Atum A3-Nano
Getting Started with the Agilex™ 3 FPGAs
Exercise Manual

ver.25.1 pro

August 2025 rev. 1

Introductory course on the Agilex™ 3 FPGA

Exercise Manual

Contents

Before reading this manual

The contents of this document are current as of July 2025.

Note that some of the software, hardware, and operating procedures described in this document may be the same for versions or devices other than those specified.

Symbols in this document

ⓘ **Note**	Provides supplementary information.
Ⓟ **Point**	Provides important points.
Reference	Provides reference materials and sites to help you better understand the manual.
⚠ **Notes**	Notes are provided.
⃠ **prohibited**	This section contains important points to be aware of and things that should not be done.

Notations in text

Underline	Click to jump to another chapter in the document or to an external site.
Bold italic	Indicates text used in menus and windows when operating windows.
xxxxxxx↵	Indicates the command string to be input.
Shade	It shows the tools to be used.

1. Introduction

This manual is intended for users of Nios® V processors. This manual is intended to give you an idea of how to create a Nios® V system. This exercise is based on the following system configuration.

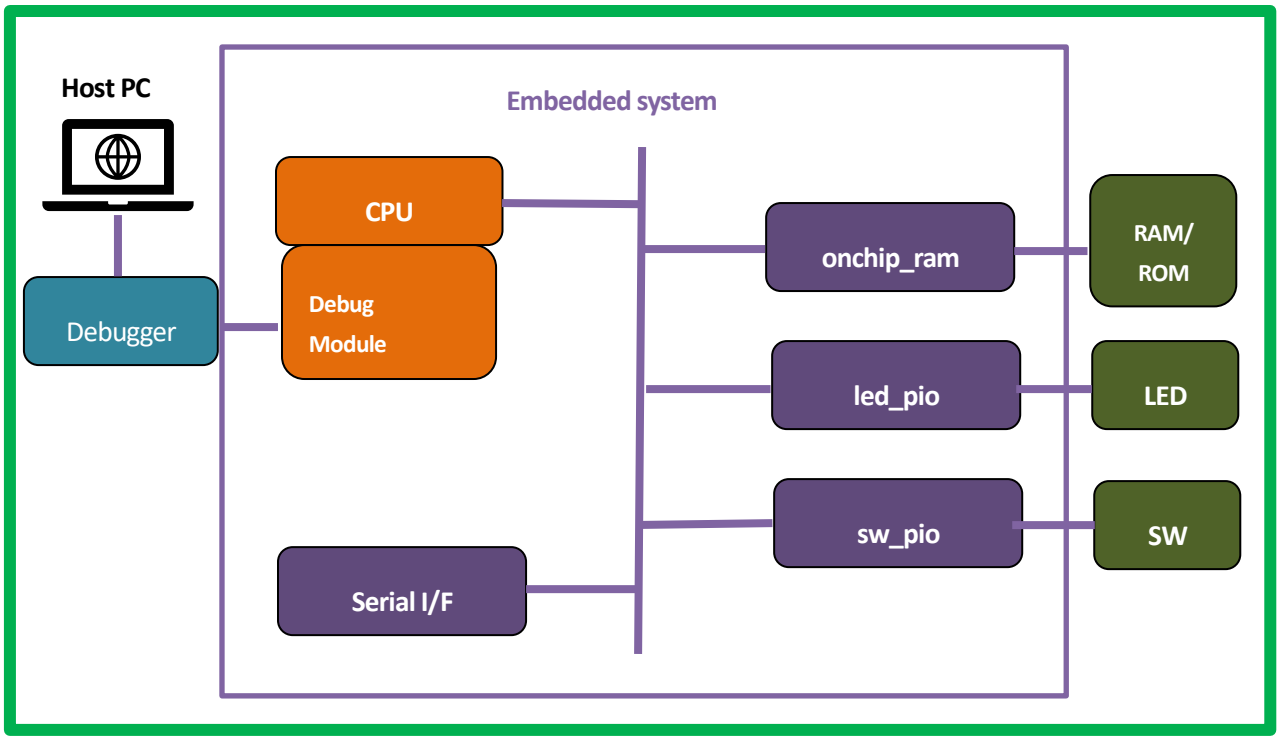

Figure 1 System configuration of the exercise subject (example of a typical embedded system)

In a typical embedded system, as shown in Figure 1, an executable file is downloaded and executed from the console host machine to the target CPU board. Downloads and simple debug commands are performed via a serial port or Ethernet. For more sophisticated debugging, a debug module on the CPU can be used via a debugger from the debug host machine.

In the configuration shown in Figure 1, a memory controller is implemented, and external memory is used as a storage and execution location for software programs.

In this exercise, we create a software program that blinks an external LED via Parallel I/O (PIO). In the system configuration shown in Figure 1, these peripherals are attached via an internal bus connected to the CPU.

Figure 2 below shows the system configuration used in the lab. We'll use the Altera® FPGA development tools as shown in Figure 2.

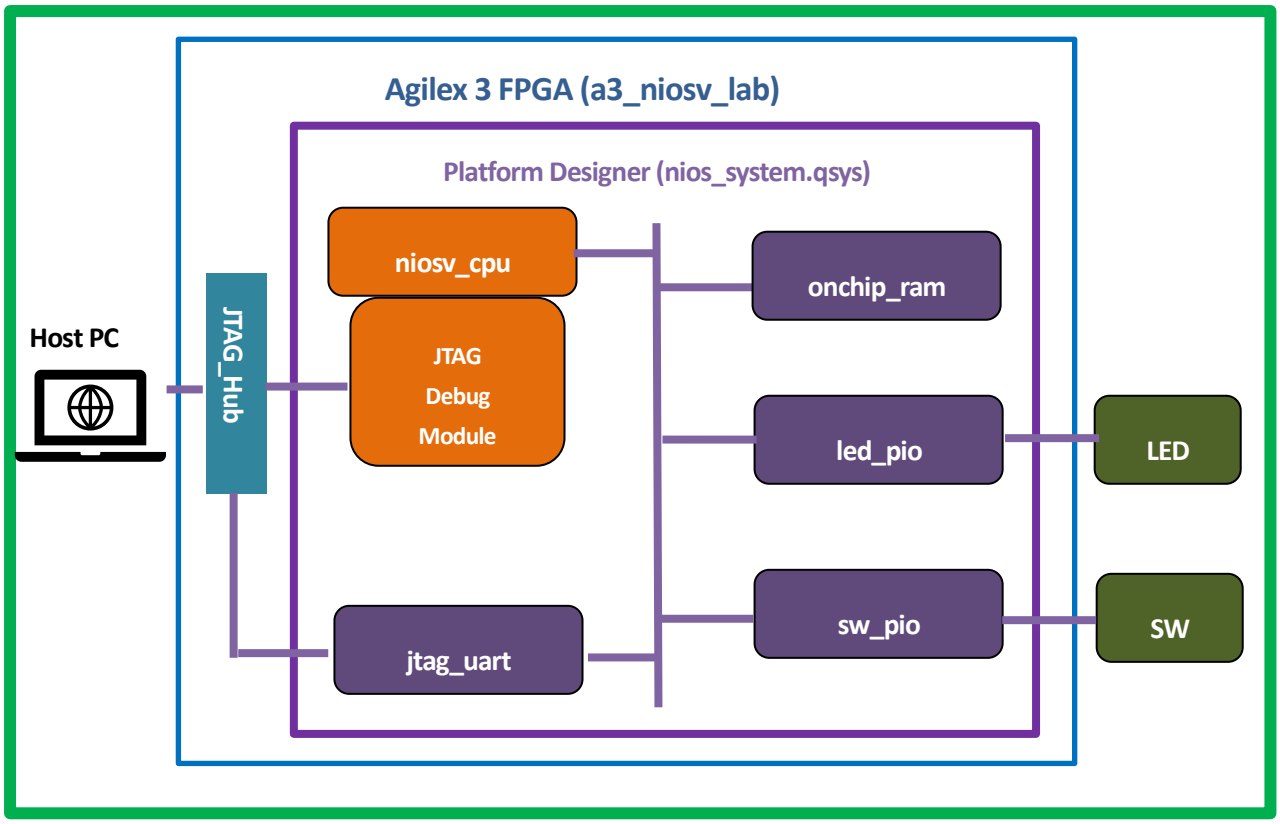

Figure 2 System Configuration of Lab Exercise(Nios ® V System)

In the configuration above, the Altera® FPGA serial port JTAG_UART (jtag_uart) enables serial communication with the host via the JTAG port using an Altera® FPGA download cable (formerly USB-Blaster™). Debugging via the JTAG debug module is also possible via the USB-Blaster™.

In this exercise, the on-chip RAM implemented in the FPGA is used as the memory and the programs are downloaded to RAM via JTAG.

In addition, the PIO and peripheral peripherals connected to the CPU using the Platform Designer interconnect, which is an internal bus for the Nios® V processor.

The embedded system configured by Platform Designer in Figure 2 is implemented in the FPGA as a block named nios_system.

2. Boards Used in Exercises

In this exercise, we will use an Atum a3-nano board manufactured by Terasic and verify the operation using the user switches and user LEDs shown below.

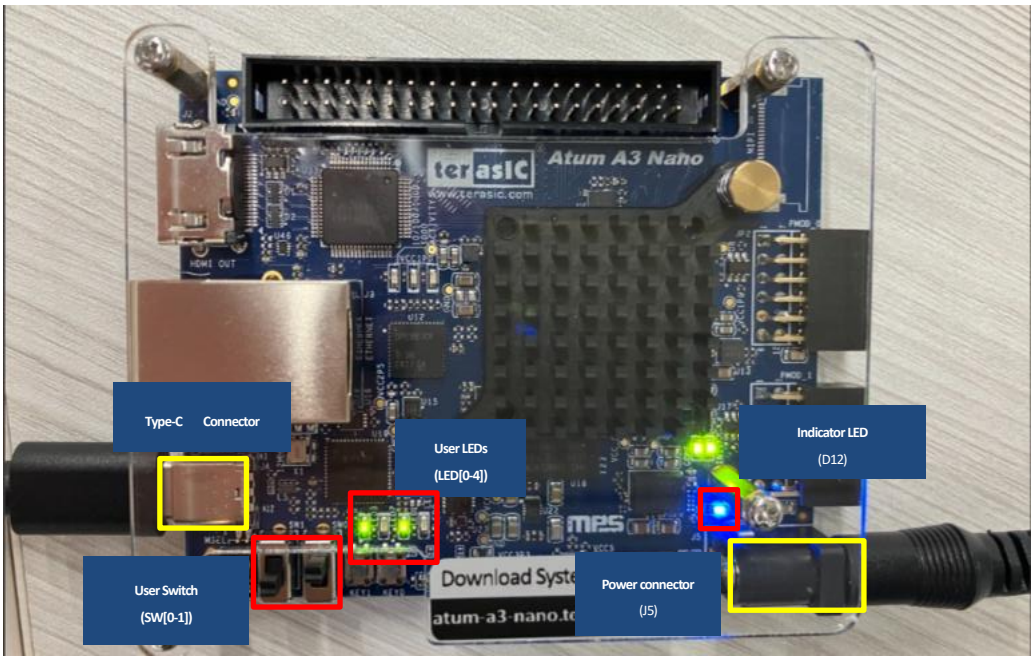

Figure 3 Atum a3-nano board

First, use the included AC adaptor to supply power. Connect it to the power connector (J5) and confirm that the 5V indicator LED lights up. Once the LED lights up, disconnect the cable from the power connector (J5).

Connect the USB cable to the Type-C connector. This development board is equipped with USB Blaster™ III, and programs can be written via JTAG by connecting it to a PC via a USB cable. It is also used as a UART terminal with the same JTAG connection.

Reference:

See the link below for more information about the board.

Terasic - All FPGA Boards - Agilex 3 - Atum A3 Nano

3. Exercise Preparation

Prepare the exercise environment.

※ If you are using a computer provided by our company, the preparation has already been completed, so please skip this section.

3-1. Preparing the Environment

Please set up the development environment referring to the preparation **web** page provided in advance.

3-2. Preparing the Exercise Data

Follow the steps below to prepare the exercise data.

1. Download ***a3_niosv_lab_v25r1_rev_1.zip***.
2. After downloading, unzip it under ***C:¥***.
3. If the ***C:¥AlteraFPGA_Lab¥a3_niosv_lab*** directory is generated, the work is complete.

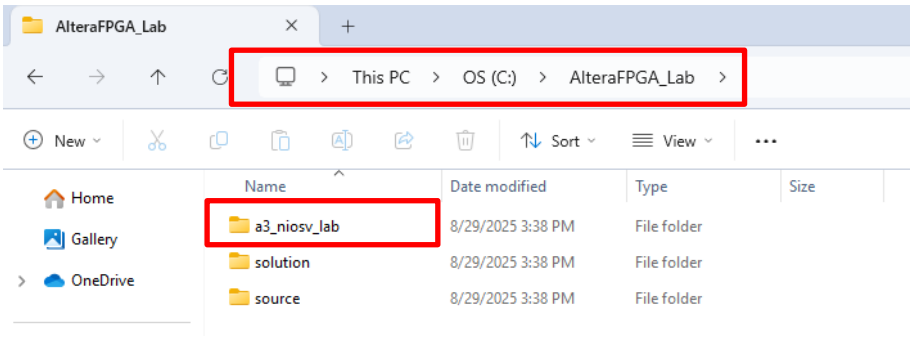

Figure 4 Unzipping Confirmation of Exercise Data

4. Creating a Platform Designer System

Create a Platform Designer system with Nios® V/g processor cores.

Figure 5 displays the system created in the exercise.

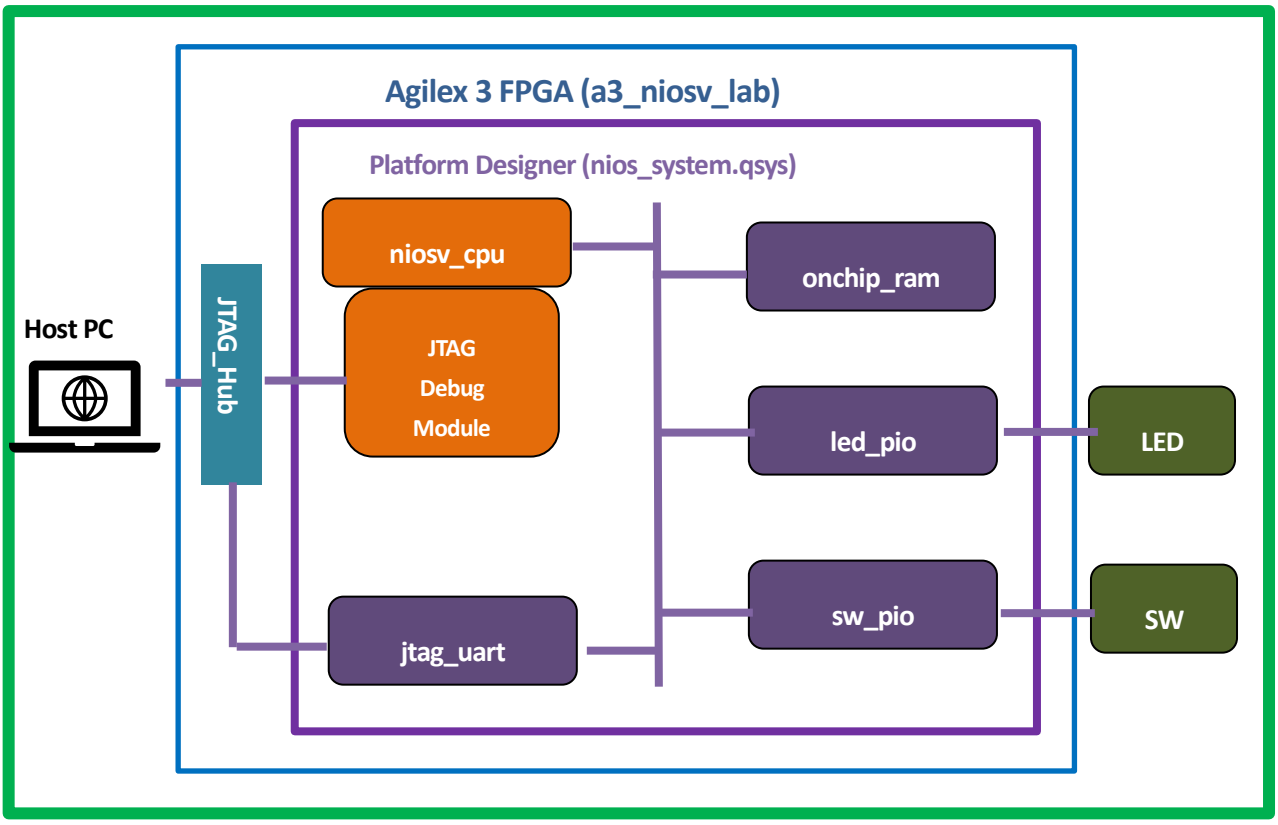

Figure 5 System to be created in this exercise (reprinted)

4-1. Opening the Quartus® Prime Project

1. Launch Quartus® Prime from the Windows Start Menu. (Our company PCs have a shortcut on the desktop.)
2. In Quartus® Prime ***File*** menu => ***Open Project***, select **golden_top.qpf** under "***C: ¥AlteraFPGA_Lab¥a3_niosv_lab***" and click Open. If a pop-up appears, click *Yes*.

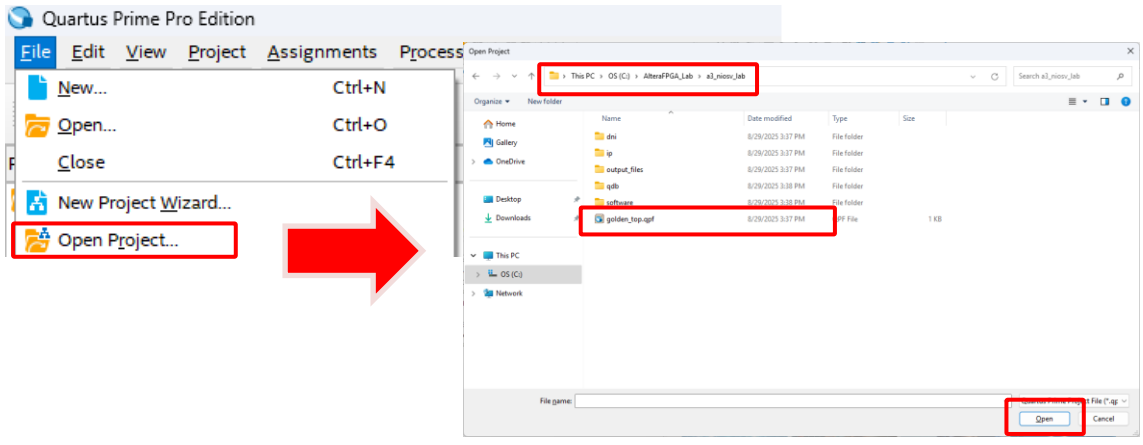

Figure 6 Opening the Quartus® Prime Project

4-2. Launching the Platform Designer

1. From the ***Tools*** menu of Quartus® Prime, select ***Platform Designer*** or click the ***Platform Designer icon***  to launch ***Platform Designer***.

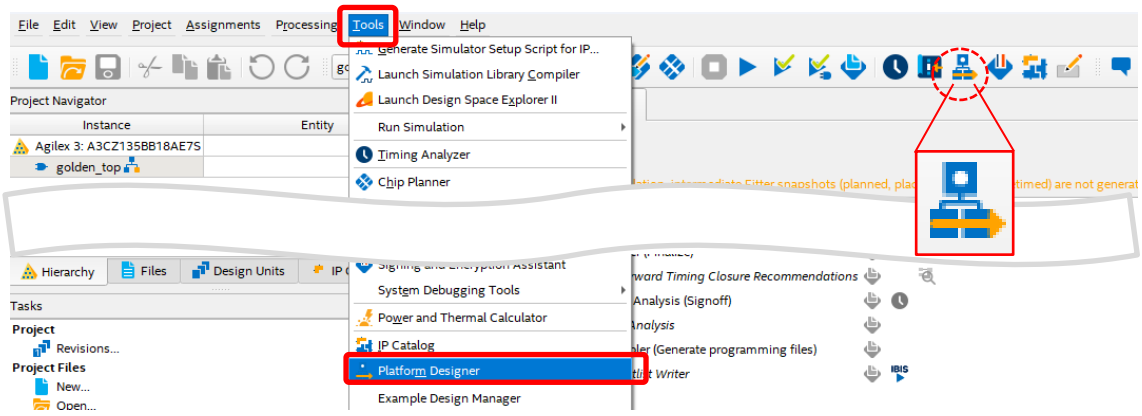

Figure 7 Launching the Platform Designer

2. Once Platform Designer is launched, open Platform Designer System.

   Select the [...] button to display the options. It's to the right of the Platform Designer System: line.

   Select ***nios_system.qsys*** as shown in [Figure 8] and click the [Select] button. Then click the ***Open*** button.

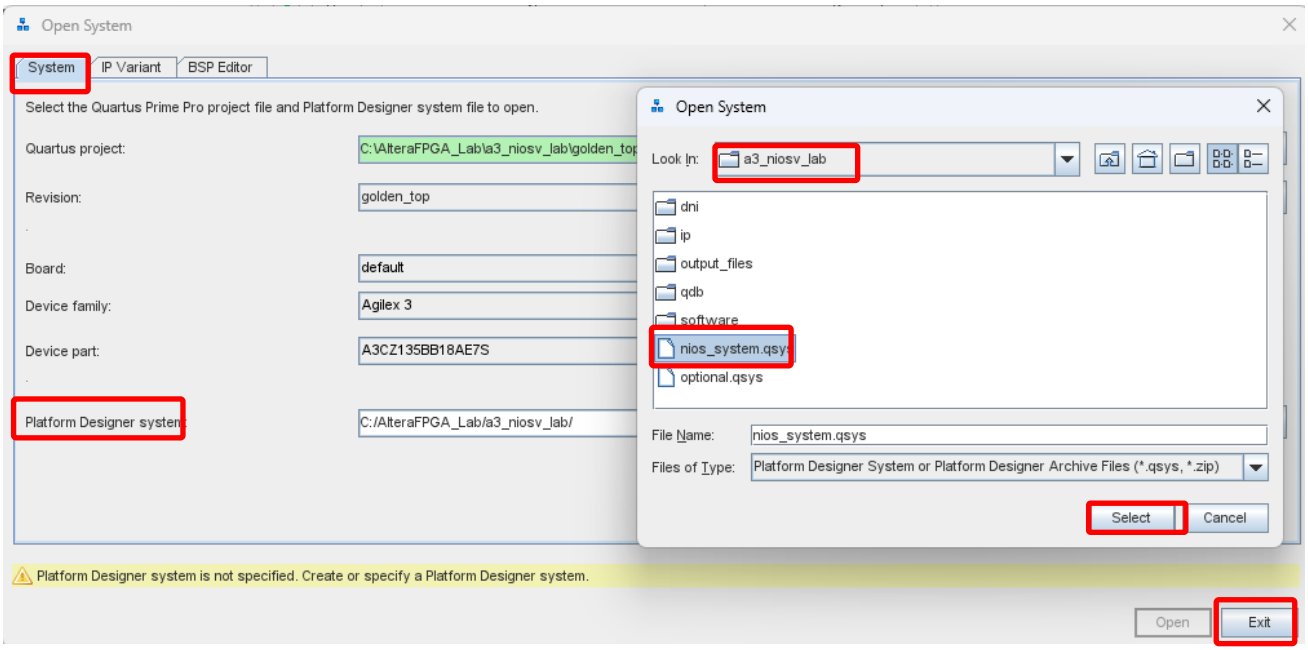

Figure 8 Open Platform Designer System

3. When the Open System Completed dialog appears, click the ***Close*** button.

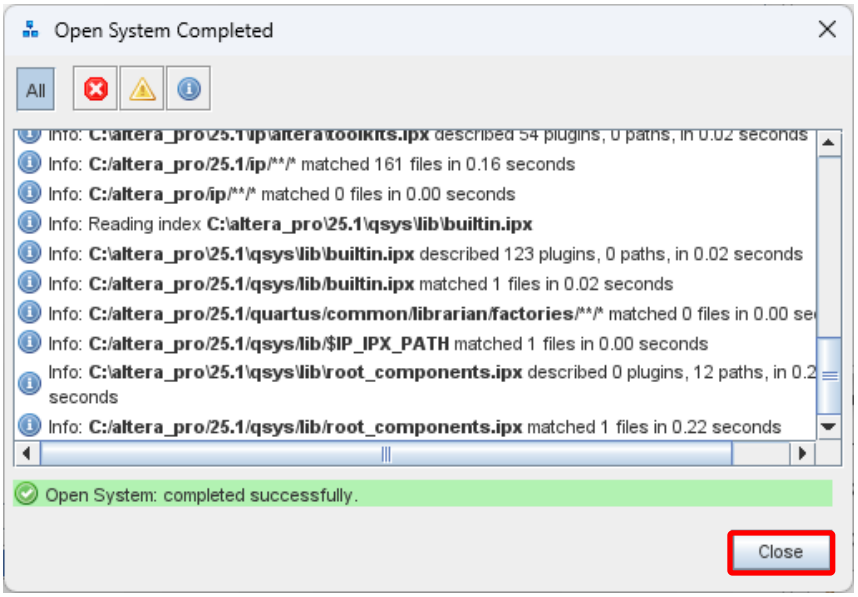

Figure 9 Close the Open System Completed dialog

4-3. Adding a Nios® V Processor Core

1. Select the ***Nios V/g General Purpose Processor Intel FPGA IP*** under ***Processors and Peripherals => Embedded Processors*** from the ***Library*** entry in the ***IP Catalog*** tab located in the upper left of Platform Designer and click the ***Add*** button.

> ⓘ Note:
> You can easily find the component by typing the word into the search input (boxed magnifying glass) in the *IP Catalog* tab. For this exercise, type niosv.

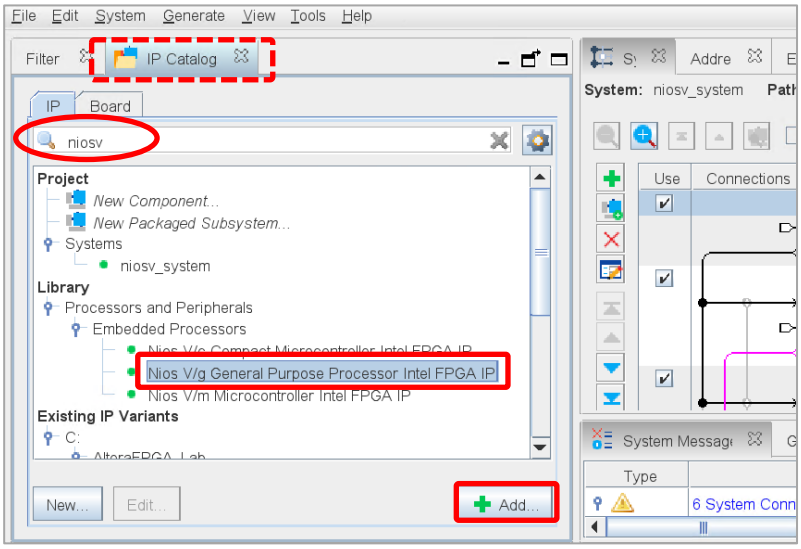

Figure 10 Adding a Nios® V Processor Core

2. The Nios® V processor configuration screen appears. Check ***Enable Debug (***checked by default) and ***Enable Reset from Debug Module***, and leave the other settings as they are. Then click the ***Finish*** button.

