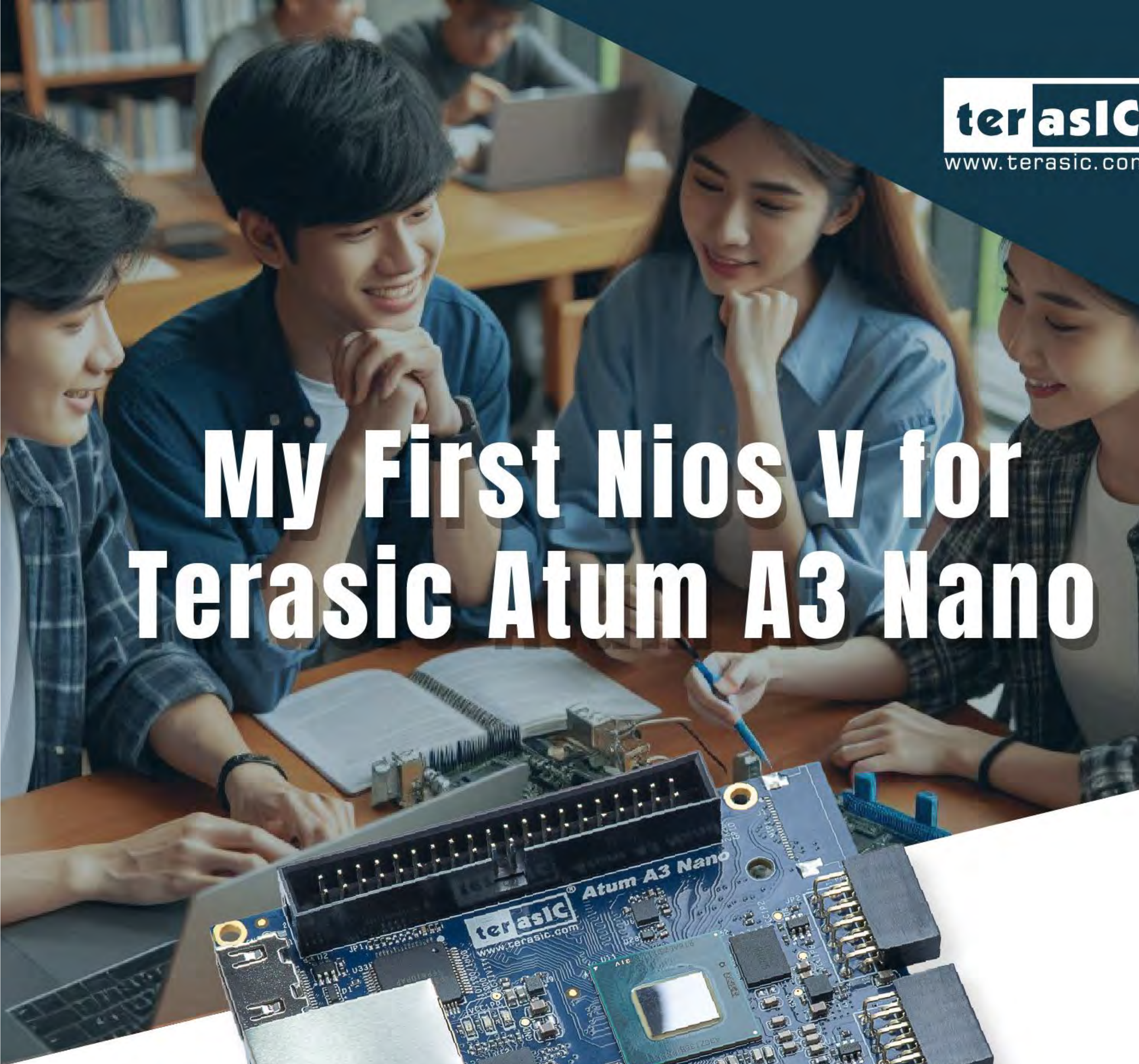


My First Nios V for Terasic Atum A3 Nano

Powering next-level performance for digital logic and embedded applications !

CONTENTS

Chapter 1

Overview

This document provides an introductory roadmap for developing a Nios® V processor system. It walks you through creating and running a basic Hello World program and controlling the LED blink feature on an FPGA board. The example setup uses the Terasic Atum A3 Nano development board, with Quartus® Prime software and the Ashling* RiscFree* IDE for Altera FPGAs as the primary development tools.

1.1 Hardware and Software Requirements

Hardware Requirements

- Terasic Atum A3 Nano development board

Software Requirements

- Quartus Prime Pro Edition (This document uses Quartus Prime Pro Edition version 25.1)
- Ashling RiscFree IDE for Altera FPGAs

Chapter 2

Hardware Design

This chapter will guide you through starting the hardware design process by adding the Nios V processor and its peripherals into Platform Designer. After setting up the system assignments and constraints, you will complete the hardware design by successfully compiling the project.

2.1 Creation of Hardware Design

This section describes the flow of how to create a hardware system including Platform Designer feature.

2.1.1 Creating a New Project

1. Launch Quartus Prime then select **File**->**New Project Wizard**, start to create a new project. See Figure 2-1.

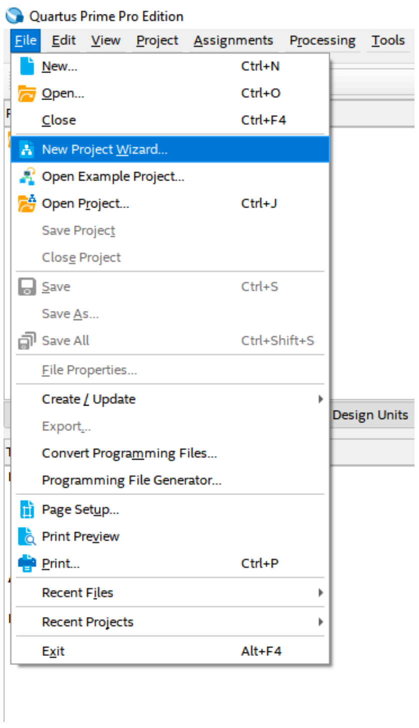
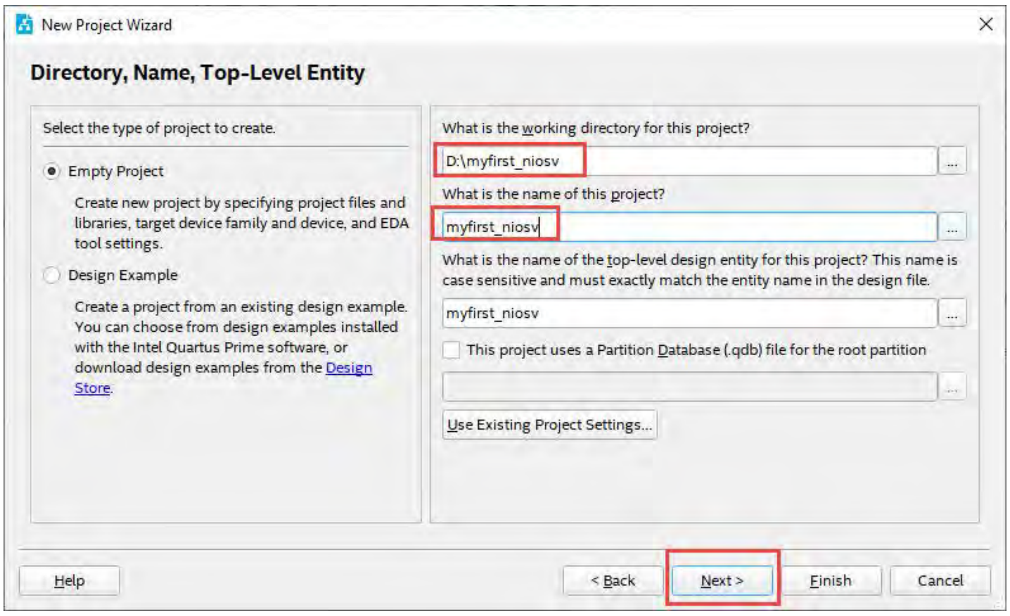


Figure 2-1 Start to Create a New Project

2. Choose a working directory for this project, type project name and top-level entity name as shown in Figure 2-2. Click **Next** to next window.



New Project Wizard

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?

D:\myfirst_niosv

What is the name of this project?

myfirst_niosv

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.

myfirst_niosv

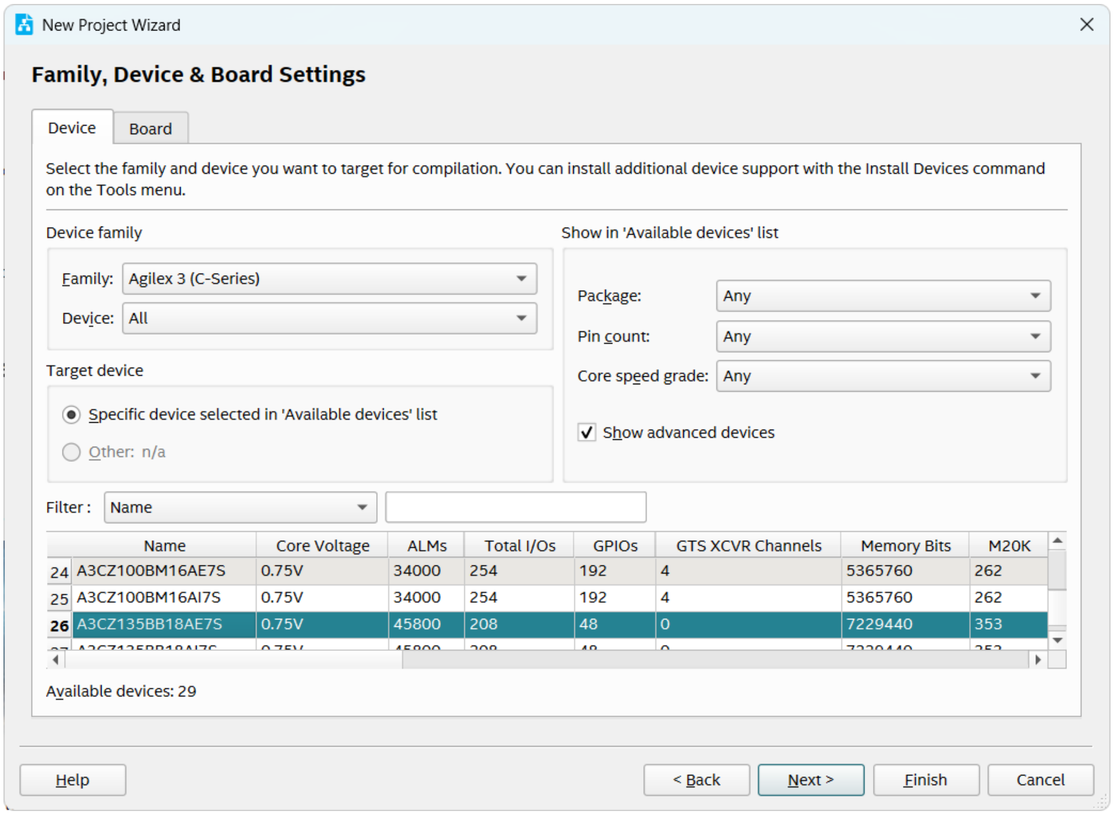
[ ] This project uses a Partition Database (.qdb) file for the root partition

Figure 2-2 Input the working directory, the name of project, top-level design entity

3. We choose device family and device settings. You should choose settings the same as the **Figure 2-3**.

- FPGA Family :**Agilex 3(C-Series)**
- FPGA Name : **A3CZ135BB18AE7S**

Then click **Finish** to finish new project. **Figure 2-4** show a new complete project.



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 3 (C-Series)

Device: All

Target device

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

( ) Other: n/a

Show in 'Available devices' list

Package: Any

Pin count: Any

Core speed grade: Any

[✓] Show advanced devices

Filter : Name

	Name	Core Voltage	ALMs	Total I/Os	GPIOs	GTS XCVR Channels	Memory Bits	M20K
24	A3CZ100BM16AE7S	0.75V	34000	254	192	4	5365760	262
25	A3CZ100BM16AI7S	0.75V	34000	254	192	4	5365760	262
26	A3CZ135BB18AE7S	0.75V	45800	208	48	0	7229440	353

Available devices: 29

Figure 2-3 New Project Wizard: Family & Device Settings

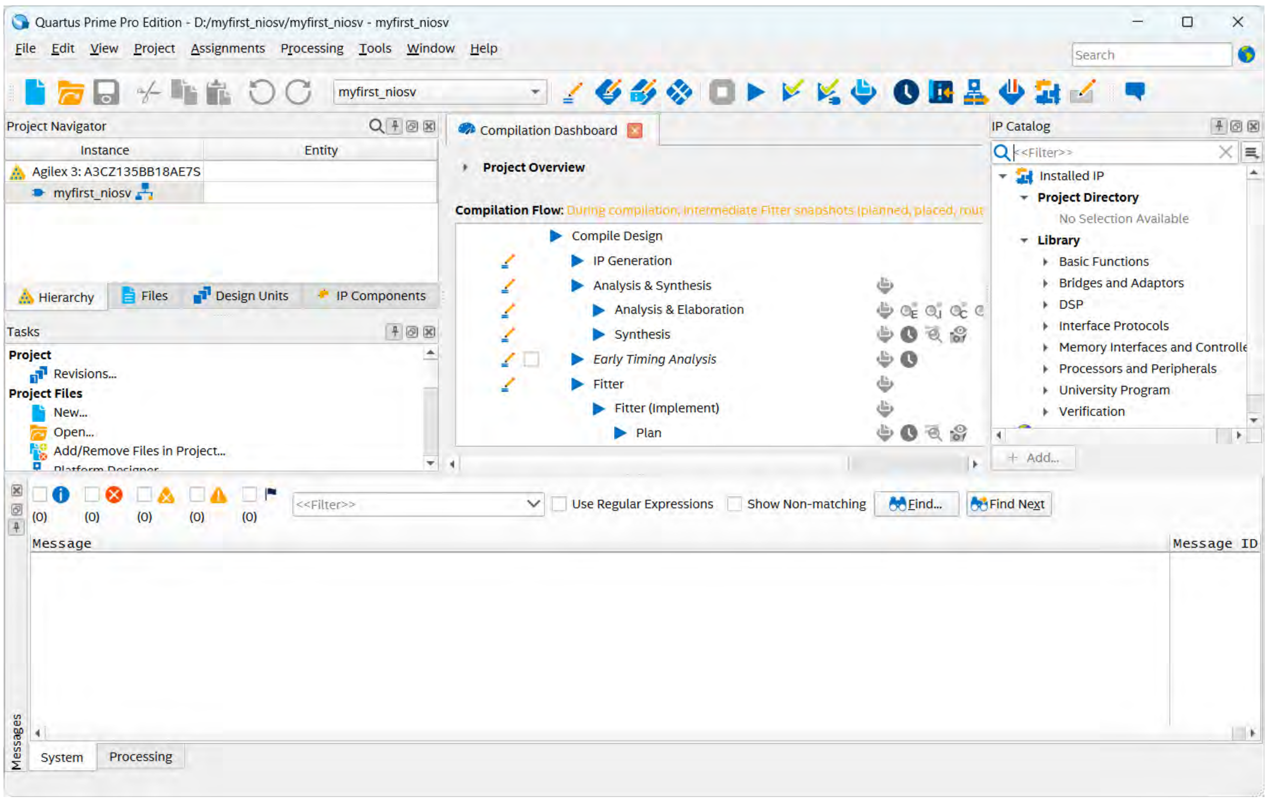
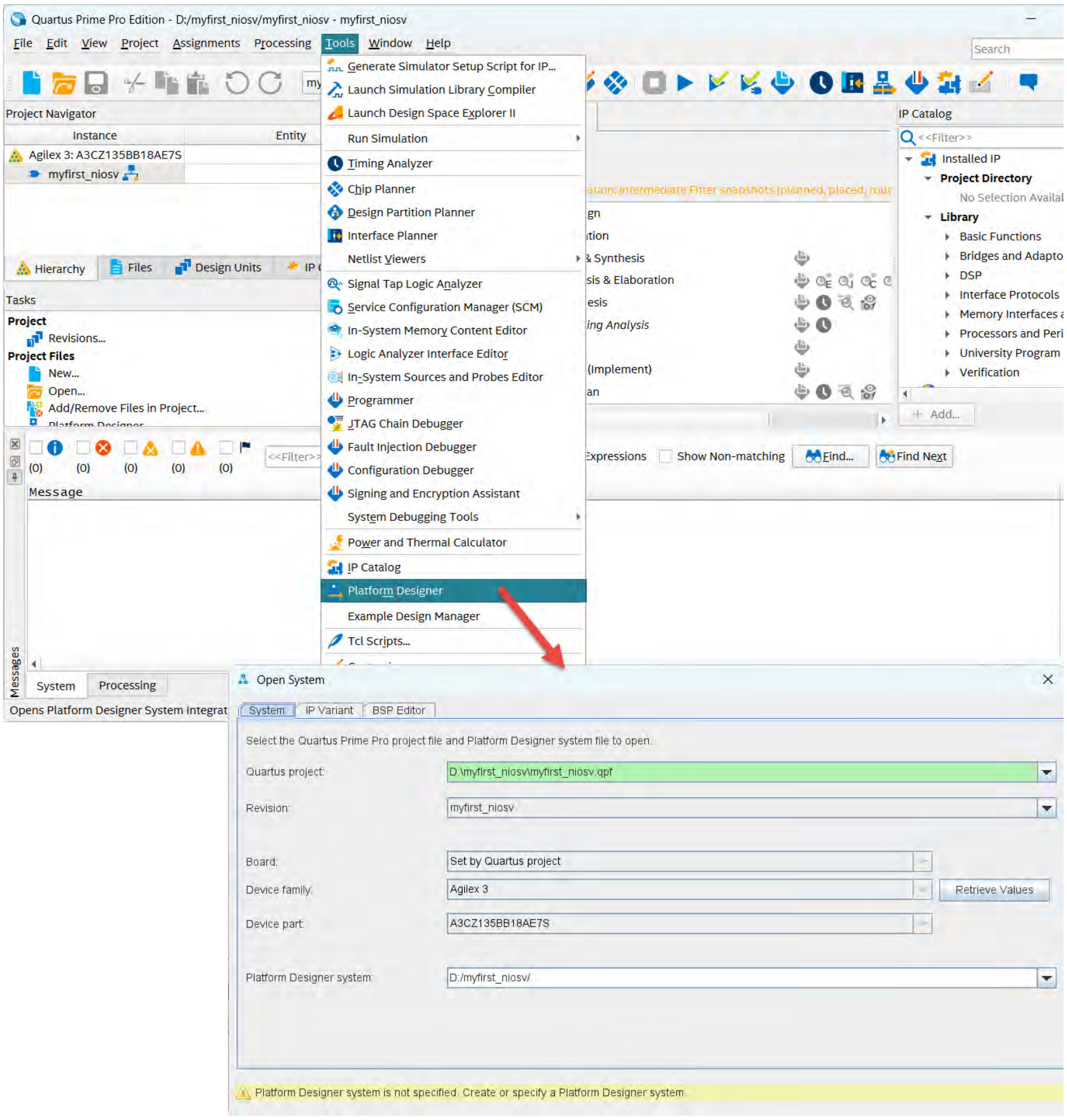


Figure 2-4 A New Complete Project

2.1.2 Creating a Platform Designer System

1. Choose **Tools** > **Platform Designer** to open new Platform Designer system wizard. See Figure 2-5.


Quartus Prime Pro Edition - D:/myfirst_niosv/myfirst_niosv - myfirst_niosv
File Edit View Project Assignments Processing Tools Window Help
Generate Simulator Setup Script for IP...
Launch Simulation Library Compiler
Launch Design Space Explorer II
Run Simulation
Timing Analyzer
Chip Planner
Design Partition Planner
Interface Planner
Netlist Viewers
Signal Tap Logic Analyzer
Service Configuration Manager (SCM)
In-System Memory Content Editor
Logic Analyzer Interface Editor
In-System Sources and Probes Editor
Programmer
JTAG Chain Debugger
Fault Injection Debugger
Configuration Debugger
Signing and Encryption Assistant
System Debugging Tools
Power and Thermal Calculator
IP Catalog
Platform Designer
Example Design Manager
Tcl Scripts...
Project Navigator
Agilex 3: A3CZ135BB18AE7S
myfirst_niosv
IP Catalog
Installed IP
Project Directory
Library
Basic Functions
DSP
Interface Protocols
Verification
Opens Platform Designer System Integrat
Open System
System
IP Variant
BSP Editor

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

Quartus project: D:\myfirst_niosv\myfirst_niosv.qpf

Revision: myfirst_niosv

Board: Set by Quartus project

Device family: Agilex 3

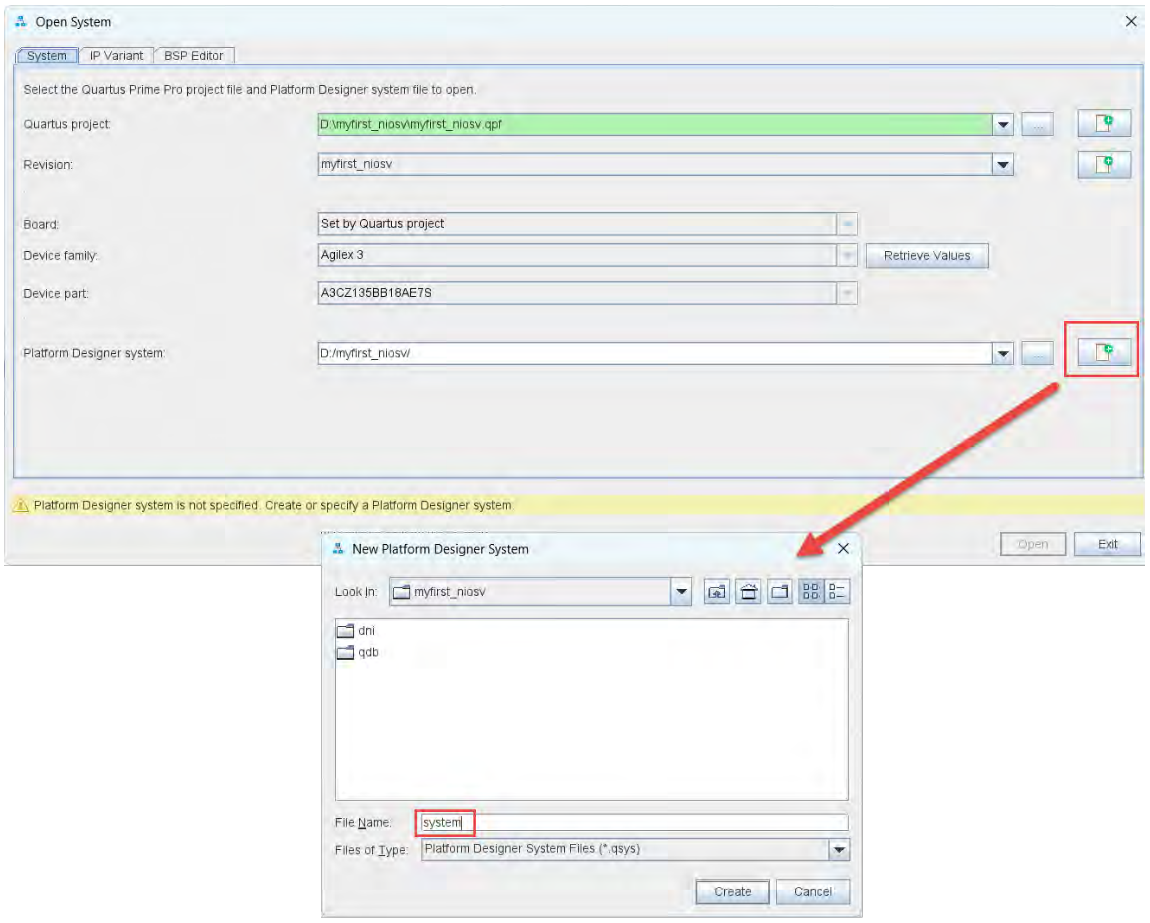
Device part: A3CZ135BB18AE7S

Platform Designer system: D:/myfirst_niosv/

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

Figure 2-5 Platform Designer Menu

2. Click the Create New Platform Designer System icon and name the new project system "**system**" (refer to Figure 2-6).


Open System

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

Quartus project: D:\myfirst_niosv\myfirst_niosv.qpf

Revision: myfirst_niosv

Board: Set by Quartus project

Device family: Agilex 3

Device part: A3CZ135BB18AE7S

Platform Designer system: D:/myfirst_niosv/

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

New Platform Designer System

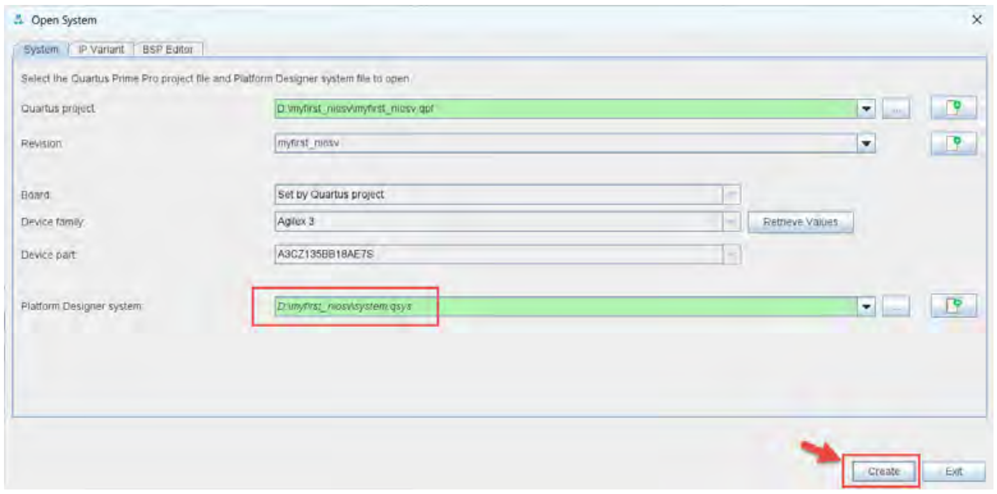
Look In: myfirst_niosv

File Name: system

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

Figure 2-6 Create New System [1]

Click Create and your will see a window as shown in Figure 2-7. Then a Platform Designer window will appear.


Open System

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

Quartus project: D:\myfirst_niosv\myfirst_niosv.qpf

Revision: myfirst_niosv

Board: Set by Quartus project

Device family: Agilex 3

Device part: A3CZ135BB18AE7S

Platform Designer system: D:\myfirst_niosv\system.qsys

Figure 2-7 Create New System [2]

■ Adding Components

1. Now, we will first add a reset component into the system.

Go to IP tab and choose **Library** > **Basic Functions** > **Configuration and Programming** > **Agilex Reset Release** to open wizard of adding reset component (See Figure 2-8)

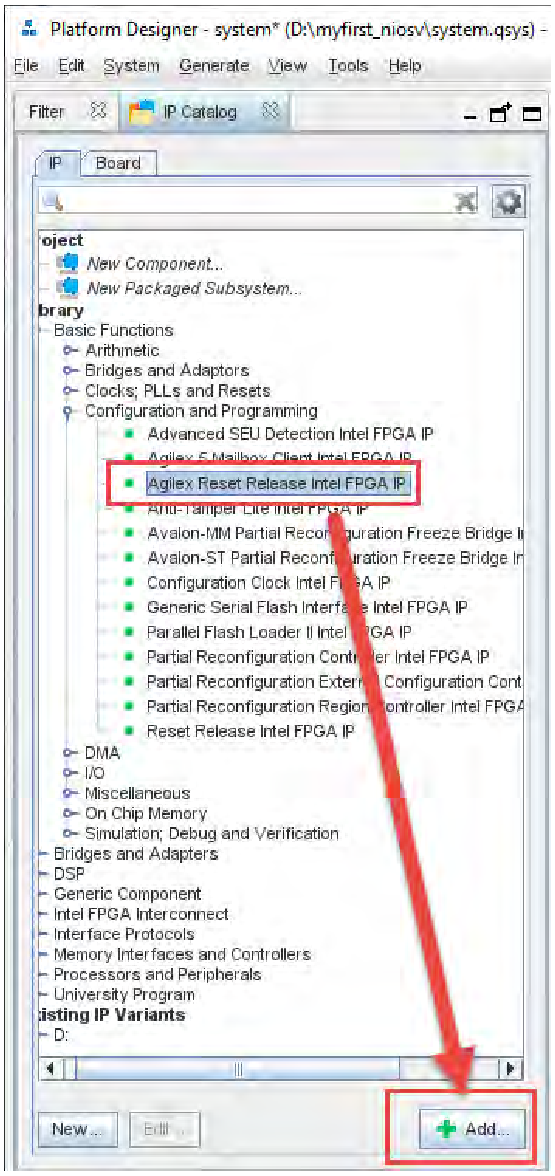

Platform Designer - system* (D:\myfirst_niosv\system.qsys) -
File Edit System Generate View Tools Help
Filter
IP Catalog
IP
Board
oject
New Component...
New Packaged Subsystem...
brary
Basic Functions
Arithmetic
Bridges and Adaptors
Clocks; PLLs and Resets
Configuration and Programming
Advanced SEU Detection Intel FPGA IP
Agilex 5 Mailbox Client Intel FPGA IP
Agilex Reset Release Intel FPGA IP
Anti-Tamper Lite Intel FPGA IP
Avalon-MM Partial Reconfiguration Freeze Bridge I
Avalon-ST Partial Reconfiguration Freeze Bridge In
Configuration Clock Intel FPGA IP
Generic Serial Flash Interface Intel FPGA IP
Parallel Flash Loader II Intel FPGA IP
Partial Reconfiguration Controller Intel FPGA IP
Partial Reconfiguration External Configuration Cont
Partial Reconfiguration Region Controller Intel FPGA
Reset Release Intel FPGA IP
DMA
I/O
Miscellaneous
On Chip Memory
Simulation; Debug and Verification
Bridges and Adapters
DSP
Generic Component
Intel FPGA Interconnect
Interface Protocols
Memory Interfaces and Controllers
Processors and Peripherals
University Program
isting IP Variants
D:
New...
Edit
Add...

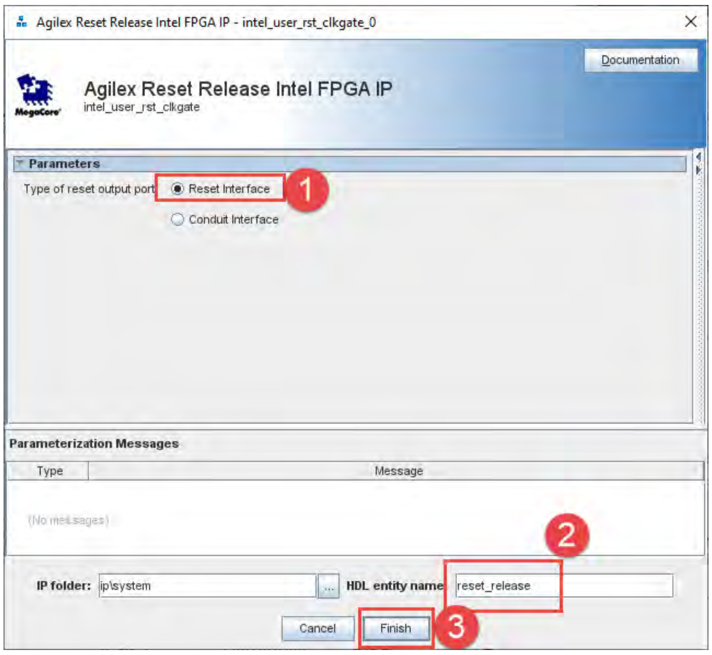
Figure 2-8 Add Agilex Reset Release

See Figure 2-9, In the Agilex Reset Release IP window:

a. Check **Reset Interface**.

b. Rename the HDL entity name to “**reset_release**”

c. Click **Finish**


Agilex Reset Release Intel FPGA IP - intel_user_rst_clkgate_0
Documentation
Agilex Reset Release Intel FPGA IP
intel_user_rst_clkgate
MegaCore
Parameters
Parameterization Messages
Type
Message
(No messages)
Cancel
Finish
1
2
3

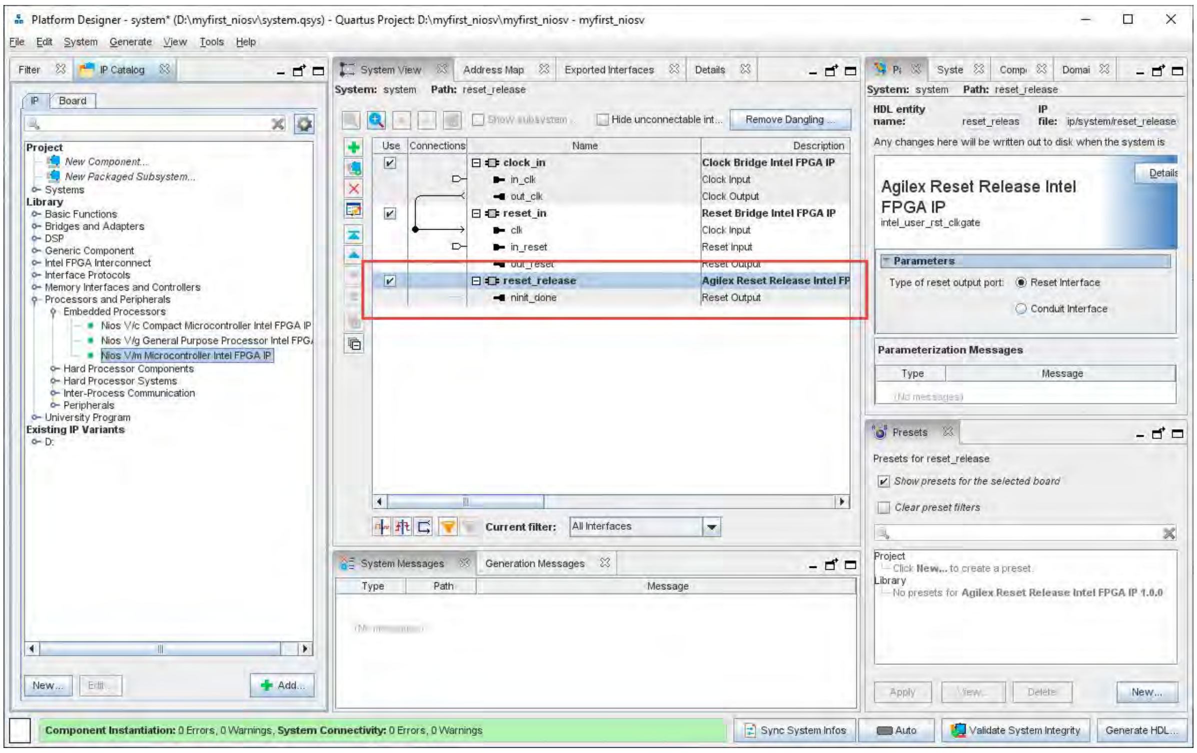
Type of reset output port: [✓ Reset Interface] Conduit Interface

IP folder: ip\system

HDL entity name: reset_release

Figure 2-9 Agilex Reset Release

A reset_release component is added in the system as shown in Figure 2-10.


Platform Designer - system* (D:\myfirst_niosv\system.qsys) - Quartus Project: D:\myfirst_niosv\myfirst_niosv - myfirst_niosv
File Edit System Generate View Tools Help
Filter
IP Catalog
IP
Board
Project
New Component...
New Packaged Subsystem...
Systems
Library
Basic Functions
Bridges and Adapters
DSP
Generic Component
Intel FPGA Interconnect
Interface Protocols
Memory Interfaces and Controllers
Processors and Peripherals
Embedded Processors
Nios V/c Compact Microcontroller Intel FPGA IP
Nios V/g General Purpose Processor Intel FPGA
Nios V/m Microcontroller Intel FPGA IP
Hard Processor Components
Hard Processor Systems
Inter-Process Communication
Peripherals
University Program
Existing IP Variants
D:
New...
Edit
Add...
System View
Address Map
Exported Interfaces
Details
System: system Path: reset_release
Show subsystem
Hide unconnectable int...
Remove Dangling ...
Current filter:
All Interfaces
System Messages
Generation Messages
Type
Path
Message
(No messages)
Presets
Presets for reset_release
Show presets for the selected board
Clear preset filters
Project
Click New... to create a preset.
Library
No presets for Agilex Reset Release Intel FPGA IP 1.0.0
Apply
Update
Delete
New...
Sync System Infos
Auto
Validate System Integrity
Generate HDL...

Use	Connections	Name	Description
✓		clock_in	Clock Bridge Intel FPGA IP
		in_clk	Clock Input
		out_clk	Clock Output
✓		reset_in	Reset Bridge Intel FPGA IP
		clk	Clock Input
		in_reset	Reset Input
		out_reset	Reset Output
✓		reset_release	Agilex Reset Release Intel FP
		ninit_done	Reset Output

System: system Path: reset_release

HDL entity name: reset_releas IP file: ip/system/reset_release

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

Agilex Reset Release Intel FPGA IP

intel_user_rst_clkgate

Parameters

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

Parameterization Messages

Type	Message
(No messages)	

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

Figure 2-10 Add reset_relese component

2. Then we will add a **Nios V microcontroller** into the system. Choose **Library** > **Processors and Peripherals > Embedded Processors** > **Nios V/m Microcontroller** to open wizard of adding cpu component. See Figure 2-11.

