ADIUVO
ENGINEERING AND TRAINING, LTD.

Getting started with Nios V and Ashling RiscFree IDE for Intel FPGAs

Adam Taylor

Agenda

- Introduction
- Hardware Design
- Software Design
- Debugging & Future Steps

ADIUVO
ENGINEERING AND TRAINING, LTD.

ADIUVO
ENGINEERING AND TRAINING, LTD.

Why SoftCore

FPGA are great for parallel processing but often need some help in the form of sequential processing for serial communications, IP configuration, Networking Etc.

- Nios 2 was one of the most popular softcore processors for this in the Intel Ecosystem

Nios V is the next generation Nios processor based on the RISC V Instruction Set Architecutre

RISC V

RISC V is NOT a processor – Rather an instruction set architecture (ISA) that processors are compliant with.

There are three main ratified ISA: the RV32I, RV64I and RV128I.

These define 32-, 64-, and 128-bit ISAs respectively.

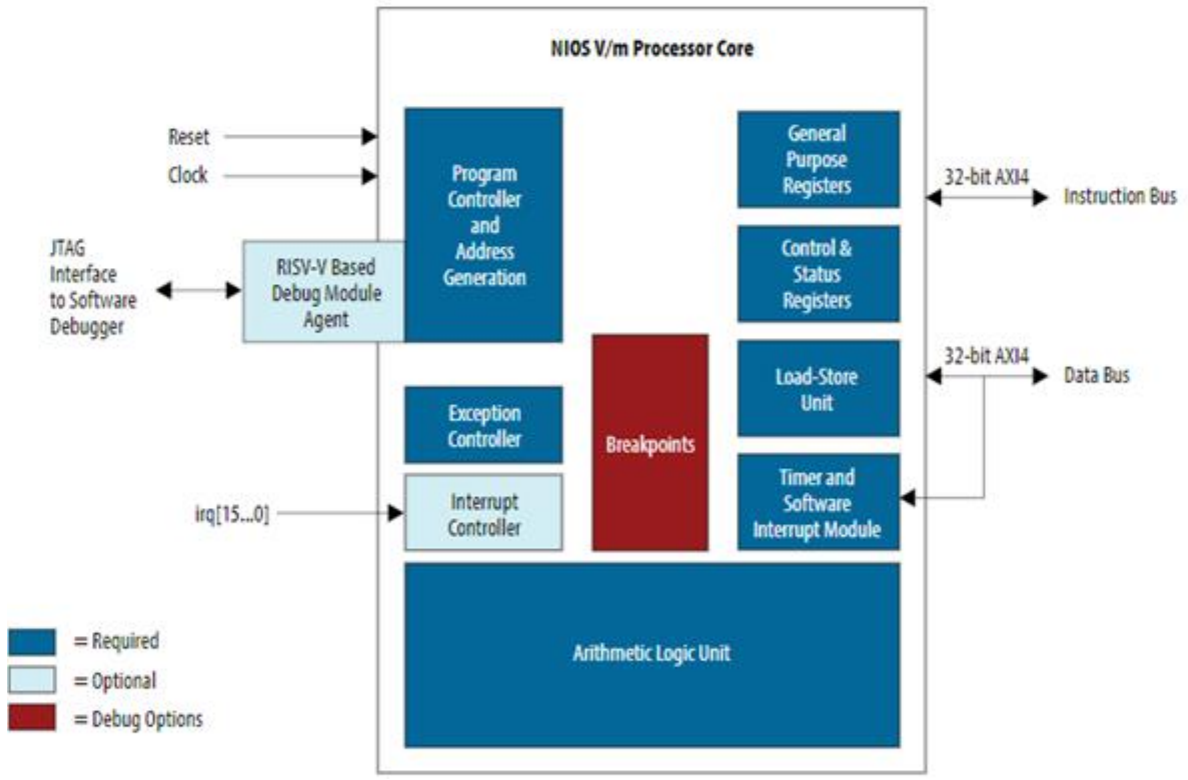
Range of Extensions

- Atomic instructions (A)
- Single and double precision floating point extensions (F and D)
- Integer multiplication and division (M)

Nios V/m

Nios V/m initial release from Intel, more advanced cores are coming including Application Class and Linux Capable.

Nios V/m is based on the 32-bit instruction set with support for atomic instructions.

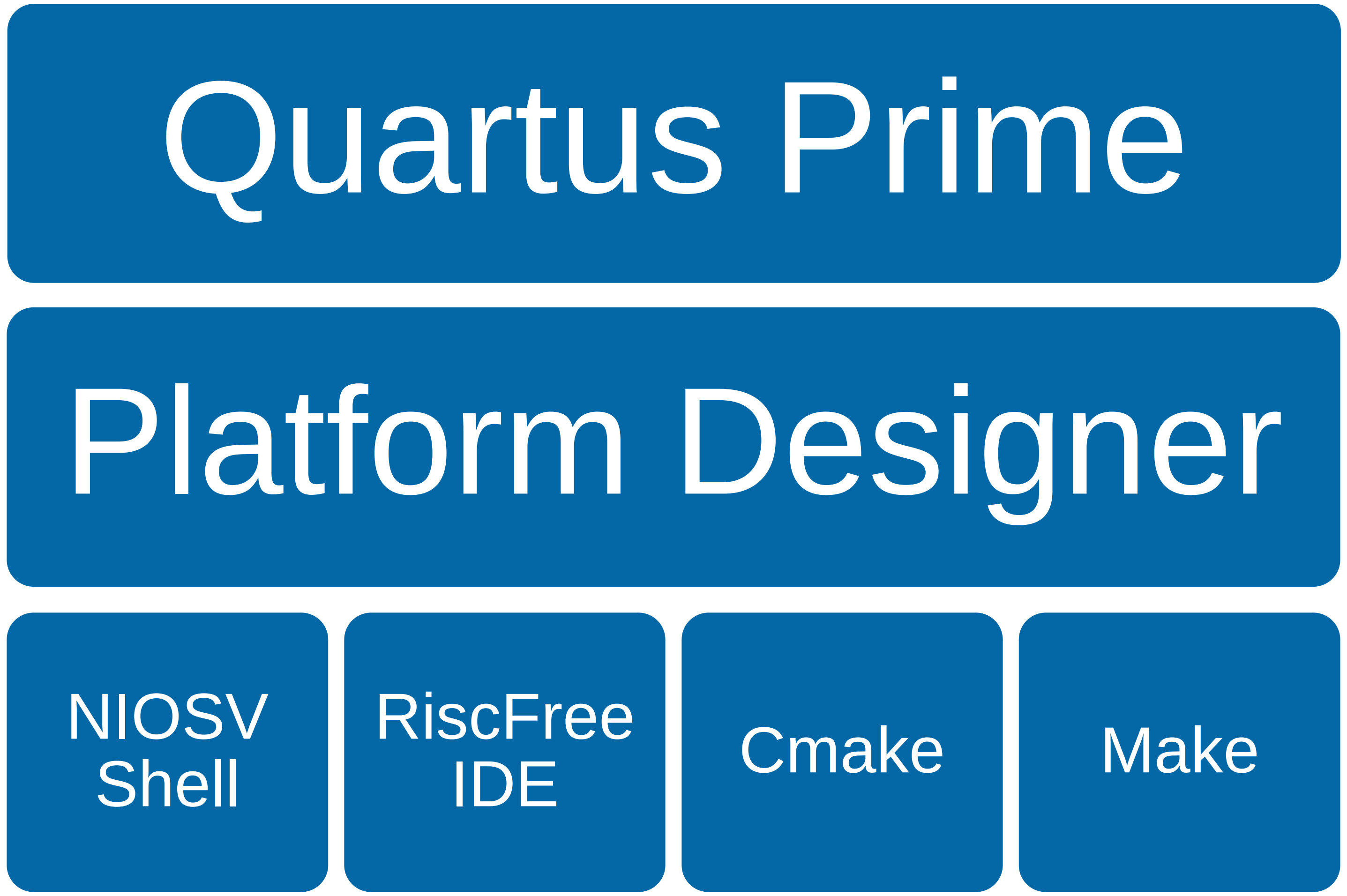


Nios V/m

Key Architectural features

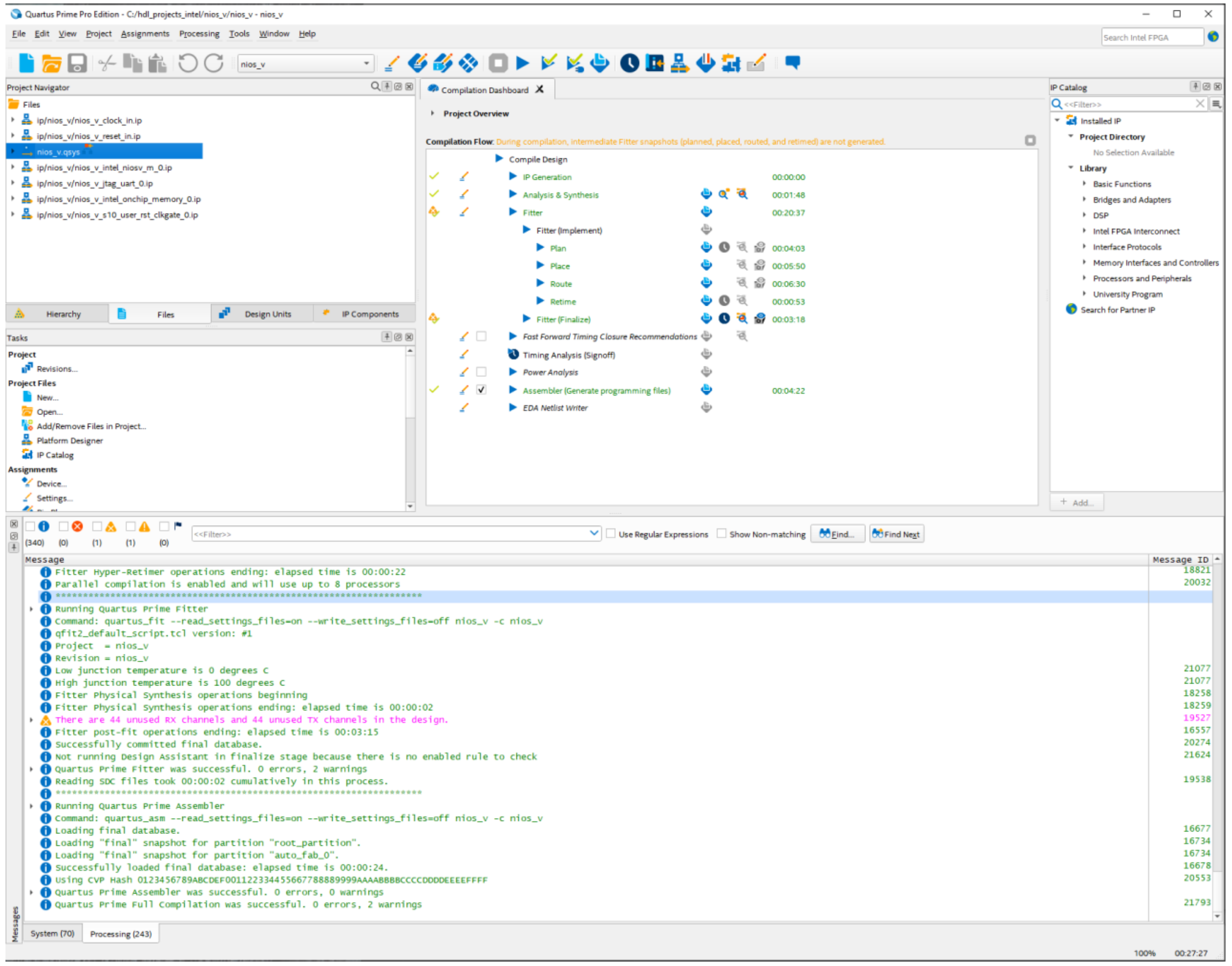
- AXI4 Busses used for Instruction and Data Memories
- Provide the developer with 0.464 DMIPs/MHZ
  - Nios II/e at 0.107 DMIPs/MHz.
- Currently Supports 10 Series devices and Agilex
- Nios V/m ranges between 1300 and 1600 ALMs
- Nios V/m provides performance between 270 MHz in a Cyclone 10 to 566 MHz in an Agilex device

Design Flow


Quartus Prime
Platform Designer
NIOSV Shell
RiscFree IDE
Cmake
Make

Quartus Prime

- Used for Implementation of the programming file.
- Pin allocation – IO standards
- SDM and security
- Timing constraints
- Timing verification
- Simulation Models



Quartus Prime Pro Edition - C:/hdl_projects_intel/nios_v/nios_v - nios_v

Files
- ip/nios_v/nios_v_clock_in.ip
- ip/nios_v/nios_v_reset_in.ip
- nios_v.qsys
- ip/nios_v/nios_v_intel_niosv_m_0.ip
- ip/nios_v/nios_v_jtag_uart_0.ip
- ip/nios_v/nios_v_intel_onchip_memory_0.ip
- ip/nios_v/nios_v_s10_user_rst_clkgate_0.ip

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

Compile Design	
IP Generation	00:00:00
Analysis & Synthesis	00:01:48
Fitter	00:20:37
Fitter (Implement)	
Plan	00:04:03
Place	00:05:50
Route	00:06:30
Retime	00:00:53
Fitter (Finalize)	00:03:18
Fast Forward Timing Closure Recommendations	
Timing Analysis (Signoff)	
Power Analysis	
Assembler (Generate programming files)	00:04:22
EDA Netlist Writer	

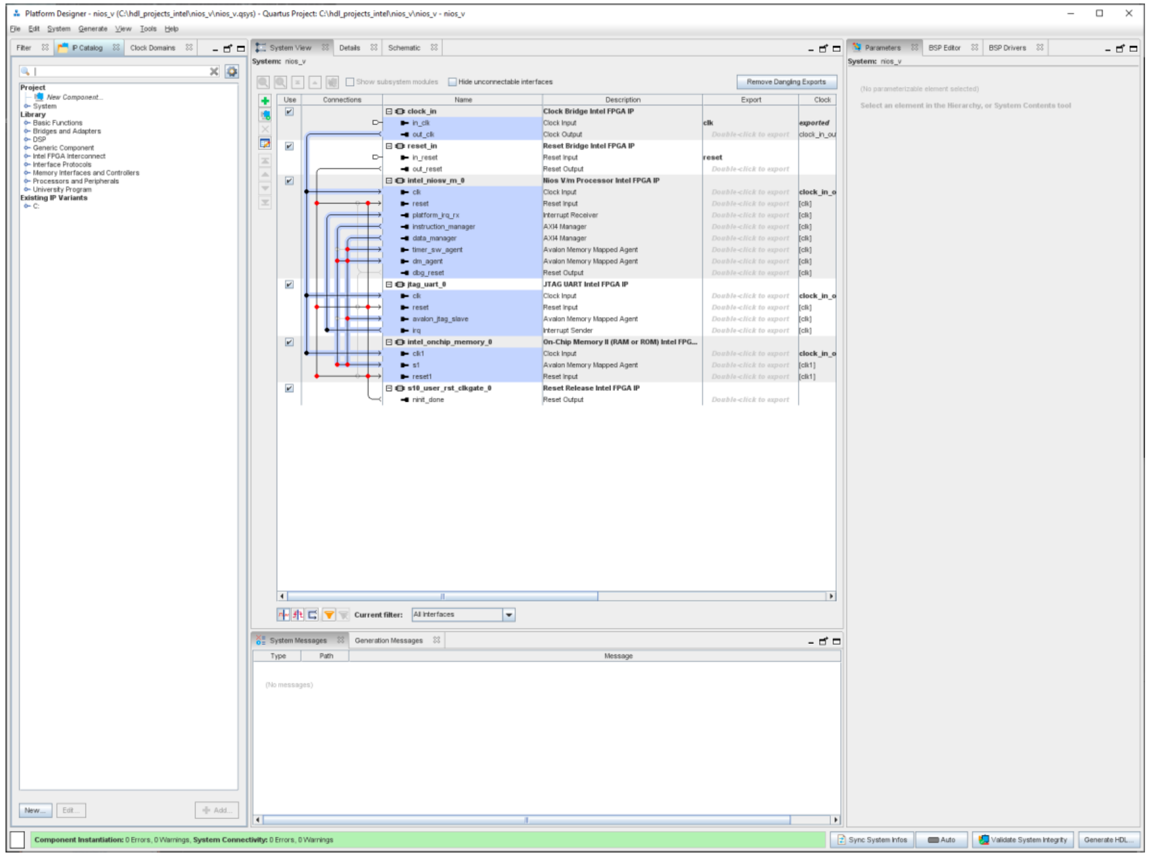
```
Message                                                                                   Message ID
Fitter Hyper-Retimer operations ending: elapsed time is 00:00:22                               18821
Parallel compilation is enabled and will use up to 8 processors                               20032
***************************************************************
Running Quartus Prime Fitter
Command: quartus_fit --read_settings_files=on --write_settings_files=off nios_v -c nios_v
qfit2_default_script.tcl version: #1
Project  = nios_v
Revision = nios_v
Low junction temperature is 0 degrees C                                                       21077
High junction temperature is 100 degrees C                                                    21077
Fitter Physical Synthesis operations beginning                                                18258
Fitter Physical Synthesis operations ending: elapsed time is 00:00:02                         18259
There are 44 unused RX channels and 44 unused TX channels in the design.                      19527
Fitter post-fit operations ending: elapsed time is 00:03:15                                   16557
Successfully committed final database.                                                        20274
Not running Design Assistant in finalize stage because there is no enabled rule to check      21624
Quartus Prime Fitter was successful. 0 errors, 2 warnings
Reading SDC files took 00:00:02 cumulatively in this process.                                 19538
***************************************************************
Running Quartus Prime Assembler
Command: quartus_asm --read_settings_files=on --write_settings_files=off nios_v -c nios_v
Loading final database.                                                                       16677
Loading "final" snapshot for partition "root_partition".                                      16734
Loading "final" snapshot for partition "auto_fab_0".                                          16734
Successfully loaded final database: elapsed time is 00:00:24.                                 16678
Using CVP Hash 0123456789ABCDEF00112233445566778889999AAAABBBBCCCCDDDDEEEEFFFF                   20553
Quartus Prime Assembler was successful. 0 errors, 0 warnings
Quartus Prime Full Compilation was successful. 0 errors, 2 warnings                           21793
```

System (70)   Processing (243)

100%   00:27:27

Platform Designer

- Create the processor subsystem
- Pick from a range of IP for easy integration in the design
- Add in Custom IP from your own library or designs
- BSP Generation for Nios V
- Make signals available to the higher level of the design / IO as desired.
- Highly configurable system development environment



Platform Designer - nios_v (C:\hdl_projects_intel\nios_v\nios_v.qsys) - Quartus Project: C:\hdl_projects_intel\nios_v\nios_v - nios_v

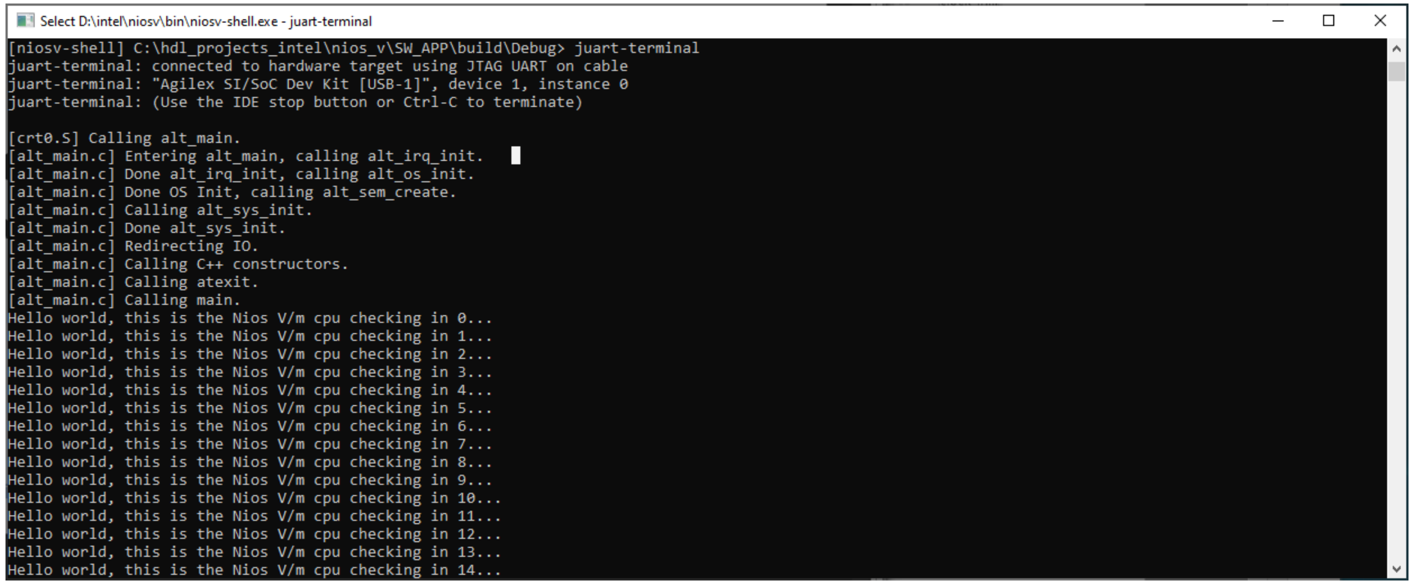
System: nios_v

Use	Name	Description	Export	Clock
✓	clock_in	Clock Bridge Intel FPGA IP		
	in_clk	Clock Input	clk	exported
	out_clk	Clock Output	Double-click to export	clock_in_ou
✓	reset_in	Reset Bridge Intel FPGA IP		
	in_reset	Reset Input	reset	
	out_reset	Reset Output	Double-click to export	
✓	intel_niosv_m_0	Nios V/m Processor Intel FPGA IP		
	clk	Clock Input	Double-click to export	clock_in_o
	reset	Reset Input	Double-click to export	[clk]
	platform_irq_rx	Interrupt Receiver	Double-click to export	[clk]
	instruction_manager	AXI4 Manager	Double-click to export	[clk]
	data_manager	AXI4 Manager	Double-click to export	[clk]
	timer_sw_agent	Avalon Memory Mapped Agent	Double-click to export	[clk]
	dm_agent	Avalon Memory Mapped Agent	Double-click to export	[clk]
	dbg_reset	Reset Output	Double-click to export	[clk]
✓	jtag_uart_0	JTAG UART Intel FPGA IP		
	clk	Clock Input	Double-click to export	clock_in_o
	reset	Reset Input	Double-click to export	[clk]
	avalon_jtag_slave	Avalon Memory Mapped Agent	Double-click to export	[clk]
	irq	Interrupt Sender	Double-click to export	[clk]
✓	intel_onchip_memory_0	On-Chip Memory II (RAM or ROM) Intel FPG...		
	clk1	Clock Input	Double-click to export	clock_in_o
	s1	Avalon Memory Mapped Agent	Double-click to export	[clk1]
	reset1	Reset Input	Double-click to export	[clk1]
✓	s10_user_rst_clkgate_0	Reset Release Intel FPGA IP		
	ninit_done	Reset Output	Double-click to export	

Component Instantiation: 0 Errors, 0 Warnings, System Connectivity: 0 Errors, 0 Warnings

NiosV Shell

- Command line shell
- Used to create / update
  - BSP
  - Applications
  - Libraries
- Download the application to target
- Provide information on the stack and heap