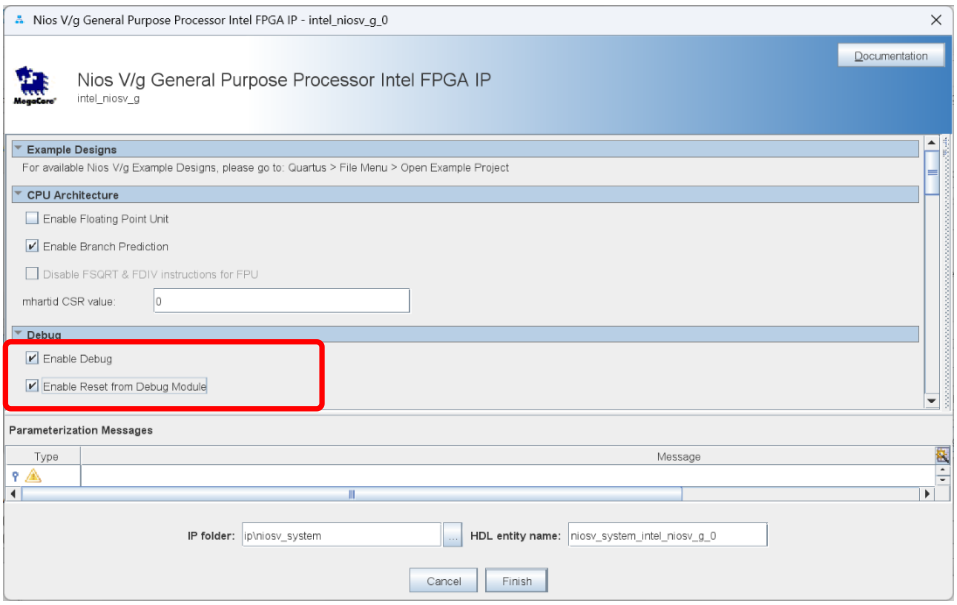

Figure 11 Nios® V Parameter Setting Screen

> ⚠ **Note:**
> If you click the Finish button, an error will appear in the Platform Designer Message window at this point.

3. To improve the readability of the design, change the display position of the Nios® V processor. After selecting ***intel_niosv_g_0***, click the ▲ symbol at the left edge of the ***System View*** tab several times to move it to just below ***pll***.

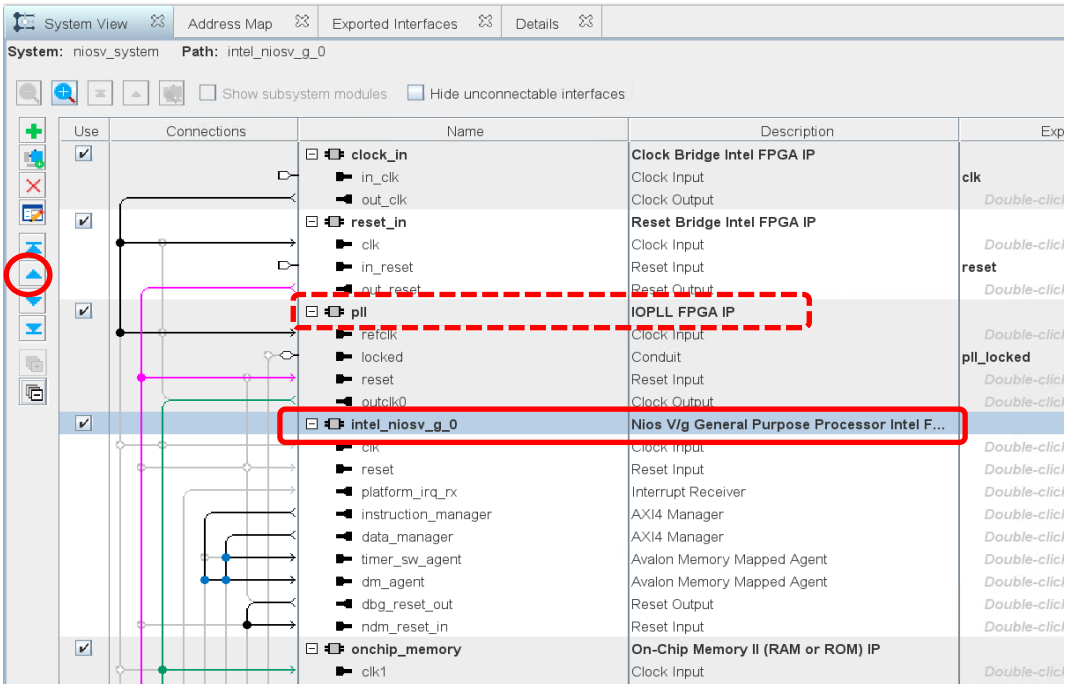

Figure 12 Changing the Nios® V Processor Display Position

4. In the ***System View*** tab, right-click ***intel_niosv_g_0*** (Name item) and select ***Rename***.After selecting, change the name from ***intel_niosv_g_0*** to ***cpu***.

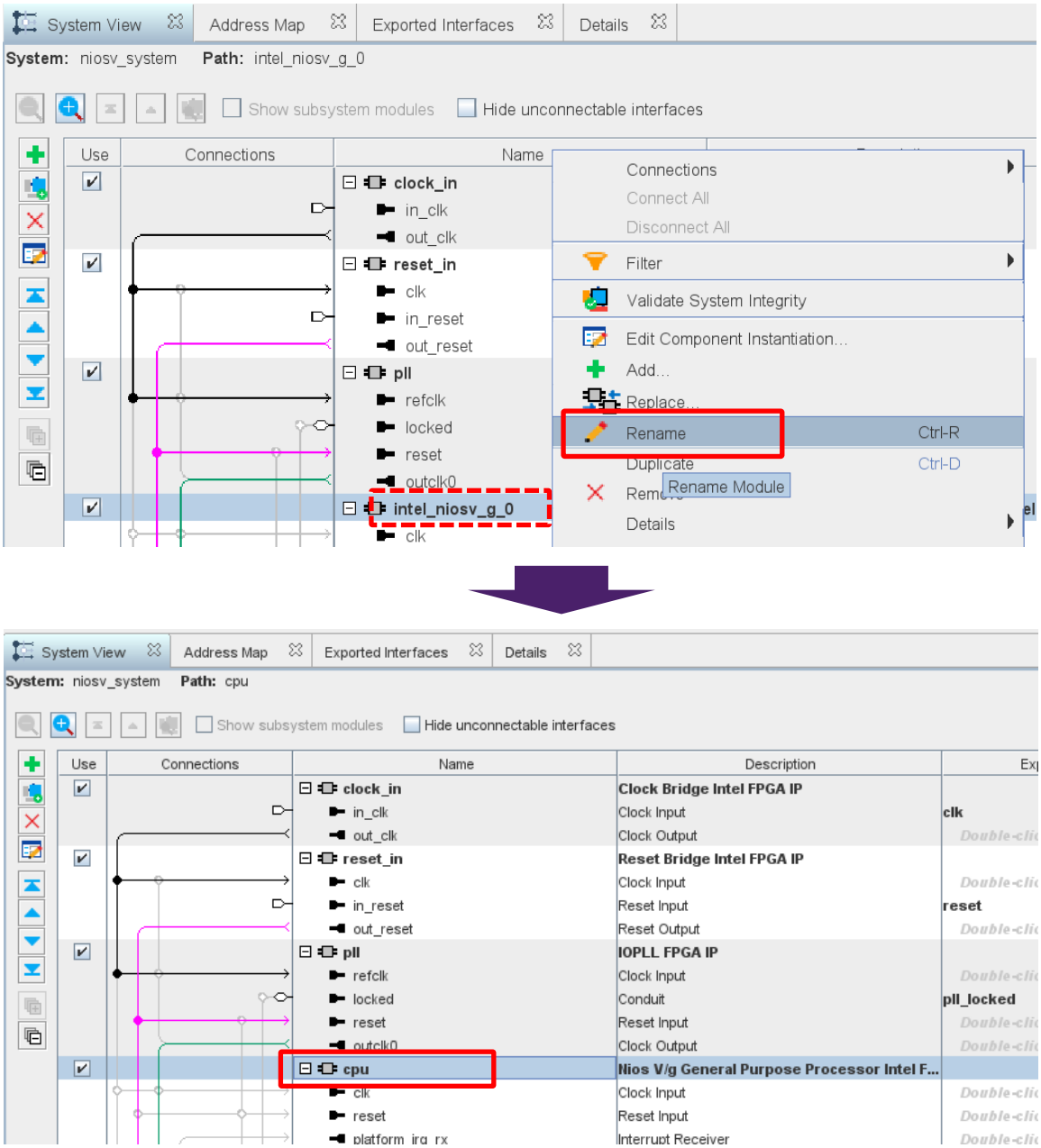

Figure 13 Changing the Module Name

4-4. Connecting the Nios® V processor to various IPs and setting Reset Agent and Peripheral Regions

1. Connect the Nios® V processor to various IPs. Referring to Table 1 and Figure 14, connect each IP to the ***instruction_manager*** and ***data_manager*** of the cpu. For the connection method, the contact points of each bus shown in [Table 1] are indicated by a white circle (O). Click with the mouse to make it a black circle (●) to connect the IPs. As shown in Figure 15, you can right-click each port name and select ***Connections: < port name >***.

Table 1 Nios® V and Various IP Connections

Name	Port Name	Connection Destination
cpu	instruction_manager	cpu.dm_agent
		onchip_memory.s1
	data_manager	cpu.timer_sw_agent
		cpu.dm_agent
		onchip_memory.s1
		jtag_uart.avalon_jtag_slave
		led_pio.s1
		sw_pio.s1
		button_pio.s1

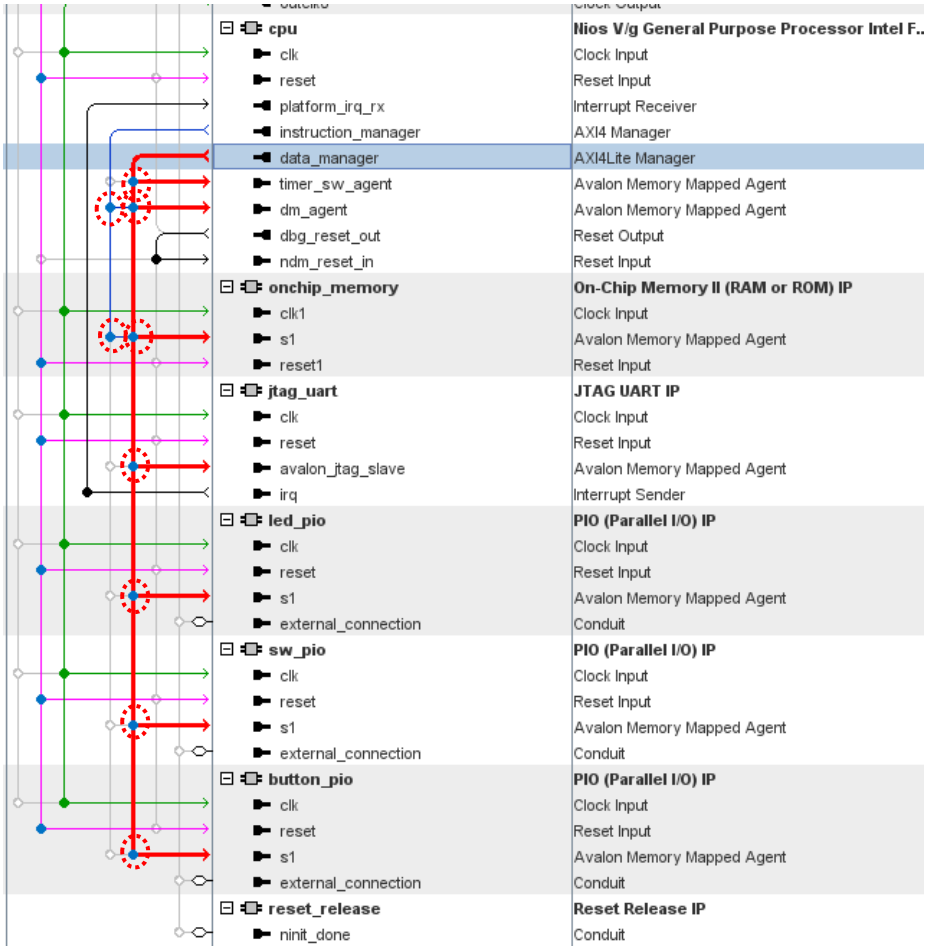

Figure 14 Nios® V and Memory Connections

> ⚠ **Note:**
>
> In the figures in this document, each bus is colored for better visibility.
> The default black bus on the lab system works fine.

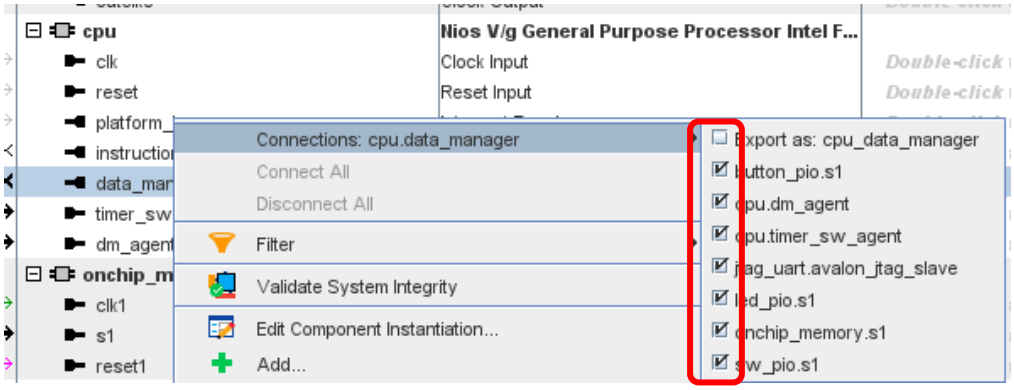

Figure 15 Connections

2. Change the base address of the ***cpu timer_sw_agent*** and ***dm_agent***. This is done in order to place them within the Peripheral Regions set next.

   Double-click the ***Base Address*** of the corresponding port to change it. Change it as follows:

   timer_sw_agent : **0x0081_0000**

   dm_agent :0x0080_0000

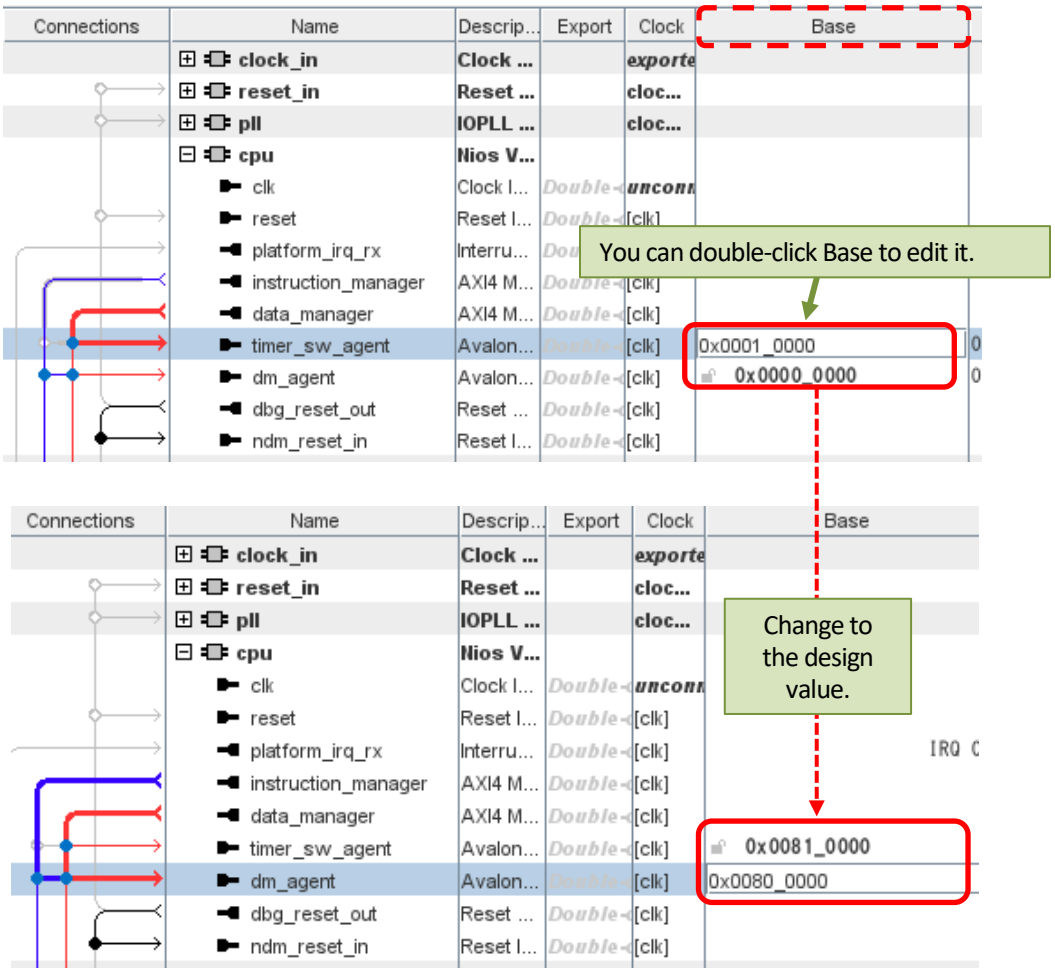

Figure 17 Changing the Base Address

3. After changing the base address, click the ***Sync System Infos*** at bottom right of the Platform Designer UI.

4. Set the Peripheral Regions. For Nios® V/g which implements cache, when performing non-cache access, the target area must be set as Peripheral Regions. Click the ***Address Map*** tab in Platform Designer and check the address map. In our system, the target area for non-cache access is all areas except On-Chip RAM. As shown in [Figure 18], the IO area is placed between 0x0080_0000 and 1MB area.

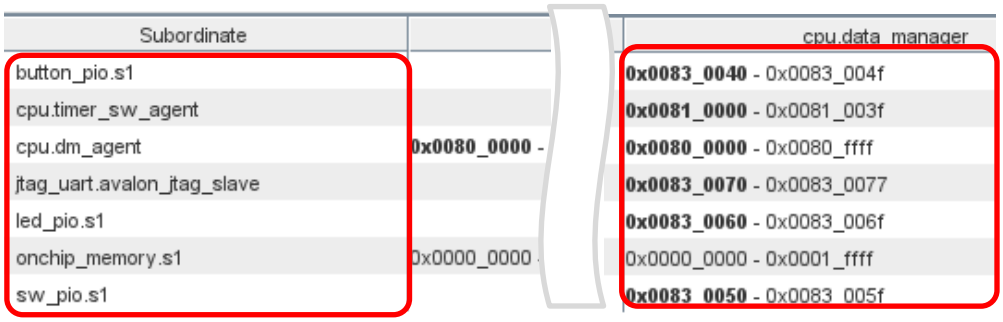

Figure 18 Checking the peripheral IO area

5. Return to the System View tab, highlight ***cpu***, right-click, and select ***Edit Parameters...***. From the Nios® V processor core parameter setting GUI window, make the following settings in the ***Peripheral Region A tab of the Peripheral Regions*** item. In our system, non-cache access is performed on all areas except On-Chip RAM. Referring to Figure 20, set the IO area between 0x0080_0000 and 1MB.

Peripheral Region A Size : **1 MBytes**

Peripheral Region A Base Address : **0x00800000**

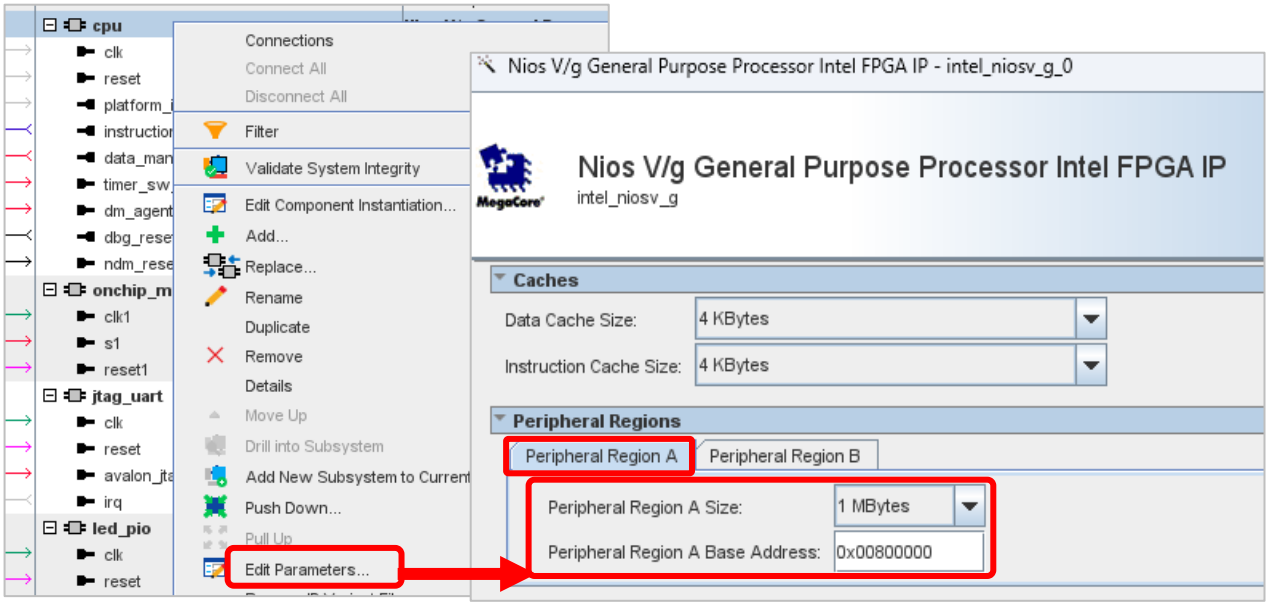

Figure 20 Configuring Peripheral Regions

6. Configure the Reset Agent. After highlighting ***cpu***, right-click and select Edit Parameters…. The GUI screen for setting parameters for the Nios® V processor core appears again.Select onchip_memory.s1 from the ***Traps, Exceptions, and Interrupts Reset Agent*** pull-down menu. Set ***Reset Vector Offset*** to **0x00000000**. After making the settings, click the [***Finish***] button.

> ⚠ **Note:**
>
> If onchip_memory.s1 is not displayed, select another parameter and check the pull-down menu again. If it is not displayed in this way, onchip_memory.s1 may not be connected to the instruction manager. Check the connection again.

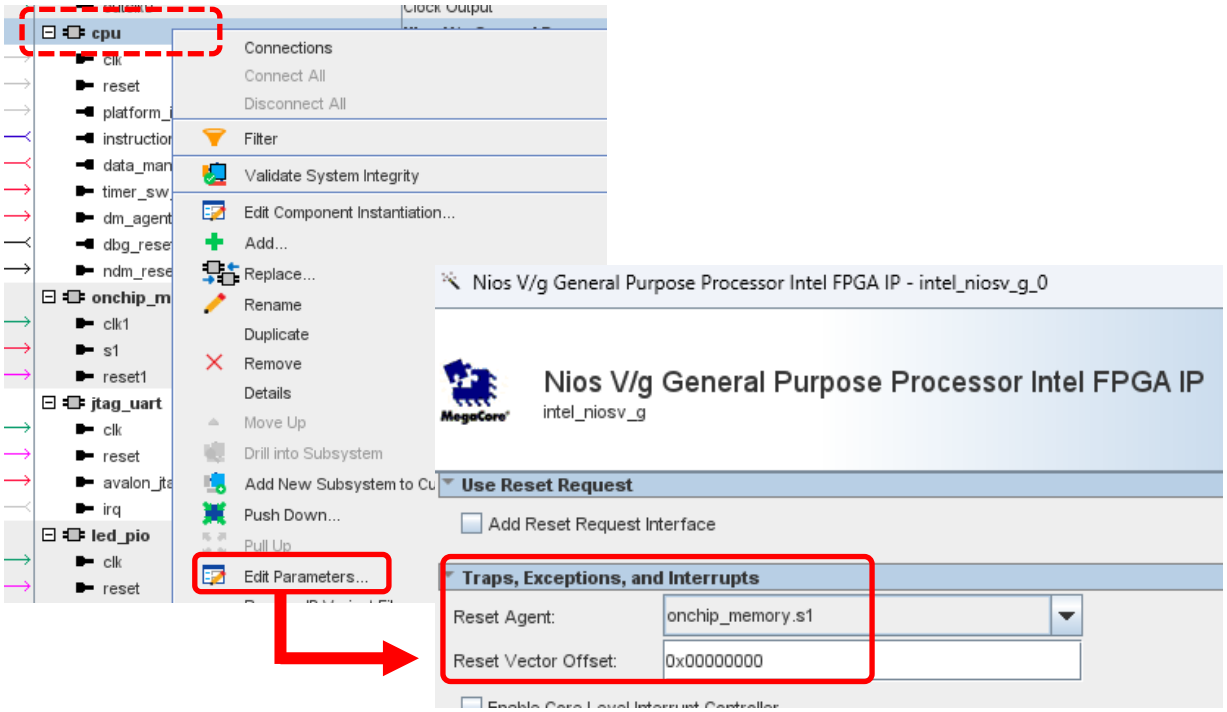

Figure 21 Reset Agent Settings

4-5. Clock, Reset and Interrupt Connections

1. Make the clock and reset connections. Return to the ***Platform Designer System View*** tab, and make the clock and reset connections.

   Refer to [Table 2] and [Figure 22] to make connections in the ***"Connections"*** section.

> ⚠ **Note:**
>
> Signals other than Nios® V are already connected.

Table 2 Clock and Reset Connections

Name	Port Name	Destination
clock_in	out_clk	reset_in.clk
		pll.refclk
reset_in	out_reset	pll.reset
		cpu.reset
		cpu.ndm_reset_in
		onchip_memory.reset1
		jtag_uart.reset
		led_pio.reset
		sw_pio.reset
		button_pio.reset
pll	outclk0	cpu.clk
		onchip_memory.clk1
		jtag_uart.clk
		led_pio.clk
		sw_pio.clk
		button_pio.clk
cpu	dbg_reset_out	cpu.ndm_reset_in
		onchip_memory.reset1
		jtag_uart.reset
		led_pio.reset
		sw_pio.reset
		button_pio.reset

2. Configure the interrupt connection from JTAG_UART to the Nios® V processor.

Referring to [Figure 22], click the white circle of ***jtag_uart*** in the ***IRQ*** column. At the same time, make sure that the ***irq*** of ***jtag_uart*** in ***Connections*** is connected to the ***platform_irq_rx*** of ***cpu***.
The numbers in the red frame IRQ column indicate the priority of interrupts to the CPU, with lower numbers indicating higher priority.

