My First Nios II for Altera DE2i-150 Board

CONTENTS

Chapter 1 Hardware Design

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

1.1 Required Features

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

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

- Nios II processor core, that's where the software will be executed
- On-chip memory to store and run the software
- JTAG link for communication between the host computer and target
- hardware (typically using a USB-Blaster cable)
- LED peripheral I/O (PIO), be used as indicators

1.2 Creation of Hardware Design

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

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

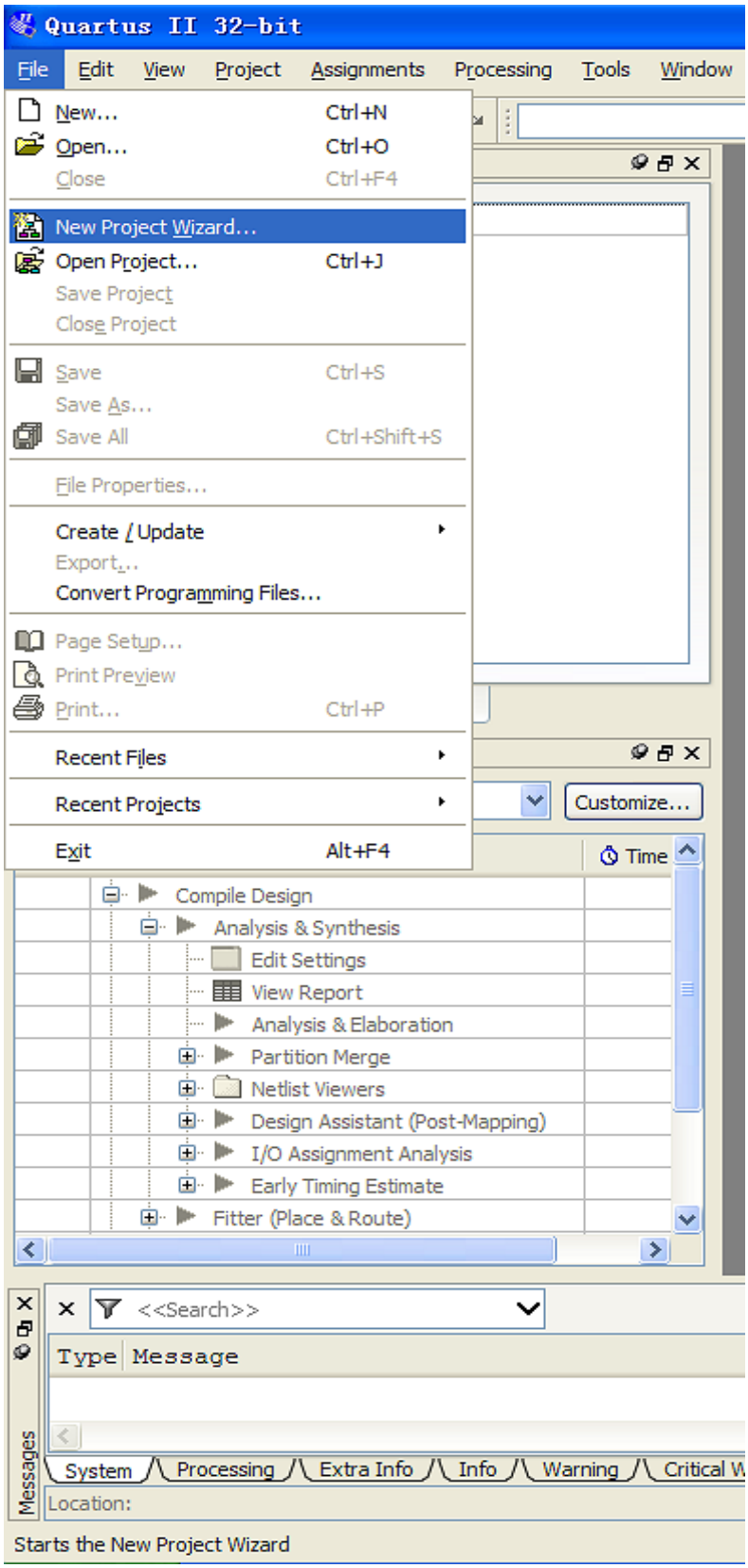

Figure 1-1 Start to Create a New Project

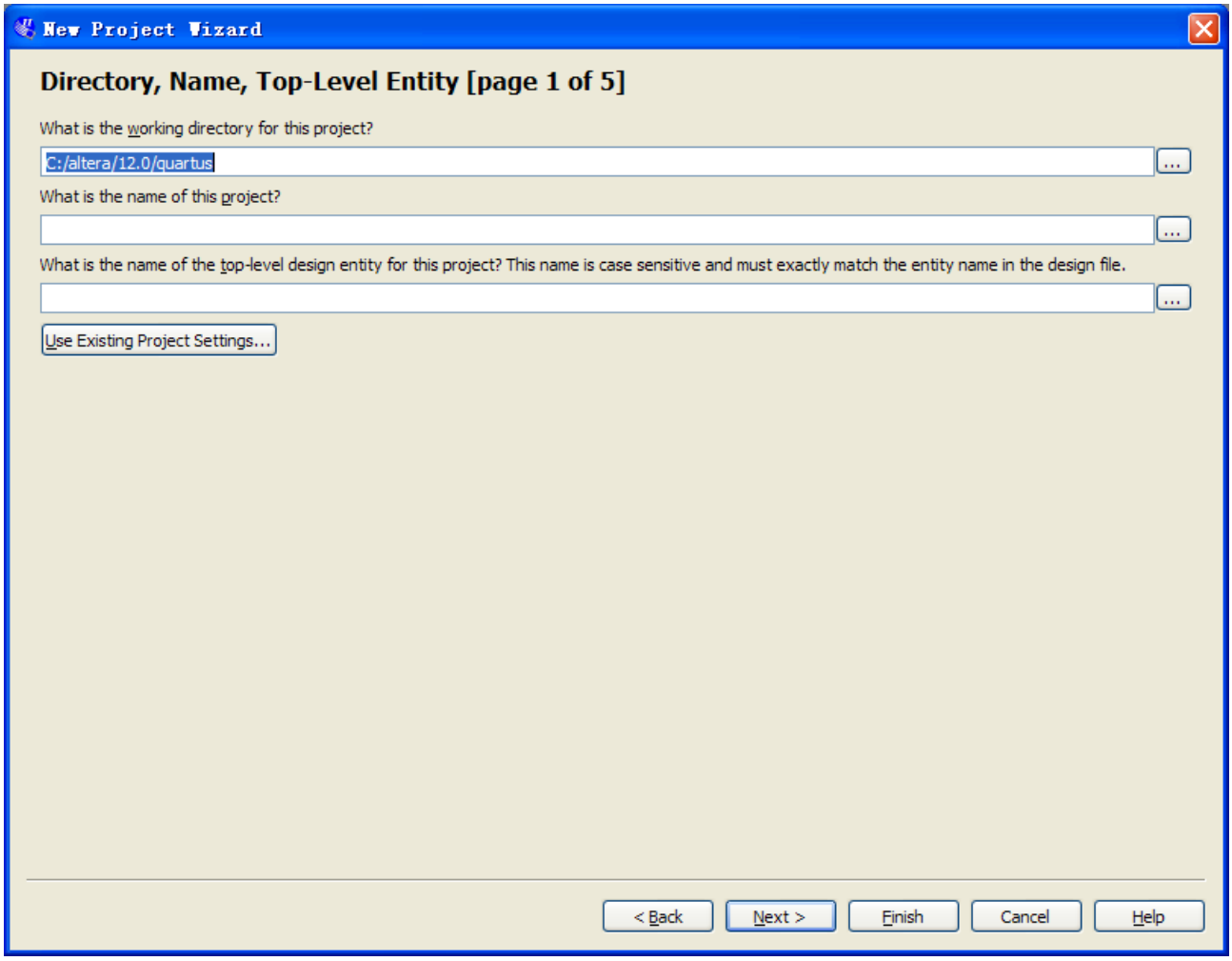

Figure 1-2 New Project Wizard

2. Choose a working directory for this project, type project name and top-level entity name as shown in Figure 1-3. Then click **Next**, you will see a window as shown in Figure 1-4.

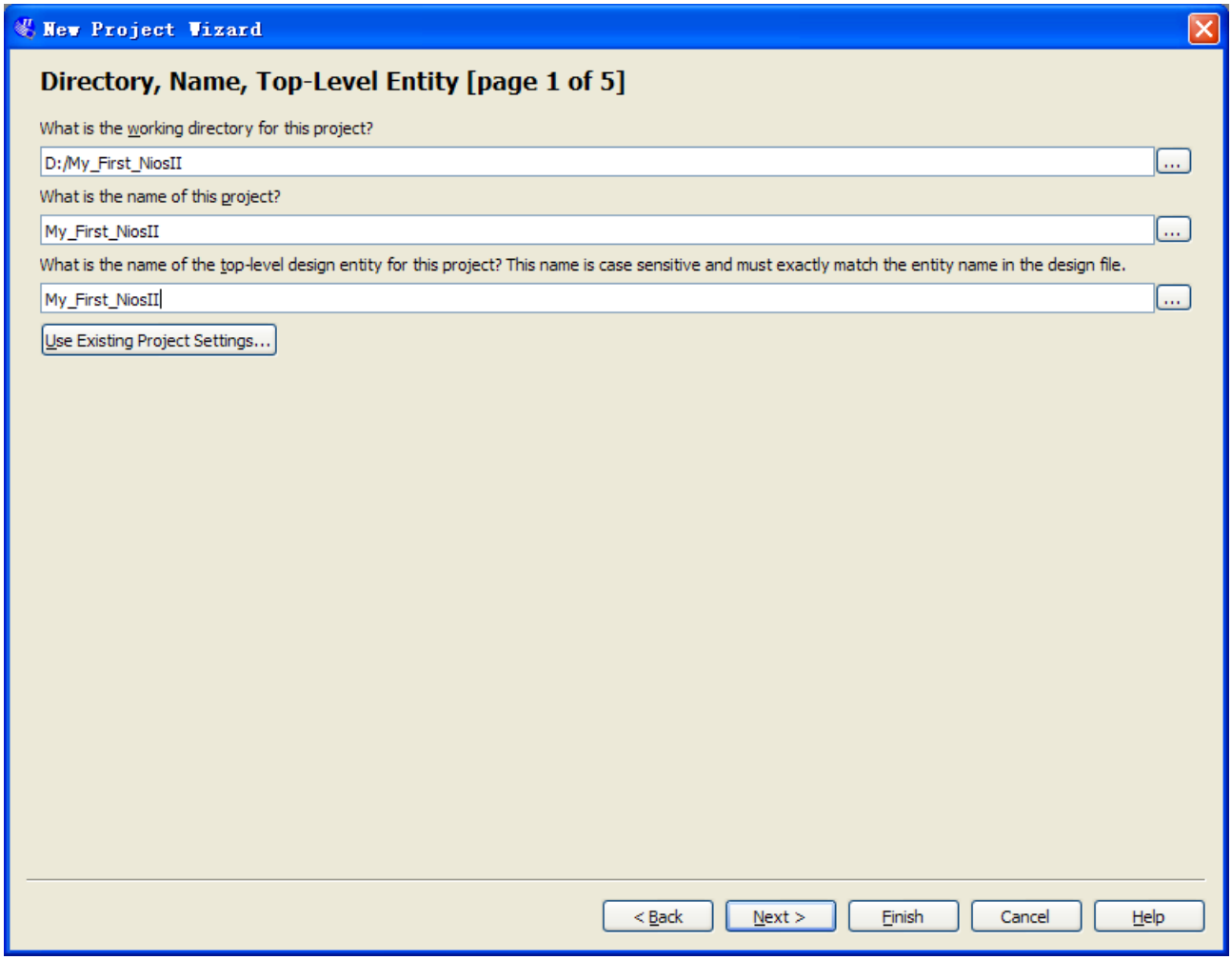

Figure 1-3 Input the working directory, the name of project, top-level design entity

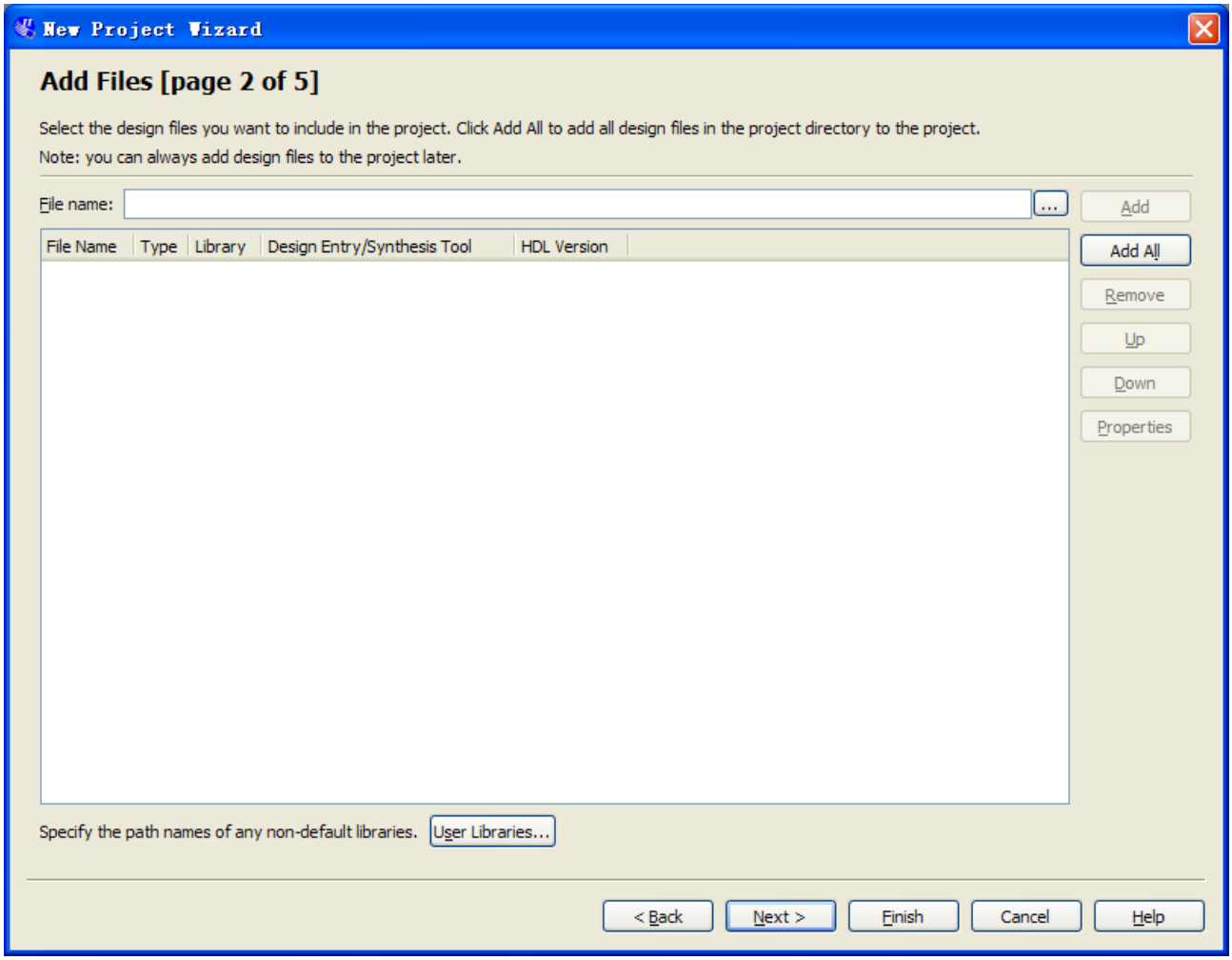

Figure 1-4 New Project Wizard: Add Files [page 2 of 5]

3. Click **Next** to next window. We choose device family and device settings. You should choose settings the same as the Figure 1-5. Then click **Next** to next window as shown in Figure 1-6.

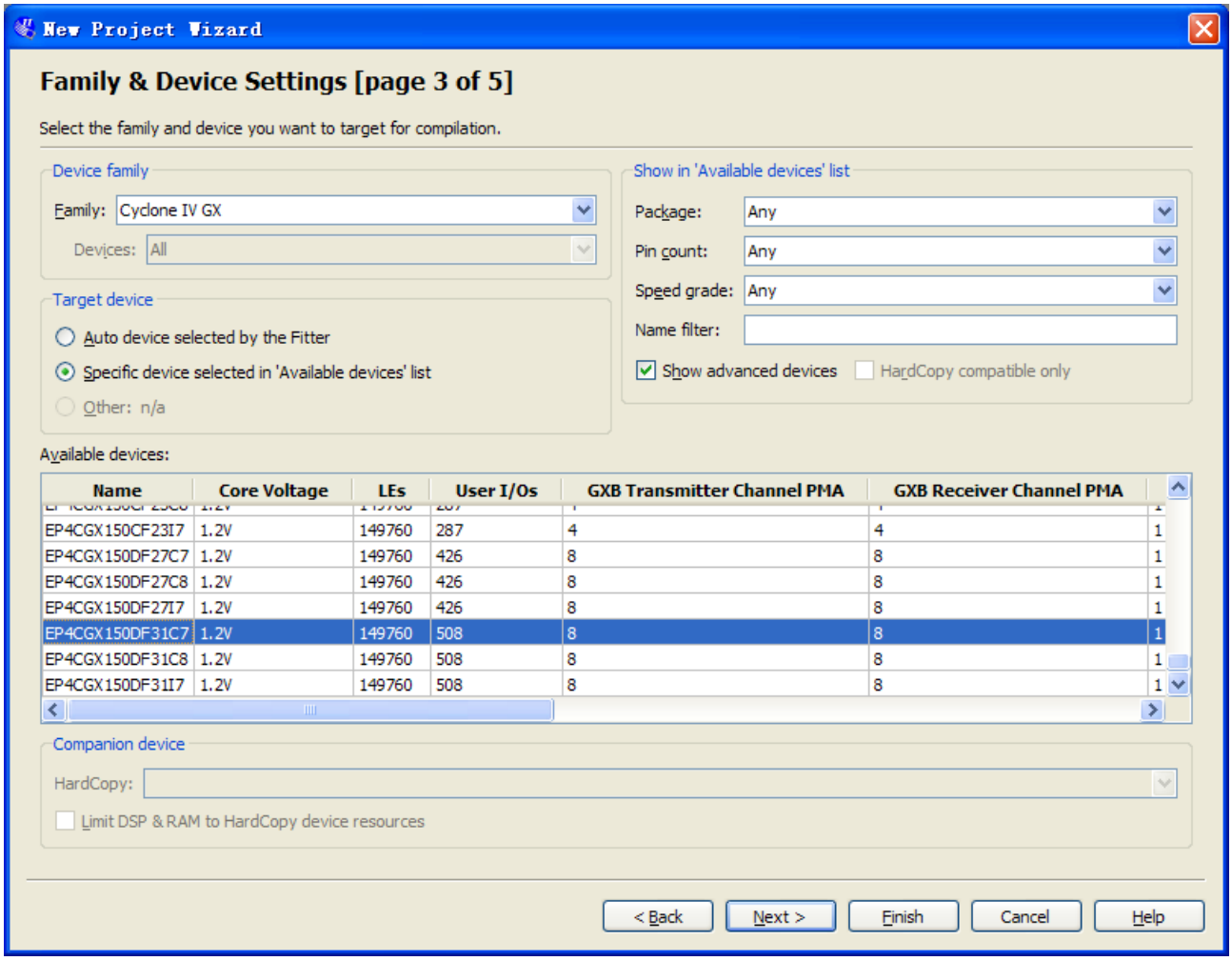

Figure 1-5 New Project Wizard: Family & Device Settings [page 3 of 5]

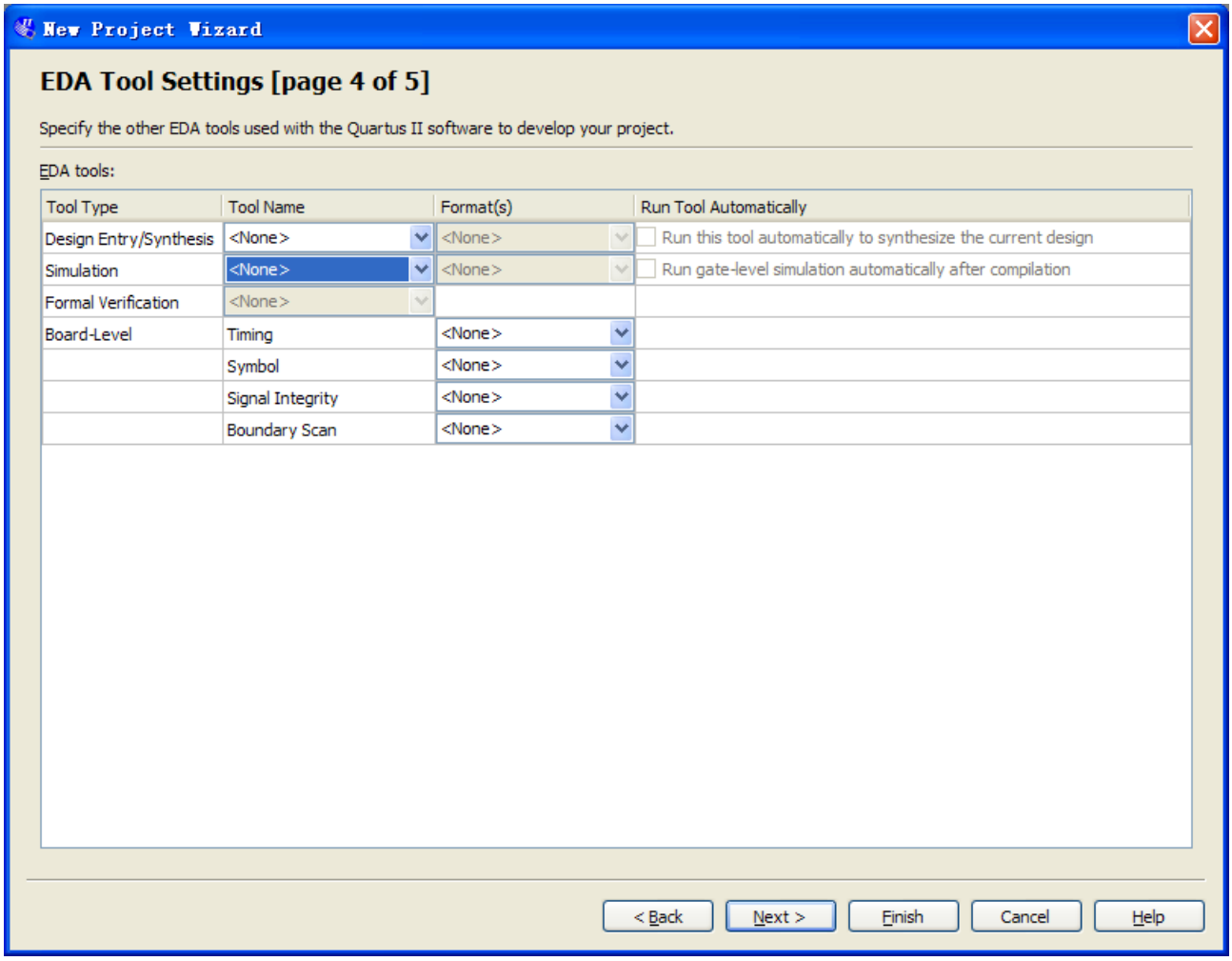

Figure 1-6 New Project Wizard: EDA Tool Settings [page 4 of 5]

4. Click **Next** and will see a window as shown in Figure 1-7. Figure 1-7 is a summary about our new project. Click **Finish** to finish new project. Figure 1-8 show a new complete project.

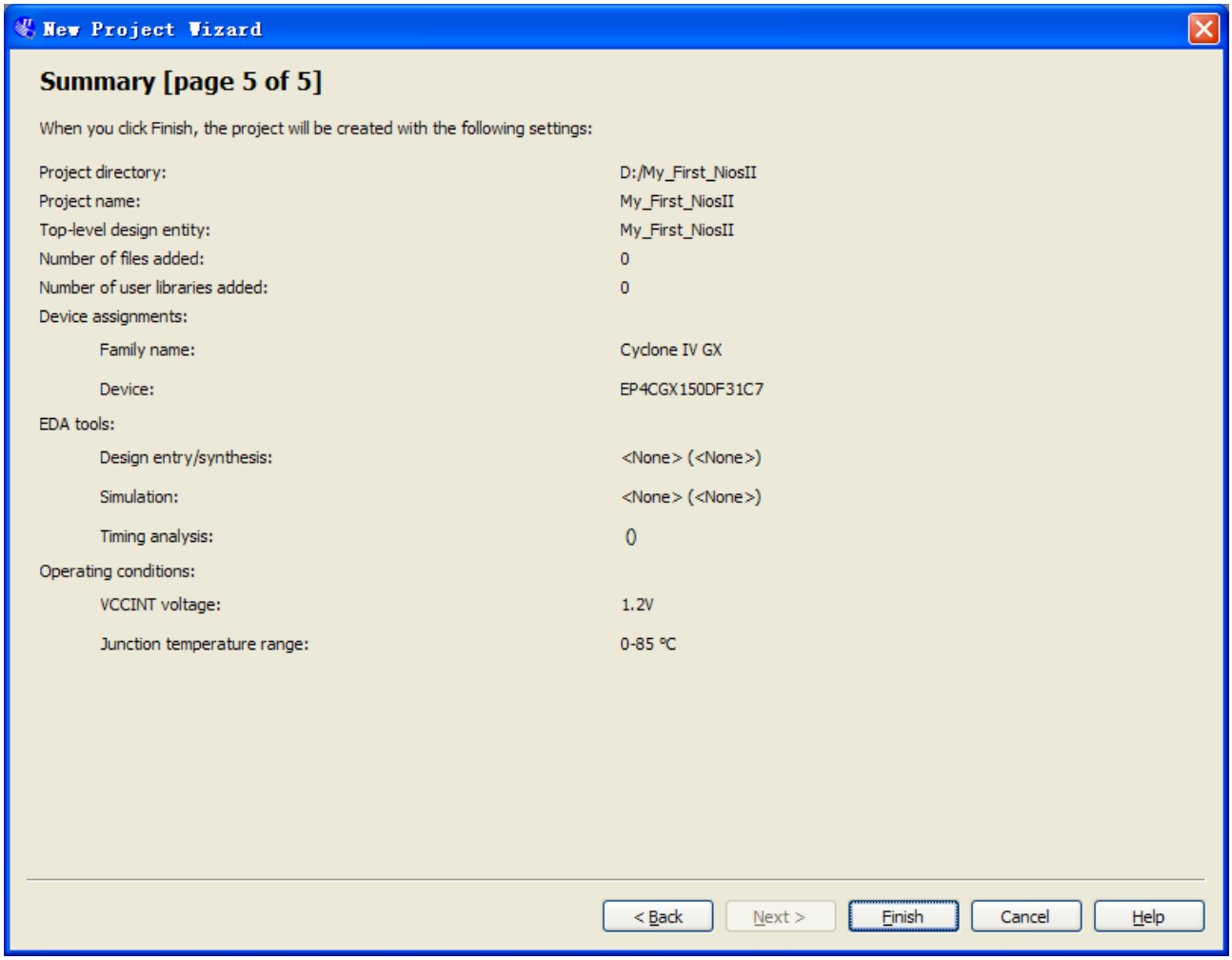

Figure 1-7 New Project Wizard: Summary [page 5 of 5]

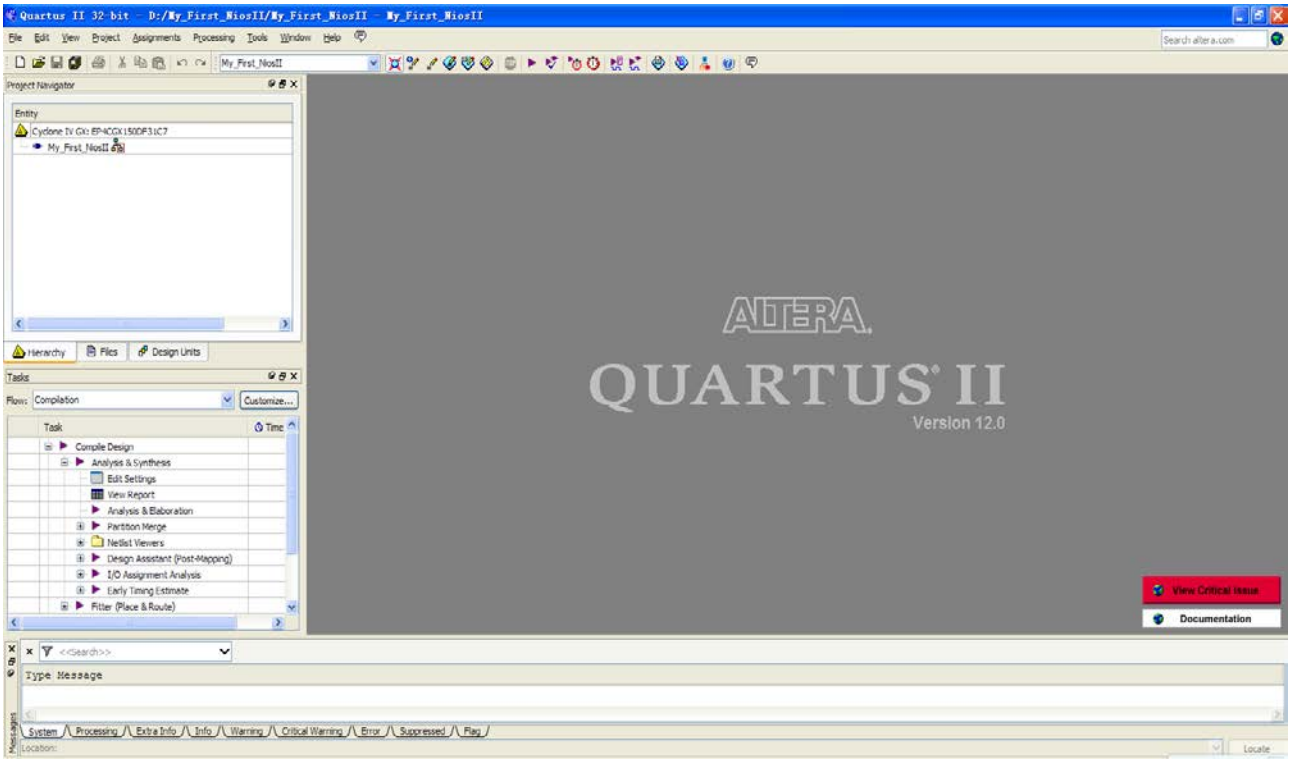

Figure 1-8 A New Complete Project

5. Choose **Tools** > **Qsys** to open new **Qsys** system wizard . See Figure 1-9 and Figure 1-10.

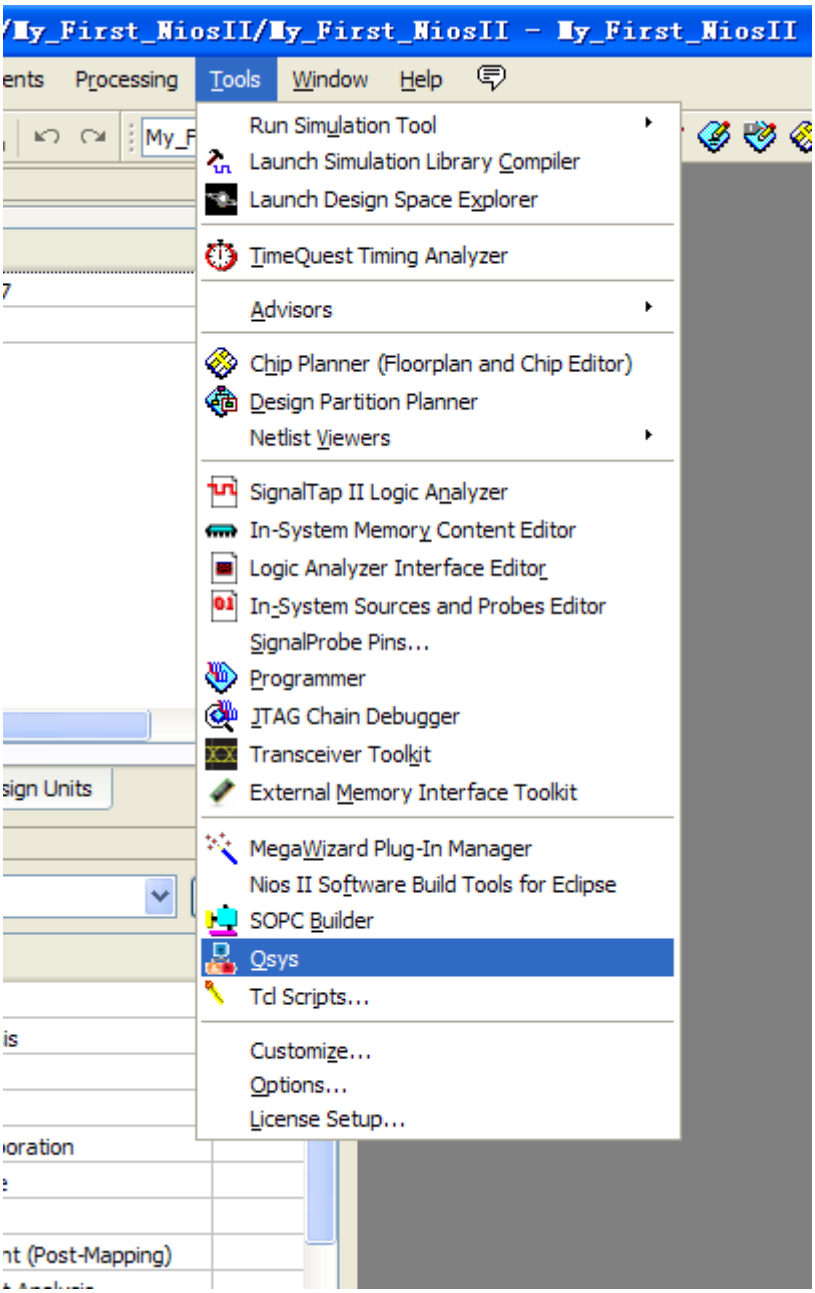

Figure 1-9 Qsys Menu

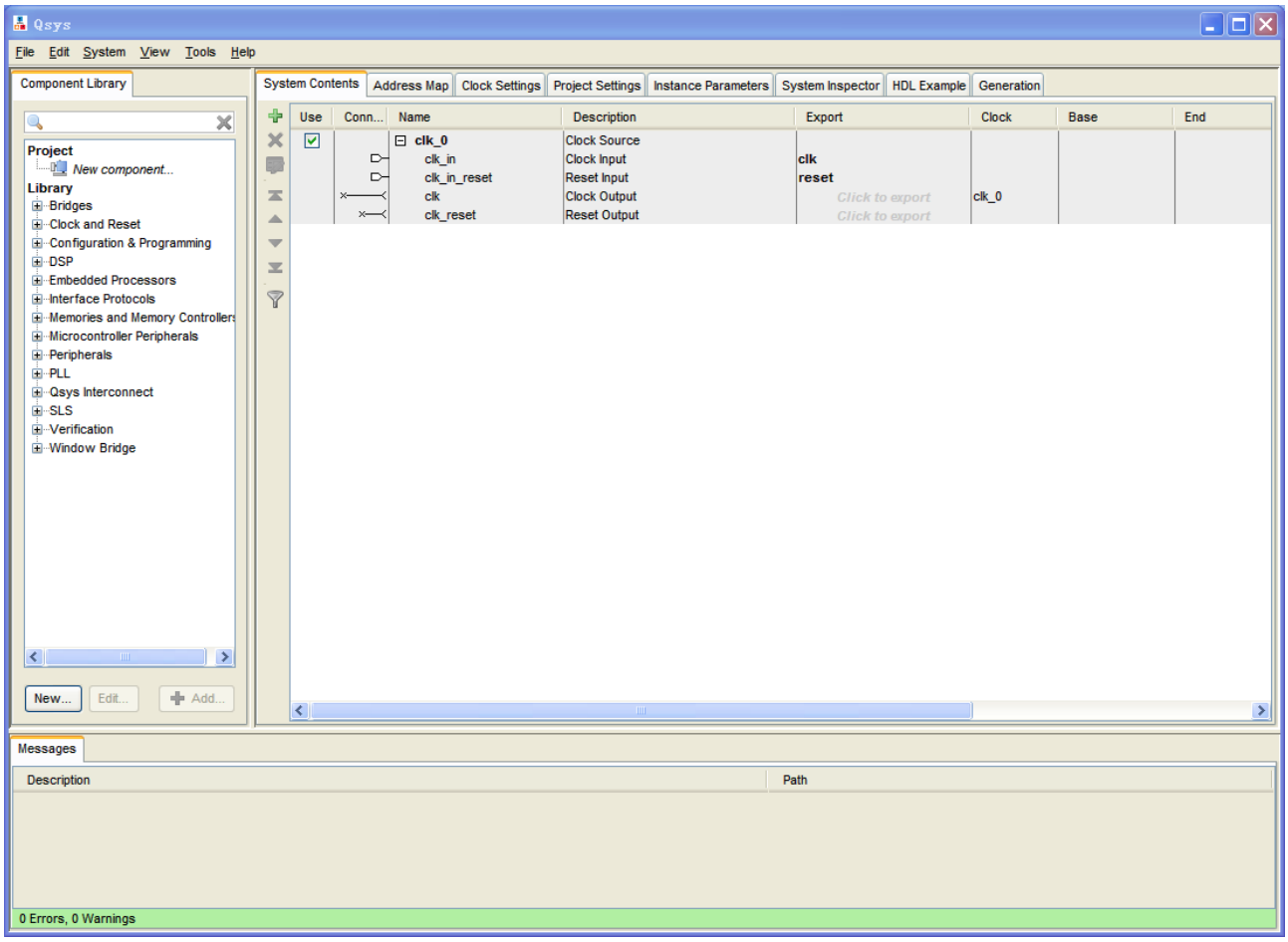

Figure 1-10 Create New Qsys System

6. **Save as** the **System** as shown in Figure 1-11.

