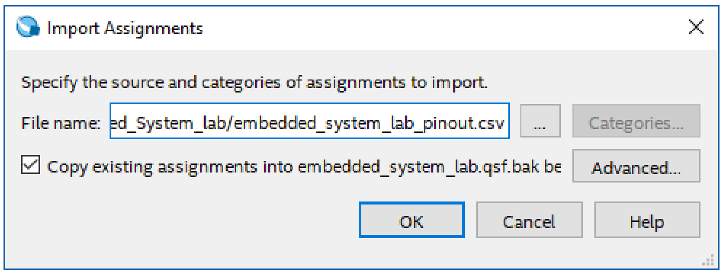
6.2.3.12 Click **Apply** and **OK**.

6.3 Compile design

6.3.1 Import pin assignments

6.3.1.1 Select **Assignments → Import Assignments…**

6.3.1.2 Add source file by clicking on the ... button and browse into the lab file folder where you will locate the provided design files and add **embedded_system_lab_pinout.csv**.



Import Assignments

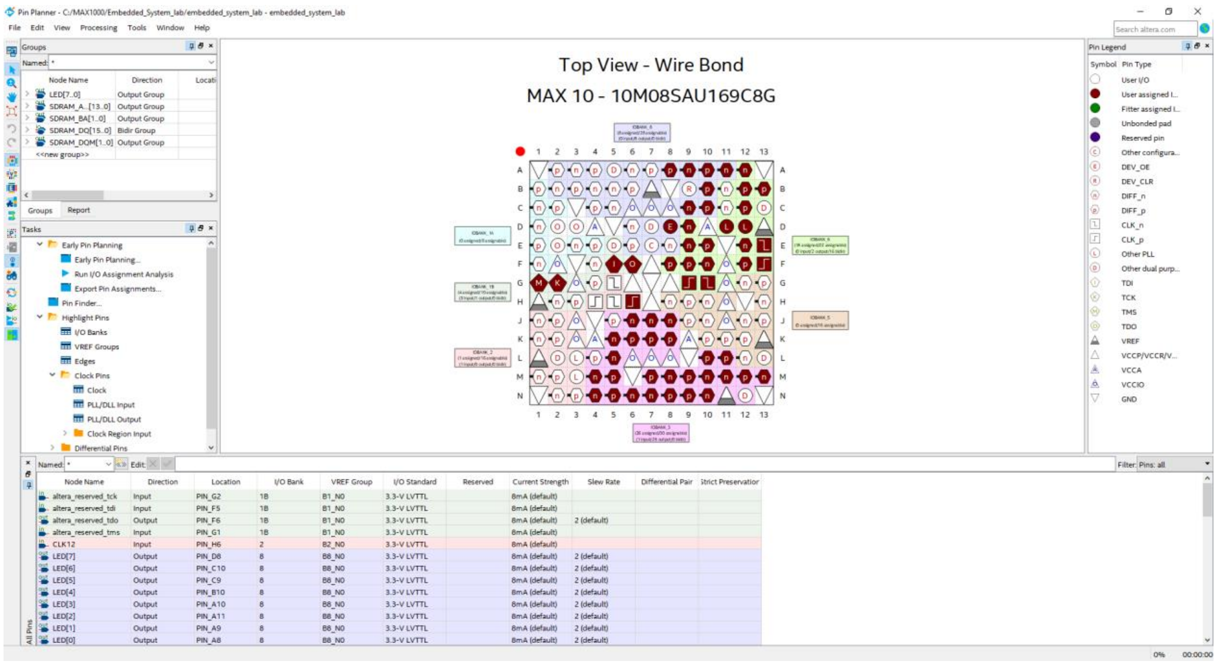
Specify the source and categories of assignments to import.

File name: ed_System_lab/embedded_system_lab_pinout.csv

☑ Copy existing assignments into embedded_system_lab.qsf.bak be

6.3.1.3 Press **OK**.

6.3.1.4 Open **Pin Planner** by clicking on button on the toolbars, or **Assignments → Pin Planner** in order to check the import. In the Pin Planner you should see the following:



Pin Planner - C:/MAX1000/Embedded_System_lab/embedded_system_lab - embedded_system_lab

Node Name	Direction
LED[7..0]	Output Group
SDRAM_A...[13..0]	Output Group
SDRAM_BA[1..0]	Output Group
SDRAM_DQ[15..0]	Bidir Group
SDRAM_DQM[1..0]	Output Group

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Strict Preservation
altera_reserved_tck	Input	PIN_G2	1B	B1_N0	3.3-V LVTTL		8mA (default)			
altera_reserved_tdi	Input	PIN_F5	1B	B1_N0	3.3-V LVTTL		8mA (default)			
altera_reserved_tdo	Output	PIN_F6	1B	B1_N0	3.3-V LVTTL		8mA (default)	2 (default)		
altera_reserved_tms	Input	PIN_G1	1B	B1_N0	3.3-V LVTTL		8mA (default)			
CLK12	Input	PIN_H6	2	B2_N0	3.3-V LVTTL		8mA (default)			
LED[7]	Output	PIN_D8	8	B8_N0	3.3-V LVTTL		8mA (default)	2 (default)		
LED[6]	Output	PIN_C10	8	B8_N0	3.3-V LVTTL		8mA (default)	2 (default)		
LED[5]	Output	PIN_C9	8	B8_N0	3.3-V LVTTL		8mA (default)	2 (default)		
LED[4]	Output	PIN_B10	8	B8_N0	3.3-V LVTTL		8mA (default)	2 (default)		
LED[3]	Output	PIN_A10	8	B8_N0	3.3-V LVTTL		8mA (default)	2 (default)		
LED[2]	Output	PIN_A11	8	B8_N0	3.3-V LVTTL		8mA (default)	2 (default)		
LED[1]	Output	PIN_A9	8	B8_N0	3.3-V LVTTL		8mA (default)	2 (default)		
LED[0]	Output	PIN_A8	8	B8_N0	3.3-V LVTTL		8mA (default)	2 (default)		

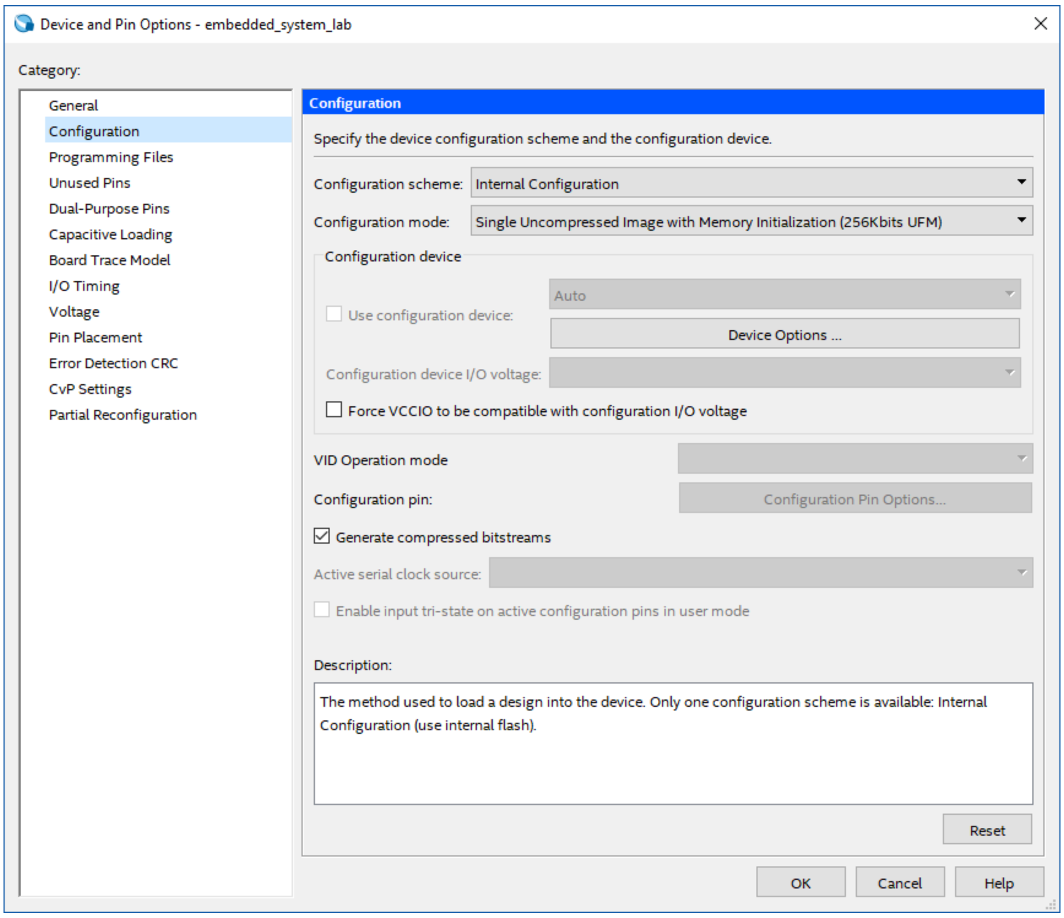
6.3.1.5 **Close** the Pin Planner.

