

SmartFusion2/IGL002 Fabric Lab Guide

Microsemi PRO BU, San Jose, CA 95134

© 2020 Microsemi Corporation. All rights reserved.

Printed in the United States of America

Release: October 2020

Microsemi makes no warranties with respect to this documentation and disclaims any implied warranties of merchantability or fitness for a particular purpose. Information in this document is subject to change without notice. Microsemi assumes no responsibility for any errors that may appear in this document.

This document contains confidential proprietary information that is not to be disclosed to any unauthorized person without prior written consent of Microsemi Corporation.

Trademarks

Microsemi, SmartFusion2, IGLOO2 and the associated logos are trademarks or registered trademarks of Microsemi Corporation. All other trademarks and service marks are the property of their respective owners.

Table of Contents

SmartFusion2/IGLOO2 Fabric Lab Guide.....	1
Table of Contents.....	3
Introduction	4
Components Used in the Design	4
Tutorial Requirements.....	5
Extracting the source files.....	5
Step 1 – Creating the Design	5
Launching Libero SoC and creating a project.....	5
Implementing the design with SmartDesign	11
Step 2 - Simulating the design	20
Step 3 – Editing the I/O Physical Constraint file	26
Step 4 - Generating Timing Constraints	28
Step 5 – Synthesis and Layout	29
Step 6 – Programming	33
Running the Application	35
Step 7 - Debugging with SmartDebug	35
SmartDebug Live Probes.....	36
SmartDebug Active Probes	38
Record answers to questions below	40
Answers to Questions (results can slightly vary on your system).....	41
List of Figures	42

Introduction

This tutorial demonstrates how to implement a basic SmartFusion2 / IGLOO2 FPGA fabric design using SmartDesign.

The design drives several LEDs on the SmartFusion2 or IGLOO2 target board. A counter output is used to drive the LEDs. The SmartDebug Live Probes are used to drive another LED on the target board.

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

- Creating a Libero SoC project
- Implementing a SmartFusion2 / IGLOO2 fabric design with SmartDesign
- Simulating the design
- Importing a PDC file, running layout and programming the SmartFusion2 / IGLOO2 silicon
- Using the SmartDebug Live Probes

Components Used in the Design

This tutorial uses the SmartFusion2 / IGLOO2 FPGA fabric, the on-chip 25/50 MHz RC oscillator and the Fabric CCC and Live Probe A.

Tutorial Requirements

Software Requirements

This tutorial requires the following software installed on your computer:

- Microsemi Libero SoC v12.5

Hardware Requirements

This tutorial can be used with the following SmartFusion2 board:

- SMF2000 SmartFusion2 board

Note: The Free Libero Silver license can be used with this kit.

Extracting the source files

Extract *SF2_Fab_Probe.zip* to extract the required lab files to the <C: or D:>\Microsemiprj folder on the HDD of your PC. Confirm that a folder named *SF2_Fab_Probe* containing a sub-folder named *Source_files* was extracted.

Step 1 – Creating the Design

In this step you will create the fabric design using SmartDesign. Some source files have been provided in the *Source_files* folder.

Launching Libero SoC and creating a project

1. Click **Start > Programs > Microsemi > Libero SoC v12.5** or click the shortcut on your desktop. The Libero SoC Project Manager will open.

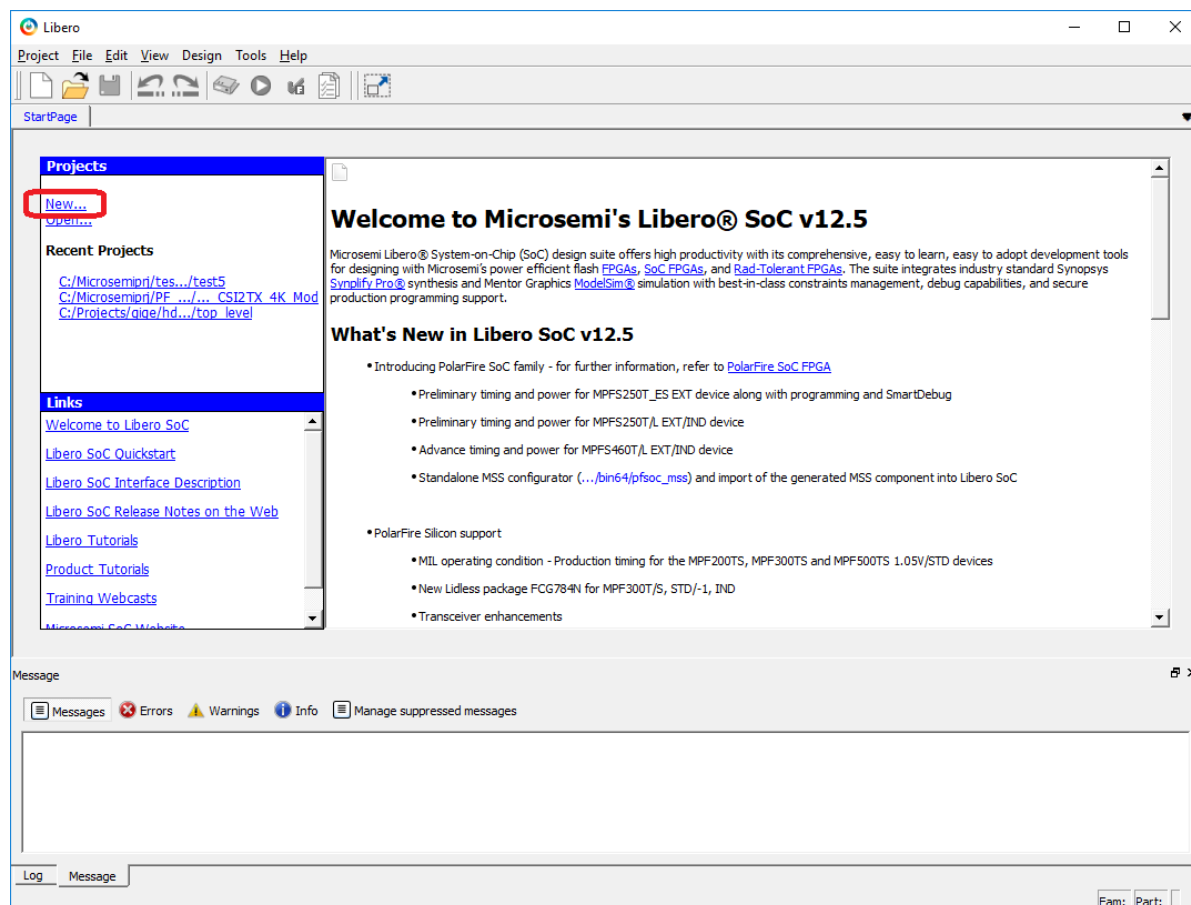
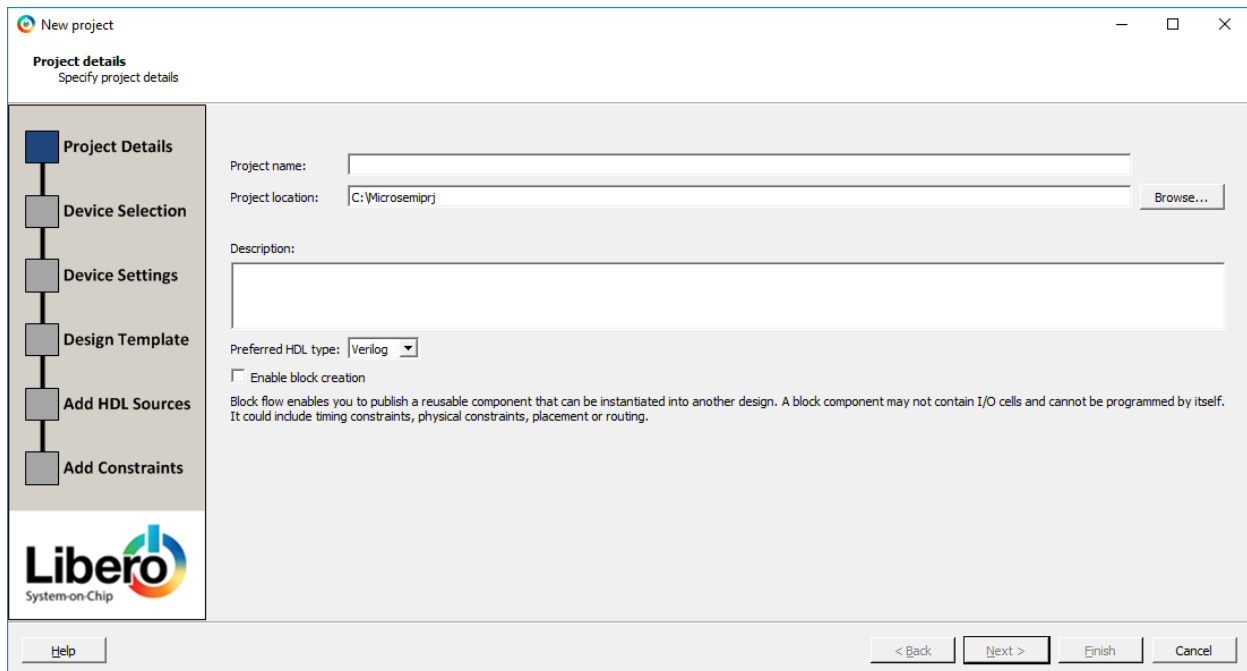


Figure 1 - Libero SoC Project Manager

2. Create a new project by selecting **New** on the Start Page tab (circled in the figure above), or by clicking **Project > New Project** from the Libero SoC menu. The New Project wizard will open.



The "New project" dialog box is shown with the "Project details" tab selected. The left sidebar contains a vertical list of steps: Project Details (selected), Device Selection, Device Settings, Design Template, Add HDL Sources, and Add Constraints. The main area contains the following fields:

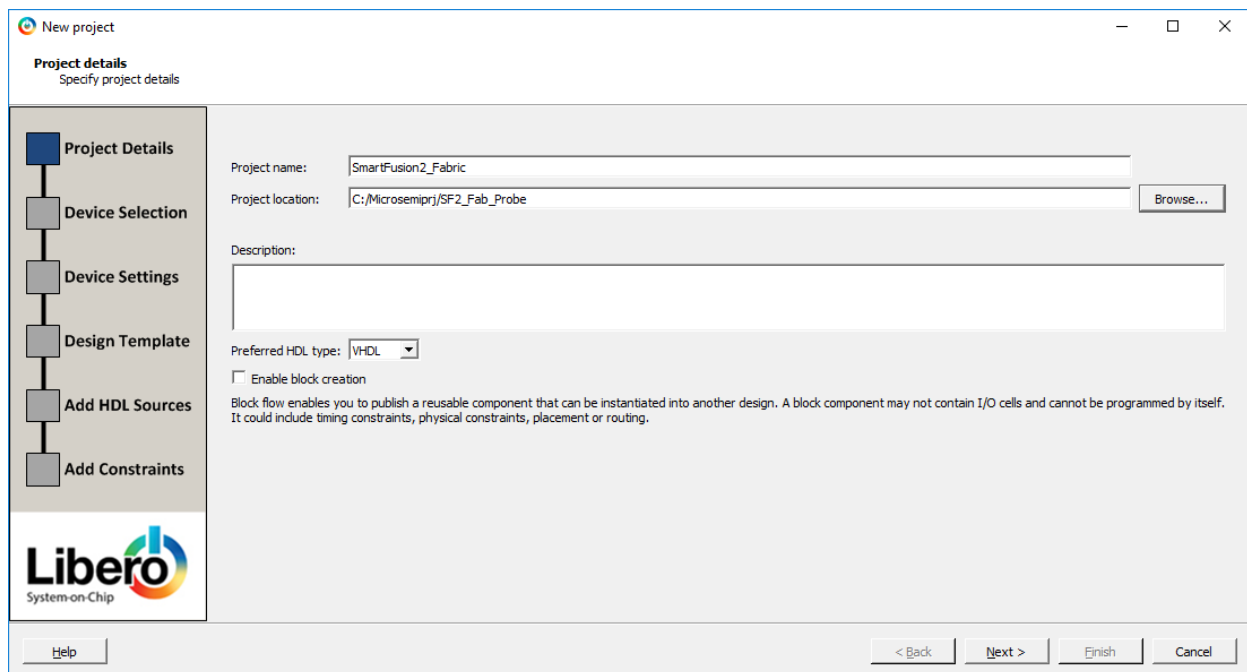
- Project name: [Empty text box]
- Project location: [C:\Microsemiprj] [Browse...]
- Description: [Empty text box]
- Preferred HDL type: Verilog [v]
- ☐ Enable block creation

Below the "Enable block creation" checkbox, a note states: "Block flow enables you to publish a reusable component that can be instantiated into another design. A block component may not contain I/O cells and cannot be programmed by itself. It could include timing constraints, physical constraints, placement or routing."

At the bottom, there are four buttons: Help, < Back, Next >, Finish, and Cancel.

Figure 2 - Libero SoC New Project dialog box

3. Enter the information shown below in the Project Details page of the New Project dialog box then click **Next**:
- Project Name: SmartFusion2_Fabric
 - Project Location: <C: or D:>\Microsemiprj\SF2_Fab_Probe (depending on where you extracted the source files)
 - Preferred HDL type: Verilog or **VHDL**, select **VHDL** for this lab example
 - Enable Block Creation: **un-checked**



The "New project" dialog box is shown with the "Project details" tab selected. The left sidebar contains a vertical list of steps: Project Details (selected), Device Selection, Device Settings, Design Template, Add HDL Sources, and Add Constraints. The main area contains the following fields:

- Project name: SmartFusion2_Fabric
- Project location: C:\Microsemiprj\SF2_Fab_Probe [Browse...]
- Description: [Empty text box]
- Preferred HDL type: VHDL [v]
- ☐ Enable block creation

Below the "Enable block creation" checkbox, a note states: "Block flow enables you to publish a reusable component that can be instantiated into another design. A block component may not contain I/O cells and cannot be programmed by itself. It could include timing constraints, physical constraints, placement or routing."

At the bottom, there are four buttons: Help, < Back, Next >, Finish, and Cancel.

Figure 3 - Project Details

4. Enter the following in the Device Selection page of the New Project dialog box, select M2S010-VF400 device then click **Next**:
- Family: SmartFusion2
 - Die: M2S010
 - Package: 400 VF
 - Speed: STD
 - Core Voltage(V): 1.2
 - Range: COM

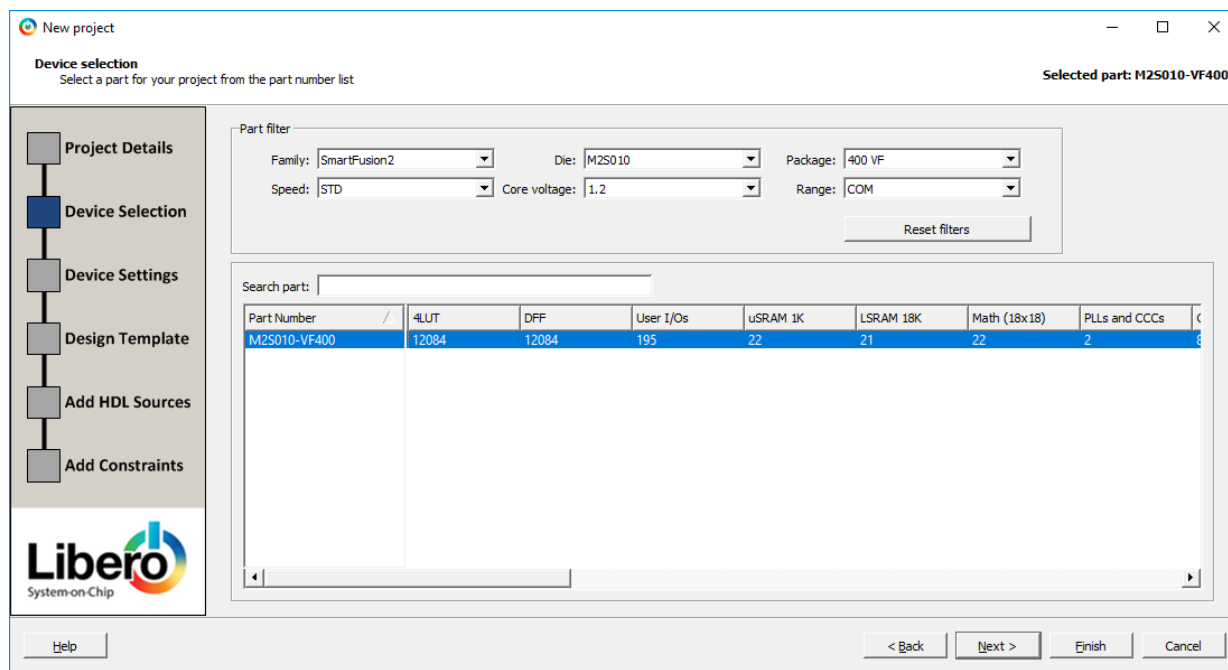
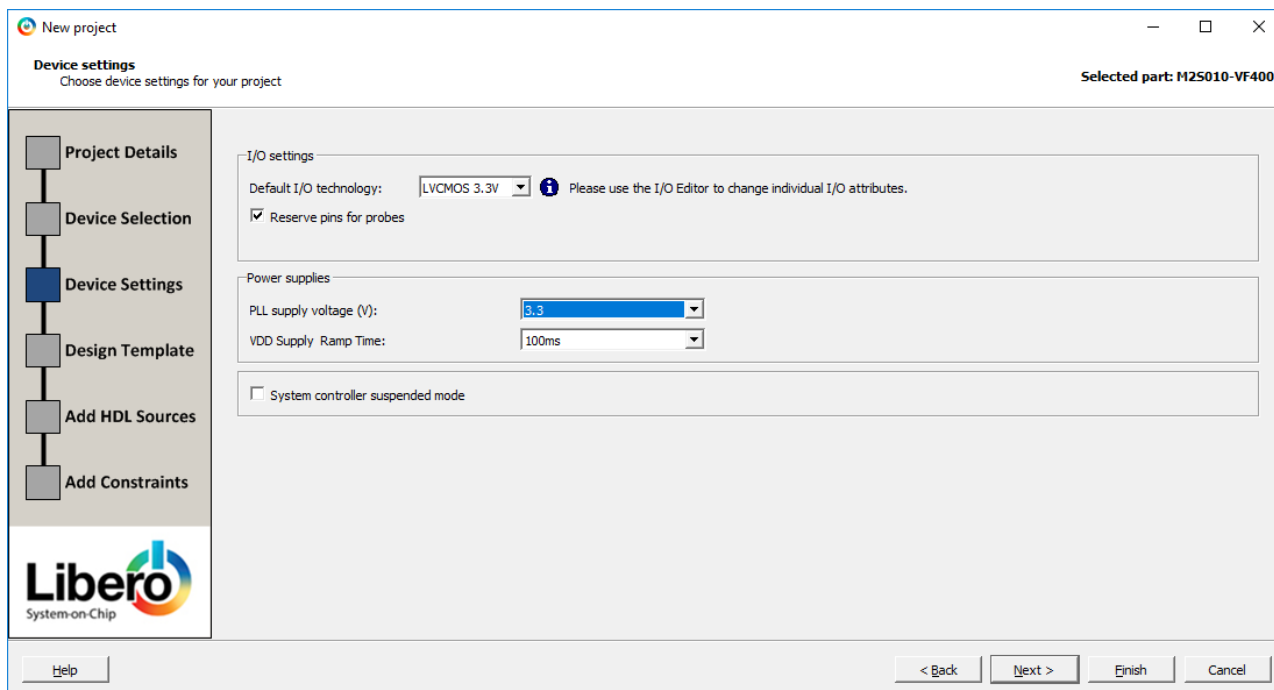


Figure 4 - Device selection (settings for SMF2000 board shown)

5. Enter the following in the Device Settings page of the New Project dialog box then click **Next**:
- Default I/O Technology: **LVC MOS 3.3V**
 - Reserve Pins for Probes: checked (default)
 - Power Supplies
 - PLL Supply Voltage (V): **3.3V**
 - VDD Supply Ramp Time: 100ms (default)
 - System Controller Suspend Mode: un-checked

The PLL Supply voltage can be either 2.5V or 3.3V. The voltage setting in the New Project dialog box must match the PLL Analog Supply voltage on the board to ensure the PLL works correctly.