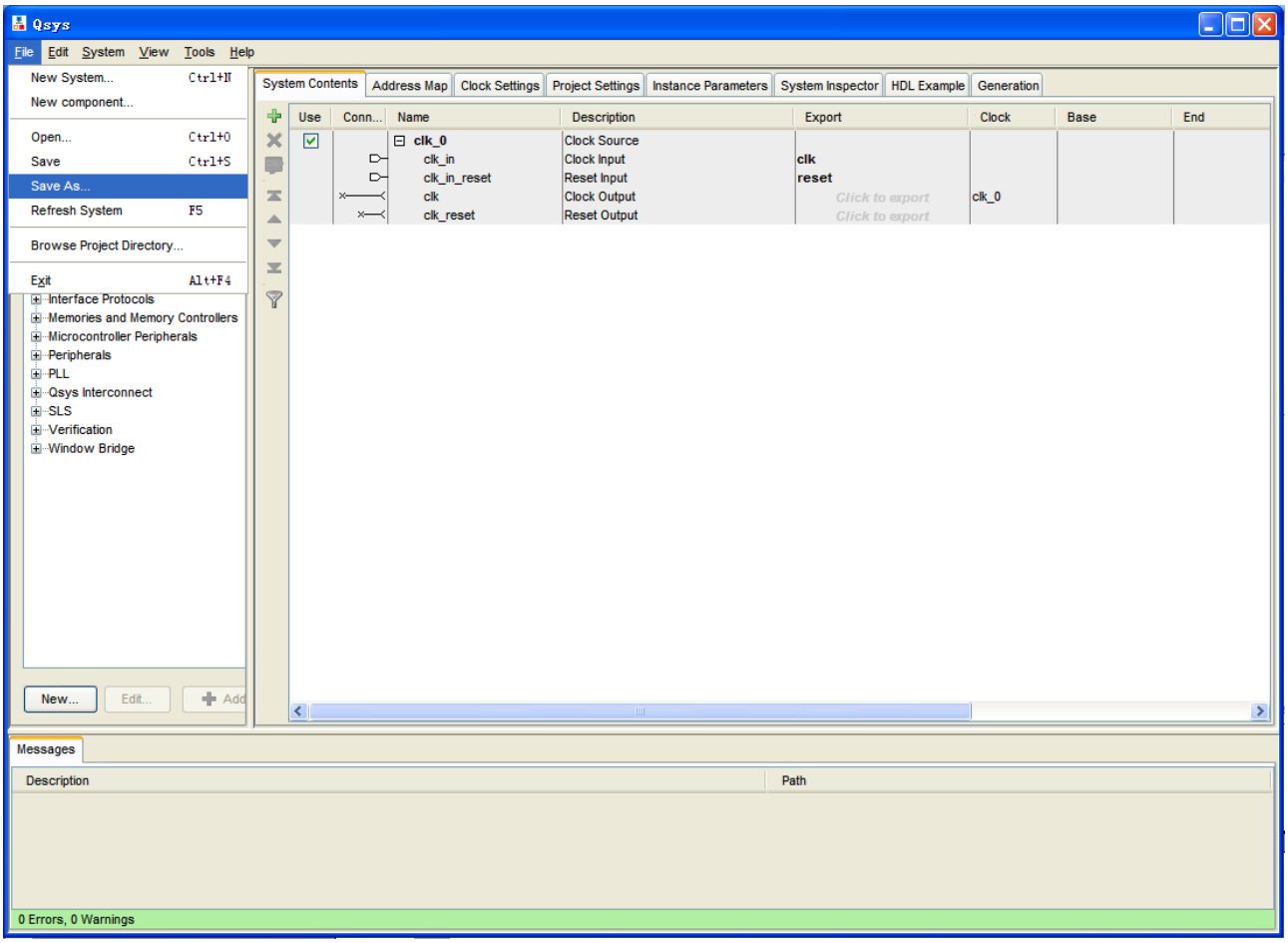

Figure 1-11 Save System

7. Rename **System Name** as shown in Figure 1-12. Click **Save** and your will see a window as shown in Figure 1-13.

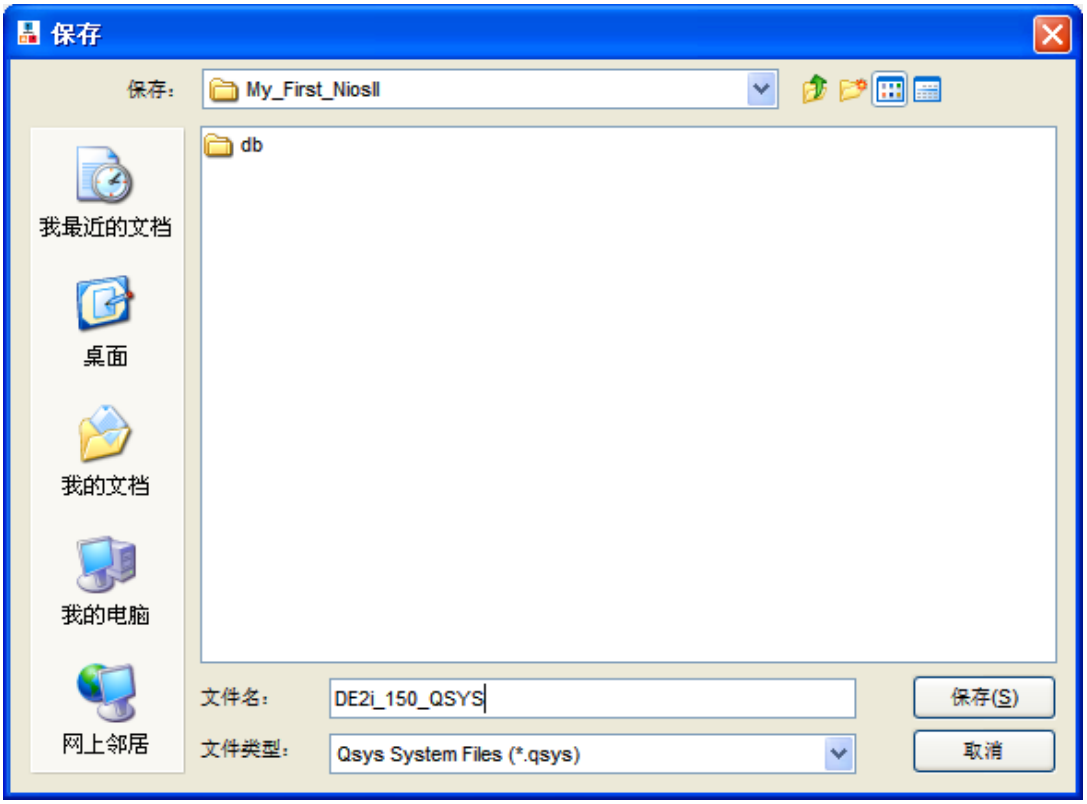

Figure 1-12 Rename System

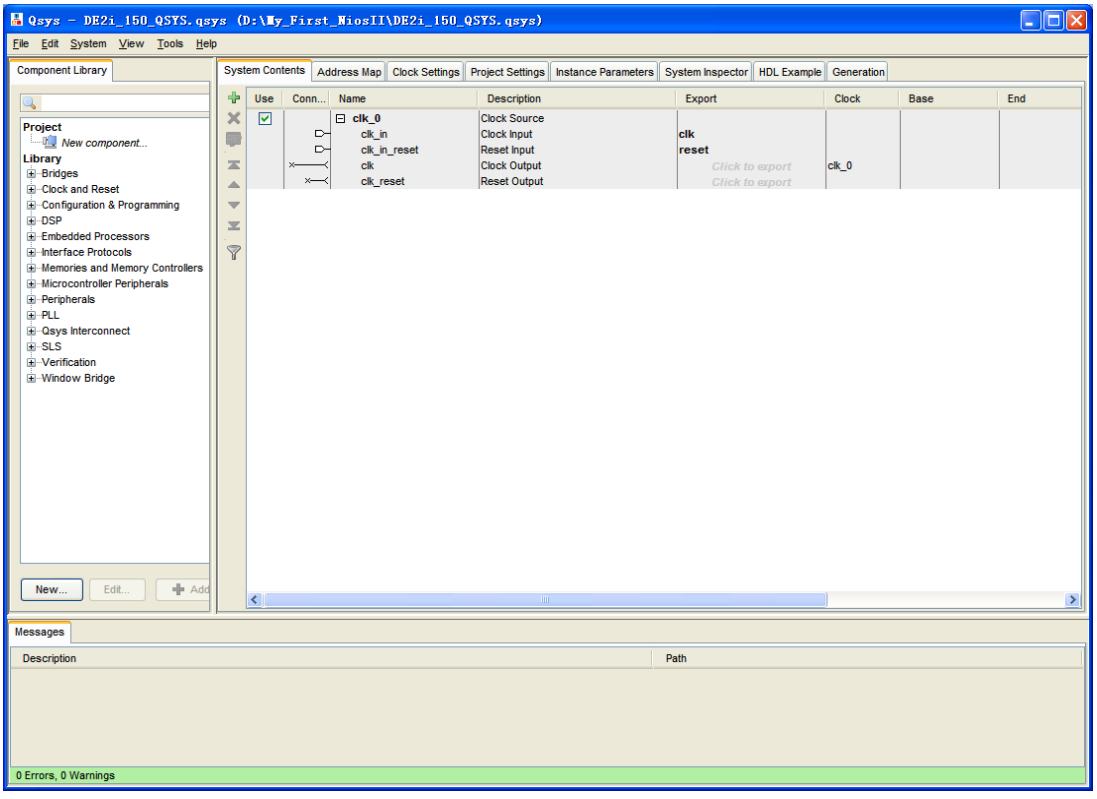

Figure 1-13 A New System

8. Click the Name of the Clock Settings table, rename **clk_0** to **clk_50**. Press Enter to complete the update. See Figure 1-14.

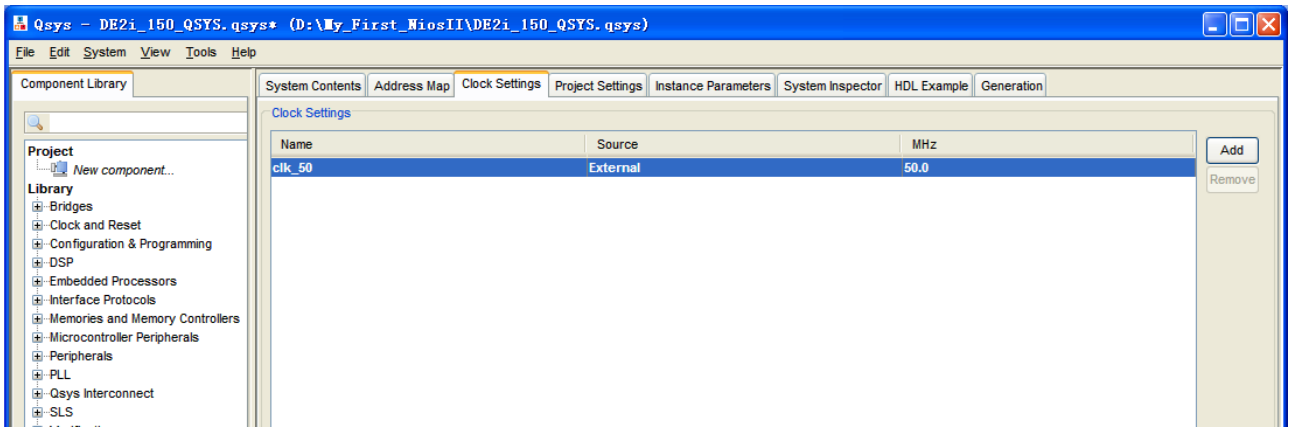

Figure 1-14 Rename Clock Name

9. Choose Library > Embedded Processors > Nios II Processor to open wizard of adding cpu component. See Figure 1-15 and Figure 1-16.

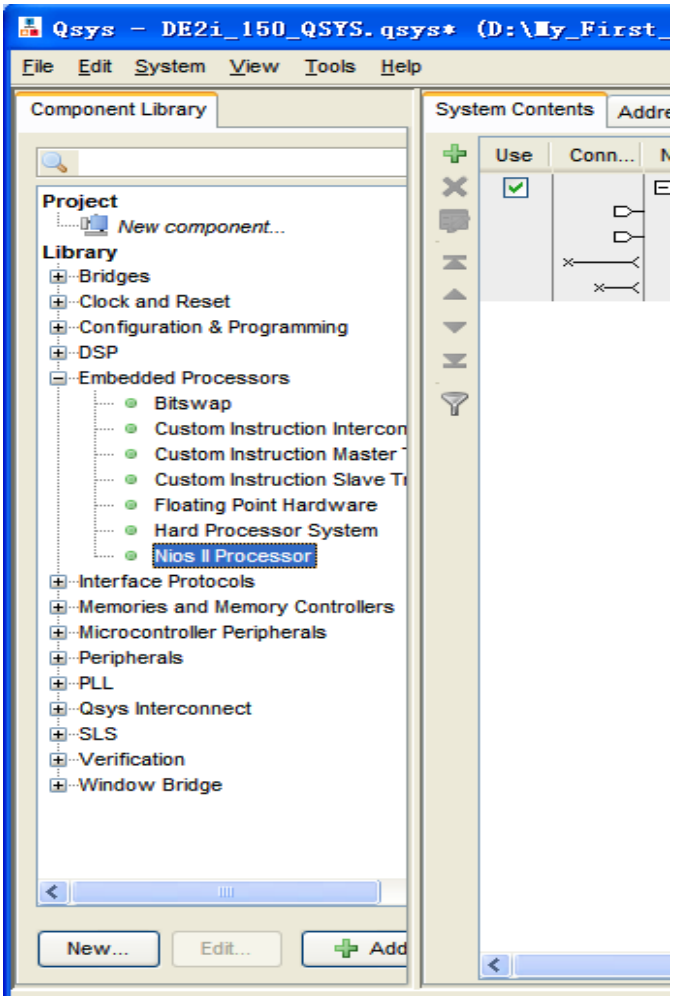

Figure 1-15 Add Nios II Processor

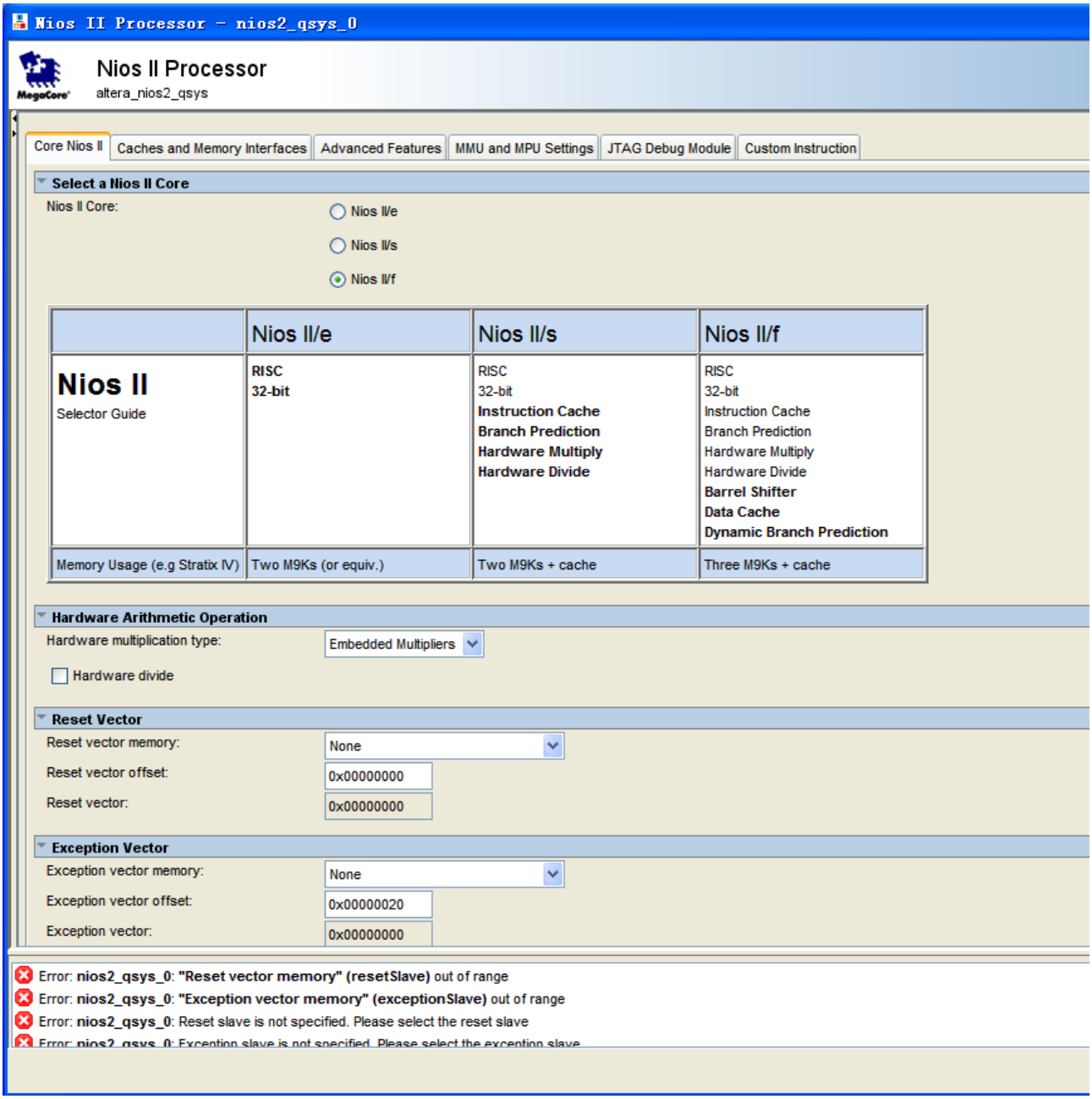

Figure 1-16 Nios II Processor

10. Click **Finish** to return to main window as shown in Figure 1-17.

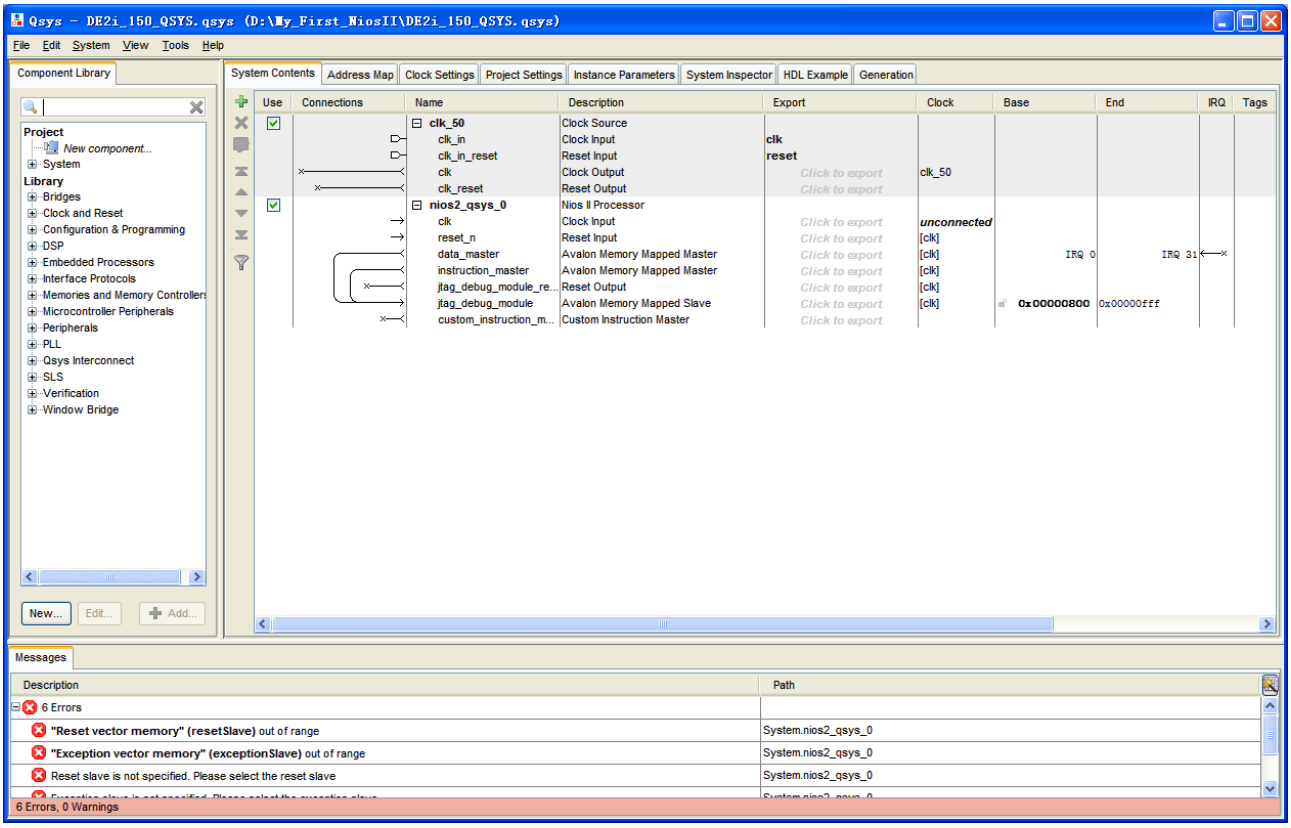

Figure 1-17 Add Nios II CPU completely

11. Choose **nios2_qsys_0** and right-click then choose **rename**, after this, you can update **nios2_qsys _0** to **nios2_qsys**. See Figure 1-18 and Figure 1-19.

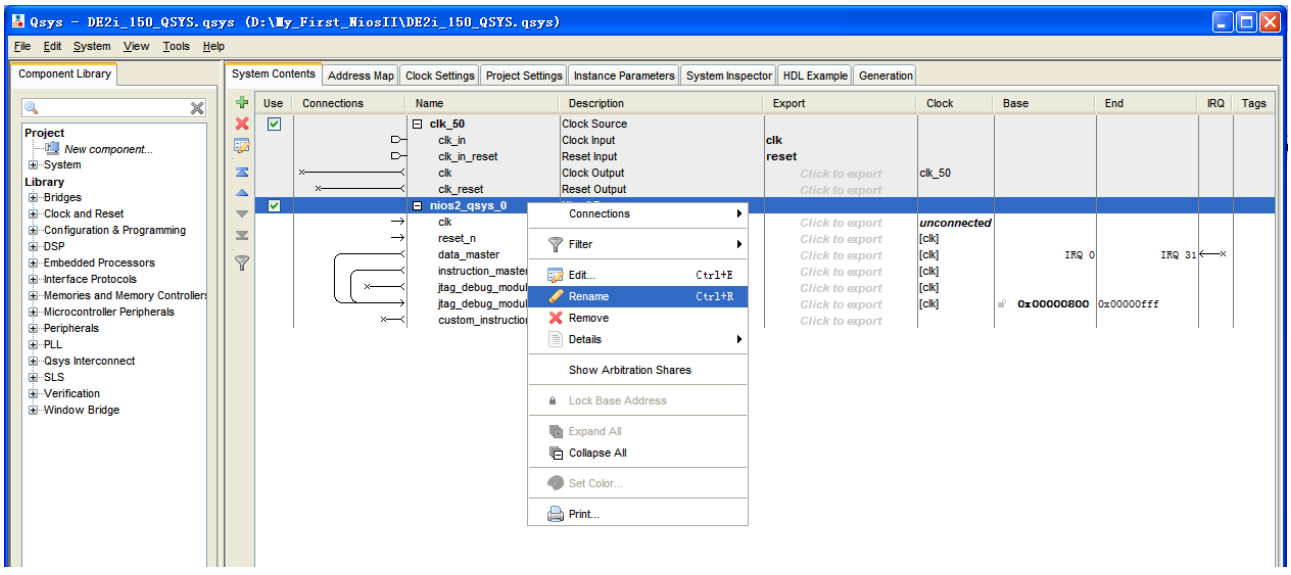

Figure 1-18 Rename CPU name (1)

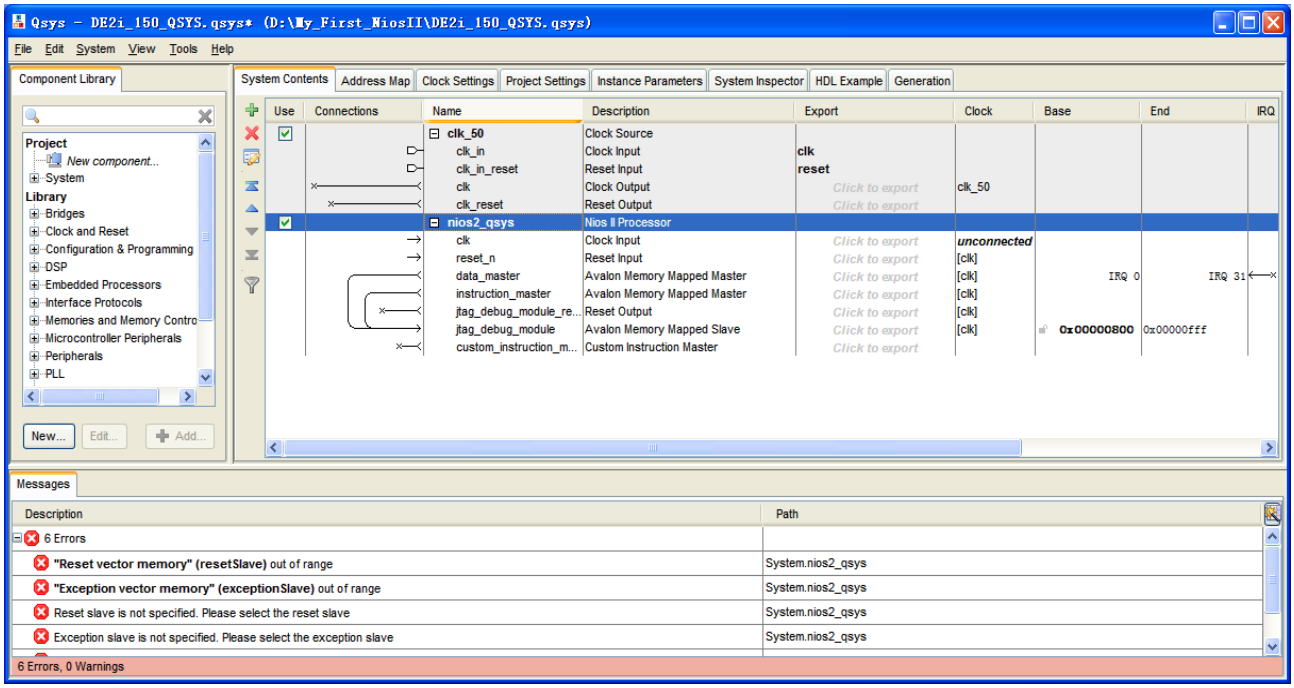

Figure 1-19 Rename CPU Name (2)

11. Connect the **clk** and **clk_reset** as shown in Figure 1-20. (clicking the hollow dots on the connection line. The dots become solid indicatingthe ports are connected.)

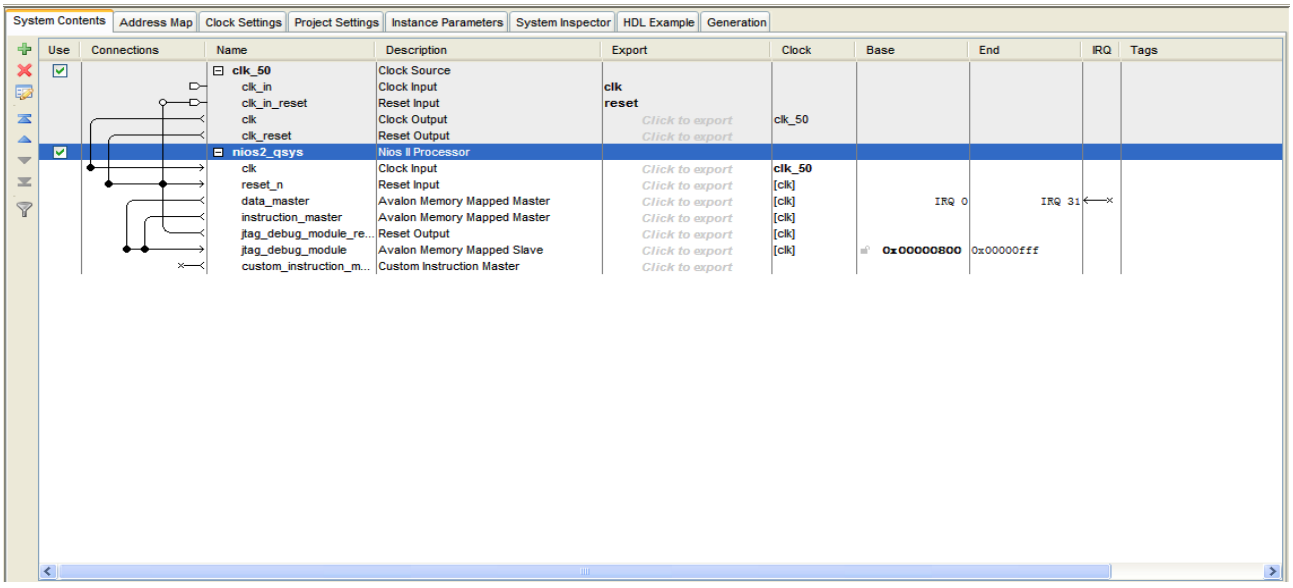

Figure 1-20 Connect the clk and clk_reset

12. Choose **Library** > **Interface Protocols** > **Serial** > **JTAG UART** to open wizard of adding **JTAG UART**. See Figure 1-21 and Figure 1-22.

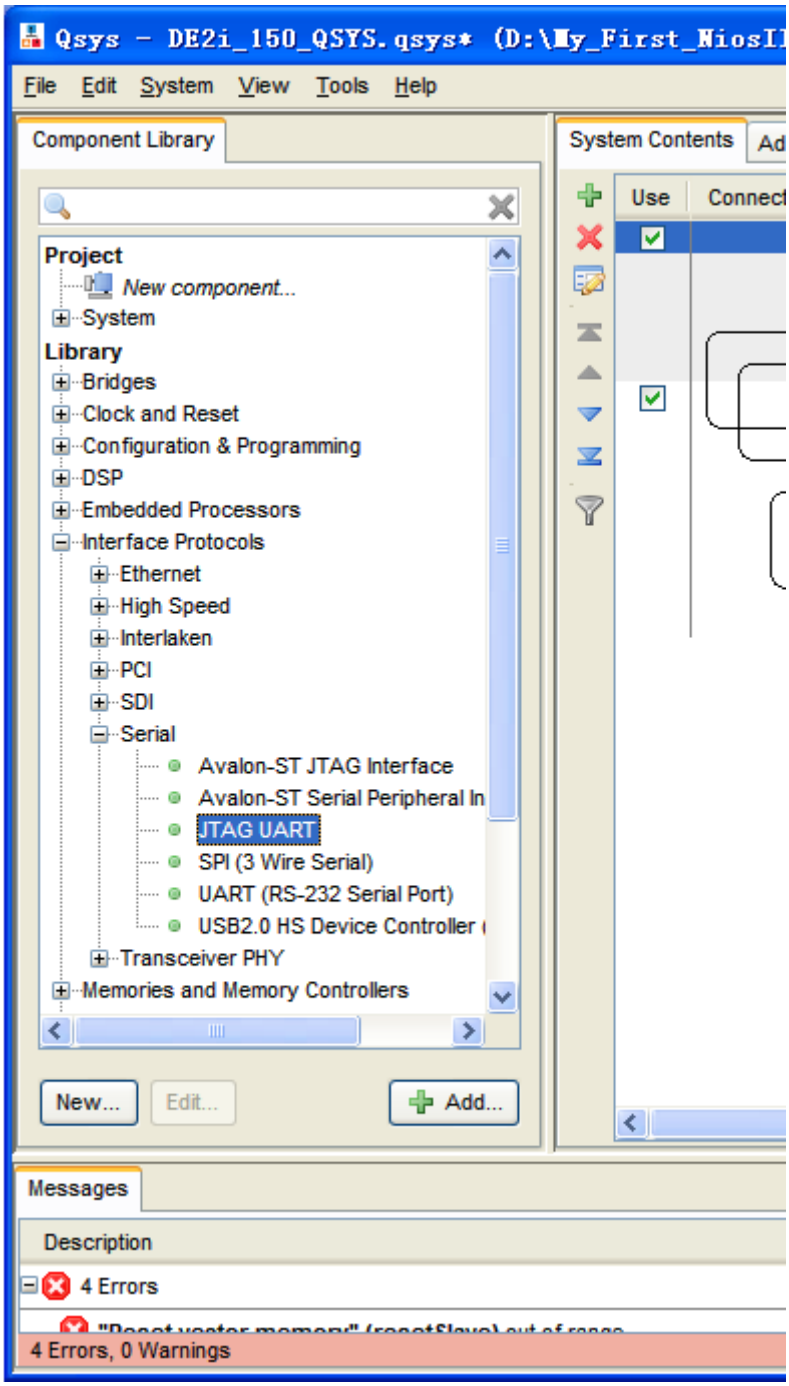

Figure 1-21 Add JTAG UART (1)

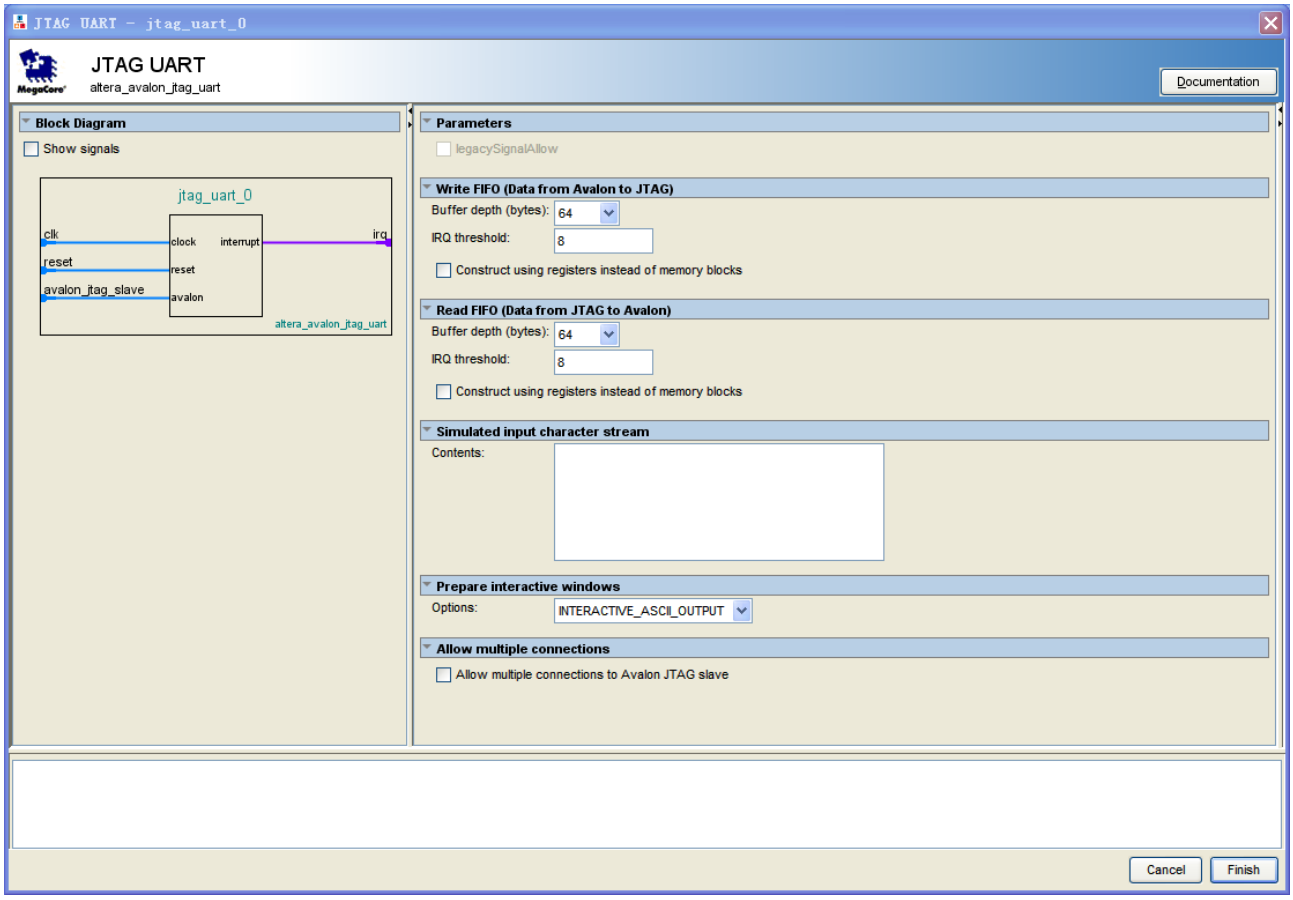

Figure 1-22 JTAG UART (2)

13. Click **Finish** to close the wizard and return to the window as shown in Figure 1-23.

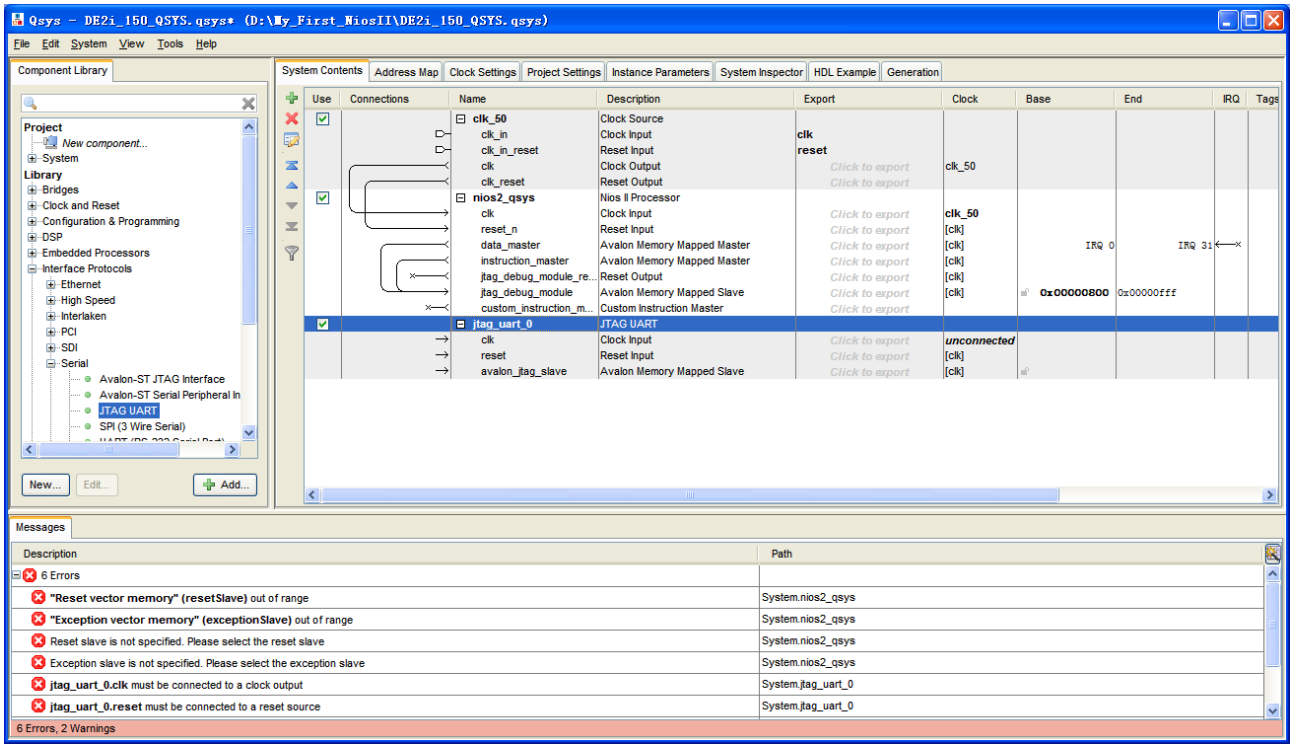

Figure 1-23 JTAG UART

14. Choose **jtag_uart_0** and rename it to **jtag_uart** as shown in Figure 1-24.

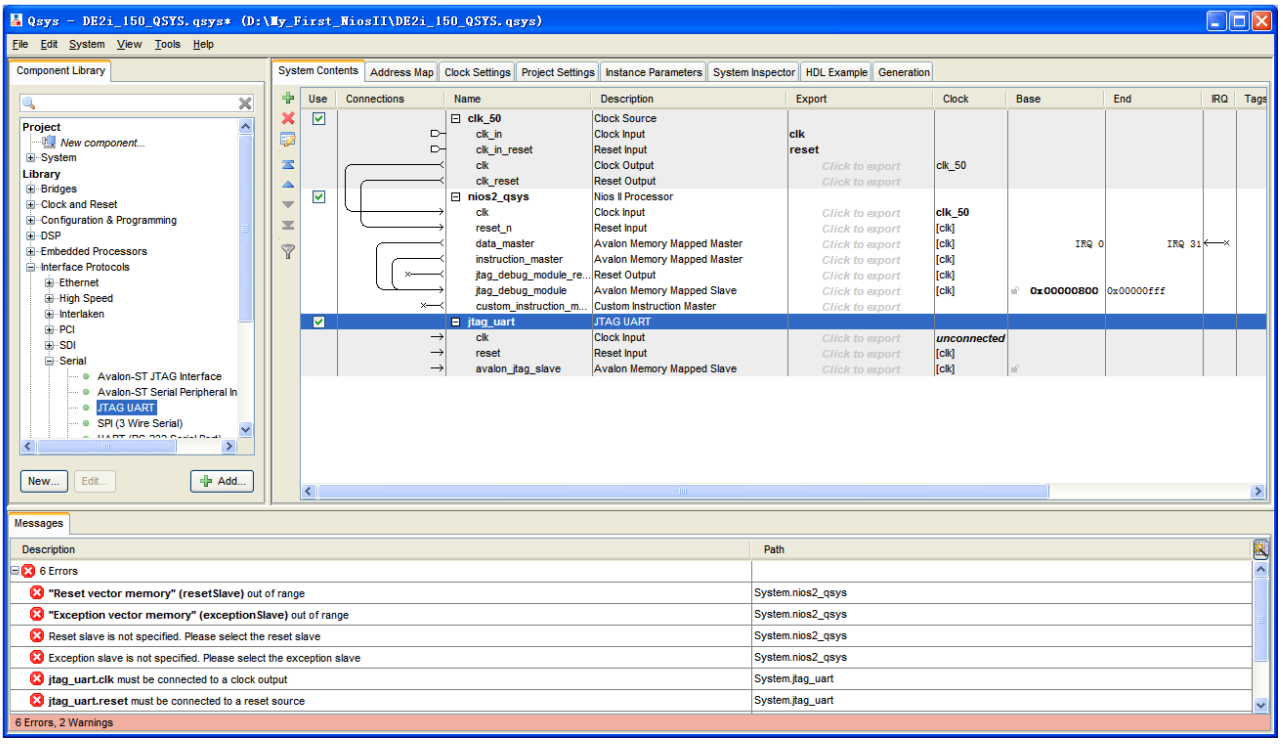

Figure 1-24 Rename JTAG UAR

15. Connect the **clk** and **clk_reset** and **data_master** as shown in Figure 1-25.

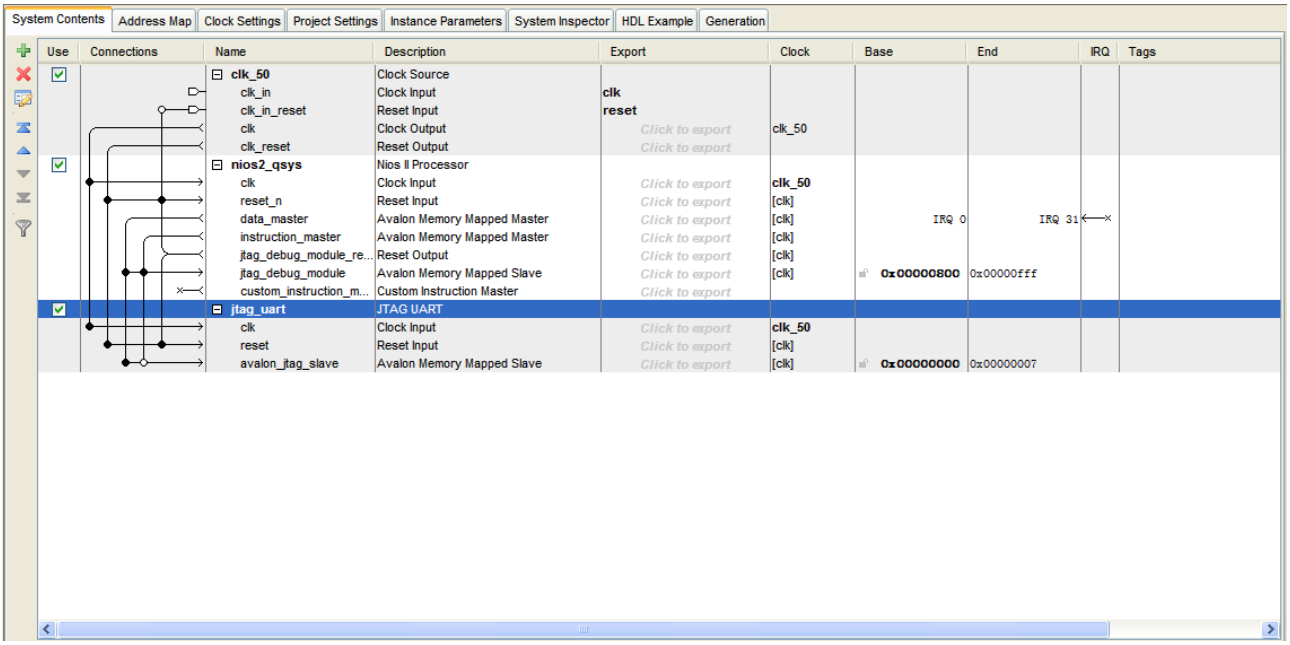

Figure 1-25 Connect JTAG UART

16. Choose **Library** > **Memories and Memory Controllers** > **On-Chip** > **On-Chip Memory (RAM or ROM)** to open wizard of adding On-Chip memory. See Figure 1-26 and Figure 1-27.

