LG0623
Lab Guide
RTG4 FPGA Fabric

Microsemi Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA
Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996
Email: sales.support@microsemi.com
www.microsemi.com

About Microsemi

50200623. 8.0 9/21

Contents

Figures

Tables

1 Revision History

The revision history describes the changes that were implemented in the document. The changes are listed by revision, starting with the current publication.

1.1 Revision 8.0

The following is a summary of the changes made in this revision.

- Updated the document for Libero SoC v2021.2.
- Updated section Demo Design, page 3 and Extracting the Source Files, page 3.
- Replaced Figure 11, page 8, Figure 12, page 9, Figure 15, page 10, Figure 18, page 12, Figure 39, page 23, and Figure 41, page 24.
- Updated Table 3, page 11.

1.2 Revision 7.0

The following is a summary of the changes made in this revision.

- Added Step 8: Programming the Device, page 28.
- Added Appendix 1: Programming the Device Using FlashPro Express, page 29.
- Added Appendix 2: Running the TCL Script, page 32.
- Removed the references to Libero version numbers.

1.3 Revision 6.0

Updated the document for Libero v11.9 SP1 software release.

1.4 Revision 5.0

Updated the document for Libero v11.8 SP2 software release.

1.5 Revision 4.0

Updated the document for Libero v11.8 software release.

1.6 Revision 3.0

Revision 3.0 was updated for Libero SoC 11.7 SP1 software release change in the document (SAR 82451).

1.7 Revision 2.0

Revision 2.0 was updated for Libero SoC 11.7 software release change in the document.

1.8 Revision 1.0

Revision 1.0 was the first release of this document.

2 RTG4 FPGA Fabric

This lab guide demonstrates how to implement a basic RTG4™ Field Programmable Gate Array (FPGA) fabric design using SmartDesign. The design drives LEDs on the RTG4 Development Kit board with different patterns based on the state of SW7, SW2, and SW1 switches (refer to the following table).

Table 1 • **State of SW7, SW2, and SW1 Switches**

Reset Switch (SW7)	User Switch (SW2)	User Switch (SW1)	LED [8:1] Behavior
Pressed	Do not care	Do not care	Off
Released	Released	Released	Toggle LED [6:5] and LED [4:3] Red and Green LEDs
Released	Released	Pressed	Shift left
Released	Pressed	Released	Shift right
Released	Pressed	Pressed	Toggle LED[8:7] and LED[2:1] Orange and Yellow LEDs

After completing this lab guide, you will be familiar with the following:

- Creating a Libero® System-on-Chip (SoC) project
- Implementing an RTG4 fabric design with SmartDesign
- Simulating the design
- Making pin assignments, running layout, and programming the RTG4 silicon

2.1 RTG4 Device Components

This lab guide uses the RTG4 FPGA fabric and fabric clock conditioning circuit (CCC).

2.2 Design Requirements

The following table lists the hardware and software design requirements to run the design.

Table 2 • **Design Requirements**

Requirement	Version
Hardware	
RTG4 FPGA Development Kit: • USB 2.0 cable • 12 V, 5A AC power adapter and cords	RT4G150-CB1657PROTO FPGA
Host PC or laptop	64-bit Windows 7 and 10
Software	
Libero SoC	**Note:** Refer to the `readme.txt` file provided in the design files for the software versions used with this reference design.
FlashPro Express	

Note: Libero SmartDesign and configuration screen shots shown in this guide are for illustration purpose only. Open the Libero design to see the latest updates.

2.3 Prerequisites

Download and install Libero SoC (as indicated in the website for this design) on the host PC from the following location: *https://www.microsemi.com/product-directory/design-resources/1750-libero-soc*

2.4 Demo Design

The demo design files are available for download at:

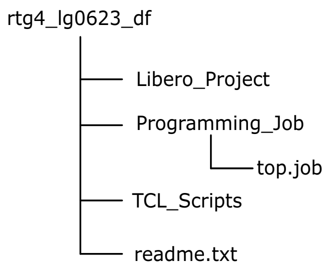
http://soc.microsemi.com/download/rsc/?f=rtg4_lg0623_df

The demo design files include:

- Source_files
- Libero_Project
- Programming_Job
- TCL_Scripts
- readme.txt

The following figure shows the top-level structure of the design files. For more information, refer to the `readme.txt` file.

Figure 1 • **Demo Design Files Top-Level Structure**



2.5 Extracting the Source Files

1. Download the design files from *http://soc.microsemi.com/download/rsc/?f=rtg4_lg0623_df*
2. Extract `rtg4_lg0623_df.zip` to extract the required lab files to the *<C:*, *D:*, or *E:>\Microsemiprj folder on the HDD of the PC*.
3. Ensure that the `rtg4_lg0623_df` folder contains a `Source_files` sub-folder, which is extracted.

2.6 Step 1: Creating the Design

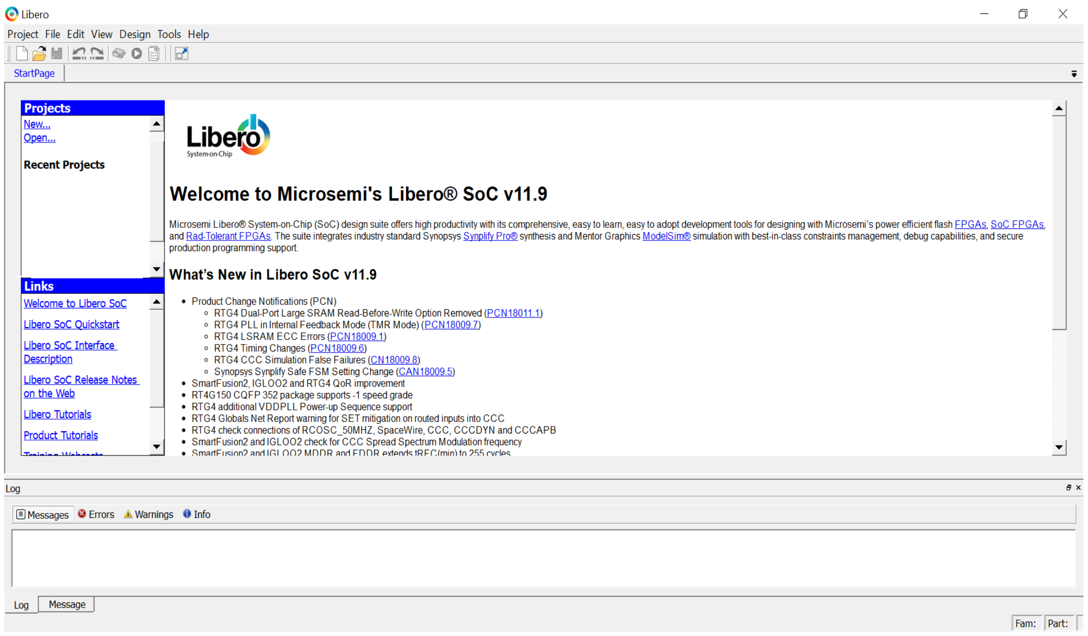
The following steps describe how to create the fabric design using SmartDesign by using the source files provided in the Programming_Job folder.

2.6.1 Launching Libero SoC Software

Follow the instructions to launch the Libero SoC software:

1. Go to **Start > Programs > Microsemi > Libero SoC v(xx.x)** or double-click the shortcut on the desktop to open the Libero SoC Project Manager.
2. Create a new project using one of the following options:
   - Click **New** on the Start Page tab, as shown in the following figure.
   - Go to **Project** > **New Project** from the Libero SoC menu.

Figure 2 • **Libero SoC Project Manager**



Welcome to Microsemi's Libero® SoC v11.9

Microsemi Libero® System-on-Chip (SoC) design suite offers high productivity with its comprehensive, easy to learn, easy to adopt development tools for designing with Microsemi's power efficient flash FPGAs, SoC FPGAs, and Rad-Tolerant FPGAs. The suite integrates industry standard Synopsys Synplify Pro® synthesis and Mentor Graphics ModelSim® simulation with best-in-class constraints management, debug capabilities, and secure production programming support.

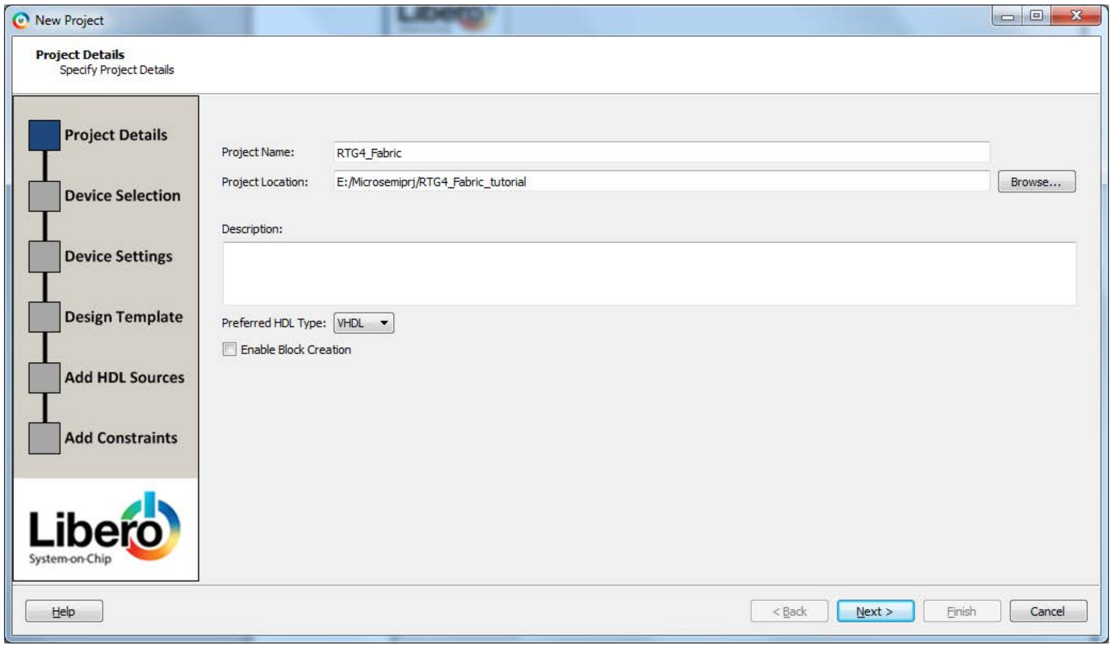
What's New in Libero SoC v11.9

- Product Change Notifications (PCN)
  - RTG4 Dual-Port Large SRAM Read-Before-Write Option Removed (PCN18011.1)
  - RTG4 PLL in Internal Feedback Mode (TMR Mode) (PCN18009.7)
  - RTG4 LSRAM ECC Errors (PCN18009.1)
  - RTG4 Timing Changes (PCN18009.6)
  - RTG4 CCC Simulation False Failures (CN18009.8)
  - Synopsys Synplify Safe FSM Setting Change (CAN18009.5)
- SmartFusion2, IGLOO2 and RTG4 QoR improvement
- RT4G150 CQFP 352 package supports -1 speed grade
- RTG4 additional VDDPLL Power-up Sequence support
- RTG4 Globals Net Report warning for SET mitigation on routed inputs into CCC
- RTG4 check connections of RCOSC_50MHZ, SpaceWire, CCC, CCCDYN and CCCAPB
- SmartFusion2 and IGLOO2 check for CCC Spread Spectrum Modulation frequency
- SmartFusion2 and IGLOO2 MDDR and FDDR extends tREC(min) to 255 cycles

3. Enter the following information in the **Project Details** page of the **New Project** dialog box as shown in the following figure:
   - **Project Name**: RTG4_Fabric
   - **Project Location**: *<C:, D:*, or *E:>\Microsemiprj\RTG4_Fabric_tutorial* (depending on where you extracted the source files)
   - **Preferred HDL Type**: Verilog or VHDL
   - **Enable Block Creation**: Leave Unchecked (default)

4. Click **Next**.

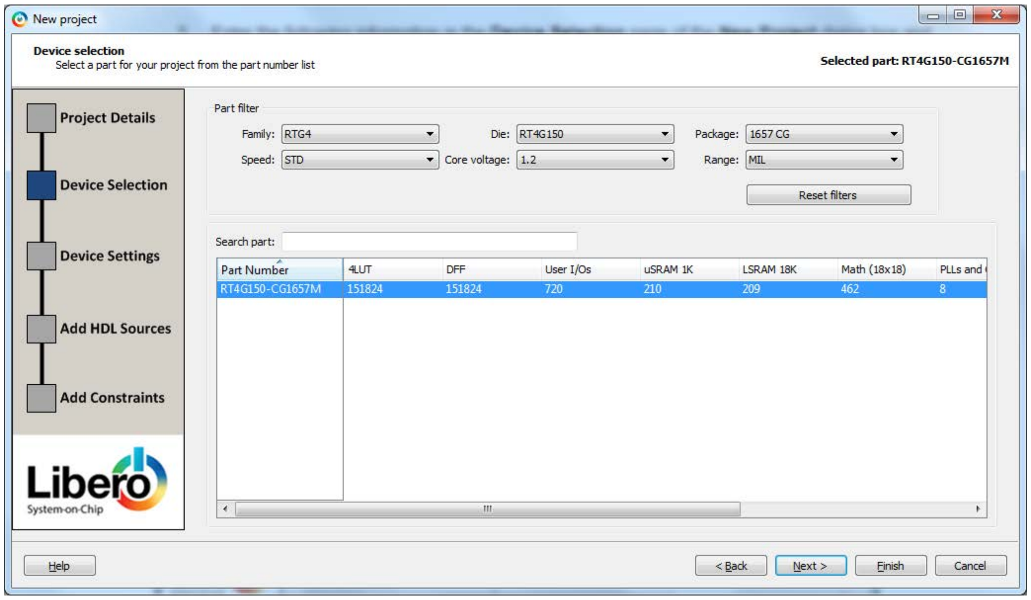
Figure 3 • **Project Details**



Project Name: RTG4_Fabric
Project Location: E:/Microsemiprj/RTG4_Fabric_tutorial
Description:
Preferred HDL Type: VHDL
Enable Block Creation

5. Enter the following information in the **Device Selection** page of the **New Project** dialog box and click **Next**:
   - **Family**: RTG4
   - **Die**: RT4G150
   - **Package**: 1657 CG
   - **Speed**: STD
   - **Core Voltage**: 1.2
   - **Range**: MIL

Figure 4 • **Device Selection Settings**



Selected part: RT4G150-CG1657M

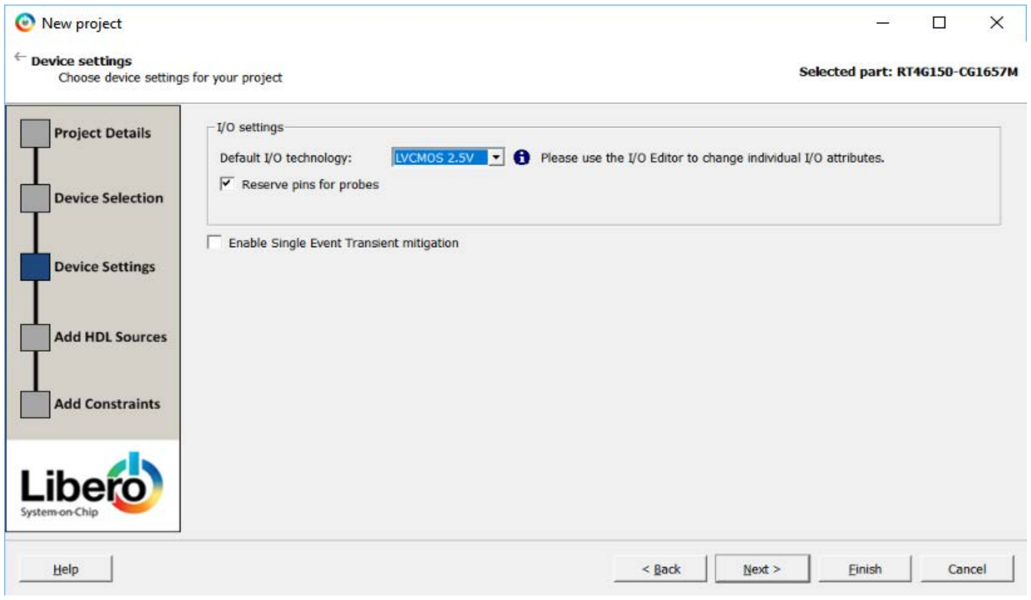
Family: RTG4 Die: RT4G150 Package: 1657 CG
Speed: STD Core voltage: 1.2 Range: MIL