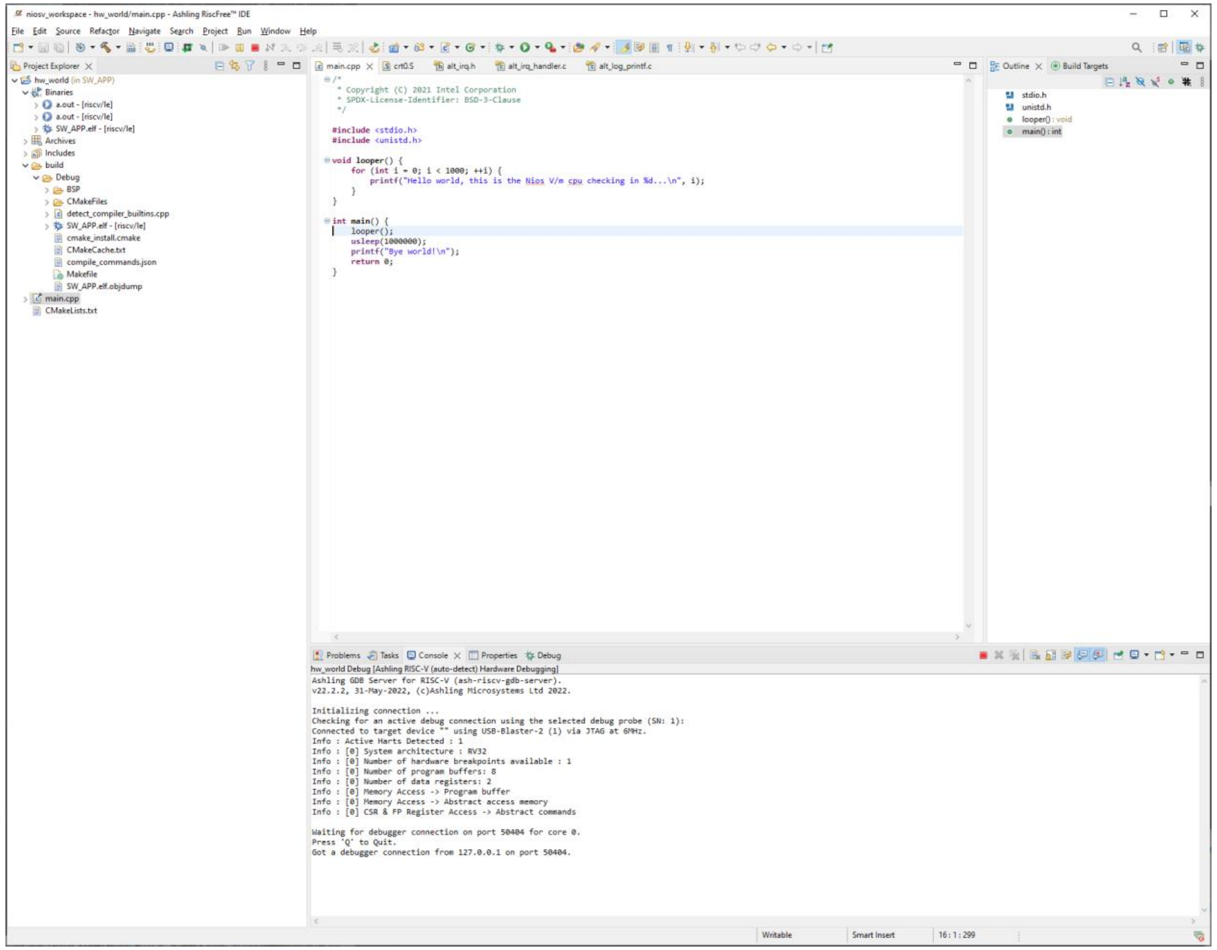


Select D:\intel\niosv\bin\niosv-shell.exe - juart-terminal

```
[niosv-shell] C:\hdl_projects_intel\nios_v\SW_APP\build\Debug> juart-terminal
juart-terminal: connected to hardware target using JTAG UART on cable
juart-terminal: "Agilex SI/SoC Dev Kit [USB-1]", device 1, instance 0
juart-terminal: (Use the IDE stop button or Ctrl-C to terminate)

[crt0.S] Calling alt_main.
[alt_main.c] Entering alt_main, calling alt_irq_init.
[alt_main.c] Done alt_irq_init, calling alt_os_init.
[alt_main.c] Done OS Init, calling alt_sem_create.
[alt_main.c] Calling alt_sys_init.
[alt_main.c] Done alt_sys_init.
[alt_main.c] Redirecting IO.
[alt_main.c] Calling C++ constructors.
[alt_main.c] Calling atexit.
[alt_main.c] Calling main.
Hello world, this is the Nios V/m cpu checking in 0...
Hello world, this is the Nios V/m cpu checking in 1...
Hello world, this is the Nios V/m cpu checking in 2...
Hello world, this is the Nios V/m cpu checking in 3...
Hello world, this is the Nios V/m cpu checking in 4...
Hello world, this is the Nios V/m cpu checking in 5...
Hello world, this is the Nios V/m cpu checking in 6...
Hello world, this is the Nios V/m cpu checking in 7...
Hello world, this is the Nios V/m cpu checking in 8...
Hello world, this is the Nios V/m cpu checking in 9...
Hello world, this is the Nios V/m cpu checking in 10...
Hello world, this is the Nios V/m cpu checking in 11...
Hello world, this is the Nios V/m cpu checking in 12...
Hello world, this is the Nios V/m cpu checking in 13...
Hello world, this is the Nios V/m cpu checking in 14...
```

Ashling RiscFree IDE

- Used for debugging and application development
- Can be used with A53 debugging also on SoC devices
- Applications and Libraries must be created using NiosV-Shell and imported into RiscFree IDE

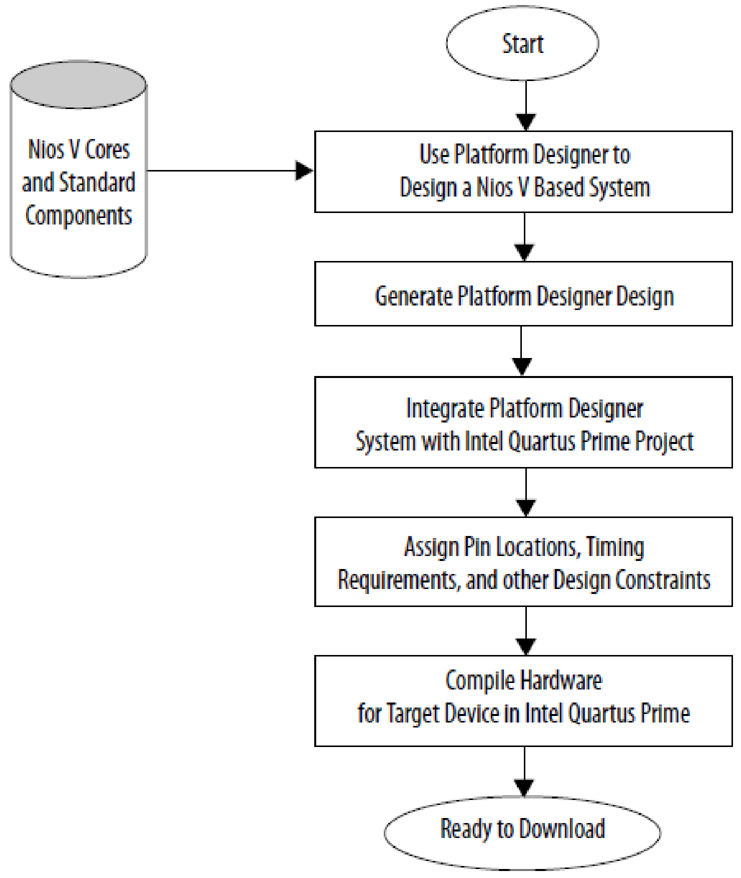


```
/*
 * Copyright (C) 2021 Intel Corporation
 * SPDX-License-Identifier: BSD-3-Clause
 */

#include <stdio.h>
#include <unistd.h>

void looper() {
    for (int i = 0; i < 1000; ++i) {
        printf("Hello world, this is the Nios V/m cpu checking in %d...\n", i);
    }
}

int main() {
    looper();
    usleep(1000000);
    printf("Bye world!\n");
    return 0;
}
```

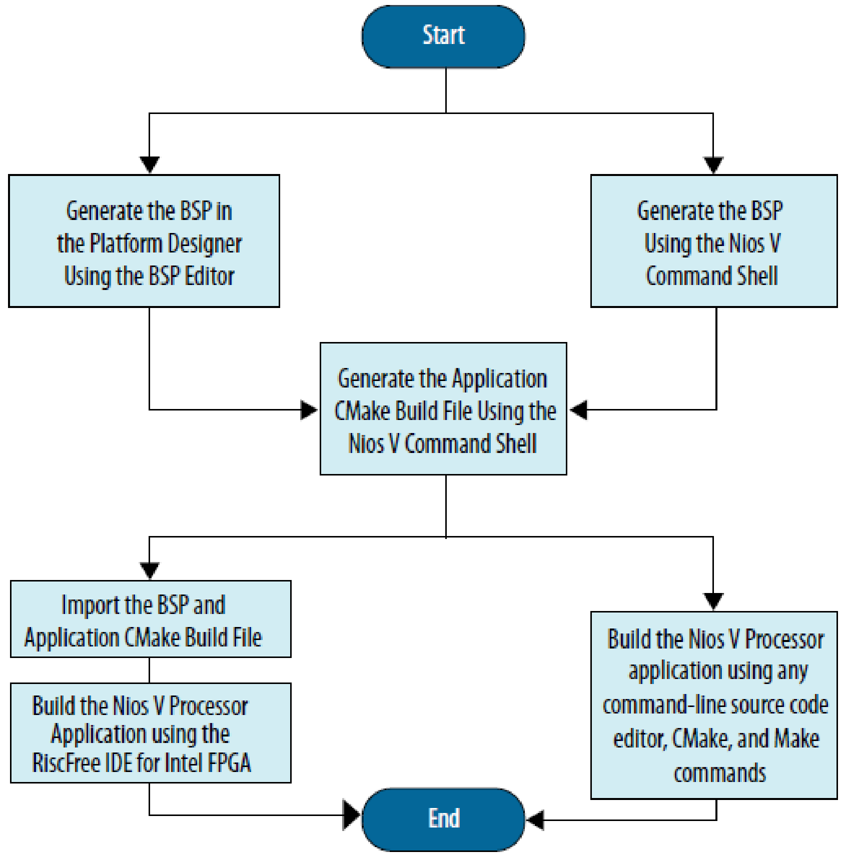
hw_world Debug [Ashling RISC-V (auto-detect) Hardware Debugging]

```
Ashling GDB Server for RISC-V (ash-riscv-gdb-server).
v22.2.2, 31-May-2022, (c)Ashling Microsystems Ltd 2022.

Initializing connection ...
Checking for an active debug connection using the selected debug probe (SN: 1):
Connected to target device "" using USB-Blaster-2 (1) via JTAG at 6MHz.
Info : Active Harts Detected : 1
Info : [0] System architecture : RV32
Info : [0] Number of hardware breakpoints available : 1
Info : [0] Number of program buffers: 8
Info : [0] Number of data registers: 2
Info : [0] Memory Access -> Program buffer
Info : [0] Memory Access -> Abstract access memory
Info : [0] CSR & FP Register Access -> Abstract commands

Waiting for debugger connection on port 50404 for core 0.
Press 'Q' to Quit.
Got a debugger connection from 127.0.0.1 on port 50404.
```

Hardware Flow


Start
Nios V Cores and Standard Components
Use Platform Designer to Design a Nios V Based System
Generate Platform Designer Design
Integrate Platform Designer System with Intel Quartus Prime Project
Assign Pin Locations, Timing Requirements, and other Design Constraints
Compile Hardware for Target Device in Intel Quartus Prime
Ready to Download

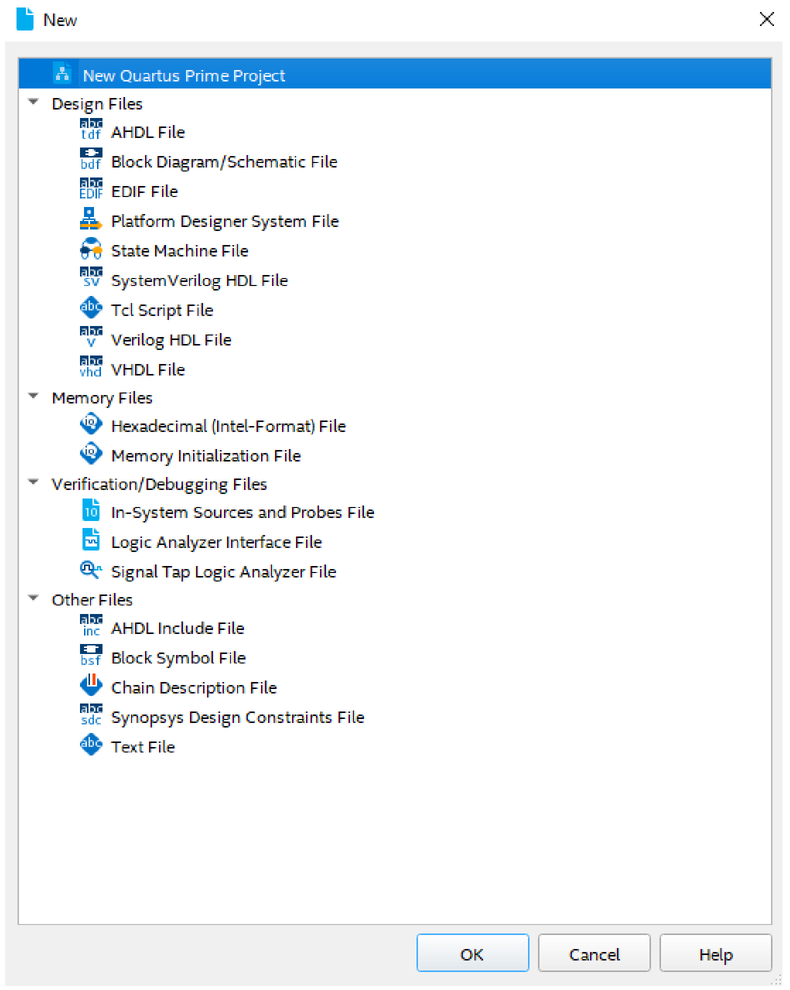
Software Flow


Start
Generate the BSP in the Platform Designer Using the BSP Editor
Generate the BSP Using the Nios V Command Shell
Generate the Application CMake Build File Using the Nios V Command Shell
Import the BSP and Application CMake Build File
Build the Nios V Processor Application using the RiscFree IDE for Intel FPGA
Build the Nios V Processor application using any command-line source code editor, CMake, and Make commands
End

ADIUVO
ENGINEERING AND TRAINING, LTD.

Lab One Hardware Build

Lab One

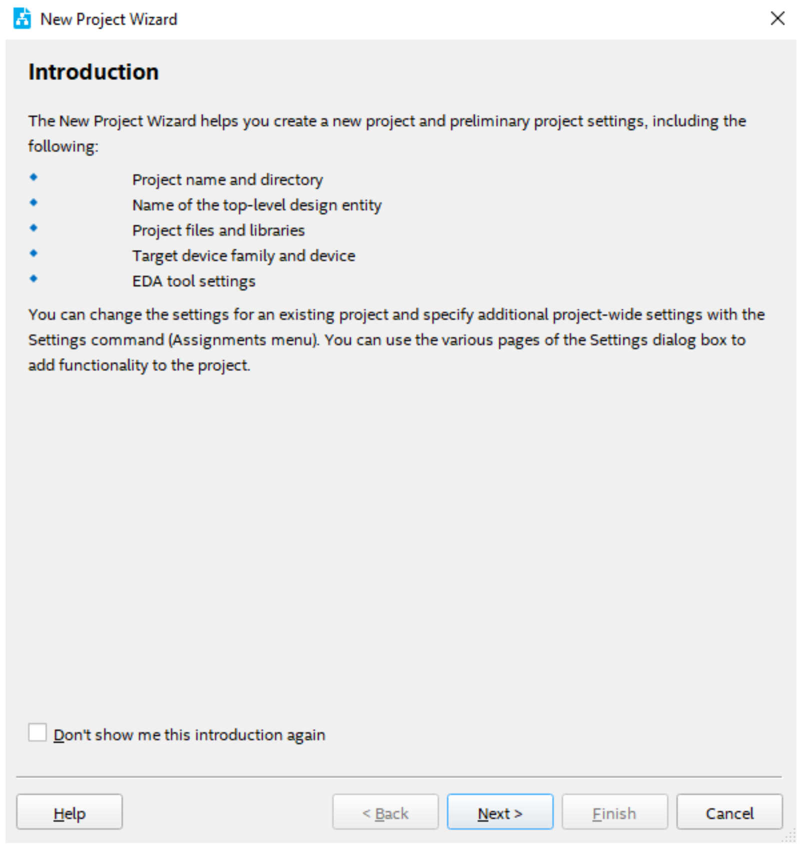


New

New Quartus Prime Project
- Design Files
  - AHDL File
  - Block Diagram/Schematic File
  - EDIF File
  - Platform Designer 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
- Other Files
  - AHDL Include File
  - Block Symbol File
  - Chain Description File
  - Synopsys Design Constraints File
  - Text File

- Open Quartus prime
  - If you do not have licenses, see the sign-up page for how to generate the free trial
  - https://app.livestorm.co/adiuvo-engineering/navigating-nios-v
- Create a new Project
- This project we will target the Agilex F Series SOC Transceiver Kit

Lab One



New Project Wizard

Introduction

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

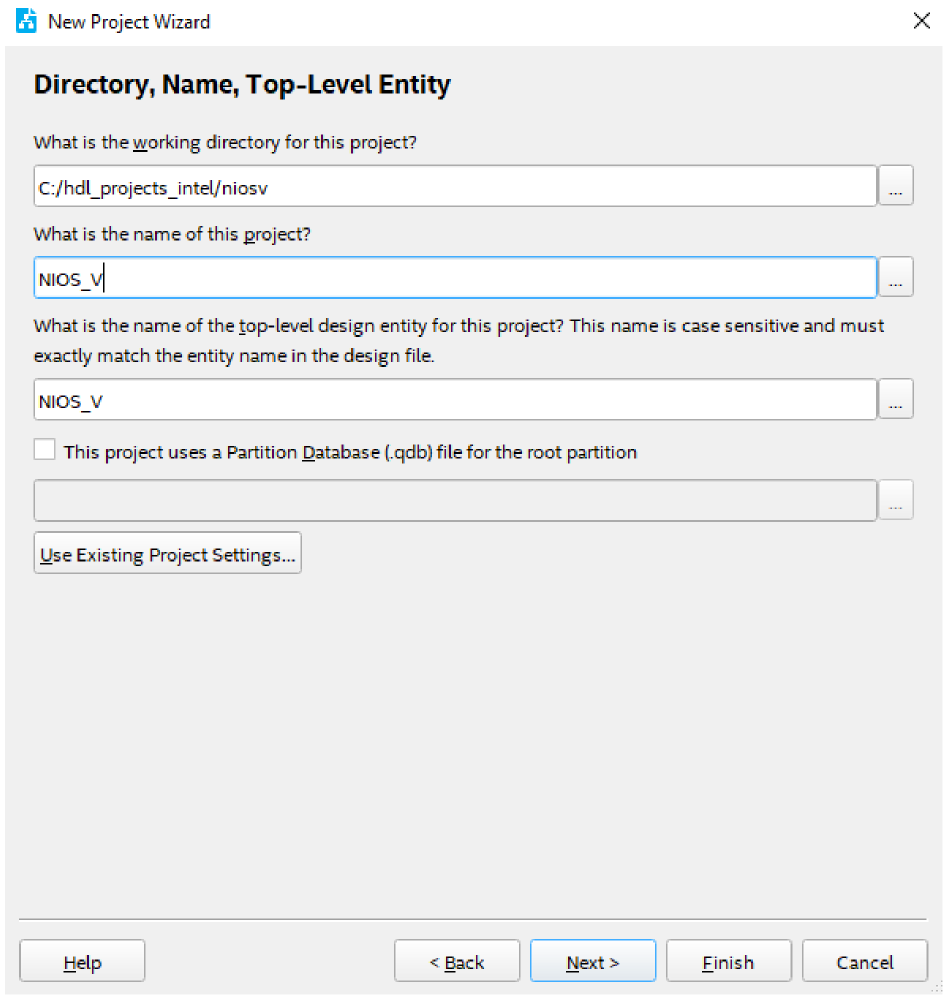
- Project name and directory
- Name of the top-level design entity
- Project files and libraries
- Target device family and device
- EDA tool settings

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

☐ Don't show me this introduction again

- Click on next

Lab One


New Project Wizard

Directory, Name, Top-Level Entity

What is the working directory for this project?

C:/hdl_projects_intel/niosv

What is the name of this project?

NIOS_V

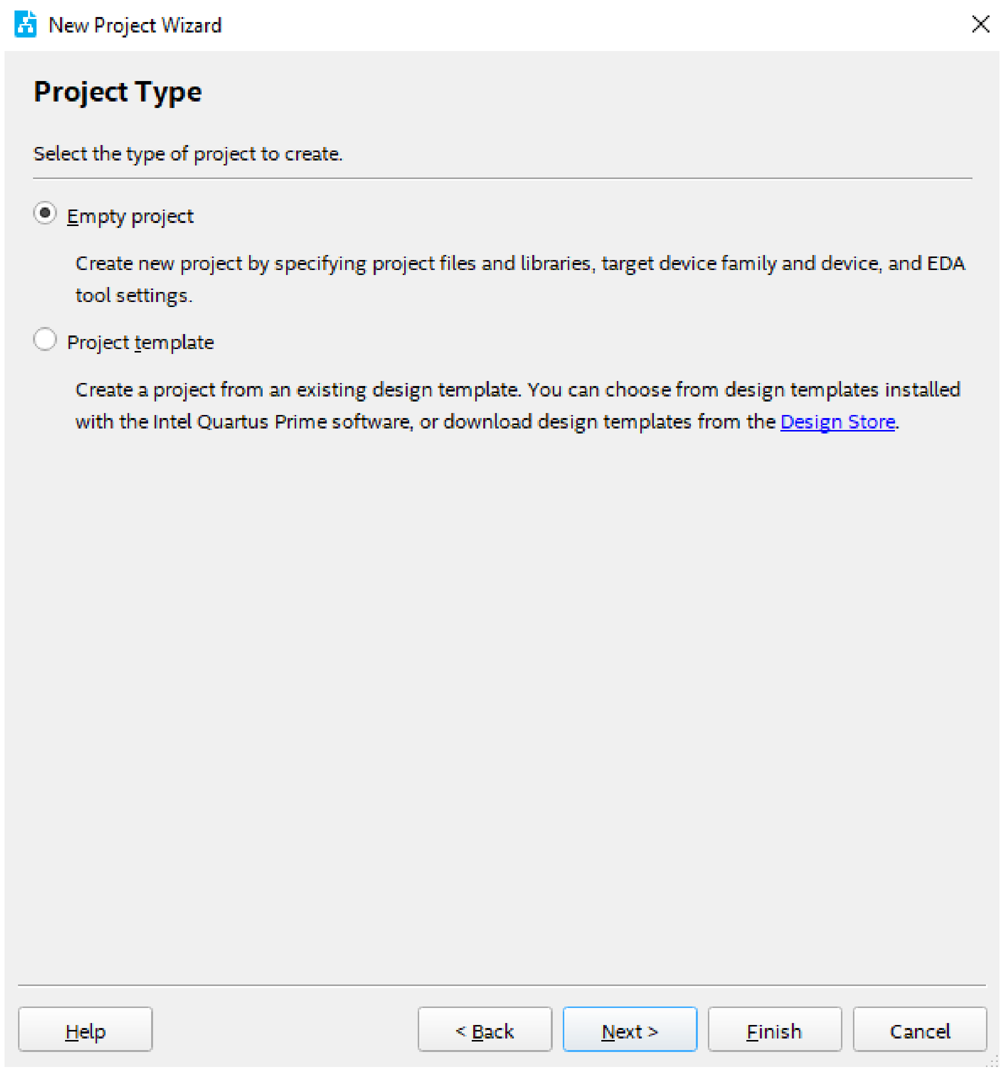
What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

NIOS_V

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

- Select a project location
- Enter a project name
- Enter the top-level entity name – I use NIOS-V
- Do not select Partition Data base or Use Existing Project Settings

Lab One


New Project Wizard

Project Type

Select the type of project to create.

(●) Empty project

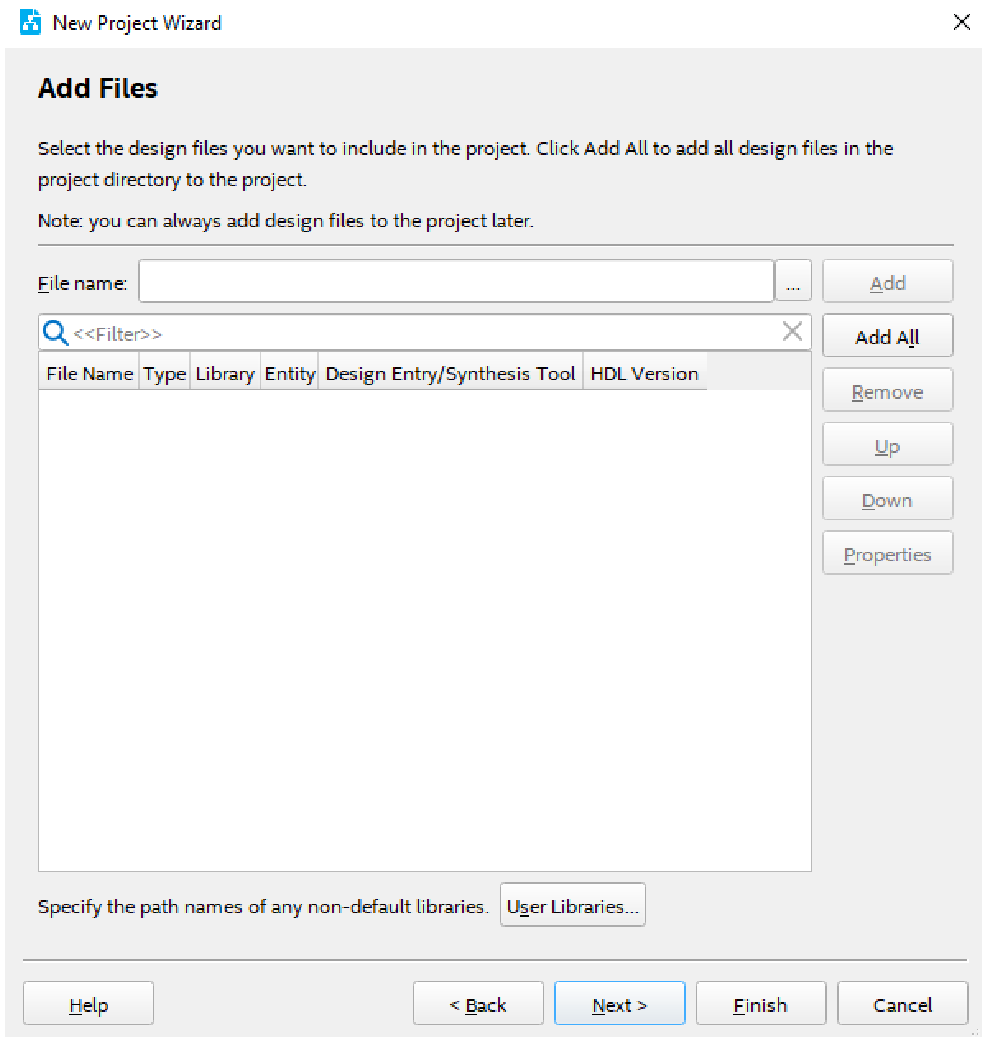
Create new project by specifying project files and libraries, target device family and device, and EDA tool settings.

