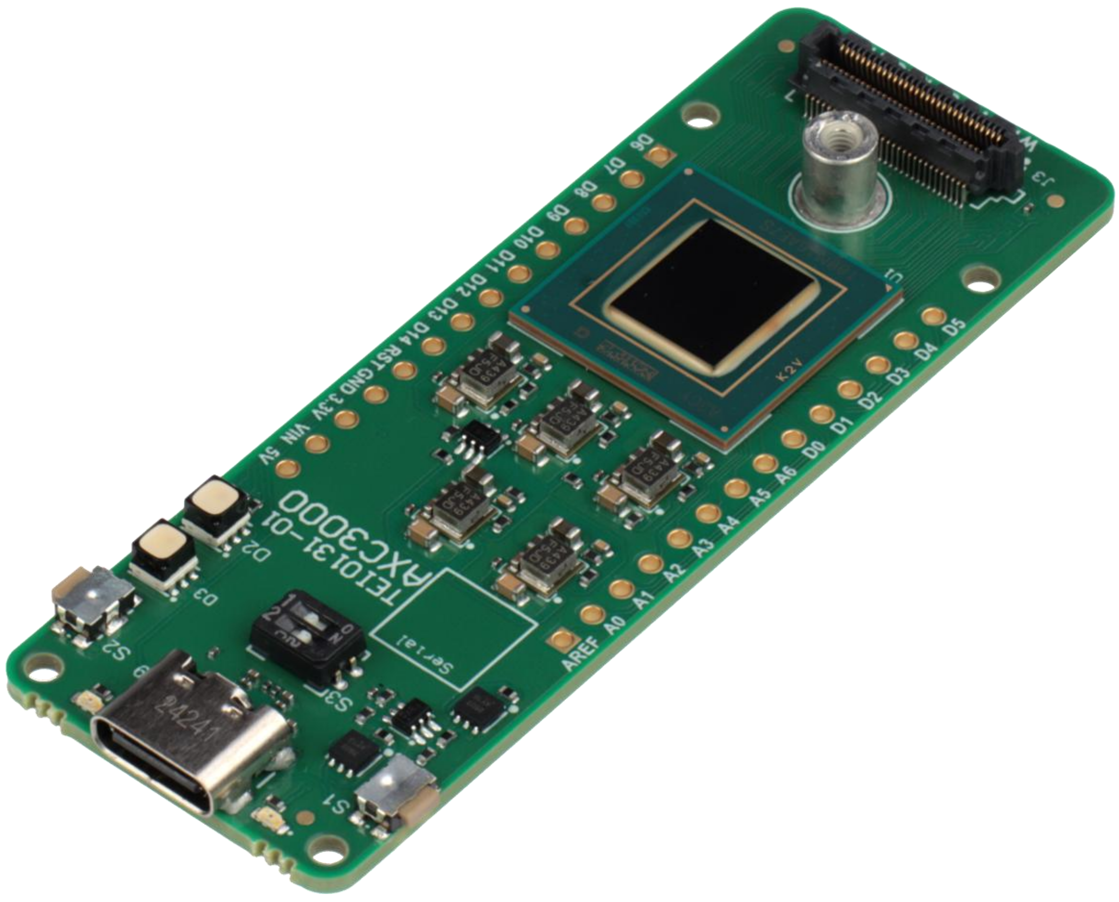
AXC3000 Simple Nios® V Lab



Software and hardware requirements to complete all exercises.

Software Requirements: Quartus® Prime Pro version 25.1.1

Hardware Requirements: AXC3000 Agilex™ 3C Development Kit

Document Control

Document Version:	Version 1.0
Document Date:	07/10/2025
Document Author(s):	Steven Kravatsky, Erika Peter, Naji Naufel, , ESC Team
Document Classification:	Released
Document Distribution:	This document is still under development. All specifications, procedures, and processes described in this document are subject to change without prior notice
Prior Version History:	

Please read the legal disclaimer at the end of this document.

Contents

1 Introduction

This tutorial provides comprehensive information to help you understand how to create a simple Nios® V Design and run it on your board. This lab will not make you an expert, but at the end, you will understand basic concepts about the embedded hardware and software flow targeting the soft core Nios V and downloading the system into the FPGA on your development board.

1.1 Objective

1. Create a Hardware Design
    a. Simple design based on a Nios V CPU in Platform Designer
    b. Top-level HDL, and timing constraints
    c. Assign pins and Compile project
2. Create a Software Design
    a. Create a Board Support Package (BSP)
    b. Create a simple C code program
    c. Compile and Generate the ELF file
3. Download and Run Application on a Development Board

Lab Notes: Many of the names that the lab asks you to use for files, components, and other objects in this exercise must be spelled exactly as directed. This nomenclature is necessary because the pre-written software application includes variables that use the names of the hardware peripherals. Naming the components differently can cause errors.

This lab can be compiled on a local machine or on a remote machine hosted by CloudLabs. Both options allow for the lab to be tested on a local AXC3000 board.

In addition, when using CloudLabs, testing can also be performed using a remote board in the Arrow board farm. Both options are documented.

2 Getting Started

The first objective is to ensure that you have all the necessary software installed so that the lab can be completed successfully. Below is a list of items required to complete this lab. A number of options are provided.

Option 1: Completing the lab using CloudLabs tools and the board farm.

- Personal computer or laptop running 64-bit Linux / Windows 10 or later with at least an Intel i3 core (or equivalent), 8GB RAM.

Option 2: Completing the lab using CloudLabs tools and local hardware.

- AXC3000 Board
- USB C cable
- Quartus Prime 25.1 .1Pro Standalone Programmer.
- Personal computer or laptop running 64-bit Linux / Windows 10 or later with at least an Intel i3 core (or equivalent), 8GB RAM.
- Terminal emulator (Putty or Tera term)

Option 3: Completing the lab using a laptop and local hardware.

- AXC3000 Board
- USB C cable
- Quartus Prime 25.1.1 Pro
- Ashling RiscFree IDE
- Personal computer or laptop running 64-bit Linux / Windows 10 or later with at least an Intel i3 core (or equivalent), 8GB RAM.
- Terminal emulator (Putty or Tera term)

A desire to learn!

3 Design Flow

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

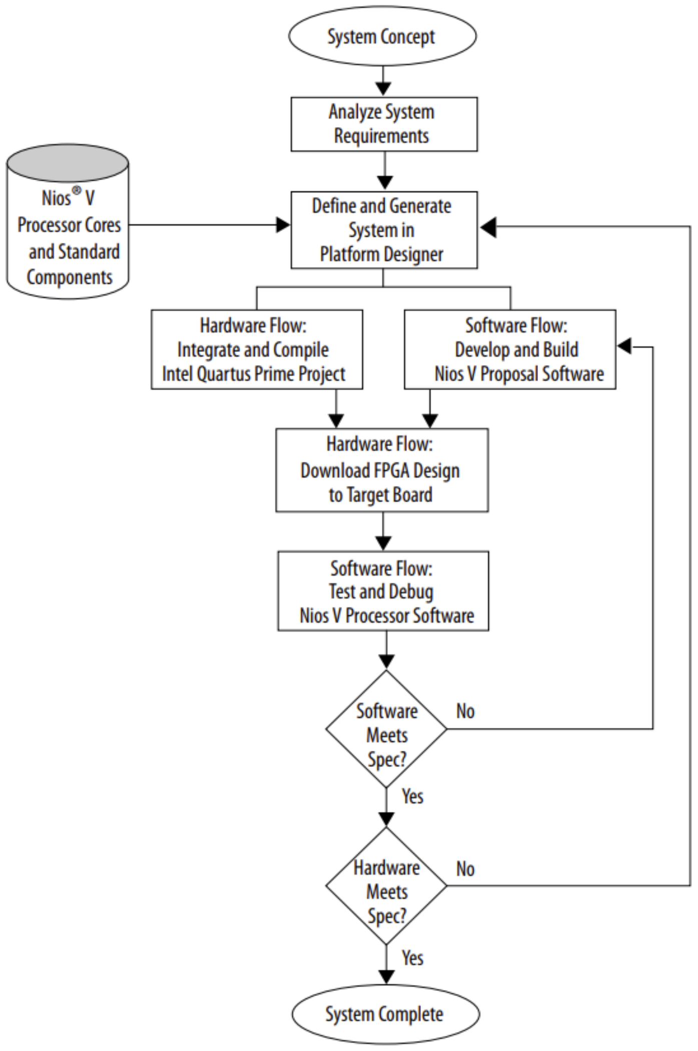

System Concept
Analyze System Requirements
Nios® V Processor Cores and Standard Components
Define and Generate System in Platform Designer
Hardware Flow: Integrate and Compile Intel Quartus Prime Project
Software Flow: Develop and Build Nios V Proposal Software
Hardware Flow: Download FPGA Design to Target Board
Software Flow: Test and Debug Nios V Processor Software
Software Meets Spec?
No
Yes
Hardware Meets Spec?
No
Yes
System Complete

Figure 3-1 : Design Flow

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

4 Hardware Project with AXC3000 board

This chapter gives instructions for creating the hardware project in Quartus® Prime Pro 25.1.

4.1 New Quartus Prime project

The hardware implementation starts with a Quartus® Prime project. A Quartus project defines the location for the HDL and IP files used, target family, target device, and initial settings. After the initial creation of the project, changes can be made. Quartus uses the project location for the generated files.

4.1.1 New project creation

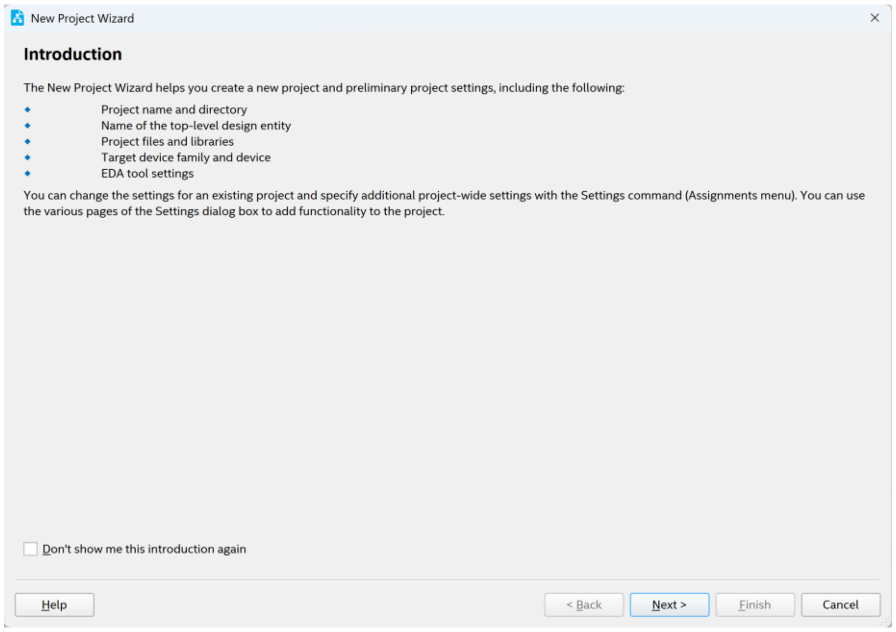
There are many screens that will need to be filled in when creating a new project. The steps below will take you through all of them.

4.1.1.1 Open Quartus Pro 25.1.1

If not already open, from the Start menu or the Desktop, open the Quartus Prime Pro 25.1 .1 IDE.

4.1.1.2 Create a new project

Using the New Project Wizard: **File** → **New Project Wizard**. This screen shows an introduction window. Click **Next.**



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

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

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

Don't show me this introduction again

Figure 4-1 : New Project Wizard Introduction

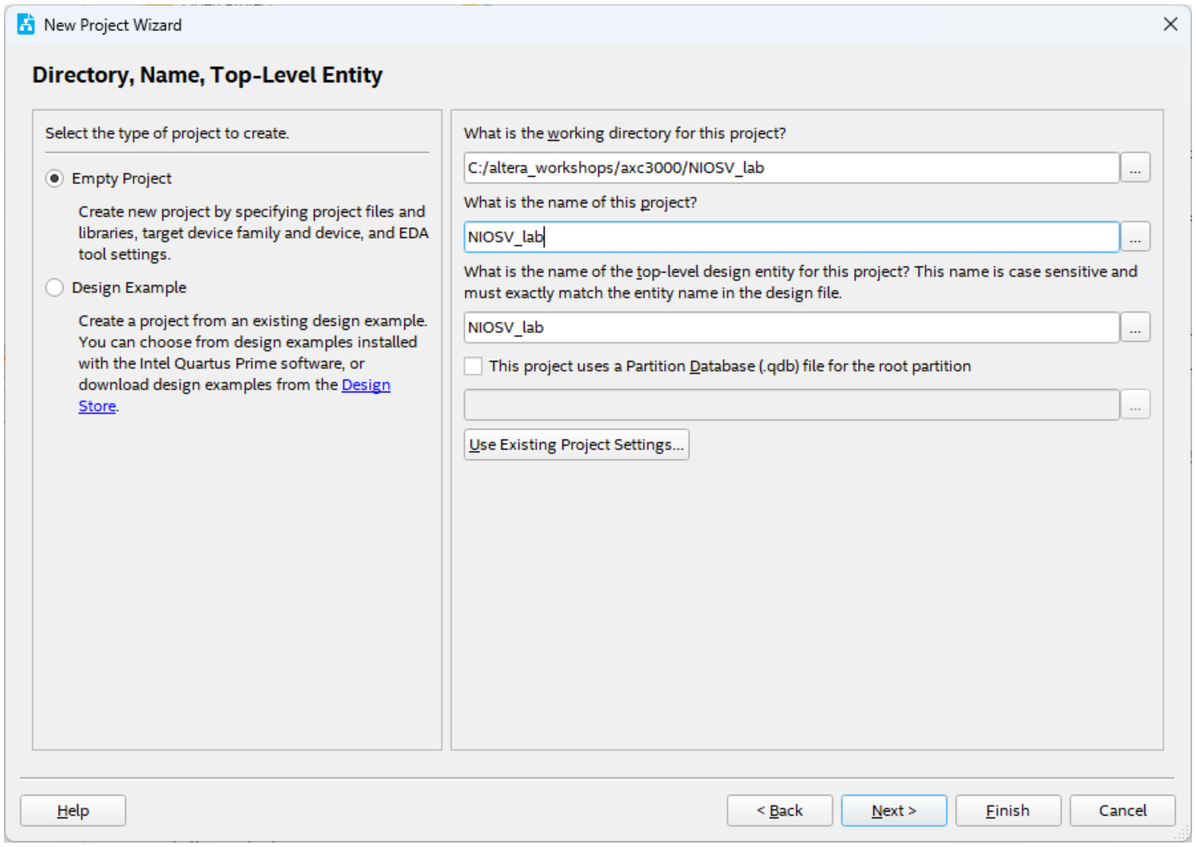
4.1.1.3 Configure the New Project Wizard

Specify project directory, name, and top-level entity information:

Note: No spaces are allowed in path or filenames.

- Ensure that the **Empty Project** radio button is selected on the left side of the window.
- Enter a directory in which you will store your Quartus project files for this design C:/altera_workshops/axc3000/NIOSV_lab
- Specify the name of the project: **NIOSV_lab**
- Specify the name of the top-level entity: **NIOSV_lab**
- Click **Next**.

If you are asked to allow creation of a non-existent folder, click **Yes**.



Directory, Name, Top-Level Entity

Select the type of project to create.

(●) Empty Project

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

( ) Design Example

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

What is the working directory for this project?
C:/altera_workshops/axc3000/NIOSV_lab

What is the name of this project?
NIOSV_lab

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

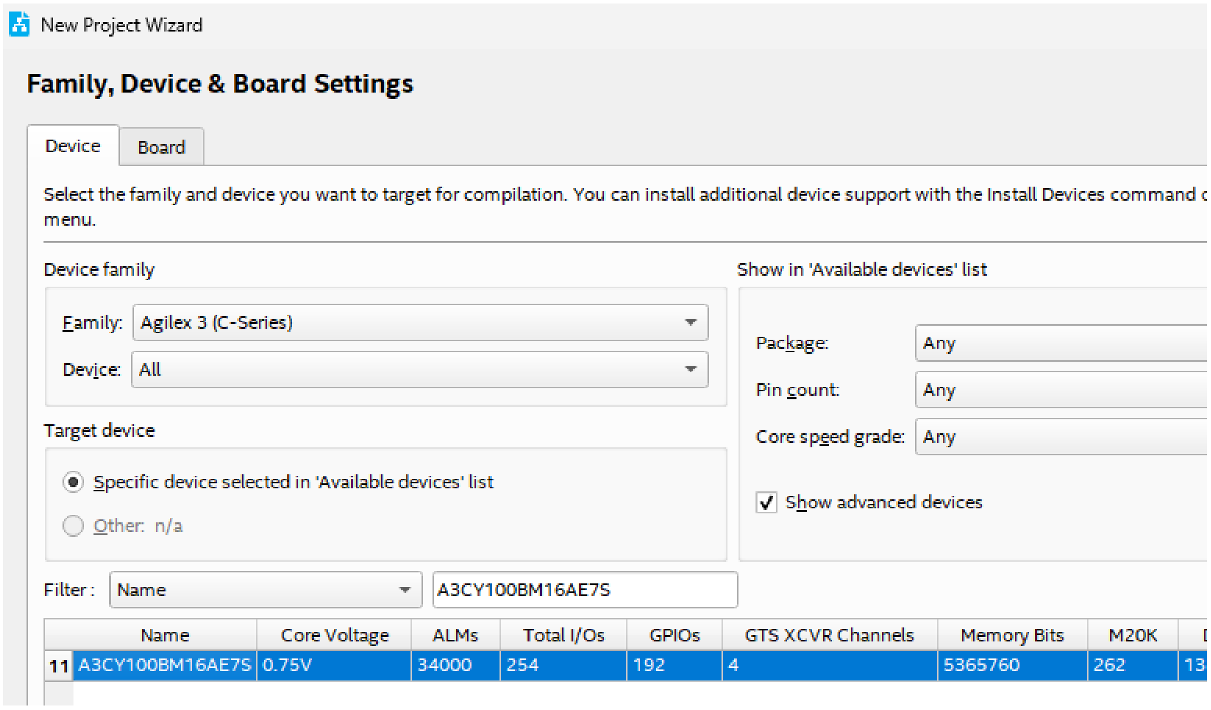
☐ This project uses a Partition Database (.qdb) file for the root partition

Use Existing Project Settings...

Figure 4-2 : New Project Wizard Folder/Project Names

4.1.1.4 Specify Family and Device Settings

In the Family field, select ***Agilex 3 (C-Series)*** and enter the part number (**A3CY100BM16AE7S**) in the Name Filter text box and select it. Click **Next**.


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 o menu.

Device family

Family: Agilex 3 (C-Series)

Device: All

Show in 'Available devices' list

Package: Any

Pin count: Any

Core speed grade: Any

[✓] Show advanced devices

Target device

(●) Specific device selected in 'Available devices' list

( ) Other: n/a

Filter: Name A3CY100BM16AE7S

	Name	Core Voltage	ALMs	Total I/Os	GPIOs	GTS XCVR Channels	Memory Bits	M20K	[
11	A3CY100BM16AE7S	0.75V	34000	254	192	4	5365760	262	13

Figure 4-3 : New Project Wizard Device Selection

4.1.1.5 Add Files

On the Add Files page, click **Next**. No files will be included.

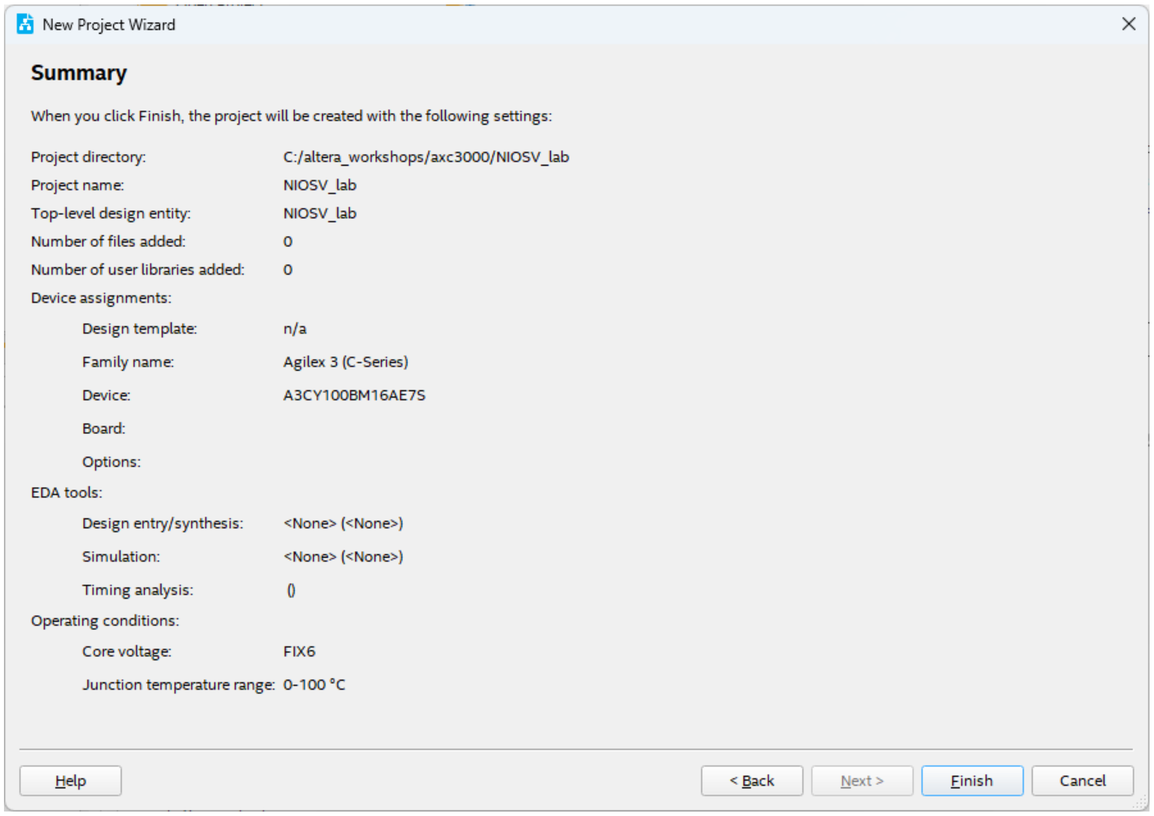
4.1.1.6 EDA Tools Settings

On the EDA Tools Settings page, select **Next**.

4.1.1.7 Summary Screen

The Summary screen will show the file location and the device selection. These settings can be changed if needed, later.

Click **Finish**.



Summary

When you click Finish, the project will be created with the following settings:

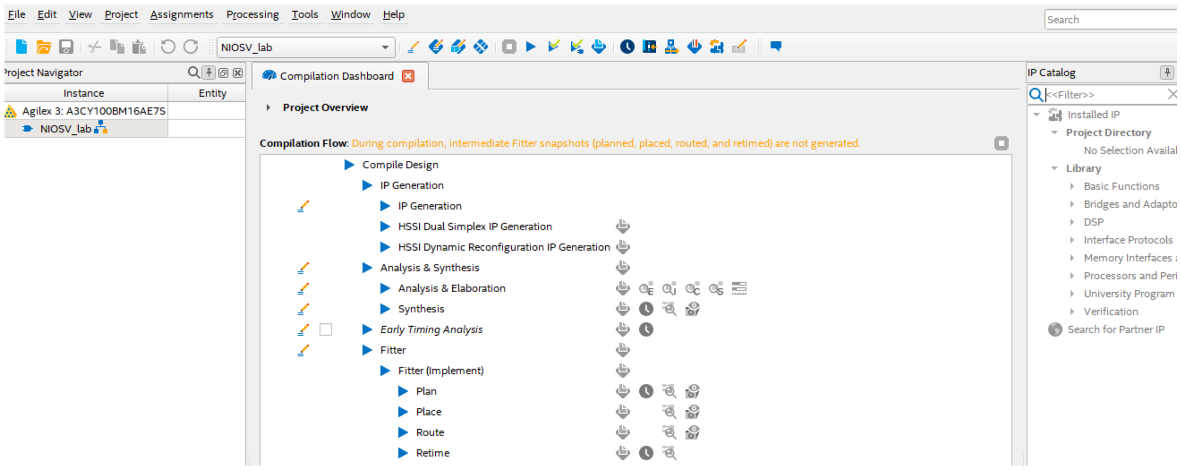
Project directory:	C:/altera_workshops/axc3000/NIOSV_lab
Project name:	NIOSV_lab
Top-level design entity:	NIOSV_lab
Number of files added:	0
Number of user libraries added:	0
Device assignments:	
Design template:	n/a
Family name:	Agilex 3 (C-Series)
Device:	A3CY100BM16AE7S
Board:	
Options:	
EDA tools:	
Design entry/synthesis:	<None> (<None>)
Simulation:	<None> (<None>)
Timing analysis:	0
Operating conditions:	
Core voltage:	FIX6
Junction temperature range:	0-100 °C

Figure 4-4 : Project Summary

4.2 Design Entry

In this module you will use Platform Designer to create hardware for the processor system. You will add components and make interface connections.

