# Creating projects with Nios II for Altera De2i-150

**By Trace Stewart**
**CPE 409**

# CONTENTS

**Chapter 1 Hardware Design ........................................................................ 1**

1.1 Required Features ........................................................................ 1

1.2 Creation of Hardware Design ........................................................................ 1

**Chapter 2 Programming the FPGA........................................................................ 33**

**Chapter 3 NIOS II Software Build Tools for Eclipse ........................................................................ 42**

3.1 Creating a simple Hello World Project ........................................................................ 42

3.2 DCT and Quantization…….................................................................... 47

# Chapter 1 *Hardware Design*

This tutorial provides comprehensive information that will help you understand how to create a FPGA based SOPC system with the Nios II processor on your FPGA development board and run software upon it.

## 1.1 Required Features

The Nios II processor core is a soft-core central processing unit that you could program onto an Altera field programmable gate array (FPGA). This tutorial illustrates you to the basic flow covering hardware creation and software building. You are assumed to have the latest Quartus II and NIOS II EDS software installed and quite familiar with the operation of Windows OS. If you use a different Quartus II and NIOS II EDS version, there will have some small difference during the operation. You are also assumed to possess a DE2i-150 development board (other kinds of dev. Board based on Altera FPGA chip also supported).

The example NIOS II standard hardware system provides the following necessary components:

- Nios II processor core, that's where the software will be executed
- On-chip memory to store and run the software
- Sdram to use other than On-chip memory to store and run software
- JTAG link for communication between the host computer and target
- Hardware (typically using a USB-Blaster cable)
- PIO registers will be used to send data between FPGA and Nios Processor

## 1.2 Creation of Hardware Design

This section describes how to create the hardware system used for this project including the SOPC feature.

1.) First step is to open the Quartus program (We will be using Quartus 17.1 in this tutorial but other Quartus versions will work will some minor tweaking). You will then **select File→New Project Wizard.** See Figures 1-1 and 1-2.

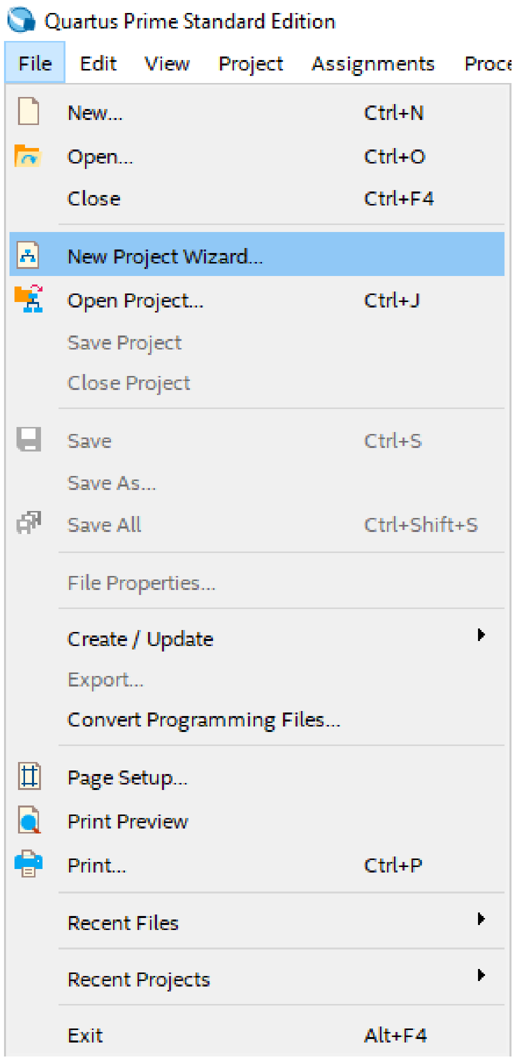
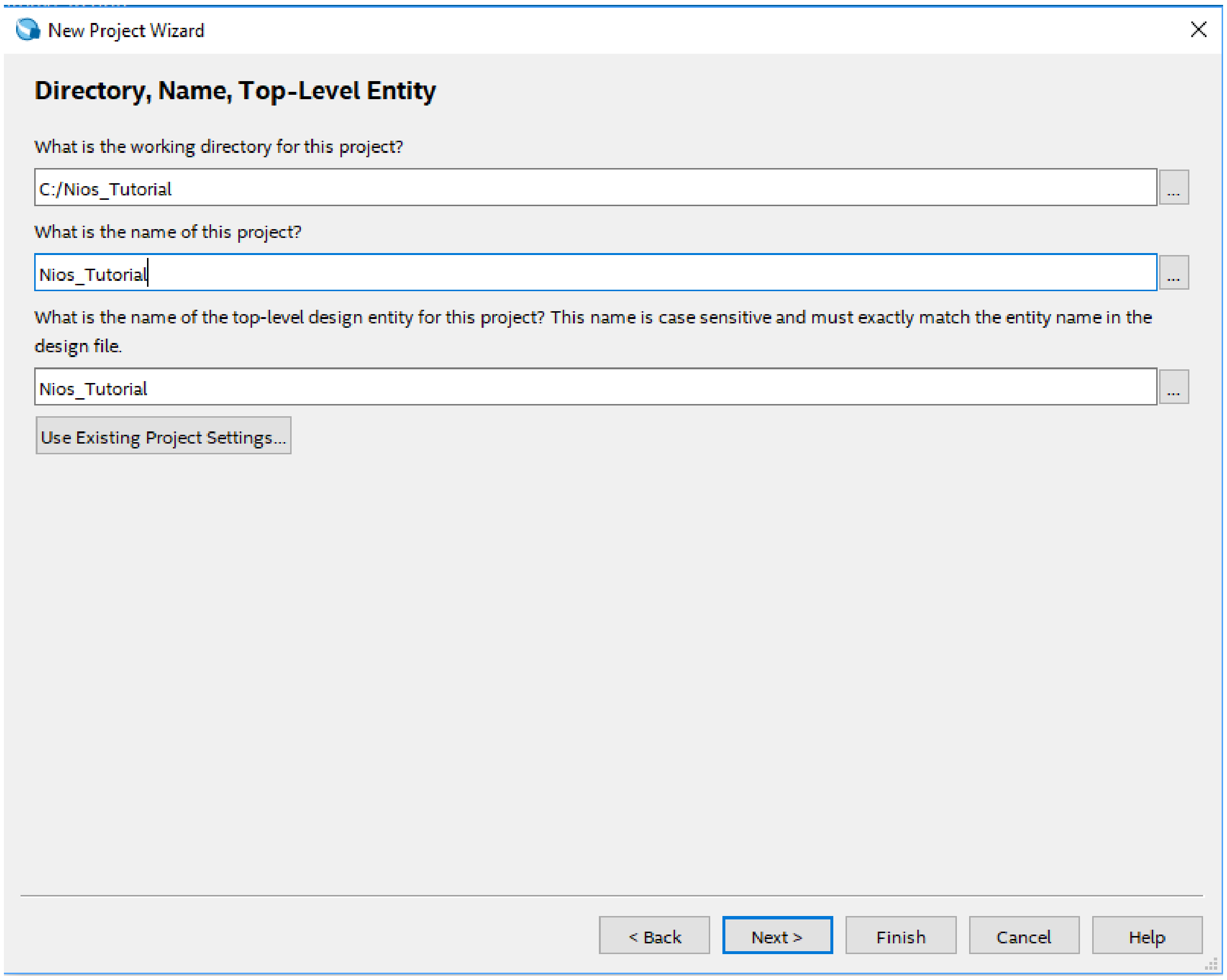


**Figure 1-1 Opening New project Wizard**


New Project Wizard

Directory, Name, Top-Level Entity

What is the working directory for this project?

C:/Nios_Tutorial

What is the name of this project?

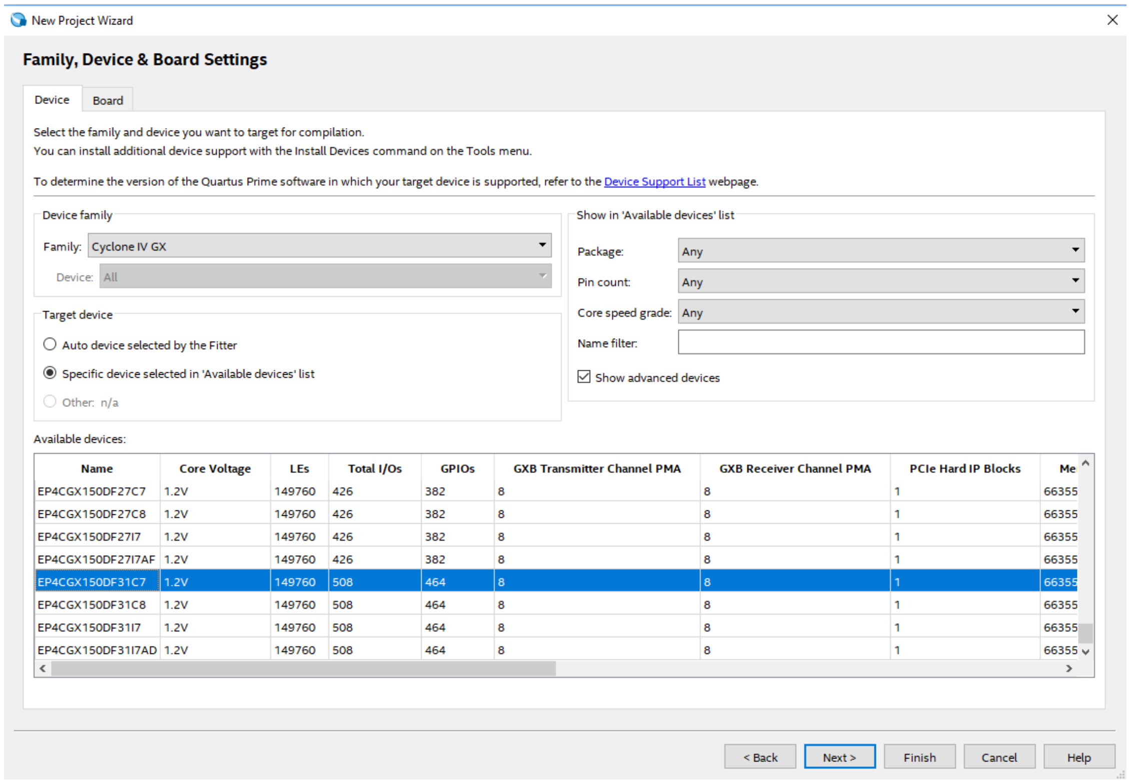
Nios_Tutorial

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.

Nios_Tutorial

**Figure 1-2 Input the working directory, the project name, and top-level design entity.**

2.) Click **Next** through the windows until you get to the **Family, Device, & Board Settings** window. You will then choose your Device Family and the device name as in Figure 1-3. We are using the De2i-150 board, so we choose **Cyclone IV GX** with name **EP4CGX150DF31C7**.


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: Cyclone IV GX
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: Any
Pin count: Any
Core speed grade: Any
Name filter:
☑ Show advanced devices

Available devices:

| Name | Core Voltage | LEs | Total I/Os | GPIOs | GXB Transmitter Channel PMA | GXB Receiver Channel PMA | PCIe Hard IP Blocks | Me |
|---|---|---|---|---|---|---|---|---|
| EP4CGX150DF27C7 | 1.2V | 149760 | 426 | 382 | 8 | 8 | 1 | 66355 |
| EP4CGX150DF27C8 | 1.2V | 149760 | 426 | 382 | 8 | 8 | 1 | 66355 |
| EP4CGX150DF27I7 | 1.2V | 149760 | 426 | 382 | 8 | 8 | 1 | 66355 |
| EP4CGX150DF27I7AF | 1.2V | 149760 | 426 | 382 | 8 | 8 | 1 | 66355 |
| EP4CGX150DF31C7 | 1.2V | 149760 | 508 | 464 | 8 | 8 | 1 | 66355 |
| EP4CGX150DF31C8 | 1.2V | 149760 | 508 | 464 | 8 | 8 | 1 | 66355 |
| EP4CGX150DF31I7 | 1.2V | 149760 | 508 | 464 | 8 | 8 | 1 | 66355 |
| EP4CGX150DF31I7AD | 1.2V | 149760 | 508 | 464 | 8 | 8 | 1 | 66355 |

**Figure 1-3 New projects wizard Family, Device, & Board Settings**

3.) Hit Next → finish and it will create a new project as in Figure 1-4.

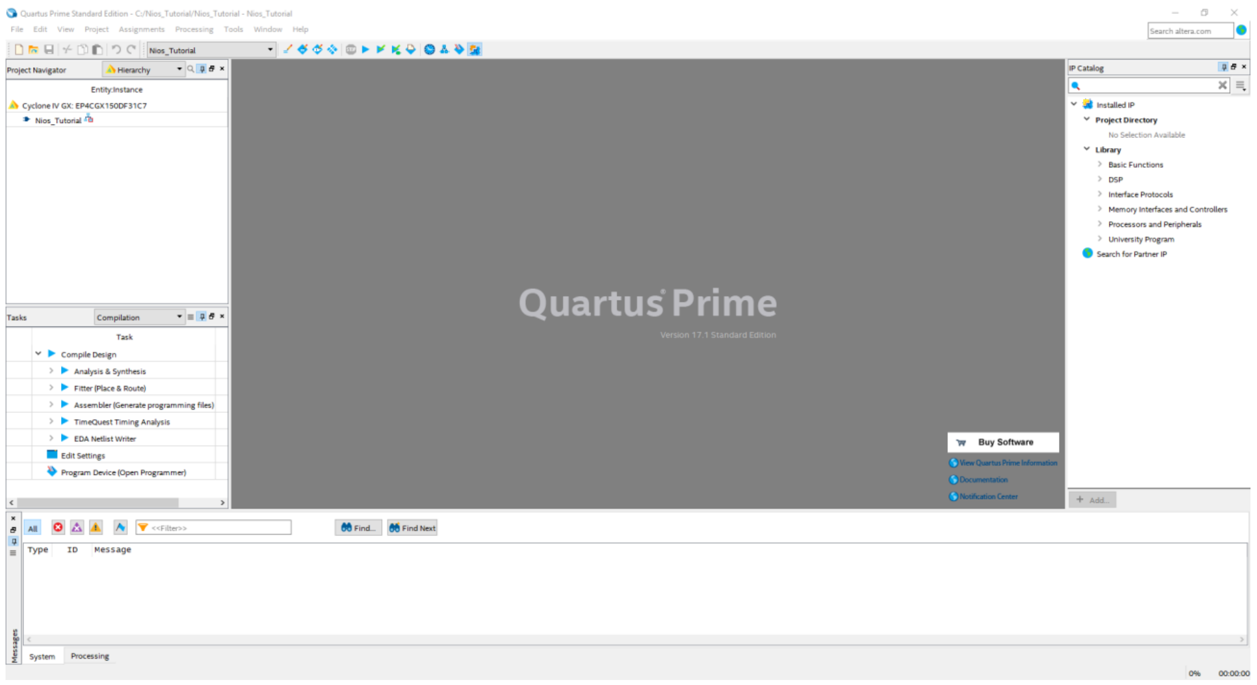


**Figure 1-4 The new project has been created**

4.) Now we will create our system with the Platform Designer tool (Formally known as Qsys). To do this choose **Tools → Platform Designer** as shown in figure 1-5. Platform Designer already creates a new system for you, but you can start a new system manually if you want. Choose **File → New System** to do that.

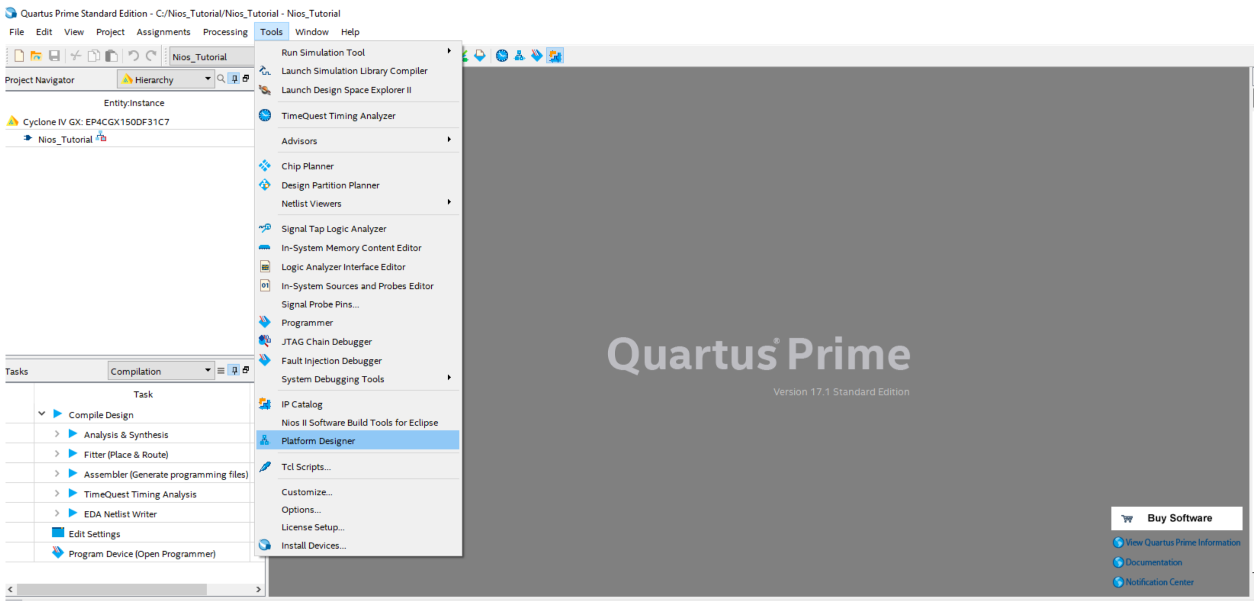
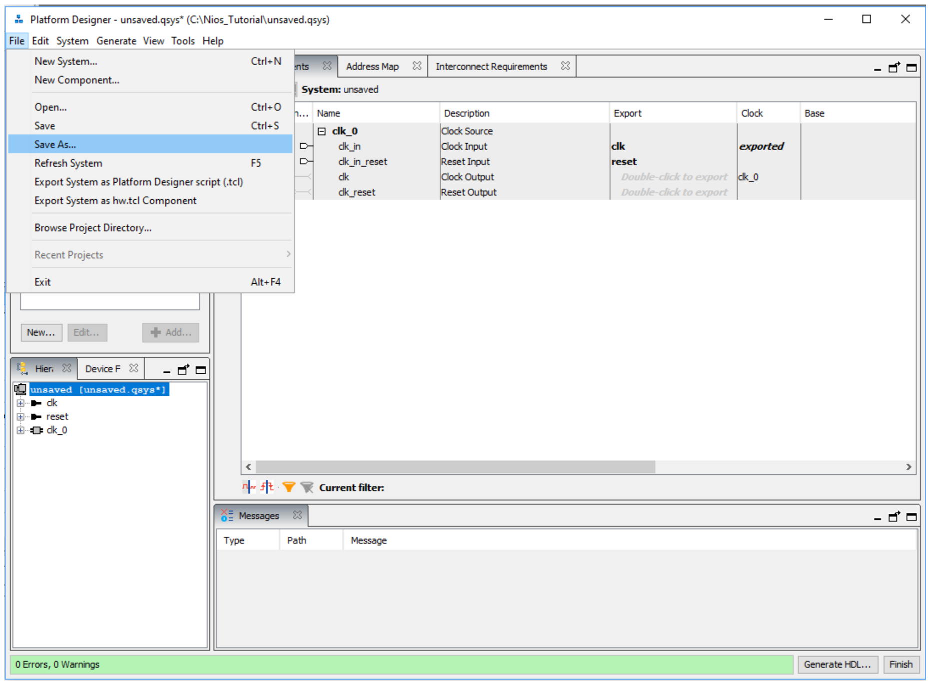

Quartus Prime Standard Edition - C:/Nios_Tutorial/Nios_Tutorial - Nios_Tutorial
File Edit View Project Assignments Processing Tools Window Help
Run Simulation Tool
Launch Simulation Library Compiler
Launch Design Space Explorer II
TimeQuest Timing Analyzer
Advisors
Chip Planner
Design Partition Planner
Netlist Viewers
Signal Tap Logic Analyzer
In-System Memory Content Editor
Logic Analyzer Interface Editor
In-System Sources and Probes Editor
Signal Probe Pins...
Programmer
JTAG Chain Debugger
Fault Injection Debugger
System Debugging Tools
IP Catalog
Nios II Software Build Tools for Eclipse
Platform Designer
Tcl Scripts...
Customize...
Options...
License Setup...
Install Devices...
Project Navigator
Hierarchy
Entity:Instance
Cyclone IV GX: EP4CGX150DF31C7
Nios_Tutorial
Tasks
Compilation
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming files)
TimeQuest Timing Analysis
EDA Netlist Writer
Edit Settings
Program Device (Open Programmer)
Quartus Prime
Version 17.1 Standard Edition
Buy Software
View Quartus Prime Information
Documentation
Notification Center

**Figure 1-5 Opening Platform Designer for creating system.**

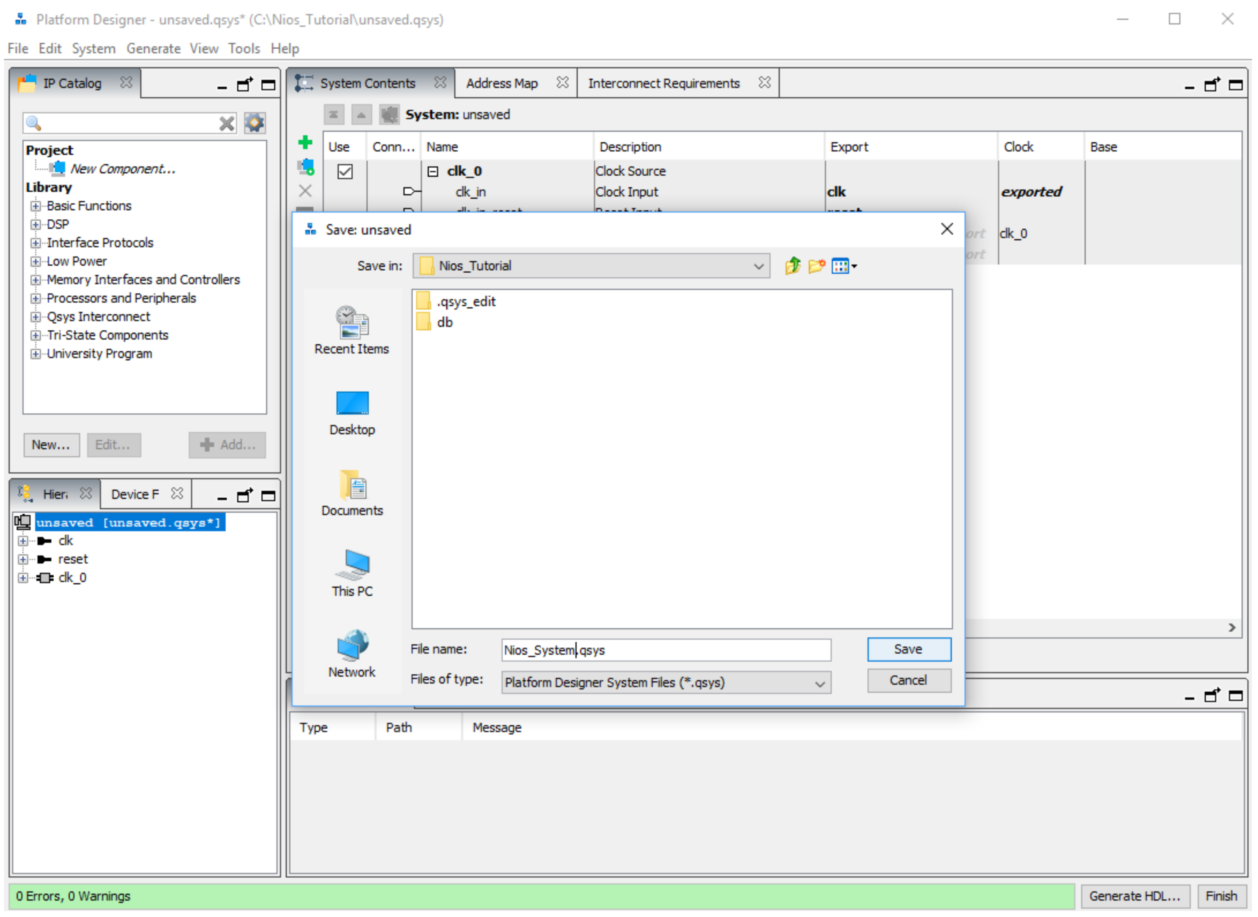
5.) After opening the new system, we want to save it under whatever name we choose as in Figure 1-6 and 1-7.



System: unsaved

| Name | Description | Export | Clock | Base |
|---|---|---|---|---|
| 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* | | |

**Figure 1-6 Saving new system under our preferred name.**



Save in: Nios_Tutorial

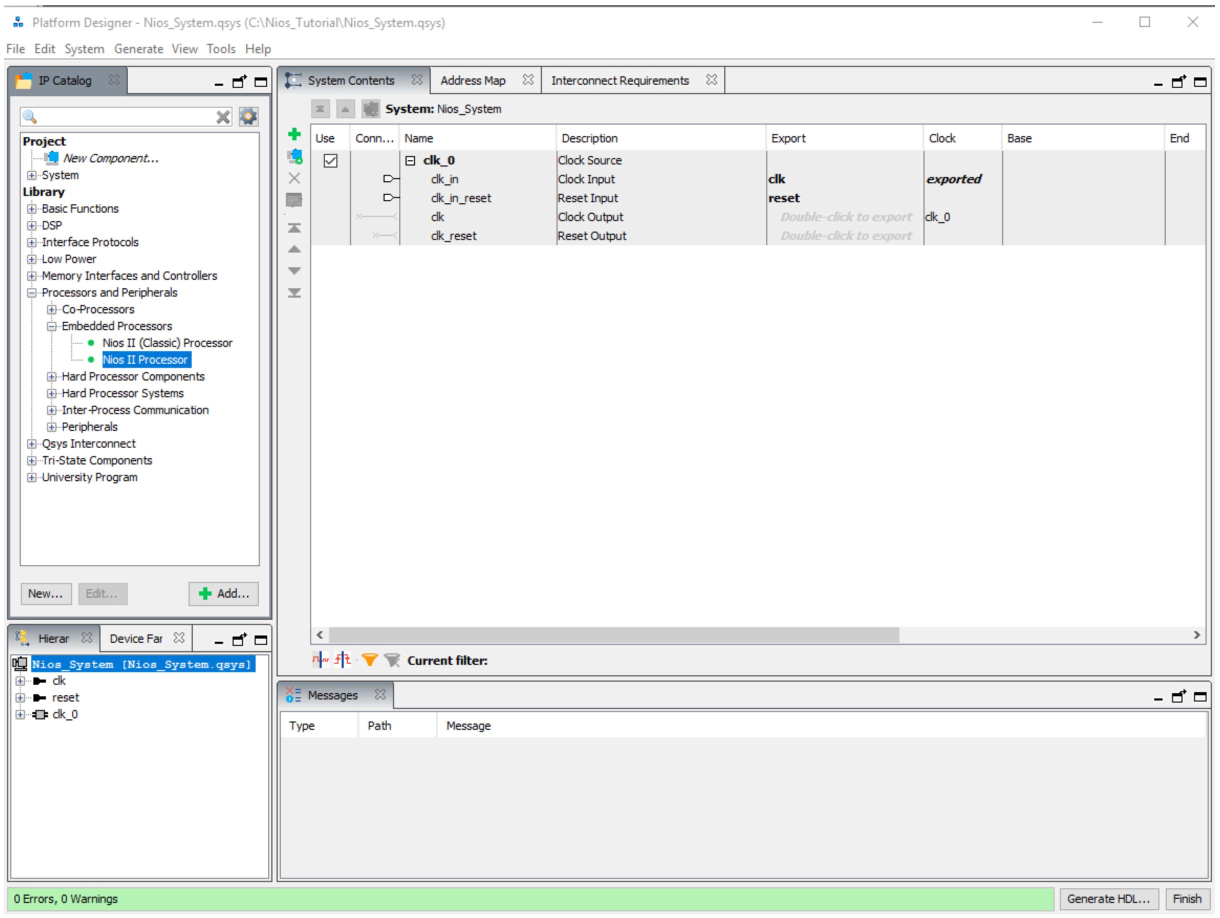
File name: Nios_System.qsys

Files of type: Platform Designer System Files (*.qsys)

0 Errors, 0 Warnings

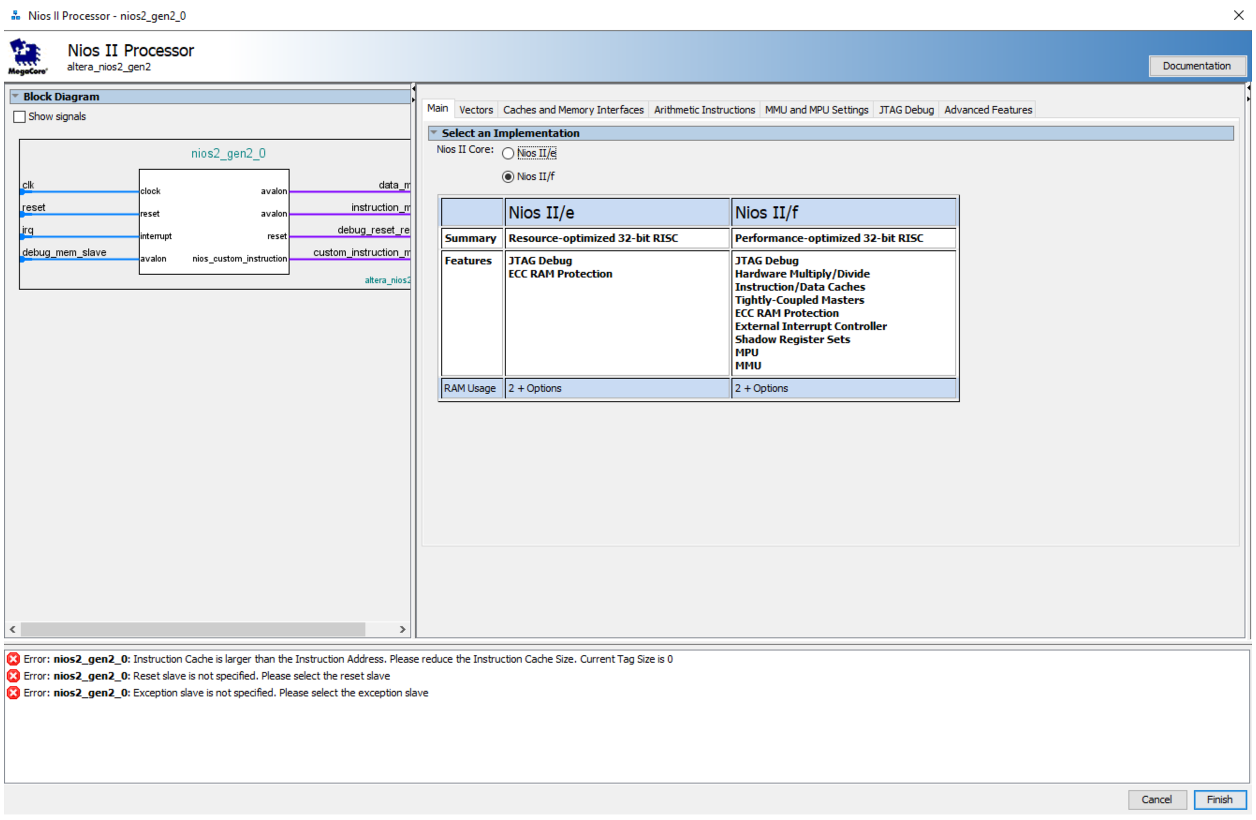
**Figure 1-7 Save new system under “Nios_System.qsys”**

6.) Now we will start adding different components to our system to get everything connected. As you can see, the system already adds a clock for us to start out with. The first thing we are going to add is our Nios II processor. To do this we will go to **Library →Processors and Peripherals→Embedded Processor→Nios II Processor** as shown in Figure 1-8. Alternatively, you can search for different components.



| Use | Conn... | Name | Description | Export | Clock | Base | End |
|---|---|---|---|---|---|---|---|
| ☑ | | ⊟ 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* | | | |

Figure 1-8 Adding Nios Processor to System



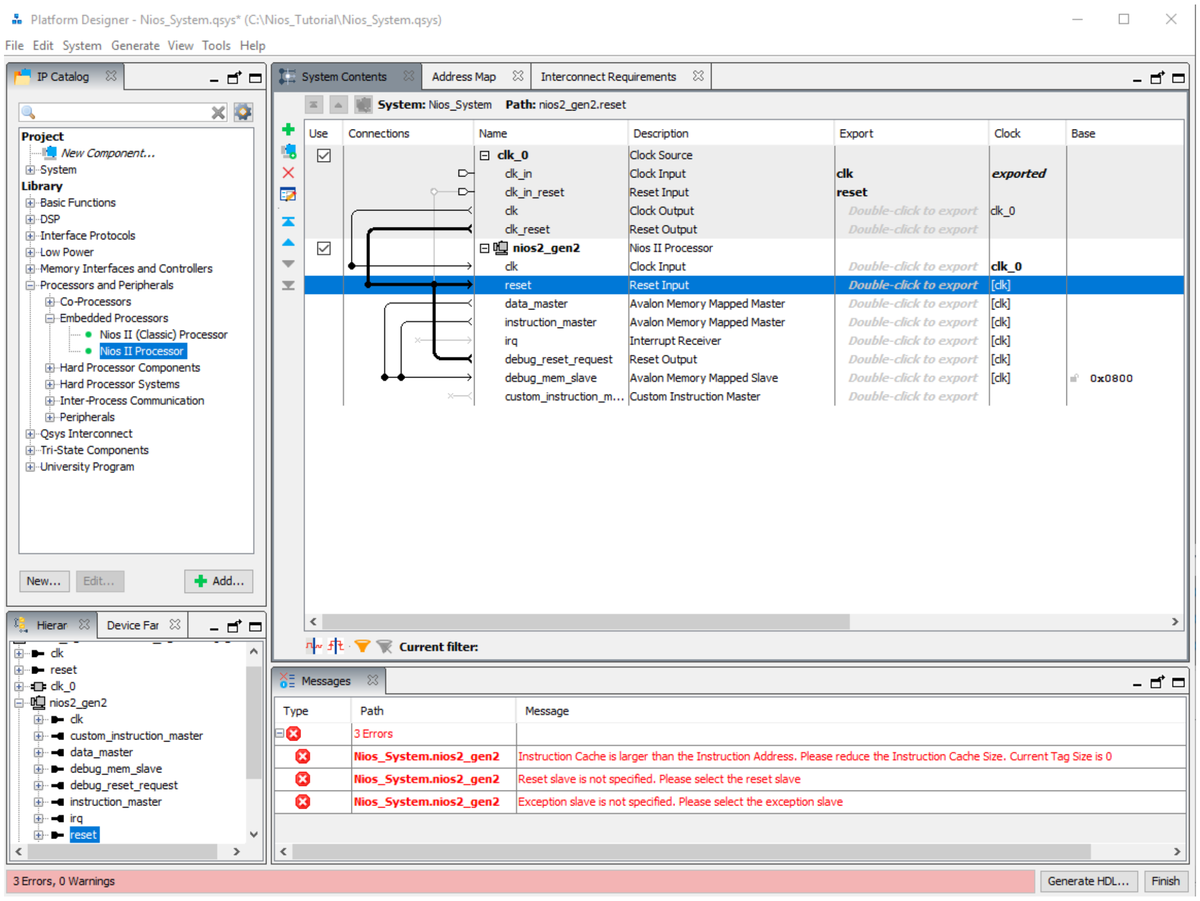
Nios II Core: Nios II/e [✓ Nios II/f]

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

```
Error: nios2_gen2_0: Instruction Cache is larger than the Instruction Address. Please reduce the Instruction Cache Size. Current Tag Size is 0
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
```

Figure 1-9 We choose Nios II/F and finish to add processor to system.