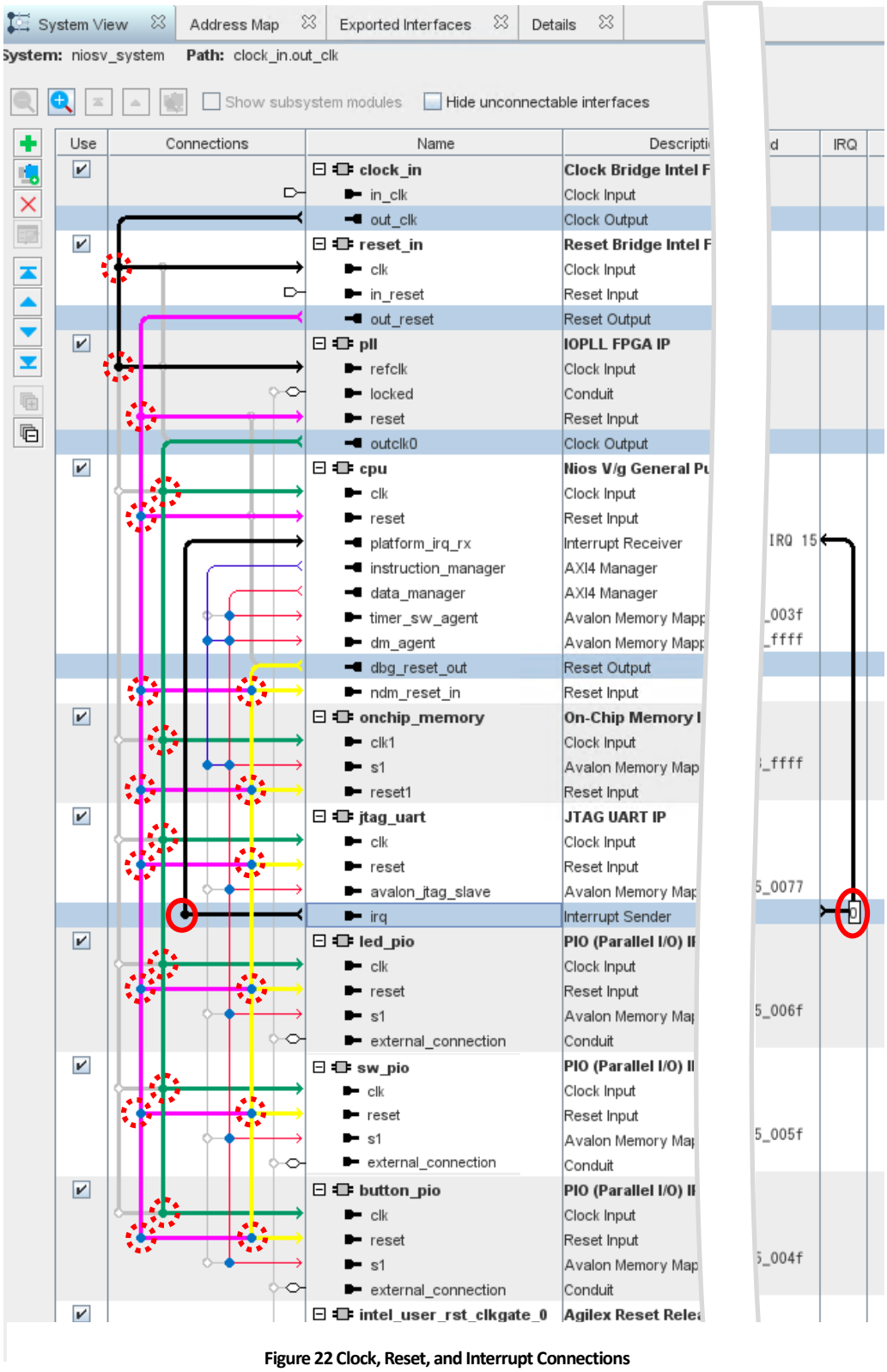

Figure 22 Clock, Reset, and Interrupt Connections

4-6. Platform Designer System Module Generation

1. Click the ***Sync System Infos*** button to bring System Information up to date.

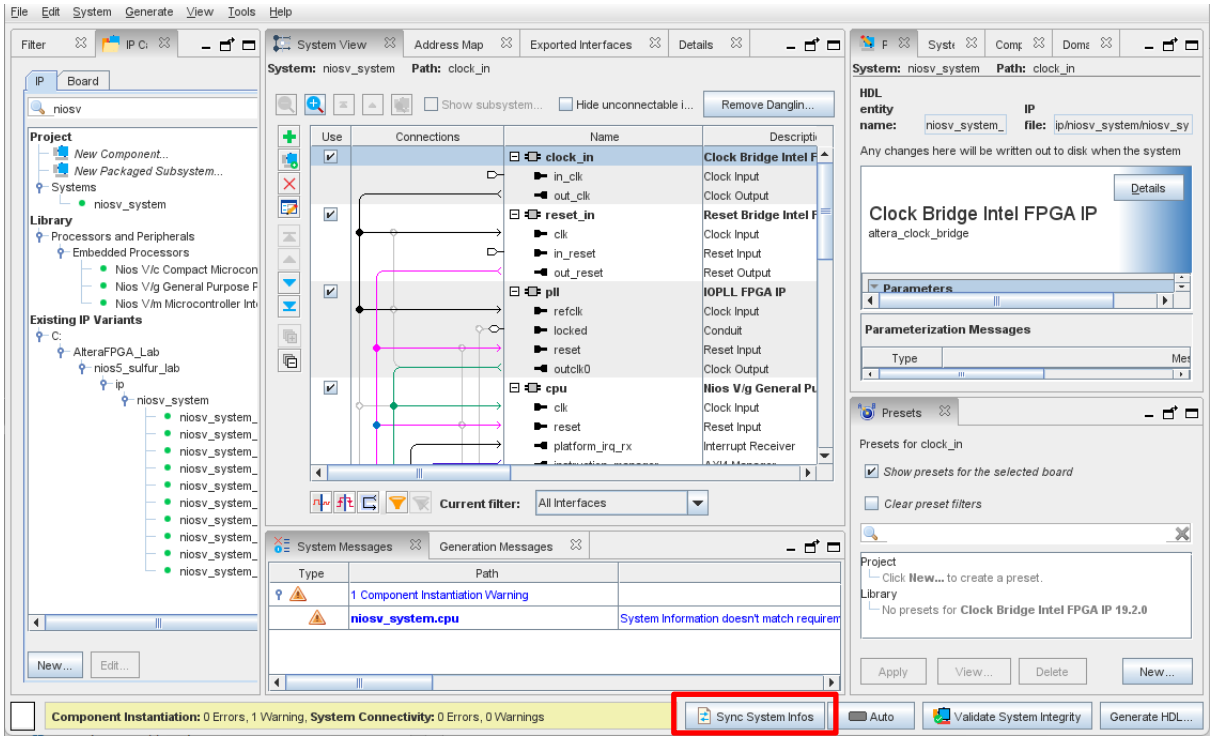

Figure 23 Sync System Infos

2. To save the Platform Designer configuration file, select ***File*** => ***Save*** and save the ***nios_system.qsys*** file. When the ***Save System Completed*** window appears, click the [***Close***] button.

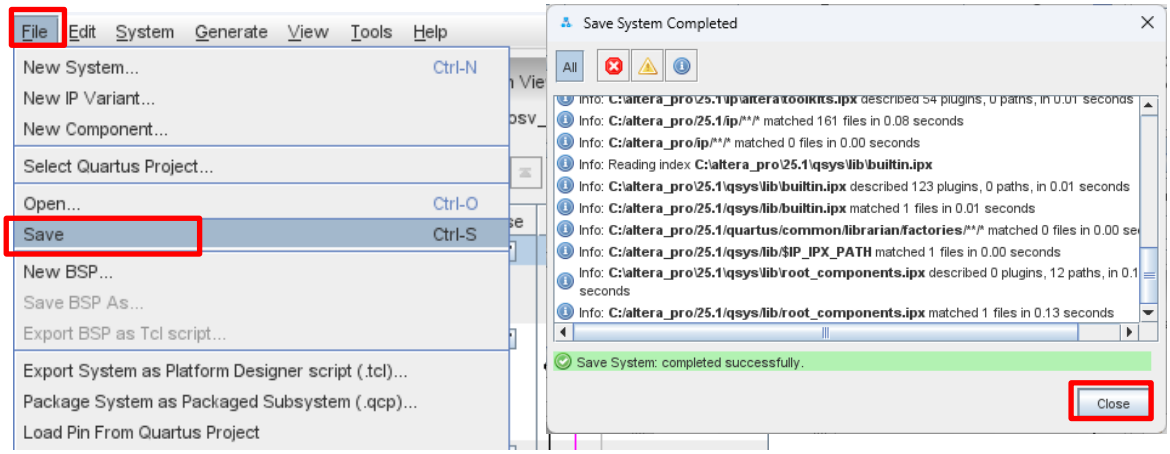

Figure 24 Platform Designer Save System

3. Select ***Generate*** menu => ***Generate HDL*** in Platform Designer to display the ***Generation*** window. Click the ***Generate*** button at the bottom right to generate the system. When the ***Generate Completed*** screen appears, click the ***Close*** button.

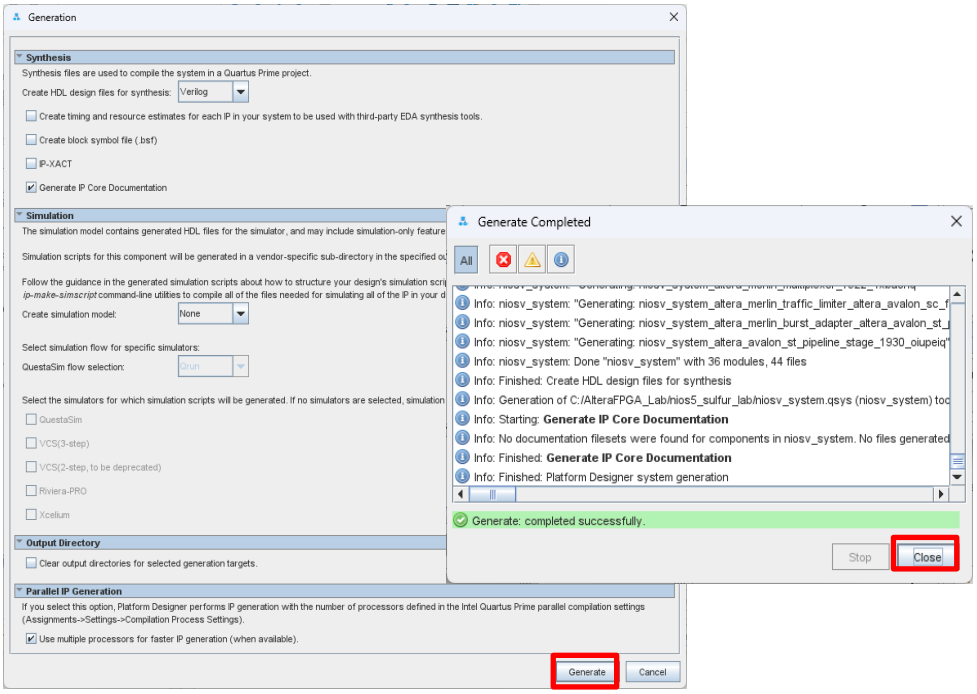

Figure 25 Generation window

4-7. Generating a template for implementing the Platform Designer system

1. From the ***Platform Designer*** main window, select ***Generate*** menu => ***Show Instantiation Template....*** The ***Instantiation Template*** window will appear, and the template for implementing the ***Platform Designer*** system on ***HDL*** in the ***Top*** hierarchy will be displayed.
   You can select a template in ***Verilog*** or VHDL from the ***HDL Language*** in the red frame of the Instantiation Template window, and copy it to the clipboard using the ***Copy*** button at the bottom right.

   In this exercise, the template has already been implemented on golden_top.v.

Figure 26 Display of the Instantiation Template

2. After closing the ***Instantiation Template*** window, use the ***File*** menu => ***Exit*** from the Platform Designer main window to close ***Platform Designer***.

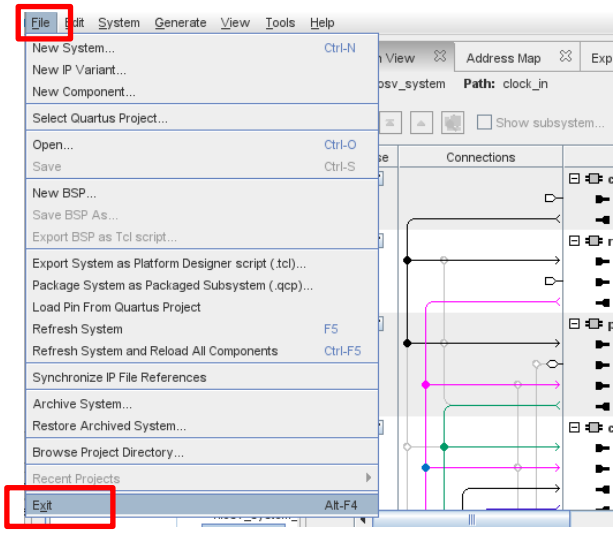

Figure 27 Exit Platform Designer

5. Creating the Hardware Design

Now that the design with Nios® V implementation is complete, we will generate the FPGA configuration file.

5-1. Various settings for FPGA hardware development.

In this exercise, we have provided a project in which all necessary operations on Quartus Prime have been performed before compiling, such as implementing the Platform Designer system, editing HDL, and setting constraints.

However, please note that the development flow described in the following link is usually required.

Altera® FPGA Development Flow - Semiconductor Business - Macnica

5-2. Compilation

Click the ***Start Compilation*** button from the main Quartus® Prime window or run it from the ***Processing*** menu.

The compilation was successful if you see a red message in the Messages window [Figure 29].

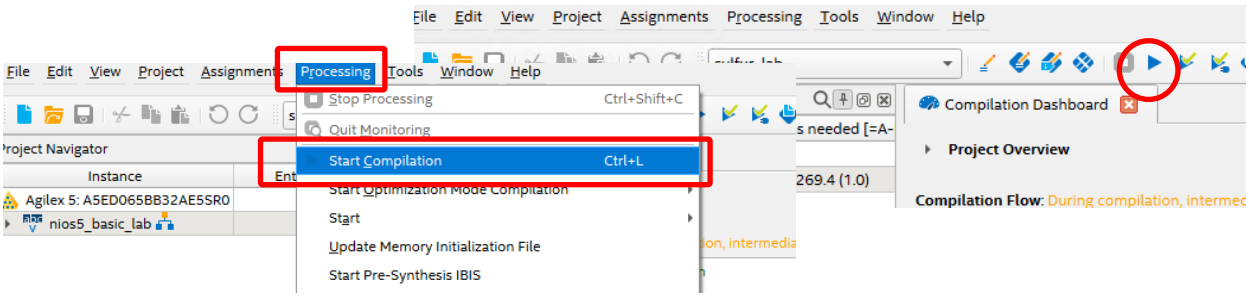

Figure 28 Start Compilation

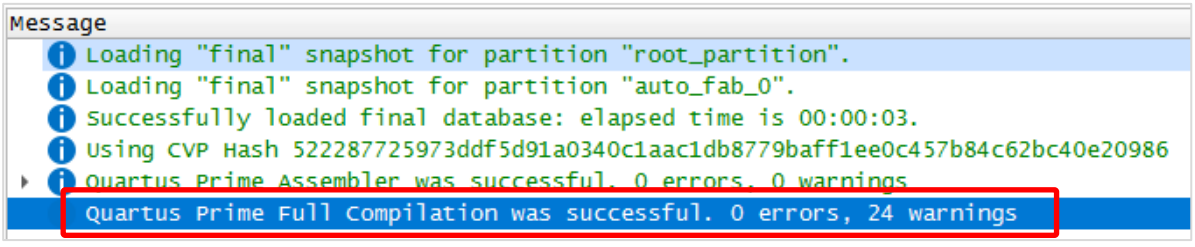

Figure 29 Compiling Complete

When the Timing Analyzer screen appears, click ***X*** to exit Timing Analyzer.

After compiling, check ***C:¥AlteraFPGA_Lab¥a3_niosv_lab¥output_files*** for golden_top.sof.

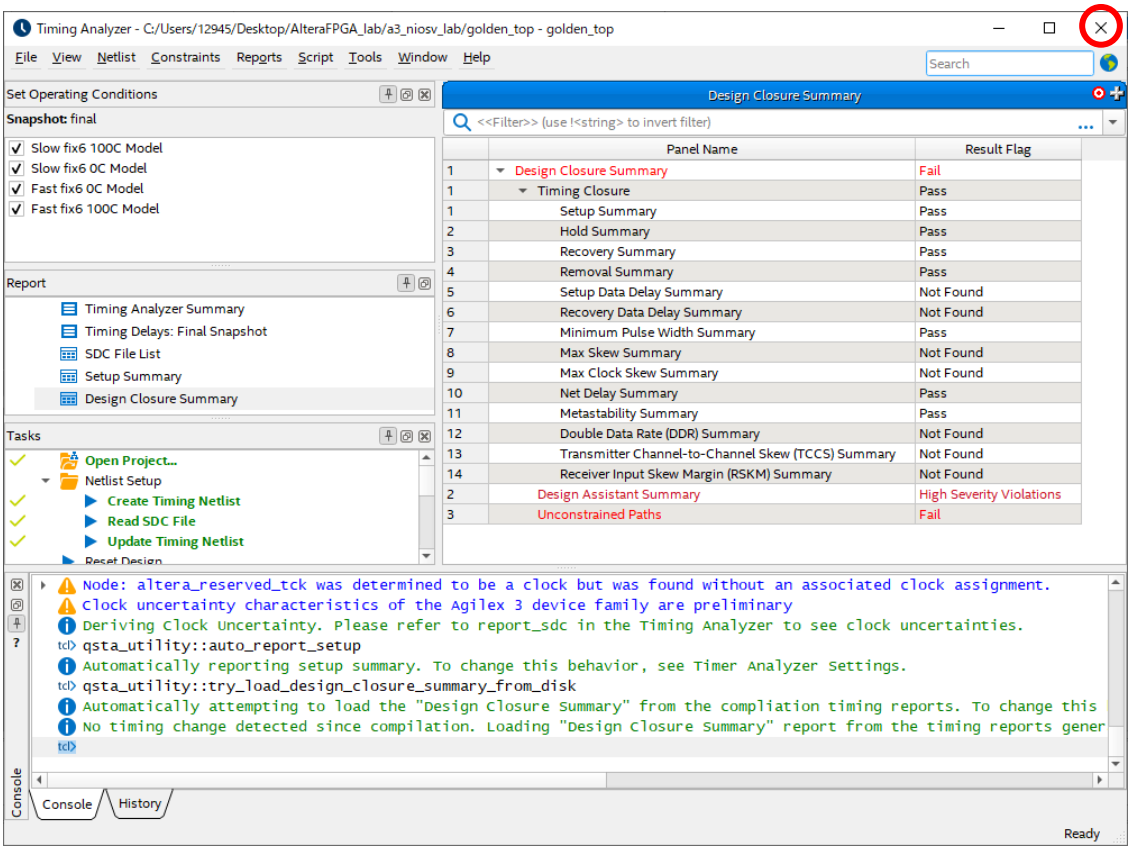

Figure 31 Timing Analyzer screen

> **⚠ Note:**
>
> The Timing Analyzer Result Flag shows Fail, which is fine for this exercise.

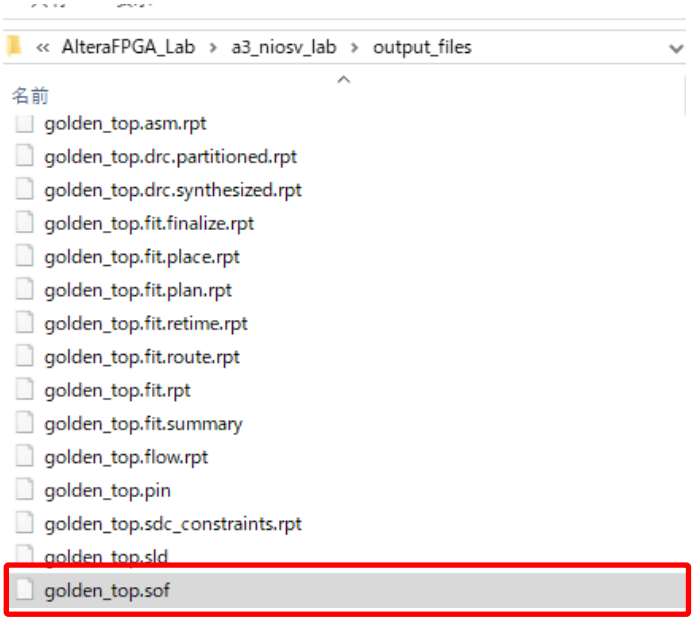

Figure 30 Checking the generated files

6. Creating the software design

This exercise uses *Ashling RiscFree IDE for Intel FPGAs**, which is available as a development environment for Nios® V processors, to write the C source code, build it, and download the program to the target board.

The C source code to be created is a program to change the LED flash interval based on signals input from the buttons on the evaluation board via the PIO.

6-1. Creating a Software Project

1. Create a workspace for Nios® V in the current hardware project directory in Quartus® Prime.
   This exercise uses the software directory for the workspace. Check that the software directory has already been generated.
   C:¥AlteraFPGA_Lab¥a3_niosv_lab¥software
2. Create a directory for the application project and a directory for the BSP project.
   Create the application project directory **app** and the BSP project directory **bsp** in the software directory as shown in the following path:
   C:¥AlteraFPGA_Lab¥a3_niosv_lab¥software¥app
   C:¥AlteraFPGA_Lab¥a3_niosv_lab¥software¥bsp

6-1-1. Create the BSP project

Start ***Platform Designer***, and select the ***BSP Editor*** tab from the Open System window that appears first. If ***Platform Designer*** is already running, select ***File*** menu => ***New BSP***. Then enter the **Open system** items in the following order.

1. ***BSP setting file:*** Click the New button on the right side of the window, and create setting.bsp directly under the ***bsp*** directory from the file selection window as shown in Figure 32.

> **Note:**
> Immediately after starting Platform Designer, the System tab is selected. Click the BSP Editor tab to switch it.

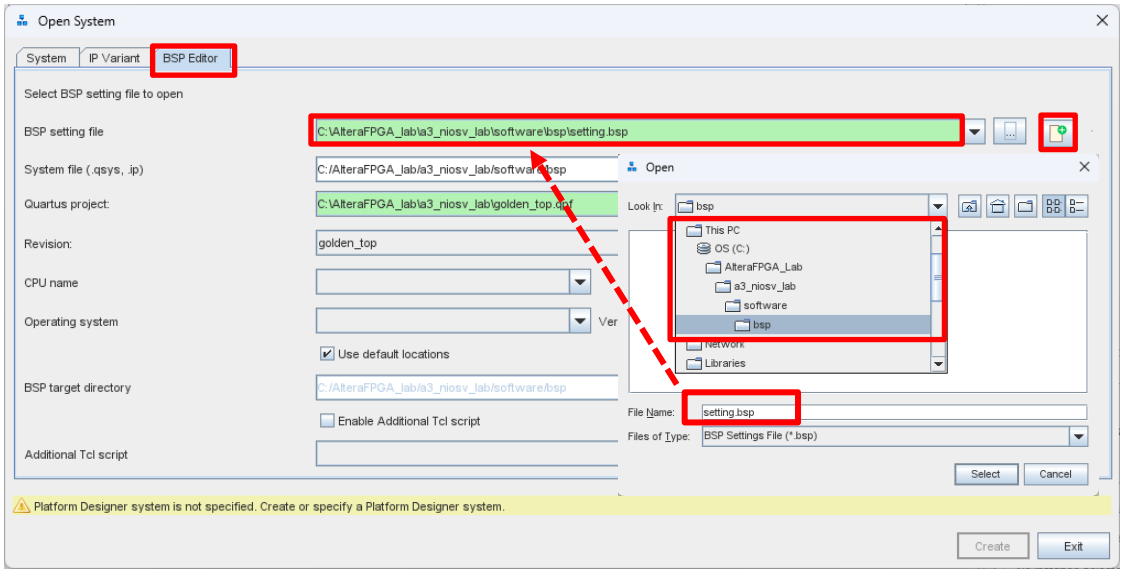

Figure 32 Generating the settings.bsp file

2. System file (qsys or sopcinfo): Select nios_system.qsys created in [4 Creating a Platform Designer System] using the file selection button on the right.

3. ***Operating system :*** Select ***Altera HAL***. (It will be selected automatically.)

4. Click ***Create*** to start ***BSP Editor***.

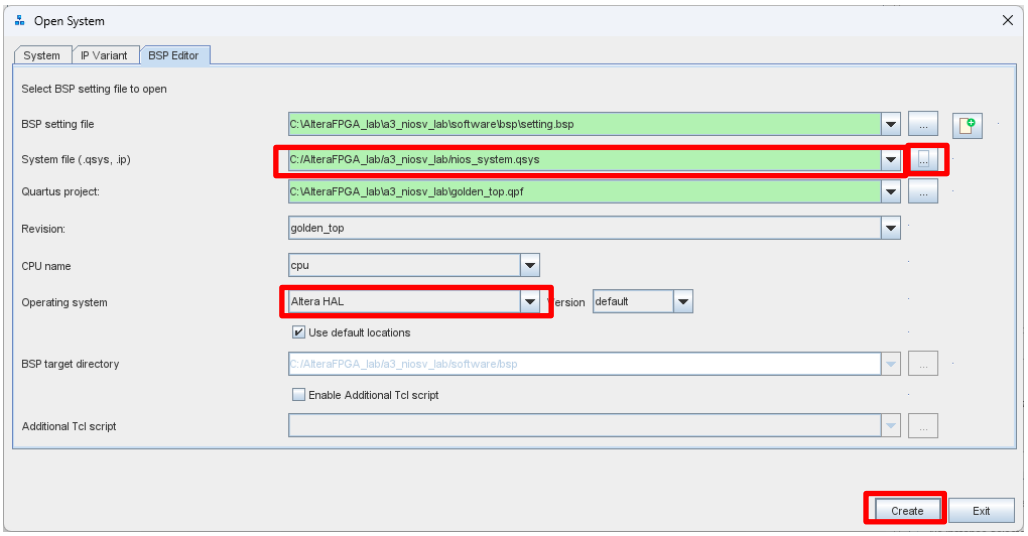

Figure 33 Open System input completion screen

5. The GUI of ***BSP Editor*** is displayed. If you start BSP Editor in a state where it is difficult to see as shown in [Figure 1], double-click the ***BSP Editor*** tab to make BSP Editor easier to see.

> ⓘ **Note:**
> Each Platform Designer window can be maximized by double-clicking the tab, and restored to its original size by double-clicking it again. This also applies to tabs other than the BSP Editor.