Search part:

Part Number	4LUT	DFF	User I/Os	uSRAM 1K	LSRAM 18K	Math (18x18)	PLLs and
RT4G150-CG1657M	151824	151824	720	210	209	462	8

6. Enter the following information in the **Device Settings** page of the **New Project** dialog box and click **Next**:
   - **Default I/O Technology**: LVCMOS 2.5 V (default)
   - **Reserve Pins for Probes**: Check (default)
   - **Reserve Pins for SPI**: Leave Unchecked (default)

Figure 5 • **Device I/O Settings**



New project

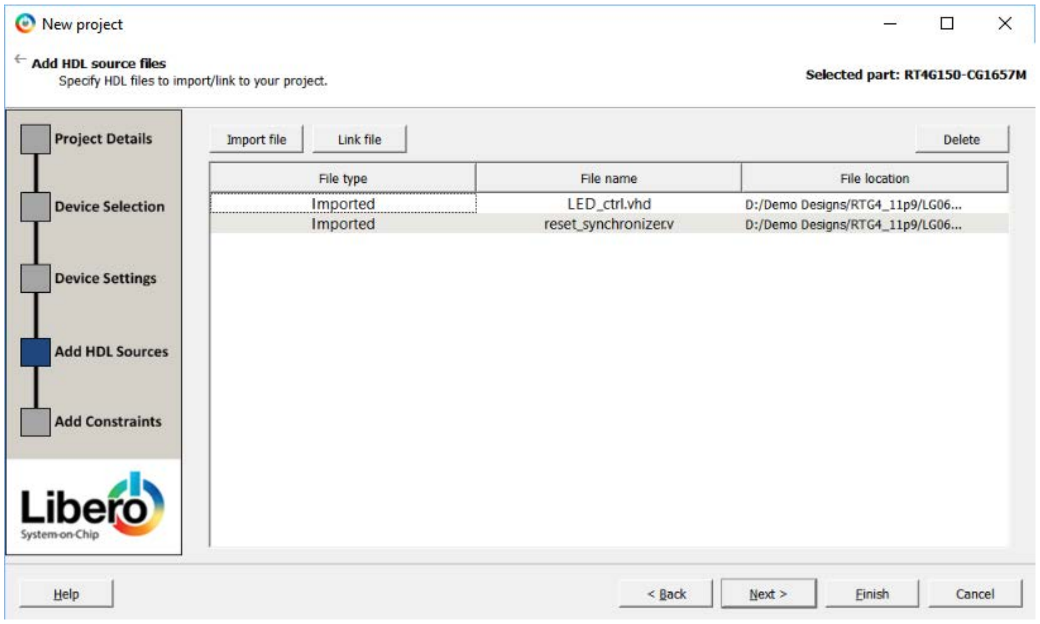
Device settings
Choose device settings for your project

Selected part: RT4G150-CG1657M

I/O settings
Default I/O technology: LVCMOS 2.5V
Please use the I/O Editor to change individual I/O attributes.
[✓] Reserve pins for probes
[ ] Enable Single Event Transient mitigation

The following figure shows the **Add HDL Source Files** page.

Figure 6 • **Imported HDL Source Files**



New project

Add HDL source files
Specify HDL files to import/link to your project.

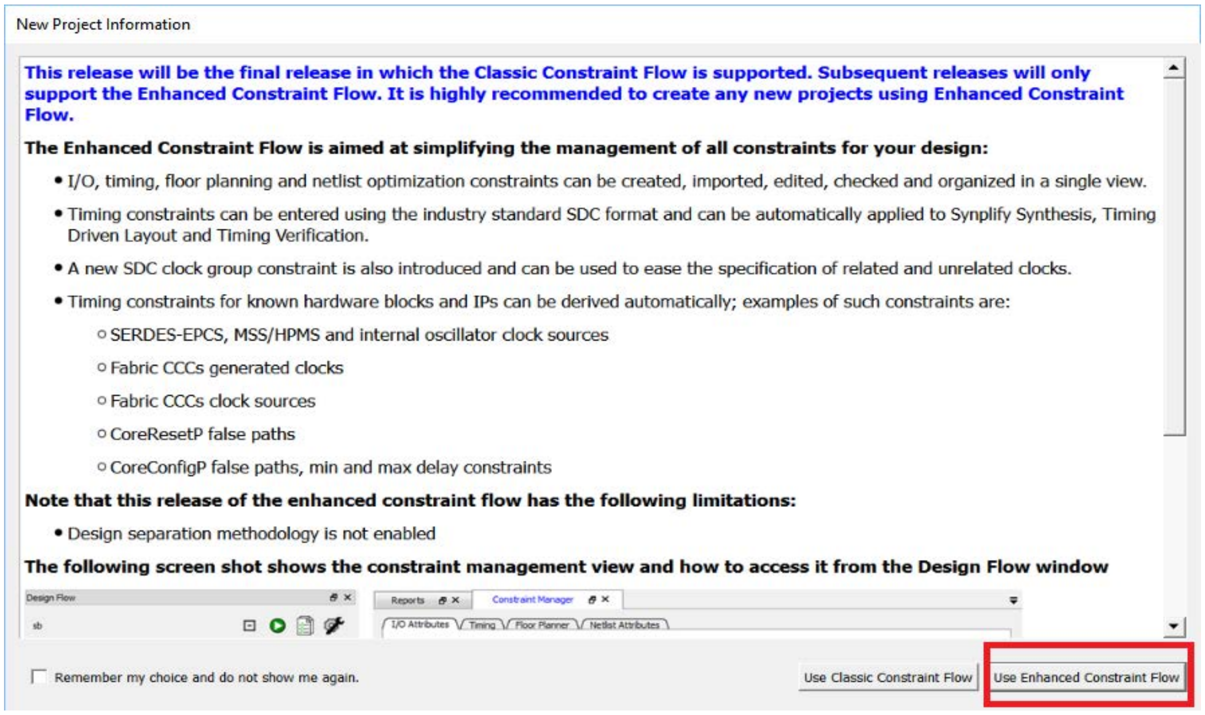
Selected part: RT4G150-CG1657M

File type	File name	File location
Imported	LED_ctrl.vhd	D:/Demo Designs/RTG4_11p9/LG06...
Imported	reset_synchronizer.v	D:/Demo Designs/RTG4_11p9/LG06...

7. Click **Import File** to import the provided VHDL and Verilog source file into the project.
8. Enter the following information in the Import Files dialog box and click **Open**:
   - **Files Type**: HDL Source Files (*.vhd *.v *.h)
   - **File Name**: `LED_ctrl.vhd` (for VHDL projects) or `LED_ctrl.v` (for Verilog projects)
   - **File Location**: <C:, D:, or E>\Microsemiprj\rtg4_lg0623_df\Source_files
9. Click **Finish**.

10. Click **Use Enhanced Constraint Flow** as shown in the following figure.

Figure 7 • **Use Enhanced Constraint Flow**



New Project Information

This release will be the final release in which the Classic Constraint Flow is supported. Subsequent releases will only support the Enhanced Constraint Flow. It is highly recommended to create any new projects using Enhanced Constraint Flow.

The Enhanced Constraint Flow is aimed at simplifying the management of all constraints for your design:

- I/O, timing, floor planning and netlist optimization constraints can be created, imported, edited, checked and organized in a single view.
- Timing constraints can be entered using the industry standard SDC format and can be automatically applied to Synplify Synthesis, Timing Driven Layout and Timing Verification.
- A new SDC clock group constraint is also introduced and can be used to ease the specification of related and unrelated clocks.
- Timing constraints for known hardware blocks and IPs can be derived automatically; examples of such constraints are:
  - SERDES-EPCS, MSS/HPMS and internal oscillator clock sources
  - Fabric CCCs generated clocks
  - Fabric CCCs clock sources
  - CoreResetP false paths
  - CoreConfigP false paths, min and max delay constraints

Note that this release of the enhanced constraint flow has the following limitations:

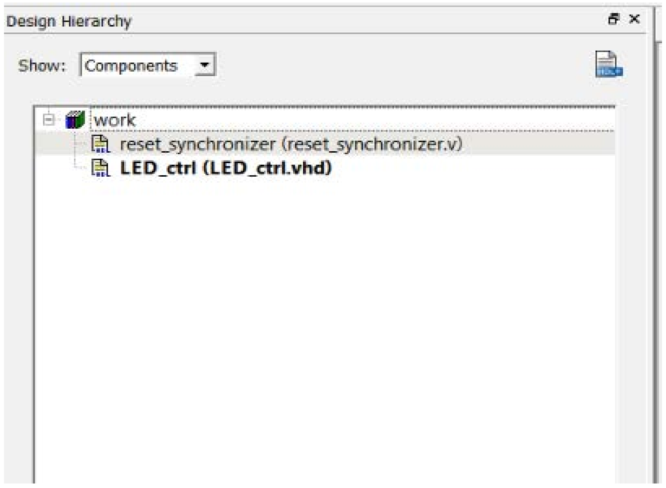
- Design separation methodology is not enabled

The following screen shot shows the constraint management view and how to access it from the Design Flow window

Design Flow
sb
Reports
Constraint Manager
I/O Attributes
Timing
Floor Planner
Netlist Attributes
Remember my choice and do not show me again.
Use Classic Constraint Flow
Use Enhanced Constraint Flow

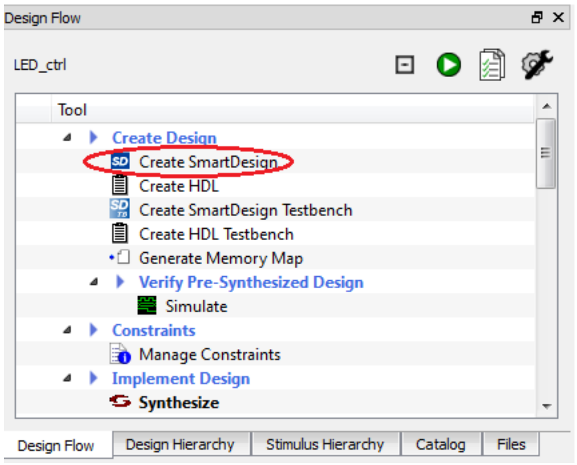
The `LED_ctrl (LED_ctrl.vhd)` file is displayed on the Libero Design Hierarchy tab as shown in the following figure.

Figure 8 • **Design Hierarchy Tab with Imported Files (VHDL shown)**



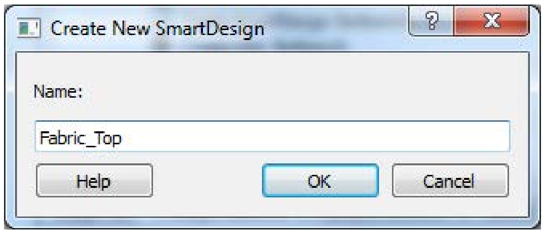
11. Open the SmartDesign canvas by selecting **File > New > SmartDesign** or by double-clicking **Create SmartDesign** under Create Design in the Design Flow tab.

Figure 9 • **Opening the SmartDesign Canvas**



12. Enter **Fabric_Top** in the **Create New SmartDesign** dialog box then click **OK**. For Verilog designs, the name is case sensitive.

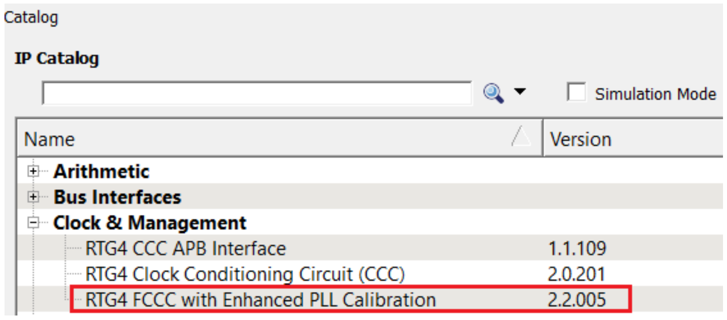
Figure 10 • **Entering SmartDesign Name**



Name: Fabric_Top

13. Drag-and-drop the `LED_ctrl` component from the Design Hierarchy tab in the SmartDesign canvas. This design uses a Fabric CCC to generate the 500 kHz internal clock. The CCC reference clock is the 50 MHz crystal oscillator on the RTG4 Development Kit. In the following steps, configure the CCC to output a 500 kHz clock.
14. Expand **Clock & Management** in the IP catalog.

Figure 11 • **Clock & Management Category of the Libero SoC IP Catalog**

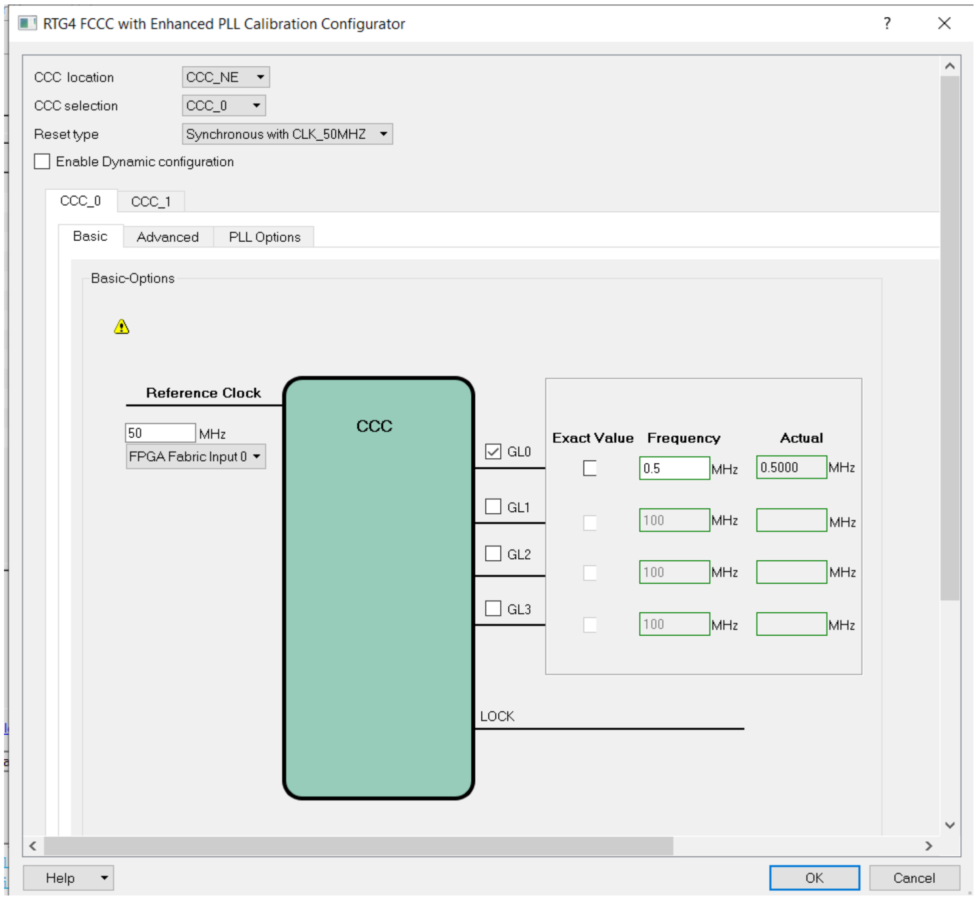


Name	Version
Arithmetic	
Bus Interfaces	
Clock & Management	
RTG4 CCC APB Interface	1.1.109
RTG4 Clock Conditioning Circuit (CCC)	2.0.201
RTG4 FCCC with Enhanced PLL Calibration	2.2.005

15. Drag-and-drop an instance of the RTG4FCCCECALIB component in the SmartDesign canvas.
16. Double-click the **RTGFCCCECALIB_C0** component in the SmartDesign canvas to open the RTG4FCCCECALIB Configurator.