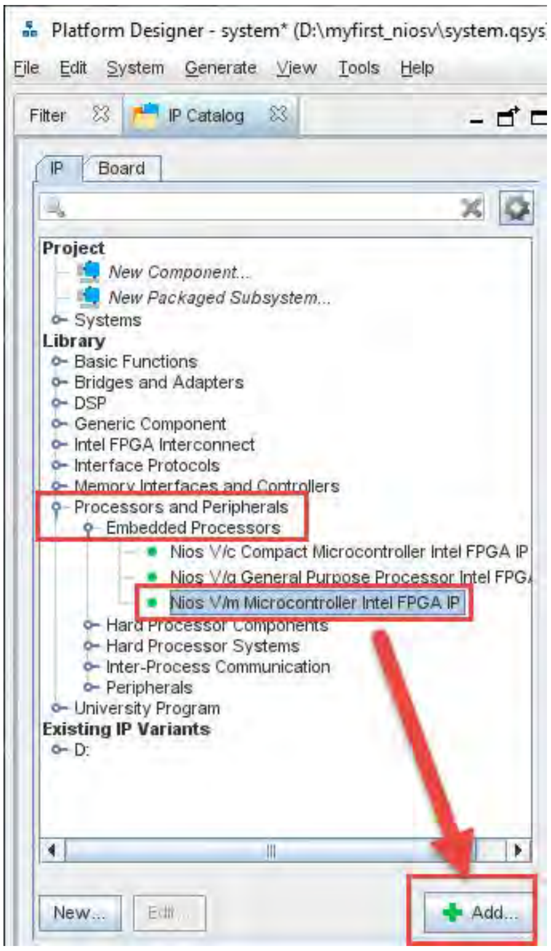
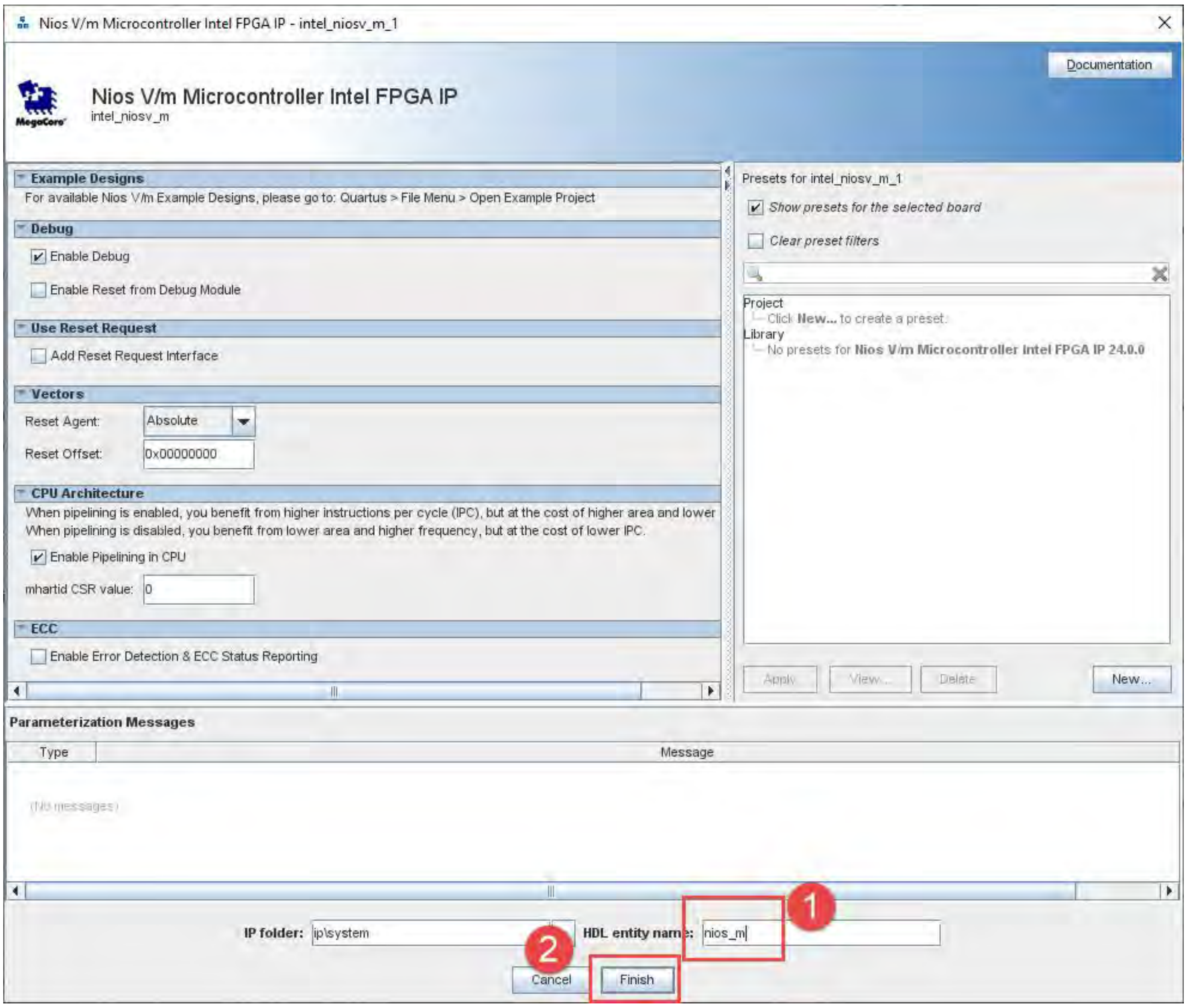

Platform Designer - system* (D:\myfirst_niosv\system.qsys
File Edit System Generate View Tools Help
Filter
IP Catalog
IP
Board
Project
New Component...
New Packaged Subsystem...
Systems
Library
Basic Functions
Bridges and Adapters
DSP
Generic Component
Intel FPGA Interconnect
Interface Protocols
Memory Interfaces and Controllers
Processors and Peripherals
Embedded Processors
Nios V/c Compact Microcontroller Intel FPGA IP
Nios V/g General Purpose Processor Intel FPG
Nios V/m Microcontroller Intel FPGA IP
Hard Processor Components
Hard Processor Systems
Inter-Process Communication
Peripherals
University Program
Existing IP Variants
D:
New...
Edit
Add...

Figure 2-11 Add Nios V/m Processor

See Figure 2-12, n the Nios V/m Microcontroller IP window:

a. Rename the HDL entity name to "**nios_m**"
b. Click **Finish**
c. Figure Figure 2-13 show the **nios_m** component is added in the system.


Nios V/m Microcontroller Intel FPGA IP - intel_niosv_m_1

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

Vectors
Reset Agent: Absolute
Reset Offset: 0x00000000

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

ECC
☐ Enable Error Detection & ECC Status Reporting

Presets for intel_niosv_m_1
☑ *Show presets for the selected board*
☐ *Clear preset filters*

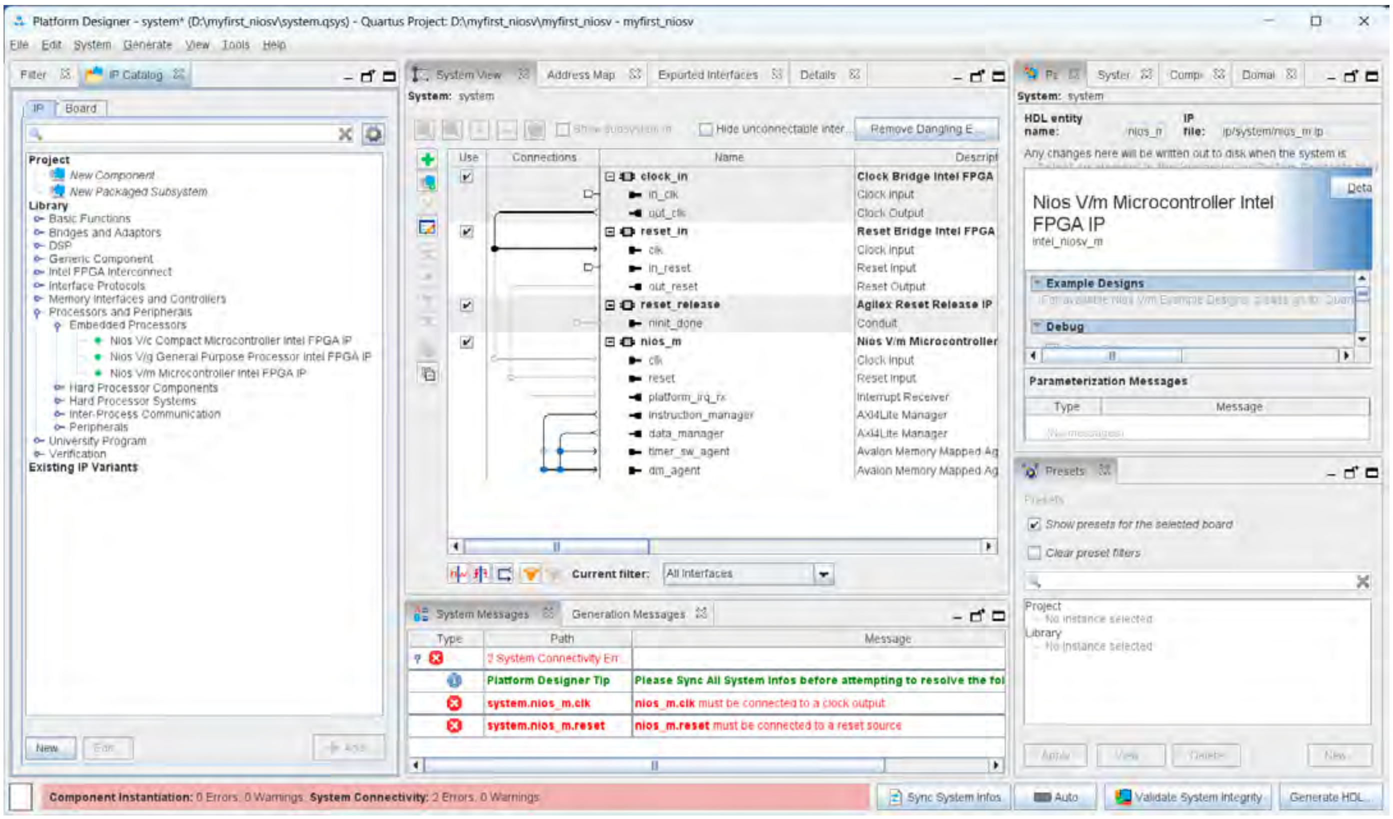
Project
Click **New...** to create a preset.
Library
No presets for **Nios V/m Microcontroller Intel FPGA IP 24.0.0**

Parameterization Messages

Type	Message
(No messages)	

IP folder: ip\system
HDL entity name: nios_m

Figure 2-12 Nios V/m Processor


Platform Designer - system* (D:\myfirst_niosv\system.qsys) - Quartus Project: D:\myfirst_niosv\myfirst_niosv - myfirst_niosv

System: system

Use	Name	Description
✓	clock_in	Clock Bridge Intel FPGA
	in_clk	Clock Input
	out_clk	Clock Output
✓	reset_in	Reset Bridge Intel FPGA
	clk	Clock Input
	in_reset	Reset Input
	out_reset	Reset Output
✓	reset_release	Agilex Reset Release IP
	ninit_done	Conduit
✓	nios_m	Nios V/m Microcontroller
	clk	Clock Input
	reset	Reset Input
	platform_irq_rx	Interrupt Receiver
	instruction_manager	AXI4Lite Manager
	data_manager	AXI4Lite Manager
	timer_sw_agent	Avalon Memory Mapped Ag
	dm_agent	Avalon Memory Mapped Ag

System Messages

Type	Path	Message
✗	2 System Connectivity Err	
●	Platform Designer Tip	Please Sync All System Infos before attempting to resolve the fol
✗	system.nios_m.clk	nios_m.clk must be connected to a clock output
✗	system.nios_m.reset	nios_m.reset must be connected to a reset source

Component Instantiation: 0 Errors, 0 Warnings System Connectivity: 2 Errors, 0 Warnings

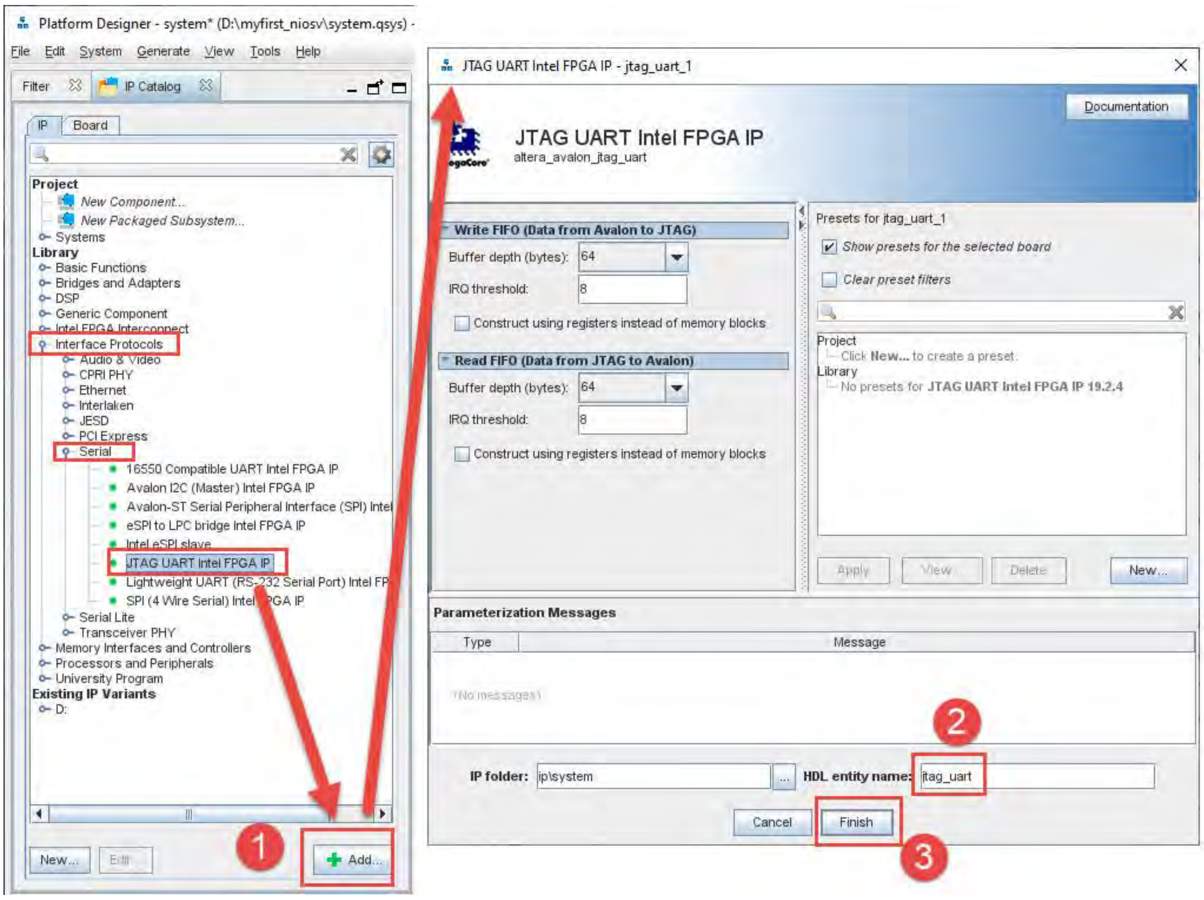
Figure 2-13 Add Nios V CPU completely

3. Choose **Library** > **Interface Protocols** > **Serial** > **JTAG UART Intel FPGA IP** to open wizard of adding JTAG UART as shown in Figure 2-14.

In the JTAG UART IP window:

a. Rename the HDL entity name to “**jatg_uart**”
b. Click **Finish**

4. Figure Figure 2-15 show the **jtag_usrt** component is added in the system.


Platform Designer - system* (D:\myfirst_niosv\system.qsys)
File Edit System Generate View Tools Help
Filter
IP Catalog
IP
Board
Project
New Component...
New Packaged Subsystem...
Systems
Library
Basic Functions
Bridges and Adapters
DSP
Generic Component
Intel FPGA Interconnect
Interface Protocols
Audio & Video
CPRI PHY
Ethernet
Interlaken
JESD
PCI Express
Serial
16550 Compatible UART Intel FPGA IP
Avalon I2C (Master) Intel FPGA IP
Avalon-ST Serial Peripheral Interface (SPI) Intel
eSPI to LPC bridge Intel FPGA IP
Intel eSPI slave
JTAG UART Intel FPGA IP
Lightweight UART (RS-232 Serial Port) Intel FP
SPI (4 Wire Serial) Intel FPGA IP
Serial Lite
Transceiver PHY
Memory Interfaces and Controllers
Processors and Peripherals
University Program
Existing IP Variants
D:
New...
Edit...
Add...
1
JTAG UART Intel FPGA IP - jtag_uart_1
Documentation
JTAG UART Intel FPGA IP
altera_avalon_jtag_uart
2
3

Write FIFO (Data from Avalon to JTAG)

Buffer depth (bytes): 64

IRQ threshold: 8

☐ Construct using registers instead of memory blocks

Read FIFO (Data from JTAG to Avalon)

Buffer depth (bytes): 64

IRQ threshold: 8

☐ Construct using registers instead of memory blocks

Presets for jtag_uart_1

☑ Show presets for the selected board

☐ Clear preset filters

Project
Click New... to create a preset.
Library
No presets for JTAG UART Intel FPGA IP 19.2.4

Apply View Delete New...

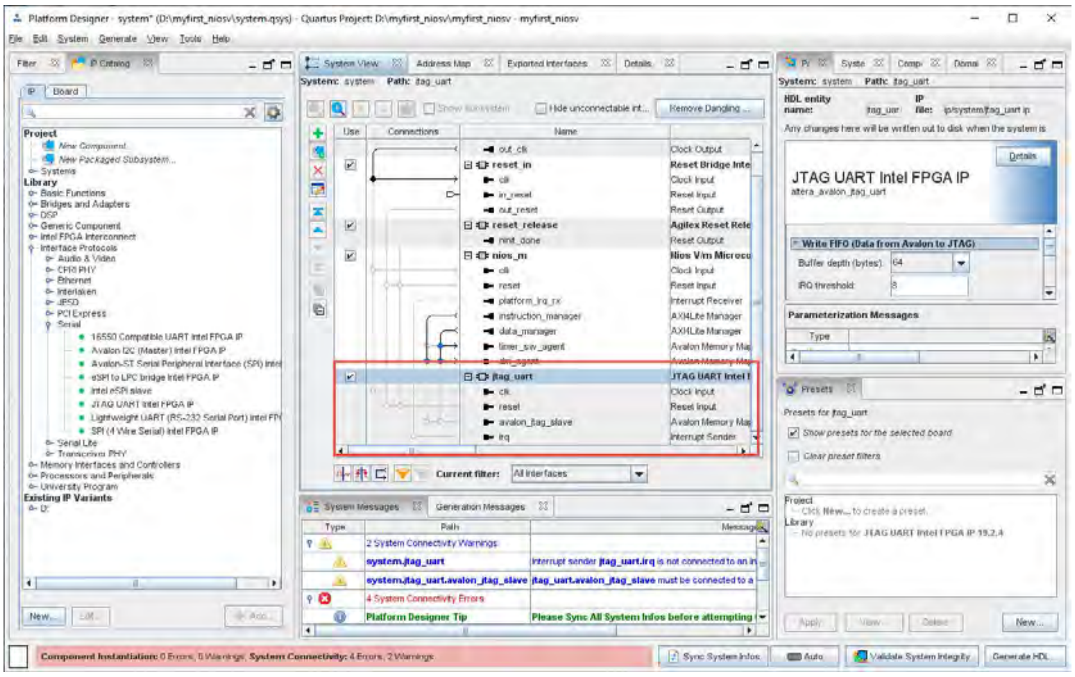
Parameterization Messages

Type	Message
(No messages)	

IP folder: ip\system HDL entity name: jtag_uart

Cancel Finish

Figure 2-14 Add JTAG UART (1)


Platform Designer - system* (D:\myfirst_niosv\system.qsys) - Quartus Project: D:\myfirst_niosv\myfirst_niosv - myfirst_niosv
File Edit System Generate View Tools Help
System View
Address Map
Exported Interfaces
Details
System: system Path: jtag_uart
Use
Connections
Name
clk
reset_in
reset_release
nios_m
jtag_uart
avalon_jtag_slave
irq
Current filter: All interfaces
HDL entity name: jtag_uart
JTAG UART Intel FPGA IP
altera_avalon_jtag_uart
Write FIFO (Data from Avalon to JTAG)
Buffer depth (bytes): 64
IRQ threshold: 8
Parameterization Messages
Presets
Sync System Infos
Auto
Validate System Integrity
Generate HDL...

System Messages Generation Messages

Type	Path	Message
⚠	2 System Connectivity Warnings	
⚠	system.jtag_uart	Interrupt sender jtag_uart.irq is not connected to an in
⚠	system.jtag_uart.avalon_jtag_slave	jtag_uart.avalon_jtag_slave must be connected to a
✖	4 System Connectivity Errors	
ⓘ	Platform Designer Tip	Please Sync All System Infos before attempting

Component Instantiation: 0 Errors, 0 Warnings System Connectivity: 4 Errors, 2 Warnings

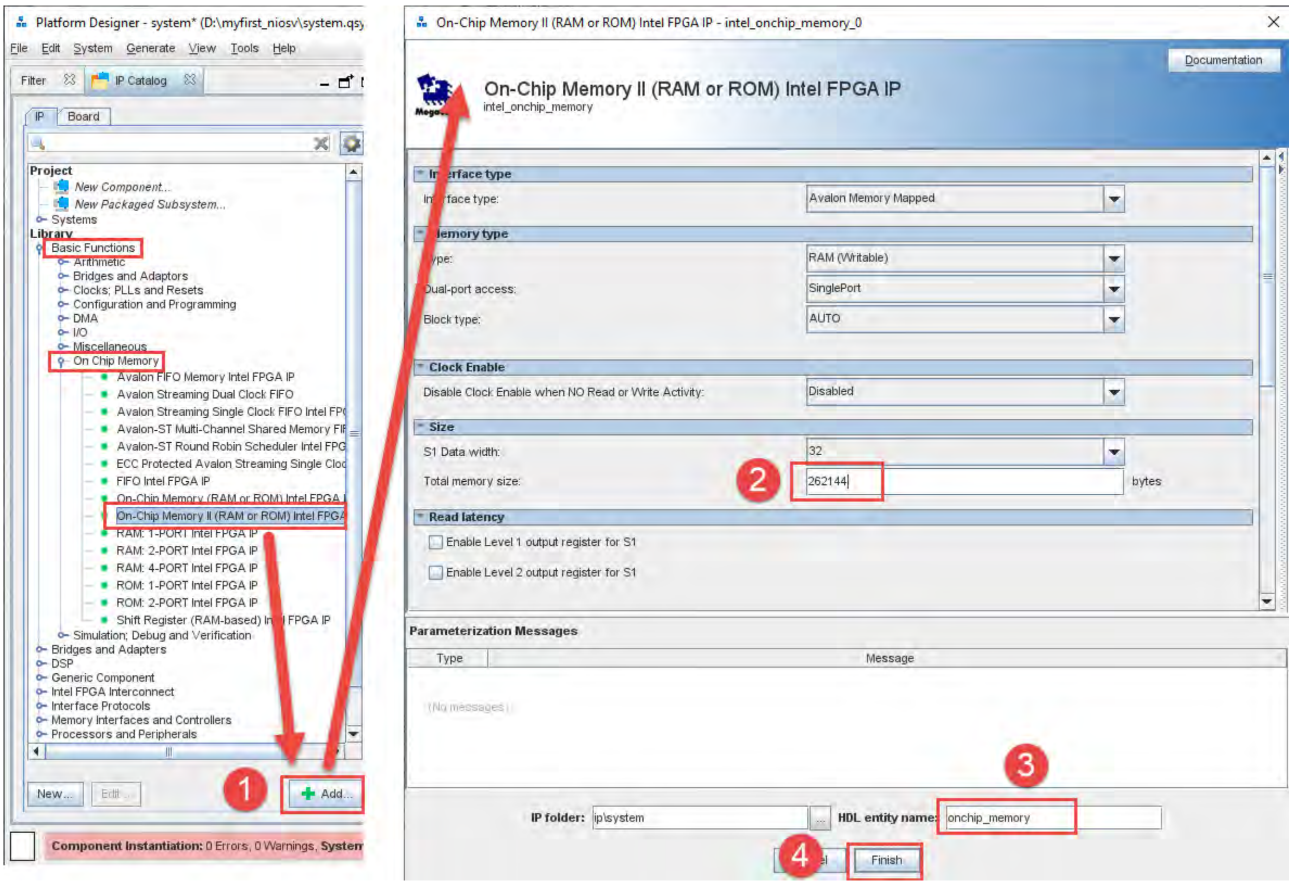
Figure 2-15 JTAG UART

5. Choose **Library** > **Basic Functions** > **On-Chip** Memory > **On-Chip Memory II (RAM or ROM) Intel FPGA IP** to open wizard of adding On-Chip memory. See Figure 2-16.

In the On-Chip Memory II IP window:

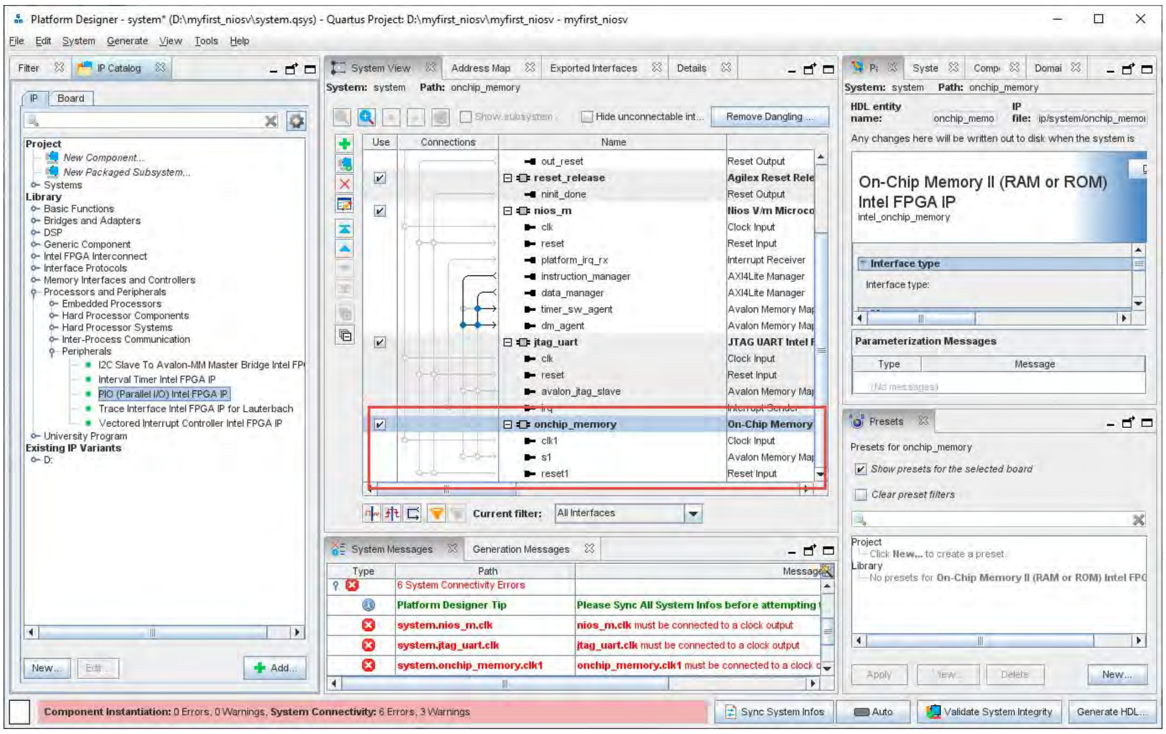
a. Change memory size to "**262144**".
b. Rename the HDL entity name to "**onchip_memory**"
c. Click **Finish**

6. Figure 2-17 show the **onchip_memory** component is added in the system.


Platform Designer - system* (D:\myfirst_niosv\system.qsys
File Edit System Generate View Tools Help
IP Catalog
Project
New Component...
New Packaged Subsystem...
Library
Basic Functions
On Chip Memory
On-Chip Memory II (RAM or ROM) Intel FPGA
1
Add...
On-Chip Memory II (RAM or ROM) Intel FPGA IP - intel_onchip_memory_0
On-Chip Memory II (RAM or ROM) Intel FPGA IP
intel_onchip_memory
Documentation
2
3
4
Finish

Interface type: Avalon Memory Mapped
Memory type
Type: RAM (Writable)
Dual-port access: SinglePort
Block type: AUTO
Clock Enable
Disable Clock Enable when NO Read or Write Activity: Disabled
Size
S1 Data width: 32
Total memory size: 262144 bytes
Read latency
Enable Level 1 output register for S1
Enable Level 2 output register for S1
Parameterization Messages
(No messages)
IP folder: ip\system
HDL entity name: onchip_memory

Figure 2-16 Add On-Chip Memory


Platform Designer - system* (D:\myfirst_niosv\system.qsys) - Quartus Project: D:\myfirst_niosv\myfirst_niosv - myfirst_niosv
File Edit System Generate View Tools Help
IP Catalog
System View
Address Map
Exported Interfaces
Details
Remove Dangling ...
Current filter: All Interfaces
Sync System Infos
Validate System Integrity
Generate HDL...

Use	Connections	Name	Description
		out_reset	Reset Output
✓		reset_release	Agilex Reset Rele
		ninit_done	Reset Output
✓		nios_m	Nios V/m Microco
		clk	Clock Input
		reset	Reset Input
		platform_irq_rx	Interrupt Receiver
		instruction_manager	AXI4Lite Manager
		data_manager	AXI4Lite Manager
		timer_sw_agent	Avalon Memory Map
		dm_agent	Avalon Memory Map
✓		jtag_uart	JTAG UART Intel F
		clk	Clock Input
		reset	Reset Input
		avalon_jtag_slave	Avalon Memory Map
		irq	Interrupt Sender
✓		onchip_memory	On-Chip Memory
		clk1	Clock Input
		s1	Avalon Memory Map
		reset1	Reset Input

Type	Path	Message
	6 System Connectivity Errors	
	Platform Designer Tip	Please Sync All System Infos before attempting
	system.nios_m.clk	nios_m.clk must be connected to a clock output
	system.jtag_uart.clk	jtag_uart.clk must be connected to a clock output
	system.onchip_memory.clk1	onchip_memory.clk1 must be connected to a clock o

Component Instantiation: 0 Errors, 0 Warnings, System Connectivity: 6 Errors, 3 Warnings

Figure 2-17 Add On-Chip memory Completely

7. Choose **Library > Processors and Peripherals > Peripherals > PIO (Parallel I/O)** to open wizard of adding PIO. See Figure 2-18.

In the PIO(Parallel I/O) IP window:

a. Rename the HDL entity name to “**pio_led**”
b. Click **Finish**

8. Figure 2-19 show the **pio_led** component is added in the system.

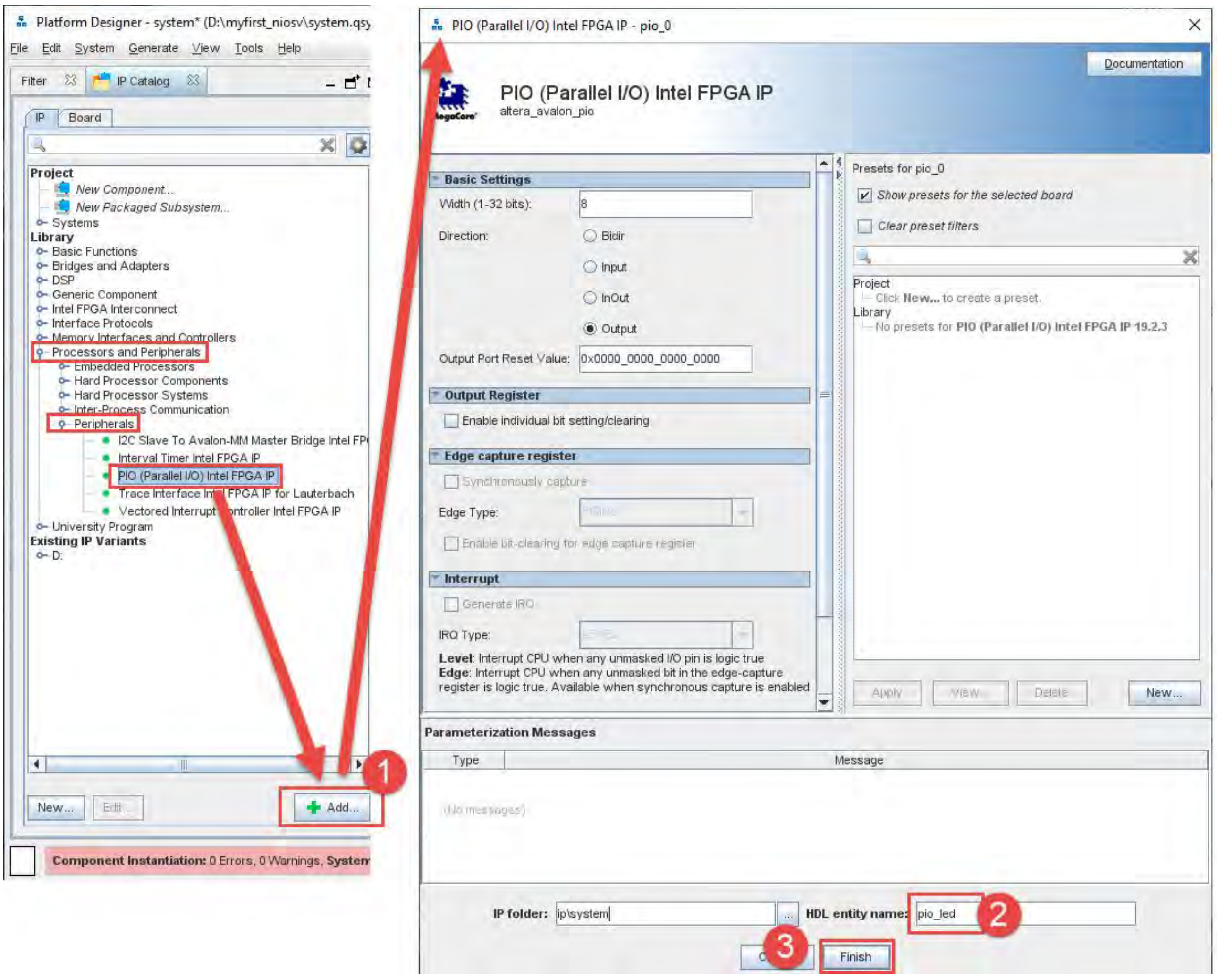
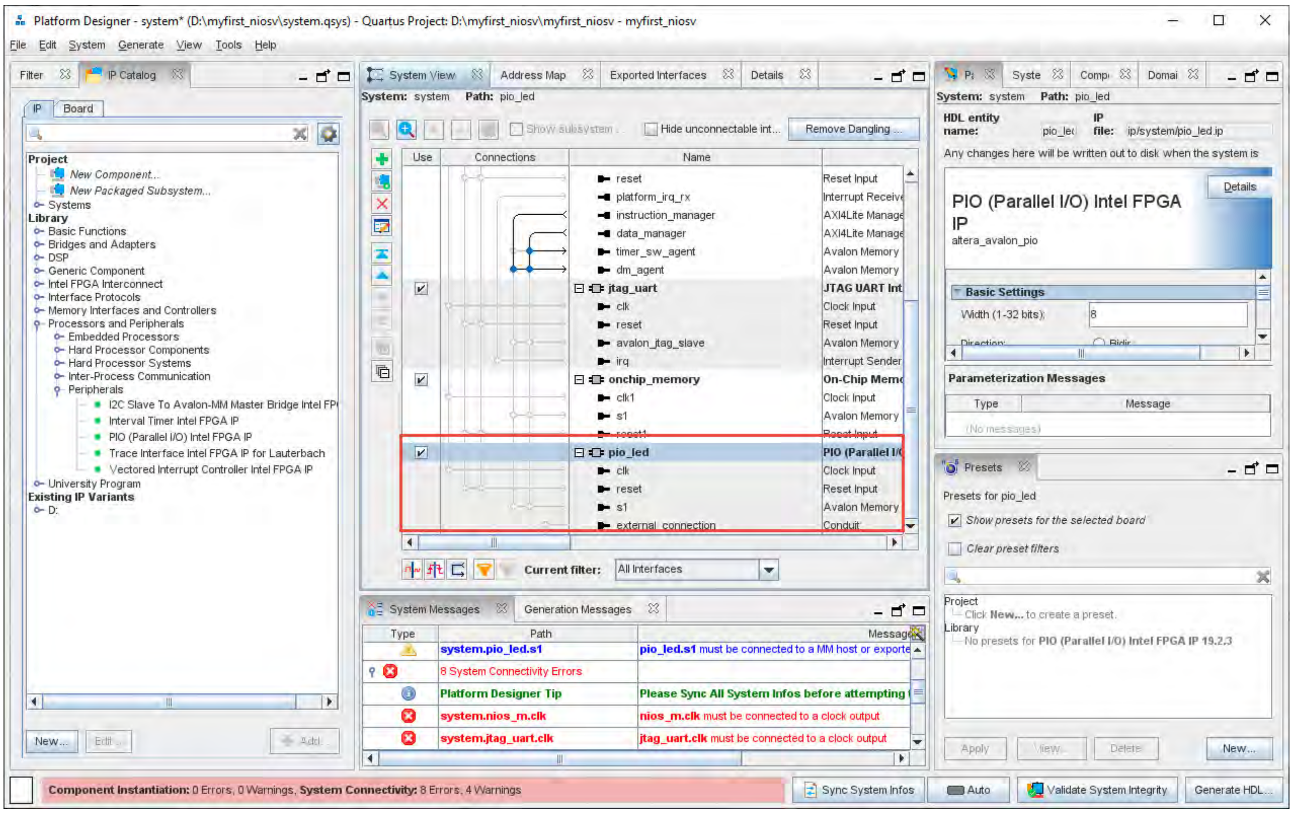

Platform Designer - system* (D:\myfirst_niosv\system.qsy
File Edit System Generate View Tools Help
Filter IP Catalog
IP Board
Project
New Component...
New Packaged Subsystem...
Systems
Library
Basic Functions
Bridges and Adapters
DSP
Generic Component
Intel FPGA Interconnect
Interface Protocols
Memory Interfaces and Controllers
Processors and Peripherals
Embedded Processors
Hard Processor Components
Hard Processor Systems
Inter-Process Communication
Peripherals
I2C Slave To Avalon-MM Master Bridge Intel FP
Interval Timer Intel FPGA IP
PIO (Parallel I/O) Intel FPGA IP
Trace Interface Intel FPGA IP for Lauterbach
Vectored Interrupt Controller Intel FPGA IP
University Program
Existing IP Variants
D:
New... Edit... Add...
1
Component Instantiation: 0 Errors, 0 Warnings, System
PIO (Parallel I/O) Intel FPGA IP - pio_0
Documentation
PIO (Parallel I/O) Intel FPGA IP
altera_avalon_pio
Basic Settings
Width (1-32 bits): 8
Direction: Bidir Input InOut [✓ Output]
Output Port Reset Value: 0x0000_0000_0000_0000
Output Register
Enable individual bit setting/clearing
Edge capture register
Synchronously capture
Edge Type:
Enable bit-clearing for edge capture register
Interrupt
Generate IRQ
IRQ Type:
Level: Interrupt CPU when any unmasked I/O pin is logic true
Edge: Interrupt CPU when any unmasked bit in the edge-capture register is logic true. Available when synchronous capture is enabled
Presets for pio_0
[✓] Show presets for the selected board
Clear preset filters
Project
Click New... to create a preset.
Library
No presets for PIO (Parallel I/O) Intel FPGA IP 19.2.3
Apply View... Delete New...
Parameterization Messages
Type Message
(No messages)
IP folder: ip\system
HDL entity name: pio_led
2
3
Finish

Figure 2-18 Add PIO


Platform Designer - system* (D:\myfirst_niosv\system.qsys) - Quartus Project: D:\myfirst_niosv\myfirst_niosv - myfirst_niosv
File Edit System Generate View Tools Help
IP Catalog
System View
Address Map
Exported Interfaces
Details
Remove Dangling
Hide unconnectable int...
Current filter: All Interfaces
System Messages
Generation Messages
Presets
Sync System Infos
Auto
Validate System Integrity
Generate HDL...