4.2.1 Project Top Level



Compilation Flow: During compilation, intermediate Fitter snapshots (planned, placed, routed, and retimed) are not generated.

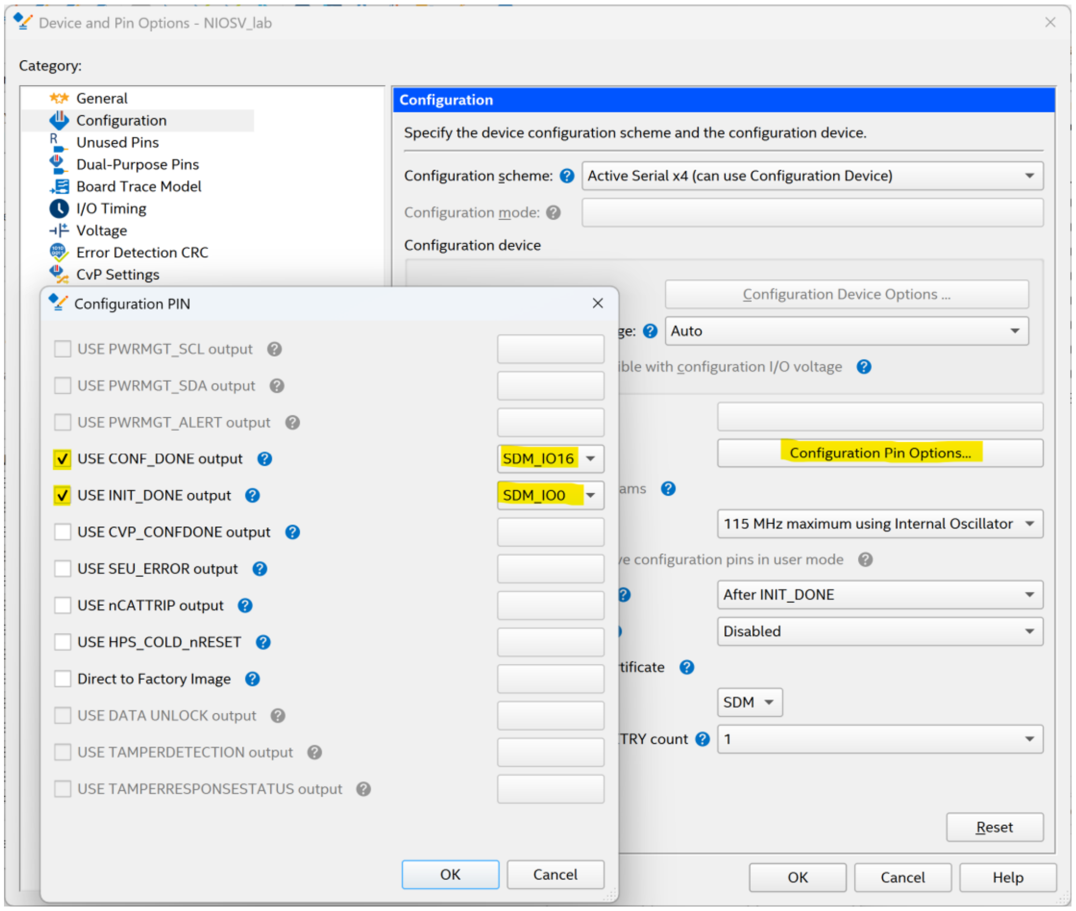
- Compile Design
  - IP Generation
    - IP Generation
    - HSSI Dual Simplex IP Generation
    - HSSI Dynamic Reconfiguration IP Generation
  - Analysis & Synthesis
    - Analysis & Elaboration
    - Synthesis
  - *Early Timing Analysis*
  - Fitter
    - Fitter (Implement)
      - Plan
      - Place
      - Route
      - Retime

Figure 4-5 : Quartus Project View

The top-level file is seen in the **Project Navigator** pane (top left). The middle section (**Compilation Flow**) shows the FPGA steps which will be taken by the compiler. These steps include analysis and synthesis, fitter and etc.

The **AXC3000** board has an LED indicator on the **CONFIG_DONE** pin. **CONFIG_DONE,** being a Dual-Purpose pin, has to be enabled to drive the LED once the FPGA enters user mode. To do so, select the menu **Assignments -> Device** then click the **Device and Pin Options...** button. When a window opens:

- Select **Configuration** in the Category pane and Click the **Configuration Pin Options** bar.
- Check the box to the left of **USE CONF_DONE** output and select ***SDM_IO16***
- Check the box to the left of **USE INIT_DONE** output and select ***SDM_IO0***
- Click **OK** in multiple windows until done with this task.



Configuration

Specify the device configuration scheme and the configuration device.

Configuration scheme: Active Serial x4 (can use Configuration Device)

Configuration mode:

Configuration device

...ge: Auto

...ible with configuration I/O voltage

...ams: 115 MHz maximum using Internal Oscillator

...e configuration pins in user mode

After INIT_DONE

Disabled

...tificate: SDM

...TRY count: 1

☐ USE PWRMGT_SCL output

☐ USE PWRMGT_SDA output

☐ USE PWRMGT_ALERT output

☑ USE CONF_DONE output — SDM_IO16

☑ USE INIT_DONE output — SDM_IO0

☐ USE CVP_CONFDONE output

☐ USE SEU_ERROR output

☐ USE nCATTRIP output

☐ USE HPS_COLD_nRESET

☐ Direct to Factory Image

☐ USE DATA UNLOCK output

☐ USE TAMPERDETECTION output

☐ USE TAMPERRESPONSESTATUS output

Figure 4-6 : Configuration Pin Assignment

4.3 Design Flow

Our design flow starts with Platform Designer where we add the Nios V CPU and its peripherals used in this workshop.

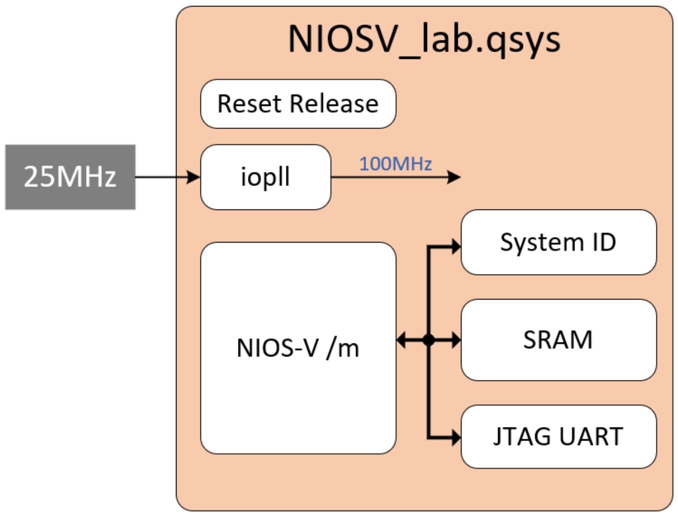


Figure 4-7 : Design Block Diagram

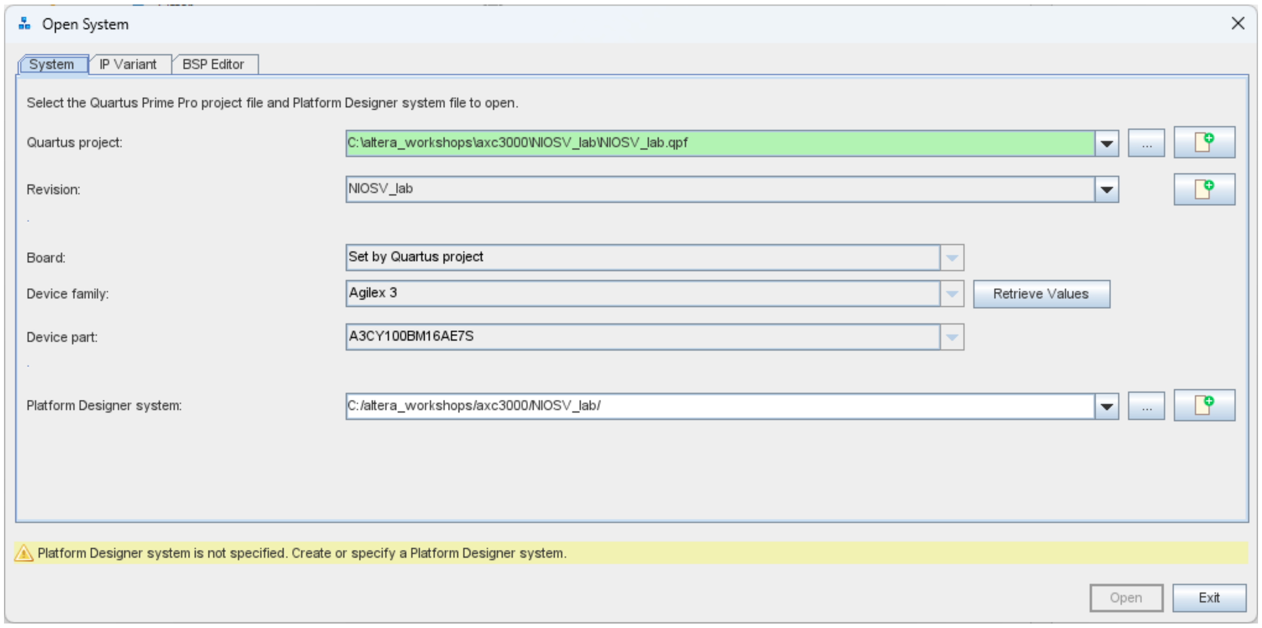
4.3.1 Add components to Platform Designer

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

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

4.3.1.1 Open Platform Designer

from the **Tools** → **Platform Designer** or clicking on button on the toolbar. The **Open System** window pops up.


Open System

System | IP Variant | BSP Editor

Select the Quartus Prime Pro project file and Platform Designer system file to open.

Quartus project: C:\altera_workshops\axc3000\NIOSV_lab\NIOSV_lab.qpf

Revision: NIOSV_lab

Board: Set by Quartus project

Device family: Agilex 3

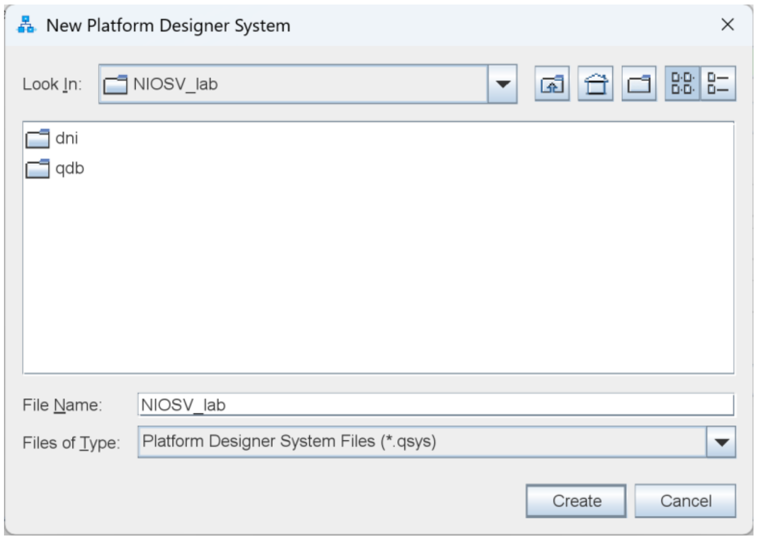
Device part: A3CY100BM16AE7S

Platform Designer system: C:/altera_workshops/axc3000/NIOSV_lab/

Platform Designer system is not specified. Create or specify a Platform Designer system.

Figure 4-8 : Open System

In the Open System window, click the icon on the Platform Designer system line to create a new system. If the system already exists, click the ... symbol instead, to open it. In the pop-up window, give it the name, then click the **Create** button.


New Platform Designer System

Look In: NIOSV_lab

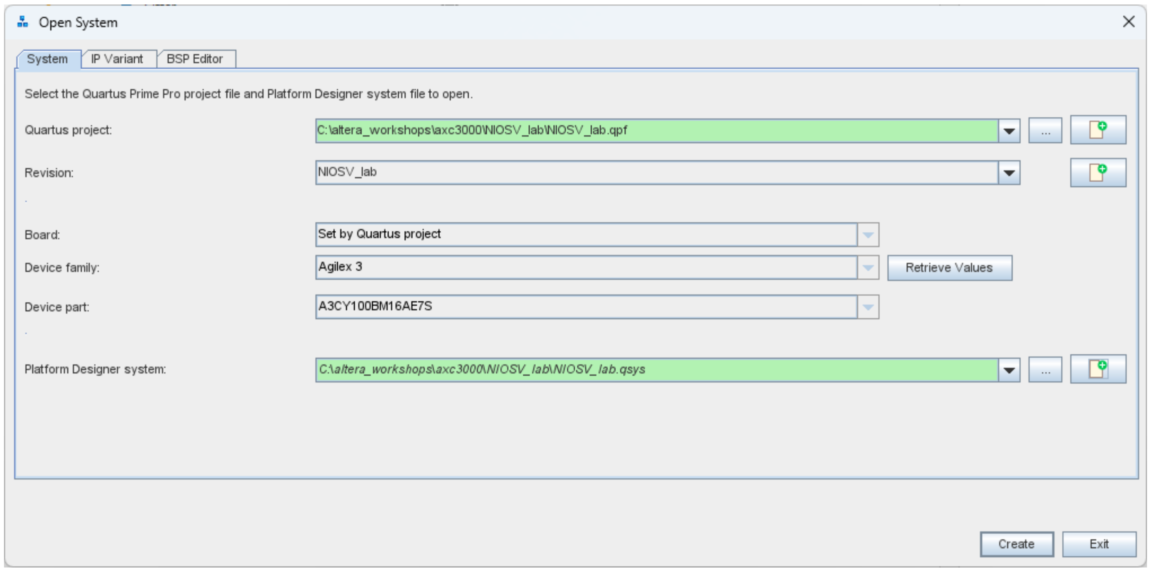
dni
qdb

File Name: NIOSV_lab

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

Figure 4-9 : New Platform Designer System window

In the **Open System** window, click the **Create** button.



Select the Quartus Prime Pro project file and Platform Designer system file to open.

Quartus project: C:\altera_workshops\axc3000\NIOSV_lab\NIOSV_lab.qpf

Revision: NIOSV_lab

Board: Set by Quartus project

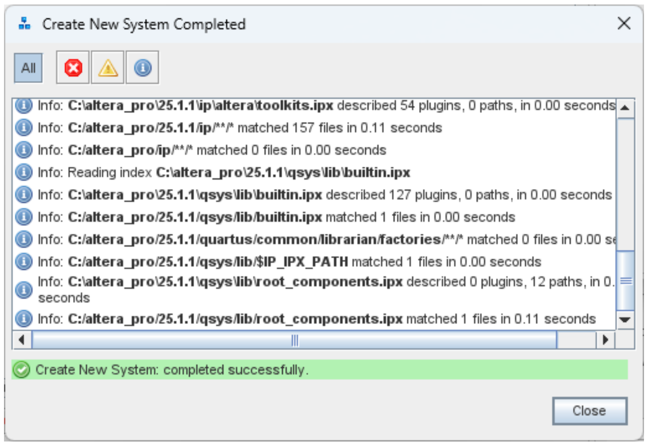
Device family: Agilex 3

Device part: A3CY100BM16AE7S

Platform Designer system: C:\altera_workshops\axc3000\NIOSV_lab\NIOSV_lab.qsys

Figure 4-10 : Completed Open System window

When the **Create New System** window shows that it completed successfully, click on the **Close** button.



```
Info: C:\altera_pro\25.1.1\ip\altera\toolkits.ipx described 54 plugins, 0 paths, in 0.00 seconds
Info: C:/altera_pro/25.1.1/ip/**/* matched 157 files in 0.11 seconds
Info: C:/altera_pro/ip/**/* matched 0 files in 0.00 seconds
Info: Reading index C:\altera_pro\25.1.1\qsys\lib\builtin.ipx
Info: C:\altera_pro\25.1.1\qsys\lib\builtin.ipx described 127 plugins, 0 paths, in 0.00 seconds
Info: C:/altera_pro/25.1.1/qsys/lib/builtin.ipx matched 1 files in 0.00 seconds
Info: C:/altera_pro/25.1.1/quartus/common/librarian/factories/**/* matched 0 files in 0.00 se
Info: C:/altera_pro/25.1.1/qsys/lib/$IP_IPX_PATH matched 1 files in 0.00 seconds
Info: C:\altera_pro\25.1.1\qsys\lib\root_components.ipx described 0 plugins, 12 paths, in 0.
seconds
Info: C:/altera_pro/25.1.1/qsys/lib/root_components.ipx matched 1 files in 0.11 seconds
```

Create New System: completed successfully.

Figure 4-11 : Create New System Completed

In the new window, you should see a single clock source component, named **clock_in** in the ***System View*** tab. This tab shows all the components currently in your system. There will be 2 components that are automatically part of the Platform Designer, the Clock Bridge and the Reset Bridge.

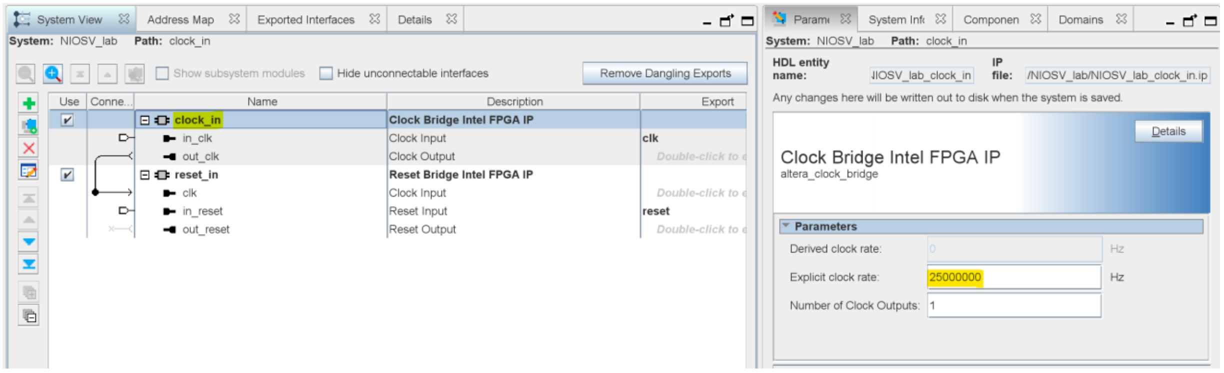
All FPGA designs need at least one Clock Bridge. Synchronous designs are more reliable, have better performance and are more verifiable than a combinational design. Combinational design is not a good design approach.

The Reset Bridge allows the reset signal to be connected to different components and sub-systems in the Platform Designer. The reset bridge defines how the reset is synchronized (i.e. synchronous/asynchronous) and if the reset is active-low/high.

4.3.1.2 Configure the Clock Bridge

The Clock Bridge has a default clock speed assigned to its input. The frequency needs to be changed to match the clock source on the **AXC3000** board. In this case, 25MHz.

Double-click the **clock_in** IP. In the Parameters window that shows up, edit the ***Explicit Clock Rate*** to ***25000000***.



System: NIOSV_lab Path: clock_in

Use	Conne...	Name	Description	Export
✓		clock_in	Clock Bridge Intel FPGA IP	
		in_clk	Clock Input	clk
		out_clk	Clock Output	Double-click to e
✓		reset_in	Reset Bridge Intel FPGA IP	
		clk	Clock Input	Double-click to e
		in_reset	Reset Input	reset
		out_reset	Reset Output	Double-click to e

HDL entity name: NIOSV_lab_clock_in IP file: /NIOSV_lab/NIOSV_lab_clock_in.ip

Any changes here will be written out to disk when the system is saved.

Clock Bridge Intel FPGA IP
altera_clock_bridge

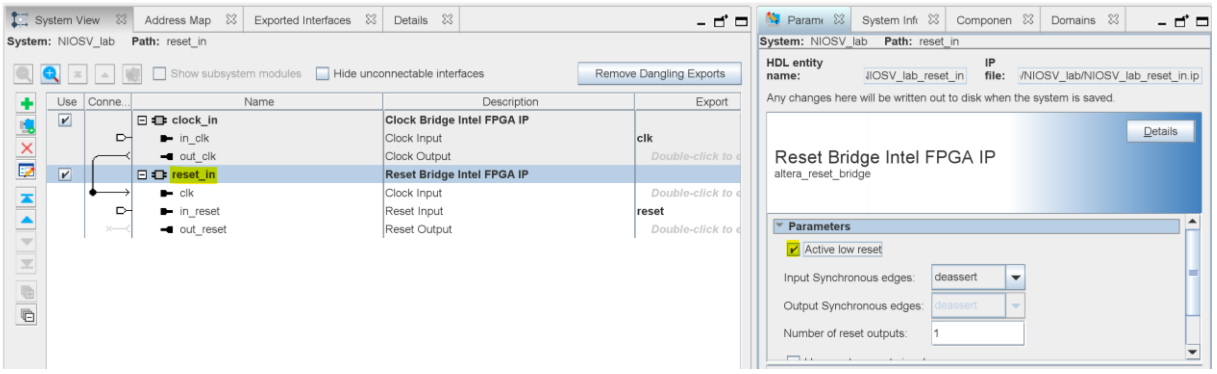
Parameters

Derived clock rate: 0 Hz
Explicit clock rate: 25000000 Hz
Number of Clock Outputs: 1

Figure 4-12 : Configure Clock Bridge

4.3.1.3 Configure the Reset Bridge

The Reset Bridge also has default settings that need to be changed. The Reset input on the board is tied to a push-button which goes low when pressed, we need to check the box for **Active low reset**.



System: NIOSV_lab Path: reset_in

Use	Conne...	Name	Description	Export
✓		clock_in	Clock Bridge Intel FPGA IP	
		in_clk	Clock Input	clk
		out_clk	Clock Output	Double-click to e
✓		reset_in	Reset Bridge Intel FPGA IP	
		clk	Clock Input	Double-click to e
		in_reset	Reset Input	reset
		out_reset	Reset Output	Double-click to e

HDL entity name: NIOSV_lab_reset_in IP file: /NIOSV_lab/NIOSV_lab_reset_in.ip

Any changes here will be written out to disk when the system is saved.

Reset Bridge Intel FPGA IP
altera_reset_bridge

Parameters

✓ Active low reset
Input Synchronous edges: deassert
Output Synchronous edges: deassert
Number of reset outputs: 1

Figure 4-13 : Configure Reset Bridge

4.3.1.4 Add PLL IP

Since the input clock frequency of 25MHz is too slow, we need a Phase-Locked Loop (PLL) IP to multiply the frequency up to 100MHz (as example).

From the IP Catalog panel on the left side, expand the menus for the **Basic Functions** → **Clocks; PLLs and Resets** → **PLL** (or type ***pll*** in the search field) and double click on **IOPLL Intel FPGA IP**.