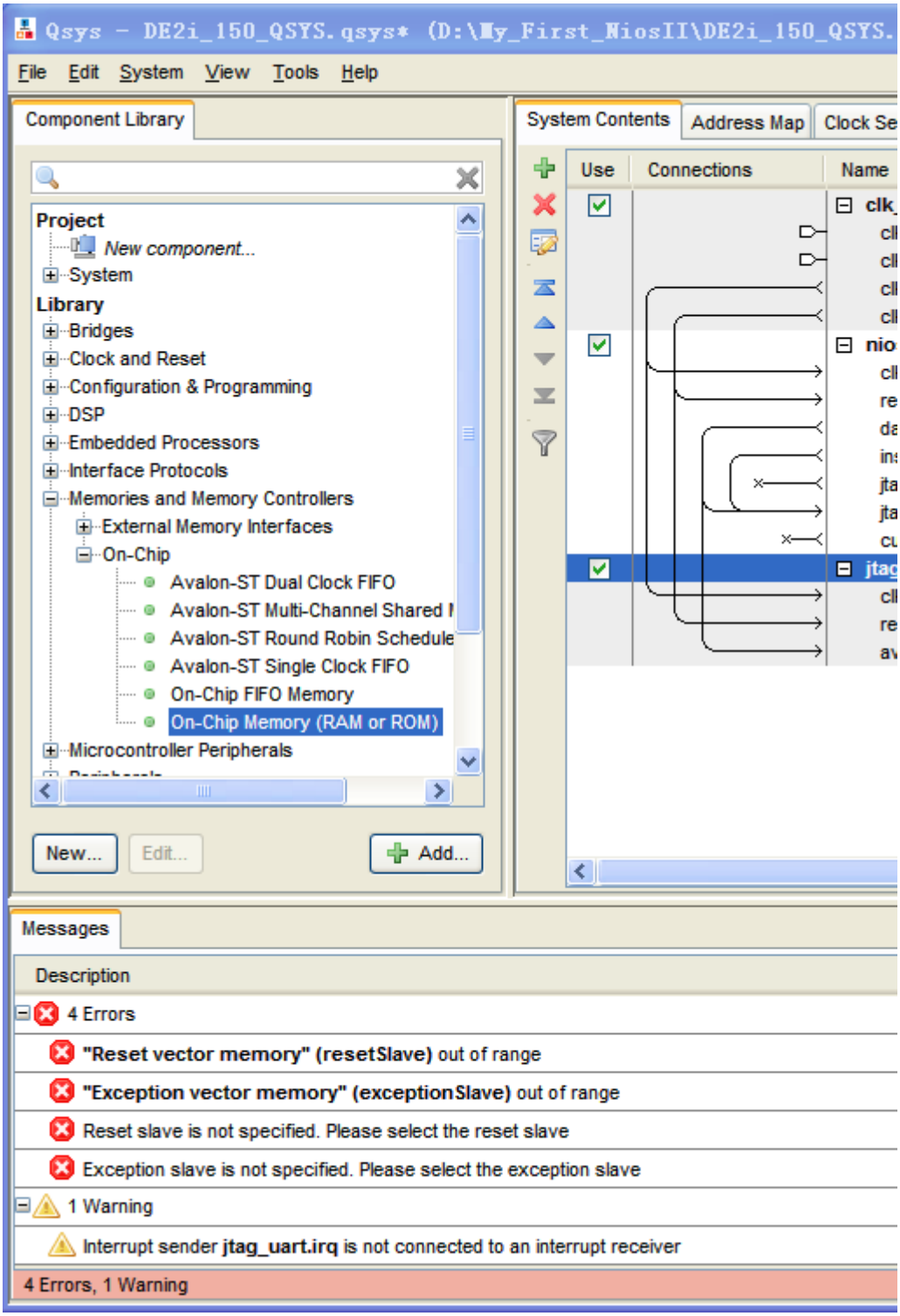

Figure 1-26 Add On-Chip Memory

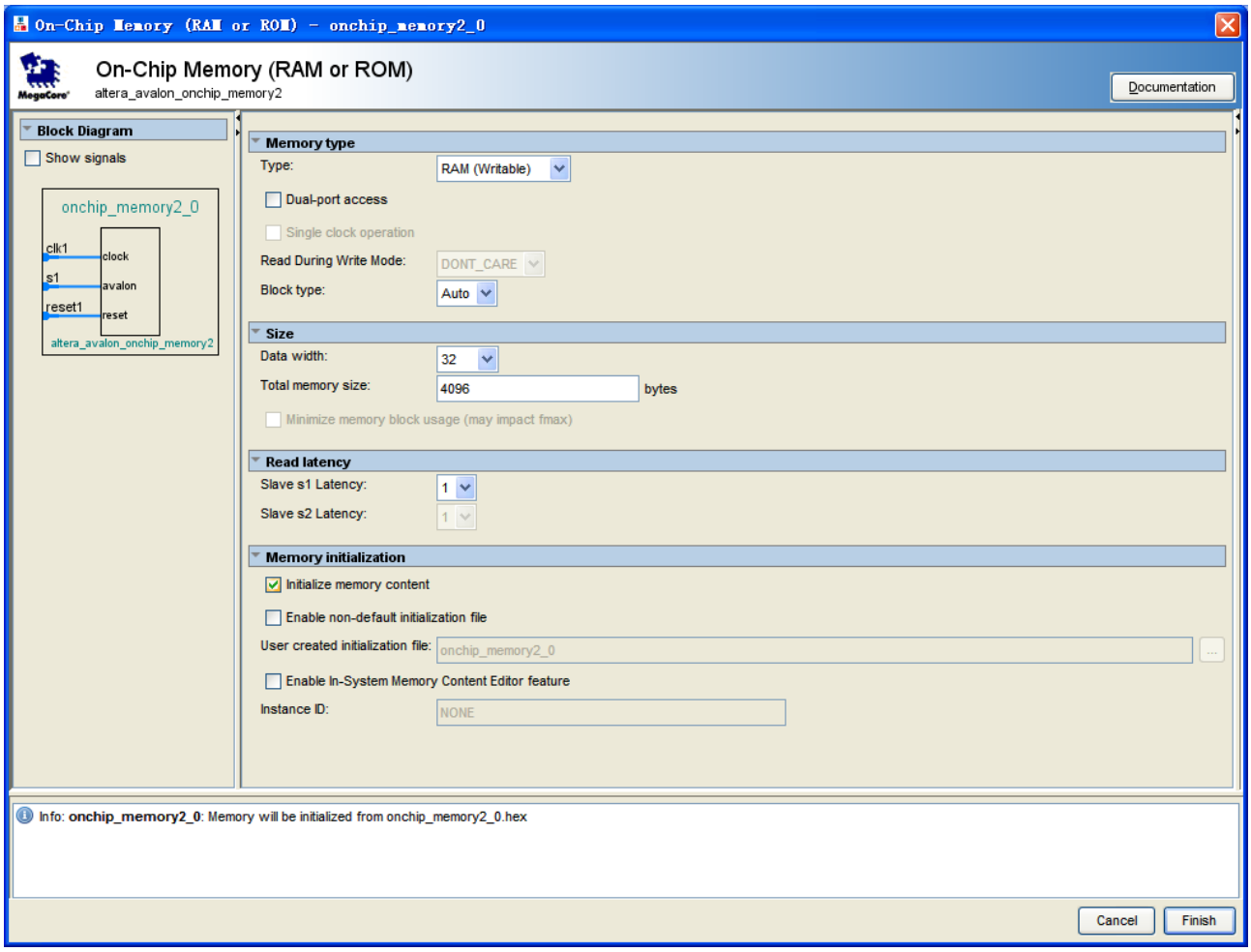

Figure 1-27 On-Chip Memory Box

17. Modify **Total memory size** to **204800** as shown in Figure 1-28. Click **Finish** to return to the window as in Figure 1-29.

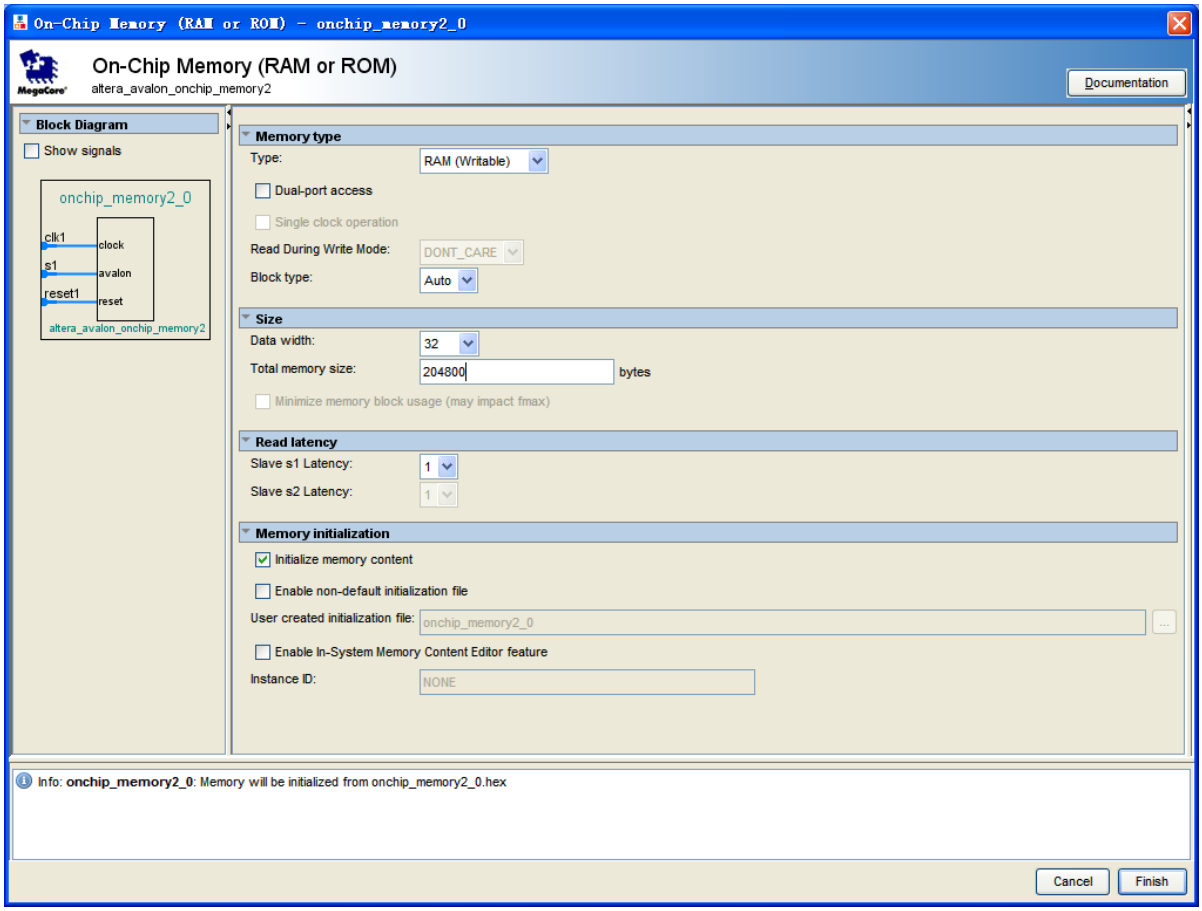

Figure 1-28 Update Total memory size

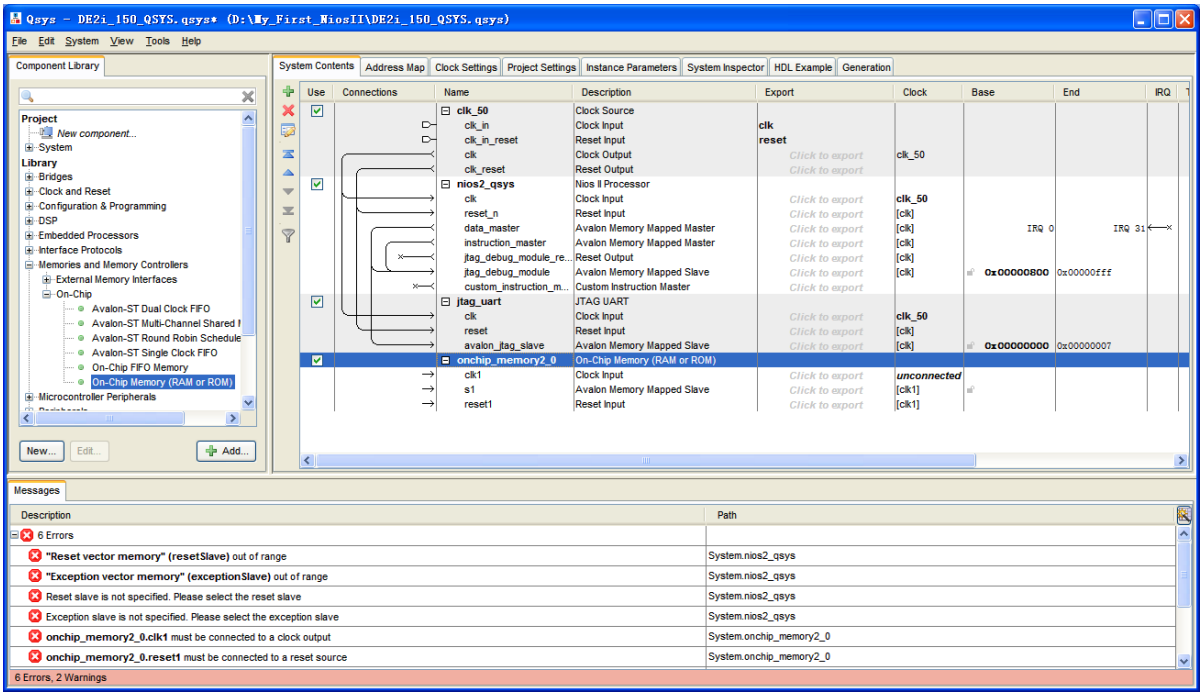

Figure 1-29Add On-Chip memory Completely

18. Rename **onchip_memory2_0** to **onchip_memory2** as shown in Figure 1-30.

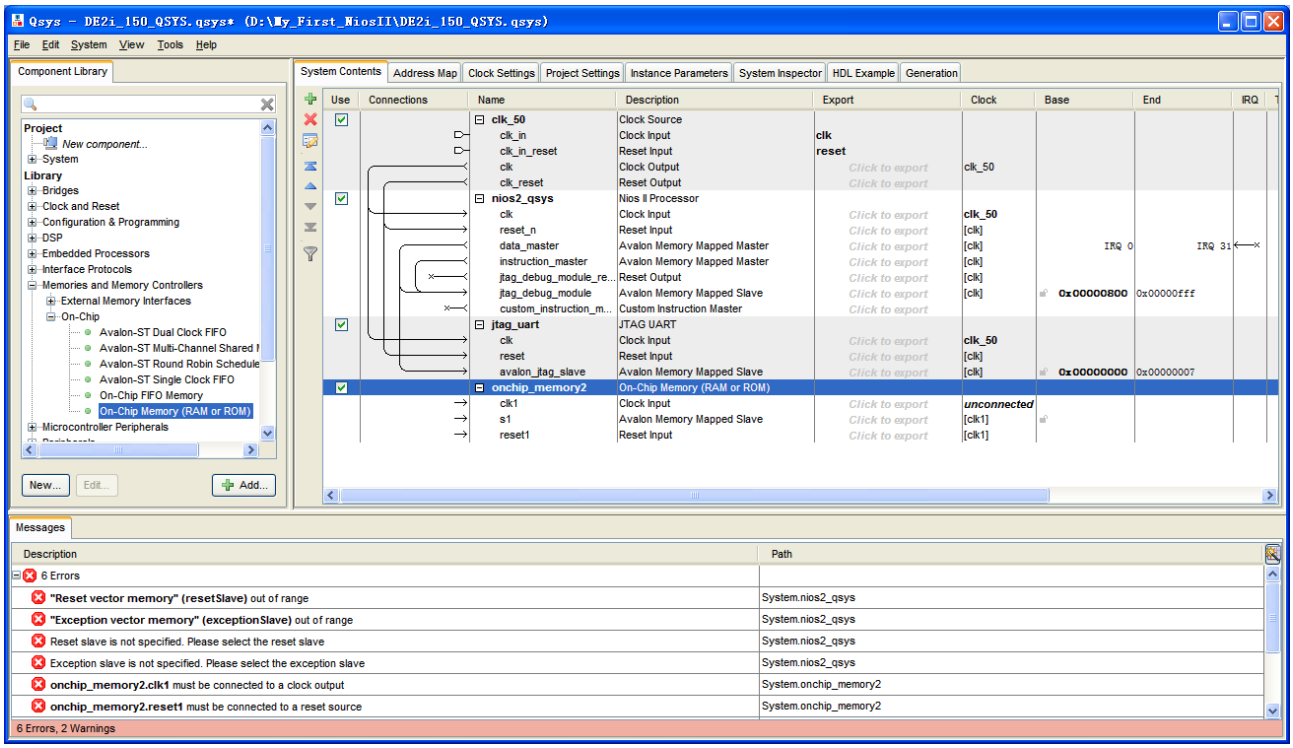

Figure 1-30 Rename On-Chip memory

19. Connect the **clk** and **clk_reset** and **data_master** as shown in Figure 1-31.

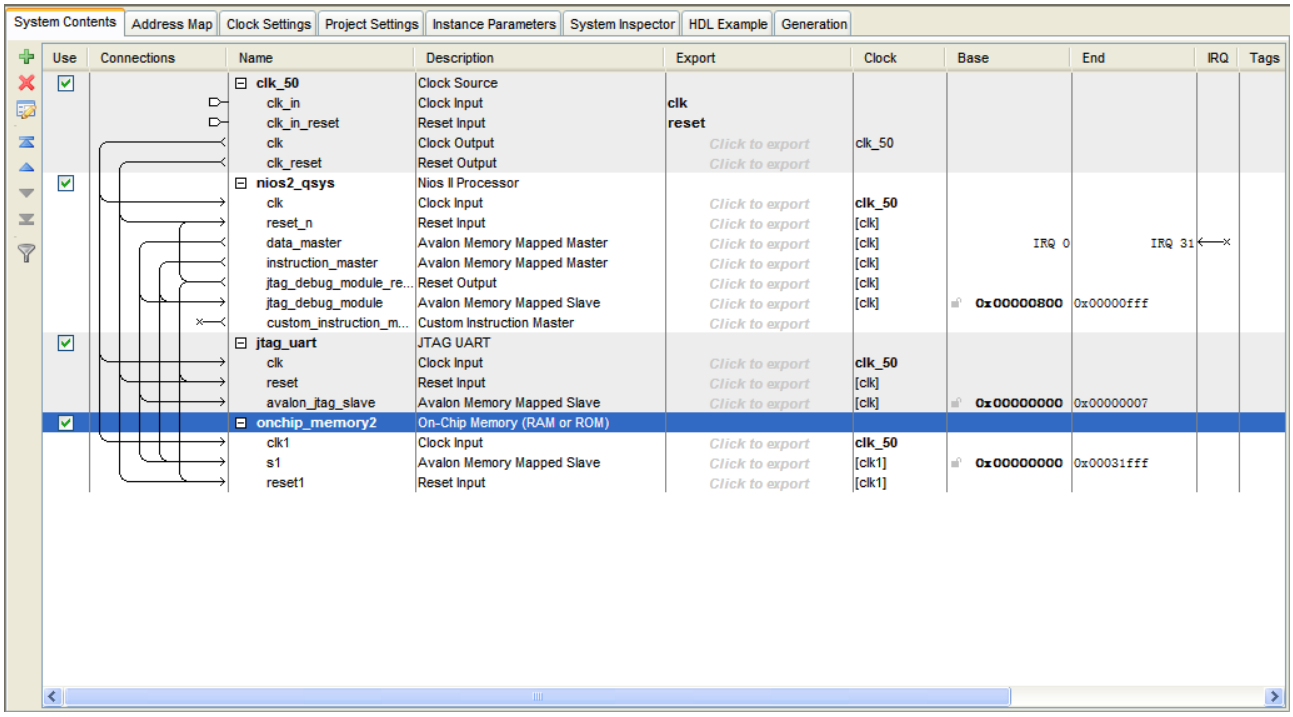

Figure 1-31 Connect On-Chip memory

20. Click **nios2_qsys** in the component list on the right part to edit the component. Update **Reset vector** and **Exception Vector** as shown in Figure 1-32. Then click **Finish** to return to the window as shown Figure 1-33.

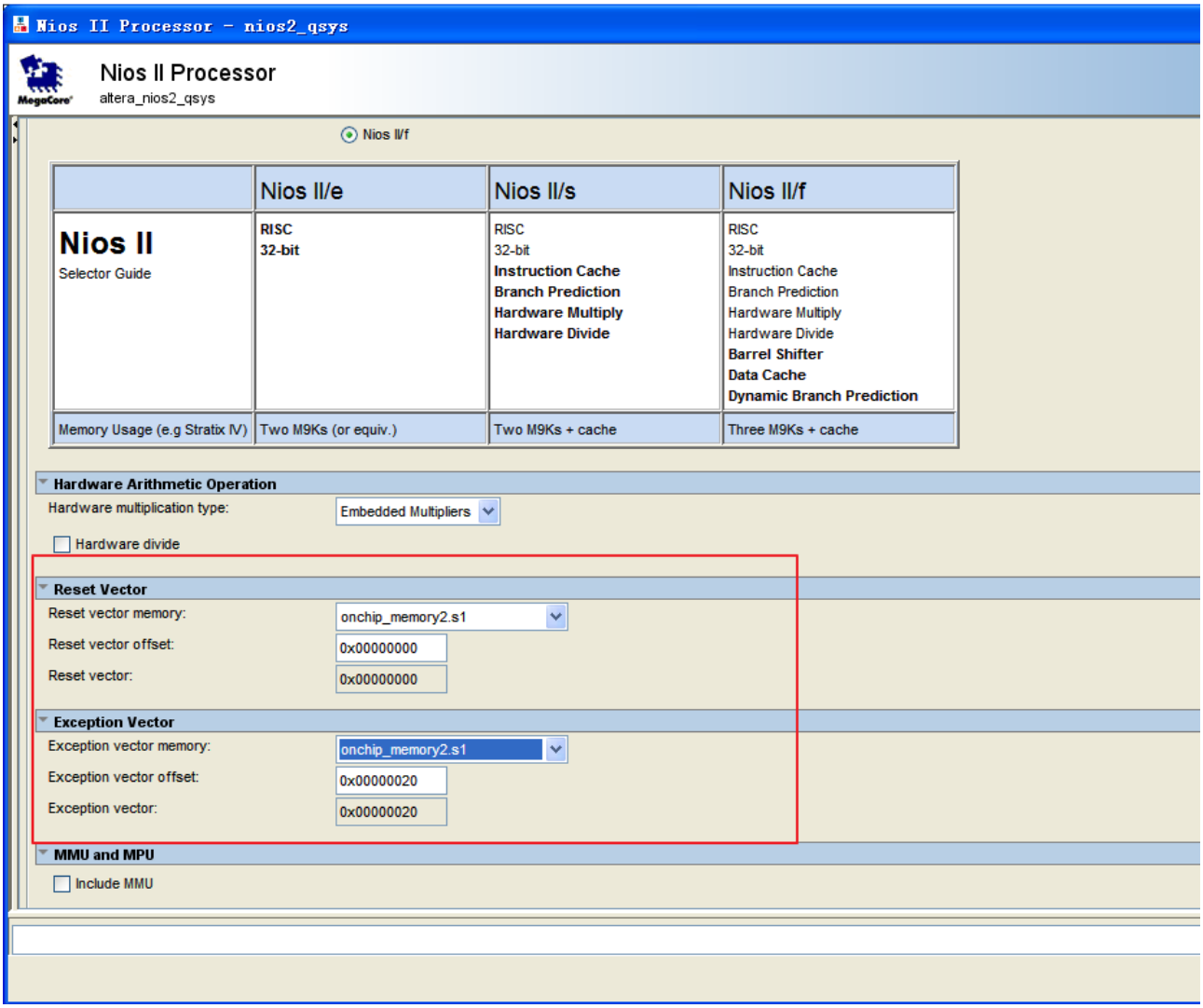

Figure 1-32 Update CPU settings

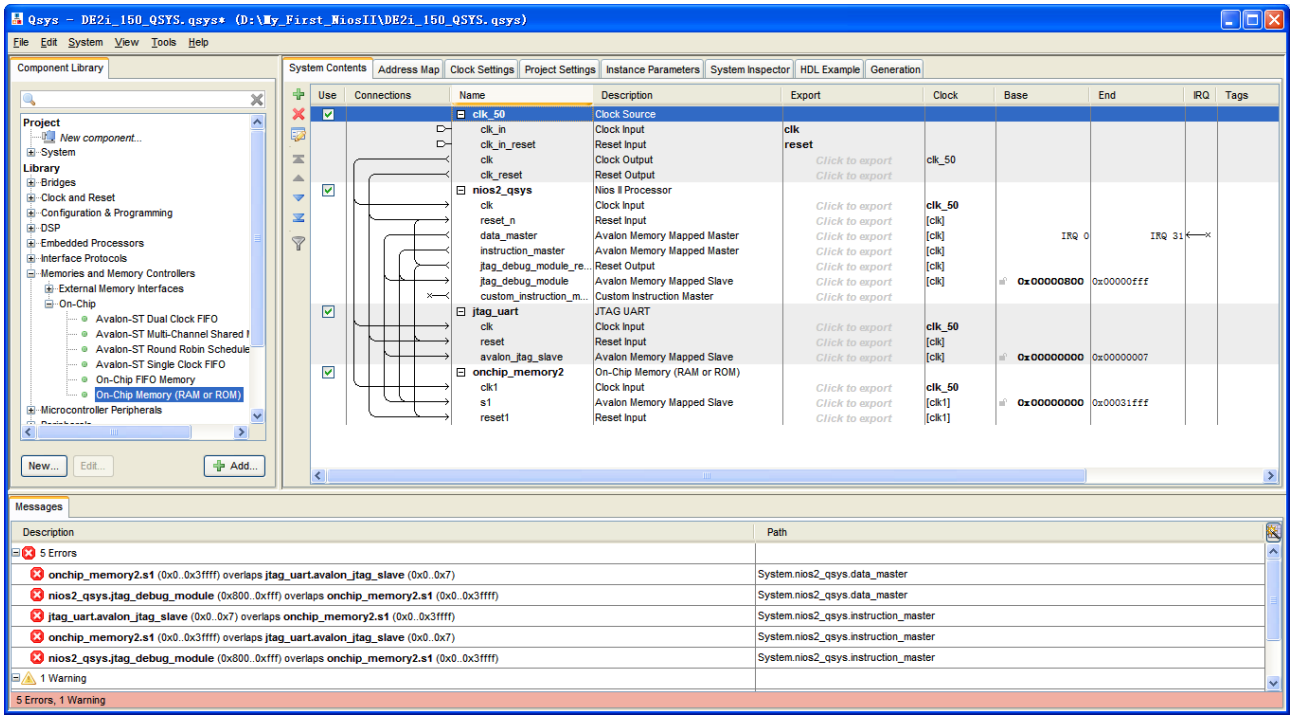

Figure 1-33 Update CPU settings Completely

21. Choose **Library** > **Peripherals** > **Debug and Performance** >**System ID Peripheral** to open wizard of adding **System ID**. See Figure 1-34 and Figure 1-35.

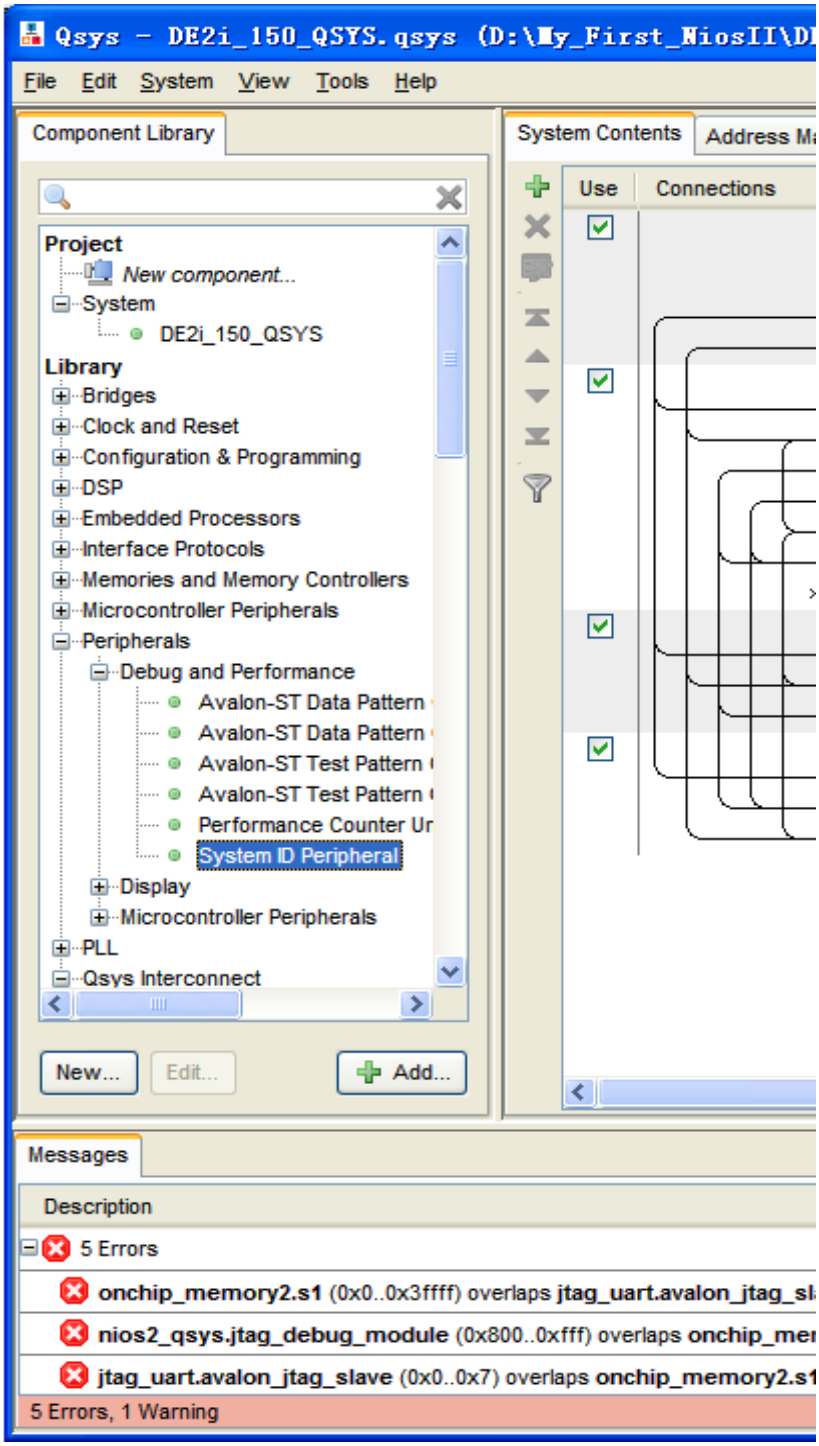

Figure 1-34 Add System ID [0]

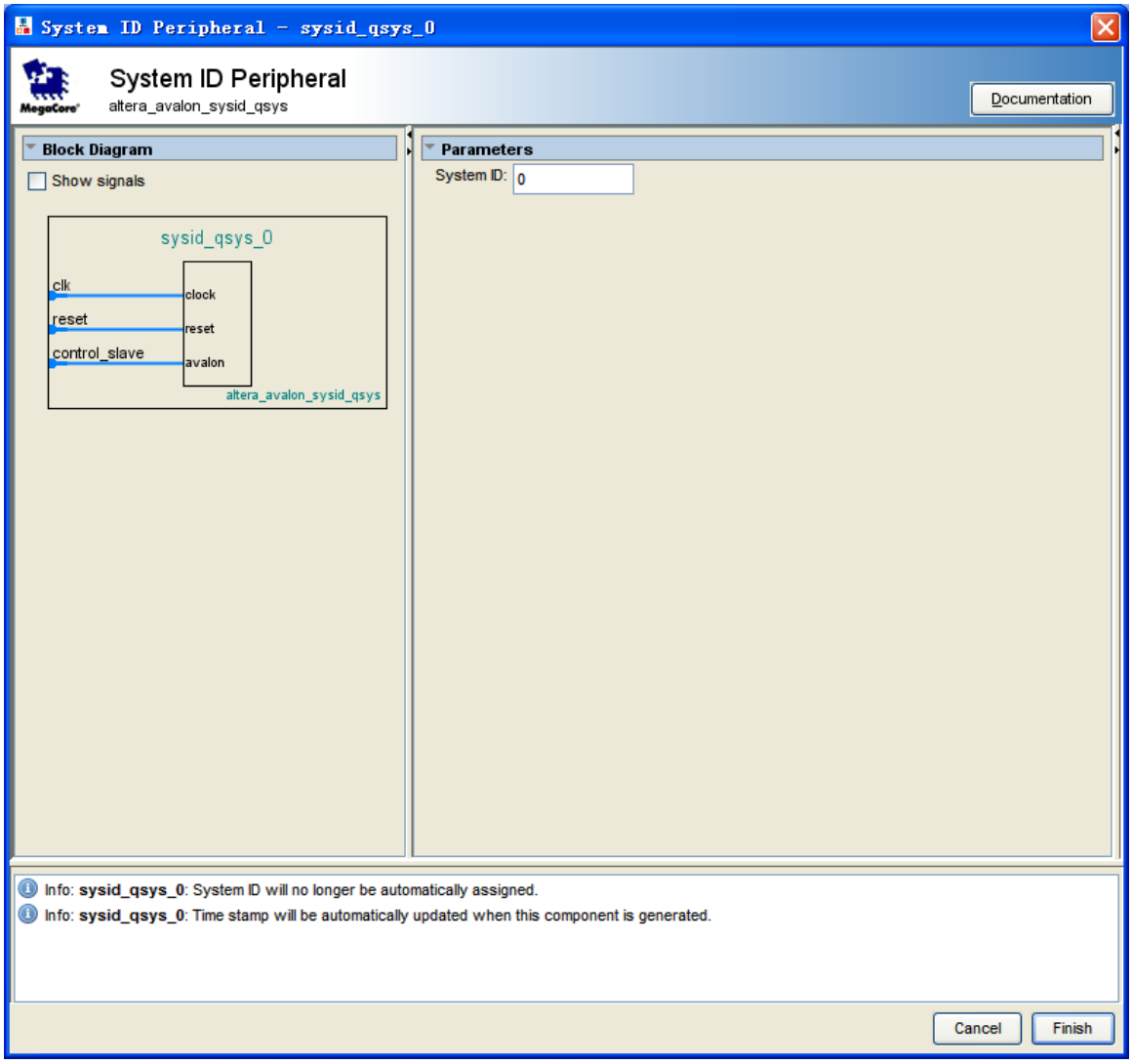

Figure 1-35 Add System ID [1]

22. Click **Finish** to close System ID Peripheral box and return to the window, rename **sysid_qsys_0** to **sysid_qsys** and connect the **clk** and **clk_reset** and **data_master** as shown in Figure 1-36.