6.3.2 Compiling the Design

6.3.2.1 Open the device settings window from **Assignments → Device…** and click on the **Device and Pin Options…** button.

6.3.2.2 Click to the **Configuration** category.

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


Device and Pin Options - embedded_system_lab

Category:

- General
- Configuration
- Programming Files
- Unused Pins
- Dual-Purpose Pins
- Capacitive Loading
- Board Trace Model
- I/O Timing
- Voltage
- Pin Placement
- Error Detection CRC
- CvP Settings
- Partial Reconfiguration

Configuration

Specify the device configuration scheme and the configuration device.

Configuration scheme: Internal Configuration

Configuration mode: Single Uncompressed Image with Memory Initialization (256Kbits UFM)

Configuration device

☐ Use configuration device: Auto

Configuration device I/O voltage:

☐ Force VCCIO to be compatible with configuration I/O voltage

VID Operation mode

Configuration pin:

☑ Generate compressed bitstreams

Active serial clock source:

☐ Enable input tri-state on active configuration pins in user mode

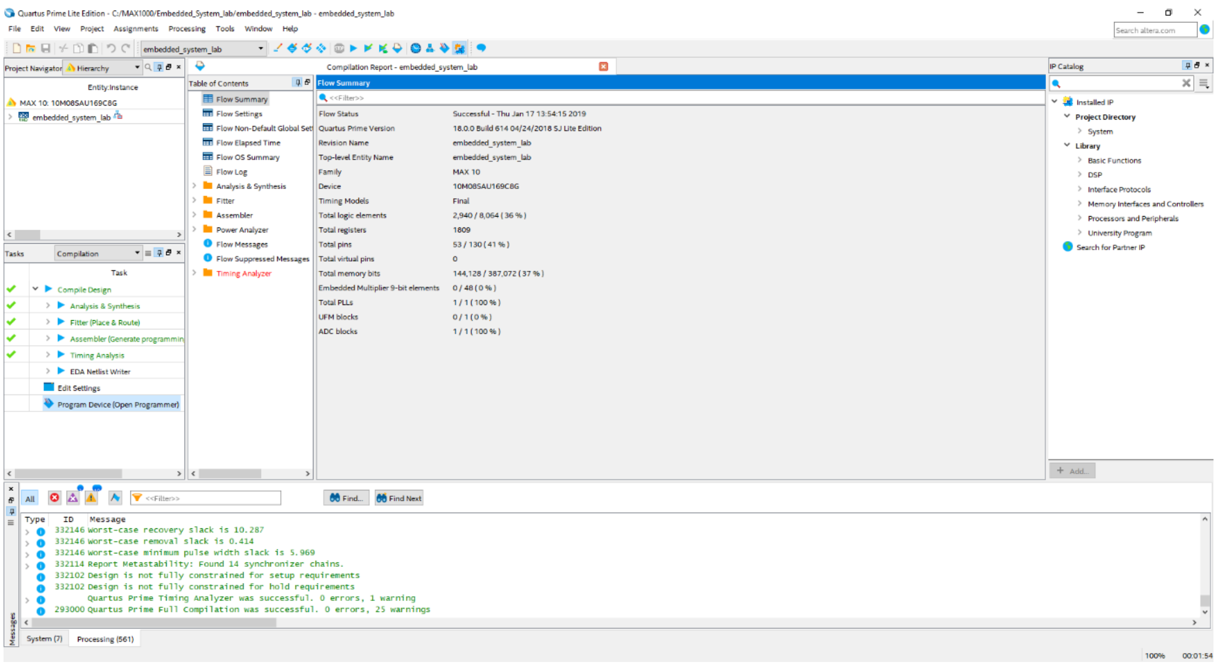
Description:

The method used to load a design into the device. Only one configuration scheme is available: Internal Configuration (use internal flash).

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

6.3.2.5 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-compilating. 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.



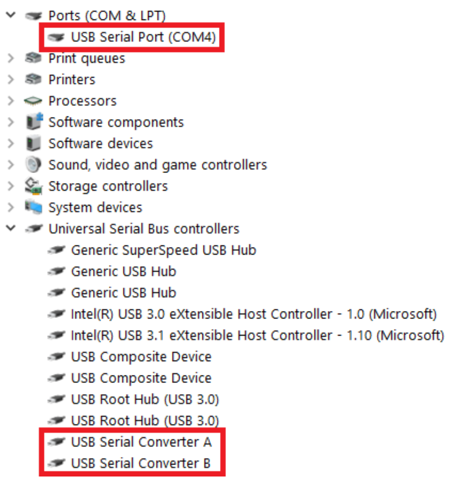
Flow Summary

Flow Status	Successful - Thu Jan 17 13:54:15 2019
Quartus Prime Version	18.0.0 Build 614 04/24/2018 SJ Lite Edition
Revision Name	embedded_system_lab
Top-level Entity Name	embedded_system_lab
Family	MAX 10
Device	10M08SAU169C8G
Timing Models	Final
Total logic elements	2,940 / 8,064 (36 %)
Total registers	1809
Total pins	53 / 130 (41 %)
Total virtual pins	0
Total memory bits	144,128 / 387,072 (37 %)
Embedded Multiplier 9-bit elements	0 / 48 (0 %)
Total PLLs	1 / 1 (100 %)
UFM blocks	0 / 1 (0 %)
ADC blocks	1 / 1 (100 %)

```
Type  ID     Message
      332146 Worst-case recovery slack is 10.287
      332146 Worst-case removal slack is 0.414
      332146 Worst-case minimum pulse width slack is 5.969
      332114 Report Metastability: Found 14 synchronizer chains.
      332102 Design is not fully constrained for setup requirements
      332102 Design is not fully constrained for hold requirements
             Quartus Prime Timing Analyzer was successful. 0 errors, 1 warning
      293000 Quartus Prime Full Compilation was successful. 0 errors, 25 warnings
```