17. Select the **Basic** tab in the FAB CCC configurator. Enter the following:
    - Enter **Reference Clock** as 50 MHz
    - Select **FPGA Fabric Input 0** from the drop-down menu
    - Select the **GL0** check box
    - Enter **Frequency** as 0.5 MHz

Figure 12 • **Configuring the RTG4 Fabric CCC**


RTG4 FCCC with Enhanced PLL Calibration Configurator

CCC location: CCC_NE
CCC selection: CCC_0
Reset type: Synchronous with CLK_50MHZ
☐ Enable Dynamic configuration

CCC_0 | CCC_1
Basic | Advanced | PLL Options

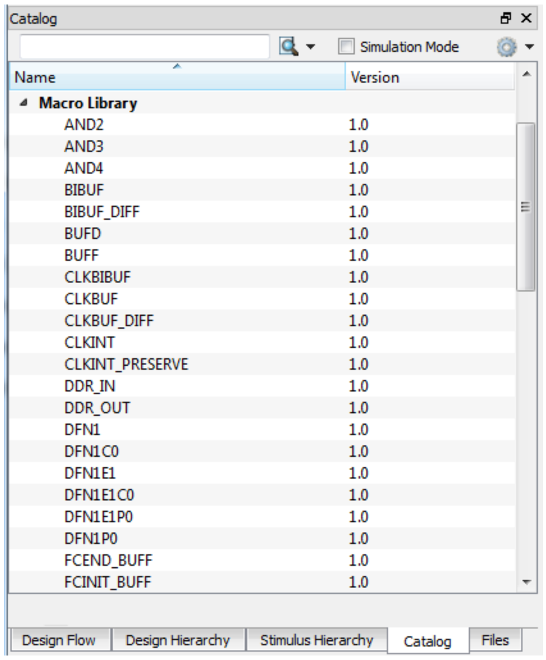
Basic-Options

Reference Clock: 50 MHz, FPGA Fabric Input 0

	Exact Value	Frequency	Actual
☑ GL0	☐	0.5 MHz	0.5000 MHz
☐ GL1		100 MHz	MHz
☐ GL2		100 MHz	MHz
☐ GL3		100 MHz	MHz

18. Click **OK** to save the changes.
19. Expand **Macro Library** in the Libero SoC IP catalog.

Figure 13 • **Macro Library Category of the Libero SoC IP Catalog**

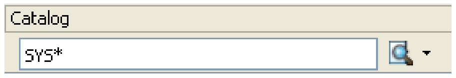


Name	Version
Macro Library	
AND2	1.0
AND3	1.0
AND4	1.0
BIBUF	1.0
BIBUF_DIFF	1.0
BUFD	1.0
BUFF	1.0
CLKBIBUF	1.0
CLKBUF	1.0
CLKBUF_DIFF	1.0
CLKINT	1.0
CLKINT_PRESERVE	1.0
DDR_IN	1.0
DDR_OUT	1.0
DFN1	1.0
DFN1C0	1.0
DFN1E1	1.0
DFN1E1C0	1.0
DFN1E1P0	1.0
DFN1P0	1.0
FCEND_BUFF	1.0
FCINIT_BUFF	1.0

20. Drag-and-drop an instance of **INBUF, and SYSRESET** in the SmartDesign Canvas.

Note: Type part of the macro name followed by * in the IP Catalog search field to find macros from the list. When finished, change the field to * to display the entire catalog.

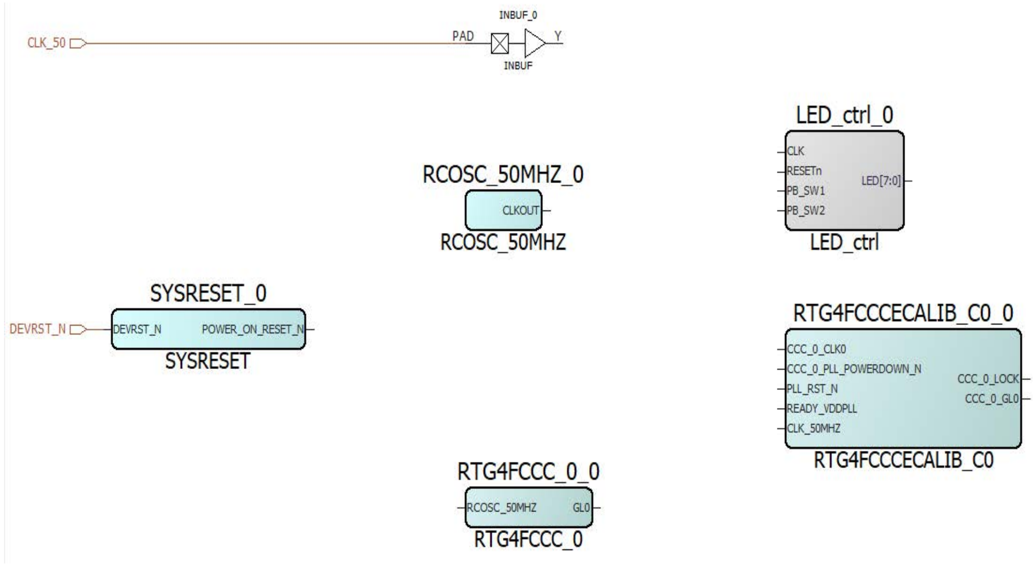
Figure 14 • **IP Catalog Search Field**



21. After adding the components, the SmartDesign is displayed, as shown in the following figure. Drag the components to improve the appearance of the canvas, if required.

Note: Expand the canvas area by selecting **View > Maximize Work Area** or click icon on the tool bar.

Figure 15 • **SmartDesign Canvas After Adding Components**



2.6.1.1 Making Connections in the Canvas

Connect the components in the SmartDesign canvas to complete the design. SmartDesign in the Libero SoC software has a connection mode that supports click, drag, and release to make connections.

1. Select **SmartDesign > Connection Mode** from the Libero SoC software or click **Connection Mode** icon .
2. Connect the **GL0** port of RTG4FCCCECALIB_C0 component to the **CLK** port of the LED_ctrl_0 as follows:
   - Click and drag-and-drop the **GL0** port of the RTG4FCCCECALIB_C0 component in the **CLK** port of the LED_ctrl_0 component.
3. Repeat step 2 to connect the ports as shown in the following table.

Table 3 • **SmartDesign Port Connections**

From		To	
Component	**Port**	**Component**	**Port**
RTG4FCCCECALIB_C0	LOCK	LED_ctrl_0	RESETn
SYSRESET_0	POWER_ON_RESET_N	RTG4FCCCECALIB_C0	PLL_RST_N and CCC_0_PLL_POWERDOWN_N
INBUF-0	Y	RTGFCCCECALIB_C0	CCC_0_CLK0
RTG4FCCC_0	GL0	RTGFCCCECALIB_C0	CLK_50MHz

4. Select **SmartDesign > Connection Mode** from the Libero SoC menu to exit the connection mode.
5. Promote the following ports to the top-level. Right-click on the port and select **Promote to Top Level**:
   - **LED_ctrl_0**: PB_SW1
   - **LED_ctrl_0**: PB_SW2
   - **LED_ctrl**: LED[7:0]
6. Rename the top-level PAD CLK_50 port by right-clicking the port and selecting **Rename Top Level Pin**.
7. Enter **CLK_50** as name in the Rename Top Level Port dialog box as shown in the following figure.

Figure 16 • **Renaming the Top-Level Port PAD**

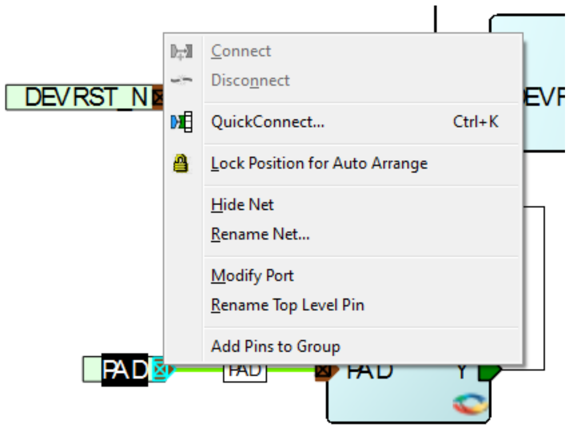
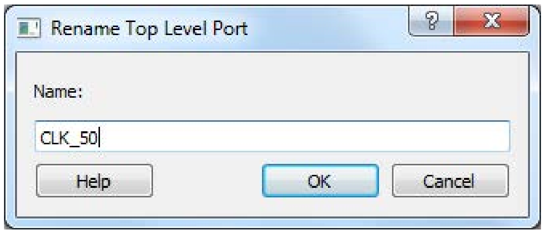
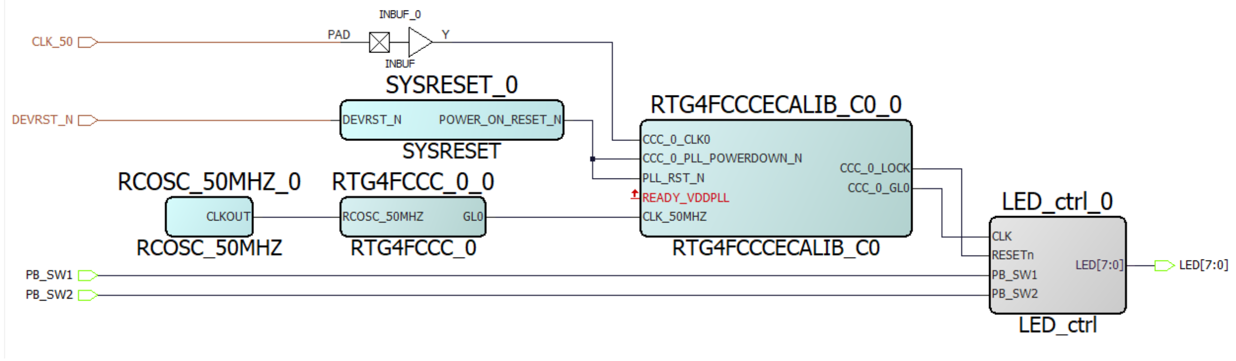


Figure 17 • **Modifying Top Level Port Names**



8. Click **OK** to save the top level port name.
9. After making the connections, the SmartDesign canvas is displayed as shown in the following figure. You can drag-and-drop the components or use the **SmartDesign Auto Arrange** feature to improve the appearance of the canvas.

Figure 18 • **SmartDesign Canvas After Connecting the Ports**



10. Save the design (**File > Save Fabric_Top**).
11. Generate the design by clicking **SmartDesign > Generate Component** or by clicking the **Generate Component** on the SmartDesign toolbar.
12. Restore the work area from **View > Restore Work Area** if you have expanded the work area earlier.
13. Ensure that **'Fabric_Top' was generated successfully**. The message appears in the Libero Message window.
14. Close the design from **File > Close Fabric_Top**.

2.7 Step 2: Simulating the Design

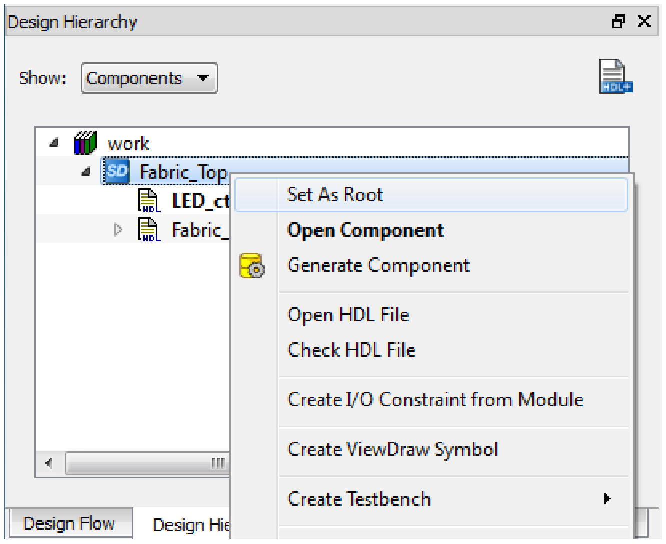
A testbench, ModelSim macro, and wave format files are provided as part of the source files. The LED_ctrl module/entity contains a 19-bit counter. To accelerate the simulation of the design, some of the counter bits are forced high in the ModelSim macro file.

Note: Simulation takes a long time with the slow clock rate such as 500 kHz.

Follow the instructions to simulate the design:

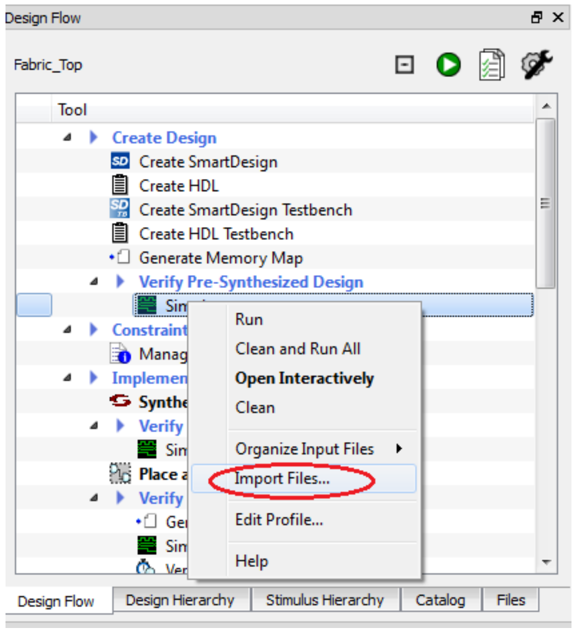
1. Ensure that **Fabric_Top** is in bold font in the Libero Design Hierarchy window. If it does not, select **Fabric_Top**, right-click and select **Set As Root**.