( ) Project template

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

- Create an Empty project – we will create this design a blank sheet design
- Intel design store does contain a range of design templates which can be used for design acceleration

Lab One



New Project Wizard

Add Files

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

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

File name:

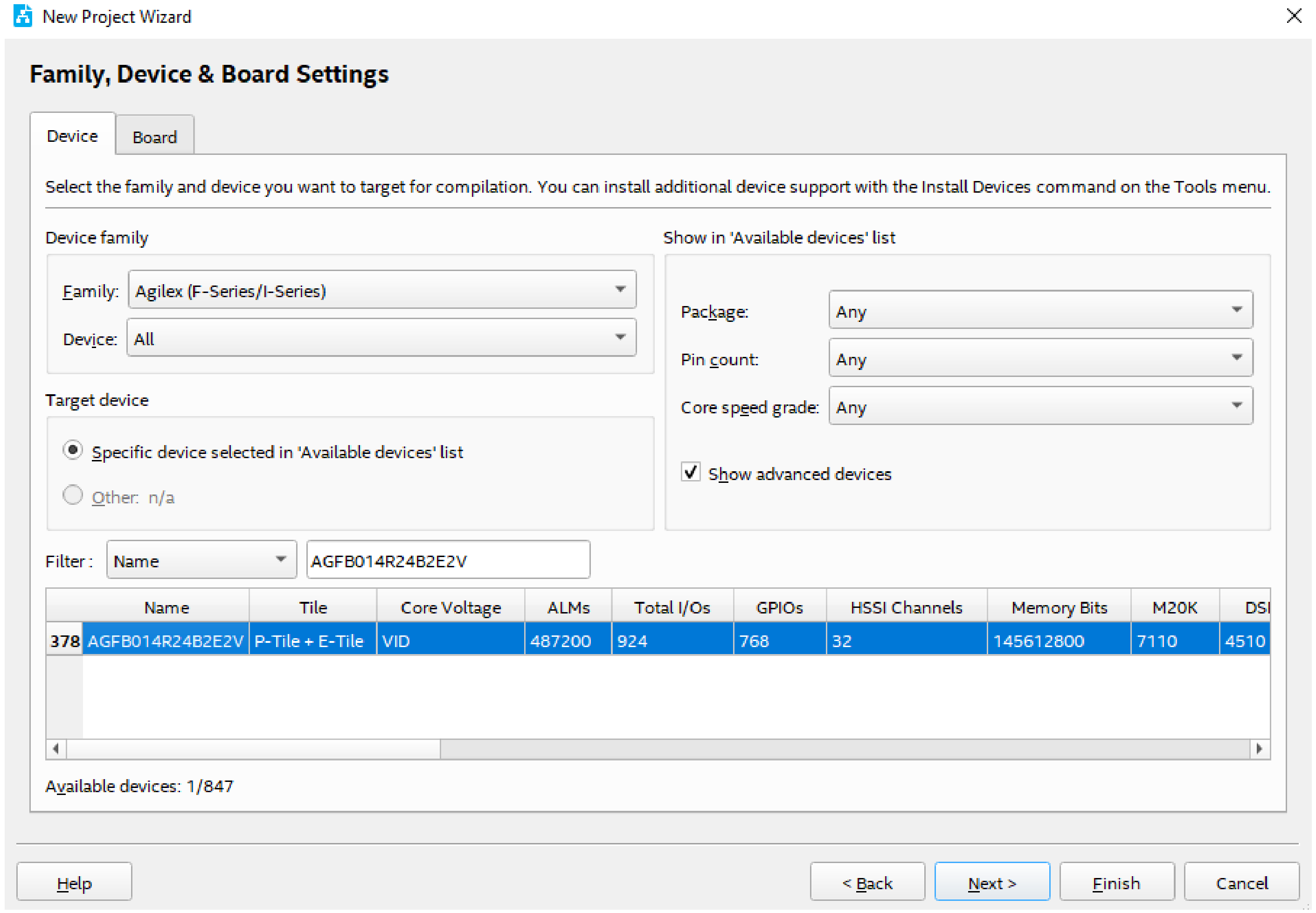
<<Filter>>

File Name	Type	Library	Entity	Design Entry/Synthesis Tool	HDL Version

Specify the path names of any non-default libraries.

- As this is a blank sheet design leave the add files dialog unchanged.

Lab One



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.

Device family

Family: Agilex (F-Series/I-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 AGFB014R24B2E2V

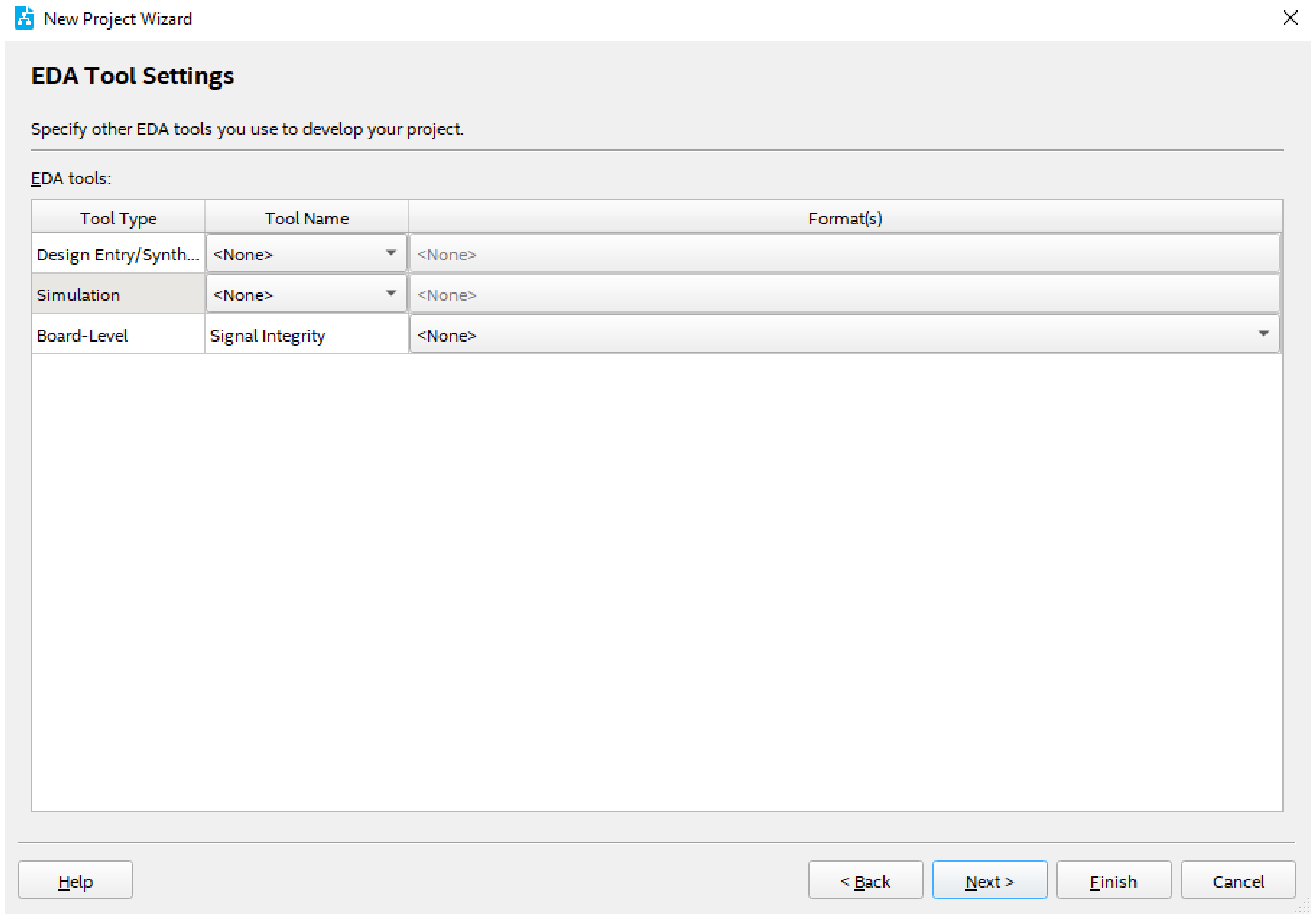
	Name	Tile	Core Voltage	ALMs	Total I/Os	GPIOs	HSSI Channels	Memory Bits	M20K	DSI
378	AGFB014R24B2E2V	P-Tile + E-Tile	VID	487200	924	768	32	145612800	7110	4510

Available devices: 1/847

Help | < Back | Next > | Finish | Cancel

- Select the device which is populated on the Agilex F Series Transceiver SOC development kit.
- Depends on serial number as to ES or Prod
- ES = AGFB014R24A2E2V
- Prod = AGFB014R24B2E2V

Lab One


New Project Wizard

EDA Tool Settings

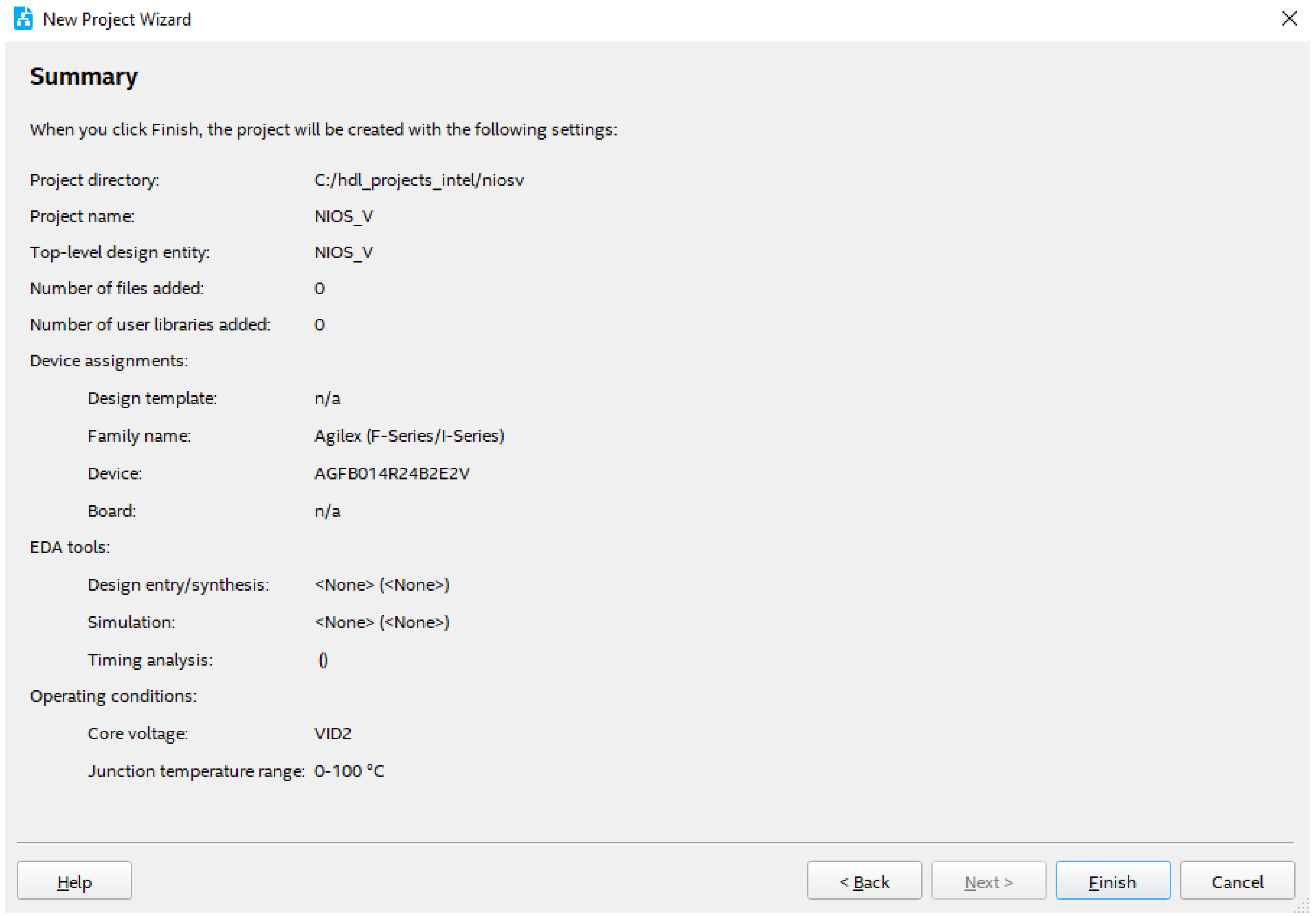
Specify other EDA tools you use to develop your project.

EDA tools:

Tool Type	Tool Name	Format(s)
Design Entry/Synth...	<None>	<None>
Simulation	<None>	<None>
Board-Level	Signal Integrity	<None>

- Leave the EDA tool settings unchanged
- We will use the inbuilt compilation tools

Lab One



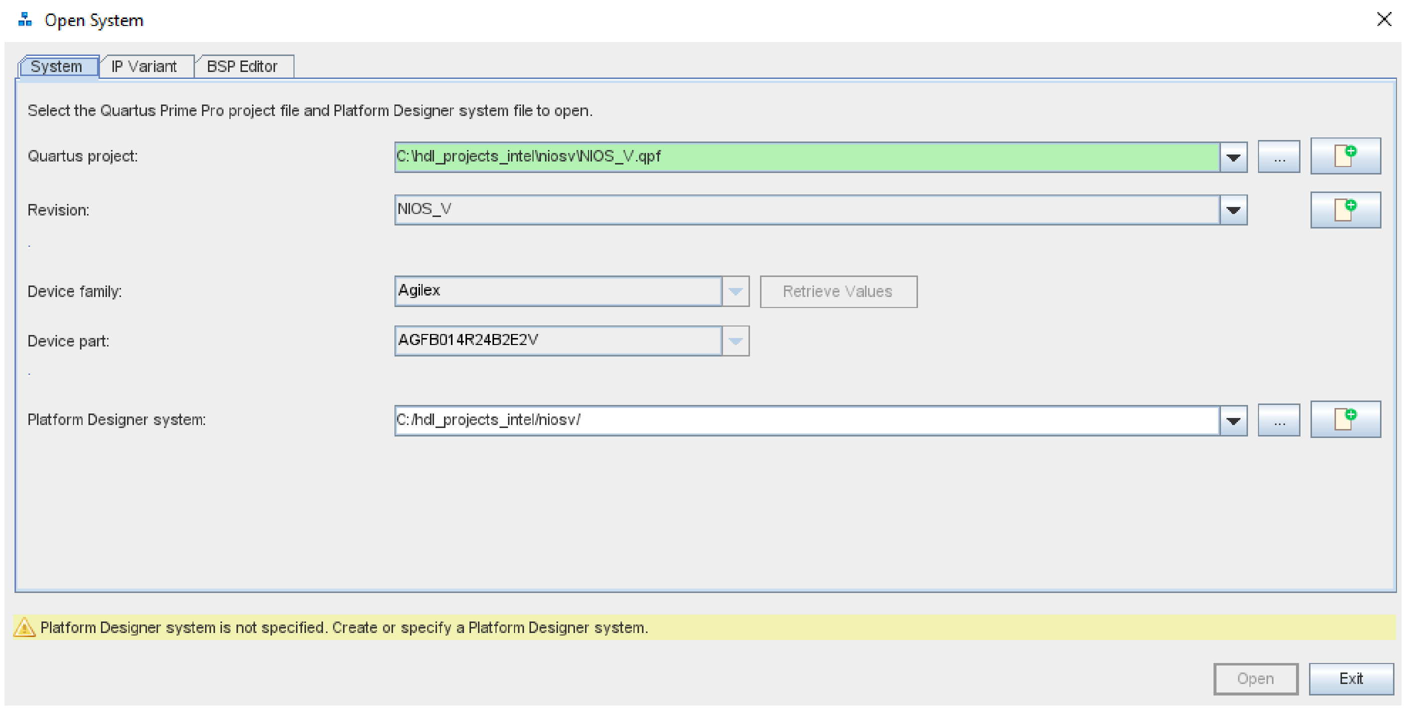
Summary

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

Project directory:	C:/hdl_projects_intel/niosv
Project name:	NIOS_V
Top-level design entity:	NIOS_V
Number of files added:	0
Number of user libraries added:	0
Device assignments:	
Design template:	n/a
Family name:	Agilex (F-Series/I-Series)
Device:	AGFB014R24B2E2V
Board:	n/a
EDA tools:	
Design entry/synthesis:	<None> (<None>)
Simulation:	<None> (<None>)
Timing analysis:	()
Operating conditions:	
Core voltage:	VID2
Junction temperature range:	0-100 °C

- The summary should show
  - Project Location
  - Project name
  - Top Level Name
  - Target Device
  - EDA Tools
  - Operating Conditions

Lab One



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

Quartus project: C:\hdl_projects_intel\niosv\NIOS_V.qpf

Revision: NIOS_V

Device family: Agilex

Device part: AGFB014R24B2E2V

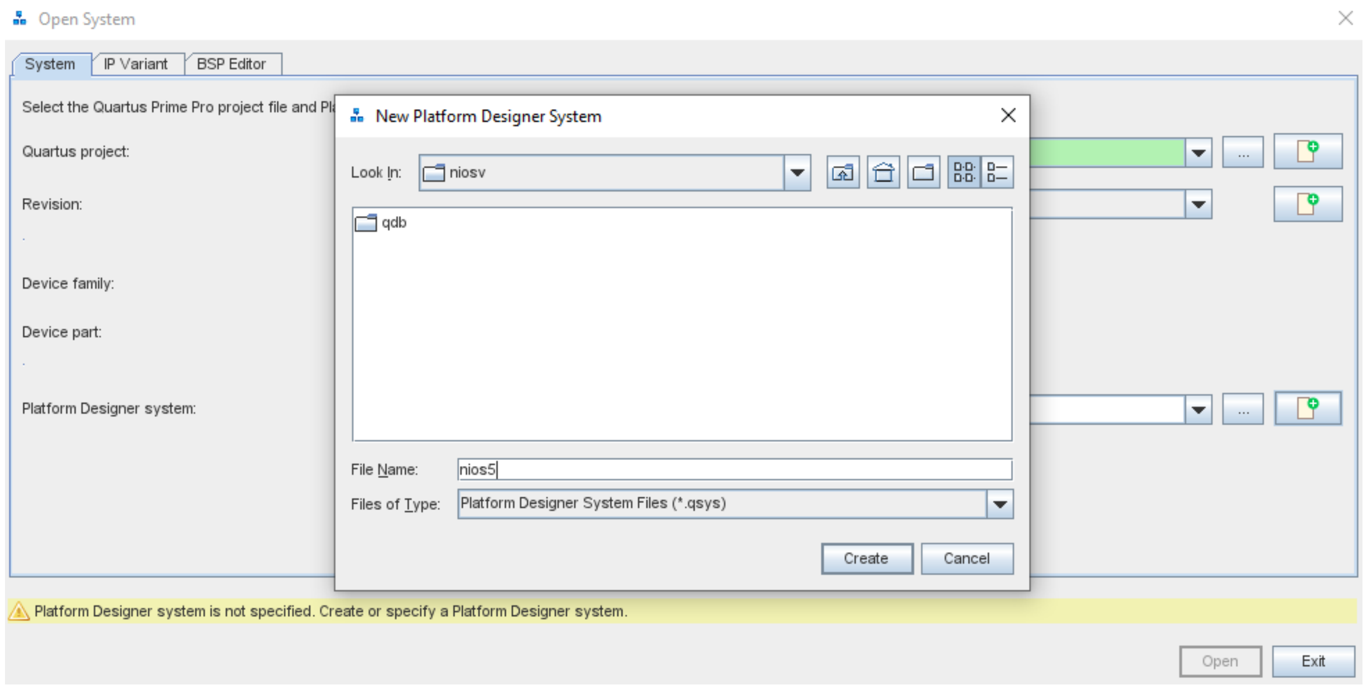
Platform Designer system: C:/hdl_projects_intel/niosv/

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

- In Quartus Prime open Platform Designer
- At the dialog prompt – for Platform Designer System select new and enter the location
- Platform Designer is where we will create our system

ADIUVO
ENGINEERING AND TRAINING, LTD.

Lab One


Open System

System | IP Variant | BSP Editor

Select the Quartus Prime Pro project file and Pl

Quartus project:

Revision:

Device family:

Device part:

Platform Designer system:

New Platform Designer System

Look In: niosv

qdb

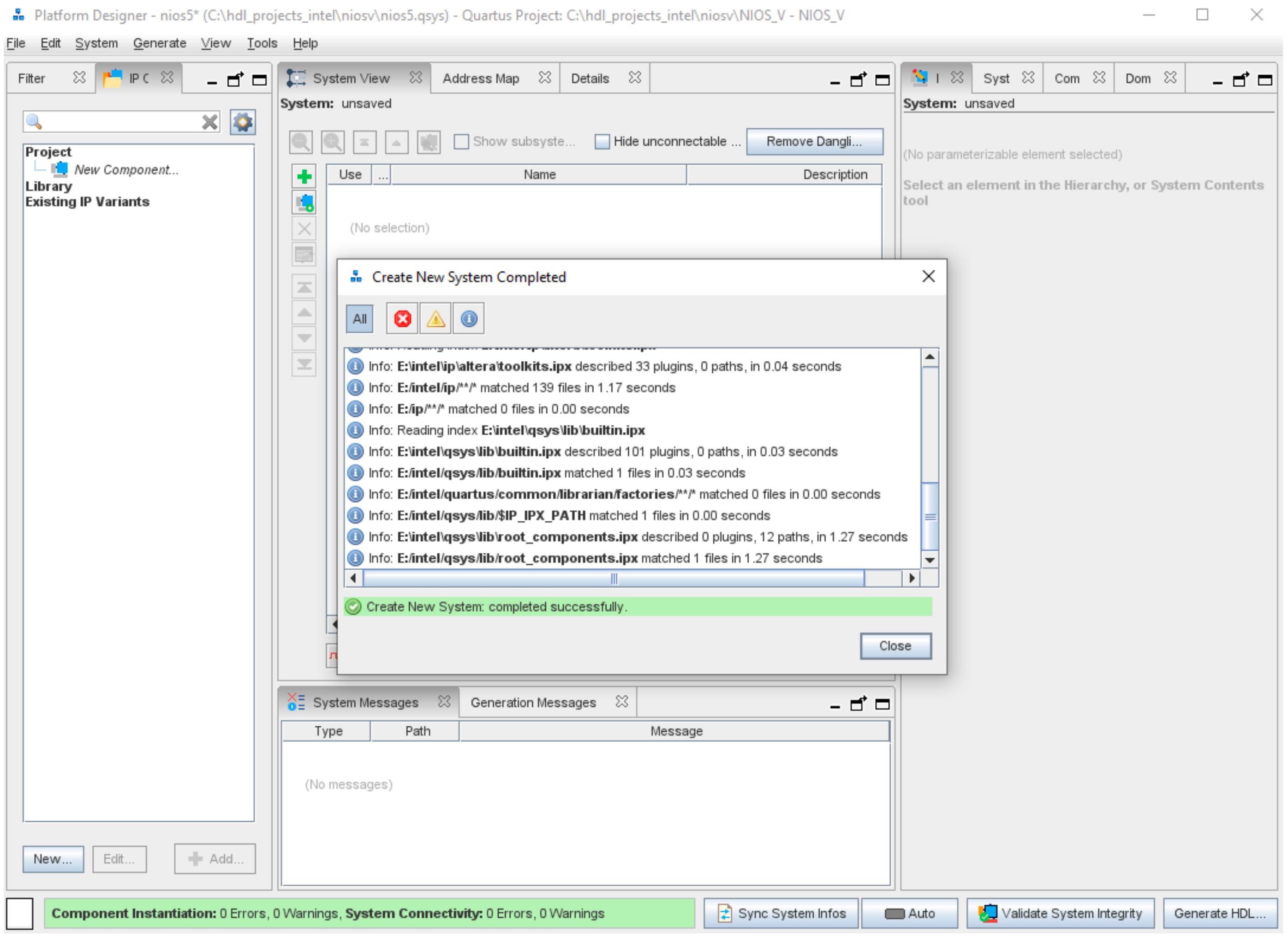
File Name: nios5

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

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

- Select the location for the QSYS file.
- This is the file which contains the Platform Designer Project
- I normally store the QSYS file in the Quartus project to keep them all located together
- Name this the same as your project Top Level Design Entity

Lab One



Platform Designer - nios5* (C:\hdl_projects_intel\niosv\nios5.qsys) - Quartus Project: C:\hdl_projects_intel\niosv\NIOS_V - NIOS_V

System: unsaved

(No selection)

Create New System Completed

```
Info: E:\intel\ip\altera\toolkits.ipx described 33 plugins, 0 paths, in 0.04 seconds
Info: E:/intel/ip/**/* matched 139 files in 1.17 seconds
Info: E:/ip/**/* matched 0 files in 0.00 seconds
Info: Reading index E:\intel\qsys\lib\builtin.ipx
Info: E:\intel\qsys\lib\builtin.ipx described 101 plugins, 0 paths, in 0.03 seconds
Info: E:/intel/qsys/lib/builtin.ipx matched 1 files in 0.03 seconds
Info: E:/intel/quartus/common/librarian/factories/**/* matched 0 files in 0.00 seconds
Info: E:/intel/qsys/lib/$IP_IPX_PATH matched 1 files in 0.00 seconds
Info: E:\intel\qsys\lib\root_components.ipx described 0 plugins, 12 paths, in 1.27 seconds
Info: E:/intel/qsys/lib/root_components.ipx matched 1 files in 1.27 seconds
```

Create New System: completed successfully.