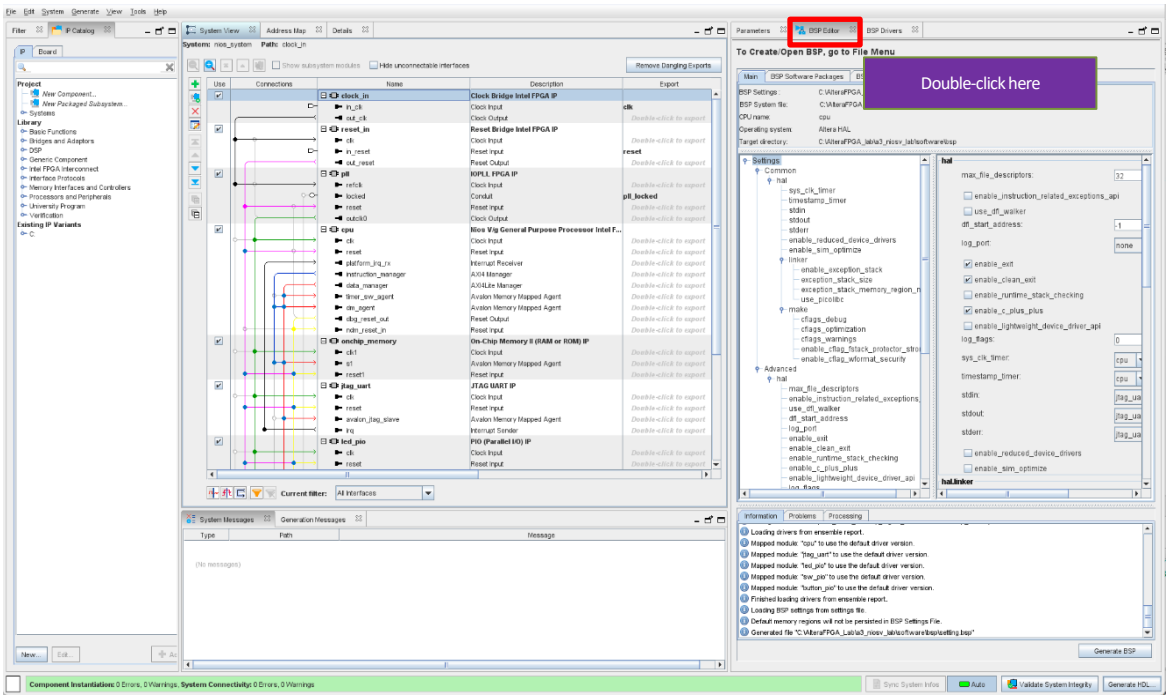

Figure 34 Starting BSP Editor

6. In this exercise, no editing will be performed. Click [***Generate BSP***] in the lower right corner of the GUI. After confirming that **Finished generating BSP ...** is displayed, click [***X***] to exit Platform Designer.

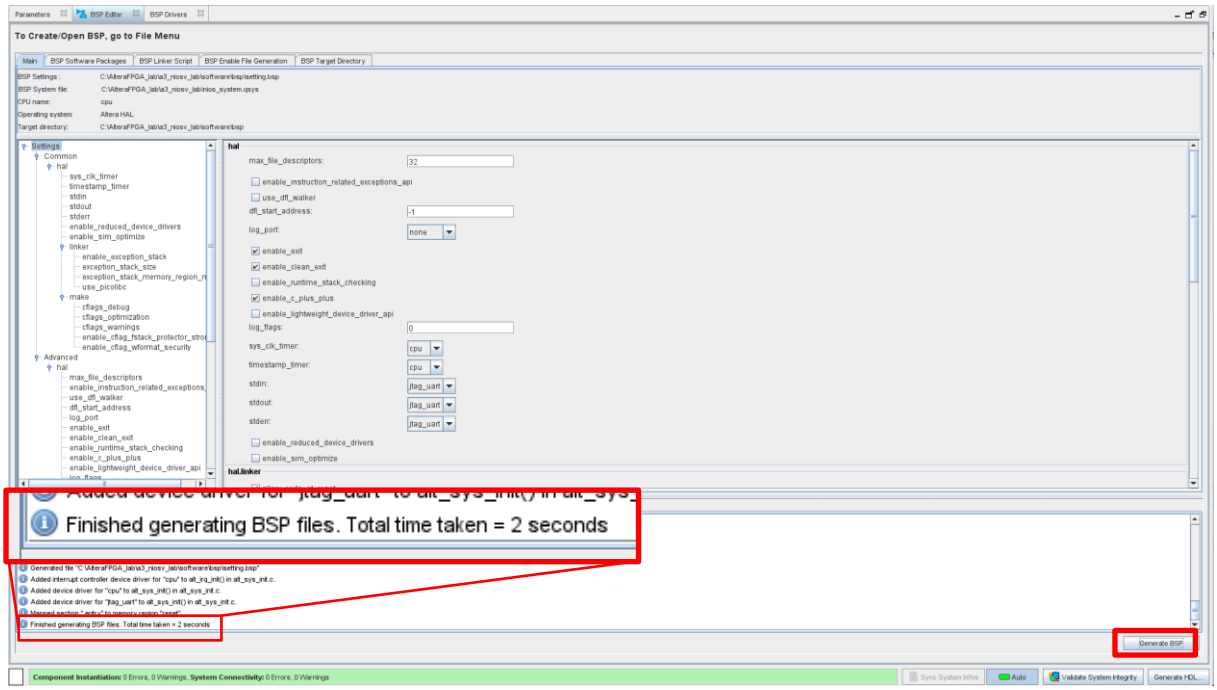

Figure 35 GUI of BSP Editor

6-1-2. Creating an Application Project

In this section you will create an application project. C For the source file, use the created ***app.c*** saved in the following directory.

C:¥AlteraFPGA_Lab¥a3_niosv_lab¥software

Follow the steps below to create an application project.

1. Copy ***app.c*** and paste it into the app directory created in [6-1. Creating a Software Project].
2. (Launch the ***Nios V command shell*** in the start menu.). Shortcut is provided on the desktop of computers lent by our company. If it is not in the start menu, browse to the following directory.
   C:¥altera_pro¥25.1¥niosv¥bin¥niosv-shell.exe
3. Specify the Quartus® Prime project directory as the current directory.

   In the ***Nios V command shell***, enter the following command:

   cd C:¥AlteraFPGA_Lab¥a3_niosv_lab↵

> ı Note
>
> You can find the commands in ***Niosv_command.txt,*** which is stored in the Quartus® Prime project directory. You can copy and use each command to simplify your work.

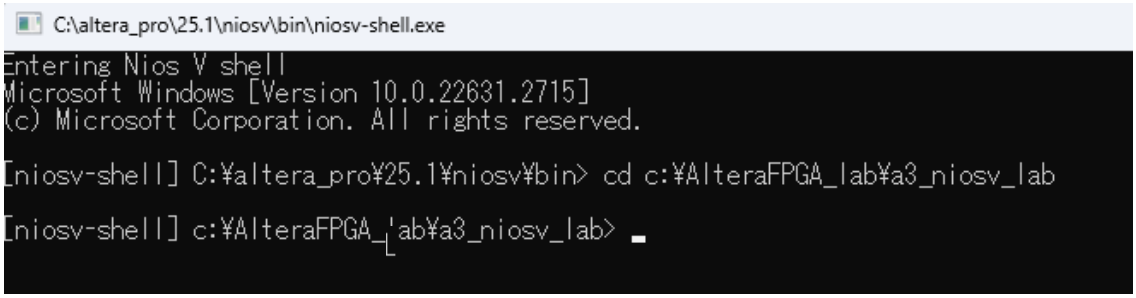

Figure 36 Specifying the Current Directory

4. Create CMakeLists.txt.

Enter the following commands in the ***Nios V Command Shell***: (You can find the commands in ***niosv_cmd.txt*** stored in the Quartus® Prime project directory. You can simplify your work by copying and using each command.)

niosv-app.exe --bsp-dir=software/bsp --app-dir=software/app --srcs=software/app/app.c --elf-name=app.elf↵

cmake.exe -G "Unix Makefiles" -DCMAKE_BUILD_TYPE=Debug -B software/app/build/Default -S software/app↵

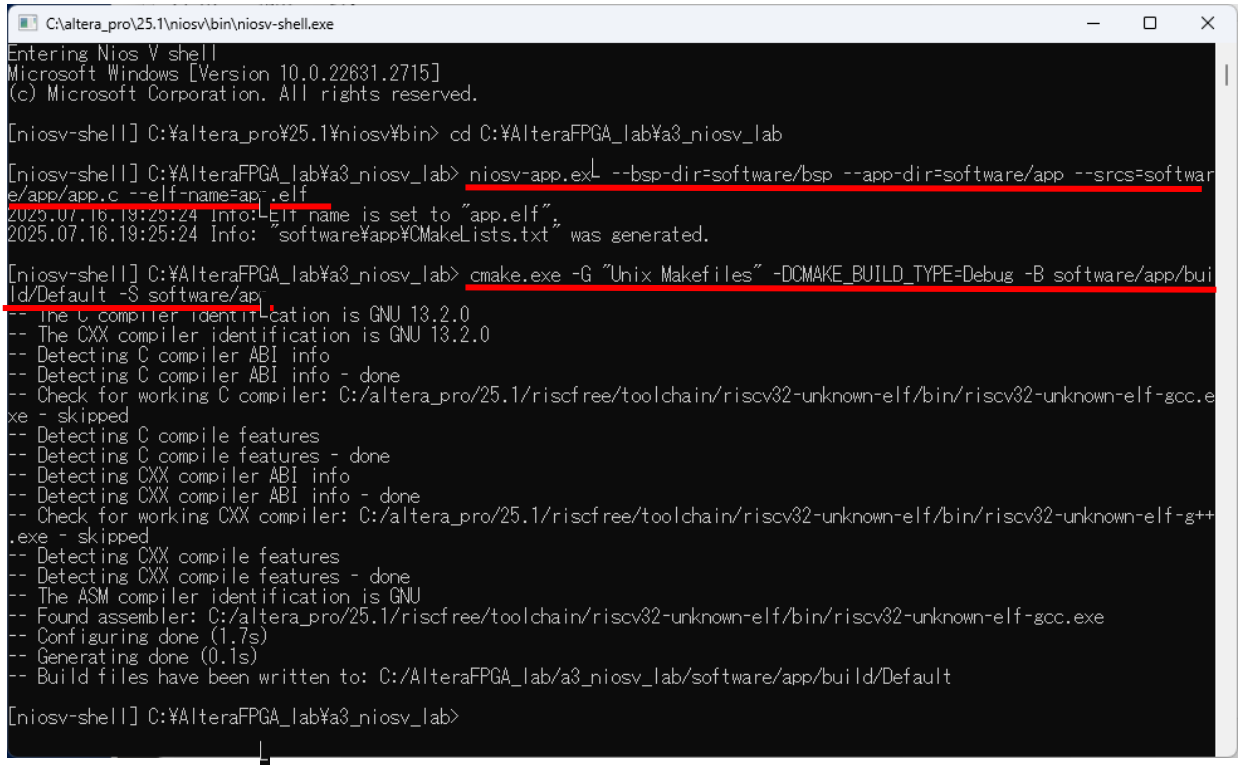

Figure 37 Command shell input screen

6-2. Starting RiscFree IDE

From the ***Nios V command shell***. Start the ***Risc Free IDE***.

1. Enter the following command into the running ***Nios V command shell*** to start the ***RiscFree IDE***.

   RiscFree.exe↵

2. The workspace selection screen appears. Use [***Browse...***] to navigate to the following workspace directory and click [***Launch***]:

   C:¥AlteraFPGA_Lab¥a3_nios5_lab¥software

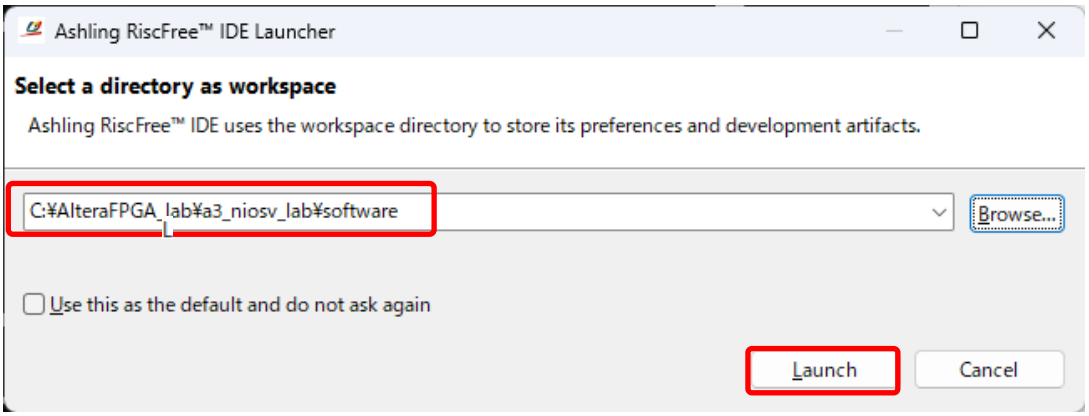

Figure 38 Workspace Selection Screen

3. If you see the Windows Security screen shown in Figure 39, click Allow.

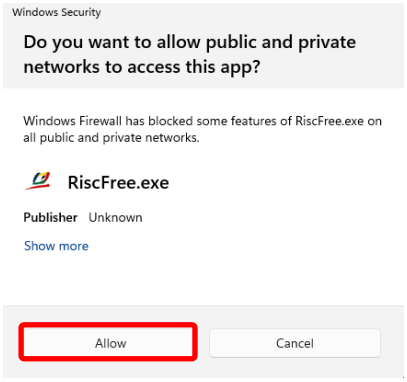

Figure 39 Windows Security Screen

4. If the pop-up shown in Figure 40 appears, check "**Exclude Ashling ...**" and click ***Proceed***.

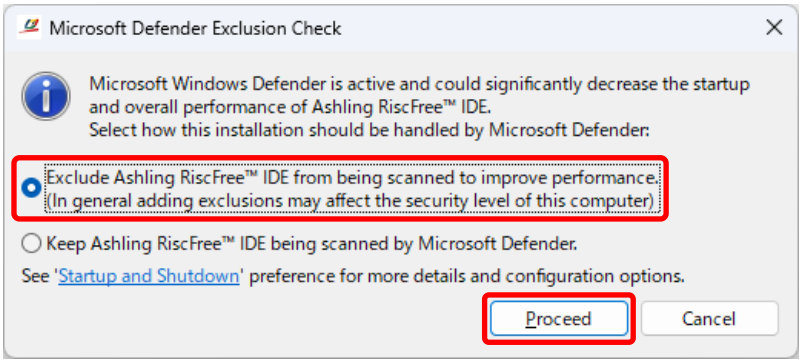

Figure 40 Microsoft Defender Exclusion Check screen

5. The ***RiscFree IDE*** GUI launches.

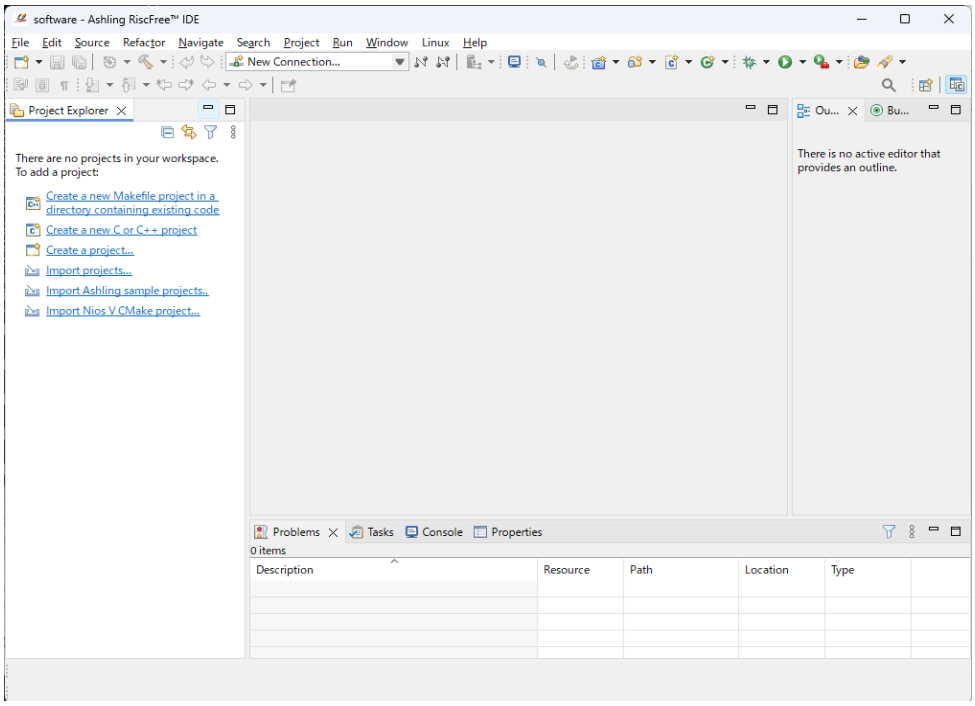

Figure 41 Launching the RiscFree IDE

6-3. Importing a Project

Import the app project you created into the ***RiscFree IDE***.

1. Select ***File ⇒ Import Nios V CMake project...*** from the RiscFree IDE menu. Alternatively, click ***Nios V CMake Project...*** in the Project Explorer of the GUI screen.

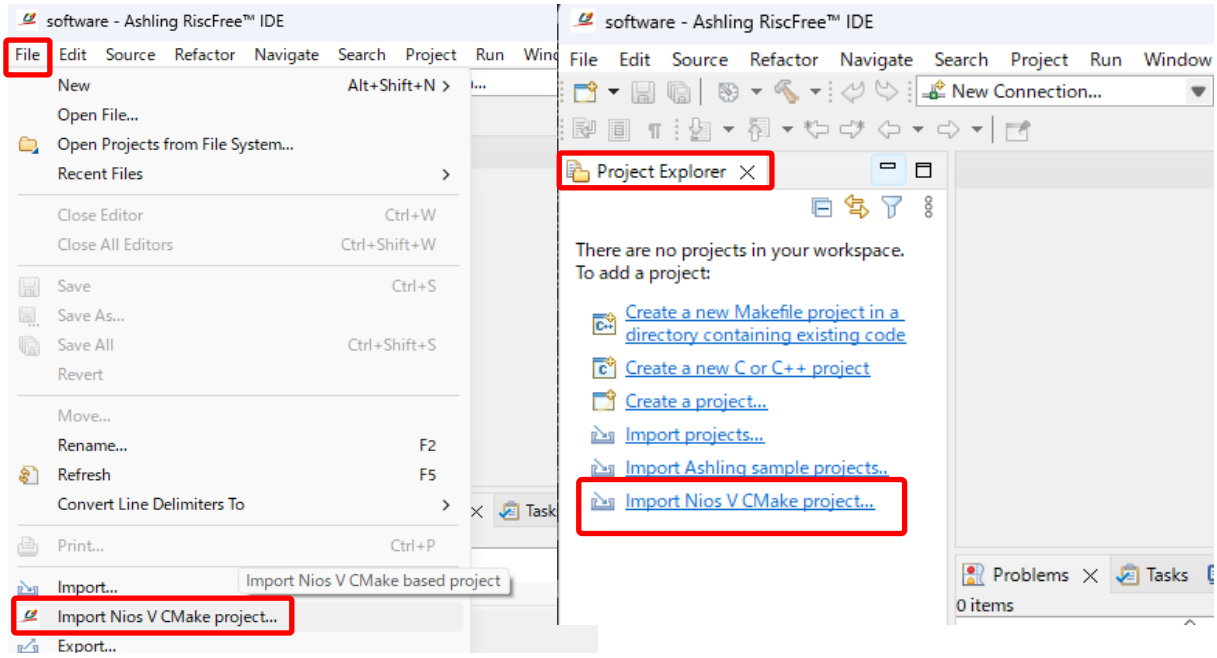

Figure 42 Importing a Project

2. In the ***Import*** screen, set the ***Project name*** and the ***Nios V CMake project location***. The ***project location*** can be selected in the GUI by clicking [***Browse***].

   After setting, click [***Finish***].

   Project Name : app

   Nios V CMake project location : **C:¥AlteraFPGA_Lab¥a3_niosv_ lab¥software¥app**

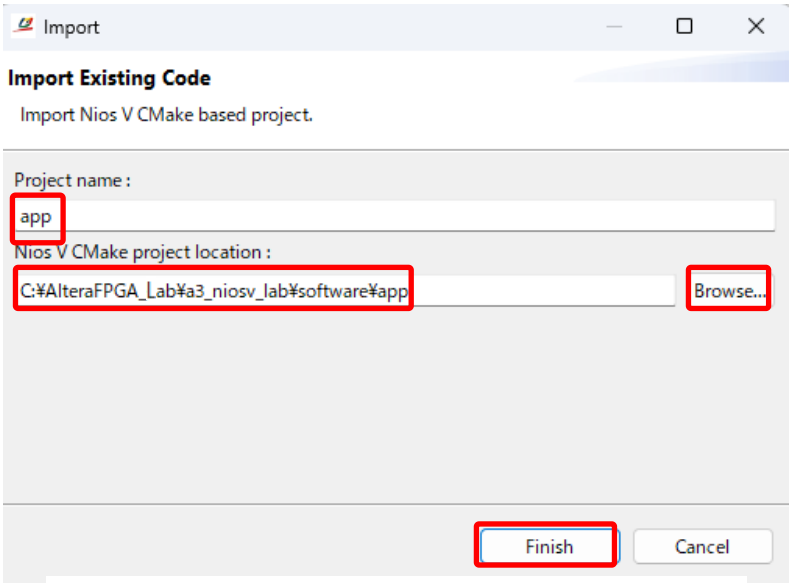

Figure 43 Import Settings

6-4. Build the project

After importing the project, perform Build to generate an executable file (elf file).

Right-click the imported app project and select ***Build Project*** from the displayed menu. After completing the ***Build Project***, an elf file (***app.elf)*** is generated as shown in Figure 45.

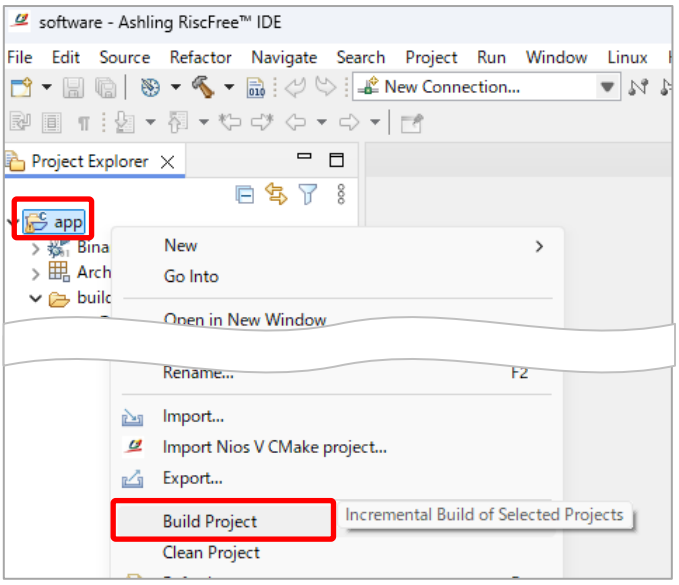

Figure 44 Steps for Build Project

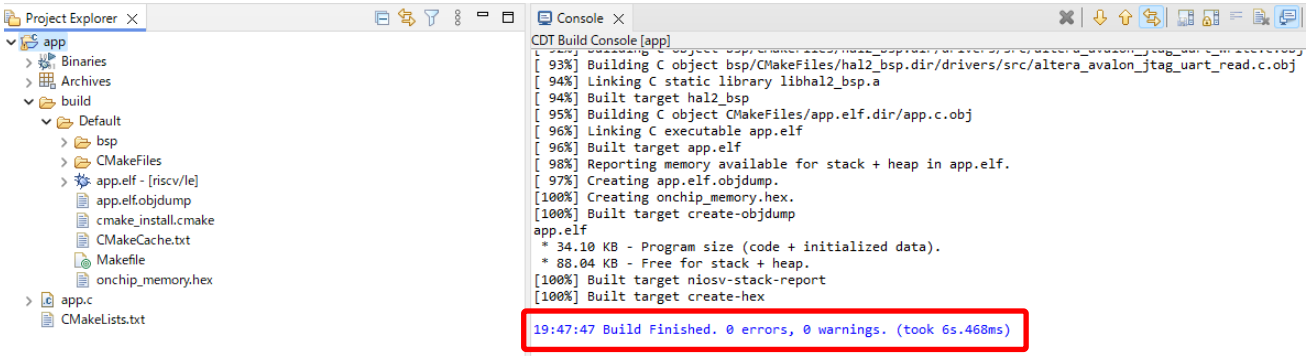

Figure 45 Reviewing the Generated elf

7. Executing Software Design

Now that the C source code has been created and built, the program (executable) generated in the previous steps is executed on the target board.

To execute the program, first connect to the serial terminal where the FPGA configuration messages are output, and then download the program to the target evaluation board.

7-1. FPGA Configuration

After compiling in [5. Creating the Hardware Design], a Software Object File (SOF file) will be generated in the following directory:

C:¥AlteraFPGA_Lab¥a3_nios5_lab¥output_files¥golden_top.sof

Using the download cable, program the sof file into the FPGA (SRAM) via the JTAG interface.

- Connect the USB Type-C cable to the Type-C connector on the Atum a3-nano board. Then connect the other end of the USB cable to the USB port on your computer to connect the board to your computer.
- Connect the AC adapter to the power connector (J5) on the Atum a3-nano board.

Follow the steps below: From the ***Quartus® Prime*** Tools menu, select ***Programmer*** to launch the Programmer.

1. Click the ***Hardware Setup*** button.

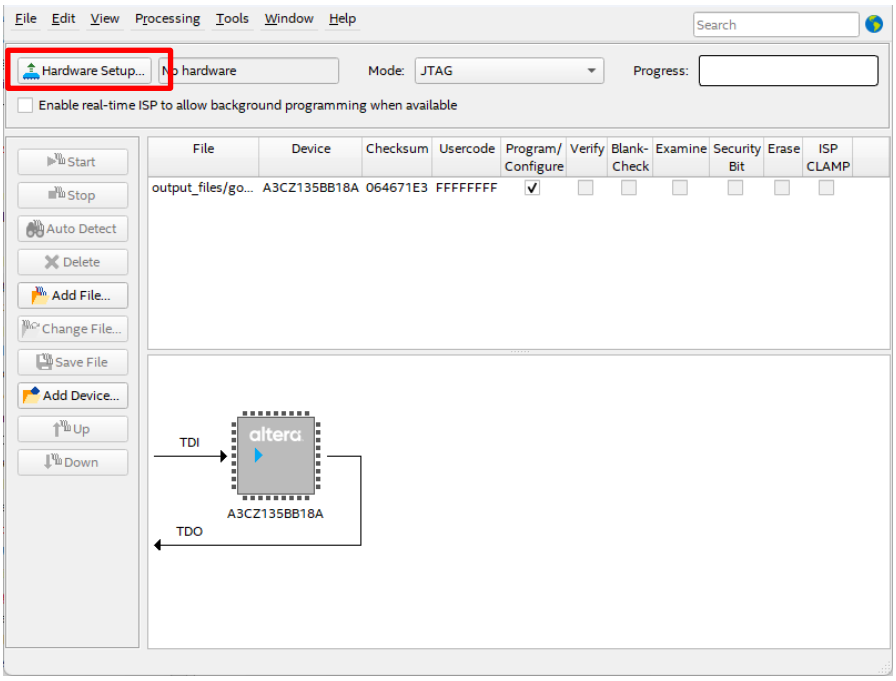

Figure 46 Hardware Setup (Programmer)

2. Under ***Currently selected hardware***, select ***Atum A3 Nano [USB-1]*** from the list box.

   After selecting, click the ***Close*** button.