Figure 19 • **Setting Fabric_Top as the Root Level**



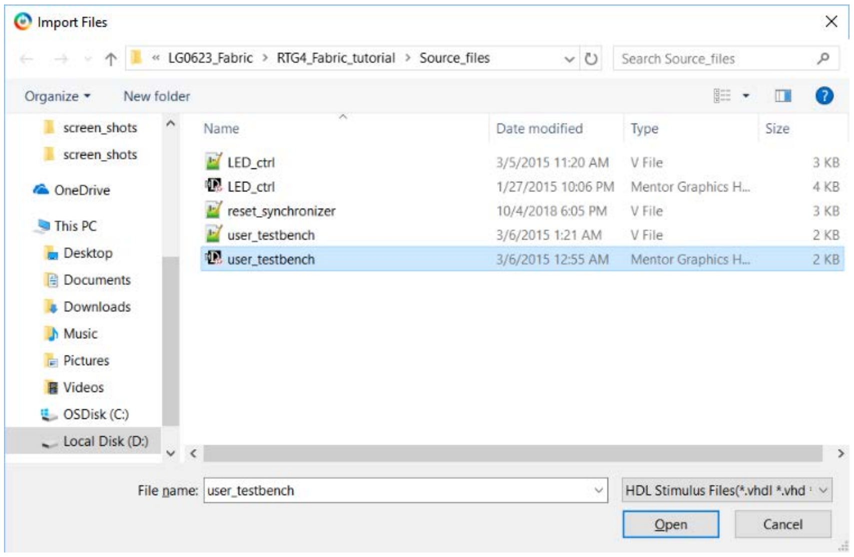
2. Expand **Verify Pre-Synthesized Design** in the Design Flow window. Right-click **Simulate** and select **Import Files**...

Figure 20 • **Importing the Testbench**



3. Enter the following in the **Import Files** dialog box and click **Open**:
   - **Look in**: <*C*:, *D*:, or *E*:>*\Microsemiprj\rtg4_lg0623_df\Source_files*
   - **Files of type**: HDL Stimulus Files (*.vhd *.v)
   - **File name**: `user_testbench.vhd` (for VHDL projects) or `user_testbench.v` (for Verilog projects)

Figure 21 • **Importing the Testbench**


Import Files

« LG0623_Fabric › RTG4_Fabric_tutorial › Source_files

Name	Date modified	Type	Size
LED_ctrl	3/5/2015 11:20 AM	V File	3 KB
LED_ctrl	1/27/2015 10:06 PM	Mentor Graphics H...	4 KB
reset_synchronizer	10/4/2018 6:05 PM	V File	3 KB
user_testbench	3/6/2015 1:21 AM	V File	2 KB
user_testbench	3/6/2015 12:55 AM	Mentor Graphics H...	2 KB

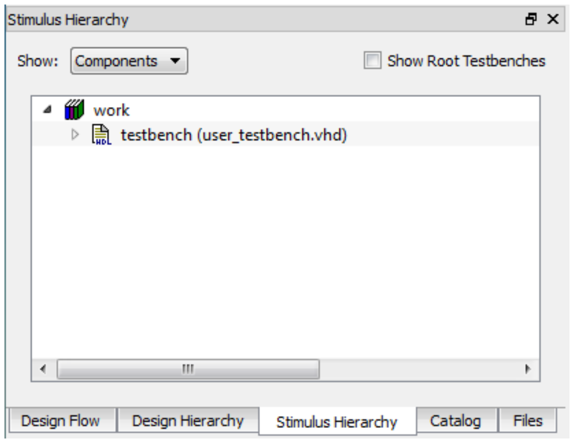
File name: user_testbench

HDL Stimulus Files(*.vhdl *.vhd

4. Select the **Stimulus Hierarchy** tab. The testbench .hdl file is displayed.

Note: If user_testbench.v or user_testbench.vhd is not visible in the Stimulus Hierarchy tab, the file is not imported as an HDL stimulus file. Re-import the file as described in step 3 and step 4.

Figure 22 • **Testbenches in the RTG4_Fabric Project (VHDL shown)**

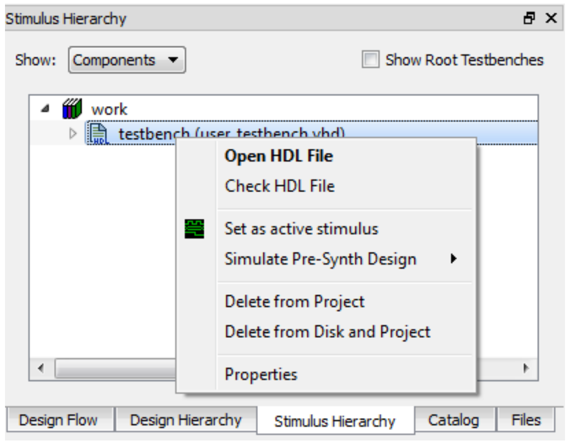

Show: Components

Show Root Testbenches

work
testbench (user_testbench.vhd)

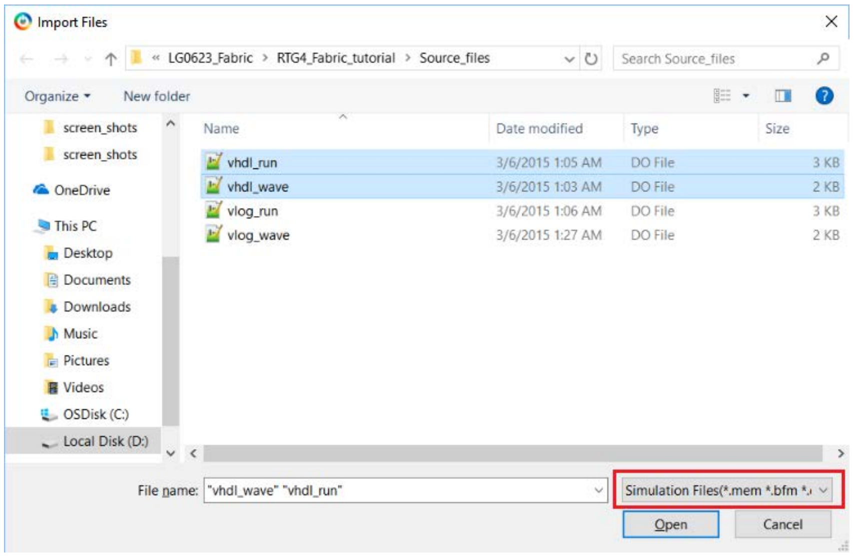
5. Right-click `user_testbench.v` or `user_testbench.vhd` and select **Set as active stimulus** to use the testbench you imported for simulation. A waveform symbol will be displayed next to the testbench name to indicate it is the active stimulus.

Figure 23 • **Selecting user_testbench for Simulation**



6. Import the ModelSim macro and Wave format files by right-clicking **Simulate** under Verify Pre-Synthesized Design in the Design Flow window and selecting **Import Files**.
7. Enter the following in the **Import Files** dialog box and click **Open**:
   - **Look in**: <*C*: or *D:* or *E:*>*\Microsemiprj\rtg4_lg0623_df\Source_files*
   - **Files of type**: Simulation Files (*.mem *.bfm *.dat *.txt *.do)
   - **File name**: Hold the shift key and select `vhdl_run.do` and `vhdl_wave.do` (VHDL projects) or `vlog_run.do` and `vlog_wave.do` (Verilog projects)

Figure 24 • **Importing the Simulation Files**



Name	Date modified	Type	Size
vhdl_run	3/6/2015 1:05 AM	DO File	3 KB
vhdl_wave	3/6/2015 1:03 AM	DO File	2 KB
vlog_run	3/6/2015 1:06 AM	DO File	3 KB
vlog_wave	3/6/2015 1:27 AM	DO File	2 KB

File name: "vhdl_wave" "vhdl_run"

Simulation Files(*.mem *.bfm *.

The testbench and simulation files are visible on the **Libero SoC** Files tab under Stimulus and Simulation, respectively.

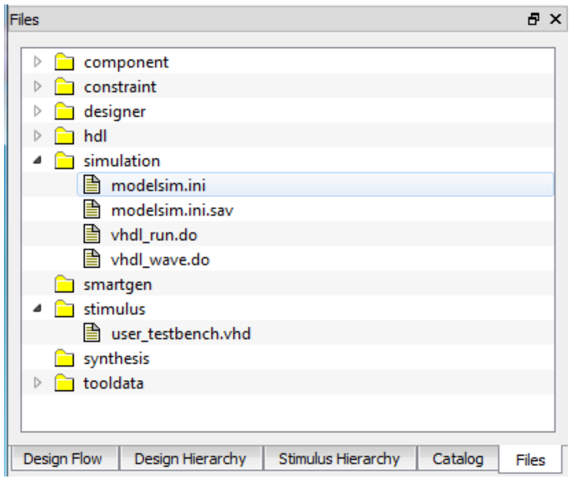


Figure 25 • **Testbench and Simulation Files (VHDL shown)**

8. Open the ModelSim macro file (`vhdl_run.do` or `vlog_run.do`) in the Libero SoC editor by double-clicking the filename on the Files tab.
9. Locate the variable PROJECT_DIR at line 11 of `vhdl_run.do` or `vlog_run.do` and confirm that it matches the location of the Libero SoC fabric tutorial. The location is displayed at the top of the Libero SoC GUI (do not include **RTG4_Fabric.prjx**). Edit the path if required.

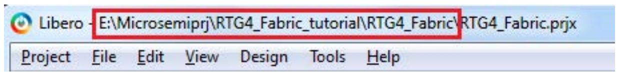
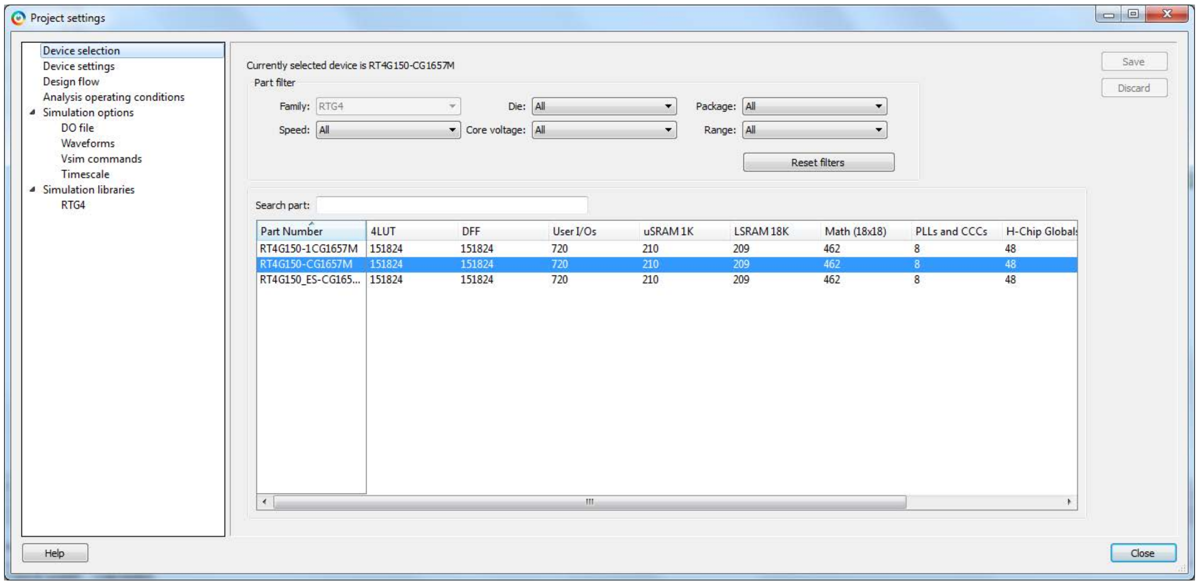

Libero - E:\Microsemiprj\RTG4_Fabric_tutorial\RTG4_Fabric\RTG4_Fabric.prjx
Project
File
Edit
View
Design
Tools
Help

Figure 26 • **Location of the Libero SoC Project**

10. Locate the variable INSTALL_DIR at line 12 of vhdl_run.do or vlog_run.do and confirm that it matches the location of the Libero SoC RTG4_Launch installation. Edit the path if required.
11. Save the file after making the changes (**File > Save vhdl_run.do or File > Save vlog_run.do**).
12. The `vhdl_run.do/vlog_run.do` file contains the force-freeze command to force the counter bits in LED_ctrl to high, This command helps to speed up the simulation.
13. Close the editor from **File > Close vhdl_run.do or File > Close vlog_run.do**.
14. Open the Libero SoC project settings from **Project > Project Settings**.



Currently selected device is RT4G150-CG1657M

Part filter

Family: RTG4
Die: All
Package: All
Speed: All
Core voltage: All
Range: All

Part Number	4LUT	DFF	User I/Os	uSRAM 1K	LSRAM 18K	Math (18x18)	PLLs and CCCs	H-Chip Global
RT4G150-1CG1657M	151824	151824	720	210	209	462	8	48
RT4G150-CG1657M	151824	151824	720	210	209	462	8	48
RT4G150_ES-CG165...	151824	151824	720	210	209	462	8	48

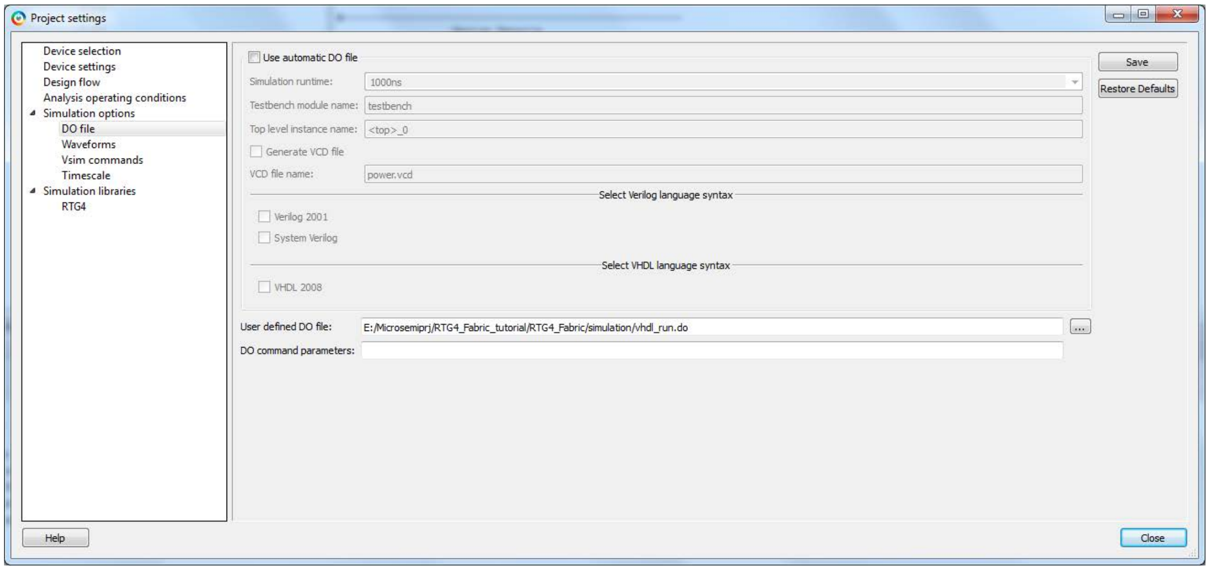
Figure 27 • **Libero SoC Project Settings Dialog Box**

15. Select **Do File** under **Simulation Options** in the Project Settings Dialog box.

16. Uncheck **Use automatic DO file** check box.
17. Click browse next to **User defined DO file**, enter the following information and click **Open**:
    - Look in: *<C: or D: or E:>\Microsemiprj\rtg4_lg0623_df/Libero_Project/Simulation*
    - Files of type:*.do
    - File name: `vhdl_run.do` (VHDL projects) or `vlog_run.do` (Verilog projects)

The ModelSim macro file (`vhdl_run.do` or `vlog_run.do`) calls the Wave format file (`vhdl_wave.do` or `vlog_wave.do`), and hence do not require any separate settings for the Wave Format file.