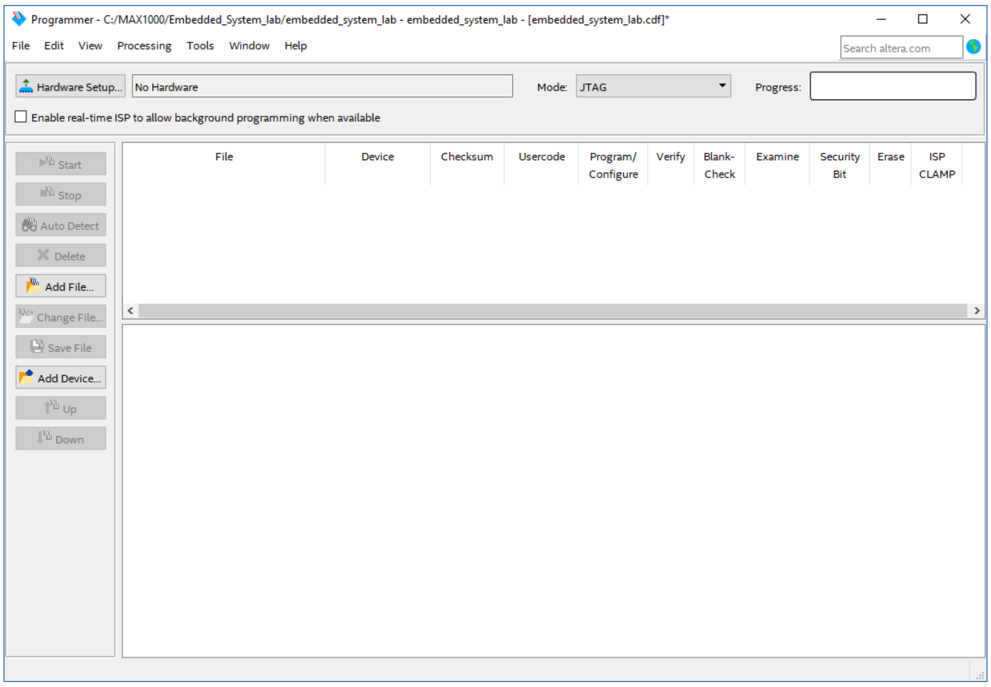
6.3.3 Configuration

6.3.3.1 Connect your MAX1000 board to your PC using an 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):



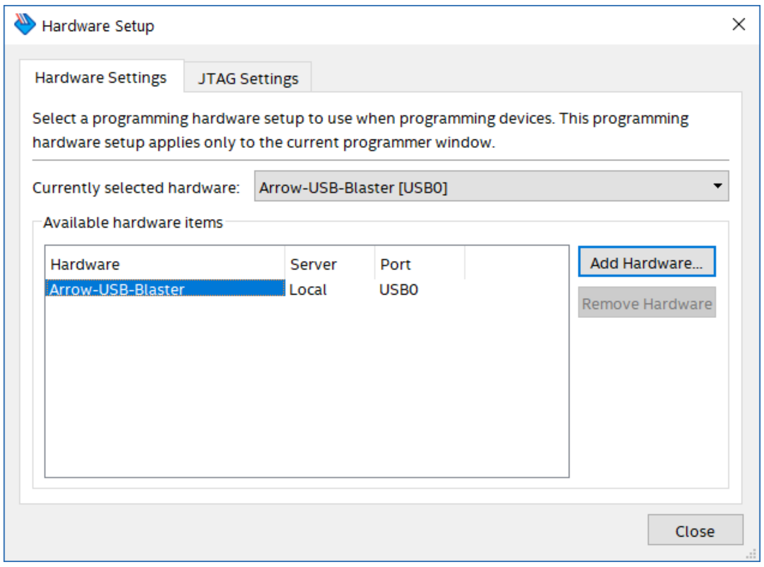
```
Ports (COM & LPT)
    USB Serial Port (COM4)
Print queues
Printers
Processors
Software components
Software devices
Sound, video and game controllers
Storage controllers
System devices
Universal Serial Bus controllers
    Generic SuperSpeed USB Hub
    Generic USB Hub
    Generic USB Hub
    Intel(R) USB 3.0 eXtensible Host Controller - 1.0 (Microsoft)
    Intel(R) USB 3.1 eXtensible Host Controller - 1.10 (Microsoft)
    USB Composite Device
    USB Composite Device
    USB Root Hub (USB 3.0)
    USB Root Hub (USB 3.0)
    USB Serial Converter A
    USB Serial Converter B
```

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


Programmer - C:/MAX1000/Embedded_System_lab/embedded_system_lab - embedded_system_lab - [embedded_system_lab.cdf]*
File Edit View Processing Tools Window Help
Hardware Setup...
Start
Stop
Auto Detect
Delete
Add File...
Change File...
Save File
Add Device...
Up
Down
File Device Checksum Usercode Program/Configure Verify Blank-Check Examine Security Bit Erase ISP CLAMP

No Hardware
Mode: JTAG
Progress:
Enable real-time ISP to allow background programming when available

6.3.3.3 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).



Select a programming hardware setup to use when programming devices. This programming hardware setup applies only to the current programmer window.

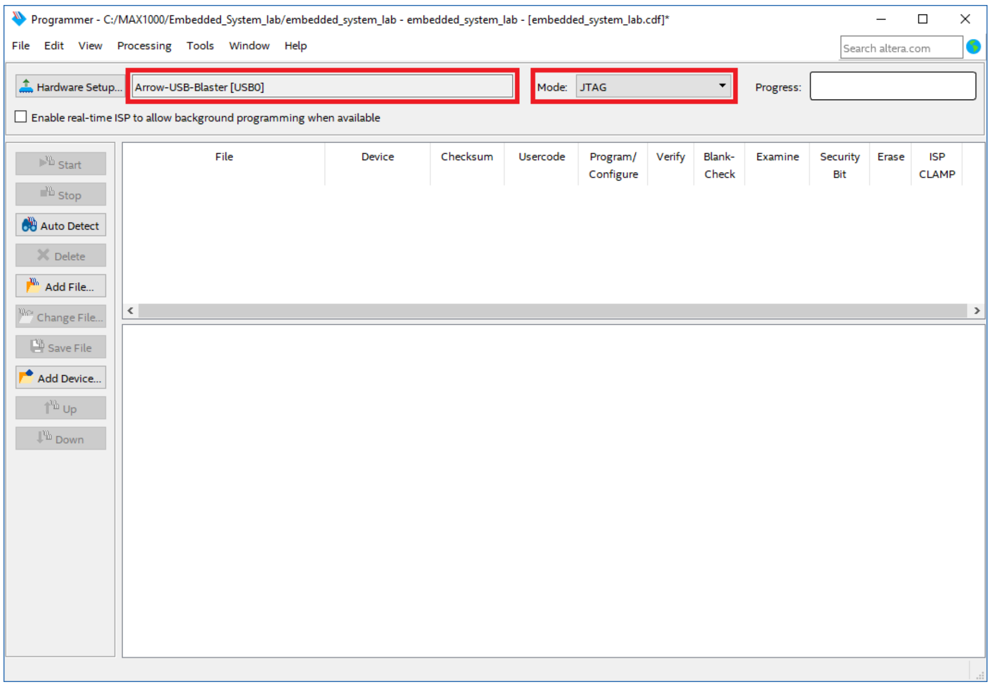
Currently selected hardware: Arrow-USB-Blaster [USB0]

Available hardware items

Hardware	Server	Port
Arrow-USB-Blaster	Local	USB0

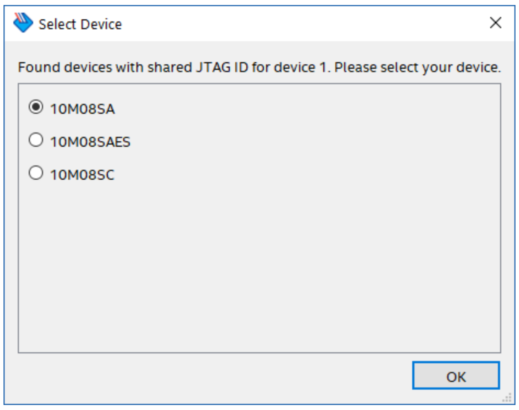
6.3.3.4 Click **Close**.

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



6.3.3.6 If the configuration has been added by default, you can skip the following steps and continue with the 6.3.4.11 point.