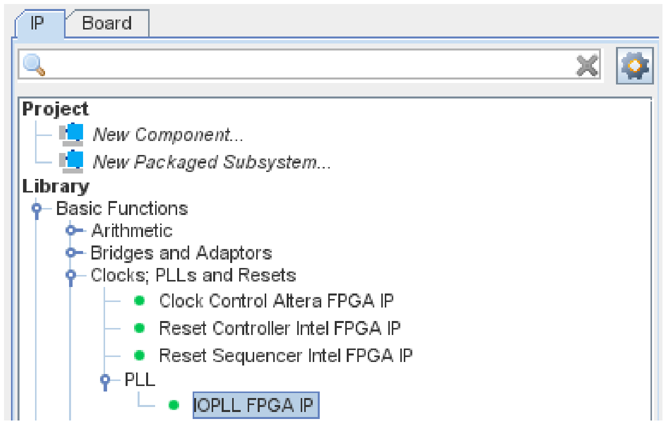


Figure 4-14 : PLL IP

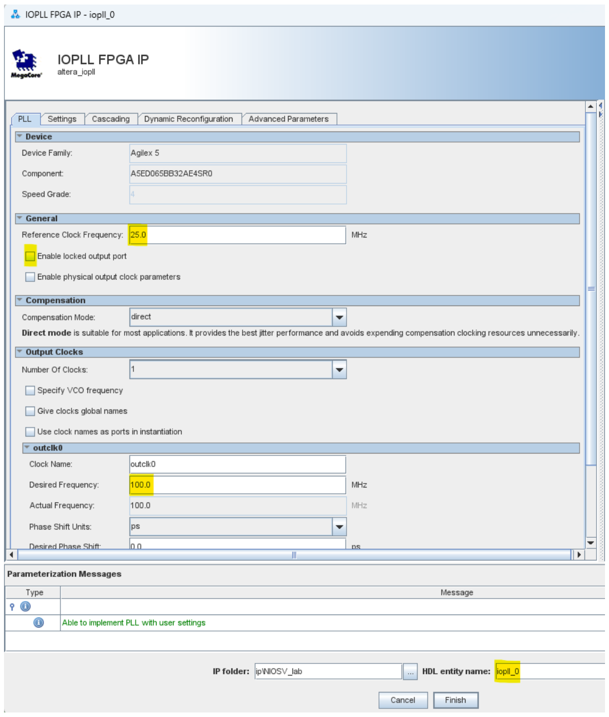
A PLL (phase lock loop) provides flexibility with clock settings. It has a clock input and multiple clock outputs which can be set to different frequencies and phases.

In the **PLL** tab of the **IOPLL FPGA IP** settings window, do:

- Change the **Reference Clock Frequency** to 25.0 MHz to match our input clock,
- Uncheck the Enable locked output port,
- Keep the **Number Of Clocks** set to **1**,
- For **outclk0**, keep the default **100.0MHz** setting,

Change the HDL entity name to *iopll_0*.

- Click the **Finish** button.



IOPLL FPGA IP - iopll_0

IOPLL FPGA IP
altera_iopll

PLL | Settings | Cascading | Dynamic Reconfiguration | Advanced Parameters

Device

Device Family: Agilex 5
Component: A5ED065BB32AE4SR0
Speed Grade: 4

General

Reference Clock Frequency: 25.0 MHz
[ ] Enable locked output port
[ ] Enable physical output clock parameters

Compensation

Compensation Mode: direct
Direct mode is suitable for most applications. It provides the best jitter performance and avoids expending compensation clocking resources unnecessarily.

Output Clocks

Number Of Clocks: 1
[ ] Specify VCO frequency
[ ] Give clocks global names
[ ] Use clock names as ports in instantiation

outclk0

Clock Name: outclk0
Desired Frequency: 100.0 MHz
Actual Frequency: 100.0 MHz
Phase Shift Units: ps
Desired Phase Shift: 0.0 ps

Parameterization Messages

Type	Message
	Able to implement PLL with user settings

IP folder: ip\NIOSV_lab
HDL entity name: iopll_0

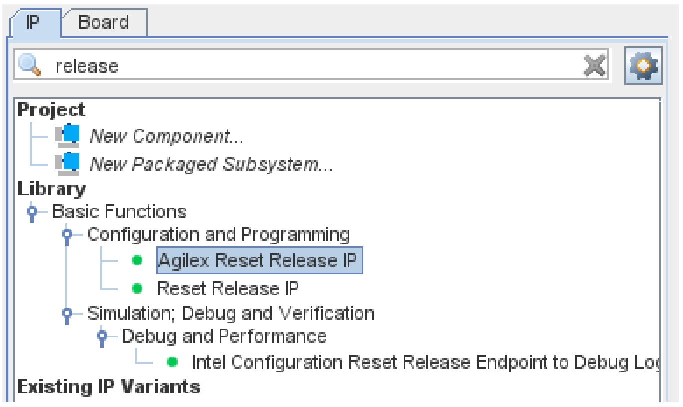
Figure 4-15 : PLL settings

By setting the reference clock frequency to 25MHz and matching it to the **clock_in** bridge IP, it gives us an input clock definition. Thus, it is not necessary to create an SDC constraint for it. Quartus uses this setting and the **outclk0** as clocks in the design.

4.3.1.5 Add Agilex Reset Release IP

The Agilex FPGA families require a coordinated release from the reset state after configuration is complete. This IP accomplishes this task.

In the **IP** pane, type **release** in the search field.



IP
Board
release
Project
New Component...
New Packaged Subsystem...
Library
Basic Functions
Configuration and Programming
Agilex Reset Release IP
Reset Release IP
Simulation; Debug and Verification
Debug and Performance
Intel Configuration Reset Release Endpoint to Debug Log
Existing IP Variants

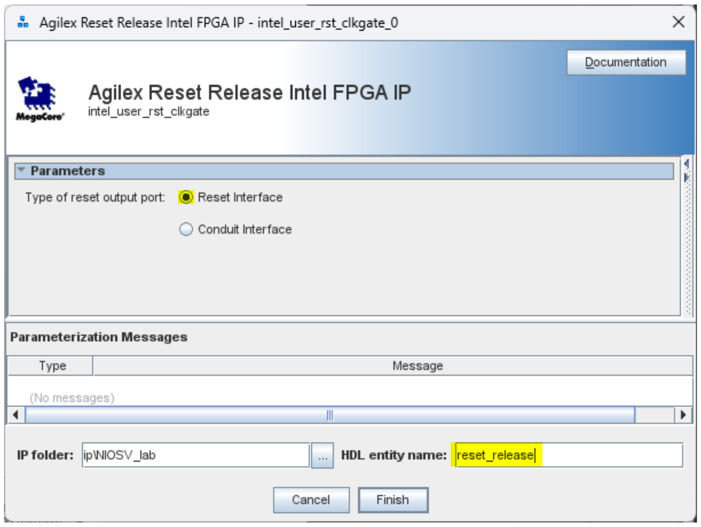
Figure 4-16 : Add Agilex Reset Release IP

Double-click Agilex Reset Release IP.

Check the **Reset Interface** radio button.

Change the HDL entity name to *reset_release*.

Click **Finish**.


Agilex Reset Release Intel FPGA IP - intel_user_rst_clkgate_0
Documentation
Agilex Reset Release Intel FPGA IP
intel_user_rst_clkgate
Parameters
Parameterization Messages
Type
Message
(No messages)
Cancel
Finish

Type of reset output port: [✓ Reset Interface] Conduit Interface

IP folder: ip\NIOSV_lab

HDL entity name: reset_release

Figure 4-17 : Reset Release IP settings

4.3.1.6 Add Nios V/m IP

The Nios V CPU comes in 3 variants, /c (compact microcontroller), /m (microcontroller), and /g (general purpose processor). More details about these variants can be found here. For the purposes of this lab, we will be using the /m variant.

In the IP search box, type **nios** and double-click **Nios V/m Microcontroller Intel FPGA IP**.

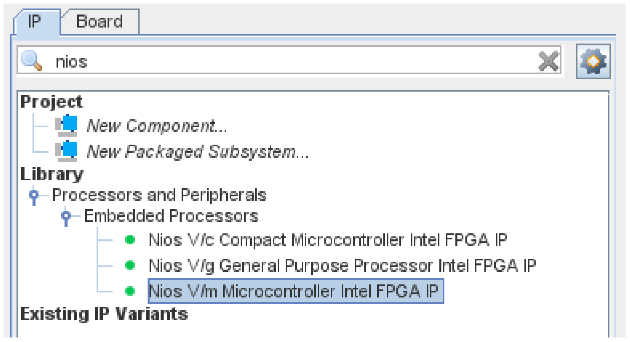
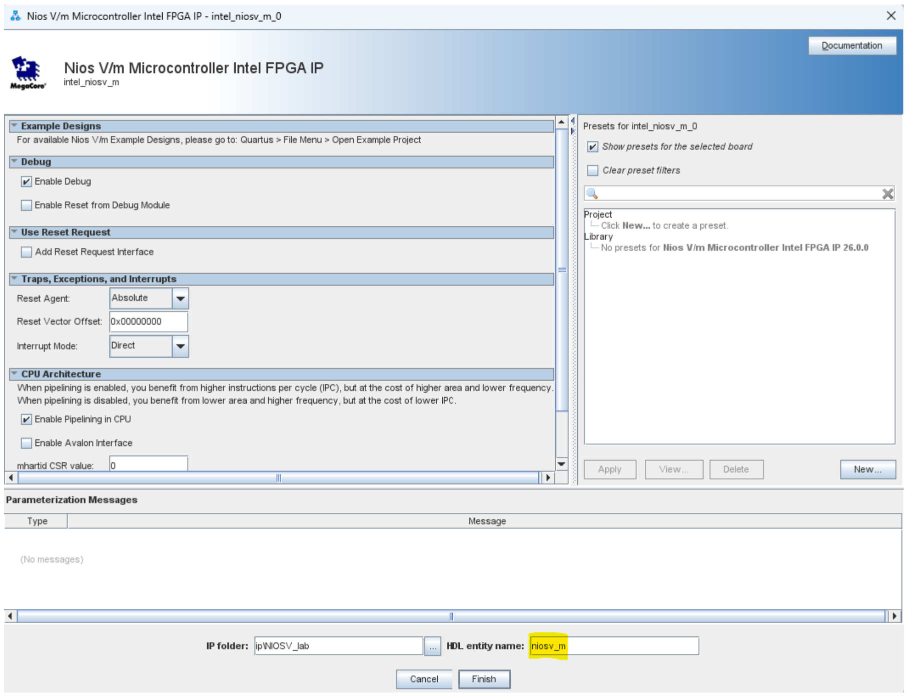

IP
Board
nios
Project
New Component...
New Packaged Subsystem...
Library
Processors and Peripherals
Embedded Processors
Nios V/c Compact Microcontroller Intel FPGA IP
Nios V/g General Purpose Processor Intel FPGA IP
Nios V/m Microcontroller Intel FPGA IP
Existing IP Variants

Figure 4-18 : Add Nios V/m IP

Change the HDL entity name to *niosv_m*. Click Finish.



Nios V/m Microcontroller Intel FPGA IP - intel_niosv_m_0

Nios V/m Microcontroller Intel FPGA IP
intel_niosv_m

Example Designs
For available Nios V/m Example Designs, please go to: Quartus > File Menu > Open Example Project

Debug
[✓] Enable Debug
[ ] Enable Reset from Debug Module

Use Reset Request
[ ] Add Reset Request Interface

Traps, Exceptions, and Interrupts
Reset Agent: Absolute
Reset Vector Offset: 0x00000000
Interrupt Mode: Direct

CPU Architecture
When pipelining is enabled, you benefit from higher instructions per cycle (IPC), but at the cost of higher area and lower frequency.
When pipelining is disabled, you benefit from lower area and higher frequency, but at the cost of lower IPC.
[✓] Enable Pipelining in CPU
[ ] Enable Avalon Interface
mhartid CSR value: 0

Presets for intel_niosv_m_0
[✓] *Show presets for the selected board*
[ ] *Clear preset filters*
Project
└ Click **New...** to create a preset.
Library
└ No presets for **Nios V/m Microcontroller Intel FPGA IP 26.0.0**

Parameterization Messages

Type	Message
(No messages)	

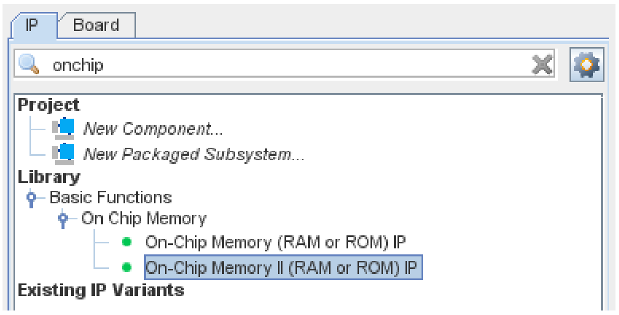
IP folder: ip\NIOSV_lab **HDL entity name:** niosv_m

Figure 4-19 : Add Nios V/m settings

4.3.1.7 Add OnChip Memory IP

The Nios V needs memory to run its code from. In this lab, we will use a 160KB onchip SRAM for this purpose. For NIOS V, Intel recommends On-Chip Memory II IP.

In the IP search field, enter **onchip.**



onchip

Project
- *New Component...*
- *New Packaged Subsystem...*

Library
- Basic Functions
  - On Chip Memory
    - On-Chip Memory (RAM or ROM) IP
    - On-Chip Memory II (RAM or ROM) IP

Existing IP Variants

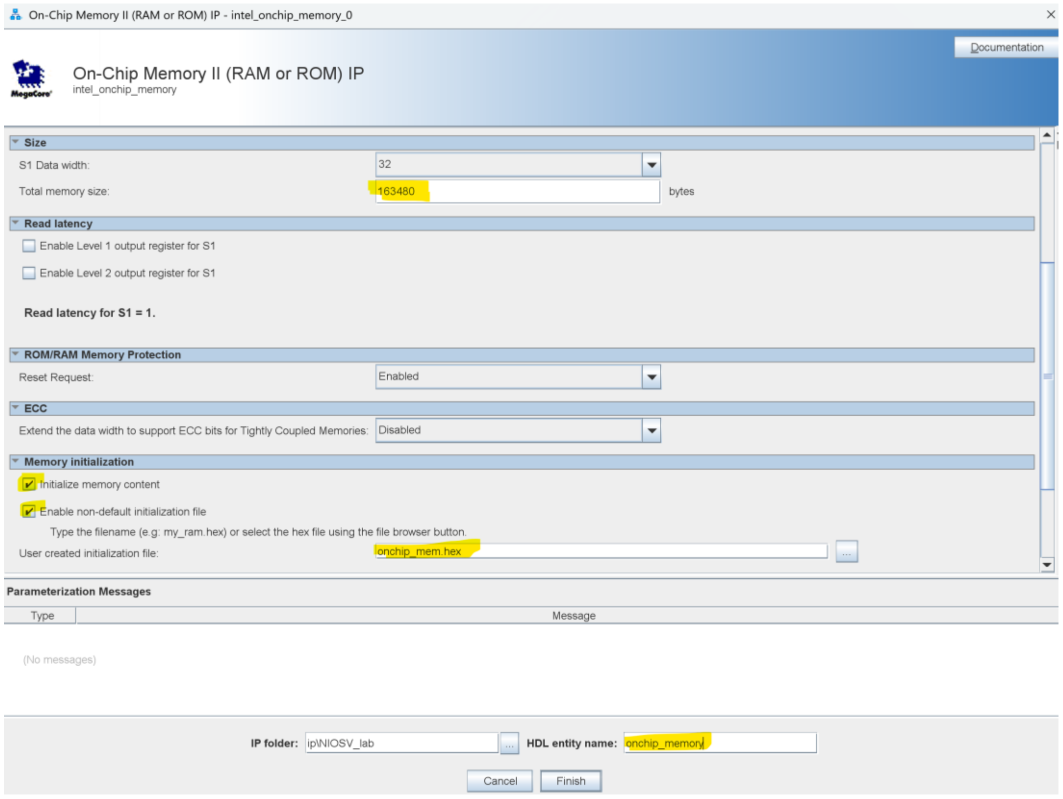
Figure 4-20 : Add on-chip SRAM IP

Double-click the **On-Chip Memory II IP**.

In the On-Chip Memory II wizard, do:

- Change the Total Memory Size to *163840*,
- Change the HDL entity name to *onchip_memory*,
- Check the **Initialize Memory Content** check box in the Memory Initialization section

- Check the **Enable non-default initialization file** check box. Use the default filename.
- Click **Finish**.


On-Chip Memory II (RAM or ROM) IP - intel_onchip_memory_0
Documentation
On-Chip Memory II (RAM or ROM) IP
intel_onchip_memory
Parameterization Messages
Type
Message
(No messages)
Cancel
Finish

Size
S1 Data width: 32
Total memory size: 163480 bytes

Read latency
☐ Enable Level 1 output register for S1
☐ Enable Level 2 output register for S1
Read latency for S1 = 1.

ROM/RAM Memory Protection
Reset Request: Enabled

ECC
Extend the data width to support ECC bits for Tightly Coupled Memories: 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

IP folder: ip\NIOSV_lab HDL entity name: onchip_memory

Figure 4-21 : Add on-chip SRAM settings

4.3.1.8 Add Lightweight UART IP

The Lightweight UART allows the user to interact with the C code using **stdio** operations without requiring an external UART device. In this lab, we will use it to display messages on a Putty or Tera term terminal emulator.

In the IP search field, type **uart**.

Double-click the **Lightweight UART IP**.

Keep all its default values.

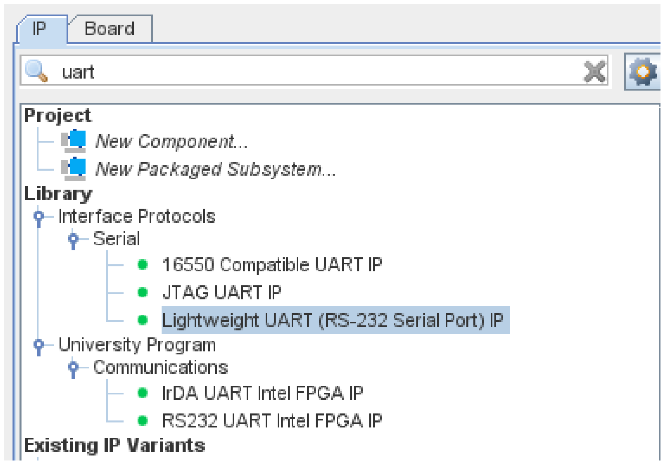
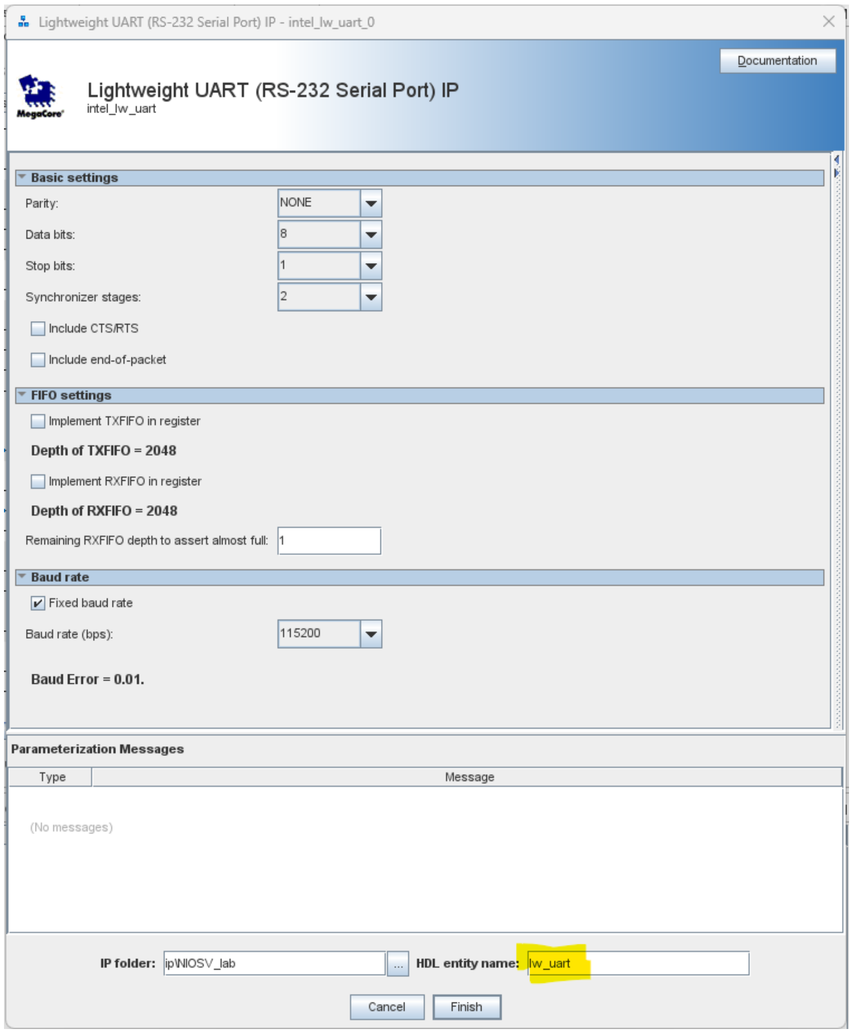


Figure 4-22 : Add JTAG UART IP

Change the HDL entity name *to lw_uart*. Click Finish.


Lightweight UART (RS-232 Serial Port) IP - intel_lw_uart_0

Lightweight UART (RS-232 Serial Port) IP
intel_lw_uart

Basic settings

Parity: NONE
Data bits: 8
Stop bits: 1
Synchronizer stages: 2
☐ Include CTS/RTS
☐ Include end-of-packet

FIFO settings

☐ Implement TXFIFO in register
Depth of TXFIFO = 2048
☐ Implement RXFIFO in register
Depth of RXFIFO = 2048
Remaining RXFIFO depth to assert almost full: 1

Baud rate

☑ Fixed baud rate
Baud rate (bps): 115200
Baud Error = 0.01.

Parameterization Messages

Type	Message
(No messages)	

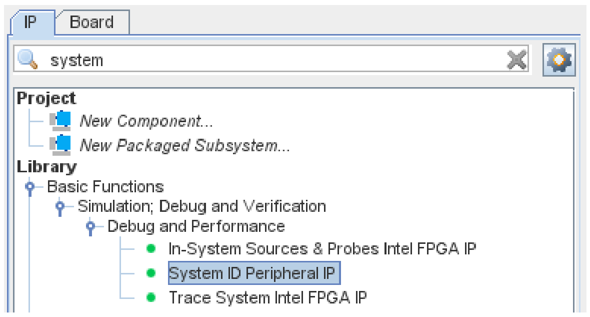
IP folder: ip\NIOSV_lab
HDL entity name: lw_uart

Figure 4-23 : Add JTAG UART settings

4.3.1.9 Add System ID IP

The system ID is a 32-bit register that can be used to give a design its unique identification such as hardware version.

In the IP search field, type **system**.



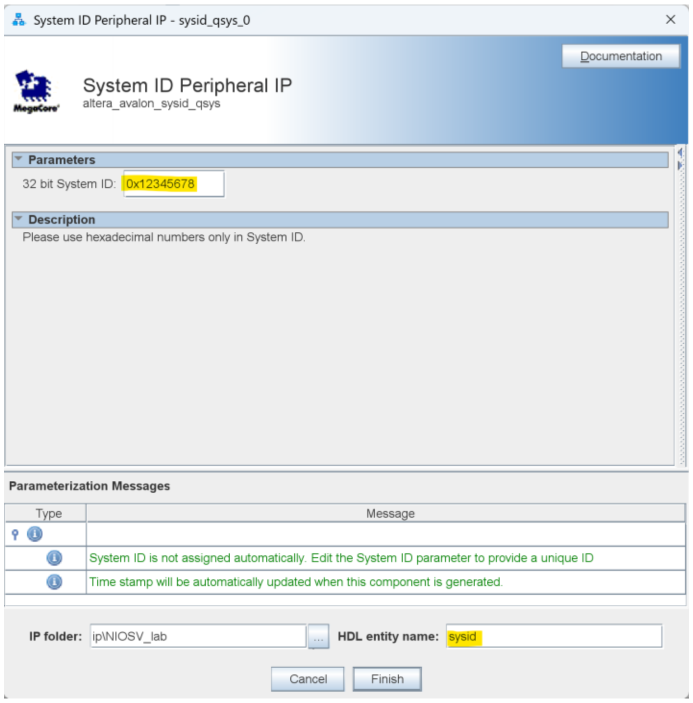
IP
Board
system
Project
New Component...
New Packaged Subsystem...
Library
Basic Functions
Simulation; Debug and Verification
Debug and Performance
In-System Sources & Probes Intel FPGA IP
System ID Peripheral IP
Trace System Intel FPGA IP

Figure 4-24 : Add System ID IP

Double-click the System ID Peripheral IP.

Change the 32-bit System ID field to *0x12345678*

Change the **HDL entity name** to ***sysid***. Click Finish.



32 bit System ID: 0x12345678

Please use hexadecimal numbers only in System ID.