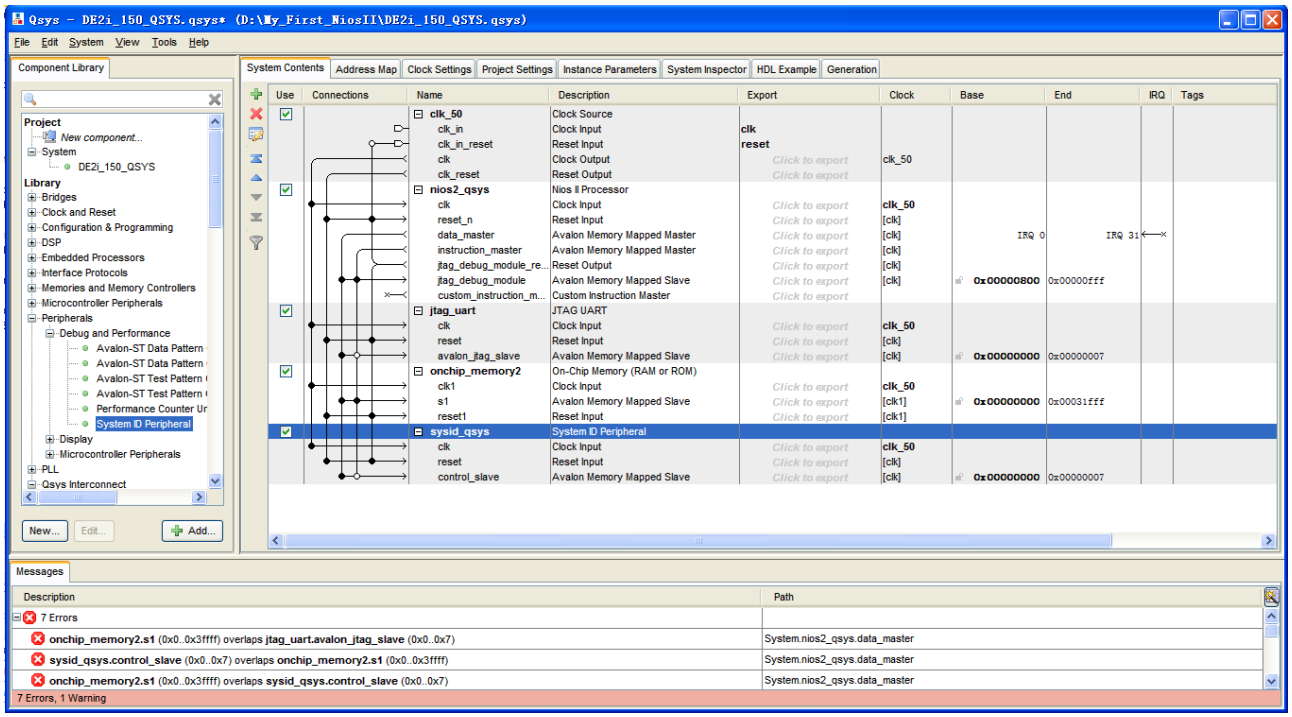

Figure 1-36 Add System ID [2]

23. Choose **Library** > **Peripherals** > **Microcontroller Peripherals** >**PIO (Parallel I/O)** to open wizard of adding PIO. See Figure 1-37 and Figure 1-38.

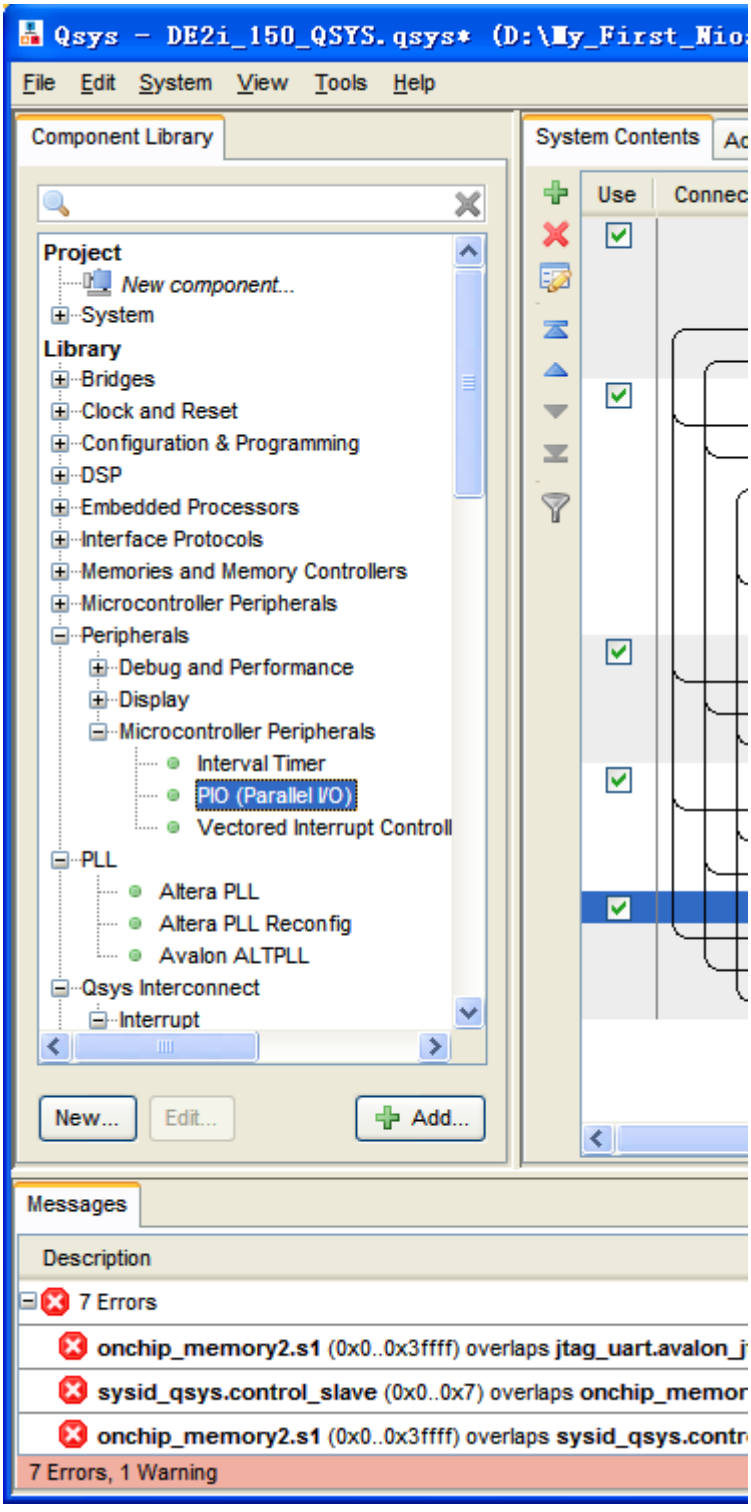

Figure 1-37 Add PIO

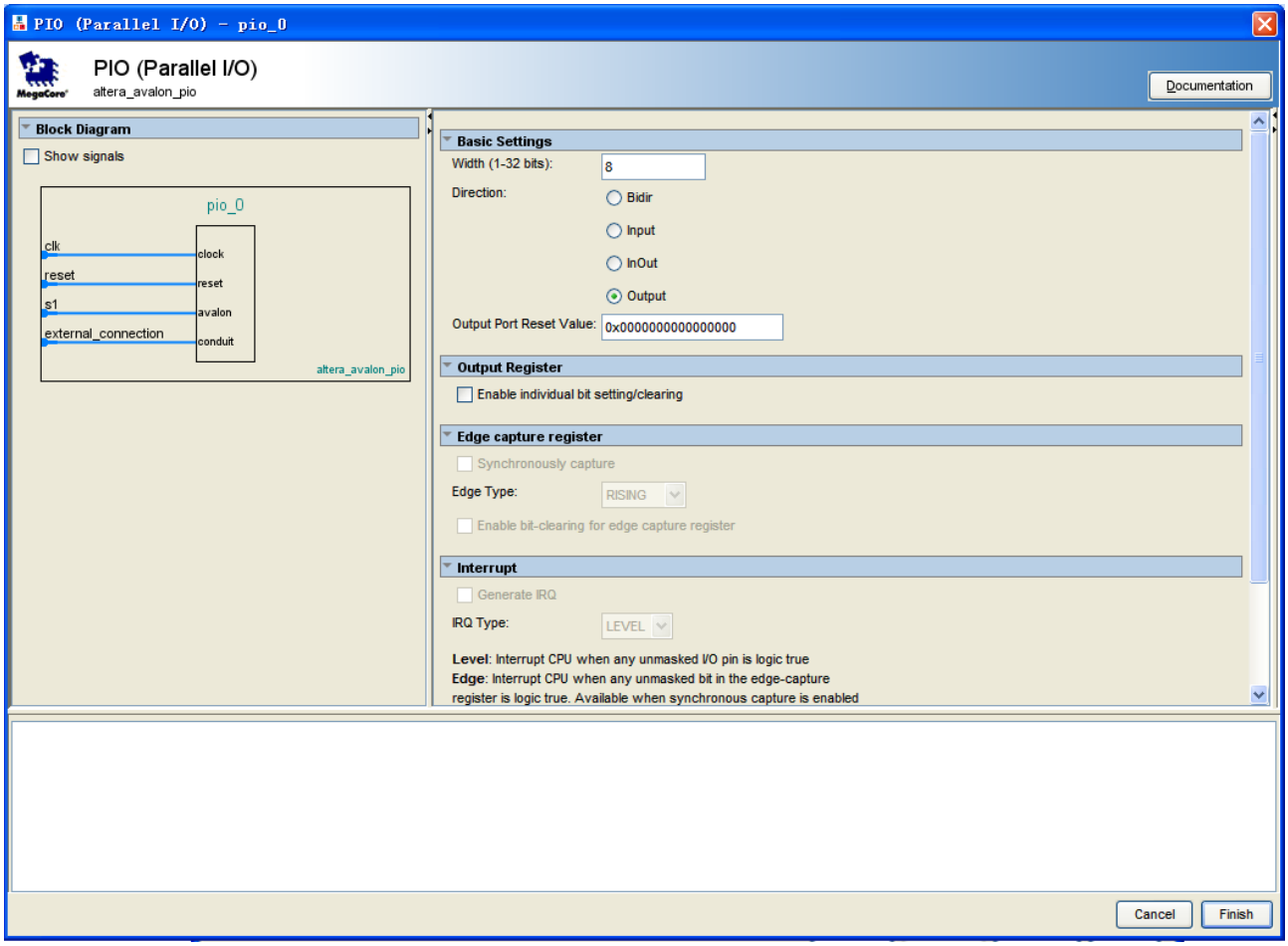

Figure 1-38 Add PIO

24. Click **Finish** to close PIO box and return to the window as shown in Figure 1-39.

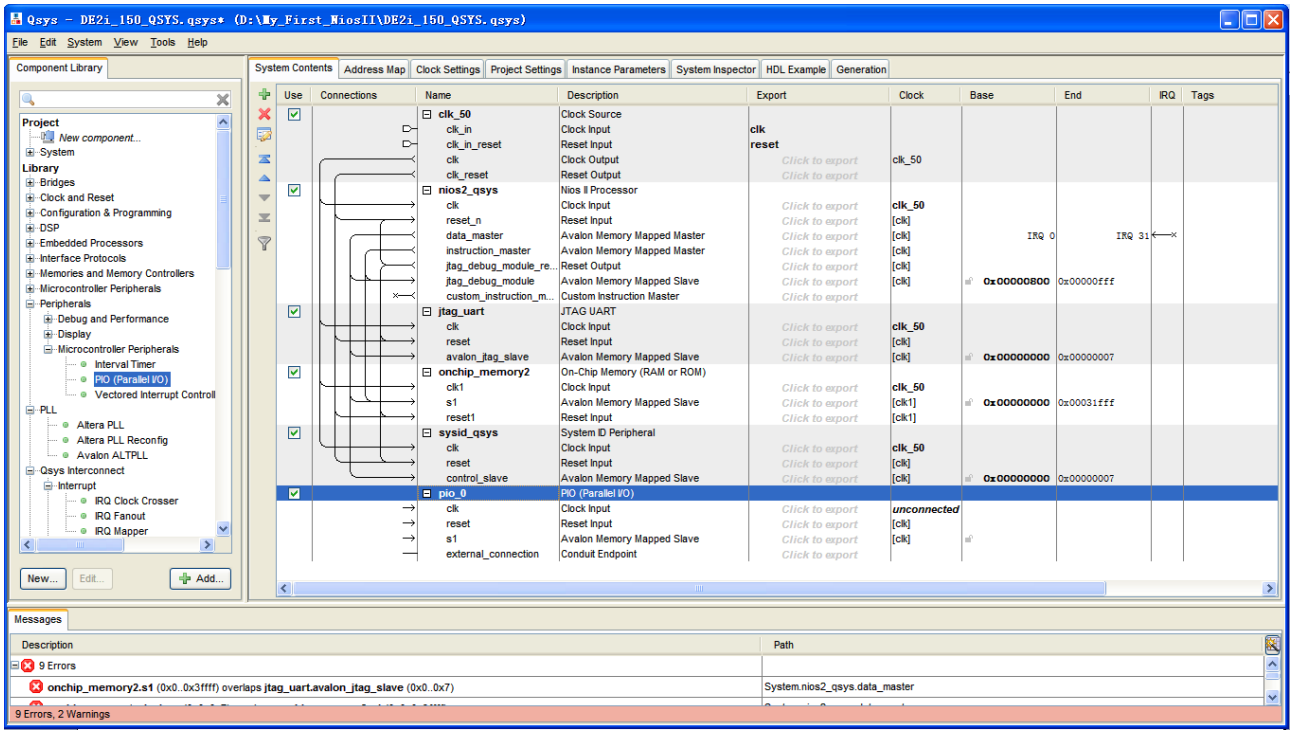

Figure 1-39 PIO

25. Rename **pio_0** to **led** as shown in Figure 1-40.

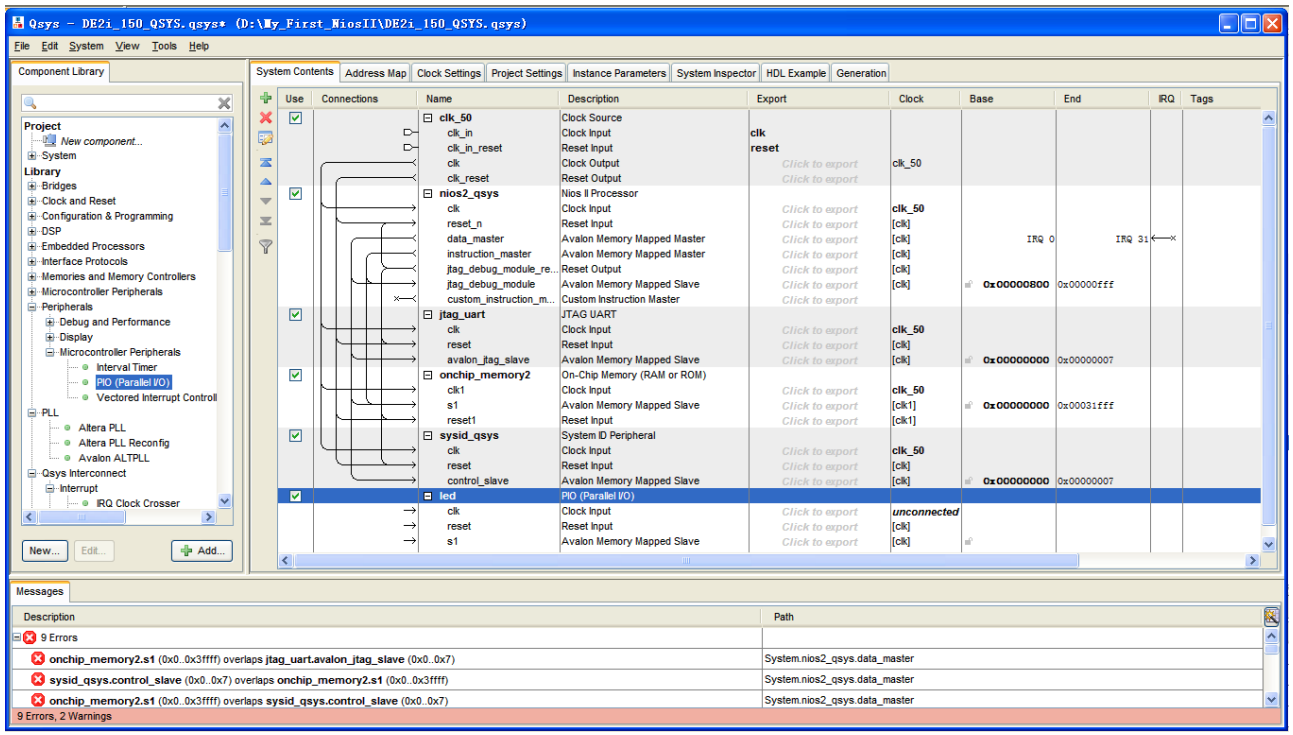

Figure 1-40 Rename PIO

26. Connect the **clk** and **clk_reset** and **data_master** as shown in Figure 1-41.

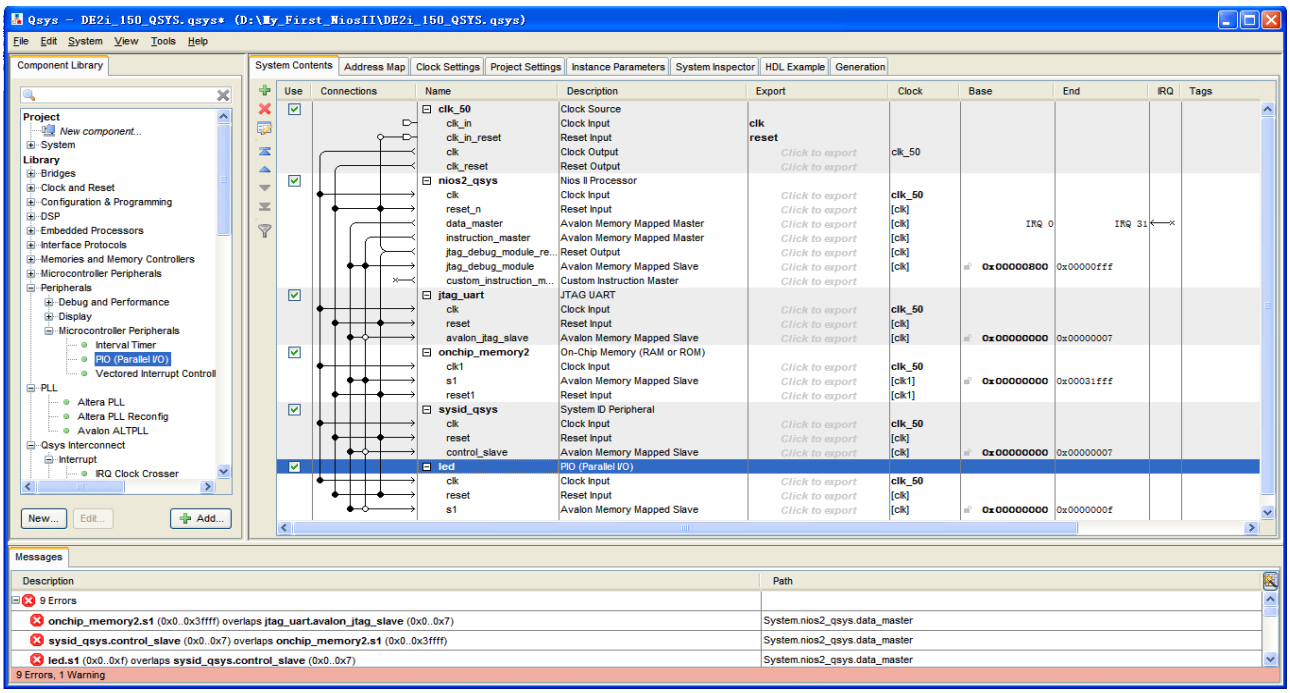

Figure 1-41 Connect PIO

27. Export **external_connection** and Rename it to **led** as shown in Figure 1-42.

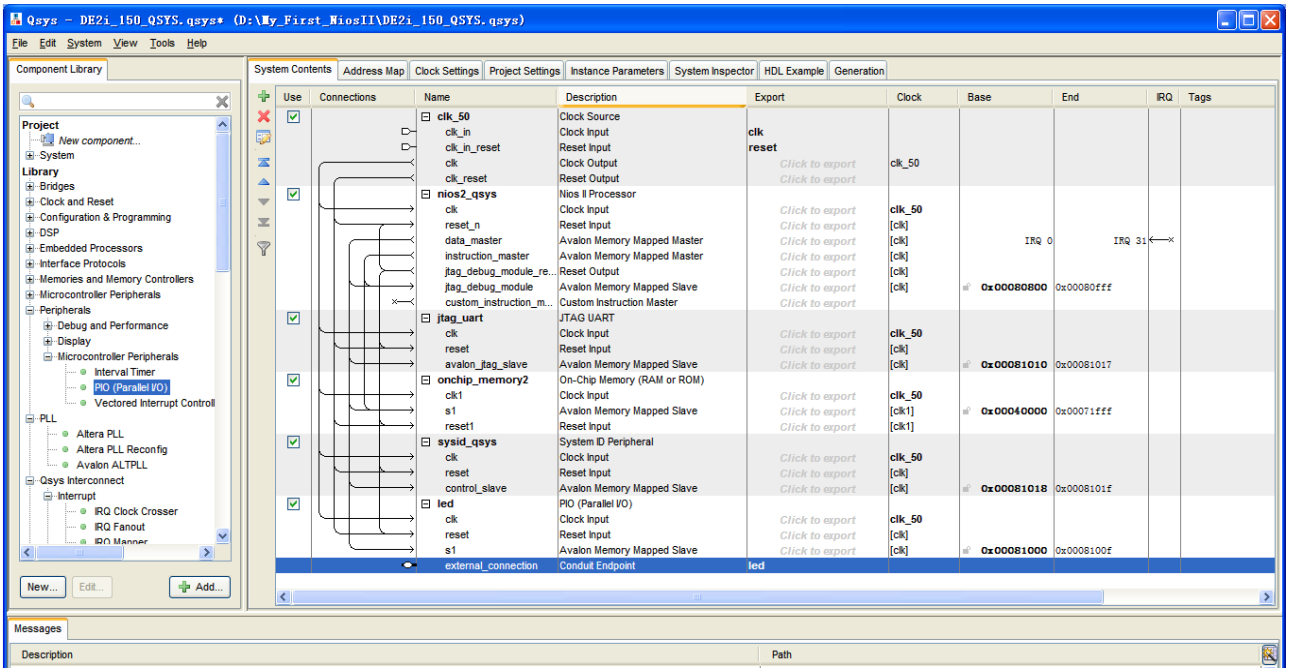

Figure 1-42 Export external_connection

28. Choose **System** > **Assign Base Addresses** as shown in Figure 1-43. After that, you will find that there is no error in the message window as shown in Figure 1-44.

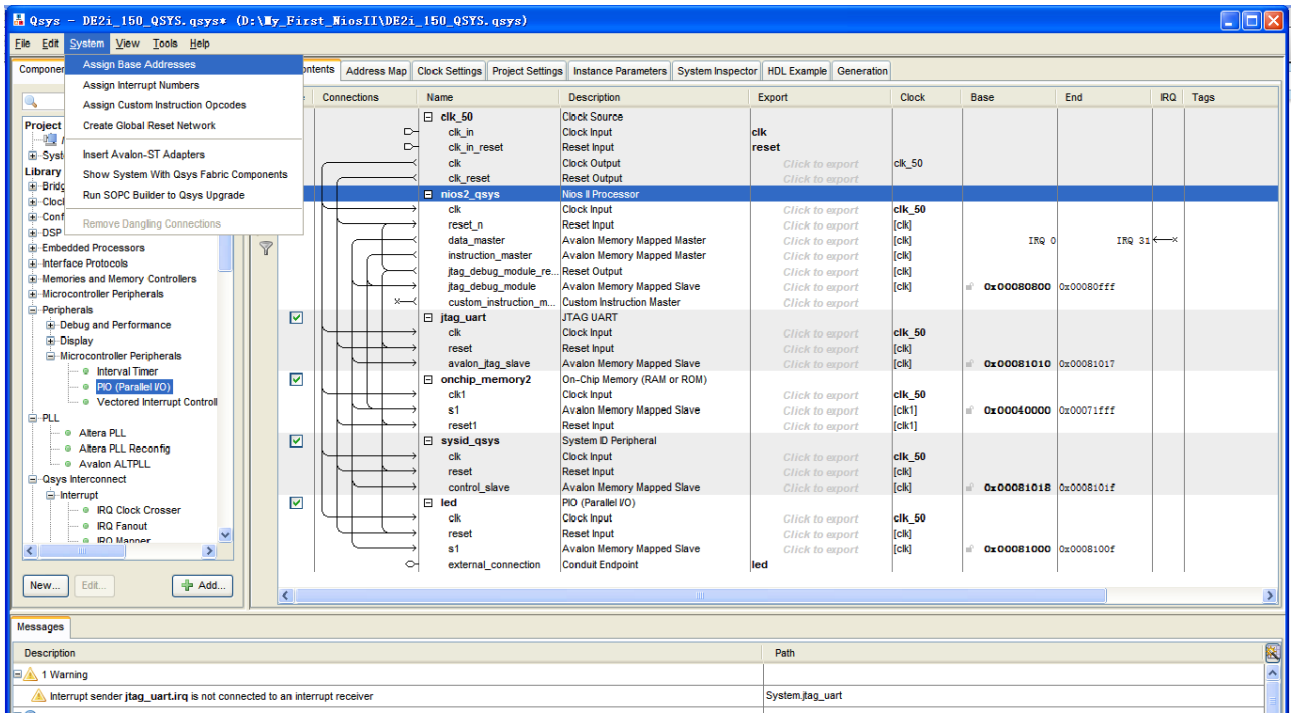

Figure 1-43 Assign Base Addresses

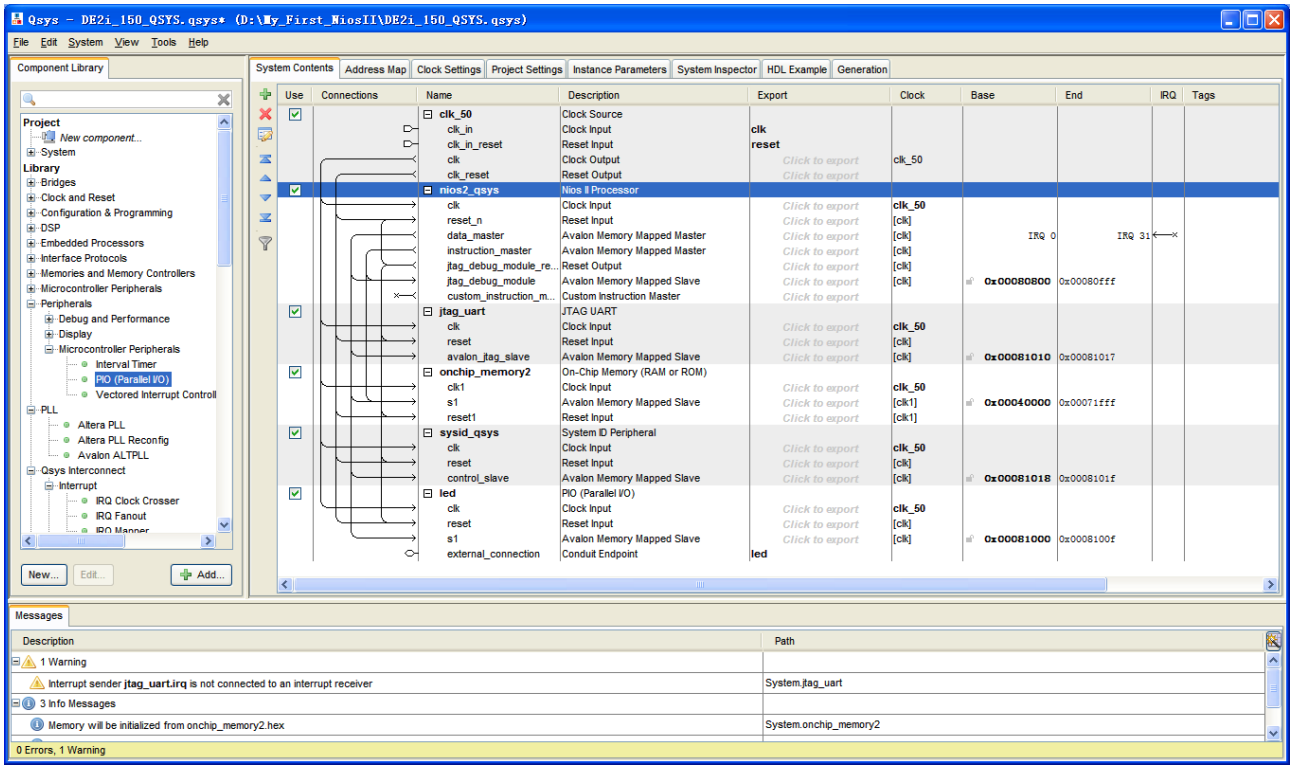

Figure 1-44 No Errors

29. Assign **Interrupt Numbers** as shown in Figure 1-45. After that, you will find that there is no warings in the message window as shown in Figure 1-46.(In the **IRQ** column, connect the Nios II processor to the JTAG UART)

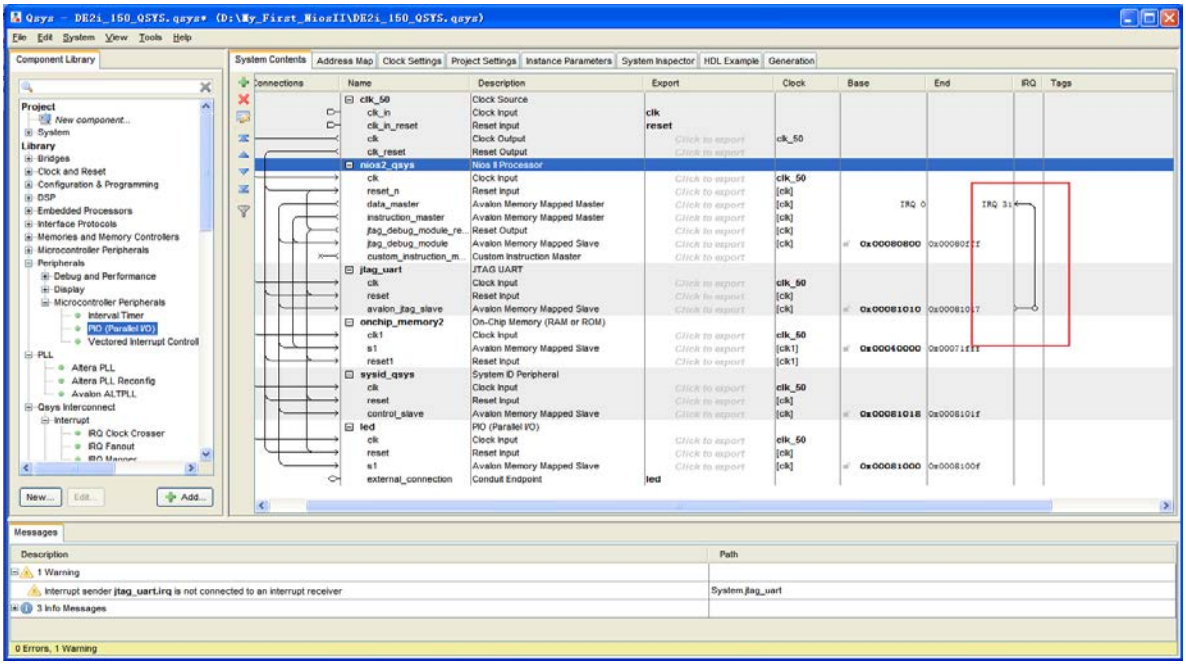

Figure 1-45 Assign IRQ

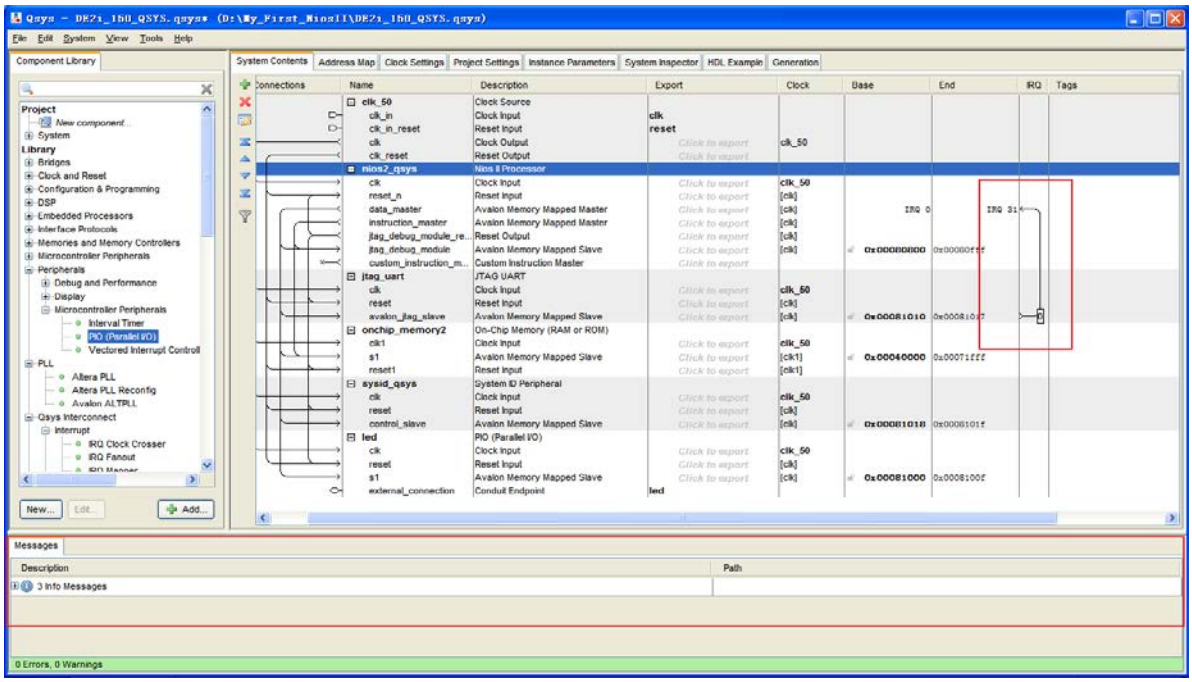

Figure 1-46 No Warings

30. Click **Generate tab** and click **Generate** then pop a window as shown in Figure 1-47. Click **Save** and the generation start. Figure 1-48 shows the generate process. If there is no error in the generation, the window will show successful as shown in Figure 1-49.