(No parameterizable element selected)

Select an element in the Hierarchy, or System Contents tool

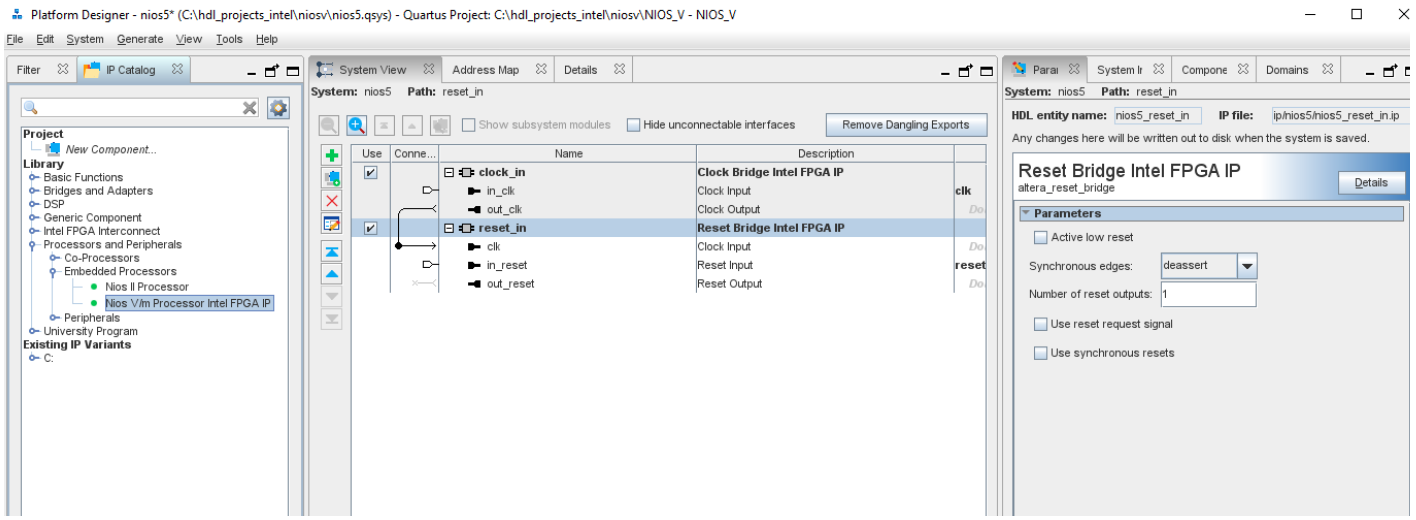
(No messages)

Component Instantiation: 0 Errors, 0 Warnings, System Connectivity: 0 Errors, 0 Warnings

- Saving the new QSYS file will open platform designer ready for a new system

- Click on close

Lab One



Platform Designer - nios5* (C:\hdl_projects_intel\niosv\nios5.qsys) - Quartus Project: C:\hdl_projects_intel\niosv\NIOS_V - NIOS_V

File Edit System Generate View Tools Help

Filter | IP Catalog

- **Project**
  - *New Component...*
- **Library**
  - Basic Functions
  - Bridges and Adapters
  - DSP
  - Generic Component
  - Intel FPGA Interconnect
  - Processors and Peripherals
    - Co-Processors
    - Embedded Processors
      - Nios II Processor
      - Nios V/m Processor Intel FPGA IP
    - Peripherals
  - University Program
- **Existing IP Variants**
  - C:

System View | Address Map | Details

System: nios5 **Path:** reset_in

Show subsystem modules · Hide unconnectable interfaces · Remove Dangling Exports

Use	Conne...	Name	Description	
✓		**clock_in**	**Clock Bridge Intel FPGA IP**	
		in_clk	Clock Input	**clk**
		out_clk	Clock Output	Do
✓		**reset_in**	**Reset Bridge Intel FPGA IP**	
		clk	Clock Input	Do
		in_reset	Reset Input	**reset**
		out_reset	Reset Output	Do

Parar | System Ir | Compone | Domains

System: nios5 **Path:** reset_in

HDL entity name: nios5_reset_in **IP file:** ip/nios5/nios5_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

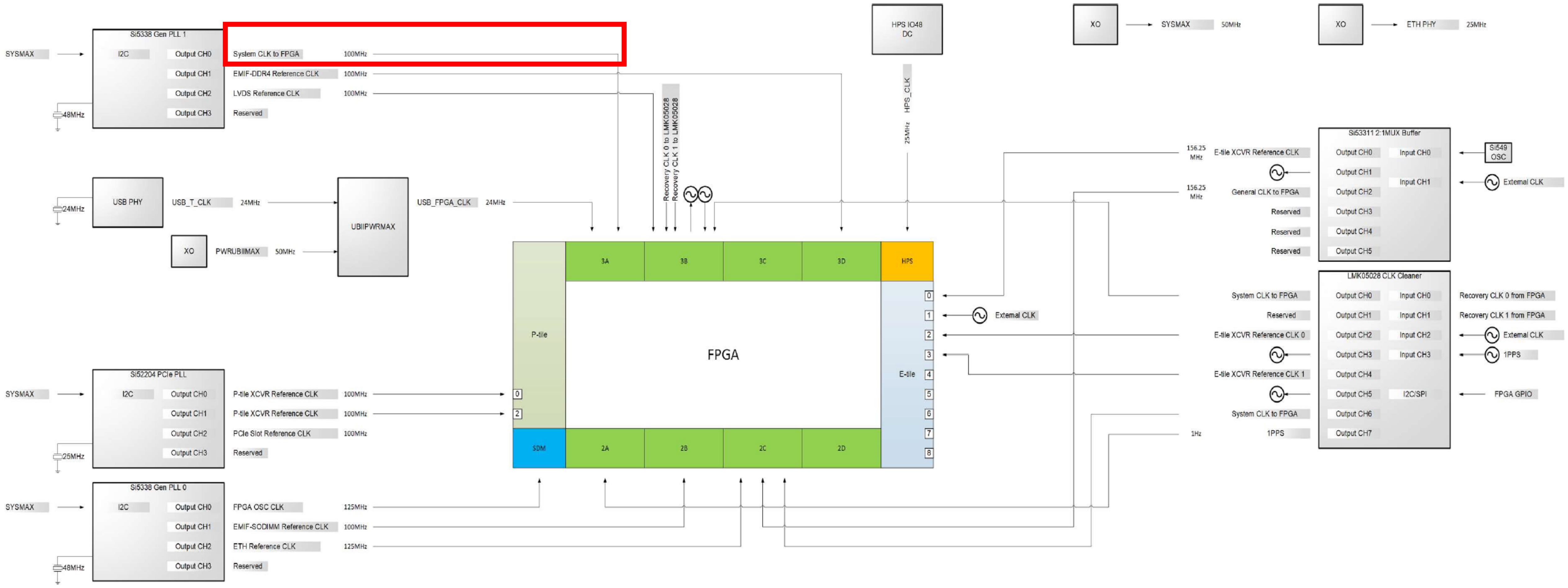
Details

Parameters

- [ ] Active low reset
- Synchronous edges: deassert
- Number of reset outputs: 1
- [ ] Use reset request signal
- [ ] Use synchronous resets

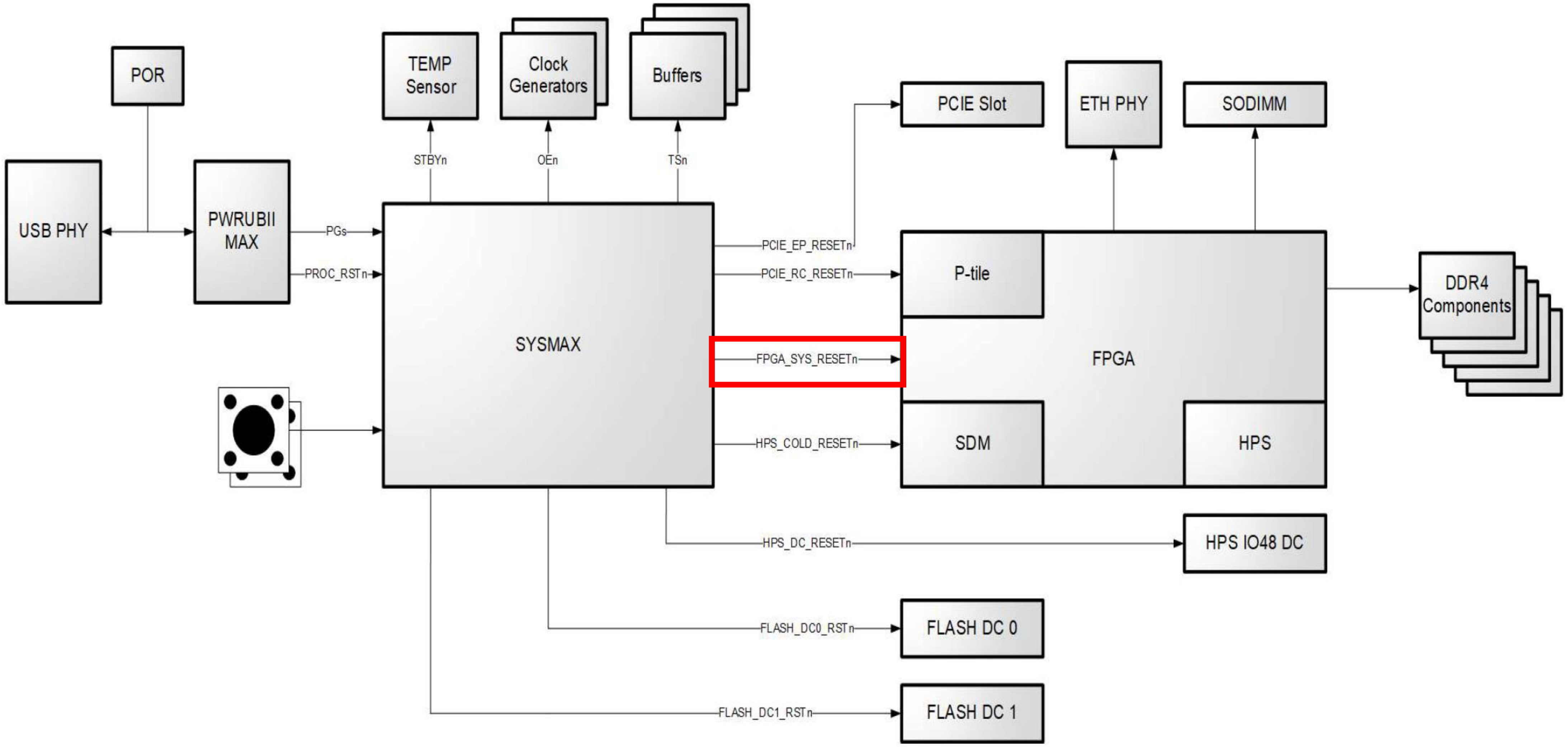
- The initial project will contain two elements a Clock Bridge and Reset Bridge
- Adjust the clock bridge to the input clock frequency – for this project the clk is 100MHz
- Select reset appropriately from the schematics – in this case it is active low

Lab One



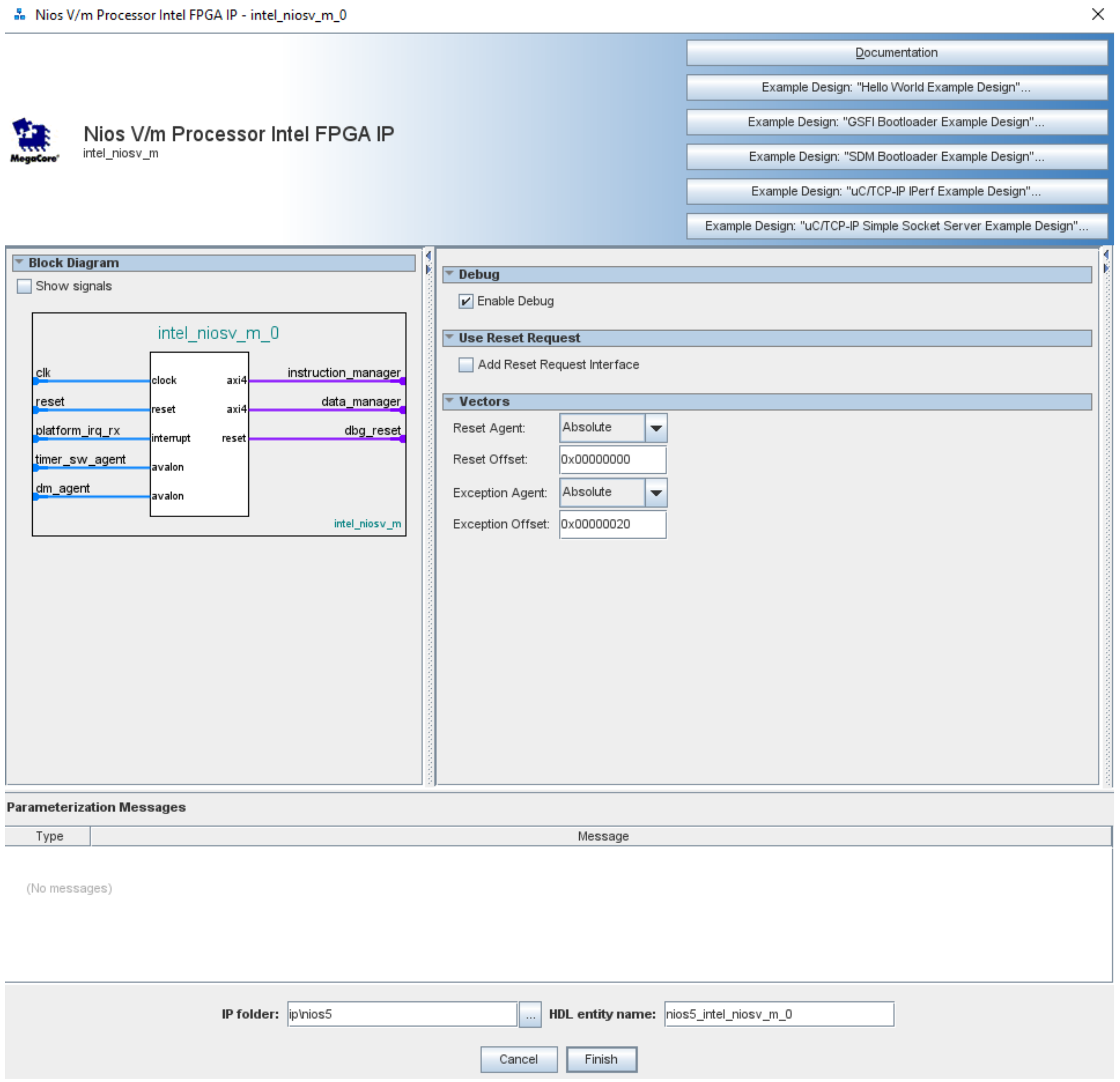
SYSMAX
Si5338 Gen PLL 1
I2C
Output CH0
Output CH1
Output CH2
Output CH3
48MHz
System CLK to FPGA
100MHz
EMIF-DDR4 Reference CLK
100MHz
LVDS Reference CLK
100MHz
Reserved
HPS IO48 DC
HPS_CLK
25MHz
XO
SYSMAX
50MHz
XO
ETH PHY
25MHz
Recovery CLK 0 to LMK05028
Recovery CLK 1 to LMK05028
USB PHY
24MHz
USB_T_CLK
24MHz
UBIIPWRMAX
USB_FPGA_CLK
24MHz
XO
PWRUBIIMAX
50MHz
3A
3B
3C
3D
HPS
P-tile
FPGA
E-tile
SDM
2A
2B
2C
2D
0
1
2
3
4
5
6
7
8
External CLK
Si53311 2:1MUX Buffer
156.25 MHz
E-tile XCVR Reference CLK
156.25 MHz
General CLK to FPGA
Reserved
Reserved
Reserved
Output CH0
Output CH1
Output CH2
Output CH3
Output CH4
Output CH5
Input CH0
Input CH1
Si549 OSC
External CLK
LMK05028 CLK Cleaner
System CLK to FPGA
Reserved
E-tile XCVR Reference CLK 0
E-tile XCVR Reference CLK 1
System CLK to FPGA
1Hz
1PPS
Output CH0
Output CH1
Output CH2
Output CH3
Output CH4
Output CH5
Output CH6
Output CH7
Input CH0
Input CH1
Input CH2
Input CH3
I2C/SPI
Recovery CLK 0 from FPGA
Recovery CLK 1 from FPGA
External CLK
1PPS
FPGA GPIO
SYSMAX
Si52204 PCIe PLL
I2C
Output CH0
Output CH1
Output CH2
Output CH3
25MHz
P-tile XCVR Reference CLK
100MHz
P-tile XCVR Reference CLK
100MHz
PCIe Slot Reference CLK
100MHz
Reserved
SYSMAX
Si5338 Gen PLL 0
I2C
Output CH0
Output CH1
Output CH2
Output CH3
48MHz
FPGA OSC CLK
125MHz
EMIF-SODIMM Reference CLK
100MHz
ETH Reference CLK
125MHz
Reserved

Lab One



POR
TEMP Sensor
Clock Generators
Buffers
USB PHY
PWRUBII MAX
STBYn
OEn
TSn
PGs
PROC_RSTn
SYSMAX
PCIE Slot
ETH PHY
SODIMM
PCIE_EP_RESETn
PCIE_RC_RESETn
FPGA_SYS_RESETn
HPS_COLD_RESETn
P-tile
FPGA
SDM
HPS
DDR4 Components
HPS_DC_RESETn
HPS IO48 DC
FLASH_DC0_RSTn
FLASH DC 0
FLASH_DC1_RSTn
FLASH DC 1

Lab One



Nios V/m Processor Intel FPGA IP - intel_niosv_m_0

Nios V/m Processor Intel FPGA IP
intel_niosv_m

Documentation
Example Design: "Hello World Example Design"...
Example Design: "GSFI Bootloader Example Design"...
Example Design: "SDM Bootloader Example Design"...
Example Design: "uC/TCP-IP IPerf Example Design"...
Example Design: "uC/TCP-IP Simple Socket Server Example Design"...
Block Diagram
Show signals
intel_niosv_m_0
clk
reset
platform_irq_rx
timer_sw_agent
dm_agent
clock
reset
interrupt
avalon
avalon
axi4
axi4
reset
instruction_manager
data_manager
dbg_reset
intel_niosv_m

Debug
- [✓] Enable Debug

Use Reset Request
- [ ] Add Reset Request Interface

Vectors

Reset Agent:	Absolute
Reset Offset:	0x00000000
Exception Agent:	Absolute
Exception Offset:	0x00000020

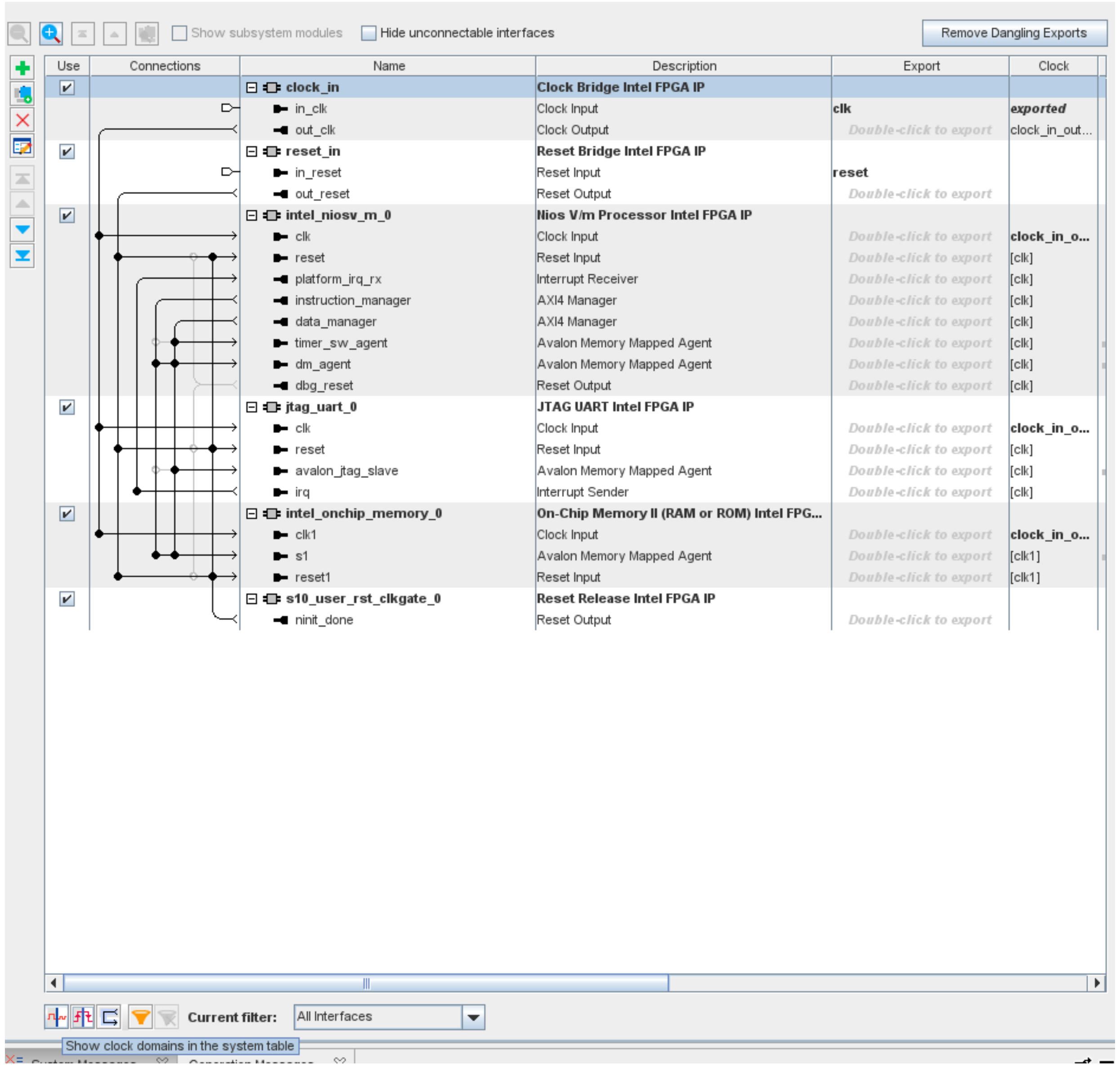
Parameterization Messages

Type	Message
(No messages)	

IP folder: ip\nios5 HDL entity name: nios5_intel_niosv_m_0

- From the IP library on the left of platform designer
- Select the Nios V/m
- Ensure the enable debug is selected