New project

Device settings
Choose device settings for your project

Selected part: M2S010-VF400

Project Details

Device Selection


Device Settings

Design Template

Add HDL Sources

Add Constraints

I/O settings

Default I/O technology: LVCMOS 3.3V  Please use the I/O Editor to change individual I/O attributes.

☒ Reserve pins for probes

Power supplies

PLL supply voltage (V): 3.3

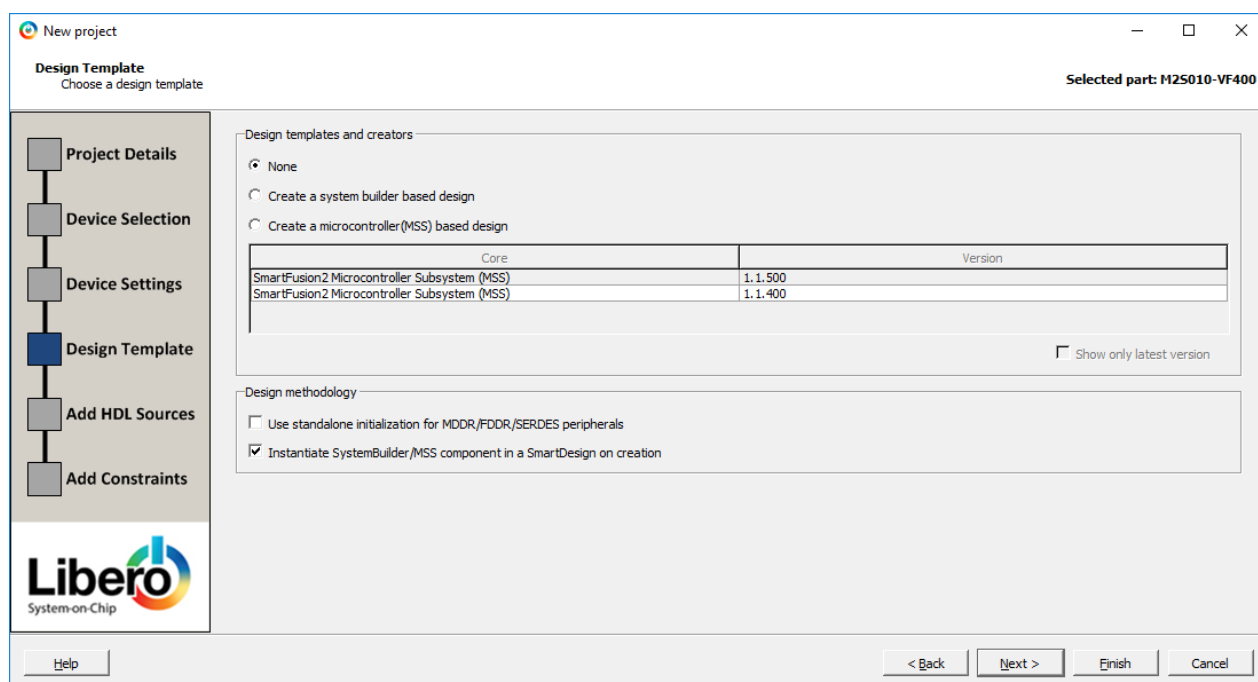
VDD Supply Ramp Time: 100ms

☐ System controller suspended mode

Help < Back Next > Finish Cancel

Figure 5 - Device Settings (settings for SMF2000 board shown)

6. Enter the following in the Design Template page of the New Project dialog box then click **Next**:
- Design Templates and Creators: **None** (this design does not use the SmartFusion MSS or IGLOO2 HPMS)
 - Design Methodology:
 - Use Standalone Initialization for MDDR/FDDR/SERDES peripherals: un-checked
 - Instantiate SystemBuilder/MSS component in a SmartDesign on creation: checked



New project

Design Template
Choose a design template

Selected part: M2S010-VF400

Project Details

Device Selection

Device Settings

Design Template

Add HDL Sources

Add Constraints

Design templates and creators

☒ None

☐ Create a system builder based design

☐ Create a microcontroller(MSS) based design

Core	Version
SmartFusion2 Microcontroller Subsystem (MSS)	1.1.500
SmartFusion2 Microcontroller Subsystem (MSS)	1.1.400

☐ Show only latest version

Design methodology

☐ Use standalone initialization for MDDR/FDDR/SERDES peripherals

☒ Instantiate SystemBuilder/MSS component in a SmartDesign on creation

Help < Back Next > Finish Cancel

Figure 6 - Design Template settings (all kits)

7. The Add HDL Source Files page will be visible. A VHDL source file has been provided. Import the file into the project by clicking **Import File**.
8. Enter the following in the Import Files dialog box then click **Open**:
 - Location: <C or D:>\Microsemiprj\SF2_Fab_Probe\Source_files
 - File name: count24.vhd
 - Files of type: HDL Source Files (*.vhd *.v *.h)

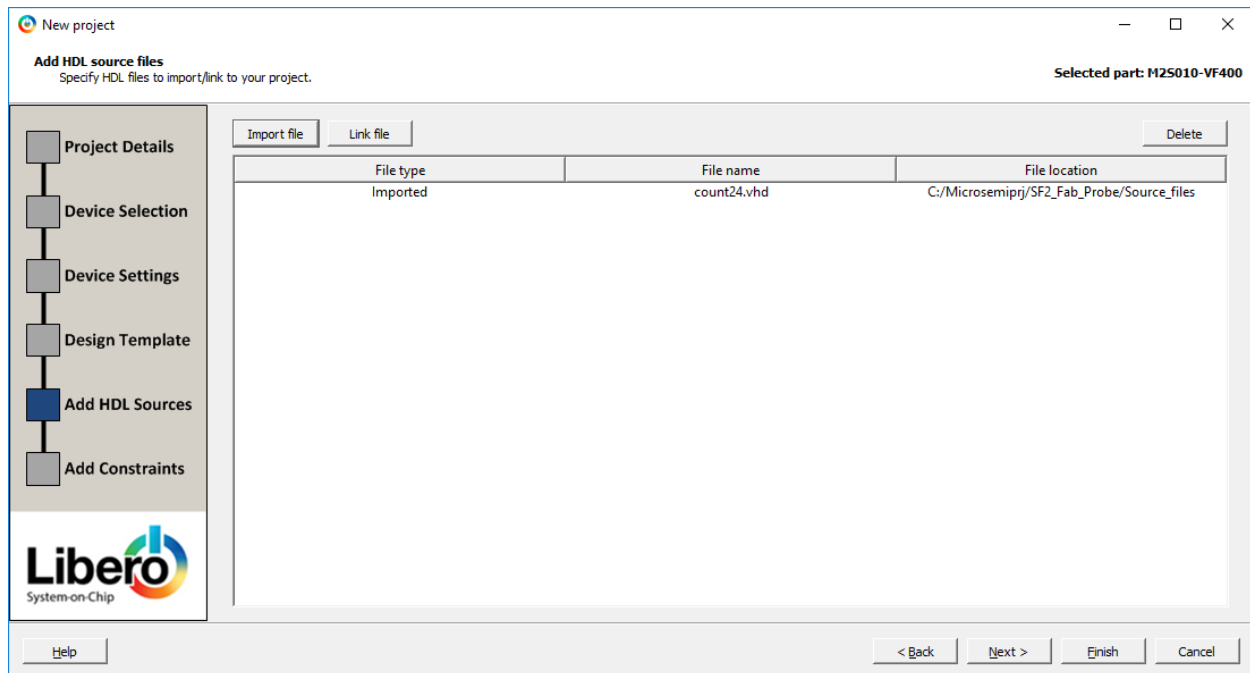


Figure 7 - Imported HDL source files

9. Click **Next**. The Add Constraints page will be visible. An I/O constraint file has been provided. Import the file into the project by clicking **Import File**.
10. Enter the following in the Import Files dialog box then click **Open**:
 - Location: <C or D:>\Microsemiprj\SF2_Fab_Probe\Source_files
 - File name: Fabric_Top.pdc
 - Files of type: I/O Constraint Files (*.pdc)

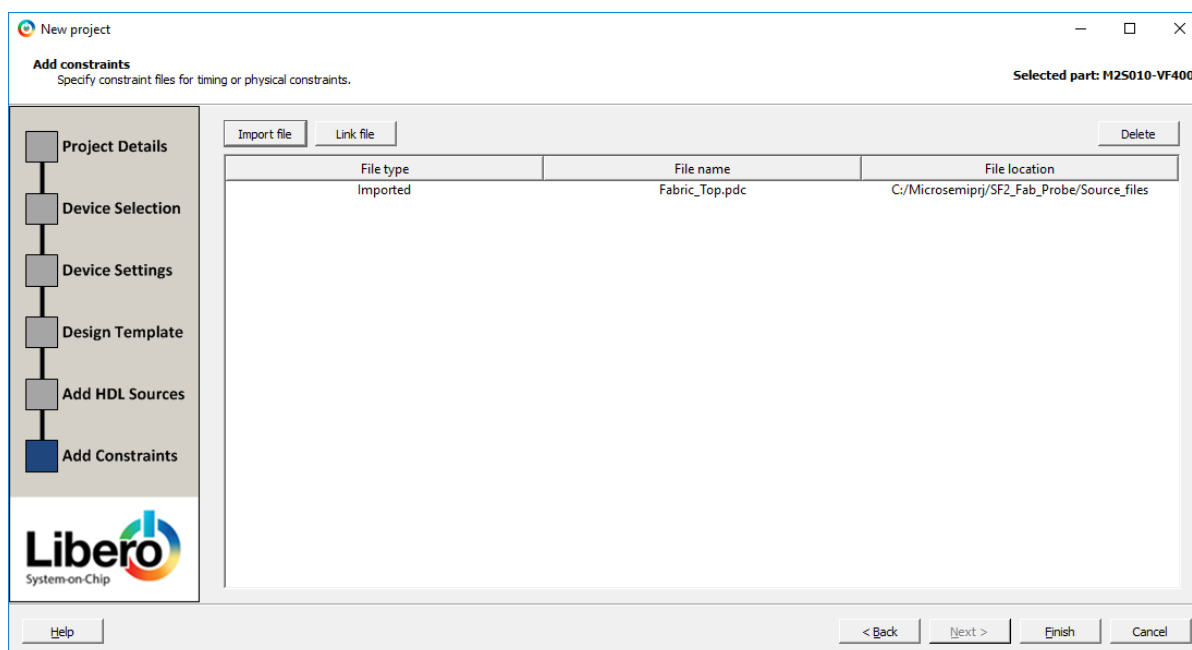


Figure 8 - Imported I/O Constraint file

11. Click **Finish** in the New Project dialog box.
12. The HDL source file will be visible on the Design Hierarchy tab after clicking **Build Hierarchy**. Both the HDL source file and the I/O constraint file will also be visible on the Files tab.

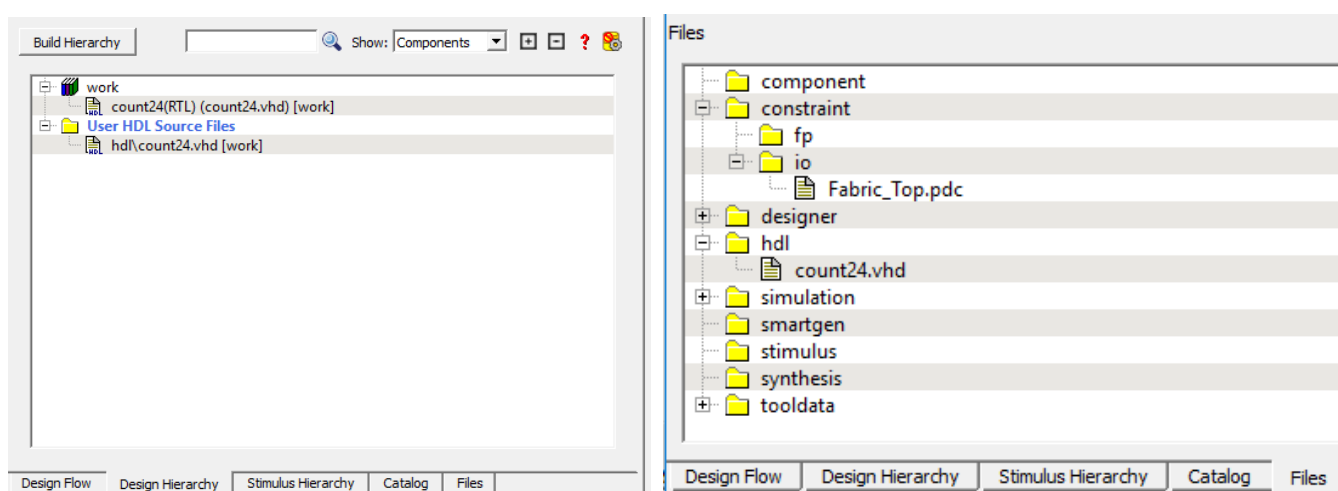


Figure 9 - HDL source file and I/O constraint file in Libero SoC

Implementing the design with SmartDesign

1. Open the SmartDesign canvas by selecting **File > New > SmartDesign** from the menu or by double-clicking **Create SmartDesign** under Create Design on the Design Flow tab.

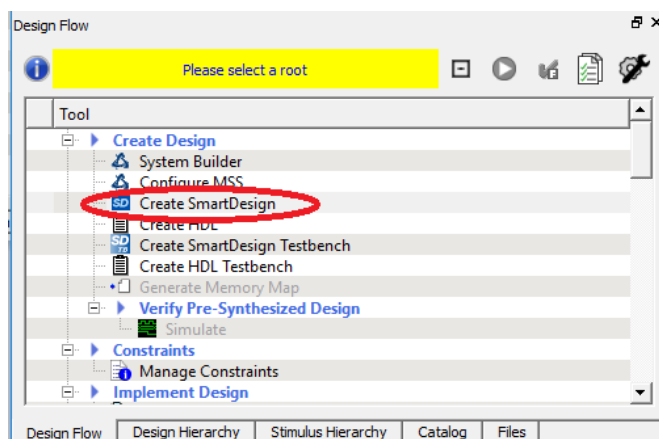


Figure 10 - Opening the SmartDesign canvas

2. Enter *Fabric_Top* in the Create New SmartDesign dialog box then click **OK**. Save the component with CTRL+SHIFT+S.

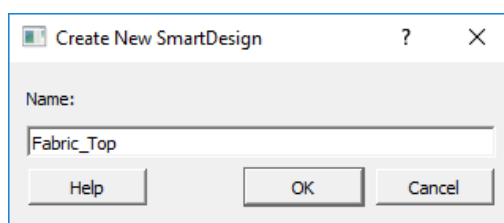


Figure 11 - Entering SmartDesign name

3. Click **Build Hierarchy** on the Design Hierarchy tab.
4. Drag the count24 component from the Design Hierarchy tab to the SmartDesign canvas.

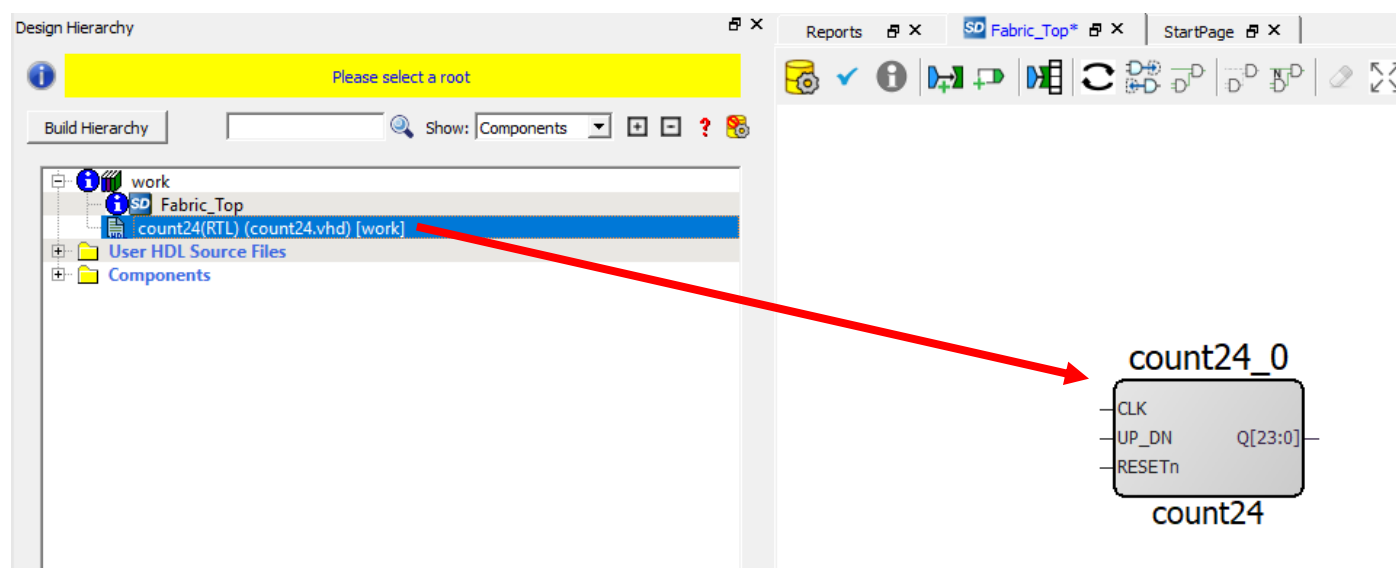


Figure 12 – Component pasting

This design uses a fabric CCC to generate the 1 MHz internal clock. The CCC reference clock is the 25/50 MHz RC oscillator. In the next steps you will configure the CCC to output a 1 MHz clock and configure the on-chip oscillator.