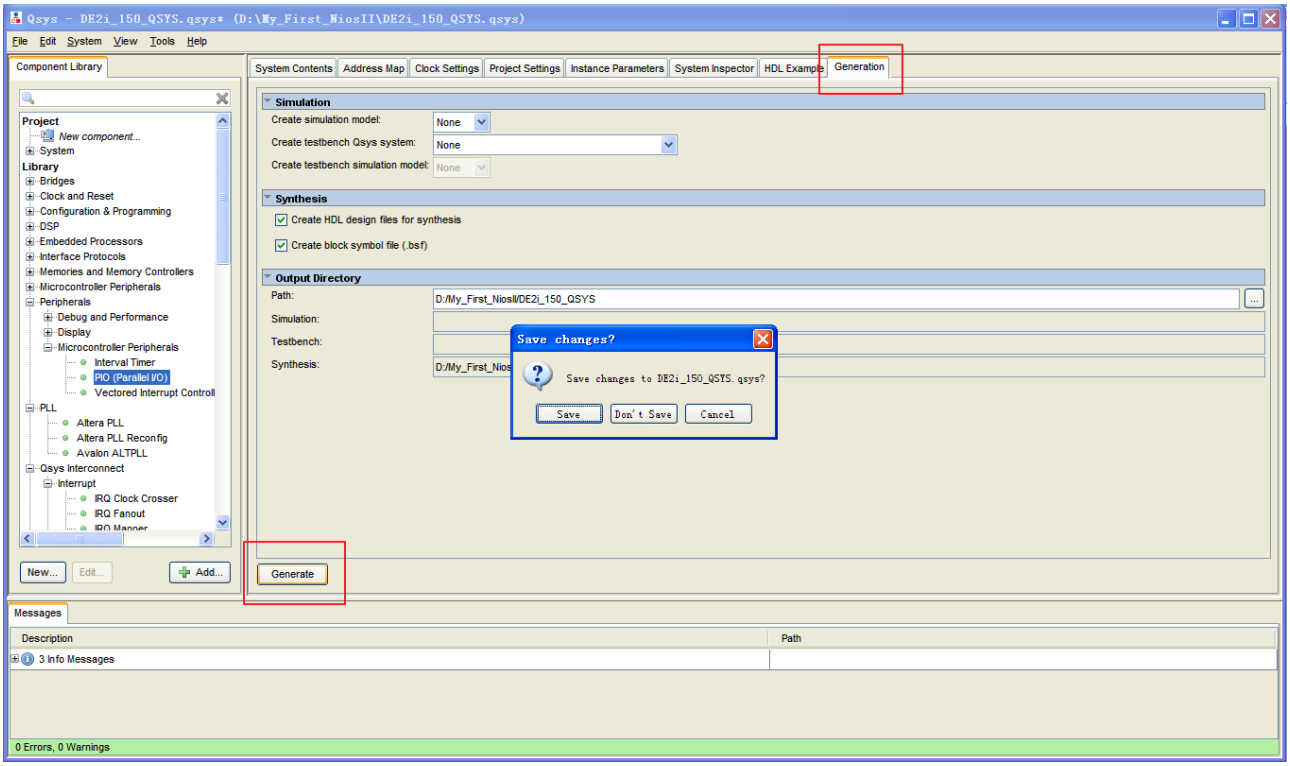

Figure 1-47 Generate Qsys

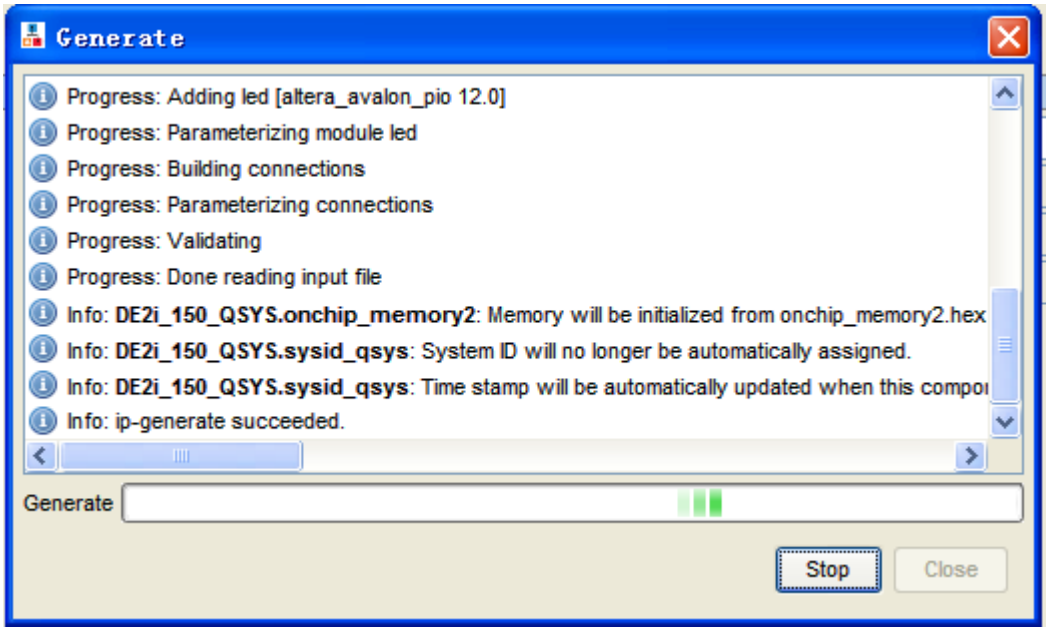

Figure 1-48 Generate Qsys

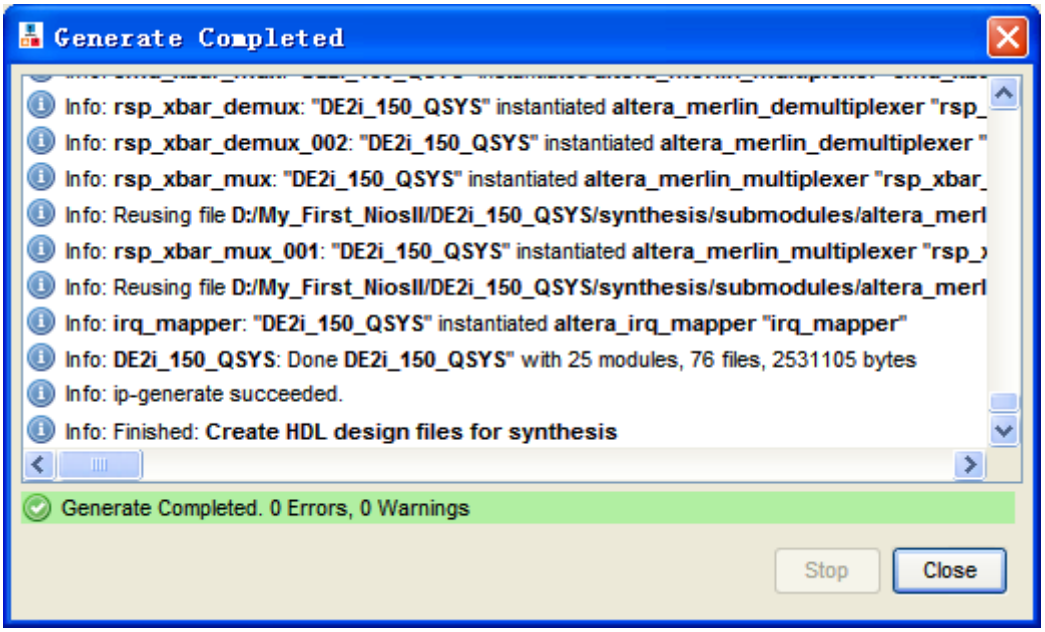

Figure 1-49 Generate Qsys Completely

31. Click **Close** to close the dialog box and **exit** the **Qsys** and return to the window as shown in Figure 1-50.

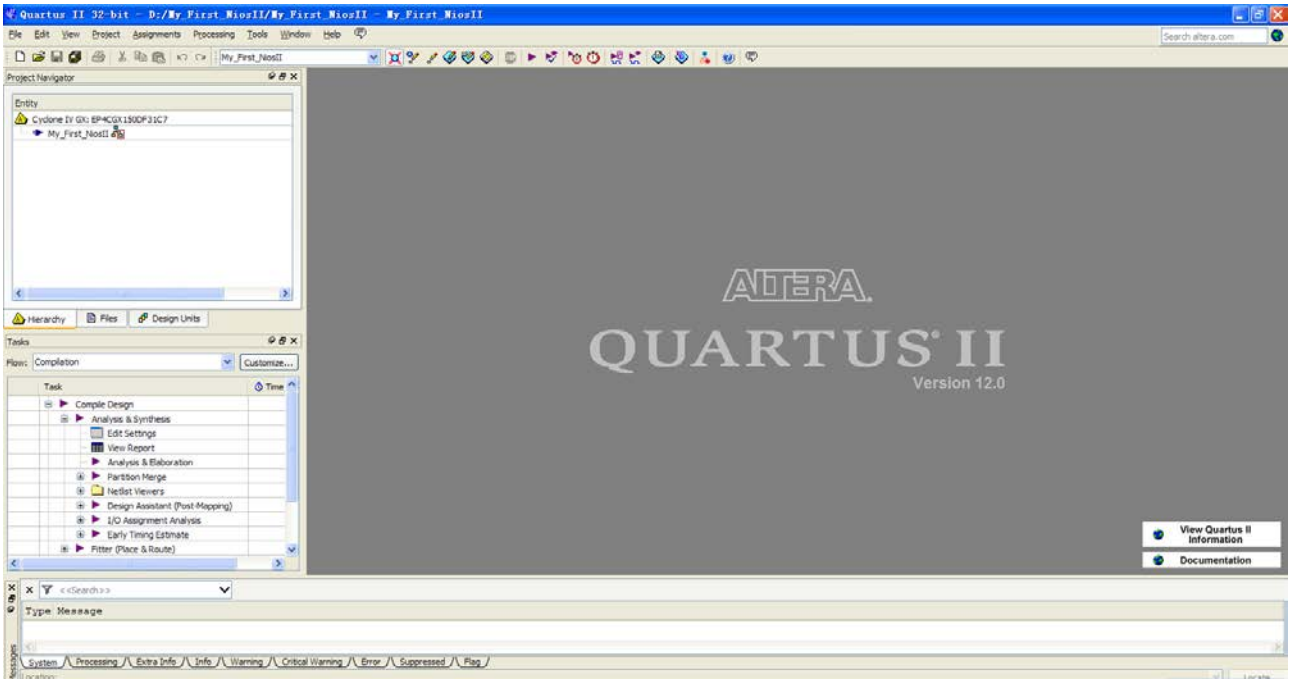

Figure 1-50 Exit Qsys

32. Choose **File** > **New** to open new files wizard. See Figure 1-51 and Figure 1-52.

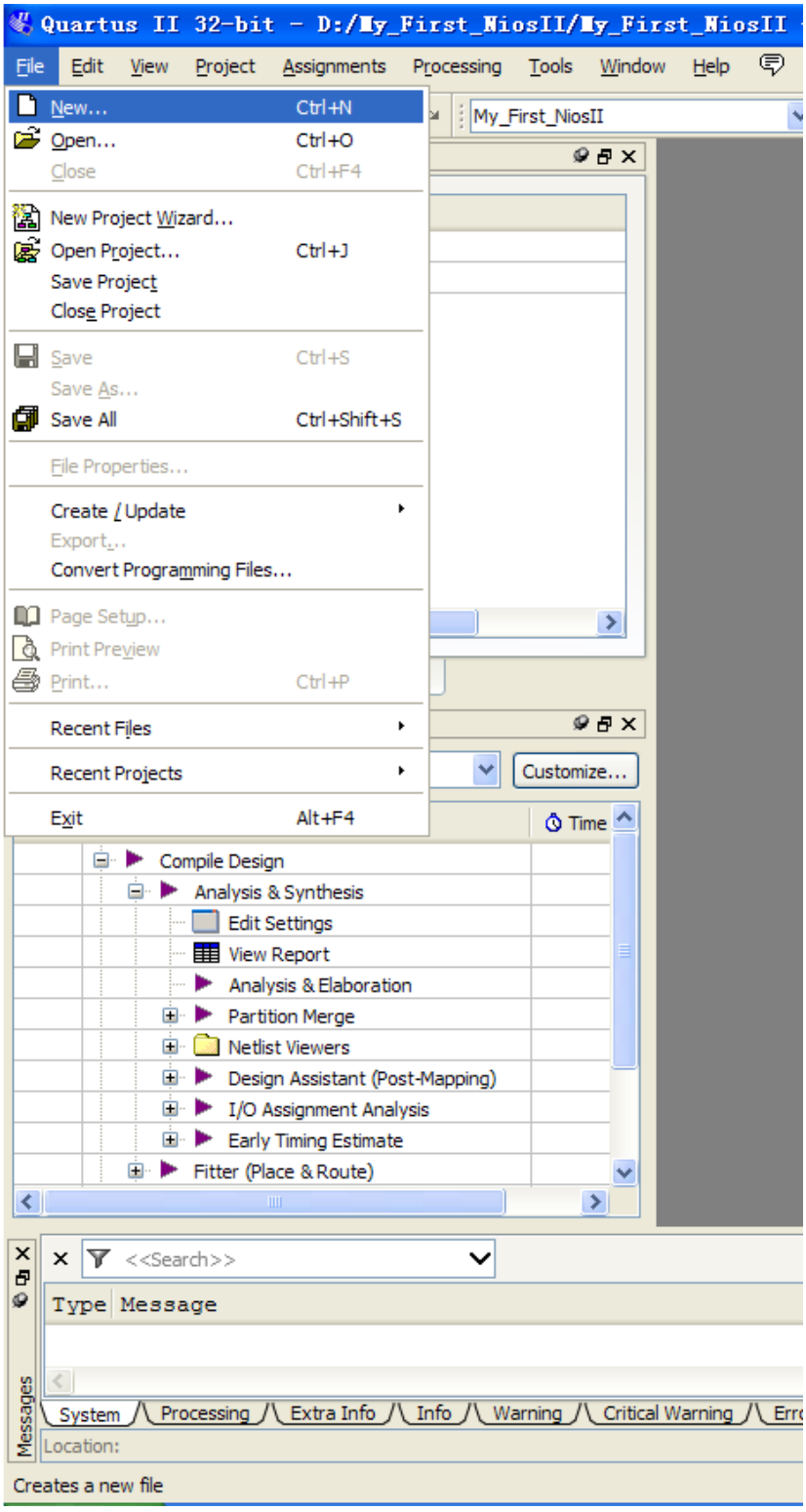

Figure 1-51 New Verilog file

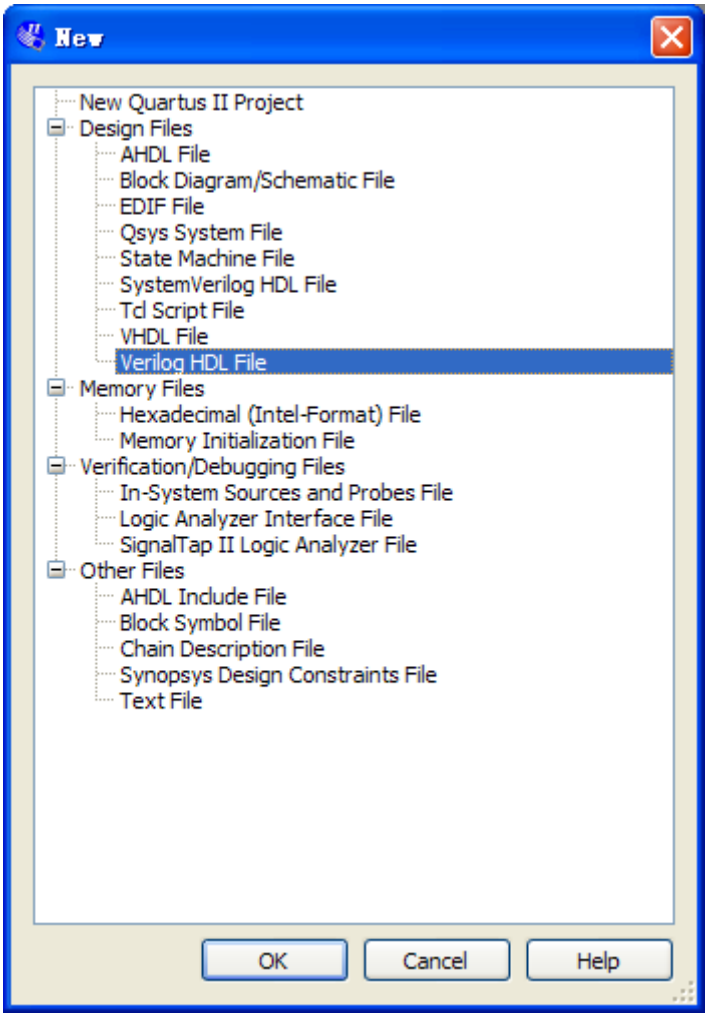

Figure 1-52 New Verilog File

33. Choose **Verilog HDL File** and click **OK** to return to the window as shown in Figure 1-53. Figure 1-53 show a blank verilog file.

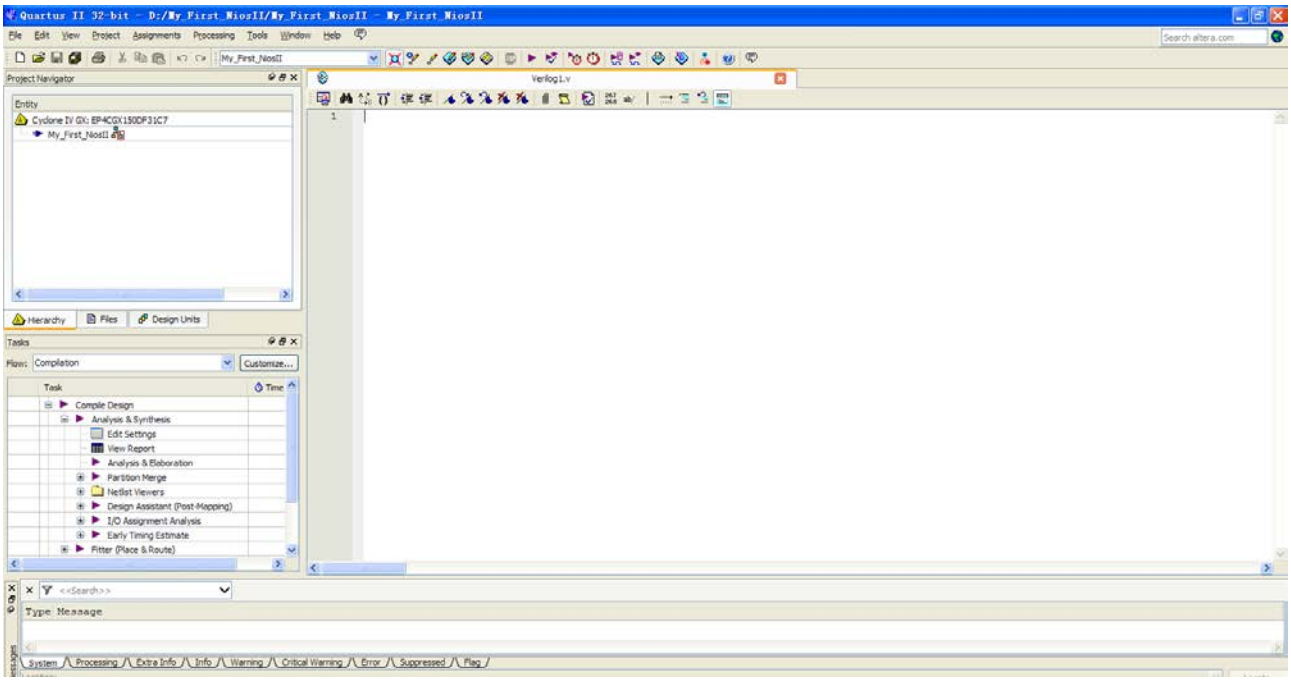

Figure 1-53 A blank verilog file

34. Type verilog the following script as shown in Figure 1-54. The module **DE2i_50_Qsys** of the code is from **DE2_115_Qsys.v** of the project. See Figure 1-55 and Figure 1-56.

```
module My_First_NiosII(

	CLOCK_50,
	LED
);
input		CLOCK_50;
output	[7:0]	LED;

DE2i_150_QSYS	u0(
			.clk_clk	(CLOCK_50),
			.led_export (LED),
			.reset_reset_n	(1'b1)
	);
endmodule
```

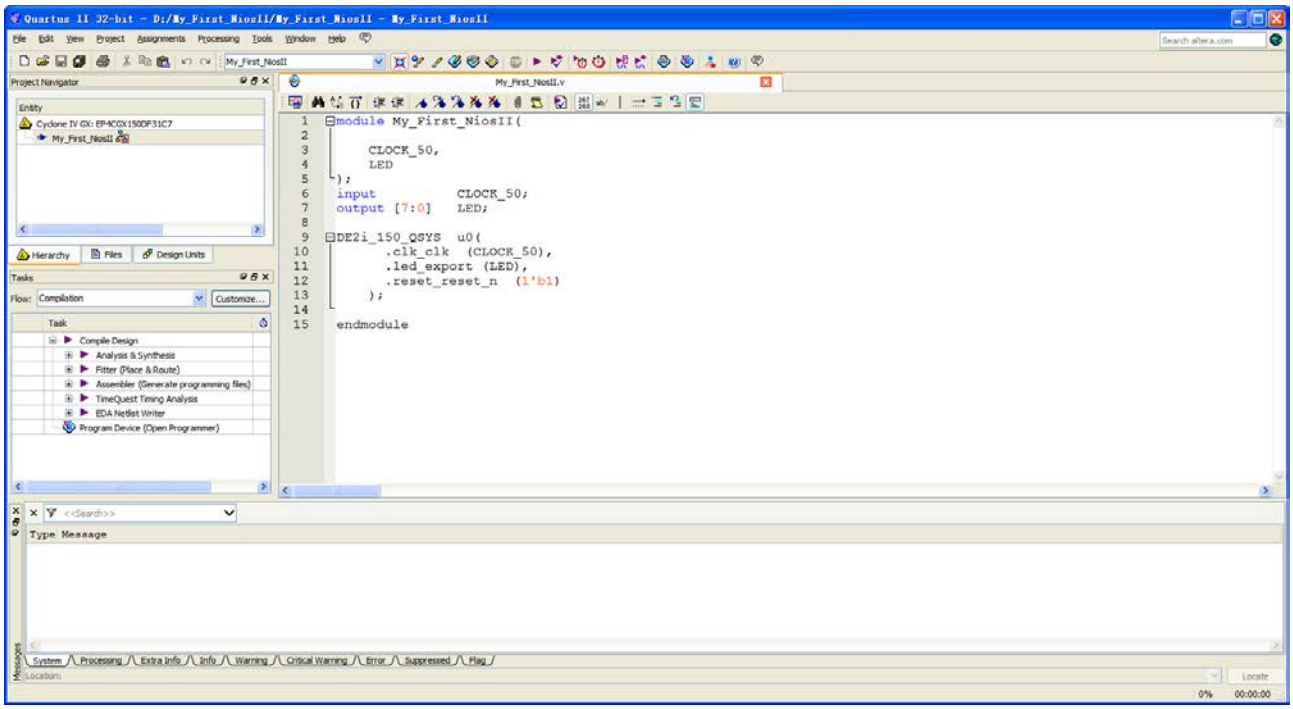

Figure 1-54 Input verilog Text

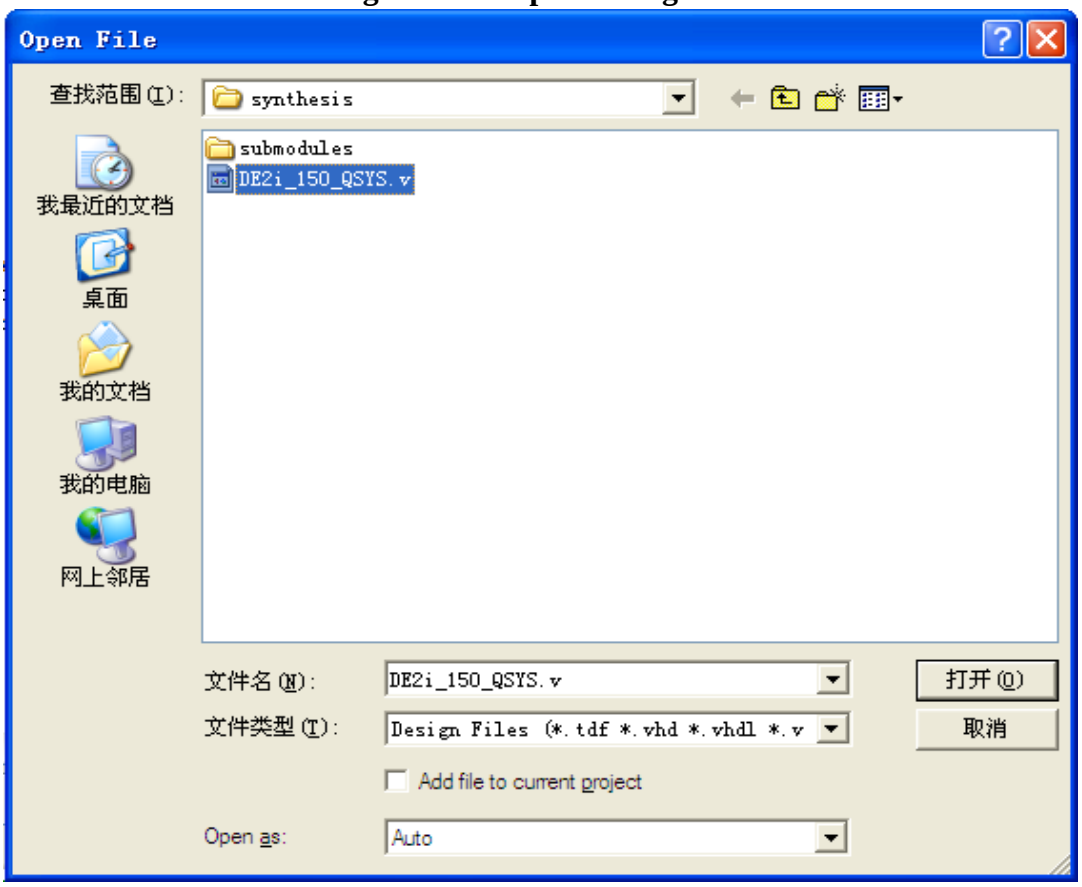

Figure 1-55 Open DE2_115_SOPC.v

```
// DE2i_150_QSYS.v

// Generated using ACDS version 12.0 178 at 2012.08.17.10:22:35

`timescale 1 ps / 1 ps
module DE2i_150_QSYS (
        output wire [7:0] led_export,    //   led.export
        input  wire       reset_reset_n, // reset.reset_n
        input  wire       clk_clk        //   clk.clk
    );

    wire         nios2_qsys_instruction_master_waitrequest;
    wire  [19:0] nios2_qsys_instruction_master_address;
    wire         nios2_qsys_instruction_master_read;
    wire  [31:0] nios2_qsys_instruction_master_readdata;
    wire         nios2_qsys_instruction_master_readdatavalid;
    wire         nios2_qsys_data_master_waitrequest;
    wire  [31:0] nios2_qsys_data_master_writedata;
    wire  [19:0] nios2_qsys_data_master_address;
    wire         nios2_qsys_data_master_write;
    wire         nios2_qsys_data_master_read;
    wire  [31:0] nios2_qsys_data_master_readdata;
    wire         nios2_qsys_data_master_debugaccess;
    wire         nios2_qsys_data_master_readdatavalid;
    wire   [3:0] nios2_qsys_data_master_byteenable;
    wire  [31:0] nios2_qsys_jtag_debug_module_translator_avalon_anti_slave_0_writedata;
```

Figure 1-56 DE2_115_SOPC module

35. Choose **Save** Icon in the tool bar. There will appear a window as shown in Figure 1-57. Click **Save**.

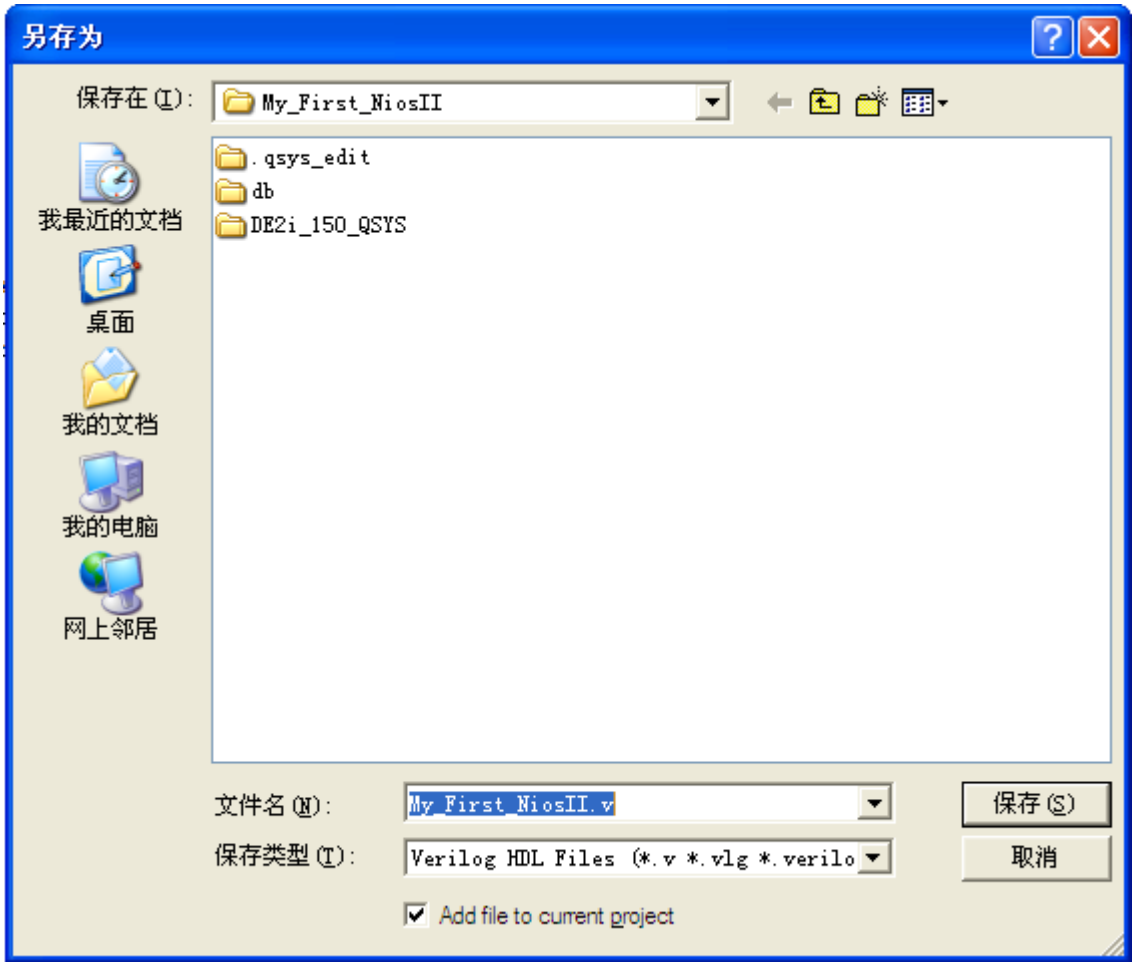

Figure 1-57 Save Verilog file

36. Add File in project as shown in Figure 1-58, add **DE2i_150_QSYS.qsys** and **DE2i_150_QSYS.v** to the project as shown in Figure 1-59 and Figure 1-60. it is completed as shown in Figure 1-61.

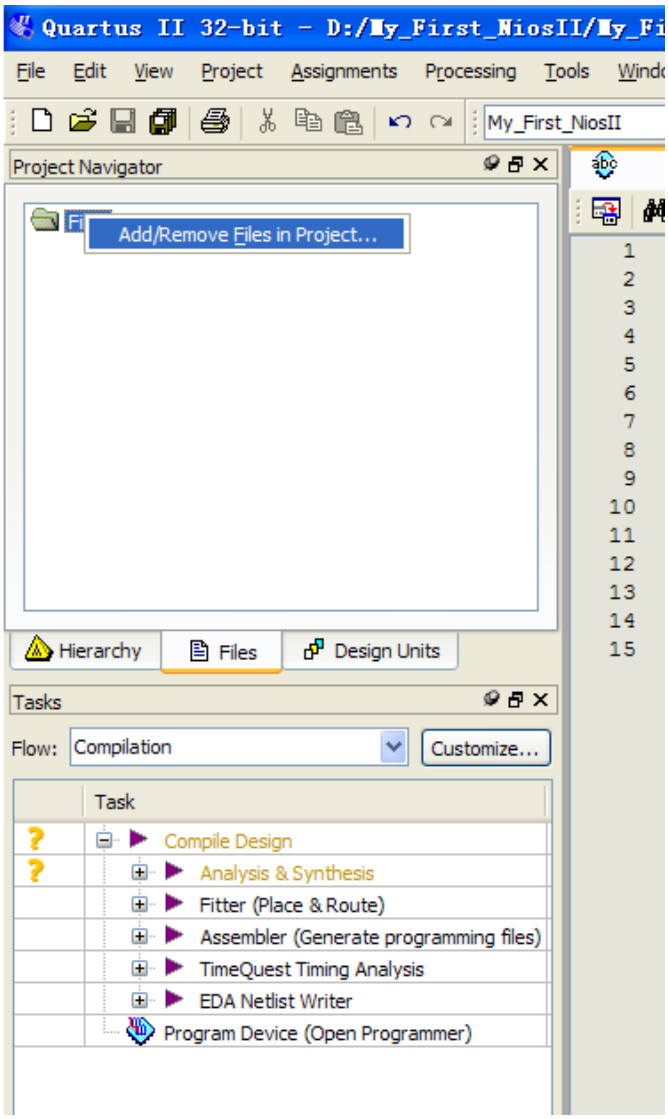

Figure 1-58 Add file

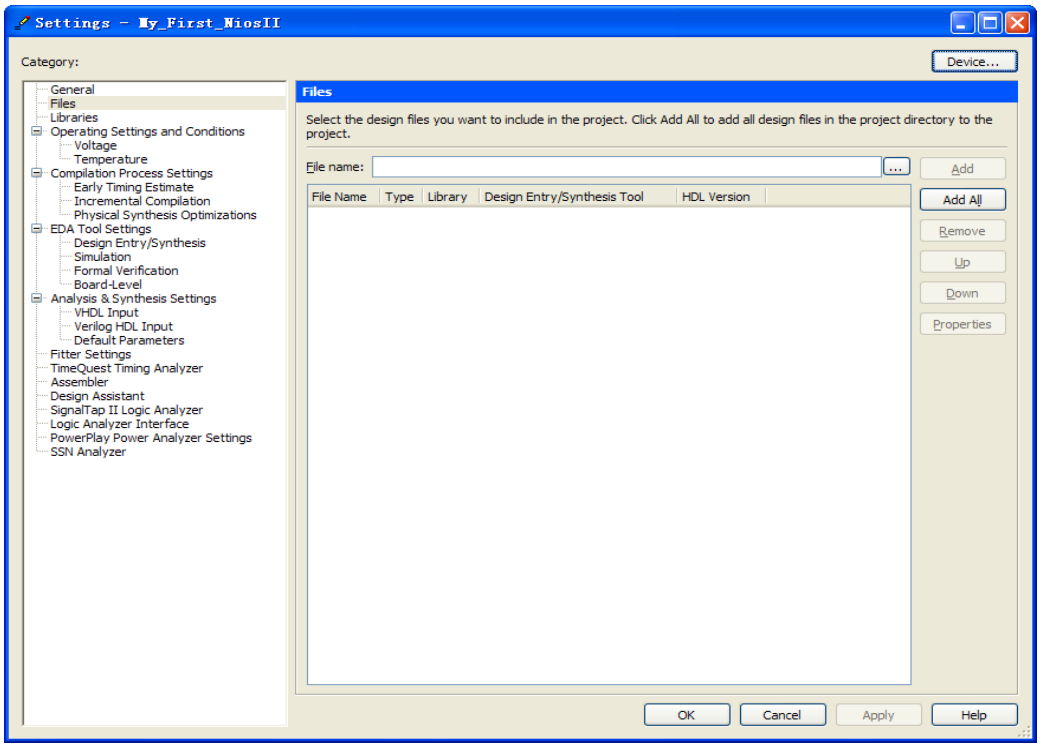

Figure 1-59 Add file

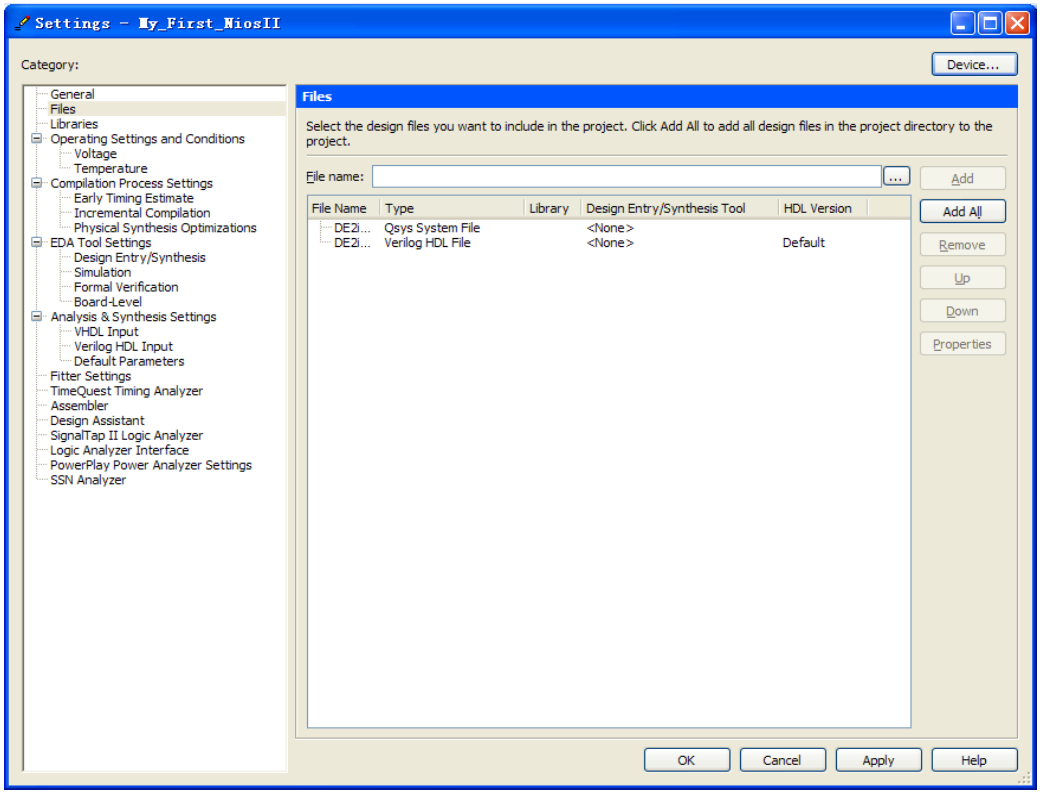

Figure 1-60 Add file

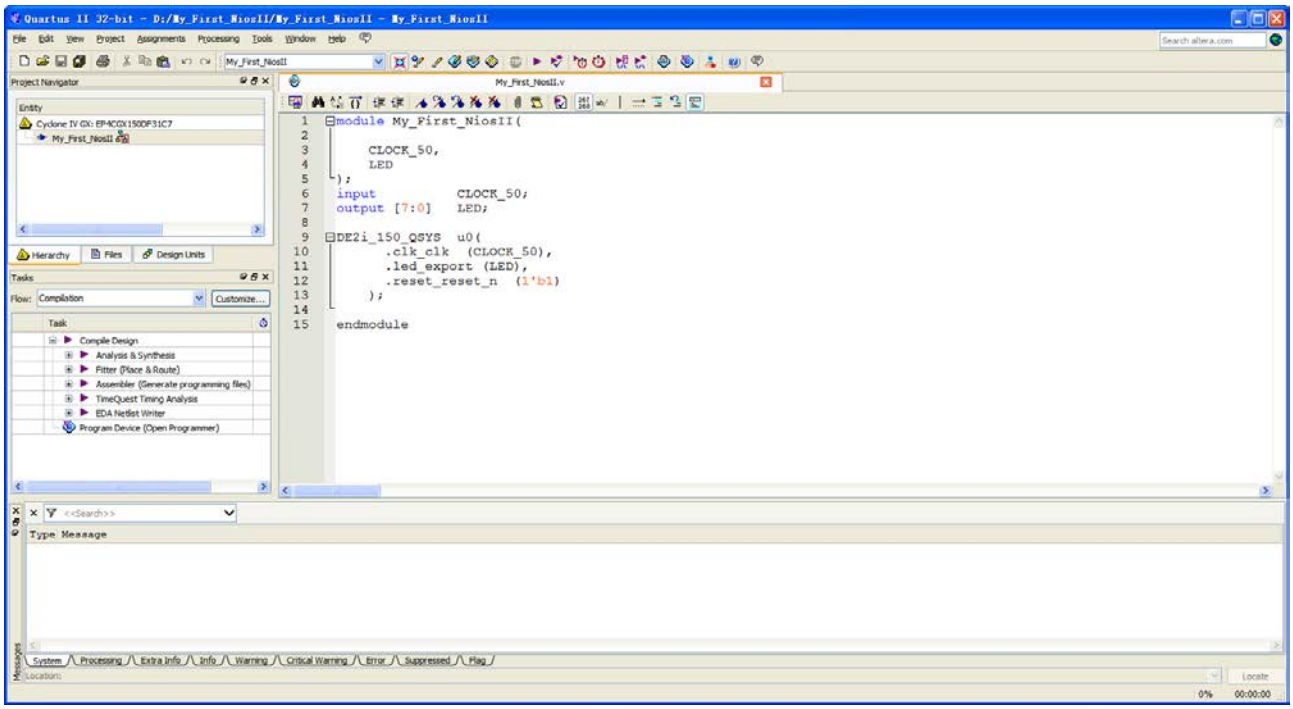

Figure 1-61 Add file completely

37. Choose **Processing** > **Start Compilation** as shown in Figure 1-62. Figure 1-63 shows the compilation process.

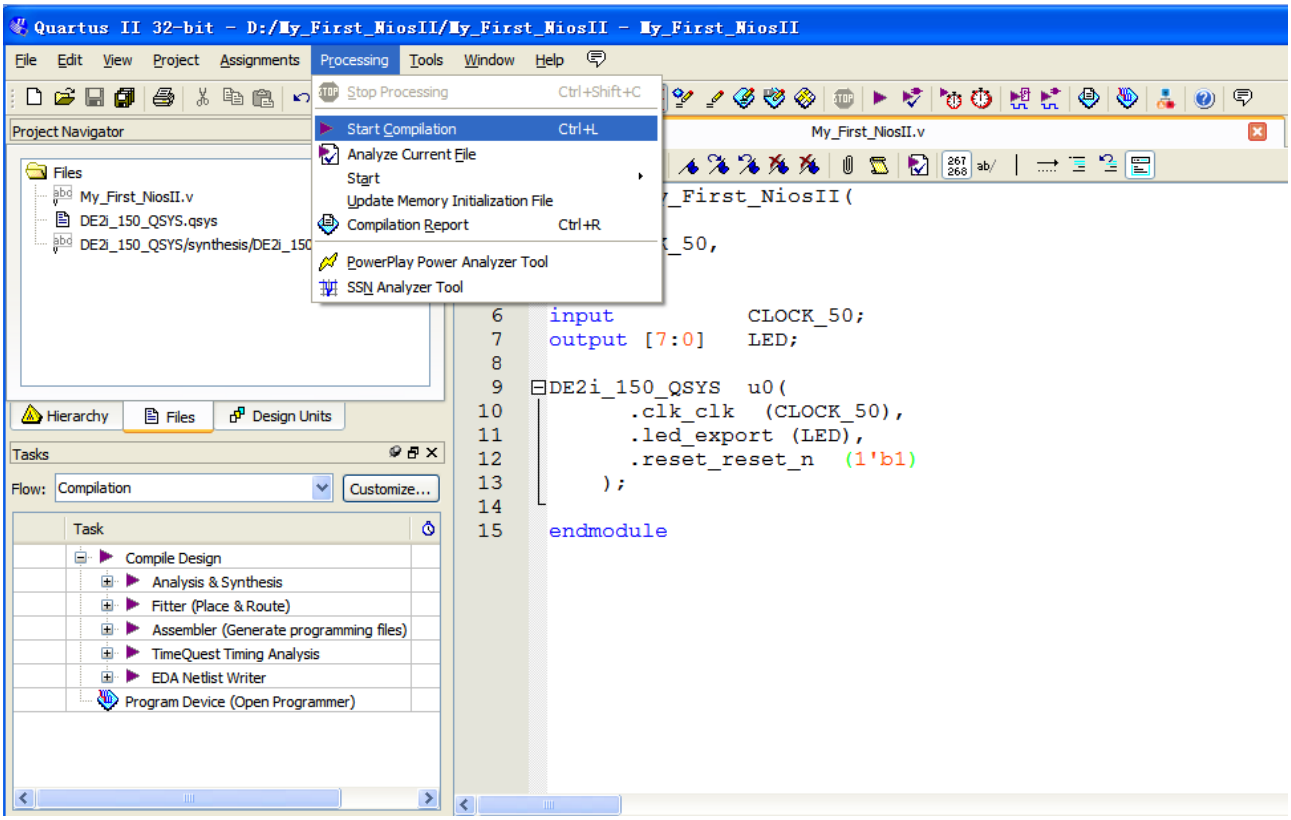

Figure 1-62 Start Compilation

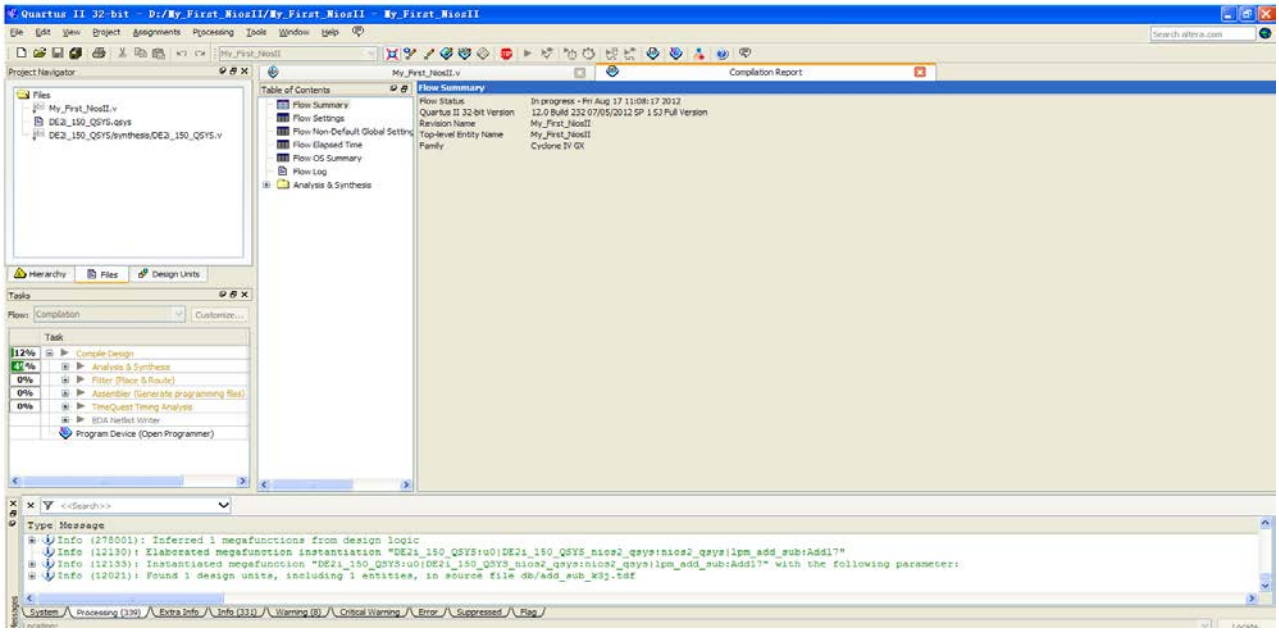

Figure 1-63 Execute Compilation

Note: In the compilation, if there is the error which shows “Error: The core supply voltage of ‘1.0v’ is illegal for the currently selected part.”, you should modify the text “set_global_assignment -name

NOMINAL_CORE_SUPPLY_VOLTAGE 1.0V" to "set_global_assignment -name NOMINAL_CORE_SUPPLY_VOLTAGE 1.2V" in the myfirst_niosii.qsf of the project.