Lab One



Show subsystem modules | Hide unconnectable interfaces | Remove Dangling Exports

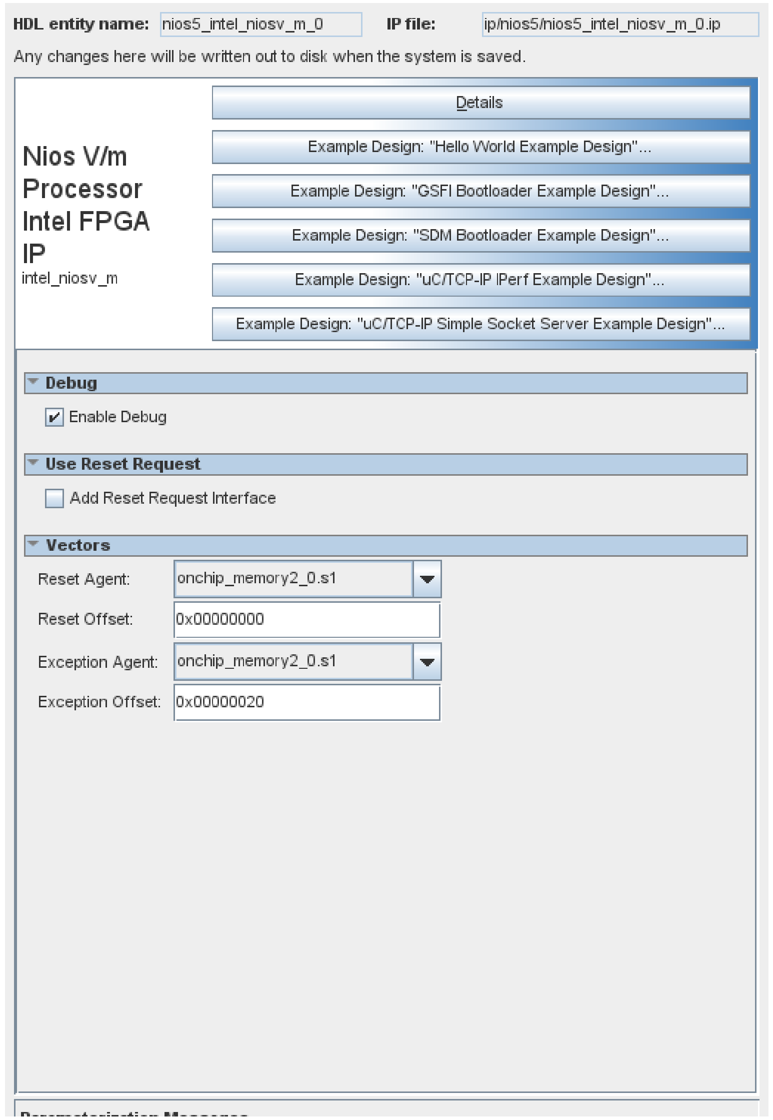
Use	Connections	Name	Description	Export	Clock
✓		**clock_in**	**Clock Bridge Intel FPGA IP**		
		in_clk	Clock Input	**clk**	***exported***
		out_clk	Clock Output	*Double-click to export*	clock_in_out...
✓		**reset_in**	**Reset Bridge Intel FPGA IP**		
		in_reset	Reset Input	**reset**	
		out_reset	Reset Output	*Double-click to export*	
✓		**intel_niosv_m_0**	**Nios V/m Processor Intel FPGA IP**		
		clk	Clock Input	*Double-click to export*	**clock_in_o...**
		reset	Reset Input	*Double-click to export*	[clk]
		platform_irq_rx	Interrupt Receiver	*Double-click to export*	[clk]
		instruction_manager	AXI4 Manager	*Double-click to export*	[clk]
		data_manager	AXI4 Manager	*Double-click to export*	[clk]
		timer_sw_agent	Avalon Memory Mapped Agent	*Double-click to export*	[clk]
		dm_agent	Avalon Memory Mapped Agent	*Double-click to export*	[clk]
		dbg_reset	Reset Output	*Double-click to export*	[clk]
✓		**jtag_uart_0**	**JTAG UART Intel FPGA IP**		
		clk	Clock Input	*Double-click to export*	**clock_in_o...**
		reset	Reset Input	*Double-click to export*	[clk]
		avalon_jtag_slave	Avalon Memory Mapped Agent	*Double-click to export*	[clk]
		irq	Interrupt Sender	*Double-click to export*	[clk]
✓		**intel_onchip_memory_0**	**On-Chip Memory II (RAM or ROM) Intel FPG...**		
		clk1	Clock Input	*Double-click to export*	**clock_in_o...**
		s1	Avalon Memory Mapped Agent	*Double-click to export*	[clk1]
		reset1	Reset Input	*Double-click to export*	[clk1]
✓		**s10_user_rst_clkgate_0**	**Reset Release Intel FPGA IP**		
		ninit_done	Reset Output	*Double-click to export*	

Current filter: All Interfaces

Show clock domains in the system table

- From the IP Library add in
  - JTAG UART
  - On Chip Memory
  - Reset Release IP
- Connect the Instruction & Data master to the OCM & dm_agent
- Connect the Data Master to the JTAG UART
- Connect the clocks and resets
- Connect the IRQ line from the JTAG UART

Lab One



HDL entity name: nios5_intel_niosv_m_0 IP file: ip/nios5/nios5_intel_niosv_m_0.ip

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

Nios V/m Processor Intel FPGA IP
intel_niosv_m

Details
Example Design: "Hello World Example Design"...
Example Design: "GSFI Bootloader Example Design"...
Example Design: "SDM Bootloader Example Design"...
Example Design: "uC/TCP-IP IPerf Example Design"...
Example Design: "uC/TCP-IP Simple Socket Server Example Design"...

Debug

[✓] Enable Debug

Use Reset Request

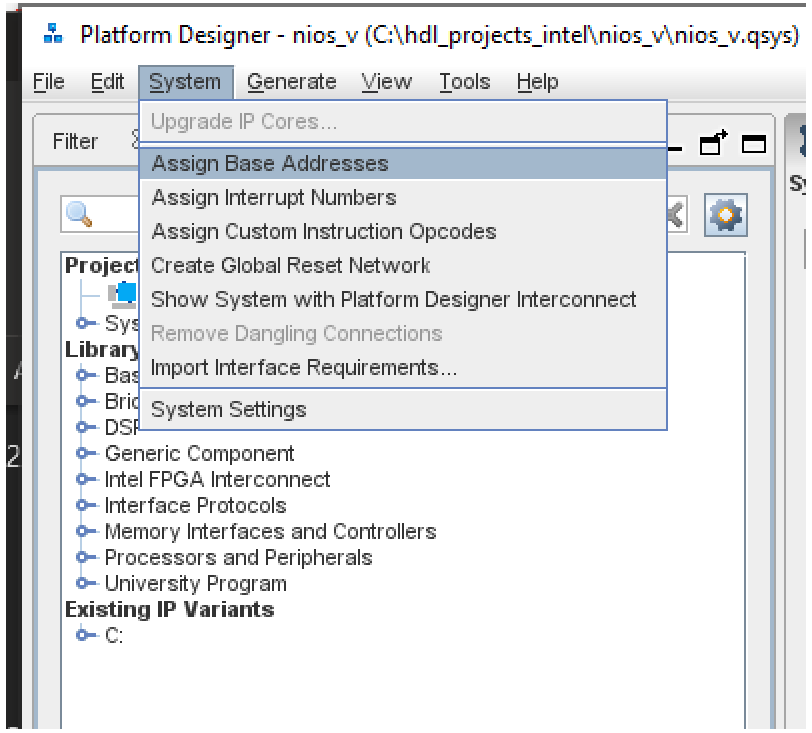
[ ] Add Reset Request Interface

Vectors

Reset Agent:	onchip_memory2_0.s1
Reset Offset:	0x00000000
Exception Agent:	onchip_memory2_0.s1
Exception Offset:	0x00000020

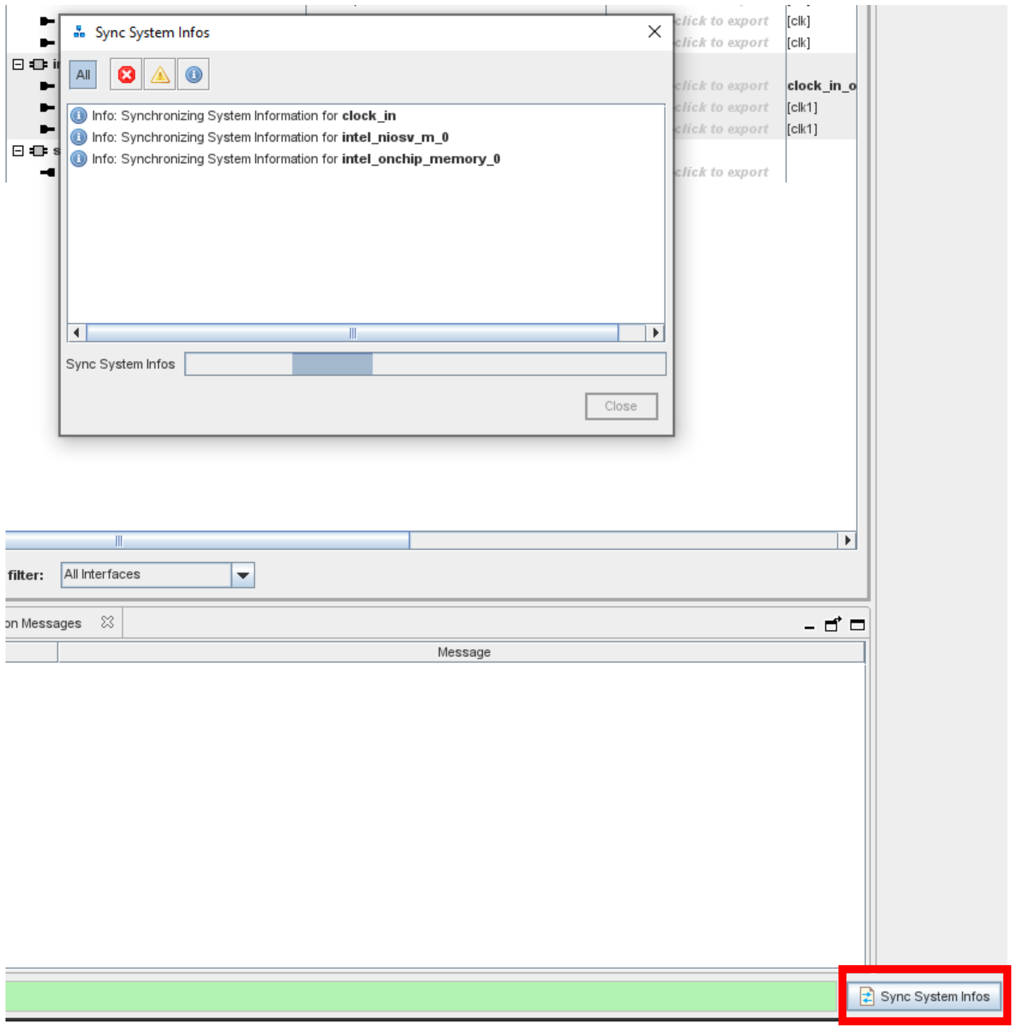
- Update the Nios V/m vectors to reference to the On Chip Memory

Lab One


Platform Designer - nios_v (C:\hdl_projects_intel\nios_v\nios_v.qsys)

- With all the IP included we need to assign the base addresses

Lab One

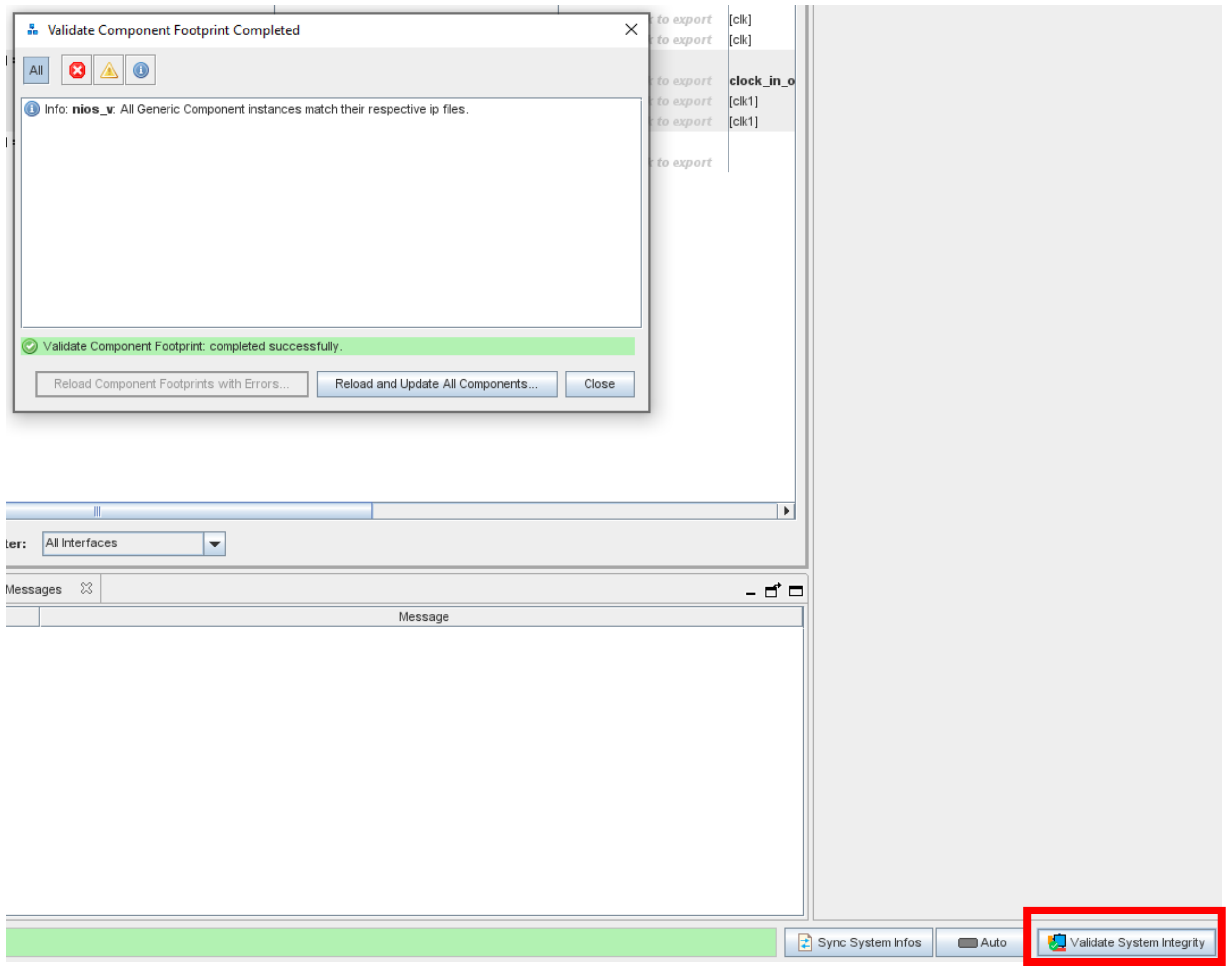


Sync System Infos

Info: Synchronizing System Information for **clock_in**
Info: Synchronizing System Information for **intel_niosv_m_0**
Info: Synchronizing System Information for **intel_onchip_memory_0**

- Select Sync System Info
- When completed close the dialog

Lab One



Validate Component Footprint Completed

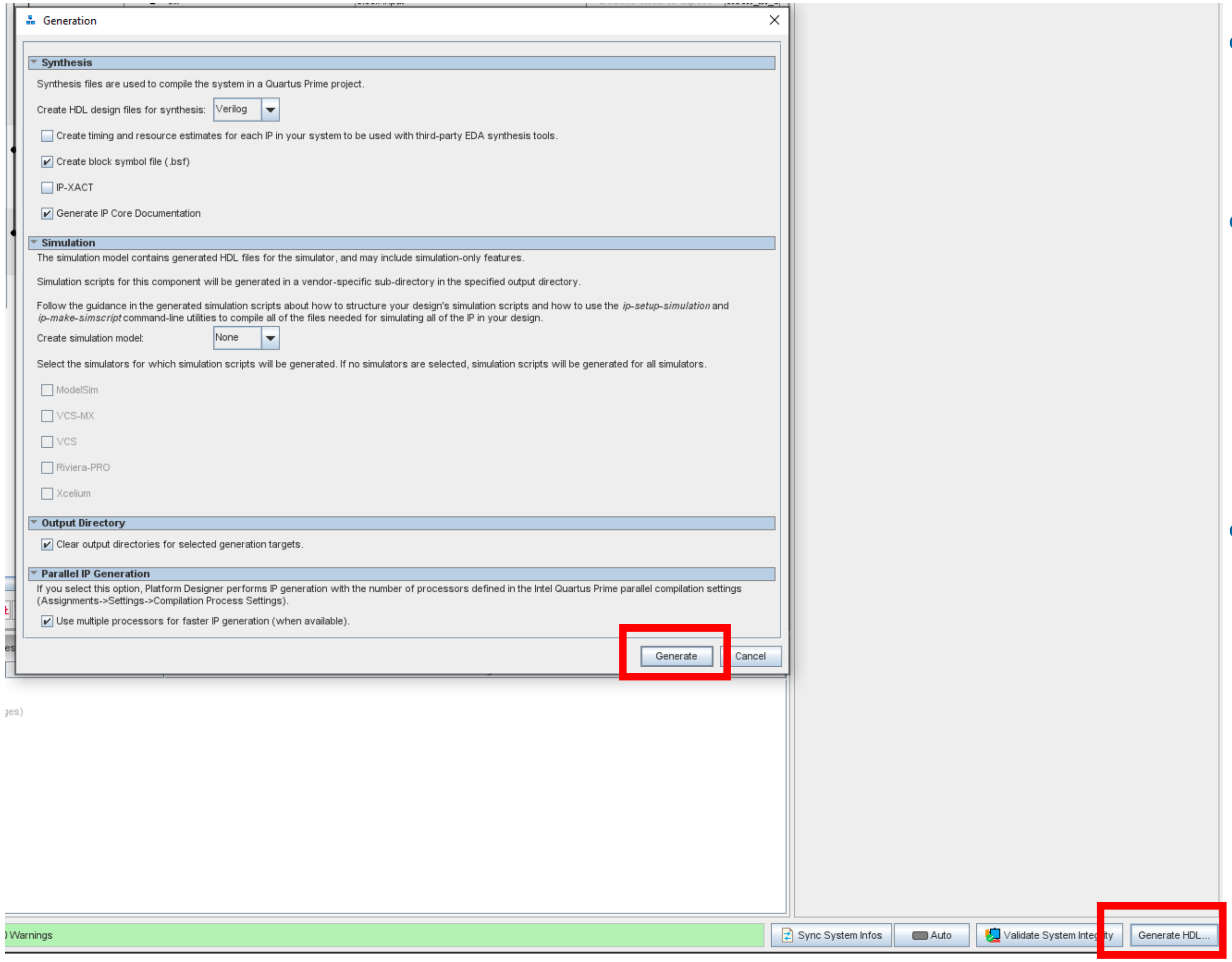
Info: **nios_v**: All Generic Component instances match their respective ip files.

Validate Component Footprint: completed successfully.

All
Reload Component Footprints with Errors...
Reload and Update All Components...
Close
to export
[clk]
clock_in_o
[clk1]
ter:
All Interfaces
Messages
Message
Sync System Infos
Auto
Validate System Integrity

- Click on Validate System Integrity
- Once completed click on close

Lab One


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.
- [x] Create block symbol file (.bsf)
- [ ] IP-XACT
- [x] 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 the simulators for which simulation scripts will be generated. If no simulators are selected, simulation scripts will be generated for all simulators.

- [ ] ModelSim
- [ ] VCS-MX
- [ ] VCS
- [ ] Riviera-PRO
- [ ] Xcelium

Output Directory

- [x] 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).

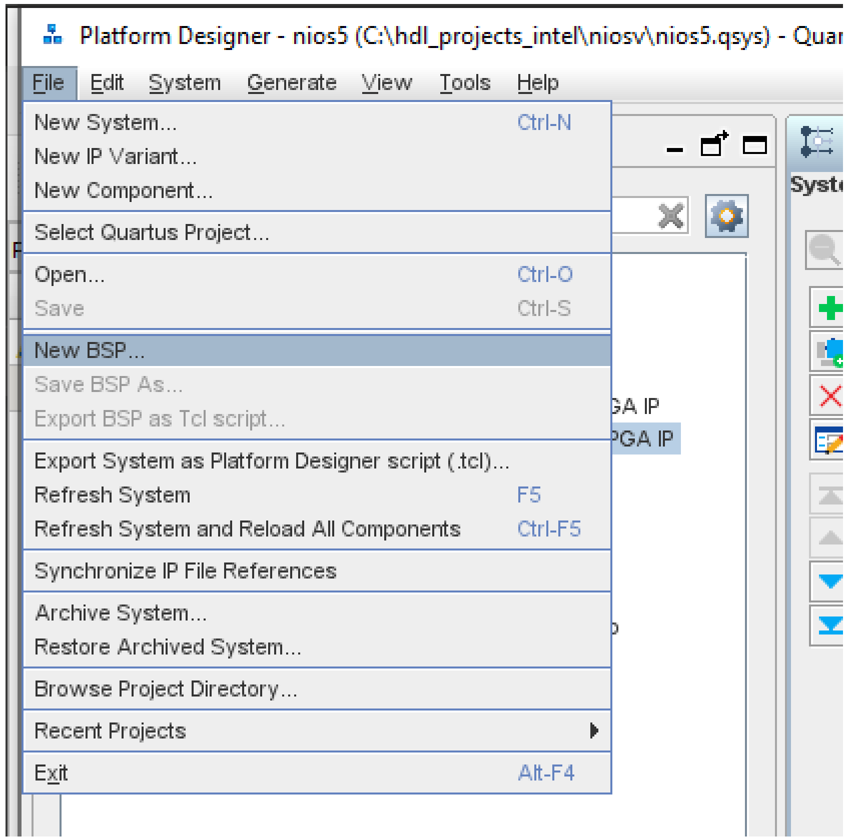
- [x] Use multiple processors for faster IP generation (when available).

Generate Cancel

Warnings Sync System Infos Auto Validate System Integrity Generate HDL...