System: system Path: pio_led

Use	Name	Description
	reset	Reset Input
	platform_irq_rx	Interrupt Receiver
	instruction_manager	AXI4Lite Manager
	data_manager	AXI4Lite Manager
	timer_sw_agent	Avalon Memory
	dm_agent	Avalon Memory
✓	jtag_uart	JTAG UART Int
	clk	Clock Input
	reset	Reset Input
	avalon_jtag_slave	Avalon Memory
	irq	Interrupt Sender
✓	onchip_memory	On-Chip Memo
	clk1	Clock Input
	s1	Avalon Memory
	reset1	Reset Input
✓	pio_led	PIO (Parallel I/O
	clk	Clock Input
	reset	Reset Input
	s1	Avalon Memory
	external_connection	Conduit

HDL entity name: pio_led IP file: ip/system/pio_led.ip

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

PIO (Parallel I/O) Intel FPGA IP
altera_avalon_pio

Basic Settings
Width (1-32 bits): 8

Type	Path	Message
⚠	system.pio_led.s1	pio_led.s1 must be connected to a MM host or exporte
✖	8 System Connectivity Errors	
ⓘ	Platform Designer Tip	Please Sync All System Infos before attempting
✖	system.nios_m.clk	nios_m.clk must be connected to a clock output
✖	system.jtag_uart.clk	jtag_uart.clk must be connected to a clock output

Presets for pio_led
Show presets for the selected board
Clear preset filters
Project
Click New... to create a preset.
Library
No presets for PIO (Parallel I/O) Intel FPGA IP 19.2.3

Component Instantiation: 0 Errors, 0 Warnings, System Connectivity: 8 Errors, 4 Warnings

Figure 2-19 PIO

■ Connect Interfaces and Signals

This section explains two ways to connect components in Platform Designer.

- Method 1: Click on an open connection circle to link compatible interfaces. Once connected, the line turns black and the circle is filled. Clicking a filled circle will remove the connection.

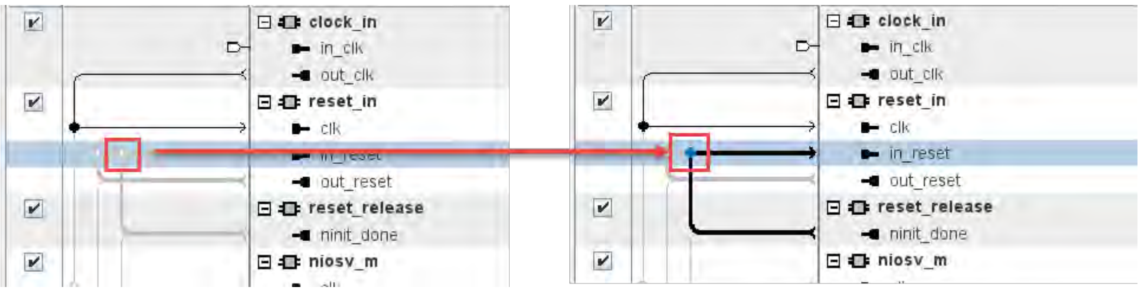


Figure 2-20 Connect interfaces and signals (1)

- Method 2:

In the Platform Designer window (see Figure 2-21), right-click **ninit_done** of **the**

reset_release component, then select:

Connections: reset_release.ninit_done > reset_in.in_reset

This connects reset_release.ninit_done to reset_in.in_reset, as shown in the figure.

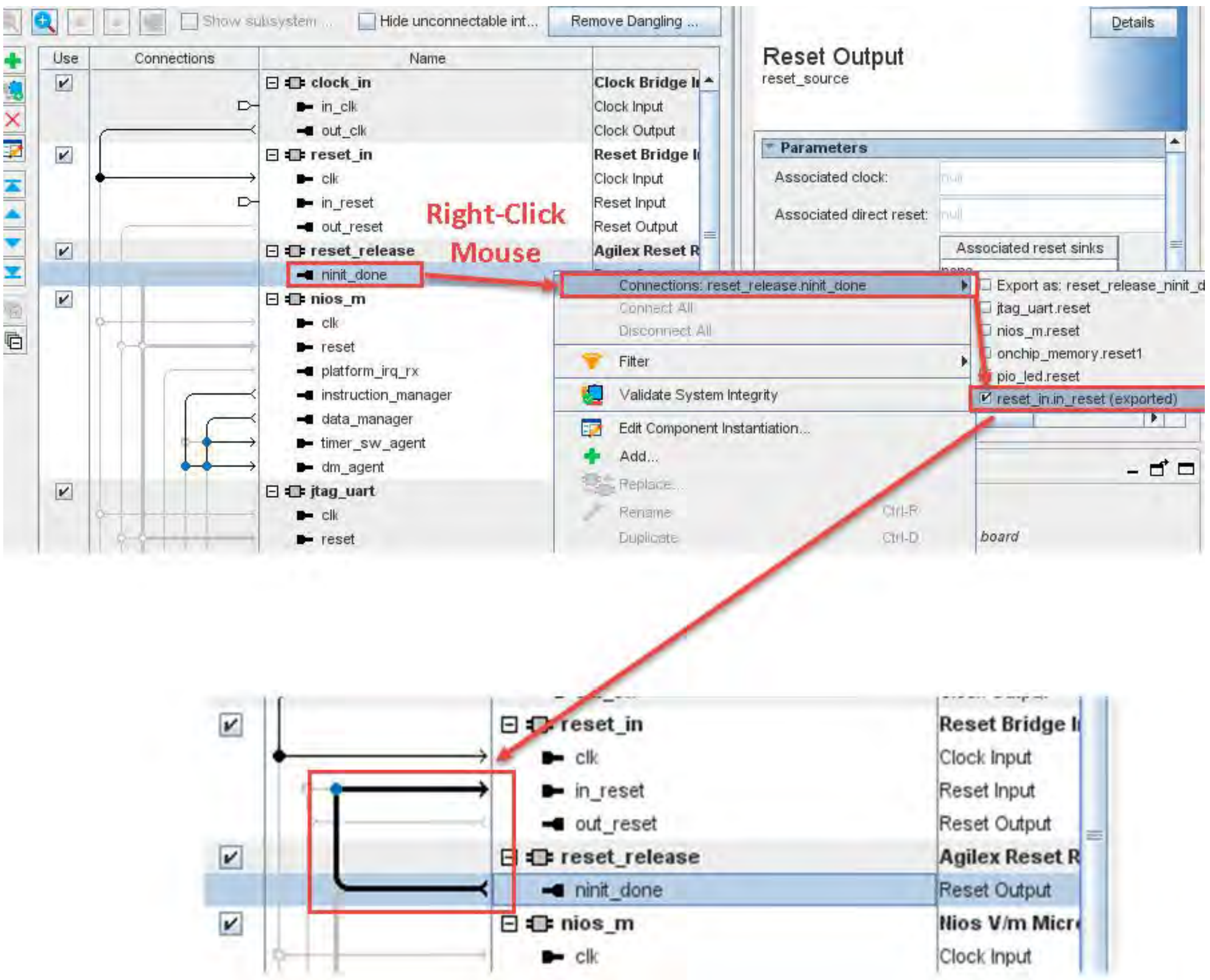


Figure 2-21 Connect interfaces and signals (2)

Use any of the methods above to connect the components in the system as specified in the table below.

Component Name	Signal Name	Connected Component	Connected Signal
reset_release	ninit_done	reset_in	reset_in.in_reset
nios_m	clk	clock_in	clock_in.out_clk
	reset	reset_in	reset_in.out_reset
jtag_uart	clk	clock_in	clock_in.out_clk
	reset	reset_in	reset_in.out_reset
	avalon_jtag_slave	niosv_m	niosv_m.data_manager
	irq	niosv_m	niosv_m.platform_irq_rx
onchip_memory	clk	clock_in	clock_in.out_clk
	s1	niosv_m	niosv_m.data_manager niosv_m.instruction_manager
	reset	reset_in	reset_in.out_reset
pio_led	clk	clock_in	clock_in.out_clk
	reset	reset_in	reset_in.out_reset
	s1	niosv_m	niosv_m.data_manager
	external_connection		Export as: pio_led_external_connection

Choose **external_connection of pio_led** and right-click then choose **Connections > Export as: pio_led_external_connection. (See** Figure 2-22**)**

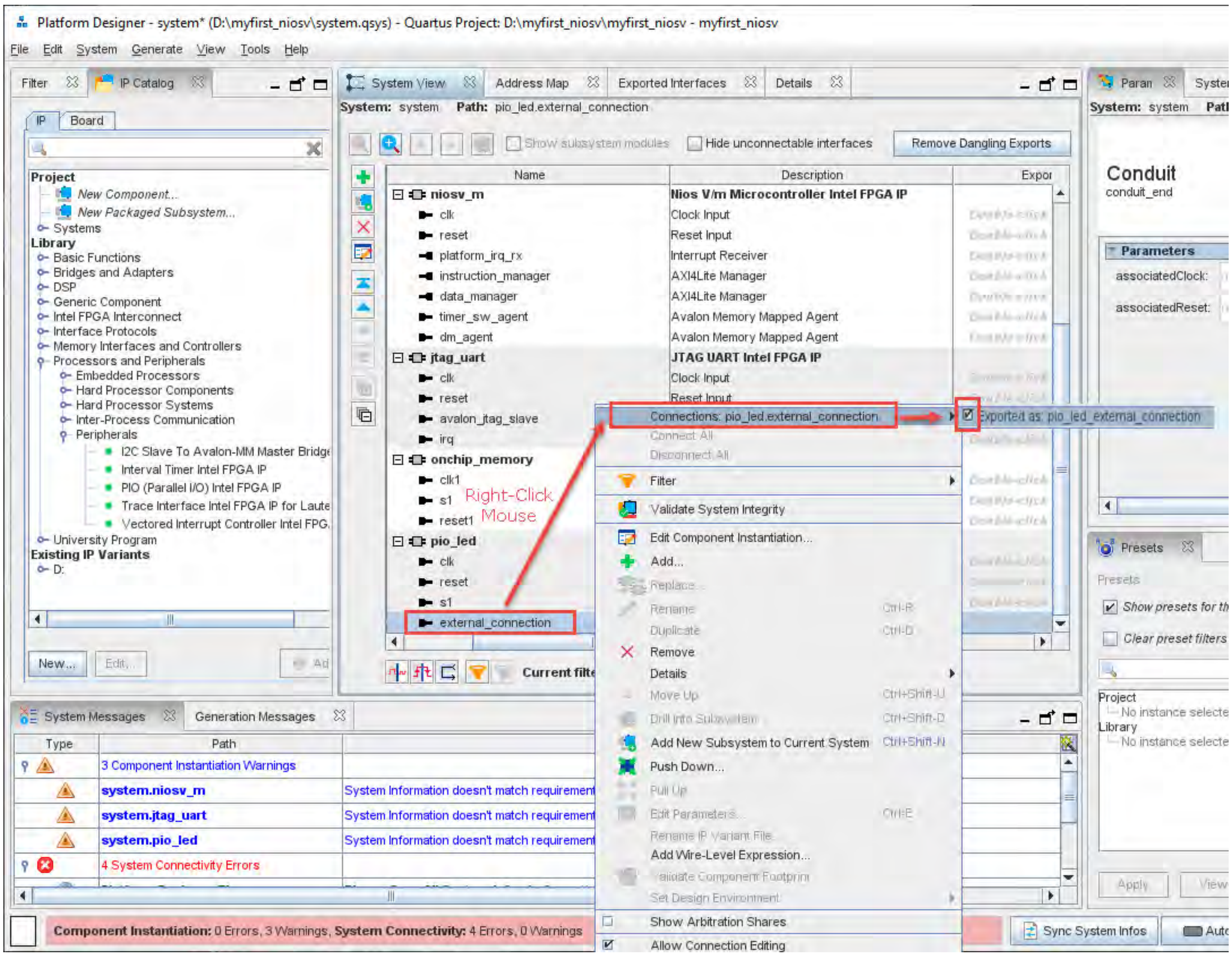
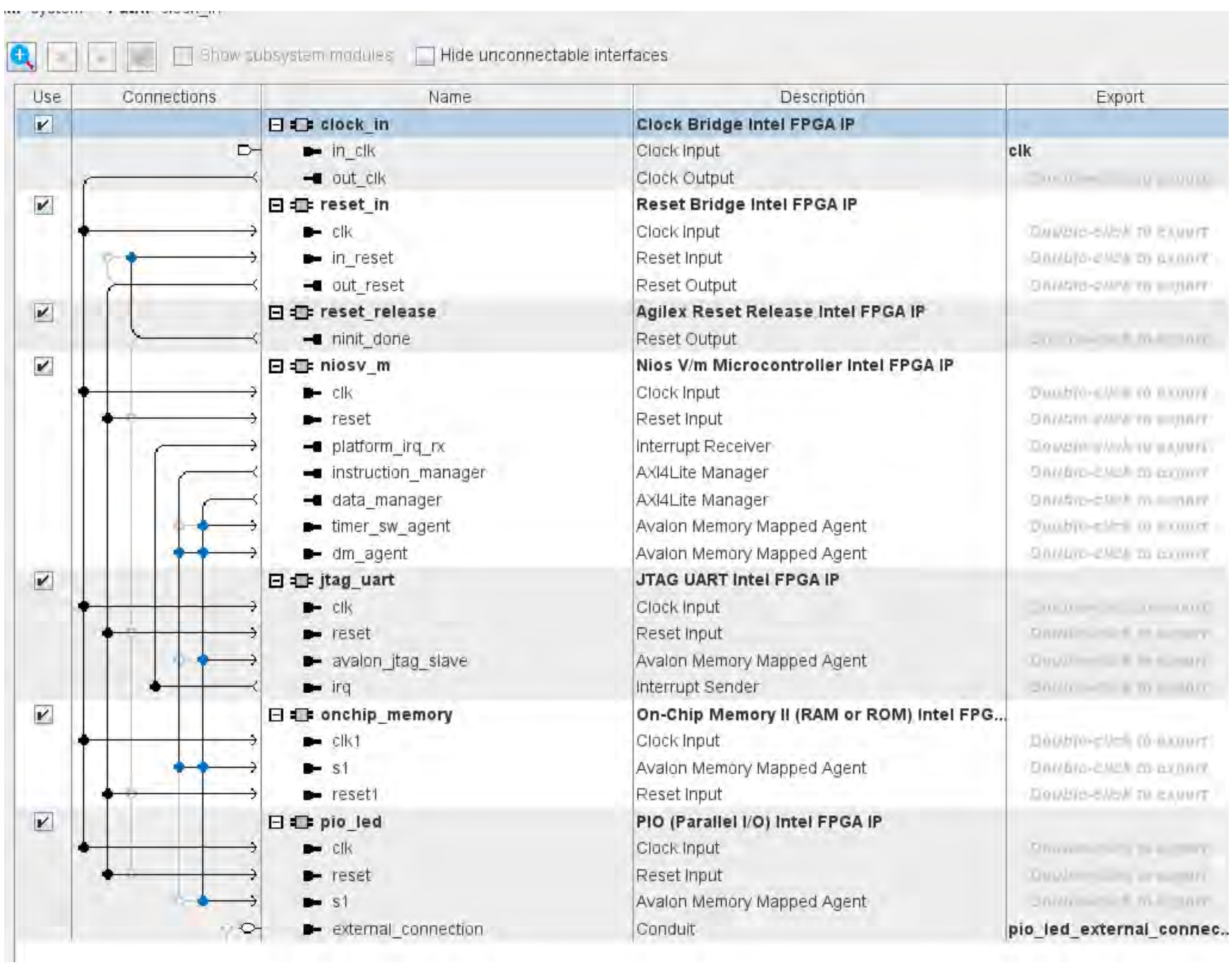

Platform Designer - system* (D:\myfirst_niosv\system.qsys) - Quartus Project: D:\myfirst_niosv\myfirst_niosv - myfirst_niosv
Right-Click Mouse
Connections: pio_led.external_connection
Exported as: pio_led_external_connection
external_connection

Figure 2-22 Setting external_connection of pio_led

Refer to Figure 2-23 for the final connection configuration.

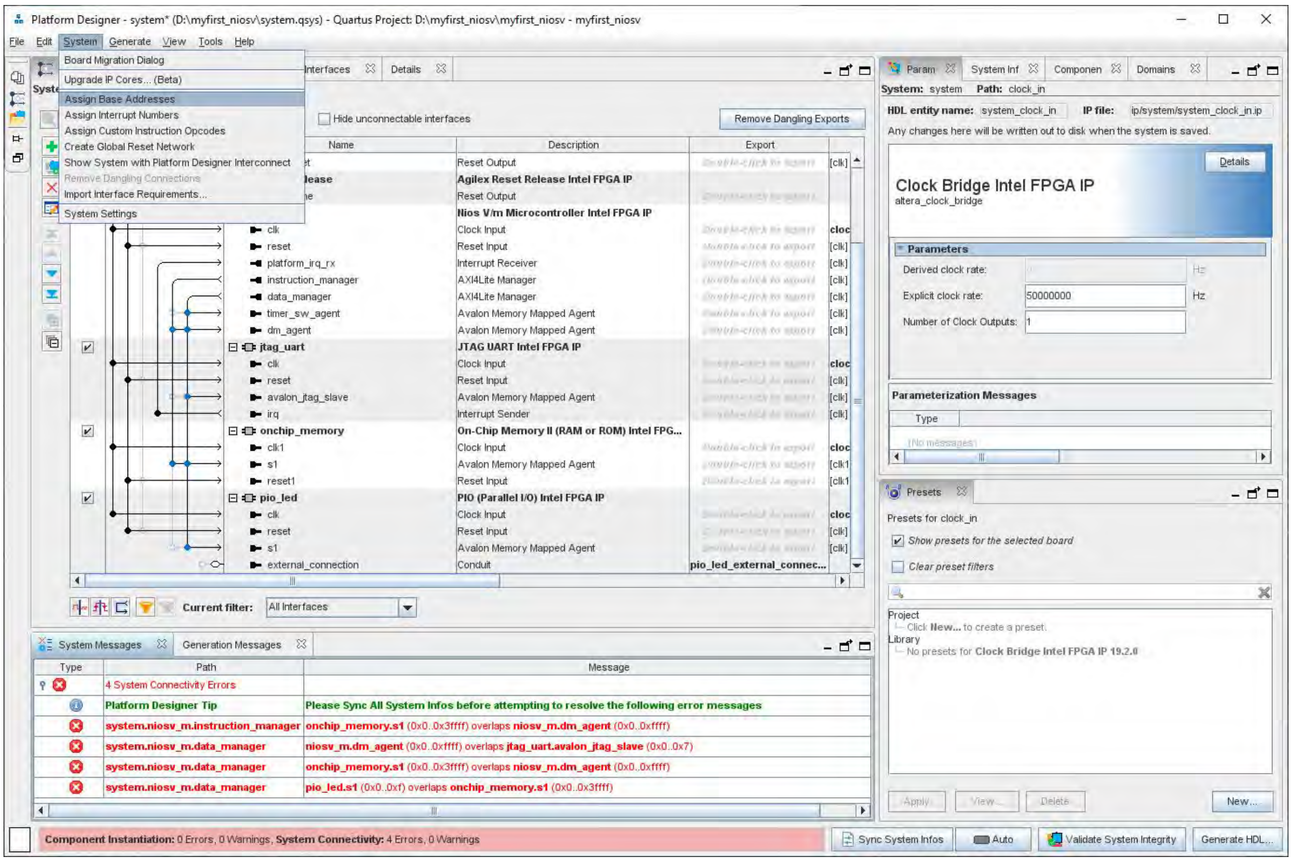

Show subsystem modules   Hide unconnectable interfaces

Use	Connections	Name	Description	Export
✓		⊟ clock_in	Clock Bridge Intel FPGA IP	
		in_clk	Clock Input	clk
		out_clk	Clock Output	Double-click to export
✓		⊟ reset_in	Reset Bridge Intel FPGA IP	
		clk	Clock Input	Double-click to export
		in_reset	Reset Input	Double-click to export
		out_reset	Reset Output	Double-click to export
✓		⊟ reset_release	Agilex Reset Release Intel FPGA 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
		reset	Reset Input	Double-click to export
		platform_irq_rx	Interrupt Receiver	Double-click to export
		instruction_manager	AXI4Lite Manager	Double-click to export
		data_manager	AXI4Lite Manager	Double-click to export
		timer_sw_agent	Avalon Memory Mapped Agent	Double-click to export
		dm_agent	Avalon Memory Mapped Agent	Double-click to export
✓		⊟ jtag_uart	JTAG UART Intel FPGA IP	
		clk	Clock Input	Double-click to export
		reset	Reset Input	Double-click to export
		avalon_jtag_slave	Avalon Memory Mapped Agent	Double-click to export
		irq	Interrupt Sender	Double-click to export
✓		⊟ onchip_memory	On-Chip Memory II (RAM or ROM) Intel FPG...	
		clk1	Clock Input	Double-click to export
		s1	Avalon Memory Mapped Agent	Double-click to export
		reset1	Reset Input	Double-click to export
✓		⊟ pio_led	PIO (Parallel I/O) Intel FPGA 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	pio_led_external_connec..

Figure 2-23 final connection configuration

■ Clear System Warnings and Errors

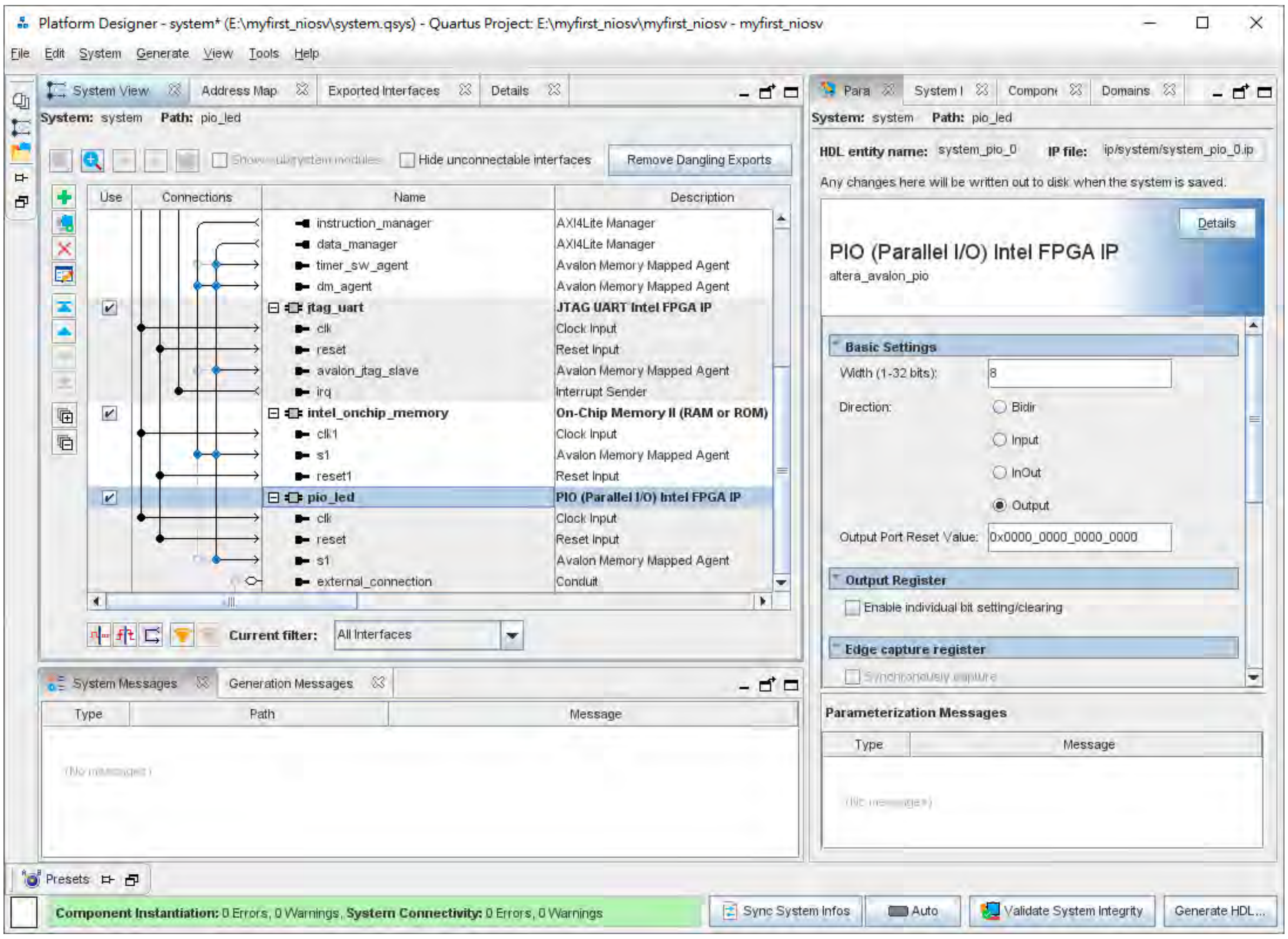
1. Choose **System** > **Assign Base Addresses** as shown in Figure 2-24. After that, you will find that there is no error in the message window as shown in Figure 2-25.


Platform Designer - system* (D:\myfirst_niosv\system.qsys) - Quartus Project: D:\myfirst_niosv\myfirst_niosv - myfirst_niosv
File
Edit
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
Clock Bridge Intel FPGA IP
Sync System Infos
Validate System Integrity
Generate HDL...

Type	Path	Message
	4 System Connectivity Errors	
	Platform Designer Tip	Please Sync All System Infos before attempting to resolve the following error messages
	system.niosv_m.instruction_manager	onchip_memory.s1 (0x0..0x3ffff) overlaps niosv_m.dm_agent (0x0..0xffff)
	system.niosv_m.data_manager	niosv_m.dm_agent (0x0..0xffff) overlaps jtag_uart.avalon_jtag_slave (0x0..0x7)
	system.niosv_m.data_manager	onchip_memory.s1 (0x0..0x3ffff) overlaps niosv_m.dm_agent (0x0..0xffff)
	system.niosv_m.data_manager	pio_led.s1 (0x0..0xf) overlaps onchip_memory.s1 (0x0..0x3ffff)

Component Instantiation: 0 Errors, 0 Warnings, **System Connectivity:** 4 Errors, 0 Warnings

Figure 2-24 Assign Base Addresses


Platform Designer - system* (E:\myfirst_niosv\system.qsys) - Quartus Project: E:\myfirst_niosv\myfirst_niosv - myfirst_niosv

System: system Path: pio_led

Use	Name	Description
	instruction_manager	AXI4Lite Manager
	data_manager	AXI4Lite Manager
	timer_sw_agent	Avalon Memory Mapped Agent
	dm_agent	Avalon Memory Mapped Agent
✓	jtag_uart	JTAG UART Intel FPGA IP
	clk	Clock Input
	reset	Reset Input
	avalon_jtag_slave	Avalon Memory Mapped Agent
	irq	Interrupt Sender
✓	intel_onchip_memory	On-Chip Memory II (RAM or ROM)
	clk1	Clock Input
	s1	Avalon Memory Mapped Agent
	reset1	Reset Input
✓	pio_led	PIO (Parallel I/O) Intel FPGA IP
	clk	Clock Input
	reset	Reset Input
	s1	Avalon Memory Mapped Agent
	external_connection	Conduit

Current filter: All Interfaces

System: system Path: pio_led

HDL entity name: system_pio_0 IP file: ip/system/system_pio_0.ip

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

PIO (Parallel I/O) Intel FPGA IP
altera_avalon_pio

Basic Settings
Width (1-32 bits): 8
Direction: Bidir / Input / InOut / [✓ Output]
Output Port Reset Value: 0x0000_0000_0000_0000

Output Register
Enable individual bit setting/clearing

Edge capture register
Synchronously capture

Parameterization Messages

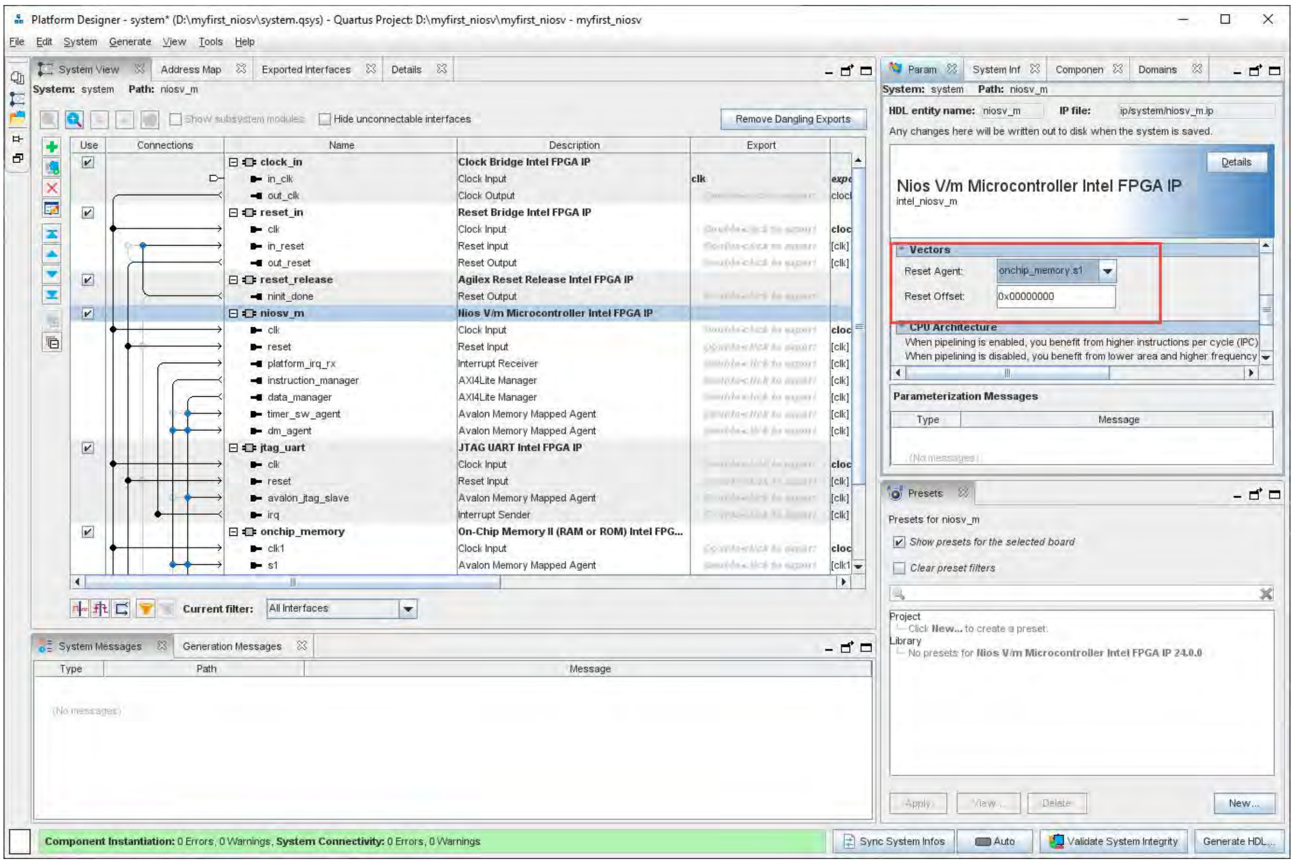
(No messages)

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

Figure 2-25 No errors or warnings

■ Configuring the Reset Vector of the Nios V Processor

2. Click **niosv_m** in the component list on the right part to edit the component. Update **Reset Agent** and **Vector Offset** as shown in Figure 2-26.



Platform Designer - system* (D:\myfirst_niosv\system.qsys) - Quartus Project: D:\myfirst_niosv\myfirst_niosv - myfirst_niosv

System: system Path: niosv_m

Name	Description	Export
clock_in	Clock Bridge Intel FPGA IP	
in_clk	Clock Input	clk
out_clk	Clock Output	
reset_in	Reset Bridge Intel FPGA IP	
clk	Clock Input	
in_reset	Reset Input	
out_reset	Reset Output	
reset_release	Agilex Reset Release Intel FPGA IP	
ninit_done	Reset Output	
niosv_m	Nios V/m Microcontroller Intel FPGA IP	
clk	Clock Input	
reset	Reset Input	
platform_irq_rx	Interrupt Receiver	
instruction_manager	AXI4Lite Manager	
data_manager	AXI4Lite Manager	
timer_sw_agent	Avalon Memory Mapped Agent	
dm_agent	Avalon Memory Mapped Agent	
jtag_uart	JTAG UART Intel FPGA IP	
clk	Clock Input	
reset	Reset Input	
avalon_jtag_slave	Avalon Memory Mapped Agent	
irq	Interrupt Sender	
onchip_memory	On-Chip Memory II (RAM or ROM) Intel FPG...	
clk1	Clock Input	
s1	Avalon Memory Mapped Agent	

HDL entity name: niosv_m IP file: ip/system/niosv_m.ip

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

Nios V/m Microcontroller Intel FPGA IP
intel_niosv_m

Vectors
Reset Agent: onchip_memory.s1
Reset Offset: 0x00000000

CPU Architecture
When pipelining is enabled, you benefit from higher instructions per cycle (IPC)
When pipelining is disabled, you benefit from lower area and higher frequency

Parameterization Messages
(No messages)

Presets for niosv_m
Click New... to create a preset.
No presets for Nios V/m Microcontroller Intel FPGA IP 24.0.0

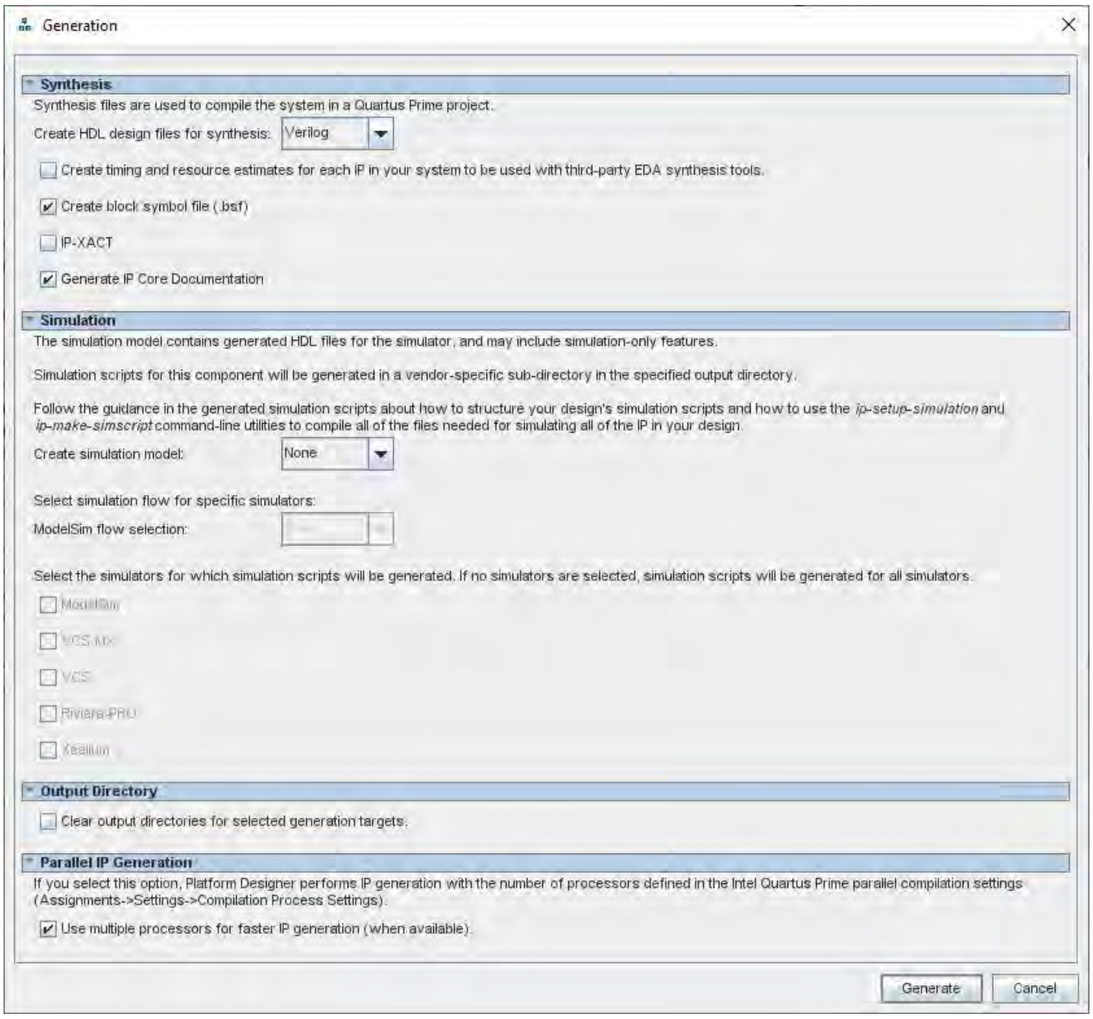
(No messages)

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

Figure 2-26 Update CPU settings

■ Saving and Generating System HDL

3. Click **Generate** > **Generate HDL** and then pop a window as shown in Figure 2-27. Click **Generate** and then pop a window as shown in Figure 2-28. Click **Save** and the generation start.


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:

ModelSim flow selection:

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

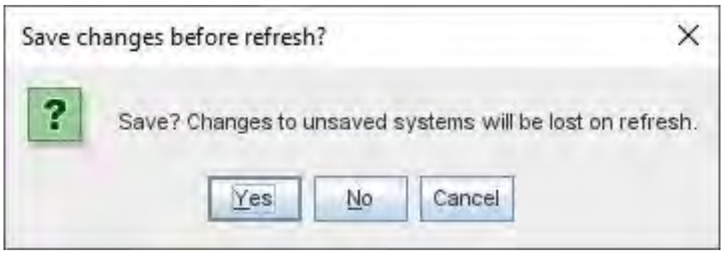
☐ Clear output directories for selected generation targets.

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).

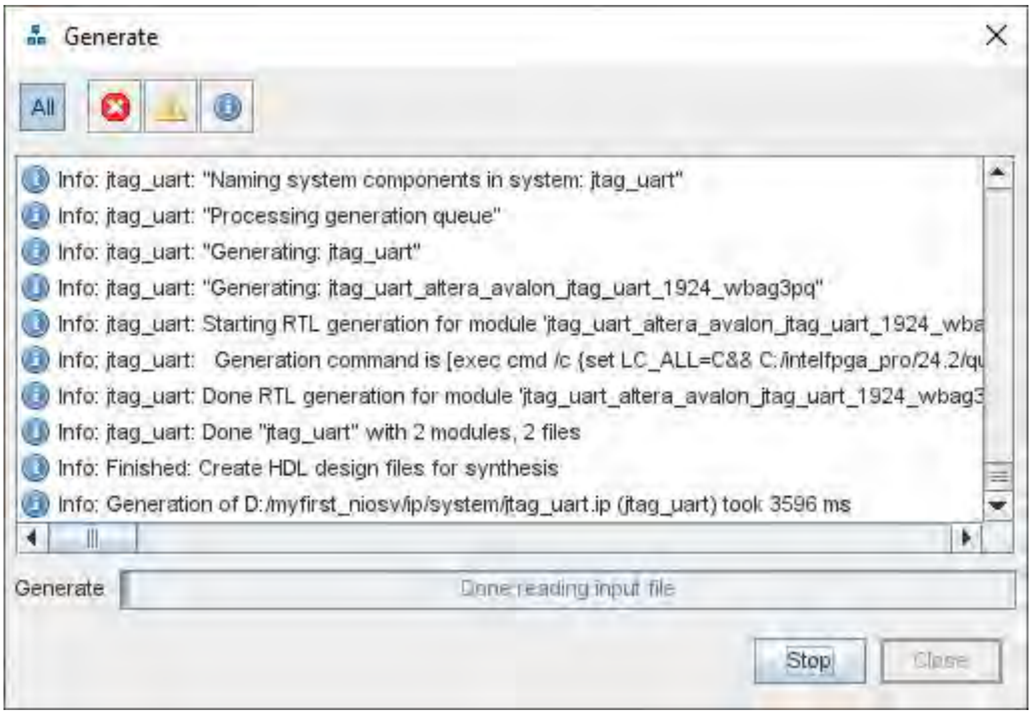
Figure 2-27 Generate Platform Designer


Save changes before refresh?