38. A window that shows successfully will appear as shown in Figure 1-64.

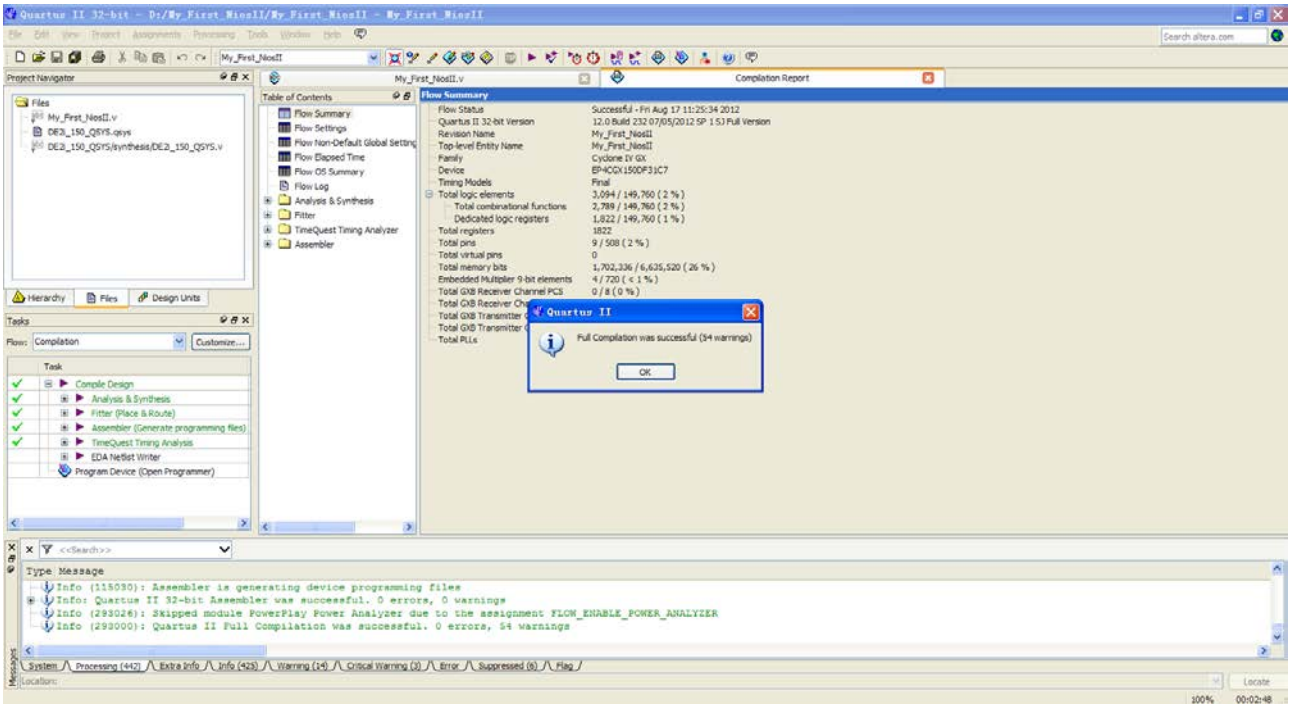

Figure 1-64 Compilation project completely

39. Choose **Assignments** > **Pins** to open pin planner as shown in Figure 1-65. Figure 1-66 show blank pins.

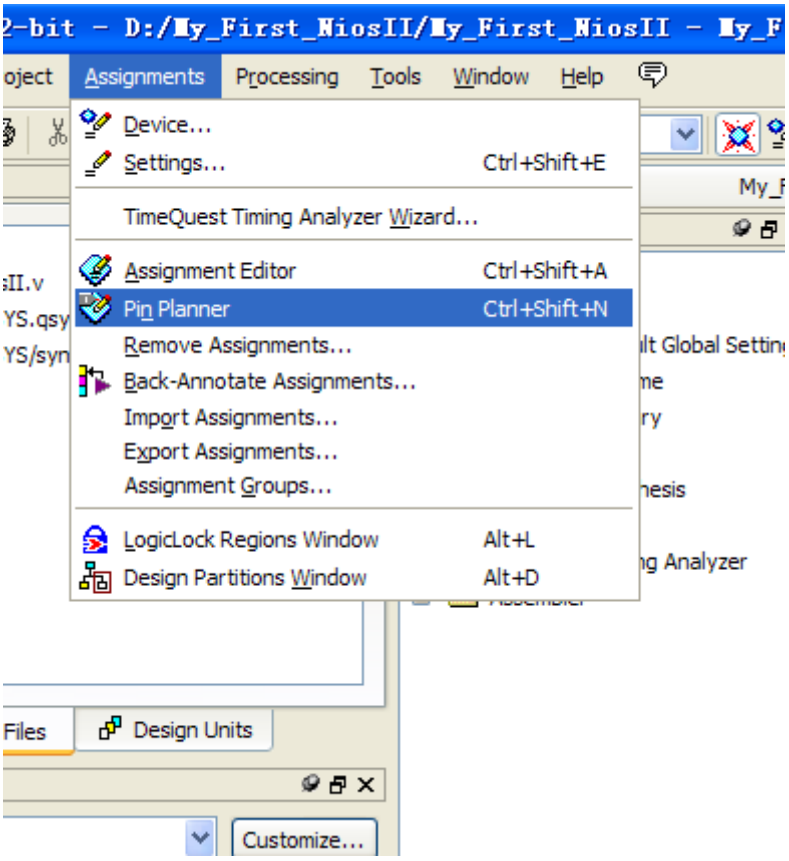

Figure 1-65 Pins menu

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair	Fitter Location
CLOCK_50	Input				2.5 V (default)		16mA (default)			PIN_W15
LED[7]	Output				2.5 V (default)		16mA (default)	2 (default)		PIN_R25
LED[6]	Output				2.5 V (default)		16mA (default)	2 (default)		PIN_A14
LED[5]	Output				2.5 V (default)		16mA (default)	2 (default)		PIN_D16
LED[4]	Output				2.5 V (default)		16mA (default)	2 (default)		PIN_R29
LED[3]	Output				2.5 V (default)		16mA (default)	2 (default)		PIN_C16
LED[2]	Output				2.5 V (default)		16mA (default)	2 (default)		PIN_F16
LED[1]	Output				2.5 V (default)		16mA (default)	2 (default)		PIN_G15
LED[0]	Output				2.5 V (default)		16mA (default)	2 (default)		PIN_AH16
<<new node>>										

Figure 1-66 Blank Pins

40. Input Location value as shown in Figure 1-67.

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate	Differential Pair
CLOCK_50	Unknown	PIN_AJ16	4	B4_N2	2.5 V (default)		16mA (default)		
LED[0]	Unknown	PIN_AA25	5	B5_N2	2.5 V (default)		16mA (default)		
LED[1]	Unknown	PIN_AB25	5	B5_N2	2.5 V (default)		16mA (default)		
LED[2]	Unknown	PIN_F27	6	B6_N0	2.5 V (default)		16mA (default)		
LED[3]	Unknown	PIN_F26	6	B6_N0	2.5 V (default)		16mA (default)		
LED[4]	Unknown	PIN_W26	5	B5_N0	2.5 V (default)		16mA (default)		
LED[5]	Unknown	PIN_Y22	5	B5_N1	2.5 V (default)		16mA (default)		
LED[6]	Unknown	PIN_Y25	5	B5_N2	2.5 V (default)		16mA (default)		
LED[7]	Unknown	PIN_AA22	5	B5_N1	2.5 V (default)		16mA (default)		
<<new node>>									

Figure 1-67 Set Pins

41. Close the **pin planner**. Restart compilation the project as shown in Figure 1-68.

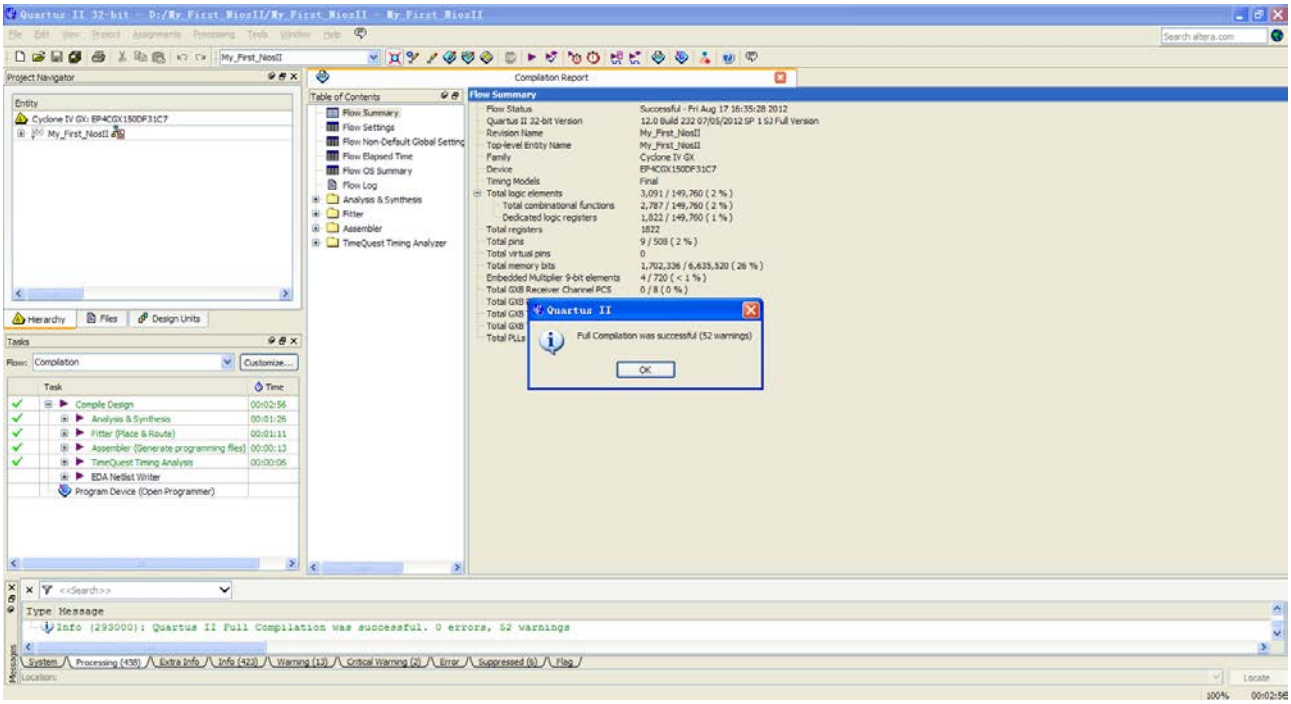

Figure 1-68 Compilation project again

1.3 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 II 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.

3. Start the Nios II Software Build Tools (SBT) for Eclipse.

4. After the welcome page appears, click **Workbench**.

5. Choose **Nios II->Quartus II Programmer**.

6. Click **Auto Detect**. The device on your development board should be detected automatically.

7. Click the top row to highlight it.

8. Click **Change File**.

9. Browse to the My_First_NiosII project directory.

10. Select the programming file (My_First_NiosII.sof) for your board.

11. Click **OK**.

12. Click **Hardware Setup** in the top, left comer of the Quartus II programmer window. The Hardware Setup dialog box appears.

13. Select USB-Blaster 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 II Help for information on how to install the driver. See Figure 1-69.

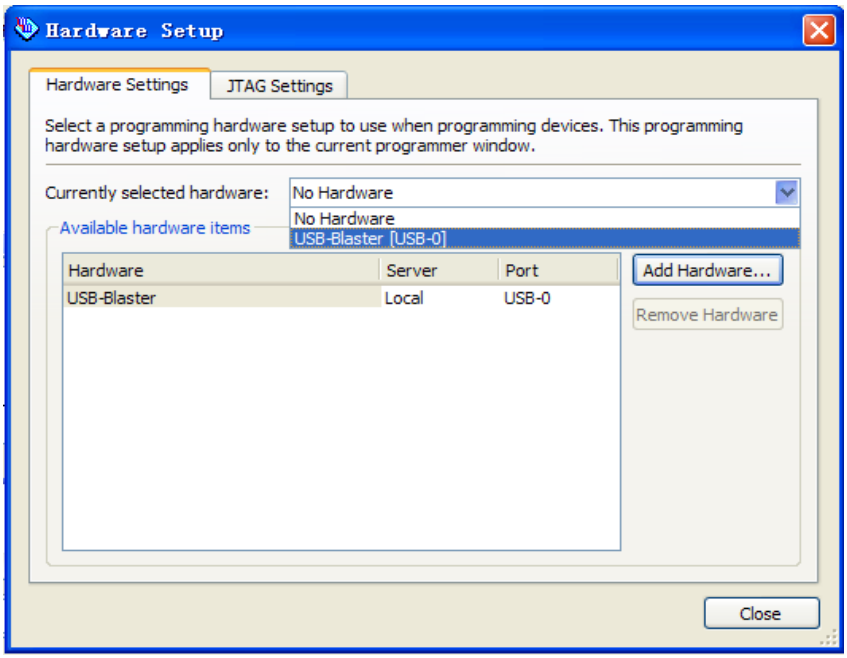

Figure 1-69 Hardware Setup Window

14. Click **Close**.

15. Turn on the **Program/Configure** option for the programming file.(See Figure 1-70 for an example).

16. Click **Start**.

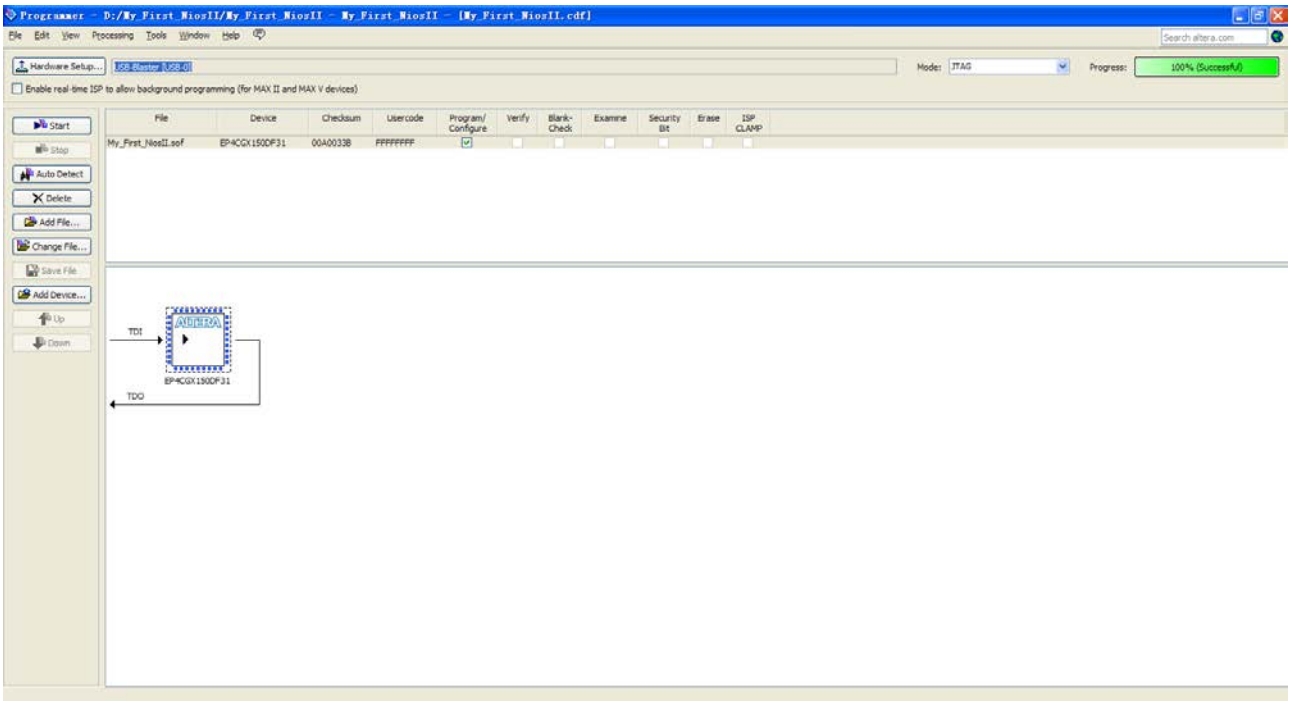

Figure 1-70 Quartus II Programmer

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

Chapter 2 NIOS II Softwar Build Tools for Eclipse

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

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

2.1 Create the hello_world Example Project

In this section you will create a new NIOS II C/C++ application project based on an installed example. To begin, perform the following steps in the NIOS II SBT for Eclipse:

1. Return to the NIOS II Software Build Tools for Eclipse.

   Note: you can close the Quartus II Programmer or leave it open in the background if you want to reload the processor system onto your development board quickly.

2. Choose **File** > **Switch Workspace** to switch workspace. See Figure 2-1 and Figure 2-2.

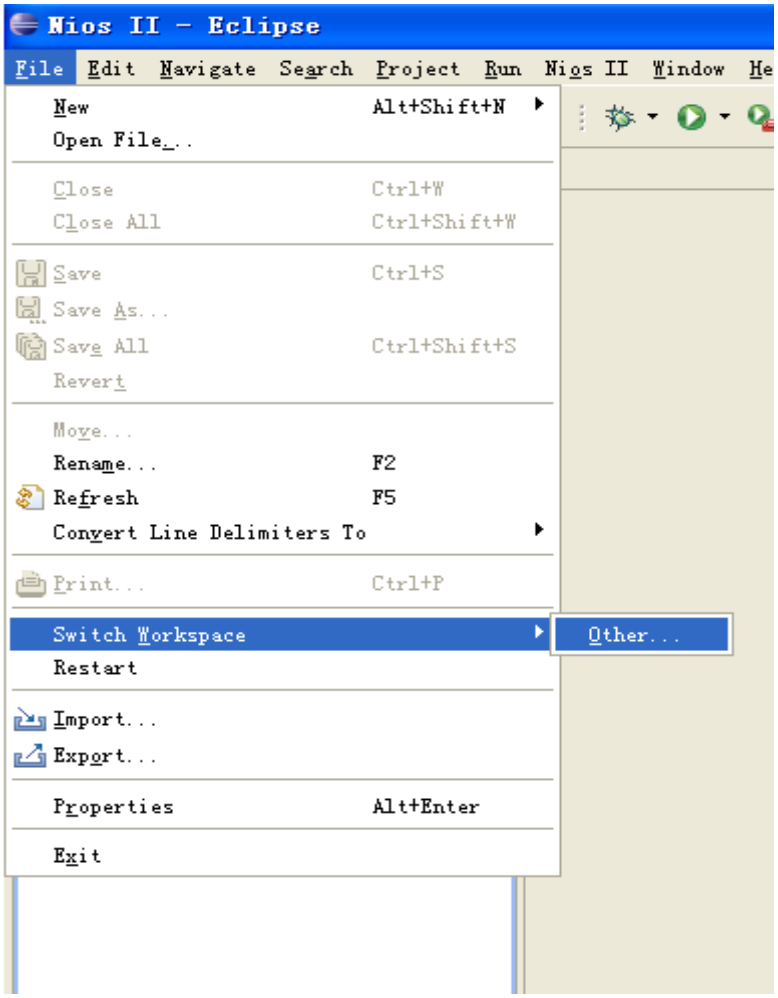

Figure 2-1 Switch Workspace (1)

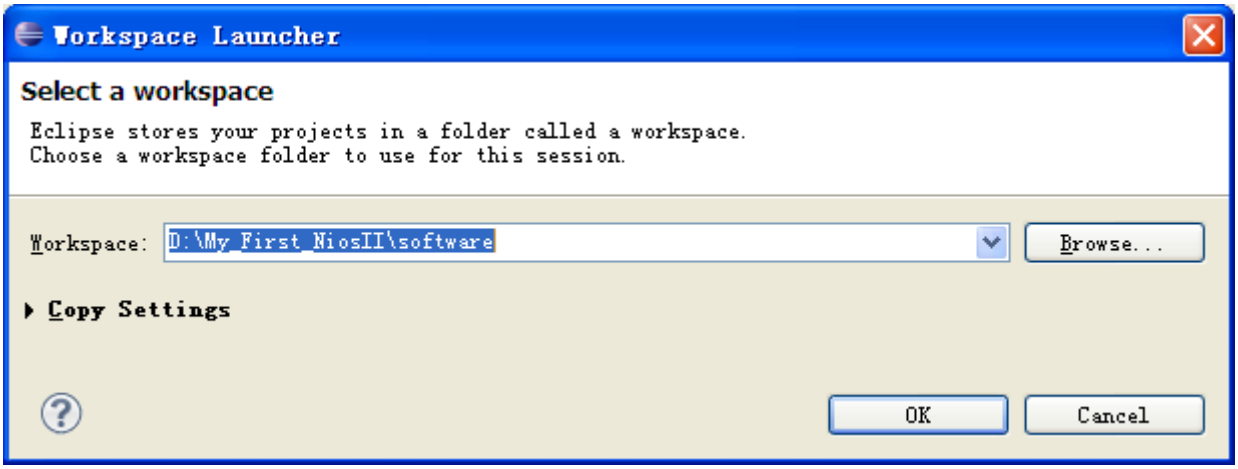

Figure 2-2 Switch Workspace (2)

3. Choose File->New->NIOS II Application and BSP from Template open the New Project Wizard.

4. In the New Project wizard, make sure the following things:

- Under **Target hardware information**, next to **SOPC Information File name**, browse to locate the <design files directory> where the previously created hardware project resides as shown in Figure 2-3.

- Select first_nios2_system.sopcinfo and click Open. You return to the Nios II Application and BSP from Template wizard showing current information for the **SOPC Information File name** and **CPU name** fields.

- Select the **Hello World** project template.

- Give the project a name. (**hello_world_0** is default name),there we rename it to My_First_NiosII.

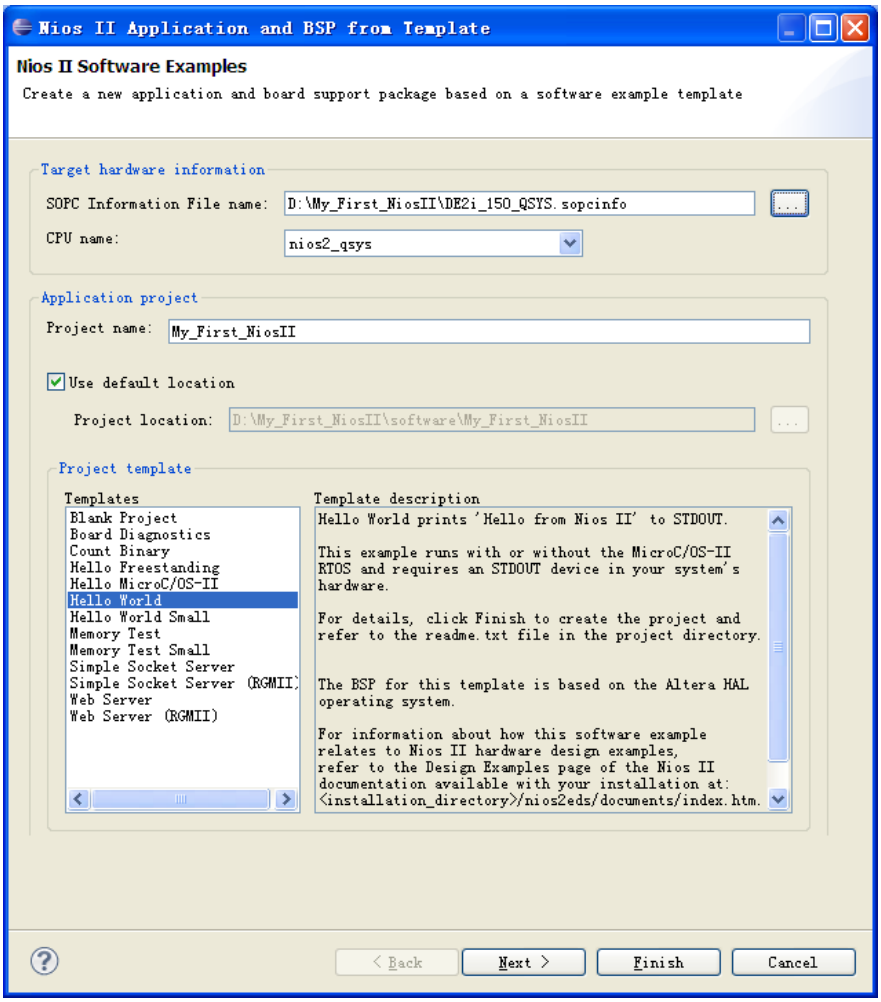

Figure 2-3 Nios II-Ecplise New Project Wizard

5. Click **Finish**. The **NIOS II SBT for Eclipse** creates the **My_First_NiosII** project and returns to the Nios II C/C++ project perspective. See Figure 2-4.

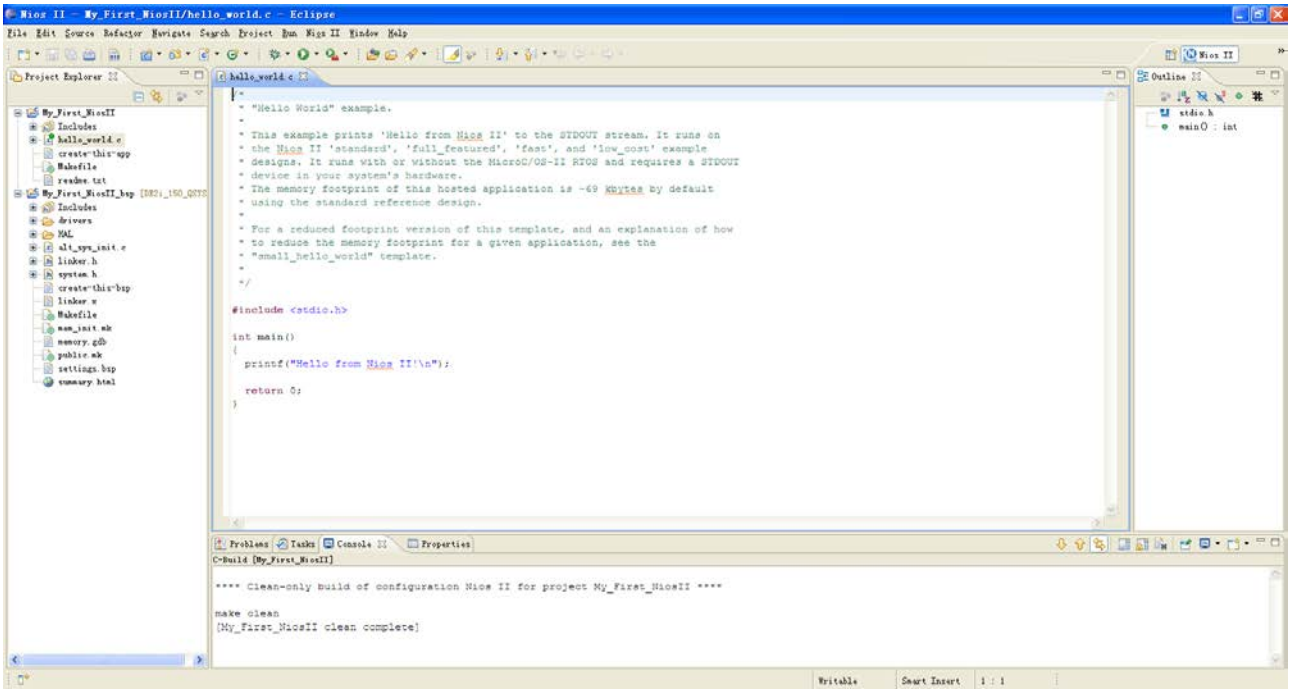

Figure 2-4 Ecplise Project Perspective for My_First_NiosII

When you create a new project, the NIOS II SBT for Eclipse creates two new projects in the NIOS II C/C++ Projects tab:

■ **My_First_NiosII** (**hello_world_0** is default name) is your C/C++ application project. This project contains the source and header files for your application.

■**My_First_NiosII_bsp** (**hello_world_0_bsp** is default name) is a board support package that encapsulates the details of theNios II system hardware.

Note:When you build the system library for the first time the NIOS II SBT for Eclipse automatically generates files useful for software development, including:

● Installed IP device drivers, including SOPC component device drivers for the NIOS II hardware system

● Newlib C library, which is a richly featured C library for the NIOS II processor.

● NIOS software packages which includes NIOS II hardware abstraction layer, NicheStack TCP/IP Network stack, NIOS II host file system, NIOS II read-only zip file system and Micrium's µC/OS-II real time operating system(RTOS).

- **system.h**, which is a header file that encapsulates your hardware system.
- **alt_sys_init.c**, which is an initialization file that initializes the devices in the system.
- **Hello_world_0.elf**, which is an executable and linked format file for the application located in hello_world_0 folder under Debug.

2.2 Build and Run the Program

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

To build the program, right-click the **My_First_NiosII** project in the Nios II C/C++ Projects tab and choose **Build Project**. The **Build Project** dialog box appears and the **Eclipse** begins compiling the project. When compilation completes, a message '[My_First_NiosII build complete]' will appear in the Console tab. The compilation time varies depending on your system. See Figure 2-5 for an example.

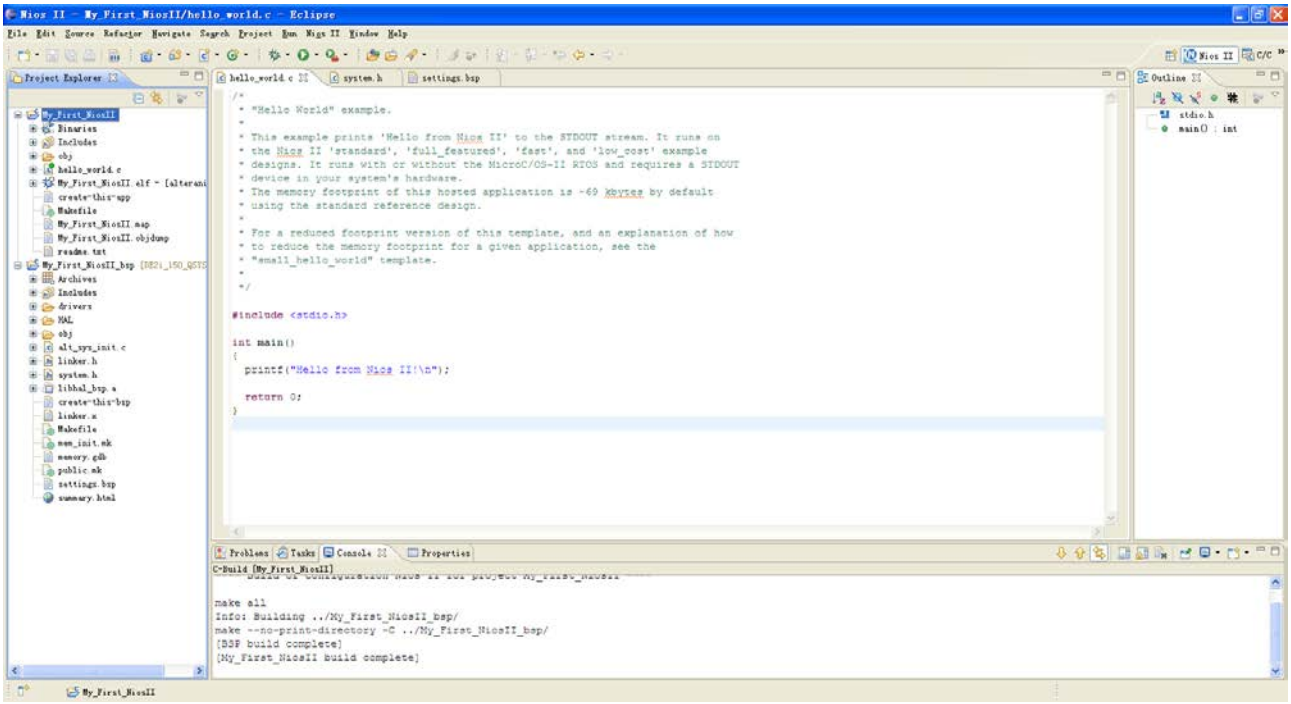

Figure 2-5 My_First_NiosII Build Completed

After compilation complete, right-click the **My_First_NiosII** project, choose **Run As**, and choose**NIOS II Hardware**. The Eclipse begins to download the program to the target FPGA developmentboard and begins execution. When the target hardware begins executing the program, the message '**Hello from Nios II!**' appears in the NIOS II SBT for Eclipse Console tab. See Figure 2-6 for an example.

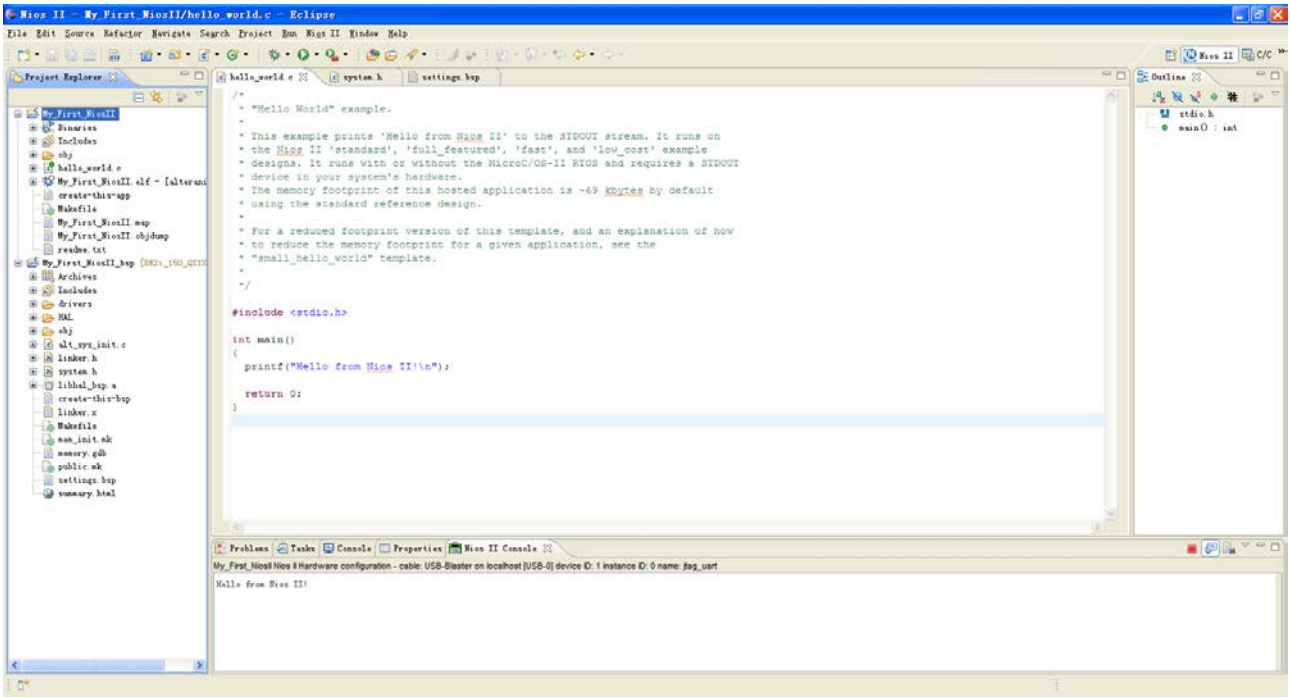

Figure 2-6 My_First_NiosII Program Output

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

2.3 Edit and Re-Run the Program

You can modify the **hello_world.c** program file in the Eclipse, 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, add the text shown in blue in the example below:

```
#include <stdio.h>

#include "system.h"

#include "altera_avalon_pio_regs.h"

int main()
```

```
{
    printf("Hello from Nios II!\n");
    int count = 0;
    int delay;
    while(1)
    {
        IOWR_ALTERA_AVALON_PIO_DATA(LED_BASE, count & 0x01);
        delay = 0;
        while(delay < 1000000)
        {
            delay++;
        }
        count++;
    }
    return 0;
}
```

2. Save the project.

3. Recompile the file by right-clicking **My_First_NiosII** in the NIOS II C/C++ Projects tab and choosing **Run** > **Run As** > **Nios II Hardware**.