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

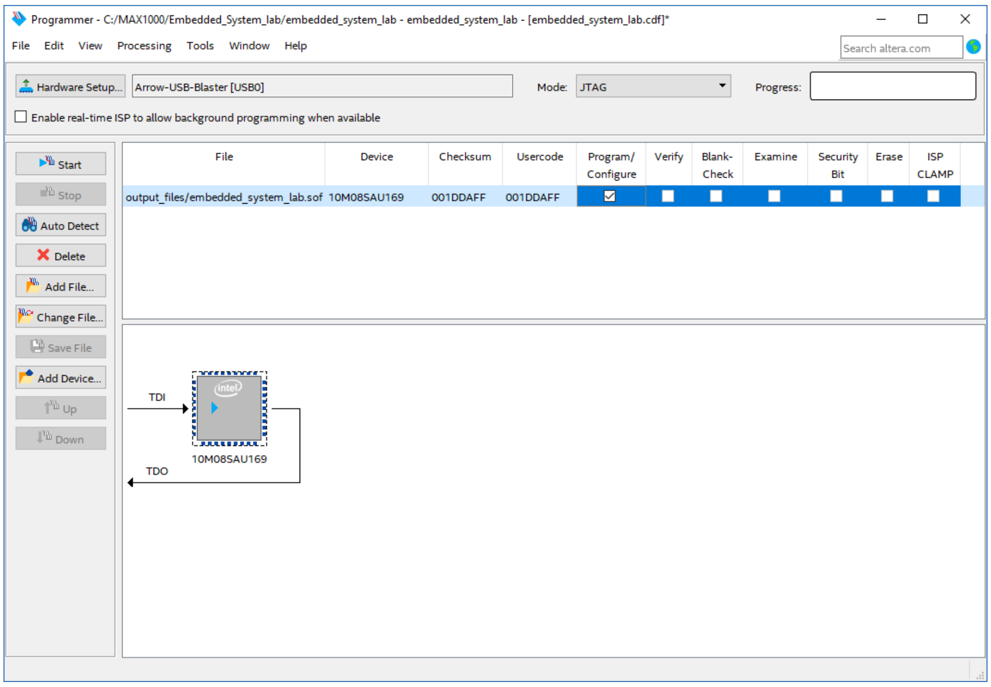

Select Device
Found devices with shared JTAG ID for device 1. Please select your device.
10M08SA
10M08SAES
10M08SC
OK

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

6.3.3.9 Navigate to **<project_directory>/output_files/** and select the **embedded_system_lab.sof** file.

6.3.3.10 Click **Open**.

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



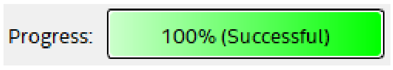
Programmer - C:/MAX1000/Embedded_System_lab/embedded_system_lab - embedded_system_lab - [embedded_system_lab.cdf]*

Hardware Setup... Arrow-USB-Blaster [USB0] Mode: JTAG Progress:

☐ Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank- Check	Examine	Security Bit	Erase	ISP CLAMP
output_files/embedded_system_lab.sof	10M08SAU169	001DDAFF	001DDAFF	☑	☐	☐	☐	☐	☐	☐

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



Progress: 100% (Successful)

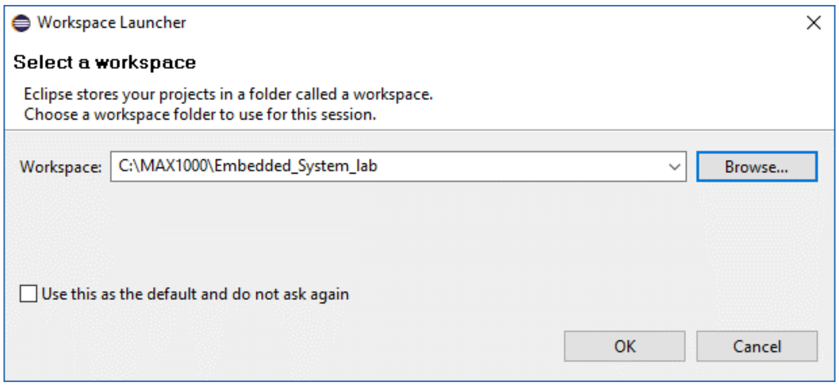
6.4 Software Design

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. The software has already been provided for you in the lab files.

6.4.1 Create a new software project

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

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

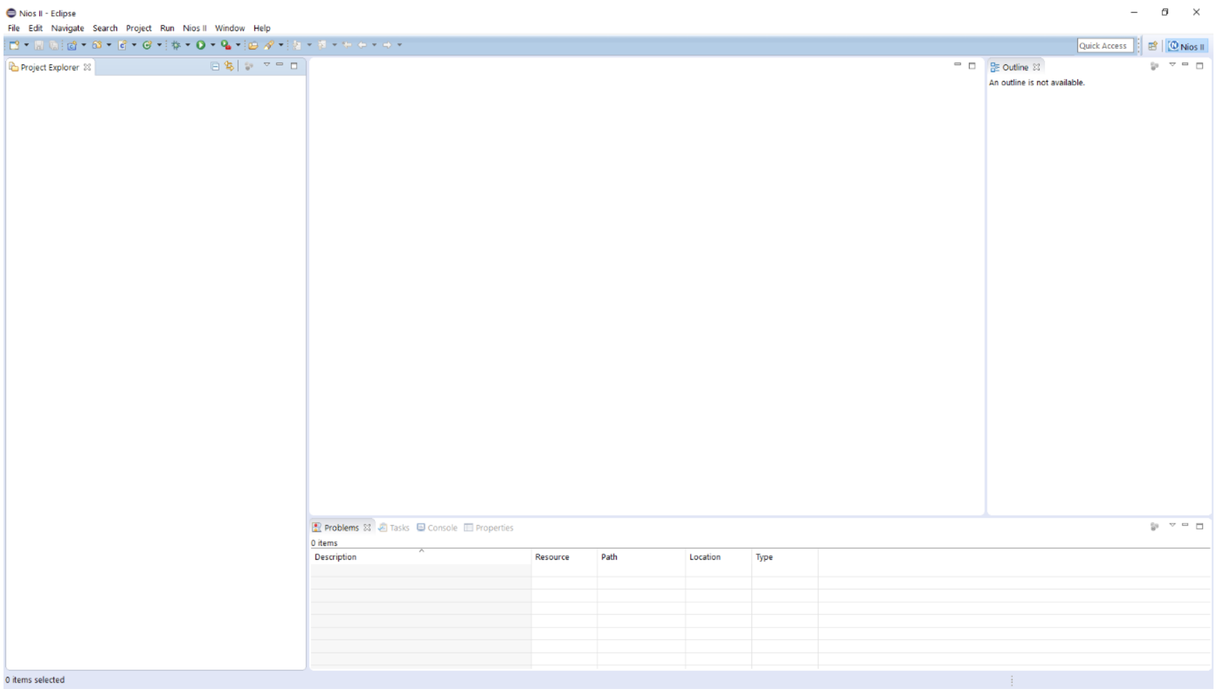

Workspace Launcher

Select a workspace

Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

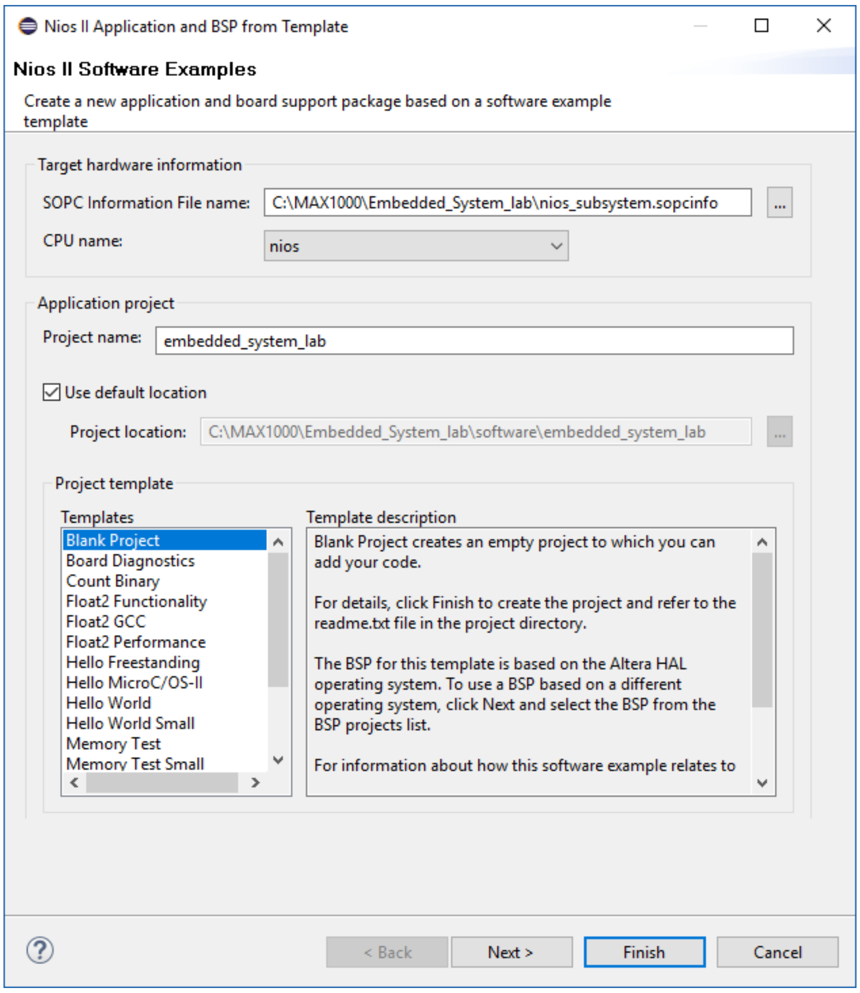
Workspace: C:\MAX1000\Embedded_System_lab