- Expand **Clock & Management** in the IP catalog.

Note: If the IP Catalog is not visible, click on View > Windows > Catalog.

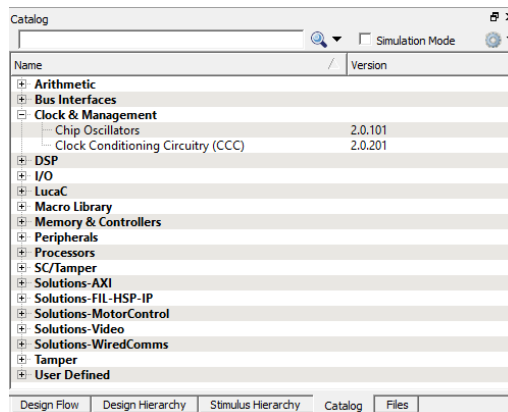


Figure 13 – Clock & Management category of the Libero SoC IO Catalog

- Drag an instance of the Clock Conditioning Circuitry (CCC) v2.0.201 component into the SmartDesign canvas.
- Keep **FCCC_C0** for the component name and click OK.

The components must be spelled exactly as directed. This nomenclature is necessary because the pre-written codes include variables that use the names of the components. Naming the components differently can cause errors.

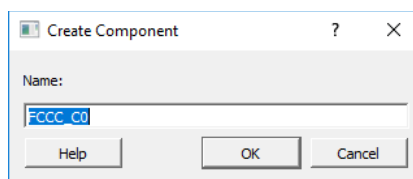


Figure 14 - Component naming

- FAB CCC Configurator will open, otherwise double-click the FCCC_C0 component in the SmartDesign canvas to open the FAB CCC Configurator.

9. Select the Basic tab in the FAB CCC configurator. Enter the following:

- Reference clock: Select Oscillators > 25/50 MHz Oscillator from the pull-down menu
- GL0: checked;
 - Exact Value: checked
 - Frequency: **1 MHz**

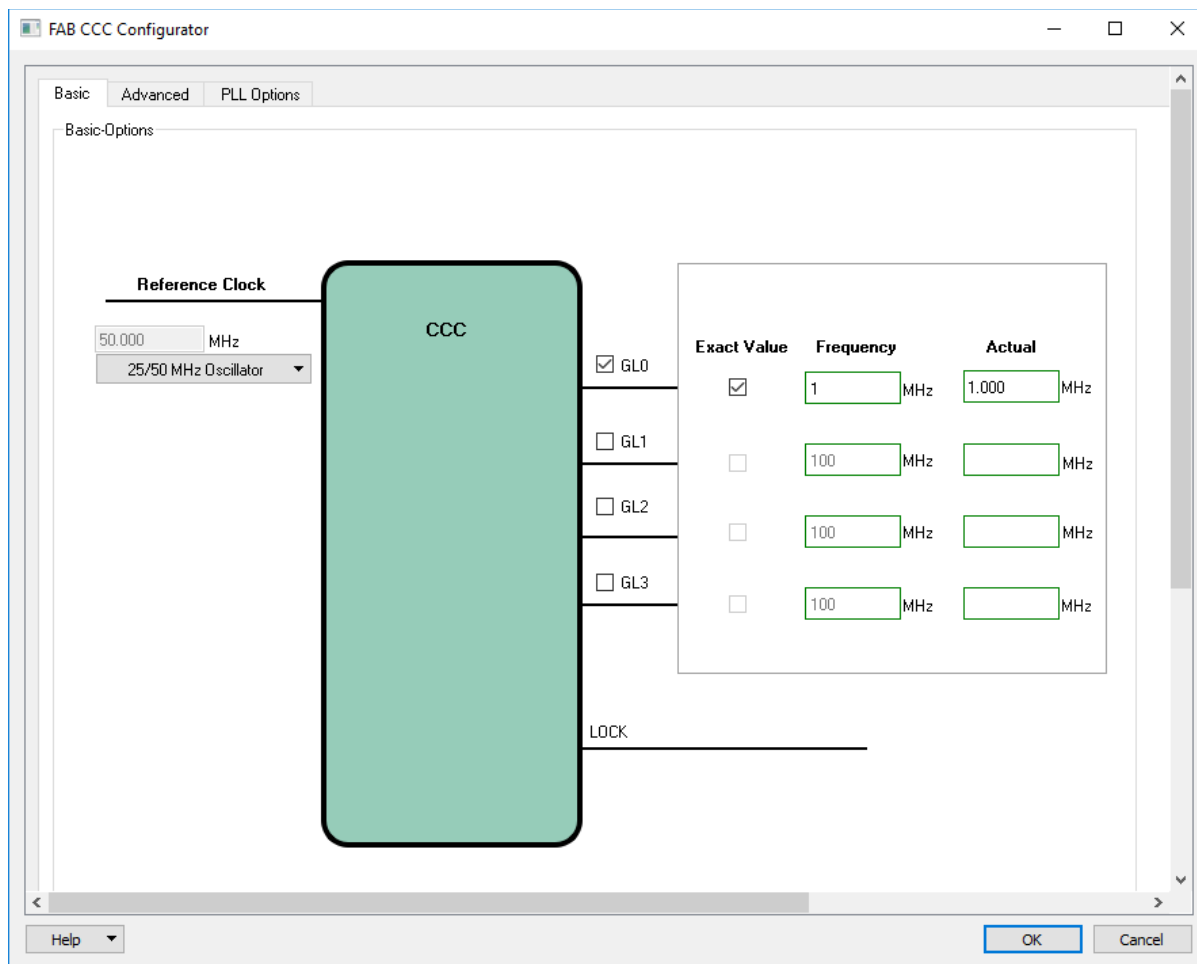


Figure 15 - Configuring the fabric CCC

10. Click **OK** to close the FAB CCC Configurator.

11. Drag an instance of the Chip Oscillators v2.0.101 component from the IP catalog into the SmartDesign canvas.

12. Keep **OSC_C0** for the component name and click OK.

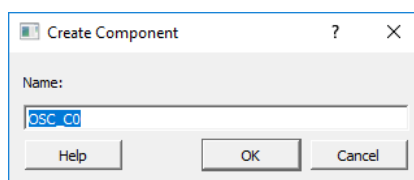


Figure 15 - Component naming

13. Chip Oscillators Configurator will open, otherwise double-click the OSC_C0 component in the SmartDesign canvas to open the Chip Oscillators configurator.

14. Configure the 50 MHz RC oscillator to drive the fabric CCC as follows:

- Enable the On-chip 25/50 MHz RC Oscillator
- Select Drives Fabric CCC(s)

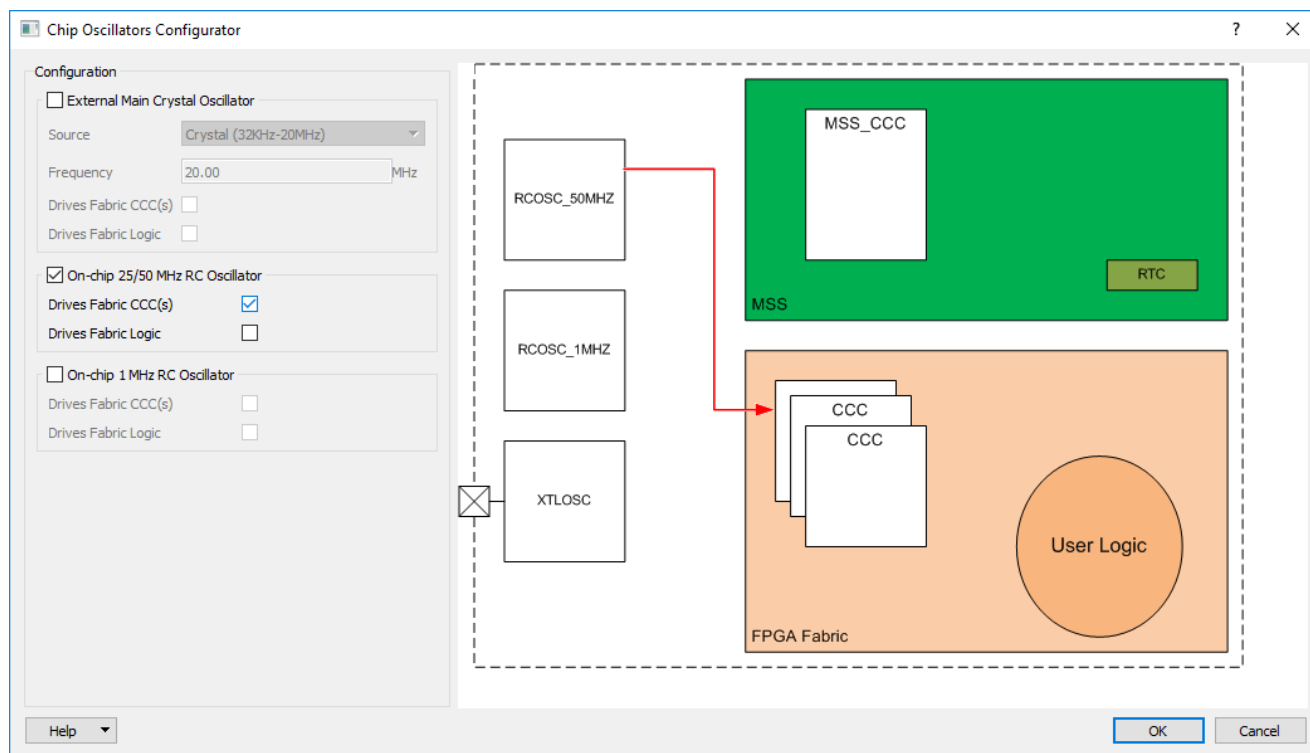


Figure 16 - Configuring the Chip Oscillators (SmartFusion2 target)

15. Click **OK** to close the Chip Oscillators configurator.

16. Expand **Macro Library** in the Libero SoC IP catalog.

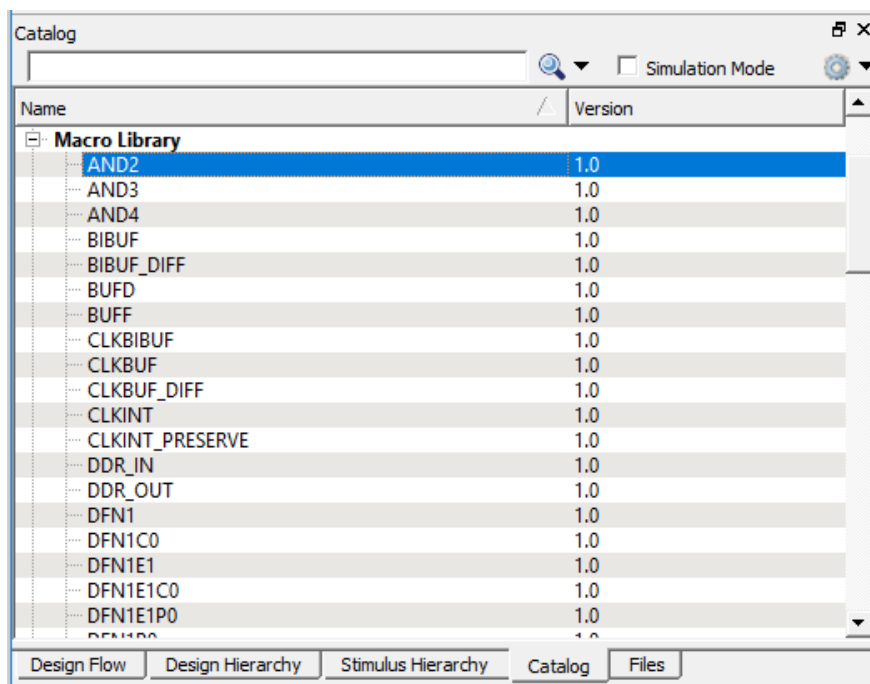


Figure 17 – Macro Library category of the Libero SoC IP Catalog

17. Drag an instance of AND2 and SYSRESET into the SmartDesign Canvas.

Tip: to make it easier to find the macros listed above, type a few letters of the macro name in the IP Catalog search field.

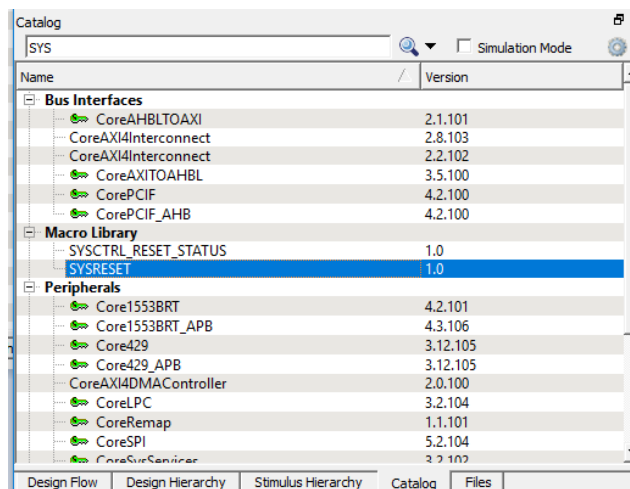


Figure 18 - IP Catalog Search field

18. Drag an instance of LIVE_PROBE_FB into the SmartDesign Canvas. The LIVE_PROBE_FB macro allows the SmartDebug Live Probes to be re-directed into the FPGA fabric. One of the outputs will be connected to an LED on the target board.

19. After adding the components, the SmartDesign will resemble the figure below. If needed, drag the components to improve the appearance of the canvas.

Tip: expand the canvas area by selecting **View > Maximize Work Area** or click the icon on the tool bar ().

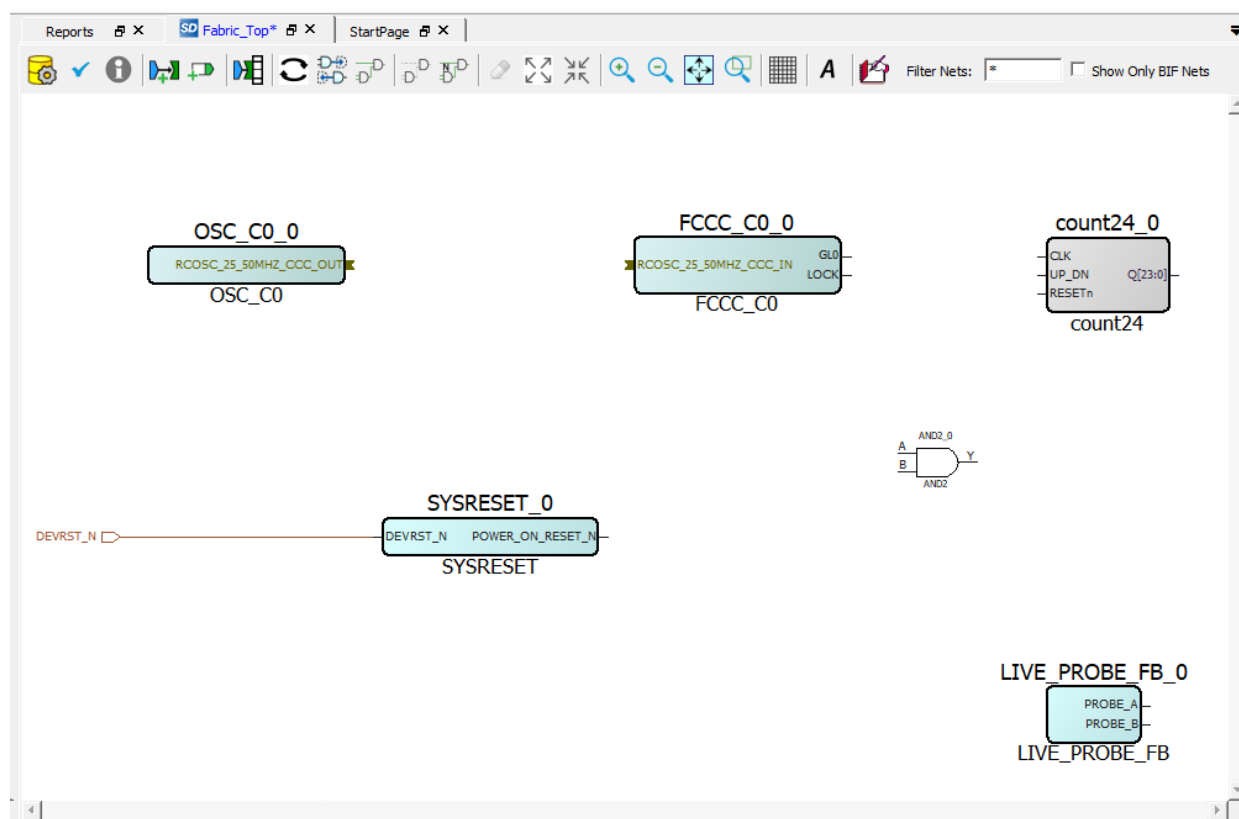



Figure 19 - SmartDesign canvas after adding components

Making connections in the canvas

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



20. Select **SmartDesign > Connection Mode** from the Libero SoC menu or click the Connection Mode icon ().
21. Connect the RCOSC_25_50MHZ_CCC_OUT port of OSC_C0_0 component to the RCOSC_25_50MHZ_CCC_IN port of the FCCC_C0_0 component as follows:
 - Click and hold mouse button on the RCOSC_25_50MHZ_CCC_OUT port of the OSC_0 component.
 - Drag mouse to the RCOSC_25_50MHZ_CCC_IN port of FCCC_0 component.
 - Release mouse button on the RCOSC_25_50MHZ_CCC_IN port to make the connection.
22. Repeat the previous steps to make the connections shown in the table below.