   Note: You do not need to build the project manually; the NIOS II SBT for Eclipse automatically re-builds the program before downloading it to the FPGA.

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

2.4 Why the LED Blinks

The Nios II system description header file, **system.h**, contains the software definitions, name, locations, base addresses, and settings for all of the components in the Nios II hardware system. The **system.h** file is located in the in the **My_First_NiosII _bsp** directory as shown in Figure 2-7.

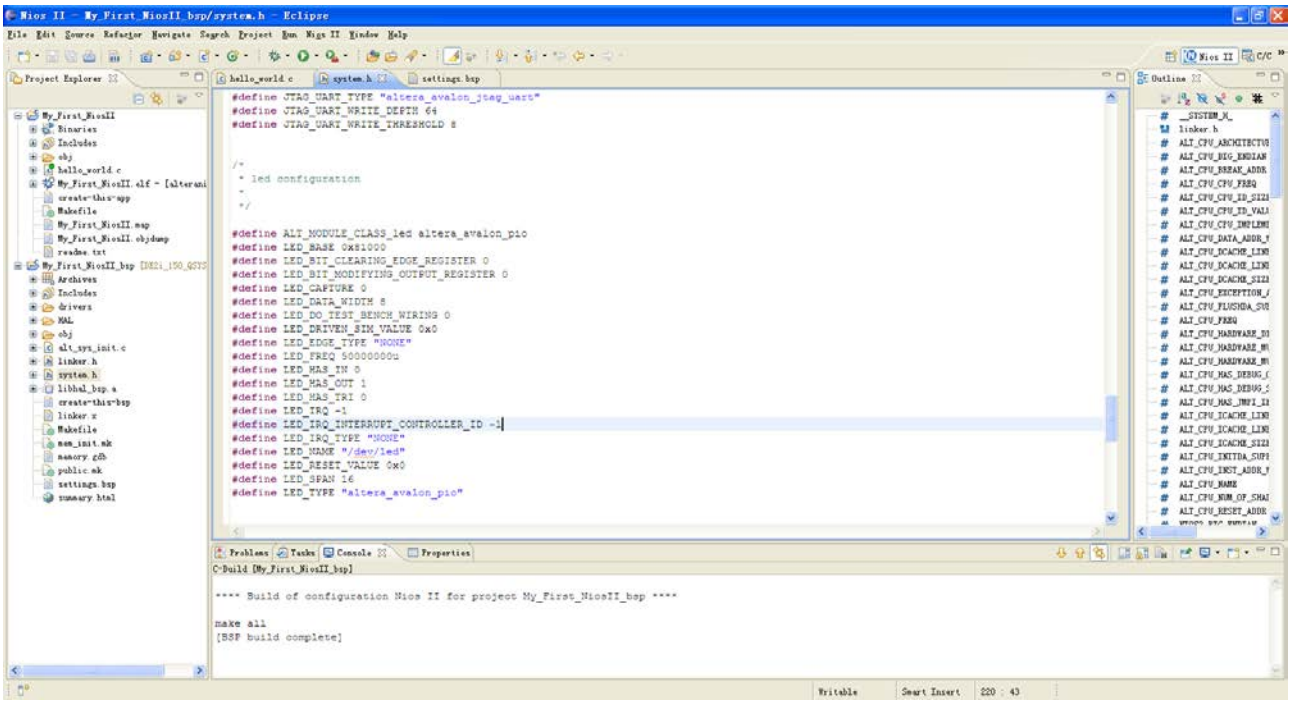

Figure 2-7 System.h Location

If you look at the **system.h** file for the Nios II project example used in this tutorial, you will notice the **led** function. This function controls the LED. The Nios II 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 **altera**\<version number>**ip\sopc_builder_ip\altera_avalon_pio**.

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 SOPC peripherals that you can use in a system. To learn about the PIO core and other embedded peripheral cores, refer to Quartus II Version <version> Handbook Volume 5: Embedded Peripherals.

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

2.5 Debugging the Application

Before you can debug a project in the NIOS II SBT for Eclipse, 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 2-8.

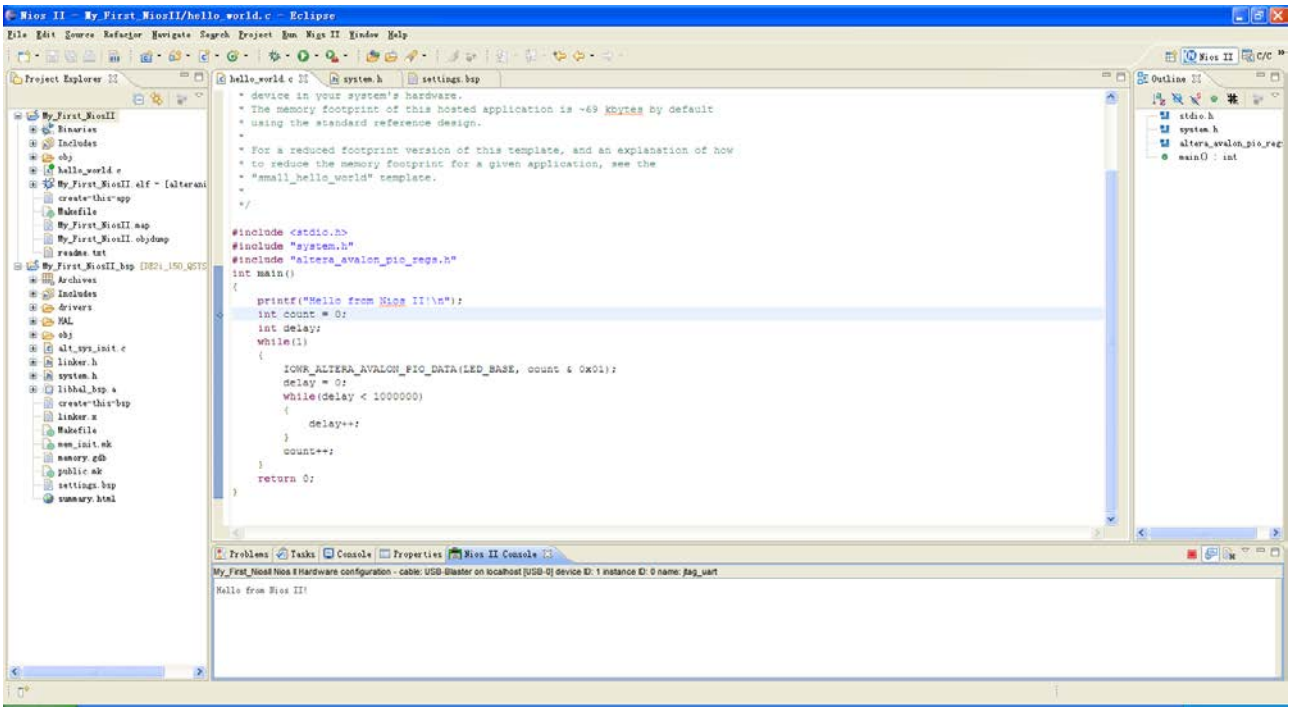

Figure 2-8 Set Breakpoint

2. To debug your application, right-click the application (**hello_world_0** by default) and choose **Debug as > Nios II Hardware**.

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

4. 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.

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

When debugging a project in the Nios II SBT for Eclipse, 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 Nios II C/C++ project perspective from the debug perspective, click the two arrows >> in the top right corner of the GUI.

2.6 Configure BSP Editor

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 the Nios II SBT for Eclipse, right-click **My_First_NiosII_bsp** and choose **Nios II-> BSP Editor**. The **BSP Editor** dialog box opens.

2. The **Main** page contains settings related to how the program interacts with the underlying hardware. The settings have names that correspond to the targeted NIOS II hardware.

3. In the **Linker Script** box, observe which memory has been assigned for **Program memory(.text)**, **Read-only data memory(.rodata)**, **Read/write data memory(.rwdata)**, **Heap memory, and Stack memory**, see Figure 0-9. These settings determine which memory is used to store the compiled executable program when the example **My_First_NiosII** programs runs. You can also specify which interface you want to use for stdio , stdin, and stderr. You can also add and configure an RTOS for your application and configure build options to support C++, reduced device drivers, etc.

4. Choose **onchip_memory2** for all the memory options in the **Linker Script** box. See Figure 2-9 for an example.

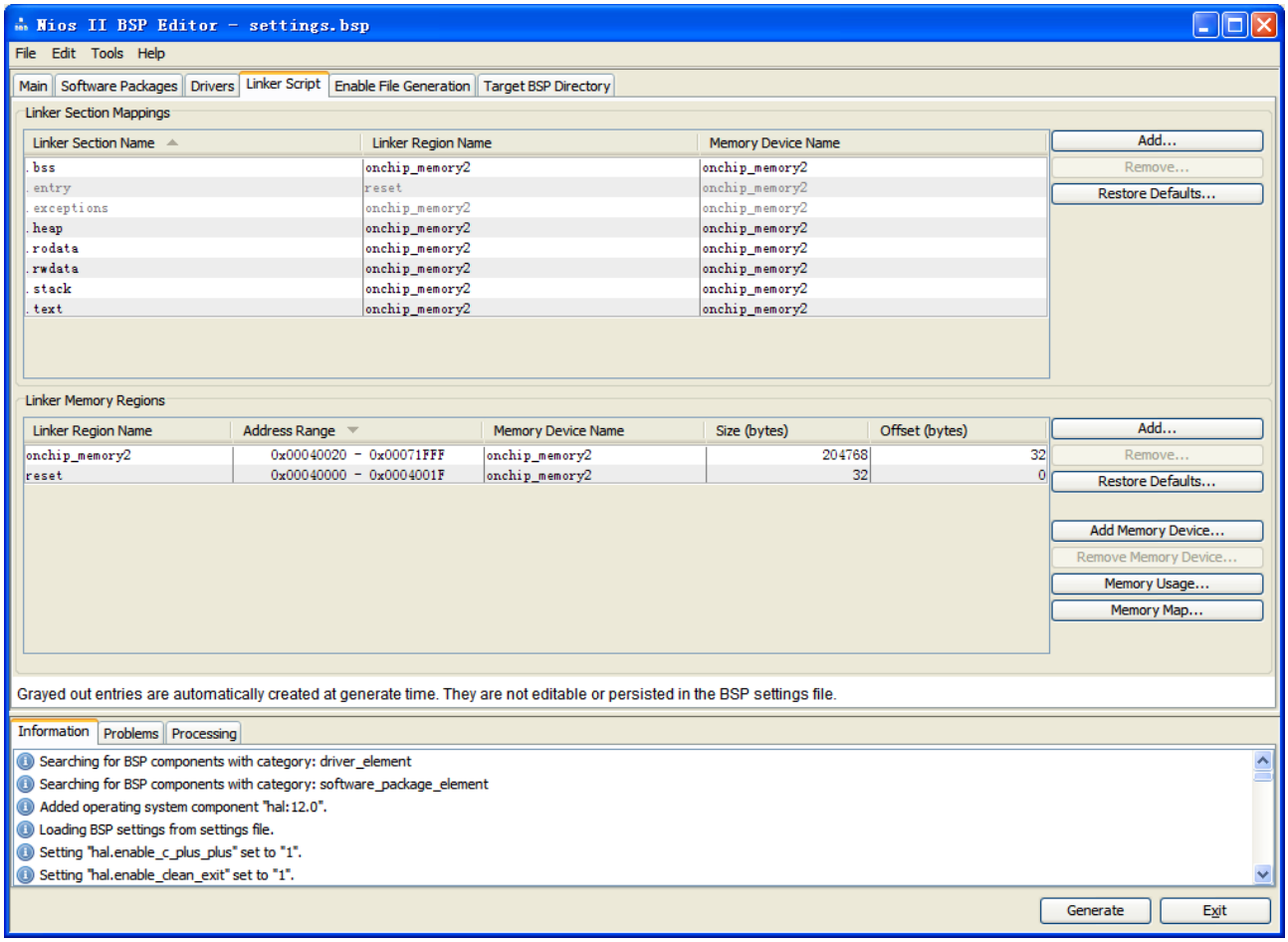

Figure 2-9 Configuring BSP

5. Click **Exit** to close the **BSP Editor** dialog box and return to the Eclipse workbench.

Note: If you make changes to the system properties or the Qsys properties or your hardware, you must rebuild your project. To rebuild, right-click the **My_First_NiosII_BSP->Nios II->Generate BSP** and then **Rebuild Project.**

Chapter 3 *Programming the CFI Flash*

With the density of FPGAs increasing, the need for larger configuration storage is also increasing. If your system contains a common flash interface (CFI) flash memory, you can use your system for FPGA configuration storage as well.

3.1 Modify the SOPC of the Project

1. Choose **Library** > **Qsys interconnect** > **Tri-State Components** > **Generic Tri-State Controller** to open the Generic Tri-State Controller wizard. See Figure 3-1.

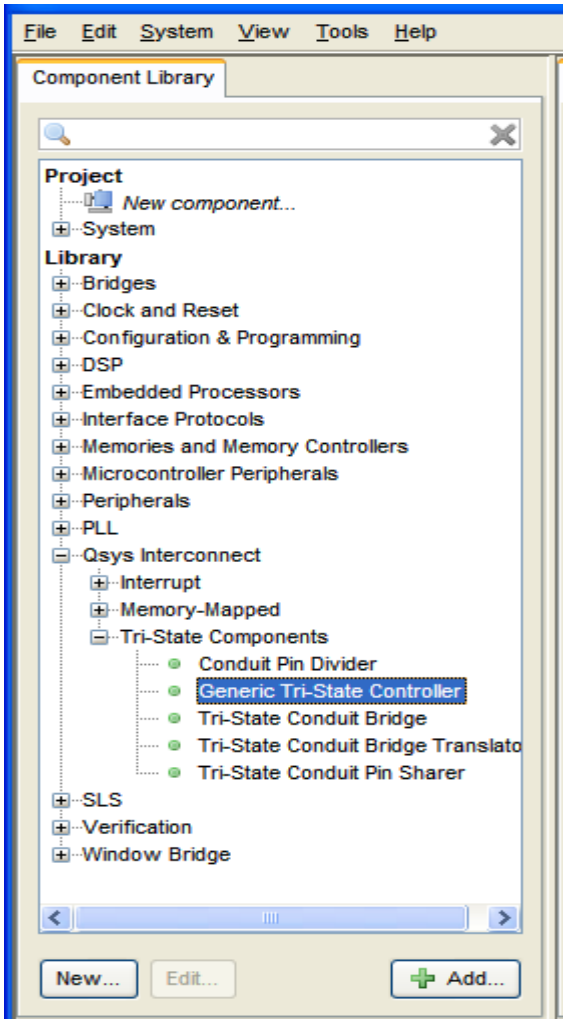

Figure 3-1 Add Generic Tri-State Controller

2. Choose **Project > Library > Flash Memory Interface(CFI)** and click **Apply** as shown in Figure 3-2.

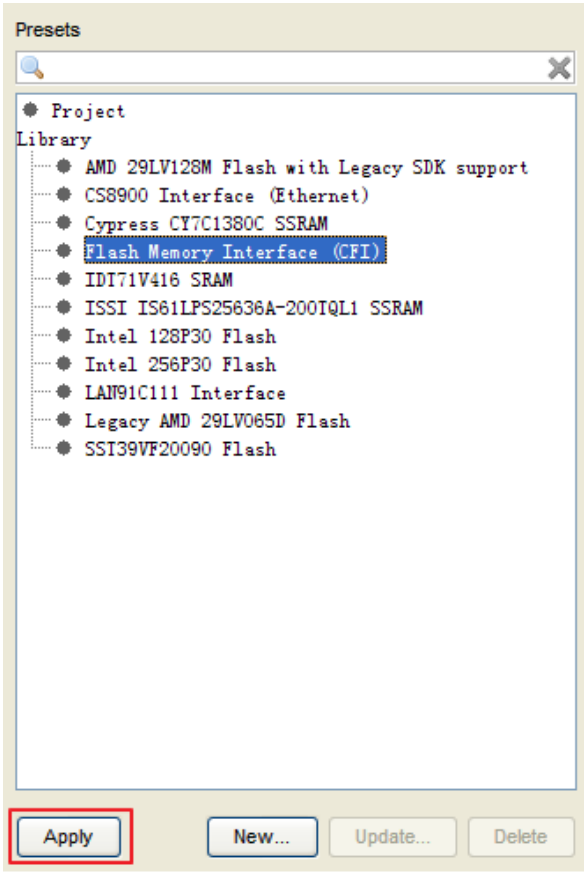

Figure 3-2 Select Library

3. Modify the **Singnal Selection** and **Signal Timing** and **Signal Polarities** as shown in Figure 3-3 to Figure 3-9.

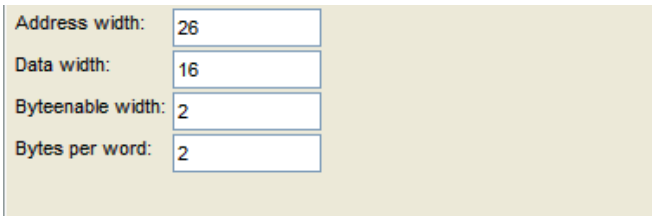

Figure 3-3 Modify CFI Flash[page 1 of 7]

Figure 3-4 Modify CFI Flash[page 2 of 7]

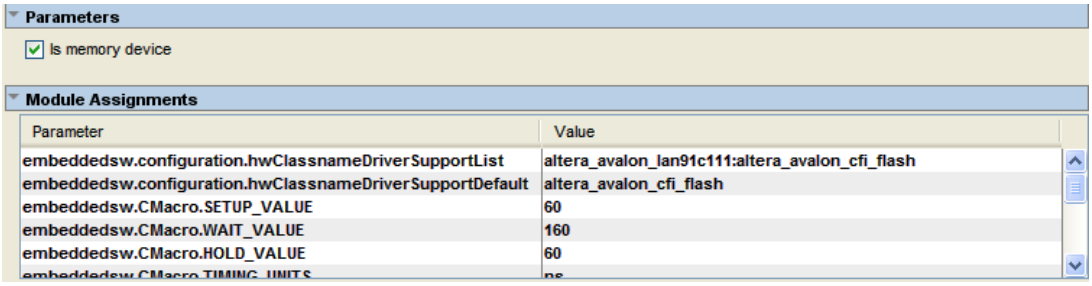

Figure 3-5 Modify CFI Flash[page 3 of 7]

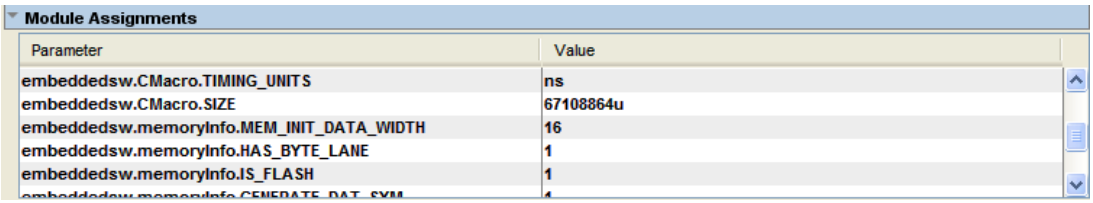

Figure 3-6 Modify CFI Flash[page 4 of 7]

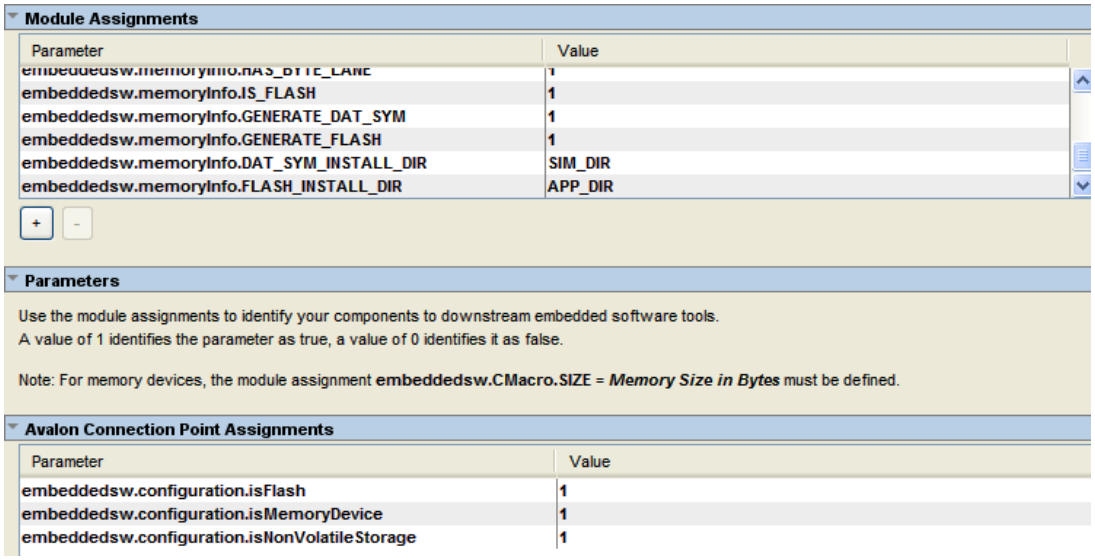

Figure 3-7 Modify CFI Flash[page 5 of 7]

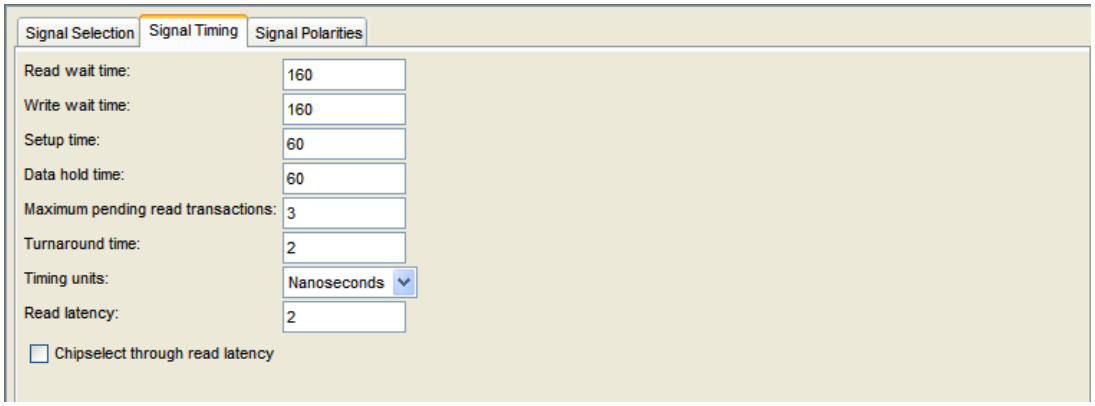

Figure 3-8 Modify CFI Flash[page 6 of 7]

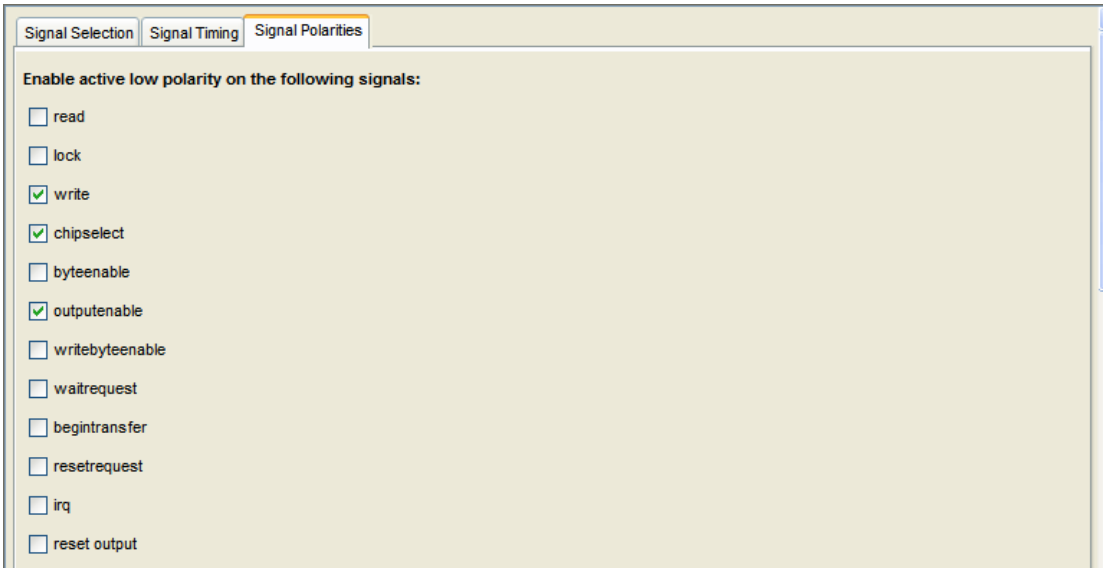

Figure 3-9 Modify CFI Flash[page 7 of 7]

4. Click **Finish** to close **Generic Tri-State Controller** box, and return to the window，then choose **generic_tristate_controller_0** and right-click then choose **Rename**, you can update **generic_tristate_controller_0** to **cfi_flash**. See Figure 3-10.

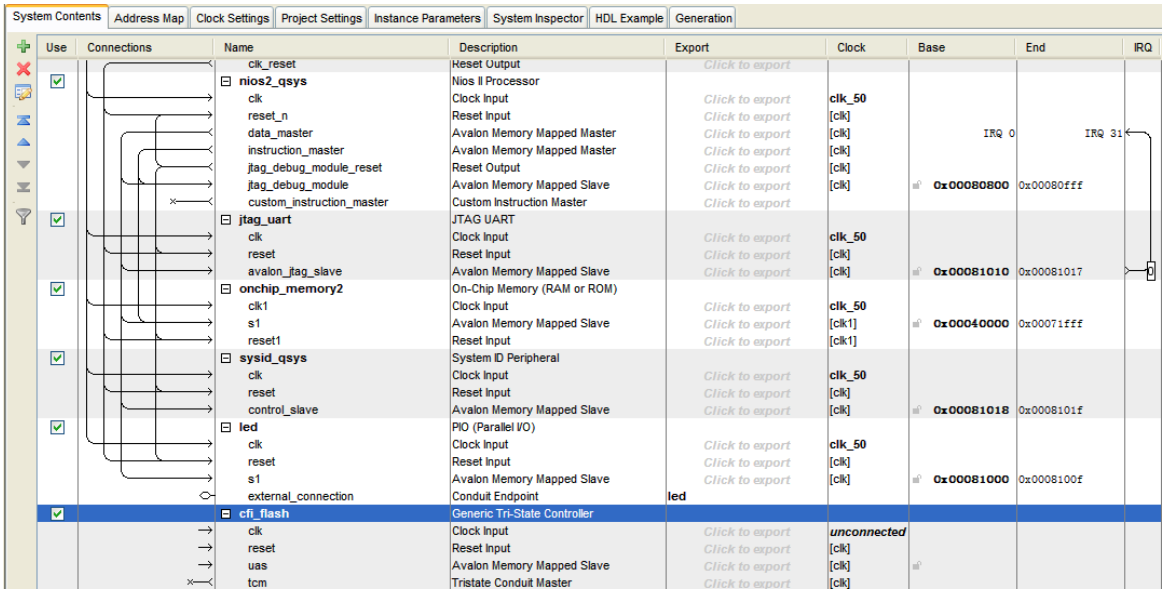

Figure 3-10 Add CFI Flash completely

5. Choose **Library** > **Qsys interconnect** > **Tri-State Components** >**Tri-State Conduit Pin Sharer** to open the Tri-State Conduit Pin Sharer wizard. See Figure 3-11.

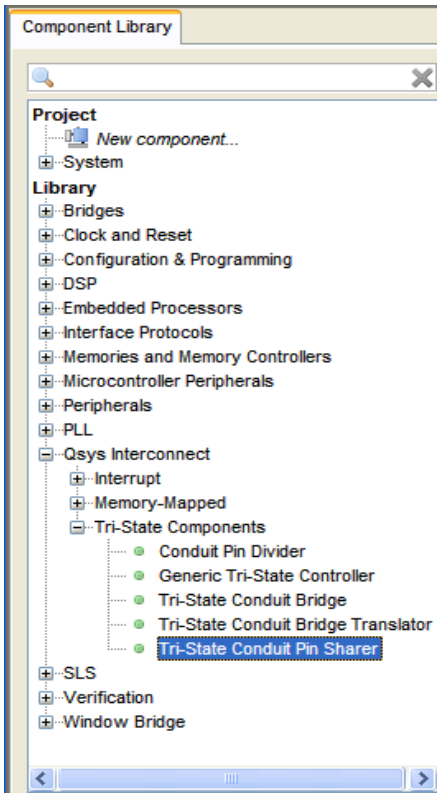

Figure 3-11 Add Tri-State Conduit Pin Sharer

6. Modify **Number of interfaces** as shown in Figure 3-12.

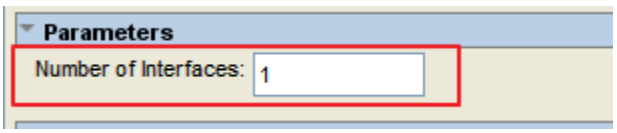

Figure 3-12 Modify Tri-State Conduit Pin Sharer

7. Click **Finish** to close **Tri-State Conduit Pin Sharer** box, then rename it and return to the window as shown in Figure 3-13.

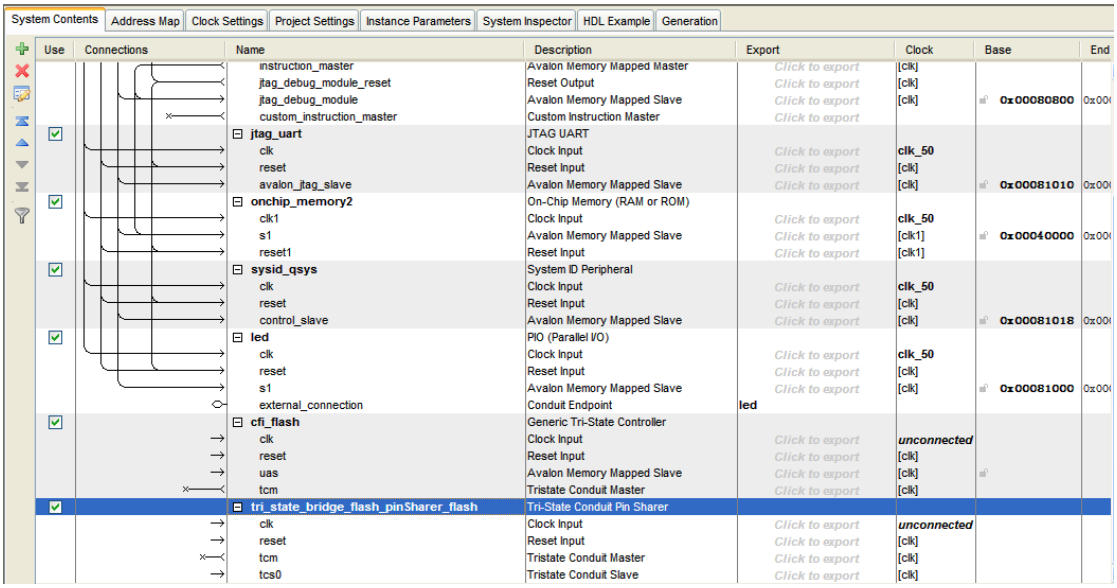

Figure 3-13 Tri-State Conduit Pin Sharer

8. Choose **Library** > **Qsys interconnect** > **Tri-State Components** >**Tri-State Conduit Bridge** to open the Tri-State Conduit Pin Bridge wizard. See Figure 3-14.

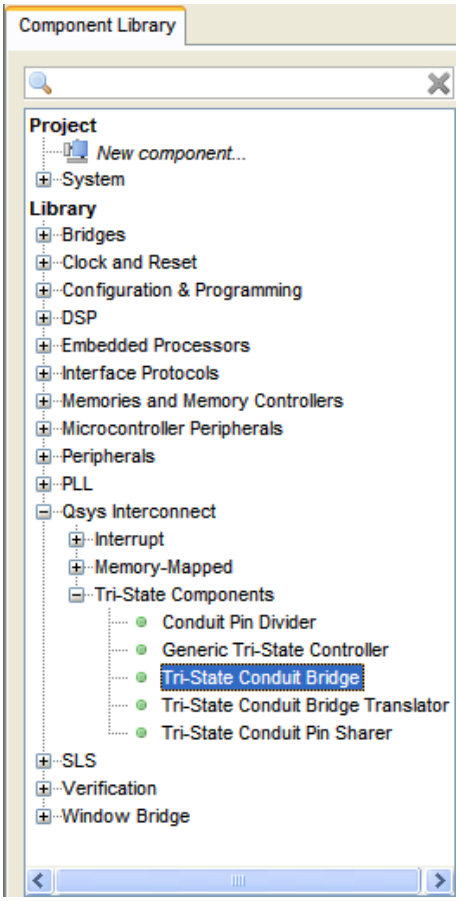

Figure 3-14 Add Tri-State Conduit Bridge

9. Click **Finish** to close **Tri-State Conduit Bridge** box, and return to the window as shown in Figure 3-15.

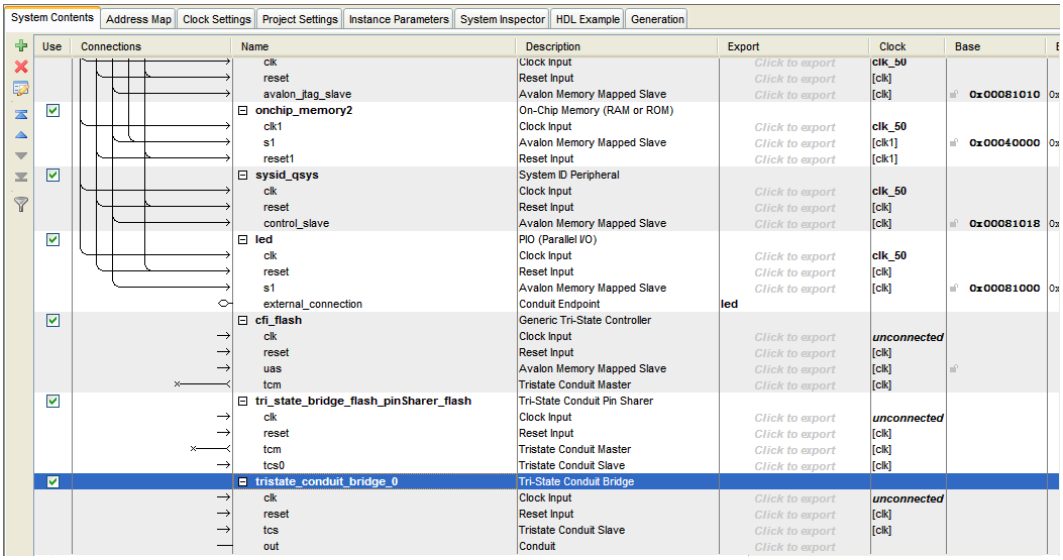

Figure 3-15 Add Tri-State Conduit Bridge Completely

10. Rename **tristate_conduit_bridge_0** as shown in Figure 3-16.

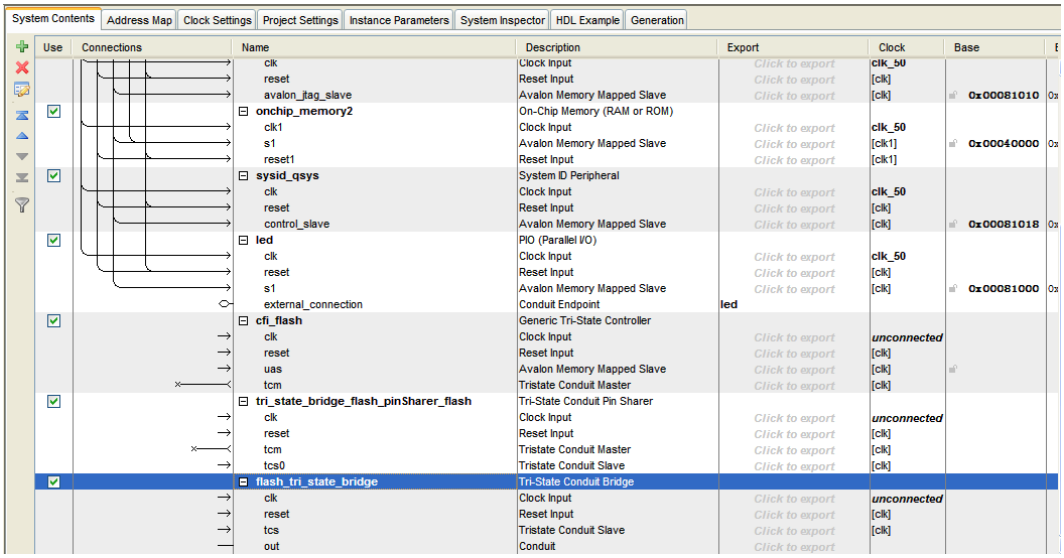

Figure 3-16 Rename Tri-State Conduit Bridge

11. Connect the **clk**、**reset** 、**tcs**、**tcm**、**tcs0** and **uas** as shown in Figure 3-17.

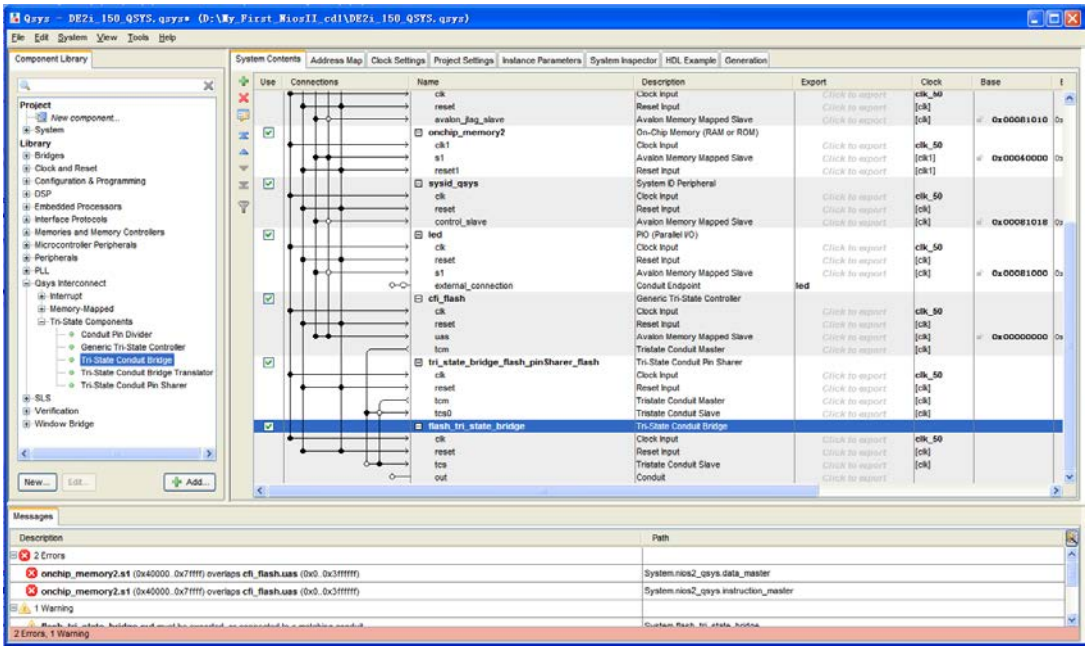

Figure 3-17 Connect signals

12. Click **tri_state_bridge_flash_pinshare_flash** in the component list on the right part to edit the component. Click **Update Interface Table** and rename the **Signal name** as shown in Figure 3-18. Then click **Finish** to return to the window.