Figure 28 • **Simulation Options**



☐ Use automatic DO file

Simulation runtime: 1000ns

Testbench module name: testbench

Top level instance name: <top>_0

☐ Generate VCD file

VCD file name: power.vcd

Select Verilog language syntax

☐ Verilog 2001

☐ System Verilog

Select VHDL language syntax

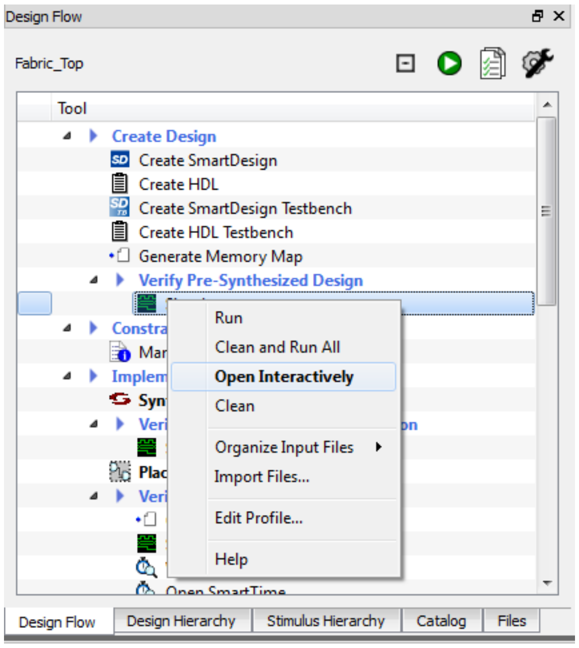
☐ VHDL 2008

User defined DO file: E:/Microsemiprj/RTG4_Fabric_tutorial/RTG4_Fabric/simulation/vhdl_run.do

DO command parameters:

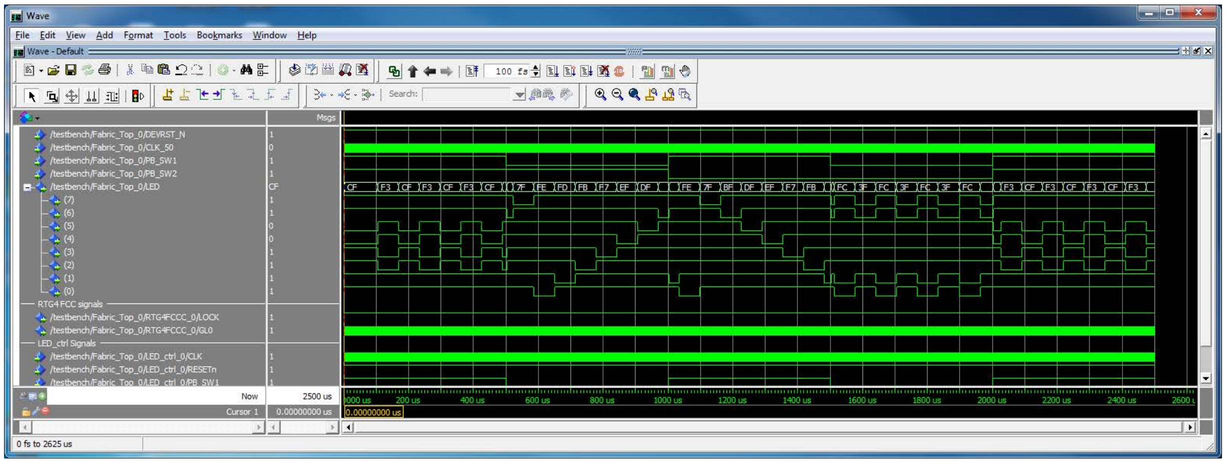
18. Click **Save** and **Close** to close the Project Settings dialog box by saving the project settings.
19. Expand Verify **Pre-Synthesized Design** in the Design Flow window. Right-click **Simulate** and select **Open Interactively** to launch ModelSim in GUI mode.

Figure 29 • **Launching Pre-Synthesis Simulation**



20. The simulation runs for 2.5 ms. When finished, the waveform window is displayed, as shown in the following figure.

Figure 30 • **ModelSim Waveform Window (VHDL results shown)**

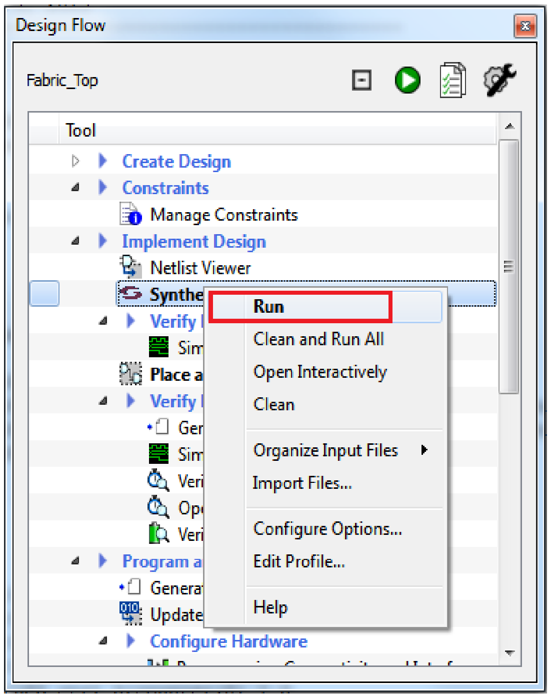

Wave
File Edit View Add Format Tools Bookmarks Window Help
Wave - Default
Search:
Msgs
/testbench/Fabric_Top_0/DEVRST_N 1
/testbench/Fabric_Top_0/CLK_50 0
/testbench/Fabric_Top_0/PB_SW1 1
/testbench/Fabric_Top_0/PB_SW2 1
/testbench/Fabric_Top_0/LED CF
(7) 1
(6) 1
(5) 0
(4) 0
(3) 1
(2) 1
(1) 1
(0) 1
RTG4 FCC signals
/testbench/Fabric_Top_0/RTG4FCCC_0/LOCK 1
/testbench/Fabric_Top_0/RTG4FCCC_0/GL0 1
LED_ctrl Signals
/testbench/Fabric_Top_0/LED_ctrl_0/CLK 1
/testbench/Fabric_Top_0/LED_ctrl_0/RESETn 1
/testbench/Fabric_Top_0/LED_ctrl_0/PB_SW1 1
Now 2500 us
Cursor 1 0.00000000 us
0 fs to 2625 us

21. Scroll in the window. The LED port represents the LED driver (1 = LED off; 0 = LED on). The PB_SW1 and PB_SW2 ports represent the switch inputs (0 = switch pressed; 1 = switch released). The DEVRST_N input represents the Reset (SW7 input).
22. Ensure that the LED output (LED) matches the description in Table 1, page 2.
23. Close the ModelSim simulator by **File > Quit**. Click **Yes** when asked if you want to quit.

2.8 Step 3: Synthesize the Design

In the **Design Flow** window, expand the **Implement Design** and right-click **Synthesize** and select **Run,** as shown in the following figure.

Figure 31 • **Design Flow - Synthesize**

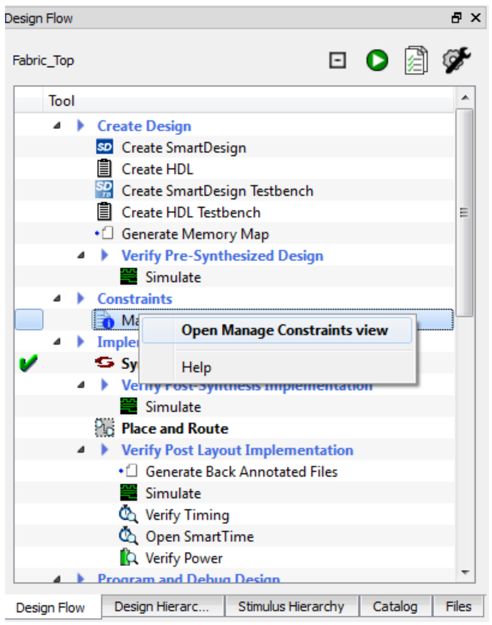


2.9 Step 4: Pin Assignments

Follow the instructions to assign the pins in the software:

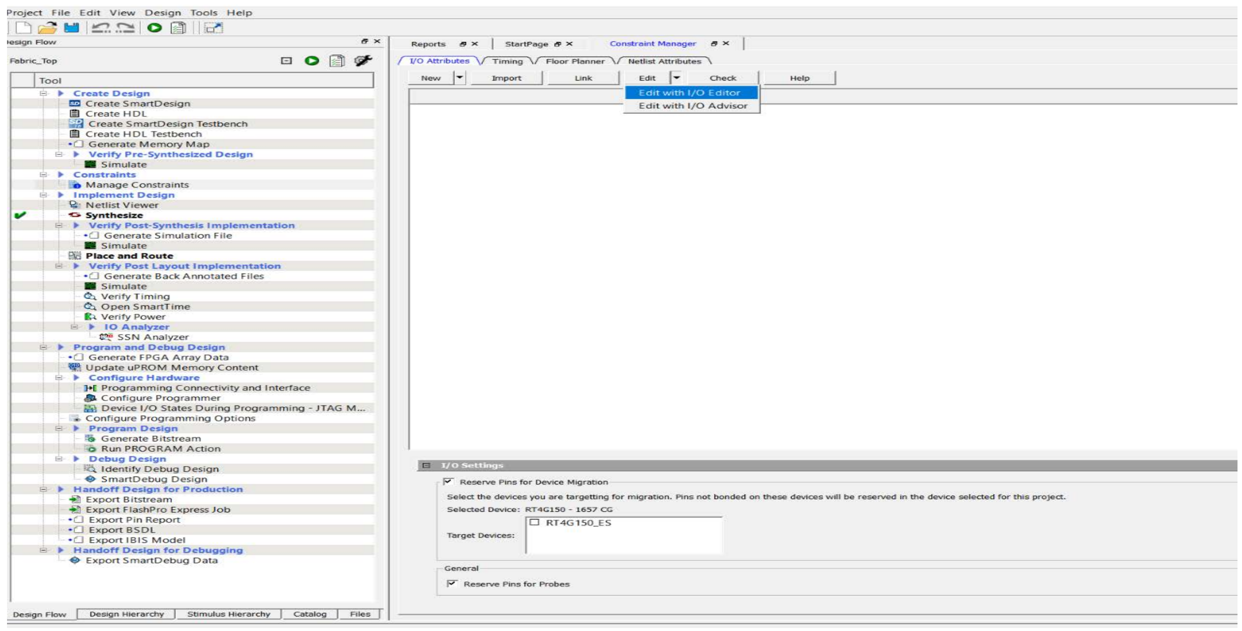
1. In the Design Flow window, expand **Constraints**. Right-click **Manage Constraints** and select **Open Manage Constraints View**.

Figure 32 • **Opening the I/O Editor**



2. In the **Constraint Manager** window, select the **I/O Attributes** tab and click **Edit with I/O Editor** as shown in the following figure.

Figure 33 • **Constraint Manager Window**



I/O Settings

☑ Reserve Pins for Device Migration

Select the devices you are targetting for migration. Pins not bonded on these devices will be reserved in the device selected for this project.

Selected Device: RT4G150 - 1657 CG

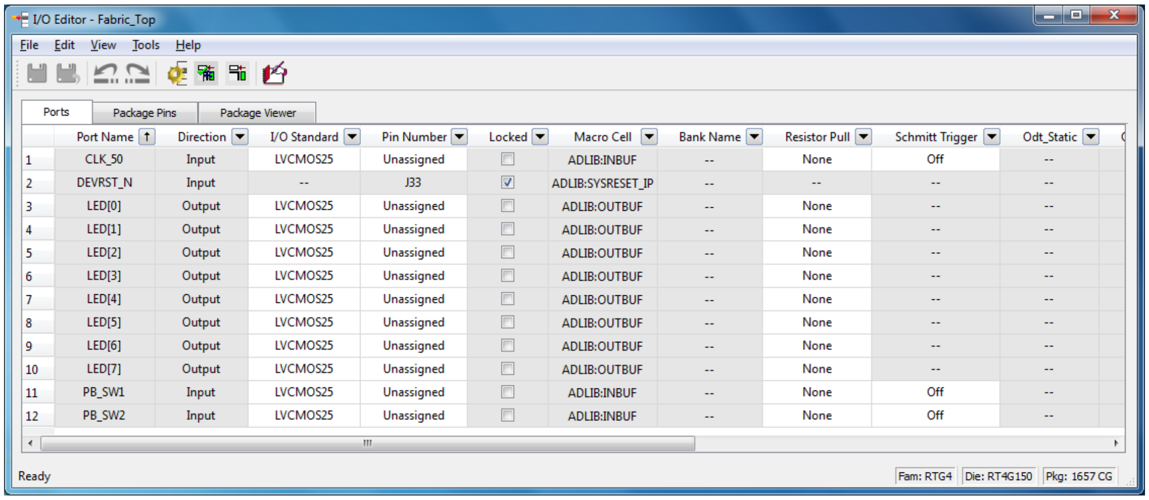
Target Devices: ☐ RT4G150_ES

General

☑ Reserve Pins for Probes

The following figure shows the I/O Editor page.

Figure 34 • **I/O Editor**



	Port Name	Direction	I/O Standard	Pin Number	Locked	Macro Cell	Bank Name	Resistor Pull	Schmitt Trigger	Odt_Static
1	CLK_50	Input	LVCMOS25	Unassigned	☐	ADLIB:INBUF	--	None	Off	--
2	DEVRST_N	Input	--	J33	☑	ADLIB:SYSRESET_IP	--	--	--	--
3	LED[0]	Output	LVCMOS25	Unassigned	☐	ADLIB:OUTBUF	--	None	--	--
4	LED[1]	Output	LVCMOS25	Unassigned	☐	ADLIB:OUTBUF	--	None	--	--
5	LED[2]	Output	LVCMOS25	Unassigned	☐	ADLIB:OUTBUF	--	None	--	--
6	LED[3]	Output	LVCMOS25	Unassigned	☐	ADLIB:OUTBUF	--	None	--	--
7	LED[4]	Output	LVCMOS25	Unassigned	☐	ADLIB:OUTBUF	--	None	--	--
8	LED[5]	Output	LVCMOS25	Unassigned	☐	ADLIB:OUTBUF	--	None	--	--
9	LED[6]	Output	LVCMOS25	Unassigned	☐	ADLIB:OUTBUF	--	None	--	--
10	LED[7]	Output	LVCMOS25	Unassigned	☐	ADLIB:OUTBUF	--	None	--	--
11	PB_SW1	Input	LVCMOS25	Unassigned	☐	ADLIB:INBUF	--	None	Off	--
12	PB_SW2	Input	LVCMOS25	Unassigned	☐	ADLIB:INBUF	--	None	Off	--

Ready

Fam: RTG4 | Die: RT4G150 | Pkg: 1657 CG

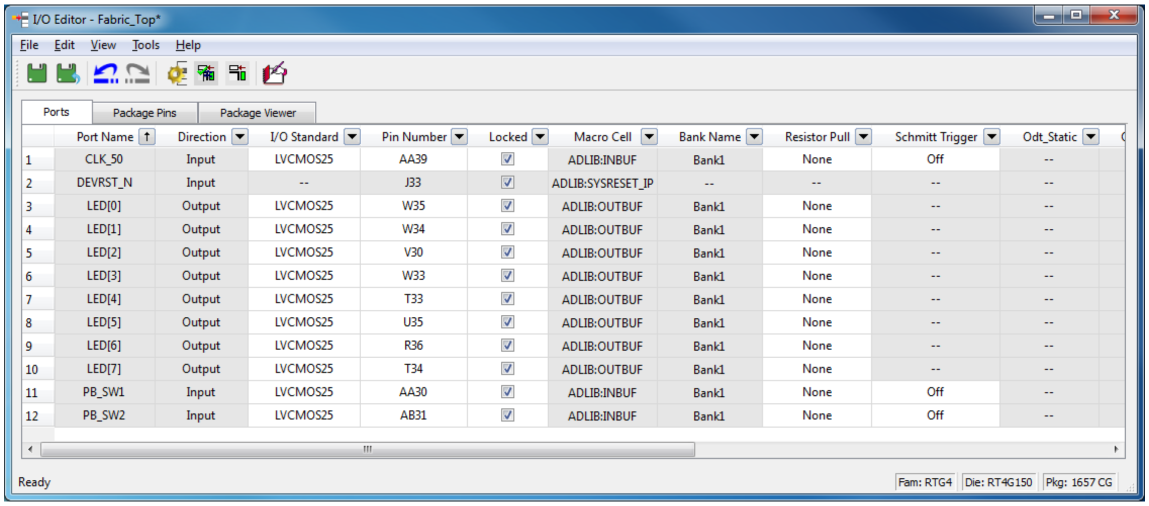
3. Make the pin assignments as listed in the following table. In **I/O Editor**, go to the **Pin Number** column and click **Unassigned** in the Pin Number column to see the available I/O pins. Accept the default settings for all other options.