7.) You can rename system components if you would like, but that won't have any impact on the system or its function. Now we will connect the Nios processor to the clk. We will **connect clk and reset** as shown in figure 1-10. These are connected by clicking the hollow dots. When the dots become solid, it means there is a connection.


Platform Designer - Nios_System.qsys* (C:\Nios_Tutorial\Nios_System.qsys)

File Edit System Generate View Tools Help

IP Catalog

Project
- New Component...
- System

Library
- Basic Functions
- DSP
- Interface Protocols
- Low Power
- Memory Interfaces and Controllers
- Processors and Peripherals
  - Co-Processors
  - Embedded Processors
    - Nios II (Classic) Processor
    - Nios II Processor
  - Hard Processor Components
  - Hard Processor Systems
  - Inter-Process Communication
  - Peripherals
- Qsys Interconnect
- Tri-State Components
- University Program

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

System Contents | Address Map | Interconnect Requirements

**System:** Nios_System **Path:** nios2_gen2.reset

| Use | Connections | Name | Description | Export | Clock | Base |
|---|---|---|---|---|---|---|
| ✓ | | 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 | | |
| ✓ | | nios2_gen2 | Nios II Processor | | | |
| | | clk | Clock Input | Double-click to export | clk_0 | |
| | | 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] | |
| | | debug_reset_request | Reset Output | Double-click to export | [clk] | |
| | | debug_mem_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0800 |
| | | custom_instruction_m... | Custom Instruction Master | Double-click to export | | |

Current filter:

Hierar | Device Far

- clk
- reset
- clk_0
- nios2_gen2
  - clk
  - custom_instruction_master
  - data_master
  - debug_mem_slave
  - debug_reset_request
  - instruction_master
  - irq
  - reset

Messages

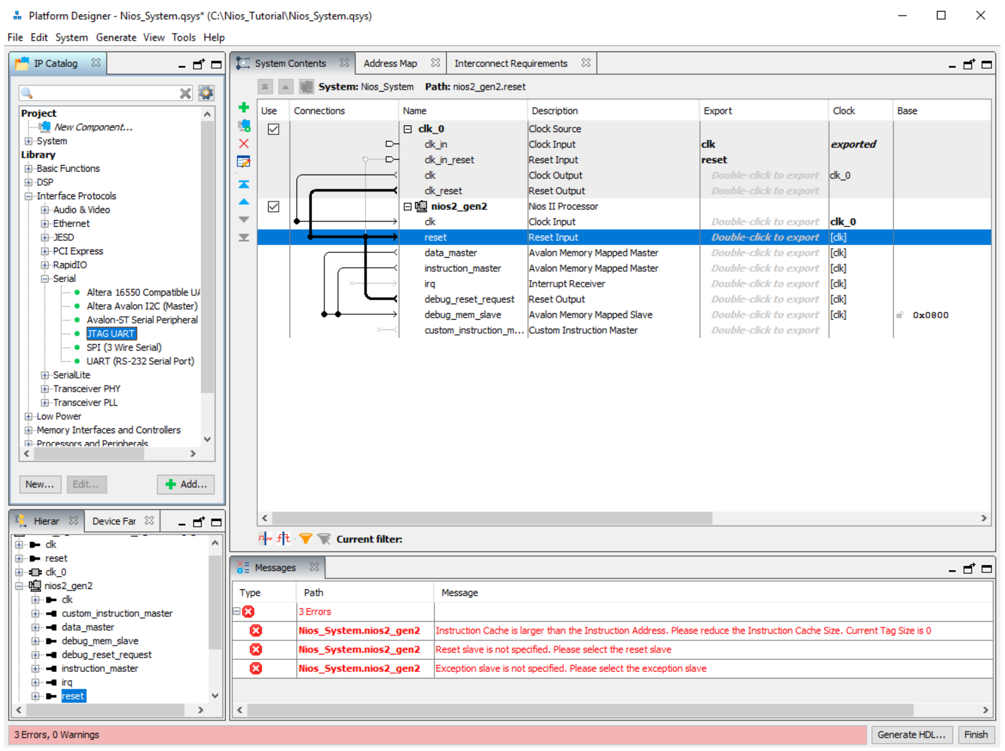
| Type | Path | Message |
|---|---|---|
| | 3 Errors | |
| | Nios_System.nios2_gen2 | Instruction Cache is larger than the Instruction Address. Please reduce the Instruction Cache Size. Current Tag Size is 0 |
| | Nios_System.nios2_gen2 | Reset slave is not specified. Please select the reset slave |
| | Nios_System.nios2_gen2 | Exception slave is not specified. Please select the exception slave |

3 Errors, 0 Warnings

Generate HDL... Finish

**Figure 1-10 Connecting clk and reset**

8.) Next thing we will add to our system is the **JTAG UART**. To do this go to **Library → Interface Protocols → Serial → JTAG UART** as shown in Figure 1-11 and 1-12.



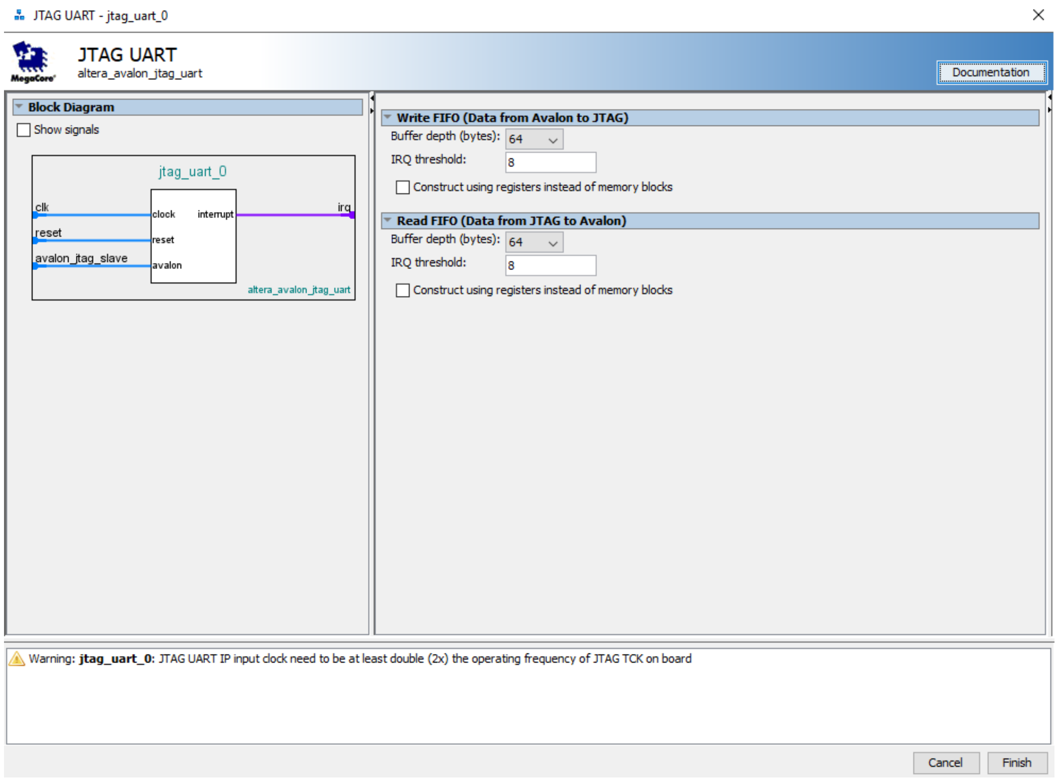
**System:** Nios_System **Path:** nios2_gen2.reset

| Use | Connections | Name | Description | Export | Clock | Base |
|---|---|---|---|---|---|---|
| ✓ | | 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* | | |
| ✓ | | nios2_gen2 | Nios II Processor | | | |
| | | clk | Clock Input | *Double-click to export* | clk_0 | |
| | | 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] | |
| | | debug_reset_request | Reset Output | *Double-click to export* | [clk] | |
| | | debug_mem_slave | Avalon Memory Mapped Slave | *Double-click to export* | [clk] | 0x0800 |
| | | custom_instruction_m... | Custom Instruction Master | *Double-click to export* | | |

| Type | Path | Message |
|---|---|---|
| | 3 Errors | |
| ✖ | Nios_System.nios2_gen2 | Instruction Cache is larger than the Instruction Address. Please reduce the Instruction Cache Size. Current Tag Size is 0 |
| ✖ | Nios_System.nios2_gen2 | Reset slave is not specified. Please select the reset slave |
| ✖ | Nios_System.nios2_gen2 | Exception slave is not specified. Please select the exception slave |

3 Errors, 0 Warnings

**Figure 1-11 Adding JTAG UART to system**



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

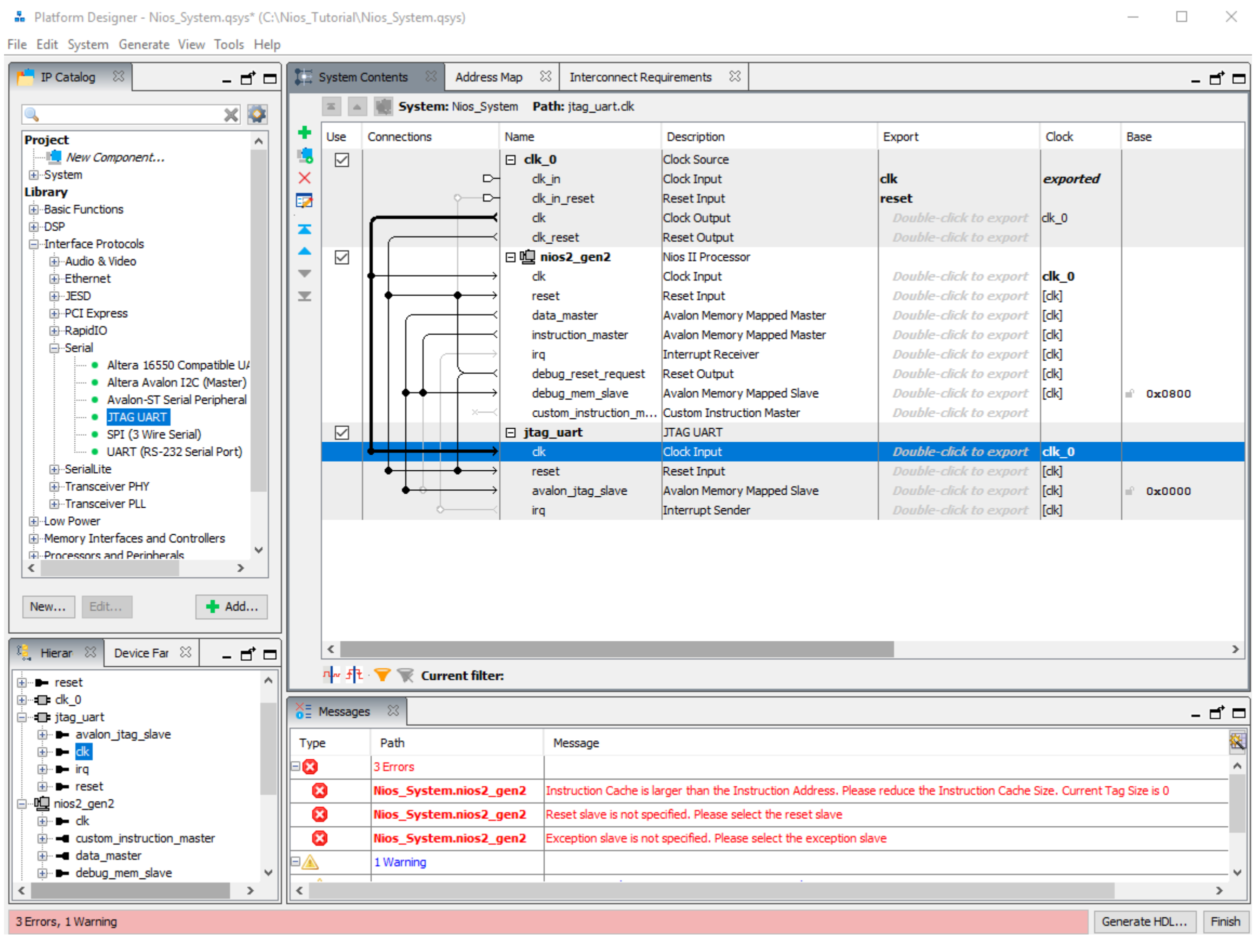
IRQ threshold: 8

☐ 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

**Figure 1-12 Adding JTAG UART.**

9.) We will now connect the clk, reset, and data master as shown in Figure 1-13.



Platform Designer - Nios_System.qsys* (C:\Nios_Tutorial\Nios_System.qsys)

System: Nios_System Path: jtag_uart.clk

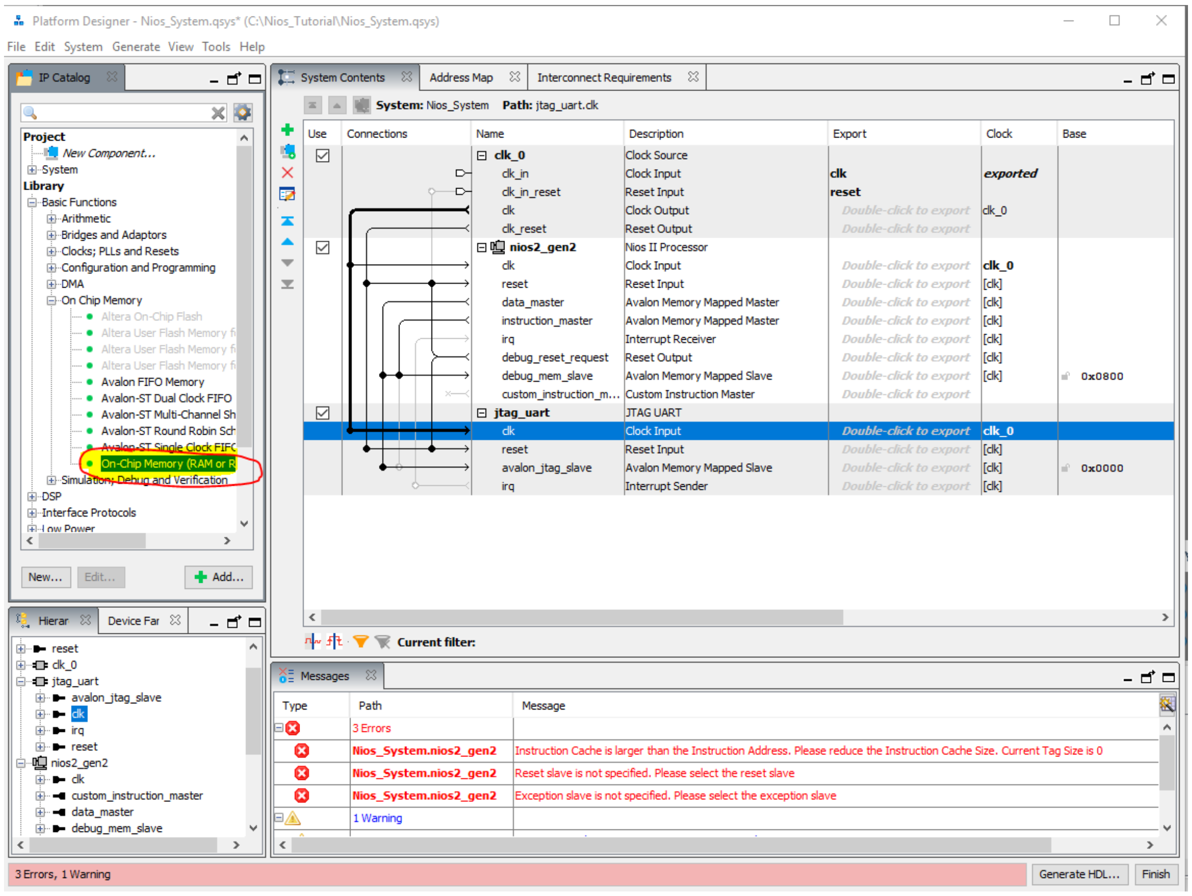
| Use | Name | Description | Export | Clock | Base |
|---|---|---|---|---|---|
| ✓ | clk_0 | Clock Source | | | |
| | clk_in | Clock Input | clk | exported | |
| | clk_in_reset | Reset Input | reset | | |
| | clk | Clock Output | Double-click to export | clk_0 | |
| | clk_reset | Reset Output | Double-click to export | | |
| ✓ | nios2_gen2 | Nios II Processor | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | 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] | |
| | debug_reset_request | Reset Output | Double-click to export | [clk] | |
| | debug_mem_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0800 |
| | custom_instruction_m... | Custom Instruction Master | Double-click to export | | |
| ✓ | jtag_uart | JTAG UART | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | avalon_jtag_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000 |
| | irq | Interrupt Sender | Double-click to export | [clk] | |

| Type | Path | Message |
|---|---|---|
| | 3 Errors | |
| | Nios_System.nios2_gen2 | Instruction Cache is larger than the Instruction Address. Please reduce the Instruction Cache Size. Current Tag Size is 0 |
| | Nios_System.nios2_gen2 | Reset slave is not specified. Please select the reset slave |
| | Nios_System.nios2_gen2 | Exception slave is not specified. Please select the exception slave |
| | 1 Warning | |

3 Errors, 1 Warning

Figure 1-13 Connecting JTAG to system

10.) Next, we will add the system memory through the on chip memory. Later we will change this to SDRAM. To add the on chip memory, choose **Library → Basic Functions → On Chip Memory → On-Chip Memory (RAM or ROM)** as shown in Figure 1-14 and 1-15.

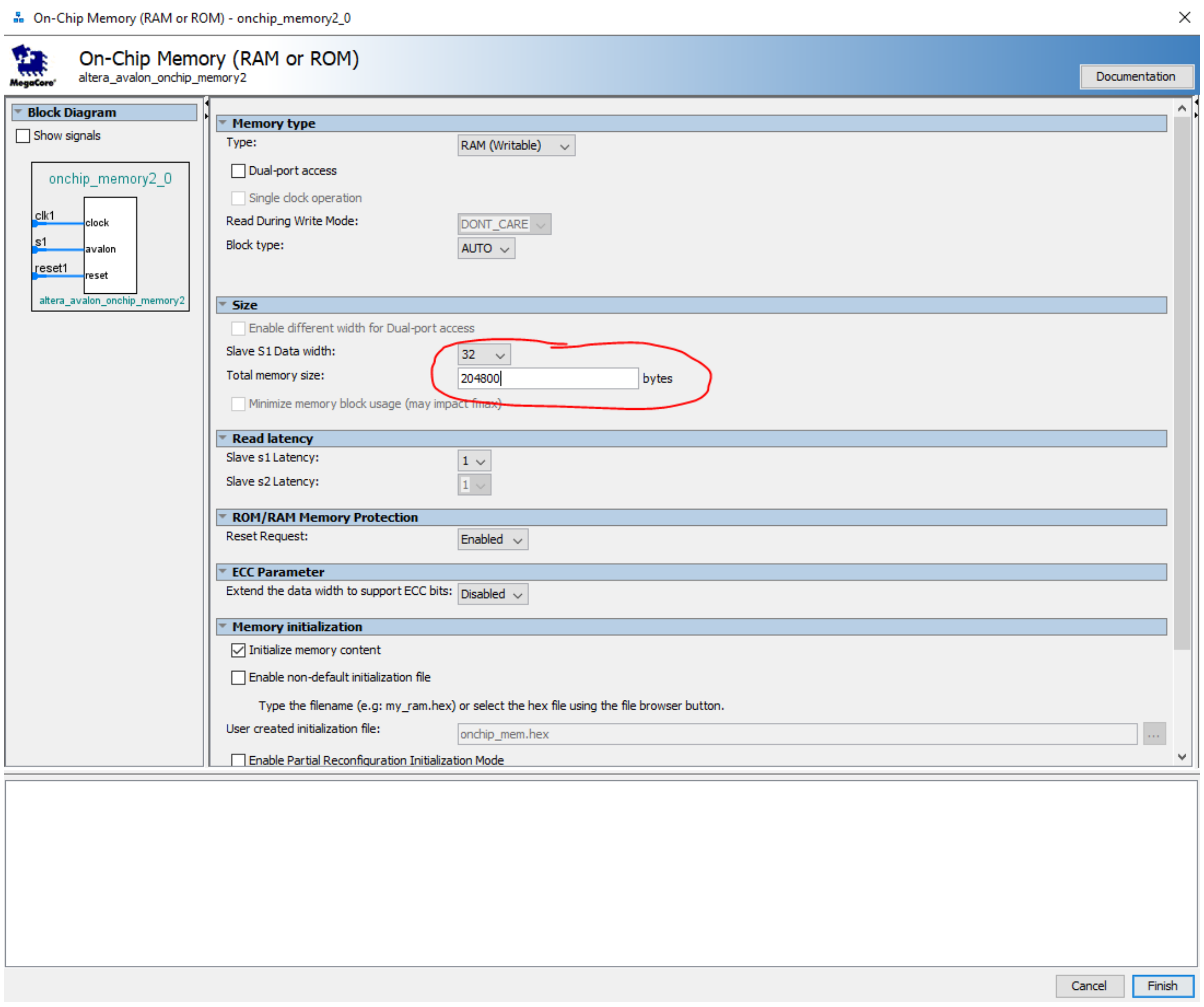

Platform Designer - Nios_System.qsys* (C:\Nios_Tutorial\Nios_System.qsys)
File Edit System Generate View Tools Help
IP Catalog
Project
New Component...
System
Library
Basic Functions
Arithmetic
Bridges and Adaptors
Clocks; PLLs and Resets
Configuration and Programming
DMA
On Chip Memory
Altera On-Chip Flash
Altera User Flash Memory f
Altera User Flash Memory f
Altera User Flash Memory f
Avalon FIFO Memory
Avalon-ST Dual Clock FIFO
Avalon-ST Multi-Channel Sh
Avalon-ST Round Robin Sch
Avalon-ST Single Clock FIFO
On-Chip Memory (RAM or R
Simulation; Debug and Verification
DSP
Interface Protocols
Low Power
New... Edit... Add...
Hierar Device Far
reset
clk_0
jtag_uart
avalon_jtag_slave
clk
irq
reset
nios2_gen2
clk
custom_instruction_master
data_master
debug_mem_slave
System Contents Address Map Interconnect Requirements
System: Nios_System Path: jtag_uart.clk
Current filter:
Messages
Generate HDL... Finish

| Use | Name | Description | Export | Clock | Base |
|---|---|---|---|---|---|
| ✓ | 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 | | |
| ✓ | nios2_gen2 | Nios II Processor | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | 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] | |
| | debug_reset_request | Reset Output | Double-click to export | [clk] | |
| | debug_mem_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0800 |
| | custom_instruction_m... | Custom Instruction Master | Double-click to export | | |
| ✓ | jtag_uart | JTAG UART | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | avalon_jtag_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000 |
| | irq | Interrupt Sender | Double-click to export | [clk] | |

| Type | Path | Message |
|---|---|---|
| | 3 Errors | |
| | Nios_System.nios2_gen2 | Instruction Cache is larger than the Instruction Address. Please reduce the Instruction Cache Size. Current Tag Size is 0 |
| | Nios_System.nios2_gen2 | Reset slave is not specified. Please select the reset slave |
| | Nios_System.nios2_gen2 | Exception slave is not specified. Please select the exception slave |
| | 1 Warning | |

3 Errors, 1 Warning

**Figure 1-14 Adding On-Chip Memory**


On-Chip Memory (RAM or ROM) - onchip_memory2_0

On-Chip Memory (RAM or ROM)
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: 204800 bytes
Minimize memory block usage (may impact fmax)

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

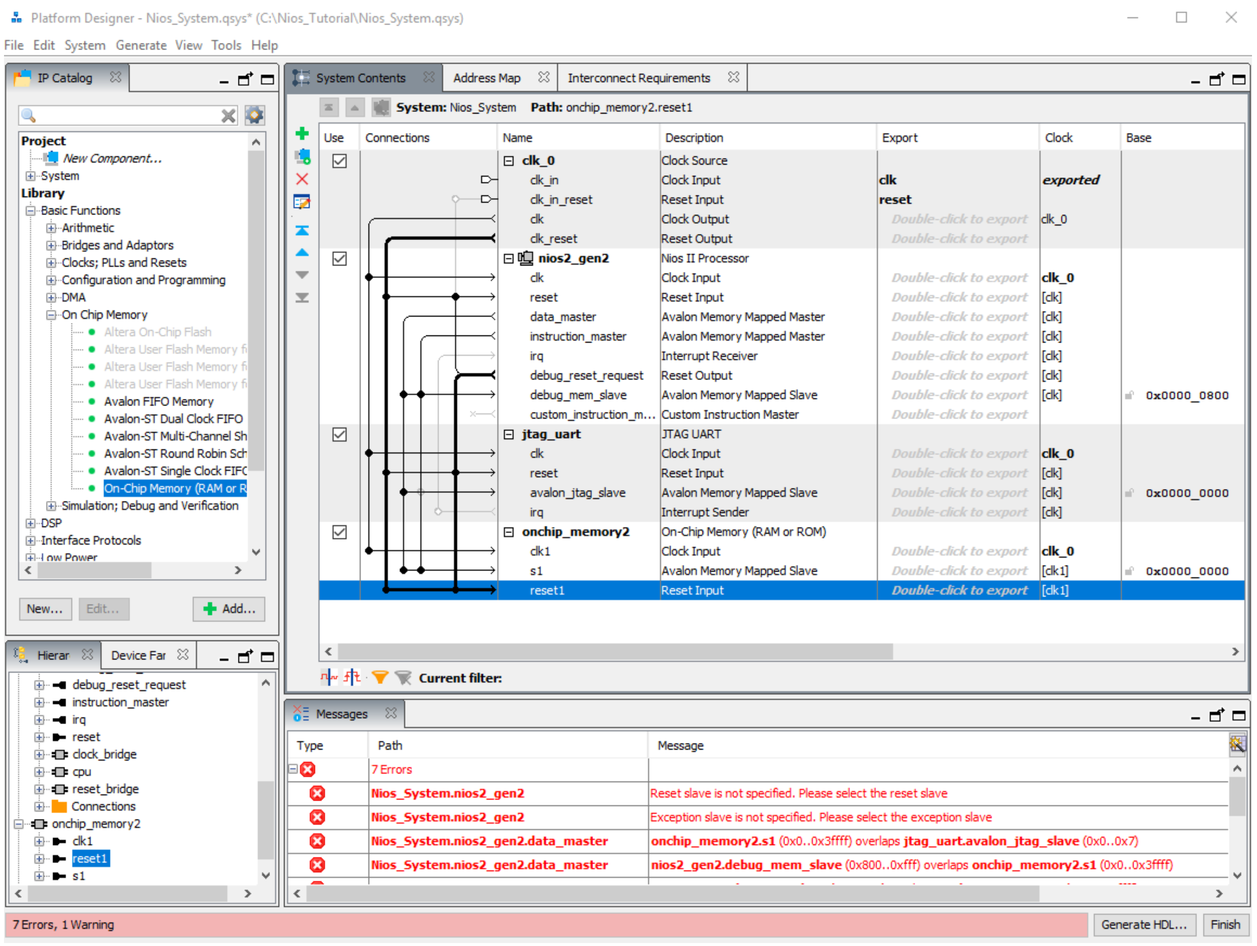
ROM/RAM Memory Protection
Reset Request: Enabled

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

Memory initialization
[✓] Initialize memory content
Enable non-default initialization file
Type the filename (e.g: my_ram.hex) or select the hex file using the file browser button.
User created initialization file: onchip_mem.hex
Enable Partial Reconfiguration Initialization Mode

**Figure 1-15 Change memory size to 204800 and select finish**

11.) Now we will connect the memory to the rest of the system by selecting the clk, reset, data master, and instruction master hollow dots as shown in Figure 1-16.