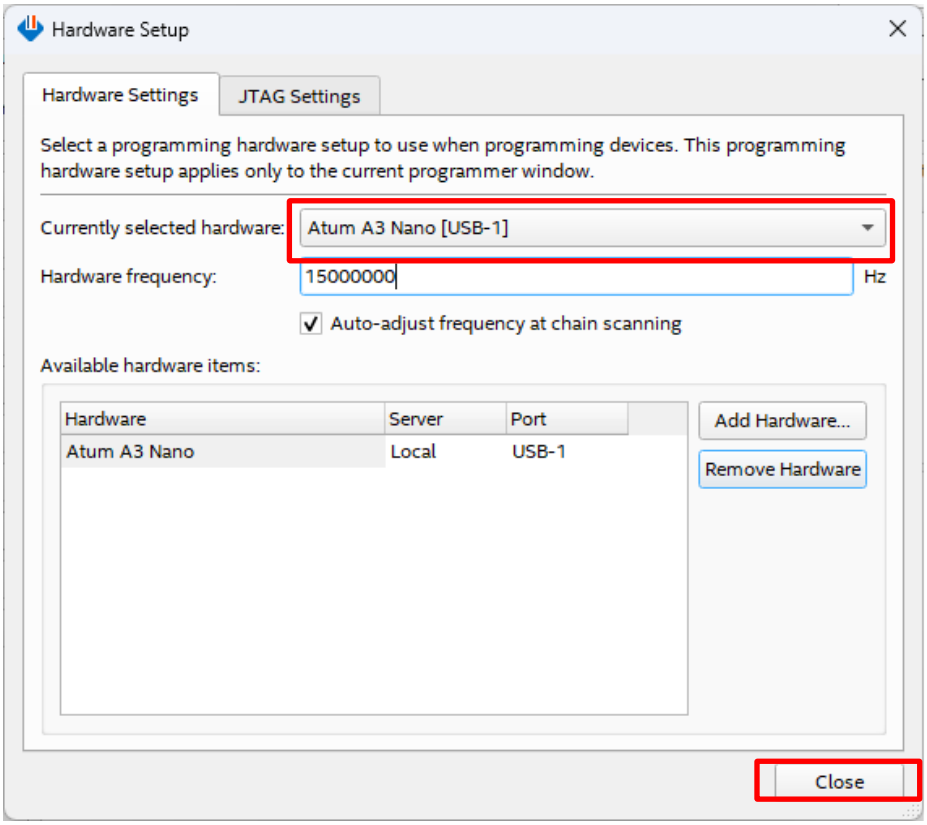

Figure 47 Selecting Hardware

If there are no ***available hardware items***, check the following:

1) Are the cables connected correctly?

2) Is the board powered?

3. Click the ***Auto Detect*** button to search for devices on the board.

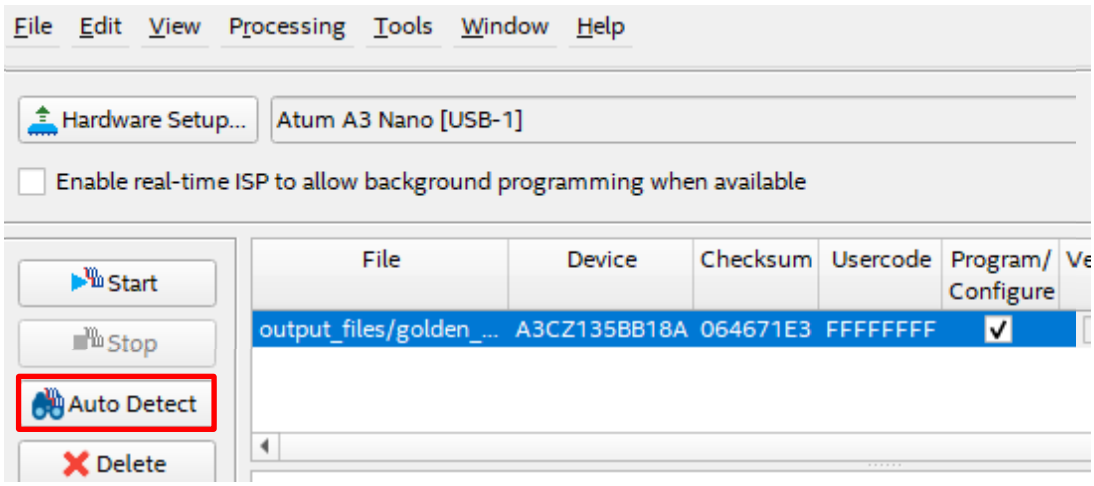

Figure 48 Auto Detect

If the pop-up Figure 49 appears, click ***Yes***.

Figure 49 List in Programmer Pop-up to Confirm Update

4. If you try clicking the ***Auto Detect*** button several times and it does not look like Figure 50, please consult your instructor.

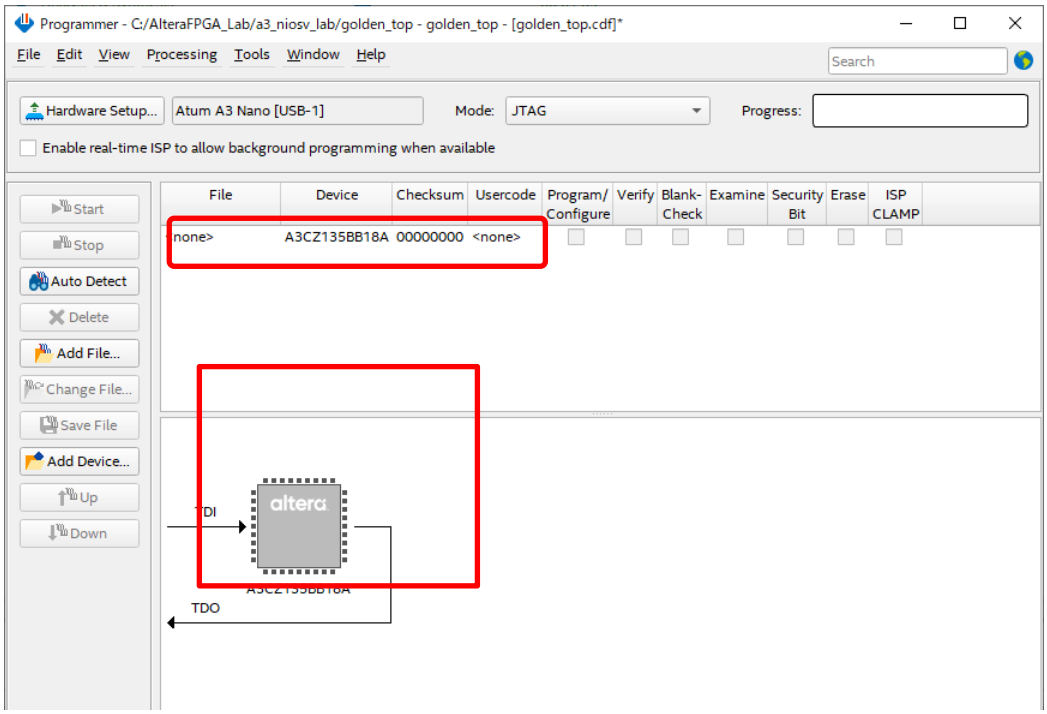

Figure 50 Display after Auto Detect

5. Select the file to write to. After highlighting the device, click ***Change File....*** When the Select New Programming File dialog box appears, select the sof file below and click ***Open***.
C:¥AlteraFPGA_Lab¥a3_niosv_lab¥output_files¥golden_top.sof

6. After selecting the sof file, check the ***Program/Configure*** field on the same line.

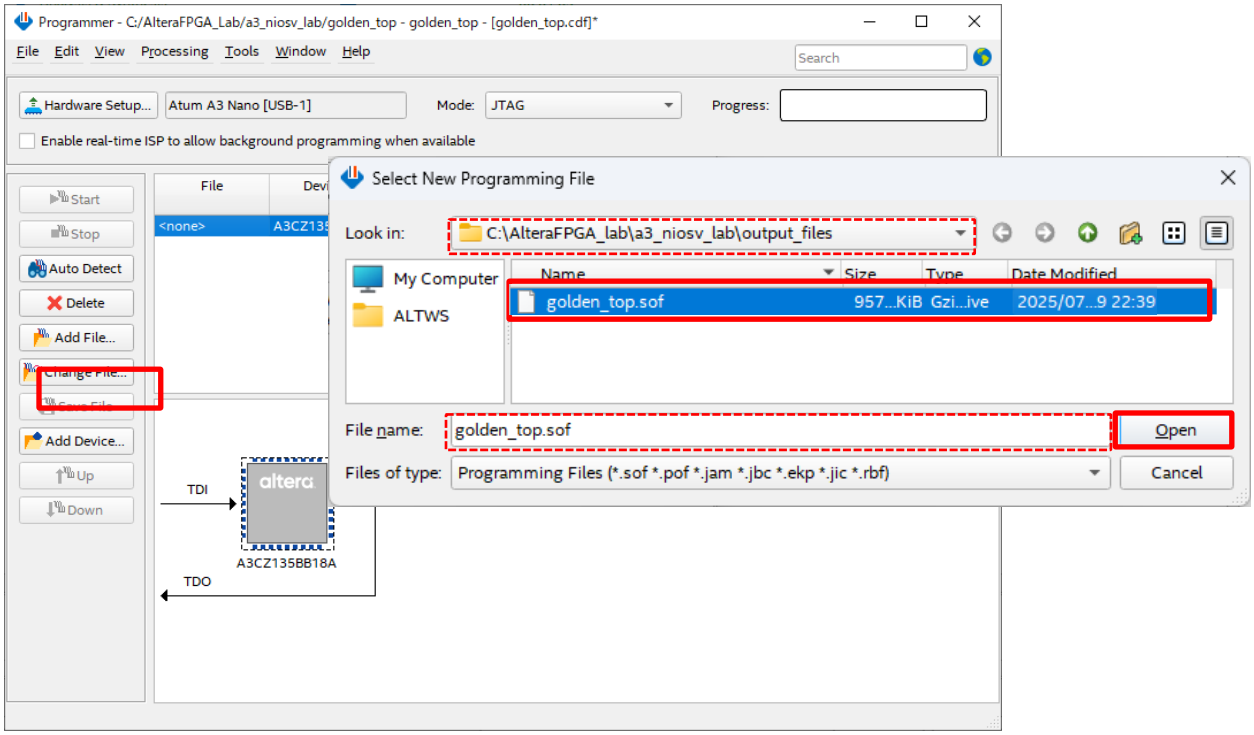

Figure 51 Select sof file

After checking, click the ***Start*** button to write the hardware design to the FPGA.

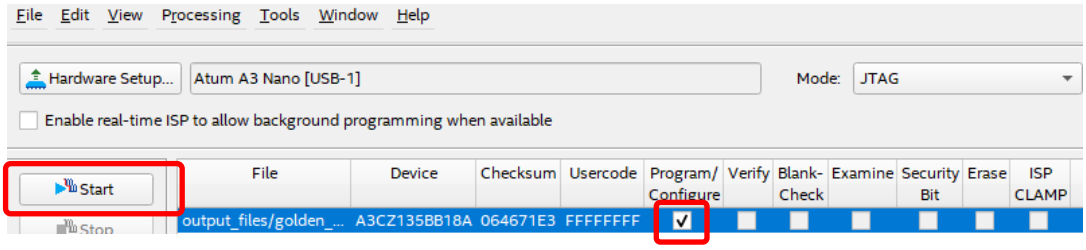

Figure 52 Selecting Options and Starting Writing

7. Writing is complete when the ***Progress*** bar is **100% (Successful).**

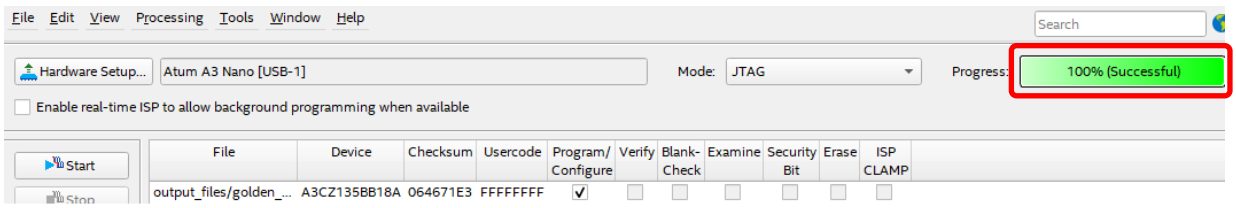

Figure 53 Writing Complete

7-2. Register Juart-terminal

The ***RiscFree IDE*** does not have a JTAG console. Therefore, you use the JTAG console by registering an external Tool.

Follow the steps below to register.

1. ***Run*** ⇒ ***External Tools*** ⇒ ***External Tools Configurations*** Launch the GUI from ….

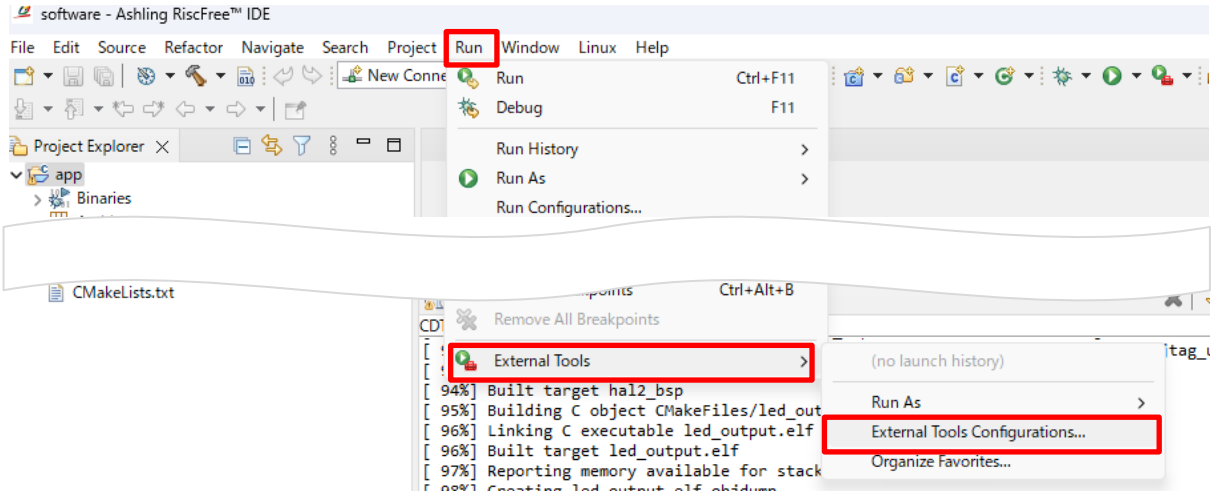

Figure 54 External Tools Configurations

2. Select ***Program*** on the left side of the External Tools Configurations screen, highlight it, and then click the ***New launch configurations***  button in the upper left.

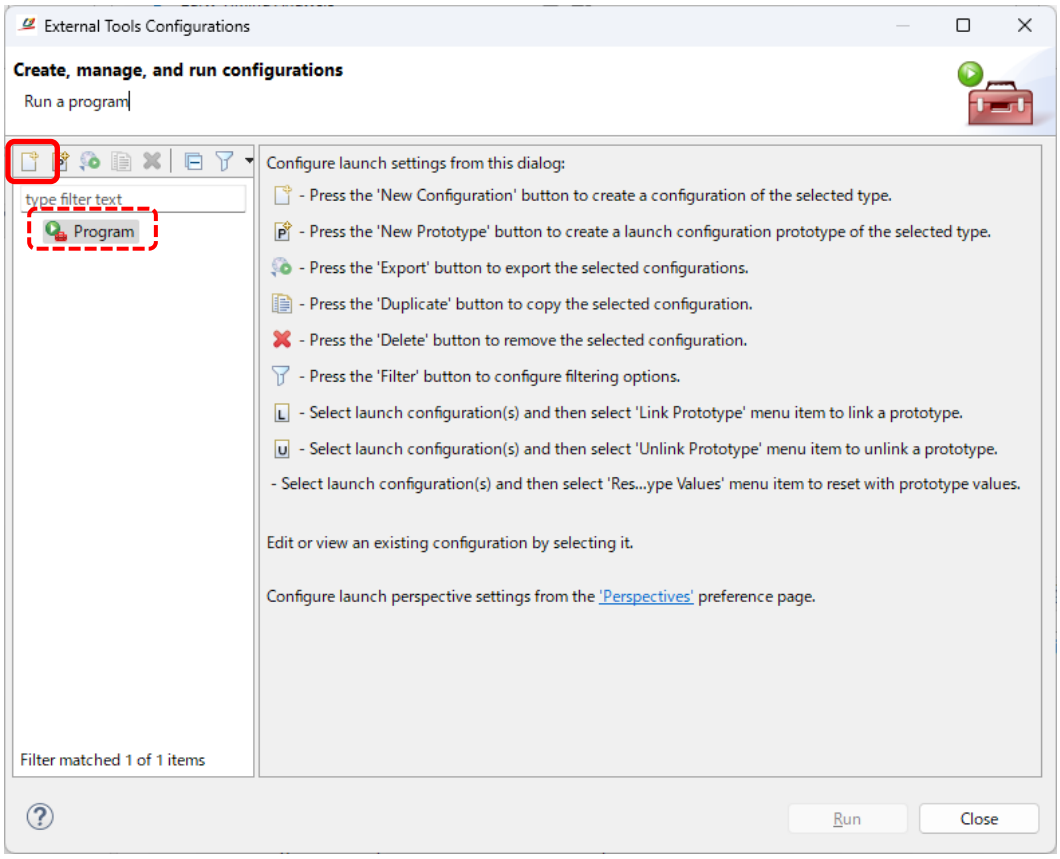

Figure 55 External Tools Configurations GUI

3. Enter the following values in the displayed GUI items.

   Name : **juart-terminal**

   Location : Absolute path to juart-terminal.exe (using the ***Browse File System*** button)

   Example : ***C:¥altera_pro¥25.1¥quartus¥bin64¥juart-terminal.exe***

4. In the ***Working Directory***, click the ***Browse Workspace...*** button. In the ***Folder Selection*** screen that appears, select ***app*** and click the ***OK*** button.

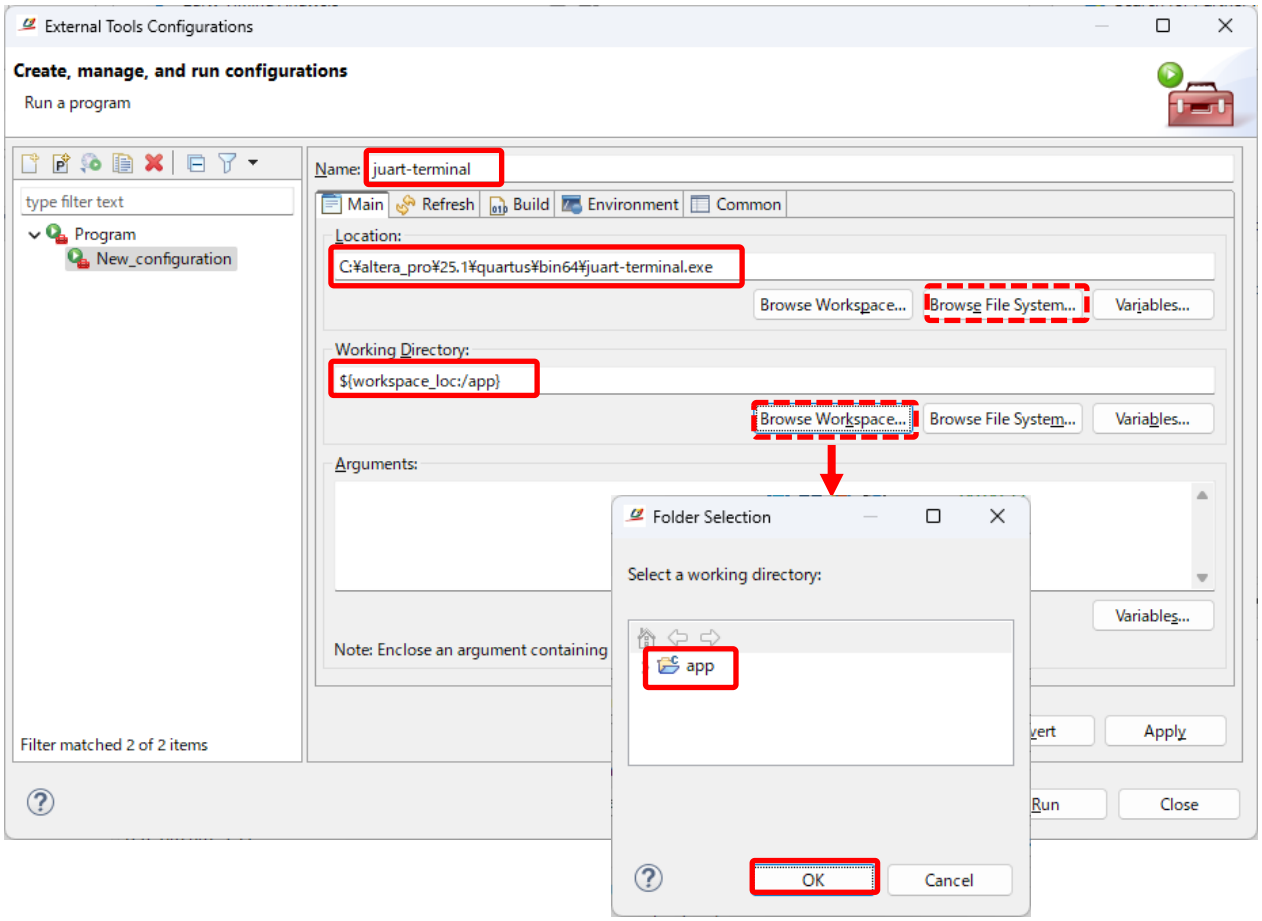

Figure 56 juart-terminal registration

5. Click the [***Apply***] button at the bottom right of External Tools Configurations to complete registration. [***Run***] Click to execute.

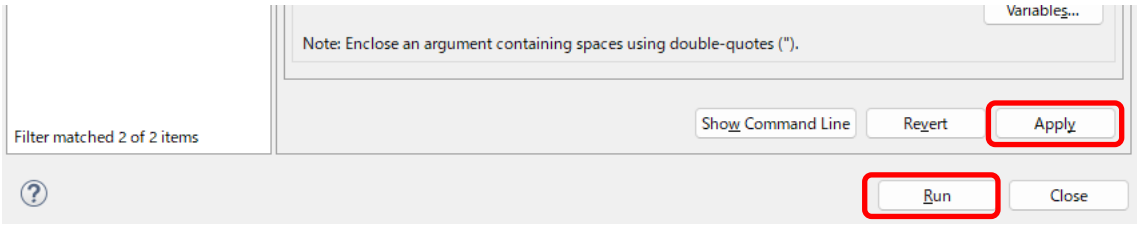

Figure 57 Apply Button/Run Button

7-3. Launch Run configuration

Download and run the elf file on the target board.

1. After right-clicking the app project ***Run As*** ⇒ ***Run configurations…*** Select.

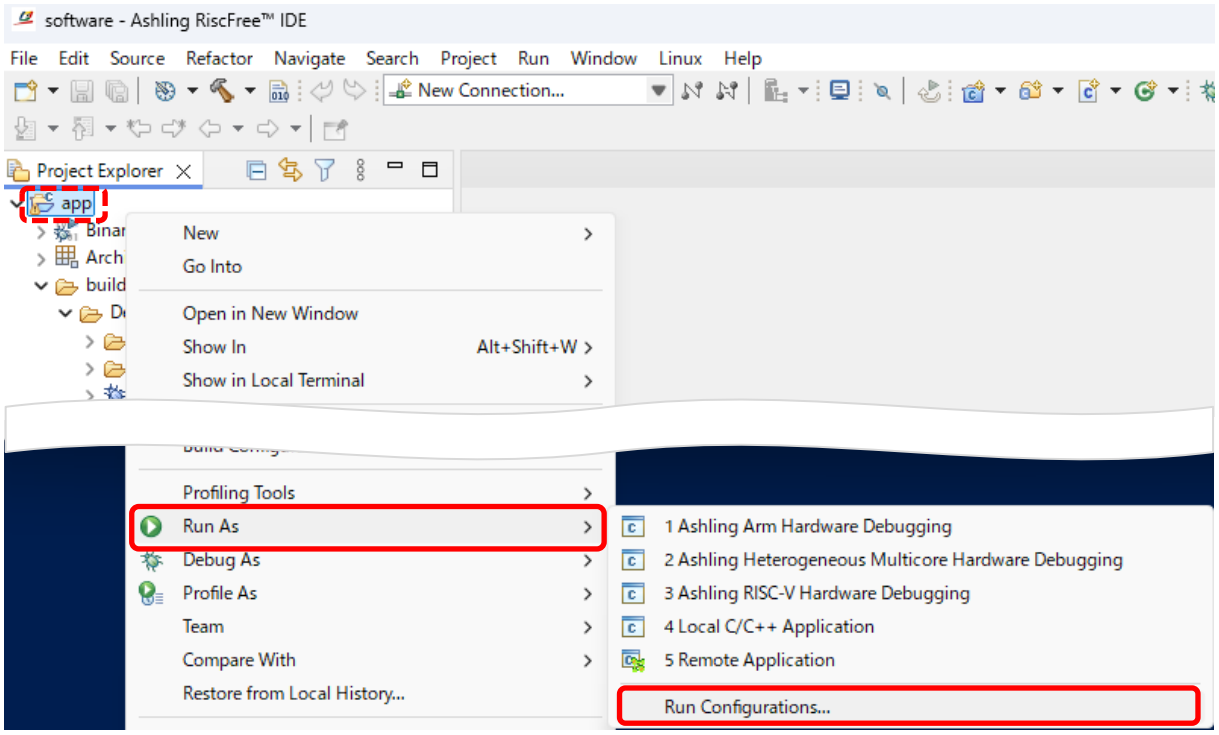

Figure 58 Run configurations

2. Double-click ***Ashling RISC-V Hardware Debugging***.

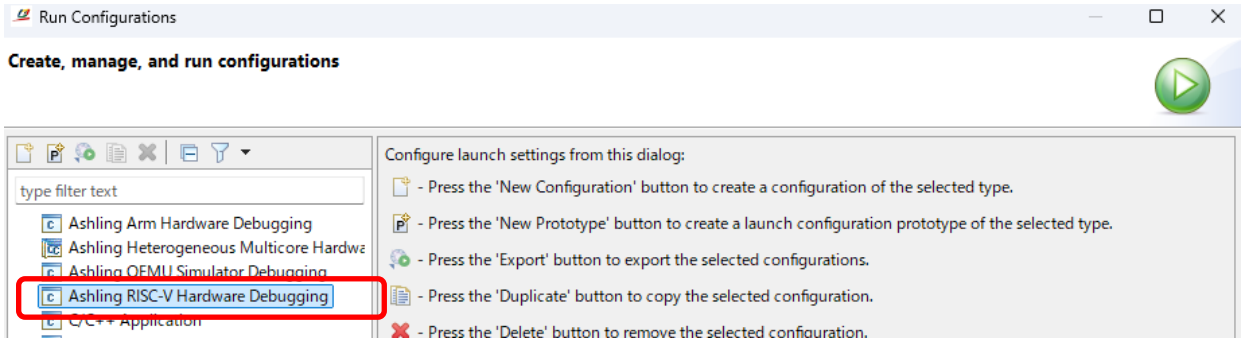

Figure 59 Ashling RISC-V Hardware Debugging (in Run configurations)

3. Click ***Search Project under Main Tab- > C/C++ Application:***

4. On the ***Program Selection*** screen, select ***app.elf*** and click the ***OK*** button.

5. In the ***Main*** tab, enter the following:
   Name : Enter any name. (***app Default*** in this exercise)
   Project : **app** is auto-populated.
   C/C++ Application : **build/Default/app.elf** is auto-populated.