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



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

6.4.1.5 Click ... to select the **nios_subsystem.sopcinfo** from your project directory and name the project **embedded_system_lab**. Select **Blank Project** from the Templates.



Nios II Application and BSP from Template

Nios II Software Examples

Create a new application and board support package based on a software example template

Target hardware information

SOPC Information File name: C:\MAX1000\Embedded_System_lab\nios_subsystem.sopcinfo

CPU name: nios

Application project

Project name: embedded_system_lab

☑ Use default location

Project location: C:\MAX1000\Embedded_System_lab\software\embedded_system_lab

Project template

Templates: Blank Project (selected), Board Diagnostics, Count Binary, Float2 Functionality, Float2 GCC, Float2 Performance, Hello Freestanding, Hello MicroC/OS-II, Hello World, Hello World Small, Memory Test, Memory Test Small

Template description

Blank Project creates an empty project to which you can add your code.

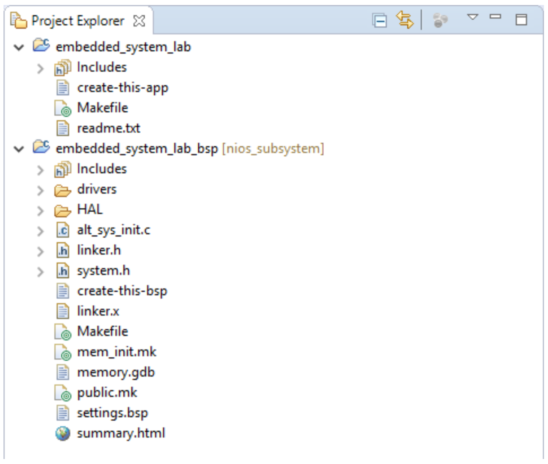
For details, click Finish to create the project and refer to the readme.txt file in the project directory.

The BSP for this template is based on the Altera HAL operating system. To use a BSP based on a different operating system, click Next and select the BSP from the BSP projects list.

For information about how this software example relates to

6.4.1.6 Click **Finish**.

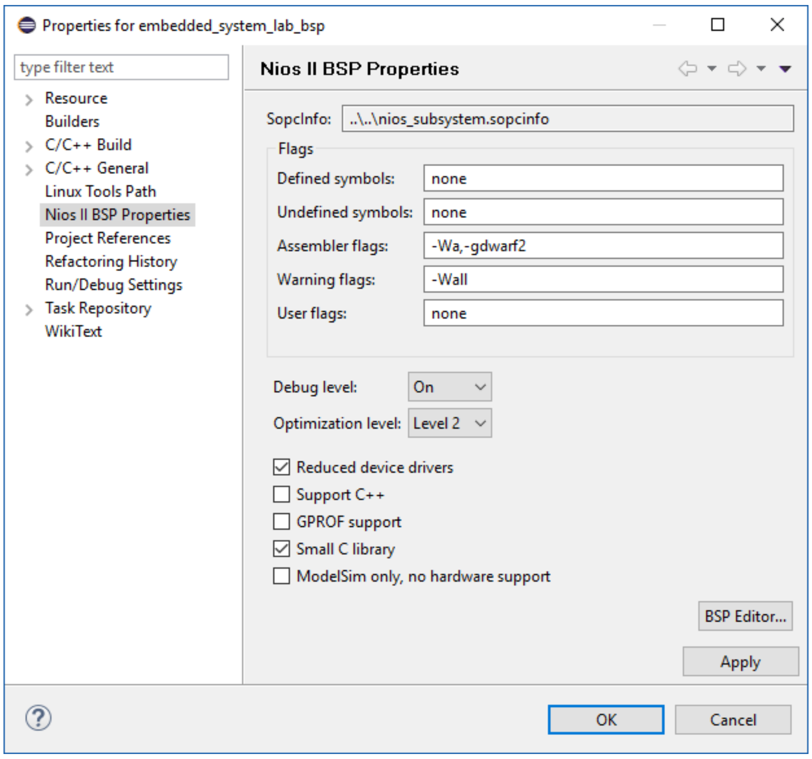
6.4.1.7 Eclipse will create two directories in the workspace, one for the application project and one for the BSP. The application directory (embedded_system_lab) is currently empty while the BSP directory (embedded_system_lab_bsp) contains software drivers, a system.h, header file, initialization source code and other software infrastructure.



6.4.1.8 Right click on the embedded_system_lab_bsp project and select **Properties** from the pop-up menu.

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

6.4.1.10 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.


Properties for embedded_system_lab_bsp

Nios II BSP Properties

SopcInfo: ..\..\nios_subsystem.sopcinfo

Flags

Defined symbols: none
Undefined symbols: none
Assembler flags: -Wa,-gdwarf2
Warning flags: -Wall
User flags: none

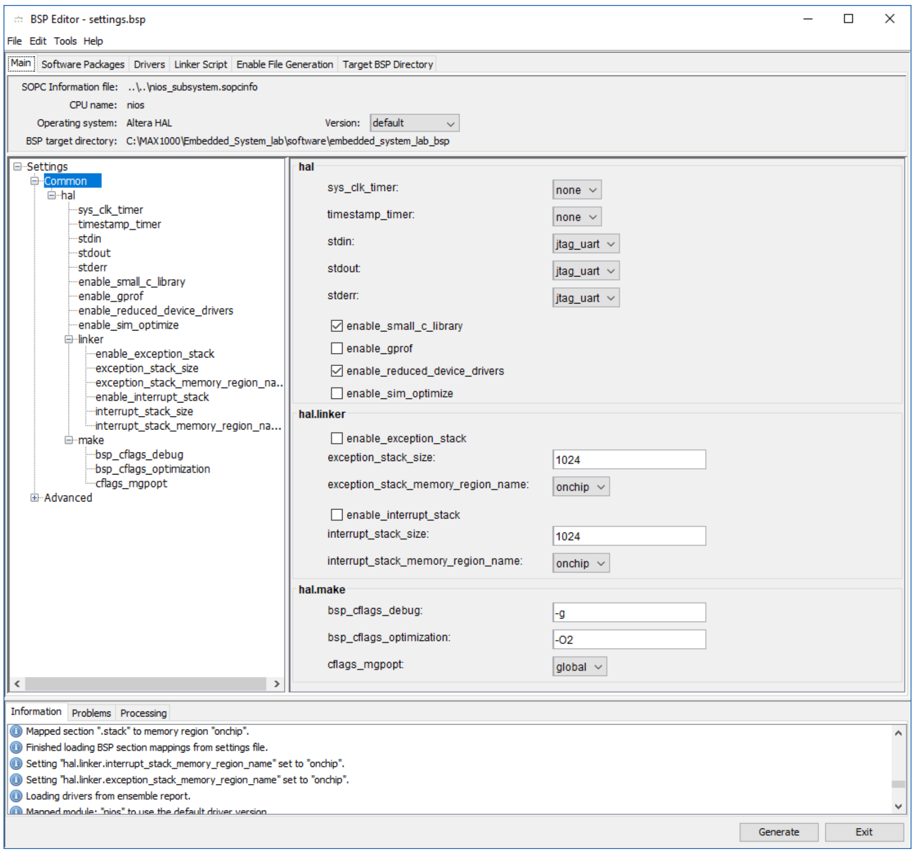
Debug level: On
Optimization level: Level 2