System ID is not assigned automatically. Edit the System ID parameter to provide a unique ID

Time stamp will be automatically updated when this component is generated.

IP folder: ip\NIOSV_lab

HDL entity name: sysid

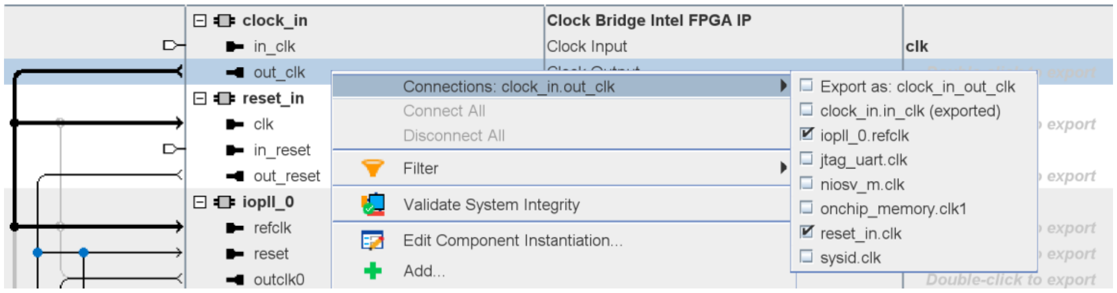
Figure 4-25 : Add System ID settings

At this point, you will see many errors and warnings in the bottom console indicating that various ports are not connected, and the memory addresses are not correct. Ignore them for now, as we will address them when we make connections.

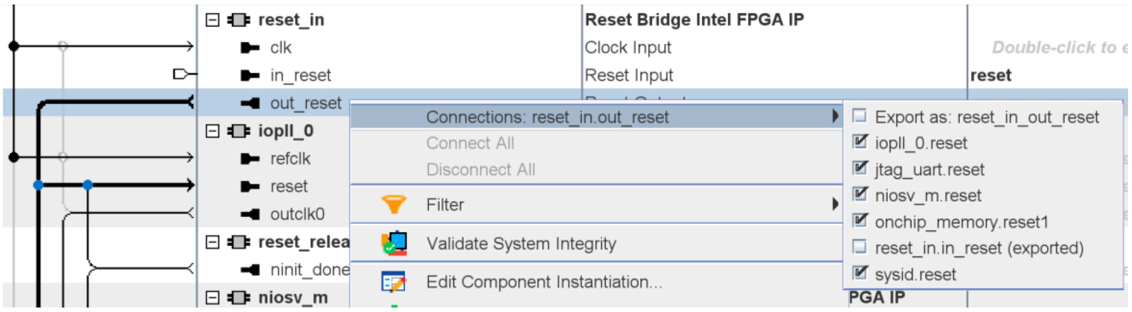
4.3.1.10 Connect the IP blocks and assign addresses

In the System View tab, make the following connections by clicking the intersection points. It is also possible to make the connections by right clicking and using the connections option shown in the images below.

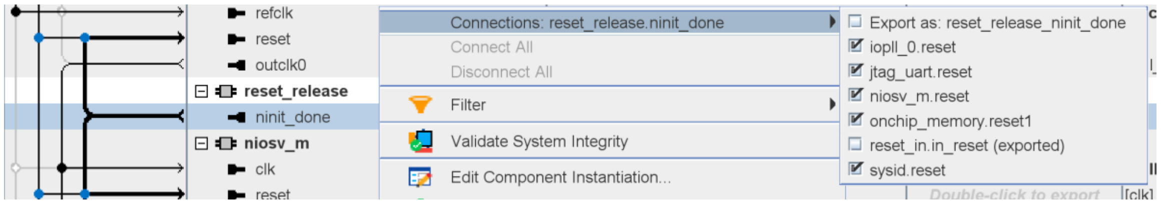
- Connect clock_in/out_clk to reset_in/clk and iopll_0/refclk.



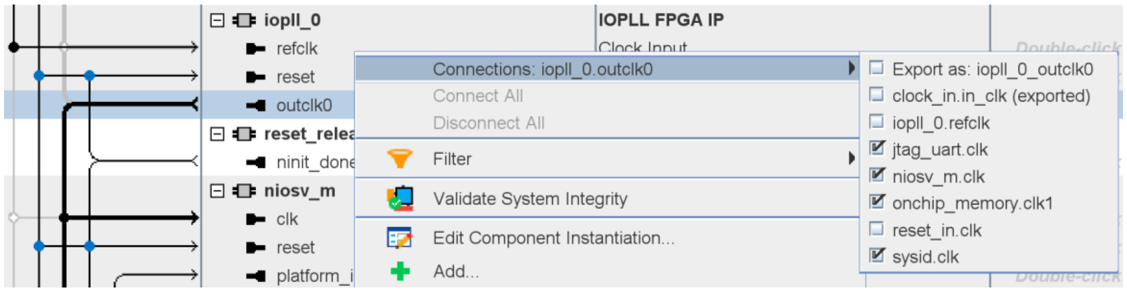
- Connect reset_in/out_reset to iopll_0/reset, lw_uart/reset, niosv_m/reset, onchip_memory/reset1, and sysid/reset.



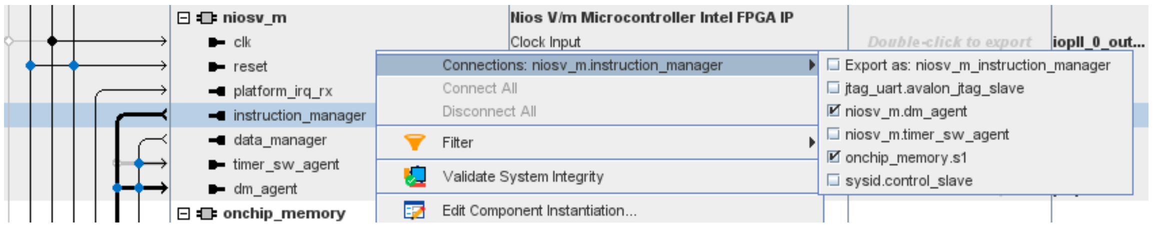
- Connect reset_release/ninit_done to iopll_0/reset, niosv_m/reset, onchip_memory/reset1, lw_uart/reset, and sysid/reset.



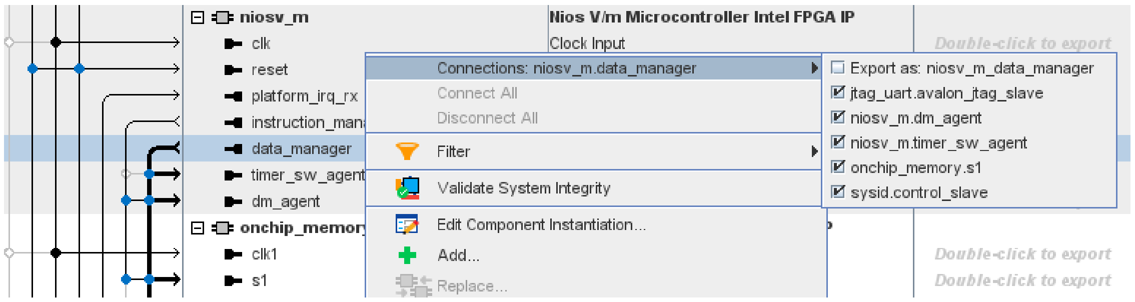
- Connect iopll_0/outclk0 to niosv_m/clk, onchip_memory/clk1, lw_uart/clk, and sysid/clk.



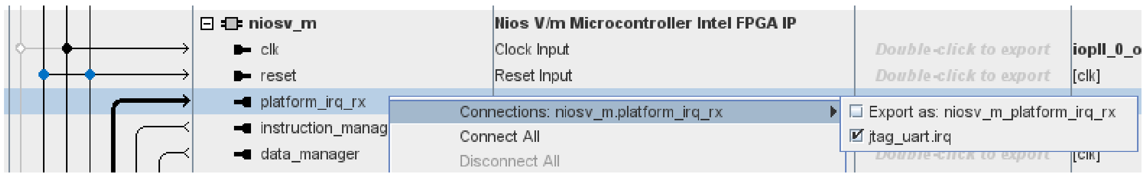
- Connect **niosv_m/instruction_manager** to **onchip_memory/s1**. This allows instruction execution from the onchip RAM.



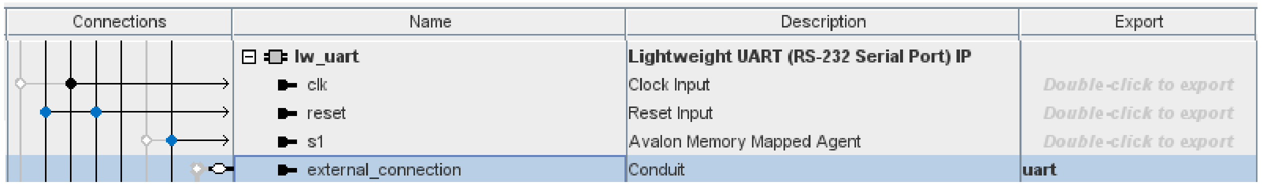
- Connect **niosv_m/data_manager** to **onchip_memory/s1**. This allows data fetch on from the onchip RAM.
- Connect niosv_m/data_manager to jtag_uart/avalon_lw_slave, and sysid/control_slave.



- Connect the niosv_m/platform_irq_rx to lw_uart/irq.



- Export the lw_uart signals. Enter uart in the export column of the external_connection field.



Connections	Name	Description	Export
	lw_uart	**Lightweight UART (RS-232 Serial Port) IP**	
	clk	Clock Input	Double-click to export
	reset	Reset Input	Double-click to export
	s1	Avalon Memory Mapped Agent	Double-click to export
	external_connection	Conduit	**uart**

From Platform Designer menu bar, execute **System → Assign Base Addresses**.

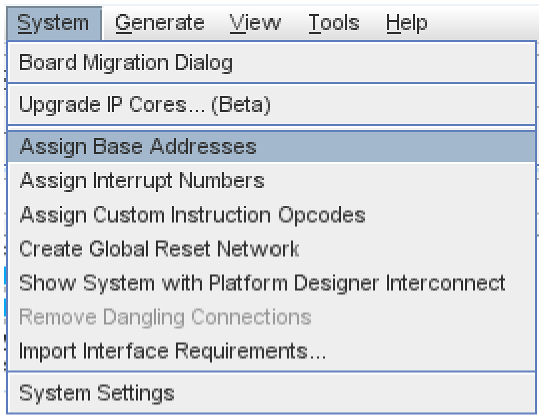
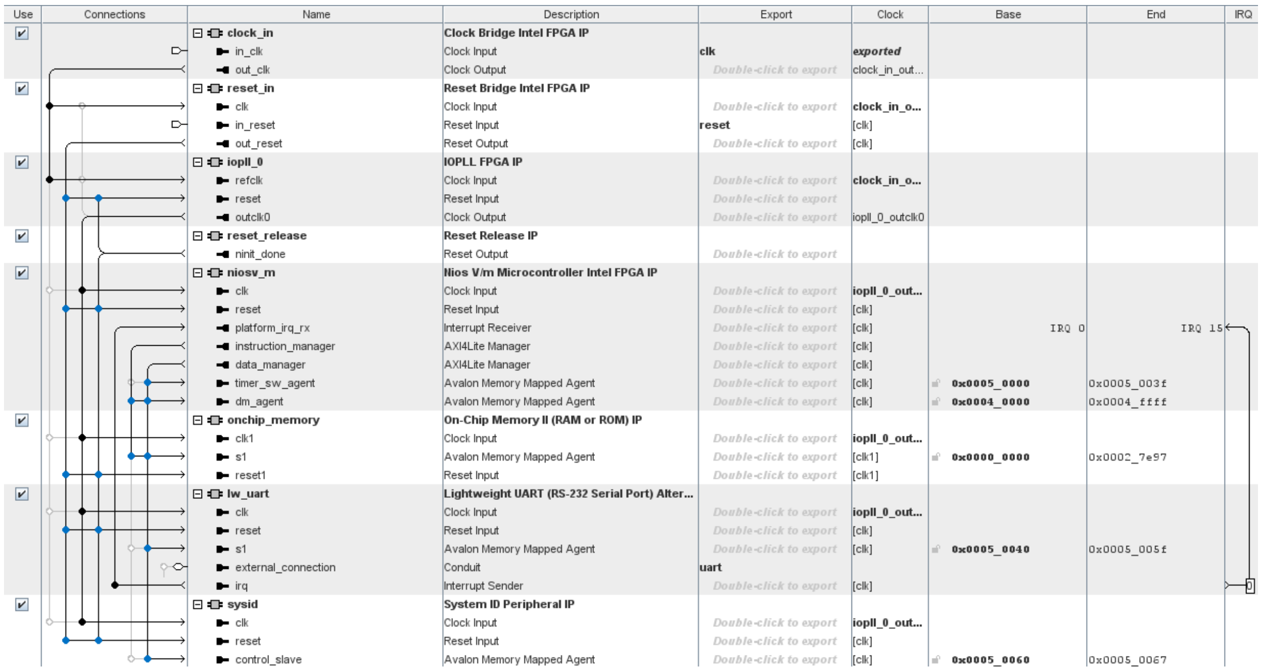

System
Generate
View
Tools
Help
Board Migration Dialog
Upgrade IP Cores... (Beta)
Assign Base Addresses
Assign Interrupt Numbers
Assign Custom Instruction Opcodes
Create Global Reset Network
Show System with Platform Designer Interconnect
Remove Dangling Connections
Import Interface Requirements...
System Settings

Figure 4-26 : Assign Base Addresses

When the **Sync System Infos** window completes, make sure it is successful with no errors. Click **Close**.

Ensure that the **onchip_memory** got assigned starting address ***0x0000_00000***. This is where the reset vector will be.

At this point, there should not be any connectivity errors. See figure below.



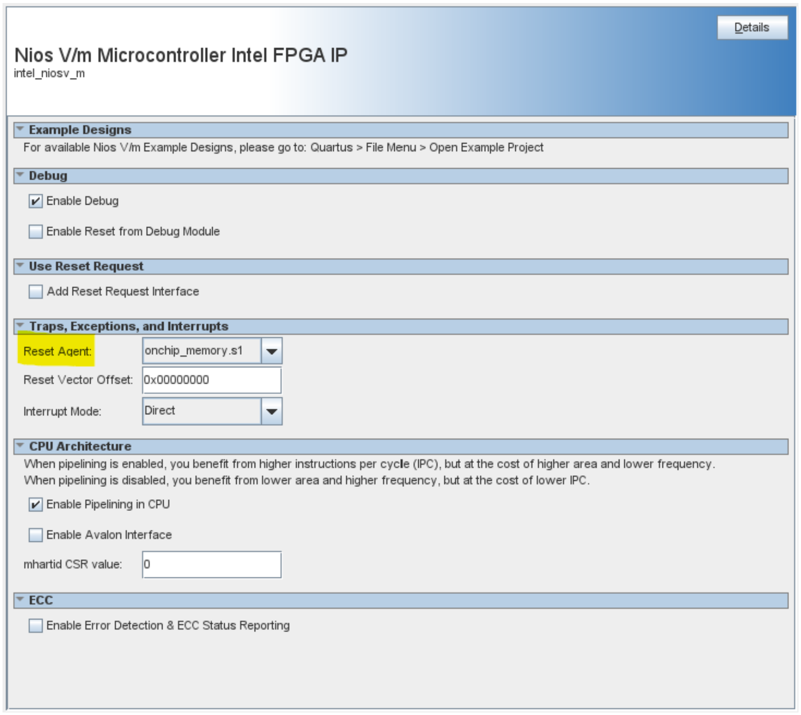
Use	Connections	Name	Description	Export	Clock	Base	End	IRQ
✓		**clock_in**	**Clock Bridge Intel FPGA IP**					
		in_clk	Clock Input	**clk**	***exported***			
		out_clk	Clock Output	*Double-click to export*	clock_in_out...			
✓		**reset_in**	**Reset Bridge Intel FPGA IP**					
		clk	Clock Input	*Double-click to export*	**clock_in_o...**			
		in_reset	Reset Input	**reset**	[clk]			
		out_reset	Reset Output	*Double-click to export*	[clk]			
✓		**iopll_0**	**IOPLL FPGA IP**					
		refclk	Clock Input	*Double-click to export*	**clock_in_o...**			
		reset	Reset Input	*Double-click to export*				
		outclk0	Clock Output	*Double-click to export*	iopll_0_outclk0			
✓		**reset_release**	**Reset Release IP**					
		ninit_done	Reset Output	*Double-click to export*				
✓		**niosv_m**	**Nios V/m Microcontroller Intel FPGA IP**					
		clk	Clock Input	*Double-click to export*	**iopll_0_out...**			
		reset	Reset Input	*Double-click to export*	[clk]			
		platform_irq_rx	Interrupt Receiver	*Double-click to export*	[clk]	IRQ 0	IRQ 15	
		instruction_manager	AXI4Lite Manager	*Double-click to export*	[clk]			
		data_manager	AXI4Lite Manager	*Double-click to export*	[clk]			
		timer_sw_agent	Avalon Memory Mapped Agent	*Double-click to export*	[clk]	**0x0005_0000**	0x0005_003f	
		dm_agent	Avalon Memory Mapped Agent	*Double-click to export*	[clk]	**0x0004_0000**	0x0004_ffff	
✓		**onchip_memory**	**On-Chip Memory II (RAM or ROM) IP**					
		clk1	Clock Input	*Double-click to export*	**iopll_0_out...**			
		s1	Avalon Memory Mapped Agent	*Double-click to export*	[clk1]	**0x0000_0000**	0x0002_7e97	
		reset1	Reset Input	*Double-click to export*	[clk1]			
✓		**lw_uart**	**Lightweight UART (RS-232 Serial Port) Alter...**					
		clk	Clock Input	*Double-click to export*	**iopll_0_out...**			
		reset	Reset Input	*Double-click to export*	[clk]			
		s1	Avalon Memory Mapped Agent	*Double-click to export*	[clk]	**0x0005_0040**	0x0005_005f	
		external_connection	Conduit	**uart**				
		irq	Interrupt Sender	*Double-click to export*	[clk]			0
✓		**sysid**	**System ID Peripheral IP**					
		clk	Clock Input	*Double-click to export*	**iopll_0_out...**			
		reset	Reset Input	*Double-click to export*	[clk]			
		control_slave	Avalon Memory Mapped Agent	*Double-click to export*	[clk]	**0x0005_0060**	0x0005_0067	

Figure 4-27 : System Connections

After Addresses are assigned, the Nios-V module needs to be re-visited to assign the Reset Vector to the proper agent.

- Double-click the **niosv_m** module to reveal its parameters page.

- Change the **Reset Agent** setting in the ***Vectors*** category to ***onchip_memory.s1***, thus setting it to point to the onchip RAM.


Nios V/m Microcontroller Intel FPGA IP
intel_niosv_m

Example Designs
For available Nios V/m Example Designs, please go to: Quartus > File Menu > Open Example Project

Debug
- [✓] Enable Debug
- [ ] Enable Reset from Debug Module

Use Reset Request
- [ ] Add Reset Request Interface

Traps, Exceptions, and Interrupts
Reset Agent: onchip_memory.s1
Reset Vector Offset: 0x00000000
Interrupt Mode: Direct

CPU Architecture
When pipelining is enabled, you benefit from higher instructions per cycle (IPC), but at the cost of higher area and lower frequency.
When pipelining is disabled, you benefit from lower area and higher frequency, but at the cost of lower IPC.
- [✓] Enable Pipelining in CPU
- [ ] Enable Avalon Interface

mhartid CSR value: 0

ECC
- [ ] Enable Error Detection & ECC Status Reporting

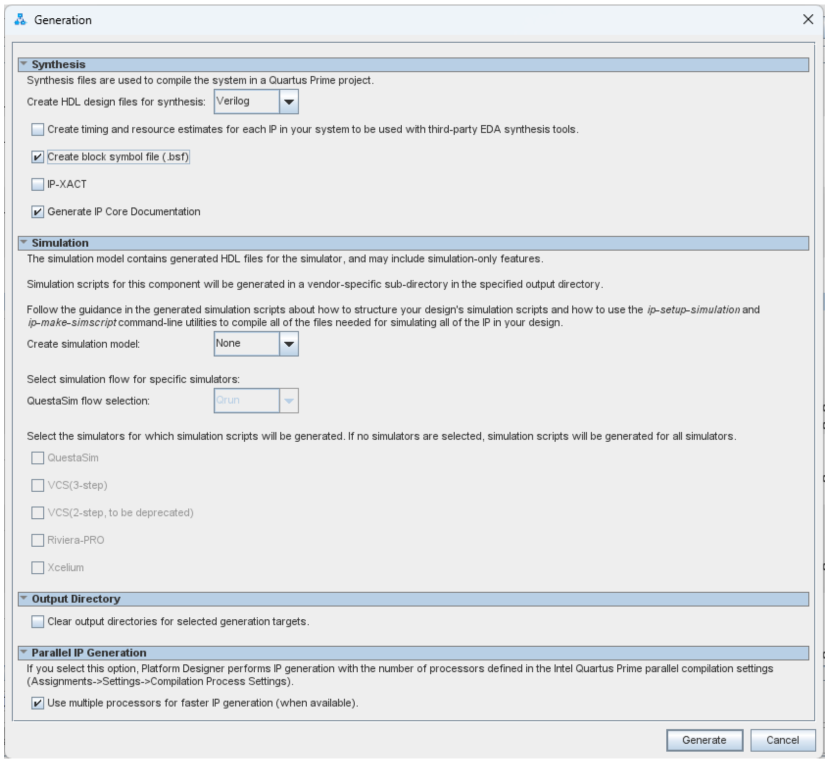
Figure 4-28 : Nios-V Reset Vector setting

Review the message window to make sure there are no errors.

4.3.2 Generate HDL for Platform Designer system

This stage generates RTL files that Quartus can use to compile the design.

- In the Platform Designer window, click the **Generate HDL ...** button in the lower right-hand corner to extract HDL code from the IP blocks added.
- In the Generation pop-up window, set the **Create HDL design files for synthesis** to ***Verilog***.
- Check the box for Create block symbol file (.bsf) and keep all other default settings.
- Click the **Generate** button.


Generation

Synthesis
Synthesis files are used to compile the system in a Quartus Prime project.
Create HDL design files for synthesis: Verilog
☐ Create timing and resource estimates for each IP in your system to be used with third-party EDA synthesis tools.
☑ Create block symbol file (.bsf)
☐ IP-XACT
☑ Generate IP Core Documentation

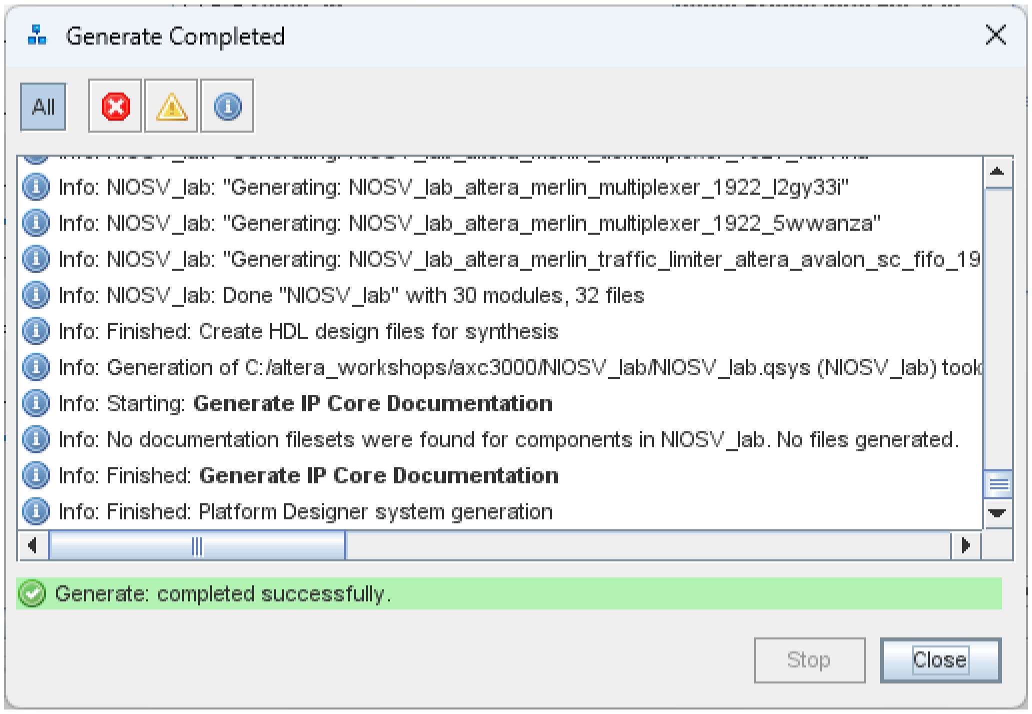
Simulation
The simulation model contains generated HDL files for the simulator, and may include simulation-only features.
Simulation scripts for this component will be generated in a vendor-specific sub-directory in the specified output directory.
Follow the guidance in the generated simulation scripts 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
Select simulation flow for specific simulators:
QuestaSim flow selection: Qrun
Select the simulators for which simulation scripts will be generated. If no simulators are selected, simulation scripts will be generated for all simulators.
☐ QuestaSim
☐ VCS(3-step)
☐ VCS(2-step, to be deprecated)
☐ Riviera-PRO
☐ Xcelium

Output Directory
☐ Clear output directories for selected generation targets.