Table 4 • **Pin Assignments**

Port Name	I/O Standard	Pin No	Pin Name
CLK_50	LVCMOS25	AA39	MSIOD73PB1/GB12_23
LED[0]	LVCMOS25	W35	MSIOD62PB1
LED[1]	LVCMOS25	W34	MSIOD46PB1
LED[2]	LVCMOS25	V30	MSIOD47PB1
LED[3]	LVCMOS25	W33	MSIOD46NB1
LED[4]	LVCMOS25	T33	MSIOD38PB1
LED[5]	LVCMOS25	U35	MSIOD41NB1
LED[6]	LVCMOS25	R36	MSIOD37PB1
LED[7]	LVCMOS25	T34	MSIOD38NB1
PB_SW1	LVCMOS25	AA30	MSIOD68NB1
PB_SW2	LVCMOS25	AB31	MSIOD65PB1

The following table shows the I/O Attribute Editor after assigning the pins.

Figure 35 • I/O Editor with Pin Assignments



	Port Name	Direction	I/O Standard	Pin Number	Locked	Macro Cell	Bank Name	Resistor Pull	Schmitt Trigger	Odt_Static
1	CLK_50	Input	LVCMOS25	AA39	✓	ADLIB:INBUF	Bank1	None	Off	--
2	DEVRST_N	Input	--	J33	✓	ADLIB:SYSRESET_IP	--	--	--	--
3	LED[0]	Output	LVCMOS25	W35	✓	ADLIB:OUTBUF	Bank1	None	--	--
4	LED[1]	Output	LVCMOS25	W34	✓	ADLIB:OUTBUF	Bank1	None	--	--
5	LED[2]	Output	LVCMOS25	V30	✓	ADLIB:OUTBUF	Bank1	None	--	--
6	LED[3]	Output	LVCMOS25	W33	✓	ADLIB:OUTBUF	Bank1	None	--	--
7	LED[4]	Output	LVCMOS25	T33	✓	ADLIB:OUTBUF	Bank1	None	--	--
8	LED[5]	Output	LVCMOS25	U35	✓	ADLIB:OUTBUF	Bank1	None	--	--
9	LED[6]	Output	LVCMOS25	R36	✓	ADLIB:OUTBUF	Bank1	None	--	--
10	LED[7]	Output	LVCMOS25	T34	✓	ADLIB:OUTBUF	Bank1	None	--	--
11	PB_SW1	Input	LVCMOS25	AA30	✓	ADLIB:INBUF	Bank1	None	Off	--
12	PB_SW2	Input	LVCMOS25	AB31	✓	ADLIB:INBUF	Bank1	None	Off	--

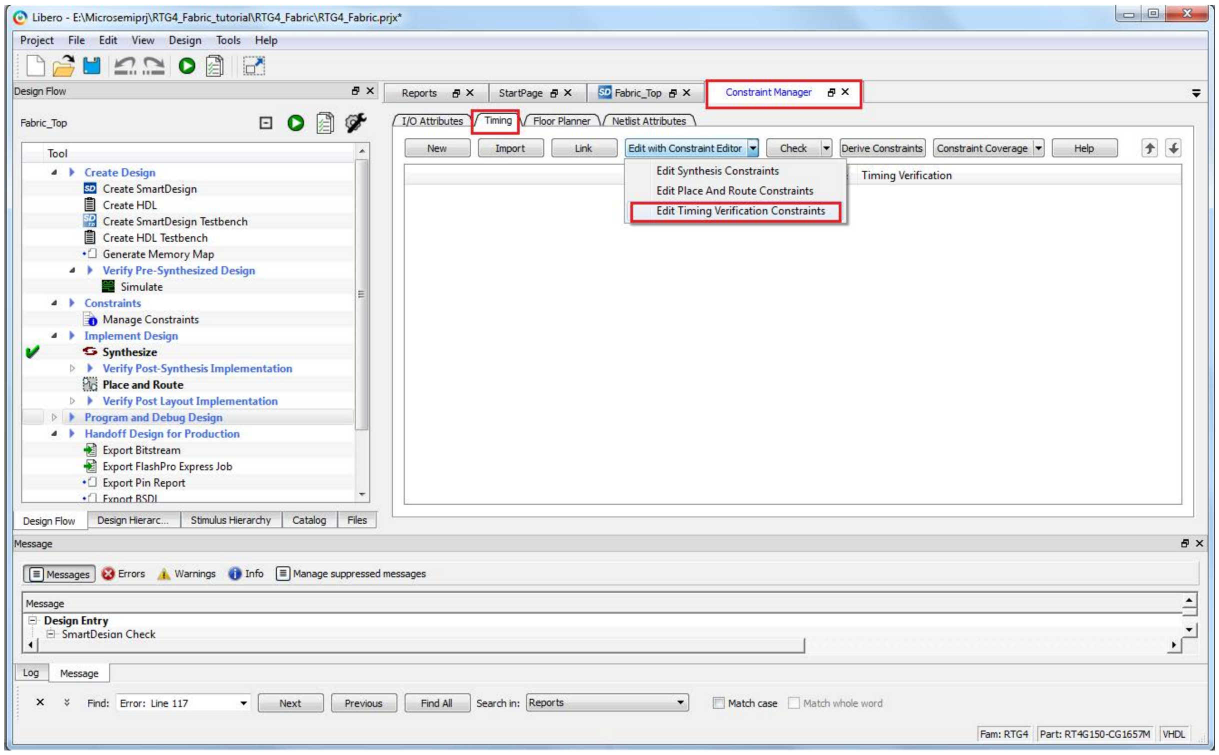
4. Select **File > Commit and Check** from the **I/O Attribute Editor** menu. Ensure that the I/O Editor Log window has no errors.
5. Close the I/O Attribute Editor using **File > Exit**.

2.10 Step 5: Entering Timing Constraints

Follow the instructions to set timing constraints in the Libero SoC software:

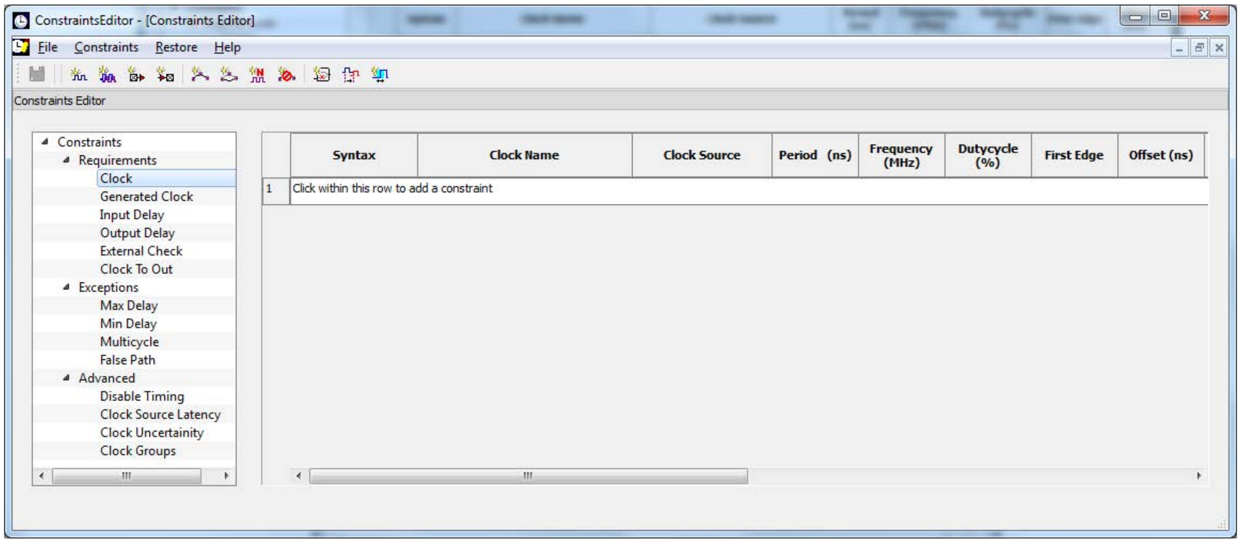
1. In the **Constraint Manager** window, select the **Timing** tab, click **Edit with Constraint Editor > Edit Timing Verification Constraints**, as shown in the following figure.

Figure 36 • Edit Timing Constraint



The following figure shows the **Constraint Editor** window.

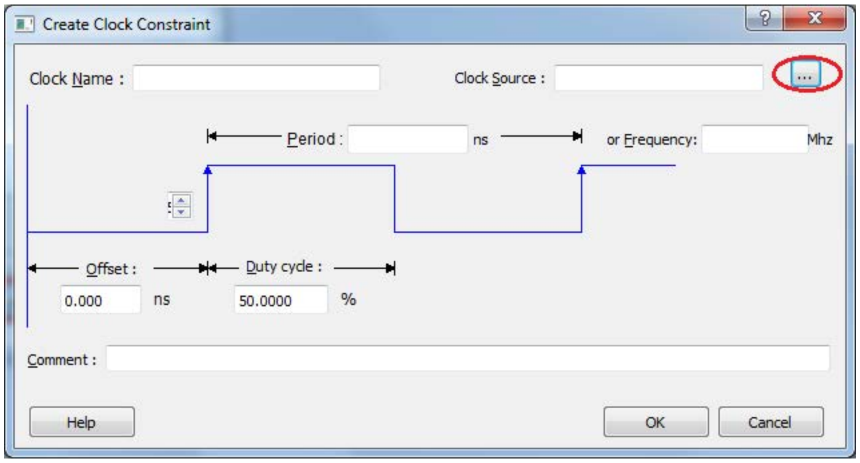
Figure 37 • **Constraint Editor**



	Syntax	Clock Name	Clock Source	Period (ns)	Frequency (MHz)	Dutycycle (%)	First Edge	Offset (ns)
1	Click within this row to add a constraint							

2. Click **Constraints > Clock** to add a Clock constraint. The Create Clock Constraint dialog box is displayed as shown in the following figure and click **Browse**.

Figure 38 • **Create Clock Constraint Dialog Box**

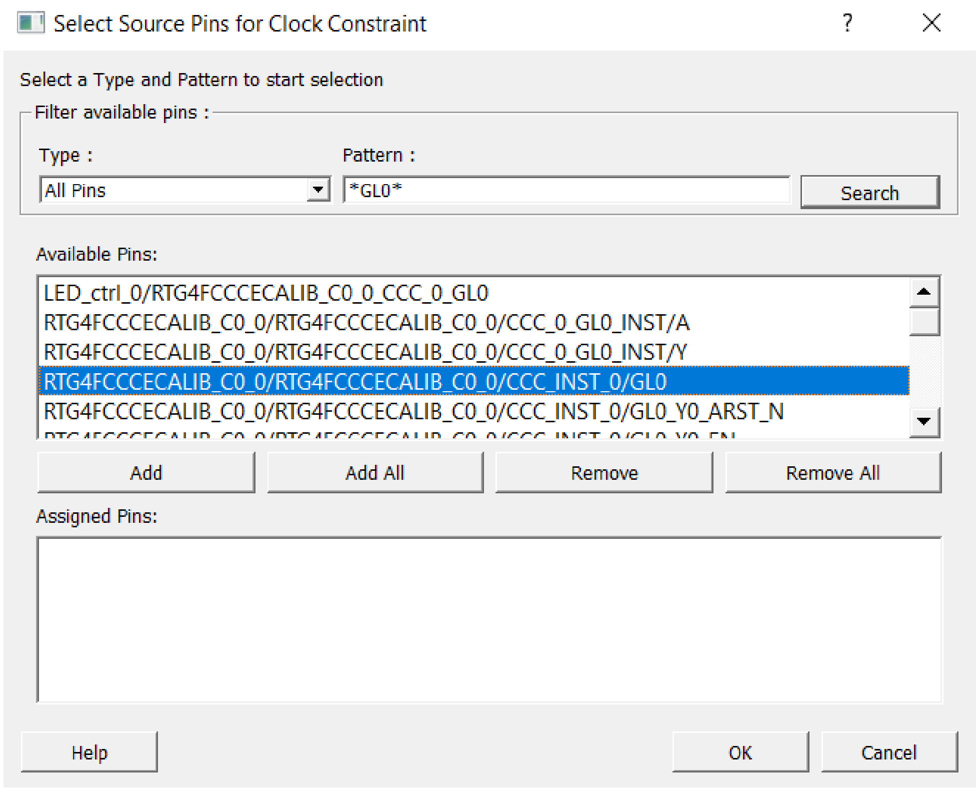


Offset : 0.000 ns

Duty cycle : 50.0000 %

3. It opens the **Select Source Pins for Clock Constraint** window. Select the **RTG4FCCCECALIB_C0/RTGFCCCECALIB_C0_0/CCC_INST_0/GL0** pin, click **Add** and click **OK** as shown in the following figure.