Save? Changes to unsaved systems will be lost on refresh.

Figure 2-28 Save changes

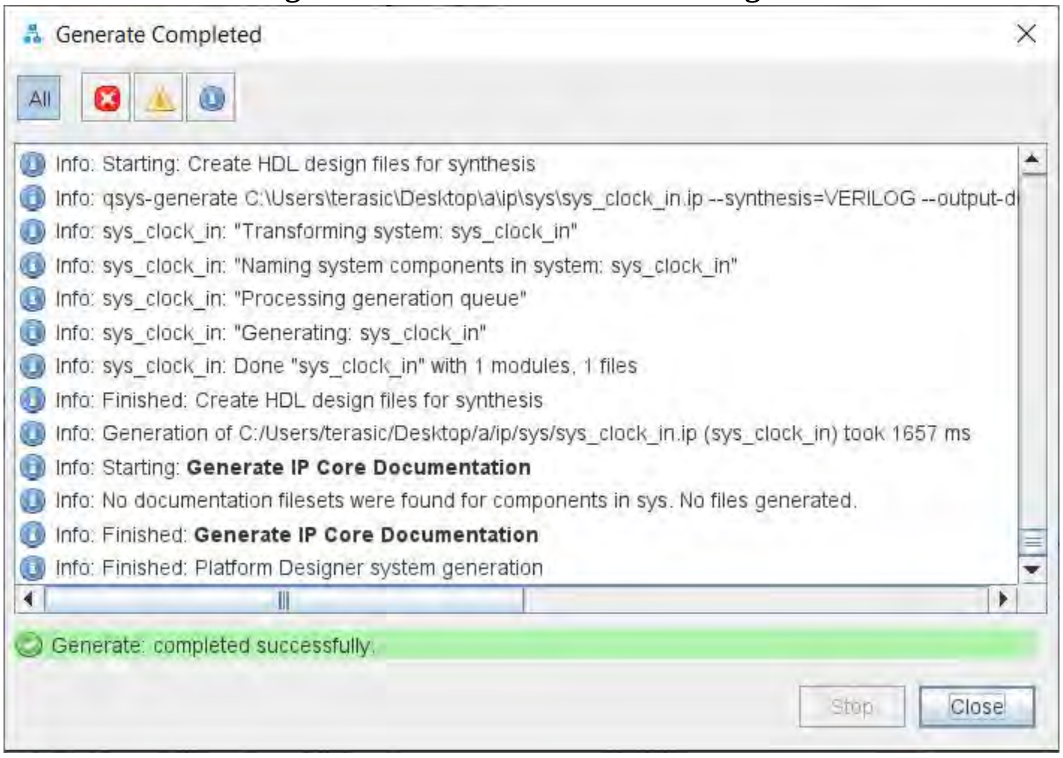
4. Figure 2-29 shows the generate process. If there is no error in the generation, the window will show successful as shown in Figure 2-30.



```
Info: jtag_uart: "Naming system components in system: jtag_uart"
Info: jtag_uart: "Processing generation queue"
Info: jtag_uart: "Generating: jtag_uart"
Info: jtag_uart: "Generating: jtag_uart_altera_avalon_jtag_uart_1924_wbag3pq"
Info: jtag_uart: Starting RTL generation for module 'jtag_uart_altera_avalon_jtag_uart_1924_wba
Info: jtag_uart:   Generation command is [exec cmd /c {set LC_ALL=C&& C:/intelfpga_pro/24.2/qu
Info: jtag_uart: Done RTL generation for module 'jtag_uart_altera_avalon_jtag_uart_1924_wbag3
Info: jtag_uart: Done "jtag_uart" with 2 modules, 2 files
Info: Finished: Create HDL design files for synthesis
Info: Generation of D:/myfirst_niosv/ip/system/jtag_uart.ip (jtag_uart) took 3596 ms
```

Done reading input file

Figure 2-29 Generate Platform Designer



```
Info: Starting: Create HDL design files for synthesis
Info: qsys-generate C:\Users\terasic\Desktop\a\ip\sys\sys_clock_in.ip --synthesis=VERILOG --output-d
Info: sys_clock_in: "Transforming system: sys_clock_in"
Info: sys_clock_in: "Naming system components in system: sys_clock_in"
Info: sys_clock_in: "Processing generation queue"
Info: sys_clock_in: "Generating: sys_clock_in"
Info: sys_clock_in: Done "sys_clock_in" with 1 modules, 1 files
Info: Finished: Create HDL design files for synthesis
Info: Generation of C:/Users/terasic/Desktop/a/ip/sys/sys_clock_in.ip (sys_clock_in) took 1657 ms
Info: Starting: Generate IP Core Documentation
Info: No documentation filesets were found for components in sys. No files generated.
Info: Finished: Generate IP Core Documentation
Info: Finished: Platform Designer system generation
```

Generate: completed successfully.

Figure 2-30 Generate Platform Designer Completely

5. Click **Close** and **File > Exit** to exit the Platform Designer and return to the window

2.1.3 Configuring Assignment and Constraint

1. In **Quartus**, choose **File** > **New** to open new files wizard. See Figure 2-31.

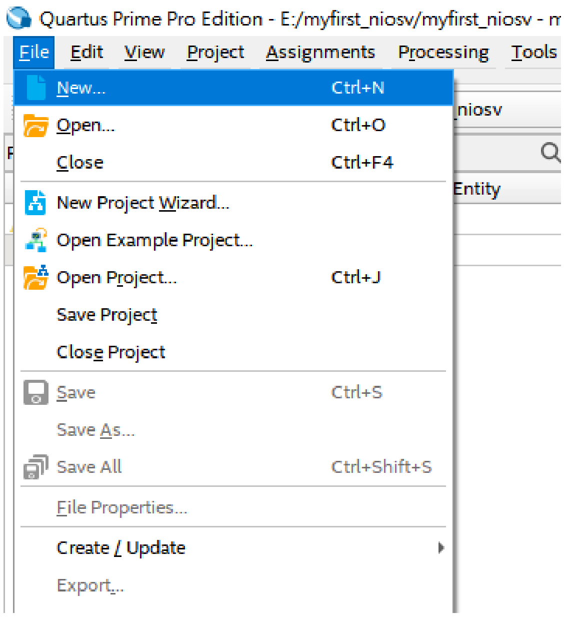


Figure 2-31 New Verilog file

2. Choose **Verilog HDL File** (Figure 2-32.**)** and click **OK** to return to the window as shown in Figure 2-33. Figure 2-33 show a blank verilog file.

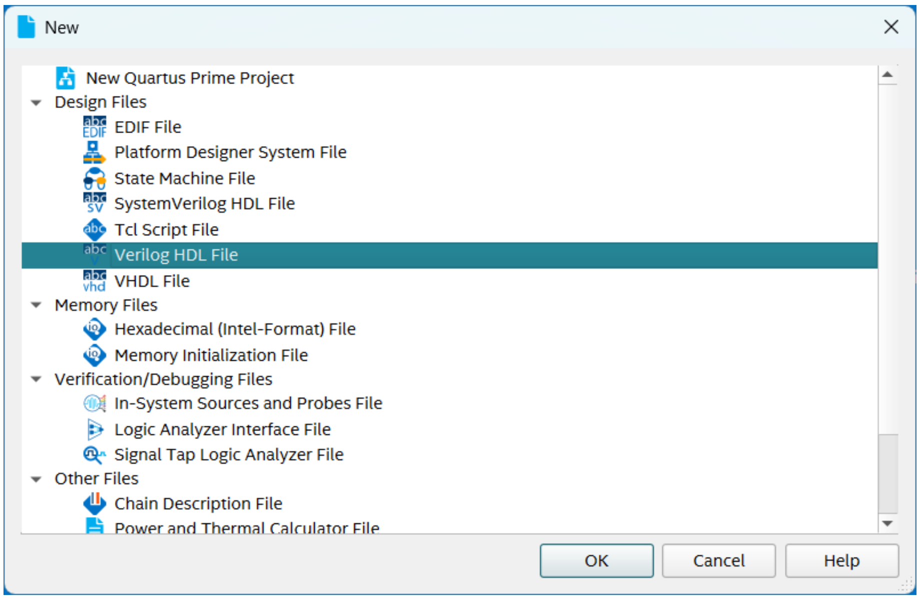

New
New Quartus Prime Project
Design Files
EDIF File
Platform Designer System File
State Machine File
SystemVerilog HDL File
Tcl Script File
Verilog HDL File
VHDL File
Memory Files
Hexadecimal (Intel-Format) File
Memory Initialization File
Verification/Debugging Files
In-System Sources and Probes File
Logic Analyzer Interface File
Signal Tap Logic Analyzer File
Other Files
Chain Description File
Power and Thermal Calculator File
OK
Cancel
Help

Figure 2-32 New Verilog File

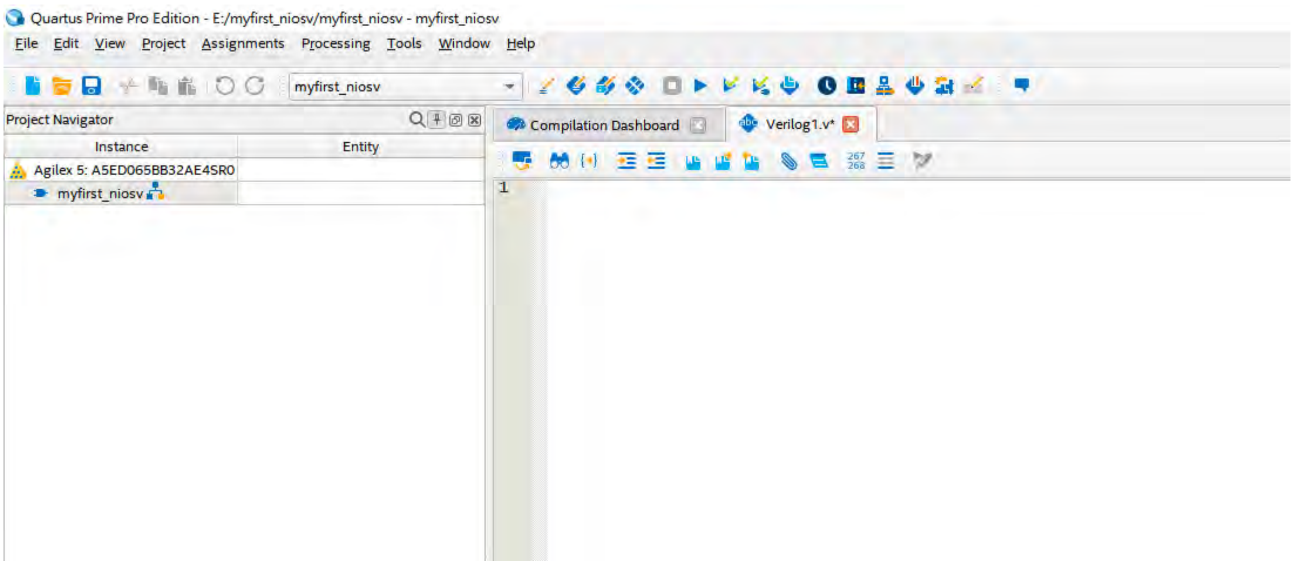
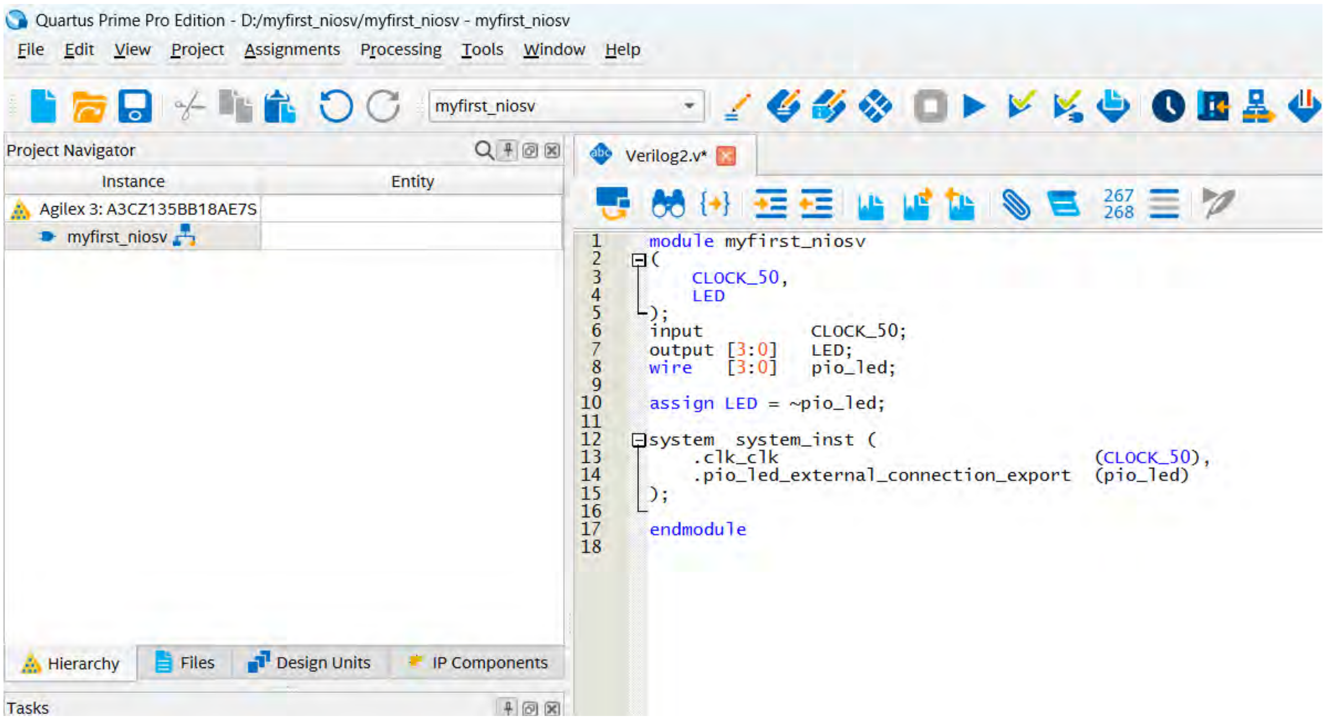

Quartus Prime Pro Edition - E:/myfirst_niosv/myfirst_niosv - myfirst_niosv
File Edit View Project Assignments Processing Tools Window Help
myfirst_niosv
Project Navigator
Instance
Entity
Agilex 5: A5ED065BB32AE4SR0
myfirst_niosv
Compilation Dashboard
Verilog1.v*

Figure 2-33 A blank verilog file

3. Type verilog the following script as shown in Figure 2 44.

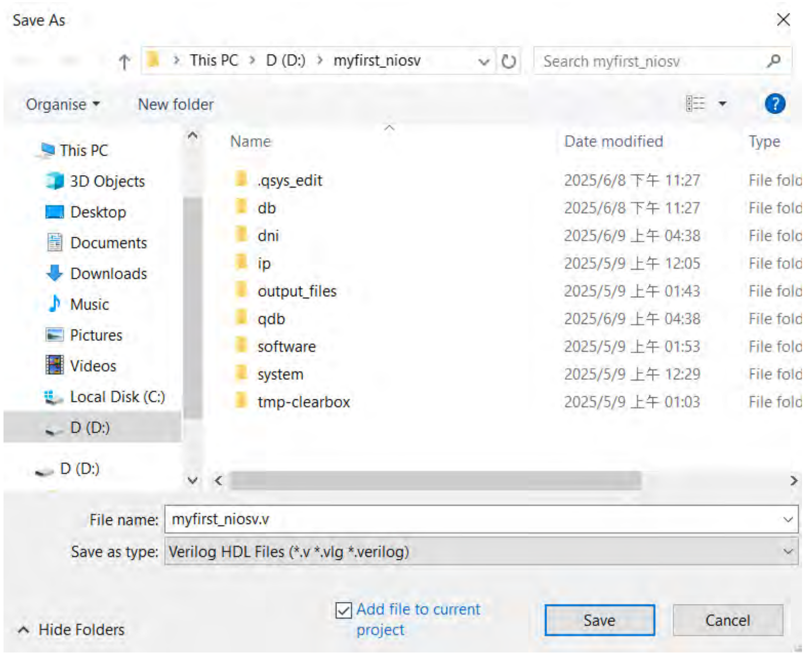
```
module myfirst_niosv
(
    CLOCK_50,
    LED
);
input          CLOCK_50;
output [3:0]    LED;
wire   [3:0]    pio_led;

assign LED = ~pio_led;

system  system_inst (
    .clk_clk                               (CLOCK_50),
    .pio_led_external_connection_export   (pio_led)
);

endmodule
```


Quartus Prime Pro Edition - D:/myfirst_niosv/myfirst_niosv - myfirst_niosv
File Edit View Project Assignments Processing Tools Window Help
myfirst_niosv
Project Navigator
Instance
Entity
Agilex 3: A3CZ135BB18AE7S
myfirst_niosv
Hierarchy
Files
Design Units
IP Components
Tasks
Verilog2.v*

```
module myfirst_niosv
(
      CLOCK_50,
      LED
);
input           CLOCK_50;
output  [3:0]   LED;
wire    [3:0]   pio_led;

assign LED = ~pio_led;

system  system_inst (
      .clk_clk                              (CLOCK_50),
      .pio_led_external_connection_export   (pio_led)
);

endmodule
```

Figure 2-34 Input verilog Text

4. Choose **Save** Icon in the tool bar. There will appear a window as shown in Figure 2-35. Makesure the file name is **myfirst_niosv.v** and click **Save.**



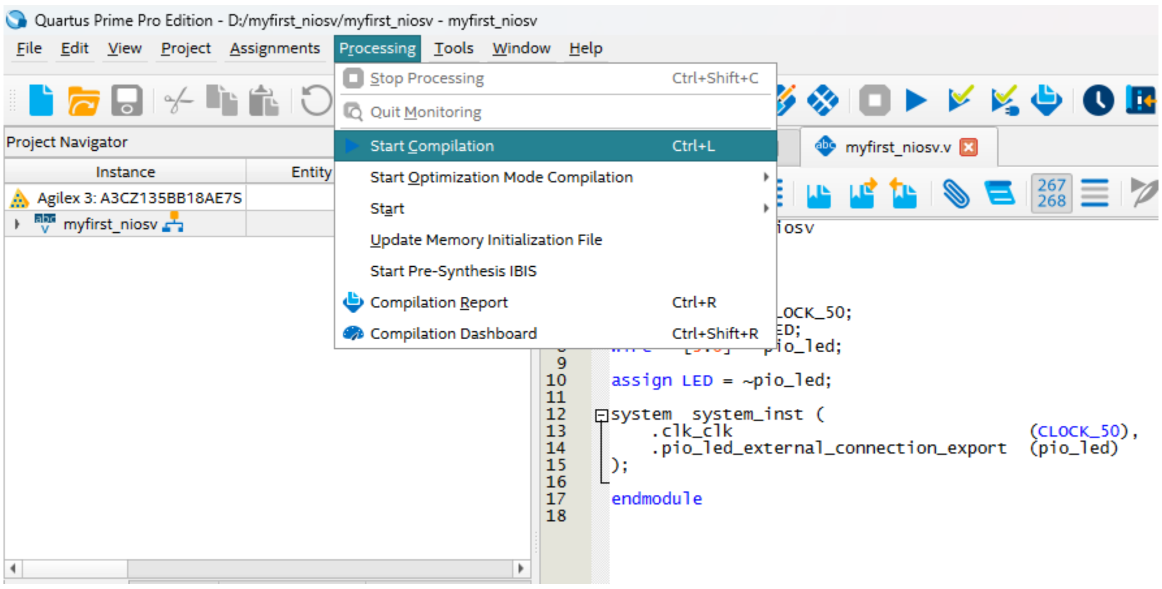
Name	Date modified	Type
.qsys_edit	2025/6/8 下午 11:27	File fold
db	2025/6/8 下午 11:27	File fold
dni	2025/6/9 上午 04:38	File fold
ip	2025/5/9 上午 12:05	File fold
output_files	2025/5/9 上午 01:43	File fold
qdb	2025/6/9 上午 04:38	File fold
software	2025/5/9 上午 01:53	File fold
system	2025/5/9 上午 12:29	File fold
tmp-clearbox	2025/5/9 上午 01:03	File fold

File name: myfirst_niosv.v
Save as type: Verilog HDL Files (*.v *.vlg *.verilog)

Figure 2-35 Save Verilog file

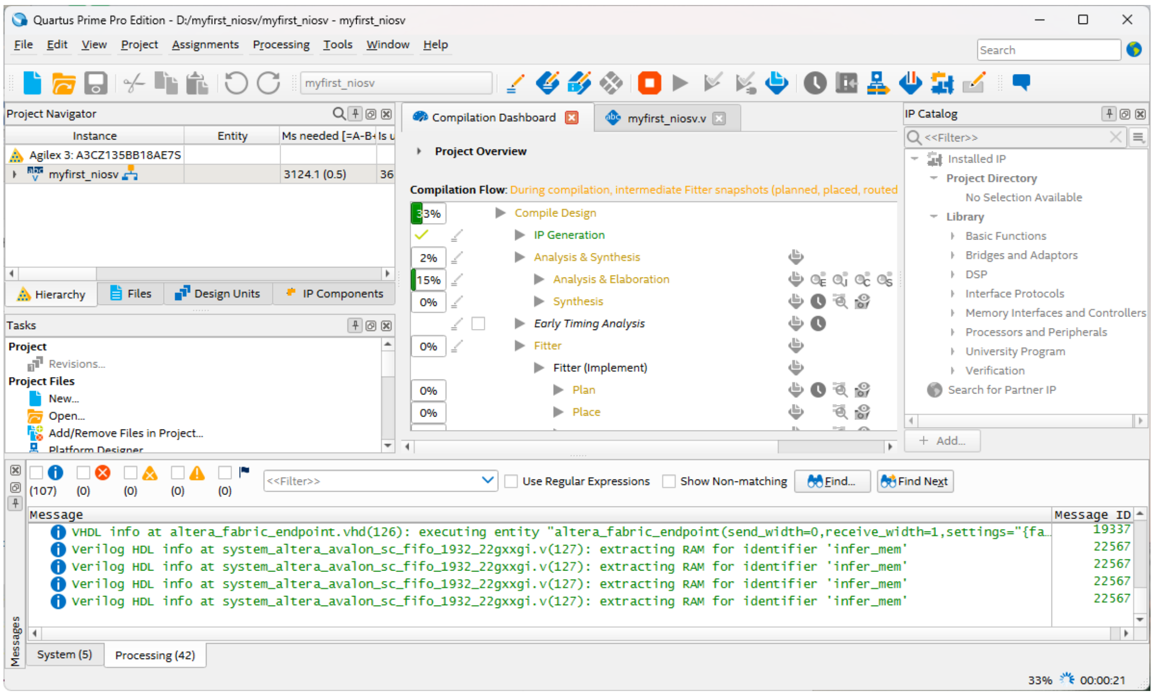
2.1.4 Compiling the Quartus Prime Software Project

1. Choose **Processing** > **Start Compilation** as shown in Figure 2-36. Figure 2-37 shows the compilation process.


Quartus Prime Pro Edition - D:/myfirst_niosv/myfirst_niosv - myfirst_niosv
File
Edit
View
Project
Assignments
Processing
Tools
Window
Help
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
Project Navigator
Instance
Entity
Agilex 3: A3CZ135BB18AE7S
myfirst_niosv
myfirst_niosv.v

```
assign LED = ~pio_led;

system  system_inst (
      .clk_clk                                 (CLOCK_50),
      .pio_led_external_connection_export      (pio_led)
);

endmodule
```

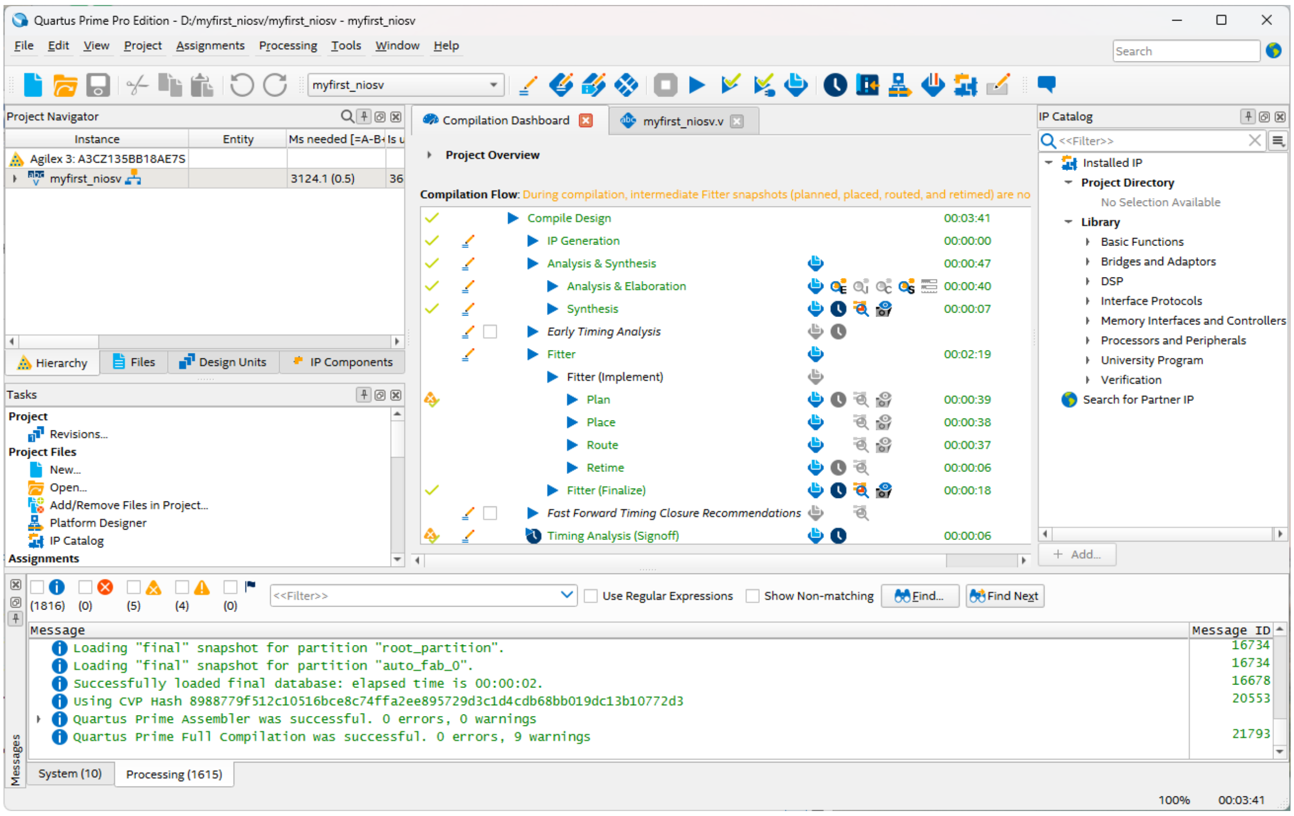
Figure 2-36 Start Compilation



```
Message                                                                                                              Message ID
VHDL info at altera_fabric_endpoint.vhd(126): executing entity "altera_fabric_endpoint(send_width=0,receive_width=1,settings="{fa…   19337
Verilog HDL info at system_altera_avalon_sc_fifo_1932_22gxxgi.v(127): extracting RAM for identifier 'infer_mem'          22567
Verilog HDL info at system_altera_avalon_sc_fifo_1932_22gxxgi.v(127): extracting RAM for identifier 'infer_mem'          22567
Verilog HDL info at system_altera_avalon_sc_fifo_1932_22gxxgi.v(127): extracting RAM for identifier 'infer_mem'          22567
Verilog HDL info at system_altera_avalon_sc_fifo_1932_22gxxgi.v(127): extracting RAM for identifier 'infer_mem'          22567
```

Figure 2-37 Execute Compilation

2. A window that shows successfully will appear as shown in Figure 2-38.

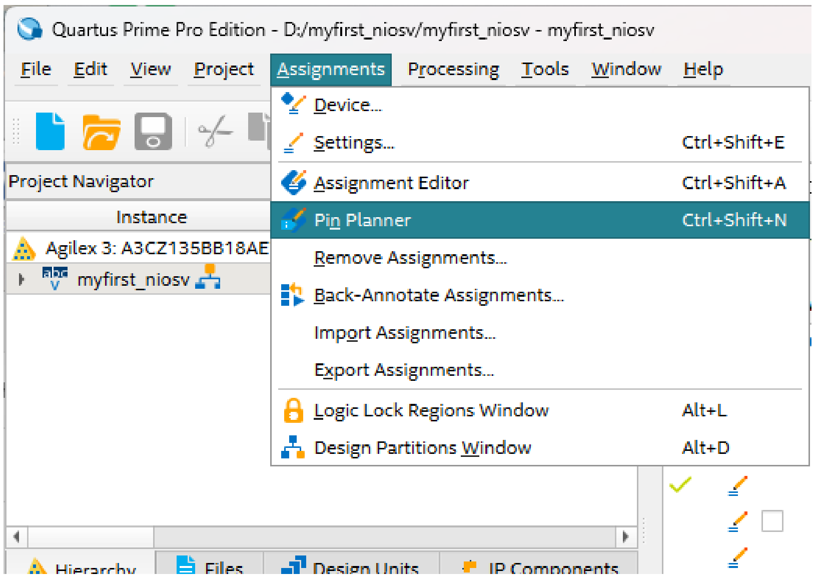


```
Message                                                                                  Message ID
  Loading "final" snapshot for partition "root_partition".                                  16734
  Loading "final" snapshot for partition "auto_fab_0".                                      16734
  Successfully loaded final database: elapsed time is 00:00:02.                             16678
  Using CVP Hash 8988779f512c10516bce8c74ffa2ee895729d3c1d4cdb68bb019dc13b10772d3           20553
▸ Quartus Prime Assembler was successful. 0 errors, 0 warnings
  Quartus Prime Full Compilation was successful. 0 errors, 9 warnings                       21793
```

Quartus Prime Pro Edition - D:/myfirst_niosv/myfirst_niosv - myfirst_niosv
Compilation Dashboard
Compile Design 00:03:41
IP Generation 00:00:00
Analysis & Synthesis 00:00:47
Analysis & Elaboration 00:00:40
Synthesis 00:00:07
Early Timing Analysis
Fitter 00:02:19
Plan 00:00:39
Place 00:00:38
Route 00:00:37
Retime 00:00:06
Fitter (Finalize) 00:00:18
Fast Forward Timing Closure Recommendations
Timing Analysis (Signoff) 00:00:06

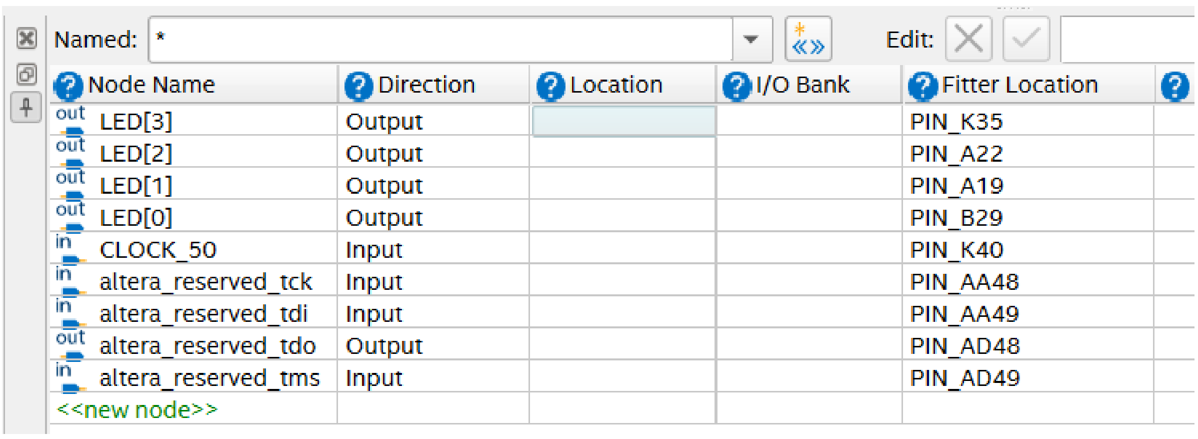
Figure 2-38 Compilation project completely

3. Choose **Assignments** > **Pin Planner** to open pin planner as shown in Figure 2-39. Figure 2-40 show blank pins.



Quartus Prime Pro Edition - D:/myfirst_niosv/myfirst_niosv - myfirst_niosv
File Edit View Project Assignments Processing Tools Window Help
Device...
Settings... Ctrl+Shift+E
Assignment Editor Ctrl+Shift+A
Pin Planner Ctrl+Shift+N
Remove Assignments...
Back-Annotate Assignments...
Import Assignments...
Export Assignments...
Logic Lock Regions Window Alt+L
Design Partitions Window Alt+D
Project Navigator
Instance
Agilex 3: A3CZ135BB18AE
myfirst_niosv
Hierarchy Files Design Units IP Components

Figure 2-39 Pins menu



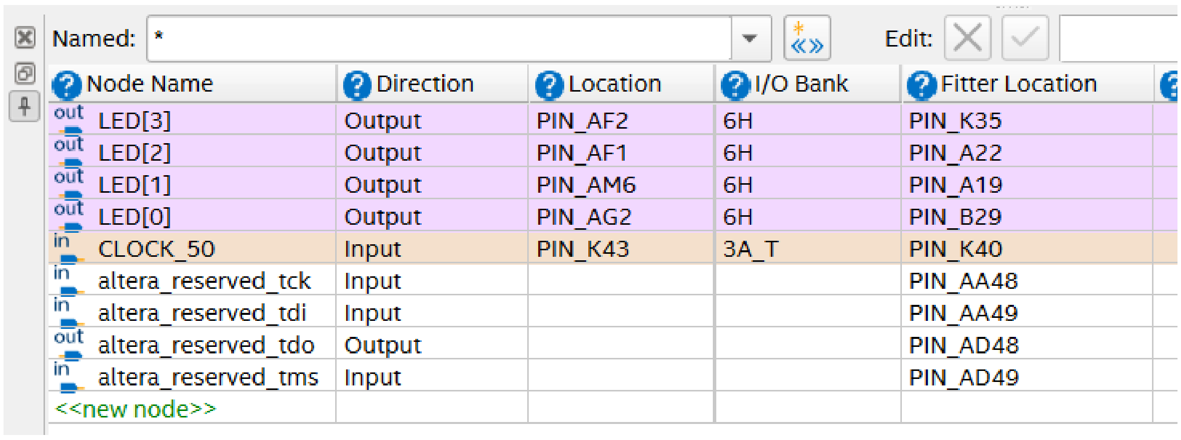
Named: *

Edit:

Node Name	Direction	Location	I/O Bank	Fitter Location
LED[3]	Output			PIN_K35
LED[2]	Output			PIN_A22
LED[1]	Output			PIN_A19
LED[0]	Output			PIN_B29
CLOCK_50	Input			PIN_K40
altera_reserved_tck	Input			PIN_AA48
altera_reserved_tdi	Input			PIN_AA49
altera_reserved_tdo	Output			PIN_AD48
altera_reserved_tms	Input			PIN_AD49
<<new node>>				

Figure 2-40 Blank Pins

4. Input Location value as shown in Figure 2-41.



Named: *

Edit:

Node Name	Direction	Location	I/O Bank	Fitter Location
LED[3]	Output	PIN_AF2	6H	PIN_K35
LED[2]	Output	PIN_AF1	6H	PIN_A22
LED[1]	Output	PIN_AM6	6H	PIN_A19
LED[0]	Output	PIN_AG2	6H	PIN_B29
CLOCK_50	Input	PIN_K43	3A_T	PIN_K40
altera_reserved_tck	Input			PIN_AA48
altera_reserved_tdi	Input			PIN_AA49
altera_reserved_tdo	Output			PIN_AD48
altera_reserved_tms	Input			PIN_AD49
<<new node>>				

Figure 2-41 Set Pins

5. Close the P**in Planner**.
6. **Restart compilation the project**.

2.2 Download Hardware Design to Target FPGA

This section describes how to download the configuration file to the board.

Download the FPGA configuration file (i.e. the SRAM Object File (.sof) that contains the NIOS V standard system) to the board by performing the following steps:

1. Connect the board to the host computer via the USB download cable.

2. Apply power to the board.

5. Choose **Tools** -> **Programmer** in Quartus.

6. Click **Hardware Setup** in the top, left comer of the Programmer window. The Hardware Setup dialog box appears.

7. Select **Atum A3 Nano** from the **Currently selected hardware** drop-down list box.

Note: If the appropriate download cable does not appear in the list, you must first install drivers for the cable. Refer to Quartus Prime Pro Help for information on how to install the driver. See Figure 2-42.