Platform Designer - Nios_System.qsys* (C:\Nios_Tutorial\Nios_System.qsys)

System: Nios_System Path: onchip_memory2.reset1

| Use | Name | Description | Export | Clock | Base |
|---|---|---|---|---|---|
| ☑ | 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 | | |
| ☑ | nios2_gen2 | Nios II Processor | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | 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] | |
| | debug_reset_request | Reset Output | Double-click to export | [clk] | |
| | debug_mem_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0800 |
| | custom_instruction_m... | Custom Instruction Master | Double-click to export | | |
| ☑ | jtag_uart | JTAG UART | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | avalon_jtag_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0000 |
| | irq | Interrupt Sender | Double-click to export | [clk] | |
| ☑ | onchip_memory2 | On-Chip Memory (RAM or ROM) | | | |
| | clk1 | Clock Input | Double-click to export | clk_0 | |
| | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk1] | 0x0000_0000 |
| | reset1 | Reset Input | Double-click to export | [clk1] | |

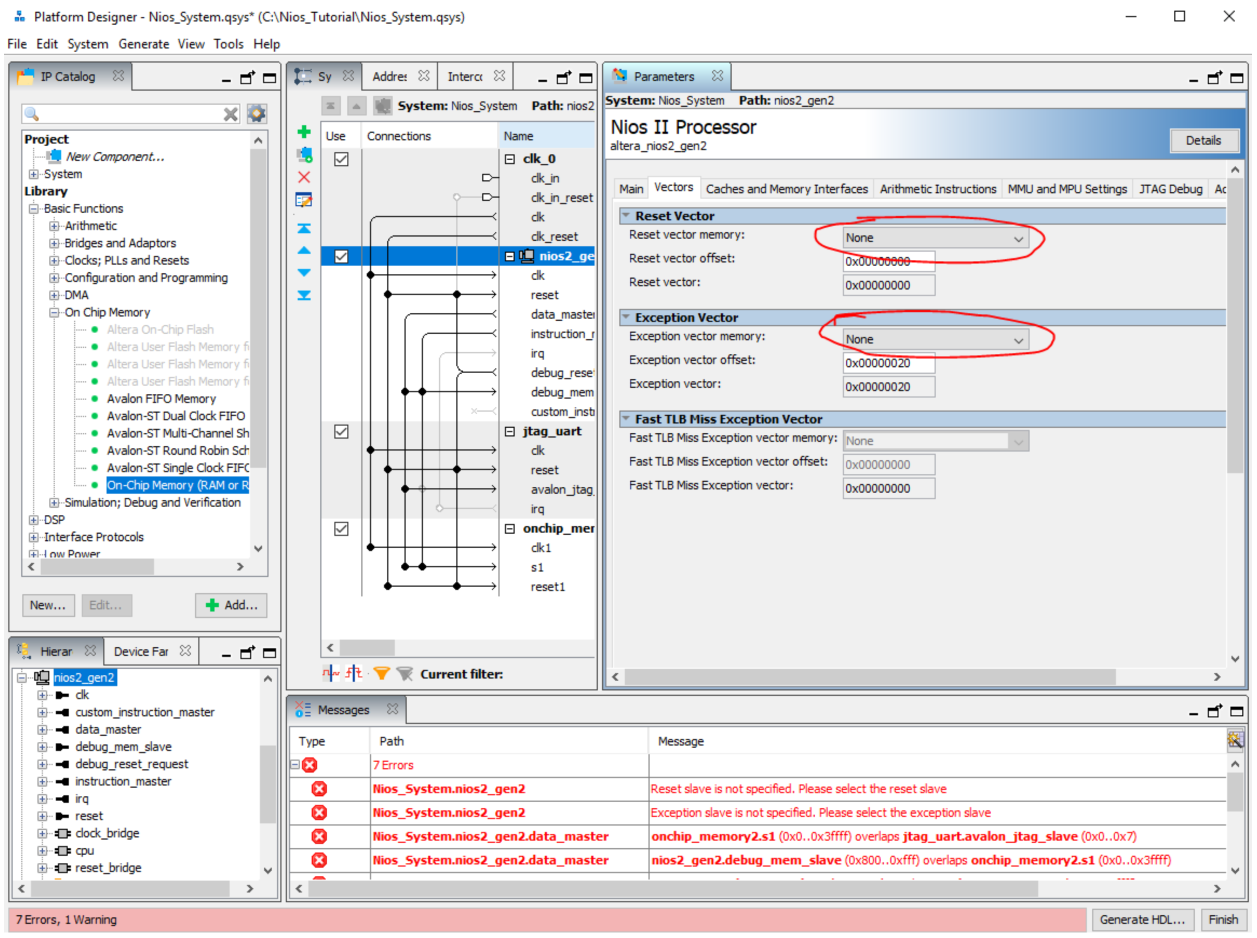
Messages

| Type | Path | Message |
|---|---|---|
| | 7 Errors | |
| | Nios_System.nios2_gen2 | Reset slave is not specified. Please select the reset slave |
| | Nios_System.nios2_gen2 | Exception slave is not specified. Please select the exception slave |
| | Nios_System.nios2_gen2.data_master | onchip_memory2.s1 (0x0..0x3ffff) overlaps jtag_uart.avalon_jtag_slave (0x0..0x7) |
| | Nios_System.nios2_gen2.data_master | nios2_gen2.debug_mem_slave (0x800..0xfff) overlaps onchip_memory2.s1 (0x0..0x3ffff) |

7 Errors, 1 Warning

**Figure 1-16 Connecting memory to system**

12.) We will now connect the Nios CPU to the memory. To do this we will click on the **Nios2_gen2** component to open its settings. Then we will go to **Vectors** and change the **Reset Vector** and **Exception Vector** to **the On-Chip memory** as shown in Figure 1-17 and 1-18.



Platform Designer - Nios_System.qsys* (C:\Nios_Tutorial\Nios_System.qsys)

System: Nios_System  Path: nios2_gen2

Nios II Processor
altera_nios2_gen2

Main | Vectors | Caches and Memory Interfaces | Arithmetic Instructions | MMU and MPU Settings | JTAG Debug

**Reset Vector**

| | |
|---|---|
| Reset vector memory: | None |
| Reset vector offset: | 0x00000000 |
| Reset vector: | 0x00000000 |

**Exception Vector**

| | |
|---|---|
| Exception vector memory: | None |
| Exception vector offset: | 0x00000020 |
| Exception vector: | 0x00000020 |

**Fast TLB Miss Exception Vector**

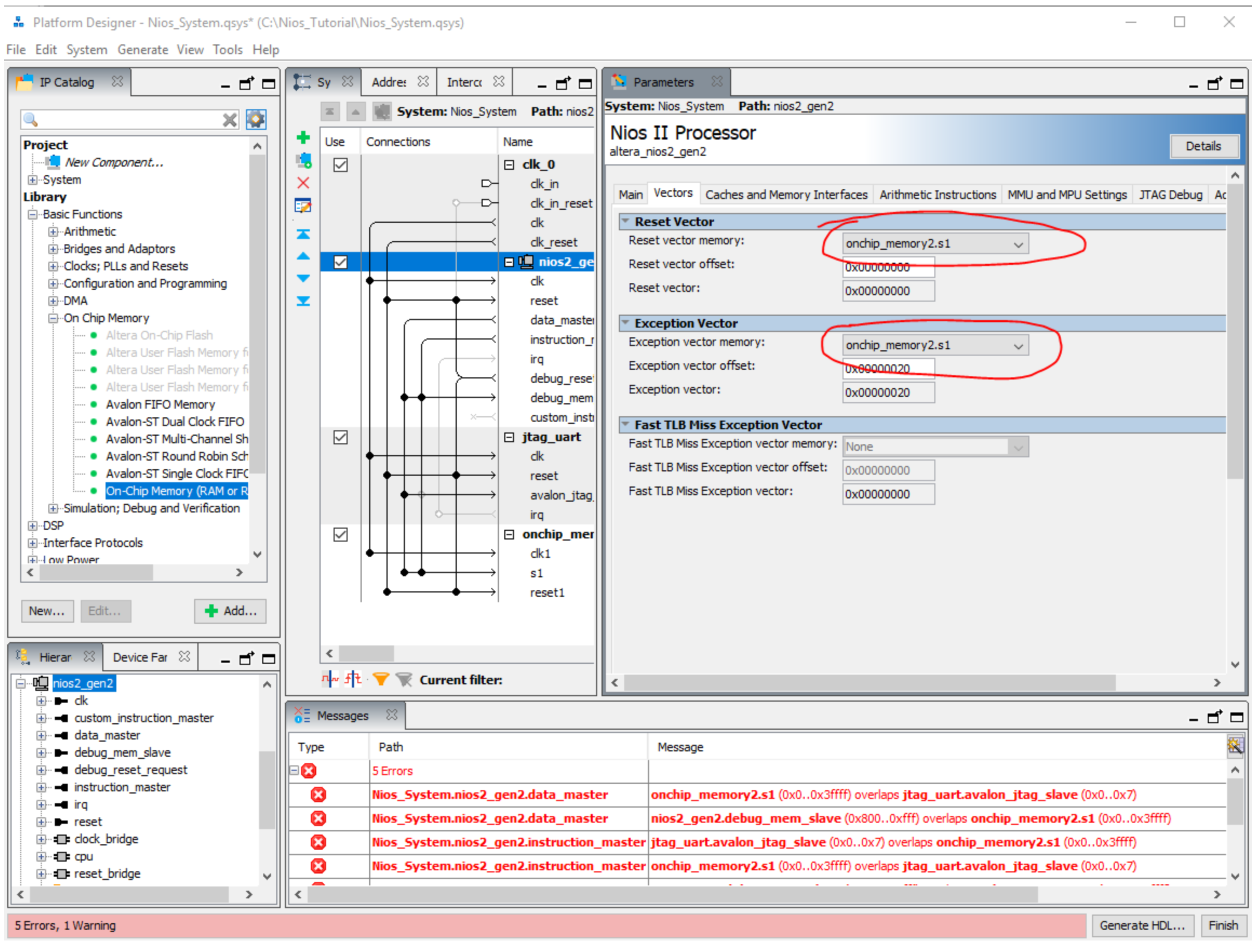
| | |
|---|---|
| Fast TLB Miss Exception vector memory: | None |
| Fast TLB Miss Exception vector offset: | 0x00000000 |
| Fast TLB Miss Exception vector: | 0x00000000 |

Messages

| Type | Path | Message |
|---|---|---|
| | 7 Errors | |
| | Nios_System.nios2_gen2 | Reset slave is not specified. Please select the reset slave |
| | Nios_System.nios2_gen2 | Exception slave is not specified. Please select the exception slave |
| | Nios_System.nios2_gen2.data_master | onchip_memory2.s1 (0x0..0x3ffff) overlaps jtag_uart.avalon_jtag_slave (0x0..0x7) |
| | Nios_System.nios2_gen2.data_master | nios2_gen2.debug_mem_slave (0x800..0xfff) overlaps onchip_memory2.s1 (0x0..0x3ffff) |

7 Errors, 1 Warning

**Figure 1-17 Changing processor to use on-chip memory.**



System: Nios_System Path: nios2_gen2

Nios II Processor
altera_nios2_gen2

**Reset Vector**

| | |
|---|---|
| Reset vector memory: | onchip_memory2.s1 |
| Reset vector offset: | 0x00000000 |
| Reset vector: | 0x00000000 |

**Exception Vector**

| | |
|---|---|
| Exception vector memory: | onchip_memory2.s1 |
| Exception vector offset: | 0x00000020 |
| Exception vector: | 0x00000020 |

**Fast TLB Miss Exception Vector**

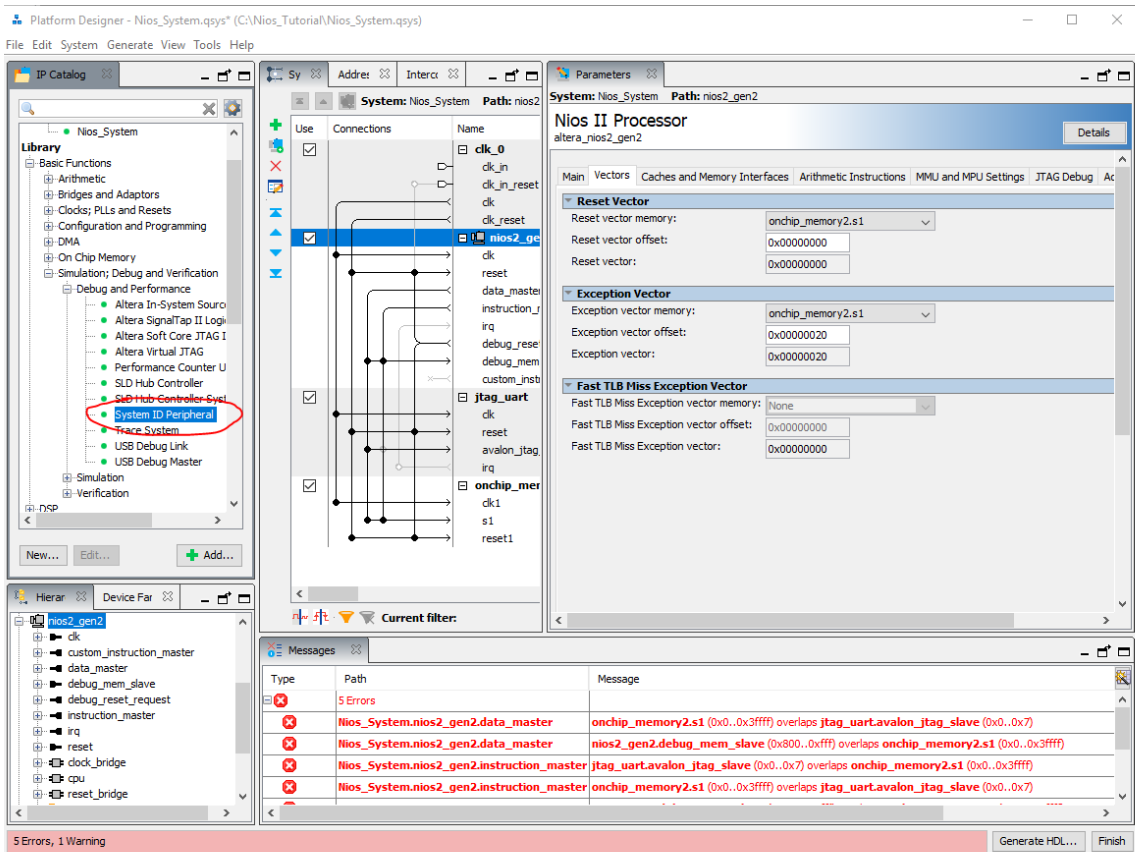
| | |
|---|---|
| Fast TLB Miss Exception vector memory: | None |
| Fast TLB Miss Exception vector offset: | 0x00000000 |
| Fast TLB Miss Exception vector: | 0x00000000 |

| Type | Path | Message |
|---|---|---|
| | 5 Errors | |
| | Nios_System.nios2_gen2.data_master | onchip_memory2.s1 (0x0..0x3ffff) overlaps jtag_uart.avalon_jtag_slave (0x0..0x7) |
| | Nios_System.nios2_gen2.data_master | nios2_gen2.debug_mem_slave (0x800..0xfff) overlaps onchip_memory2.s1 (0x0..0x3ffff) |
| | Nios_System.nios2_gen2.instruction_master | jtag_uart.avalon_jtag_slave (0x0..0x7) overlaps onchip_memory2.s1 (0x0..0x3ffff) |
| | Nios_System.nios2_gen2.instruction_master | onchip_memory2.s1 (0x0..0x3ffff) overlaps jtag_uart.avalon_jtag_slave (0x0..0x7) |

5 Errors, 1 Warning

**Figure 1-18 Changed Reset and Exception Vector**

13.) Next up is to add the System ID to the system. Choose **Library → Basic Functions → Simulation; Debug and Verification → Debug and Performance → System ID Peripheral** as shown in Figure 1-19 and 1-20.



**Reset Vector**

| | |
|---|---|
| Reset vector memory: | onchip_memory2.s1 |
| Reset vector offset: | 0x00000000 |
| Reset vector: | 0x00000000 |

**Exception Vector**

| | |
|---|---|
| Exception vector memory: | onchip_memory2.s1 |
| Exception vector offset: | 0x00000020 |
| Exception vector: | 0x00000020 |

**Fast TLB Miss Exception Vector**

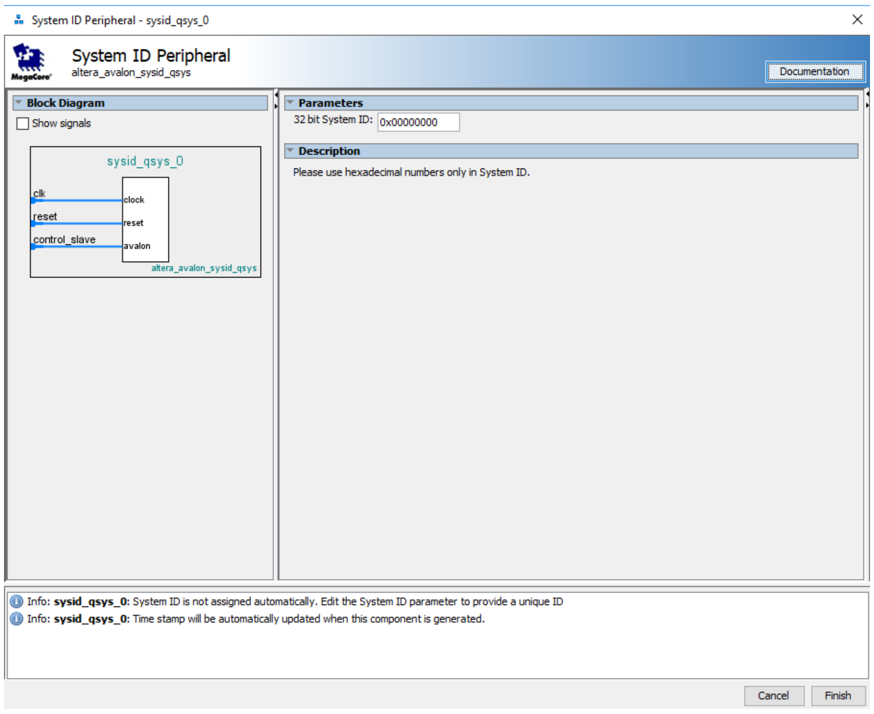
| | |
|---|---|
| Fast TLB Miss Exception vector memory: | None |
| Fast TLB Miss Exception vector offset: | 0x00000000 |
| Fast TLB Miss Exception vector: | 0x00000000 |

Messages

| Type | Path | Message |
|---|---|---|
| | 5 Errors | |
| | Nios_System.nios2_gen2.data_master | onchip_memory2.s1 (0x0..0x3ffff) overlaps jtag_uart.avalon_jtag_slave (0x0..0x7) |
| | Nios_System.nios2_gen2.data_master | nios2_gen2.debug_mem_slave (0x800..0xfff) overlaps onchip_memory2.s1 (0x0..0x3ffff) |
| | Nios_System.nios2_gen2.instruction_master | jtag_uart.avalon_jtag_slave (0x0..0x7) overlaps onchip_memory2.s1 (0x0..0x3ffff) |
| | Nios_System.nios2_gen2.instruction_master | onchip_memory2.s1 (0x0..0x3ffff) overlaps jtag_uart.avalon_jtag_slave (0x0..0x7) |

5 Errors, 1 Warning

**Figure 1-19 Adding system ID**



**Parameters**

32 bit System ID: 0x00000000

**Description**

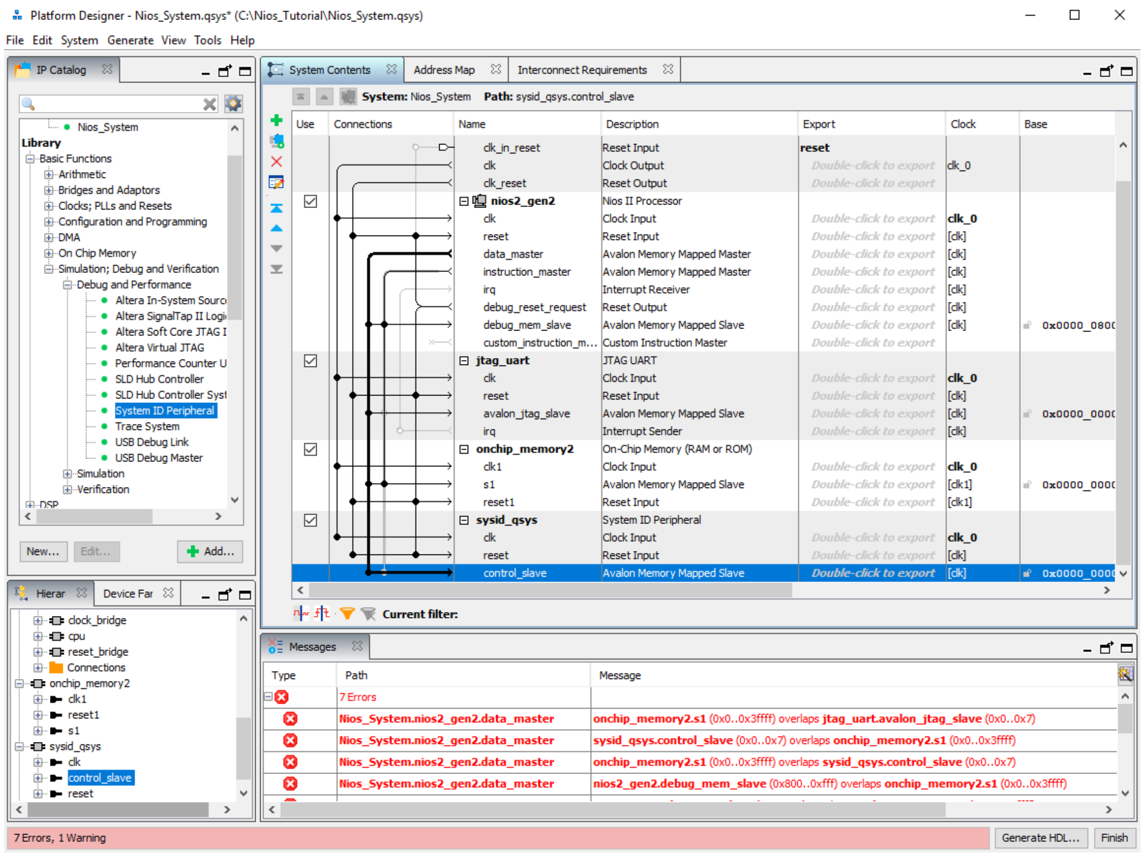
Please use hexadecimal numbers only in System ID.

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.

**Figure 1-20**

14.) Connect the clk, reset, and data master dots as shown in figure 1-21.



Platform Designer - Nios_System.qsys* (C:\Nios_Tutorial\Nios_System.qsys)

System: Nios_System Path: sysid_qsys.control_slave

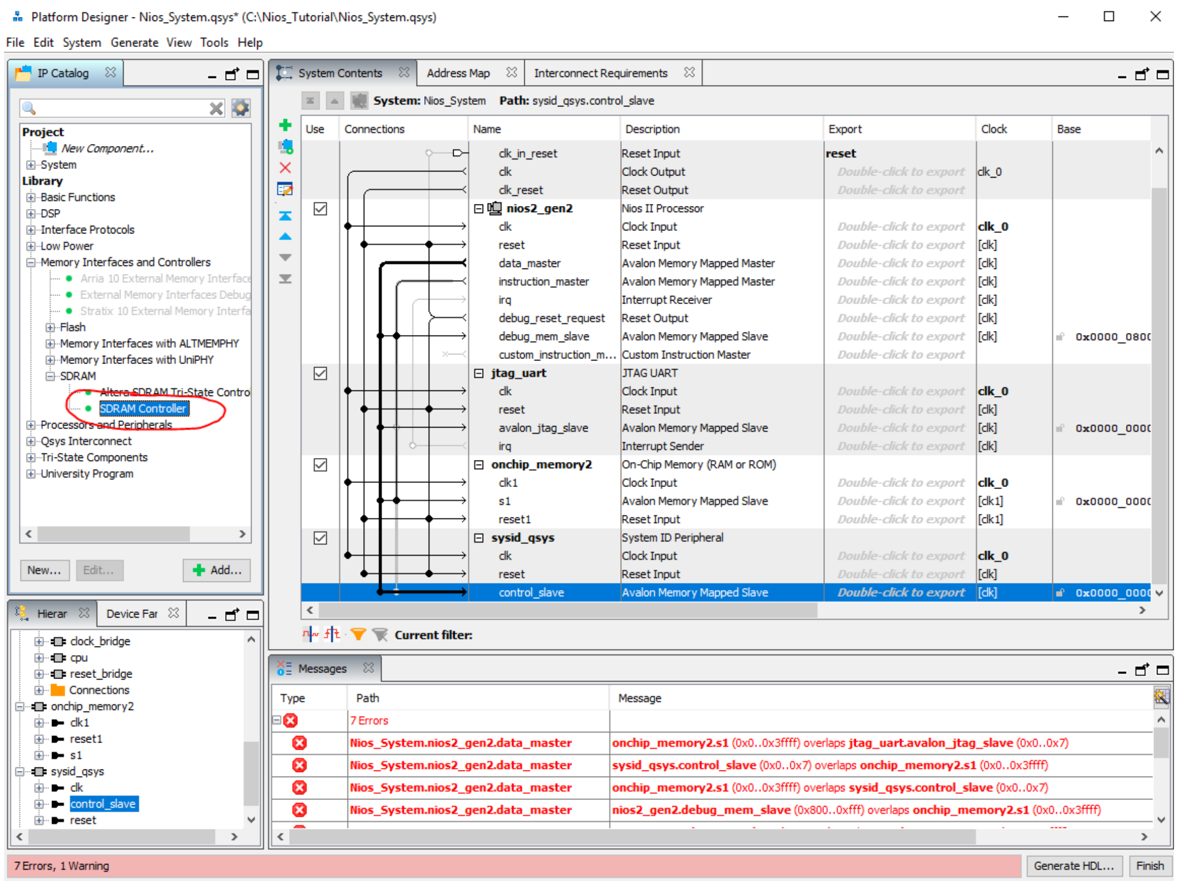
| Use | Name | Description | Export | Clock | Base |
|---|---|---|---|---|---|
| | clk_in_reset | Reset Input | reset | | |
| | clk | Clock Output | Double-click to export | clk_0 | |
| | clk_reset | Reset Output | Double-click to export | | |
| ✓ | nios2_gen2 | Nios II Processor | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | 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] | |
| | debug_reset_request | Reset Output | Double-click to export | [clk] | |
| | debug_mem_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_080 |
| | custom_instruction_m… | Custom Instruction Master | Double-click to export | | |
| ✓ | jtag_uart | JTAG UART | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | avalon_jtag_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_000 |
| | irq | Interrupt Sender | Double-click to export | [clk] | |
| ✓ | onchip_memory2 | On-Chip Memory (RAM or ROM) | | | |
| | clk1 | Clock Input | Double-click to export | clk_0 | |
| | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk1] | 0x0000_000 |
| | reset1 | Reset Input | Double-click to export | [clk1] | |
| ✓ | sysid_qsys | System ID Peripheral | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | control_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_000 |

| Type | Path | Message |
|---|---|---|
| ⊗ | 7 Errors | |
| ⊗ | Nios_System.nios2_gen2.data_master | onchip_memory2.s1 (0x0..0x3ffff) overlaps jtag_uart.avalon_jtag_slave (0x0..0x7) |
| ⊗ | Nios_System.nios2_gen2.data_master | sysid_qsys.control_slave (0x0..0x7) overlaps onchip_memory2.s1 (0x0..0x3ffff) |
| ⊗ | Nios_System.nios2_gen2.data_master | onchip_memory2.s1 (0x0..0x3ffff) overlaps sysid_qsys.control_slave (0x0..0x7) |
| ⊗ | Nios_System.nios2_gen2.data_master | nios2_gen2.debug_mem_slave (0x800..0xfff) overlaps onchip_memory2.s1 (0x0..0x3ffff) |

7 Errors, 1 Warning

**Figure 1-21 Connecting System ID connections**

15.) Now we will add the SDRAM to the system which will be used instead of the on-chip memory. To do this choose **Library → Memory Interfaces and Controlelrs → SDRAM → Controller** as shown in Figure 1-22 and 1-23.



Platform Designer - Nios_System.qsys* (C:\Nios_Tutorial\Nios_System.qsys)

File Edit System Generate View Tools Help

IP Catalog

Project
- New Component...
- System

Library
- Basic Functions
- DSP
- Interface Protocols
- Low Power
- Memory Interfaces and Controllers
  - Arria 10 External Memory Interface
  - External Memory Interfaces Debug
  - Stratix 10 External Memory Interfa
  - Flash
  - Memory Interfaces with ALTMEMPHY
  - Memory Interfaces with UniPHY
  - SDRAM
    - Altera SDRAM Tri-State Contro
    - SDRAM Controller
- Processors and Peripherals
- Qsys Interconnect
- Tri-State Components
- University Program

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

System Contents | Address Map | Interconnect Requirements

**System:** Nios_System **Path:** sysid_qsys.control_slave

| Use | Connections | Name | Description | Export | Clock | Base |
|---|---|---|---|---|---|---|
| | | clk_in_reset | Reset Input | **reset** | | |
| | | clk | Clock Output | Double-click to export | clk_0 | |
| | | clk_reset | Reset Output | Double-click to export | | |
| ✓ | | **nios2_gen2** | Nios II Processor | | | |
| | | clk | Clock Input | Double-click to export | **clk_0** | |
| | | 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] | |
| | | debug_reset_request | Reset Output | Double-click to export | [clk] | |
| | | debug_mem_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_080 |
| | | custom_instruction_m... | Custom Instruction Master | Double-click to export | | |
| ✓ | | **jtag_uart** | JTAG UART | | | |
| | | clk | Clock Input | Double-click to export | **clk_0** | |
| | | reset | Reset Input | Double-click to export | [clk] | |
| | | avalon_jtag_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_000 |
| | | irq | Interrupt Sender | Double-click to export | [clk] | |
| ✓ | | **onchip_memory2** | On-Chip Memory (RAM or ROM) | | | |
| | | clk1 | Clock Input | Double-click to export | **clk_0** | |
| | | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk1] | 0x0000_000 |
| | | reset1 | Reset Input | Double-click to export | [clk1] | |
| ✓ | | **sysid_qsys** | System ID Peripheral | | | |
| | | clk | Clock Input | Double-click to export | **clk_0** | |
| | | reset | Reset Input | Double-click to export | [clk] | |
| | | control_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_000 |

Current filter:

Hierar | Device Far

- clock_bridge
- cpu
- reset_bridge
- Connections
- onchip_memory2
  - clk1
  - reset1
  - s1
- sysid_qsys
  - clk
  - control_slave
  - reset

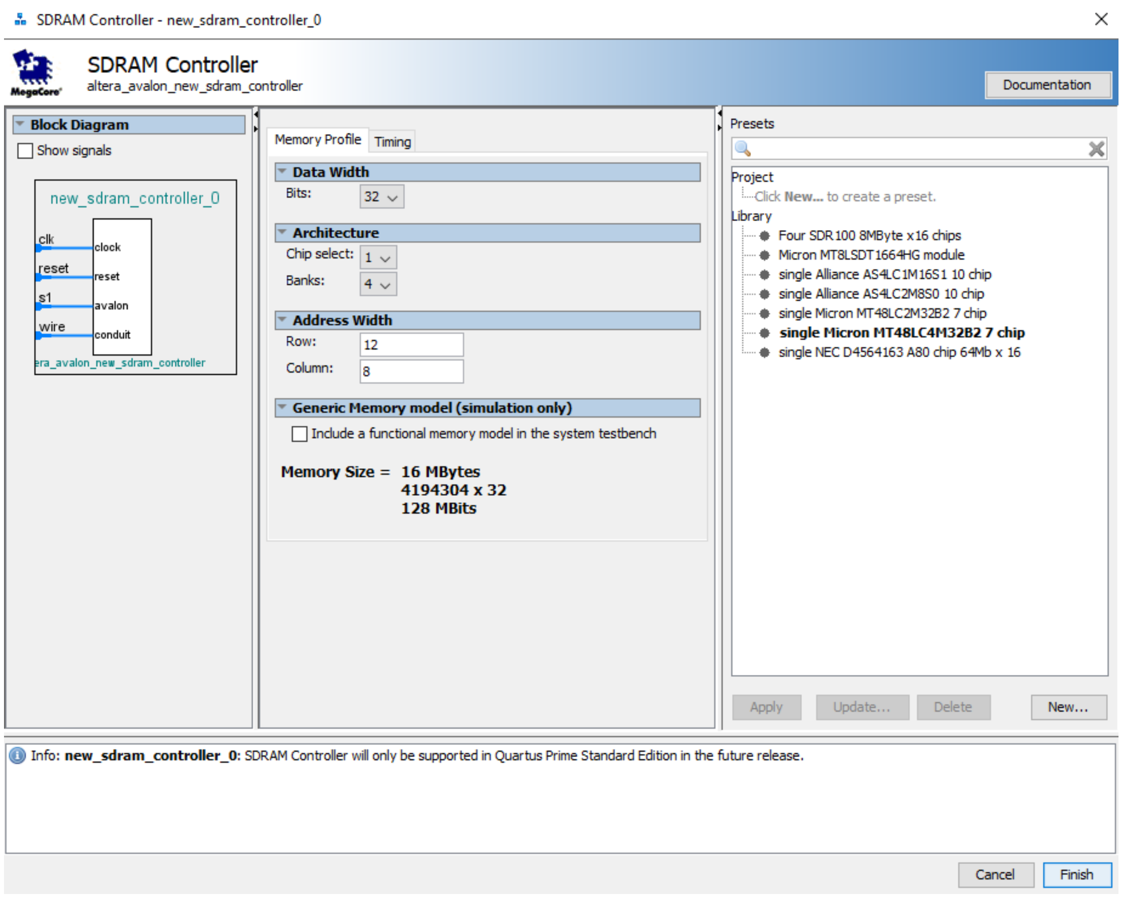
Messages

| Type | Path | Message |
|---|---|---|
| | 7 Errors | |
| ✖ | Nios_System.nios2_gen2.data_master | **onchip_memory2.s1** (0x0..0x3ffff) overlaps **jtag_uart.avalon_jtag_slave** (0x0..0x7) |
| ✖ | Nios_System.nios2_gen2.data_master | **sysid_qsys.control_slave** (0x0..0x7) overlaps **onchip_memory2.s1** (0x0..0x3ffff) |
| ✖ | Nios_System.nios2_gen2.data_master | **onchip_memory2.s1** (0x0..0x3ffff) overlaps **sysid_qsys.control_slave** (0x0..0x7) |
| ✖ | Nios_System.nios2_gen2.data_master | **nios2_gen2.debug_mem_slave** (0x800..0xfff) overlaps **onchip_memory2.s1** (0x0..0x3ffff) |

7 Errors, 1 Warning

Generate HDL... Finish

**Figure 1-22 Adding SDRAM to System**


SDRAM Controller - new_sdram_controller_0

SDRAM Controller
altera_avalon_new_sdram_controller

Memory Profile | Timing

**Data Width**
Bits: 32

**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

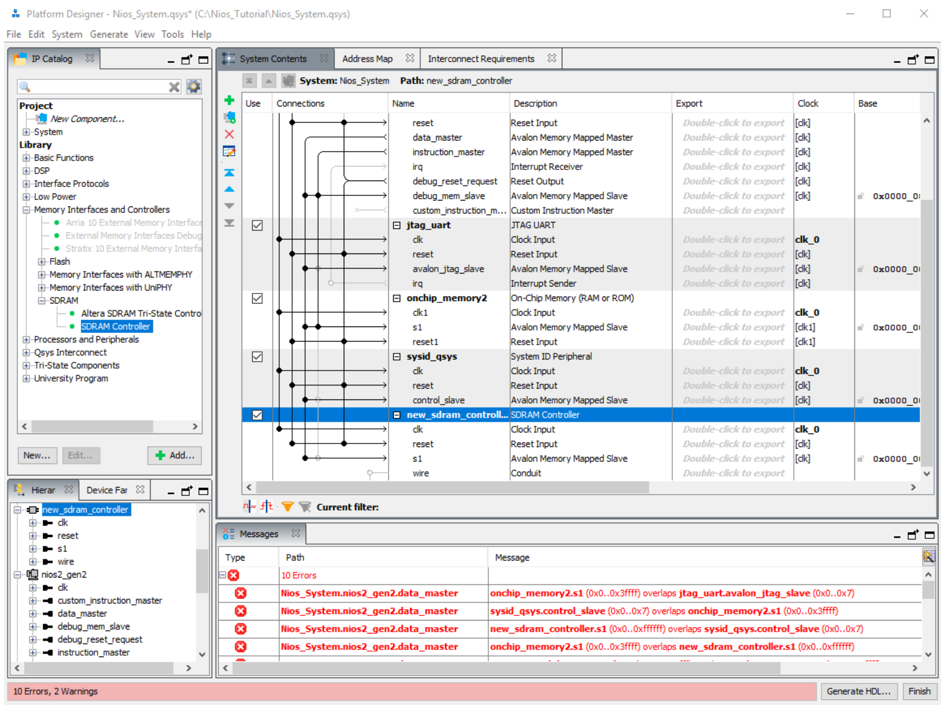
**Memory Size = 16 MBytes**
**4194304 x 32**
**128 MBits**

Presets
Project
Click **New...** to create a preset.
Library
- Four SDR100 8MByte x16 chips
- Micron MT8LSDT1664HG module
- single Alliance AS4LC1M16S1 10 chip
- single Alliance AS4LC2M8S0 10 chip
- single Micron MT48LC2M32B2 7 chip
- **single Micron MT48LC4M32B2 7 chip**
- single NEC D4564163 A80 chip 64Mb x 16

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

**Figure 1-23**

16.) Now we will connect the SDRAM to the system by connecting the clk, reset, and data master wires as shown in Figure 1-24.