☑ Reduced device drivers
☐ Support C++
☐ GPROF support
☑ Small C library
☐ ModelSim only, no hardware support

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

6.4.1.12 Right click on the embedded_system_lab_bsp project and select **Nios II → BSP Editor…** from the pop-up menu.

6.4.1.13 The Nios II BSP Editor will open. In the Common settings under the main tab, ensure the settings are configured as below:



SOPC Information file: ..\..\nios_subsystem.sopcinfo
CPU name: nios
Operating system: Altera HAL
Version: default
BSP target directory: C:\MAX1000\Embedded_System_lab\software\embedded_system_lab_bsp

hal

sys_clk_timer: none
timestamp_timer: none
stdin: jtag_uart
stdout: jtag_uart
stderr: jtag_uart
☑ enable_small_c_library
☐ enable_gprof
☑ enable_reduced_device_drivers
☐ enable_sim_optimize

hal.linker

☐ enable_exception_stack
exception_stack_size: 1024
exception_stack_memory_region_name: onchip
☐ enable_interrupt_stack
interrupt_stack_size: 1024
interrupt_stack_memory_region_name: onchip

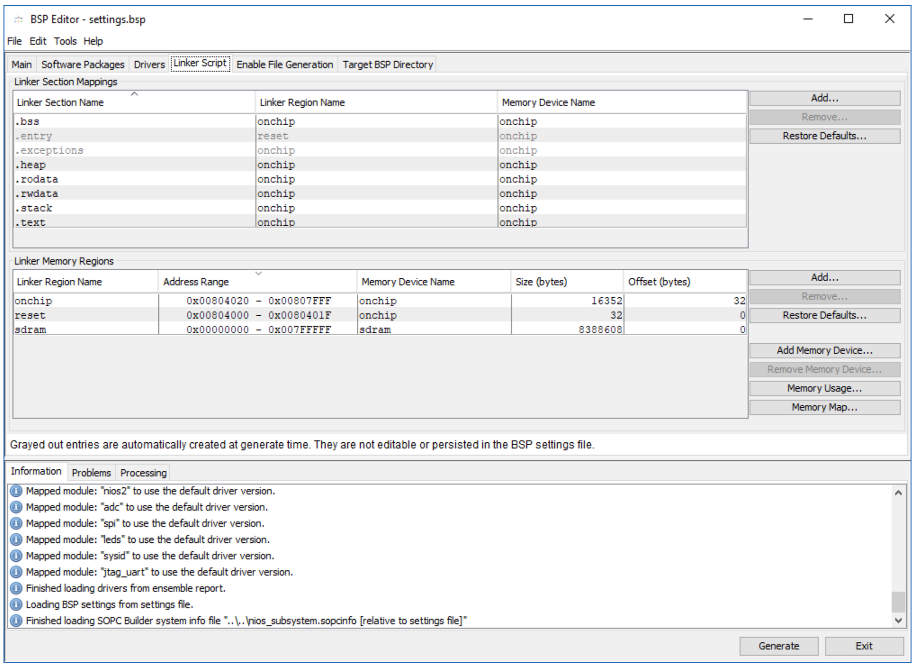
hal.make

bsp_cflags_debug: -g
bsp_cflags_optimization: -O2
cflags_mgpopt: global

```
Mapped section ".stack" to memory region "onchip".
Finished loading BSP section mappings from settings file.
Setting "hal.linker.interrupt_stack_memory_region_name" set to "onchip".
Setting "hal.linker.exception_stack_memory_region_name" set to "onchip".
Loading drivers from ensemble report.
Mapped module: "nios" to use the default driver version
```

Notice that since there is no operating system in this tab, the stdout, stdin, and stderr messages are reported through the JTAG UART that you will be able to see in the Nios II Console in Eclipse. On-chip memory will be used for processor code storage, data storage, the exception and interrupt stack.

6.4.1.14 Click on the Linker Script tab and set **onchip** in the Linker Region Name column for .heap, .rodata, .rwdata and .stack.



Linker Section Mappings

Linker Section Name	Linker Region Name	Memory Device Name
.bss	onchip	onchip
.entry	reset	onchip
.exceptions	onchip	onchip
.heap	onchip	onchip
.rodata	onchip	onchip
.rwdata	onchip	onchip
.stack	onchip	onchip
.text	onchip	onchip

Linker Memory Regions

Linker Region Name	Address Range	Memory Device Name	Size (bytes)	Offset (bytes)
onchip	0x00804020 - 0x00807FFF	onchip	16352	32
reset	0x00804000 - 0x0080401F	onchip	32	0
sdram	0x00000000 - 0x007FFFFF	sdram	8388608	0

Grayed out entries are automatically created at generate time. They are not editable or persisted in the BSP settings file.

```
Mapped module: "nios2" to use the default driver version.
Mapped module: "adc" to use the default driver version.
Mapped module: "spi" to use the default driver version.
Mapped module: "leds" to use the default driver version.
Mapped module: "sysid" to use the default driver version.
Mapped module: "jtag_uart" to use the default driver version.
Finished loading drivers from ensemble report.
Loading BSP settings from settings file.
Finished loading SOPC Builder system info file "..\..\nios_subsystem.sopcinfo [relative to settings file]"
```

BSP Editor - settings.bsp
File Edit Tools Help
Main Software Packages Drivers Linker Script Enable File Generation Target BSP Directory
Add...
Remove...
Restore Defaults...
Add Memory Device...
Remove Memory Device...
Memory Usage...
Memory Map...
Information Problems Processing
Generate
Exit

Feel free to explore the BSP editor. The Drivers tab gives the user control over built into the BSP. The Linker Script tab provides a mechanism to adjust what memory regions are utilized for certain purposes.

6.4.1.15 Click **Generate** button to update the PSB and select **Exit** to close it once the process is finished.

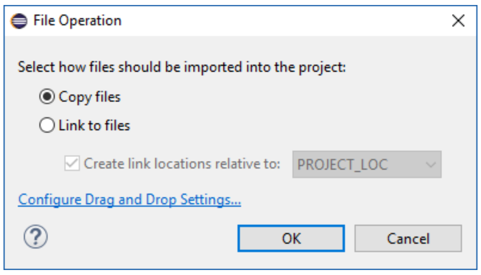
6.5 Accelerometer lab

6.5.1 Add source code to the project

Note: The C source file have been provided for you in this lab. All that needs to be done is to copy it to your workspace.

6.5.1.1 From Windows Explorer, navigate to your main project directory. There you will find a file named **accelerometer.c** which you will need to copy to this project.

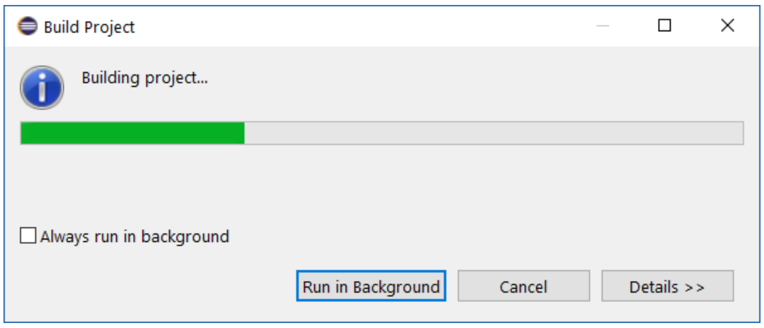
6.5.1.2 Select the accelerometer.c file and drag it into the embedded_system_lab directory in Eclipse. Select the **Copy files** option in the pop-up and click **OK**.