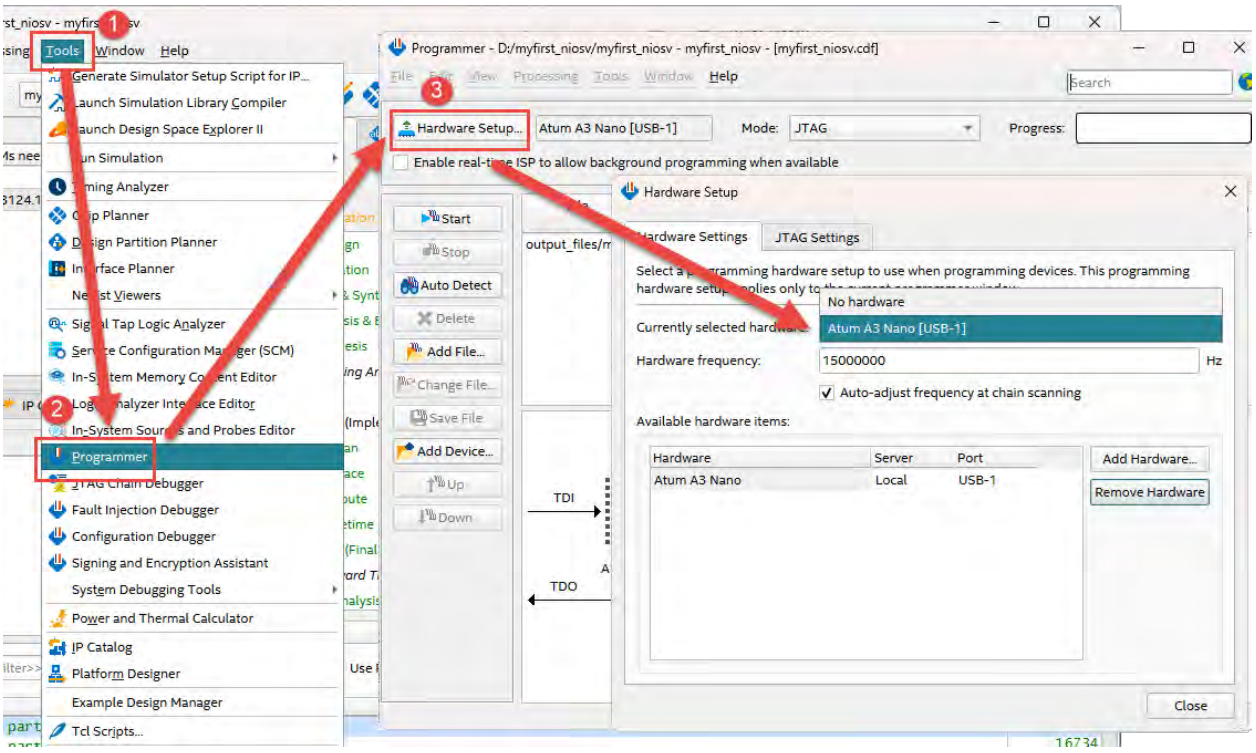
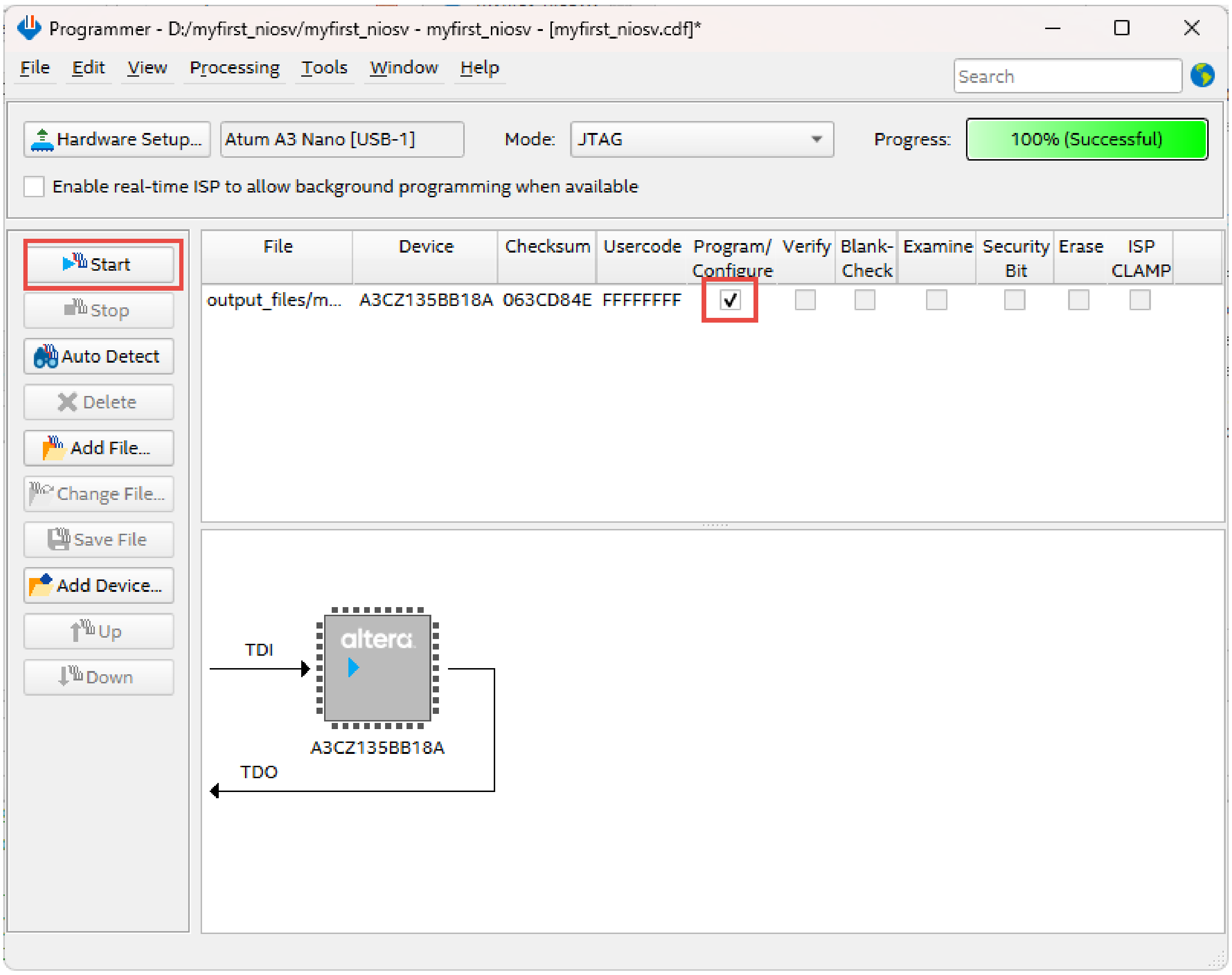


Figure 2-42 Hardware Setup Window

15. Turn on the **Program/Configure** option for the programming file. (See Figure 2-43 for an example).

16. Click **Start**.



Hardware: Atum A3 Nano [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
output_files/m...	A3CZ135BB18A	063CD84E	FFFFFFFF	✓	☐	☐	☐	☐	☐	☐

Figure 2-43 Programmer

The Progress meter sweeps to 100% after the configuration finished. When configuration is complete, the FPGA is configured with the Nios V system, but it does not yet have a C program in memory to execute.

Chapter 3

Ashling RiscFree IDE Build Flow

This chapter covers the build process for C software programs targeting the Nios V processor. The Ashling RiscFree IDE for Altera FPGAs provides an easy-to-use graphical user interface (GUI) that automates the build process and the management of CMakeLists.txt files. The IDE also integrates a text editor and a debugger.

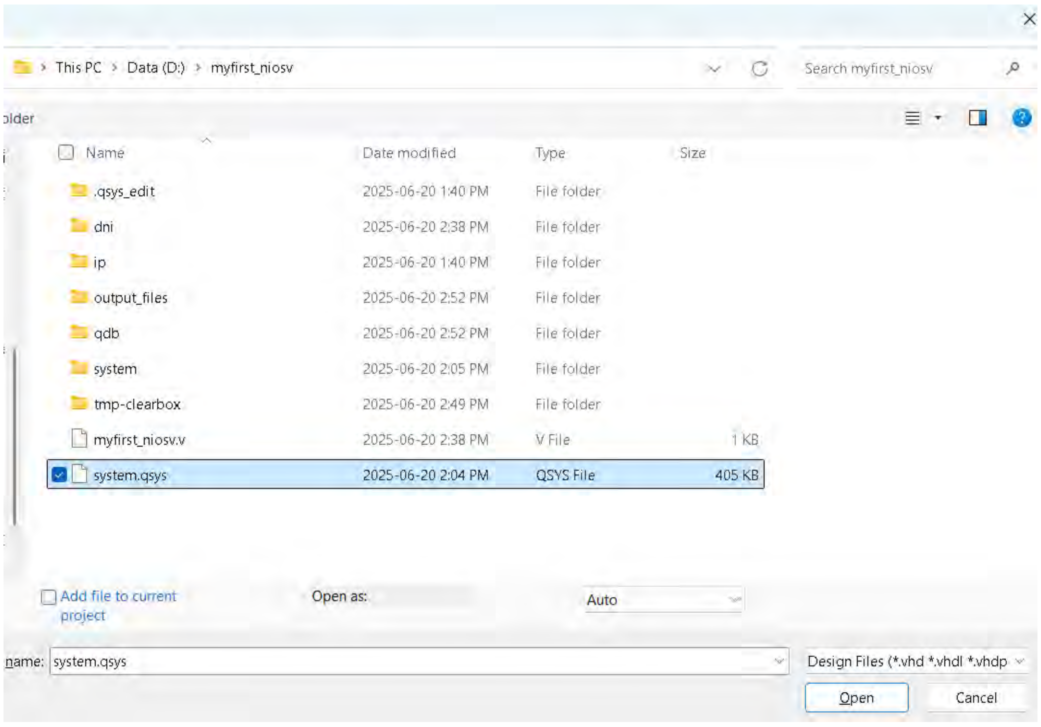
In this chapter, you will use the Ashling RiscFree IDE for Altera FPGAs to compile a simple C example program, which will run on a standard Nios V system configured on the FPGA of your development board. You will create a new software project, build it, and execute it on the target hardware. You will also learn to edit the project, rebuild it, and initiate a debugging session.

3.1 Create the hello_world Example Project

In this section you will create a new NIOS V C/C++ application project. To begin, perform the following steps:

■ **Creating a Board Support Package**

1. In Quartus Prime, select **File** > **Open** to open system.qsys, as shown in the figure below.



Name	Date modified	Type	Size
.qsys_edit	2025-06-20 1:40 PM	File folder	
dni	2025-06-20 2:38 PM	File folder	
ip	2025-06-20 1:40 PM	File folder	
output_files	2025-06-20 2:52 PM	File folder	
qdb	2025-06-20 2:52 PM	File folder	
system	2025-06-20 2:05 PM	File folder	
tmp-clearbox	2025-06-20 2:49 PM	File folder	
myfirst_niosv.v	2025-06-20 2:38 PM	V File	1 KB
system.qsys	2025-06-20 2:04 PM	QSYS File	405 KB

name: system.qsys

Design Files (*.vhd *.vhdl *.vhdp

Figure 3-1 Open .qsys

2. Click File > New BSP to create a new BSP, as shown in the figure below.

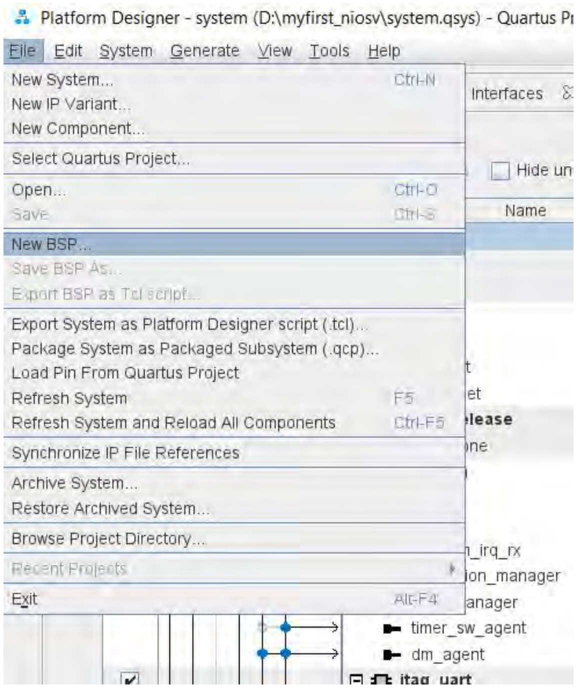
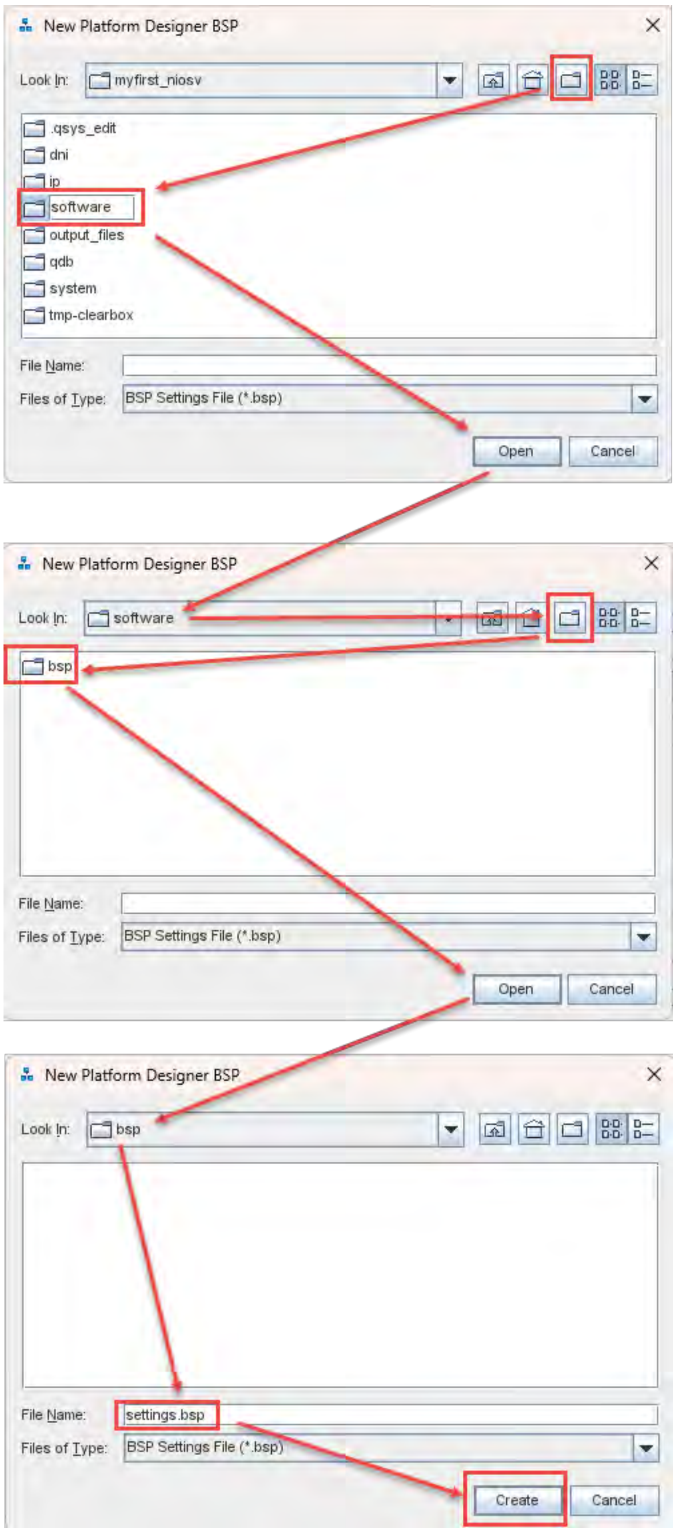

Platform Designer - system (D:\myfirst_niosv\system.qsys) - Quartus Pr
File
Edit
System
Generate
View
Tools
Help
New System... Ctrl-N
New IP Variant...
New Component...
Select Quartus Project...
Open... Ctrl-O
Save Ctrl-S
New BSP...
Save BSP As...
Export BSP as Tcl script...
Export System as Platform Designer script (.tcl)...
Package System as Packaged Subsystem (.qcp)...
Load Pin From Quartus Project
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
Interfaces
Hide un
Name
timer_sw_agent
dm_agent
jtag_uart

Figure 3-2 Creating a new bsp

3. Create a **software** folder and a **bsp** subfolder within it. Name the file **settings.bsp** and click Create, as shown below.



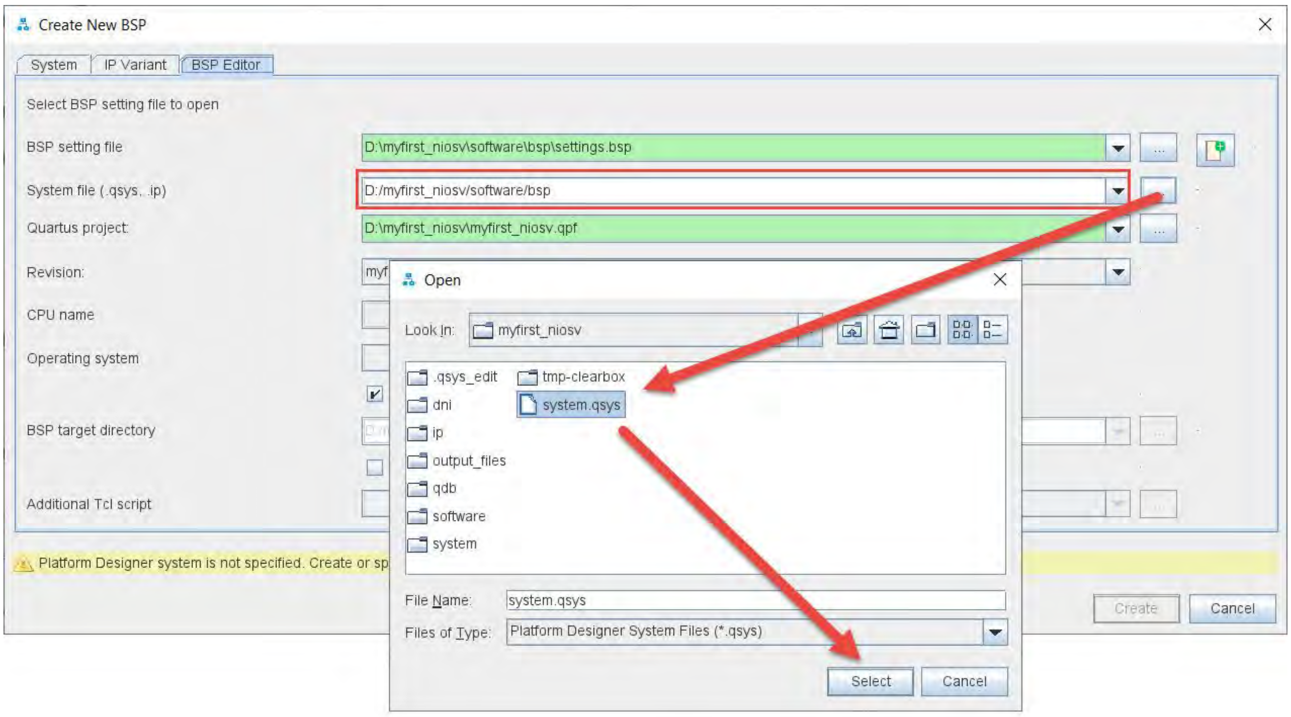
Look In: myfirst_niosv
File Name:
Files of Type: BSP Settings File (*.bsp)

Look In: software
File Name:
Files of Type: BSP Settings File (*.bsp)

Look In: bsp
File Name: settings.bsp
Files of Type: BSP Settings File (*.bsp)

Figure 3-3 Creating a new bsp

4. In the **System File** field, browse to and select **system.qsys**, as shown below.


Create New BSP

System | IP Variant | BSP Editor

Select BSP setting file to open

BSP setting file: D:\myfirst_niosv\software\bsp\settings.bsp

System file (.qsys, .ip): D:/myfirst_niosv/software/bsp

Quartus project: D:\myfirst_niosv\myfirst_niosv.qpf

Revision: myf

CPU name

Operating system

BSP target directory

Additional Tcl script

Platform Designer system is not specified. Create or sp

Open

Look In: myfirst_niosv

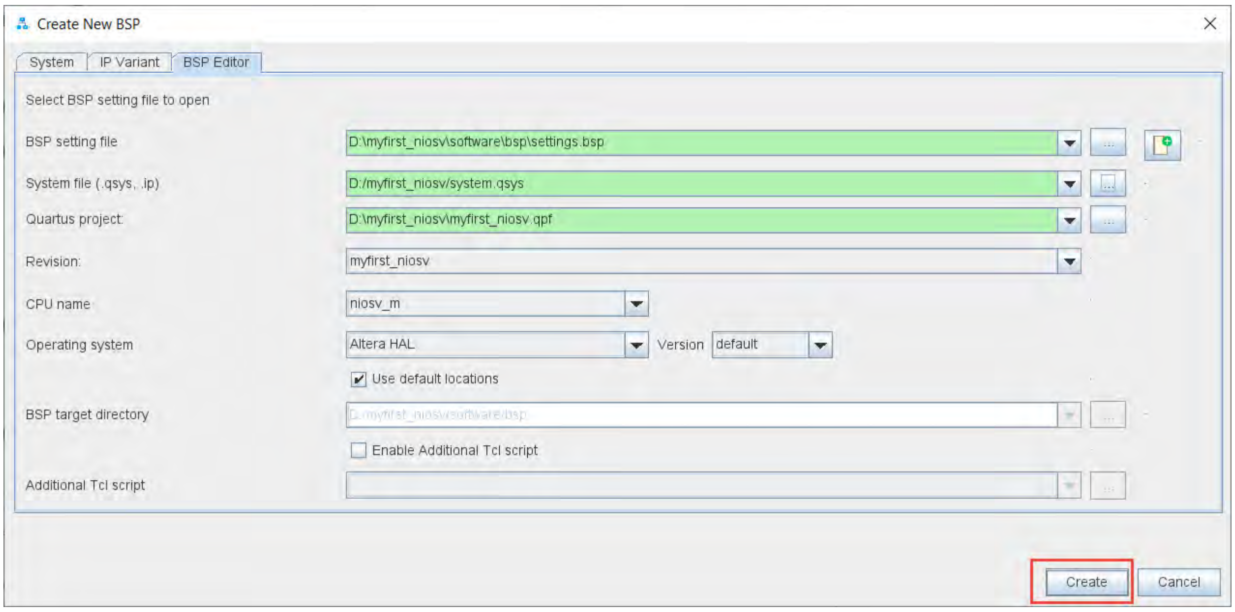
.qsys_edit, tmp-clearbox, dni, system.qsys, ip, output_files, qdb, software, system

File Name: system.qsys

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

Figure 3-4 Select system file

5. Click **Create** to generate the BSP.


Create New BSP

System | IP Variant | BSP Editor

Select BSP setting file to open

BSP setting file: D:\myfirst_niosv\software\bsp\settings.bsp

System file (.qsys, .ip): D:/myfirst_niosv/system.qsys

Quartus project: D:\myfirst_niosv\myfirst_niosv.qpf

Revision: myfirst_niosv

CPU name: niosv_m

Operating system: Altera HAL Version: default

☑ Use default locations

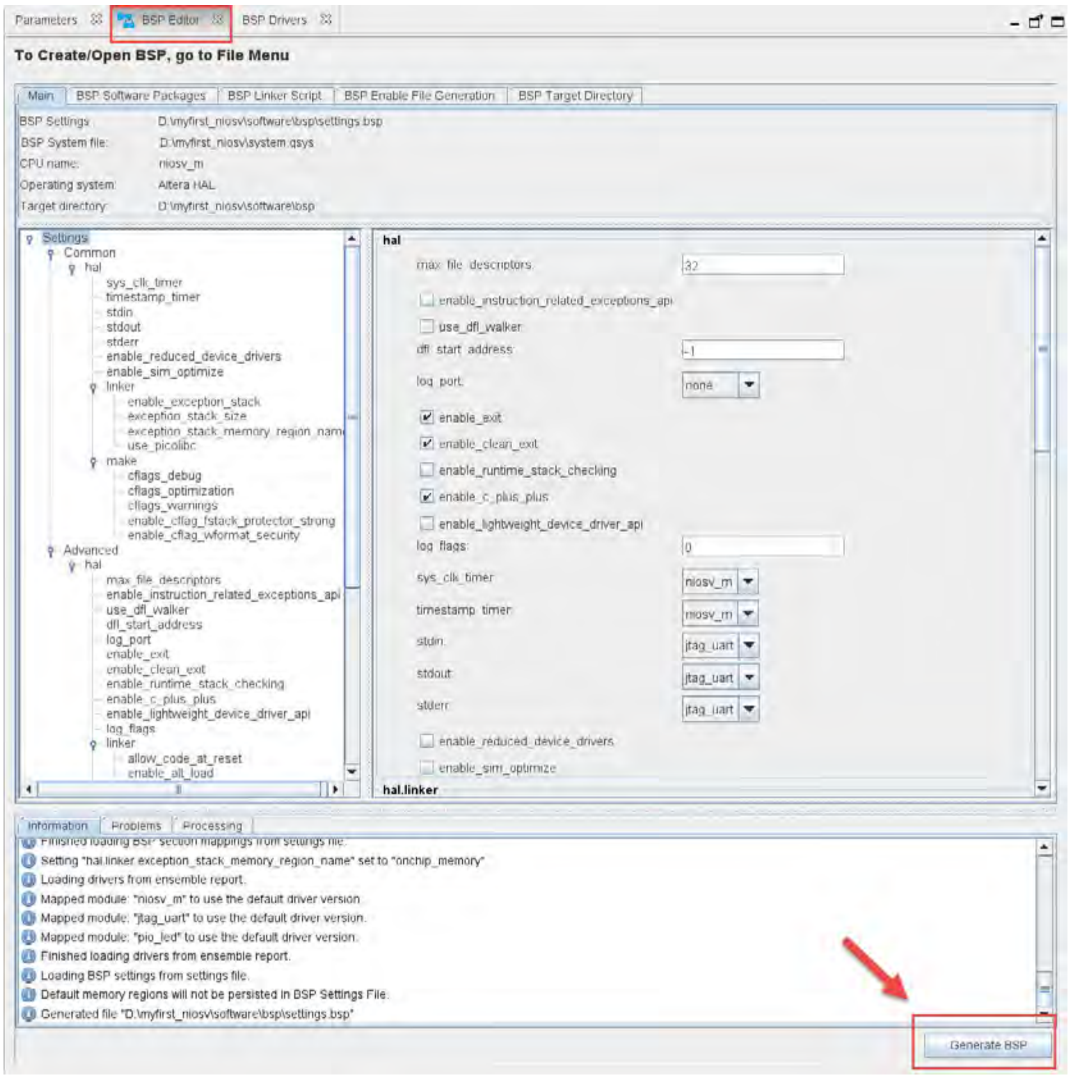
BSP target directory: D:/myfirst_niosv/software/bsp

☐ Enable Additional Tcl script

Additional Tcl script

Figure 3-5 Creating bsp

6. Double-click the **BSP Editor** tab to maximize it, as shown below. Click **Generate BSP.**



To Create/Open BSP, go to File Menu

BSP Settings:	D:\myfirst_niosv\software\bsp\settings.bsp
BSP System file:	D:\myfirst_niosv\system.qsys
CPU name:	niosv_m
Operating system:	Altera HAL
Target directory:	D:\myfirst_niosv\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_nam
        use_picolibc
      make
        cflags_debug
        cflags_optimization
        cflags_warnings
        enable_cflag_fstack_protector_strong
        enable_cflag_wformat_security
  Advanced
    hal
      max_file_descriptors
      enable_instruction_related_exceptions_api
      use_dfl_walker
      dfl_start_address
      log_port
      enable_exit
      enable_clean_exit
      enable_runtime_stack_checking
      enable_c_plus_plus
      enable_lightweight_device_driver_api
      log_flags
      linker
        allow_code_at_reset
        enable_alt_load
```

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	
log_flags	0
sys_clk_timer	niosv_m
timestamp_timer	niosv_m
stdin	jtag_uart
stdout	jtag_uart
stderr	jtag_uart
☐ enable_reduced_device_drivers	
☐ enable_sim_optimize	

hal.linker

```
Finished loading BSP section mappings from settings file.
Setting "hal.linker.exception_stack_memory_region_name" set to "onchip_memory"
Loading drivers from ensemble report.
Mapped module: "niosv_m" to use the default driver version.
Mapped module: "jtag_uart" to use the default driver version.
Mapped module: "pio_led" to use the default driver version.
Finished loading drivers from ensemble report.
Loading BSP settings from settings file.
Default memory regions will not be persisted in BSP Settings File.
Generated file "D:\myfirst_niosv\software\bsp\settings.bsp"
```

Figure 3-6 Creating bsp

7. Launch **Nios V Command Shell** from the Start Menu.

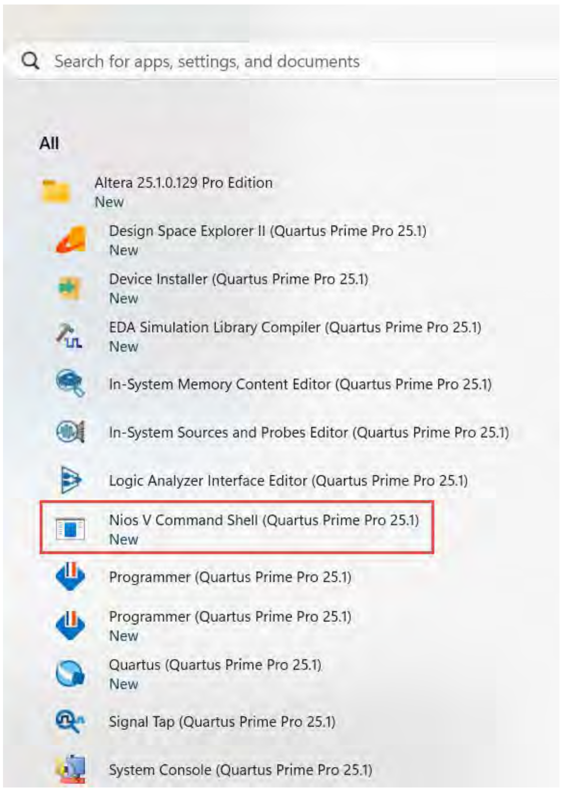
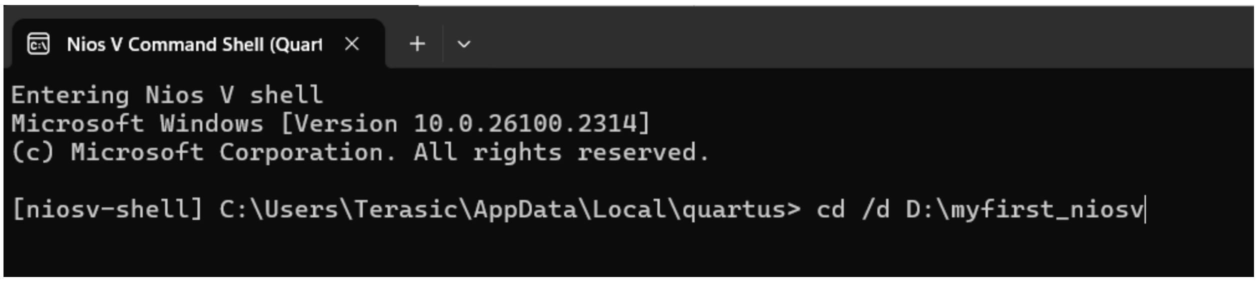

Search for apps, settings, and documents
All
Altera 25.1.0.129 Pro Edition
New
Design Space Explorer II (Quartus Prime Pro 25.1)
New
Device Installer (Quartus Prime Pro 25.1)
New
EDA Simulation Library Compiler (Quartus Prime Pro 25.1)
New
In-System Memory Content Editor (Quartus Prime Pro 25.1)
In-System Sources and Probes Editor (Quartus Prime Pro 25.1)
Logic Analyzer Interface Editor (Quartus Prime Pro 25.1)
Nios V Command Shell (Quartus Prime Pro 25.1)
New
Programmer (Quartus Prime Pro 25.1)
Programmer (Quartus Prime Pro 25.1)
New
Quartus (Quartus Prime Pro 25.1)
New
Signal Tap (Quartus Prime Pro 25.1)
System Console (Quartus Prime Pro 25.1)

Figure 3-7 Launch Nios V Command Shell

8. Run the command below to switch to the working directory (see Figure 3-8).

```
cd /d <myfirst_niosv path>
```



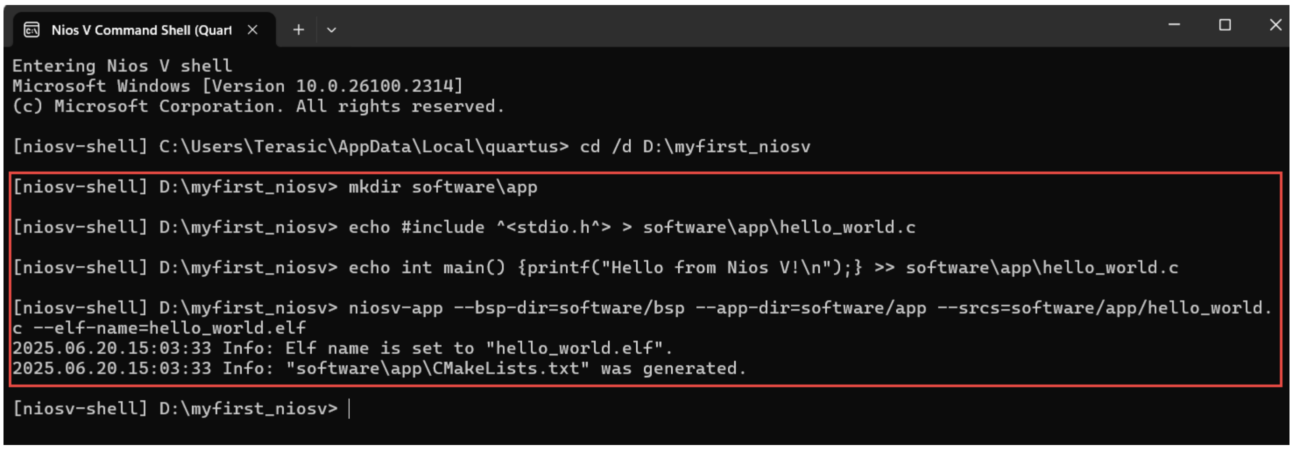
```
Entering Nios V shell
Microsoft Windows [Version 10.0.26100.2314]
(c) Microsoft Corporation. All rights reserved.

[niosv-shell] C:\Users\Terasic\AppData\Local\quartus> cd /d D:\myfirst_niosv
```

Figure 3-8 Switch working directory

9. Run the commands shown to create the application (see Figure 3-9).

```
mkdir software\app

echo #include ^<stdio.h^> > software\app\hello_world.c

echo int main() {printf("Hello from Nios V!\n");} >> software\app\hello_world.c
```

```
niosv-app --bsp-dir=software/bsp --app-dir=software/app --srcs=software/app/hello_world.c --elf-name=hello_world.elf
```



```
Entering Nios V shell
Microsoft Windows [Version 10.0.26100.2314]
(c) Microsoft Corporation. All rights reserved.

[niosv-shell] C:\Users\Terasic\AppData\Local\quartus> cd /d D:\myfirst_niosv

[niosv-shell] D:\myfirst_niosv> mkdir software\app

[niosv-shell] D:\myfirst_niosv> echo #include ^<stdio.h^> > software\app\hello_world.c

[niosv-shell] D:\myfirst_niosv> echo int main() {printf("Hello from Nios V!\n");} >> software\app\hello_world.c

[niosv-shell] D:\myfirst_niosv> niosv-app --bsp-dir=software/bsp --app-dir=software/app --srcs=software/app/hello_world.
c --elf-name=hello_world.elf
2025.06.20.15:03:33 Info: Elf name is set to "hello_world.elf".
2025.06.20.15:03:33 Info: "software\app\CMakeLists.txt" was generated.