Platform Designer - Nios_System.qsys* (C:\Nios_Tutorial\Nios_System.qsys)

System: Nios_System  Path: new_sdram_controller

| Use | Name | Description | Export | Clock | Base |
|---|---|---|---|---|---|
| | 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] | |
| | debug_reset_request | Reset Output | Double-click to export | [clk] | |
| | debug_mem_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| | custom_instruction_m... | Custom Instruction Master | Double-click to export | | |
| ✓ | jtag_uart | JTAG UART | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | avalon_jtag_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| | irq | Interrupt Sender | Double-click to export | [clk] | |
| ✓ | onchip_memory2 | On-Chip Memory (RAM or ROM) | | | |
| | clk1 | Clock Input | Double-click to export | clk_0 | |
| | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk1] | 0x0000_0 |
| | reset1 | Reset Input | Double-click to export | [clk1] | |
| ✓ | sysid_qsys | System ID Peripheral | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | control_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| ✓ | new_sdram_controll... | SDRAM Controller | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| | wire | Conduit | Double-click to export | | |

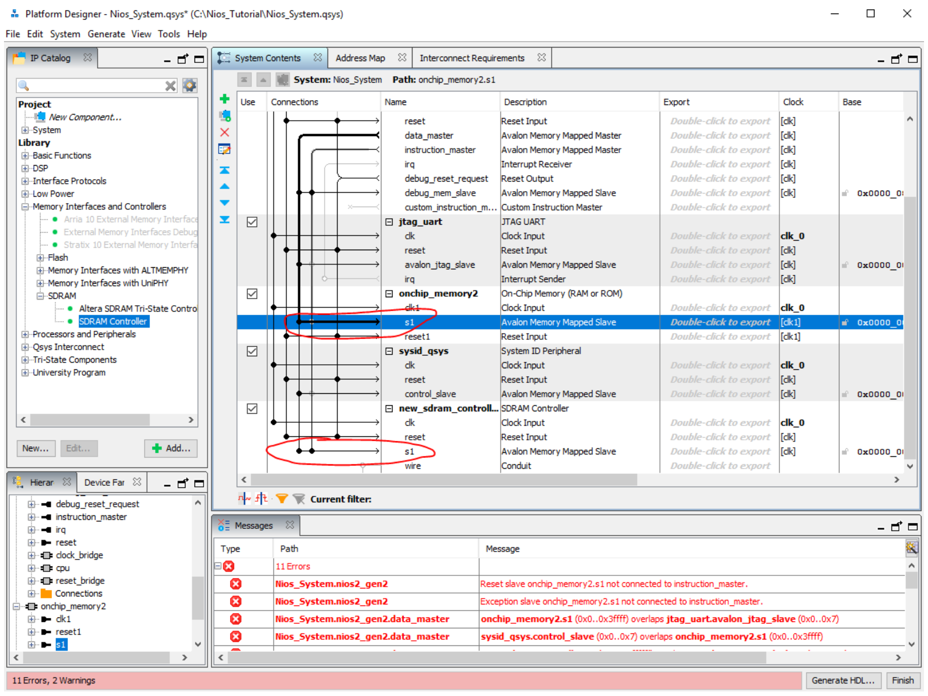
Messages

| Type | Path | Message |
|---|---|---|
| | 10 Errors | |
| ⊗ | Nios_System.nios2_gen2.data_master | **onchip_memory2.s1** (0x0..0x3ffff) overlaps **jtag_uart.avalon_jtag_slave** (0x0..0x7) |
| ⊗ | Nios_System.nios2_gen2.data_master | **sysid_qsys.control_slave** (0x0..0x7) overlaps **onchip_memory2.s1** (0x0..0x3ffff) |
| ⊗ | Nios_System.nios2_gen2.data_master | **new_sdram_controller.s1** (0x0..0xffffff) overlaps **sysid_qsys.control_slave** (0x0..0x7) |
| ⊗ | Nios_System.nios2_gen2.data_master | **onchip_memory2.s1** (0x0..0x3ffff) overlaps **new_sdram_controller.s1** (0x0..0xffffff) |

10 Errors, 2 Warnings

**Figure 1-24 Connecting SDRAM**

17.) To change the Nios system to use the SDRAM instead of the on-chip memory, we will connect the SDRAM to the instruction master and disconnect the On-Chip memory from the instruction master as shown in Figure 1-25. We will then go to the CPU setting by clicking on the Nios component and changing the Reset and Exception Vectors to the SDRAM shown in Figure 1-26.



Platform Designer - Nios_System.qsys* (C:\Nios_Tutorial\Nios_System.qsys)

System: Nios_System  Path: onchip_memory2.s1

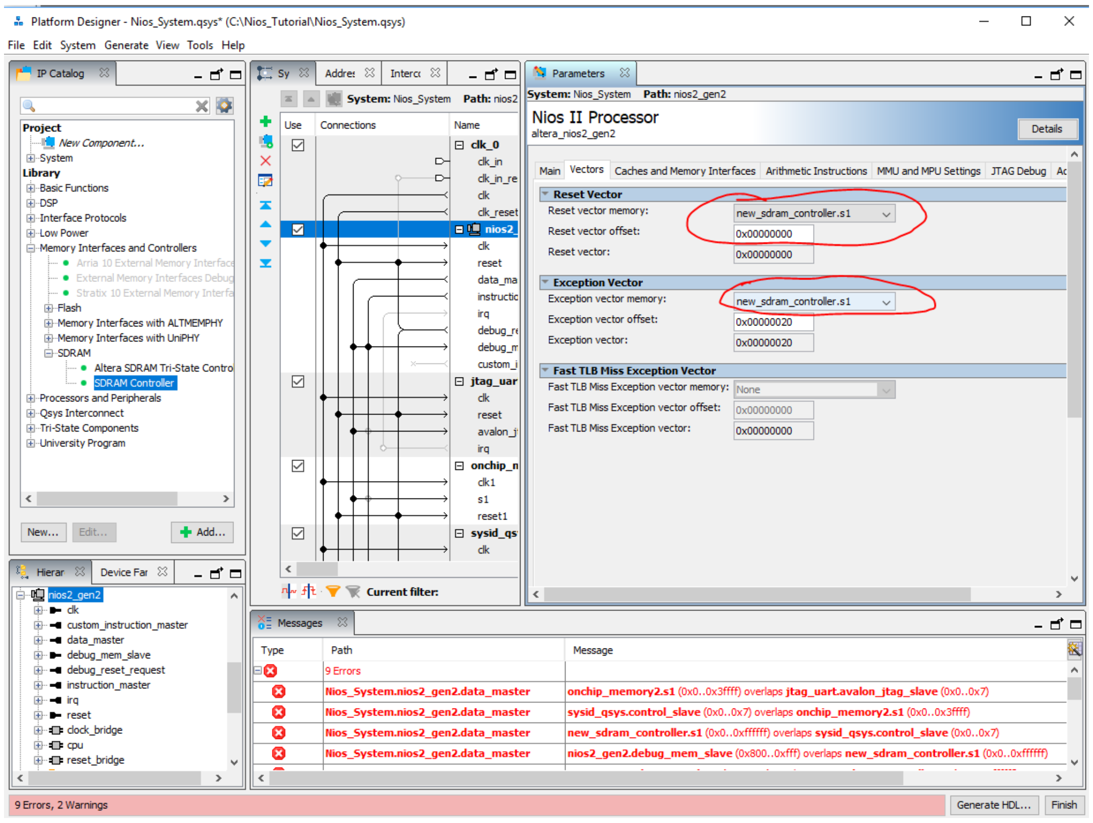
| Use | Name | Description | Export | Clock | Base |
|---|---|---|---|---|---|
| | reset | Reset Input | Double-click to export | [clk] | |
| | data_master | Avalon Memory Mapped Master | Double-click to export | [clk] | |
| | instruction_master | Avalon Memory Mapped Master | Double-click to export | [clk] | |
| | irq | Interrupt Receiver | Double-click to export | [clk] | |
| | debug_reset_request | Reset Output | Double-click to export | [clk] | |
| | debug_mem_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| | custom_instruction_m... | Custom Instruction Master | Double-click to export | | |
| ✓ | jtag_uart | JTAG UART | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | avalon_jtag_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| | irq | Interrupt Sender | Double-click to export | [clk] | |
| ✓ | onchip_memory2 | On-Chip Memory (RAM or ROM) | | | |
| | clk1 | Clock Input | Double-click to export | clk_0 | |
| | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk1] | 0x0000_0 |
| | reset1 | Reset Input | Double-click to export | [clk1] | |
| ✓ | sysid_qsys | System ID Peripheral | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | control_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| ✓ | new_sdram_controll... | SDRAM Controller | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| | wire | Conduit | Double-click to export | | |

Messages

| Type | Path | Message |
|---|---|---|
| | 11 Errors | |
| ⊗ | Nios_System.nios2_gen2 | Reset slave onchip_memory2.s1 not connected to instruction_master. |
| ⊗ | Nios_System.nios2_gen2 | Exception slave onchip_memory2.s1 not connected to instruction_master. |
| ⊗ | Nios_System.nios2_gen2.data_master | **onchip_memory2.s1** (0x0..0x3ffff) overlaps **jtag_uart.avalon_jtag_slave** (0x0..0x7) |
| ⊗ | Nios_System.nios2_gen2.data_master | **sysid_qsys.control_slave** (0x0..0x7) overlaps **onchip_memory2.s1** (0x0..0x3ffff) |

11 Errors, 2 Warnings

**Figure 1-25 Connecting SDRAM to instruction master**



Platform Designer - Nios_System.qsys* (C:\Nios_Tutorial\Nios_System.qsys)

System: Nios_System Path: nios2_gen2

Nios II Processor
altera_nios2_gen2

**Reset Vector**

| | |
|---|---|
| Reset vector memory: | new_sdram_controller.s1 |
| Reset vector offset: | 0x00000000 |
| Reset vector: | 0x00000000 |

**Exception Vector**

| | |
|---|---|
| Exception vector memory: | new_sdram_controller.s1 |
| Exception vector offset: | 0x00000020 |
| Exception vector: | 0x00000020 |

**Fast TLB Miss Exception Vector**

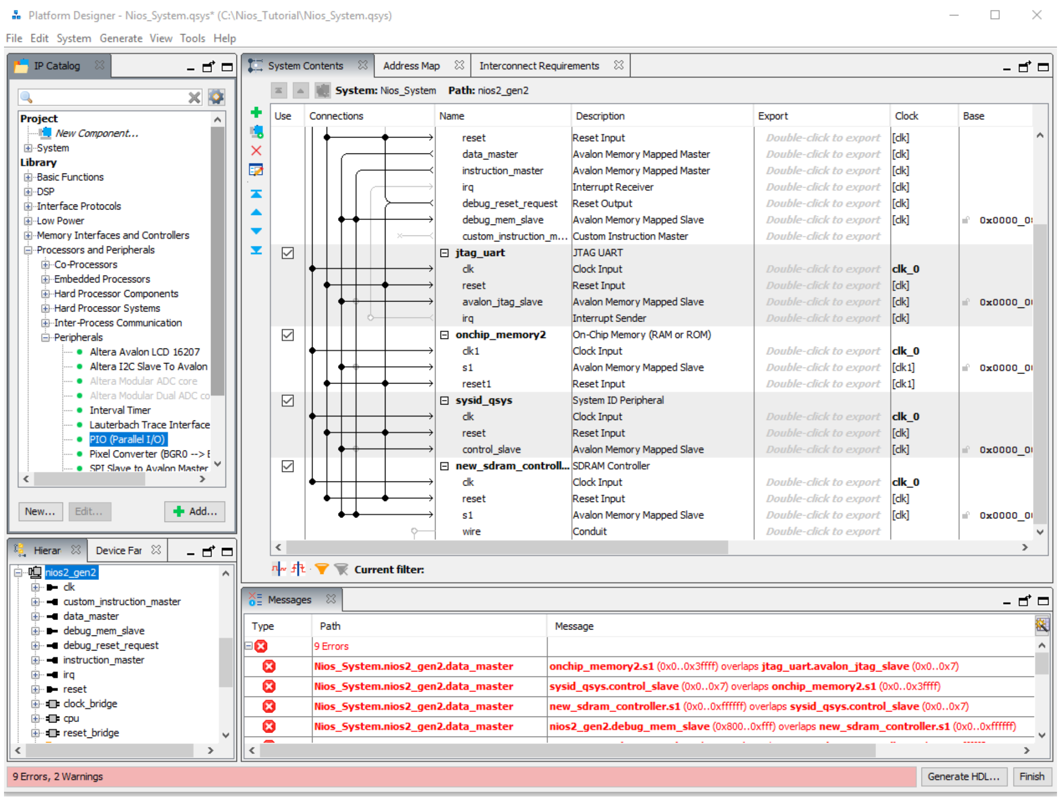
| | |
|---|---|
| Fast TLB Miss Exception vector memory: | None |
| Fast TLB Miss Exception vector offset: | 0x00000000 |
| Fast TLB Miss Exception vector: | 0x00000000 |

| Type | Path | Message |
|---|---|---|
| | 9 Errors | |
| | Nios_System.nios2_gen2.data_master | onchip_memory2.s1 (0x0..0x3ffff) overlaps jtag_uart.avalon_jtag_slave (0x0..0x7) |
| | Nios_System.nios2_gen2.data_master | sysid_qsys.control_slave (0x0..0x7) overlaps onchip_memory2.s1 (0x0..0x3ffff) |
| | Nios_System.nios2_gen2.data_master | new_sdram_controller.s1 (0x0..0xffffff) overlaps sysid_qsys.control_slave (0x0..0x7) |
| | Nios_System.nios2_gen2.data_master | nios2_gen2.debug_mem_slave (0x800..0xfff) overlaps new_sdram_controller.s1 (0x0..0xffffff) |

9 Errors, 2 Warnings

**Figure 1-26 Changing CPU to use SDRAM**

18.) We will now start connecting our PIO’s to send data to and from the CPU to the FPGA. Choose **Library → Processors and Peripherals → PIO (Parallel I/O)** as shown in figure 1-27 and 1-28. We will be using 32 bit In Out PIO’s for what we will be doing. Choose accordingly depending on your use of the PIO.



Platform Designer - Nios_System.qsys* (C:\Nios_Tutorial\Nios_System.qsys)

System: Nios_System Path: nios2_gen2

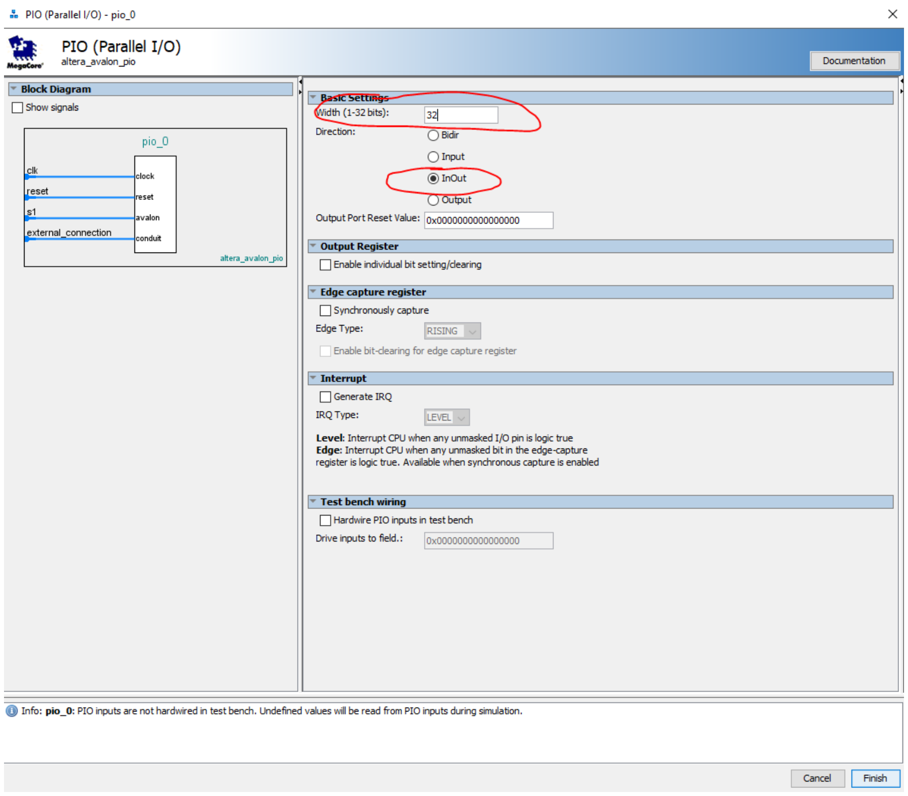
| Use | Name | Description | Export | Clock | Base |
|---|---|---|---|---|---|
| | reset | Reset Input | Double-click to export | [clk] | |
| | data_master | Avalon Memory Mapped Master | Double-click to export | [clk] | |
| | instruction_master | Avalon Memory Mapped Master | Double-click to export | [clk] | |
| | irq | Interrupt Receiver | Double-click to export | [clk] | |
| | debug_reset_request | Reset Output | Double-click to export | [clk] | |
| | debug_mem_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| | custom_instruction_m… | Custom Instruction Master | Double-click to export | | |
| ✓ | jtag_uart | JTAG UART | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | avalon_jtag_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| | irq | Interrupt Sender | Double-click to export | [clk] | |
| ✓ | onchip_memory2 | On-Chip Memory (RAM or ROM) | | | |
| | clk1 | Clock Input | Double-click to export | clk_0 | |
| | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk1] | 0x0000_0 |
| | reset1 | Reset Input | Double-click to export | [clk1] | |
| ✓ | sysid_qsys | System ID Peripheral | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | control_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| ✓ | new_sdram_controll… | SDRAM Controller | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| | wire | Conduit | Double-click to export | | |

Messages

| Type | Path | Message |
|---|---|---|
| | 9 Errors | |
| | Nios_System.nios2_gen2.data_master | onchip_memory2.s1 (0x0..0x3ffff) overlaps jtag_uart.avalon_jtag_slave (0x0..0x7) |
| | Nios_System.nios2_gen2.data_master | sysid_qsys.control_slave (0x0..0x7) overlaps onchip_memory2.s1 (0x0..0x3ffff) |
| | Nios_System.nios2_gen2.data_master | new_sdram_controller.s1 (0x0..0xffffff) overlaps sysid_qsys.control_slave (0x0..0x7) |
| | Nios_System.nios2_gen2.data_master | nios2_gen2.debug_mem_slave (0x800..0xfff) overlaps new_sdram_controller.s1 (0x0..0xffffff) |

9 Errors, 2 Warnings

## Figure 1-27 Adding PIO to system



PIO (Parallel I/O) - pio_0

**Basic Settings**

Width (1-32 bits): 32

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**

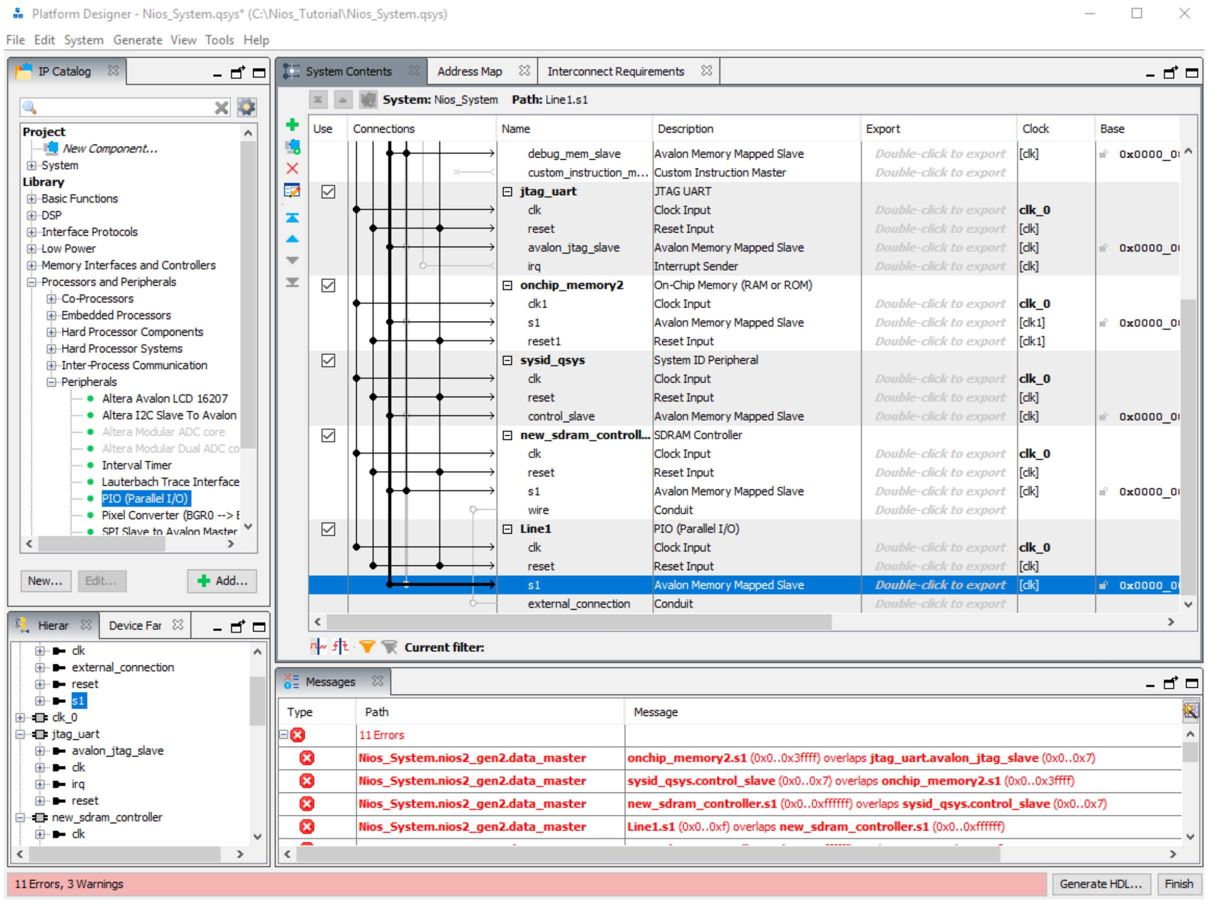
☐ Hardwire PIO inputs in test bench

Drive inputs to field.: 0x0000000000000000

Info: **pio_0:** PIO inputs are not hardwired in test bench. Undefined values will be read from PIO inputs during simulation.

## Figure 1-28 Adding 32 bit inout PIO

19.) We will connect the PIO clk, reset, and data master as shown in Figure 1-29. Also we will export the PIO shown in Figure 1-30.



Platform Designer - Nios_System.qsys* (C:\Nios_Tutorial\Nios_System.qsys)

System: Nios_System  Path: Line1.s1

| Use | Name | Description | Export | Clock | Base |
|---|---|---|---|---|---|
| | debug_mem_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| | custom_instruction_m… | Custom Instruction Master | Double-click to export | | |
| ☑ | jtag_uart | JTAG UART | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | avalon_jtag_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| | irq | Interrupt Sender | Double-click to export | [clk] | |
| ☑ | onchip_memory2 | On-Chip Memory (RAM or ROM) | | | |
| | clk1 | Clock Input | Double-click to export | clk_0 | |
| | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk1] | 0x0000_0 |
| | reset1 | Reset Input | Double-click to export | [clk1] | |
| ☑ | sysid_qsys | System ID Peripheral | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | control_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| ☑ | new_sdram_controll… | SDRAM Controller | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| | wire | Conduit | Double-click to export | | |
| ☑ | Line1 | PIO (Parallel I/O) | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| | external_connection | Conduit | Double-click to export | | |

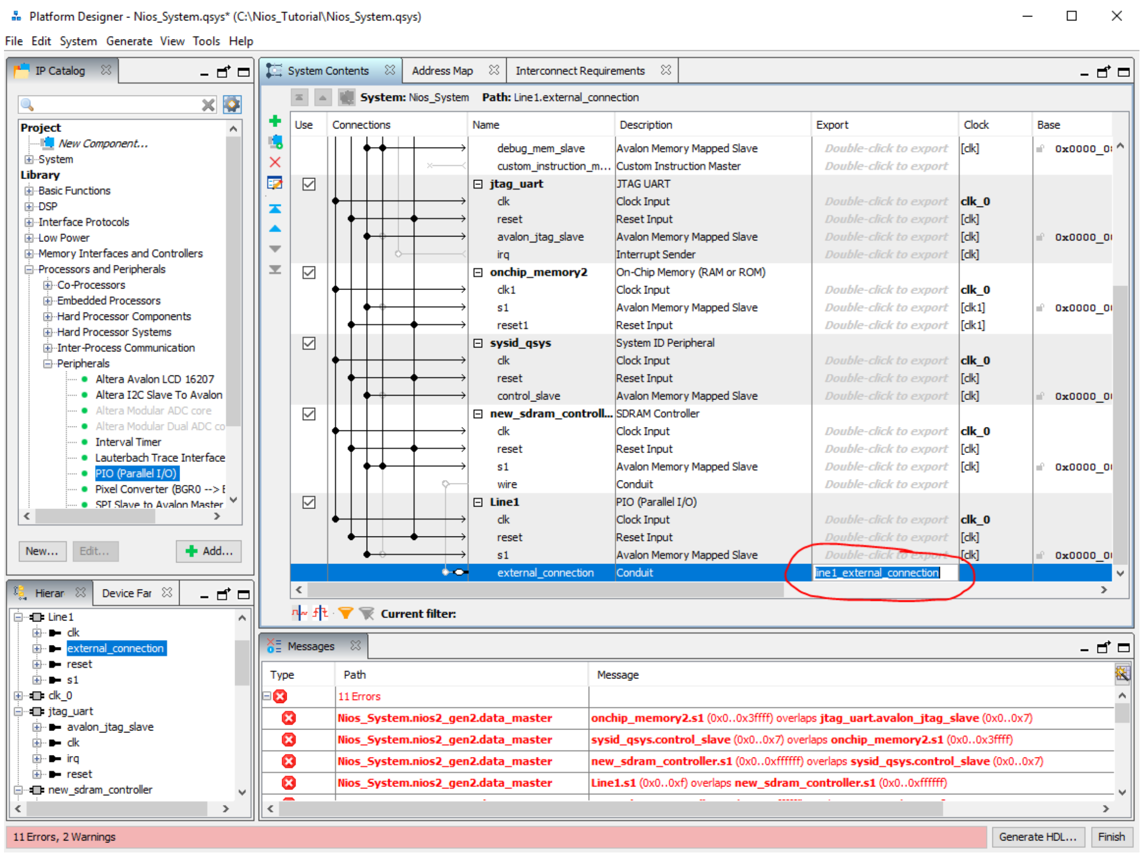
Current filter:

Messages

| Type | Path | Message |
|---|---|---|
| ⊗ | 11 Errors | |
| ⊗ | Nios_System.nios2_gen2.data_master | onchip_memory2.s1 (0x0..0x3ffff) overlaps jtag_uart.avalon_jtag_slave (0x0..0x7) |
| ⊗ | Nios_System.nios2_gen2.data_master | sysid_qsys.control_slave (0x0..0x7) overlaps onchip_memory2.s1 (0x0..0x3ffff) |
| ⊗ | Nios_System.nios2_gen2.data_master | new_sdram_controller.s1 (0x0..0xffffff) overlaps sysid_qsys.control_slave (0x0..0x7) |
| ⊗ | Nios_System.nios2_gen2.data_master | Line1.s1 (0x0..0xf) overlaps new_sdram_controller.s1 (0x0..0xffffff) |

11 Errors, 3 Warnings

**Figure 1-29 Connecting PIO connections**



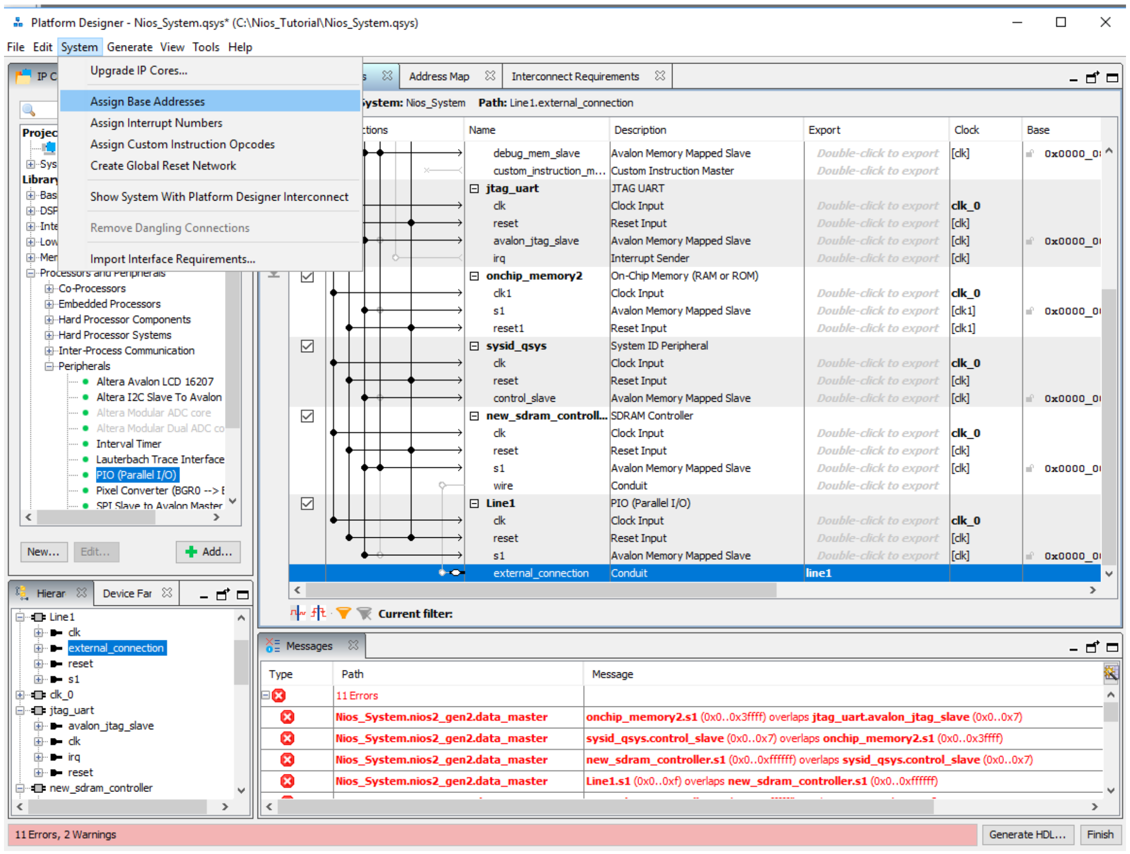
System: Nios_System Path: Line1.external_connection

| Use | Name | Description | Export | Clock | Base |
|---|---|---|---|---|---|
| | debug_mem_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| | custom_instruction_m... | Custom Instruction Master | Double-click to export | | |
| ✓ | jtag_uart | JTAG UART | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | avalon_jtag_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| | irq | Interrupt Sender | Double-click to export | [clk] | |
| ✓ | onchip_memory2 | On-Chip Memory (RAM or ROM) | | | |
| | clk1 | Clock Input | Double-click to export | clk_0 | |
| | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk1] | 0x0000_0 |
| | reset1 | Reset Input | Double-click to export | [clk1] | |
| ✓ | sysid_qsys | System ID Peripheral | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | control_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| ✓ | new_sdram_controll... | SDRAM Controller | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| | wire | Conduit | Double-click to export | | |
| ✓ | Line1 | PIO (Parallel I/O) | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| | external_connection | Conduit | line1_external_connection | | |

| Type | Path | Message |
|---|---|---|
| ⊗ | 11 Errors | |
| ⊗ | Nios_System.nios2_gen2.data_master | onchip_memory2.s1 (0x0..0x3ffff) overlaps jtag_uart.avalon_jtag_slave (0x0..0x7) |
| ⊗ | Nios_System.nios2_gen2.data_master | sysid_qsys.control_slave (0x0..0x7) overlaps onchip_memory2.s1 (0x0..0x3ffff) |
| ⊗ | Nios_System.nios2_gen2.data_master | new_sdram_controller.s1 (0x0..0xffffff) overlaps sysid_qsys.control_slave (0x0..0x7) |
| ⊗ | Nios_System.nios2_gen2.data_master | Line1.s1 (0x0..0xf) overlaps new_sdram_controller.s1 (0x0..0xffffff) |

**Figure 1-30 We will export the PIO**

20.) Now we need to assign base addresses to all these components. To do this **choose System → Assign Base Addresses** as shown in Figure 1-31.