Parallel IP Generation
If you select this option, Platform Designer performs IP generation with the number of processors defined in the Intel Quartus Prime parallel compilation settings (Assignments->Settings->Compilation Process Settings).
☑ Use multiple processors for faster IP generation (when available).

Figure 4-29 : HDL Generation

- When prompted to save, click **Yes**.
- The **Generate** window should complete successfully, with no errors. Click **Close**.
- Minimize the Platform Designer Window since the next part of the Lab will be done in Quartus.



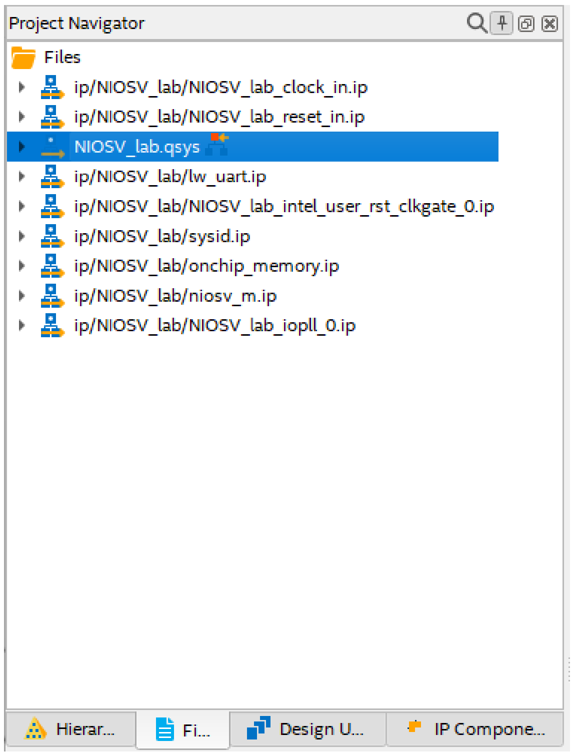
```
Info: NIOSV_lab: "Generating: NIOSV_lab_altera_merlin_multiplexer_1922_l2gy33i"
Info: NIOSV_lab: "Generating: NIOSV_lab_altera_merlin_multiplexer_1922_5wwanza"
Info: NIOSV_lab: "Generating: NIOSV_lab_altera_merlin_traffic_limiter_altera_avalon_sc_fifo_19
Info: NIOSV_lab: Done "NIOSV_lab" with 30 modules, 32 files
Info: Finished: Create HDL design files for synthesis
Info: Generation of C:/altera_workshops/axc3000/NIOSV_lab/NIOSV_lab.qsys (NIOSV_lab) took
Info: Starting: Generate IP Core Documentation
Info: No documentation filesets were found for components in NIOSV_lab. No files generated.
Info: Finished: Generate IP Core Documentation
Info: Finished: Platform Designer system generation
```

Generate: completed successfully.

Figure 4-30 : Generate Complete

4.4 Complete the Quartus Design

During the creation of the new project, the top-level design entity is named NIOSV_lab . Therefore, the file **NIOSV_lab.qsys** is automatically the top-level design entity. Verify the assignment of the top-level design entity in the Project Navigator ***Files*** tab.


Project Navigator

Files
- ip/NIOSV_lab/NIOSV_lab_clock_in.ip
- ip/NIOSV_lab/NIOSV_lab_reset_in.ip
- NIOSV_lab.qsys
- ip/NIOSV_lab/lw_uart.ip
- ip/NIOSV_lab/NIOSV_lab_intel_user_rst_clkgate_0.ip
- ip/NIOSV_lab/sysid.ip
- ip/NIOSV_lab/onchip_memory.ip
- ip/NIOSV_lab/niosv_m.ip
- ip/NIOSV_lab/NIOSV_lab_iopll_0.ip

Figure 4-31 : Top-Level selection

The compile process consists of the steps shown below:

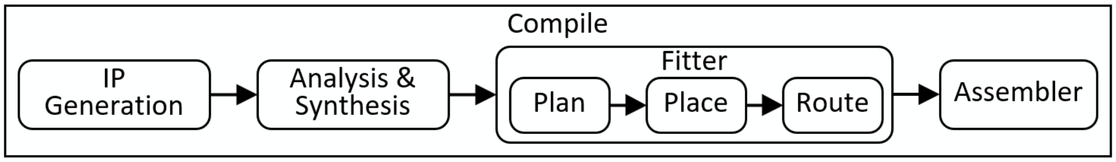


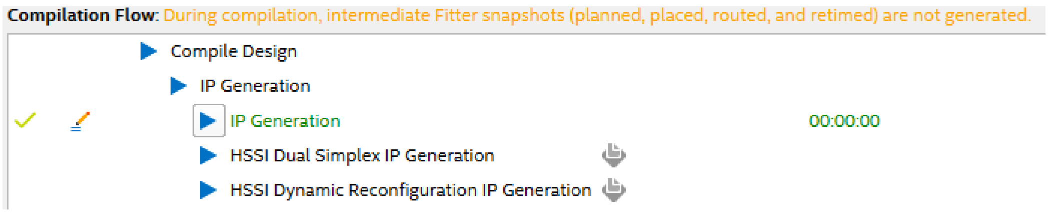
Figure 4-32 : Compile Process

We will be compiling the design in stages.

4.4.1 IP Generation

The IP Generation step makes sure that HDL code exists for all IP blocks. If not, it will generate them.

Click the IP Generation Blue arrow in the ***Compilation Flow*** window.


Compilation Flow: During compilation, intermediate Fitter snapshots (planned, placed, routed, and retimed) are not generated.

```
Compile Design
  IP Generation
✓   IP Generation                                   00:00:00
    HSSI Dual Simplex IP Generation
    HSSI Dynamic Reconfiguration IP Generation
```

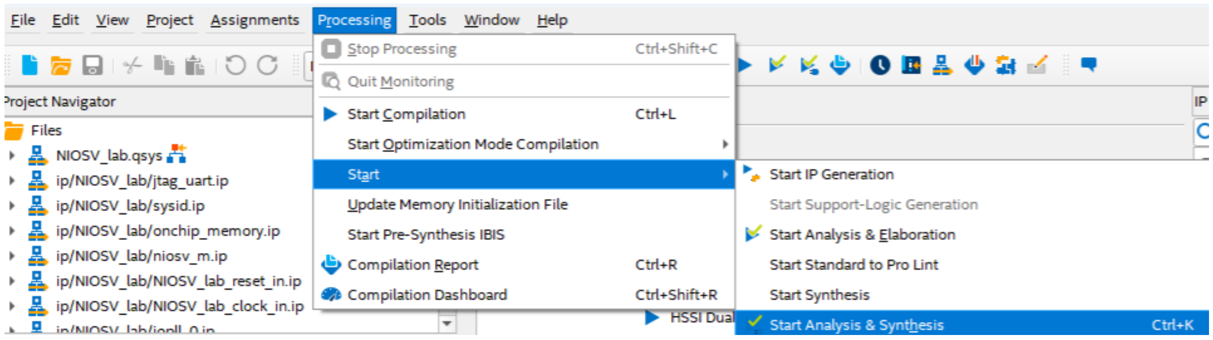
Figure 4-33 : IP Generation step

There should not be any errors generated by this step.

4.4.2 Analysis and Synthesis

Analysis and Synthesis, is the step where the design files are converted into a netlist that Quartus can use in later steps. Quartus also checks the legality of the connections. The netlist is used by Quartus so that IO constraints can be assigned.

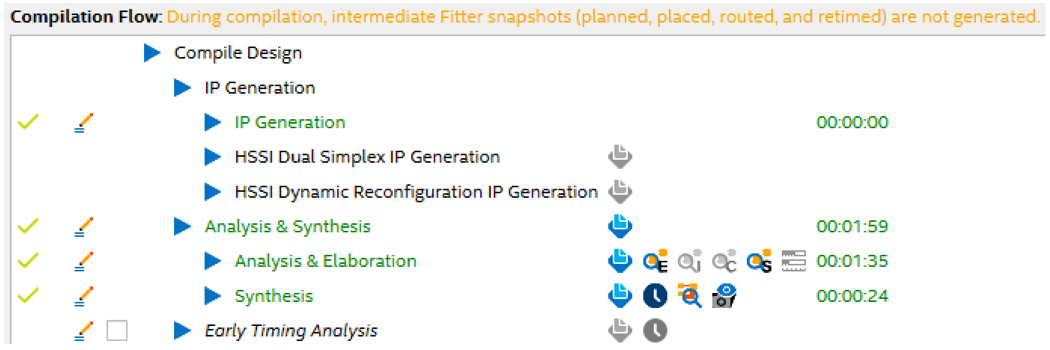
This step creates a database in Quartus which allows us to assign I/O and other constraints. On the Quartus menu bar, select **Processing -> Start -> Start Analysis and Synthesis**, or click the icon.



File
Edit
View
Project
Assignments
Processing
Tools
Window
Help
Project Navigator
Files
NIOSV_lab.qsys
ip/NIOSV_lab/jtag_uart.ip
ip/NIOSV_lab/sysid.ip
ip/NIOSV_lab/onchip_memory.ip
ip/NIOSV_lab/niosv_m.ip
ip/NIOSV_lab/NIOSV_lab_reset_in.ip
ip/NIOSV_lab/NIOSV_lab_clock_in.ip
Stop Processing Ctrl+Shift+C
Quit Monitoring
Start Compilation Ctrl+L
Start Optimization Mode Compilation
Start
Update Memory Initialization File
Start Pre-Synthesis IBIS
Compilation Report Ctrl+R
Compilation Dashboard Ctrl+Shift+R
Start IP Generation
Start Support-Logic Generation
Start Analysis & Elaboration
Start Standard to Pro Lint
Start Synthesis
Start Analysis & Synthesis Ctrl+K
HSSI Dual

Figure 4-34 : Run Analysis & Synthesis

Upon completion, there should be green check marks to the left of the Analysis & Synthesis steps.


Compilation Flow: During compilation, intermediate Fitter snapshots (planned, placed, routed, and retimed) are not generated.

```
Compile Design
  IP Generation
✓   IP Generation                                   00:00:00
    HSSI Dual Simplex IP Generation
    HSSI Dynamic Reconfiguration IP Generation
✓ Analysis & Synthesis                              00:01:59
✓   Analysis & Elaboration                          00:01:35
✓   Synthesis                                       00:00:24
  Early Timing Analysis
```

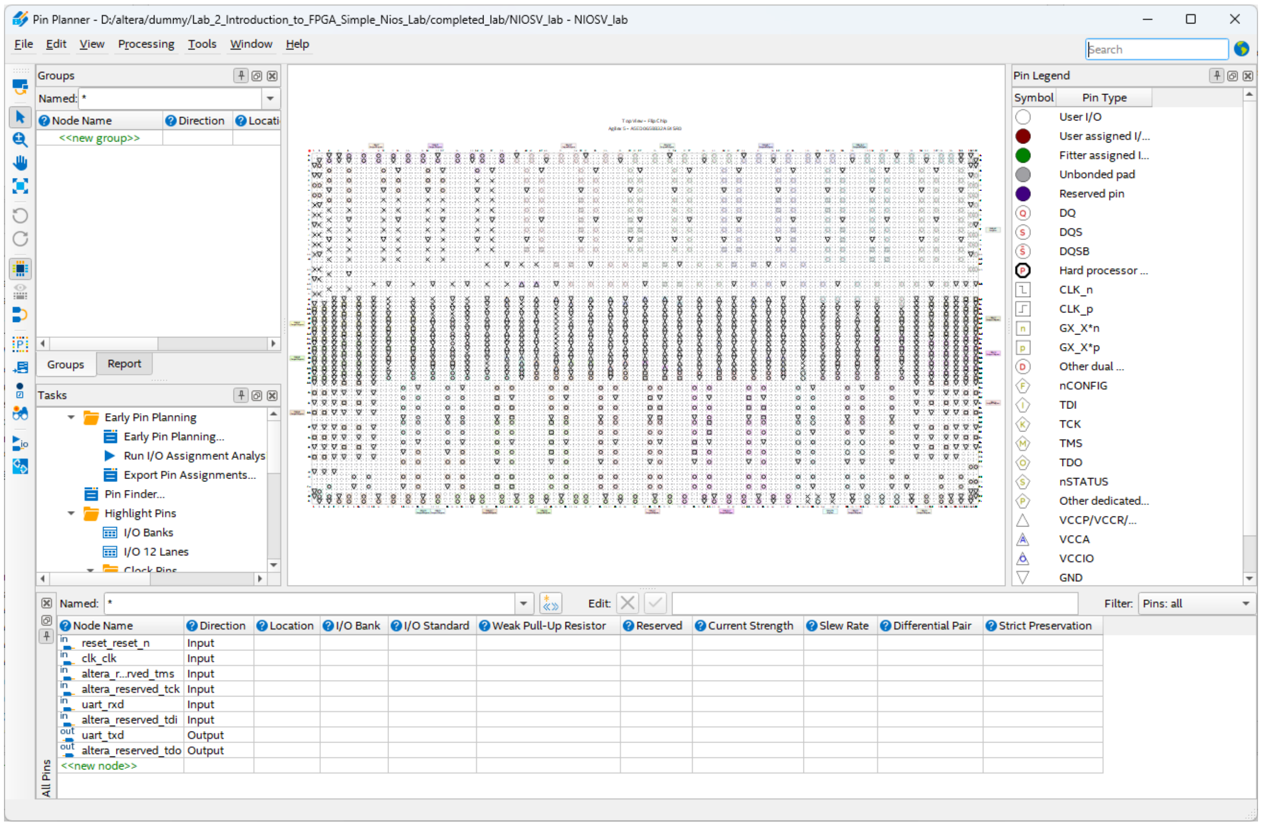
Figure 4-35 : Successful Analysis & Synthesis steps

4.4.3 Pin Assignment

Pin Assignments are needed so the design ports are assigned to the correct pins on the board. Without pin assignments, Quartus will use default pin assignments which will not correspond to the board pinout, and thus cause issues.

4.4.3.1 Open Pin Planner

Open the **Pin Planner** by clicking on symbol on the toolbar, or **Assignments → Pin Planner**



Named: *

Node Name	Direction	Location	I/O Bank	I/O Standard	Weak Pull-Up Resistor	Reserved	Current Strength	Slew Rate	Differential Pair	Strict Preservation
reset_reset_n	Input									
clk_clk	Input									
altera_r...rved_tms	Input									
altera_reserved_tck	Input									
uart_rxd	Input									
altera_reserved_tdi	Input									
uart_txd	Output									
altera_reserved_tdo	Output									
<<new node>>										

Figure 4-36 : Pin Planner

4.4.3.2 Assign Pins

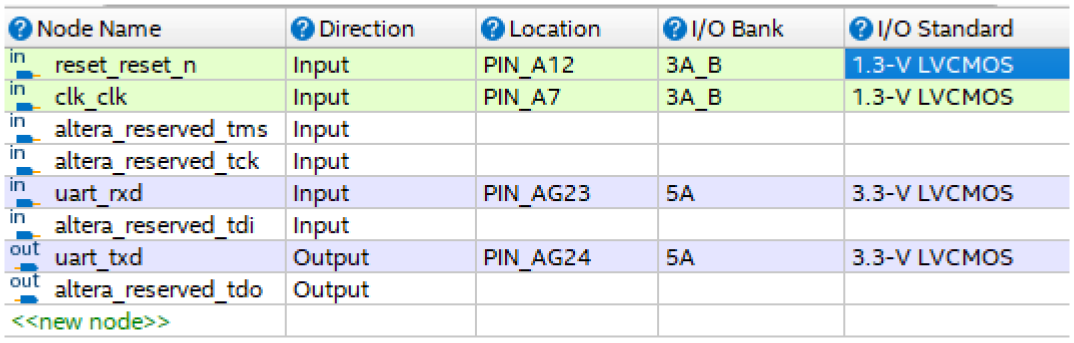
The bottom section has a table which lists the design nodes that can be assigned to pins. Double-click the **Location** cell for node ***clk_clk*** and type ***A7***.The "**PIN_**" prefix gets added by Quartus. Set its **I/O Standard** to ***1.3-V LVCMOS***.

Double-click the **Location** cell for node ***reset_reset_n*** and type ***A12***.The "**PIN_**" prefix gets added by Quartus. Set its **I/O Standard** to ***1.3-V LVCMOS***. This pin is connected to a push-button labeled **S2** on the board.

Double-click the **Location** cell for node ***uart_rxd*** and type ***AG23***.The "**PIN_**" prefix gets added by Quartus. Set its **I/O Standard** to ***3.3-V LVCMOS***. This pin is connected to a UART to USB communication device.

Double-click the **Location** cell for node ***uart_txd*** and type ***AG24***.The "**PIN_**" prefix gets added by Quartus. Set its **I/O Standard** to ***3.3-V LVCMOS***. This pin is connected to a UART to USB communication device.

The JTAG pins are *reserved* and Quartus takes care of them. Leave them open.



Node Name	Direction	Location	I/O Bank	I/O Standard
reset_reset_n	Input	PIN_A12	3A_B	1.3-V LVCMOS
clk_clk	Input	PIN_A7	3A_B	1.3-V LVCMOS
altera_reserved_tms	Input			
altera_reserved_tck	Input			
uart_rxd	Input	PIN_AG23	5A	3.3-V LVCMOS
altera_reserved_tdi	Input			
uart_txd	Output	PIN_AG24	5A	3.3-V LVCMOS
altera_reserved_tdo	Output			
<<new node>>				

Figure 4-37 : Pin Assignment

4.4.3.3 Customize Columns

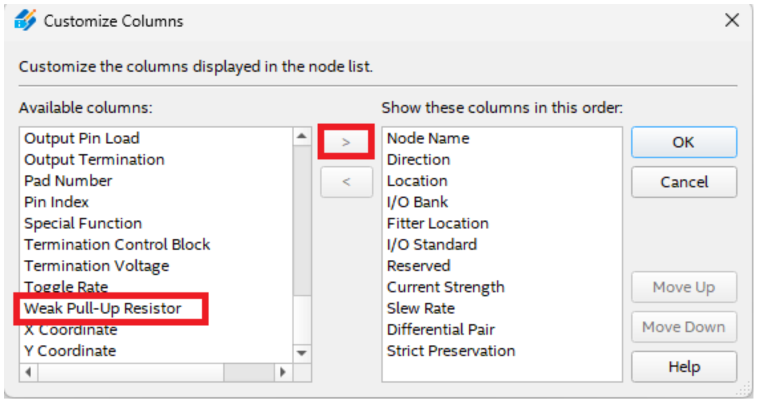
Right click in the spreadsheet part of Pin Planner. Select Customize Columns.by selecting File → Close. The settings are automatically saved.

4.4.3.4 Add the **Weak Pull-Up Resistor** column.

Select Weak Pull-up Resistor in Available Columns and press the > button. Select Weak Pull-up Resistor in the right hand column and then use the Move-Up button to place at under the I/O Standard column selection.

4.4.3.5 Close the Customize Columns window.

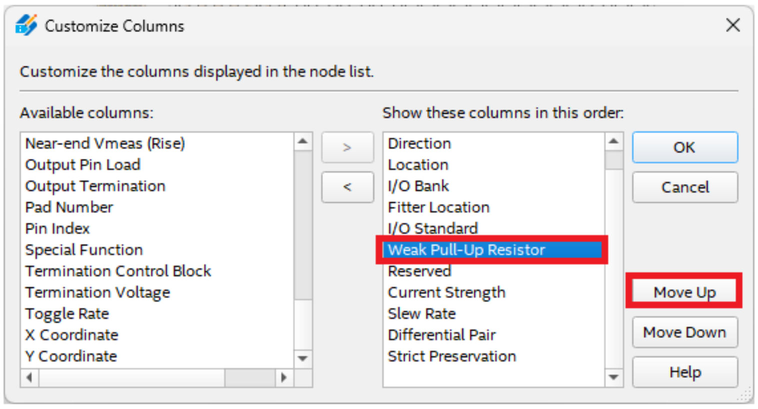
Select **OK**.


Customize Columns

Customize the columns displayed in the node list.

Available columns:
- Output Pin Load
- Output Termination
- Pad Number
- Pin Index
- Special Function
- Termination Control Block
- Termination Voltage
- Toggle Rate
- Weak Pull-Up Resistor
- X Coordinate
- Y Coordinate

Show these columns in this order:
- Node Name
- Direction
- Location
- I/O Bank
- Fitter Location
- I/O Standard
- Reserved
- Current Strength
- Slew Rate
- Differential Pair
- Strict Preservation


Customize Columns

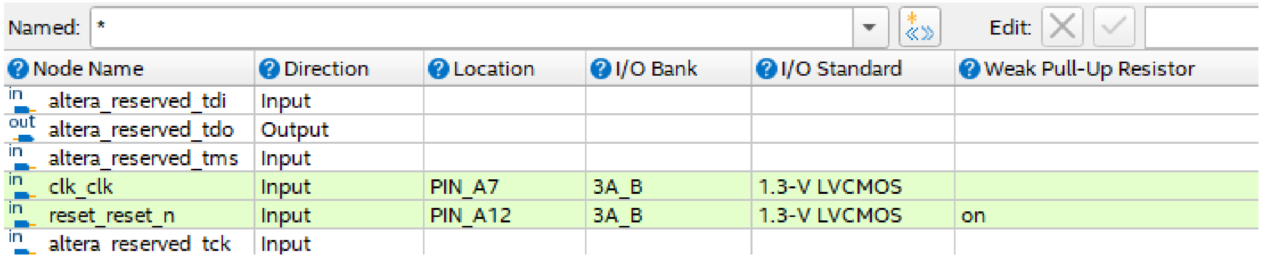
Customize the columns displayed in the node list.

Available columns:
- Near-end Vmeas (Rise)
- Output Pin Load
- Output Termination
- Pad Number
- Pin Index
- Special Function
- Termination Control Block
- Termination Voltage
- Toggle Rate
- X Coordinate
- Y Coordinate

Show these columns in this order:
- Direction
- Location
- I/O Bank
- Fitter Location
- I/O Standard
- Weak Pull-Up Resistor
- Reserved
- Current Strength
- Slew Rate
- Differential Pair
- Strict Preservation

4.4.3.6 Enable Pullup

Select on in the Weak Pull-up Resistor column for the reset_reset_n Node Name


Named: *

Edit:

Node Name	Direction	Location	I/O Bank	I/O Standard	Weak Pull-Up Resistor
altera_reserved_tdi	Input				
altera_reserved_tdo	Output				
altera_reserved_tms	Input				
clk_clk	Input	PIN_A7	3A_B	1.3-V LVCMOS	
reset_reset_n	Input	PIN_A12	3A_B	1.3-V LVCMOS	on
altera_reserved_tck	Input				

Close the Pin Planner. The settings are automatically saved.

5 Software Project with AXC3000 board

Once the processor hardware system is ready, you can start building the software design using the **Ashling RiscFree™ IDE software for Intel FPGAs**.

RiscFree IDE is an Eclipse-based IDE that supports C and C++ software development. With the RiscFree environment, there are three main steps consisting of:

a) Creating a C project environment with the hardware project imported,

b) building an application C code

c) compiling the code

Before the RiscFree IDE can be used, there are some set up items that need to be done first.

5.1 Create a New Software Project

To ensure an optimized compilation flow, it is recommended that you create a directory tree within your working directory. In your design project folder, create a folder named **software**. In the **software** folder, create two folders named ***app*** and ***bsp***.