From	To
FCCC_C0_0:GL0	count24_0:CLK
FCCC_C0_0:LOCK	AND2_0:A
SYSRESET_0:POWER_ON_RESET_N	AND2_0:B
AND2_0:Y	count24_0:RESETn

Table 2 – SmartDesign canvas connections

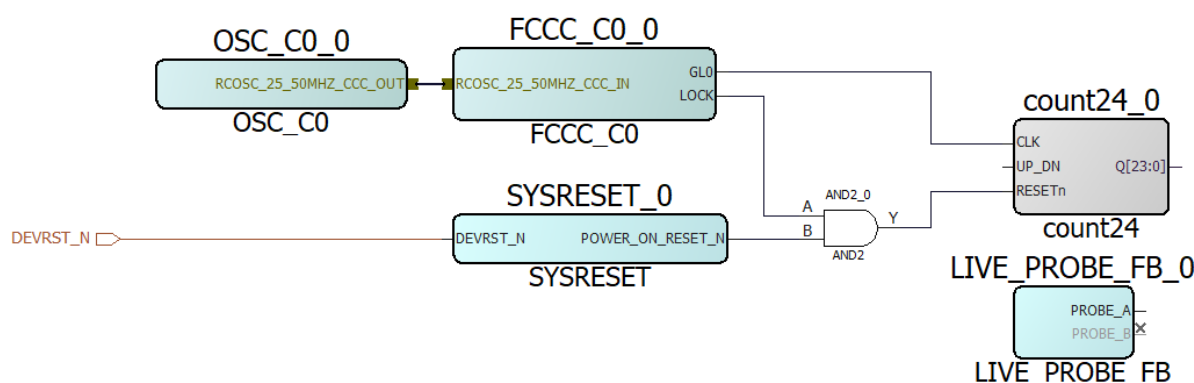


Figure 20 – Connection creating

23. Disable the SmartDesign connection mode by selecting **SmartDesign > Connection Mode** from the Libero SoC menu.
24. Promote the UP_DN port from count24_0 to the Top Level by selecting the UP_DN port and choosing **Promote to Top Level** from the popup menu (click right mouse button).
25. Rename UP_DN port to USER_BTN by choosing **Modify/Rename** from the popup menu (click right mouse button).

26. Expose bits 16 to 23 of count24_0 by selecting port Q[23:0] of count24_0 then right-clicking and selecting **Edit Slice**.

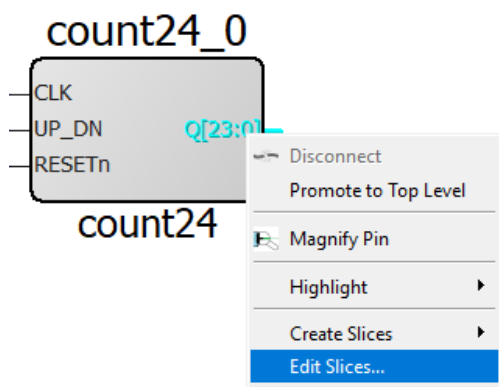


Figure 21 - Adding a slice to the Q[23:0] output

27. Click the + sign (circled in the figure below) in the Edit Slices – Q[23:0] dialog box.

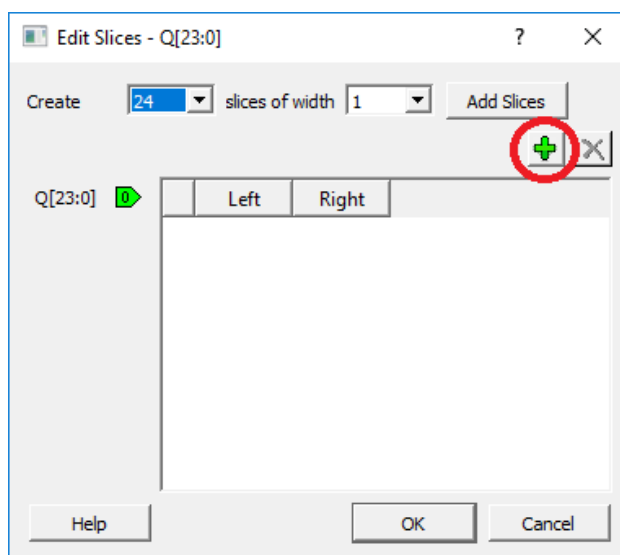


Figure 22 - Edit Slices dialog box

28. Enter 23 for the Left and 16 for the Right index to expose count24_0[23:16] and click **OK**.

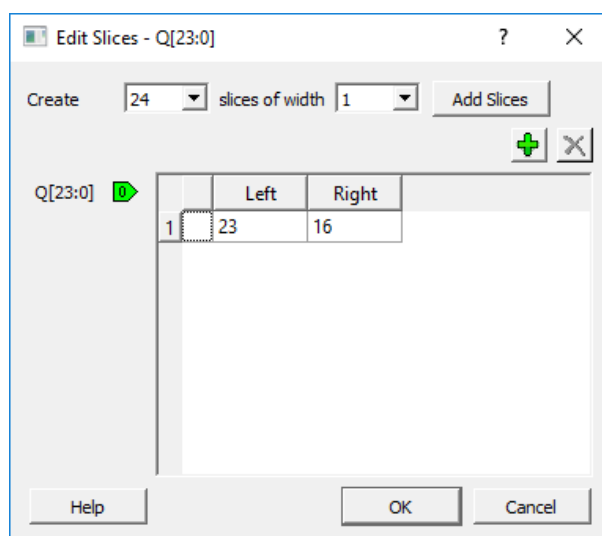


Figure 23 - Adding a slice to the count24_0 output

29. Mark the PROBE_B port of the LIVE_PROBE_FB_0 component unused by selecting the port then right-clicking and selecting **Mark Unused**.

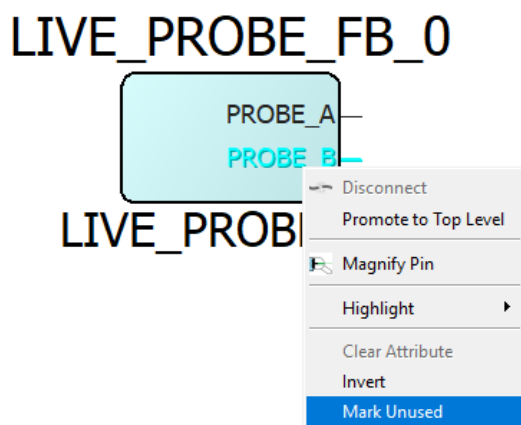



Figure 24 - Marking the PROBE_B port unused

30. Click on  icon on the toolbar or right-click anywhere in Design Canvas and select “Add Port”. The Add Port dialog box will appear.

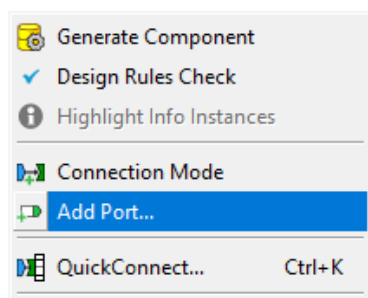


Figure 25 – Adding a port in Design Canvas

31. Enter the following in the Add Port dialog box then click **OK**:

- Name: LED[7:0]
- Direction : Output

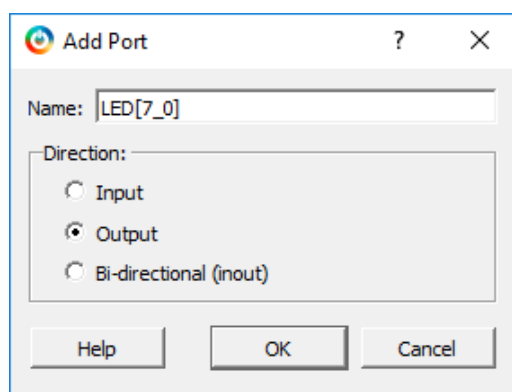


Figure 26 – Adding LED[1:8] output port in Design Canvas

32. If required, click the + sign on the Q[23:0] port of count24_0 to expose the slice that was created above.
33. Connect the Q[23:16] port of count24_0 to the LED[7:0] output port.
34. Add another output port called USER_LED.
35. Connect the PROBE_A port of the LIVE_PROBE_FB_0 component to the USER_LED output port.

36. After making the connections listed above the SmartDesign canvas will appear as shown in the figure below. You can drag the components or use the SmartDesign Auto Arrange feature to improve the appearance of the canvas.

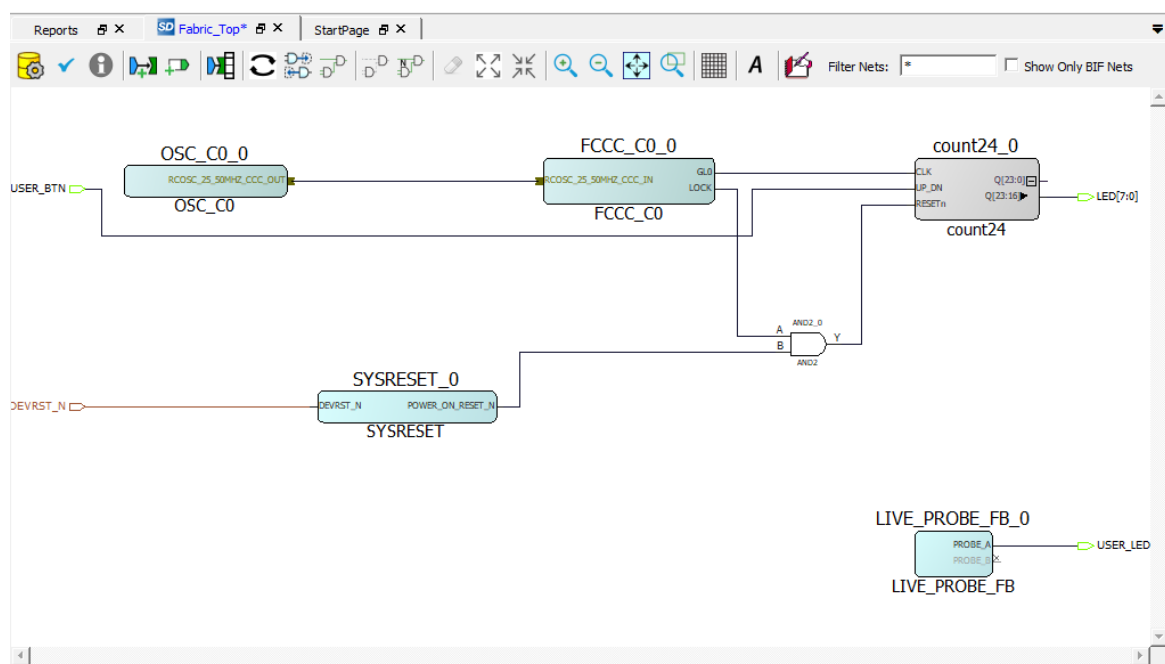



Figure 27 - SmartDesign canvas after making connections

37. Save the design (**File > Save Fabric_Top**).
38. Generate the design by clicking **SmartDesign > Generate Component** or by clicking the Generate Component icon on the SmartDesign toolbar ().
39. Restore the work area (**View > Restore Work Area**) if you expanded the work area earlier.
40. Confirm that the message "'Fabric_Top' was successfully generated" appears in the Libero Log window. The warning message "Floating output bus pin count24_0:Q[23:0]" can be ignored.
41. Set Design Top Level to "Fabric_Top" by right-clicking and selecting **Set As Root**, the active Top Level component is shown in **bold** font in the Design Hierarchy window.

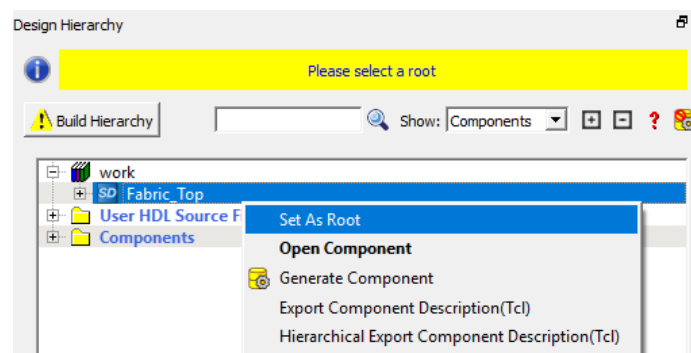


Figure 28 – Setting Fabric_Top as the Top Level (Root)

42. Click **Build Hierarchy** to update the Design Hierarchy.
43. Close the design (**File > Close Fabric_Top**).

Step 2 - Simulating the design

The next step is to simulate the design. A testbench, ModelSim macro file and a wave format file have been provided in the source files. The Fabric_Top module/entity contains a 24 bit counter. The simulation would take a long time because of the 24bit counter where we only need the upper 8 bits and because of the 1:50 clock ratio on the PLL. In order to accelerate the simulation of the design some of the counter bits are forced high in the ModelSim macro file during a second phase of the simulation.

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

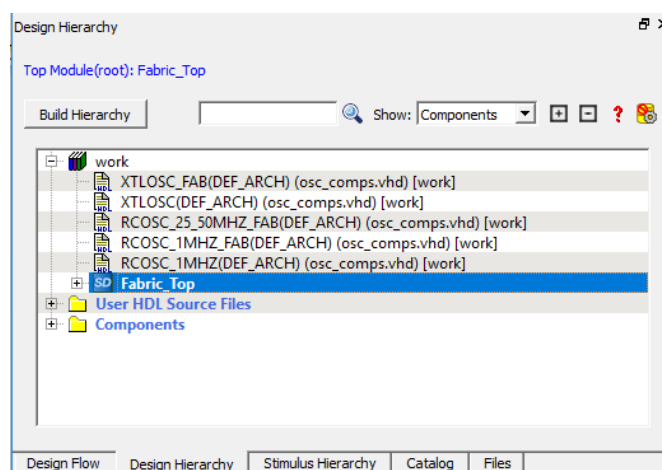


Figure 29 - Design Hierarchy with Fabric_Top as the Top Level

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

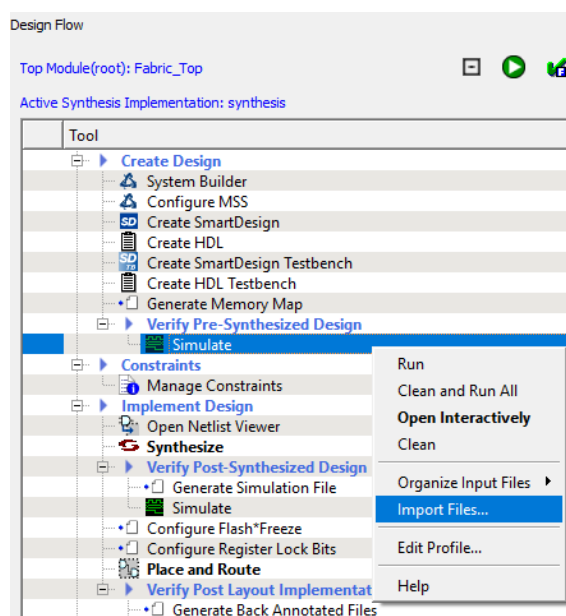


Figure 30 - Importing Simulation files

3. Import the ModelSim macro file and Wave format file.

Enter the following in the Import Files dialog box then click **Open**:

- Look in: <C: or D:>\Microsemiprj\SF2_Fab_Probe\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

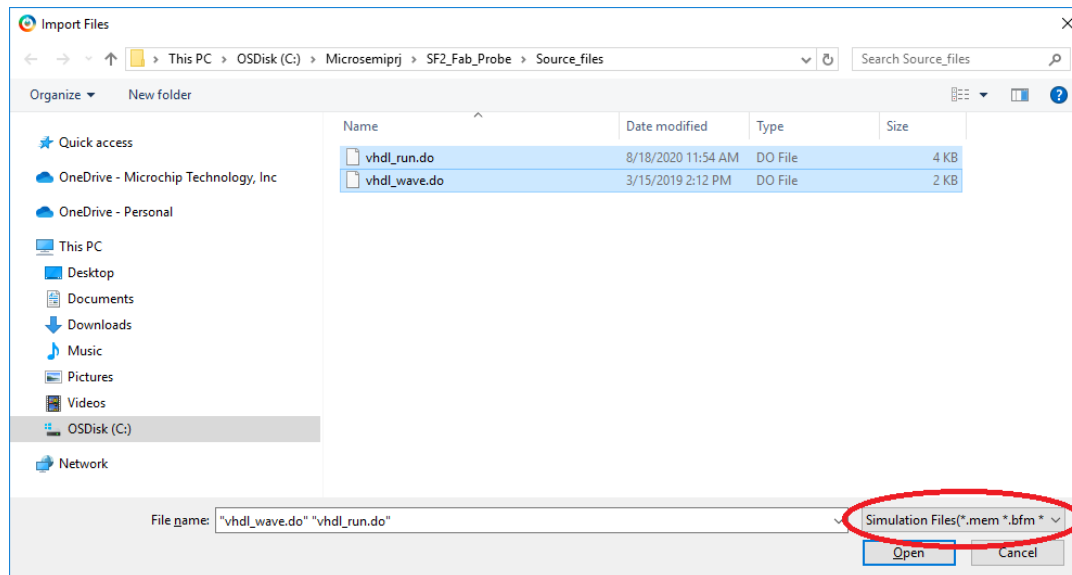


Figure 31 - Importing the ModelSim simulation files

4. Import the testbench by right-clicking **Simulate** under Verify Pre-Synthesized Design in the Design Flow window and selecting **Import Files**.