- Click generate HDL
- In the dialog which opens select your preferred language type generate the output files
- Once completed close the dialog

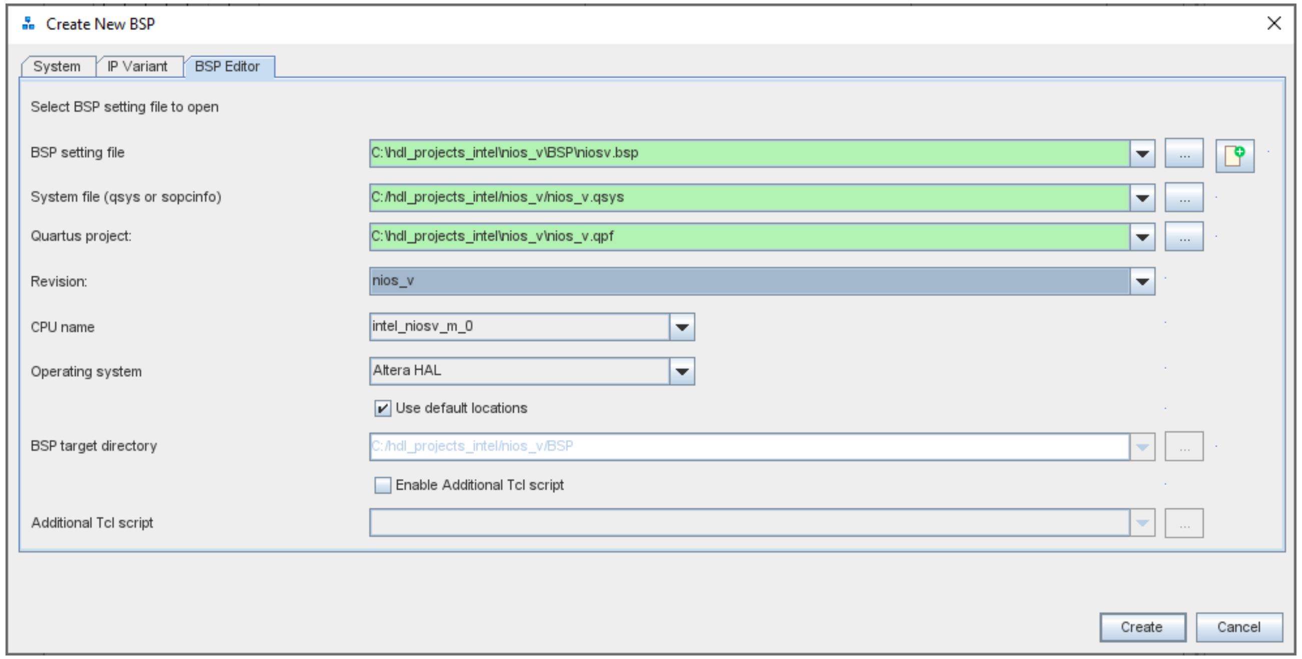
Lab One



Platform Designer - nios5 (C:\hdl_projects_intel\niosv\nios5.qsys) - Quar
File Edit System Generate View Tools Help
New System... Ctrl-N
New IP Variant...
New Component...
Select Quartus Project...
Open... Ctrl-O
Save Ctrl-S
New BSP...
Save BSP As...
Export BSP as Tcl script...
Export System as Platform Designer script (.tcl)...
Refresh System F5
Refresh System and Reload All Components Ctrl-F5
Synchronize IP File References
Archive System...
Restore Archived System...
Browse Project Directory...
Recent Projects
Exit Alt-F4

- In a change from Nios 2 systems we generate the BSP in platform designer for Nios V/m
- From the File menu select New BSP

Lab One

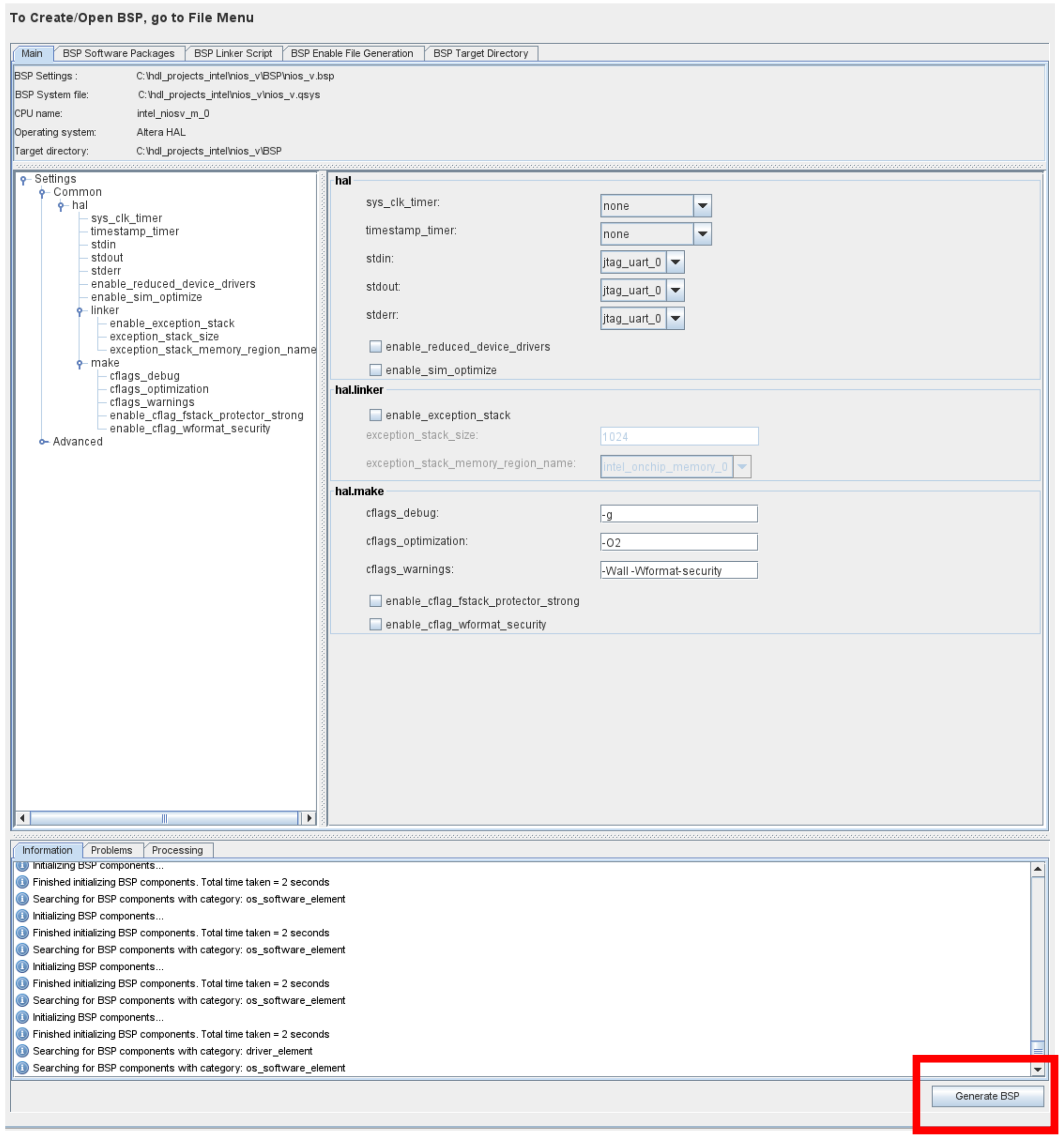


Select BSP setting file to open

BSP setting file	C:\hdl_projects_intel\nios_v\BSP\niosv.bsp
System file (qsys or sopcinfo)	C:/hdl_projects_intel/nios_v/nios_v.qsys
Quartus project:	C:\hdl_projects_intel\nios_v\nios_v.qpf
Revision:	nios_v
CPU name	intel_niosv_m_0
Operating system	Altera HAL
	☑ Use default locations
BSP target directory	C:/hdl_projects_intel/nios_v/BSP
	☐ Enable Additional Tcl script
Additional Tcl script	

- Select the location for the BSP to be generated
- Provide the QSYS File & Quartus Project
- Ensure the processor is selected
- Select the Altera HAL

Lab One



To Create/Open BSP, go to File Menu

BSP Settings :	C:\hdl_projects_intel\nios_v\BSP\nios_v.bsp
BSP System file:	C:\hdl_projects_intel\nios_v\nios_v.qsys
CPU name:	intel_niosv_m_0
Operating system:	Altera HAL
Target directory:	C:\hdl_projects_intel\nios_v\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_name
      make
        cflags_debug
        cflags_optimization
        cflags_warnings
        enable_cflag_fstack_protector_strong
        enable_cflag_wformat_security
  Advanced
```

hal

sys_clk_timer:	none
timestamp_timer:	none
stdin:	jtag_uart_0
stdout:	jtag_uart_0
stderr:	jtag_uart_0

☐ enable_reduced_device_drivers
☐ enable_sim_optimize

hal.linker

☐ enable_exception_stack

exception_stack_size:	1024
exception_stack_memory_region_name:	intel_onchip_memory_0

hal.make

cflags_debug:	-g
cflags_optimization:	-O2
cflags_warnings:	-Wall -Wformat-security

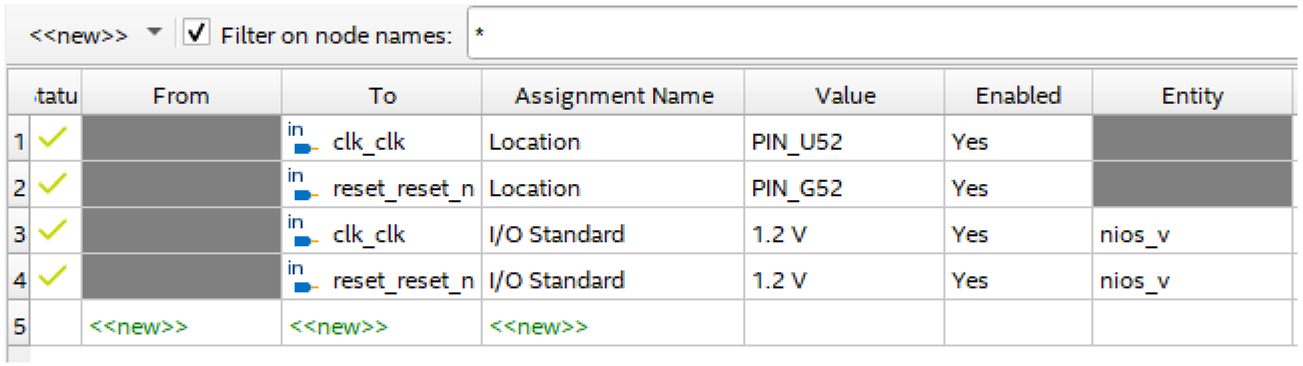
☐ enable_cflag_fstack_protector_strong
☐ enable_cflag_wformat_security

```
Initializing BSP components...
Finished initializing BSP components. Total time taken = 2 seconds
Searching for BSP components with category: os_software_element
Initializing BSP components...
Finished initializing BSP components. Total time taken = 2 seconds
Searching for BSP components with category: os_software_element
Initializing BSP components...
Finished initializing BSP components. Total time taken = 2 seconds
Searching for BSP components with category: os_software_element
Initializing BSP components...
Finished initializing BSP components. Total time taken = 2 seconds
Searching for BSP components with category: driver_element
Searching for BSP components with category: os_software_element
```

- This will open the BSP editor it is here we are able to select configurations for the BSP
- Set the Sys Clk Timer and Timestamp Timer to none
- Generate the BSP
- We can now close Platform Designer

ADIUVO
ENGINEERING AND TRAINING, LTD.

Lab One

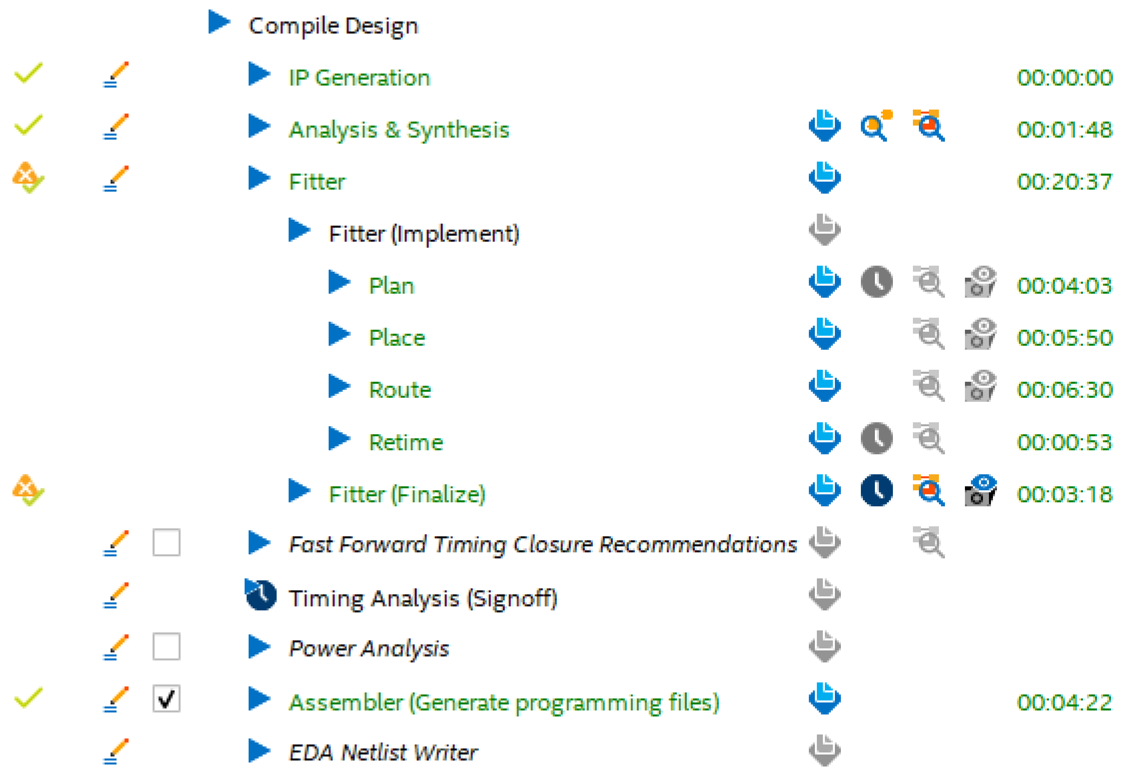


<<new>> Filter on node names: *

tatu		From	To	Assignment Name	Value	Enabled	Entity
1	✓		clk_clk	Location	PIN_U52	Yes	
2	✓		reset_reset_n	Location	PIN_G52	Yes	
3	✓		clk_clk	I/O Standard	1.2 V	Yes	nios_v
4	✓		reset_reset_n	I/O Standard	1.2 V	Yes	nios_v
5		<<new>>	<<new>>	<<new>>			

- In Quartus open the assignment editor and select the pin locations for the board in question.
- On the Agilex F Series Transceiver SoC Development Kit
  - Clk = U52
  - Resetn = G52

Lab One



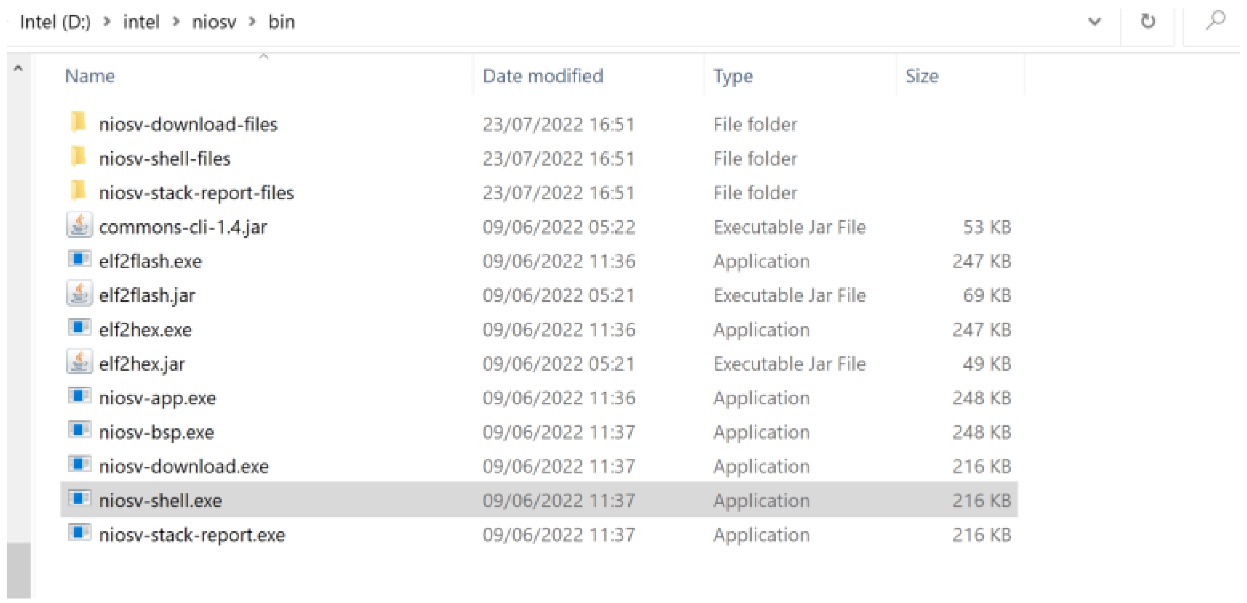
✓	Compile Design	
✓	IP Generation	00:00:00
✓	Analysis & Synthesis	00:01:48
	Fitter	00:20:37
	Fitter (Implement)	
	Plan	00:04:03
	Place	00:05:50
	Route	00:06:30
	Retime	00:00:53
	Fitter (Finalize)	00:03:18
☐	*Fast Forward Timing Closure Recommendations*	
	Timing Analysis (Signoff)	
☐	*Power Analysis*	
☑	Assembler (Generate programming files)	00:04:22
	EDA Netlist Writer	

- Compile the design and generate the programming file

ADIUVO
ENGINEERING AND TRAINING, LTD.

Lab Two

Lab Two

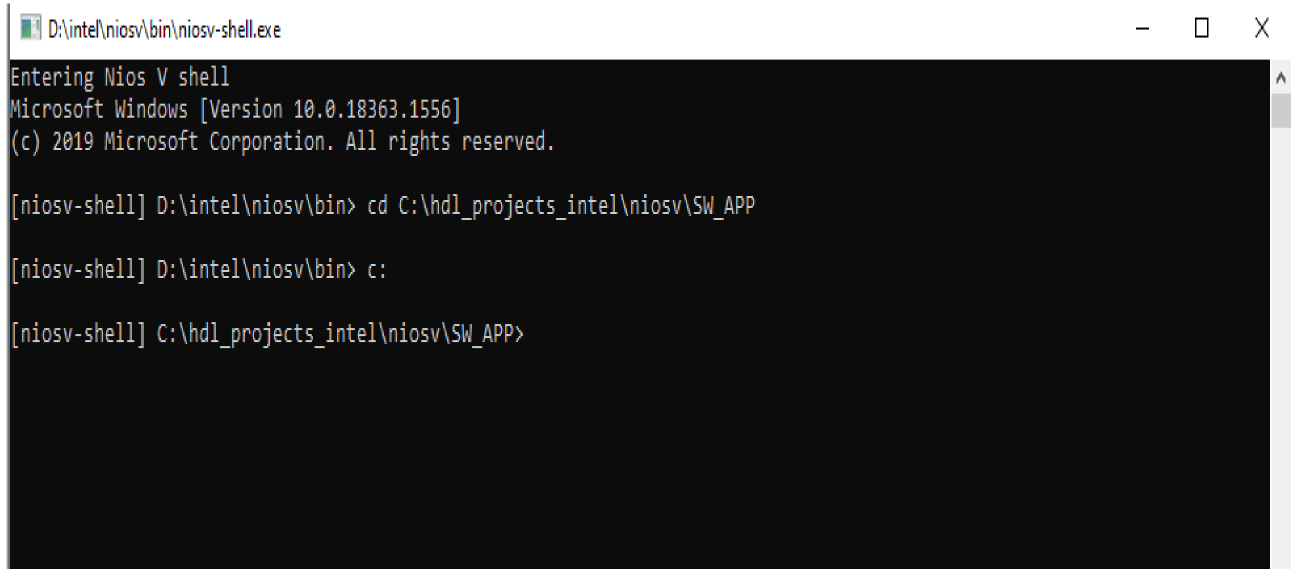


Intel (D:) › intel › niosv › bin

Name	Date modified	Type	Size
niosv-download-files	23/07/2022 16:51	File folder	
niosv-shell-files	23/07/2022 16:51	File folder	
niosv-stack-report-files	23/07/2022 16:51	File folder	
commons-cli-1.4.jar	09/06/2022 05:22	Executable Jar File	53 KB
elf2flash.exe	09/06/2022 11:36	Application	247 KB
elf2flash.jar	09/06/2022 05:21	Executable Jar File	69 KB
elf2hex.exe	09/06/2022 11:36	Application	247 KB
elf2hex.jar	09/06/2022 05:21	Executable Jar File	49 KB
niosv-app.exe	09/06/2022 11:36	Application	248 KB
niosv-bsp.exe	09/06/2022 11:37	Application	248 KB
niosv-download.exe	09/06/2022 11:37	Application	216 KB
niosv-shell.exe	09/06/2022 11:37	Application	216 KB
niosv-stack-report.exe	09/06/2022 11:37	Application	216 KB

- Navigate to the Intel Install directory
- Under Nios V / Bin
- Run the NIOSV Shell
- This uses the BSP previously created to the application we will Run on the target processor.

Lab Two



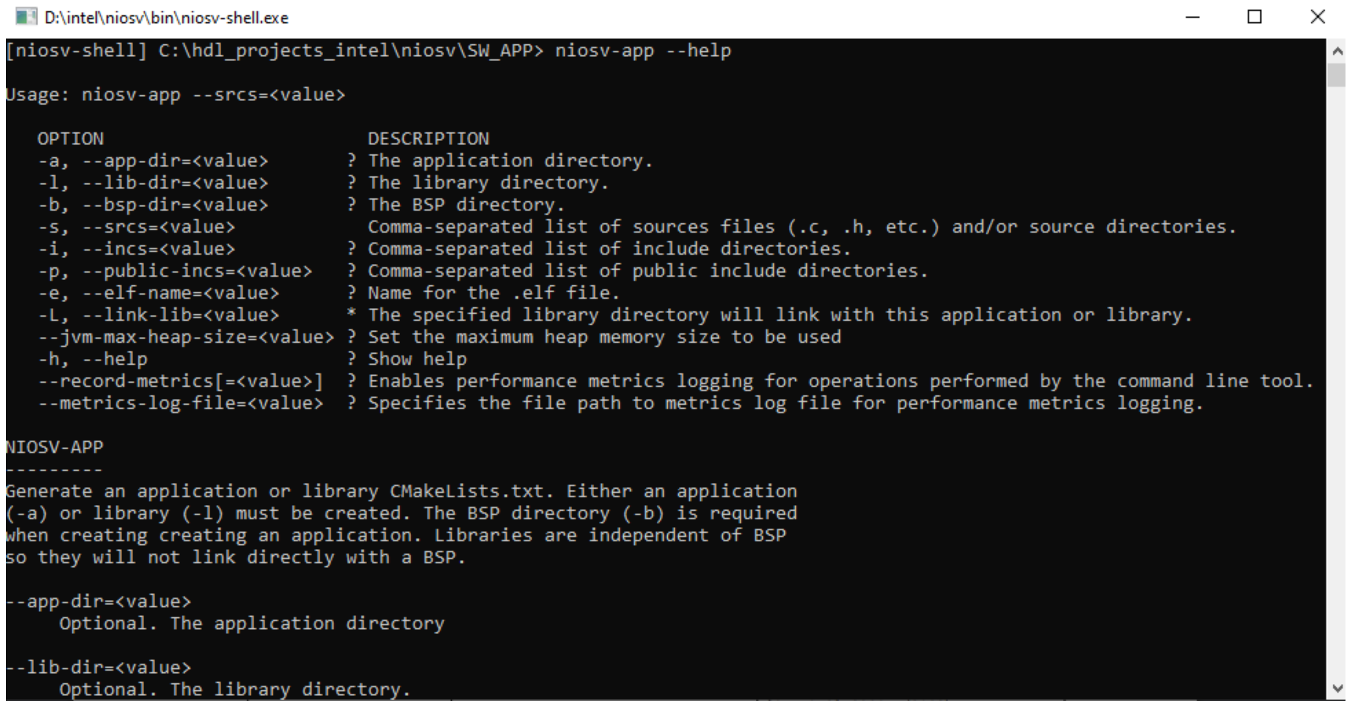
D:\intel\niosv\bin\niosv-shell.exe

```
Entering Nios V shell
Microsoft Windows [Version 10.0.18363.1556]
(c) 2019 Microsoft Corporation. All rights reserved.

[niosv-shell] D:\intel\niosv\bin> cd C:\hdl_projects_intel\niosv\SW_APP

[niosv-shell] D:\intel\niosv\bin> c:

[niosv-shell] C:\hdl_projects_intel\niosv\SW_APP>
```

- Change directory into the directory containing the SW source we wish to use in our application.

ADIUVO
ENGINEERING AND TRAINING, LTD.

Lab Two

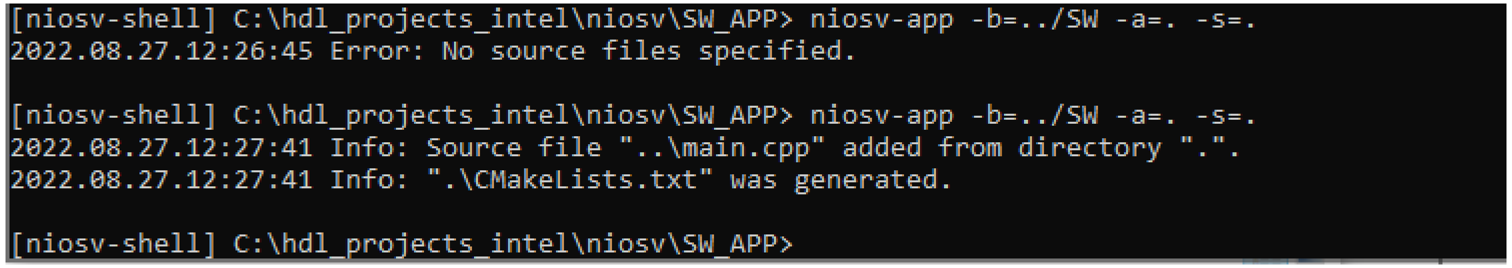


```
[niosv-shell] C:\hdl_projects_intel\niosv\SW_APP> niosv-app --help

Usage: niosv-app --srcs=<value>

   OPTION                        DESCRIPTION
   -a, --app-dir=<value>       ? The application directory.
   -l, --lib-dir=<value>       ? The library directory.
   -b, --bsp-dir=<value>       ? The BSP directory.
   -s, --srcs=<value>            Comma-separated list of sources files (.c, .h, etc.) and/or source directories.
   -i, --incs=<value>          ? Comma-separated list of include directories.
   -p, --public-incs=<value>   ? Comma-separated list of public include directories.
   -e, --elf-name=<value>      ? Name for the .elf file.
   -L, --link-lib=<value>      * The specified library directory will link with this application or library.
   --jvm-max-heap-size=<value> ? Set the maximum heap memory size to be used
   -h, --help                  ? Show help
   --record-metrics[=<value>]  ? Enables performance metrics logging for operations performed by the command line tool.
   --metrics-log-file=<value>  ? Specifies the file path to metrics log file for performance metrics logging.

NIOSV-APP
---------
Generate an application or library CMakeLists.txt. Either an application
(-a) or library (-l) must be created. The BSP directory (-b) is required
when creating creating an application. Libraries are independent of BSP
so they will not link directly with a BSP.