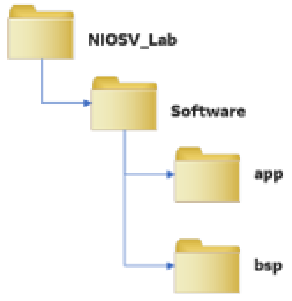


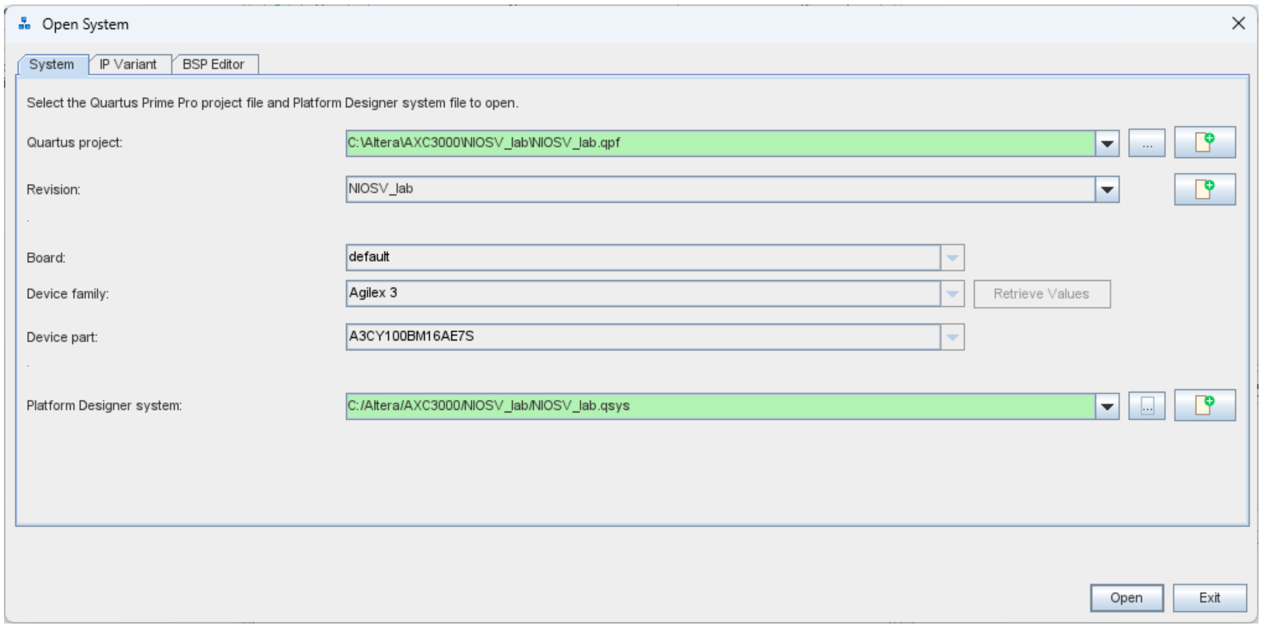
Figure 5-1 : Software folder tree structure

The ***app*** folder will have the C program and the ***bsp*** folder will have the Board Support Package data.

5.2 Generate the project BSP

The Board Support Package (BSP) is an integral part of the software development. It takes in the memory-mapped addresses assigned to memory and peripherals attached to the CPU bus and creates a Hardware Abstraction Layer (HAL) which contains the system definitions in a file called **system.h**, and collects the available API routines for the peripherals being used.

The BSP generation takes place in Platform Designer (PD). If PD is not already open with the NIOS-V system loaded, open it with **Tools → Platform Designer**. In the **Platform Designer system** entry box, navigate to ***NIOSV_lab.qsys*** and **select** it. Click **Open**.



Select the Quartus Prime Pro project file and Platform Designer system file to open.

Quartus project: C:\Altera\AXC3000\NIOSV_lab\NIOSV_lab.qpf

Revision: NIOSV_lab

Board: default

Device family: Agilex 3

Device part: A3CY100BM16AE7S

Platform Designer system: C:/Altera/AXC3000/NIOSV_lab/NIOSV_lab.qsys

Figure 5-2 : Open System window from Platform Designer

Once the system is opened, you can **close** the dialog box.

In the Platform Designer window, go to **File → New BSP**.

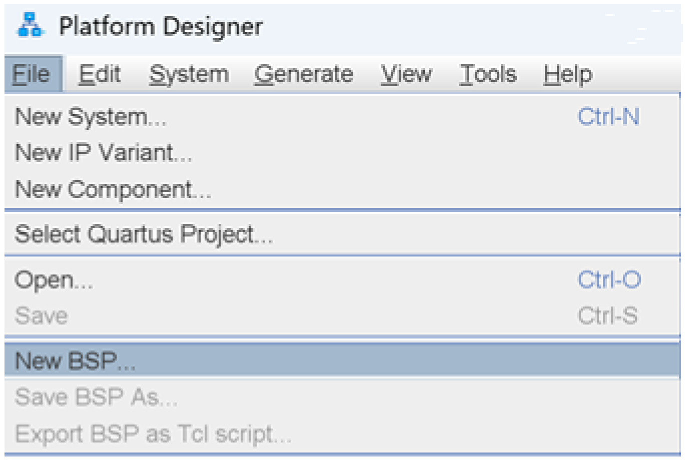
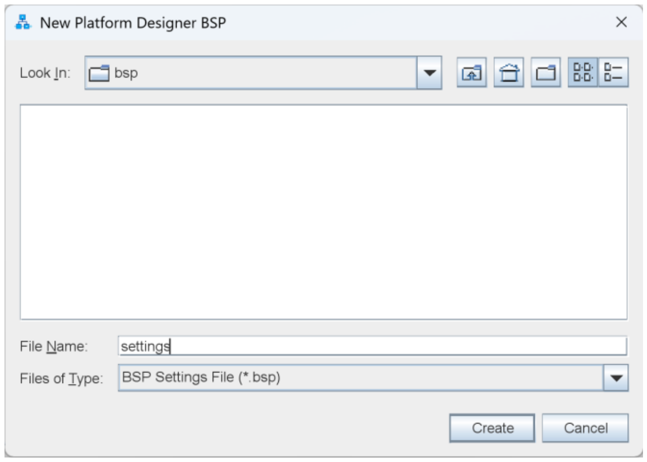


Figure 5-3 : New BSP

You will be asked to select the BSP name. Navigate to the **software → bsp** folder and use the file name ***settings.*** Click the **Create** button.



New Platform Designer BSP

Look In: bsp

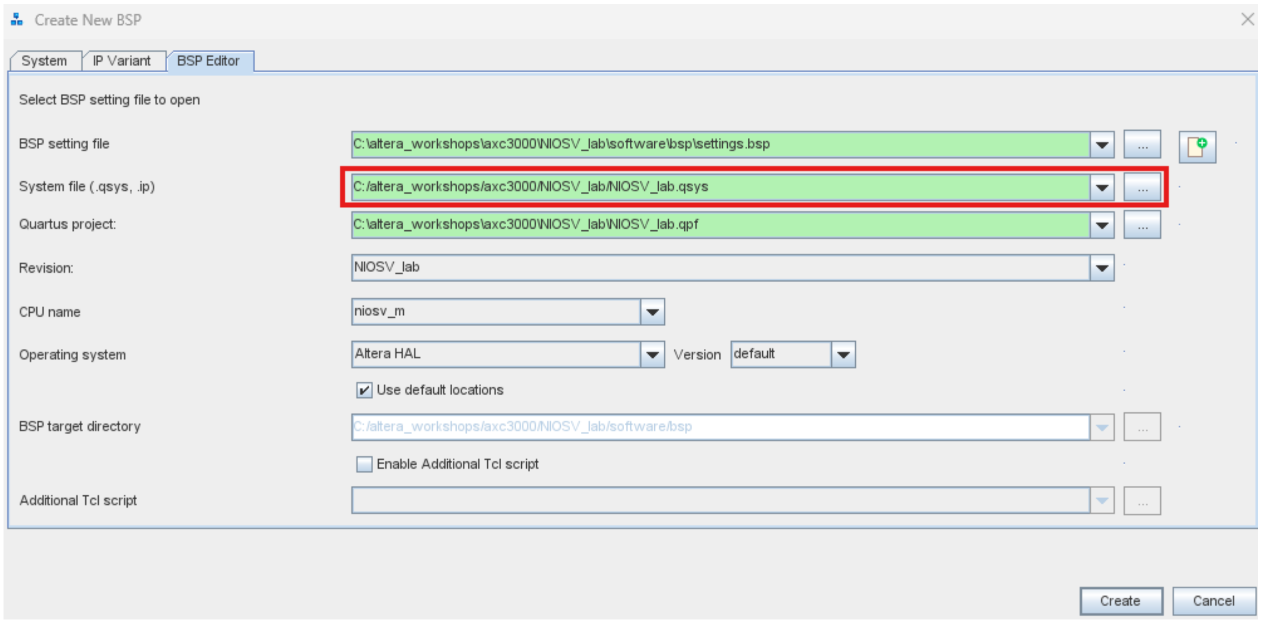
File Name: settings

Files of Type: BSP Settings File (*.bsp)

Figure 5-4 : BSP Settings file name

The next entry is to select the **System file**.

Navigate to the ***NIOSV_lab.qsys*** file and select it.



Create New BSP

Select BSP setting file to open

BSP setting file: C:\altera_workshops\axc3000\NIOSV_lab\software\bsp\settings.bsp

System file (.qsys, .ip): C:/altera_workshops/axc3000/NIOSV_lab/NIOSV_lab.qsys

Quartus project: C:\altera_workshops\axc3000\NIOSV_lab\NIOSV_lab.qpf

Revision: NIOSV_lab

CPU name: niosv_m

Operating system: Altera HAL Version: default

[✓] Use default locations

BSP target directory: C:/altera_workshops/axc3000/NIOSV_lab/software/bsp

[ ] Enable Additional Tcl script

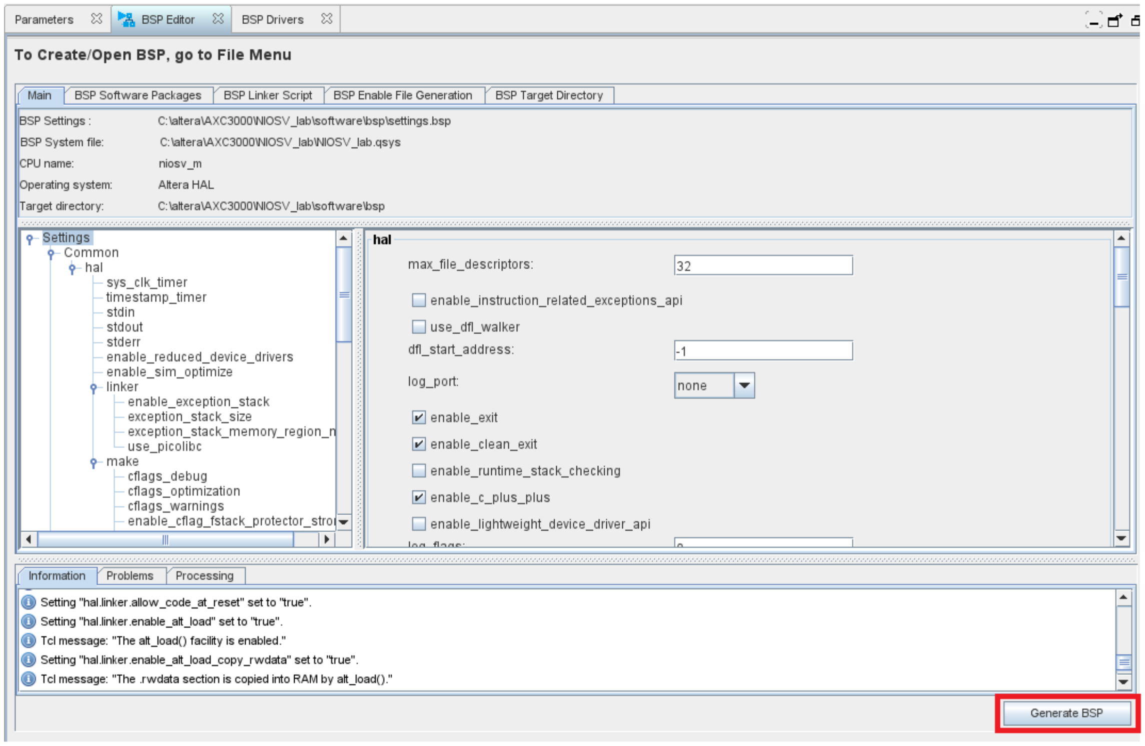
Additional Tcl script:

Figure 5-5 : Select System File name

Click **Create**.

When the BSP creation operation is done, the BSP Editor will display on the right-hand side of the Platform Designer.

Click the **Generate BSP** button. You may have to make that side of the window wider to see this button.



BSP Settings : C:\altera\AXC3000\NIOSV_lab\software\bsp\settings.bsp
BSP System file: C:\altera\AXC3000\NIOSV_lab\NIOSV_lab.qsys
CPU name: niosv_m
Operating system: Altera HAL
Target directory: C:\altera\AXC3000\NIOSV_lab\software\bsp

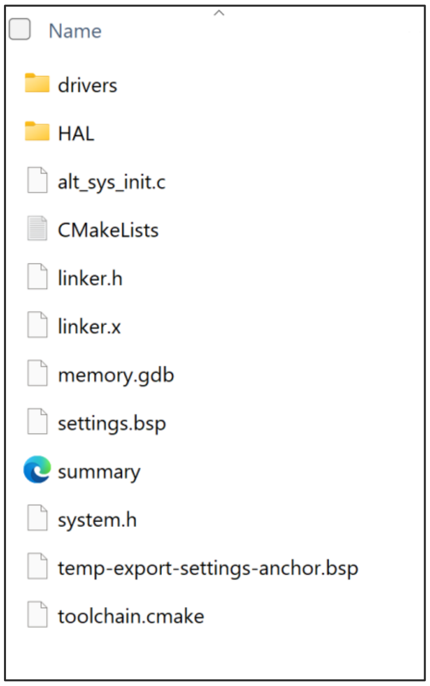
```
Settings
  Common
    hal
      sys_clk_timer
      timestamp_timer
      stdin
      stdout
      stderr
      enable_reduced_device_drivers
      enable_sim_optimize
      linker
        enable_exception_stack
        exception_stack_size
        exception_stack_memory_region_n
        use_picolibc
      make
        cflags_debug
        cflags_optimization
        cflags_warnings
        enable_cflag_fstack_protector_stro
```

hal
max_file_descriptors: 32
☐ enable_instruction_related_exceptions_api
☐ use_dfl_walker
dfl_start_address: -1
log_port: none
☑ enable_exit
☑ enable_clean_exit
☐ enable_runtime_stack_checking
☑ enable_c_plus_plus
☐ enable_lightweight_device_driver_api

```
Setting "hal.linker.allow_code_at_reset" set to "true".
Setting "hal.linker.enable_alt_load" set to "true".
Tcl message: "The alt_load() facility is enabled."
Setting "hal.linker.enable_alt_load_copy_rwdata" set to "true".
Tcl message: "The .rwdata section is copied into RAM by alt_load()."
```

Figure 5-6 : Generate BSP

The BSP files are generated in the ***software/bsp*** folder.



Name
- drivers
- HAL
- alt_sys_init.c
- CMakeLists
- linker.h
- linker.x
- memory.gdb
- settings.bsp
- summary
- system.h
- temp-export-settings-anchor.bsp
- toolchain.cmake

Figure 5-7 : BSP files

The **system.h** files describes Platform Designer peripherals. **alt_sys_init.c** has initialization drivers. **linker.h** describes the memory layout.

5.3 Building software project using Ashling RiscFree IDE for Intel FPGA

We already have a **software/app** folder.

- We will create a **HelloWorld.c** file and place in it.
- The application project needs to be created using the ***settings.bsp*** files created earlier. This creates more files needed by the Ashling RiscFree compiler.

5.3.1 Create the C code program

In the software directory ***software/app***, create a C source file with code below and name it **HelloWorld.c**. For convenience and correctness, copy and paste the code below in Figure 57.

The BSP generation added some Hardware Abstraction Layer (HAL) API routines for you to use.

The **system.h** file has the address assigned to **SYSID_BASE** variable.

The IORD_ALTERA_AVALON_SYSID_QSYS_ID command is an API found in altera_avalon_sysid_qsys.h.

```
#include "system.h"
#include "stdio.h"
#include "alt_types.h"
#include <sys/alt_sys_init.h>
#include "altera_avalon_sysid_qsys.h"
#include "altera_avalon_sysid_qsys_regs.h"

alt_u32   temp_data;

int main()
{
   // Initialize all of the HAL drivers used in this design
   alt_sys_init();

   // Read and display the system ID value created in hardware
   temp_data = IORD_ALTERA_AVALON_SYSID_QSYS_ID(SYSID_BASE);
   printf("\r\nHello World! My System ID is: 0x%08lX\r\n", temp_data);

   while (1);

   return 0;
}
```

Figure 5-8 : Application C code

5.3.2 Create the Application Project

In addition to the BSP, the software project needs an application project which is created using a CMake flow. It uses the C code program, the BSP, and the HAL to generate an application environment used by the Ashling RiscFree IDE.

5.3.2.1 Open a Nios V Command Shell

Open the Nios V command shell from the Windows Start button -> All apps -> Altera 25.1.1.125 Pro Edition -> Nios V Command Shell (Quartus Prime Pro 25.1.1)

This will open a command shell window.

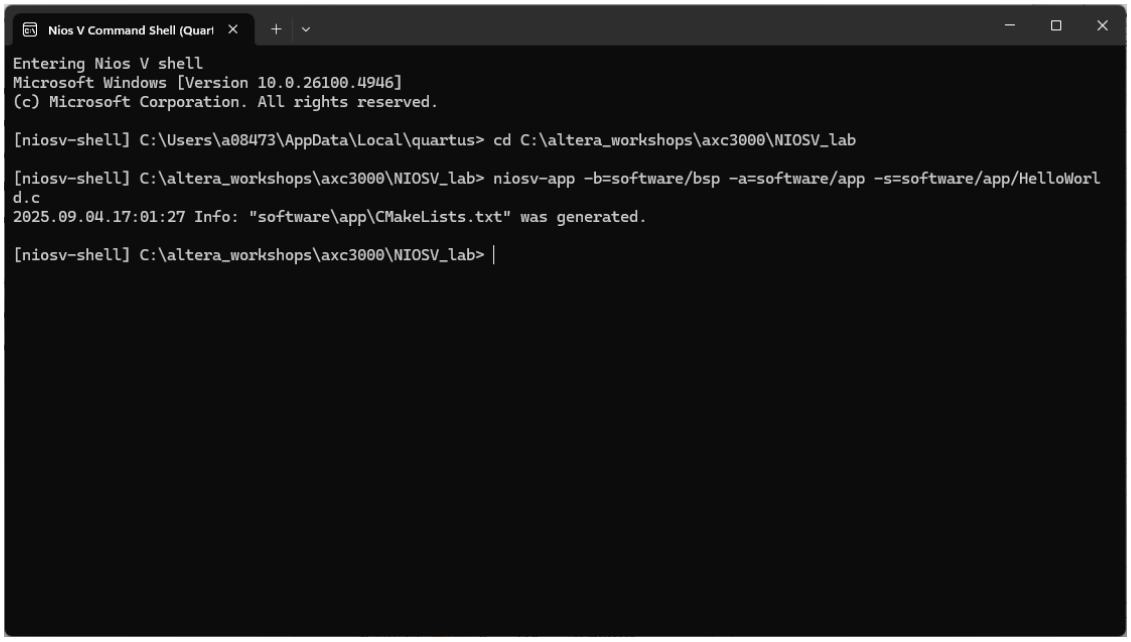
Navigate to the project folder C:\altera_workshops\axc3000\NIOSV_lab

You'll need to remove the existing local path before entering the project folder path. To do this, type cd in the command shell, followed by the full path to your project folder (see example in Figure 5-9).

5.3.2.2 Create Application

In the Nios V command shell, execute the command shown below,
to generate the CMake application file ***CMakeLists.txt***. This file is used by the Ashling RiscFree compiler.

```
niosv-app -b=software/bsp -a=software/app -s=software/app/HelloWorld.c
```

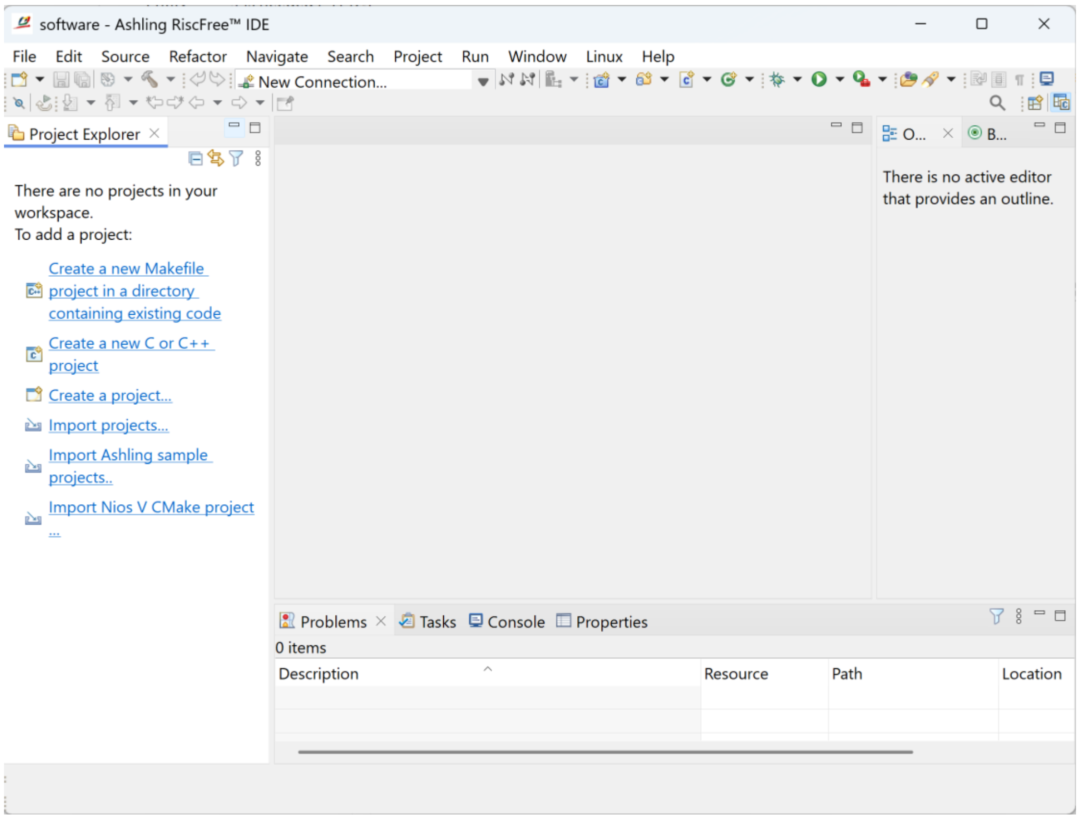


```
Entering Nios V shell
Microsoft Windows [Version 10.0.26100.4946]
(c) Microsoft Corporation. All rights reserved.

[niosv-shell] C:\Users\a08473\AppData\Local\quartus> cd C:\altera_workshops\axc3000\NIOSV_lab

[niosv-shell] C:\altera_workshops\axc3000\NIOSV_lab> niosv-app -b=software/bsp -a=software/app -s=software/app/HelloWorl
d.c
2025.09.04.17:01:27 Info: "software\app\CMakeLists.txt" was generated.

[niosv-shell] C:\altera_workshops\axc3000\NIOSV_lab> |
```

Figure 5-9 : App Generation with Nios V Command Shell

5.3.2.3 Use Ashling RiscFree IDE

Opening the **Ashling RiscFree IDE** can be done by typing **riscfree -data software** in the Nios V Command Shell. This points the RiscFree Workspace to the **software** folder.


software - Ashling RiscFree™ IDE
File Edit Source Refactor Navigate Search Project Run Window Linux Help
New Connection...
Project Explorer
O...
B...
Problems
Tasks
Console
Properties
Create a new Makefile project in a directory containing existing code
Create a new C or C++ project
Create a project...
Import projects...
Import Ashling sample projects..
Import Nios V CMake project ...
Description
Resource
Path
Location

There are no projects in your workspace.
To add a project:

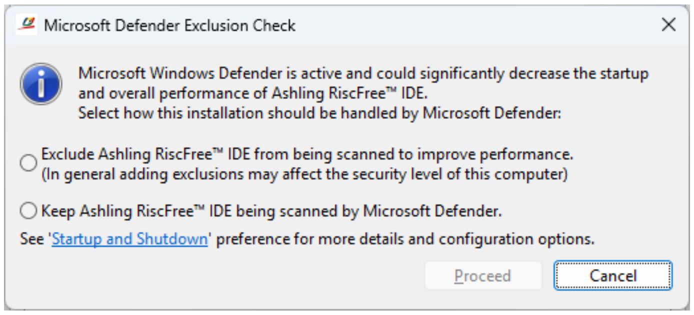
There is no active editor that provides an outline.