File Operation
Select how files should be imported into the project:
Copy files
Link to files
Create link locations relative to:
PROJECT_LOC
Configure Drag and Drop Settings...
OK
Cancel

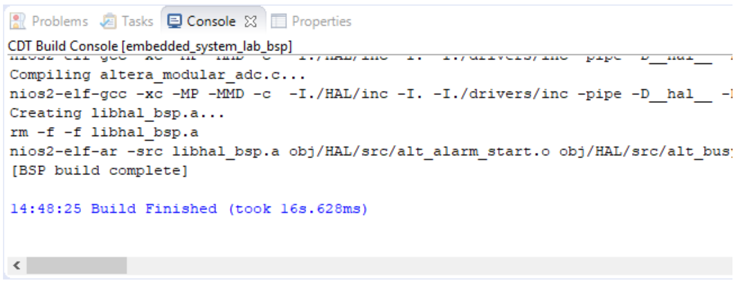
You should now see the new file appear under the embedded_system_lab project in the Project Explorer.

6.5.2 Build the software

6.5.2.1 Right click on the embedded_system_lab_bsp project and select **Build Project** from the pop-up menu.



6.5.2.2 When it finished, repeat the previous step for the embedded_system_lab application project.

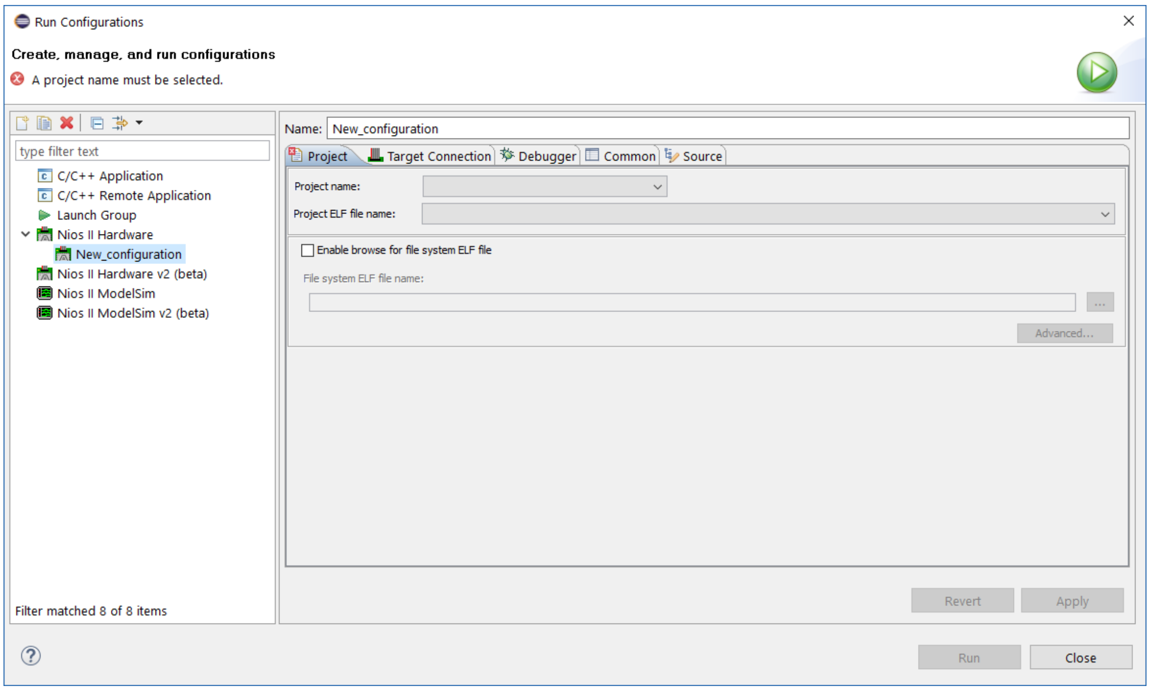


CDT Build Console [embedded_system_lab_bsp]

```
Compiling altera_modular_adc.c...
nios2-elf-gcc -xc -MP -MMD -c  -I./HAL/inc -I. -I./drivers/inc -pipe -D__hal__ -
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/alt_bus
[BSP build complete]

14:48:25 Build Finished (took 16s.628ms)
```

6.5.3 Run the application

6.5.3.1 Select embedded_system_lab and go to **Run → Run Configurations…** and double click to **Nios II Hardware** to add a new configuration.



Create, manage, and run configurations

A project name must be selected.

type filter text

- C/C++ Application
- C/C++ Remote Application
- Launch Group
- Nios II Hardware
  - New_configuration
- Nios II Hardware v2 (beta)
- Nios II ModelSim
- Nios II ModelSim v2 (beta)

Filter matched 8 of 8 items

Name: New_configuration

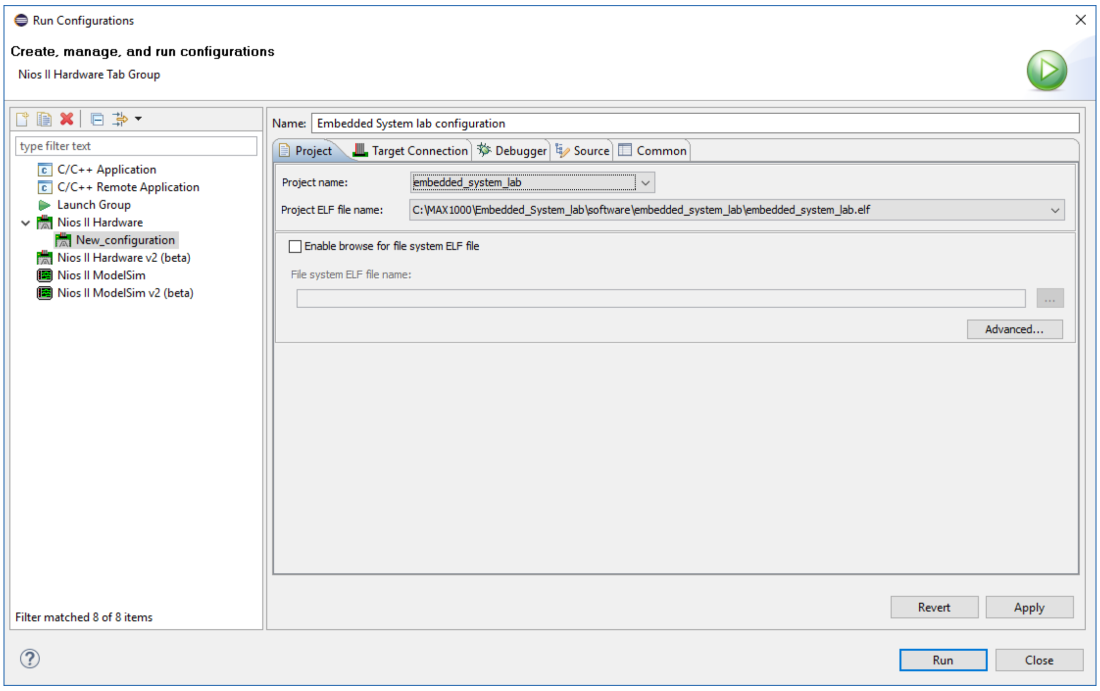
Project name:

Project ELF file name:

Enable browse for file system ELF file

File system ELF file name:

6.5.3.2 Rename it to **Embedded System lab configuration** and on the Project tab select **embedded_system_lab** from the drop-down menu for the Project name.