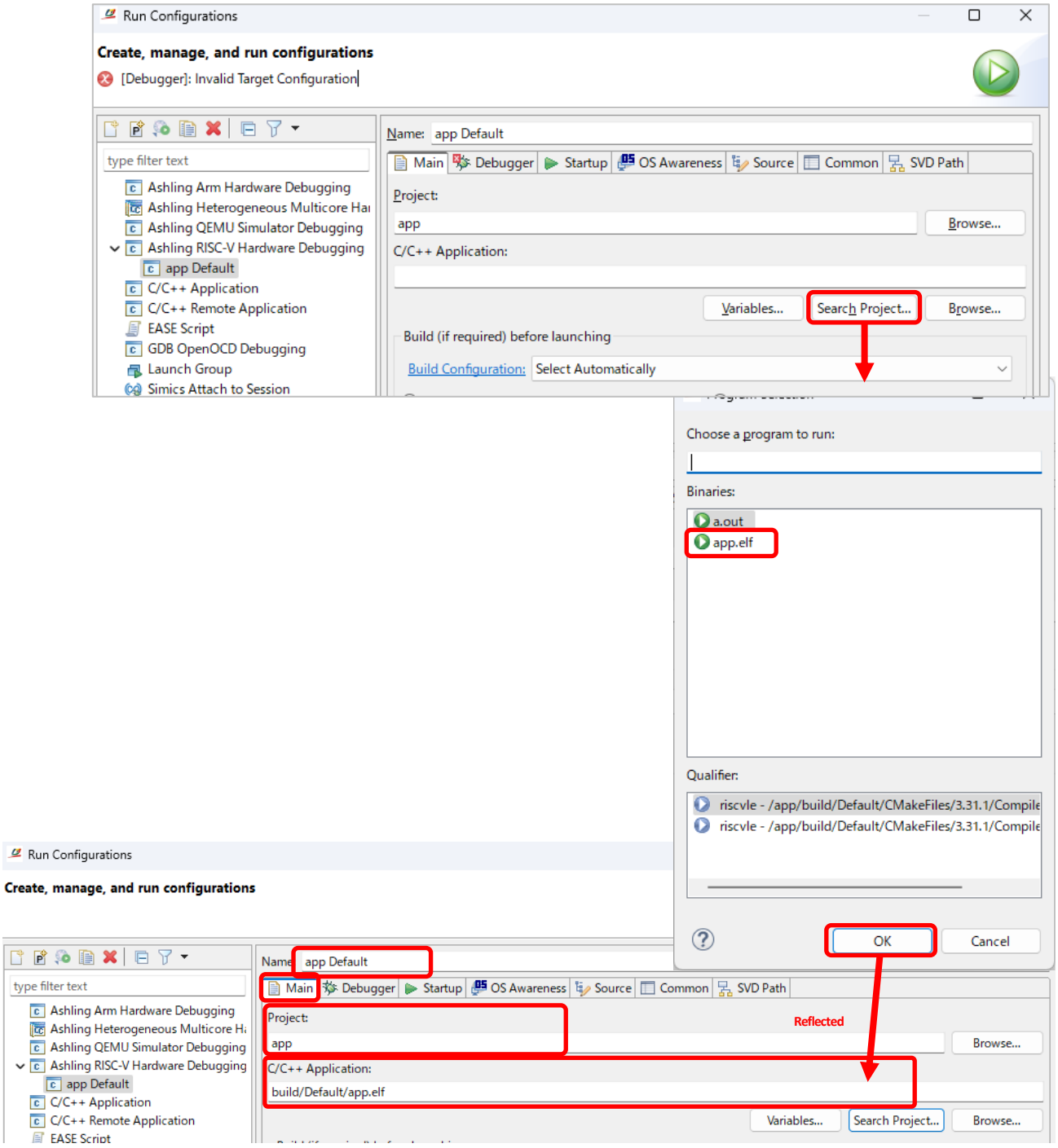

Figure 60 Run configurations configuration steps

6. Go to the ***Debugger*** tab.
7. Click ***Auto-detect Scan Chain*** and make sure Device/TAP select is ***A3C (Y135BB18A|Z135BB18A)*** and Core selection is ***Nios V (0x08986E00).***
8. Click the ***Apply*** button.

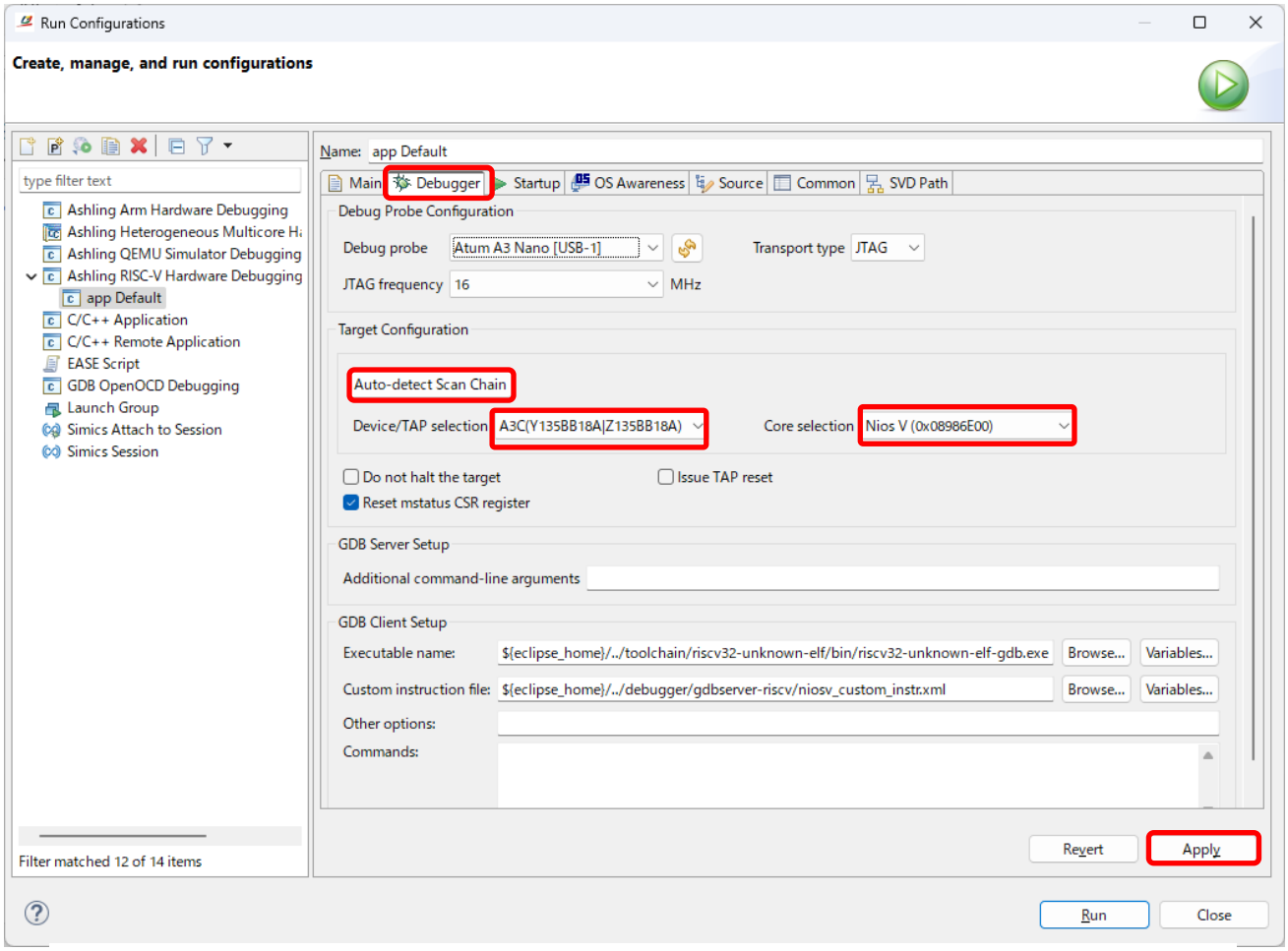

Figure 61 Scan Chain Settings

7-4. Run AS

1. Click the ***Run*** button at the bottom right of the ***Run configuration*** GUI. If the Windows Security screen appears, click **Allow**.

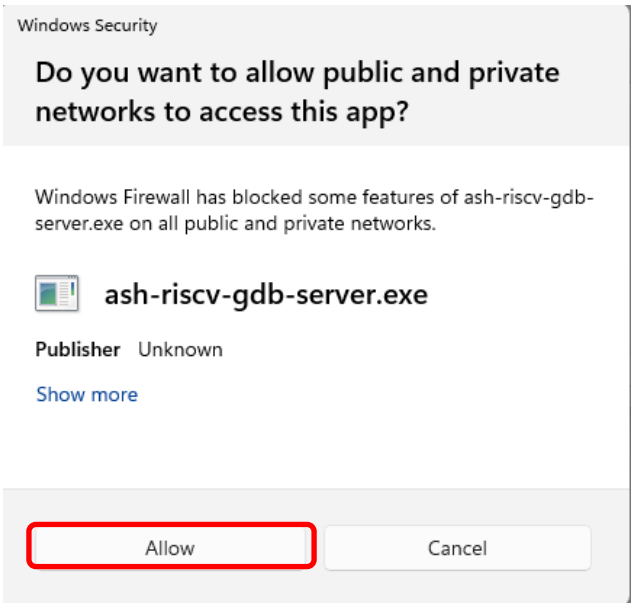

Figure 62 Windows Security Screen

2. The RiscFree IDE ***Console*** displays "**Hello from Nios V, start!!**" when the program is executed .

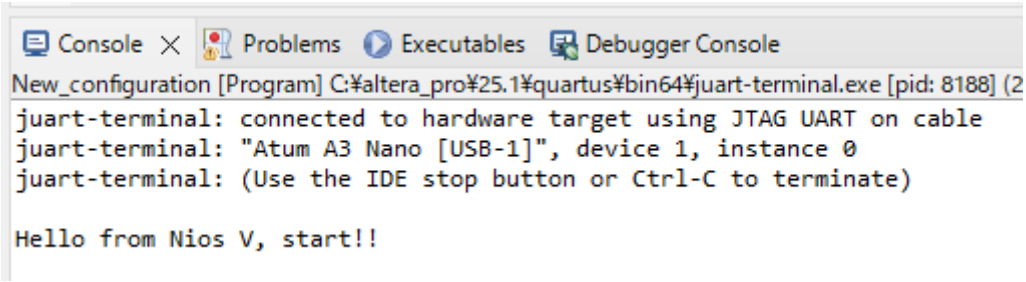

Figure 63 Console Output

3. You can see that the LEDs on the board are flashing as shown in [Figure 64]. Switch (SW13-1) ON/OFF to change the LED flashing speed.

Figure 64 LED Flashing and Speed Change by DIP Switch

4. After checking the operation, click the button in the Console of the RiscFree IDE to exit the program.

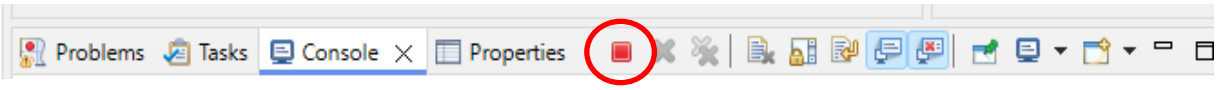

Figure 65 Stopping the Program

> **Note:**
>
> After pressing the stop button on the console, there may be an active console behind the first console. Press the button next to the stop button. Make sure the button is grayed out.

7-5. Starting the Debug Configuration

Configure the target board for debugging.

1. Right-click the App project and select ***Debug As => Debug configuration***.

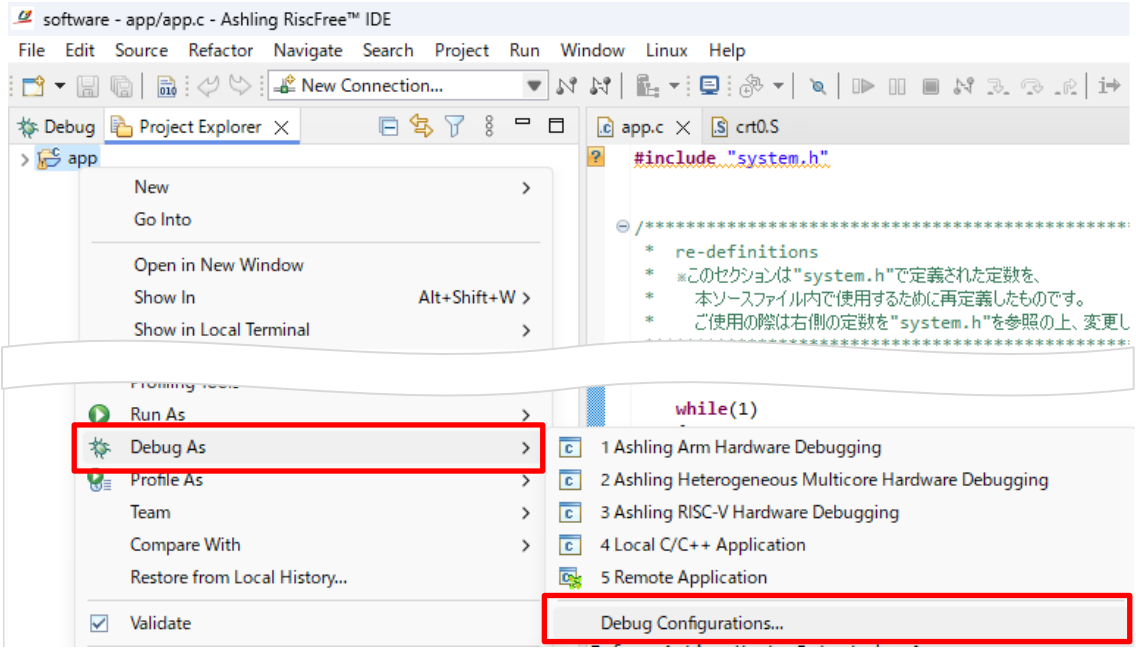

Figure 66 Starting the Debug Configuration

2. Select ***app Default under Ashling RISC-V Hardware Debugging*** created in Run Configuration.

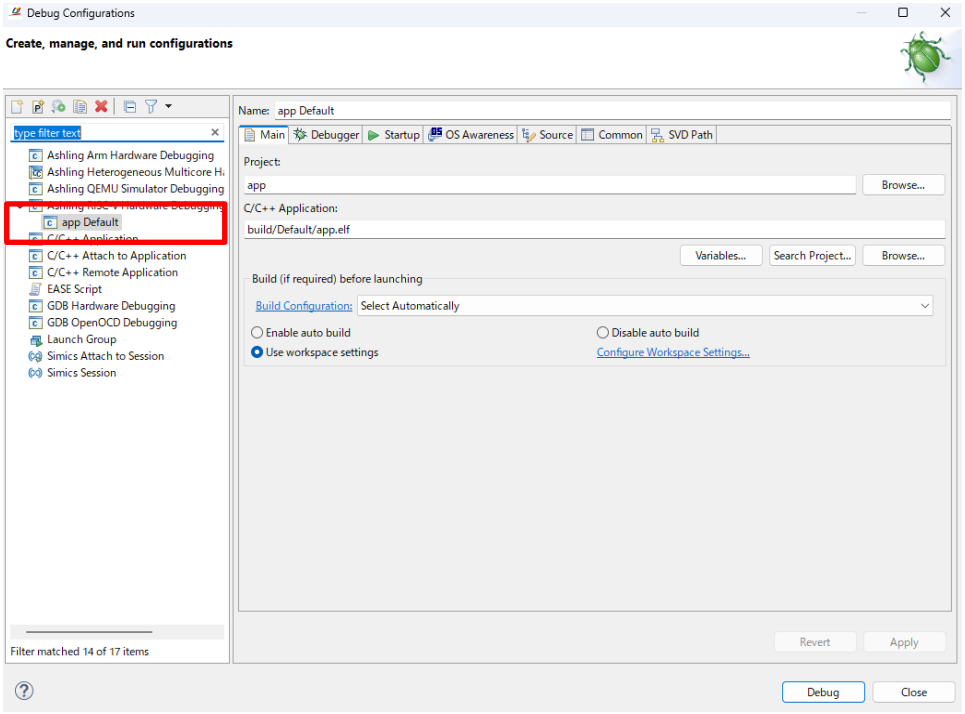

Figure 67 Debug Configuration Settings

3. Select the ***Debugger*** tab and from the ***Target Configuration***, make sure that ***Device/TAP select*** is ***A3C (Y135BB18A|Z135BB18A)*** and ***Core selection*** is ***Nios V (0x08986E00).*** If not, click ***Auto-detect Scan Chain*** and make sure that Core is selected again.

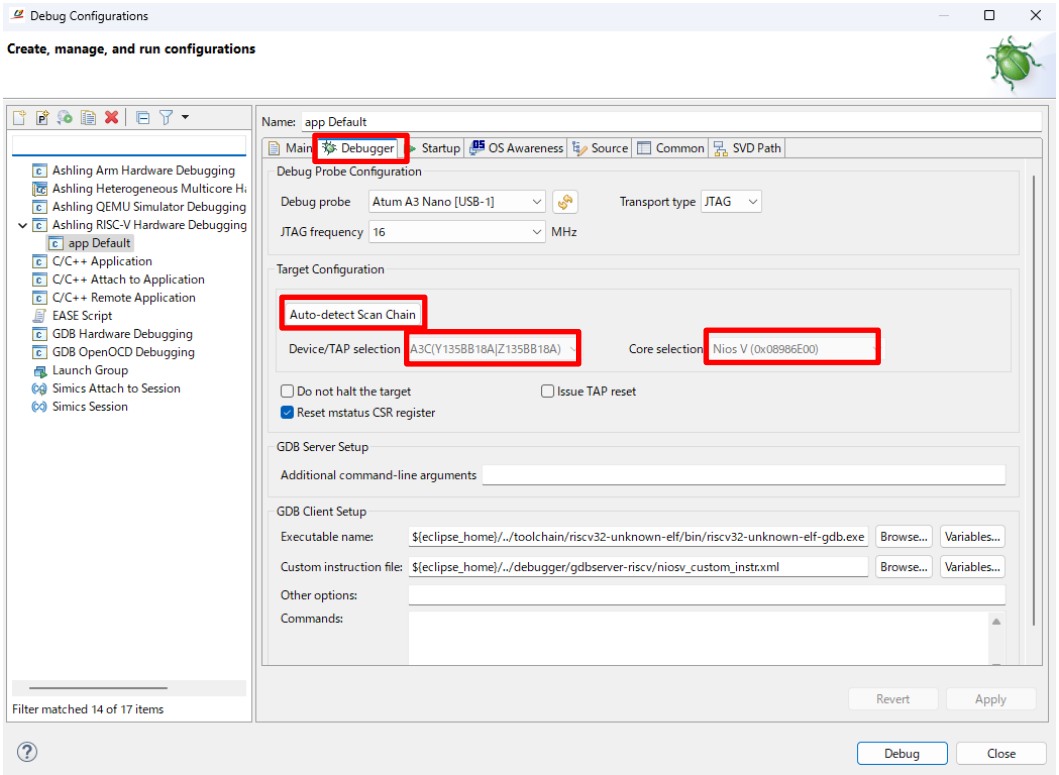

Figure 68 Scan Chain Settings

7-6. Executing Debug As

1. Click the ***Debug*** button at the bottom right of the Debug Configuration GUI.
2. A pop-up will appear asking you to confirm moving the Perspective, select the ***Switch*** button.

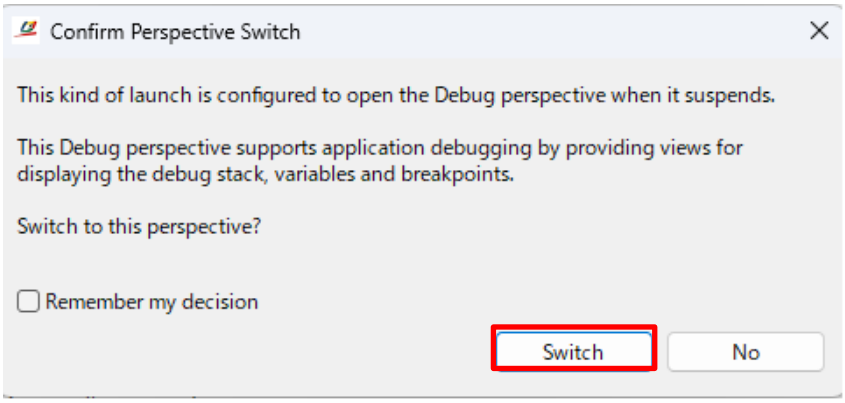

Figure 69 Changing Perspective

3. In the Debug Perspective, you can see in the center of the screen that a break occurs at the beginning of the main function.

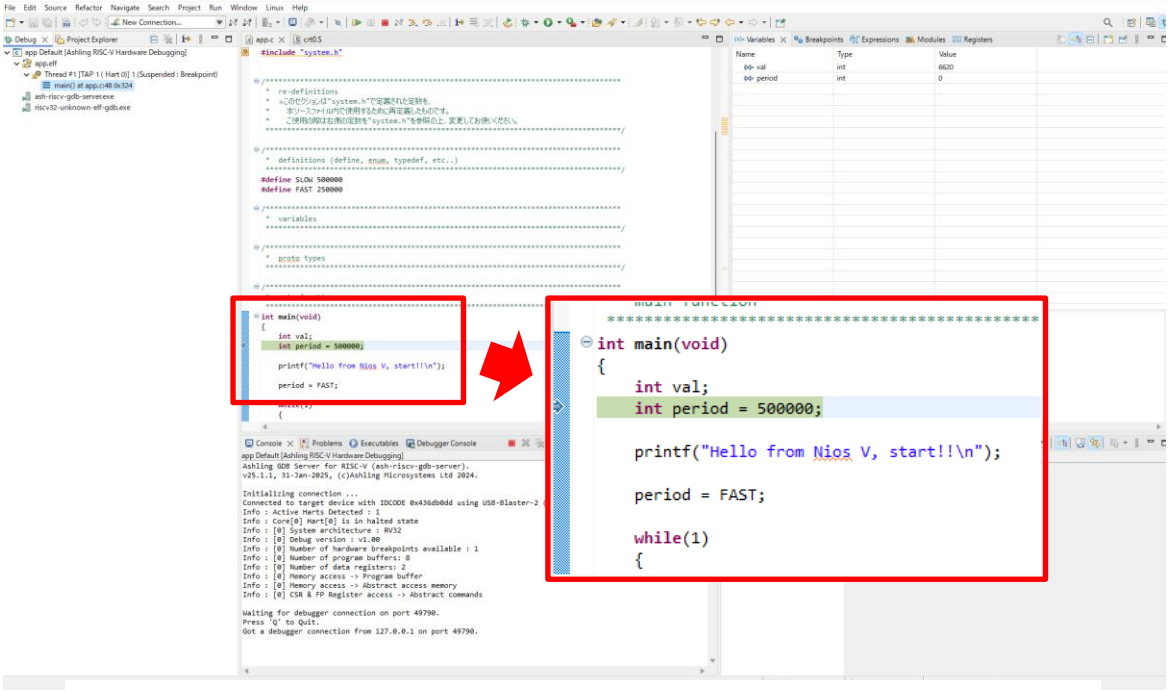

Figure 70 Starting Debug Perspective

7-7. Various Debug Functions

The following describes the various functions available in the Debug Perspective.

7-7-1. Using the Debug Toolbar

The RiscFree IDE's Debug toolbar provides Eclipse-based functionality that enables you to debug programs efficiently.

The buttons have the following functions:

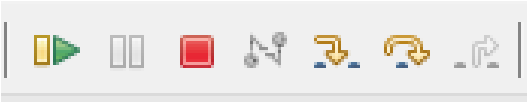

Figure 71 Debug Tool Bar Specifications

[ ] Resume: Resumes at the next break point or until the process terminates.

[ ] Suspend: Pauses the debugging code.

[ ]Terminate: Terminates the debugging code.

[ ]Step into: Steps into the next state statement of the executing code to see further processing. [*] [*]

[ ]Step over: Steps over to the next state statement of the executing code and advances to the next line.

[ ]Step out: Returns from the state statement stepped into.

7-7-2. Using Break Points

When debugging a program, you can use break points as markers to pause execution at a particular line. To the left of the code in the source view, there is a blue highlighted area that you can double-click at any line you want to stop.

You can see that code execution stops at the line where you set the break point.

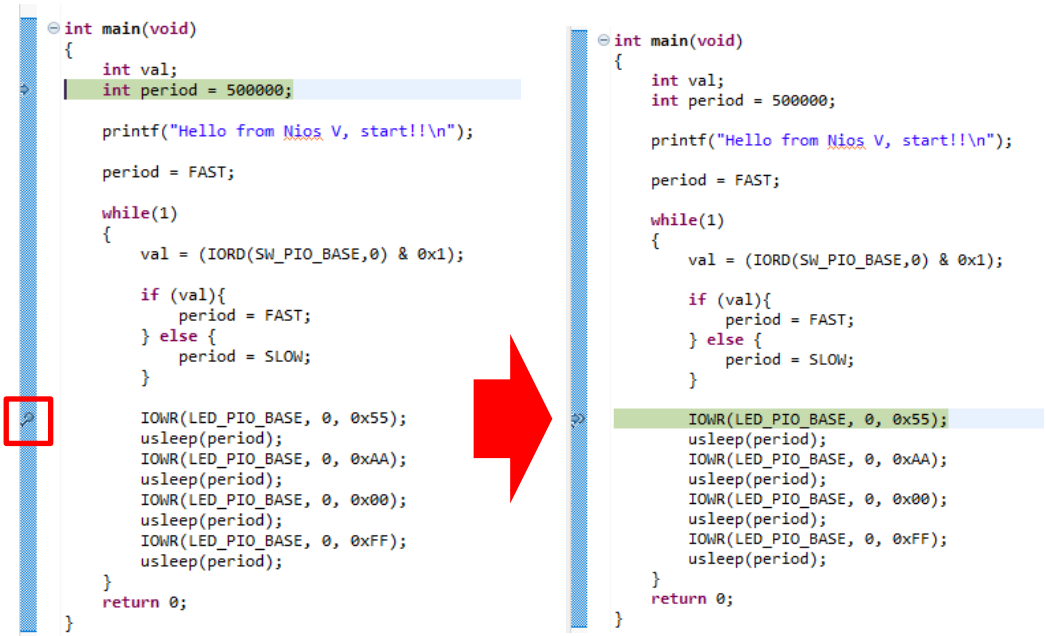

Figure 72 Using Break Points

7-7-3. Using Various Views

●Variable View

The Variables View allows you to view the values of variables being debugged. Variables that have changed during step execution are highlighted in yellow, making it easier to visualize the changes. You can also change the values of variables directly from the Variables View.