--app-dir=<value>
     Optional. The application directory

--lib-dir=<value>
     Optional. The library directory.
```

- To see the commands available, in the Nios V Shall use the --help command

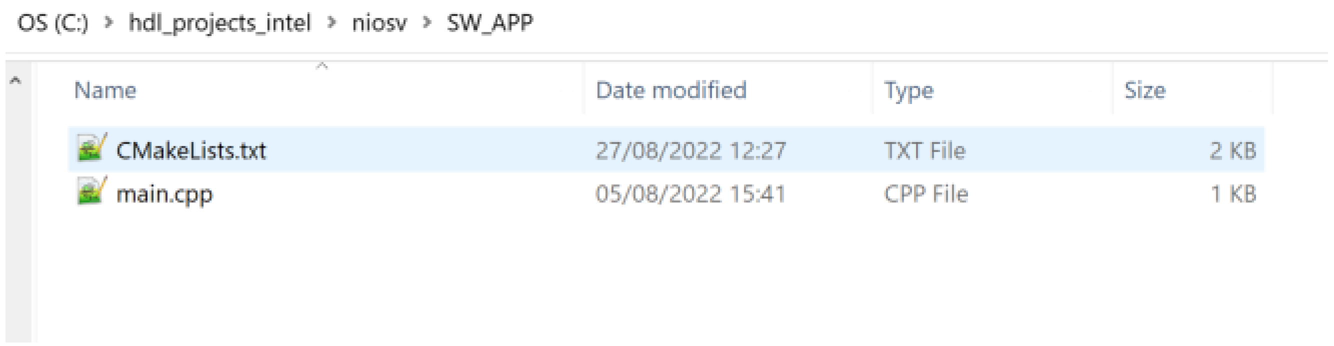
Lab Two

- To create a new application, we need to define
  - BSP Directory
  - Application Directory
  - Source Files



```
[niosv-shell] C:\hdl_projects_intel\niosv\SW_APP> niosv-app -b=../SW -a=. -s=.
2022.08.27.12:26:45 Error: No source files specified.

[niosv-shell] C:\hdl_projects_intel\niosv\SW_APP> niosv-app -b=../SW -a=. -s=.
2022.08.27.12:27:41 Info: Source file "..\main.cpp" added from directory ".".
2022.08.27.12:27:41 Info: ".\CMakeLists.txt" was generated.

[niosv-shell] C:\hdl_projects_intel\niosv\SW_APP>
```

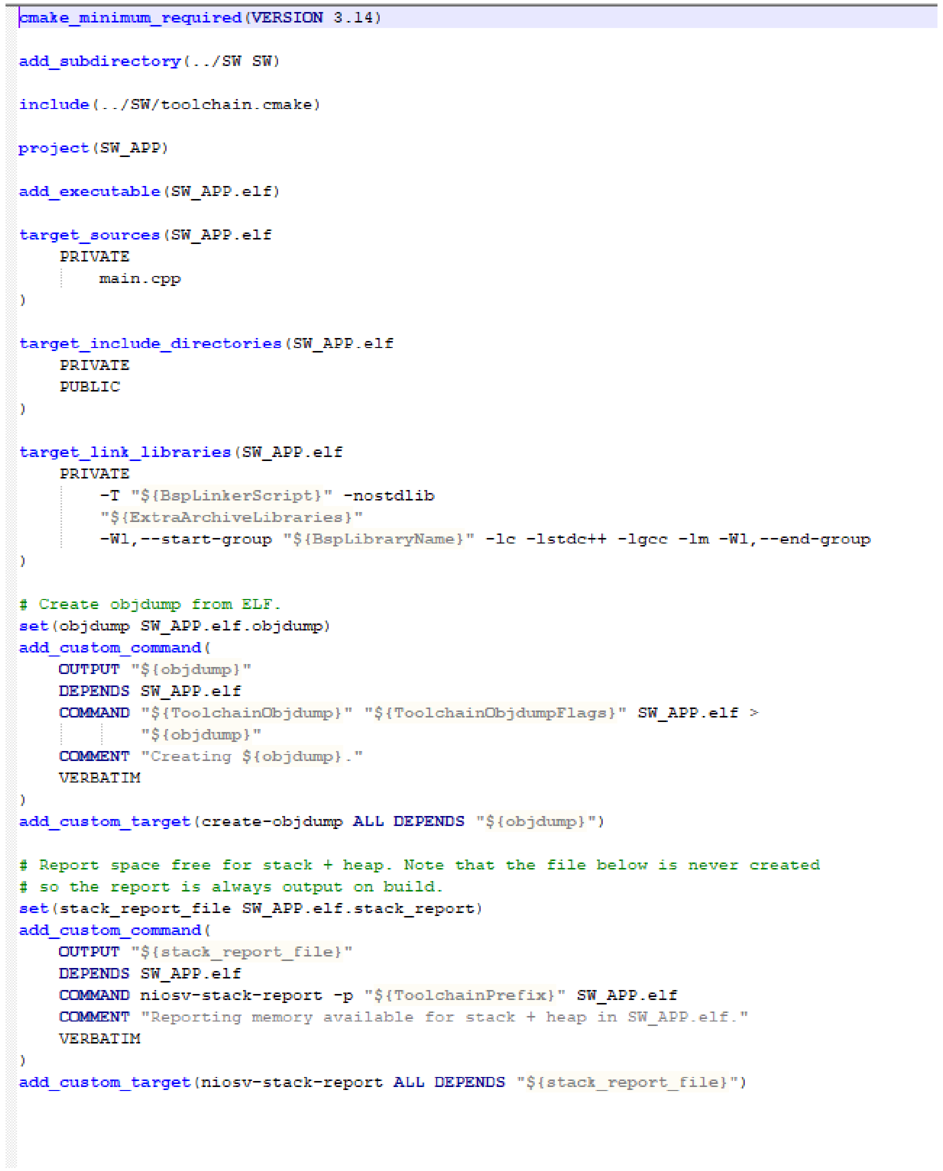
Lab Two


OS (C:) > hdl_projects_intel > niosv > SW_APP

Name	Date modified	Type	Size
CMakeLists.txt	27/08/2022 12:27	TXT File	2 KB
main.cpp	05/08/2022 15:41	CPP File	1 KB

- This will create the Cmake file which is used to compile the application and BSP just created into a ELF for download

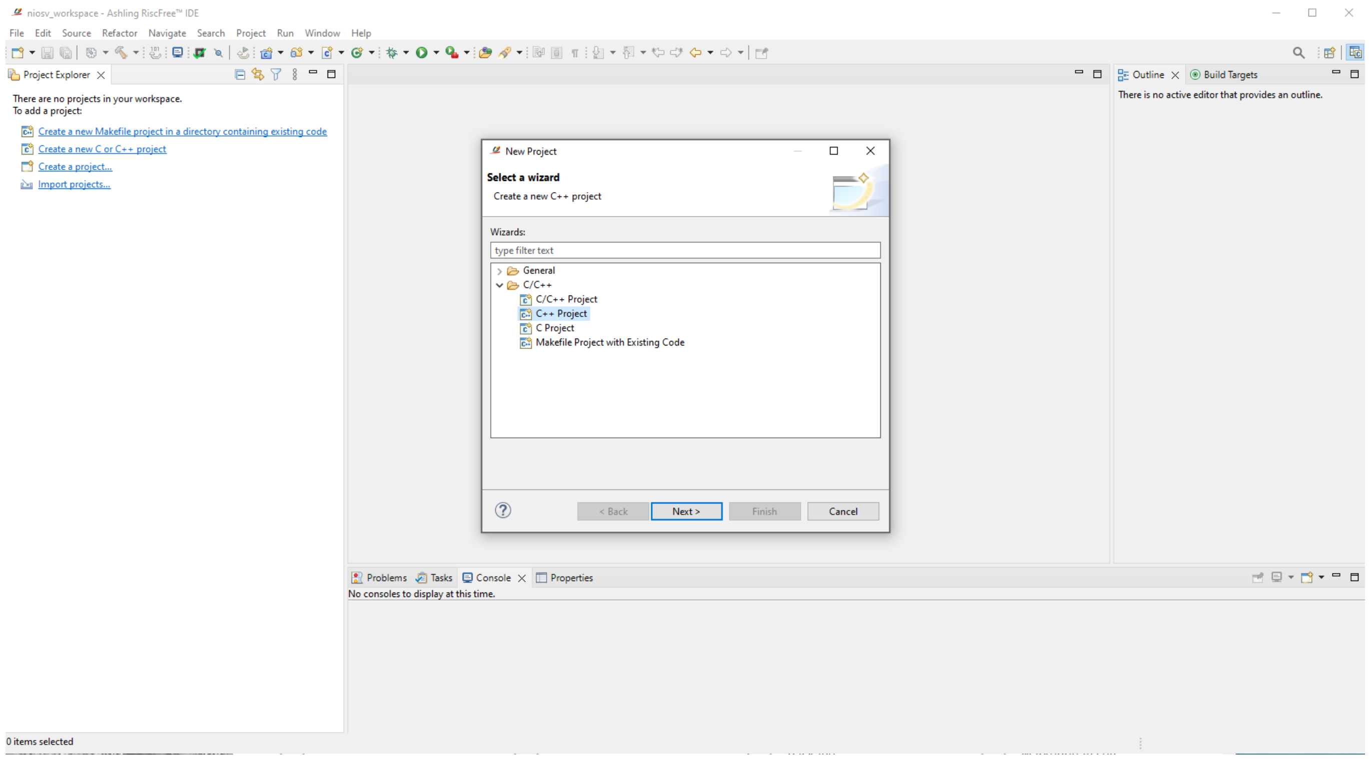
Lab Two



```cmake
cmake_minimum_required(VERSION 3.14)

add_subdirectory(../SW SW)

include(../SW/toolchain.cmake)

project(SW_APP)

add_executable(SW_APP.elf)

target_sources(SW_APP.elf
    PRIVATE
        main.cpp
)

target_include_directories(SW_APP.elf
    PRIVATE
    PUBLIC
)

target_link_libraries(SW_APP.elf
    PRIVATE
        -T "${BspLinkerScript}" -nostdlib
        "${ExtraArchiveLibraries}"
        -Wl,--start-group "${BspLibraryName}" -lc -lstdc++ -lgcc -lm -Wl,--end-group
)

# Create objdump from ELF.
set(objdump SW_APP.elf.objdump)
add_custom_command(
    OUTPUT "${objdump}"
    DEPENDS SW_APP.elf
    COMMAND "${ToolchainObjdump}" "${ToolchainObjdumpFlags}" SW_APP.elf >
            "${objdump}"
    COMMENT "Creating ${objdump}."
    VERBATIM
)
add_custom_target(create-objdump ALL DEPENDS "${objdump}")

# Report space free for stack + heap. Note that the file below is never created
# so the report is always output on build.
set(stack_report_file SW_APP.elf.stack_report)
add_custom_command(
    OUTPUT "${stack_report_file}"
    DEPENDS SW_APP.elf
    COMMAND niosv-stack-report -p "${ToolchainPrefix}" SW_APP.elf
    COMMENT "Reporting memory available for stack + heap in SW_APP.elf."
    VERBATIM
)
add_custom_target(niosv-stack-report ALL DEPENDS "${stack_report_file}")
```

- Examine the CMAKE file and observe the structure of the file

Lab Two



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

- Create a new Makefile project in a directory containing existing code
- Create a new C or C++ project
- Create a project...
- Import projects...

There is no active editor that provides an outline.

Select a wizard

Create a new C++ project

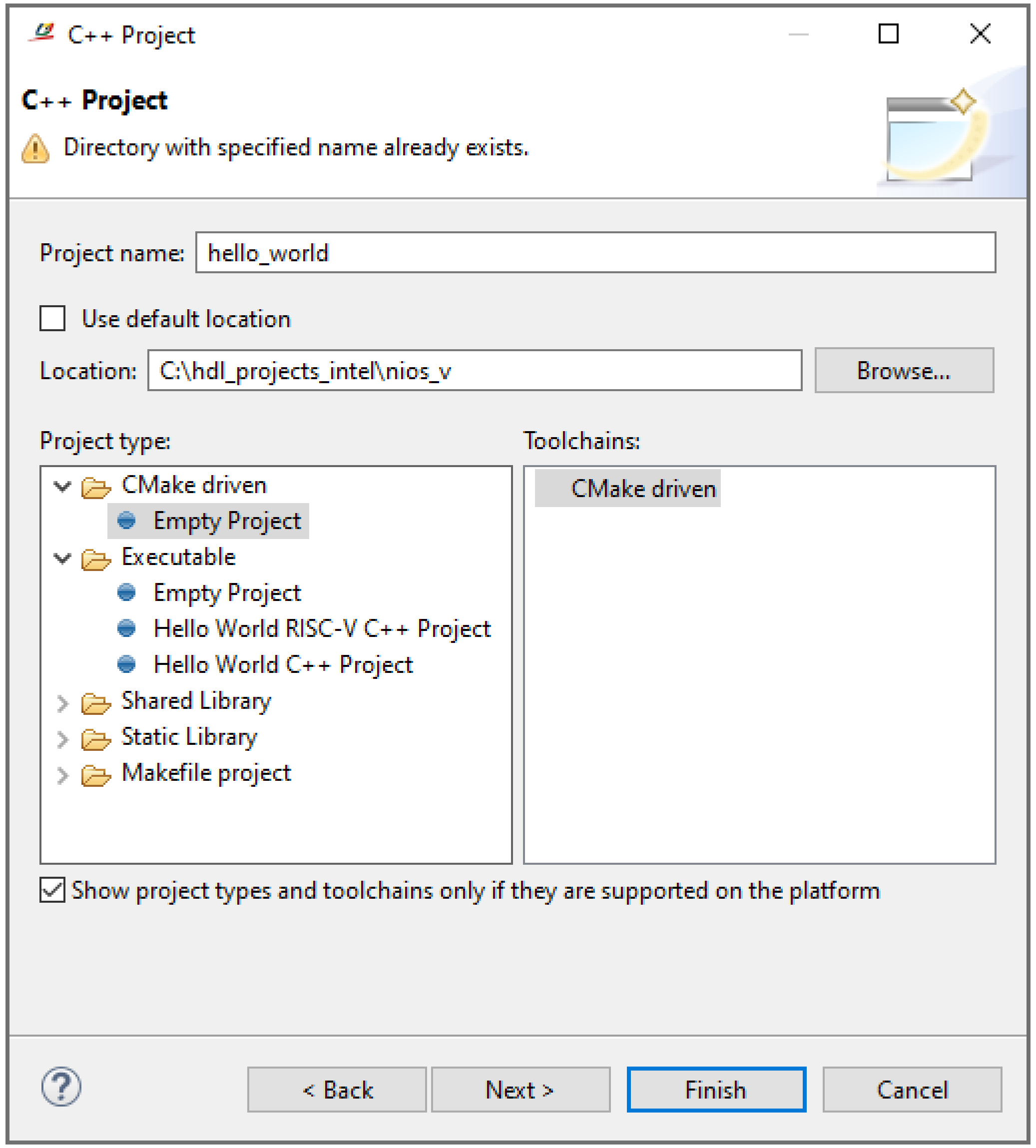
No consoles to display at this time.

0 items selected

- From the Nios Shell start the Ashling RiscFree IDE by typing
  - Riscfree
- Select the project workspace
- Select Create Project in the dialog which appears select C++ project

ADIUVO ENGINEERING AND TRAINING, LTD.

Lab Two


C++ Project

C++ Project

⚠ Directory with specified name already exists.

Project name: hello_world

☐ Use default location

Location: C:\hdl_projects_intel\nios_v

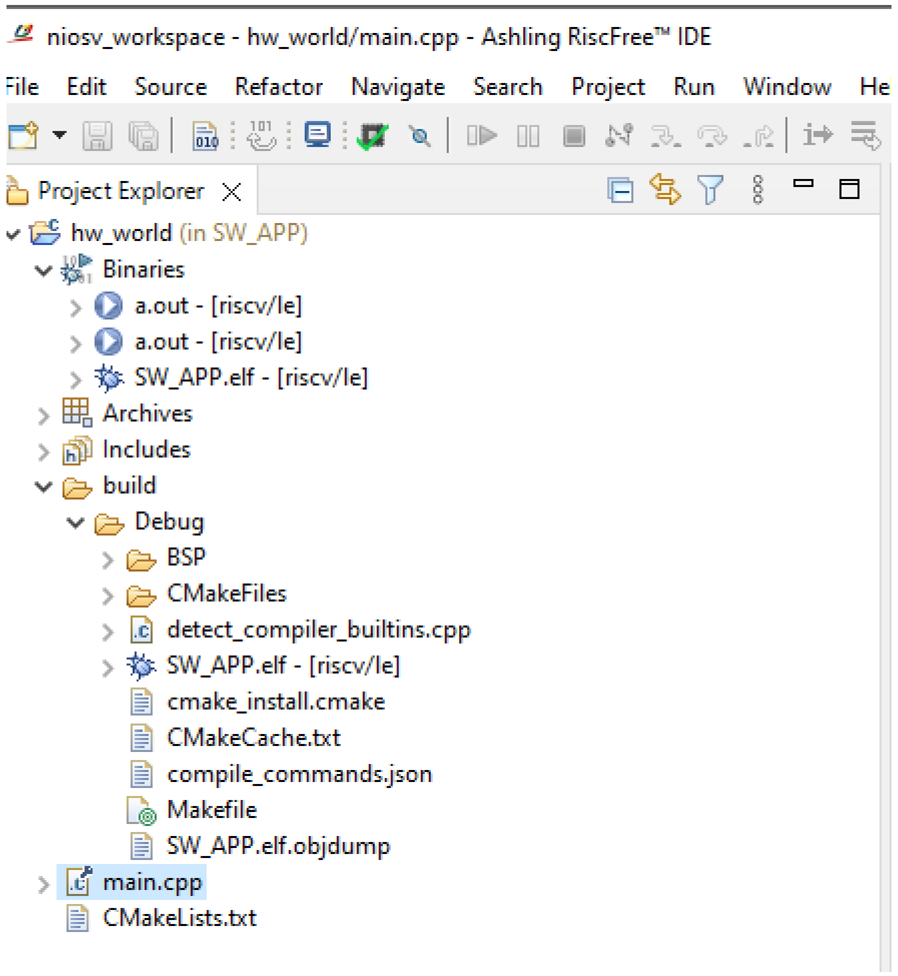
Project type:
- CMake driven
  - Empty Project
- Executable
  - Empty Project
  - Hello World RISC-V C++ Project
  - Hello World C++ Project
- Shared Library
- Static Library
- Makefile project

Toolchains:
- CMake driven

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

- Enter a project name
- Point the project location to the just created C Make files
- Select Cmake Driven project
- Click Finish

Lab Two

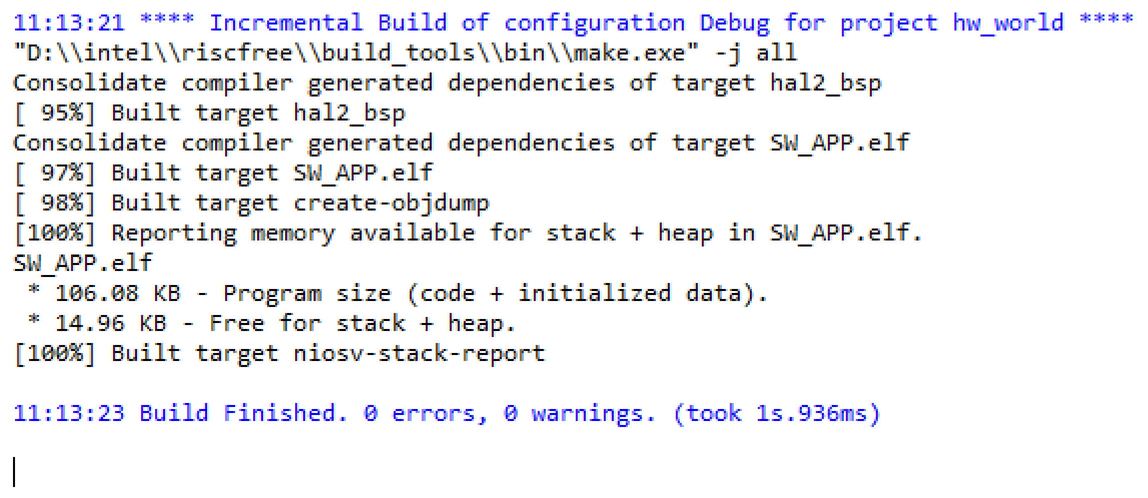

niosv_workspace - hw_world/main.cpp - Ashling RiscFree™ IDE

File Edit Source Refactor Navigate Search Project Run Window He
Project Explorer

```
hw_world (in SW_APP)
  Binaries
    a.out - [riscv/le]
    a.out - [riscv/le]
    SW_APP.elf - [riscv/le]
  Archives
  Includes
  build
    Debug
      BSP
      CMakeFiles
      detect_compiler_builtins.cpp
      SW_APP.elf - [riscv/le]
      cmake_install.cmake
      CMakeCache.txt
      compile_commands.json
      Makefile
      SW_APP.elf.objdump
  main.cpp
  CMakeLists.txt
```

- This will open the project

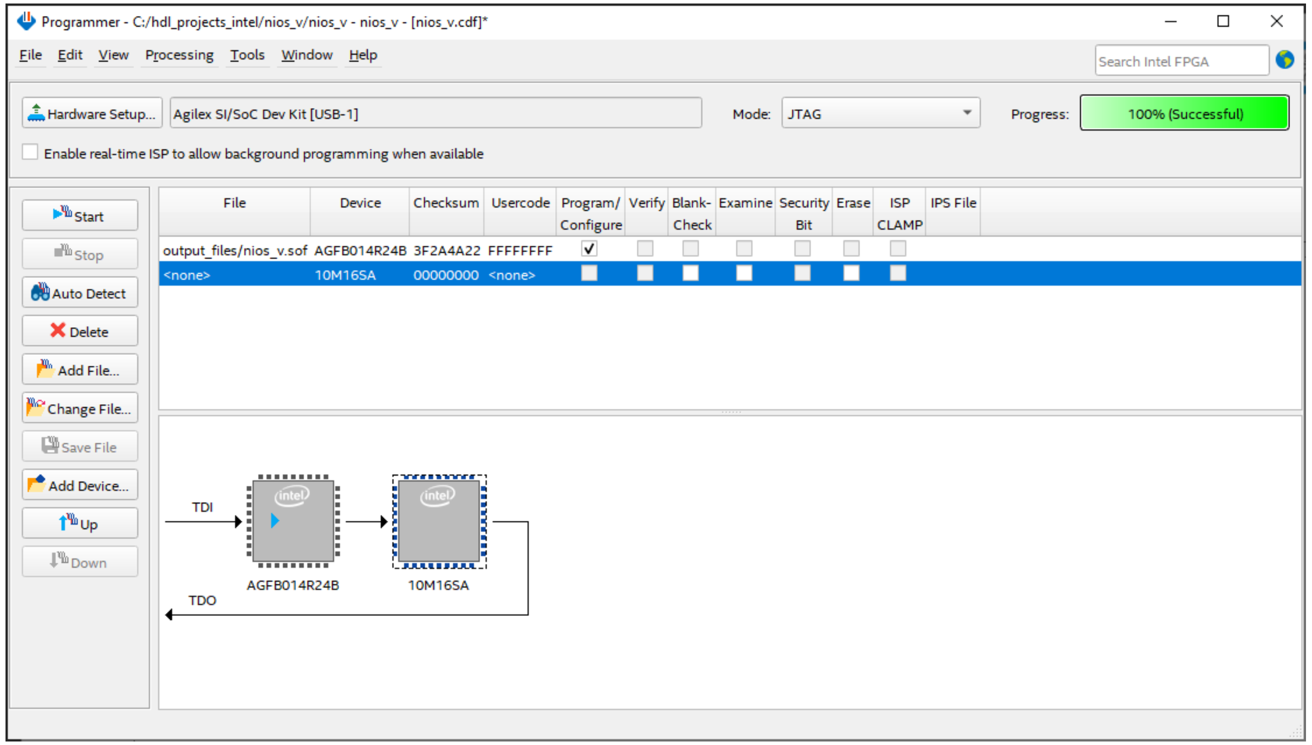
Lab Two



```
11:13:21 **** Incremental Build of configuration Debug for project hw_world ****
"D:\\intel\\riscfree\\build_tools\\bin\\make.exe" -j all
Consolidate compiler generated dependencies of target hal2_bsp
[ 95%] Built target hal2_bsp
Consolidate compiler generated dependencies of target SW_APP.elf
[ 97%] Built target SW_APP.elf
[ 98%] Built target create-objdump
[100%] Reporting memory available for stack + heap in SW_APP.elf.
SW_APP.elf
 * 106.08 KB - Program size (code + initialized data).
 * 14.96 KB - Free for stack + heap.
[100%] Built target niosv-stack-report