Platform Designer - Nios_System.qsys* (C:\Nios_Tutorial\Nios_System.qsys)
File Edit System Generate View Tools Help
Upgrade IP Cores...
Assign Base Addresses
Assign Interrupt Numbers
Assign Custom Instruction Opcodes
Create Global Reset Network
Show System With Platform Designer Interconnect
Remove Dangling Connections
Import Interface Requirements...
Address Map
Interconnect Requirements
New... Edit... Add...
Generate HDL... Finish

ystem: Nios_System  Path: Line1.external_connection

| Name | Description | Export | Clock | Base |
|---|---|---|---|---|
| debug_mem_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| custom_instruction_m... | Custom Instruction Master | Double-click to export | | |
| jtag_uart | JTAG UART | | | |
| clk | Clock Input | Double-click to export | clk_0 | |
| reset | Reset Input | Double-click to export | [clk] | |
| avalon_jtag_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| irq | Interrupt Sender | Double-click to export | [clk] | |
| onchip_memory2 | On-Chip Memory (RAM or ROM) | | | |
| clk1 | Clock Input | Double-click to export | clk_0 | |
| s1 | Avalon Memory Mapped Slave | Double-click to export | [clk1] | 0x0000_0 |
| reset1 | Reset Input | Double-click to export | [clk1] | |
| sysid_qsys | System ID Peripheral | | | |
| clk | Clock Input | Double-click to export | clk_0 | |
| reset | Reset Input | Double-click to export | [clk] | |
| control_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| new_sdram_controll... | SDRAM Controller | | | |
| clk | Clock Input | Double-click to export | clk_0 | |
| reset | Reset Input | Double-click to export | [clk] | |
| s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| wire | Conduit | Double-click to export | | |
| Line1 | PIO (Parallel I/O) | | | |
| clk | Clock Input | Double-click to export | clk_0 | |
| reset | Reset Input | Double-click to export | [clk] | |
| s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0 |
| external_connection | Conduit | line1 | | |

Current filter:

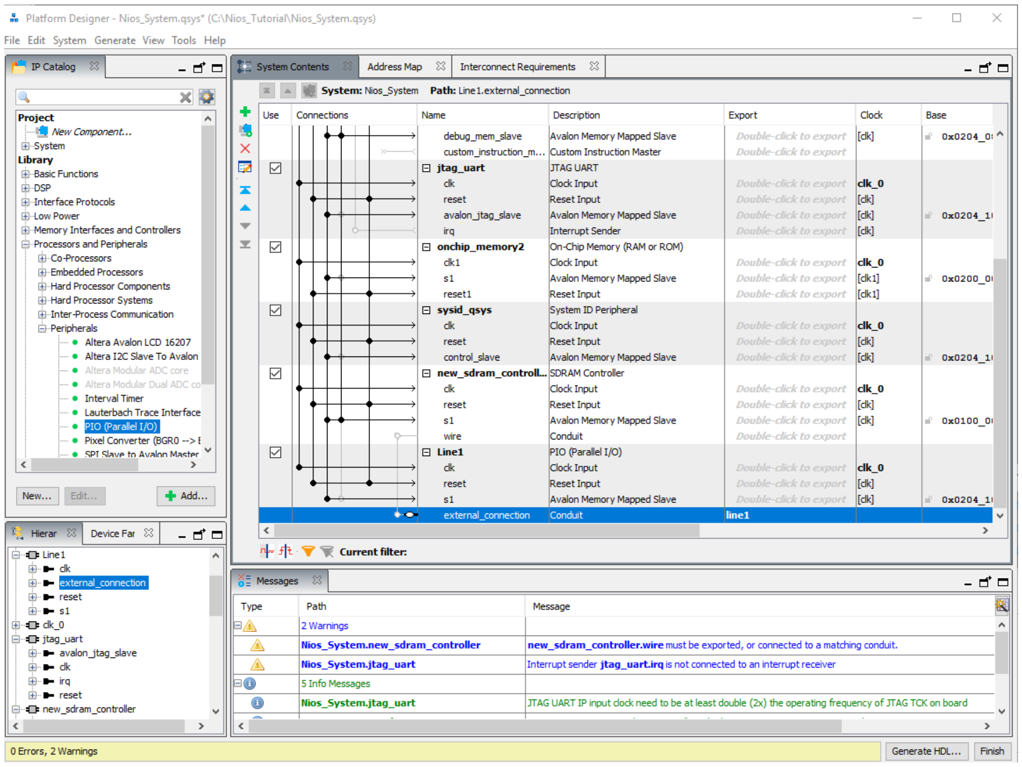
Messages

| Type | Path | Message |
|---|---|---|
| | 11 Errors | |
| | Nios_System.nios2_gen2.data_master | onchip_memory2.s1 (0x0..0x3ffff) overlaps jtag_uart.avalon_jtag_slave (0x0..0x7) |
| | Nios_System.nios2_gen2.data_master | sysid_qsys.control_slave (0x0..0x7) overlaps onchip_memory2.s1 (0x0..0x3ffff) |
| | Nios_System.nios2_gen2.data_master | new_sdram_controller.s1 (0x0..0xffffff) overlaps sysid_qsys.control_slave (0x0..0x7) |
| | Nios_System.nios2_gen2.data_master | Line1.s1 (0x0..0xf) overlaps new_sdram_controller.s1 (0x0..0xffffff) |

11 Errors, 2 Warnings

**Figure 1-31 Assigning base addresses**

21.) Now you see that we only have a couple of warning shown in Figure 1-32. To get rid of those warnings, we will assign Interrupt Numbers shown in Figure 1-33 and 1-34.


Platform Designer - Nios_System.qsys* (C:\Nios_Tutorial\Nios_System.qsys)

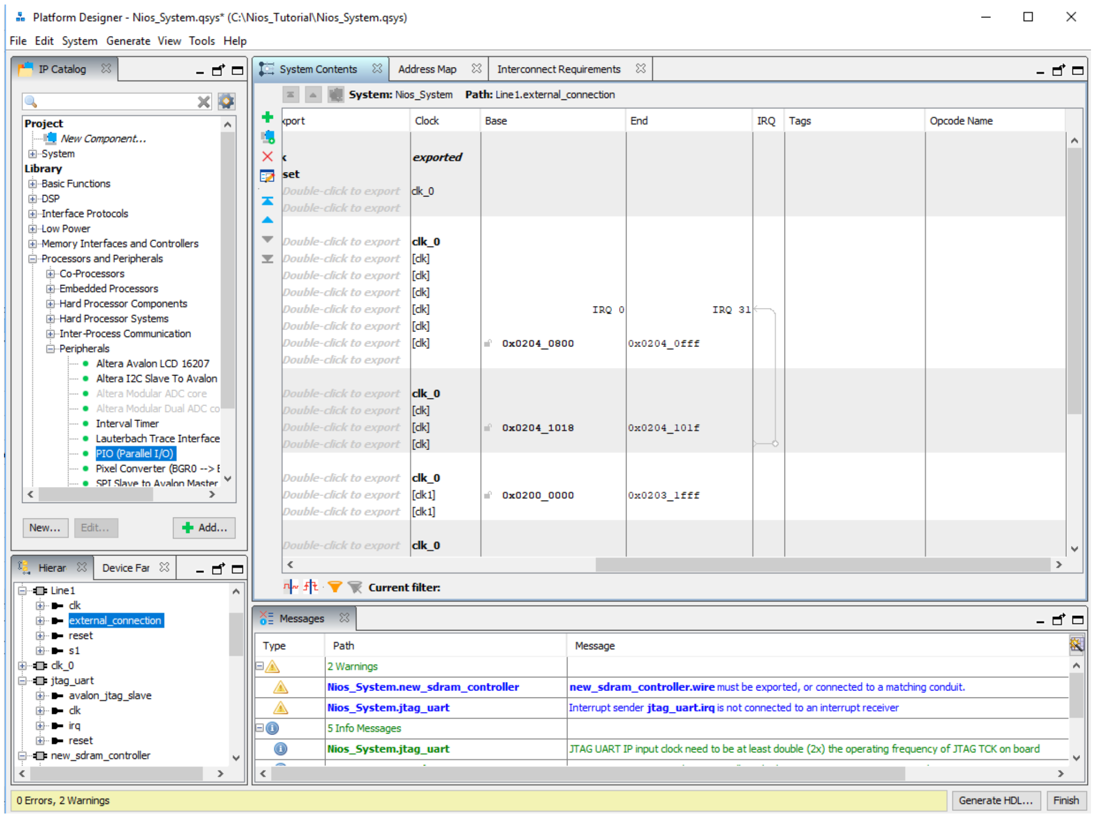
System: Nios_System Path: Line1.external_connection

| Use | Name | Description | Export | Clock | Base |
|---|---|---|---|---|---|
| | debug_mem_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0204_0 |
| | custom_instruction_m... | Custom Instruction Master | Double-click to export | | |
| ✓ | jtag_uart | JTAG UART | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | avalon_jtag_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0204_1 |
| | irq | Interrupt Sender | Double-click to export | [clk] | |
| ✓ | onchip_memory2 | On-Chip Memory (RAM or ROM) | | | |
| | clk1 | Clock Input | Double-click to export | clk_0 | |
| | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk1] | 0x0200_0 |
| | reset1 | Reset Input | Double-click to export | [clk1] | |
| ✓ | sysid_qsys | System ID Peripheral | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | control_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0204_1 |
| ✓ | new_sdram_controll... | SDRAM Controller | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0100_0 |
| | wire | Conduit | Double-click to export | | |
| ✓ | Line1 | PIO (Parallel I/O) | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0204_1 |
| | external_connection | Conduit | line1 | | |

| Type | Path | Message |
|---|---|---|
| | 2 Warnings | |
| | Nios_System.new_sdram_controller | new_sdram_controller.wire must be exported, or connected to a matching conduit. |
| | Nios_System.jtag_uart | Interrupt sender jtag_uart.irq is not connected to an interrupt receiver |
| | 5 Info Messages | |
| | Nios_System.jtag_uart | JTAG UART IP input clock need to be at least double (2x) the operating frequency of JTAG TCK on board |

0 Errors, 2 Warnings

**Figure 1-32**


Platform Designer - Nios_System.qsys* (C:\Nios_Tutorial\Nios_System.qsys)

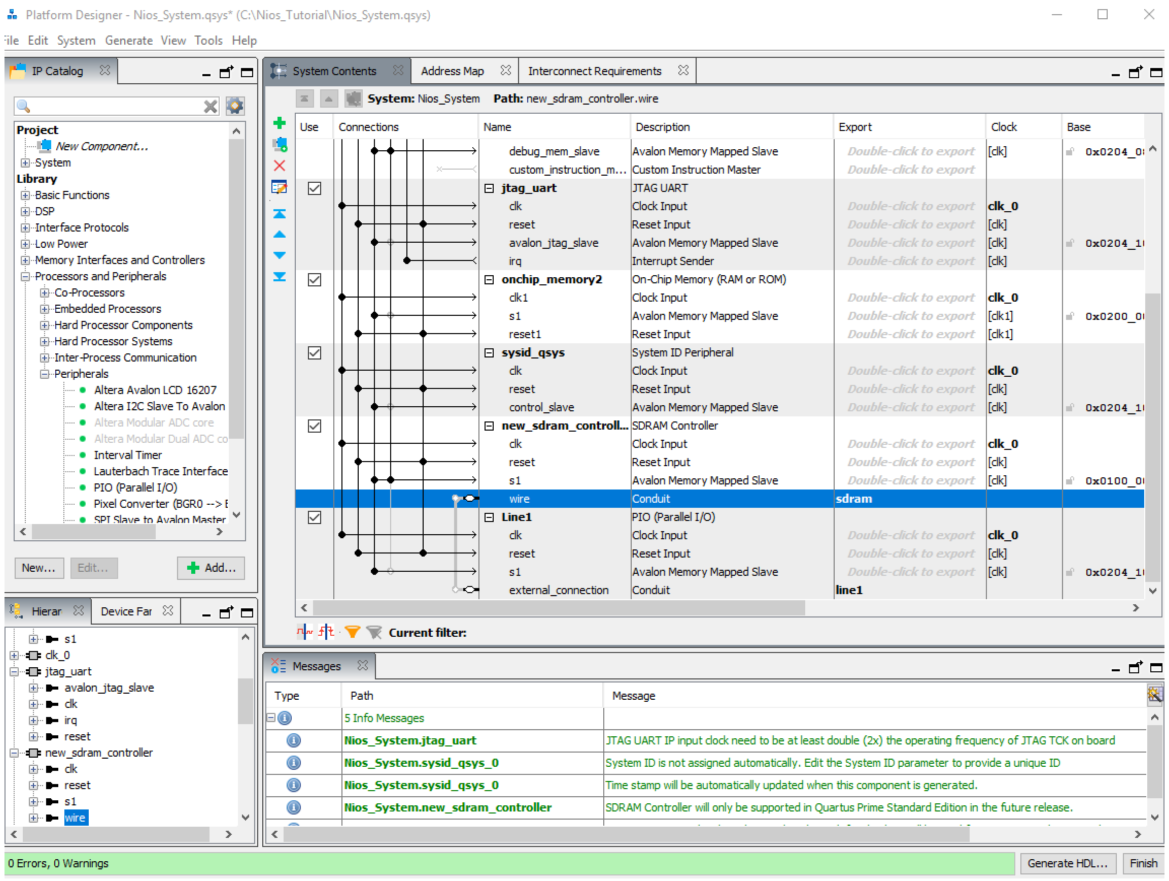
System: Nios_System Path: Line1.external_connection

| xport | Clock | Base | End | IRQ | Tags | Opcode Name |
|---|---|---|---|---|---|---|
| k | exported | | | | | |
| set | | | | | | |
| Double-click to export | clk_0 | | | | | |
| Double-click to export | | | | | | |
| Double-click to export | clk_0 | | | | | |
| Double-click to export | [clk] | | | | | |
| Double-click to export | [clk] | | | | | |
| Double-click to export | [clk] | | | | | |
| Double-click to export | [clk] | IRQ 0 | IRQ 31 | | | |
| Double-click to export | [clk] | | | | | |
| Double-click to export | [clk] | 0x0204_0800 | 0x0204_0fff | | | |
| Double-click to export | | | | | | |
| Double-click to export | clk_0 | | | | | |
| Double-click to export | [clk] | | | | | |
| Double-click to export | [clk] | 0x0204_1018 | 0x0204_101f | | | |
| Double-click to export | [clk] | | | | | |
| Double-click to export | clk_0 | | | | | |
| Double-click to export | [clk1] | 0x0200_0000 | 0x0203_1fff | | | |
| Double-click to export | [clk1] | | | | | |
| Double-click to export | clk_0 | | | | | |

| Type | Path | Message |
|---|---|---|
| | 2 Warnings | |
| | Nios_System.new_sdram_controller | new_sdram_controller.wire must be exported, or connected to a matching conduit. |
| | Nios_System.jtag_uart | Interrupt sender jtag_uart.irq is not connected to an interrupt receiver |
| | 5 Info Messages | |
| | Nios_System.jtag_uart | JTAG UART IP input clock need to be at least double (2x) the operating frequency of JTAG TCK on board |

0 Errors, 2 Warnings

**Figure 1-33 Assigning interrupt numbers**


Platform Designer - Nios_System.qsys* (C:\Nios_Tutorial\Nios_System.qsys)
File Edit System Generate View Tools Help
IP Catalog
System Contents
Address Map
Interconnect Requirements
System: Nios_System Path: new_sdram_controller.wire
Project
New Component...
System
Library
Basic Functions
DSP
Interface Protocols
Low Power
Memory Interfaces and Controllers
Processors and Peripherals
Co-Processors
Embedded Processors
Hard Processor Components
Hard Processor Systems
Inter-Process Communication
Peripherals
Altera Avalon LCD 16207
Altera I2C Slave To Avalon
Altera Modular ADC core
Altera Modular Dual ADC co
Interval Timer
Lauterbach Trace Interface
PIO (Parallel I/O)
Pixel Converter (BGR0 --> I
SPI Slave to Avalon Master
New...
Edit...
Add...
Hierar
Device Far
s1
clk_0
jtag_uart
avalon_jtag_slave
clk
irq
reset
new_sdram_controller
clk
reset
s1
wire
Current filter:
Messages
Generate HDL...
Finish

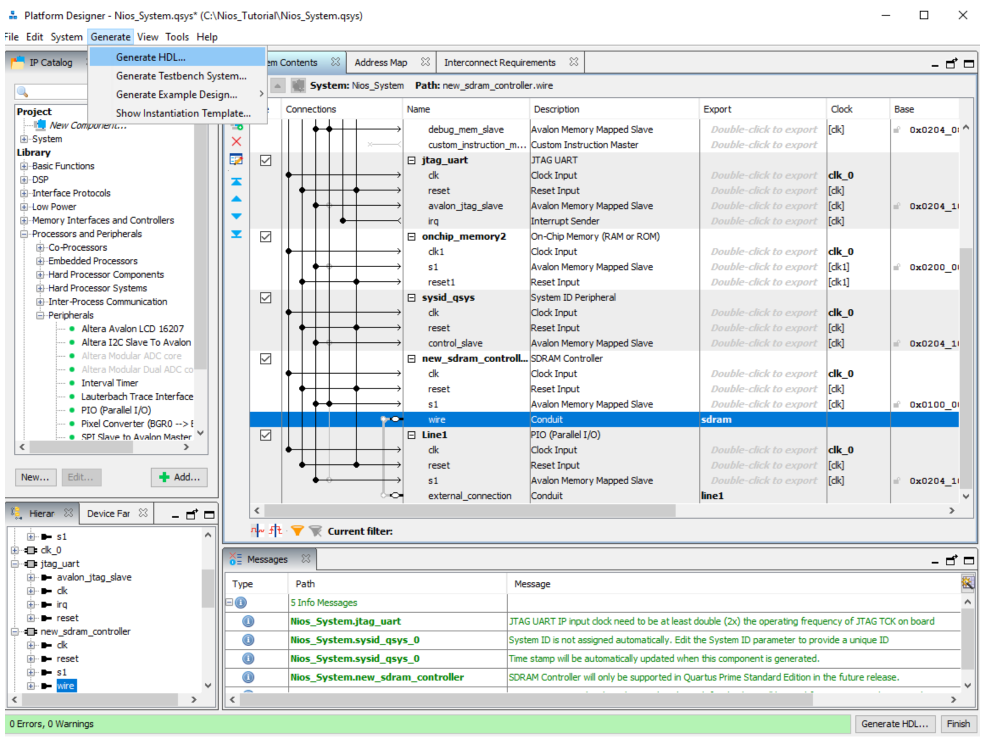
| Use | Name | Description | Export | Clock | Base |
|---|---|---|---|---|---|
| | debug_mem_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0204_0 |
| | custom_instruction_m... | Custom Instruction Master | Double-click to export | | |
| ☑ | jtag_uart | JTAG UART | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | avalon_jtag_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0204_1 |
| | irq | Interrupt Sender | Double-click to export | [clk] | |
| ☑ | onchip_memory2 | On-Chip Memory (RAM or ROM) | | | |
| | clk1 | Clock Input | Double-click to export | clk_0 | |
| | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk1] | 0x0200_0 |
| | reset1 | Reset Input | Double-click to export | [clk1] | |
| ☑ | sysid_qsys | System ID Peripheral | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | control_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0204_1 |
| ☑ | new_sdram_controll... | SDRAM Controller | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0100_0 |
| | wire | Conduit | sdram | | |
| ☑ | Line1 | PIO (Parallel I/O) | | | |
| | clk | Clock Input | Double-click to export | clk_0 | |
| | reset | Reset Input | Double-click to export | [clk] | |
| | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0204_1 |
| | external_connection | Conduit | line1 | | |

| Type | Path | Message |
|---|---|---|
| | 5 Info Messages | |
| | Nios_System.jtag_uart | JTAG UART IP input clock need to be at least double (2x) the operating frequency of JTAG TCK on board |
| | Nios_System.sysid_qsys_0 | System ID is not assigned automatically. Edit the System ID parameter to provide a unique ID |
| | Nios_System.sysid_qsys_0 | Time stamp will be automatically updated when this component is generated. |
| | Nios_System.new_sdram_controller | SDRAM Controller will only be supported in Quartus Prime Standard Edition in the future release. |

0 Errors, 0 Warnings

**Figure 1-34 Select the Hollow dot to connect the Interrupt Numbers**

22.) Now we have no more warning and we can Generate our system. To do that select the **Generate** tab at the top of the screen as shown in figures 1-35 and 1-36.

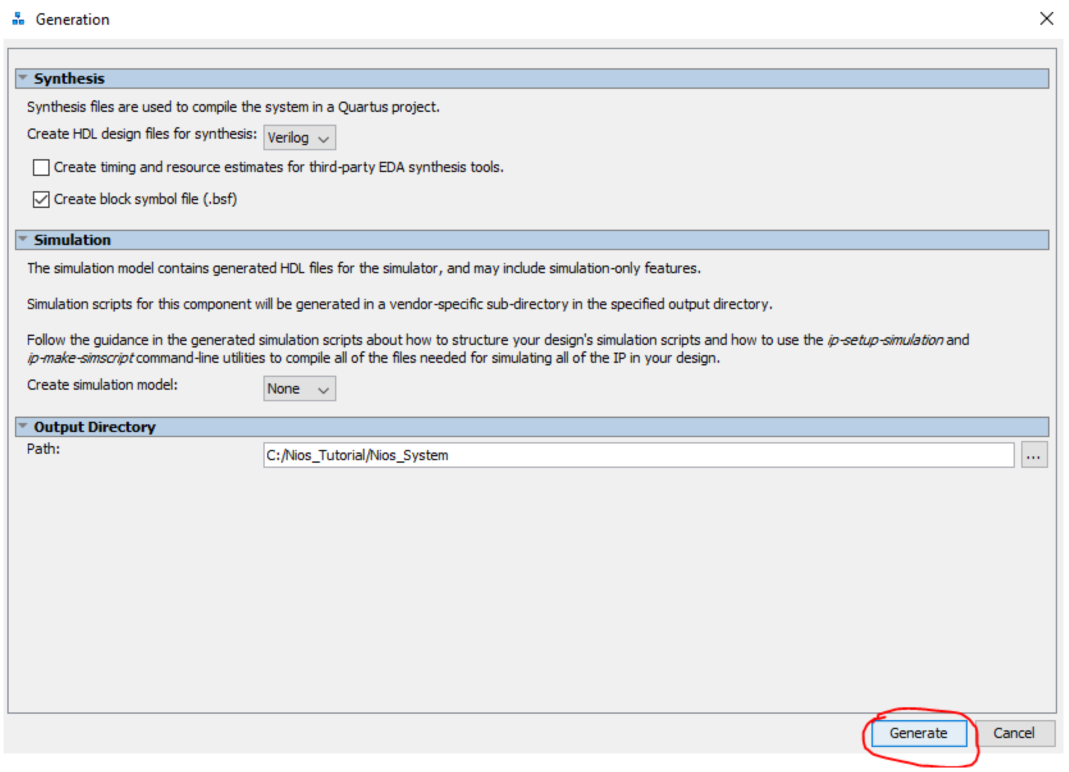

Platform Designer - Nios_System.qsys* (C:\Nios_Tutorial\Nios_System.qsys)
File Edit System Generate View Tools Help
Generate HDL...
Generate Testbench System...
Generate Example Design...
Show Instantiation Template...
IP Catalog
Project
System
Library
Basic Functions
DSP
Interface Protocols
Low Power
Memory Interfaces and Controllers
Processors and Peripherals
Co-Processors
Embedded Processors
Hard Processor Components
Hard Processor Systems
Inter-Process Communication
Peripherals
Altera Avalon LCD 16207
Altera I2C Slave To Avalon
Altera Modular ADC core
Altera Modular Dual ADC co
Interval Timer
Lauterbach Trace Interface
PIO (Parallel I/O)
Pixel Converter (BGR0 --> B
SPI Slave to Avalon Master
New... Edit... Add...
Hierar Device Far
s1
clk_0
jtag_uart
avalon_jtag_slave
clk
irq
reset
new_sdram_controller
clk
reset
s1
wire
System Contents Address Map Interconnect Requirements
Current filter:
Messages
0 Errors, 0 Warnings
Generate HDL... Finish

System: Nios_System Path: new_sdram_controller.wire

| Name | Description | Export | Clock | Base |
|---|---|---|---|---|
| debug_mem_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0204_0 |
| custom_instruction_m... | Custom Instruction Master | Double-click to export | | |
| jtag_uart | JTAG UART | | | |
| clk | Clock Input | Double-click to export | clk_0 | |
| reset | Reset Input | Double-click to export | [clk] | |
| avalon_jtag_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0204_1 |
| irq | Interrupt Sender | Double-click to export | [clk] | |
| onchip_memory2 | On-Chip Memory (RAM or ROM) | | | |
| clk1 | Clock Input | Double-click to export | clk_0 | |
| s1 | Avalon Memory Mapped Slave | Double-click to export | [clk1] | 0x0200_0 |
| reset1 | Reset Input | Double-click to export | [clk1] | |
| sysid_qsys | System ID Peripheral | | | |
| clk | Clock Input | Double-click to export | clk_0 | |
| reset | Reset Input | Double-click to export | [clk] | |
| control_slave | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0204_1 |
| new_sdram_controll... | SDRAM Controller | | | |
| clk | Clock Input | Double-click to export | clk_0 | |
| reset | Reset Input | Double-click to export | [clk] | |
| s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0100_0 |
| wire | Conduit | sdram | | |
| Line1 | PIO (Parallel I/O) | | | |
| clk | Clock Input | Double-click to export | clk_0 | |
| reset | Reset Input | Double-click to export | [clk] | |
| s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0204_1 |
| external_connection | Conduit | line1 | | |

| Type | Path | Message |
|---|---|---|
| | 5 Info Messages | |
| | Nios_System.jtag_uart | JTAG UART IP input clock need to be at least double (2x) the operating frequency of JTAG TCK on board |
| | Nios_System.sysid_qsys_0 | System ID is not assigned automatically. Edit the System ID parameter to provide a unique ID |
| | Nios_System.sysid_qsys_0 | Time stamp will be automatically updated when this component is generated. |
| | Nios_System.new_sdram_controller | SDRAM Controller will only be supported in Quartus Prime Standard Edition in the future release. |

**Figure 1-35 Generate HDL tab**


Generation

**Synthesis**

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

Create HDL design files for synthesis: Verilog

☐ 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.

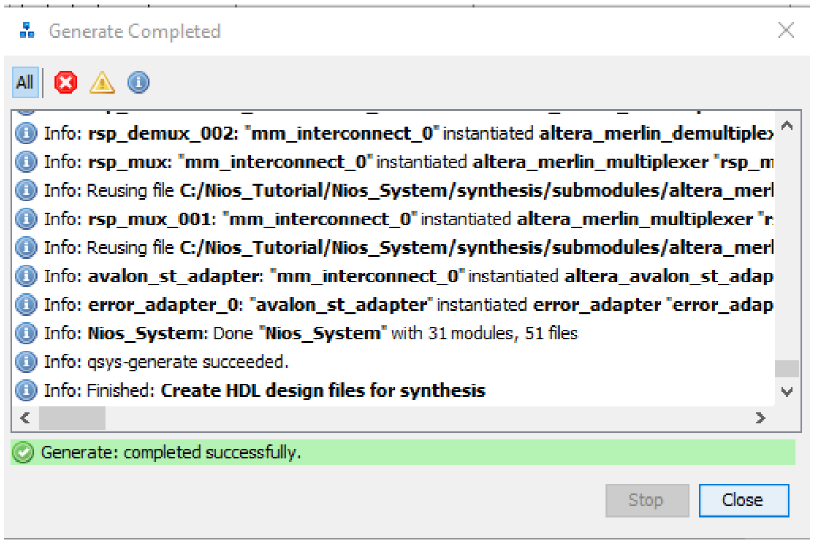
Follow the guidance in the generated simulation scripts about how to structure your design's simulation scripts and how to use the *ip-setup-simulation* and *ip-make-simscript* command-line utilities to compile all of the files needed for simulating all of the IP in your design.