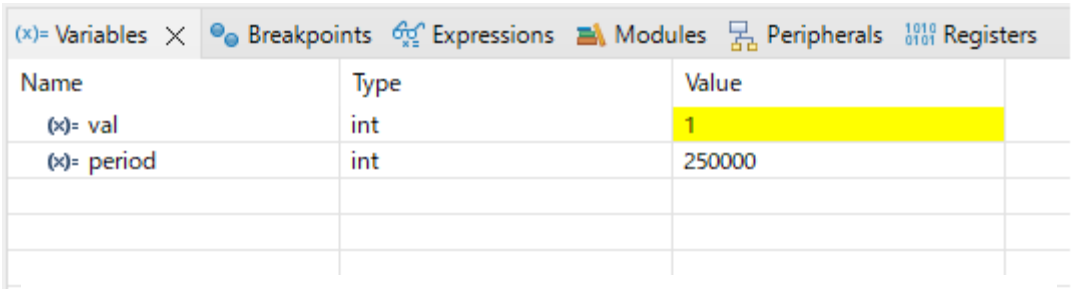

Figure 73 Variables View

●Register View

Register View allows you to view the register contents of a Nios® V processor during debugging. Similar to the Variables View, modified registers are highlighted in yellow, and you can also change values directly from the view.

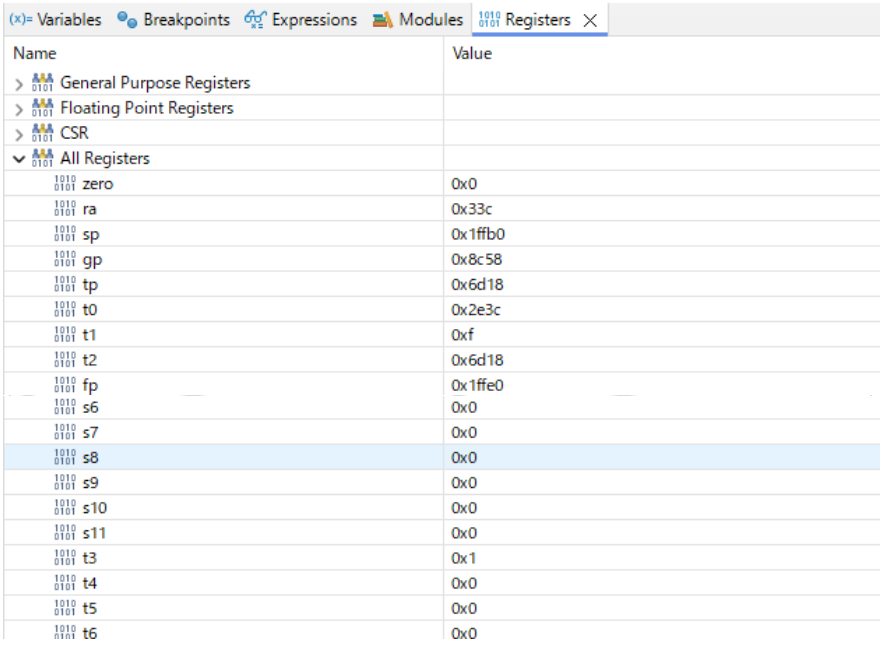

Figure 74 Register View

●Memory View

Memory View allows you to check the memory contents of a specified address during debugging. Changes are displayed in red text for easy visual identification. You can also edit values directly from the memory view, which is useful for real-time operation checking and adjustment. To use the address, use the procedure shown in to Figure 76 add a reference address.

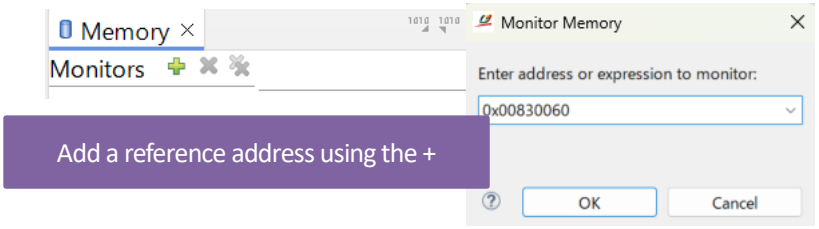

Figure 75 Adding Memory View

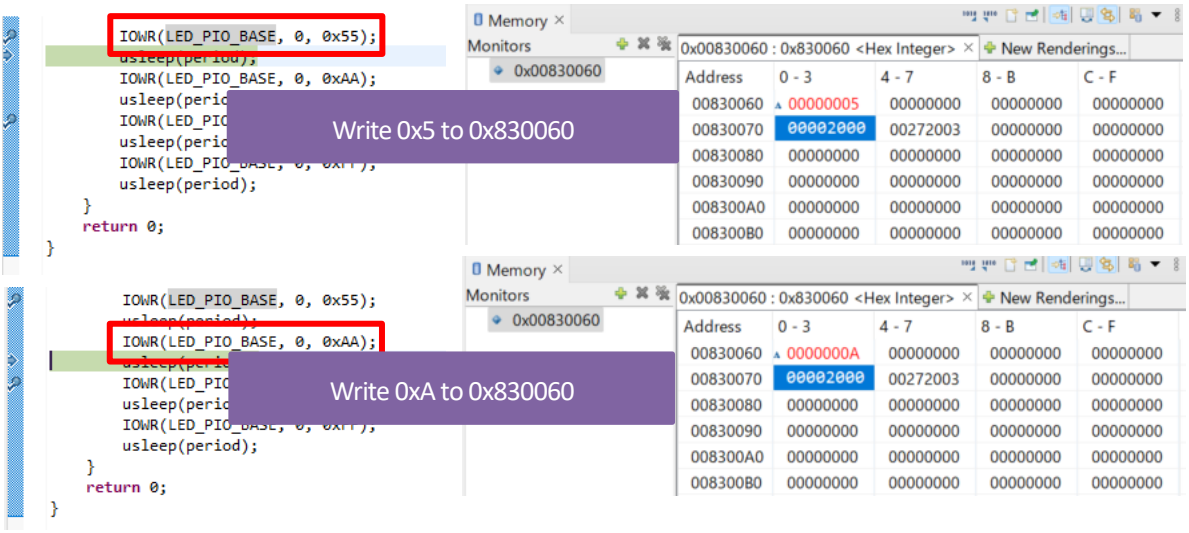

Figure 76 Memory View Debugging

●Dis assembly View

Disassembly View allows you to see the results of disassembling compiled binaries in a debugger. Because you can set breakpoints and step through them, you can see more detailed register behavior than you can with source view debugging.

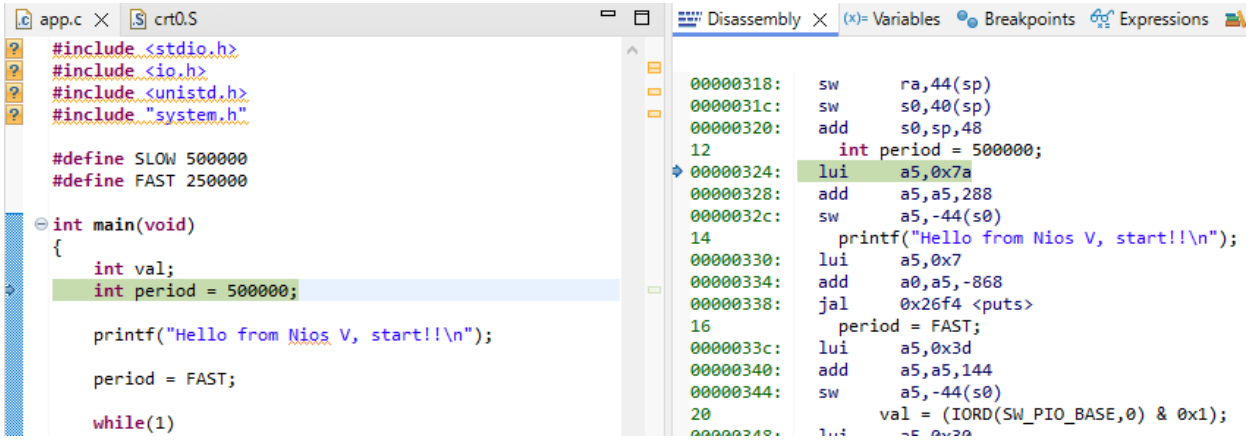

Figure 78 Disassembly View

Note

If the various views are not displayed, you can add them from the ***Window*** menu ⇒ ***Show View*** as shown in Figure 79.

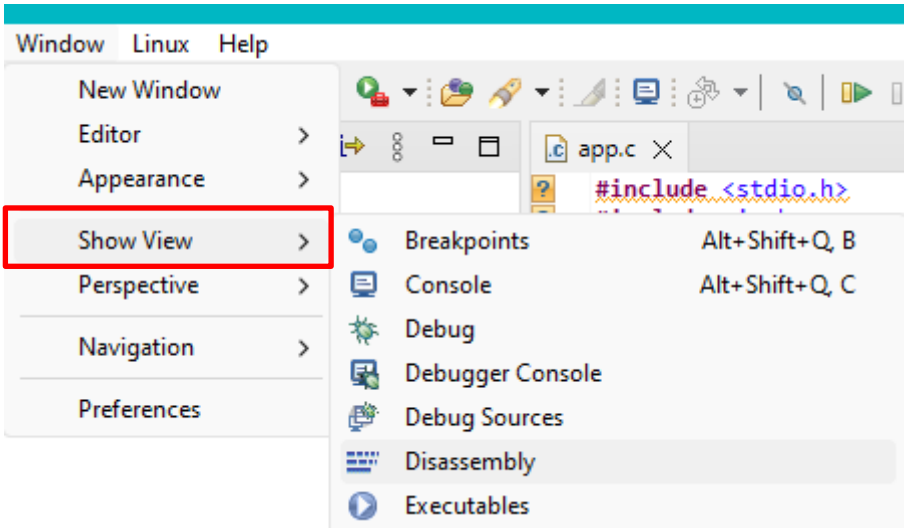

Figure 79 Disassembly View

7-8. Exit the program

1. From the RiscFree IDE File menu, select ***Exit*** to exit the RiscFree IDE.

2. Enter the Exit command into the Nios V command shell and close the shell window.

```
exit
```

3. From the Quartus® Prime File menu, select ***Exit*** to exit Quartus® Prime.

4. Disconnect the AC adapter cable from the board and turn off the power.

This concludes the exercise.

If you have time left, please try the following exercises.

8. Optional

8-1. Signal Tap Logic Analyzer

Signal Tap Logic Analyzer(Signal Tap) is a debugging tool that enables you to check the status of registers and terminals inside an FPGA in real time while the FPGA is operating. It is ideal for real-time operation analysis and is widely used for design verification and troubleshooting.

Signal Tap uses a reference clock signal (CLK) as a trigger to store and capture the status of the terminals registered on the GUI in the FPGA internal memory. This makes it possible to directly observe the signals in the FPGA without using an external logic analyzer.

8-1-1. How to use

In this exercise, the project is created with the Signal Tap configuration file (.stp) embedded.
Because FPGA logic is used, you must write the .sof file before operating.

1. From the Quartus Prime menu, open ***Toos => Signal Tap Logic Analyzer***.

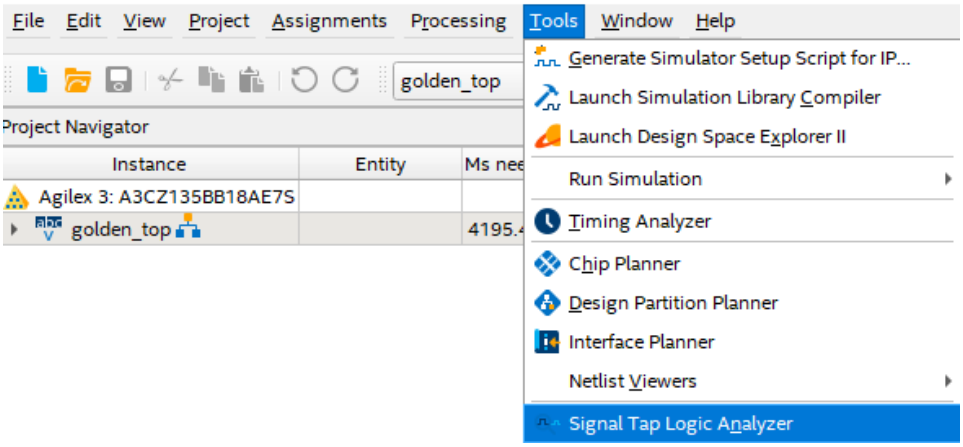

Figure 80 Starting Signal Tap

> ⚠ **Note:**
>
> In Figure 81 through Figure 86 of this document, 8 bit signals are displayed due to the size of the document. You can check the 24 bit counter waveform in the lab used in the exercise.

2. The Signal Tap window opens. In the center of the window, terminals and registers for waveform acquisition are shown. In the .stp of this exercise, reset terminals and counter registers have already been added.
   If yellow Waiting for JTAG or red Program the Device Continue is shown in the upper part of the window, ***set up*** and ***Scan Chain*** in JTAG Chain Configuration in the upper right of the screen to recognize the device.

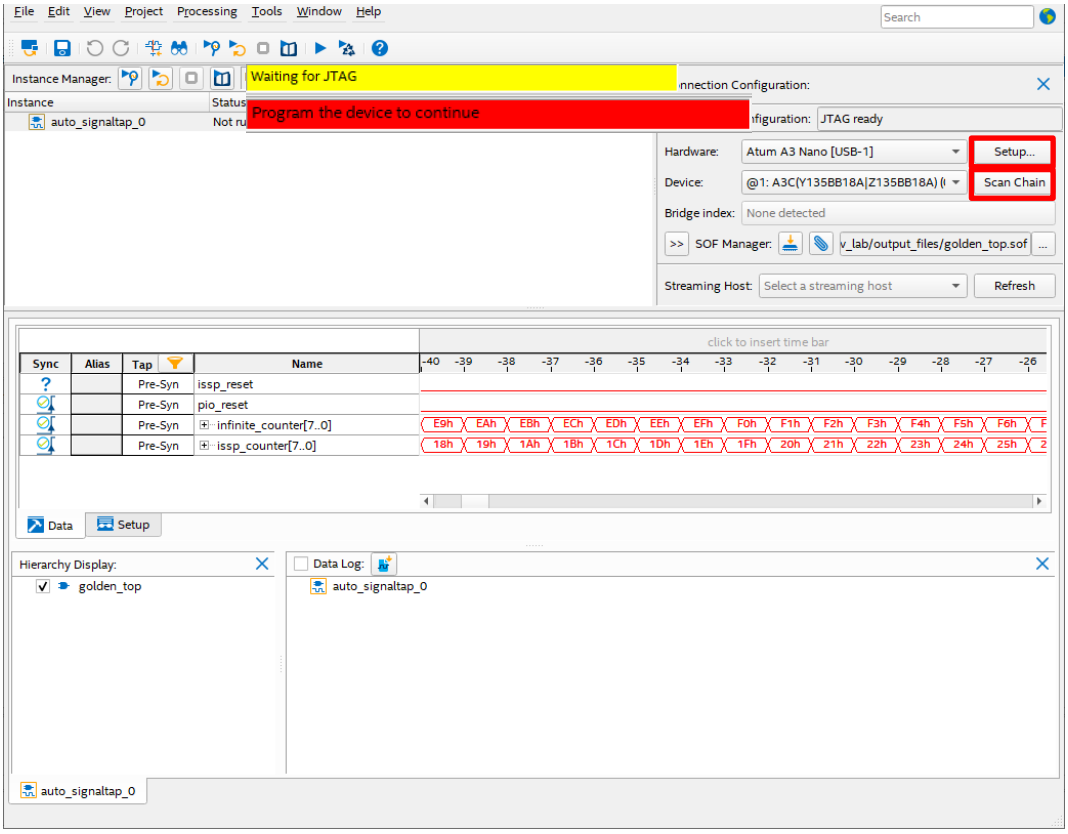

Figure 81 Signal Tap window

3. Select the Set up tab in the center of the window. You can usually add any signal in this window.

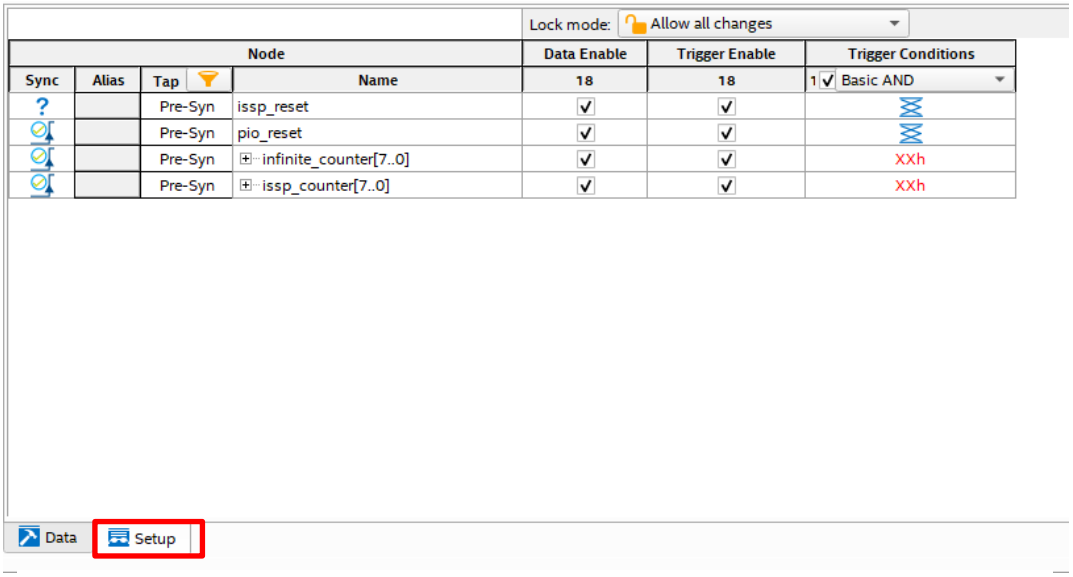

Figure 82 Show Setup tab

4. In the ***Set up*** tab, you can select the trigger for signal tap acquisition. In this exercise, use the inifinite_counter as a trigger to acquire a waveform. Select the ***+*** symbol of the grouped inifinite_counter from the Node Name field in the center of the window to display the terminals in the group.

Node				Data Enable	Trigger Enable	Trigger Conditions
Sync	Alias	Tap	Name	18	18	1 Basic AND
?		Pre-Syn	issp_reset	☑	☑	
		Pre-Syn	pio_reset	☑	☑	
		Pre-Syn	⊟ infinite_counter[7..0]	☑	☑	XXh
		Pre-Syn	infinite_counter[7]	☑	☑	
		Pre-Syn	infinite_counter[6]	☑	☑	
		Pre-Syn	infinite_counter[5]	☑	☑	
		Pre-Syn	infinite_counter[4]	☑	☑	
		Pre-Syn	infinite_counter[3]	☑	☑	
		Pre-Syn	infinite_counter[2]	☑	☑	
		Pre-Syn	infinite_counter[1]	☑	☑	
		Pre-Syn	infinite_counter[0]	☑	☑	
		Pre-Syn	⊞ issp_counter[7..0]	☑	☑	XXh

Figure 83 Display of grouped terminals

5. Right-click the ***Trigger Condition*** column of Infinite _counter [0] and select ***Rising Edge***.

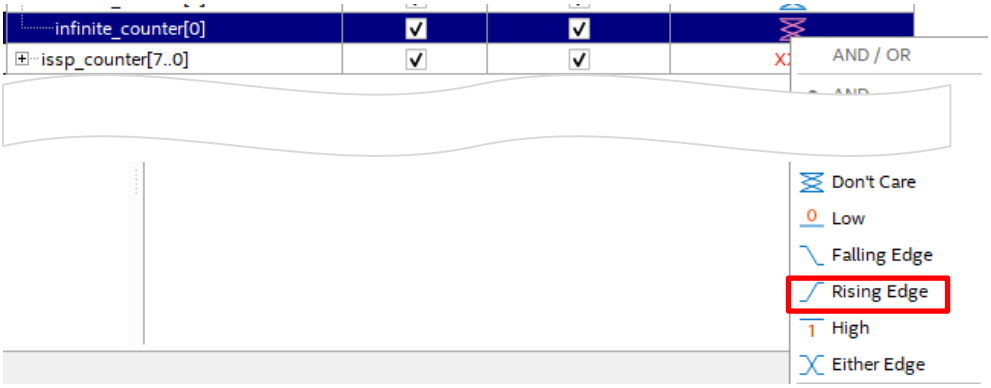

Figure 84 Selecting trigger edges

6. Select ***Run Analysis*** from Instance Manger at the top of the window. The one-shot waveform with the set trigger is displayed in the ***Data*** tab.

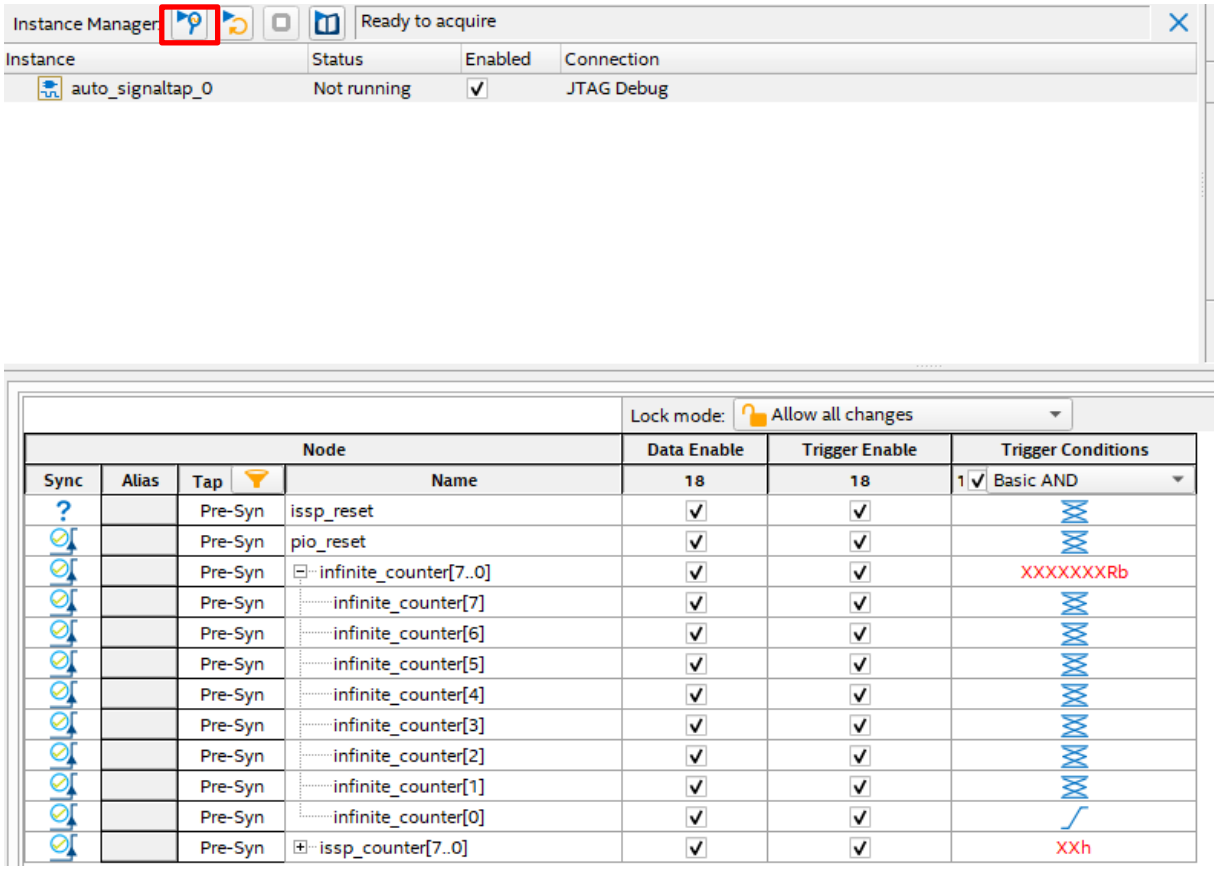

Figure 85 Settings before measurement

8-1-2. Execution result

The following figure shows the execution result. You can check the register value of each counter and the signal of the Reset terminal. You can also select ***Auto run Analysis*** from Instance Manger to continuously acquire waveforms for each trigger condition.

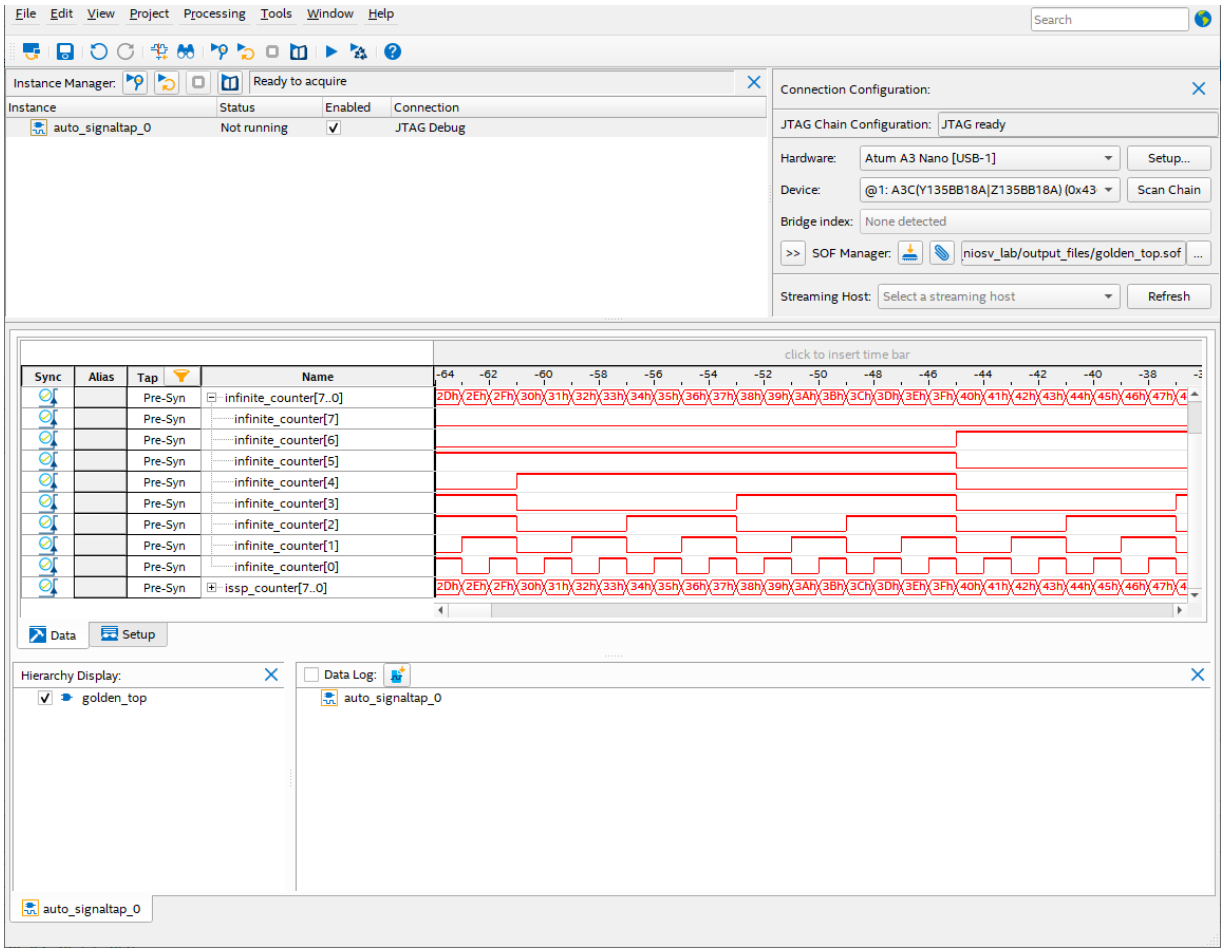

Figure 86 Signal Tap waveform

8-2. ISSP (In System Souce and Probe)

ISSP (In-System Source and Probe) is a tool that incorporates externally accessible control logic into FPGA logic and enables you to input signals (Source) and observe signals (Probe) from the GUI.