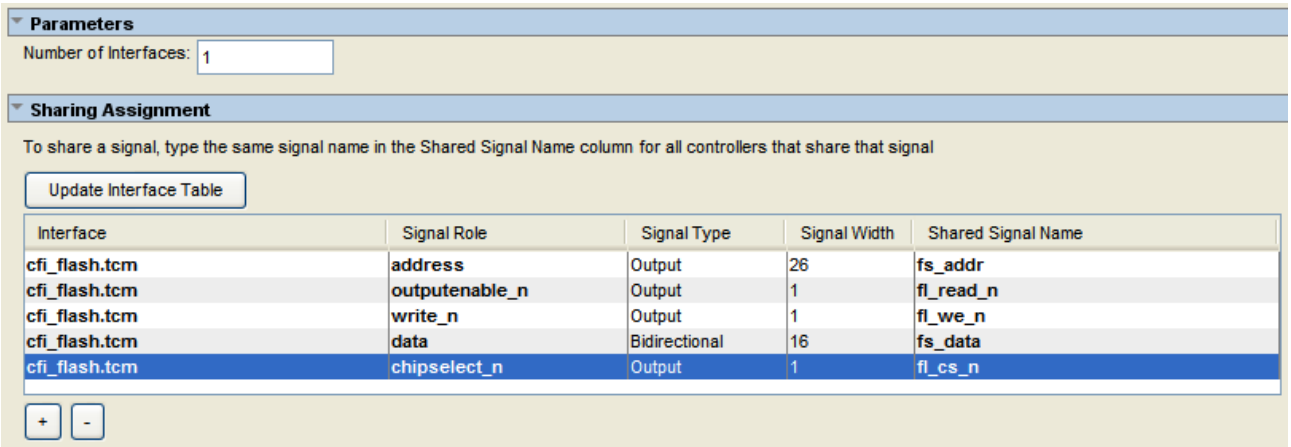

Figure 3-18 Update Tri_state_bridge_flash_pinshare_flash

13. Export **out** and Rename it to **flash_tri_state_bridge_out** as shown in Figure 3-19.

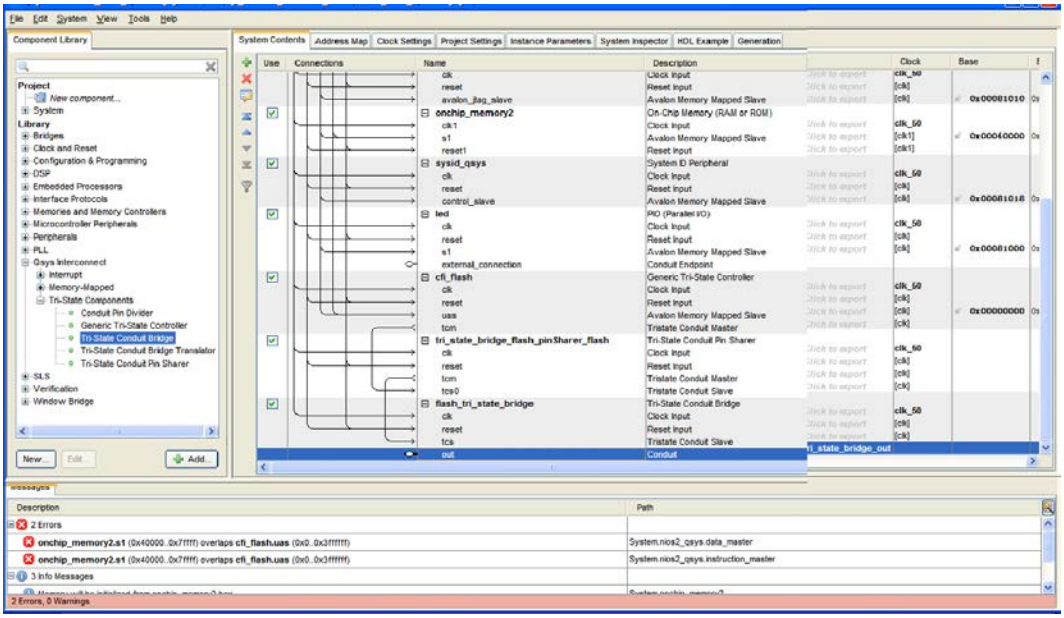

Figure 3-19 Export out and Rename

14. Click **nios2_qsys** in the component list on the right part to edit the component. Change **Reset vector** to cfi_flash as shown in Figure 3-20. Then click **Finish** to return to the window.

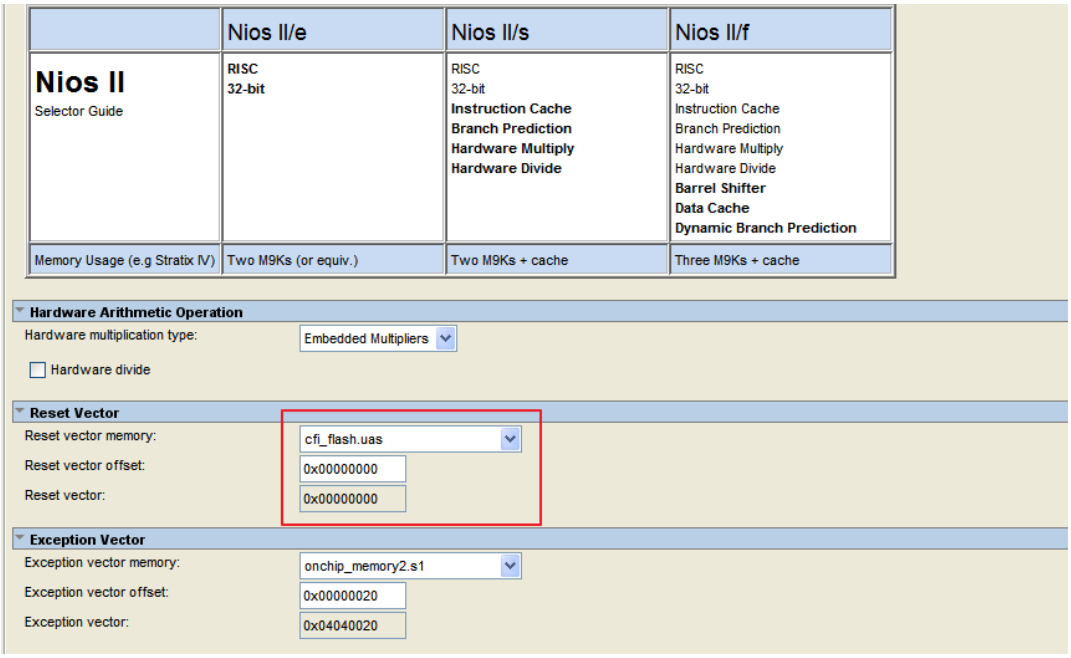

Figure 3-20 Update CPU settings

14. Choose **System** > **Auto-Assign Base Addresses**, then click **Generate** to generate the Qsys as shown in Figure 3-21.

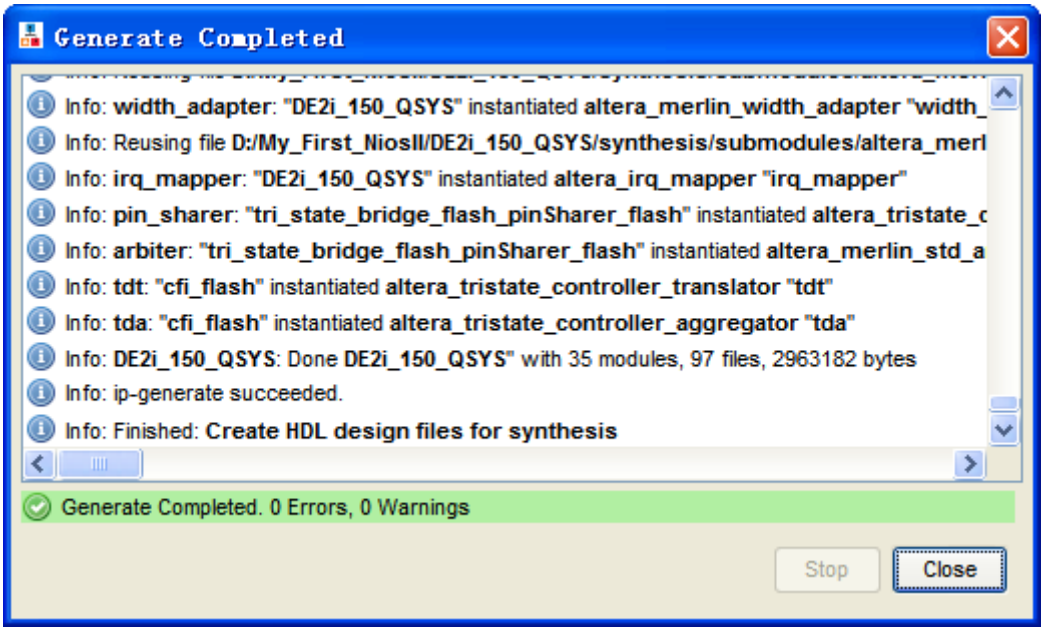

Figure 3-21 Generate Qsys

3.2 Modify the myfirst_niosii.v

1. In the Quartus II, modify **myfirst_niosii.v** as below.

```
module My_First_NiosII(
    ////////////CLOCK/////////
    CLOCK_50,
    ////////////FLASH//////
    FL_CE_N,
    FL_OE_N,
    FL_RESET_N,
    FL_RY,
    FL_WE_N,
    FL_WP_N,
    /////////Data and Address bus shared by Flash ////////
    FS_ADDR,
    FS_DQ,
    //////LED//////
    LED
);
input           CLOCK_50;
output          FL_CE_N;
output          FL_OE_N;
```

```
output          FL_RESET_N;
inout           FL_RY;
output          FL_WE_N;
output          FL_WP_N;
output  [26:0]  FS_ADDR;
inout   [15:0]  FS_DQ;
output  [7:0]   LED;
DE2i_150_QSYS   u0(
                        .clk_clk    (CLOCK_50),
                        .reset_reset_n    (1'b1),
                        .flash_tri_state_bridge_out_fs_addr(FS_ADDR),
                        .flash_tri_state_bridge_out_fl_we_n(FL_WE_N),
                        .flash_tri_state_bridge_out_fl_read_n(FL_OE_N),
                        .flash_tri_state_bridge_out_fs_data(FS_DQ),
                        .flash_tri_state_bridge_out_fl_cs_n(FL_CE_N),
                        .led_export (LED),
                    );
//Flash Config
assign FL_RESET_N = 1'b1;
assign FL_WP_N    = 1'b1;
endmodule
```

2. Re-compilation myfirst_niosii project.

3.3 Re-assign pins

1. re-assign pins. The pins as shown in Table 3-1.

Node Name	Location
CLOCK_50	PIN_AJ16
FL_ADDR[0]	
FL_ADDR[1]	PIN_AB22
FL_ADDR[2]	PIN_AH19
FL_ADDR[3]	PIN_AK19
FL_ADDR[4]	PIN_AJ18
FL_ADDR[5]	PIN_AA18
FL_ADDR[6]	PIN_AH18
FL_ADDR[7]	PIN_AK17
FL_ADDR[8]	PIN_Y20
FL_ADDR[9]	PIN_AK21
FL_ADDR[10]	PIN_AH21
FL_ADDR[11]	PIN_AG21
FL_ADDR[12]	PIN_AG22
FL_ADDR[13]	PIN_AD22
FL_ADDR[14]	PIN_AE24
FL_ADDR[15]	PIN_AD23
FL_ADDR[16]	PIN_AB21
FL_ADDR[17]	PIN_AH17
FL_ADDR[18]	PIN_AE17
FL_ADDR[19]	PIN_AG20
FL_ADDR[20]	PIN_AK20
FL_ADDR[21]	PIN_AE19
FL_ADDR[22]	PIN_AA16
FL_ADDR[23]	PIN_AF15
FL_ADDR[24]	PIN_AG15
FL_ADDR[25]	PIN_Y17
FL_ADDR[26]	PIN_AB16
FL_CE_N	PIN_AG19
FL_DQ[0]	PIN_AK29
FL_DQ[1]	PIN_AE23
FL_DQ[2]	PIN_AH24
FL_DQ[3]	PIN_AH23
FL_DQ[4]	PIN_AA21

FL_DQ[5]	PIN_AE20
FL_DQ[6]	PIN_Y19
FL_DQ[7]	PIN_AA17
FL_DQ[8]	PIN_AB17
FL_DQ[9]	PIN_Y18
FL_DQ[10]	PIN_AA20
FL_DQ[11]	PIN_AE21
FL_DQ[12]	PIN_AH22
FL_DQ[13]	PIN_AJ24
FL_DQ[14]	PIN_AE22
FL_DQ[15]	PIN_AK28
FL_OE_N	PIN_AJ19
FL_RESET_N	PIN_AG18
FL_RY	PIN_AF19
FL_WE_N	PIN_AG17
FL_WP_N	PIN_AK18
LED[0]	PIN_AA25
LED[1]	PIN_AB25
LED[2]	PIN_F27
LED[3]	PIN_F26
LED[4]	PIN_W26
LED[5]	PIN_Y22
LED[6]	PIN_Y25
LED[7]	PIN_AA22

2.Re-compilation My_First_NiosII project and re-download My_First_NiosII.sof to the development board.

3.4 Re-Configure Nios II BSP Editor

1. In the NIOS II SBT for Eclipse, right-click **My_First_Niosii_bsp** and choose **Nios II > Generate BSP**. See Figure 3-22.

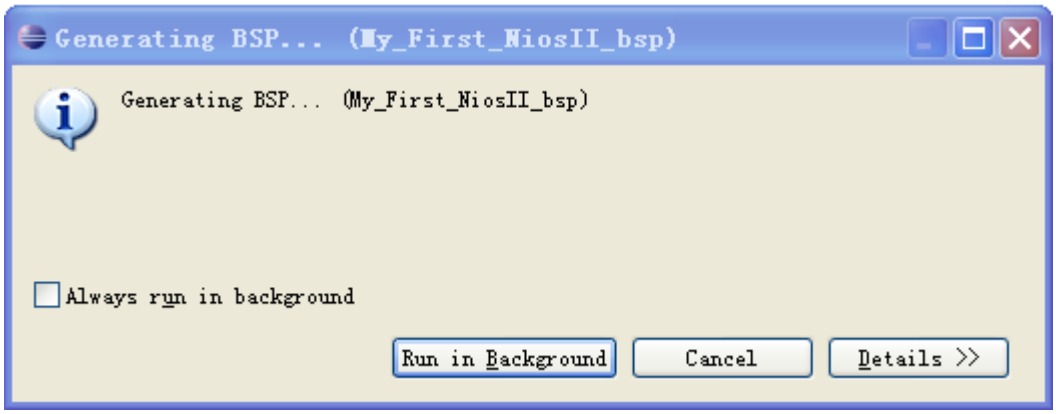

Figure 3-22 Generate BSP

2. Re-build My_First_NiosII project.

3.5 Programming the CFI Flash

1. Choose **Nios II**> **Flash Programmer** to open Nios II Flash Programmer box. See Figure 3-23.

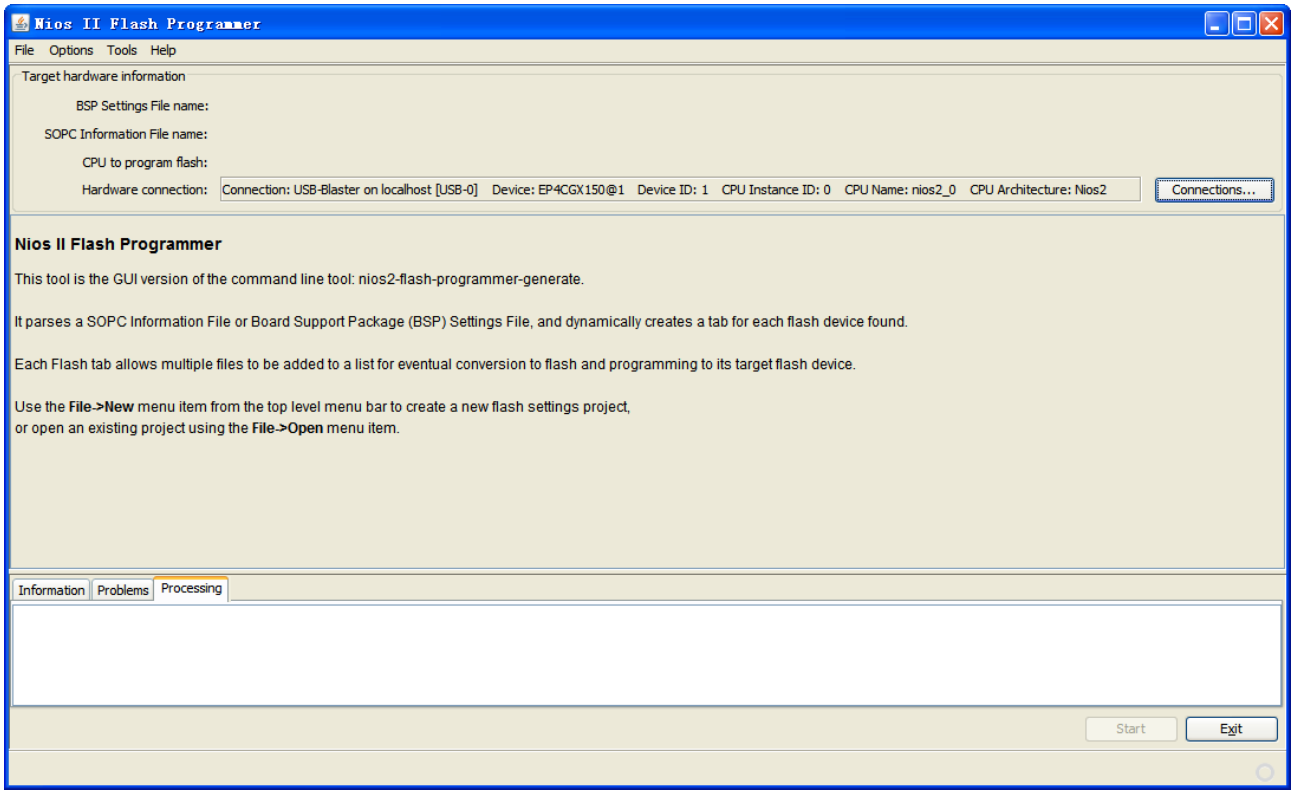

Figure 3-23 Flash Programmer Box

2.Choose **File** > **New** to open New Flash Programmer Setting File box as shown in Figure 3-24.

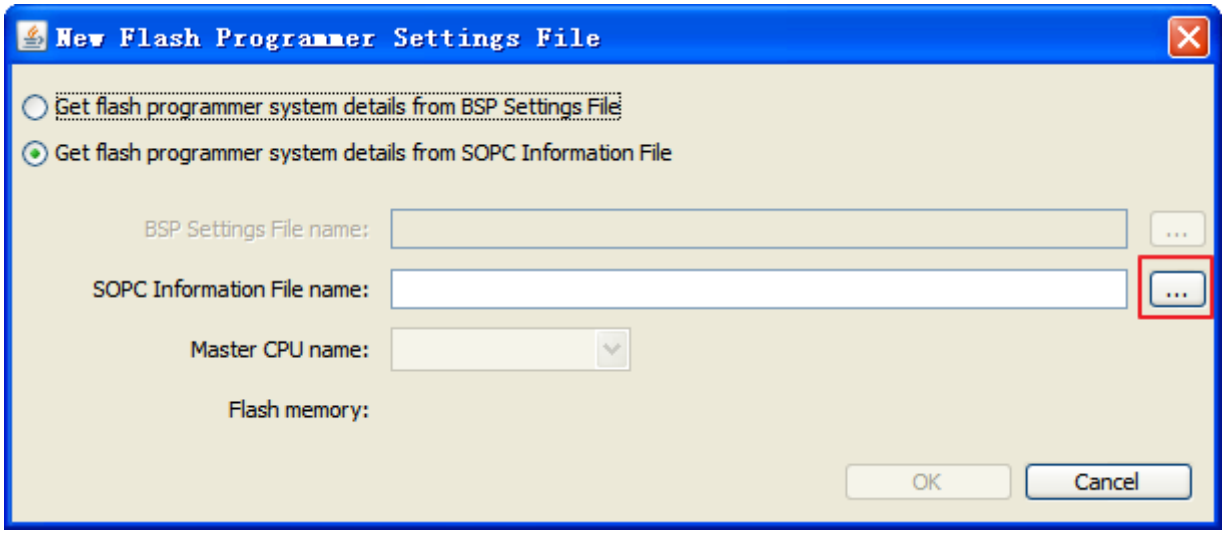

Figure 3-24 New Flash Programmer Settings File

3.Click the botton which is mark in Figure 3-24,and open Select SOPC Information Design File box as shown in Figure 3-25.

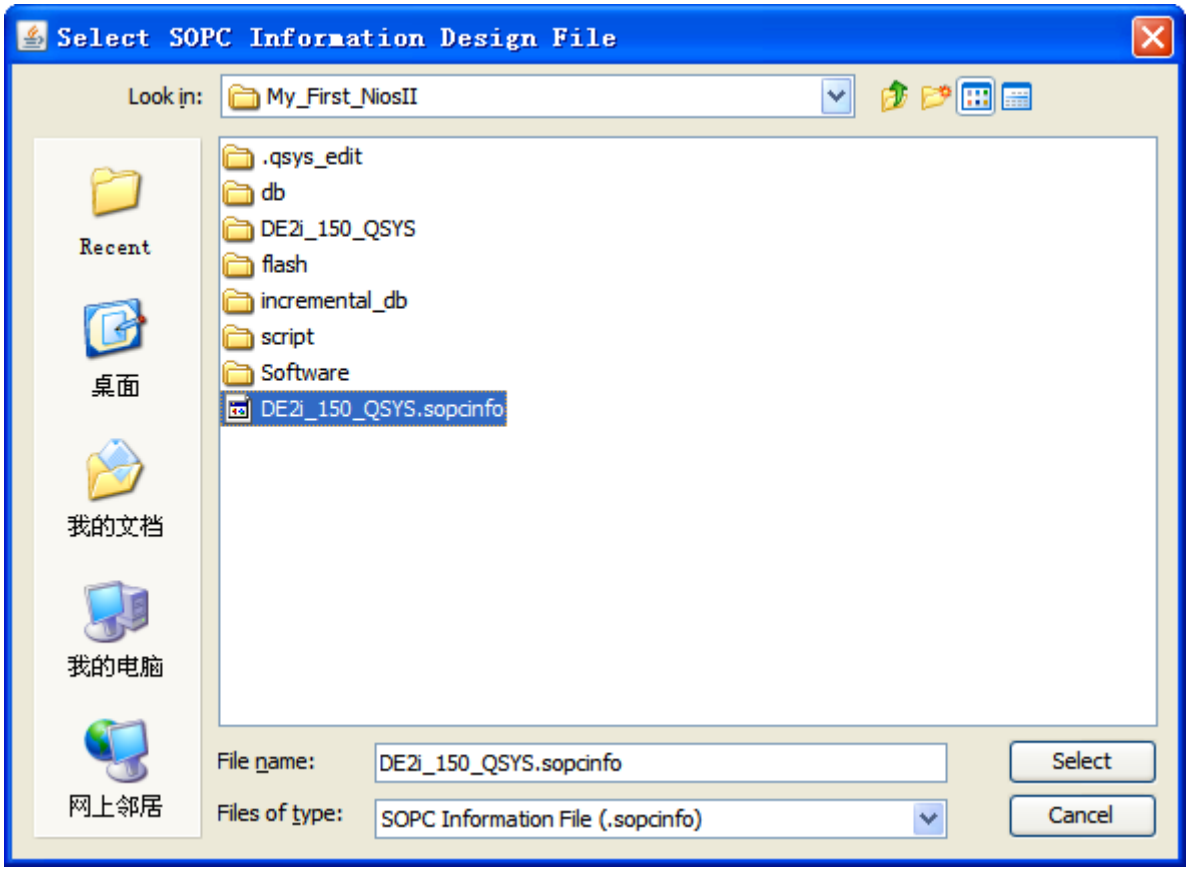

Figure 3-25 Select SOPC Information Design File[1]

4.Select “DE2I_150_QSYS.sopcinfo” file, and click Select to back as shown in Figure 3-26.

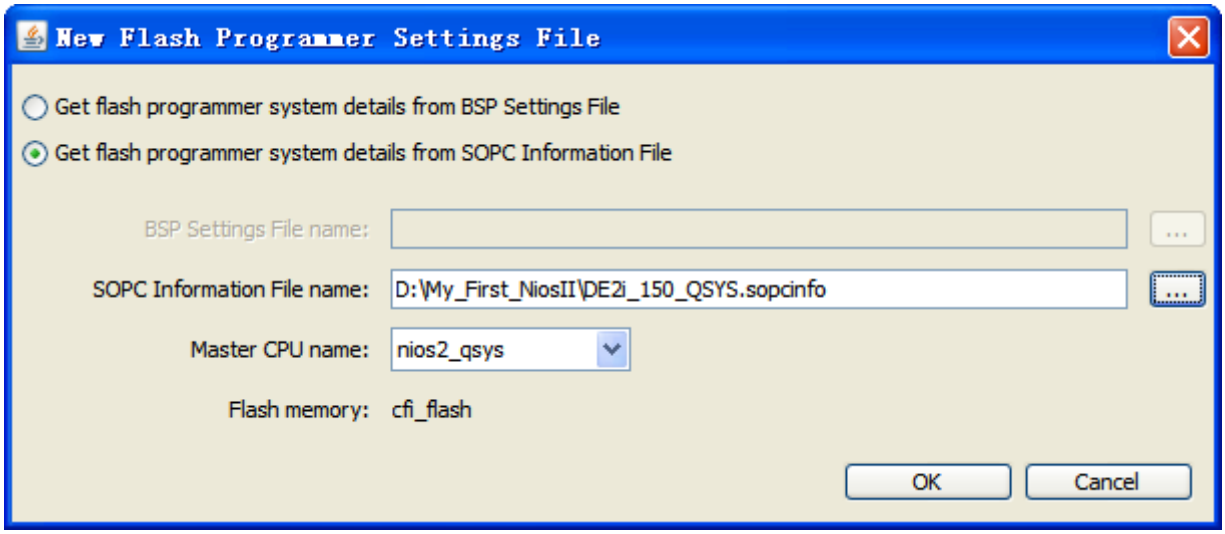

Figure 3-26 Select SOPC Information Design File[2]

5.Click ok to back the Nios II Flash Programmer box as shown in Figure 3-27,and click add.. to select file to Flash Conversion as shown in Figure 3-28.

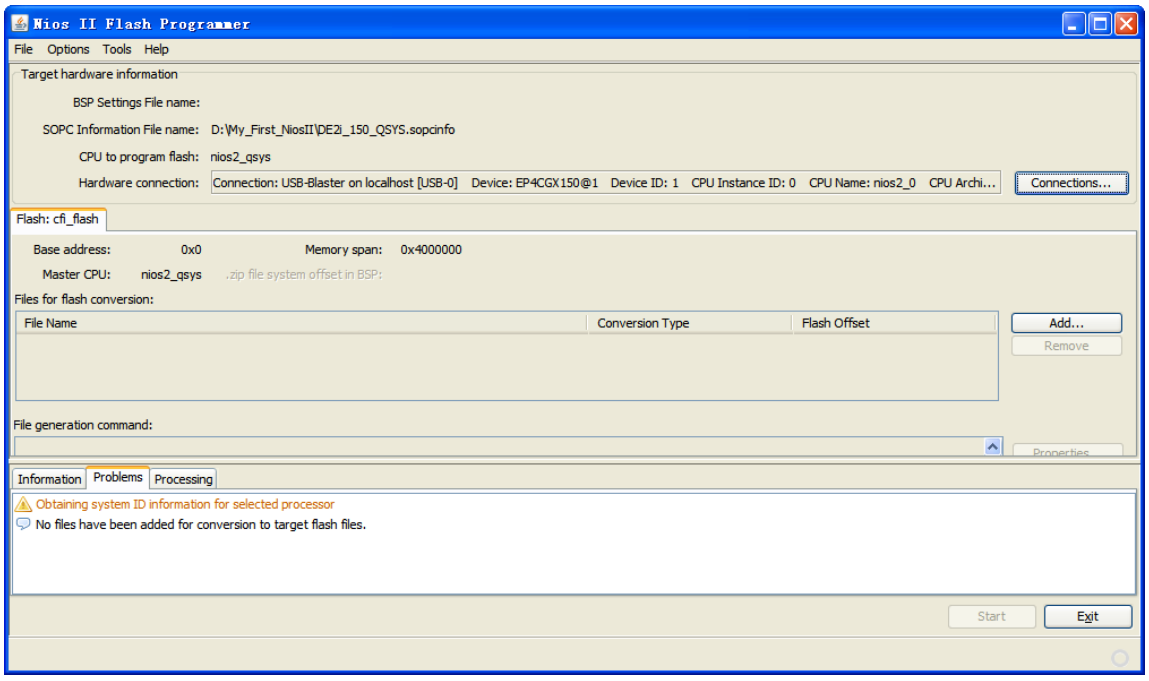

Figure 3-27 Back Nios II Flash Programmer

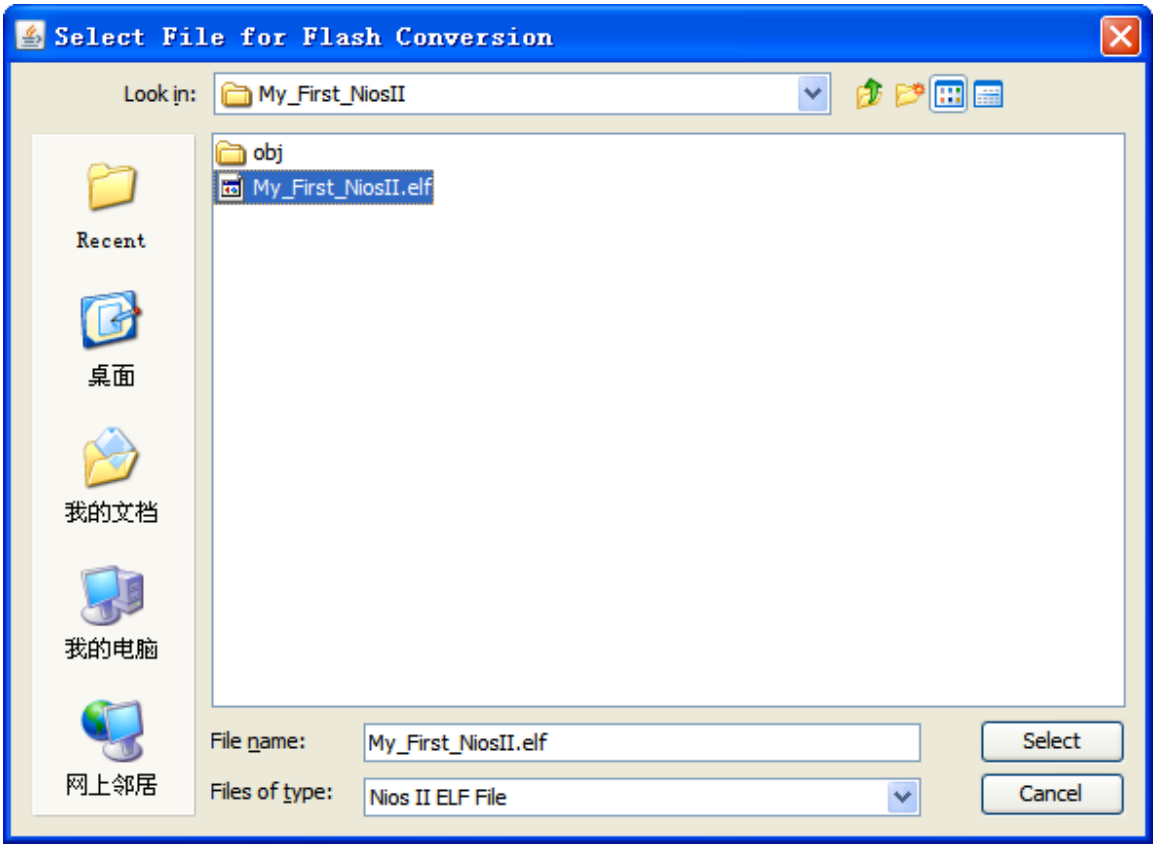

Figure 3-28 Add File for Flash Conversion

6.Add My_First_NiosII.elf to Flash Conversion and click Start to Program Flash as shown in Figure 3-29.

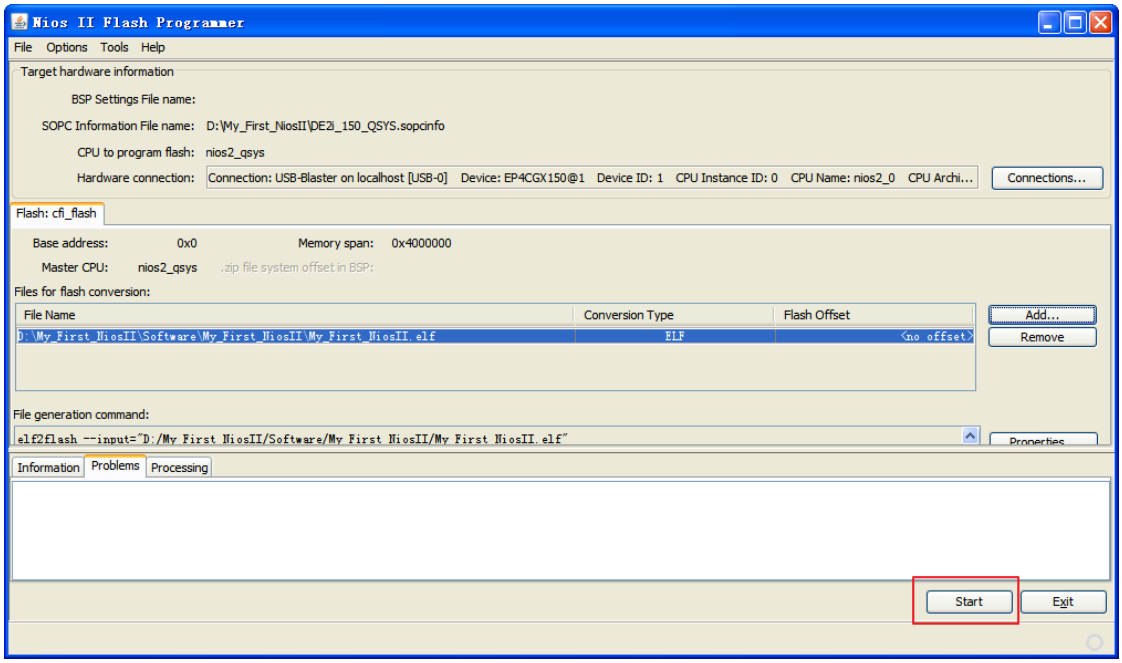

Figure 3-29 Start Program Flash

7.When program flash completely,the console tab displays as shown in Figure 3-30.

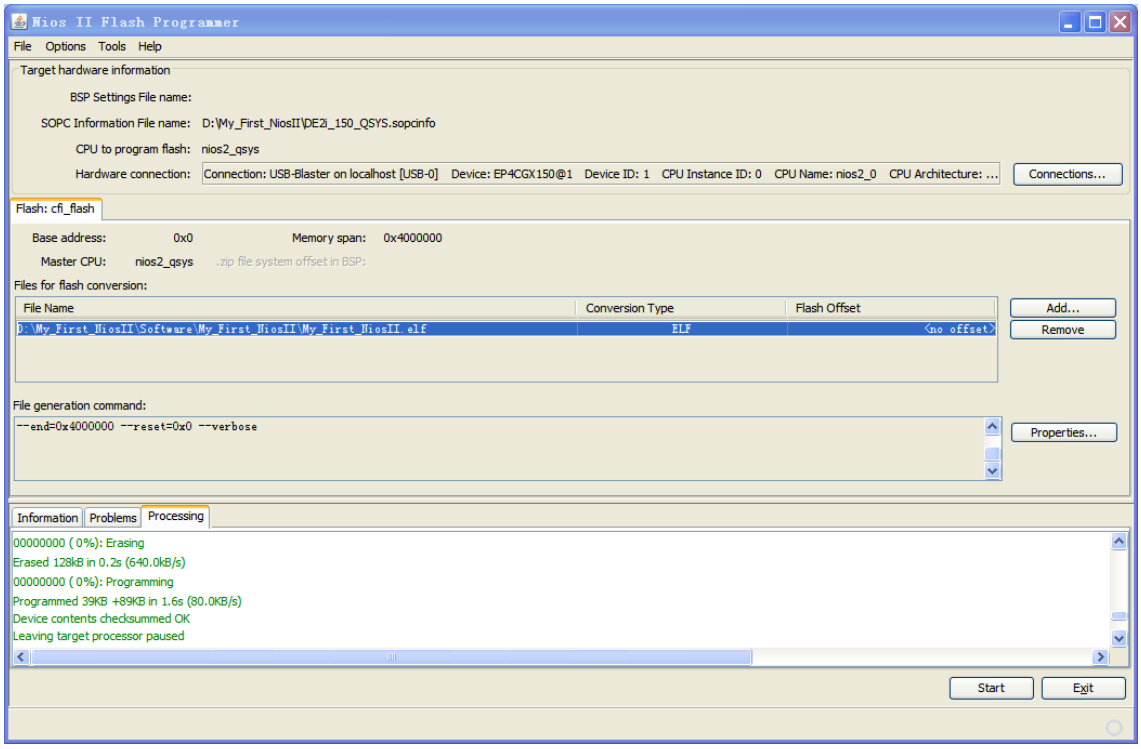

Figure 3-30 Program Flash completel

8. Restart power on the board. Download My_First_NiosII.sof of your project “My_First_NiosII”. You will see that the led blinks.