Create simulation model: None

**Output Directory**

Path: C:/Nios_Tutorial/Nios_System

**Figure 1-36 Generate the system**



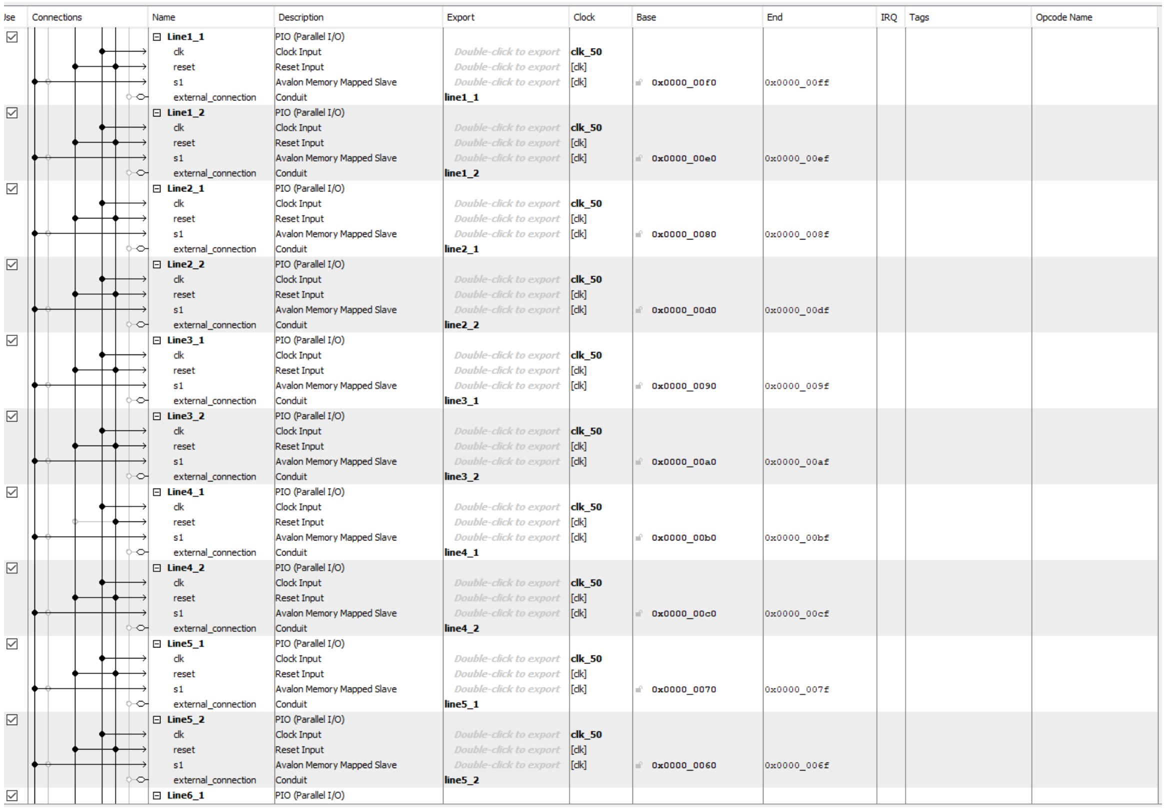
Generate Completed

```
Info: rsp_demux_002: "mm_interconnect_0" instantiated altera_merlin_demultiplex
Info: rsp_mux: "mm_interconnect_0" instantiated altera_merlin_multiplexer "rsp_m
Info: Reusing file C:/Nios_Tutorial/Nios_System/synthesis/submodules/altera_merl
Info: rsp_mux_001: "mm_interconnect_0" instantiated altera_merlin_multiplexer "r
Info: Reusing file C:/Nios_Tutorial/Nios_System/synthesis/submodules/altera_merl
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_System: Done "Nios_System" with 31 modules, 51 files
Info: qsys-generate succeeded.
Info: Finished: Create HDL design files for synthesis
```

Generate: completed successfully.

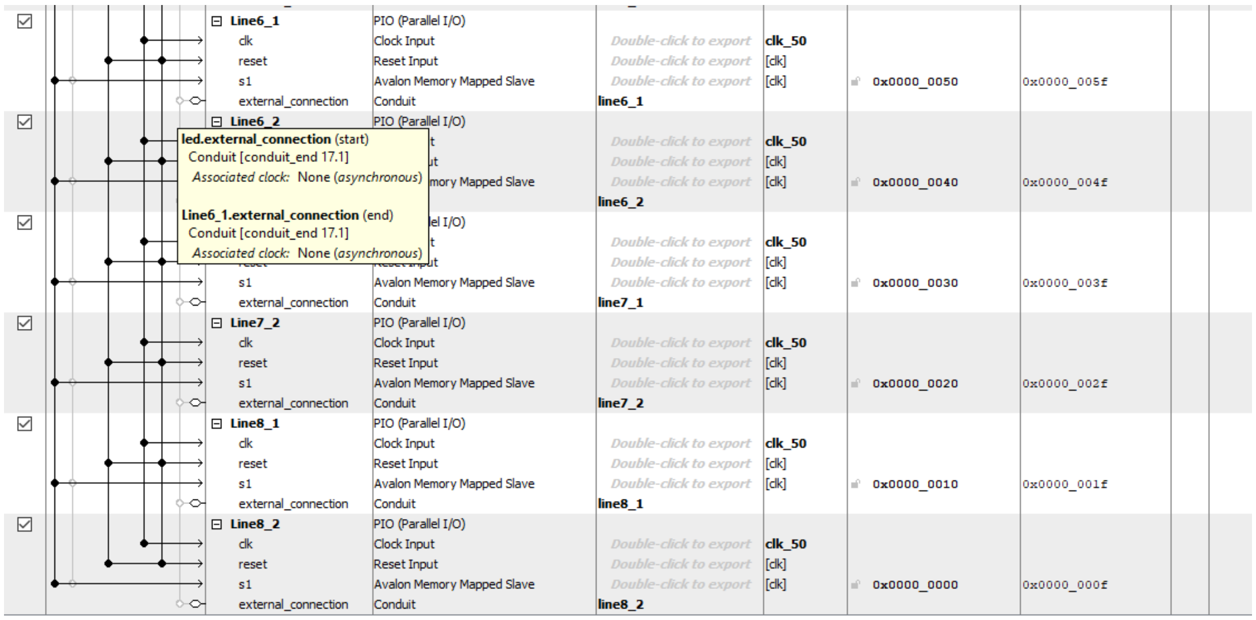
**Figure 1-37 Generation is successful**

23.) We will be sending an 8x8 matrix of 8-bit numbers to and from the Nios processor so we will need a lot more PIO's. To do this we will do it exactly how it was shown previously. For our purposes we used 16 PIO's to make up the 8x8 matrix. The final product of this is shown in Figure 1-38 and 1-39.



| Ise | Connections | Name | Description | Export | Clock | Base | End | IRQ | Tags | Opcode Name |
|---|---|---|---|---|---|---|---|---|---|---|
| ✓ | | Line1_1 | PIO (Parallel I/O) | | | | | | | |
| | | clk | Clock Input | Double-click to export | clk_50 | | | | | |
| | | reset | Reset Input | Double-click to export | [clk] | | | | | |
| | | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_00f0 | 0x0000_00ff | | | |
| | | external_connection | Conduit | line1_1 | | | | | | |
| ✓ | | Line1_2 | PIO (Parallel I/O) | | | | | | | |
| | | clk | Clock Input | Double-click to export | clk_50 | | | | | |
| | | reset | Reset Input | Double-click to export | [clk] | | | | | |
| | | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_00e0 | 0x0000_00ef | | | |
| | | external_connection | Conduit | line1_2 | | | | | | |
| ✓ | | Line2_1 | PIO (Parallel I/O) | | | | | | | |
| | | clk | Clock Input | Double-click to export | clk_50 | | | | | |
| | | reset | Reset Input | Double-click to export | [clk] | | | | | |
| | | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0080 | 0x0000_008f | | | |
| | | external_connection | Conduit | line2_1 | | | | | | |
| ✓ | | Line2_2 | PIO (Parallel I/O) | | | | | | | |
| | | clk | Clock Input | Double-click to export | clk_50 | | | | | |
| | | reset | Reset Input | Double-click to export | [clk] | | | | | |
| | | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_00d0 | 0x0000_00df | | | |
| | | external_connection | Conduit | line2_2 | | | | | | |
| ✓ | | Line3_1 | PIO (Parallel I/O) | | | | | | | |
| | | clk | Clock Input | Double-click to export | clk_50 | | | | | |
| | | reset | Reset Input | Double-click to export | [clk] | | | | | |
| | | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0090 | 0x0000_009f | | | |
| | | external_connection | Conduit | line3_1 | | | | | | |
| ✓ | | Line3_2 | PIO (Parallel I/O) | | | | | | | |
| | | clk | Clock Input | Double-click to export | clk_50 | | | | | |
| | | reset | Reset Input | Double-click to export | [clk] | | | | | |
| | | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_00a0 | 0x0000_00af | | | |
| | | external_connection | Conduit | line3_2 | | | | | | |
| ✓ | | Line4_1 | PIO (Parallel I/O) | | | | | | | |
| | | clk | Clock Input | Double-click to export | clk_50 | | | | | |
| | | reset | Reset Input | Double-click to export | [clk] | | | | | |
| | | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_00b0 | 0x0000_00bf | | | |
| | | external_connection | Conduit | line4_1 | | | | | | |
| ✓ | | Line4_2 | PIO (Parallel I/O) | | | | | | | |
| | | clk | Clock Input | Double-click to export | clk_50 | | | | | |
| | | reset | Reset Input | Double-click to export | [clk] | | | | | |
| | | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_00c0 | 0x0000_00cf | | | |
| | | external_connection | Conduit | line4_2 | | | | | | |
| ✓ | | Line5_1 | PIO (Parallel I/O) | | | | | | | |
| | | clk | Clock Input | Double-click to export | clk_50 | | | | | |
| | | reset | Reset Input | Double-click to export | [clk] | | | | | |
| | | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0070 | 0x0000_007f | | | |
| | | external_connection | Conduit | line5_1 | | | | | | |
| ✓ | | Line5_2 | PIO (Parallel I/O) | | | | | | | |
| | | clk | Clock Input | Double-click to export | clk_50 | | | | | |
| | | reset | Reset Input | Double-click to export | [clk] | | | | | |
| | | s1 | Avalon Memory Mapped Slave | Double-click to export | [clk] | 0x0000_0060 | 0x0000_006f | | | |
| | | external_connection | Conduit | line5_2 | | | | | | |
| ✓ | | Line6_1 | PIO (Parallel I/O) | | | | | | | |

**Figure 1-37**



| | | | | | | |
|---|---|---|---|---|---|---|
| ☑ | ⊟ **Line6_1** | PIO (Parallel I/O) | | | | |
| | clk | Clock Input | *Double-click to export* | **clk_50** | | |
| | reset | Reset Input | *Double-click to export* | [clk] | | |
| | s1 | Avalon Memory Mapped Slave | *Double-click to export* | [clk] | 0x0000_0050 | 0x0000_005f |
| | external_connection | Conduit | **line6_1** | | | |
| ☑ | ⊟ **Line6_2** | PIO (Parallel I/O) | | | | |
| | | t | *Double-click to export* | **clk_50** | | |
| | | ut | *Double-click to export* | [clk] | | |
| | | mory Mapped Slave | *Double-click to export* | [clk] | 0x0000_0040 | 0x0000_004f |
| | | | **line6_2** | | | |
| ☑ | | el I/O) | | | | |
| | | t | *Double-click to export* | **clk_50** | | |
| | | ut | *Double-click to export* | [clk] | | |
| | s1 | Avalon Memory Mapped Slave | *Double-click to export* | [clk] | 0x0000_0030 | 0x0000_003f |
| | external_connection | Conduit | **line7_1** | | | |
| ☑ | ⊟ **Line7_2** | PIO (Parallel I/O) | | | | |
| | clk | Clock Input | *Double-click to export* | **clk_50** | | |
| | reset | Reset Input | *Double-click to export* | [clk] | | |
| | s1 | Avalon Memory Mapped Slave | *Double-click to export* | [clk] | 0x0000_0020 | 0x0000_002f |
| | external_connection | Conduit | **line7_2** | | | |
| ☑ | ⊟ **Line8_1** | PIO (Parallel I/O) | | | | |
| | clk | Clock Input | *Double-click to export* | **clk_50** | | |
| | reset | Reset Input | *Double-click to export* | [clk] | | |
| | s1 | Avalon Memory Mapped Slave | *Double-click to export* | [clk] | 0x0000_0010 | 0x0000_001f |
| | external_connection | Conduit | **line8_1** | | | |
| ☑ | ⊟ **Line8_2** | PIO (Parallel I/O) | | | | |
| | clk | Clock Input | *Double-click to export* | **clk_50** | | |
| | reset | Reset Input | *Double-click to export* | [clk] | | |
| | s1 | Avalon Memory Mapped Slave | *Double-click to export* | [clk] | 0x0000_0000 | 0x0000_000f |
| | external_connection | Conduit | **line8_2** | | | |

**led.external_connection** (start)
Conduit [conduit_end 17.1]
*Associated clock:* None (*asynchronous*)

**Line6_1.external_connection** (end)
Conduit [conduit_end 17.1]
*Associated clock:* None (*asynchronous*)

**Figure 1-38**

# Chapter 2 *Programming the FPGA*

Now we will create a Verilog file that will instantiate and run everything that we have created above. It will be what connects all the components to the FPGA.

1.) First thing we want to do is create our Verilog File. To do this, choose **File → New → Verilog HDL File** as shown in Figure 2-1 and 2-2. You will get a blank Verilog file as shown in Figure 2-3.

2.)

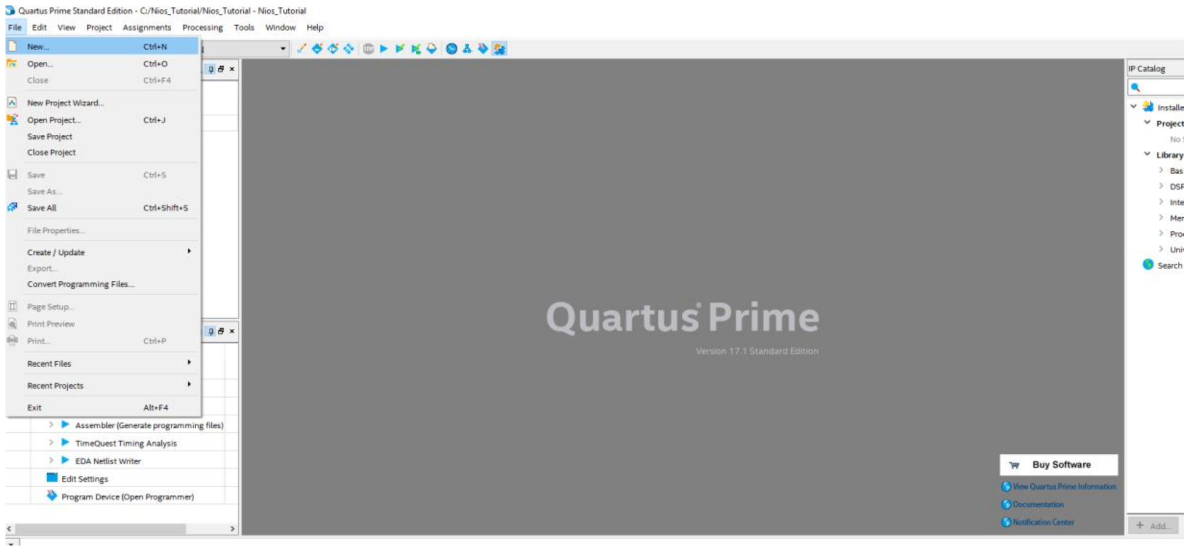


**Figure 2-1 Creating New Verilog file**

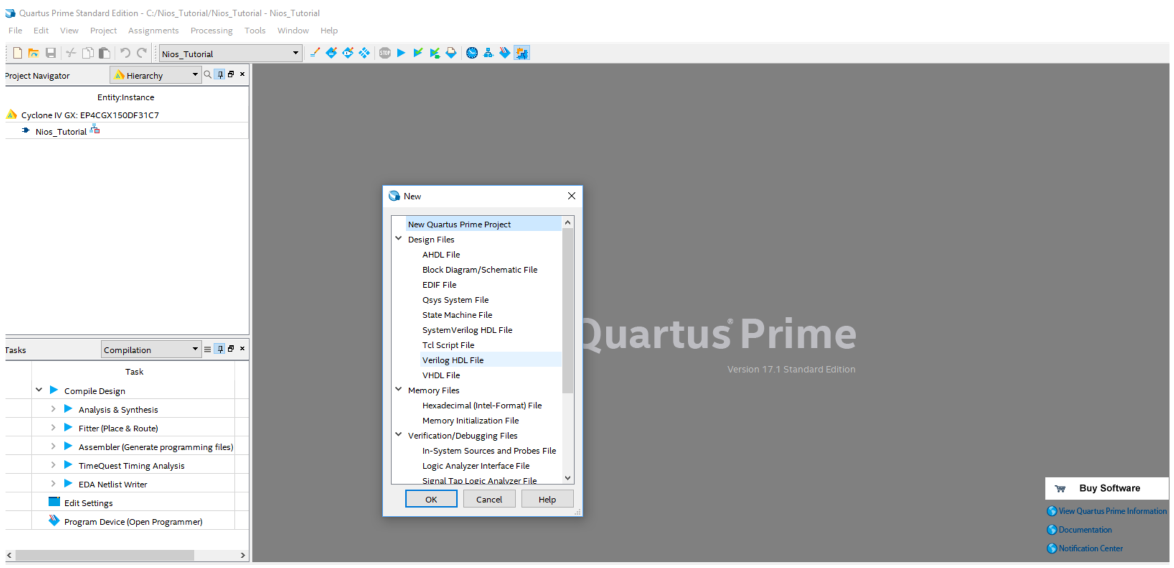

Quartus Prime Standard Edition - C:/Nios_Tutorial/Nios_Tutorial - Nios_Tutorial
File Edit View Project Assignments Processing Tools Window Help
Nios_Tutorial
Project Navigator
Hierarchy
Entity:Instance
Cyclone IV GX: EP4CGX150DF31C7
Nios_Tutorial
Tasks
Compilation
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming files)
TimeQuest Timing Analysis
EDA Netlist Writer
Edit Settings
Program Device (Open Programmer)
New
New Quartus Prime Project
Design Files
AHDL File
Block Diagram/Schematic File
EDIF File
Qsys System File
State Machine File
SystemVerilog HDL File
Tcl Script File
Verilog HDL File
VHDL File
Memory Files
Hexadecimal (Intel-Format) File
Memory Initialization File
Verification/Debugging Files
In-System Sources and Probes File
Logic Analyzer Interface File
Signal Tap Logic Analyzer File
OK
Cancel
Help
Quartus Prime
Version 17.1 Standard Edition
Buy Software
View Quartus Prime Information
Documentation
Notification Center

**Figure 2-2 Creating new Verilog File**

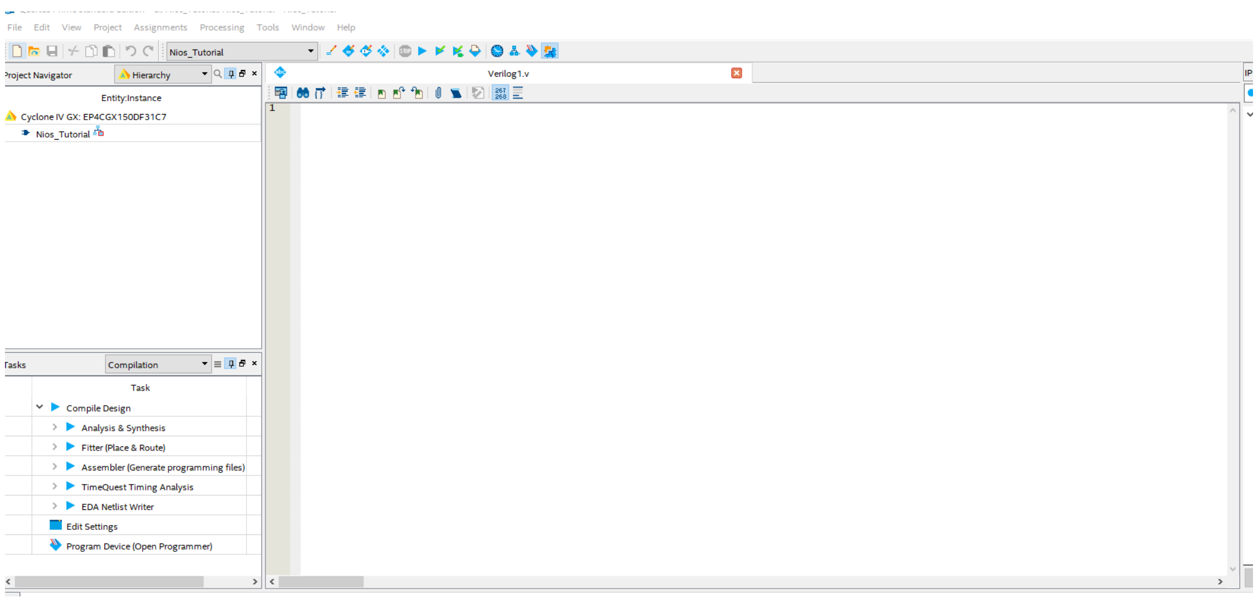
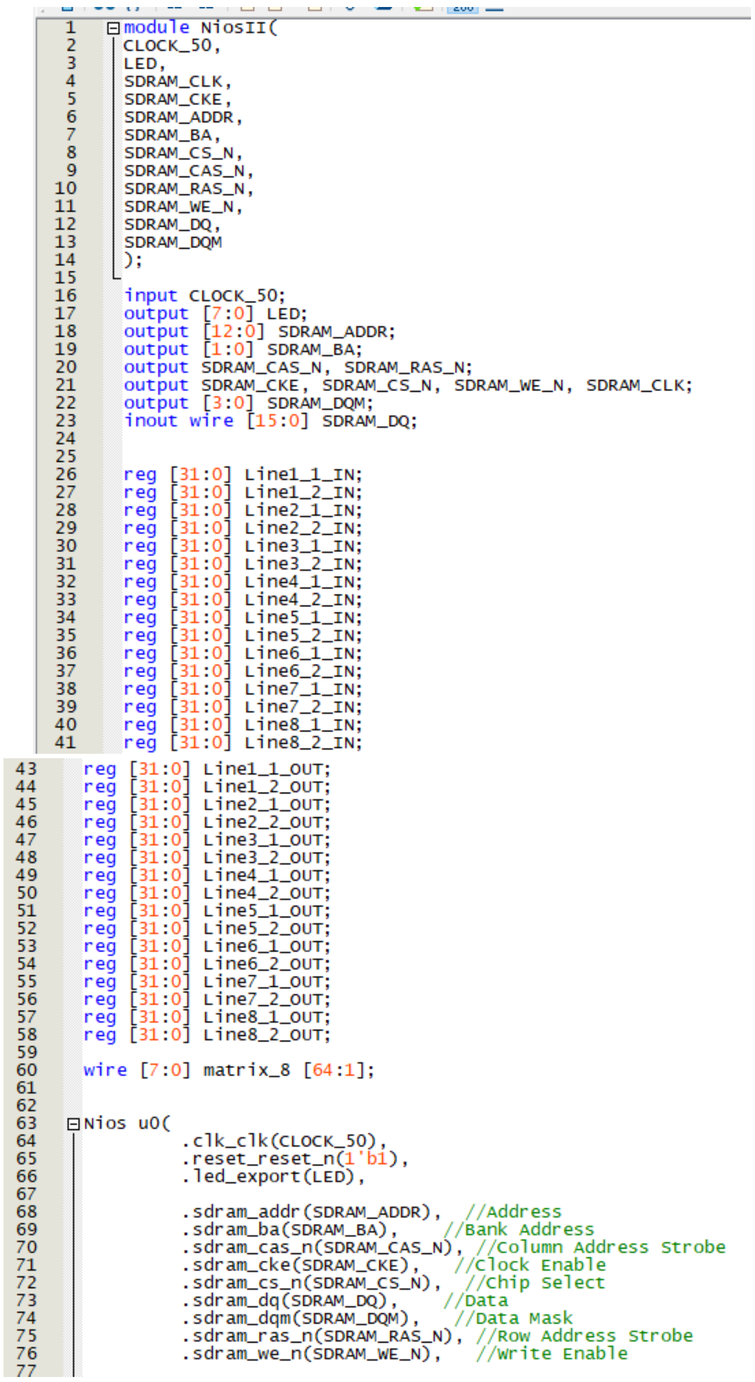

File Edit View Project Assignments Processing Tools Window Help
Nios_Tutorial
Project Navigator
Hierarchy
Entity:Instance
Cyclone IV GX: EP4CGX150DF31C7
Nios_Tutorial
Verilog1.v
Tasks
Compilation
Task
Compile Design
Analysis & Synthesis
Fitter (Place & Route)
Assembler (Generate programming files)
TimeQuest Timing Analysis
EDA Netlist Writer
Edit Settings
Program Device (Open Programmer)

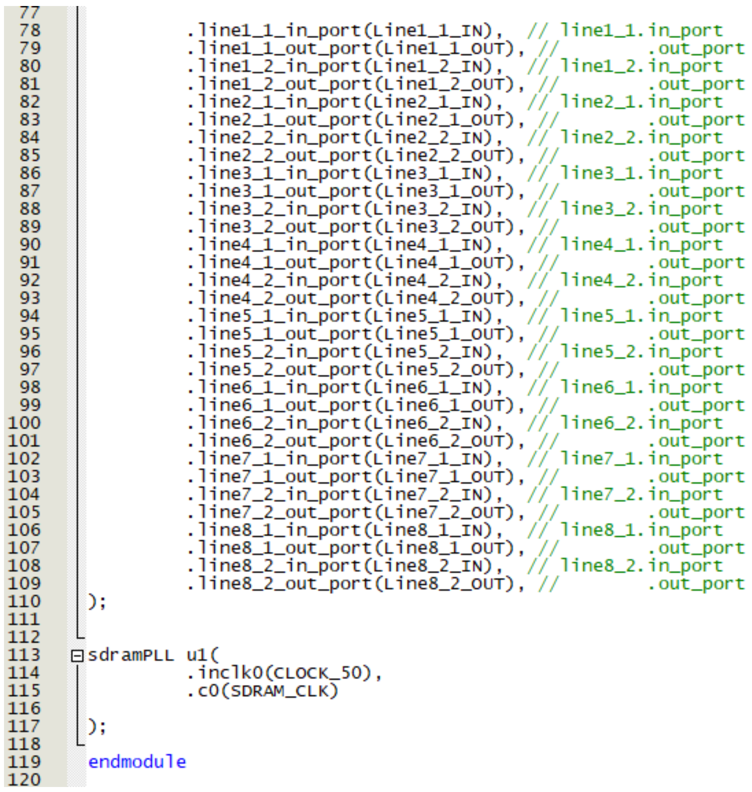
**Figure 2-3 Blank Verilog File**

3.) We will need to write code for input and outputs as well as instantiating all the modules inside of the program. The code is shown in Figure 2-4 and 2-5. The input and output names can be found inside the Nios Verilog file that was created with the system we created. This is shown in Figure 2-6.



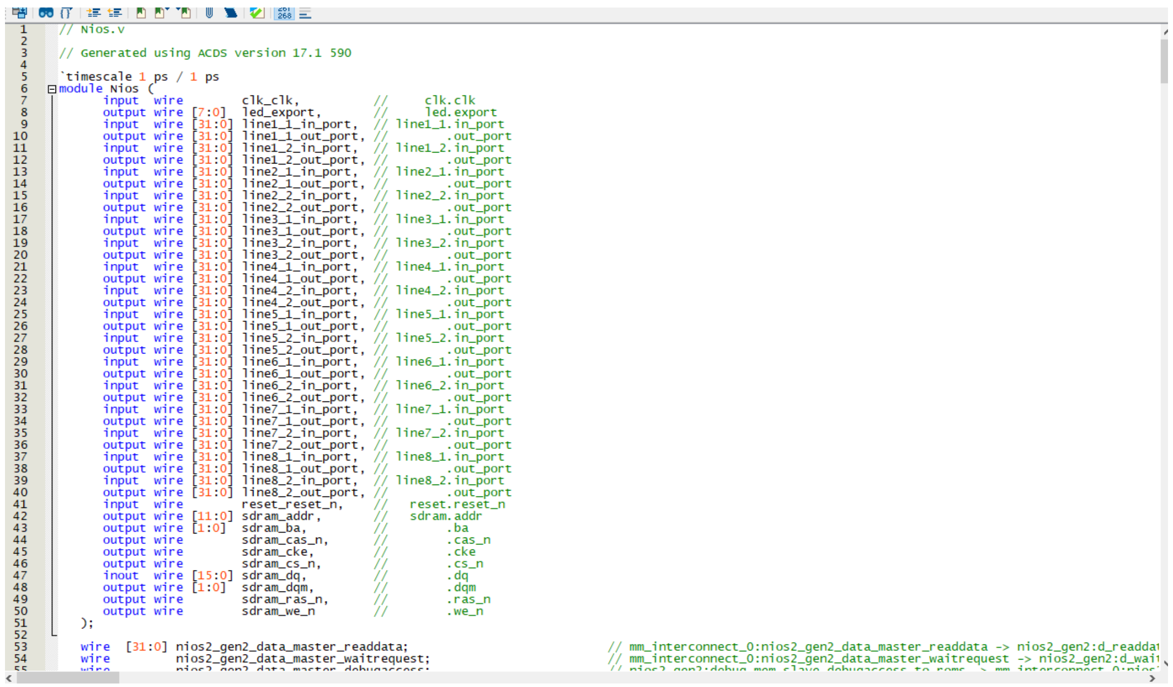
```verilog
module NiosII(
CLOCK_50,
LED,
SDRAM_CLK,
SDRAM_CKE,
SDRAM_ADDR,
SDRAM_BA,
SDRAM_CS_N,
SDRAM_CAS_N,
SDRAM_RAS_N,
SDRAM_WE_N,
SDRAM_DQ,
SDRAM_DQM
);

input CLOCK_50;
output [7:0] LED;
output [12:0] SDRAM_ADDR;
output [1:0] SDRAM_BA;
output SDRAM_CAS_N, SDRAM_RAS_N;
output SDRAM_CKE, SDRAM_CS_N, SDRAM_WE_N, SDRAM_CLK;
output [3:0] SDRAM_DQM;
inout wire [15:0] SDRAM_DQ;


reg [31:0] Line1_1_IN;
reg [31:0] Line1_2_IN;
reg [31:0] Line2_1_IN;
reg [31:0] Line2_2_IN;
reg [31:0] Line3_1_IN;
reg [31:0] Line3_2_IN;
reg [31:0] Line4_1_IN;
reg [31:0] Line4_2_IN;
reg [31:0] Line5_1_IN;
reg [31:0] Line5_2_IN;
reg [31:0] Line6_1_IN;
reg [31:0] Line6_2_IN;
reg [31:0] Line7_1_IN;
reg [31:0] Line7_2_IN;
reg [31:0] Line8_1_IN;
reg [31:0] Line8_2_IN;
reg [31:0] Line1_1_OUT;
reg [31:0] Line1_2_OUT;
reg [31:0] Line2_1_OUT;
reg [31:0] Line2_2_OUT;
reg [31:0] Line3_1_OUT;
reg [31:0] Line3_2_OUT;
reg [31:0] Line4_1_OUT;
reg [31:0] Line4_2_OUT;
reg [31:0] Line5_1_OUT;
reg [31:0] Line5_2_OUT;
reg [31:0] Line6_1_OUT;
reg [31:0] Line6_2_OUT;
reg [31:0] Line7_1_OUT;
reg [31:0] Line7_2_OUT;
reg [31:0] Line8_1_OUT;
reg [31:0] Line8_2_OUT;

wire [7:0] matrix_8 [64:1];


Nios u0(
        .clk_clk(CLOCK_50),
        .reset_reset_n(1'b1),
        .led_export(LED),

        .sdram_addr(SDRAM_ADDR),   //Address
        .sdram_ba(SDRAM_BA),     //Bank Address
        .sdram_cas_n(SDRAM_CAS_N), //Column Address Strobe
        .sdram_cke(SDRAM_CKE),    //Clock Enable
        .sdram_cs_n(SDRAM_CS_N),   //Chip Select
        .sdram_dq(SDRAM_DQ),     //Data
        .sdram_dqm(SDRAM_DQM),    //Data Mask
        .sdram_ras_n(SDRAM_RAS_N), //Row Address Strobe
        .sdram_we_n(SDRAM_WE_N),    //Write Enable
```