[niosv-shell] D:\myfirst_niosv> |
```

Figure 3-9 Creating application

■ Importing Projects into Ashling RiscFree IDE for Altera FPGAs

1. User need to install WSL in your windows system.
2. Launch **Ashling RiscFree IDE** for Intel FPGAs from the Start Menu.

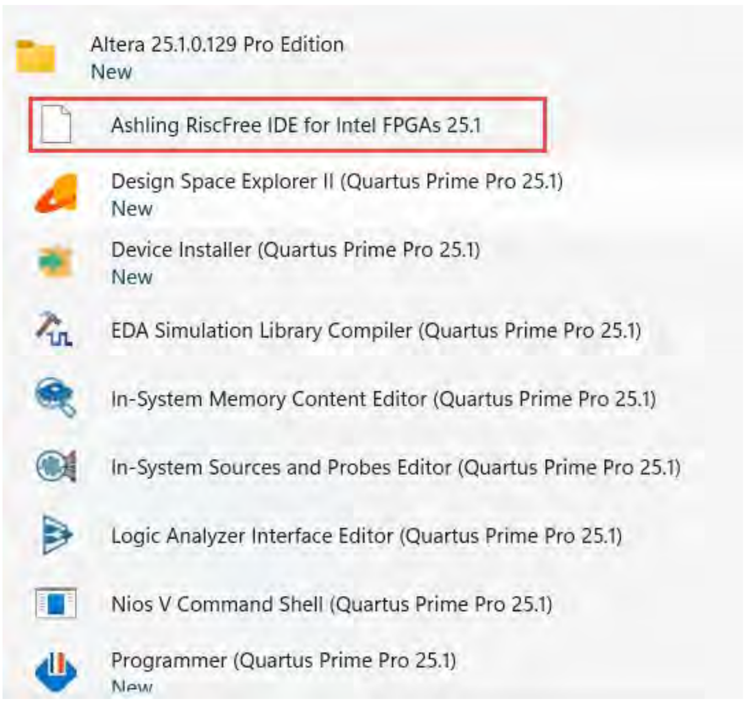
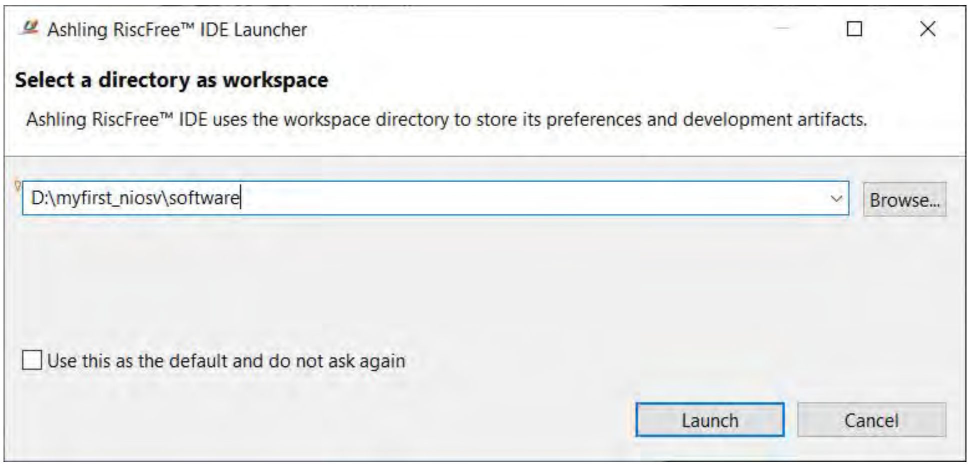

Altera 25.1.0.129 Pro Edition
New
Ashling RiscFree IDE for Intel FPGAs 25.1
Design Space Explorer II (Quartus Prime Pro 25.1)
New
Device Installer (Quartus Prime Pro 25.1)
New
EDA Simulation Library Compiler (Quartus Prime Pro 25.1)
In-System Memory Content Editor (Quartus Prime Pro 25.1)
In-System Sources and Probes Editor (Quartus Prime Pro 25.1)
Logic Analyzer Interface Editor (Quartus Prime Pro 25.1)
Nios V Command Shell (Quartus Prime Pro 25.1)
Programmer (Quartus Prime Pro 25.1)
New

Figure 3-10 Select Workspace

Note: If the user does not see the Ashling RiscFree IDE Shortcut in the Quartus 25.1 Pro folder of the Windows Start Menu, please refer to the link below to resolve the issue : Resolving the Missing

Ashling RiscFree IDE Shortcut in the Quartus 25.1 Pro Start Menu

3. Then click Launch Set the workspace as shown in Figure 3-11, then click Launch.


Ashling RiscFree™ IDE Launcher
Select a directory as workspace
Browse...
Launch
Cancel

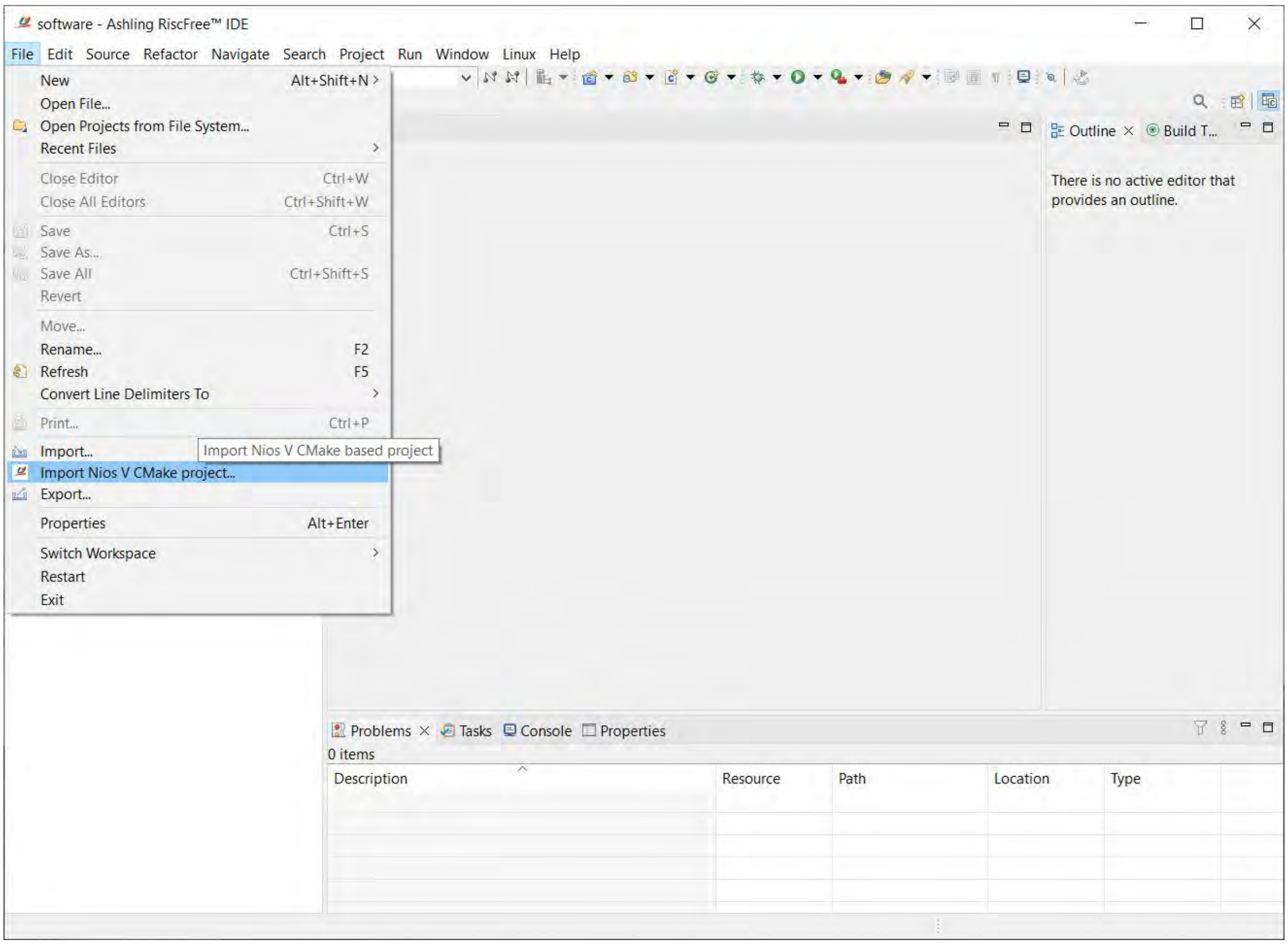
Ashling RiscFree™ IDE uses the workspace directory to store its preferences and development artifacts.

D:\myfirst_niosv\software

☐ Use this as the default and do not ask again

Figure 3-11 Select Workspace

4. Choose **File->Import Nios V CMake Project** to open the Import Window.



There is no active editor that provides an outline.

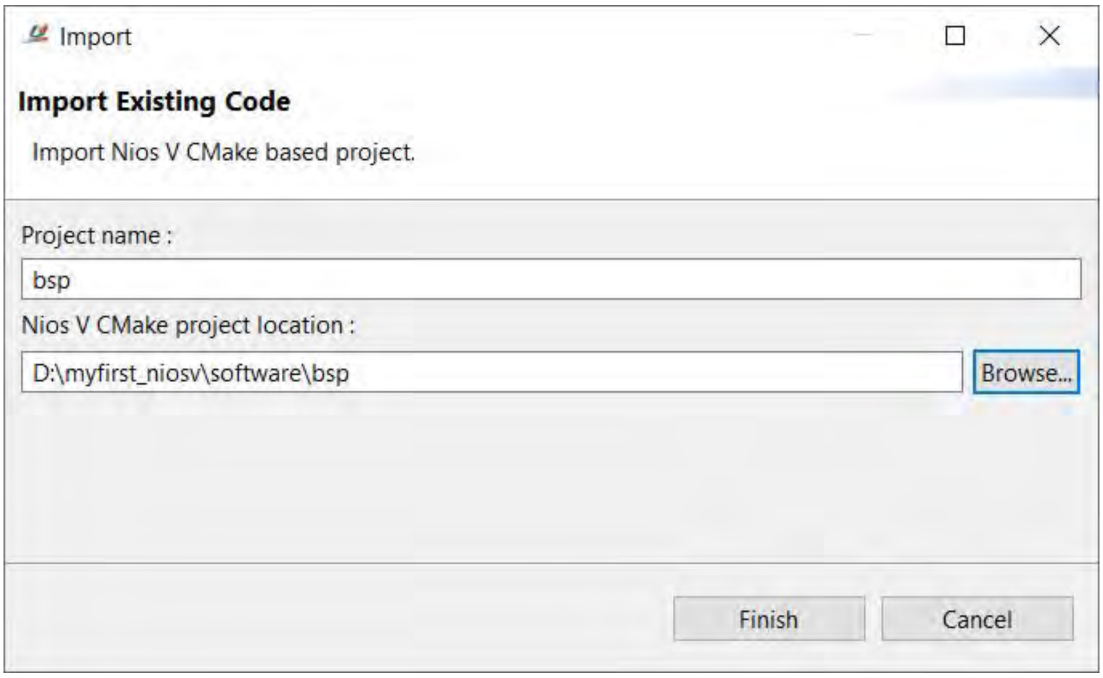
0 items

Figure 3-12 Import Nios V CMake Project

5. In the Import Window, make sure the following things:

● Browse and select the **bsp** project folder.

● Give the project a name. (**bsp** is default name)



Import Existing Code

Import Nios V CMake based project.

Project name :

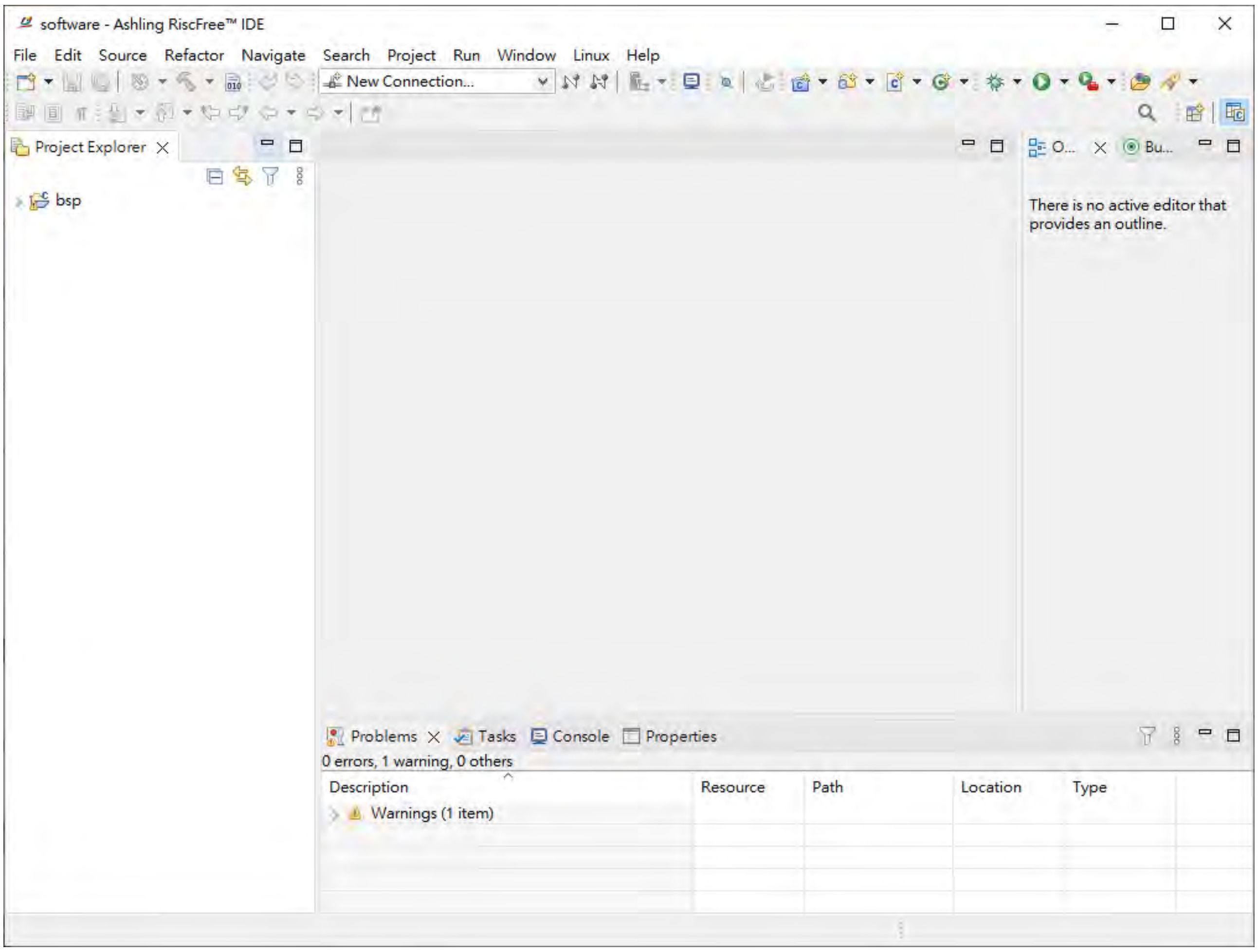
bsp

Nios V CMake project location :

D:\myfirst_niosv\software\bsp

Figure 3-13 Ashling RiscFree IDE Import Project Wizard

6. Click **Finish**. The Ashling RiscFree IDE import the **bsp** project and returns to the Nios V C/C++ project perspective. See Figure 3-14.


software - Ashling RiscFree™ IDE
File Edit Source Refactor Navigate Search Project Run Window Linux Help
New Connection...
Project Explorer
bsp
O...
Bu...

There is no active editor that provides an outline.

Problems Tasks Console Properties

0 errors, 1 warning, 0 others

Description	Resource	Path	Location	Type
Warnings (1 item)				

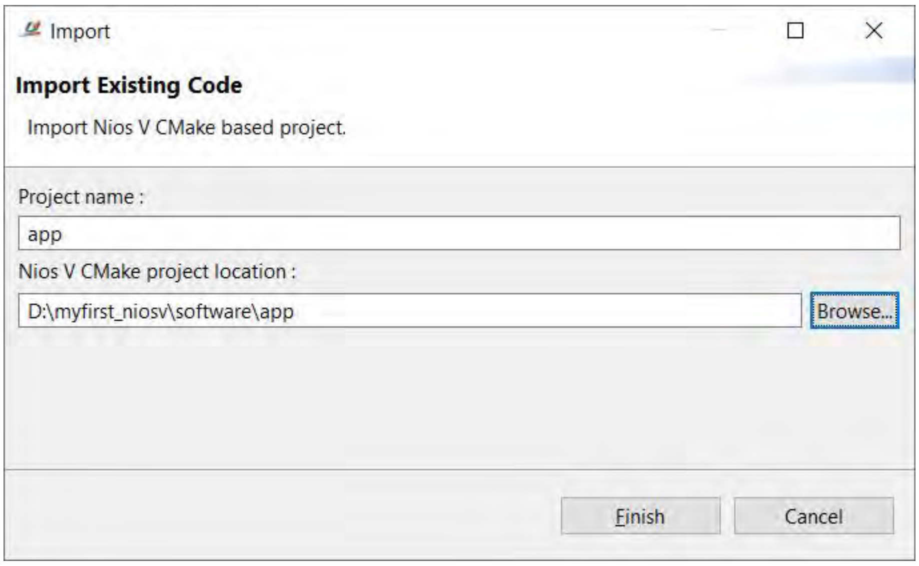
Figure 3-14 Ashling RiscFree IDE C++ Project Perspective for bsp

7. Choose **File->Import Nios V CMake Project** to open the Import Window.

8. In the Import Window, make sure the following things:

● Browse and select the **app** project folder.

● Give the project a name. (**app** is default name)



Import

Import Existing Code

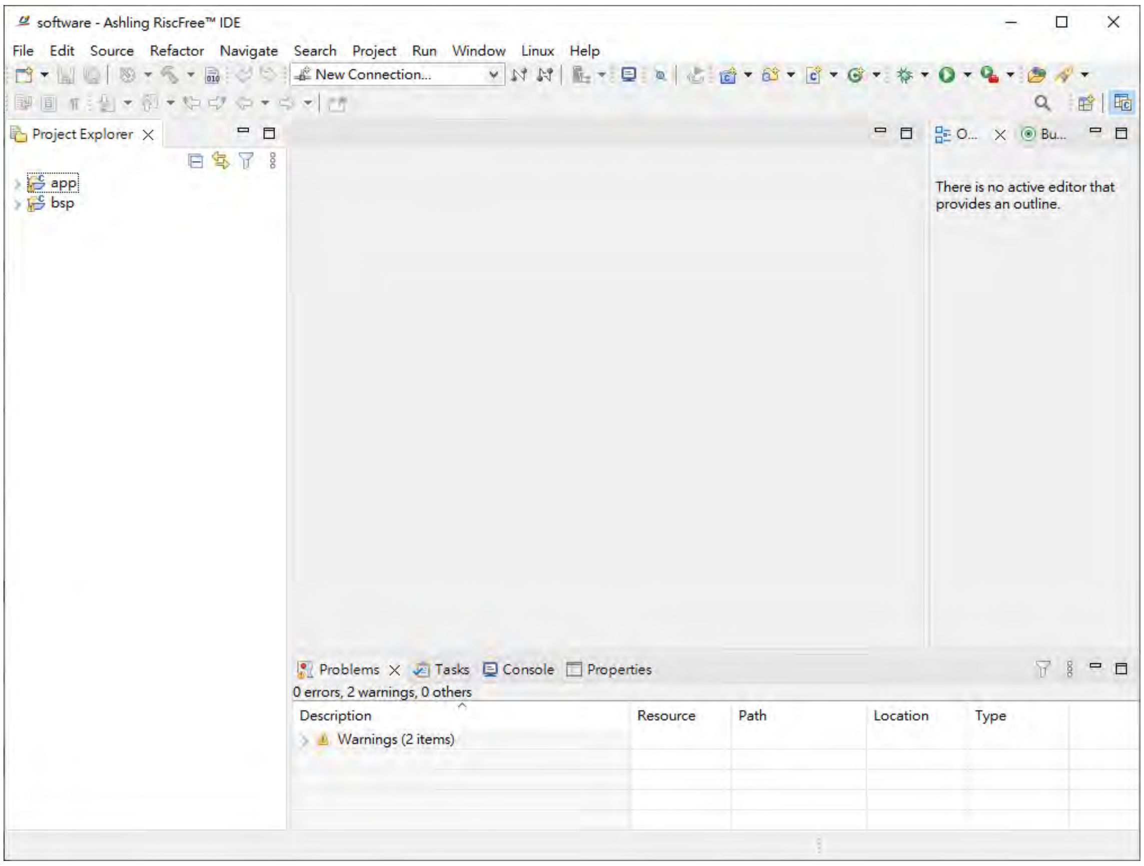
Import Nios V CMake based project.

Project name : app

Nios V CMake project location : D:\myfirst_niosv\software\app

Figure 3-15 Ashling RiscFree IDE Import Project Wizard

9. Click **Finish**. The Ashling RiscFree IDE import the **app** project and returns to the Nios V C/C++ project perspective. See Figure 3-16.



software - Ashling RiscFree™ IDE

There is no active editor that provides an outline.

0 errors, 2 warnings, 0 others

Description	Resource	Path	Location	Type
Warnings (2 items)				

Figure 3-16 Ashling RiscFree IDE C++ Project Perspective for bsp and app

When you import the project, the Ashling RiscFree IDE add two new projects in the Projects Explorer tab:

■ **app** is your C/C++ application project. This project contains the source and header files for your application.

■ **bsp** is a system library that encapsulates the details of the Nios V system hardware.

3.2 Build and Run the Program

In this section you will build and run the program to execute the compiled code.

1. Before building the project, we would like to confirm with you that the .sof file (hardware design file) in the Quartus project has been downloaded (refer to Section 2.2).

2. To build the program, right-click the **app** project in the Projects Explorer tab and choose **Build Project**. The **Build Project** dialog box appears and the IDE begins compiling the project.

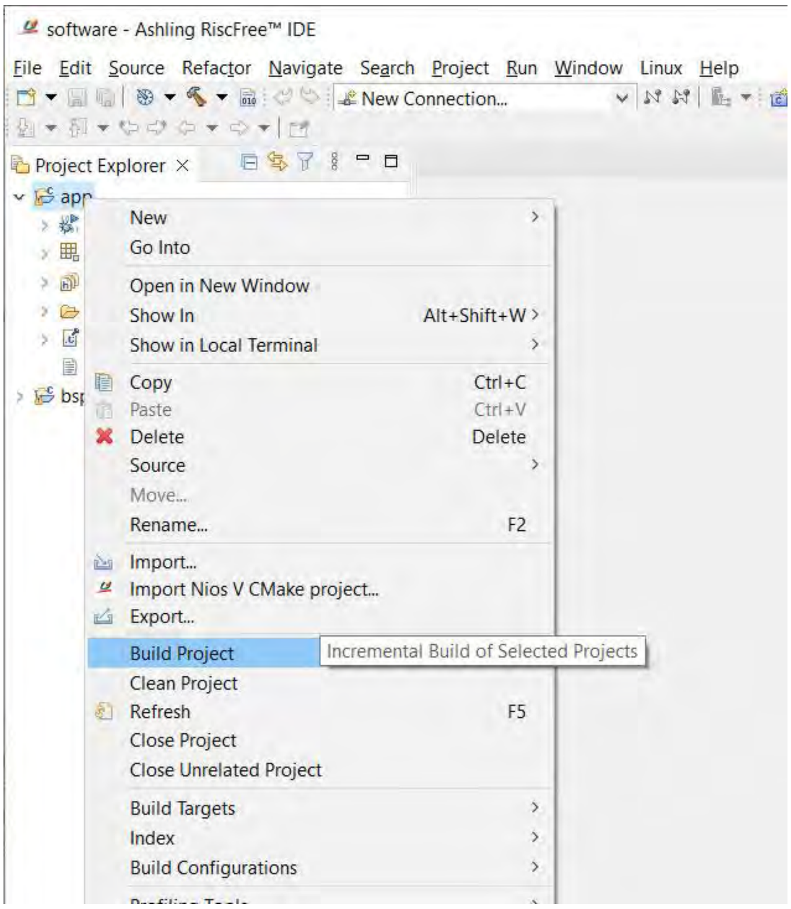
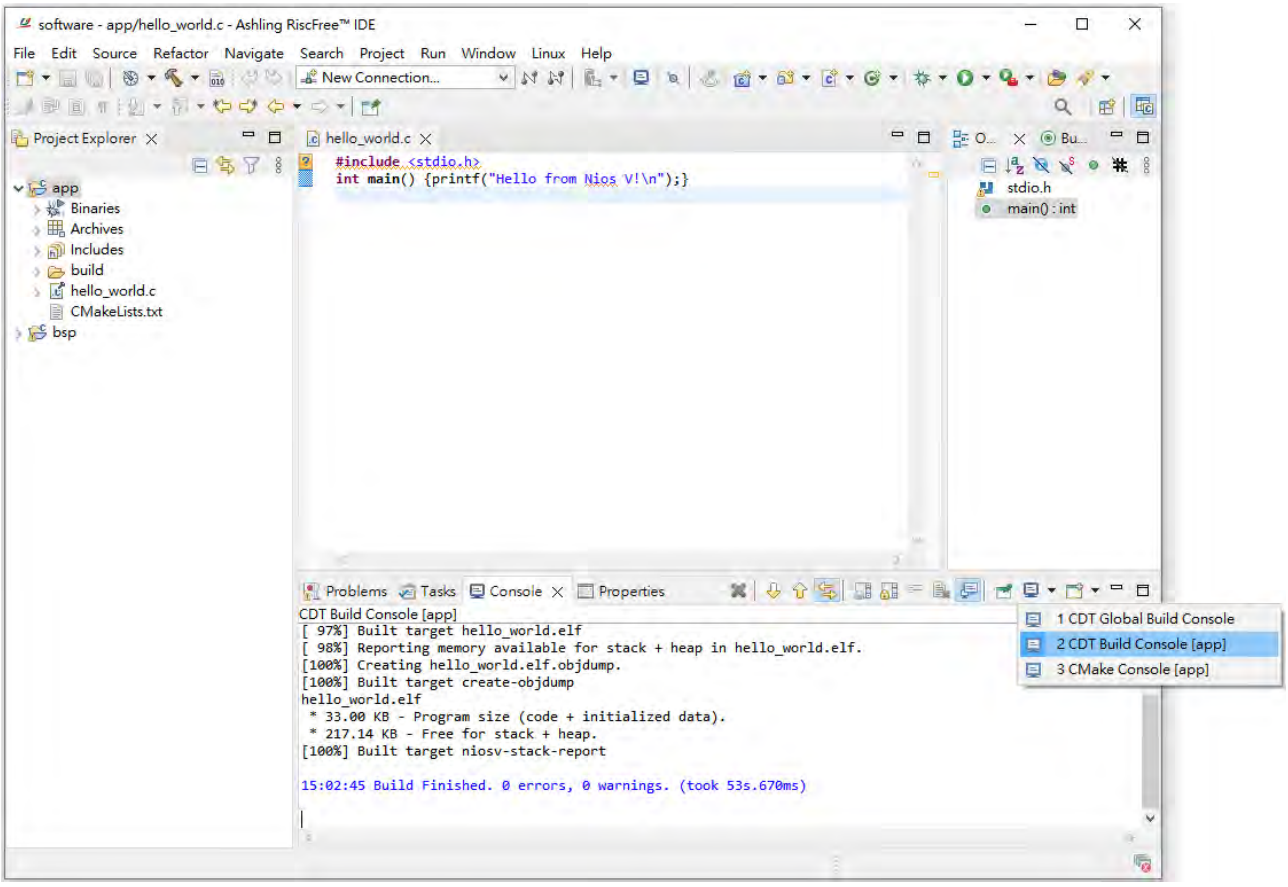


Figure 3-17 Ashling RiscFree IDE C++ Project Perspective for bsp and app

3. When the compilation is complete, a **Build Finished** message will appear in the CDT Build Console. The compilation time will vary depending on your system. Refer to Figure 3-18 for an example.


software - app/hello_world.c - Ashling RiscFree™ IDE
File Edit Source Refactor Navigate Search Project Run Window Linux Help
New Connection...
Project Explorer
app
Binaries
Archives
Includes
build
hello_world.c
CMakeLists.txt
bsp
hello_world.c
stdio.h
main() : int
Problems Tasks Console Properties
1 CDT Global Build Console
2 CDT Build Console [app]
3 CMake Console [app]

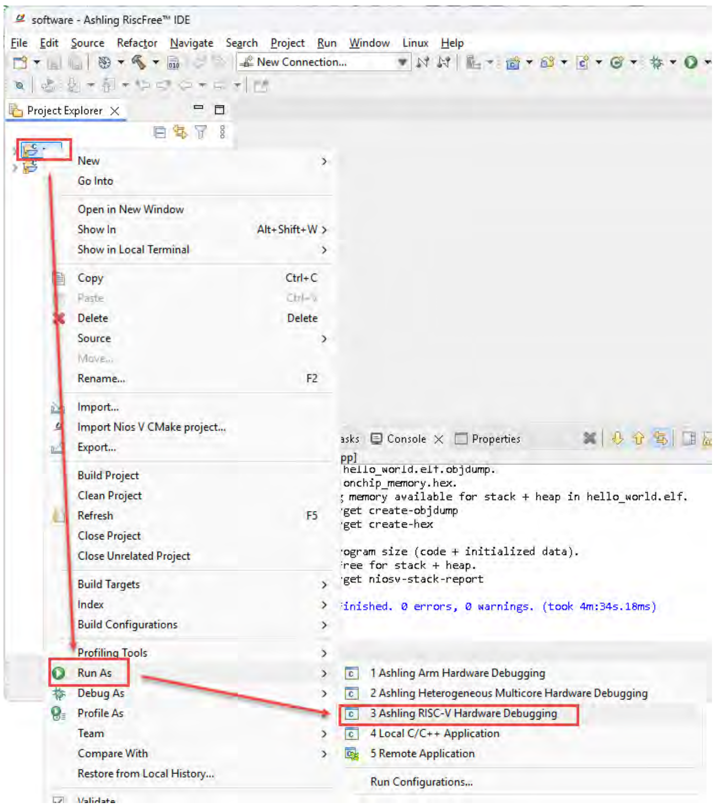
```
#include <stdio.h>
int main() {printf("Hello from Nios V!\n");}
```

CDT Build Console [app]

```
[ 97%] Built target hello_world.elf
[ 98%] Reporting memory available for stack + heap in hello_world.elf.
[100%] Creating hello_world.elf.objdump.
[100%] Built target create-objdump
hello_world.elf
 * 33.00 KB - Program size (code + initialized data).
 * 217.14 KB - Free for stack + heap.
[100%] Built target niosv-stack-report

15:02:45 Build Finished. 0 errors, 0 warnings. (took 53s.670ms)
```

Figure 3-18 Ashling RiscFree IDE app Build Completed

4. After compilation complete, right-click the **app** project, choose **Run As**, and choose **Ashling RISC-V Hardware Debugging**.



```
pp]
 hello_world.elf.objdump.
 onchip_memory.hex.
; memory available for stack + heap in hello_world.elf.
get create-objdump
get create-hex

ogram size (code + initialized data).
ree for stack + heap.
get niosv-stack-report

inished. 0 errors, 0 warnings. (took 4m:34s.18ms)
```

Figure 3-19 Ashling RiscFree IDE Import Project Wizard

5. In the **C Local Application** window, select hello_world.elf and then click OK.

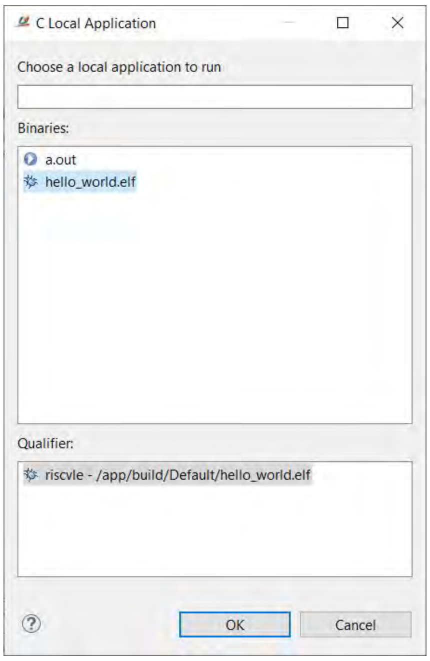
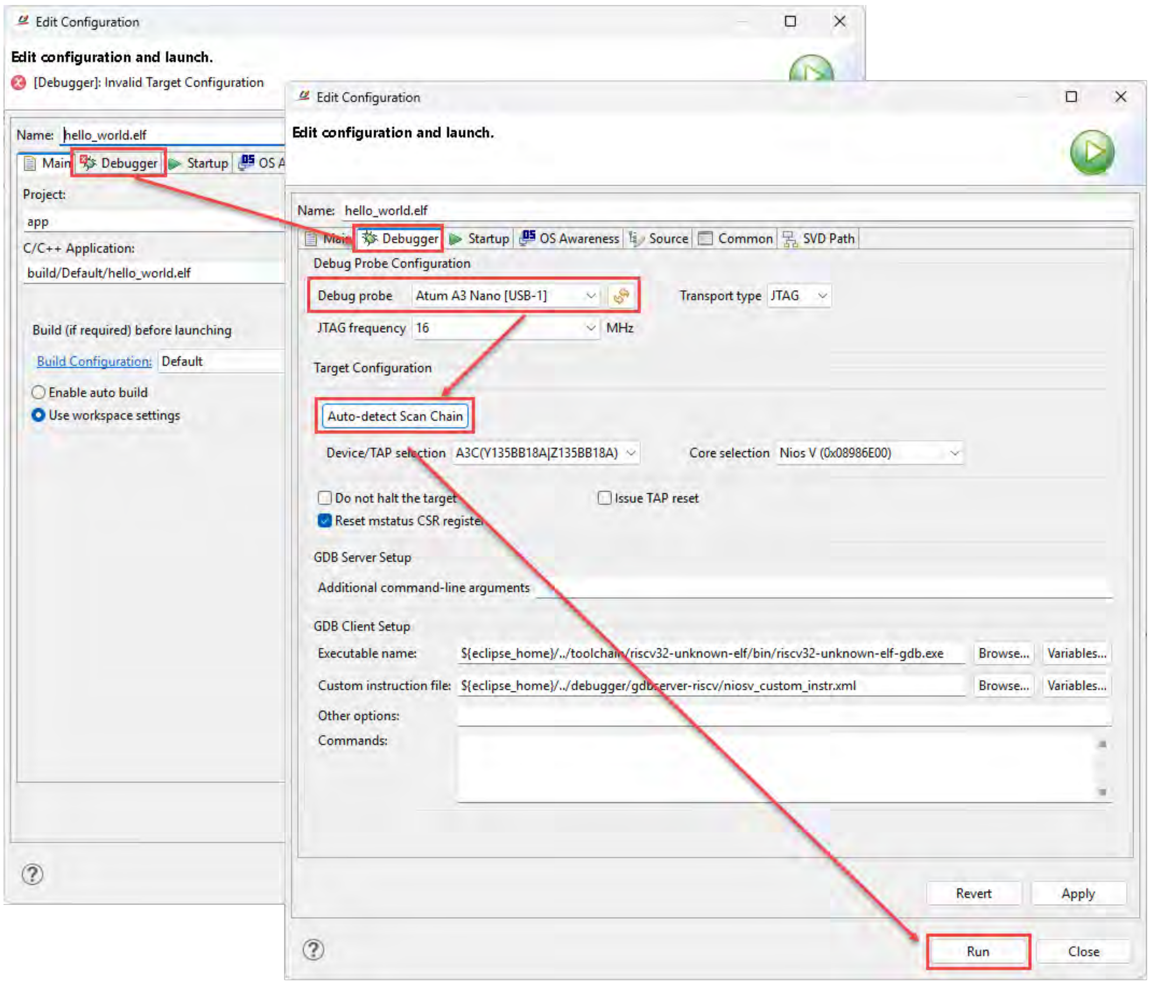

C Local Application
Choose a local application to run
Binaries:
a.out
hello_world.elf
Qualifier:
riscvle - /app/build/Default/hello_world.elf
OK
Cancel

Figure 3-20 Ashling RiscFree IDE Import Project Wizard

6. In the Edit Configuration window:

- Select the Debugger tab.
  - o For Debug Probe, select Atum A3 Nano Development board.
  - o Click Auto-detect Scan Chain.
  - o Click Run.


Edit Configuration

Edit configuration and launch.

[Debugger]: Invalid Target Configuration

Name: hello_world.elf

Project: app

C/C++ Application: build/Default/hello_world.elf

Build (if required) before launching

Build Configuration: Default

Enable auto build

Use workspace settings

Debug Probe Configuration

Debug probe: Atum A3 Nano [USB-1]

Transport type: JTAG

JTAG frequency: 16 MHz

Target Configuration

Device/TAP selection: A3C(Y135BB18A|Z135BB18A)

Core selection: Nios V (0x08986E00)

Do not halt the target

Issue TAP reset

Reset mstatus CSR register

GDB Server Setup

Additional command-line arguments

GDB Client Setup

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

Custom instruction file: ${eclipse_home}/../debugger/gdbserver-riscv/niosv_custom_instr.xml

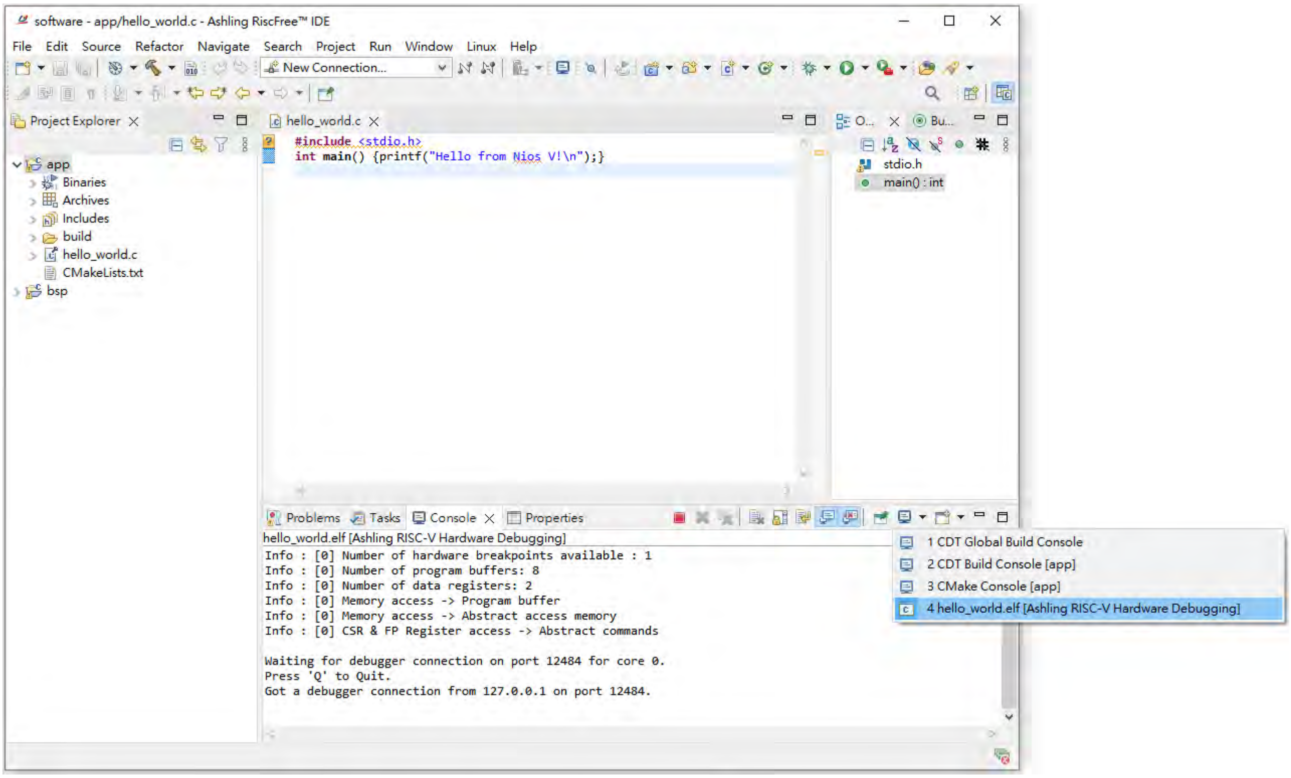
Other options:

Commands:

Figure 3-21 Ashling RiscFree IDE Import Project Wizard

The IDE begins to download the program to the target FPGA development board and begins execution.

When the target hardware begins executing the program, a message ‘Got a debugger connection’ will appear in the Console tab (switch to 4. hello_world.elf[Ashing RISC-V Hardware Debuggin]).


software - app/hello_world.c - Ashling RiscFree™ IDE
File Edit Source Refactor Navigate Search Project Run Window Linux Help
New Connection...
Project Explorer
app
Binaries
Archives
Includes
build
hello_world.c
CMakeLists.txt
bsp
hello_world.c
stdio.h
main() : int
Problems Tasks Console Properties
1 CDT Global Build Console
2 CDT Build Console [app]
3 CMake Console [app]
4 hello_world.elf [Ashling RISC-V Hardware Debugging]

```
#include <stdio.h>
int main() {printf("Hello from Nios V!\n");}
```

hello_world.elf [Ashling RISC-V Hardware Debugging]

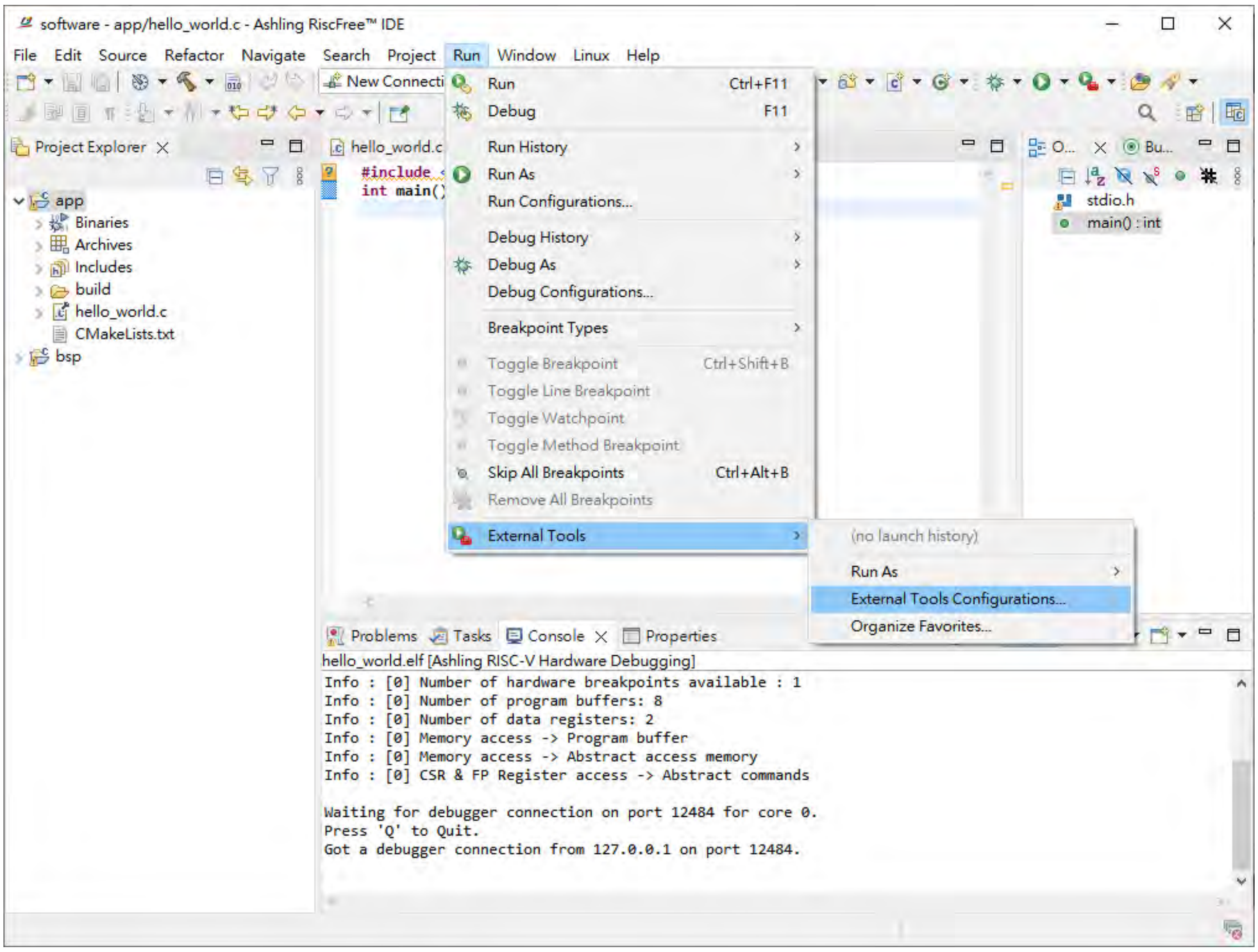
```
Info : [0] Number of hardware breakpoints available : 1
Info : [0] Number of program buffers: 8
Info : [0] Number of data registers: 2
Info : [0] Memory access -> Program buffer
Info : [0] Memory access -> Abstract access memory
Info : [0] CSR & FP Register access -> Abstract commands