0 items

Figure 5-10 : Open Ashling RiscFree IDE

Click on **Create a project...** in the thumbnails.

Click **Cancel** if you see this pop up.



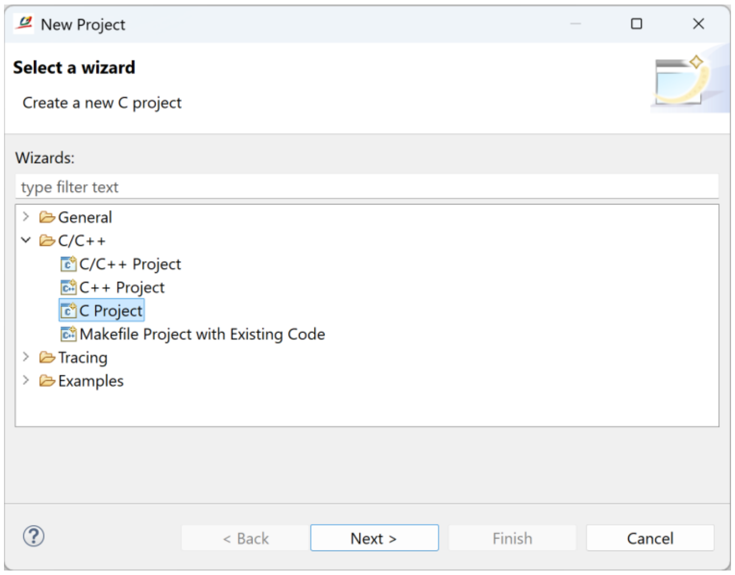
Microsoft Windows Defender is active and could significantly decrease the startup and overall performance of Ashling RiscFree™ IDE.
Select how this installation should be handled by Microsoft Defender:

○ Exclude Ashling RiscFree™ IDE from being scanned to improve performance.
(In general adding exclusions may affect the security level of this computer)

○ Keep Ashling RiscFree™ IDE being scanned by Microsoft Defender.

See 'Startup and Shutdown' preference for more details and configuration options.

Expand the **C/C++** wizard and select the **C Project** item, then click **Next.**


New Project

Select a wizard

Create a new C project

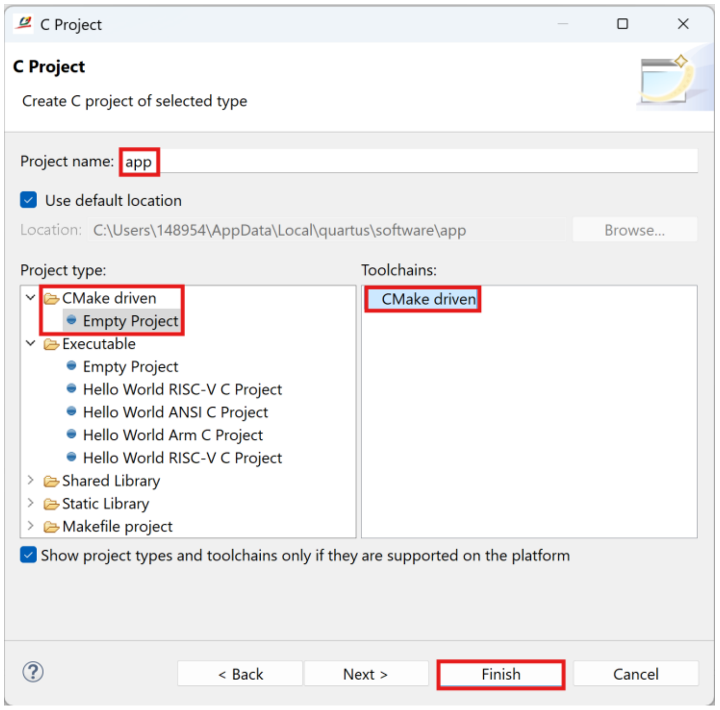
Wizards:

type filter text

- General
- C/C++
  - C/C++ Project
  - C++ Project
  - C Project
  - Makefile Project with Existing Code
- Tracing
- Examples

Figure 5-11 : Select C Project type

Expand the **CMake driven** Project type and select **Empty Project**. In the Toolchains pane, select **CMake driven**.


C Project

C Project

Create C project of selected type

Project name: app

☑ Use default location

Location: C:\Users\148954\AppData\Local\quartus\software\app

Project type:

- CMake driven
  - Empty Project
- Executable
  - Empty Project
  - Hello World RISC-V C Project
  - Hello World ANSI C Project
  - Hello World Arm C Project
  - Hello World RISC-V C Project
- Shared Library
- Static Library
- Makefile project

Toolchains:

CMake driven

☑ Show project types and toolchains only if they are supported on the platform

Figure 5-32 : Create CMake driven demo_app project

Type **app** for the Project name. This matches the pre-existing folder containing the application project.

Click **Finish**.

In the **Ashling RiscFree IDE**, you should see the **app** folder.

Right-click the ***app*** project and select **Build Project**.

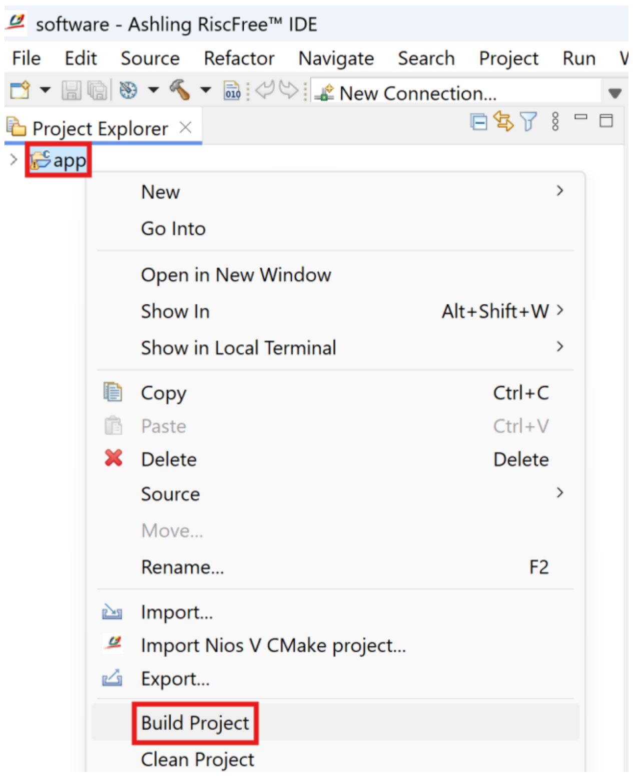
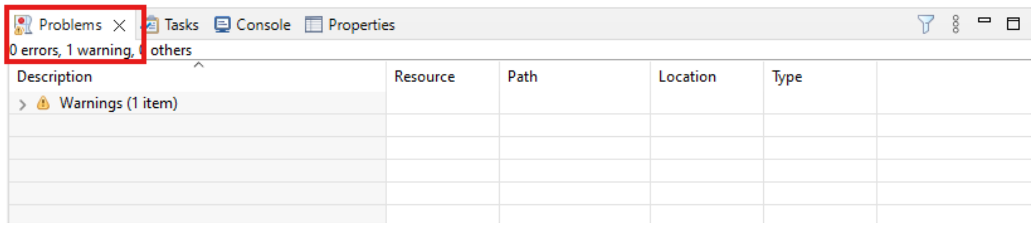


Figure 5-13 : Build Software Project

When the ***Build Project*** progress meter in the lower right corner reached 100%, always check the ***Problems*** messages window for error message.



0 errors, 1 warning, 0 others

Description	Resource	Path	Location	Type
Warnings (1 item)				

Figure 5-14 : Compiler Problems Window

5.3.3 Convert ELF code to HEX

The NIOS-V C code has been successfully compiled, but still needs to be converted into a .hex format for placement in the FPGA onchip RAM during Quartus compilation. Execute the command below in the Nios V Command Shell.

```
elf2hex software/app/build/Debug/app.elf -b 0x00000000 -w 32 -e 0x00027E98 --
record=4 -o onchip_mem.hex
```

5.4 Compile the Design in Quartus

The hardware project needs to be compiled with no errors in order to get a **.sof** configuration file. There are multiple steps performed in Quartus. These steps are the **Fitter** and **Assembler** steps. Ensure that the box to the left of the **Assembler** step is checked. This is the step that generates the **.sof** configuration file.

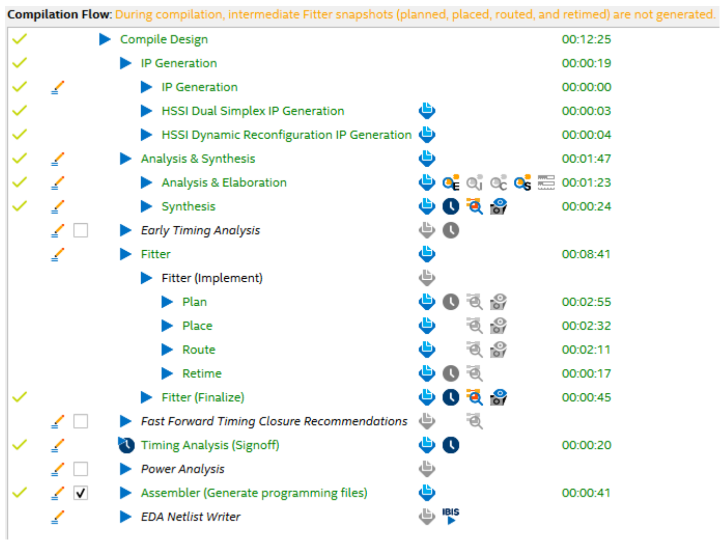
You can either re-start Compilation from the beginning, by clicking on ▶ button on the toolbars, or ***Processing* → *Start Compilation***, or resume to the **Fitter** and **Assembler** by clicking the blue arrow to the left of **Assembler**.

There should be no errors during compilation. If there are errors, they should be fixed before re-compilating. A green checkmark next to the Compile steps in the **Compilation Flow** window indicates that the compilation was successful.

As Quartus compiles the design, there will be a progress bar that shows the steps that it does and the checkmarks as the steps are completed. Notice the checkmarks as steps are completed. The time used for each step depends on the computer used such as the number of processors, amount of memory, and processor clock speed.

If you did a re-compile from the beginning, there will be a **Timing Analysis** Window that appears at the end, but the focus is on the compilation.

The on-chip memory is initialized with the software project compiled code in a HEX format. This will be included in the FPGA bitstream when the device is programmed and will be run when the Nios V processor is released from reset.



```
Compilation Flow: During compilation, intermediate Fitter snapshots (planned, placed, routed, and retimed) are not generated.
Compile Design                                          00:12:25
  IP Generation                                         00:00:19
    IP Generation                                       00:00:00
    HSSI Dual Simplex IP Generation                     00:00:03
    HSSI Dynamic Reconfiguration IP Generation          00:00:04
  Analysis & Synthesis                                  00:01:47
    Analysis & Elaboration                              00:01:23
    Synthesis                                           00:00:24
  Early Timing Analysis
  Fitter                                                00:08:41
    Fitter (Implement)
      Plan                                              00:02:55
      Place                                             00:02:32
      Route                                             00:02:11
      Retime                                            00:00:17
    Fitter (Finalize)                                   00:00:45
  Fast Forward Timing Closure Recommendations
  Timing Analysis (Signoff)                             00:00:20
  Power Analysis
  Assembler (Generate programming files)                00:00:41
  EDA Netlist Writer
```

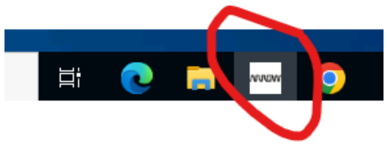
Figure 5-15 : Completed Compile flow

6 Validate the design Locally

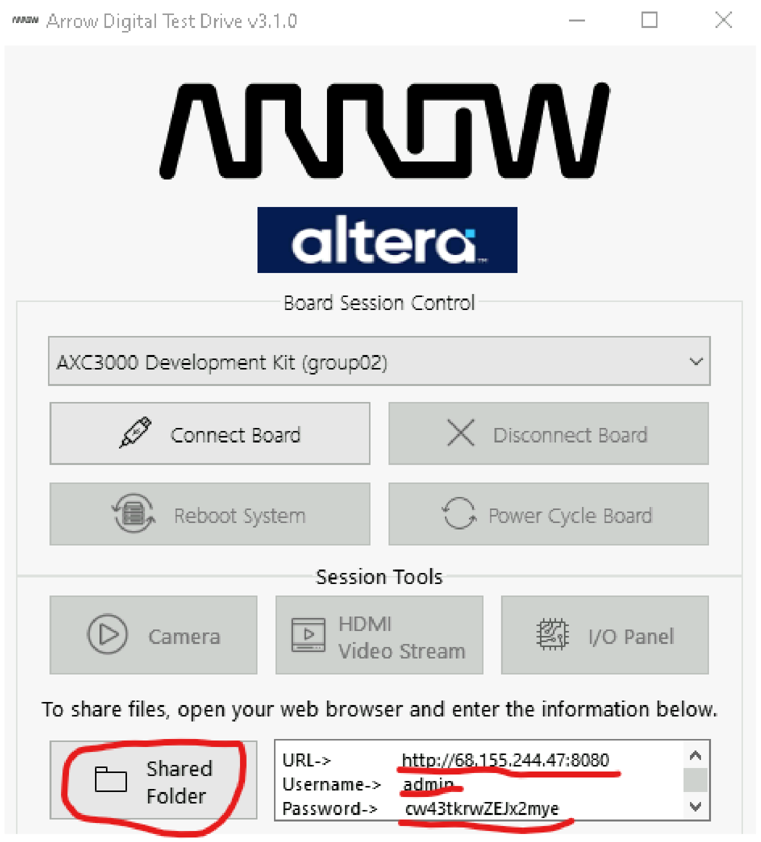
The next step is to see if the design works. If you have built the lab using CloudLabs then complete section 6.1. If you have built the lab on a local PC then skip to section 6.2.

6.1 File Download

6.1.1 Click on the ArrowDTD icon in CloudLabs to launch the tool.



6.1.2 Open the **Shared Folder** utility in the CloudLab VM. Open a browser in your local PC. Navigate to the **URL** indicated. Use the provided username and password.

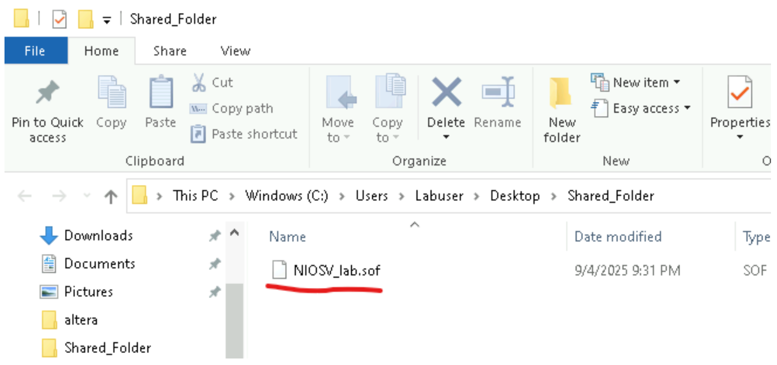


To share files, open your web browser and enter the information below.

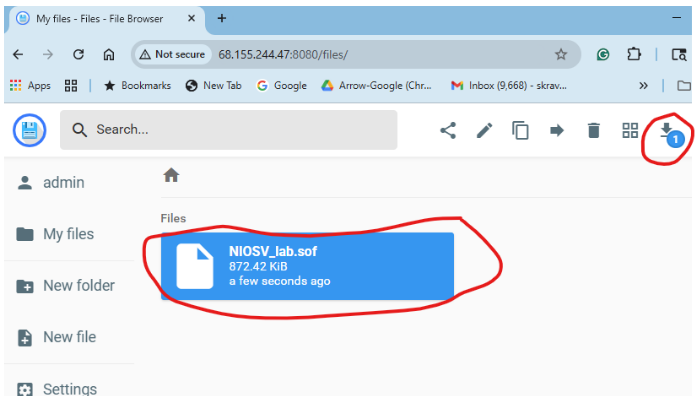
URL-> http://68.155.244.47:8080
Username-> admin
Password-> cw43tkrwZEJx2mye

6.1.3 **Drag** and drop the following file into the **Shared** folder.

c:\altera_workshops\axc3000\My_First_Lab\output_files\NIOSV_lab.sof



6.1.4 This will be reflected in the **browser** on the local PC. **Download** the SOF file to a convenient location.



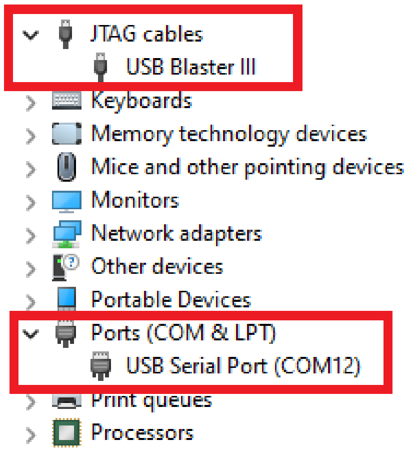
The rest of the instructions will be done on the **local computer**.

6.2 Program the FPGA

The next step is to program the design into the FPGA on the board. During the Assembler stage of the compilation, a programming file was generated.

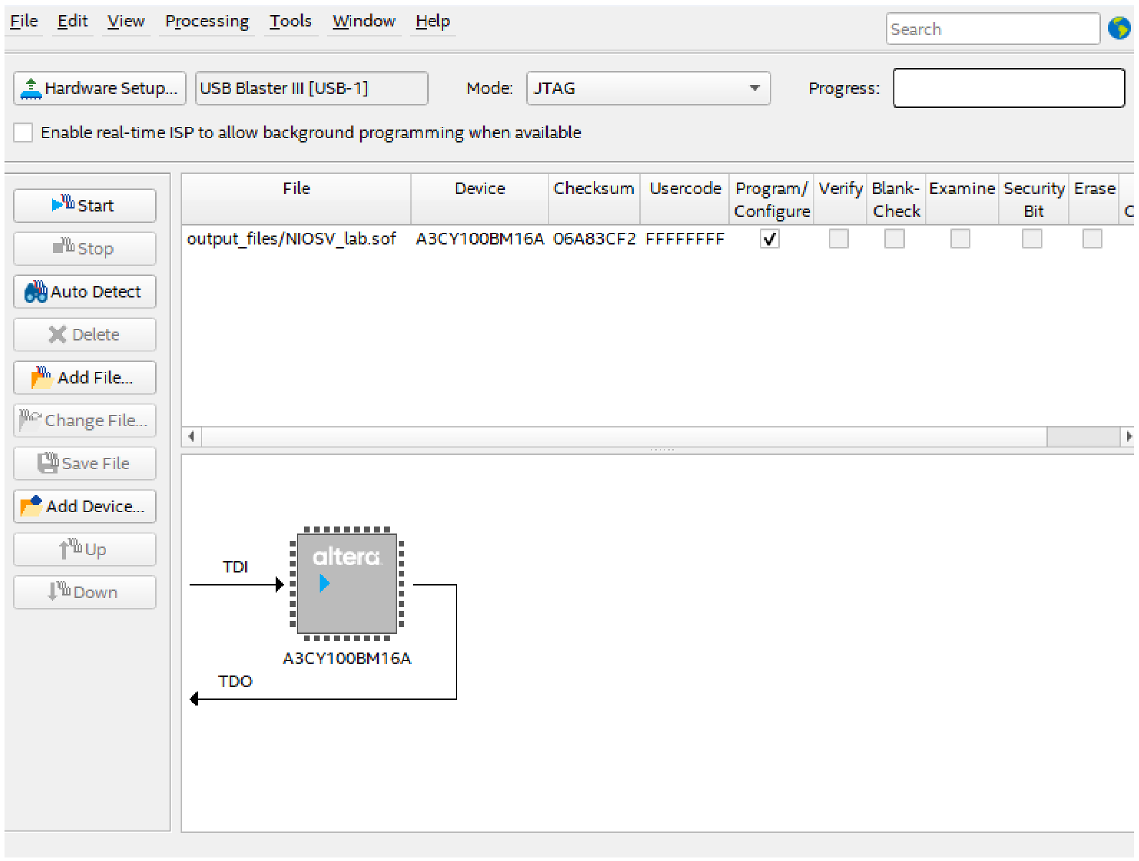
6.2.1 Connect the AXC3000 to your PC using a USB C cable.

Since the Altera USB Blaster III should be already installed, the Window's Device Manager should display the following entries, highlighted in red (COM port number may differ depending on your PC). Use the COM port later when setting up the terminal emulator.



JTAG cables
USB Blaster III
Keyboards
Memory technology devices
Mice and other pointing devices
Monitors
Network adapters
Other devices
Portable Devices
Ports (COM & LPT)
USB Serial Port (COM12)
Print queues
Processors

6.2.2 From the Start menu open Altera 25.1.1.125 Pro Edition --> Programmer.



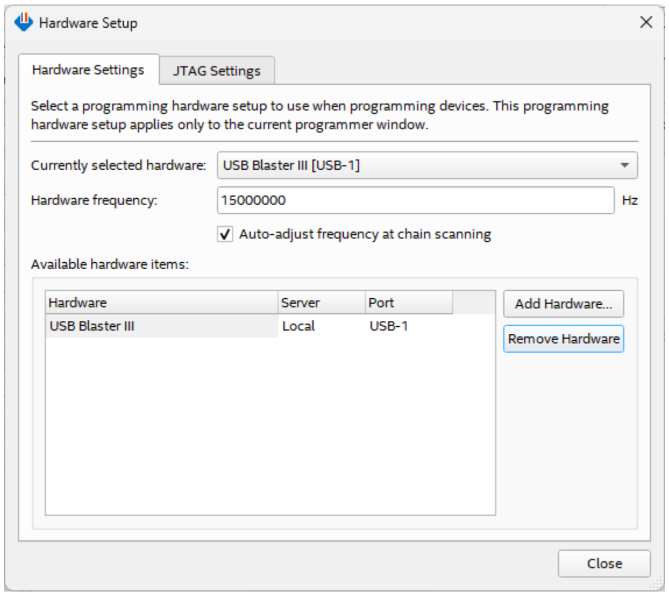
Hardware Setup... USB Blaster III [USB-1] Mode: JTAG Progress:

Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase
output_files/NIOSV_lab.sof	A3CY100BM16A	06A83CF2	FFFFFFFF	✓					

Figure 6-1 : Quartus Programmer

6.2.3 Click the **Hardware Setup** button and double-click on the **USB Blaster III**, then click **Close**.



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

Currently selected hardware: USB Blaster III [USB-1]

Hardware frequency: 15000000 Hz

✓ Auto-adjust frequency at chain scanning

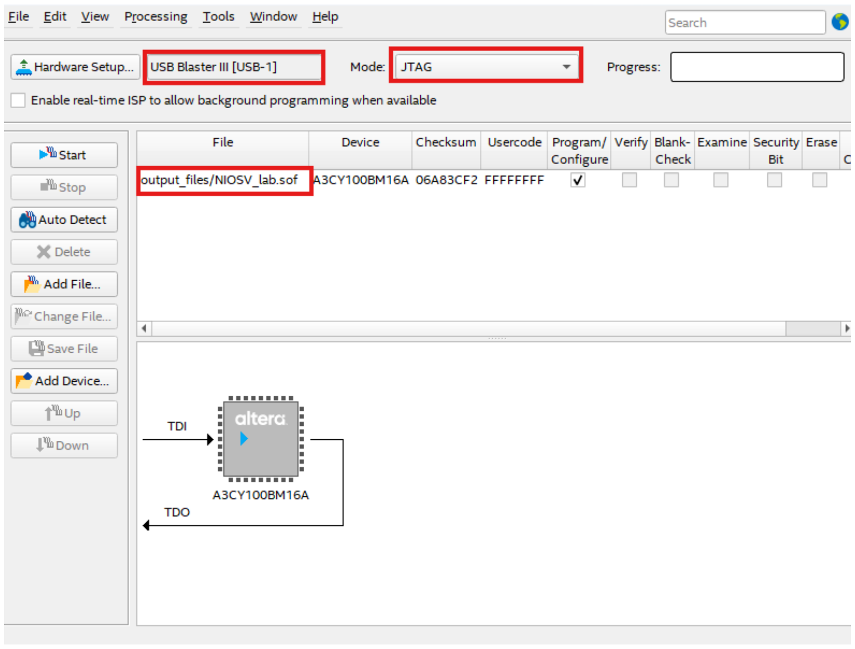
Available hardware items:

Hardware	Server	Port
USB Blaster III	Local	USB-1

Figure 6-2 : USB Blaster III Hardware Setup

6.2.4 Ensure that the **USB Blaster III** is selected, and the mode is set to **JTAG**.

6.2.5 If the .sof file is not already selected, click on the **Change File** button and navigate to the ***output_files/NIOSV_lab.sof***. If the file was downloaded from Cloudlabs then navigate to the **folder where the sof file was downloaded** and select the **.sof** file.