Run Configurations

Create, manage, and run configurations

Nios II Hardware Tab Group

Name: Embedded System lab configuration

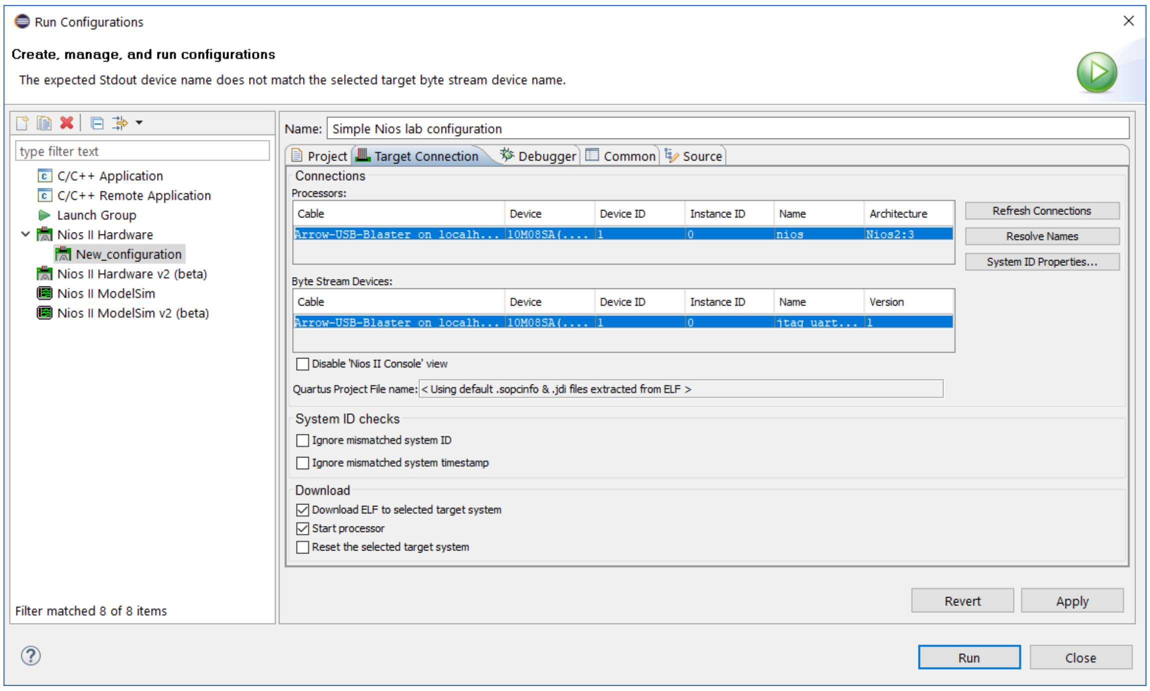
Project name: embedded_system_lab

Project ELF file name: C:\MAX1000\Embedded_System_lab\software\embedded_system_lab\embedded_system_lab.elf

☐ Enable browse for file system ELF file

File system ELF file name:

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


Run Configurations

Create, manage, and run configurations

The expected Stdout device name does not match the selected target byte stream device name.

Name: Simple Nios lab configuration

Connections

Processors:

Cable	Device	Device ID	Instance ID	Name	Architecture
Arrow-USB-Blaster on localh...	10M08SA(....	1	0	nios	Nios2:3

Byte Stream Devices:

Cable	Device	Device ID	Instance ID	Name	Version
Arrow-USB-Blaster on localh...	10M08SA(....	1	0	jtag_uart...	1

☐ Disable 'Nios II Console' view

Quartus Project File name: < Using default .sopcinfo & .jdi files extracted from ELF >

System ID checks

☐ Ignore mismatched system ID

☐ Ignore mismatched system timestamp

Download

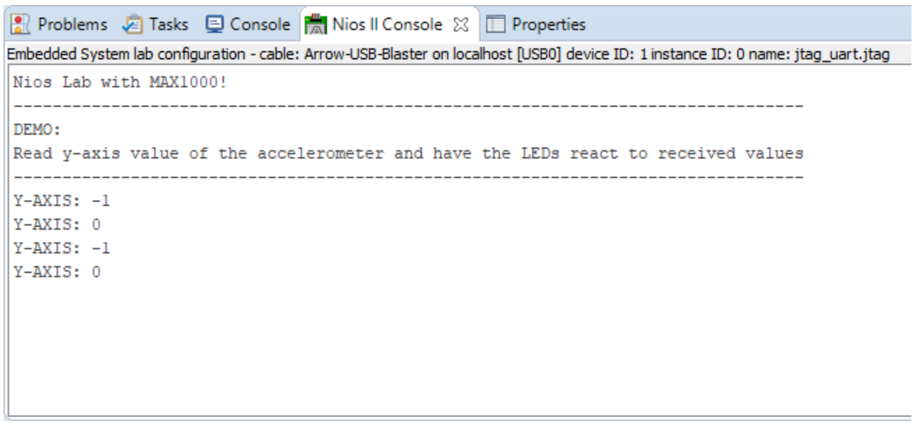
☑ Download ELF to selected target system

☑ Start processor

☐ Reset the selected target system

6.5.3.4 Click **Apply** and **Run**.

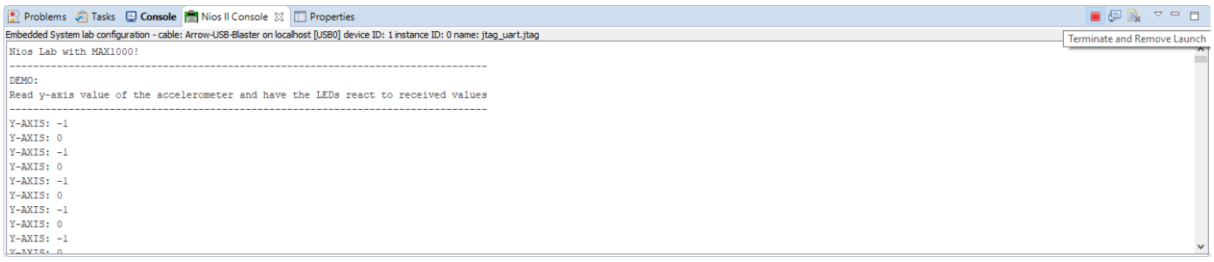
6.5.3.5 After a few second, the Nios II Console should open at the bottom of the Eclipse.



```
Embedded System lab configuration - cable: Arrow-USB-Blaster on localhost [USB0] device ID: 1 instance ID: 0 name: jtag_uart.jtag
Nios Lab with MAX1000!
--------------------------------------------------------------------------------
DEMO:
Read y-axis value of the accelerometer and have the LEDs react to received values
--------------------------------------------------------------------------------
Y-AXIS: -1
Y-AXIS: 0
Y-AXIS: -1
Y-AXIS: 0
```

After this message, the software downloaded to MAX1000 will obtain the Y-axis data from its onboard accelerometer and toggle it's LEDs accordingly to the tilt level. Every 10ms the Y-axis value will be sent to Nios II Console window.

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



```
Embedded System lab configuration - cable: Arrow-USB-Blaster on localhost [USB0] device ID: 1 instance ID: 0 name: jtag_uart.jtag
Nios Lab with MAX1000!
--------------------------------------------------------------------------------
DEMO:
Read y-axis value of the accelerometer and have the LEDs react to received values
--------------------------------------------------------------------------------
Y-AXIS: -1
Y-AXIS: 0
Y-AXIS: -1
Y-AXIS: 0
Y-AXIS: -1
Y-AXIS: 0
Y-AXIS: -1
Y-AXIS: 0
Y-AXIS: -1
```

CONGRATULATIONS! YOU HAVE SUCCESSFULLY COMPLETED THE EMBEDDED SYSTEM LAB!

5 Revision History

Version	Change Log	Date of Change
V1.0	Initial Version	17/08/2017
V2.0	- Update the naming of the tools within Quartus to accommodate the recent version changes - Update website links - Corrections / clarity changes - Added usage of areset conduit of the PLL component - New version of tools generated new names for conduit signals/base addresses, appropriate changes in nios_lab_top.vhd and main.c file to address that - Size of the onchip_ram block reduced to 16kB from 32kB	27/09/2018
V3.0	- Improving the lab into the embedded system lab - Update the design, and make preparation for additional software labs: add ADC, SDRAM controller, renaming modules and files - Corrections / clarity changes - Formal changes	17/01/2019

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 designees.

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.