Waiting for debugger connection on port 12484 for core 0.
Press 'Q' to Quit.
Got a debugger connection from 127.0.0.1 on port 12484.
```

Figure 3-22 Debugger connection message in the Console

You can configure the Ashling RiscFree IDE to integrate external tools. As an example, the following steps detail how to set up the juart-terminal tool to display a "Hello World" message via the JTAG UART IP.

To perform external tool configuration for juart-terminal, follow these steps:

1. Go to Run > External Tools > External Tools Configurations.



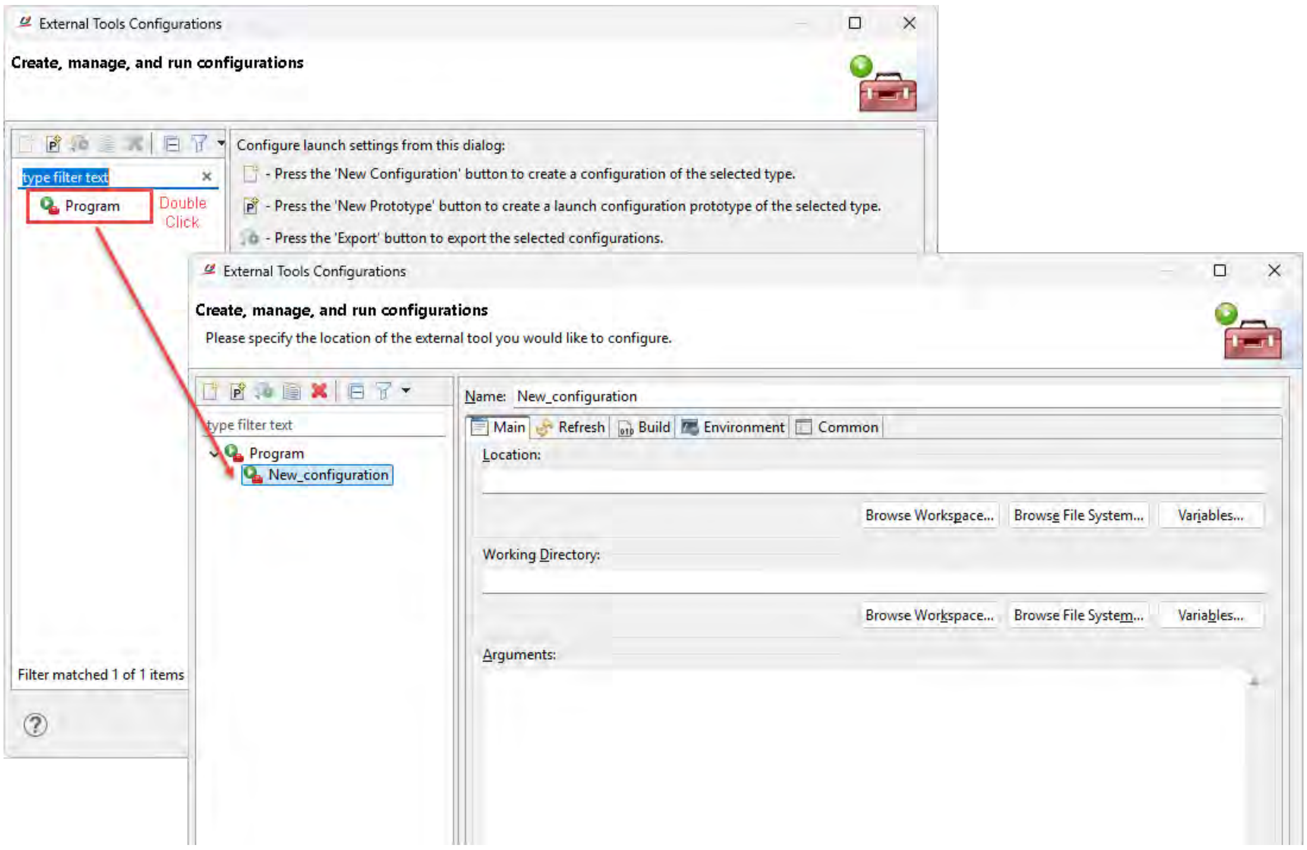
```
#include <stdio.h>
int main()
```

hello_world.elf [Ashling RISC-V Hardware Debugging]

```
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 12484 for core 0.
Press 'Q' to Quit.
Got a debugger connection from 127.0.0.1 on port 12484.
```

Figure 3-23 Navigating to External Tools Configurations

2. Double click Program to open a New_configuration window.


External Tools Configurations

Create, manage, and run configurations

Configure launch settings from this dialog:

- Press the 'New Configuration' button to create a configuration of the selected type.
- Press the 'New Prototype' button to create a launch configuration prototype of the selected type.
- Press the 'Export' button to export the selected configurations.

Filter matched 1 of 1 items

External Tools Configurations

Create, manage, and run configurations

Please specify the location of the external tool you would like to configure.

Name: New_configuration

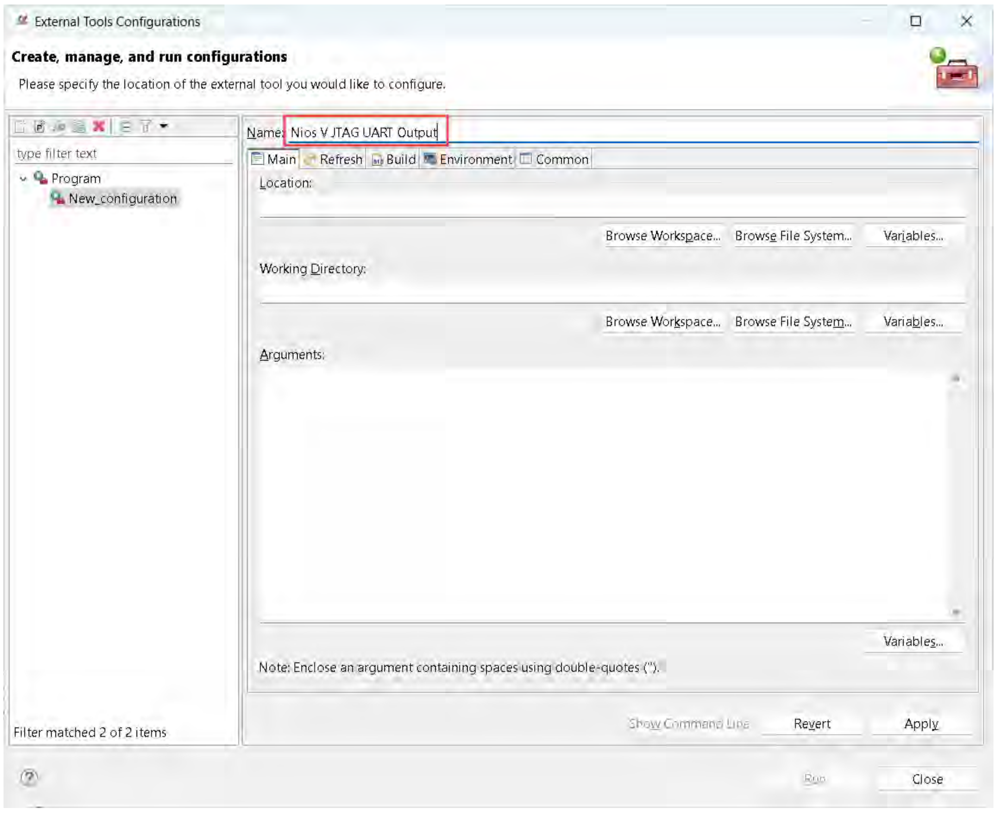
Location:

Working Directory:

Arguments:

Figure 3-24 Creating a new Program configuration

3. Rename the configuration as Nios V JTAG UART Output.


External Tools Configurations

Create, manage, and run configurations

Please specify the location of the external tool you would like to configure.

Name: Nios V JTAG UART Output

Location:

Working Directory:

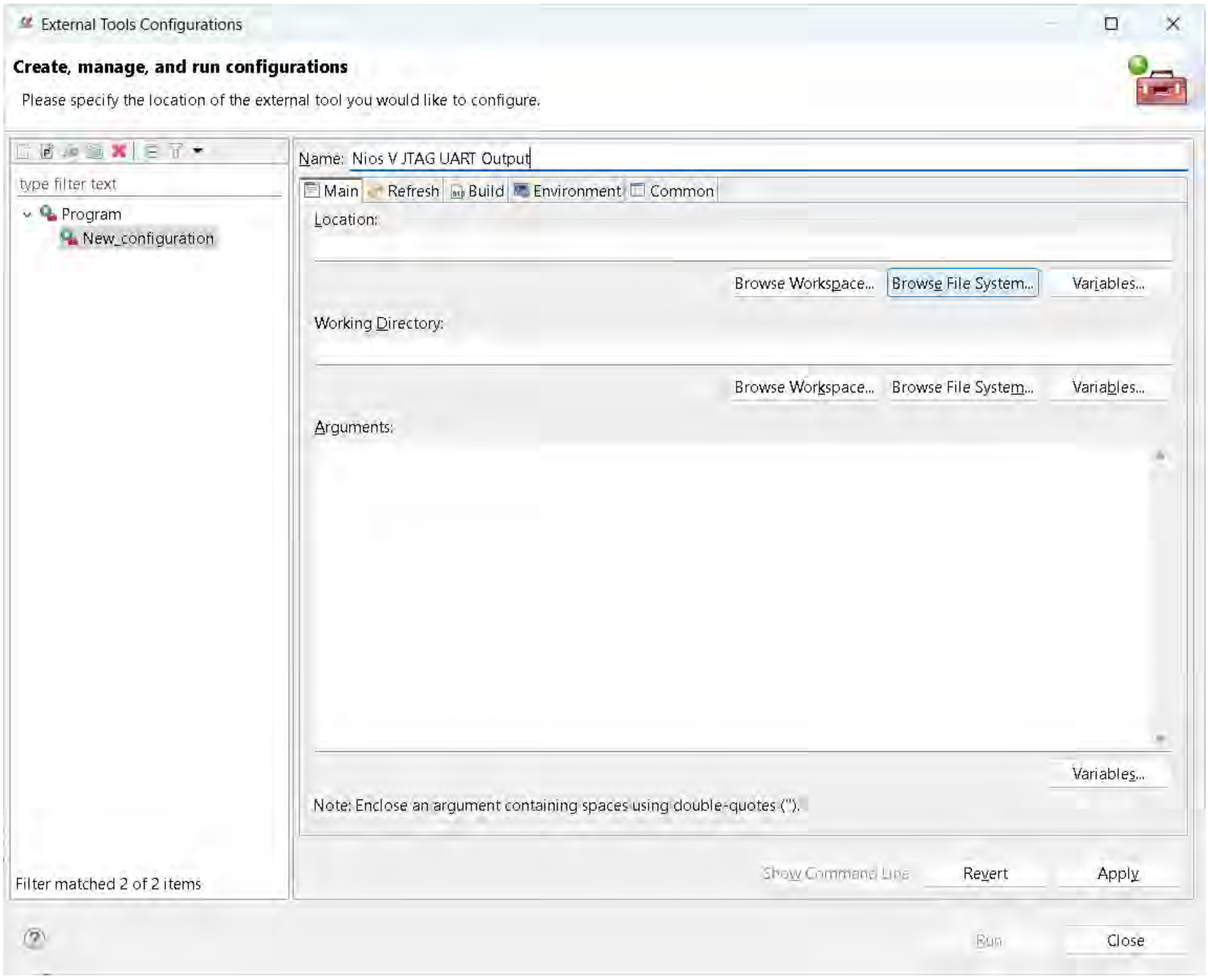
Arguments:

Note: Enclose an argument containing spaces using double-quotes (").

Filter matched 2 of 2 items

Figure 3-25 Renaming the configuration to 'Nios V JTAG UART Output'

4. In Location, click Browse File system.


External Tools Configurations

Create, manage, and run configurations

Please specify the location of the external tool you would like to configure.

type filter text

Program

New_configuration

Filter matched 2 of 2 items

Name: Nios V JTAG UART Output

Main | Refresh | Build | Environment | Common

Location:

Working Directory:

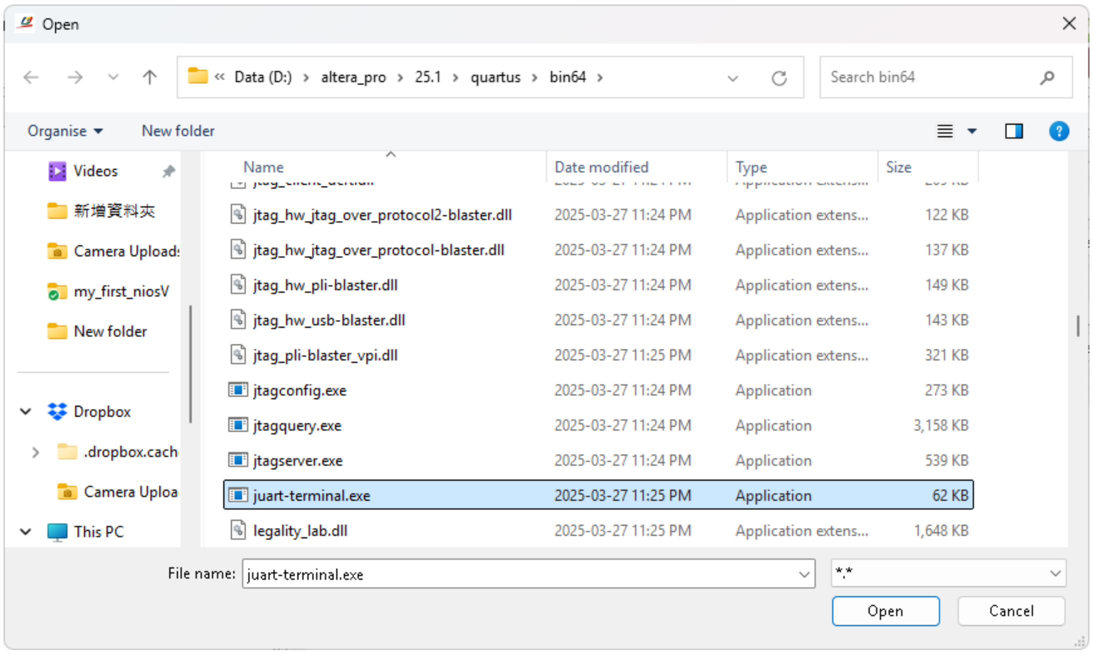
Arguments:

Note: Enclose an argument containing spaces using double-quotes (").

Figure 3-26 Browse for the executable file in the Location field

5. Browse and select the juart-terminal file in the following paths:

   <Intel Quartus Prime installation directory>/quartus/bin64/juart-terminal.exe



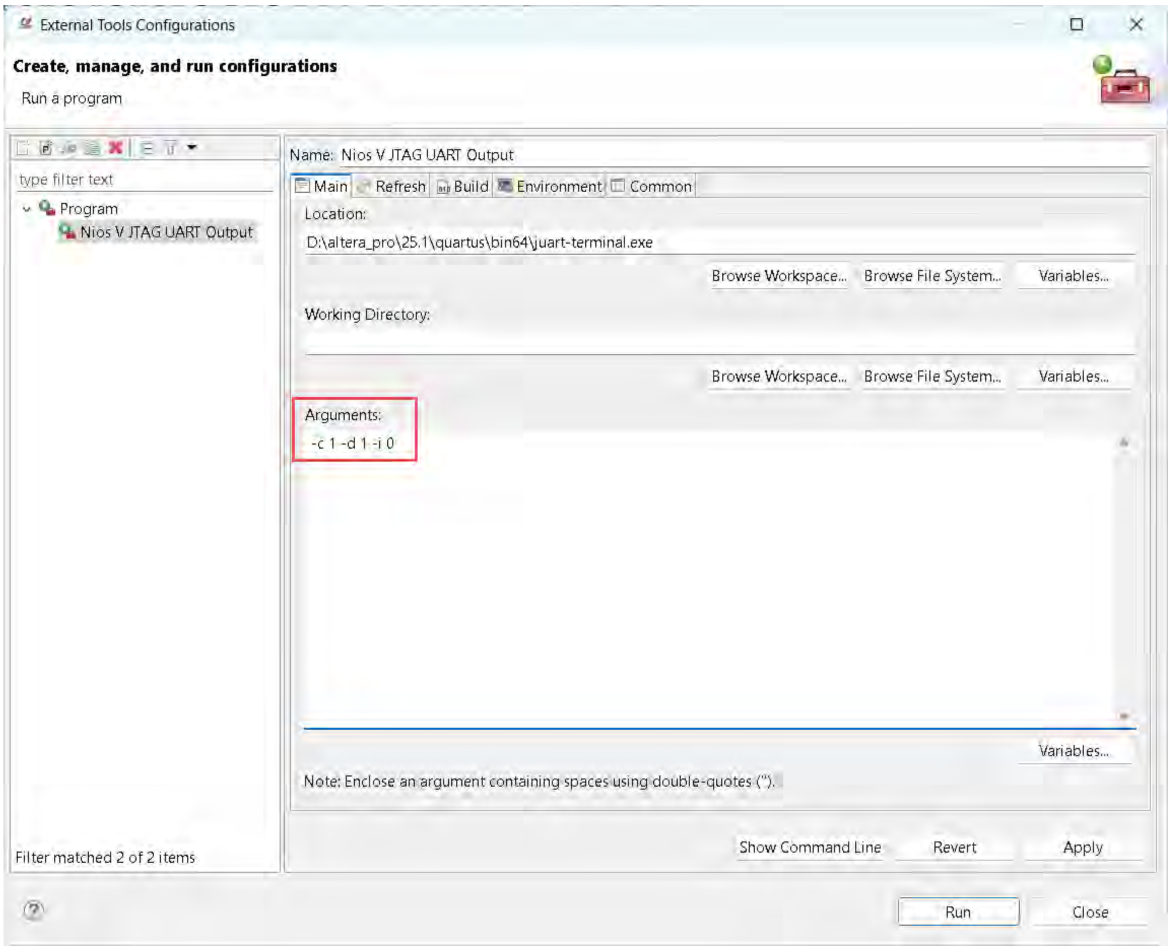
Name	Date modified	Type	Size
jtag_hw_jtag_over_protocol2-blaster.dll	2025-03-27 11:24 PM	Application extens...	122 KB
jtag_hw_jtag_over_protocol-blaster.dll	2025-03-27 11:24 PM	Application extens...	137 KB
jtag_hw_pli-blaster.dll	2025-03-27 11:24 PM	Application extens...	149 KB
jtag_hw_usb-blaster.dll	2025-03-27 11:24 PM	Application extens...	143 KB
jtag_pli-blaster_vpi.dll	2025-03-27 11:25 PM	Application extens...	321 KB
jtagconfig.exe	2025-03-27 11:24 PM	Application	273 KB
jtagquery.exe	2025-03-27 11:24 PM	Application	3,158 KB
jtagserver.exe	2025-03-27 11:24 PM	Application	539 KB
juart-terminal.exe	2025-03-27 11:25 PM	Application	62 KB
legality_lab.dll	2025-03-27 11:25 PM	Application extens...	1,648 KB

File name: juart-terminal.exe

.

Figure 3-27 Selecting 'juart-terminal.exe'

6. Set the Arguments as "-c 1 -d 1 -i 0". This configures the JTAG UART connection is towards the JTAG UART IP at cable 1, device 1, and instance 0.


External Tools Configurations

Create, manage, and run configurations

Run a program

type filter text

Program

Nios V JTAG UART Output

Filter matched 2 of 2 items

Name: Nios V JTAG UART Output

Main | Refresh | Build | Environment | Common

Location:

D:\altera_pro\25.1\quartus\bin64\juart-terminal.exe

Working Directory:

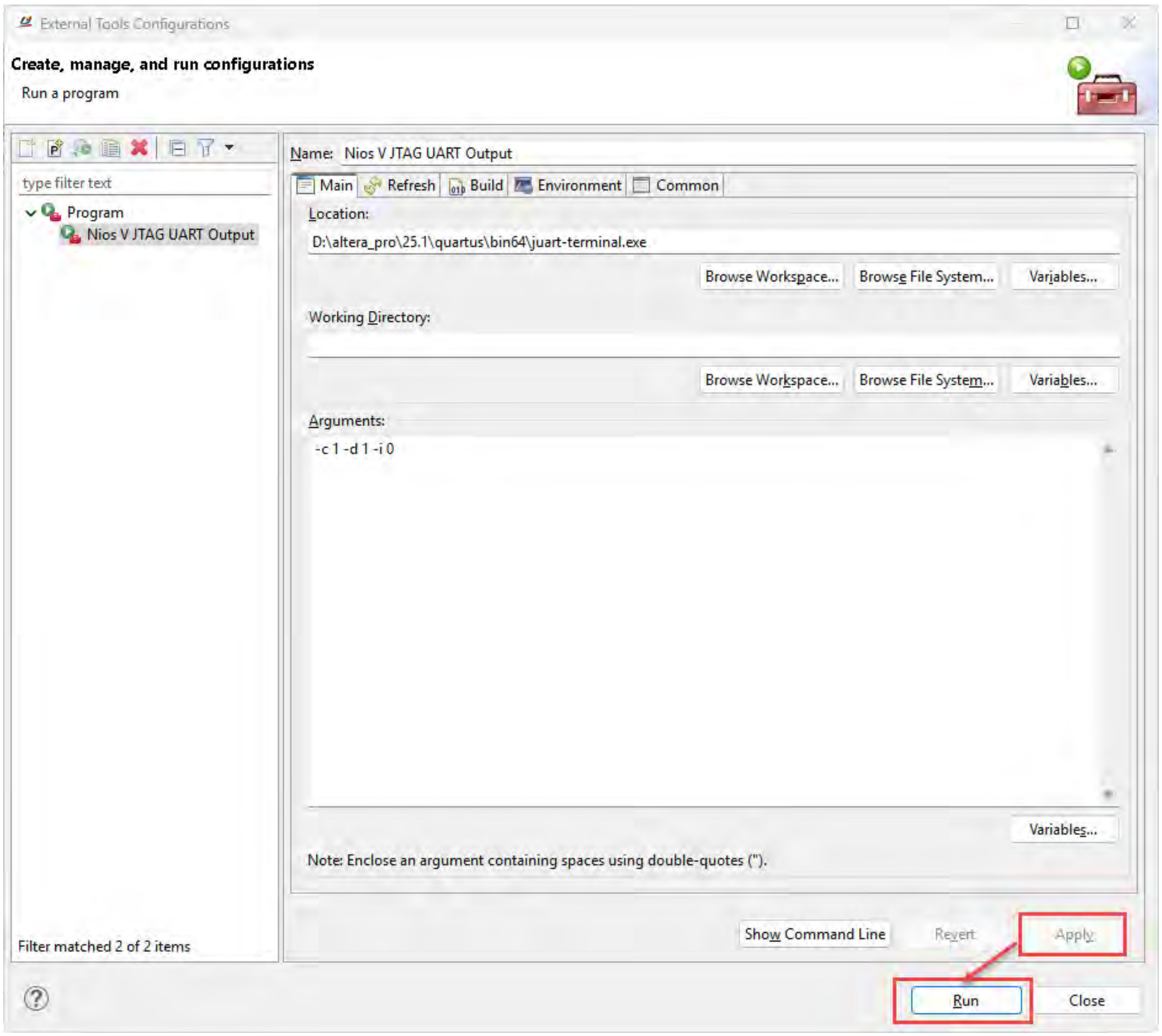
Arguments:

-c 1 -d 1 -i 0

Note: Enclose an argument containing spaces using double-quotes (").

Figure 3-28 Setting arguments for the JTAG UART

7. Click Apply, then click Run


External Tools Configurations

Create, manage, and run configurations

Run a program

type filter text

Program
Nios V JTAG UART Output

Filter matched 2 of 2 items

Name: Nios V JTAG UART Output

Main | Refresh | Build | Environment | Common

Location:
D:\altera_pro\25.1\quartus\bin64\juart-terminal.exe

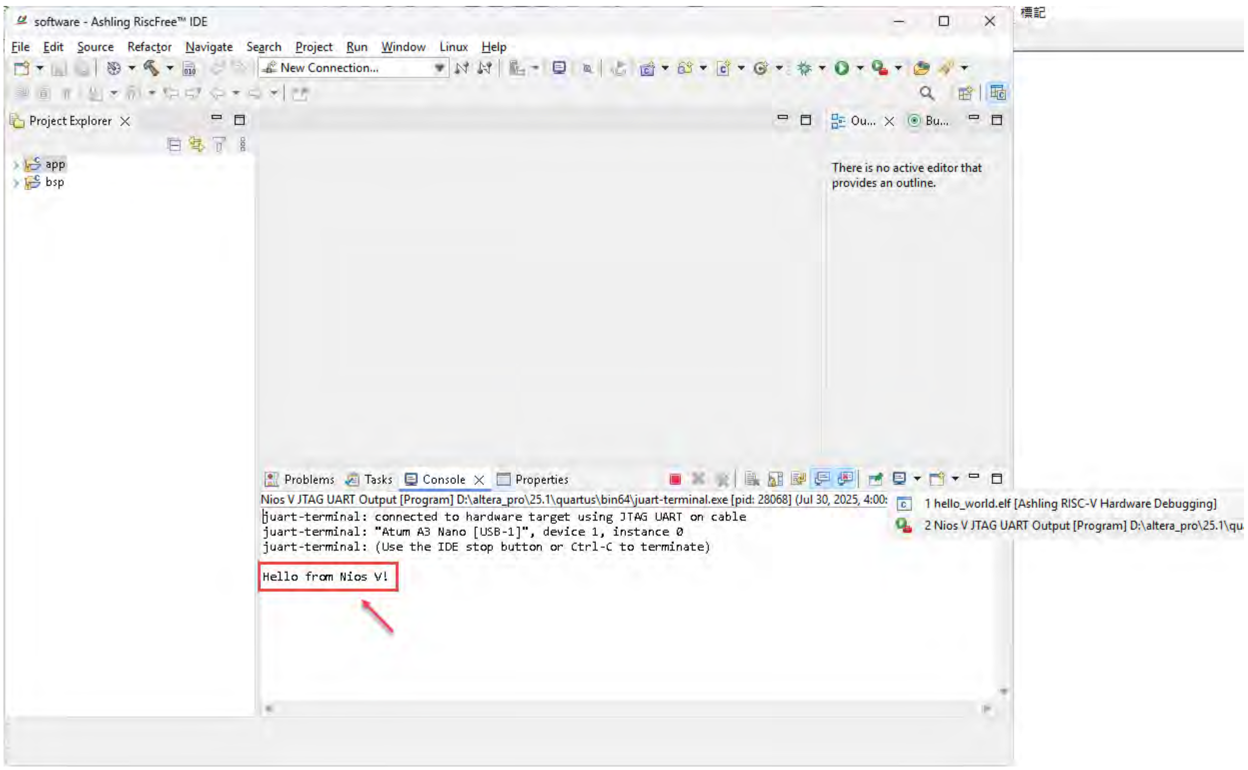
Working Directory:

Arguments:
-c 1 -d 1 -i 0

Note: Enclose an argument containing spaces using double-quotes (").

Figure 3-29 Applying and running the external tool

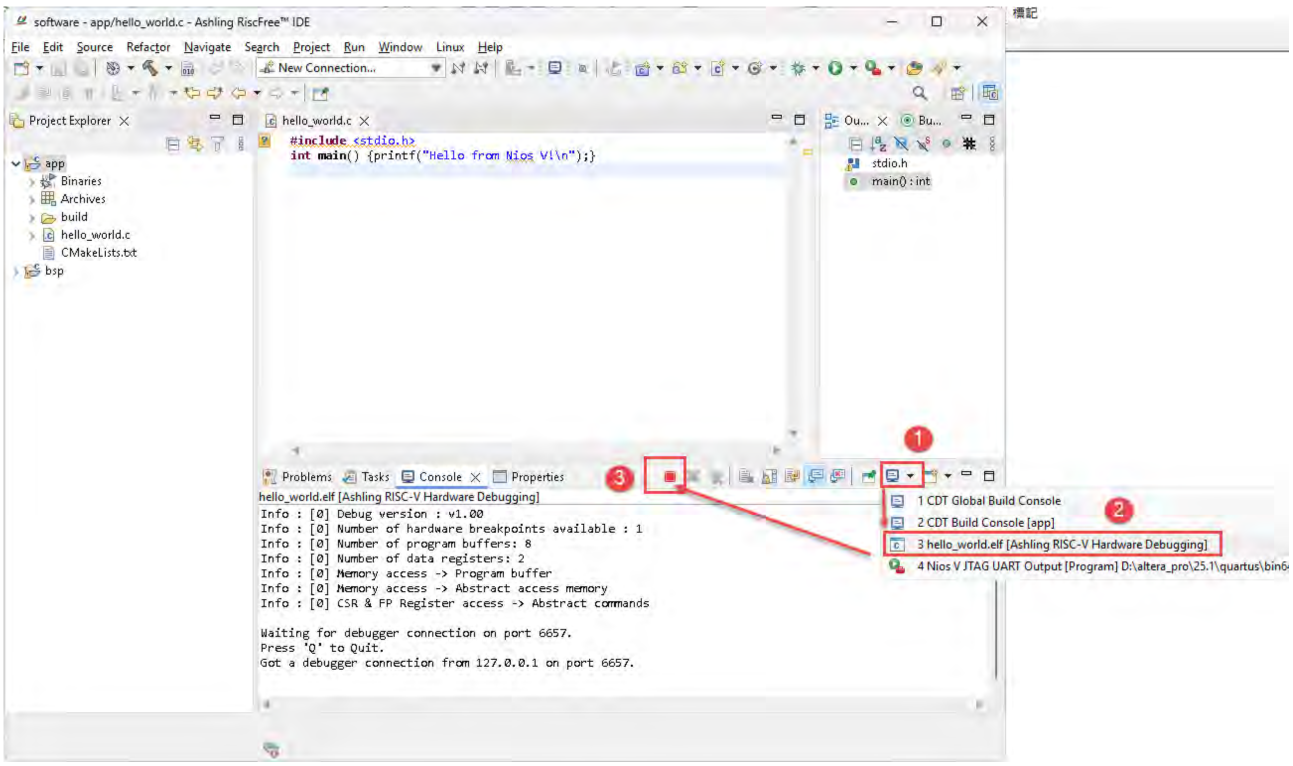
8. The message ’Hello from Nios V!’ appears in the Nios V JTAG UART Output tab.


software - Ashling RiscFree™ IDE
File Edit Source Refactor Navigate Search Project Run Window Linux Help
New Connection...
Project Explorer
app
bsp
Ou...
Bu...
There is no active editor that provides an outline.
Problems Tasks Console Properties
1 hello_world.elf [Ashling RISC-V Hardware Debugging]
2 Nios V JTAG UART Output [Program] D:\altera_pro\25.1\qu

```
Nios V JTAG UART Output [Program] D:\altera_pro\25.1\quartus\bin64\juart-terminal.exe [pid: 28068] (Jul 30, 2025, 4:00:
juart-terminal: connected to hardware target using JTAG UART on cable
juart-terminal: "Atum A3 Nano [USB-1]", device 1, instance 0
juart-terminal: (Use the IDE stop button or Ctrl-C to terminate)

Hello from Nios V!
```

Figure 3-30 "Hello from Nios V!" message in the output tab.

9. Before proceeding to the next step, the user needs to switch to the hello_world.elf console and stop Terminate . This will ensure that the next step can be executed without any issues.


software - app/hello_world.c - Ashling RiscFree™ IDE
File Edit Source Refactor Navigate Search Project Run Window Linux Help
Project Explorer
app
Binaries
Archives
build
hello_world.c
CMakeLists.txt
bsp
hello_world.c
stdio.h
main() : int
Problems Tasks Console Properties
1
2
3
1 CDT Global Build Console
2 CDT Build Console [app]
3 hello_world.elf [Ashling RISC-V Hardware Debugging]
4 Nios V JTAG UART Output [Program] D:\altera_pro\25.1\quartus\bin6

```
#include <stdio.h>
int main() {printf("Hello from Nios V!\n");}
```

hello_world.elf [Ashling RISC-V Hardware Debugging]

```
Info : [0] Debug version : v1.00
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 6657.
Press 'Q' to Quit.
Got a debugger connection from 127.0.0.1 on port 6657.
```

Figure 3-31 Stop Terminate.

Now you have created, compiled, and run your first software program based on NIOS V. And you can perform additional operations such as configuring the system properties, editing and re-building the application, and debugging the source code.

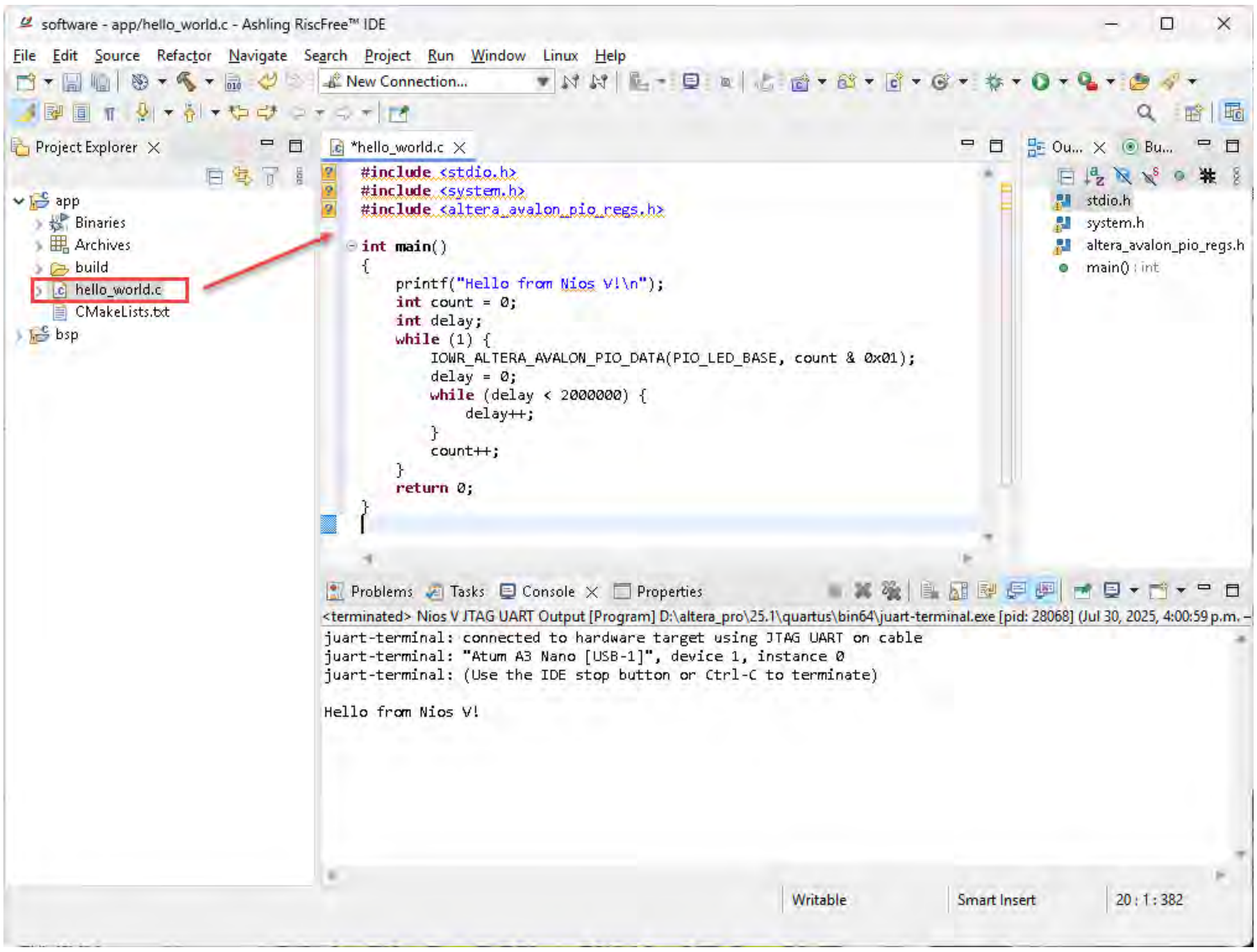
3.3 Edit and Re-Run the Program

You can modify the **hello_world.c** program file in the IDE, build it, and re-run the program to observe your changes executing on the target board. In this section you will add code that will make LEDG blink.

Perform the following steps to modify and re-run the program:

1. In the hello_world.c file, modify the text shown in the example below:

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

int main()
{
    printf("Hello from Nios V!\n");
    int count = 0;
    int delay;
    while (1) {
        IOWR_ALTERA_AVALON_PIO_DATA(PIO_LED_BASE, count & 0x01);
        delay = 0;
        while (delay < 2000000) {
            delay++;
        }
        count++;
    }
    return 0;
}
```