Figure 39 • **Clock Constraint Window**



Select Source Pins for Clock Constraint

Select a Type and Pattern to start selection

Filter available pins :

Type : All Pins

Pattern : *GL0*

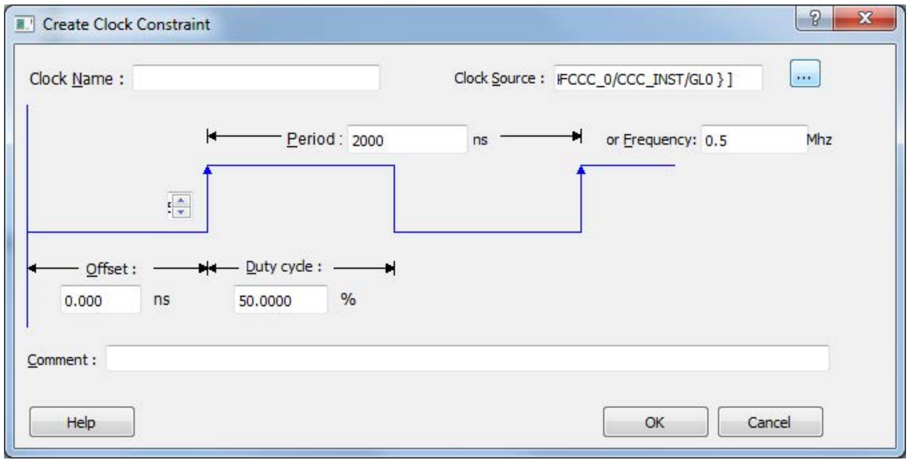
Available Pins:

LED_ctrl_0/RTG4FCCCECALIB_C0_0_CCC_0_GL0
RTG4FCCCECALIB_C0_0/RTG4FCCCECALIB_C0_0/CCC_0_GL0_INST/A
RTG4FCCCECALIB_C0_0/RTG4FCCCECALIB_C0_0/CCC_0_GL0_INST/Y
RTG4FCCCECALIB_C0_0/RTG4FCCCECALIB_C0_0/CCC_INST_0/GL0
RTG4FCCCECALIB_C0_0/RTG4FCCCECALIB_C0_0/CCC_INST_0/GL0_Y0_ARST_N

Assigned Pins:

4. Enter the following information in the **Create Clock Constraint** dialog box and click **OK**:
   - **Frequency**: 0.5 MHz

Figure 40 • **Entering the Clock Constraint**



Create Clock Constraint

Clock Name :

Clock Source : FCCC_0/CCC_INST/GL0 }]

Period : 2000 ns

or Frequency: 0.5 Mhz

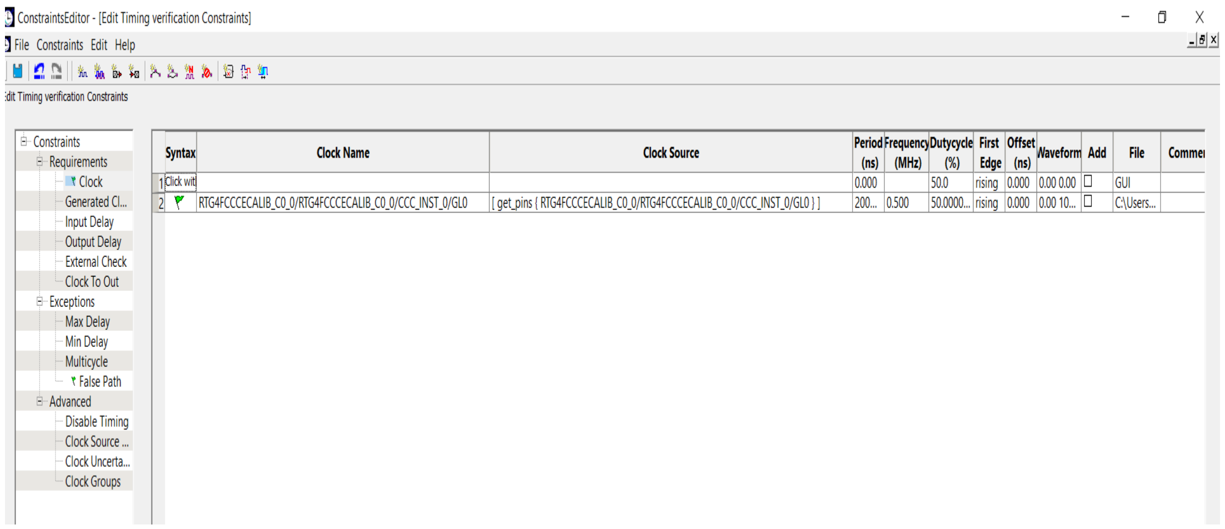
Offset : 0.000 ns

Duty cycle : 50.0000 %

Comment :

The following figure shows the clock constraint in the **Constraints Editor**.

Figure 41 • **Constraints Editor with Clock Constraint**



	Syntax	Clock Name	Clock Source	Period (ns)	Frequency (MHz)	Dutycycle (%)	First Edge	Offset (ns)	Waveform	Add	File	Commen
1	Click wit			0.000		50.0	rising	0.000	0.00 0.00	☐	GUI	
2		RTG4FCCCECALIB_C0_0/RTG4FCCCECALIB_C0_0/CCC_INST_0/GL0	[get_pins { RTG4FCCCECALIB_C0_0/RTG4FCCCECALIB_C0_0/CCC_INST_0/GL0 }]	200...	0.500	50.0000...	rising	0.000	0.00 10...	☐	C:\Users...	

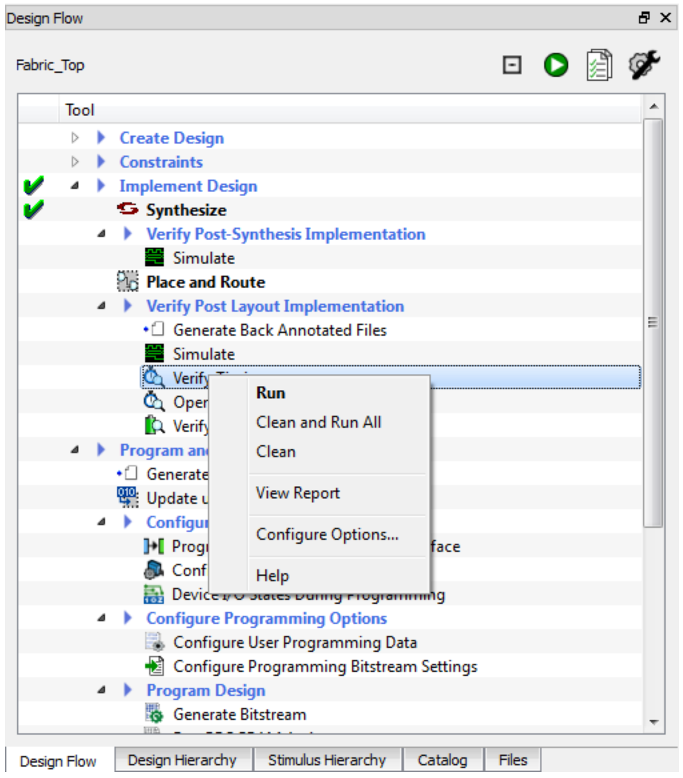
5. Close the Constraints Editor (**File > Exit**). Click **Yes** when prompted to save the changes.

2.11 Step 6: Layout and Timing Verification

2.11.1 Layout Verification

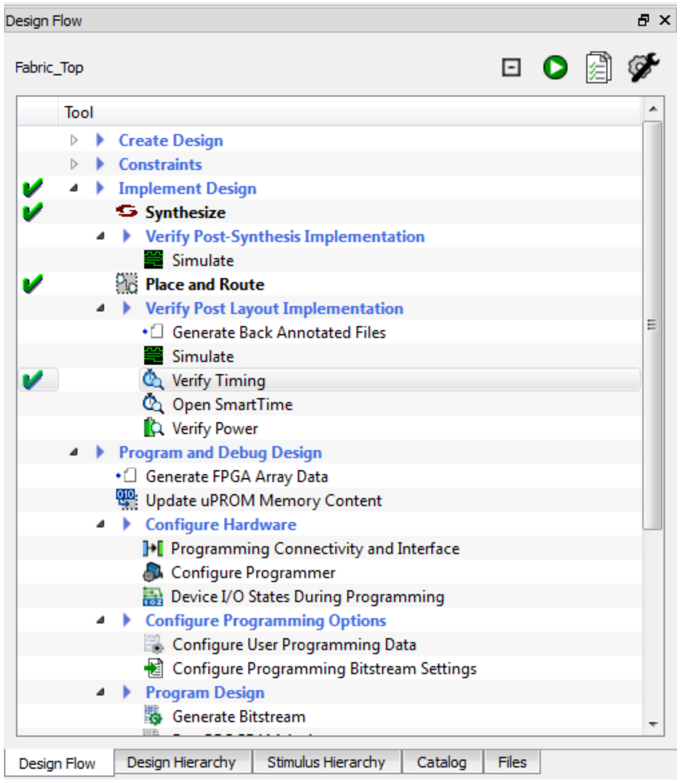
Expand **Verify Post Layout Implementation** on the Design Flow tab. Right-click **Verify Timing** and select **Run**. Place-and-Route tool is executed and a timing report is generated.

Figure 42 • **Running Layout and Verifying Timing**



A green checkmark is displayed in the Design Flow window next to Place and Route and Verify Timing indicating that the layout is completed without any errors and a timing report is generated.

Figure 43 • **Design Flow Tab after Layout and Timing Report Generation**

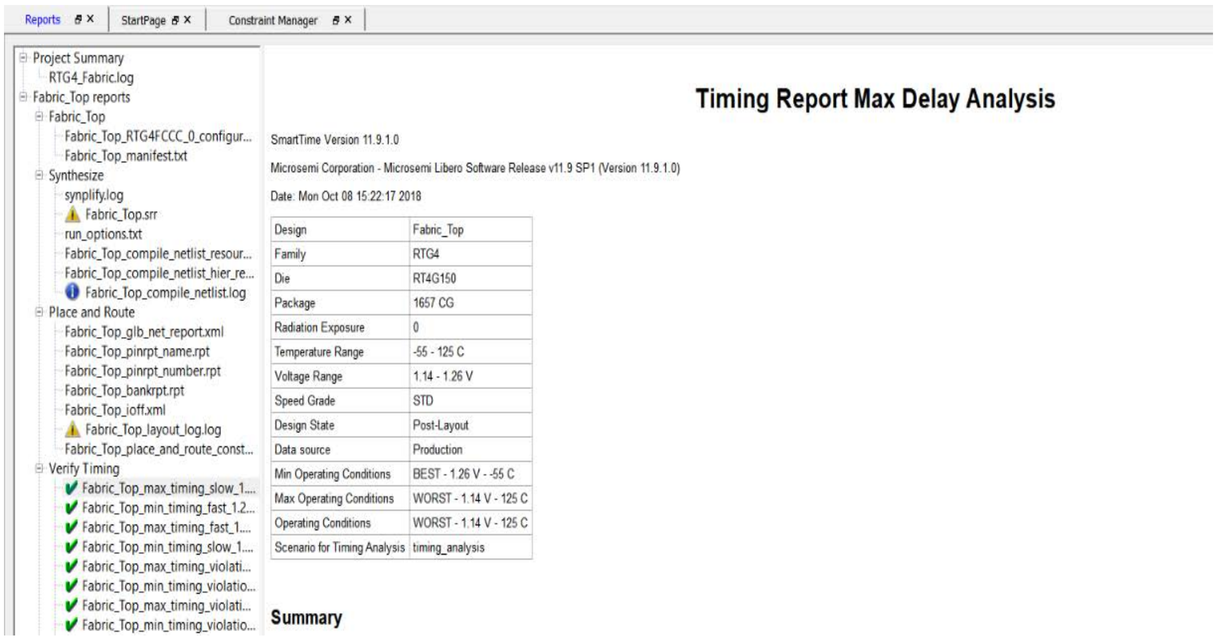


- ✓ Create Design
- Constraints
- ✓ Implement Design
  - ✓ Synthesize
  - Verify Post-Synthesis Implementation
    - Simulate
  - ✓ Place and Route
  - Verify Post Layout Implementation
    - Generate Back Annotated Files
    - Simulate
    - ✓ Verify Timing
    - Open SmartTime
    - Verify Power
- Program and Debug Design
  - Generate FPGA Array Data
  - Update uPROM Memory Content
  - Configure Hardware
    - Programming Connectivity and Interface
    - Configure Programmer
    - Device I/O States During Programming
  - Configure Programming Options
    - Configure User Programming Data
    - Configure Programming Bitstream Settings
  - Program Design
    - Generate Bitstream

2.11.2 Timing Verification

- The timing reports are in the **Reports tab**.
- Maximum Delay Analysis is contained in the Fabric_Top_max_timing_slow_1.14V_125C.xml report. Minimum Delay Analysis is contained in the Fabric_Top_min_timing_fast_1.26V_-55C.xml report. Both the reports are available in the **Reports** window.

Figure 44 • **Timing Reports**


Reports
StartPage
Constraint Manager
Project Summary
RTG4_Fabric.log
Fabric_Top reports
Fabric_Top
Fabric_Top_RTG4FCCC_0_configur...
Fabric_Top_manifest.txt
Synthesize
synplify.log
Fabric_Top.srr
run_options.txt
Fabric_Top_compile_netlist_resour...
Fabric_Top_compile_netlist_hier_re...
Fabric_Top_compile_netlist.log
Place and Route
Fabric_Top_glb_net_report.xml
Fabric_Top_pinrpt_name.rpt
Fabric_Top_pinrpt_number.rpt
Fabric_Top_bankrpt.rpt
Fabric_Top_ioff.xml
Fabric_Top_layout_log.log
Fabric_Top_place_and_route_const...
Verify Timing
Fabric_Top_max_timing_slow_1....
Fabric_Top_min_timing_fast_1.2...
Fabric_Top_max_timing_fast_1....
Fabric_Top_min_timing_slow_1....
Fabric_Top_max_timing_violati...
Fabric_Top_min_timing_violatio...
Fabric_Top_max_timing_violati...
Fabric_Top_min_timing_violatio...

Timing Report Max Delay Analysis

SmartTime Version 11.9.1.0

Microsemi Corporation - Microsemi Libero Software Release v11.9 SP1 (Version 11.9.1.0)

Date: Mon Oct 08 15:22:17 2018

Design	Fabric_Top
Family	RTG4
Die	RT4G150
Package	1657 CG
Radiation Exposure	0
Temperature Range	-55 - 125 C
Voltage Range	1.14 - 1.26 V
Speed Grade	STD
Design State	Post-Layout
Data source	Production
Min Operating Conditions	BEST - 1.26 V - -55 C
Max Operating Conditions	WORST - 1.14 V - 125 C
Operating Conditions	WORST - 1.14 V - 125 C
Scenario for Timing Analysis	timing_analysis

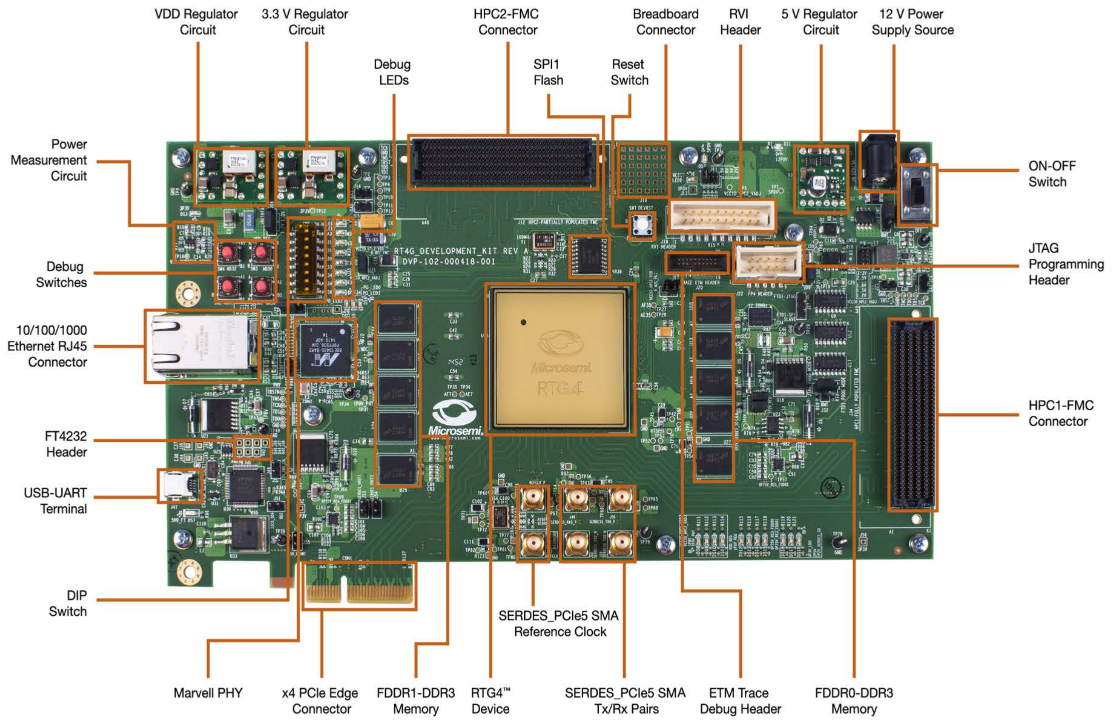
Summary

- Timing violations can quickly be identified by looking at the timing violations reports such as `Fabric_Top_max_timing_violations_slow_1.14V_125C.xml` or `Fabric_Top_min_timing_violations_fast_1.26V_-55C.xml`.