**Figure 2-4 Verilog Code for Nios processor**



```
            .line1_1_in_port(Line1_1_IN),   // line1_1.in_port
            .line1_1_out_port(Line1_1_OUT), //        .out_port
            .line1_2_in_port(Line1_2_IN),   // line1_2.in_port
            .line1_2_out_port(Line1_2_OUT), //        .out_port
            .line2_1_in_port(Line2_1_IN),   // line2_1.in_port
            .line2_1_out_port(Line2_1_OUT), //        .out_port
            .line2_2_in_port(Line2_2_IN),   // line2_2.in_port
            .line2_2_out_port(Line2_2_OUT), //        .out_port
            .line3_1_in_port(Line3_1_IN),   // line3_1.in_port
            .line3_1_out_port(Line3_1_OUT), //        .out_port
            .line3_2_in_port(Line3_2_IN),   // line3_2.in_port
            .line3_2_out_port(Line3_2_OUT), //        .out_port
            .line4_1_in_port(Line4_1_IN),   // line4_1.in_port
            .line4_1_out_port(Line4_1_OUT), //        .out_port
            .line4_2_in_port(Line4_2_IN),   // line4_2.in_port
            .line4_2_out_port(Line4_2_OUT), //        .out_port
            .line5_1_in_port(Line5_1_IN),   // line5_1.in_port
            .line5_1_out_port(Line5_1_OUT), //        .out_port
            .line5_2_in_port(Line5_2_IN),   // line5_2.in_port
            .line5_2_out_port(Line5_2_OUT), //        .out_port
            .line6_1_in_port(Line6_1_IN),   // line6_1.in_port
            .line6_1_out_port(Line6_1_OUT), //        .out_port
            .line6_2_in_port(Line6_2_IN),   // line6_2.in_port
            .line6_2_out_port(Line6_2_OUT), //        .out_port
            .line7_1_in_port(Line7_1_IN),   // line7_1.in_port
            .line7_1_out_port(Line7_1_OUT), //        .out_port
            .line7_2_in_port(Line7_2_IN),   // line7_2.in_port
            .line7_2_out_port(Line7_2_OUT), //        .out_port
            .line8_1_in_port(Line8_1_IN),   // line8_1.in_port
            .line8_1_out_port(Line8_1_OUT), //        .out_port
            .line8_2_in_port(Line8_2_IN),   // line8_2.in_port
            .line8_2_out_port(Line8_2_OUT), //        .out_port
);


sdramPLL u1(
            .inclk0(CLOCK_50),
            .c0(SDRAM_CLK)

);

endmodule
```

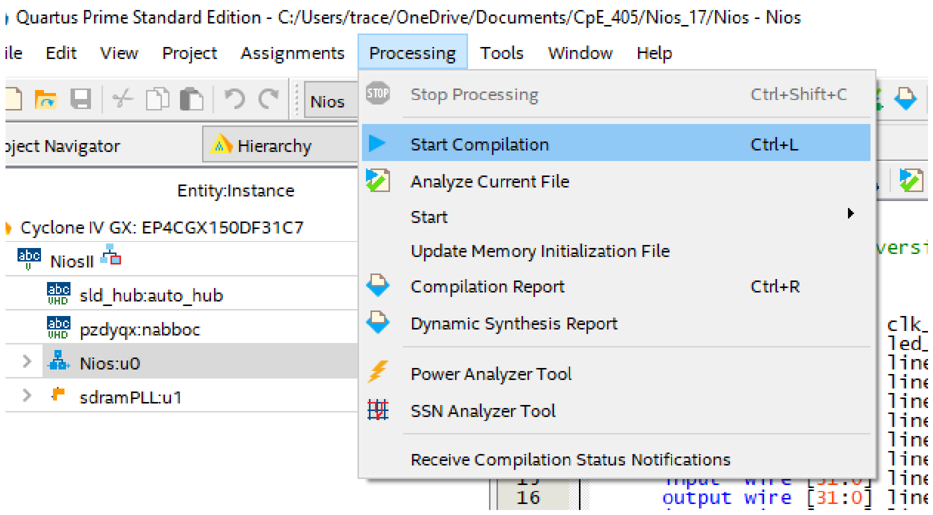
**Figure 2-5 SdramPLL is used for SDRAM Clock offset**



```
// Nios.v

// Generated using ACDS version 17.1 590

`timescale 1 ps / 1 ps
module Nios (
        input  wire        clk_clk,           //     clk.clk
        output wire [7:0]  led_export,        //     led.export
        input  wire [31:0] line1_1_in_port,   // line1_1.in_port
        output wire [31:0] line1_1_out_port,  //        .out_port
        input  wire [31:0] line1_2_in_port,   // line1_2.in_port
        output wire [31:0] line1_2_out_port,  //        .out_port
        input  wire [31:0] line2_1_in_port,   // line2_1.in_port
        output wire [31:0] line2_1_out_port,  //        .out_port
        input  wire [31:0] line2_2_in_port,   // line2_2.in_port
        output wire [31:0] line2_2_out_port,  //        .out_port
        input  wire [31:0] line3_1_in_port,   // line3_1.in_port
        output wire [31:0] line3_1_out_port,  //        .out_port
        input  wire [31:0] line3_2_in_port,   // line3_2.in_port
        output wire [31:0] line3_2_out_port,  //        .out_port
        input  wire [31:0] line4_1_in_port,   // line4_1.in_port
        output wire [31:0] line4_1_out_port,  //        .out_port
        input  wire [31:0] line4_2_in_port,   // line4_2.in_port
        output wire [31:0] line4_2_out_port,  //        .out_port
        input  wire [31:0] line5_1_in_port,   // line5_1.in_port
        output wire [31:0] line5_1_out_port,  //        .out_port
        input  wire [31:0] line5_2_in_port,   // line5_2.in_port
        output wire [31:0] line5_2_out_port,  //        .out_port
        input  wire [31:0] line6_1_in_port,   // line6_1.in_port
        output wire [31:0] line6_1_out_port,  //        .out_port
        input  wire [31:0] line6_2_in_port,   // line6_2.in_port
        output wire [31:0] line6_2_out_port,  //        .out_port
        input  wire [31:0] line7_1_in_port,   // line7_1.in_port
        output wire [31:0] line7_1_out_port,  //        .out_port
        input  wire [31:0] line7_2_in_port,   // line7_2.in_port
        output wire [31:0] line7_2_out_port,  //        .out_port
        input  wire [31:0] line8_1_in_port,   // line8_1.in_port
        output wire [31:0] line8_1_out_port,  //        .out_port
        input  wire [31:0] line8_2_in_port,   // line8_2.in_port
        output wire [31:0] line8_2_out_port,  //        .out_port
        input  wire        reset_reset_n,     //   reset.reset_n
        output wire [11:0] sdram_addr,        //   sdram.addr
        output wire [1:0]  sdram_ba,          //        .ba
        output wire        sdram_cas_n,       //        .cas_n
        output wire        sdram_cke,         //        .cke
        output wire        sdram_cs_n,        //        .cs_n
        inout  wire [15:0] sdram_dq,          //        .dq
        output wire [1:0]  sdram_dqm,         //        .dqm
        output wire        sdram_ras_n,       //        .ras_n
        output wire        sdram_we_n         //        .we_n
    );

    wire  [31:0] nios2_gen2_data_master_readdata;                    // mm_interconnect_0:nios2_gen2_data_master_readdata -> nios2_gen2:d_readdat
    wire         nios2_gen2_data_master_waitrequest;                 // mm_interconnect_0:nios2_gen2_data_master_waitrequest -> nios2_gen2:d_wait
```

**Figure 2-6 Verilog module for Nios**

4.) We will now need to compile the program. **Choose Processing → Start Compilation** as shown in Figures 2-7 and 2-8.



```
16        output wire [31:0] line
```

**Figure 2-7 Compiling Project**

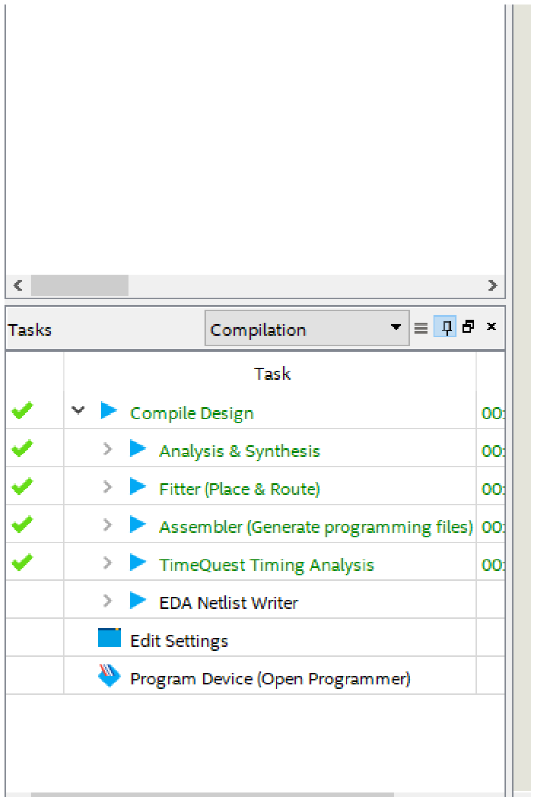
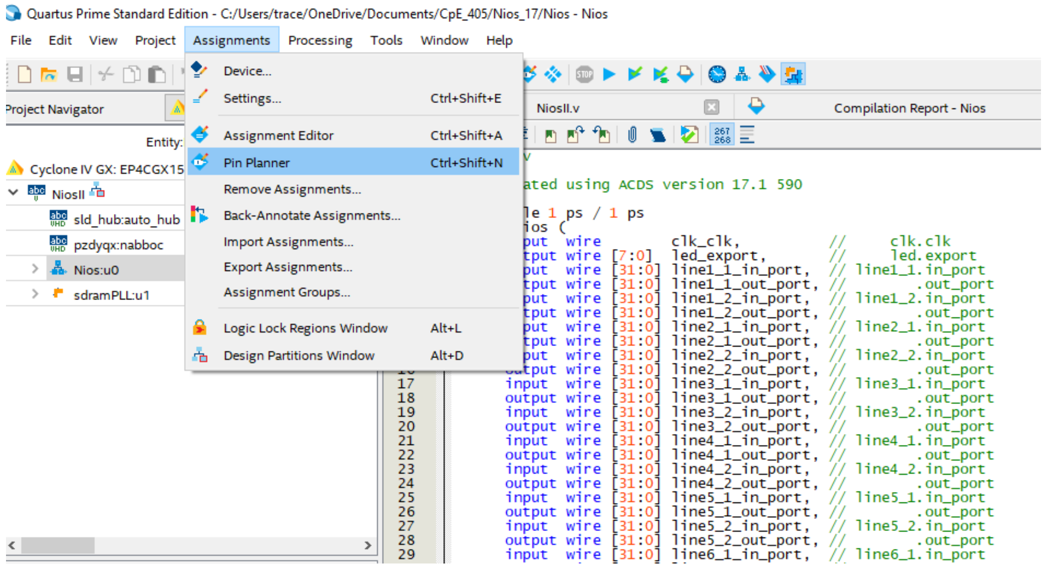


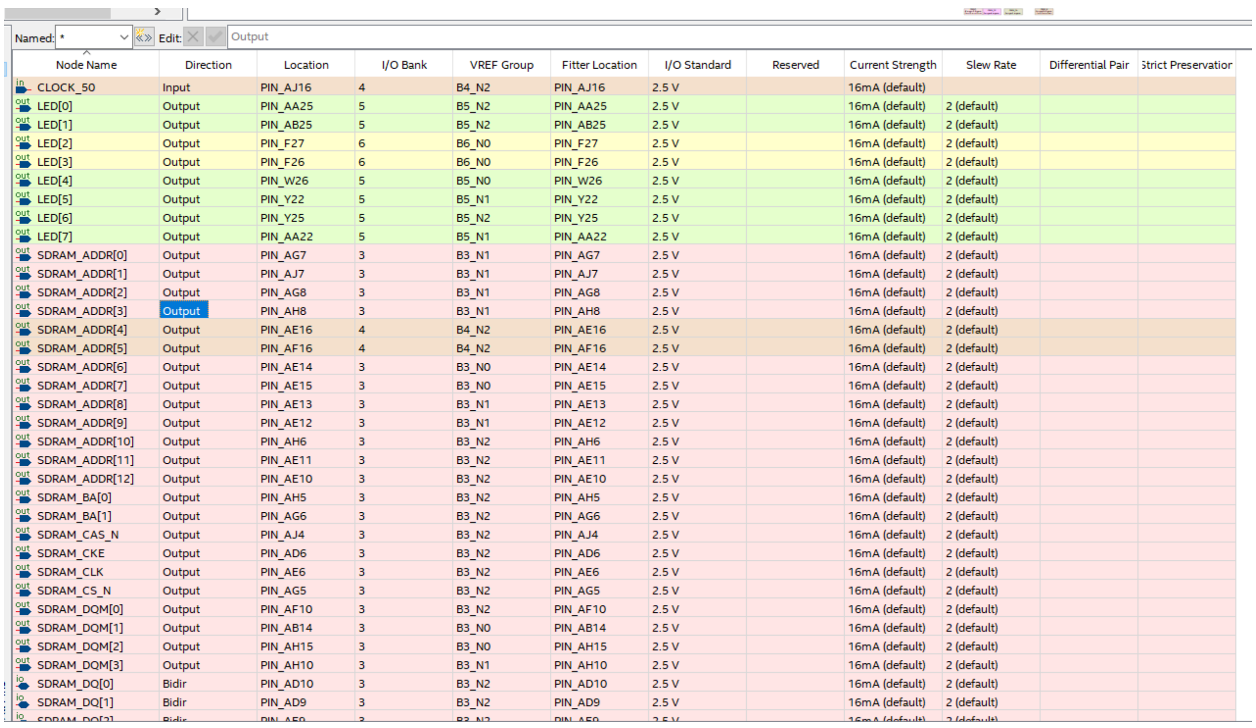
**Figure 2-8 Compiling is complete**

5.) You will need to assign pins to Clocks and any external input/output pins like LED's or Switches. You can do this through **Assignments → Pin Planner** as shown in Figures 2-9 and 2-10. We will then re-compile the project as shown above.



```
ated using ACDS version 17.1 590

le 1 ps / 1 ps
ios (
put   wire         clk_clk,            //      clk.clk
tput  wire [7:0]   led_export,         //      led.export
put   wire [31:0]  line1_1_in_port,    // line1_1.in_port
tput  wire [31:0]  line1_1_out_port,   //        .out_port
put   wire [31:0]  line1_2_in_port,    // line1_2.in_port
tput  wire [31:0]  line1_2_out_port,   //        .out_port
put   wire [31:0]  line2_1_in_port,    // line2_1.in_port
tput  wire [31:0]  line2_1_out_port,   //        .out_port
put   wire [31:0]  line2_2_in_port,    // line2_2.in_port
tput  wire [31:0]  line2_2_out_port,   //        .out_port
input  wire [31:0] line3_1_in_port,    // line3_1.in_port
output wire [31:0] line3_1_out_port,   //        .out_port
input  wire [31:0] line3_2_in_port,    // line3_2.in_port
output wire [31:0] line3_2_out_port,   //        .out_port
input  wire [31:0] line4_1_in_port,    // line4_1.in_port
output wire [31:0] line4_1_out_port,   //        .out_port
input  wire [31:0] line4_2_in_port,    // line4_2.in_port
output wire [31:0] line4_2_out_port,   //        .out_port
input  wire [31:0] line5_1_in_port,    // line5_1.in_port
output wire [31:0] line5_1_out_port,   //        .out_port
input  wire [31:0] line5_2_in_port,    // line5_2.in_port
output wire [31:0] line5_2_out_port,   //        .out_port
input  wire [31:0] line6_1_in_port,    // line6_1.in_port
```

**Figure 2-9**



| Node Name | Direction | Location | I/O Bank | VREF Group | Fitter Location | I/O Standard | Reserved | Current Strength | Slew Rate | Differential Pair | Strict Preservation |
|---|---|---|---|---|---|---|---|---|---|---|---|
| CLOCK_50 | Input | PIN_AJ16 | 4 | B4_N2 | PIN_AJ16 | 2.5 V | | 16mA (default) | | | |
| LED[0] | Output | PIN_AA25 | 5 | B5_N2 | PIN_AA25 | 2.5 V | | 16mA (default) | 2 (default) | | |
| LED[1] | Output | PIN_AB25 | 5 | B5_N2 | PIN_AB25 | 2.5 V | | 16mA (default) | 2 (default) | | |
| LED[2] | Output | PIN_F27 | 6 | B6_N0 | PIN_F27 | 2.5 V | | 16mA (default) | 2 (default) | | |
| LED[3] | Output | PIN_F26 | 6 | B6_N0 | PIN_F26 | 2.5 V | | 16mA (default) | 2 (default) | | |
| LED[4] | Output | PIN_W26 | 5 | B5_N0 | PIN_W26 | 2.5 V | | 16mA (default) | 2 (default) | | |
| LED[5] | Output | PIN_Y22 | 5 | B5_N1 | PIN_Y22 | 2.5 V | | 16mA (default) | 2 (default) | | |
| LED[6] | Output | PIN_Y25 | 5 | B5_N2 | PIN_Y25 | 2.5 V | | 16mA (default) | 2 (default) | | |
| LED[7] | Output | PIN_AA22 | 5 | B5_N1 | PIN_AA22 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_ADDR[0] | Output | PIN_AG7 | 3 | B3_N1 | PIN_AG7 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_ADDR[1] | Output | PIN_AJ7 | 3 | B3_N1 | PIN_AJ7 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_ADDR[2] | Output | PIN_AG8 | 3 | B3_N1 | PIN_AG8 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_ADDR[3] | Output | PIN_AH8 | 3 | B3_N1 | PIN_AH8 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_ADDR[4] | Output | PIN_AE16 | 4 | B4_N2 | PIN_AE16 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_ADDR[5] | Output | PIN_AF16 | 4 | B4_N2 | PIN_AF16 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_ADDR[6] | Output | PIN_AE14 | 3 | B3_N0 | PIN_AE14 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_ADDR[7] | Output | PIN_AE15 | 3 | B3_N0 | PIN_AE15 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_ADDR[8] | Output | PIN_AE13 | 3 | B3_N1 | PIN_AE13 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_ADDR[9] | Output | PIN_AE12 | 3 | B3_N1 | PIN_AE12 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_ADDR[10] | Output | PIN_AH6 | 3 | B3_N2 | PIN_AH6 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_ADDR[11] | Output | PIN_AE11 | 3 | B3_N2 | PIN_AE11 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_ADDR[12] | Output | PIN_AE10 | 3 | B3_N2 | PIN_AE10 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_BA[0] | Output | PIN_AH5 | 3 | B3_N2 | PIN_AH5 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_BA[1] | Output | PIN_AG6 | 3 | B3_N2 | PIN_AG6 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_CAS_N | Output | PIN_AJ4 | 3 | B3_N2 | PIN_AJ4 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_CKE | Output | PIN_AD6 | 3 | B3_N2 | PIN_AD6 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_CLK | Output | PIN_AE6 | 3 | B3_N2 | PIN_AE6 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_CS_N | Output | PIN_AG5 | 3 | B3_N2 | PIN_AG5 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_DQM[0] | Output | PIN_AF10 | 3 | B3_N2 | PIN_AF10 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_DQM[1] | Output | PIN_AB14 | 3 | B3_N0 | PIN_AB14 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_DQM[2] | Output | PIN_AH15 | 3 | B3_N0 | PIN_AH15 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_DQM[3] | Output | PIN_AH10 | 3 | B3_N1 | PIN_AH10 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_DQ[0] | Bidir | PIN_AD10 | 3 | B3_N2 | PIN_AD10 | 2.5 V | | 16mA (default) | 2 (default) | | |
| SDRAM_DQ[1] | Bidir | PIN_AD9 | 3 | B3_N2 | PIN_AD9 | 2.5 V | | 16mA (default) | 2 (default) | | |

**Figure 2-10 Assigning pins to nodes**

6.) Now we are ready to download the program to the board. To do this we will choose **Tools → Programmer** as shown in Figure 2-11.

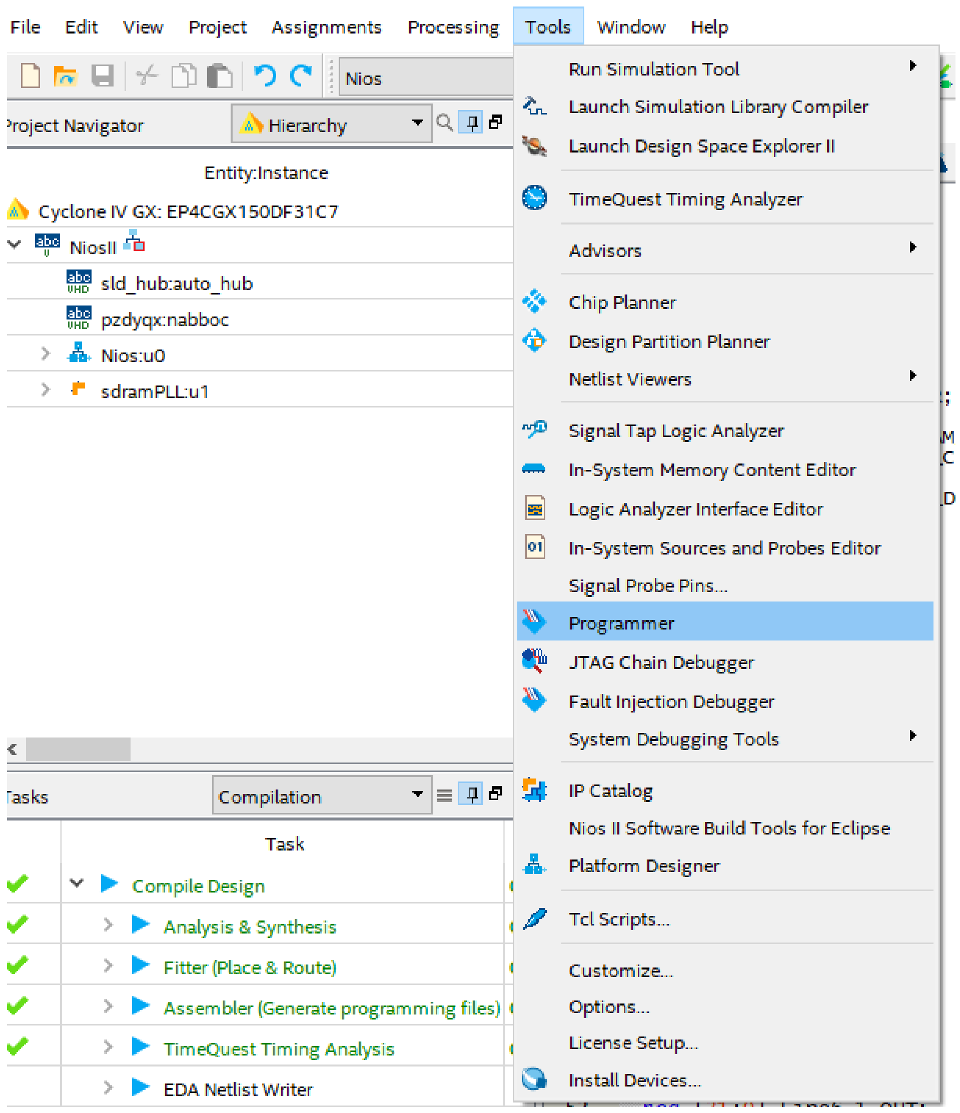
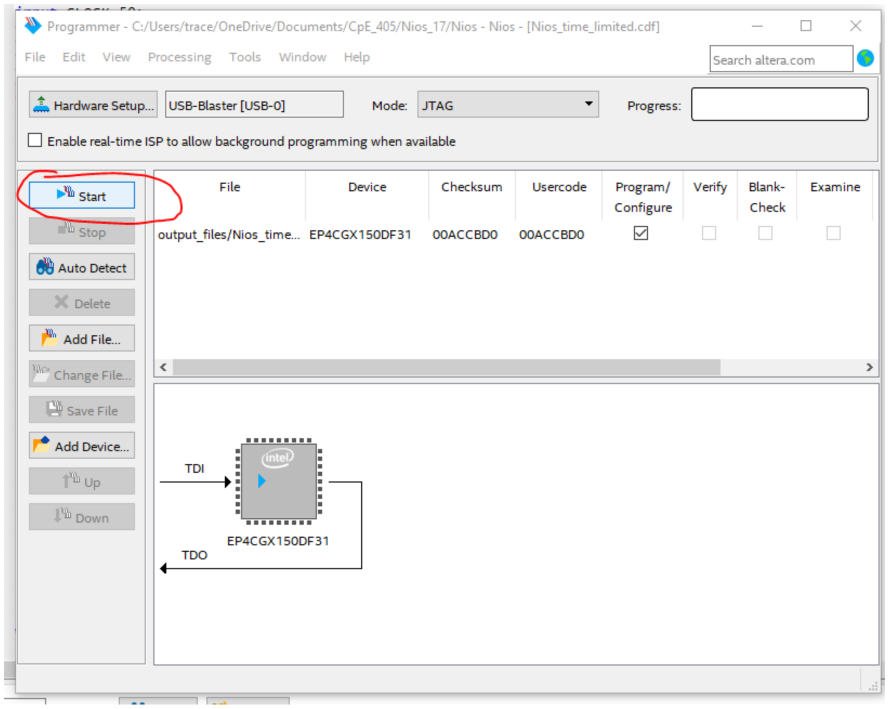


**Figure 2-11 Quartus Programmer**

7.) Now we will select our file and device we want to program and then program the board as shown in Figure 2-12. Make sure your board is connected to your computer and it is turned on.



Programmer - C:/Users/trace/OneDrive/Documents/CpE_405/Nios_17/Nios - Nios - [Nios_time_limited.cdf]

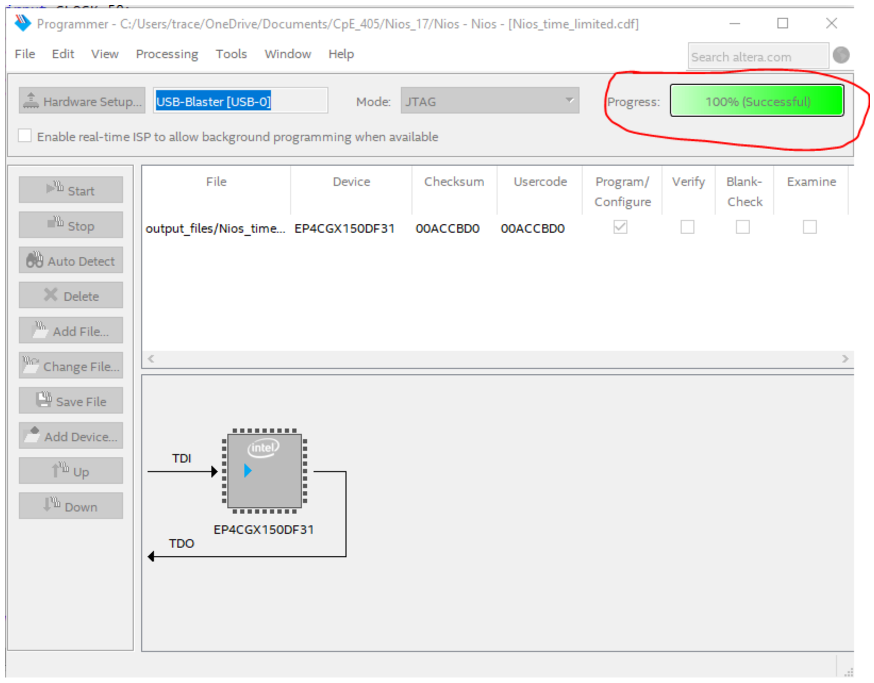
USB-Blaster [USB-0] Mode: JTAG Progress:

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

| File | Device | Checksum | Usercode | Program/ Configure | Verify | Blank-Check | Examine |
|---|---|---|---|---|---|---|---|
| output_files/Nios_time... | EP4CGX150DF31 | 00ACCBD0 | 00ACCBD0 | ☑ | ☐ | ☐ | ☐ |

Figure 2-12 Quartus Programmer

8.) Once programmed, the progress meter should be at 100% shown in Figure 2-13. The FPGA is now configured with the Nios System. Now we need to write our C program to execute.



Programmer - C:/Users/trace/OneDrive/Documents/CpE_405/Nios_17/Nios - Nios - [Nios_time_limited.cdf]

USB-Blaster [USB-0] Mode: JTAG Progress: 100% (Successful)

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

| File | Device | Checksum | Usercode | Program/ Configure | Verify | Blank-Check | Examine |
|---|---|---|---|---|---|---|---|
| output_files/Nios_time... | EP4CGX150DF31 | 00ACCBD0 | 00ACCBD0 | ☑ | ☐ | ☐ | ☐ |

Figure 2-13 FPGA is programmed with Nios System

# Chapter 3 *NIOS II Software Build Tools for Eclipse*

This Chapter covers build flow of Nios II C coded software program.

The Nios II Software Build Tools (SBT) for Eclipse is an easy-to-use graphical user interface (GUI) that automates build and makefile management. The Nios II SBT for Eclipse integrates a text editor, debugger, the BSP editor, the Nios II flash programmer and the Quartus II Programmer. The included example software application templates make it easy for new software programmers to get started quickly. In this section you will use the Nios II SBT for Eclipse to compile a simple C language example software program to run on the Nios II standard system configured onto the FPGA on your development board. You will create a new software project, build it, and run it on the target hardware. You will also edit the project, re-build it, and set up a debug session.

## 3.1 Creating a simple Hello World Example Project

In this section we will be creating a simple C project that will print Hello World! Onto the console.

1.) Choose Tools → NIOS II Software Build Tools for Eclipse shown in Figure 3-1.

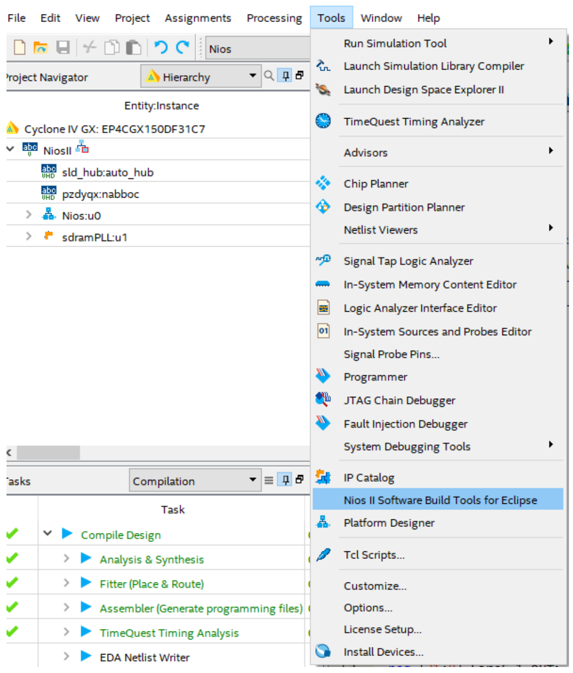
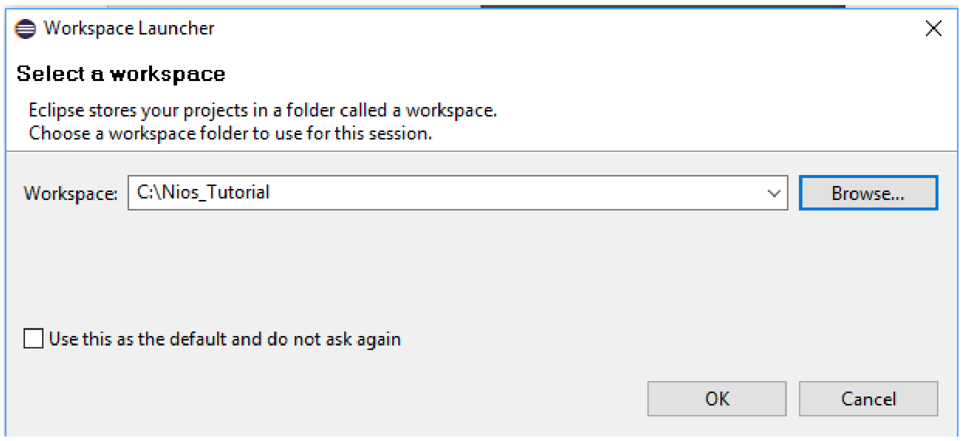


**Figure 3-1**

2.) You will then select your workspace you want to work in as shown in Figure 3-2.


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:\Nios_Tutorial

**Figure 3-2**

3.) Next will be creating the Hello World Example Project. **Choose File → New → Nios II Application and BSP from Template** as shown in Figure 3-3.