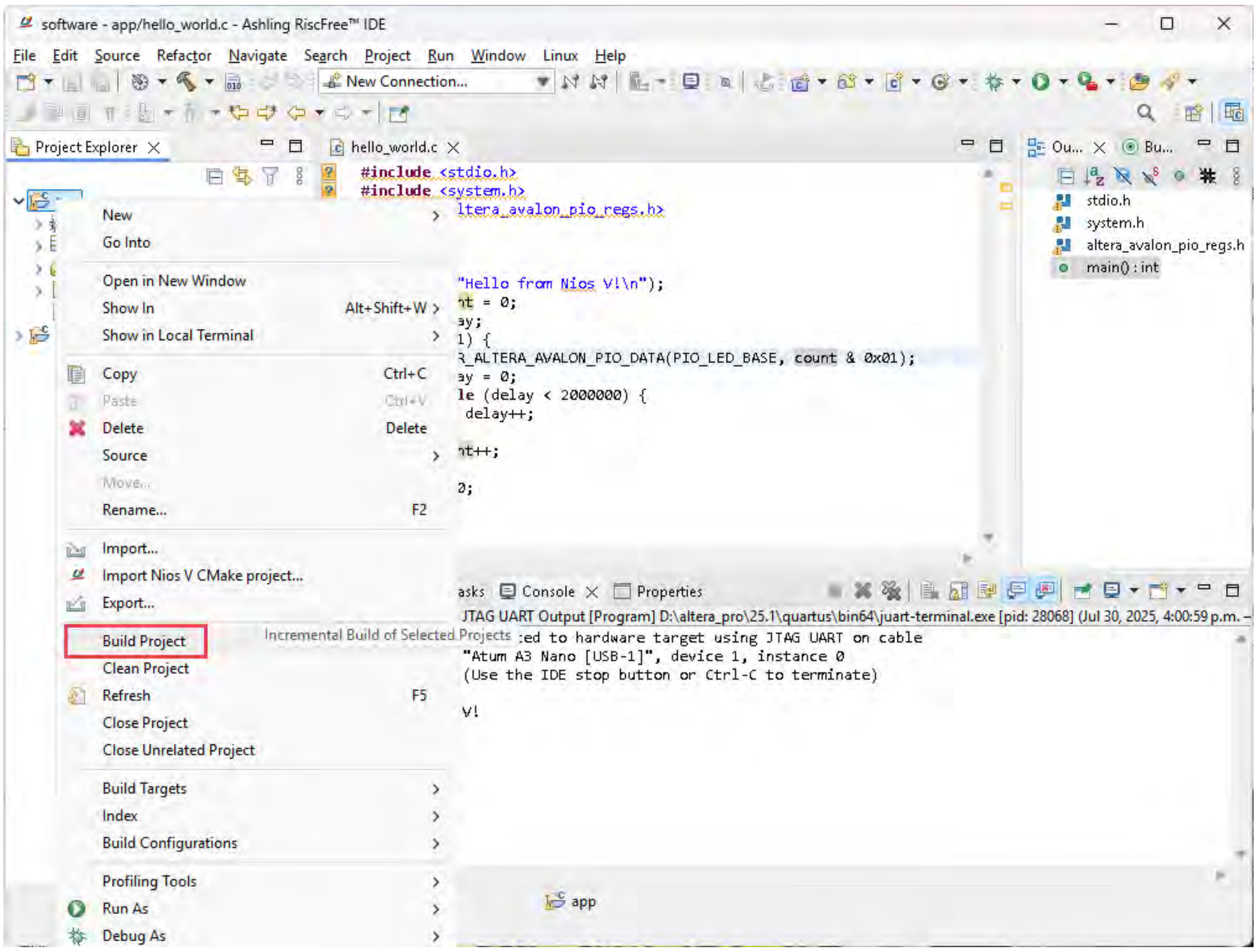
```
#include <stdio.h>
#include <system.h>
#include <altera_avalon_pio_regs.h>

int main()
{
    printf("Hello from Nios V!\n");
    int count = 0;
    int delay;
    while (1) {
        IOWR_ALTERA_AVALON_PIO_DATA(PIO_LED_BASE, count & 0x01);
        delay = 0;
        while (delay < 2000000) {
            delay++;
        }
        count++;
    }
    return 0;
}
```

```
<terminated> Nios V JTAG UART Output [Program] D:\altera_pro\25.1\quartus\bin64\juart-terminal.exe [pid: 28068] (Jul 30, 2025, 4:00:59 p.m. -
juart-terminal: connected to hardware target using JTAG UART on cable
juart-terminal: "Atum A3 Nano [USB-1]", device 1, instance 0
juart-terminal: (Use the IDE stop button or Ctrl-C to terminate)

Hello from Nios V!
```

Figure 3-32 Adding LED blink code to 'hello_world.c'

2. Save the project.

3. Recompile the file by right-clicking **app** in the Projects Explorer tab and choosing **Build Project**

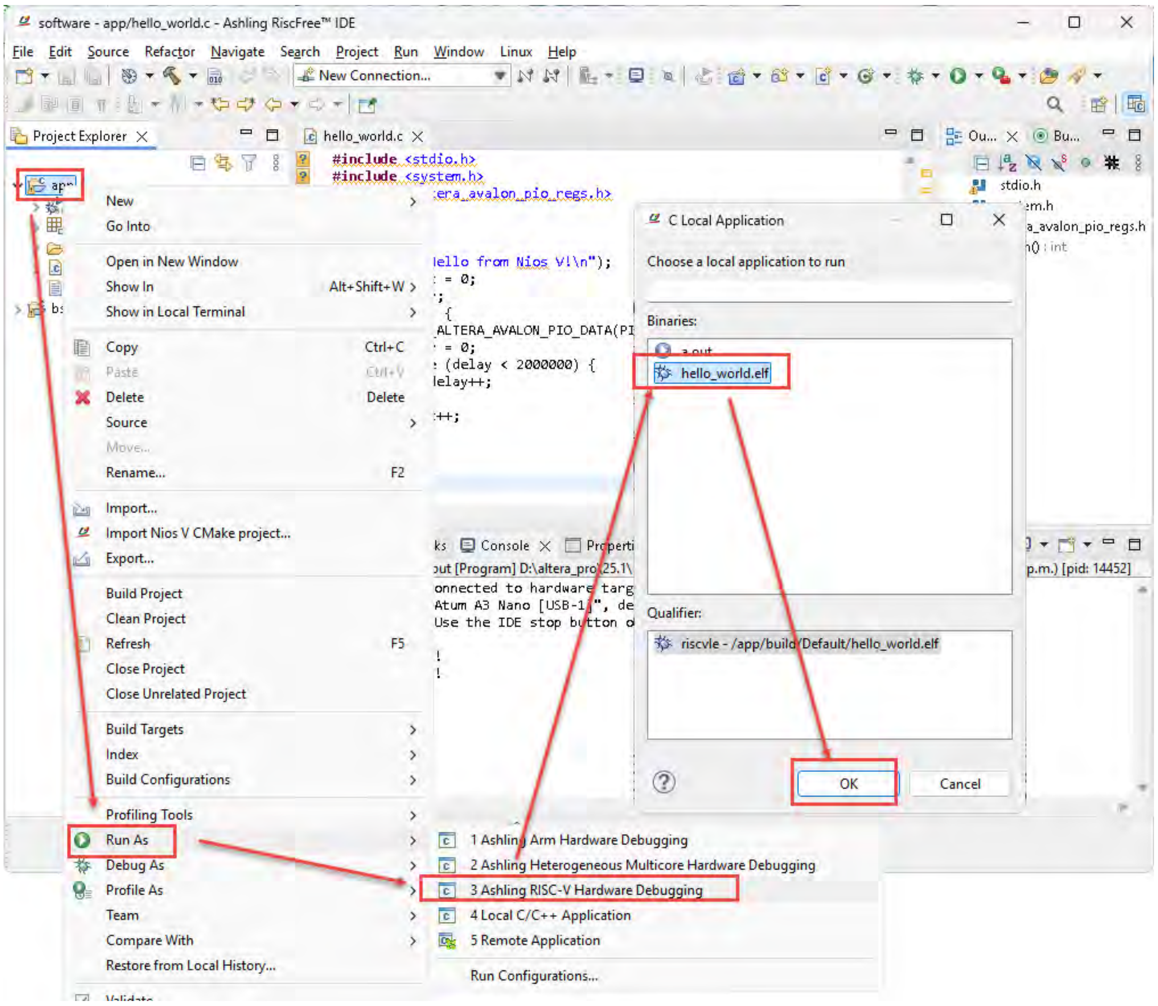


```
#include <stdio.h>
#include <system.h>
ltera_avalon_pio_regs.h>

"Hello from Nios V!\n");
nt = 0;
ay;
1) {
R_ALTERA_AVALON_PIO_DATA(PIO_LED_BASE, count & 0x01);
ay = 0;
le (delay < 2000000) {
 delay++;

nt++;

0;
```

```
JTAG UART Output [Program] D:\altera_pro\25.1\quartus\bin64\juart-terminal.exe [pid: 28068] (Jul 30, 2025, 4:00:59 p.m. –
:ed to hardware target using JTAG UART on cable
"Atum A3 Nano [USB-1]", device 1, instance 0
(Use the IDE stop button or Ctrl-C to terminate)

V!
```

Figure 3-33 Rebuilding the project after code changes

4. After **Build Porject** done**, Run** > **Run As** > **Ashling RISC-V Hardware Debugging**.



```
#include <stdio.h>
#include <system.h>
:era_avalon_pio_regs.h>

Jello from Nios V!\n");
: = 0;
;
 {
_ALTERA_AVALON_PIO_DATA(PI
: = 0;
: (delay < 2000000) {
lelay++;

:++;
```

```
out [Program] D:\altera_pro\25.1\
onnected to hardware targ
Atum A3 Nano [USB-1]", de
Use the IDE stop button o
```

Figure 3-34 Run program

5. Orient your development board so that you can observe LEDG blinking.

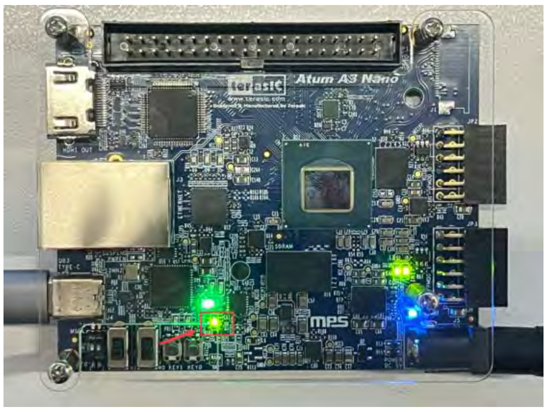
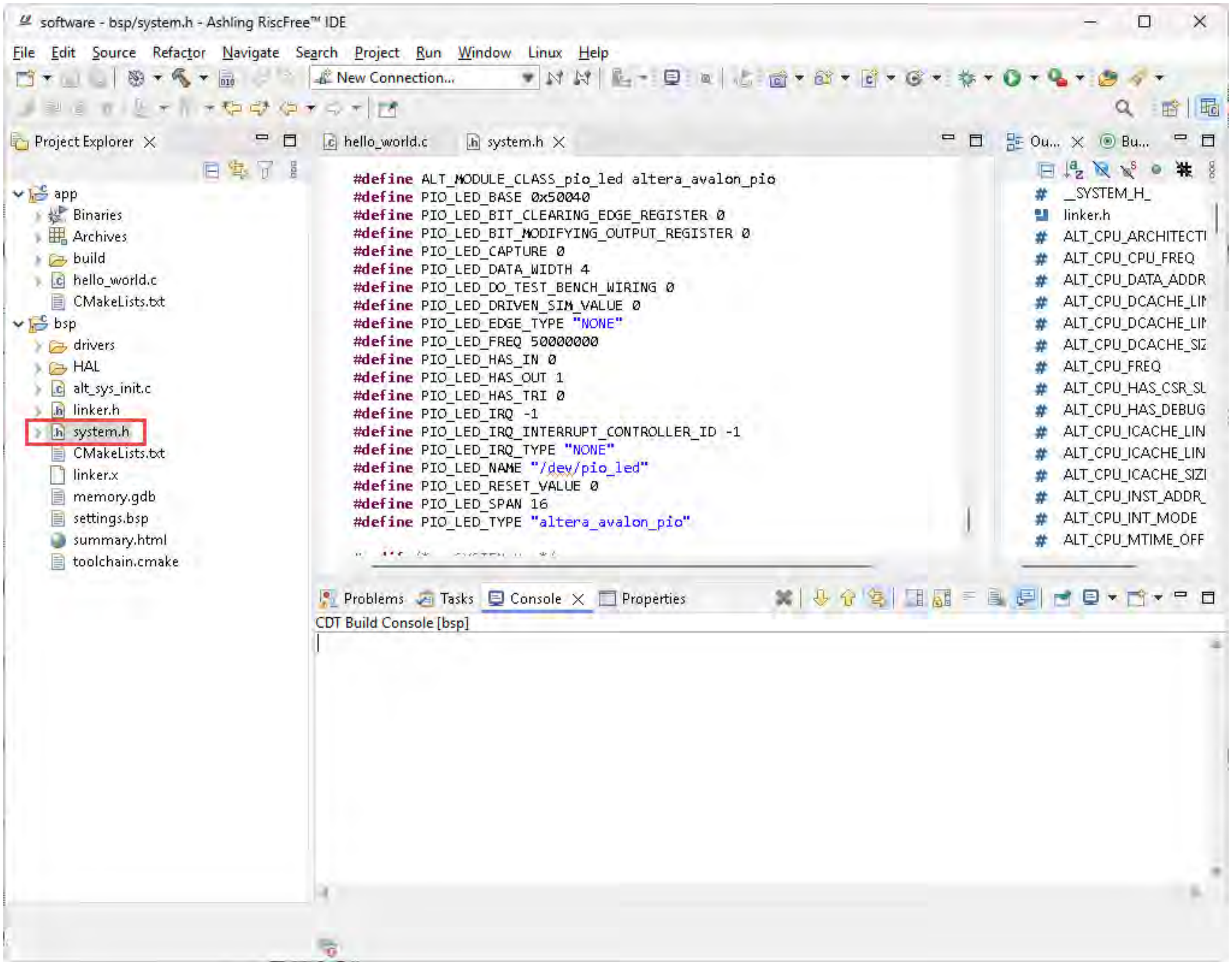


Figure 3-35 LEDG

3.4 Why the LED Blinks

The Nios V system description header file, **system.h**, contains the software definitions, name, locations, base addresses, and settings for all of the components in the Nios V hardware system. The **system.h** file is located in the in the **bsp\system.h** directory as shown in Figure 3-36.



```
#define ALT_MODULE_CLASS_pio_led altera_avalon_pio
#define PIO_LED_BASE 0x50040
#define PIO_LED_BIT_CLEARING_EDGE_REGISTER 0
#define PIO_LED_BIT_MODIFYING_OUTPUT_REGISTER 0
#define PIO_LED_CAPTURE 0
#define PIO_LED_DATA_WIDTH 4
#define PIO_LED_DO_TEST_BENCH_WIRING 0
#define PIO_LED_DRIVEN_SIM_VALUE 0
#define PIO_LED_EDGE_TYPE "NONE"
#define PIO_LED_FREQ 50000000
#define PIO_LED_HAS_IN 0
#define PIO_LED_HAS_OUT 1
#define PIO_LED_HAS_TRI 0
#define PIO_LED_IRQ -1
#define PIO_LED_IRQ_INTERRUPT_CONTROLLER_ID -1
#define PIO_LED_IRQ_TYPE "NONE"
#define PIO_LED_NAME "/dev/pio_led"
#define PIO_LED_RESET_VALUE 0
#define PIO_LED_SPAN 16
#define PIO_LED_TYPE "altera_avalon_pio"
```

Figure 3-36 system.h Location

If you look at the **system.h** file for the Nios V project example used in this tutorial, you will notice the **pio_led** function. This function controls the LED. The Nios V processor controls the PIO ports (and thereby the LED) by reading and writing to the register map. For the PIO, there are four registers: **data, direction, interrupt mask, and edge capture**. To turn the LED on and off, the application writes to the PIO data register.

The PIO core has an associated software file **altera_avalon_pio_regs.h**. This file defines the core's register map, providing symbolic constants to access the low-level hardware.

The **altera_avalon_pio_regs.h** file is located in **bsp\drivers\inc**.

When you include the **altera_avalon_pio_regs.h** file, several useful functions that manipulate the PIO core registers are available to your program. In particular, the function

IOWR_ALTERA_AVALON_PIO_DATA (base, data)

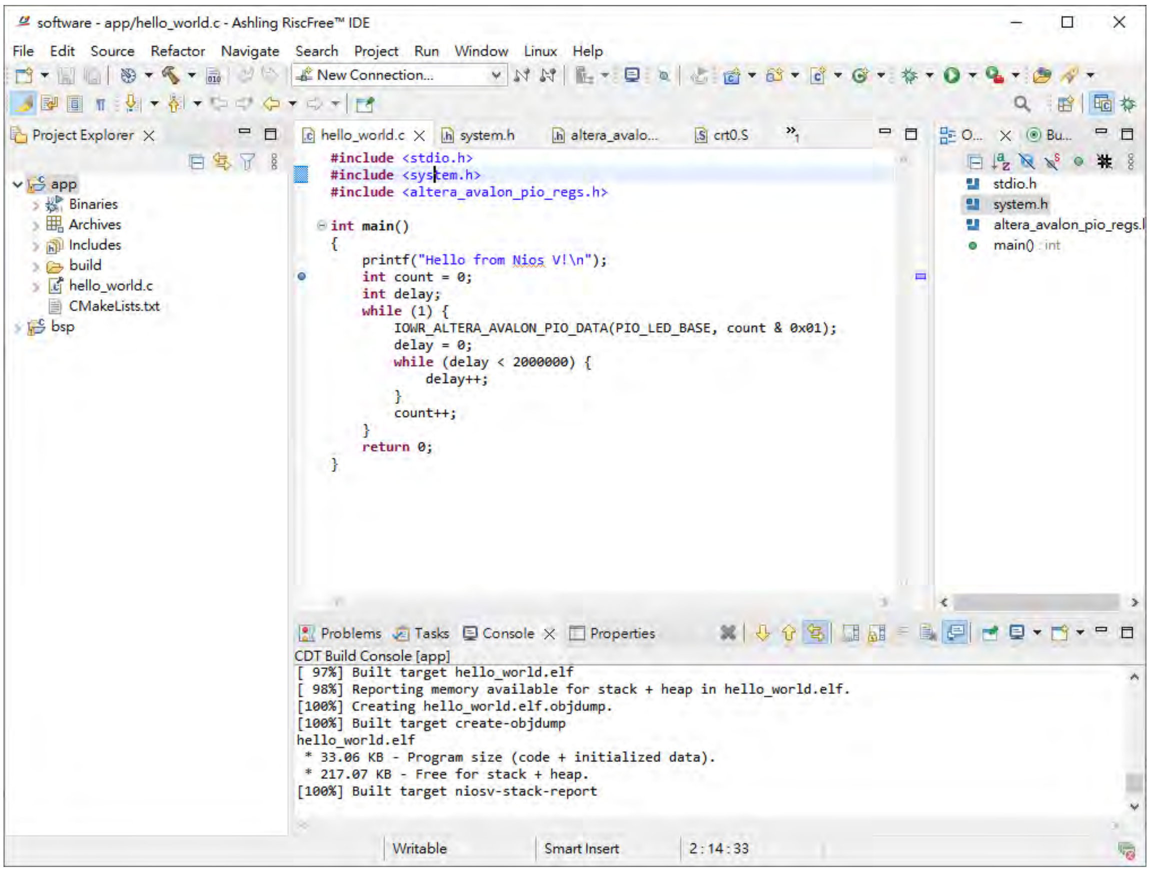
can write to the PIO data register, turning the LED on and off. The PIO is just one of many Platform Designer peripherals that you can use in a system. To learn about the PIO core and other embedded peripheral cores, refer to Embedded Peripherals IP User Guide.

When developing your own designs, you can use the software functions and resources that are provided with the Nios V HAL. Refer to the Nios V Processor Software Developer Handbook for extensive documentation on developing your own Nios V processor-based software applications.

3.5 Debugging the Application

Before you can debug a project in the Ashling RiscFree IDE, you need to create a debug configuration that specifies how to run the software. To set up a debug configuration, perform the following steps:

1. In the **hello_world.c**, double-click the front of the line which is needed to set breakpoint. See Figure 3-37.



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

int main()
{
    printf("Hello from Nios V!\n");
    int count = 0;
    int delay;
    while (1) {
        IOWR_ALTERA_AVALON_PIO_DATA(PIO_LED_BASE, count & 0x01);
        delay = 0;
        while (delay < 2000000) {
            delay++;
        }
        count++;
    }
    return 0;
}
```

```
CDT Build Console [app]
[ 97%] Built target hello_world.elf
[ 98%] Reporting memory available for stack + heap in hello_world.elf.
[100%] Creating hello_world.elf.objdump.
[100%] Built target create-objdump
hello_world.elf
 * 33.06 KB - Program size (code + initialized data).
 * 217.07 KB - Free for stack + heap.
[100%] Built target niosv-stack-report
```

Figure 3-37 Set Breakpoint

2. To debug your application, right-click the application (**app** by default) and choose **Debug as > Ashling RISC-V Hardware Debugging**.

3. If the **Confirm Perspective Switch** message box appears, click **Switch**.

After a moment, the main () function appears in the editor. A blue arrow next to the first line of code indicates that execution stopped at that line.

4. Choose **Run-> Resume** to resume execution.

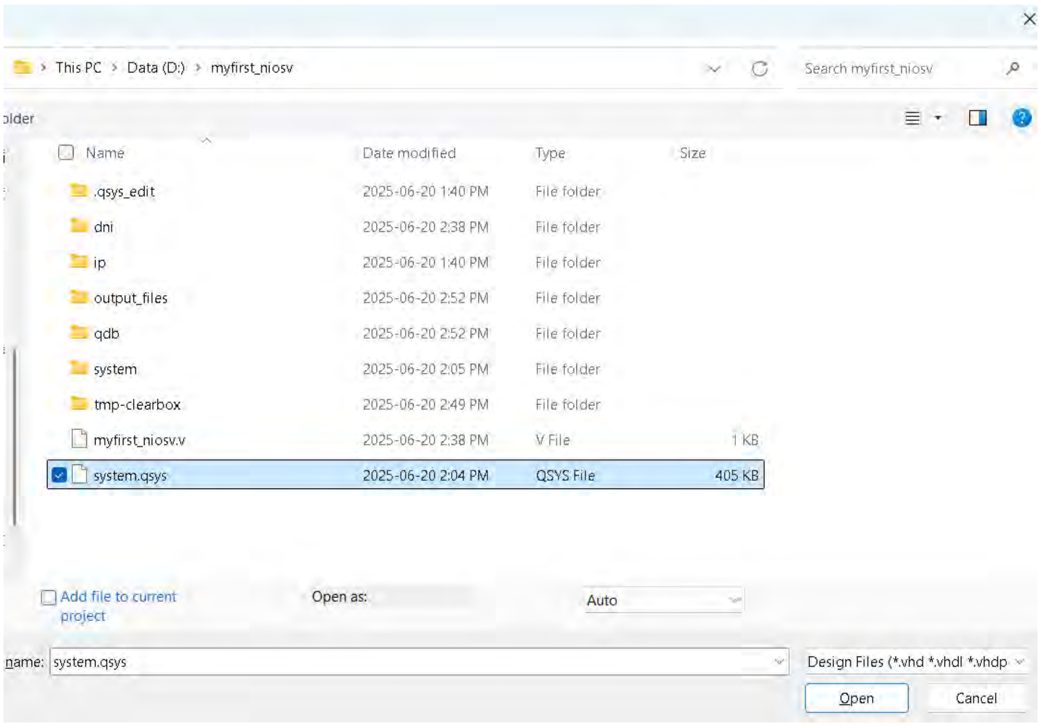
When debugging a project in the Ashling RiscFree IDE, you can pause, stop or single step the program, set breakpoints, examine variables, and perform many other common debugging tasks.

Note: To return to the C/C++ project perspective from the debug perspective, click the C/C++ icon in the top right corner of the GUI.

3.6 Configure System Library

In this section you will learn how to configure some advanced options about the target memory or other things. By performing the following steps, you can charge all the available settings:

1. In Quartus Prime, select File > Open, and open system.qsys, as shown in the figure below.



This PC > Data (D:) > myfirst_niosv

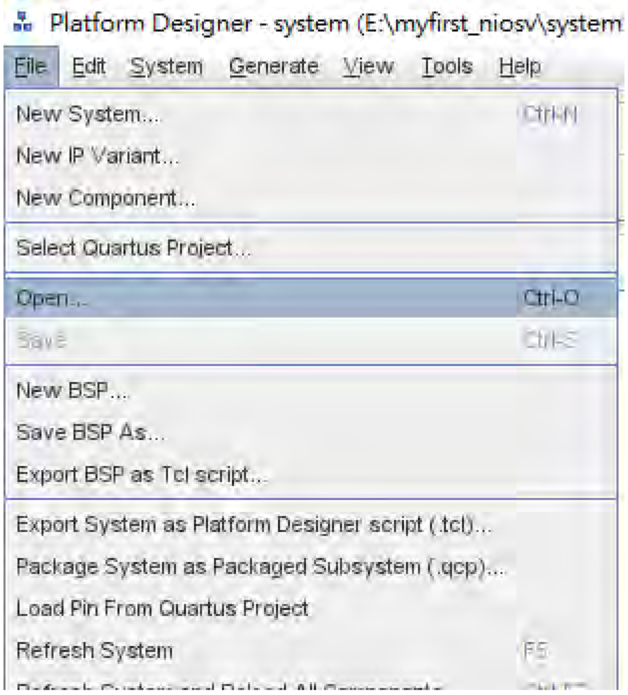
Name	Date modified	Type	Size
.qsys_edit	2025-06-20 1:40 PM	File folder	
dni	2025-06-20 2:38 PM	File folder	
ip	2025-06-20 1:40 PM	File folder	
output_files	2025-06-20 2:52 PM	File folder	
qdb	2025-06-20 2:52 PM	File folder	
system	2025-06-20 2:05 PM	File folder	
tmp-clearbox	2025-06-20 2:49 PM	File folder	
myfirst_niosv.v	2025-06-20 2:38 PM	V File	1 KB
system.qsys	2025-06-20 2:04 PM	QSYS File	405 KB

Add file to current project   Open as: Auto

name: system.qsys   Design Files (*.vhd *.vhdl *.vhdp

Figure 3-38 Opening 'system.qsys' in Quartus Prime

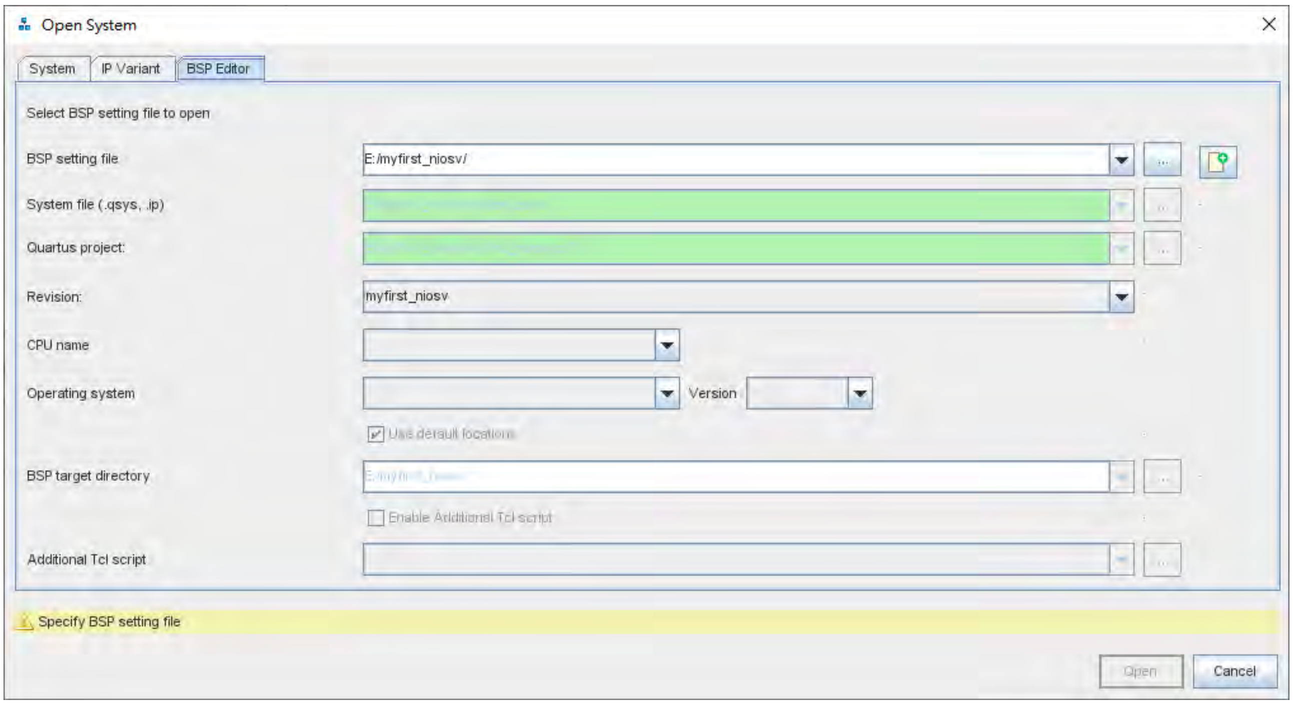
2. Click File > Open to open the BSP, as shown in the figure below.



Platform Designer - system (E:\myfirst_niosv\system
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)...
Package System as Packaged Subsystem (.qcp)...
Load Pin From Quartus Project
Refresh System F5

Figure 3-39 Opening the BSP file

3. Select the BSP Editor tab.



Select BSP setting file to open

BSP setting file: E:/myfirst_niosv/

System file (.qsys, .ip)

Quartus project:

Revision: myfirst_niosv

CPU name

Operating system Version

Use default locations

BSP target directory

Enable Additional Tcl script

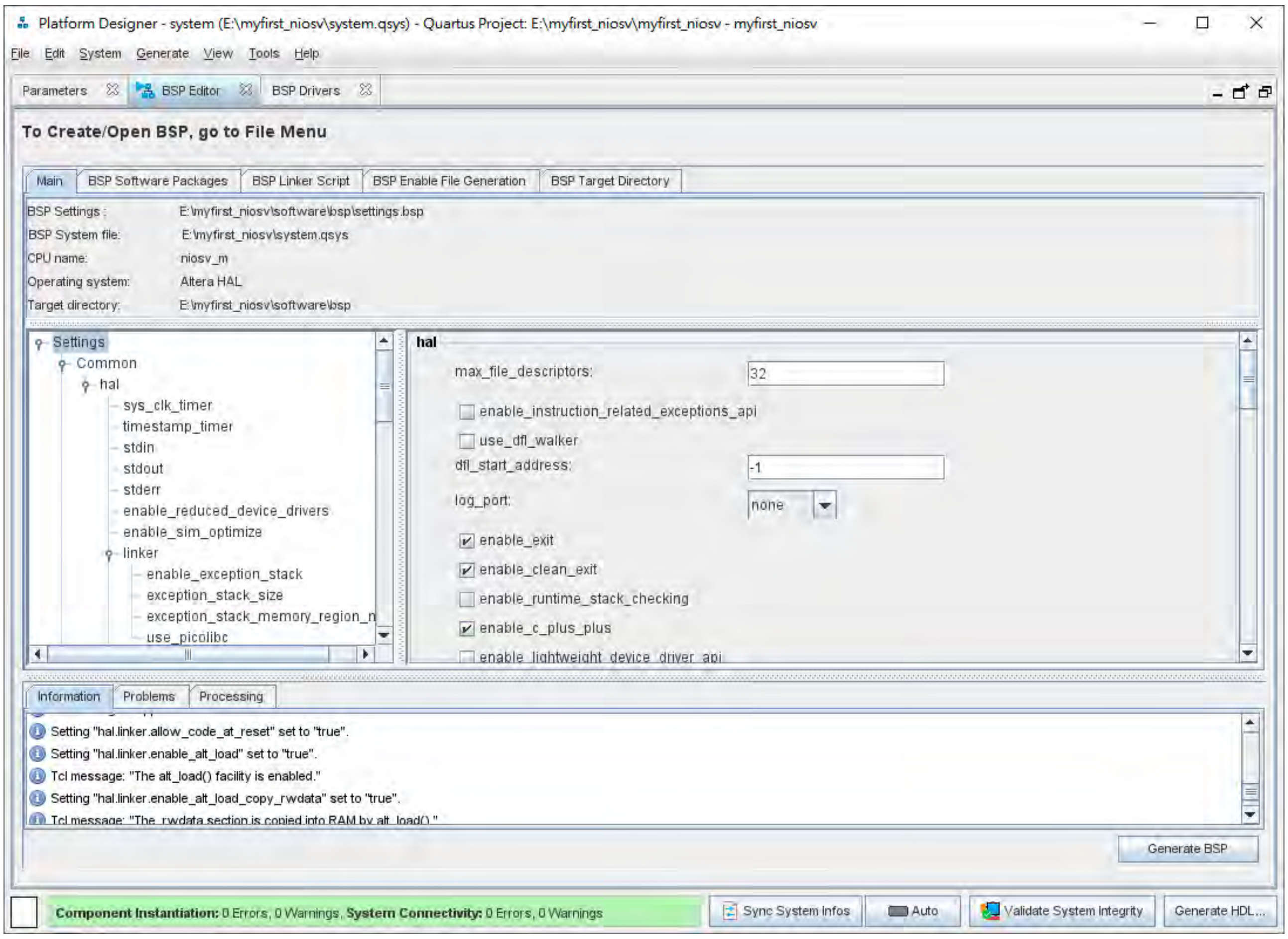
Additional Tcl script

Specify BSP setting file

Figure 3-40 Selecting the 'BSP Editor' tab

4. Browse to and open software/bsp/settings.bsp.
5. Click Open to open the BSP Editor.

6. Double-click the BSP Editor Tab to maximize the tab, as shown in the figure below.


Platform Designer - system (E:\myfirst_niosv\system.qsys) - Quartus Project: E:\myfirst_niosv\myfirst_niosv - myfirst_niosv
File Edit System Generate View Tools Help
Parameters
BSP Editor
BSP Drivers
Main
BSP Software Packages
BSP Linker Script
BSP Enable File Generation
BSP Target Directory
Information
Problems
Processing
Generate BSP
Sync System Infos
Auto
Validate System Integrity
Generate HDL...

To Create/Open BSP, go to File Menu

BSP Settings : E:\myfirst_niosv\software\bsp\settings.bsp
BSP System file: E:\myfirst_niosv\system.qsys
CPU name: niosv_m
Operating system: Altera HAL
Target directory: E:\myfirst_niosv\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

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()."
```

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

Figure 3-41 Maximizing the 'BSP Editor' tab.

7. Click Generate BSP

The following content will introduce how to configure a software project to use hardware memory. By modifying the System Library Properties in the Ashling RiscFree IDE, you can precisely configure the memory mapping.

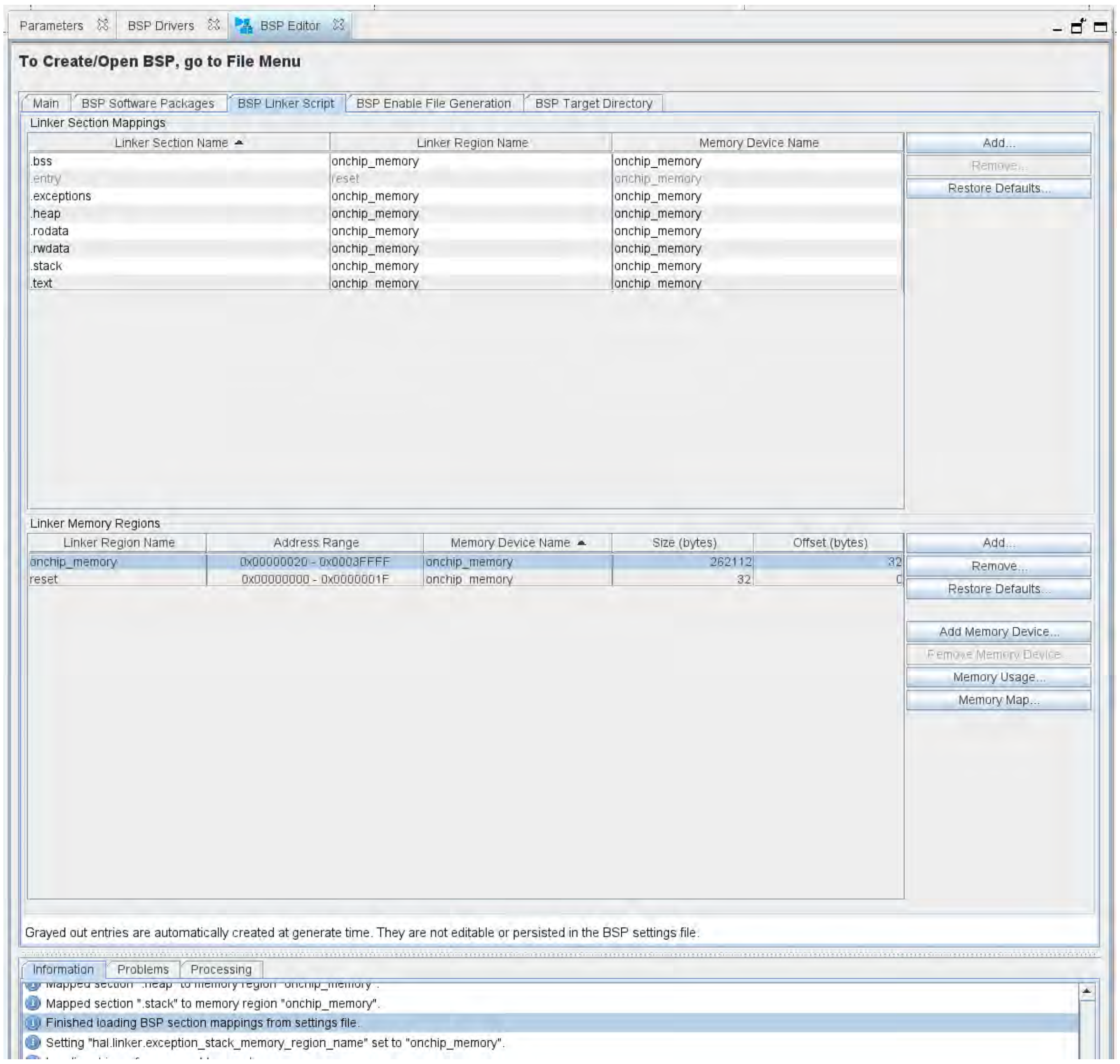
1. In the Ashling RiscFree IDE, right-click **app** and choose **System Library Properties**. The **Properties for app_syslib** dialog box opens.

2. Click **System Library**. The **System Library** page contains settings related to how the program interacts with the underlying hardware. The settings have names that correspond to the targeted NIOS V hardware.

3. In the **BSP Linker Script** tab, observe which memory has been assigned for **Block Started by Symbol memory (.bss), exception handler memory (.exceptions), Heap memory, Read-only**

data memory (.rodata), Read/write data memory (.rwdata), Stack memory and Program memory (.text), see Figure 3-42. These settings determine which memory is used to store the compiled executable program when the example **app** programs runs. In the **Main** tab, You can also specify which interface you want to use for stdio, stdin, and stderr. You can also configure build options to support C++, reduced device drivers, etc.

4. Choose **onchip_memory** for all the memory options in the **BSP Linker Script** tab. See Figure 3-42 for an example.



To Create/Open BSP, go to File Menu

Linker Section Mappings

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

Linker Memory Regions

Linker Region Name	Address Range	Memory Device Name	Size (bytes)	Offset (bytes)
onchip_memory	0x00000020 - 0x0003FFFF	onchip_memory	262112	32
reset	0x00000000 - 0x0000001F	onchip_memory	32	0

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

```
Mapped section ".heap" to memory region "onchip_memory".
Mapped section ".stack" to memory region "onchip_memory".
Finished loading BSP section mappings from settings file.
Setting "hal.linker.exception_stack_memory_region_name" set to "onchip_memory".
```

Figure 3-42 Configuring System Library Properties

5. Click **Generate BSP** to reg**enerate BSP.** Close the **Platform Designer** Windows and return to the IDE workbench.

Note: If you make changes to the system properties you must rebuild your project. To rebuild, right-click the **app** project in the **Projects Explorer** tab and choose **Build Project.**

Chapter 4

Appendix

4.1 Revision History

Version	*Change Log*
V1.0	**Initail version**

4.2 Copyright Statement