In this exercise, we will use the Source function and control counter reset by inputting signals from the GUI. This function is very useful for debugging purposes such as test input and operation check.

8-2-1. Usage

It can be accessed from the GUI by incorporating dedicated logic as shown below.

```
// ISSP Reset
altsource_probe #(
    .probe_width(0),
    .source_width(1),
    .source_initial_value("0")
) issp_reset_inst (
    .source(issp_reset)
);
```

Figure 87 HDL in ISSP

In this exercise, the top design (golden_top.v) that incorporates logic has already been created. In addition, since FPGA logic is used, it must be operated after writing sof.

1. From the Quartus Prime menu, open ***Tools => In System Source and Probe Editor***.

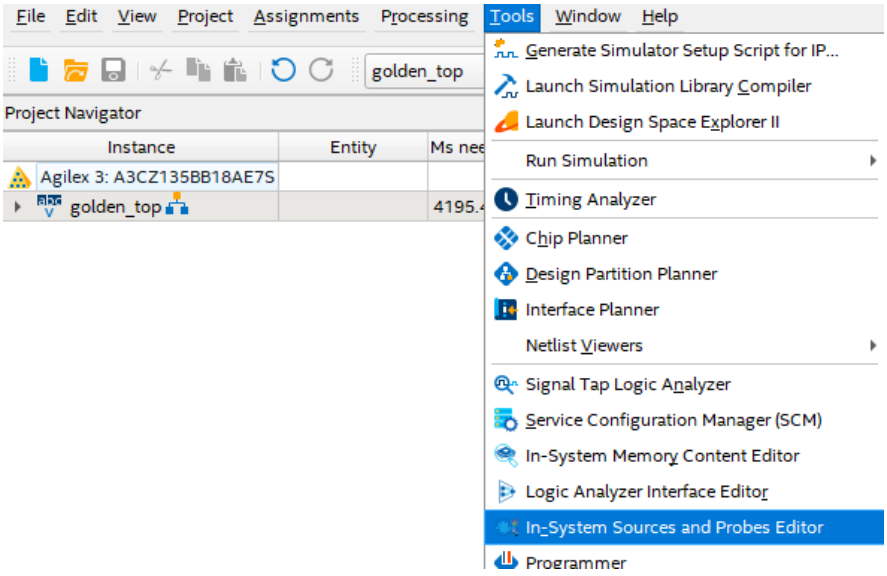

Figure 88 Launching ISSP

2. Make sure you see Source [0.. 0] in the Name column of the list at the bottom of the window. If you see Waiting for JTAG in yellow letters at the top of the screen, check ***Set up*** and ***Scan Chain*** in ***JTAG Chain Configuration*** at the top right of the screen to recognize the device.

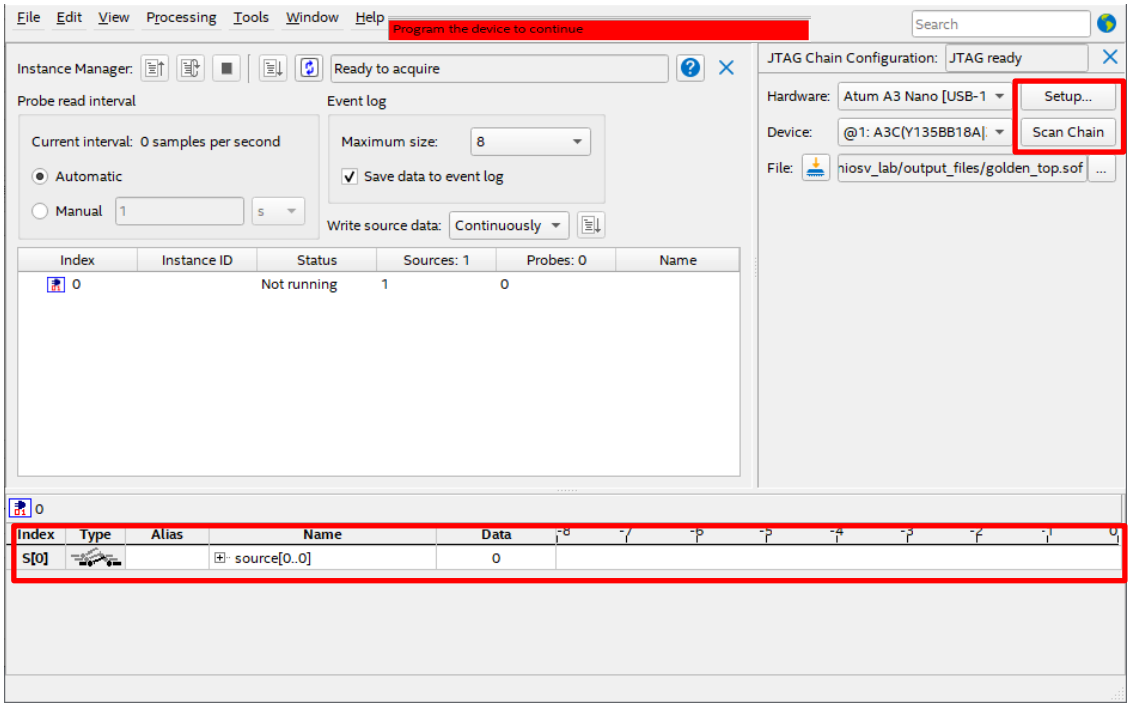

Figure 89 ISSP Window

3. Since the ***Source*** signals in the ***Name*** column in the currently displayed list are grouped, click the plus on the left to display the signals in the group.

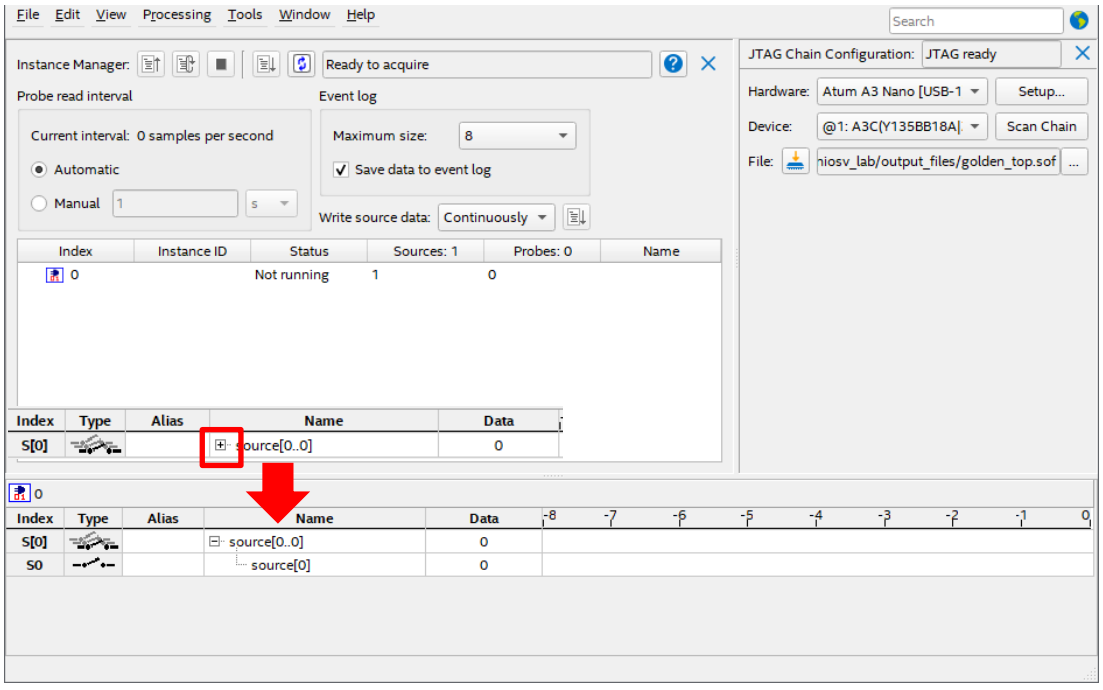

Figure 90 Displaying Source Signals

4. The ***Data*** field of the Source signal in the Group shows the current Source signal status. Click the ***Data*** field to change the signal status.

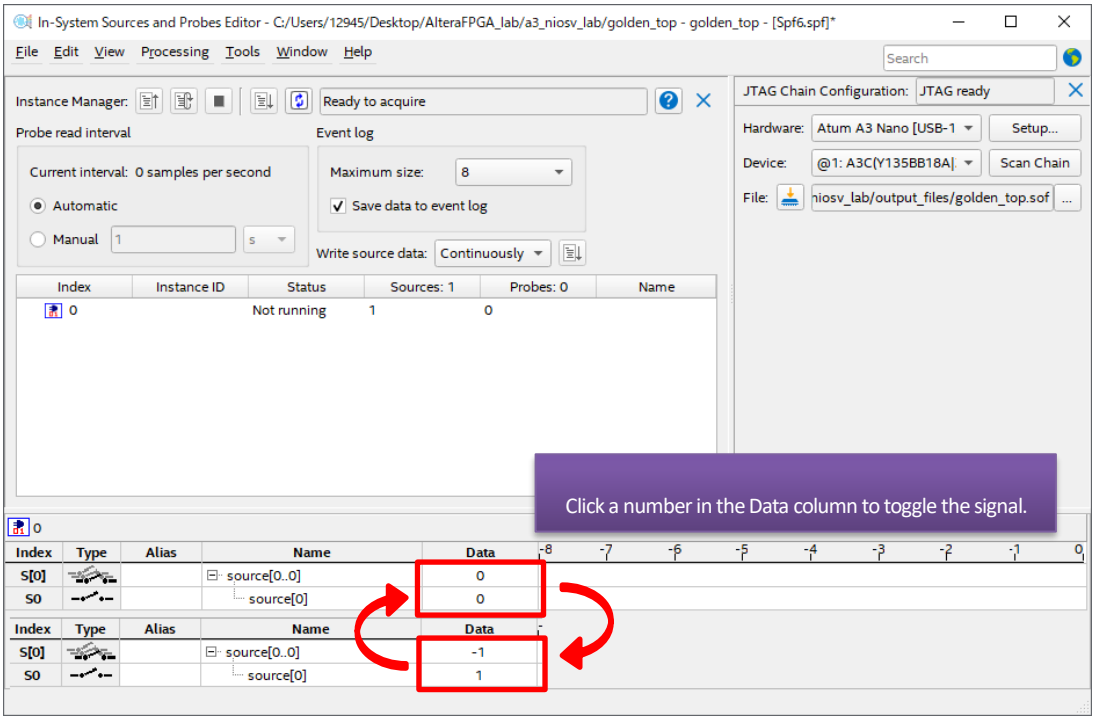

Figure 91 Manipulating Source Signals

8-2-2. Execution result

In this exercise, a counter and the Source signal of ISSP are added to Signal Tap, so check from Signal Tap.

You can see that the ***Data*** column of the ISSP Source signal is linked with the ISSP Source of Signal TAP. You can also verify that the counter is reset.

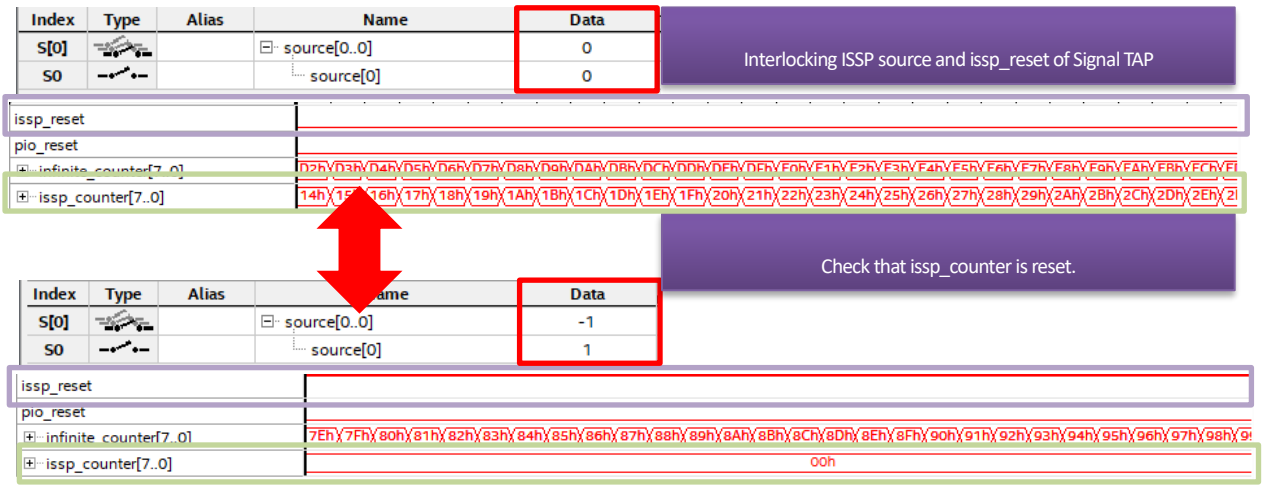

Figure 92 Interlocking ISSP and Signal Tap

8-3. System Consoloe

System Console is a tool that allows you to control FPGA internal signals via JTAG. It can be automated using TCL scripts, and register operations and debugging can be performed in real time without recompiling. It is also easy to link with the IP core, which is useful for efficient verification.

8-3-1. Usage

By installing a JTAG-accessible Master IP, you can perform command-based debugging from a dedicated GUI. In this exercise, we have prepared a tcl script file that describes all the commands to be executed, and you can also operate a GUI that uses the Tool Kit API that can be operated on the System Console. Because this function uses an FPGA IP, you must download sof before performing the operation.

1. From the Quartus Prime menu, select ***Tools => System Debugging Tools => System Console***.

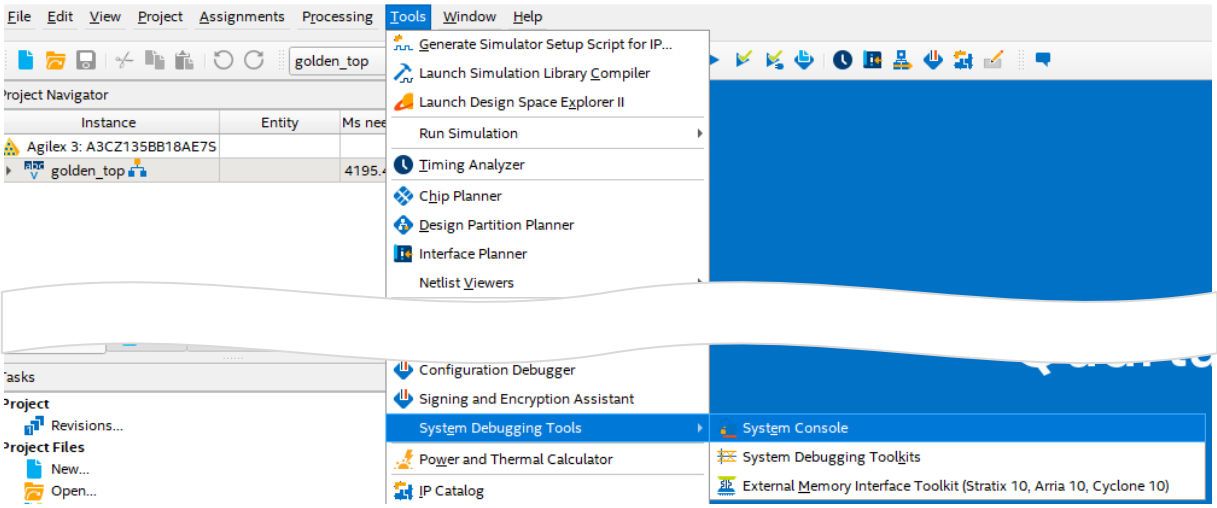

Figure 93 Starting the System Console

2. After the System Console window appears, wait until the ***Tcl Console*** window in the lower right corner becomes operable (%).

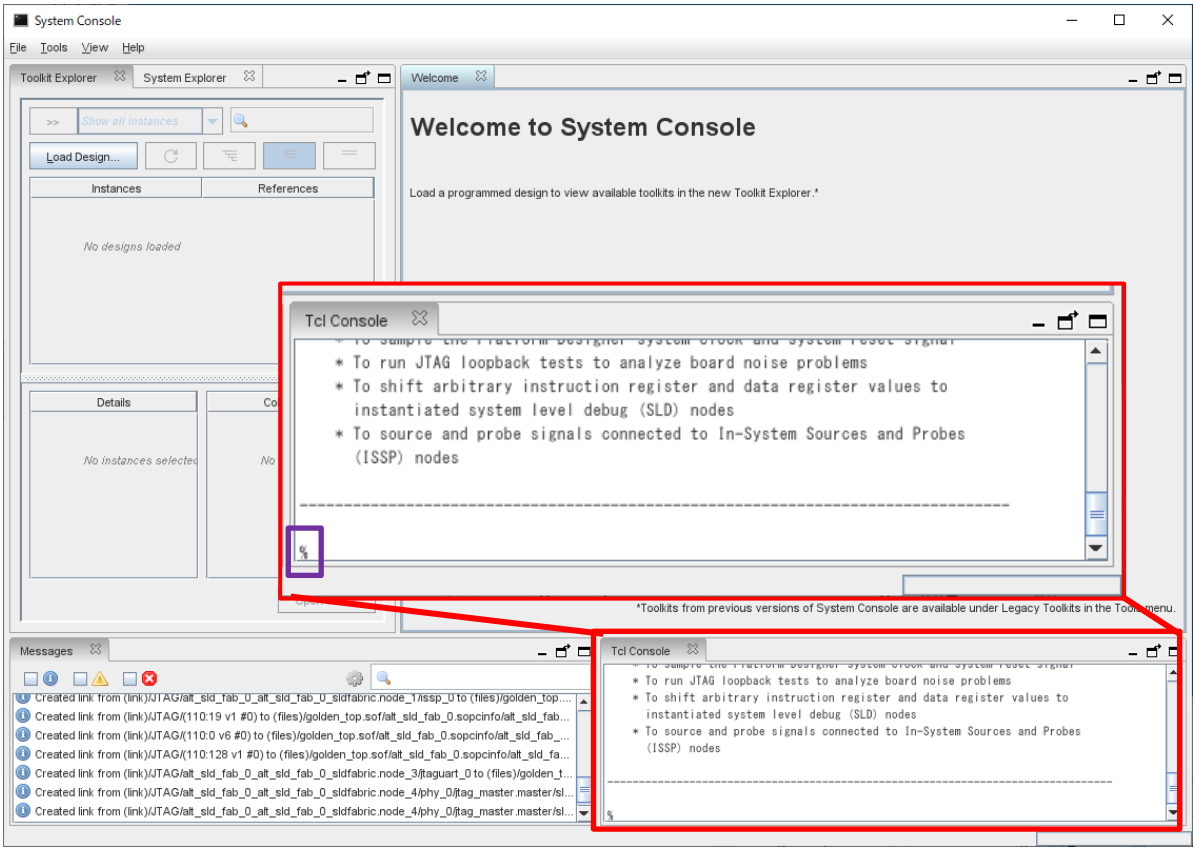

Figure 94 System Console window

3. Enter ***source demo.tcl*** in the Tcl Console window.

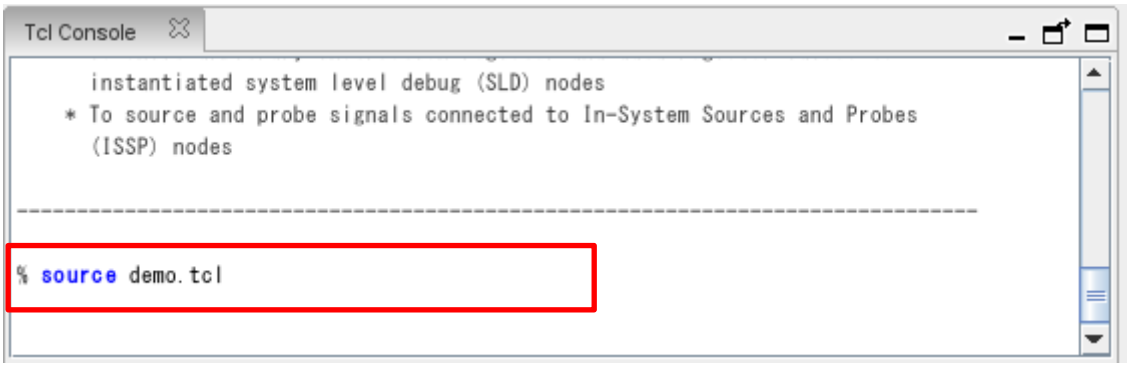

Figure 95 Executing a tcl File

4. The GUI developed with the Tool kit API is displayed.

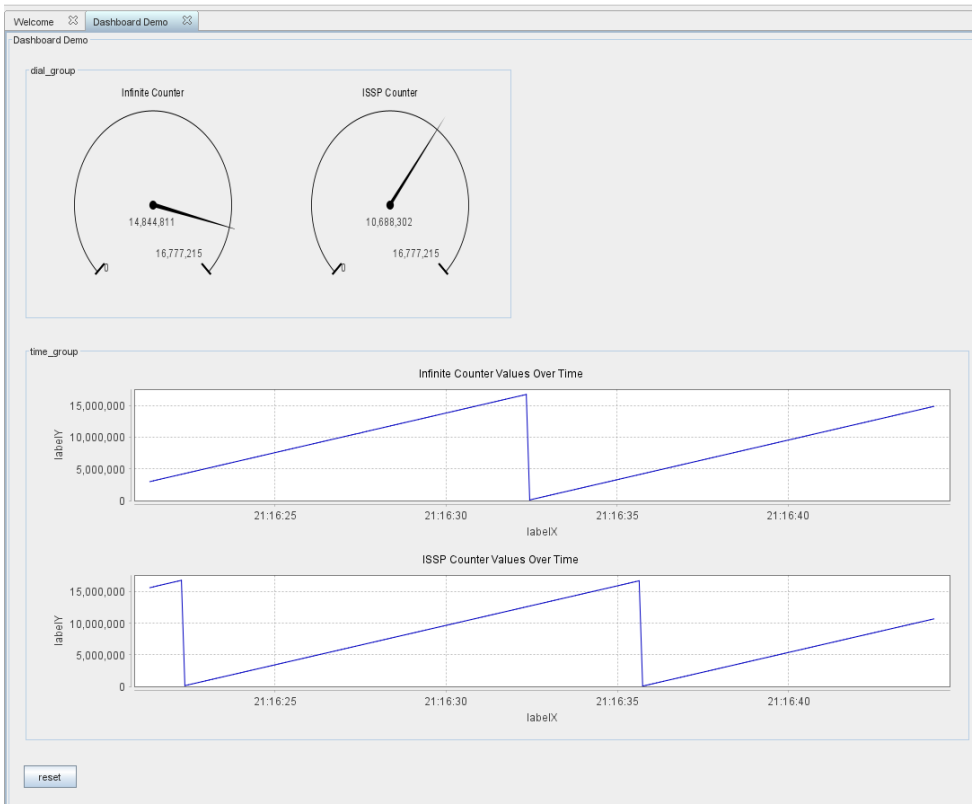

Figure 96 Displaying the GUI

8-3-2. Execution result

The displayed meter and analog value are the graphic representation of the counter value shown by Signal Tap. You can confirm that the counter is reset by pressing the Reset button on the GUI. You can also confirm that the ISSP Counter is reset by toggling the Source signal from the ISSP GUI.

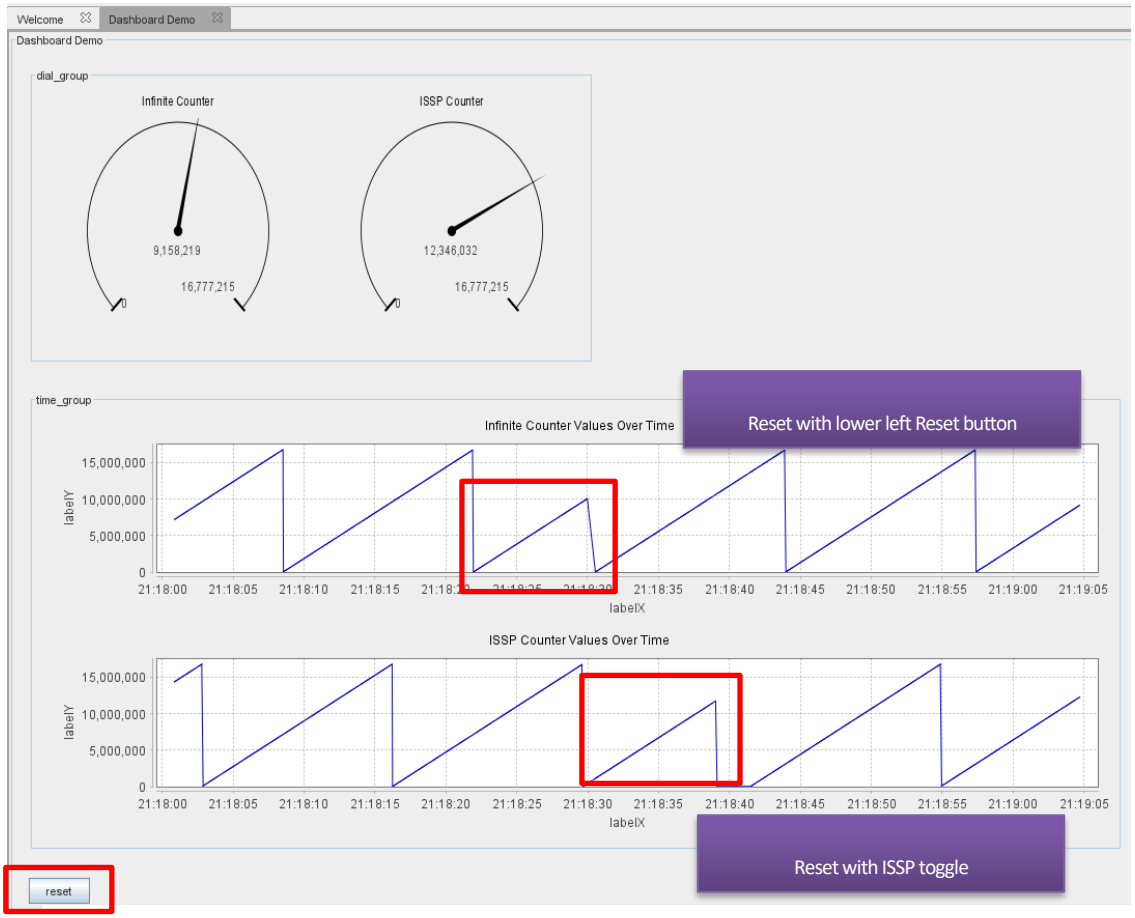

Figure 97 GUI operation

Reference:

For a .tcl file for creating a dashboard for the System Console, please refer to the following link for our company article on how to create a dashboard.

Using the Toolkit API GUI for System Console Commands - Semiconductor Business - Macnica

This concludes all of the exercises. Congratulations!