5. Enter the following in the Import Files dialog box then click **Open**:

- Look in: <C: or D:>\Microsemiprj\SF2_Fab_Probe\Source_files
- Files of type: HDL Stimulus Files (*.vhd *.v)
- File name: user_testbench.vhd

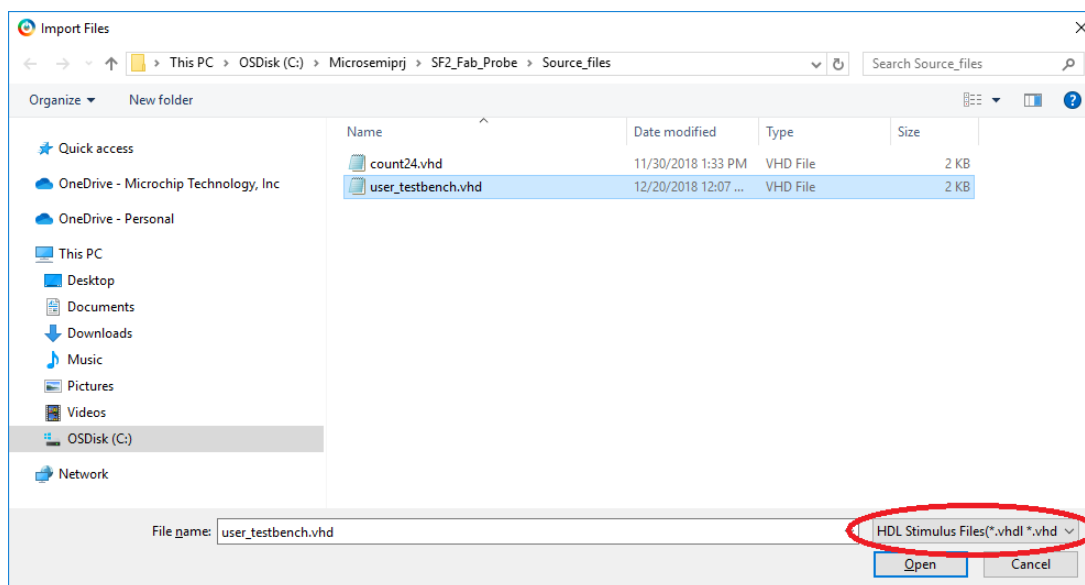


Figure 32 - Importing the testbench file

6. Select the Stimulus Hierarchy tab & click **Build Hierarchy**. The testbench (user_testbench.vhd) will be visible. Select the testbench, right-click and select **Set As Active stimulus**. A waveform symbol will indicate the active stimulus (circled below).

Note: If user_testbench.vhd is not visible on the Stimulus Hierarchy tab, the file was not imported as an HDL stimulus file. Re-import the file as described above.

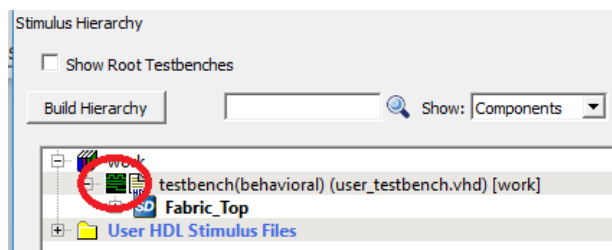


Figure 33 – Active Testbench in the project

The testbench and simulation files will be visible on the Libero SoC Files tab under Stimulus and Simulation.

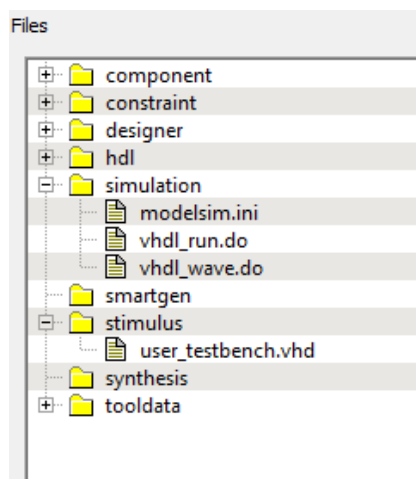


Figure 34 - Testbench and simulation files

7. Open the ModelSim macro file (vhdl_run.do) in the Libero SoC editor by double-clicking the filename on the Files tab.
8. Locate the variable PROJECT_DIR on line 17 of vhdl_run.do and confirm that it matches the location of your Libero SoC fabric tutorial project. The location is displayed at the top of the Libero SoC GUI (do not include "SmartFusion2_Fabric.prjx"). Edit the path if necessary (note that backward & forward slash usage differs between the Libero and Modelsim tool on Windows OS).

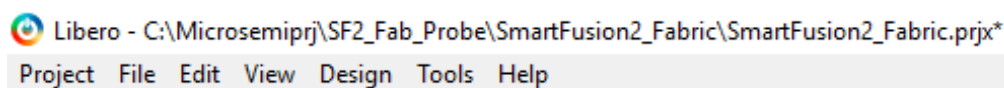


Figure 35 - Location of Libero SoC project

9. Locate the variable INSTALL_DIR on line 19 of vhdl_run.do and confirm that it matches the location of your Libero SoC installation. Edit the path if necessary. Ask the lab instructor if you do not know the Libero SoC installation path.
10. If you made changes, save the file (**File > Save vhdl_run.do**).
11. Scroll in the file to become familiar with the commands it contains. The command force -freeze forces some of the counter bits in Fabric_Top_0 high to speed up the simulation in the second phase of the simulation.
12. Close the editor (**File > Close vhdl_run.do**).

13. Open the Libero SoC project settings (**Project > Project Settings...**).

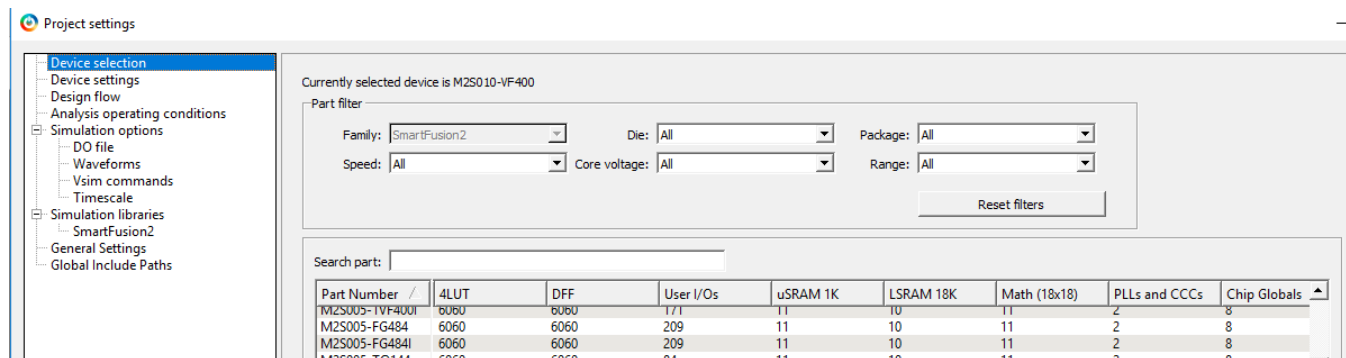


Figure 36 - Libero SoC Project Settings dialog box

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

15. Un-check "Use automatic DO file".

16. Click the browse button () next to "User defined DO file" and enter the following then click **Open**:

- Look in: <C: or D:>\Microsemiprj\SF2_Fab_Probe\SmartFusion2_Fabric\simulation
- Files of type: *.do
- File name: vhdl_run.do

The ModelSim macro file vhdl_run.do calls the Wave format file vhdl_wave.do, so there are no settings required for the Wave Format file.

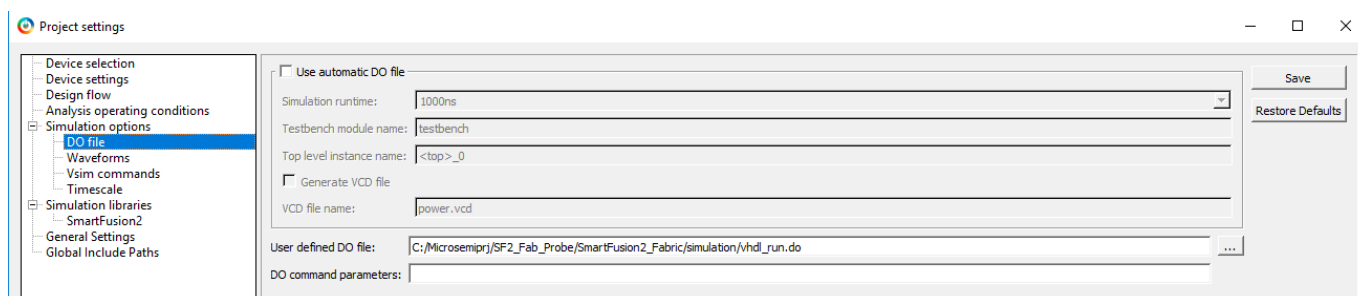


Figure 37 - Simulation options

17. Click **Save** then **Close** to close the Project settings dialog box.

18. Select **Project** -> **Tool Profiles...**

In order to use a ModelSim tool version which is supported by the Silver license, a new Simulation profile has to be added.

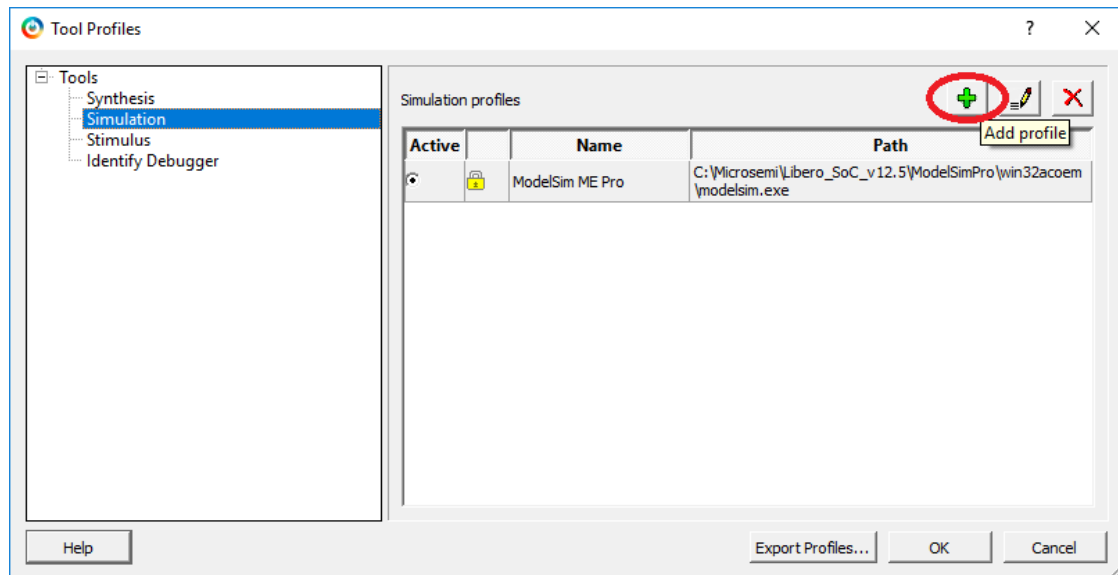


Figure 38 - Tool Profiles

19. Select Simulation under **Tools**, click Add profile + button and enter settings below, then click **OK**:

- Name: ModelSim ME
- Tool integration: ModelSim
- Browse to Location and select, e.g.: C:\Microsemi\Libero_SoC_v12.5\ModelSim\win32acoem\modelsim.exe

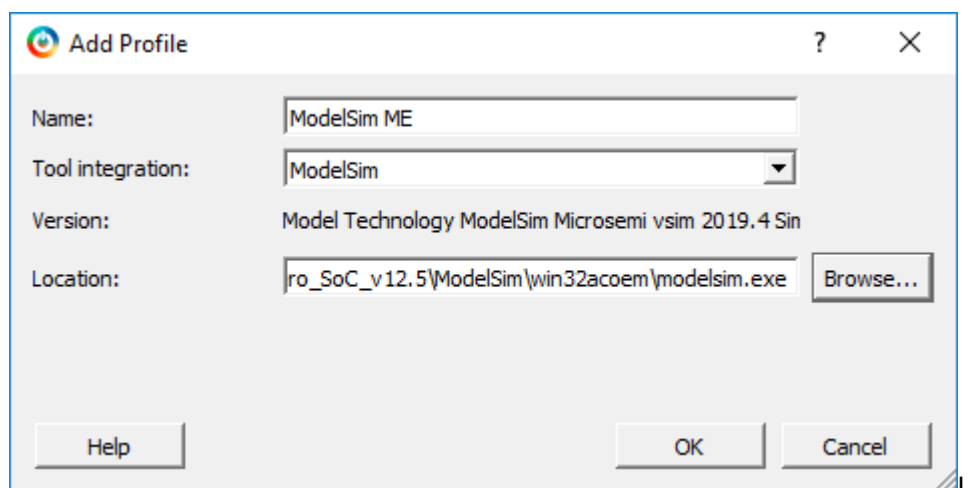


Figure 39 - ModelSim ME Profile

20. Make sure ModelSim ME is set as Active Simulation Profile, then press **OK**

21. Expand Verify Pre-Synthesized Design in the Design Flow window. Right-click **Simulate** and select **Open Interactively** to launch ModelSim in GUI mode.

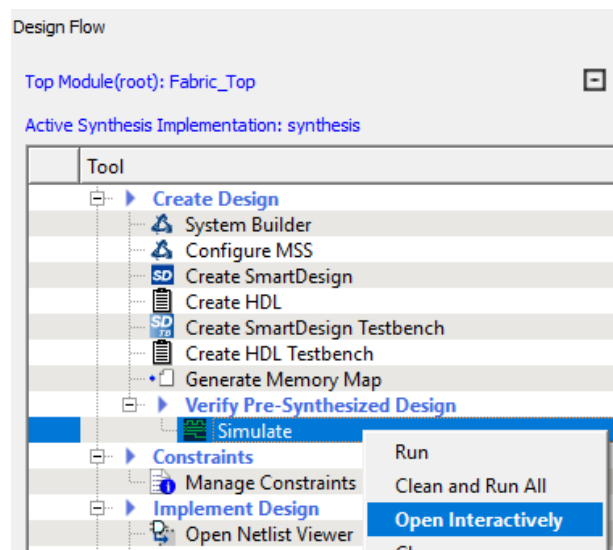


Figure 40 - Launching pre-synthesis simulation

22. The simulation will run for 2.0 ms. When finished, the Wave window should appear as shown below after selecting **Zoom full** from the popup menu (click right mouse button inside Wave – Default window).

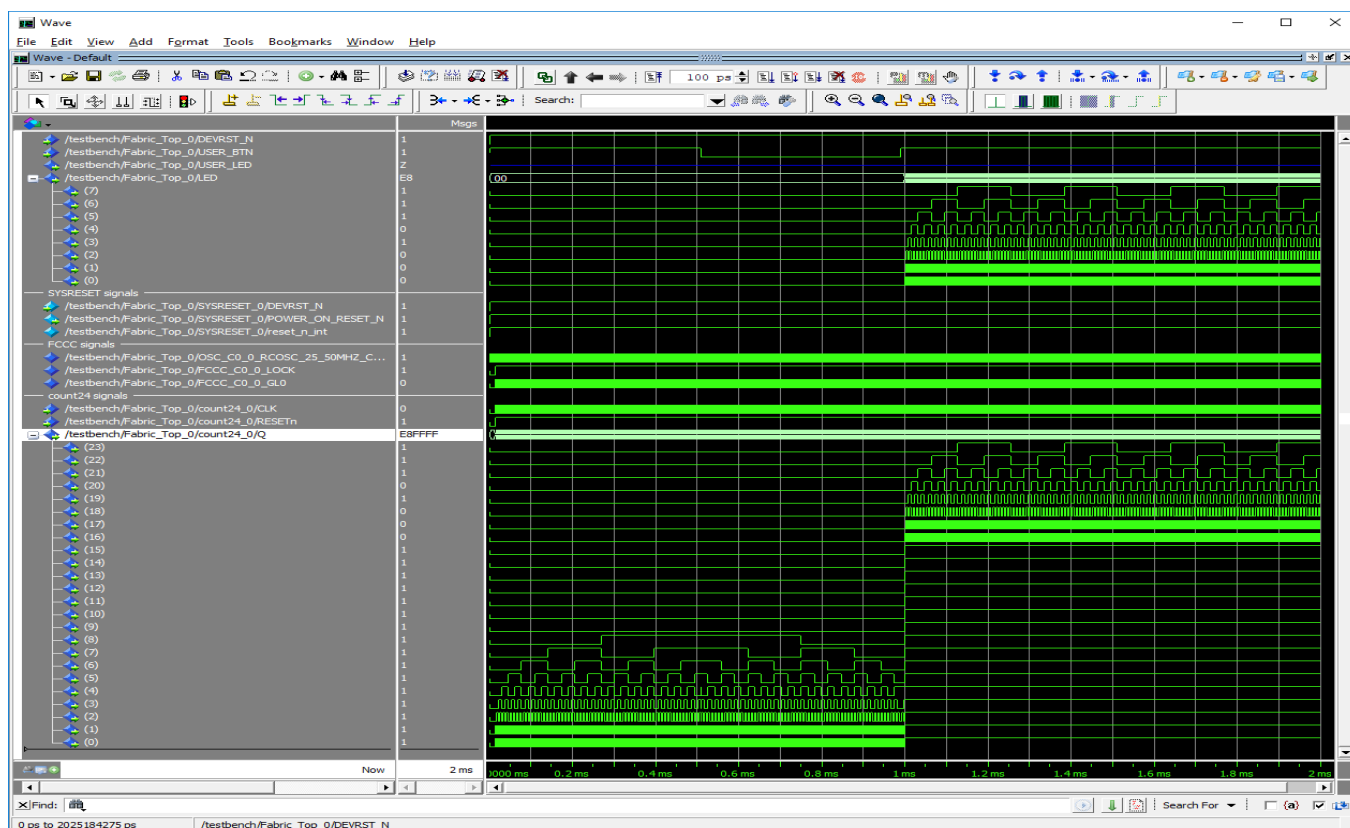


Figure 41 - ModelSim Wave window

23. Scroll in the wave window. The LED[7:0] port represents the LED driver (0 = LED off; 1 = LED on). The USER_LED port is the re-directed Live Probe output. In the simulation this appears as a Hi-Z signal. When SmartDebug is used, the output of any flip-flop can be re-directed to this output port.
24. Contact the lab instructor if your results do not match the figure above.
25. Close the ModelSim simulator (**File > Quit**). Click **Yes** when asked if you want to quit.

Step 3 – Editing the I/O Physical Constraint file

There are multiple ways to make I/O Assignments. In this lab we will use the I/O Physical Design Constraint (PDC) file that was imported when the project was created. The I/O pdc file must be edited to match the target board.