Hardware Setup... USB Blaster III [USB-1] Mode: JTAG Progress:

Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase
output_files/NIOSV_lab.sof	A3CY100BM16A	06A83CF2	FFFFFFFF	✓					

Figure 6-3 : USB Blaster III and .sof Selected

6.2.6 Select the .sof file then click the **Start** button to begin the programming operation and wait for the progress bar to hit 100%.

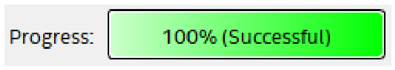
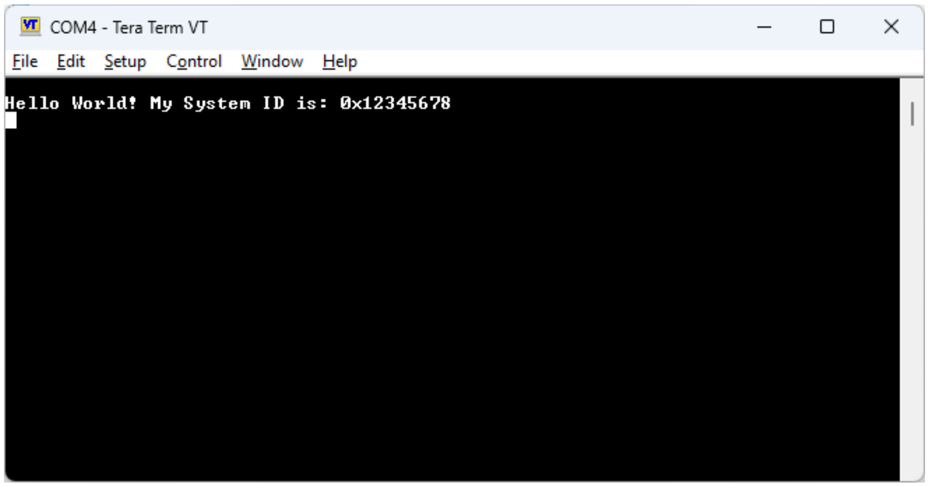


Figure 6-4 : Programmer progress bar

6.3 Open a Terminal Emulator

Open a terminal emulator (Tera term or Putty). Select the COM port shown earlier. Set the emulator to have 8 databits, No stop bits and one Parity bit. Set the baud rate to 115,200.

Observe the "**Hello World**" message printed.



```
Hello World! My System ID is: 0x12345678
```

Figure 6-5 : Terminal Emulator

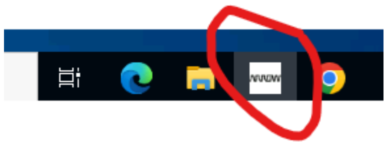
CONGRATULATIONS! YOU HAVE SUCCESSFULLY COMPLETED THE Nios V DESIGN LAB!

7 Validate the Design Remotely

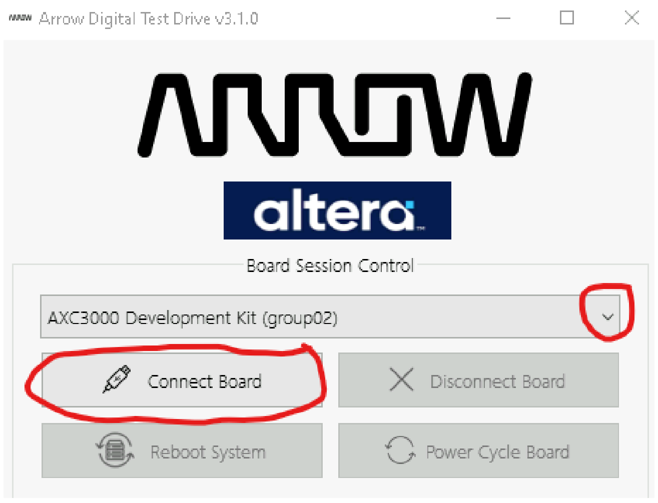
The following instructions require the user to be utilizing a CloudLabs Windows Virtual Machine.

7.1 Connect to the Board Farm

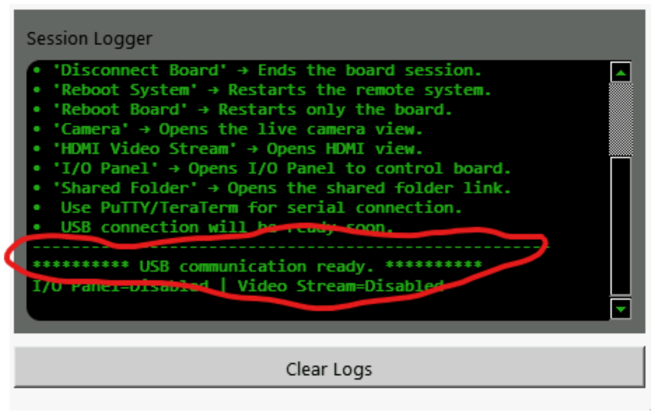
7.1.1 Connect to the remote AXC3000 Board using the ArrowDTD tool. Click on the ArrowDTD icon to launch the tool.



7.1.2 Select an available board from the dropdown menu and click connect board.

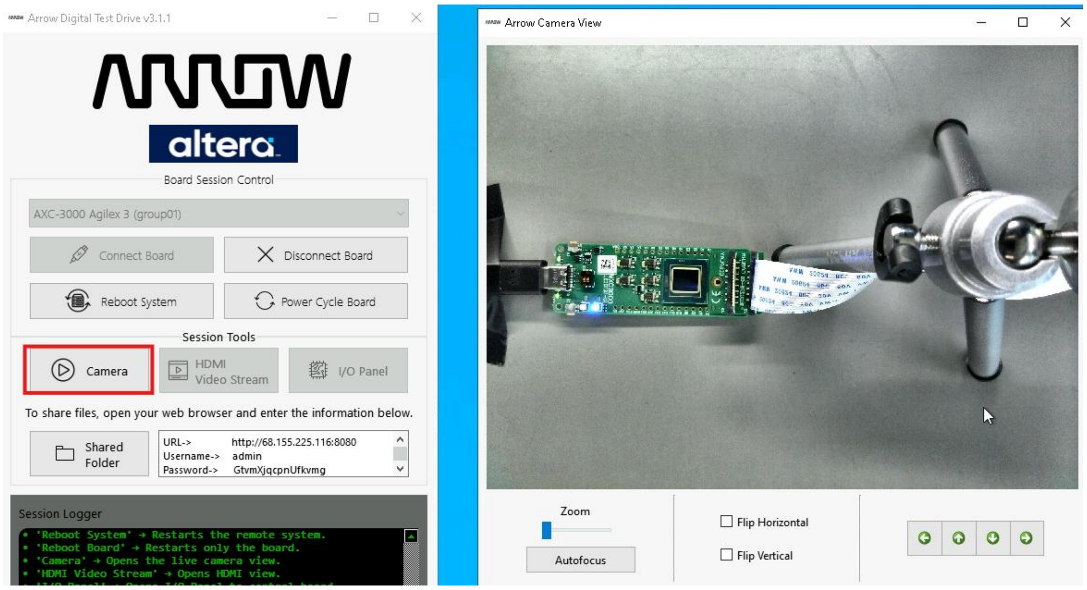


Wait until the board is ready.



```
• 'Disconnect Board' → Ends the board session.
• 'Reboot System' → Restarts the remote system.
• 'Reboot Board' → Restarts only the board.
• 'Camera' → Opens the live camera view.
• 'HDMI Video Stream' → Opens HDMI view.
• 'I/O Panel' → Opens I/O Panel to control board.
• 'Shared Folder' → Opens the shared folder link.
•  Use PuTTY/TeraTerm for serial connection.
•  USB connection will be ready soon.
------------------------------------------------------------
********** USB communication ready. **********
I/O Panel=Disabled | Video Stream=Disabled
```

7.1.3 Open the Camera View of the board.



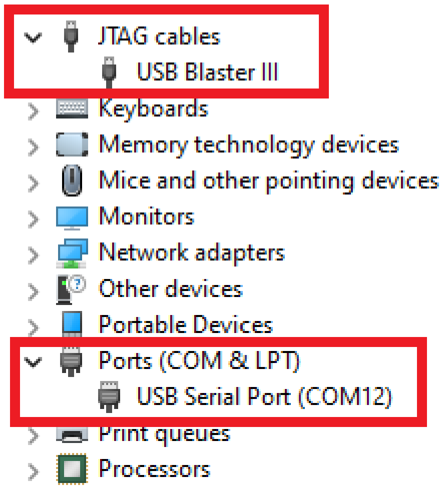
AXC-3000 Agilex 3 (group01)

To share files, open your web browser and enter the information below.

```
URL->        http://68.155.225.116:8080
Username->   admin
Password->   GtvmXjqcpnUfkvmg
```

```
• 'Reboot System' → Restarts the remote system.
• 'Reboot Board' → Restarts only the board.
• 'Camera' → Opens the live camera view.
• 'HDMI Video Stream' → Opens HDMI view.
```

7.1.4 Window's Device Manager should display the following entries, highlighted in red (COM port number may differ depending on your PC). Use the COM port later when setting up the terminal emulator.

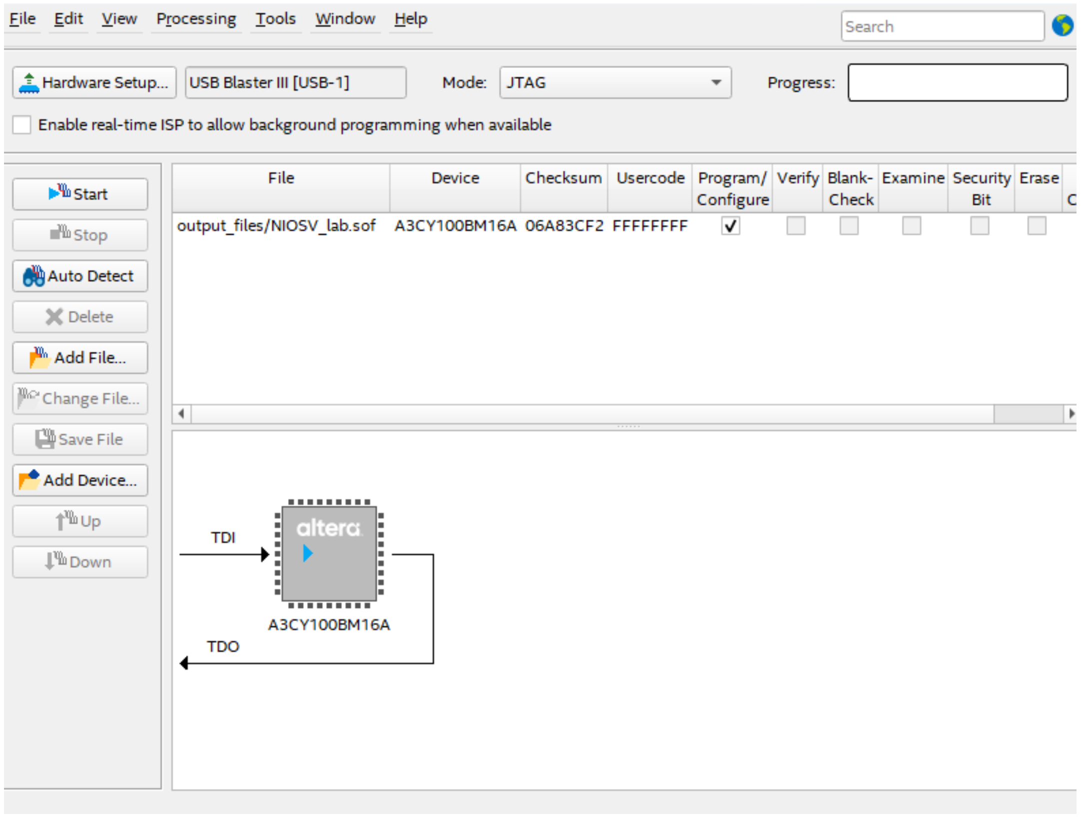


- JTAG cables
  - USB Blaster III
- Keyboards
- Memory technology devices
- Mice and other pointing devices
- Monitors
- Network adapters
- Other devices
- Portable Devices
- Ports (COM & LPT)
  - USB Serial Port (COM12)
- Print queues
- Processors

7.2 Program the FPGA

7.2.1 Open the Quartus Prime Programmer

On the Quartus Prime Pro, click **Tools → Programmer**, or click on the Programmer icon in the tool bar. The **Hardware Setup** field might show ***No Hardware***.



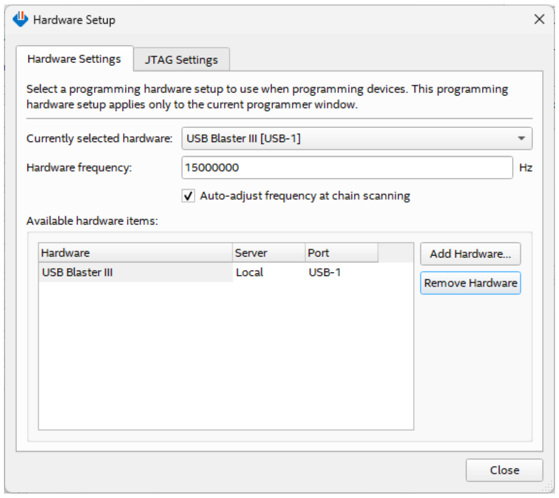
Hardware Setup... USB Blaster III [USB-1] Mode: JTAG Progress:

Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase
output_files/NIOSV_lab.sof	A3CY100BM16A	06A83CF2	FFFFFFFF	✓					

Figure 7-1 : Quartus Programmer

7.2.2 Click the **Hardware Setup** button and double-click on the **USB Blaster III**, then click **Close**.



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

Currently selected hardware: USB Blaster III [USB-1]

Hardware frequency: 15000000 Hz

✓ Auto-adjust frequency at chain scanning

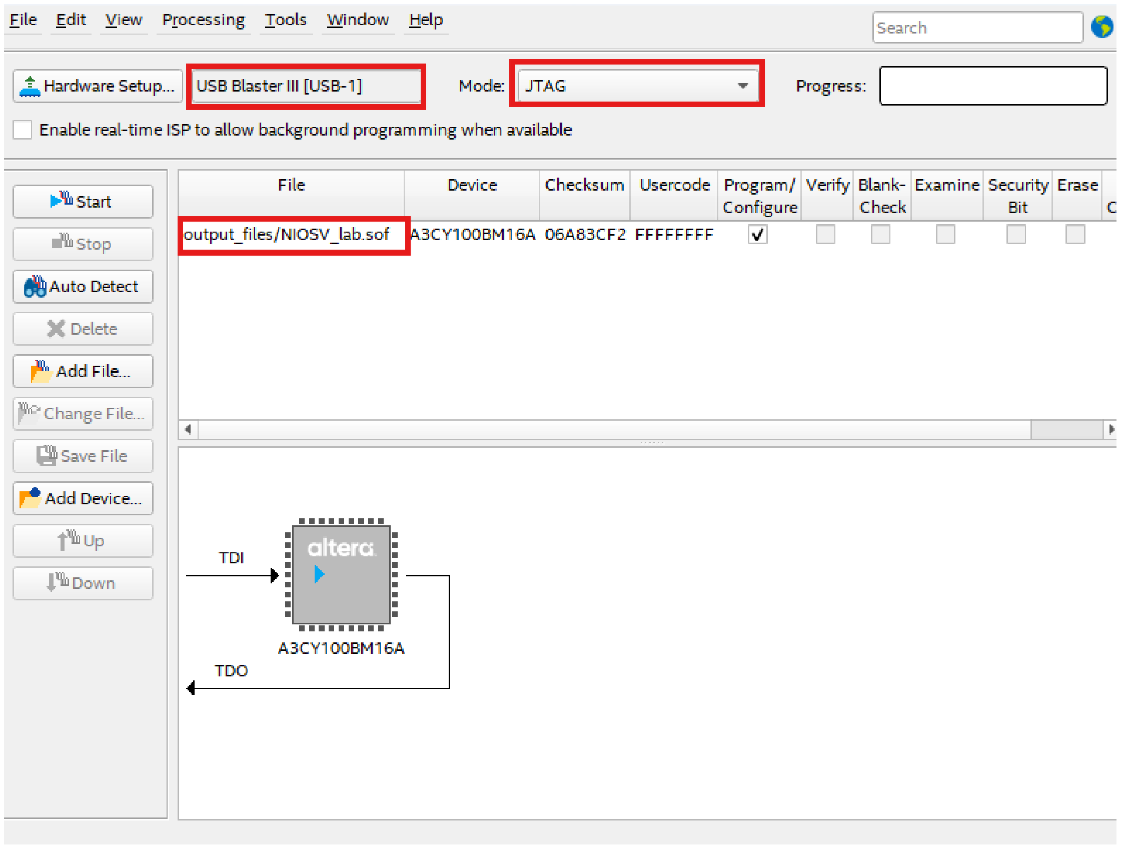
Available hardware items:

Hardware	Server	Port
USB Blaster III	Local	USB-1

Figure 7-2 : USB Blaster III Hardware Setup

7.2.3 Ensure that the **USB Blaster III** is selected, and the mode is set to **JTAG**.

If the .sof file is not already selected, click on the **Change File** button and navigate to the ***output_files/NIOSV_lab.sof***.



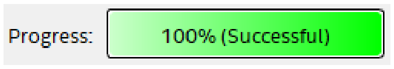
USB Blaster III [USB-1] Mode: JTAG Progress:

Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase
output_files/NIOSV_lab.sof	A3CY100BM16A	06A83CF2	FFFFFFFF	✓					

Figure 7-3 : USB Blaster III and .sof Selected

Select the .sof file then click the **Start** button to begin the programming operation and wait for the progress bar to hit 100%.



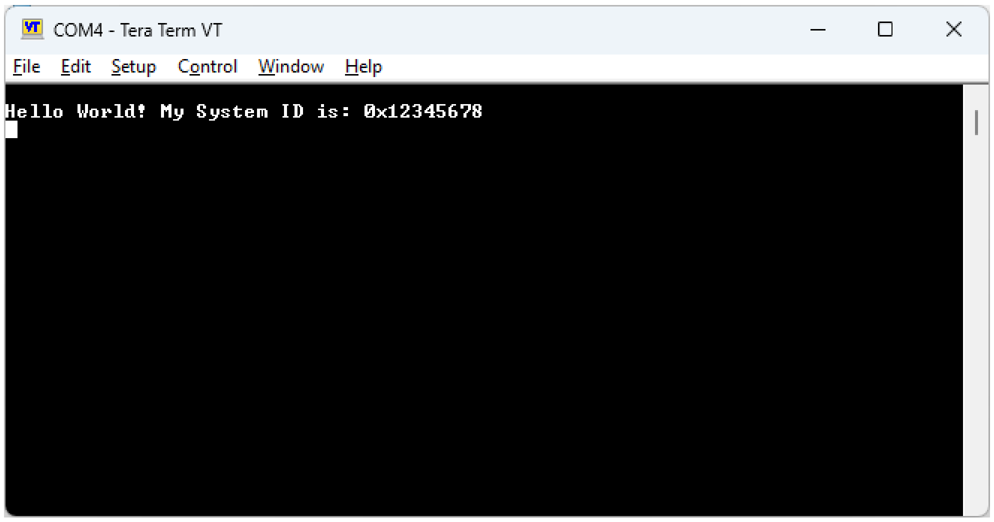
Progress: 100% (Successful)

Figure 7-4 : Programmer progress bar

7.3 Open a Terminal Emulator

Open a terminal emulator (Tera term or Putty). Select the COM port shown earlier. Set the emulator to have 8 databits, No stop bits and one Parity bit. Set the baud rate to 115,200.

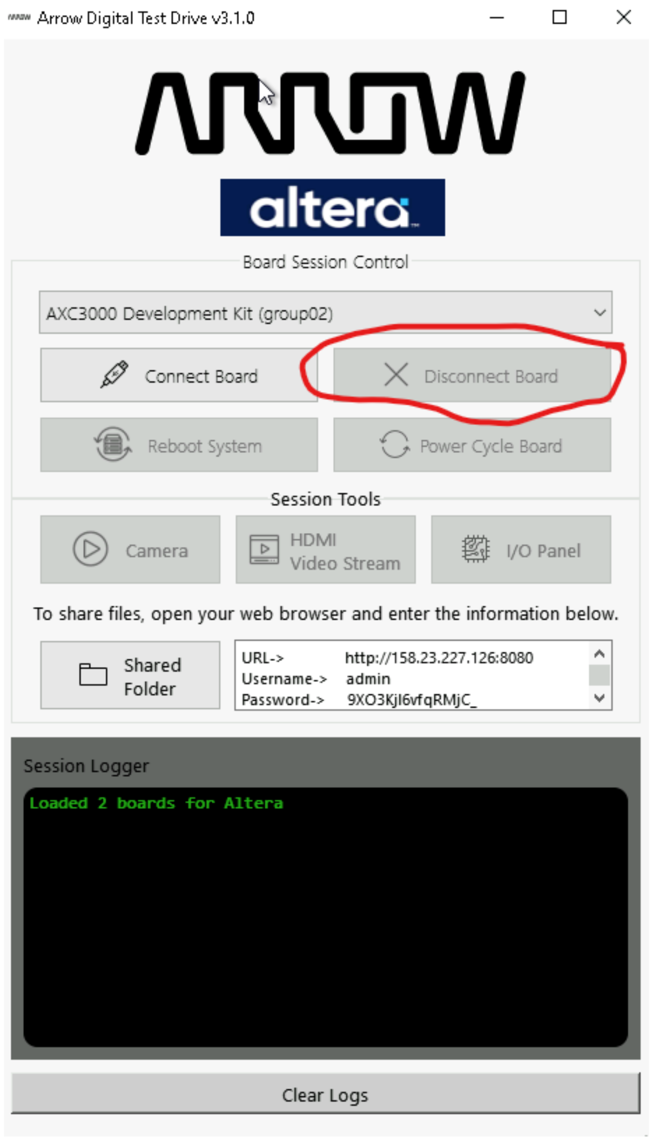
Observe the “**Hello World**” message printed.



```
Hello World! My System ID is: 0x12345678
```

Figure 7-5 : Terminal Emulator

7.3.1.1 Please disconnect the board when debugging is complete. This will free up the board resource for others to use.



To share files, open your web browser and enter the information below.

```
URL->          http://158.23.227.126:8080
Username->     admin
Password->     9XO3KjI6vfqRMjC_
```

```
Loaded 2 boards for Altera
```

CONGRATULATIONS! YOU HAVE SUCCESSFULLY COMPLETED THE Nios V DESIGN LAB!

8 Legal Disclaimer

ARROW ELECTRONICS

EVALUATION BOARD LICENSE AGREEMENT

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

PURPOSE

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

LICENSE

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

EVALUATION BOARD STATUS

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

OWNERSHIP AND COPYRIGHT

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

RESTRICTIONS AND WARNINGS

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

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

WARRANTY

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

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

LIMITATION OF LIABILITIES

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

IDENTIFICATION

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

RECYCLING

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