2.12 Step 7: Setting Up the Demo Design

The RTG4 Development Kit board is shown in the following figure.

Figure 45 • **RTG4 Development Kit Board**



The following steps describe how to set up the RTG4 Development Kit to run the demo design:

1. Ensure that the board is powered OFF using **SW6** switch. Also ensure that AC input connector is not connected to JP before connecting the cable or changing the jumper connections.
2. Plug the AC adapter to the J9 connector and plug into a 120 V AC outlet.

3. Before programming (and powering up) the RTG4 Development Kit board, ensure that the jumpers are positioned as show in the following table.

Table 5 • **Jumper Settings**

Jumper	Location	Purpose	Settings
J16	Above SW3	Select VDD core voltage	2-3 installed
J17	Below J9 AC connector	Select either SW6 input or signal ENABLE_FT4232 from FT4232H chip	1-2 installed
J19	Below J9 AC connector		1-2 installed
J21	To the right of the dip switch bank	Bank 7 supply voltage	1-2 installed
J23	To the right of the FP4 programming header		1-2 installed
J26	Below ETM Trace header	Bank 2 supply voltage	1-2 installed
J28	Below the dip switch bank		1-2 installed
J32	To the left of the FMC connector (HPC1)	Enable FlashPro5 for programming	1-2 installed
J33	Below FP4 programming header		1-2 installed 3-4 installed

4. Connect USB cable (mini USB to Type A USB cable) to J47 of the RTG4 Development Kit board and another end of the cable to the USB port of the Host PC.
5. Switch ON the power supply switch, **SW6**.

2.13 Step 8: Programming the Device

To program the RTG4 Development Kit with the job file provided as part of the design files using FlashPro Express software, refer to Appendix 1: Programming the Device Using FlashPro Express, page 29.

2.14 Step 9: Running the Application

Follow the instructions to run the application:

1. Reset the board by holding the **SW7** reset switch. The LEDs must be in the OFF position.
2. Release SW7 and observe the pattern of the LEDs. LEDs [6:5] and LEDs [4:3] (the Red and Green LEDs) must be toggled.
3. Press and hold the SW1 switch and observe the pattern of the LEDs. Release the SW1 switch and press the SW2 switch and observe the pattern of the LEDs. Press and hold SW1 and SW2 switches together. The LED pattern must match the description in Table 1, page 2.
4. When finished, change the SW6 switch to the OFF position and power OFF the board.

3 Appendix 1: Programming the Device Using FlashPro Express

This section describes how to program the RTG4 device with the programming job file using FlashPro Express.

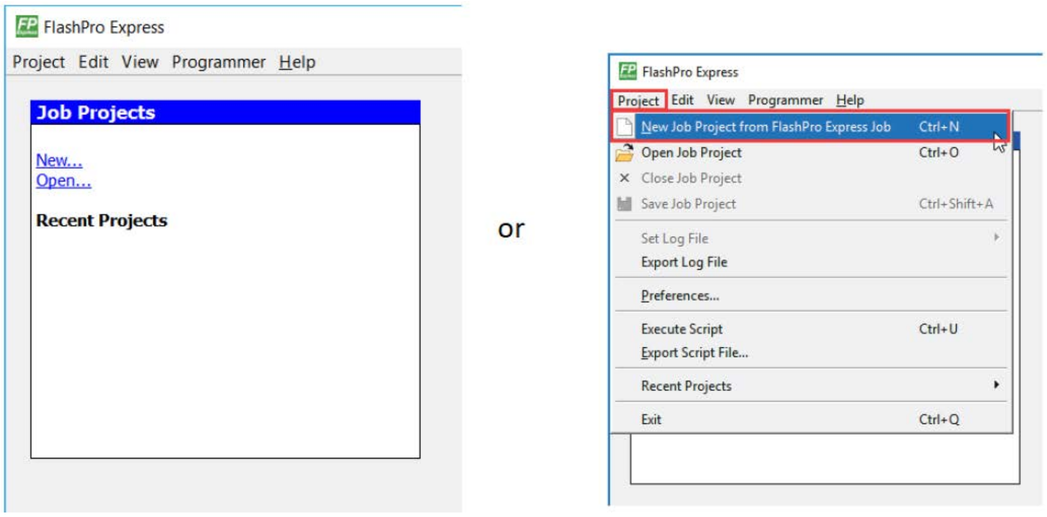
To program the device, perform the following steps:

1. Ensure that the jumper settings on the board are the same as those listed in *Table 3 of UG0617: RTG4 Development Kit User Guide*.
2. Optionally, jumper **J32** can be set to connect pins 2-3 when using an external FlashPro4, FlashPro5, or FlashPro6 programmer instead of the default jumper setting to use the embedded FlashPro5.

Note: The power supply switch, **SW6** must be switched **OFF** while making the jumper connections.

3. Connect the power supply cable to the **J9** connector on the board.
4. Power **ON** the power supply switch **SW6**.
5. If using the embedded FlashPro5, connect the USB cable to connector **J47** and the host PC. Alternatively, if using an external programmer, connect the ribbon cable to the JTAG header **J22** and connect the programmer to the host PC.
6. On the host PC, launch the **FlashPro Express** software.
7. Click **New** or select **New Job Project from FlashPro Express Job** from **Project** menu to create a new job project, as shown in the following figure.

Figure 46 • **FlashPro Express Job Project**



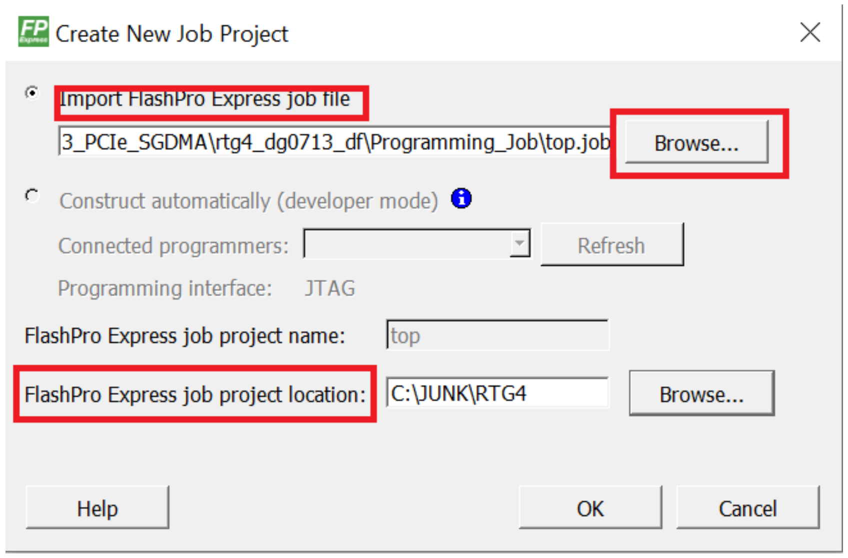
8. Enter the following in the **New Job Project from FlashPro Express Job** dialog box:

- **Programming job file:** Click **Browse**, and navigate to the location where the .job file is located and select the file. The default location is:

  `<download_folder>\rtg4_lg0623_df\Programming_Job`

- **FlashPro Express job project location:** Click **Browse** and navigate to the desired FlashPro Express project location.

Figure 47 • **New Job Project from FlashPro Express Job**



Create New Job Project

Import FlashPro Express job file

3_PCIe_SGDMA\rtg4_dg0713_df\Programming_Job\top.job

Construct automatically (developer mode)

Connected programmers:

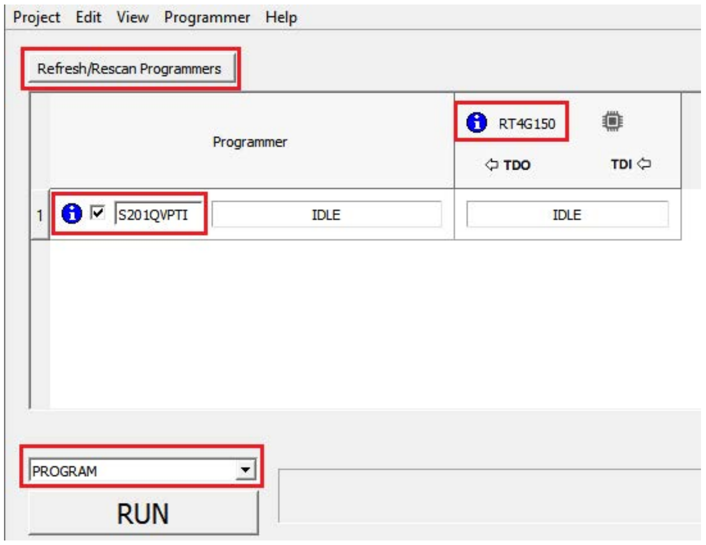
Programming interface: JTAG

FlashPro Express job project name: top

FlashPro Express job project location: C:\JUNK\RTG4

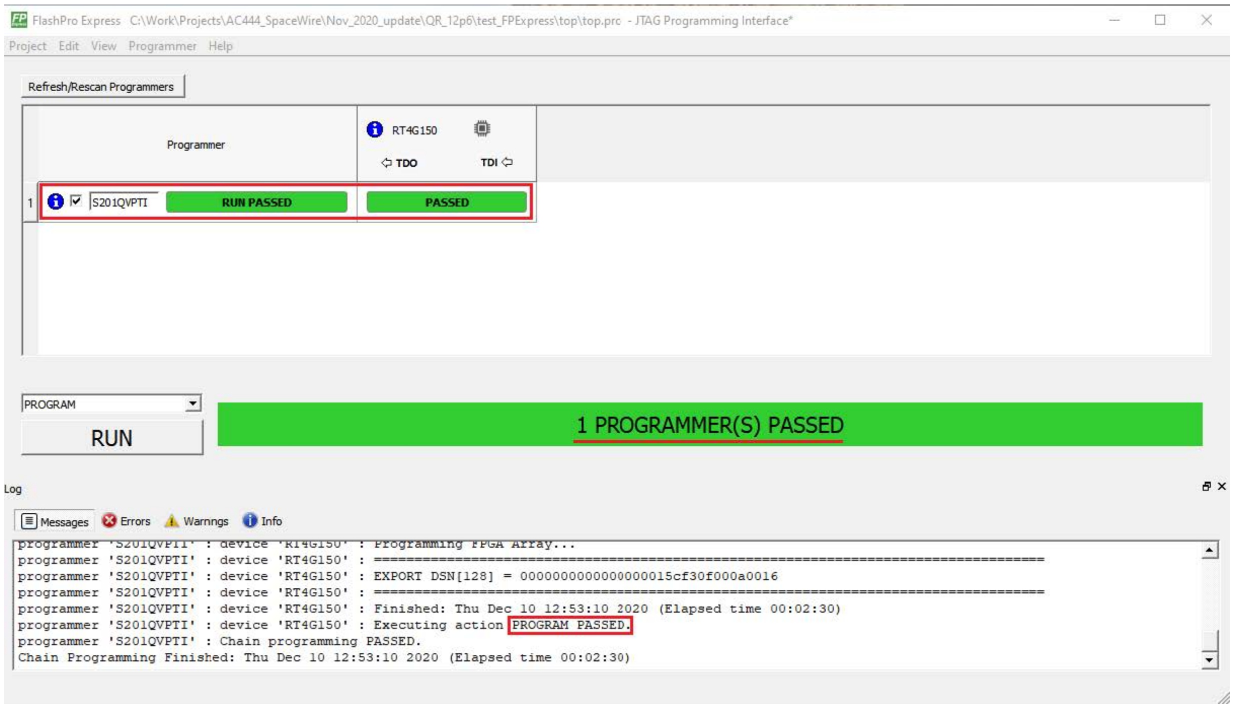
9. Click **OK**. The required programming file is selected and ready to be programmed in the device.
10. The FlashPro Express window appears as shown in the following figure. Confirm that a programmer number appears in the Programmer field. If it does not, confirm the board connections and click **Refresh/Rescan** Programmers.

Figure 48 • **Programming the Device**



11. Click **RUN**. When the device is programmed successfully, a **RUN PASSED** status is displayed as shown in the following figure.

Figure 49 • **FlashPro Express—RUN PASSED**


FlashPro Express C:\Work\Projects\AC444_SpaceWire\Nov_2020_update\QR_12p6\test_FPExpress\top\top.pro - JTAG Programming Interface*
Project Edit View Programmer Help
Refresh/Rescan Programmers
Programmer
RT4G150
TDO
TDI
1
S201QVPTI
RUN PASSED
PASSED
PROGRAM
RUN
1 PROGRAMMER(S) PASSED
Log
Messages
Errors
Warnngs
Info

```
programmer 'S201QVPTI' : device 'RT4G150' : Programming FPGA Array...
programmer 'S201QVPTI' : device 'RT4G150' : ================================================================================
programmer 'S201QVPTI' : device 'RT4G150' : EXPORT DSN[128] = 0000000000000000015cf30f000a0016
programmer 'S201QVPTI' : device 'RT4G150' : ================================================================================
programmer 'S201QVPTI' : device 'RT4G150' : Finished: Thu Dec 10 12:53:10 2020 (Elapsed time 00:02:30)
programmer 'S201QVPTI' : device 'RT4G150' : Executing action PROGRAM PASSED.
programmer 'S201QVPTI' : Chain programming PASSED.
Chain Programming Finished: Thu Dec 10 12:53:10 2020 (Elapsed time 00:02:30)
```

12. Close **FlashPro Express** or click **Exit** in the Project tab.

4 Appendix 2: Running the TCL Script

TCL scripts are provided in the design files folder under directory TCL_Scripts. If required, the design flow can be reproduced from Design Implementation till generation of job file.

To run the TCL, follow the steps below:

1. Launch the Libero software
2. Select **Project > Execute Script....**
3. Click Browse and select `script.tcl` from the downloaded TCL_Scripts directory.
4. Click **Run**.

After successful execution of TCL script, Libero project is created within TCL_Scripts directory.

For more information about TCL scripts, refer to **rtg4_lg0623_df/TCL_Scripts/readme.txt.**

Refer to *Libero® SoC TCL Command Reference Guide* for more details on TCL commands. Contact Technical Support for any queries encountered when running the TCL script.