11:13:23 Build Finished. 0 errors, 0 warnings. (took 1s.936ms)
```

- Click the hammer to build the project
- Note the program size and stack and heap size will be reported

Lab Two



Programmer - C:/hdl_projects_intel/nios_v/nios_v - nios_v - [nios_v.cdf]*

Hardware Setup... Agilex SI/SoC Dev Kit [USB-1]

Mode: JTAG

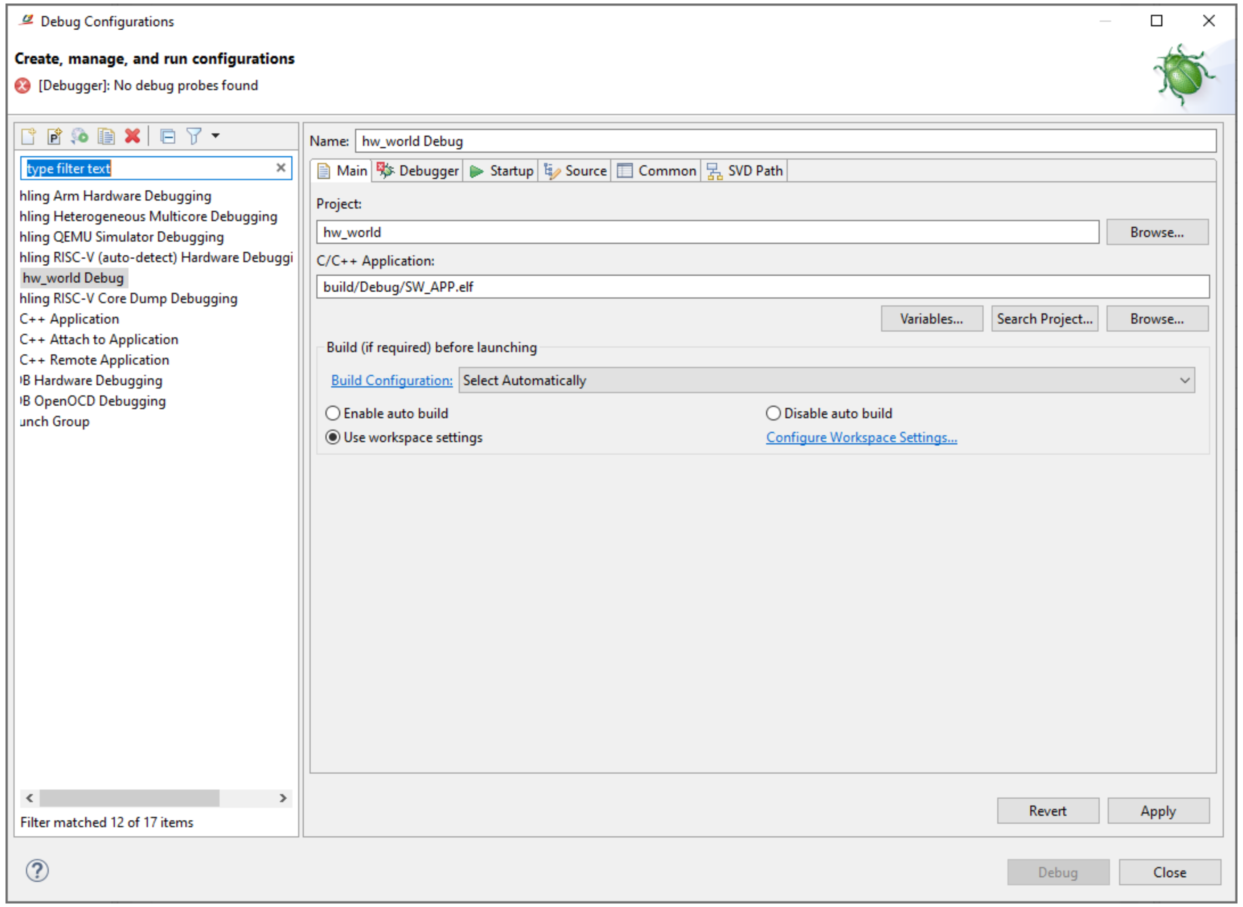
Progress: 100% (Successful)

Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase	ISP CLAMP	IPS File
output_files/nios_v.sof	AGFB014R24B	3F2A4A22	FFFFFFFF	✓							
<none>	10M16SA	00000000	<none>								

- Use the Quartus Programmer to download the SOF programming file.

Lab Two



Debug Configurations

Create, manage, and run configurations

[Debugger]: No debug probes found

type filter text

- hling Arm Hardware Debugging
- hling Heterogeneous Multicore Debugging
- hling QEMU Simulator Debugging
- hling RISC-V (auto-detect) Hardware Debuggi
- hw_world Debug
- hling RISC-V Core Dump Debugging
- C++ Application
- C++ Attach to Application
- C++ Remote Application
- B Hardware Debugging
- B OpenOCD Debugging
- unch Group

Filter matched 12 of 17 items

Name: hw_world Debug

Main | Debugger | Startup | Source | Common | SVD Path

Project: hw_world

C/C++ Application: build/Debug/SW_APP.elf

Variables... Search Project... Browse...

Build (if required) before launching

Build Configuration: Select Automatically

( ) Enable auto build ( ) Disable auto build

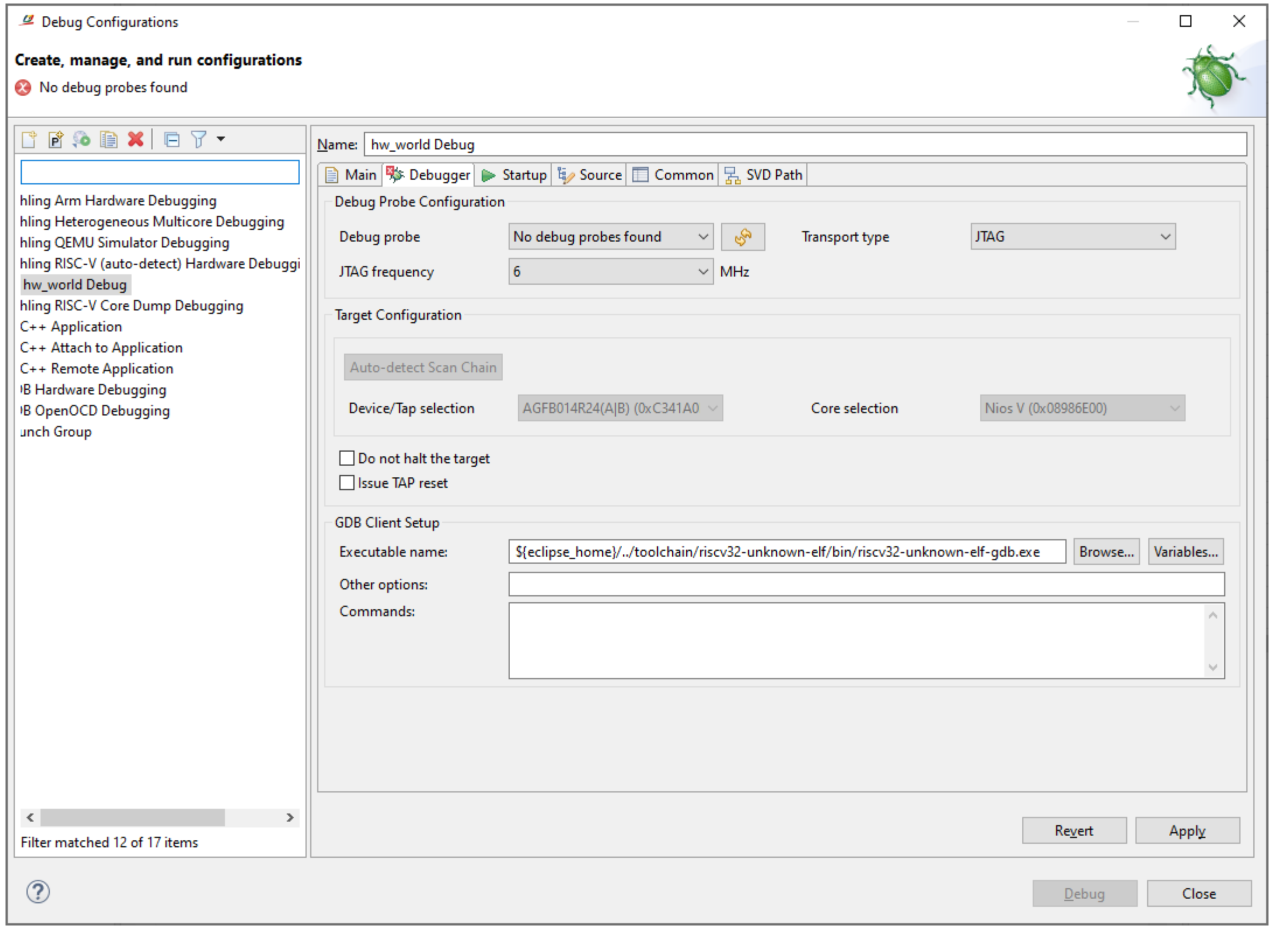
(●) Use workspace settings Configure Workspace Settings...

Revert Apply

Debug Close

- Open the debug configuration and create a new Ashling RISC-V (auto detect) Hardware Debugging
- Select the ELF just built

Lab Two



Debug Configurations

Create, manage, and run configurations

No debug probes found

hling Arm Hardware Debugging
hling Heterogeneous Multicore Debugging
hling QEMU Simulator Debugging
hling RISC-V (auto-detect) Hardware Debuggi
hw_world Debug
hling RISC-V Core Dump Debugging
C++ Application
C++ Attach to Application
C++ Remote Application
B Hardware Debugging
B OpenOCD Debugging
unch Group

Filter matched 12 of 17 items

Name: hw_world Debug

Debug Probe Configuration

Debug probe	No debug probes found	Transport type	JTAG
JTAG frequency	6 MHz		

Target Configuration

Device/Tap selection	AGFB014R24(A|B) (0xC341A0	Core selection	Nios V (0x08986E00)

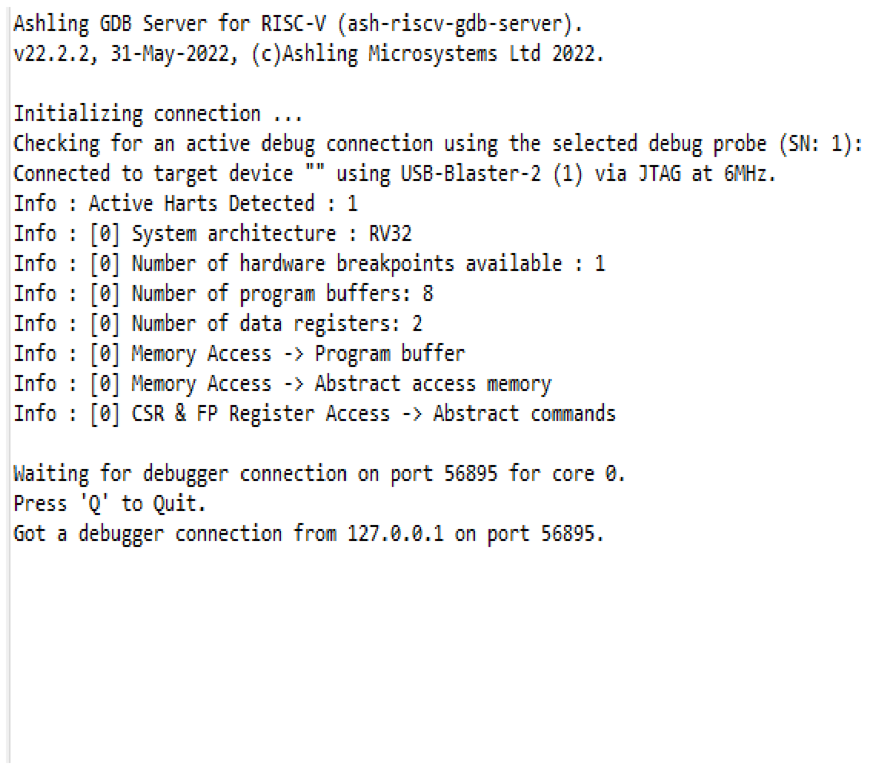
☐ Do not halt the target
☐ Issue TAP reset

GDB Client Setup

Executable name: ${eclipse_home}/../toolchain/riscv32-unknown-elf/bin/riscv32-unknown-elf-gdb.exe
Other options:
Commands:

- On the debug tab select the JTAG frequency of 6MHz
- With the target powered up auto detect the scan change and select the Nios V core
- Leave all other settings unchanged

Lab Two

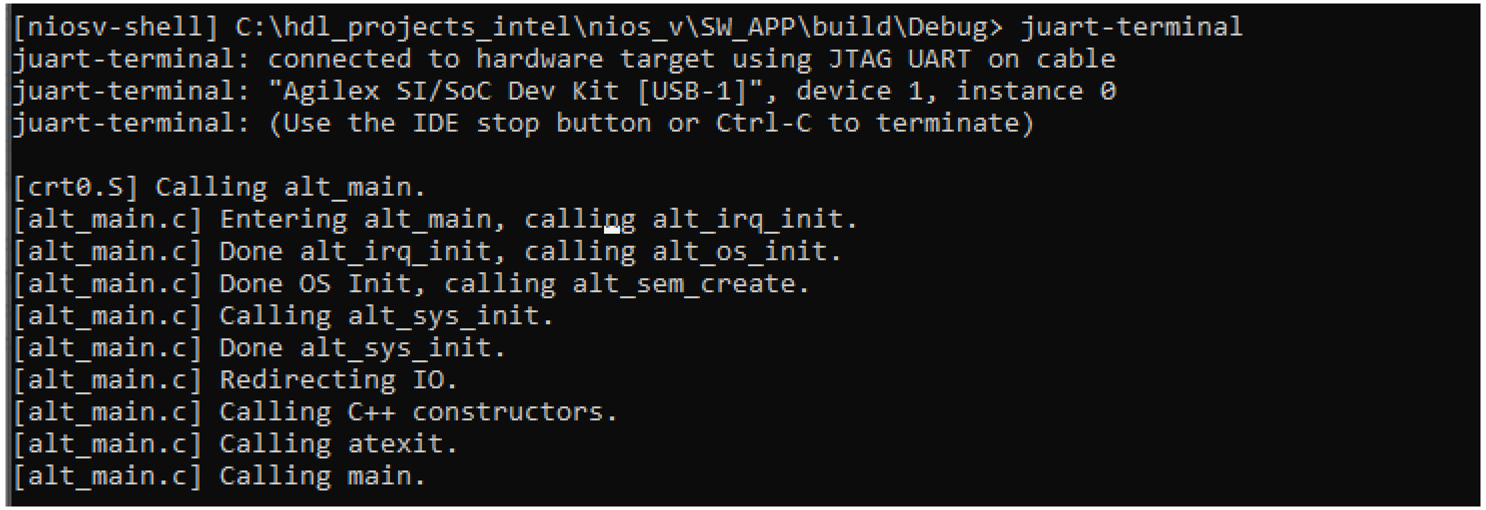


```
Ashling GDB Server for RISC-V (ash-riscv-gdb-server).
v22.2.2, 31-May-2022, (c)Ashling Microsystems Ltd 2022.

Initializing connection ...
Checking for an active debug connection using the selected debug probe (SN: 1):
Connected to target device "" using USB-Blaster-2 (1) via JTAG at 6MHz.
Info : Active Harts Detected : 1
Info : [0] System architecture : RV32
Info : [0] Number of hardware breakpoints available : 1
Info : [0] Number of program buffers: 8
Info : [0] Number of data registers: 2
Info : [0] Memory Access -> Program buffer
Info : [0] Memory Access -> Abstract access memory
Info : [0] CSR & FP Register Access -> Abstract commands

Waiting for debugger connection on port 56895 for core 0.
Press 'Q' to Quit.
Got a debugger connection from 127.0.0.1 on port 56895.
```

- The terminal should show the JTAG connection and application software downloading

Lab Two

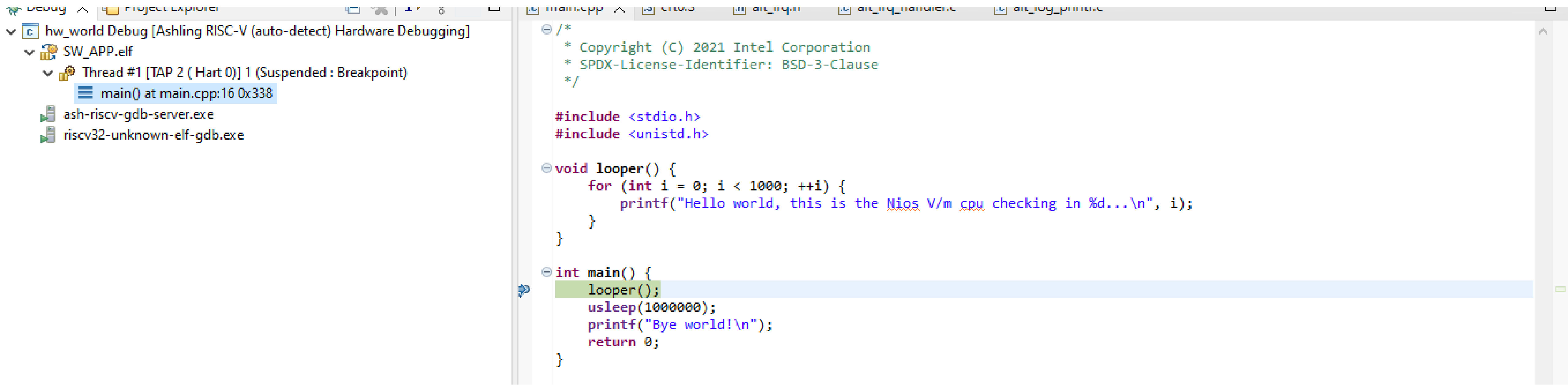
- In the NIOSV Shell open the JTAG UART
  - juart-terminal



```
[niosv-shell] C:\hdl_projects_intel\nios_v\SW_APP\build\Debug> juart-terminal
juart-terminal: connected to hardware target using JTAG UART on cable
juart-terminal: "Agilex SI/SoC Dev Kit [USB-1]", device 1, instance 0
juart-terminal: (Use the IDE stop button or Ctrl-C to terminate)

[crt0.S] Calling alt_main.
[alt_main.c] Entering alt_main, calling alt_irq_init.
[alt_main.c] Done alt_irq_init, calling alt_os_init.
[alt_main.c] Done OS Init, calling alt_sem_create.
[alt_main.c] Calling alt_sys_init.
[alt_main.c] Done alt_sys_init.
[alt_main.c] Redirecting IO.
[alt_main.c] Calling C++ constructors.
[alt_main.c] Calling atexit.
[alt_main.c] Calling main.
```

Lab Two

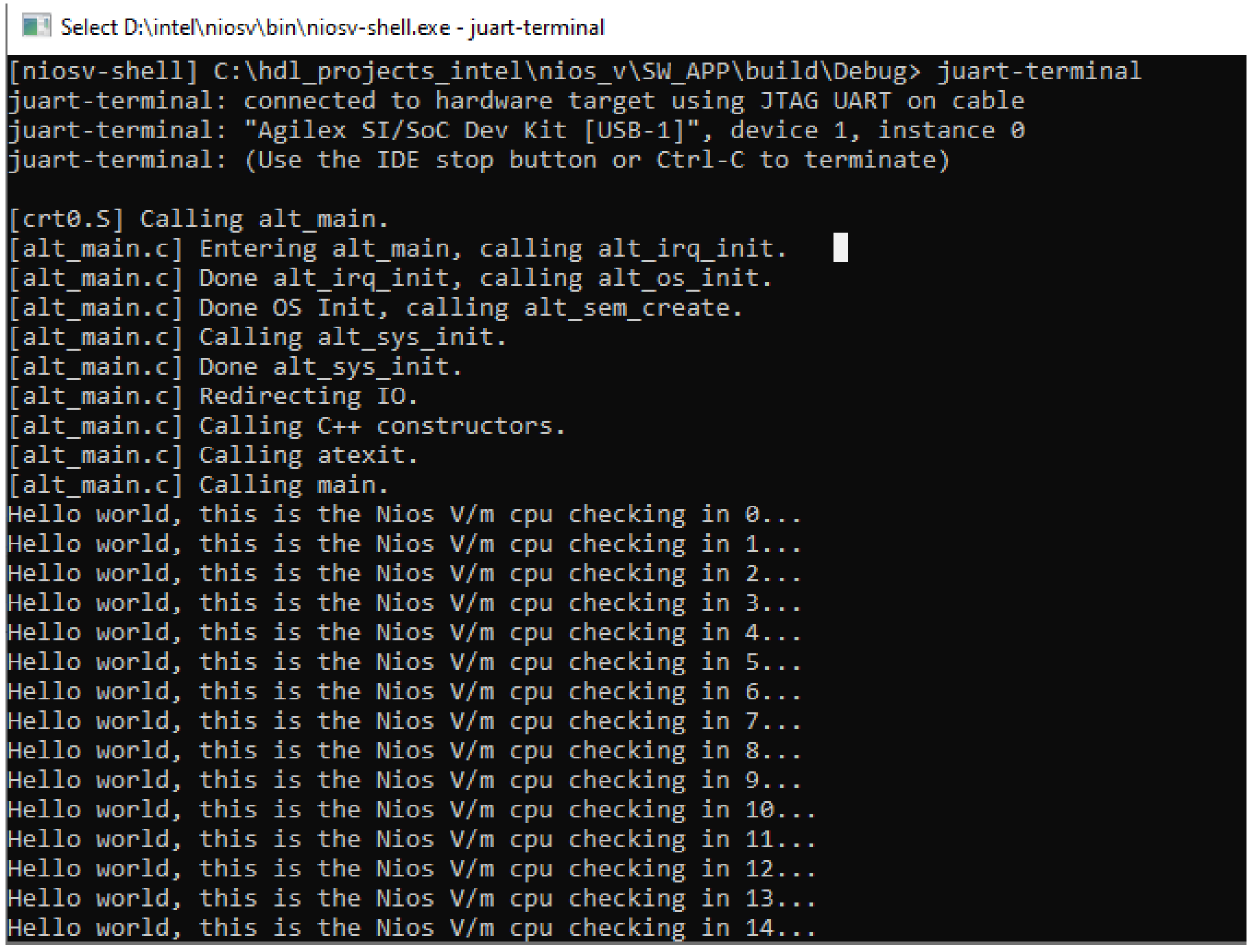
- The application will be downloaded and paused
- Run the program



hw_world Debug [Ashling RISC-V (auto-detect) Hardware Debugging]
SW_APP.elf
Thread #1 [TAP 2 (Hart 0)] 1 (Suspended : Breakpoint)
main() at main.cpp:16 0x338
ash-riscv-gdb-server.exe
riscv32-unknown-elf-gdb.exe

```
/*
 * Copyright (C) 2021 Intel Corporation
 * SPDX-License-Identifier: BSD-3-Clause
 */

#include <stdio.h>
#include <unistd.h>

void looper() {
    for (int i = 0; i < 1000; ++i) {
        printf("Hello world, this is the Nios V/m cpu checking in %d...\n", i);
    }
}

int main() {
    looper();
    usleep(1000000);
    printf("Bye world!\n");
    return 0;
}
```

Lab Two



Select D:\intel\niosv\bin\niosv-shell.exe - juart-terminal

```
[niosv-shell] C:\hdl_projects_intel\nios_v\SW_APP\build\Debug> juart-terminal
juart-terminal: connected to hardware target using JTAG UART on cable
juart-terminal: "Agilex SI/SoC Dev Kit [USB-1]", device 1, instance 0
juart-terminal: (Use the IDE stop button or Ctrl-C to terminate)

[crt0.S] Calling alt_main.
[alt_main.c] Entering alt_main, calling alt_irq_init.
[alt_main.c] Done alt_irq_init, calling alt_os_init.
[alt_main.c] Done OS Init, calling alt_sem_create.
[alt_main.c] Calling alt_sys_init.
[alt_main.c] Done alt_sys_init.
[alt_main.c] Redirecting IO.
[alt_main.c] Calling C++ constructors.
[alt_main.c] Calling atexit.
[alt_main.c] Calling main.
Hello world, this is the Nios V/m cpu checking in 0...
Hello world, this is the Nios V/m cpu checking in 1...
Hello world, this is the Nios V/m cpu checking in 2...
Hello world, this is the Nios V/m cpu checking in 3...
Hello world, this is the Nios V/m cpu checking in 4...
Hello world, this is the Nios V/m cpu checking in 5...
Hello world, this is the Nios V/m cpu checking in 6...
Hello world, this is the Nios V/m cpu checking in 7...
Hello world, this is the Nios V/m cpu checking in 8...
Hello world, this is the Nios V/m cpu checking in 9...
Hello world, this is the Nios V/m cpu checking in 10...
Hello world, this is the Nios V/m cpu checking in 11...
Hello world, this is the Nios V/m cpu checking in 12...
Hello world, this is the Nios V/m cpu checking in 13...
Hello world, this is the Nios V/m cpu checking in 14...
```

- Observe the JTAG UART Terminal

Wrap Up

- Introduced the Tools used & Development Flow
- Demonstrated how to build the hardware
- Demonstrated how to create a software app
- Shown the ASW application running on the HW
- Questions?

ADIUVO
ENGINEERING AND TRAINING, LTD.

ADIUVO

ENGINEERING AND TRAINING, LTD.

www.adiuvoengineering.com

adam@adiuvoengineering.com