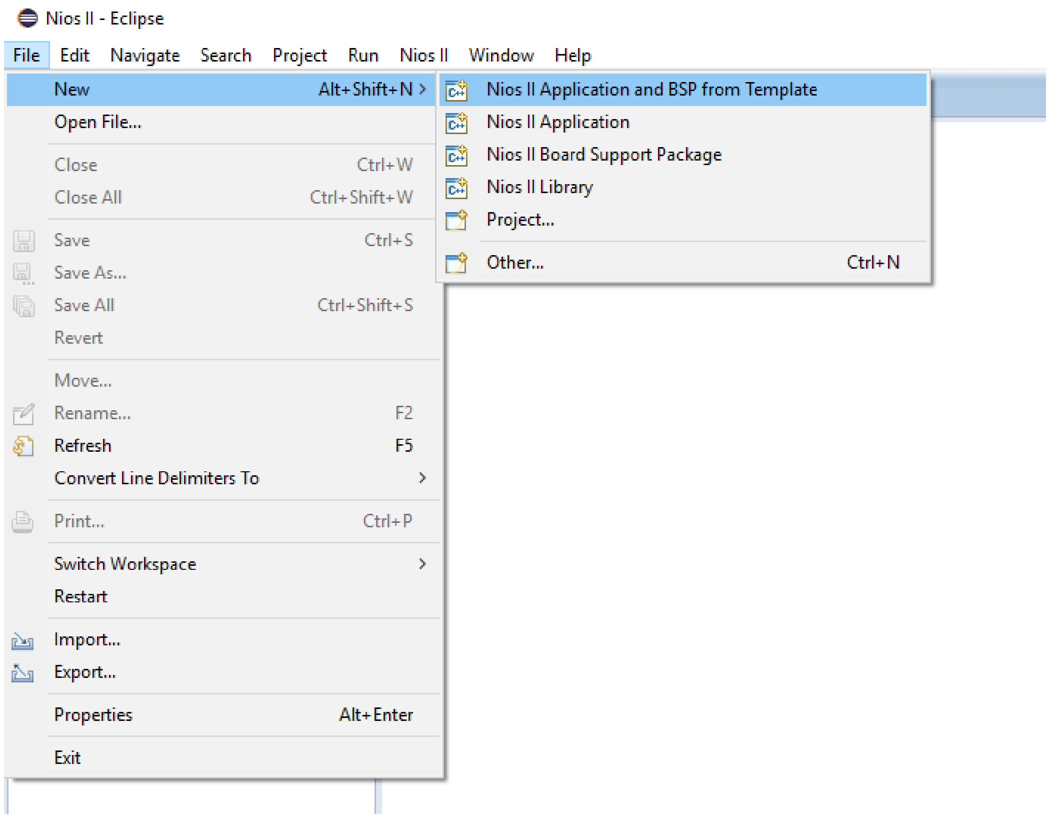
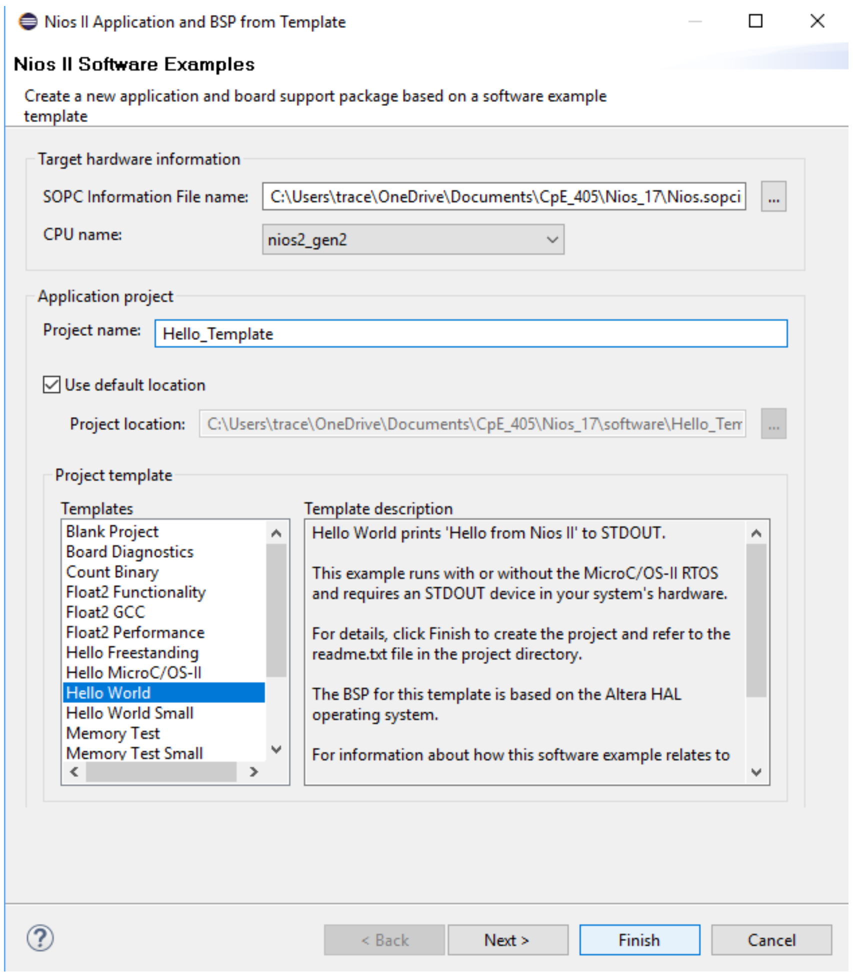


**Figure 3-3**

4.) Now we will choose your SOPC File which will be located in the directory that you created your Qsys in. We will then create our name and select the Hello World Template and select Finish.



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:\Users\trace\OneDrive\Documents\CpE_405\Nios_17\Nios.sopci

CPU name: nios2_gen2

Application project

Project name: Hello_Template

☑ Use default location

Project location: C:\Users\trace\OneDrive\Documents\CpE_405\Nios_17\software\Hello_Tem

Project template

Templates

- Blank Project
- 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

Hello World prints 'Hello from Nios II' to STDOUT.

This example runs with or without the MicroC/OS-II RTOS and requires an STDOUT device in your system's hardware.

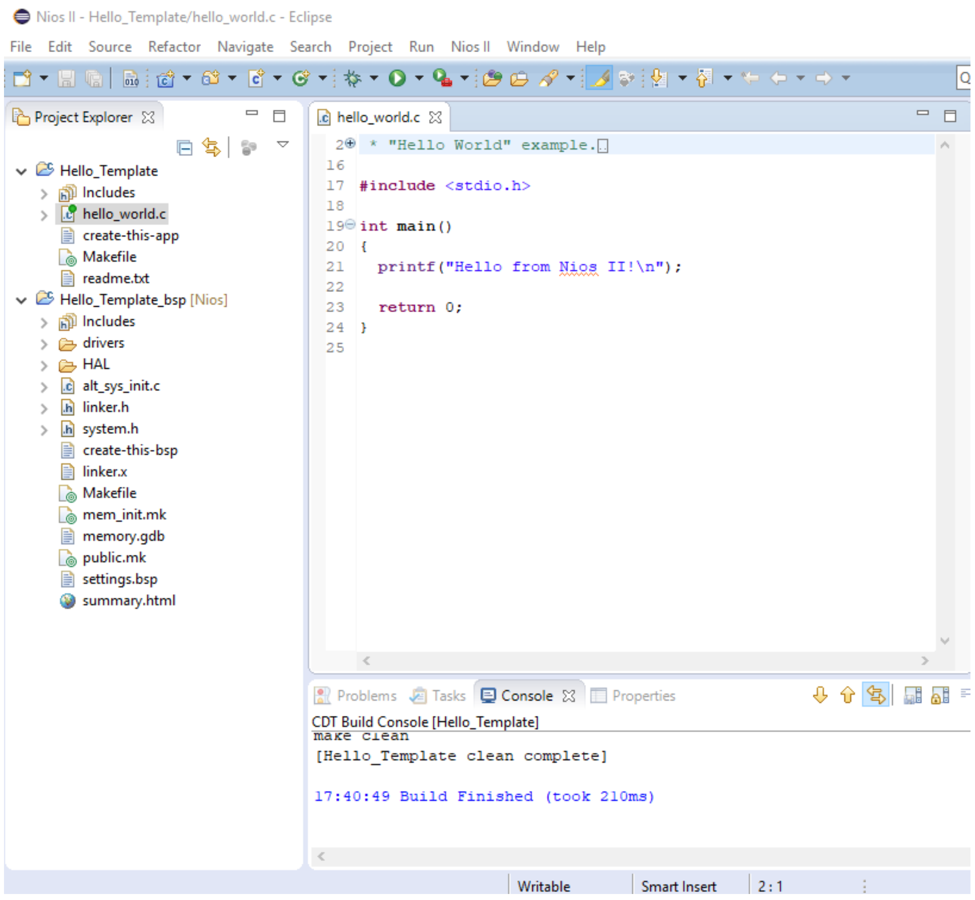
For details, click Finish to create the project and refer to the readme.txt file in the project directory.

The BSP for this template is based on the Altera HAL operating system.

For information about how this software example relates to

**Figure 3-4 Creating Hello World Template**

5.) The system will create everything you will need. You can click on the hello_world.c file to see the C code that was created.



```
 * "Hello World" example.

#include <stdio.h>

int main()
{
  printf("Hello from Nios II!\n");

  return 0;
}
```

```
CDT Build Console [Hello_Template]
make clean
[Hello_Template clean complete]

17:40:49 Build Finished (took 210ms)
```

**Figure 3-5 C code for Hello World**

6.) Now we will need to build and run this program. Right click on the **Hello_Template → Build Project** as shown in Figure 3-6. Next we will run this program by right clicking again on **Hello_Template → Run As → Nios II Hardware** as shown in Figure 3-7.

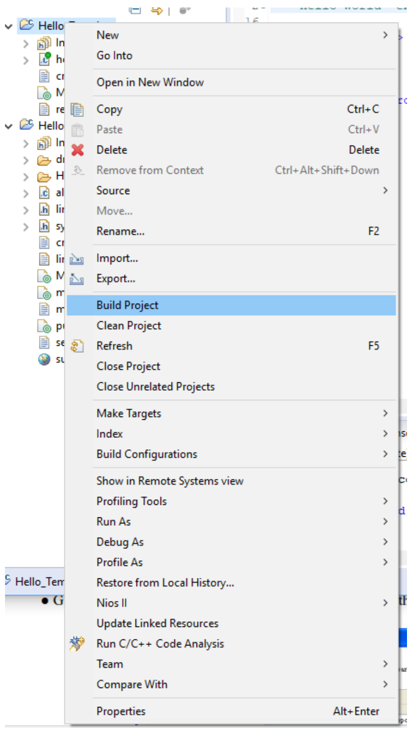


**Figure 3-6 Building Project**

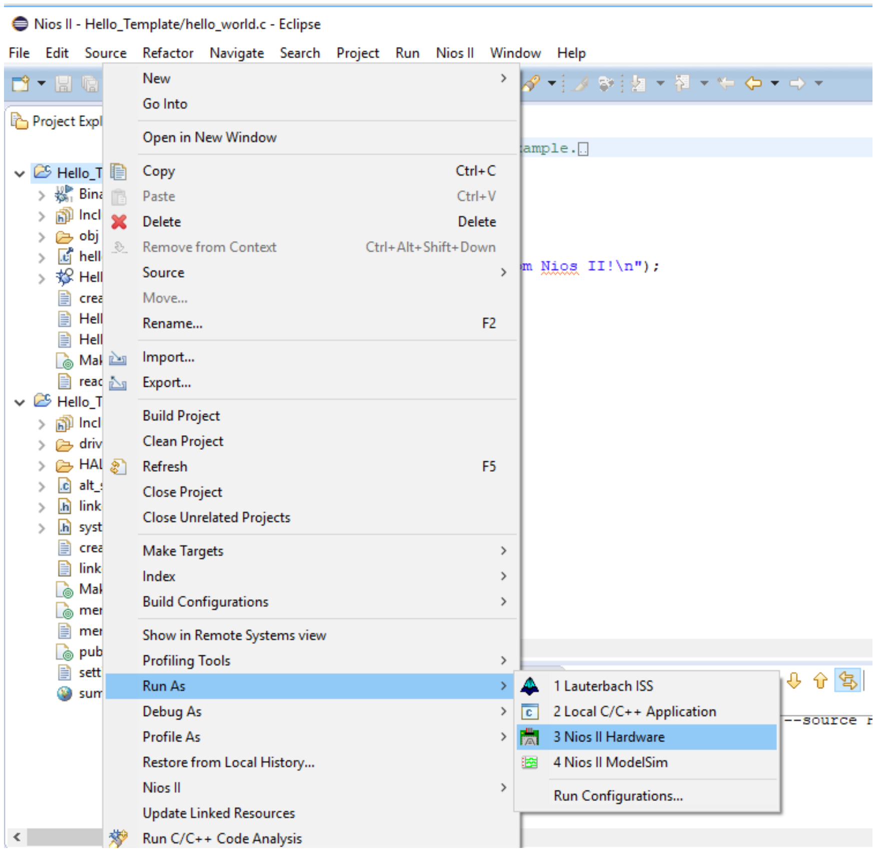
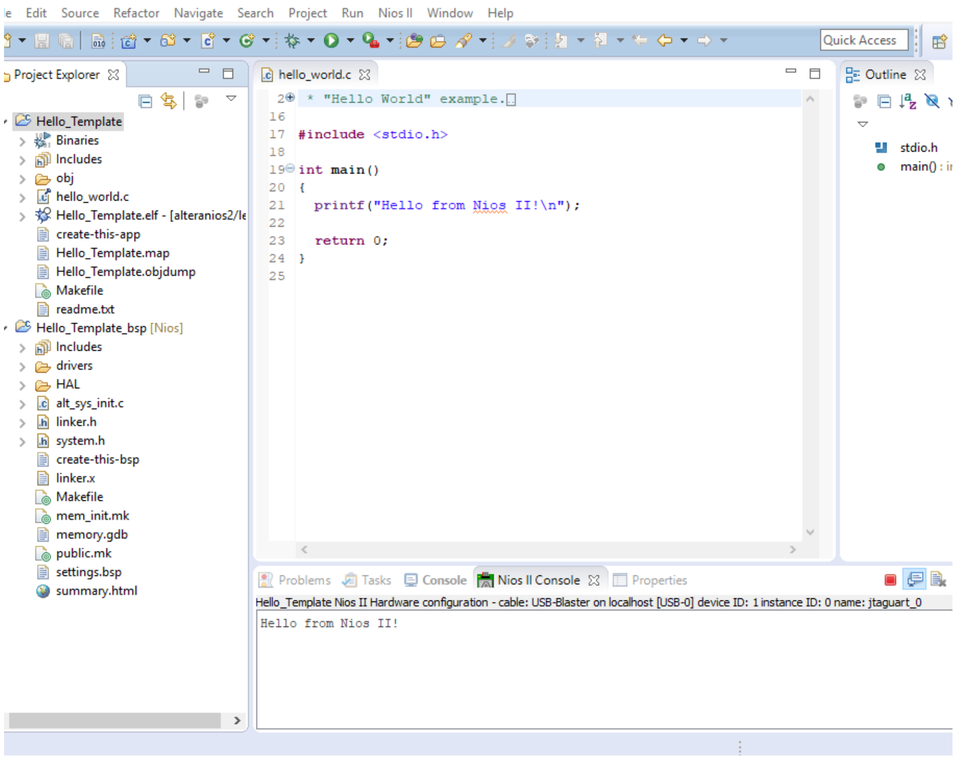


**Figure 3-7 Running Nios Project**

7.) You should now see in the console a message displaying “Hello From Nios II!”.



```
 * "Hello World" example.

#include <stdio.h>

int main()
{
  printf("Hello from Nios II!\n");

  return 0;
}
```

```
Hello_Template Nios II Hardware configuration - cable: USB-Blaster on localhost [USB-0] device ID: 1 instance ID: 0 name: jtaguart_0
Hello from Nios II!
```

**Figure 3-8**

## 3.2 DCT and Quantization with Nios Processor

1.) We are now going to edit the project we just created for hello world to do DCT on an 8x8 matrix. First, we will need to change the code. The DCT code is shown below.

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

#define pi 3.142857
const int m = 8, n = 8;


// Function to find discrete cosine transform and print it
int dctTransform(int matrix[][n], int Qmatrix[][n])
{
    int i, j, k, l;
```

```c
    // dct will store the discrete cosine transform
    float dct[m][n];
    float Qdct[m][n];

    float ci, cj, dct1, sum;

    for (i = 0; i < m; i++) {
        for (j = 0; j < n; j++) {

            // ci and cj depends on frequency as well as
            // number of row and columns of specified matrix
            if (i == 0)
                ci = 1 / sqrt(m);
            else
                ci = sqrt(2) / sqrt(m);
            if (j == 0)
                cj = 1 / sqrt(n);
            else
                cj = sqrt(2) / sqrt(n);

            // sum will temporarily store the sum of
            // cosine signals
            sum = 0;
            for (k = 0; k < m; k++) {
                for (l = 0; l < n; l++) {
                  matrix[k][l] = matrix[k][l] - 128;
                    dct1 = matrix[k][l] *
                           cos((2 * k + 1) * i * pi / (2 * m)) *
                           cos((2 * l + 1) * j * pi / (2 * n));
                    sum = sum + dct1;
                }
            }
            Qdct[i][j] = ci * cj * sum;
            dct[i][j] = Qdct[i][j] / Qmatrix[i][j];

        }
    }

    for (i = 0; i < m; i++) {
        for (j = 0; j < n; j++) {
            printf("%f\t", dct[i][j]);
        }
        printf("\n");
    }
    return 0;
}
// Driver code
int main()
{
    int matrix[8][8];/* = { { 255, 255, 255, 255, 255, 255, 255, 255 },
                          { 255, 255, 255, 255, 255, 255, 255, 255 },
                          { 255, 255, 255, 255, 255, 255, 255, 255 },
                          { 255, 255, 255, 255, 255, 255, 255, 255 },
                          { 255, 255, 255, 255, 255, 255, 255, 255 },
                          { 255, 255, 255, 255, 255, 255, 255, 255 },
                          { 255, 255, 255, 255, 255, 255, 255, 255 },
                          { 255, 255, 255, 255, 255, 255, 255, 255 } };*/
```

```c
int Qmatrix[8][8] = { { 16, 11, 10, 16, 24, 40, 51, 61 },
                     { 12, 12, 14, 19, 26, 58, 60, 55 },
                     { 14, 13, 16, 24, 40, 57, 69, 56 },
                     { 14, 17, 22, 29, 51, 87, 80, 62 },
                     { 18, 22, 37, 56, 68, 109, 103, 77 },
                     { 24, 35, 55, 64, 81, 104, 113, 92 },
                     { 49, 64, 78, 87, 103, 121, 120, 101 },
                     { 72, 92, 95, 98, 112, 100, 103, 99 } };


int temp;

int t, r;

  for(t = 0; t < 8; t++)
  {
        for(r = 0; r < 8; r++)
        {
              if(t == 0 && r < 4)
              {
                    matrix[t][r] = IORD_8DIRECT(LINE1_1_BASE, r);
              }
              if(t == 0 && r >= 4)
              {
                    matrix[t][r] = IORD_8DIRECT(LINE1_2_BASE, r-4);
              }
              if(t == 1 && r < 4)
              {
                    matrix[t][r] = IORD_8DIRECT(LINE2_1_BASE, r);
              }
              if(t == 1 && r >= 4)
              {
                    matrix[t][r] = IORD_8DIRECT(LINE2_2_BASE, r-4);
              }
              if(t == 2 && r < 4)
              {
                    matrix[t][r] = IORD_8DIRECT(LINE3_1_BASE, r);
              }
              if(t == 2 && r >= 4)
              {
                    matrix[t][r] = IORD_8DIRECT(LINE3_2_BASE, r-4);
              }
              if(t == 3 && r < 4)
              {
                    matrix[t][r] = IORD_8DIRECT(LINE4_1_BASE, r);
              }
              if(t == 3 && r >= 4)
              {
                    matrix[t][r] = IORD_8DIRECT(LINE4_2_BASE, r-4);
              }
              if(t == 4 && r < 4)
              {
                    matrix[t][r] = IORD_8DIRECT(LINE5_1_BASE, r);
              }
              if(t == 4 && r >= 4)
              {
```

```
                                matrix[t][r] = IORD_8DIRECT(LINE5_2_BASE, r-4);
                        }
                        if(t == 5 && r < 4)
                        {
                                matrix[t][r] = IORD_8DIRECT(LINE6_1_BASE, r);
                        }
                        if(t == 5 && r >= 4)
                        {
                                matrix[t][r] = IORD_8DIRECT(LINE6_2_BASE, r-4);
                        }
                        if(t == 6 && r < 4)
                        {
                                matrix[t][r] = IORD_8DIRECT(LINE7_1_BASE, r);
                        }
                        if(t == 6 && r >= 4)
                        {
                                matrix[t][r] = IORD_8DIRECT(LINE7_2_BASE, r-4);
                        }
                        if(t == 7 && r < 4)
                        {
                                matrix[t][r] = IORD_8DIRECT(LINE8_1_BASE, r);
                        }
                        if(t == 7 && r >= 4)
                        {
                                matrix[t][r] = IORD_8DIRECT(LINE8_2_BASE, r-4);
                        }
                }
        }
          dctTransform(matrix, Qmatrix);


    return 0;
}
```

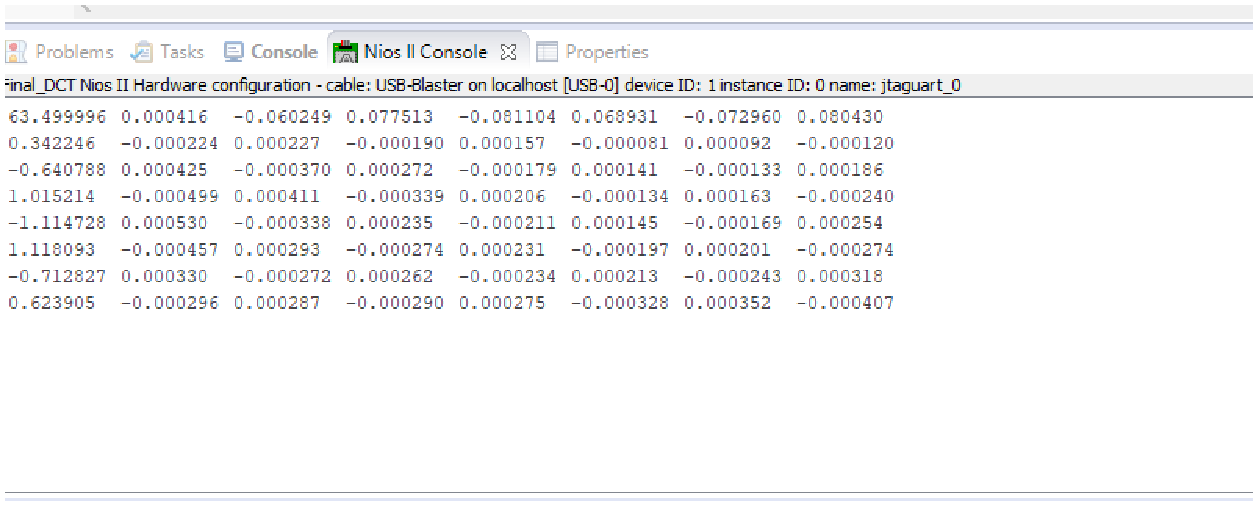
2.) Now we will go back to the Verilog file and change that to send data to the Nios Processor. Our new code is shown below. We generate a 8x8 matrix with all values of 255 for testing and put those values into our PIO's to be sent into the Nios processor. We will need to recompile this program as well.

```
module NiosII(
CLOCK_50,
LED,
SDRAM_CLK,
SDRAM_CKE,
SDRAM_ADDR,
SDRAM_BA,
SDRAM_CS_N,
SDRAM_CAS_N,
SDRAM_RAS_N,
SDRAM_WE_N,
SDRAM_DQ,
SDRAM_DQM
);

input CLOCK_50;
output [7:0] LED;
output [12:0] SDRAM_ADDR;
output [1:0] SDRAM_BA;
output SDRAM_CAS_N, SDRAM_RAS_N;
output SDRAM_CKE, SDRAM_CS_N, SDRAM_WE_N, SDRAM_CLK;
output [3:0] SDRAM_DQM;
inout wire [15:0] SDRAM_DQ;


reg [31:0] Line1_1_IN;
reg [31:0] Line1_2_IN;
reg [31:0] Line2_1_IN;
reg [31:0] Line2_2_IN;
reg [31:0] Line3_1_IN;
reg [31:0] Line3_2_IN;
reg [31:0] Line4_1_IN;
reg [31:0] Line4_2_IN;
reg [31:0] Line5_1_IN;
reg [31:0] Line5_2_IN;
reg [31:0] Line6_1_IN;
reg [31:0] Line6_2_IN;
reg [31:0] Line7_1_IN;
reg [31:0] Line7_2_IN;
reg [31:0] Line8_1_IN;
reg [31:0] Line8_2_IN;

reg [31:0] Line1_1_OUT;
reg [31:0] Line1_2_OUT;
reg [31:0] Line2_1_OUT;
reg [31:0] Line2_2_OUT;
reg [31:0] Line3_1_OUT;
reg [31:0] Line3_2_OUT;
reg [31:0] Line4_1_OUT;
reg [31:0] Line4_2_OUT;
reg [31:0] Line5_1_OUT;
reg [31:0] Line5_2_OUT;
reg [31:0] Line6_1_OUT;
reg [31:0] Line6_2_OUT;
reg [31:0] Line7_1_OUT;
reg [31:0] Line7_2_OUT;
reg [31:0] Line8_1_OUT;
reg [31:0] Line8_2_OUT;

wire [7:0] matrix_8 [64:1];


Nios u0(
                        .clk_clk(CLOCK_50),
                        .reset_reset_n(1'b1),
                        .led_export(LED),

                        .sdram_addr(SDRAM_ADDR),  //Address
                        .sdram_ba(SDRAM_BA),    //Bank Address
                        .sdram_cas_n(SDRAM_CAS_N), //Column Address Strobe
                        .sdram_cke(SDRAM_CKE),   //Clock Enable
                        .sdram_cs_n(SDRAM_CS_N),  //Chip Select
                        .sdram_dq(SDRAM_DQ),    //Data
                        .sdram_dqm(SDRAM_DQM),   //Data Mask
                        .sdram_ras_n(SDRAM_RAS_N), //Row Address Strobe
                        .sdram_we_n(SDRAM_WE_N),   //Write Enable

                        .line1_1_in_port(Line1_1_IN),  // line1_1.in_port
                        .line1_1_out_port(Line1_1_OUT), //        .out_port
                        .line1_2_in_port(Line1_2_IN),  // line1_2.in_port
                        .line1_2_out_port(Line1_2_OUT), //        .out_port
                        .line2_1_in_port(Line2_1_IN),  // line2_1.in_port
                        .line2_1_out_port(Line2_1_OUT), //        .out_port
                        .line2_2_in_port(Line2_2_IN),  // line2_2.in_port
                        .line2_2_out_port(Line2_2_OUT), //        .out_port
                        .line3_1_in_port(Line3_1_IN),  // line3_1.in_port
                        .line3_1_out_port(Line3_1_OUT), //        .out_port
                        .line3_2_in_port(Line3_2_IN),  // line3_2.in_port
                        .line3_2_out_port(Line3_2_OUT), //        .out_port
                        .line4_1_in_port(Line4_1_IN),  // line4_1.in_port
                        .line4_1_out_port(Line4_1_OUT), //        .out_port
                        .line4_2_in_port(Line4_2_IN),  // line4_2.in_port
                        .line4_2_out_port(Line4_2_OUT), //        .out_port
                        .line5_1_in_port(Line5_1_IN),  // line5_1.in_port
                        .line5_1_out_port(Line5_1_OUT), //        .out_port
                        .line5_2_in_port(Line5_2_IN),  // line5_2.in_port
                        .line5_2_out_port(Line5_2_OUT), //        .out_port
                        .line6_1_in_port(Line6_1_IN),  // line6_1.in_port
                        .line6_1_out_port(Line6_1_OUT), //        .out_port
```

```
                                .line6_2_in_port(Line6_2_IN),  // line6_2.in_port
                                .line6_2_out_port(Line6_2_OUT), //       .out_port
                                .line7_1_in_port(Line7_1_IN),  // line7_1.in_port
                                .line7_1_out_port(Line7_1_OUT), //       .out_port
                                .line7_2_in_port(Line7_2_IN),  // line7_2.in_port
                                .line7_2_out_port(Line7_2_OUT), //       .out_port
                                .line8_1_in_port(Line8_1_IN),  // line8_1.in_port
                                .line8_1_out_port(Line8_1_OUT), //       .out_port
                                .line8_2_in_port(Line8_2_IN),  // line8_2.in_port
                                .line8_2_out_port(Line8_2_OUT), //       .out_port
);


sdramPLL          u1(
                                .inclk0(CLOCK_50),
                                .c0(SDRAM_CLK)

);



genvar f;
generate
for(f=1; f<65; f=f+1) begin: kForl
begin
assign matrix_8[f] = 255;
end
end
endgenerate


always @(posedge SDRAM_CLK) begin



Line1_1_IN <= {matrix_8[1], matrix_8[2], matrix_8[3], matrix_8[4]};
Line1_2_IN <= {matrix_8[5], matrix_8[6], matrix_8[7], matrix_8[8]};

Line2_1_IN <= {matrix_8[9], matrix_8[10], matrix_8[11], matrix_8[12]};
Line2_2_IN <= {matrix_8[13], matrix_8[14], matrix_8[15], matrix_8[16]};

Line3_1_IN <= {matrix_8[17], matrix_8[18], matrix_8[19], matrix_8[20]};
Line3_2_IN <= {matrix_8[21], matrix_8[22], matrix_8[23], matrix_8[24]};

Line4_1_IN <= {matrix_8[25], matrix_8[26], matrix_8[27], matrix_8[28]};
```

```
Line4_2_IN <= {matrix_8[29], matrix_8[30], matrix_8[31], matrix_8[32]};

Line5_1_IN <= {matrix_8[33], matrix_8[34], matrix_8[35], matrix_8[36]};
Line5_2_IN <= {matrix_8[37], matrix_8[38], matrix_8[39], matrix_8[40]};

Line6_1_IN <= {matrix_8[41], matrix_8[42], matrix_8[43], matrix_8[44]};
Line6_2_IN <= {matrix_8[45], matrix_8[46], matrix_8[47], matrix_8[48]};

Line7_1_IN <= {matrix_8[49], matrix_8[50], matrix_8[51], matrix_8[52]};
Line7_2_IN <= {matrix_8[53], matrix_8[54], matrix_8[55], matrix_8[56]};

Line8_1_IN <= {matrix_8[57], matrix_8[58], matrix_8[59], matrix_8[60]};
Line8_2_IN <= {matrix_8[61], matrix_8[62], matrix_8[63], matrix_8[64]};

end

endmodule
```

3.) Now that we have recompiled the program, we will reopen the Nios software builder where we had just updated our program. We will now run that program and we should see a 8x8 matrix with the values after the DCT and Quantization as shown in Figure 3-9.



```
Final_DCT Nios II Hardware configuration - cable: USB-Blaster on localhost [USB-0] device ID: 1 instance ID: 0 name: jtaguart_0
63.499996 0.000416  -0.060249 0.077513  -0.081104 0.068931  -0.072960 0.080430
0.342246  -0.000224 0.000227  -0.000190 0.000157  -0.000081 0.000092  -0.000120
-0.640788 0.000425  -0.000370 0.000272  -0.000179 0.000141  -0.000133 0.000186
1.015214  -0.000499 0.000411  -0.000339 0.000206  -0.000134 0.000163  -0.000240
-1.114728 0.000530  -0.000338 0.000235  -0.000211 0.000145  -0.000169 0.000254
1.118093  -0.000457 0.000293  -0.000274 0.000231  -0.000197 0.000201  -0.000274
-0.712827 0.000330  -0.000272 0.000262  -0.000234 0.000213  -0.000243 0.000318
0.623905  -0.000296 0.000287  -0.000290 0.000275  -0.000328 0.000352  -0.000407
```

**Figure 3-9 The Values after DCT and Quantization**

You can use this program and incorporate it into multiple different types of projects. The next step after this will be doing full JPEG compression after sending the values that we had gotten in Figure 3-9 back to the FPGA.