1. Double-click **Manage Constraints** in the Libero SoC Design Flow window to open the Libero SoC Constraint Manager.

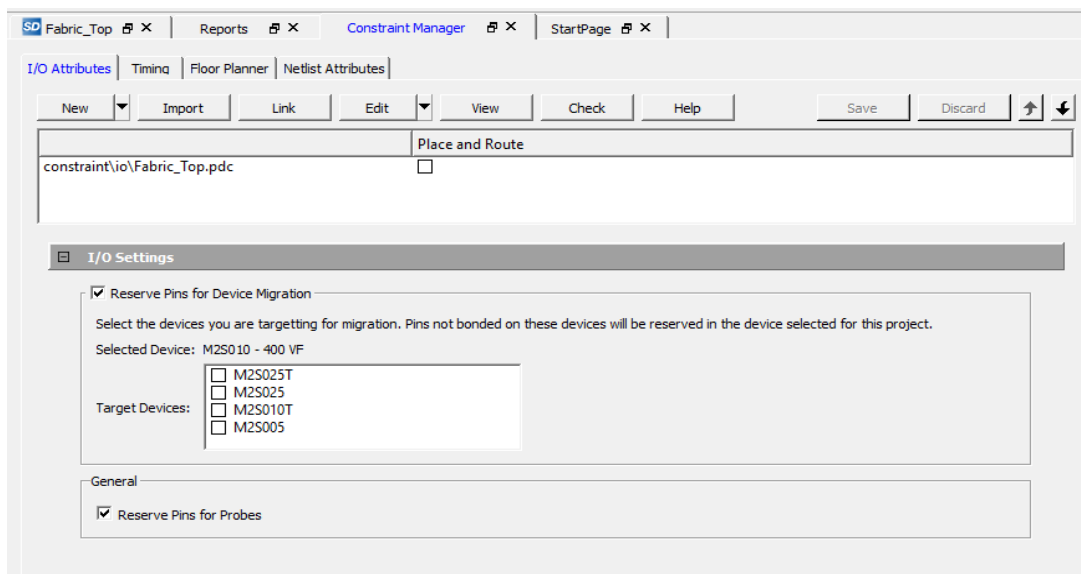


Figure 42 - Libero SoC Constraint Manager

2. Select the Constraint Manager I/O Attributes tab. Double-click Fabric_Top.pdc to open the file in the Libero SoC editor. Scroll in the file to become familiar with the syntax. The constraint set_iobank sets the input voltage of the bank to determine the allowable I/O standards; the constraint set_io sets the pin number and I/O specific attributes. The # symbol is a comment.

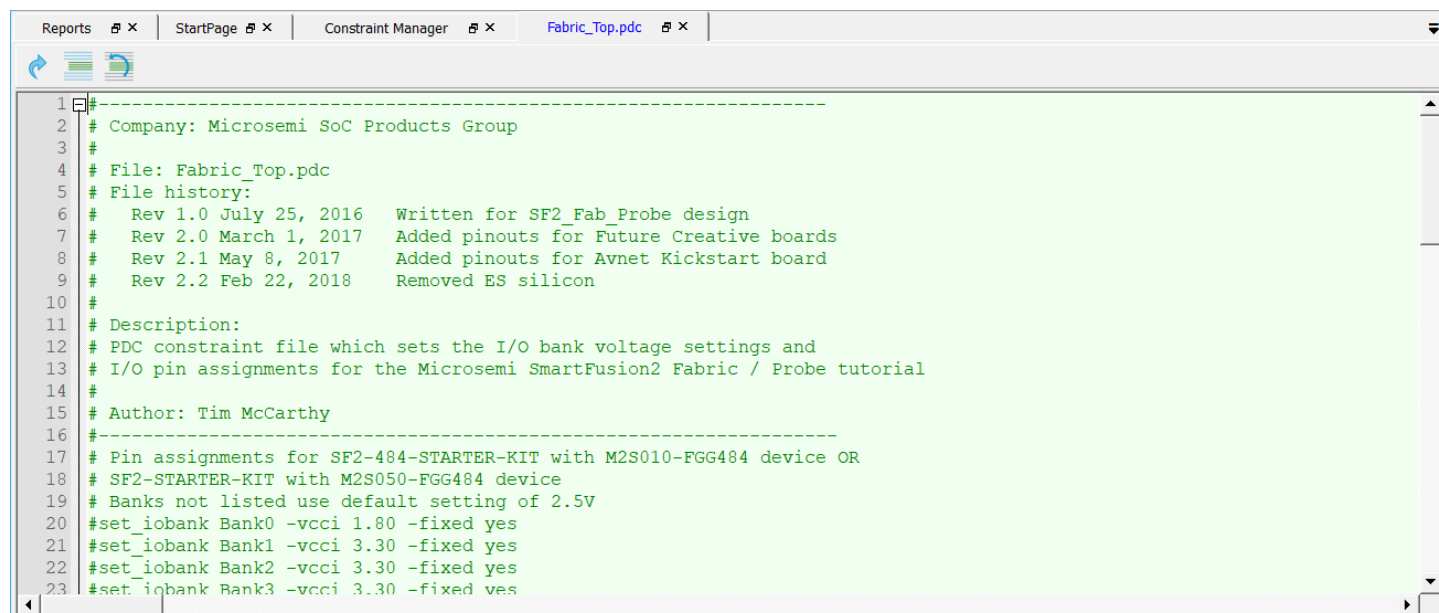


Figure 43 - PDC file in the Libero SoC constraint editor

A description of the PDC constraints syntax is available in the Libero Help (**Help > Help Topics > Manage Constraints > Assigning Design Constraints > Design Constraints Guide > Reference > Constraints by File Format > PDC Command Reference**).

For more information regarding pin assignments for the SmartFusion2 target board, refer to the schematic for the board, which is available from the Trenz Electronic website (http://www.trenz-electronic.de/fileadmin/docs/Trenz_Electronic/Modules_and_Module_Carriers/2.5x6.15/TEM0001/REV01/Documents/SCH-TEM0001-01-010C.PDF).

3. All assignments in the PDC file are commented out. Uncomment the lines starting with **set_iobank** and **set_io**.
4. Save the PDC file after making modifying (**File > Save Fabric_Top.pdc**).
5. Close the PDC file (**File > Close Fabric_Top.pdc**).
6. Enable (checkbox) Fabric_Top.pdc for **Place and Route** on the I/O Attributes tab of the Libero Constraint Manager.
7. Press **Save** to store the change in the I/O Attributes constraint file selection.
Confirm saving of changes in warning notice (design flow invalidation) with **OK**.

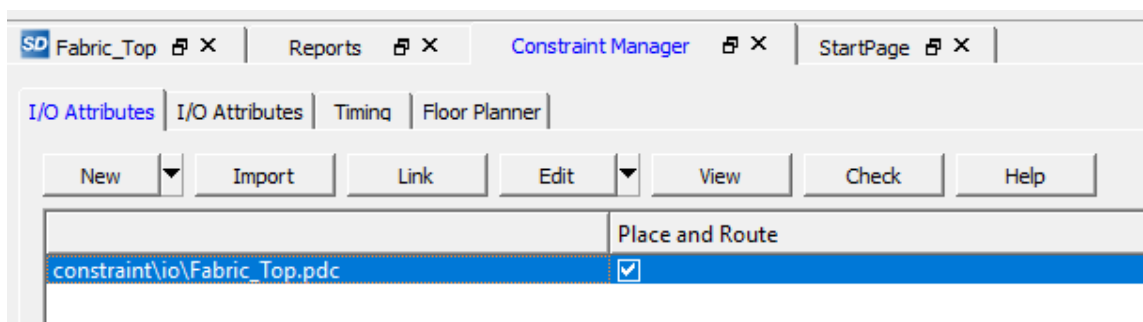


Figure 44 - Selecting the I/O PDC constraint file in the Design Flow window

Step 4 - Generating Timing Constraints

In this step you derive timing constraints for the design. Timing constraints for known blocks (such as the RC oscillators and the PLLs) can be generated automatically.

1. Select the Timing tab in the Constraint manager window. Click on **Derive Constraints**.

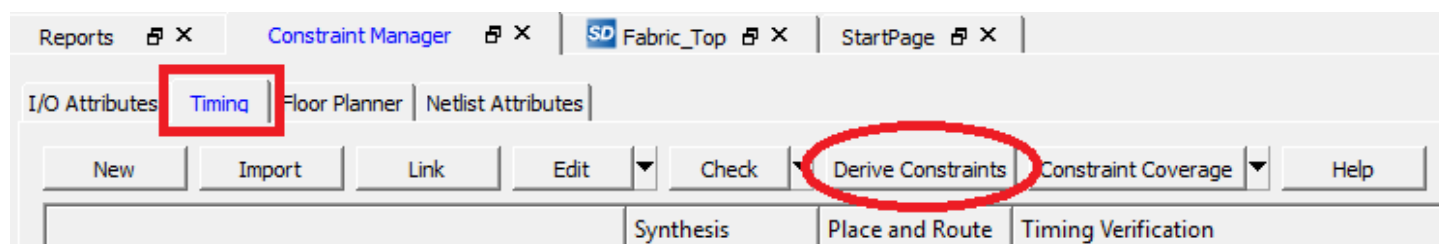


Figure 45 - Deriving Timing Constraints

2. Click on **Yes** in the Message window to automatically associate the derived constraints SDC file to the 'Synthesis', 'Place and Route' and 'Timing Verification'.

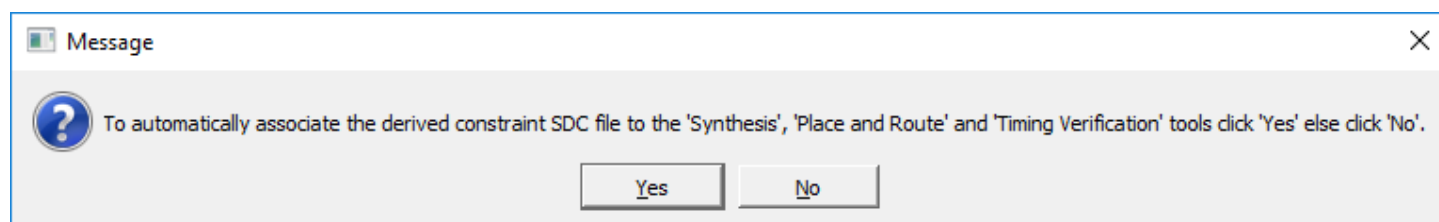


Figure 46 - Message Window

3. Double click on the Top_derived_constraints.sdc to view the automatically generated clock constraints in the text editor.

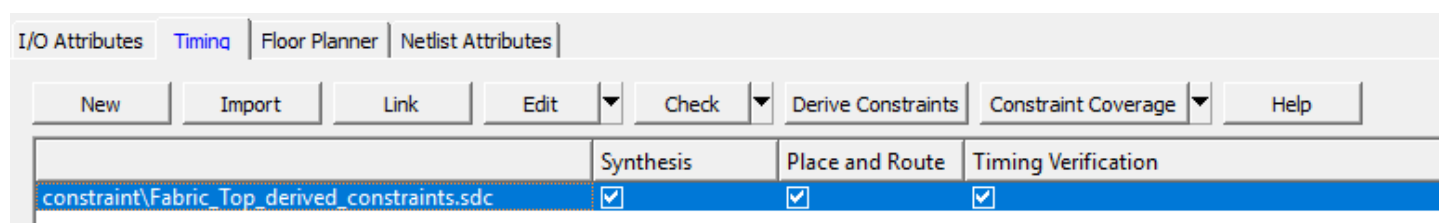


Figure 47 - Derived Timing Constraints

4. For this design a Clock constraint and Generated Clock constraint are generated automatically. The Clock constraint is the constraint for the 50 MHz RC oscillator, which is the reference clock for the fabric CCC. The Generated Clock constraint is the constraint for the 1 MHz Fabric PLL output.

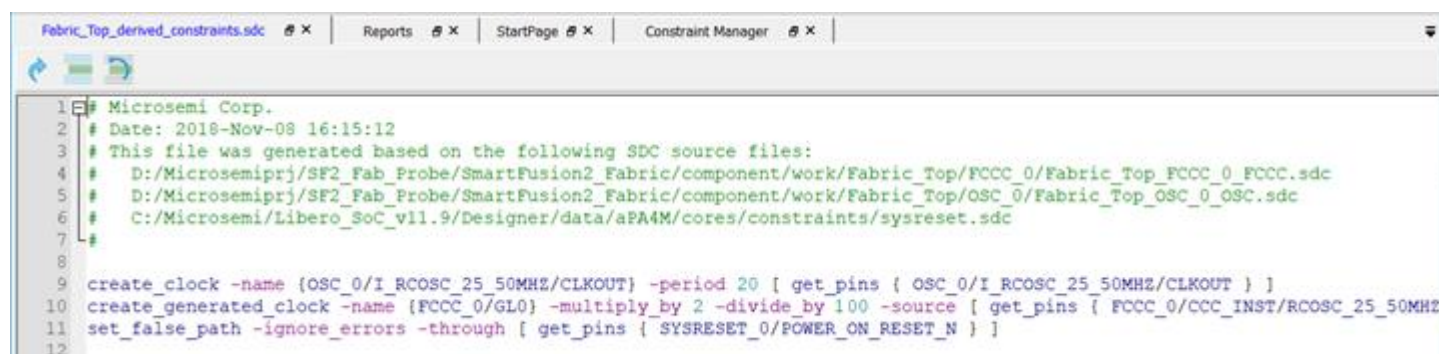


Figure 48 - Derived timing constraints

5. Close the Constraint Manager Editor (**File > Close Fabric_Top_derived_constraints.sdc**).

Step 5 – Synthesis and Layout

In this step you will use the push-button flow to synthesize the design with Synplify Pro, run layout and generate the programming file.

1. Double-click the Generate Bitstream in the Design Flow window to synthesize the design, run layout using the I/O constraints that were created and generate the programming file.

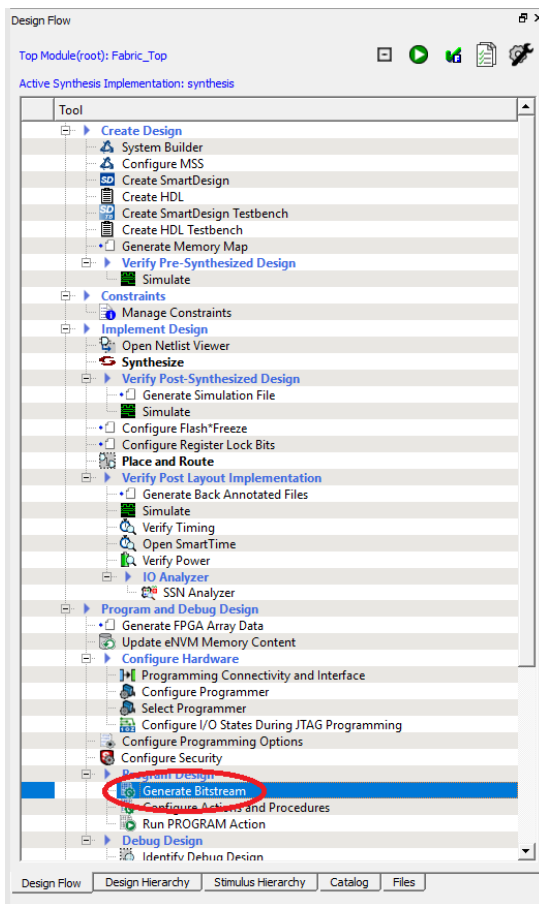


Figure 49 - Generate Bitstream

The design implementation tools will run in batch mode. Successful completion of a design step will be indicated by a green check mark next to the item in the Design Flow window.

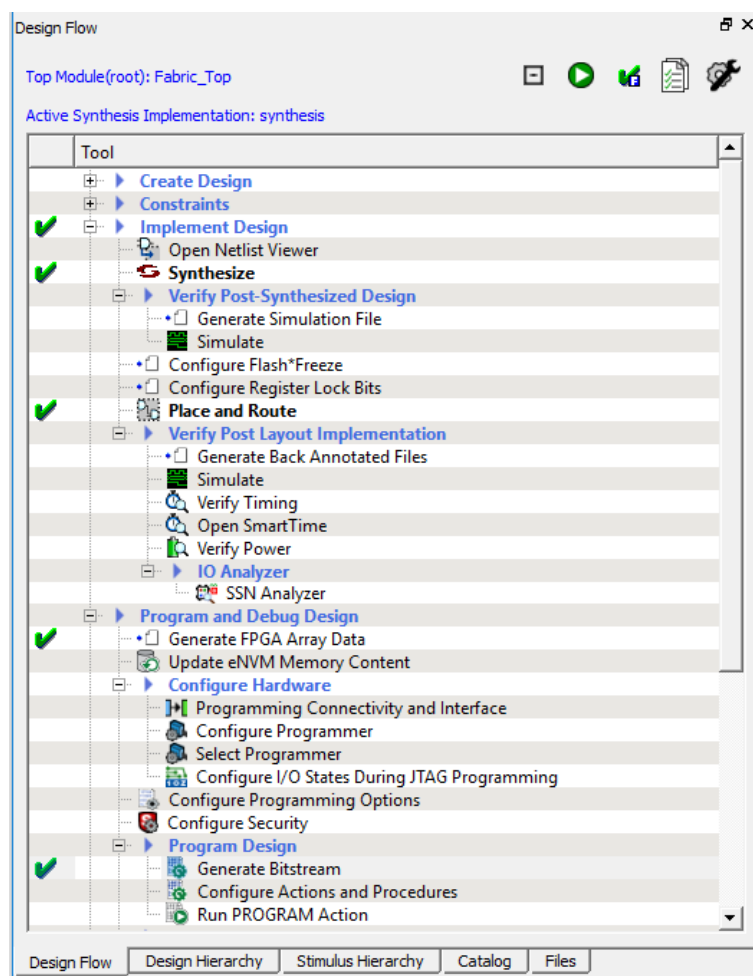


Figure 50 – Successful completion of design implementation

- The Reports tab will display reports for the tools used to implement the design.

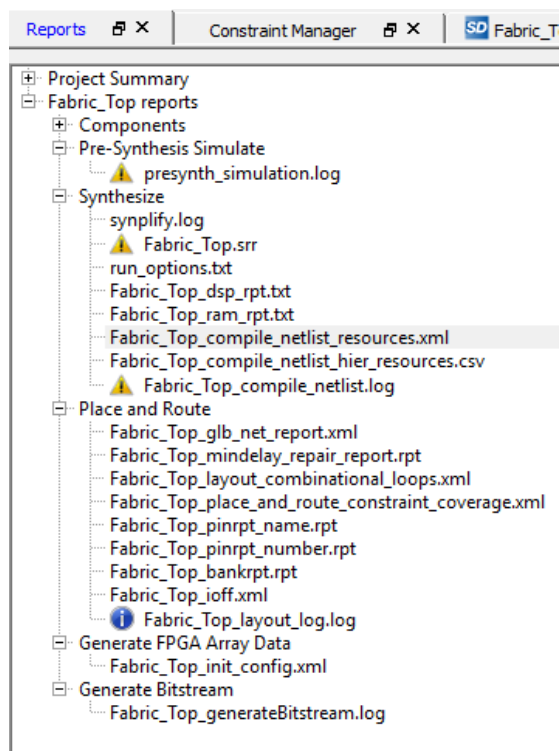


Figure 51 - Reports tab after implementing the design

3. Select the Compile report (Fabric_Top_compile_netlist_resources.xml) under Synthesize on the Reports tab to view the resource usage. Record the number of combinatorial cells (LUTs) and sequential cells (Flip-flops) used in the design below.

LUTs (4LUT) _____

Flip-flops (DFF) _____

4. Configure the timing reports to be generated by right-clicking Verify Timing under Verify Post Layout Implementation on the Design Flow tab and selecting **Configure Options**. Enable all available timing report options and select **OK**.

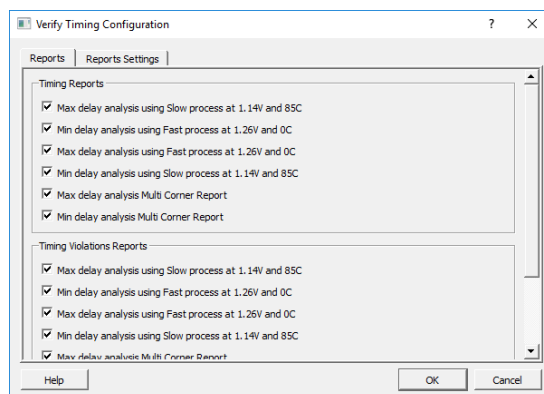


Figure 52 - Timing reports configuration

5. Generate the timing reports by right-clicking Verify Timing under Verify Post Layout Implementation on the Design Flow tab and selecting **Run**.

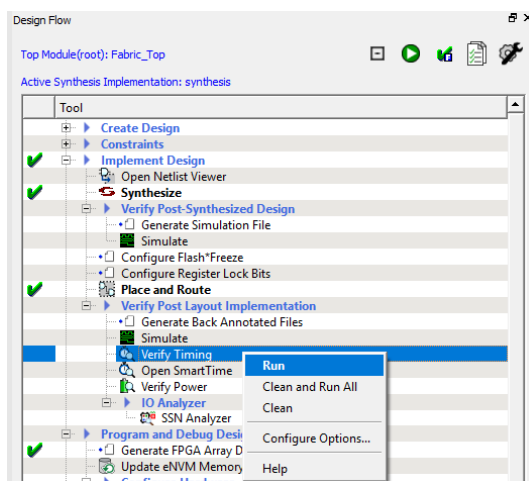


Figure 53 - Generating the post-layout timing report

6. Select the timing report named Fabric_Top_max_timing_slow_1.14V_85C.xml under Verify Timing on the Reports tab. Scroll to the timing Summary in the Report window. Record the maximum frequency for the clock FCCC_0/ GL0 below:

FCCC_0/ GL0: _____

7. Timing violations can quickly be identified by looking at the timing violations reports (Fabric_Top_max/min_timing_violations_slow_1.14V_85C.xml and Fabric_Top_max/min_timing_violations_fast_1.26V_0C.xml).
8. The output files from Synplify Pro and Designer (Place&Route, Timing Analyzer) will be visible on the Libero SoC Files tab.

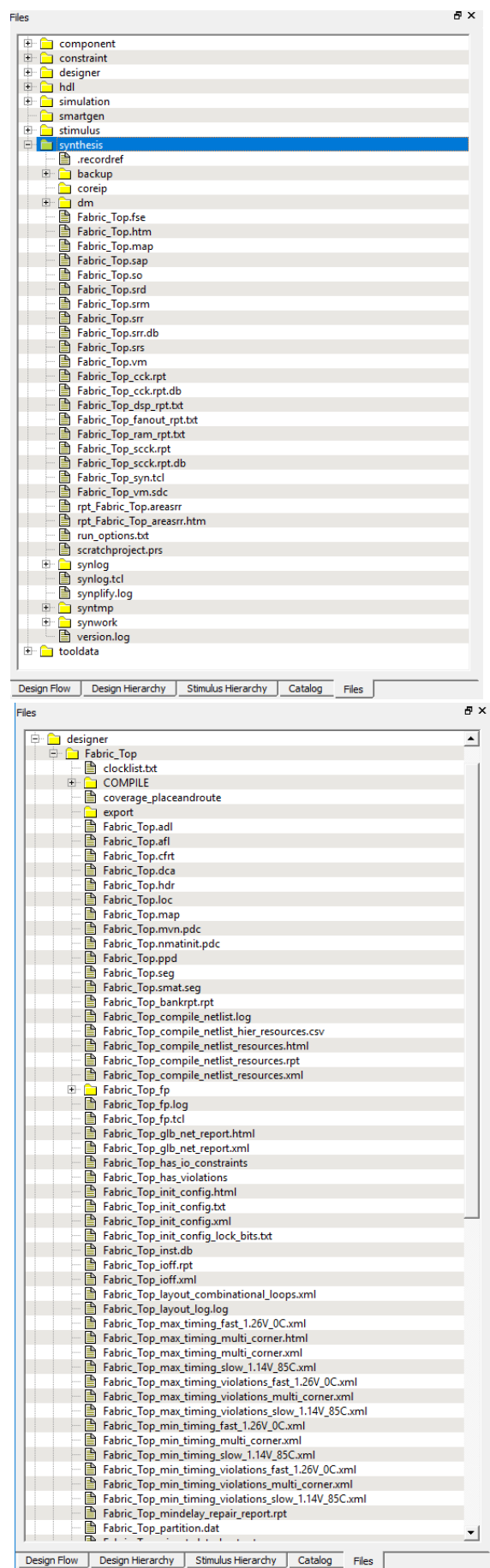


Figure 54 - Synthesis and Designer files on Libero SoC Files tab

Step 6 – Programming

In this step, you will run FlashPro in batch mode to program the SmartFusion2 device on the SmartFusion2 target board. Connect the SMF2000 board using a Micro-USB cable to the PC USB port (board power is provided from PC USB port). Since the FlashPro5 driver should be already installed by the Microsemi Libero SoC v12.5 installation, the Window's Device Manager should display the following entries (port number may differ depending on your PC):

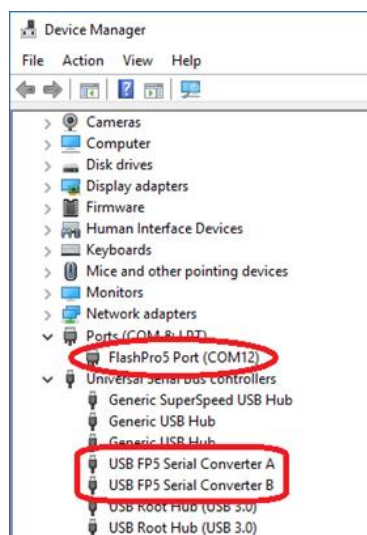


Figure 55 – Device Manager

1. Expand Program Design in the Design Flow window. Right-click **Run PROGRAM Action** and select **Run** to begin programming.

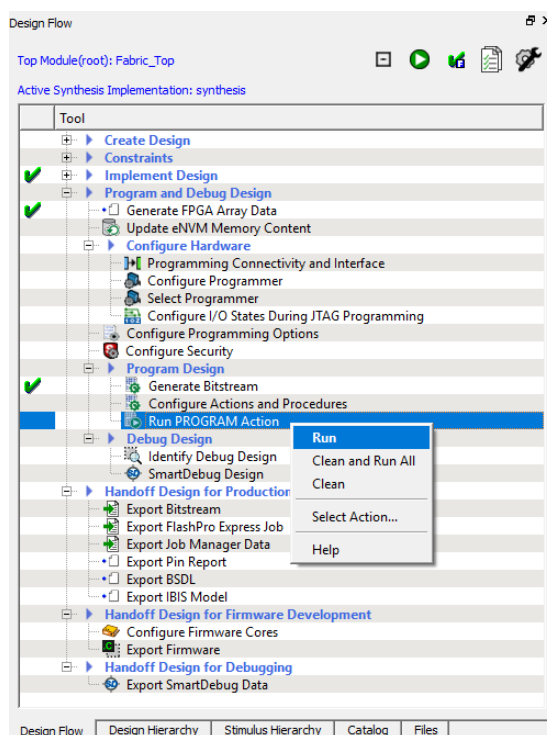


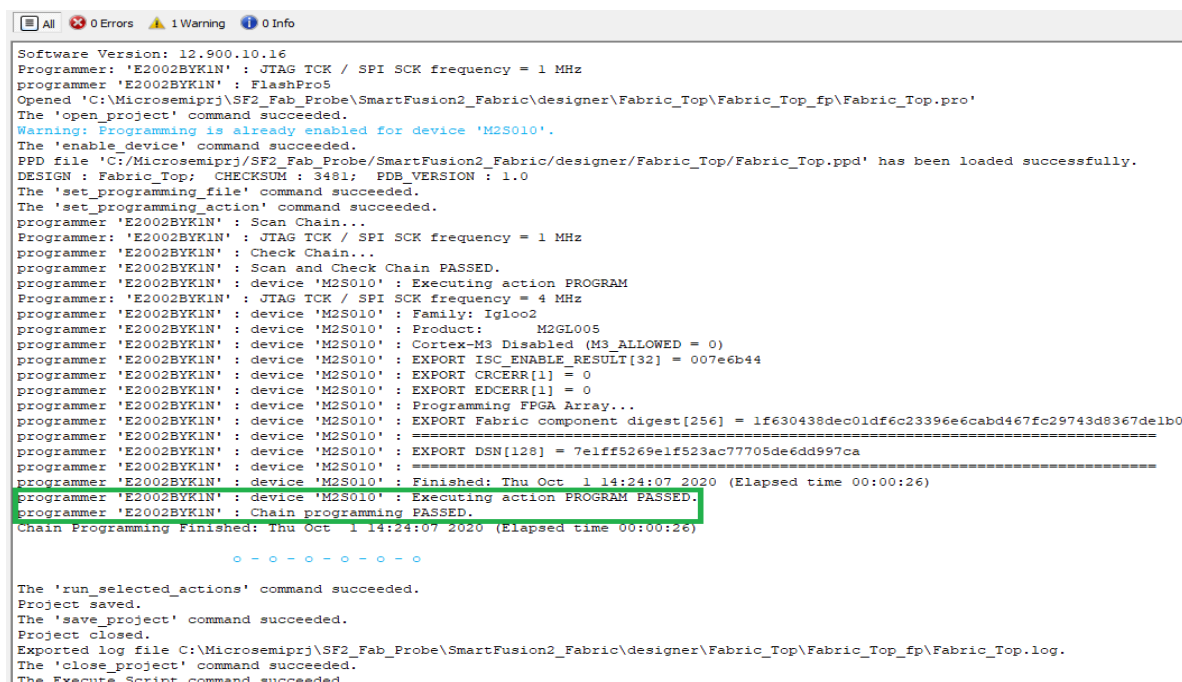
Figure 56 - Launching Programming software from Design Flow window

2. FlashPro will run in batch mode and program the device. Programming messages will be visible in the Libero SoC log window (programmer number and device number may differ).

Note: Do not interrupt the programming sequence; it may damage the device or the programmer.

- The following message should be visible in the Reports view under Program Device when the device is programmed successfully (programmer number will differ; device will differ depending on the target board):

programmer 'E2002BYK1N' : device 'M2S010' : Executing action PROGRAM PASSED.



```

Software Version: 12.900.10.16
Programmer: 'E2002BYK1N' : JTAG TCK / SPI SCK frequency = 1 MHz
programmer 'E2002BYK1N' : FlashPro5
Opened 'C:\Microsemiprj\SF2_Fab_Probe\SmartFusion2_Fabric\designer\Fabric_Top\Fabric_Top_fp\Fabric_Top.pro'
The 'open_project' command succeeded.
Warning: Programming is already enabled for device 'M2S010'.
The 'enable_device' command succeeded.
FPD file 'C:\Microsemiprj\SF2_Fab_Probe\SmartFusion2_Fabric\designer\Fabric_Top\Fabric_Top.ppd' has been loaded successfully.
DESIGN : Fabric_Top; CHECKSUM : 3481; PDB_VERSION : 1.0
The 'set_programming_file' command succeeded.
The 'set_programming_action' command succeeded.
programmer 'E2002BYK1N' : Scan Chain...
programmer 'E2002BYK1N' : JTAG TCK / SPI SCK frequency = 1 MHz
programmer 'E2002BYK1N' : Check Chain...
programmer 'E2002BYK1N' : Scan and Check Chain PASSED.
programmer 'E2002BYK1N' : device 'M2S010' : Executing action PROGRAM
programmer 'E2002BYK1N' : JTAG TCK / SPI SCK frequency = 4 MHz
programmer 'E2002BYK1N' : device 'M2S010' : Family: Igloo2
programmer 'E2002BYK1N' : device 'M2S010' : Product: M2GL005
programmer 'E2002BYK1N' : device 'M2S010' : Cortex-M3 Disabled (M3_ALLOWED = 0)
programmer 'E2002BYK1N' : device 'M2S010' : EXPORT ISC_ENABLE_RESULT[32] = 007e6b44
programmer 'E2002BYK1N' : device 'M2S010' : EXPORT CRCERR[1] = 0
programmer 'E2002BYK1N' : device 'M2S010' : EXPORT EDCERR[1] = 0
programmer 'E2002BYK1N' : device 'M2S010' : Programming FPGA Array...
programmer 'E2002BYK1N' : device 'M2S010' : EXPORT Fabric component digest[256] = 1f630438dec01df6c23396e6cabd467fc29743d8367delb0
programmer 'E2002BYK1N' : device 'M2S010' : EXPORT DSN[128] = 7e1ff5269e1f523ac77705de6dd997ca
programmer 'E2002BYK1N' : device 'M2S010' : Finished: Thu Oct 1 14:24:07 2020 (Elapsed time 00:00:26)
programmer 'E2002BYK1N' : device 'M2S010' : Executing action PROGRAM PASSED.
programmer 'E2002BYK1N' : Chain programming PASSED.
Chain Programming Finished: Thu Oct 1 14:24:07 2020 (Elapsed time 00:00:26)

The 'run_selected_actions' command succeeded.
Project saved.
The 'save_project' command succeeded.
Project closed.
Exported log file C:\Microsemiprj\SF2_Fab_Probe\SmartFusion2_Fabric\designer\Fabric_Top\Fabric_Top_fp\Fabric_Top.log.
The 'close_project' command succeeded.
The Execute Script command succeeded.
  
```

Figure 57 - Programming messages in Libero SoC log window

- A green check mark will appear next to Program Design and Run PROGRAM Action in the Design Flow window to indicate programming completed successfully.

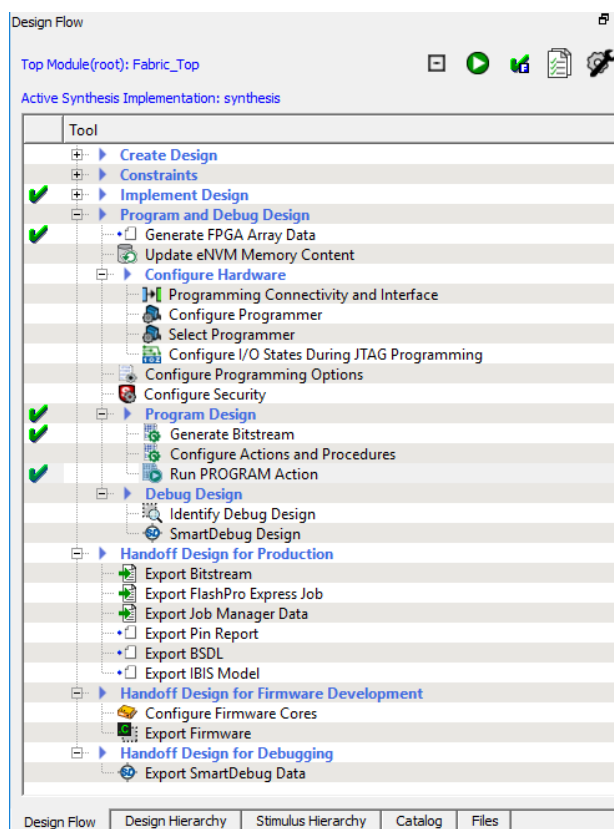


Figure 58 - Design Flow window after programming

Running the Application

5. Reset the board by pressing and releasing the Reset button (the button on the opposite side of the LEDs) and observe the pattern of the LEDs.
6. Press the USER_BTN (the button near the 8 LEDs) and observe the counter changing the count direction.

Step 7 - Debugging with SmartDebug

In this step you will use SmartDebug to view internal circuits in the design.

SmartDebug tool is a new approach to debug the Microsemi FPGA array and SERDES without using an internal logic analyzer (ILA). SmartDebug provides the following methods to debug the SmartFusion2 / IGLOO2 FPGA array:

- **Live Probe:** Two dedicated probes can be configured to observe a Probe Point which is any input or output of a logic element. The probe data can then be sent to an oscilloscope or logic analyzer.
- **Active Probe:** Active Probe allows dynamic asynchronous read and write to a flip-flop or probe point. This enables a user to quickly observe the output of the logic internally or to quickly experiment on how the logic will be affected by writing to a probe point.
- **Memory Debug:** Memory debug is used to debug embedded FPGA fabric memories (uSRAM and LSRAM) by reading and writing to the memory block. Memory Debug also supports reading eNVM memory content.
- **Probe Insertion:** Probe insertion is used to insert probes into the design and bring signals out to the FPGA package pins to evaluate and debug the design.
- **SERDES Debug:** SmartDebug is used to debug high speed serial interfaces by utilizing on-chip PRBS features and signal integrity controls.
- **FPGA Hardware Breakpoint:** Enables single stepping and synchronous signal capture capabilities on selected clock domains. Available for logic running on an internally generated PLL clock (clock gating logic will be added). See Project settings "Enable FPGA Hardware Breakpoint Auto Instantiation".

SmartDebug Live Probes

In this step we will use Live Probe A to drive one of the LEDs on the target board. The LIVE_PROBE_FB macro re-directs the Live Probe output. For this design we connected the PROBE_A output of the LIVE_PROBE_FB macro to an output pin which is connected to the USER_LED on the target board.

1. Expand Debug Design in the Libero Design Flow window. Select SmartDebug Design then right-click and select **Open Interactively**.

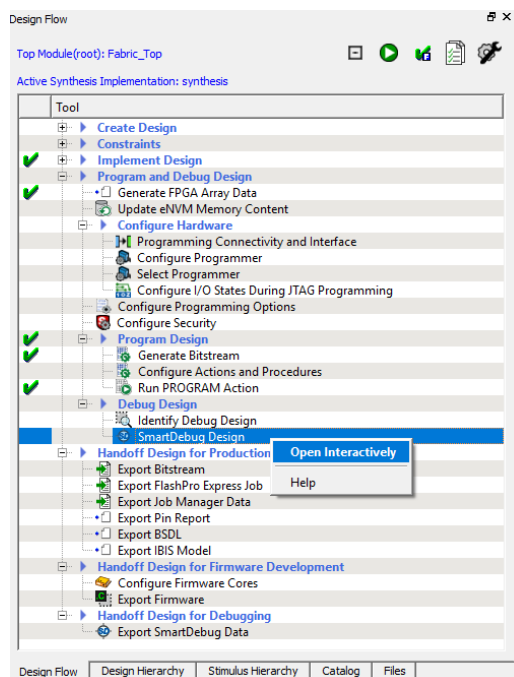


Figure 59 - Launching SmartDebug

2. The SmartDebug GUI will open. Click the Debug FPGA Array button (circled in the figure below). The Debug SERDES option may not be available for some target boards.

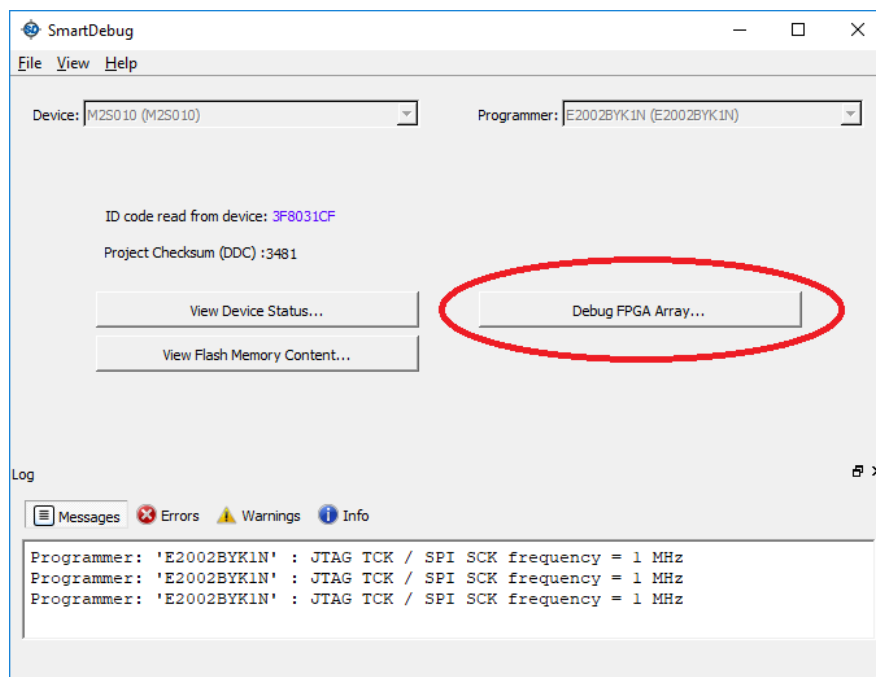


Figure 60 - SmartDebug GUI

- Select the Live Probes tab in the **Debug FPGA Array** GUI. The design hierarchy is displayed in the left window. Signals selected for probing are displayed in the right window.

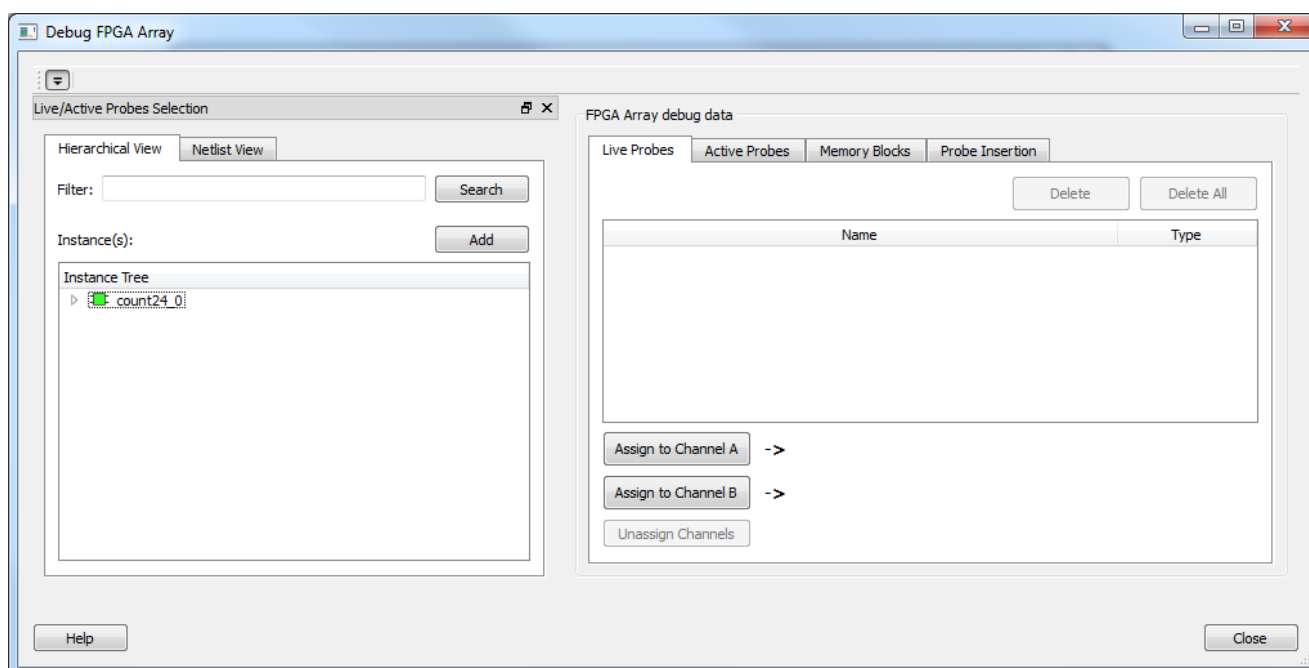


Figure 61 - SmartDebug Live Probes tab

- Select count24_0 in the Instance Tree on the Hierarchical View then click **Add**. The signals will appear in the Live Probes tab.

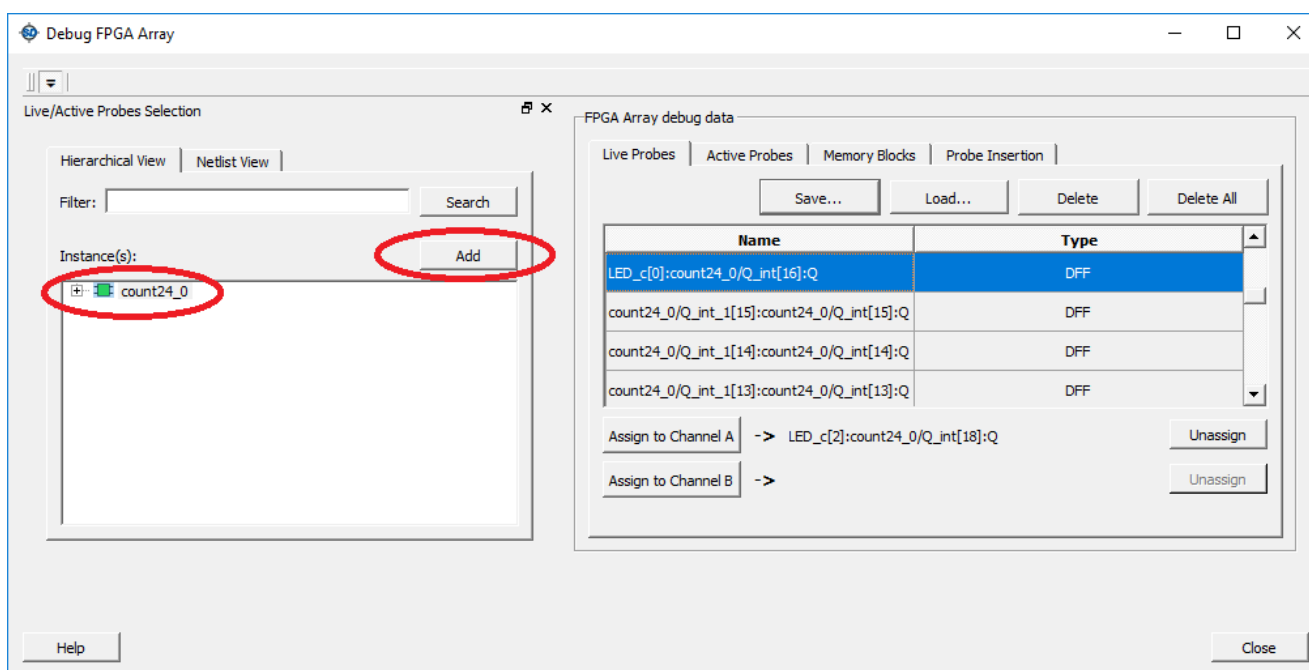


Figure 62 - count24_0 signals in the Live Probes tab

- Select the signal LED_c[0]:count24_0/Q_int[16]:Q then click **Assign to Channel A**.
- The USER_LED starts blinking.
Note: the USER_LED is located directly next to the USB connector.
- Select a different signal and assign it to Channel A. Observe the change in the LED patterns. Live Probe allows observation of any flip-flop output without having to re-run layout on the design.

SmartDebug Active Probes

In this step, we will use the Active Probes to observe the output of count_24_0.

10. Select the Active Probes tab in the Debug FPGA Array GUI. The design hierarchy is displayed in the left window. Signals selected for probing are displayed in the right window.

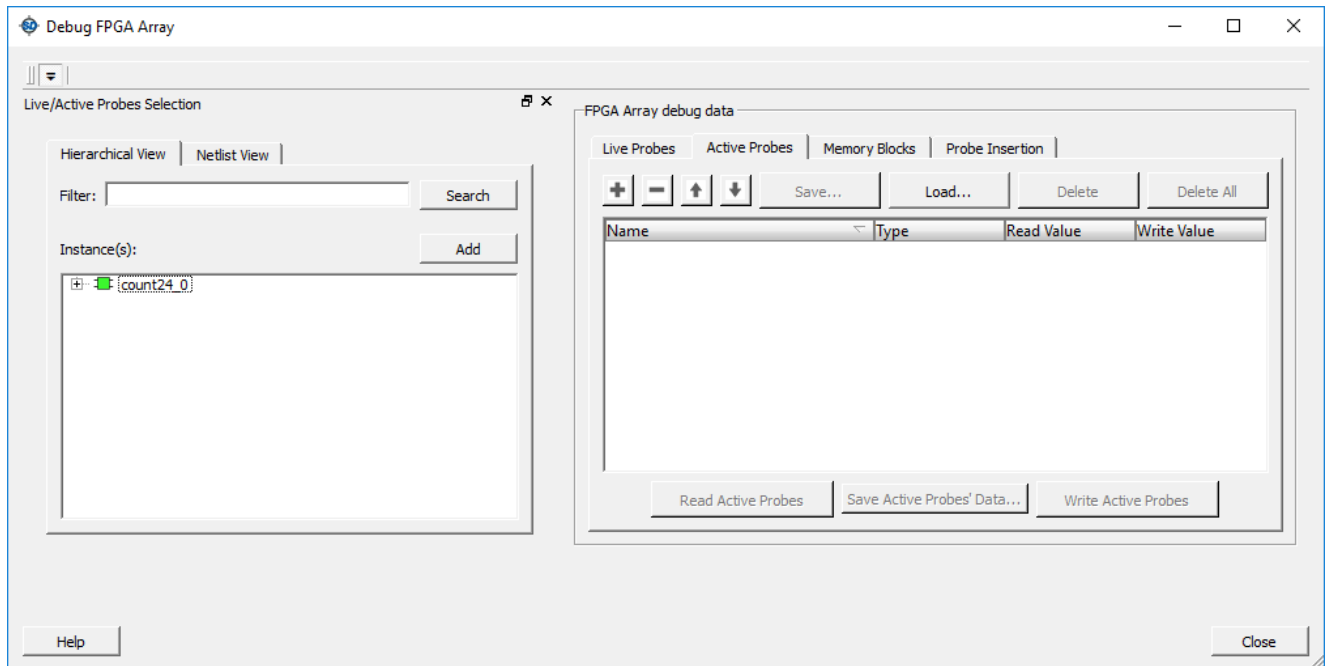


Figure 63 - SmartDebug Active Probes tab

11. Select count24_0 in the Instance Tree on the Hierarchical View then click **Add**. The signals will appear in the Active Probes tab.

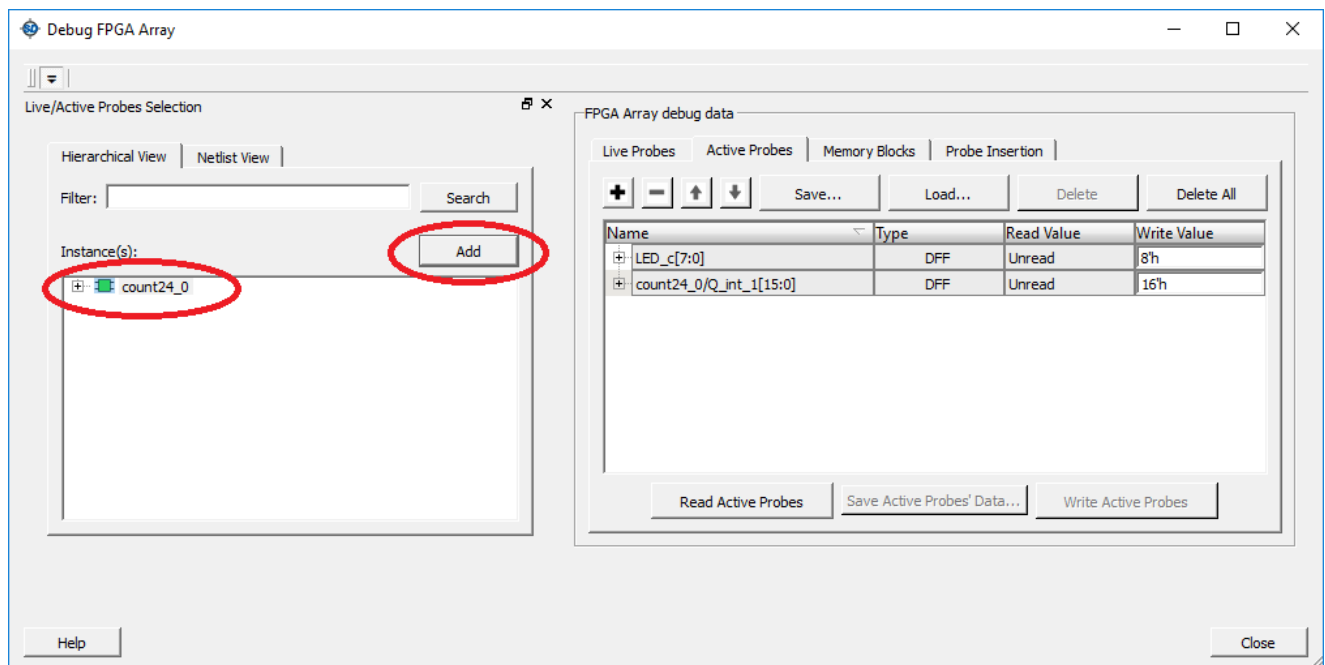


Figure 64 - Adding the count24_0 signals to the Active Probes tab

12. Click the **Read Active Probes** button (circled in the figure below). The state of the signal will be displayed. Clicking **Read Active Probes** repeatedly will allow observation of the counter output.

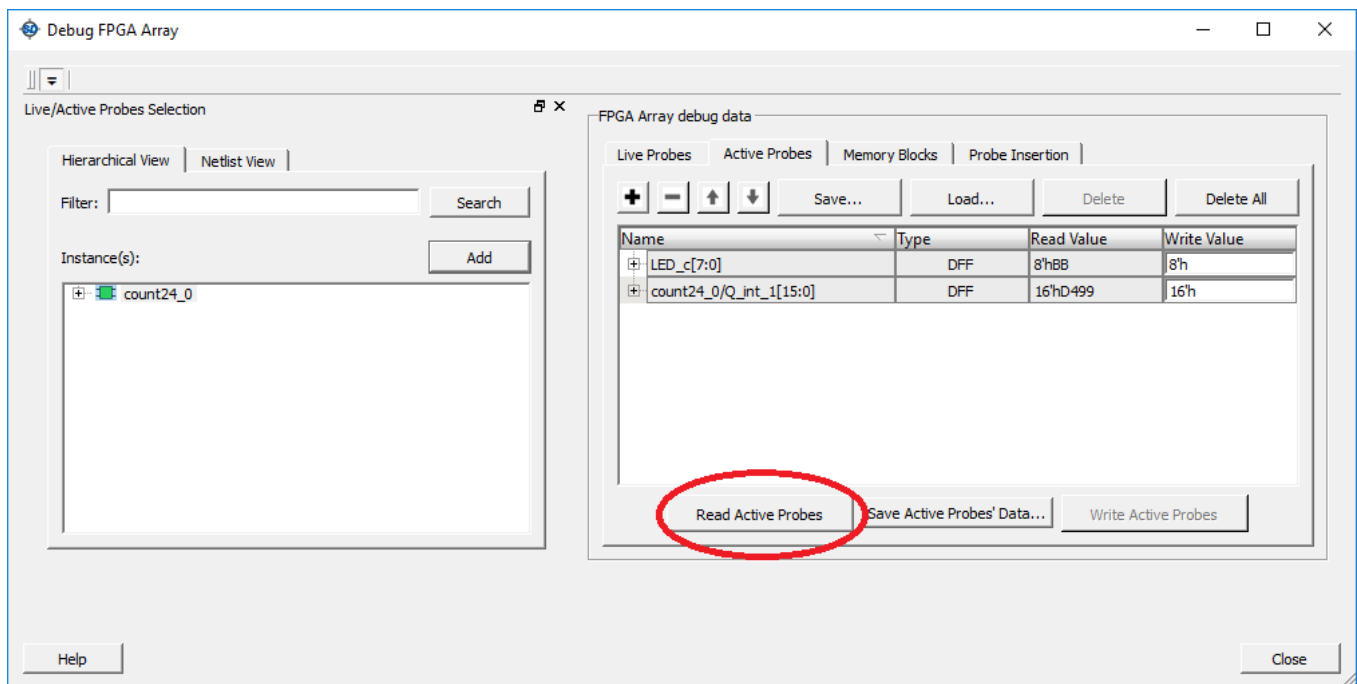


Figure 65 - Reading the counter with the Active Probes

13. Click **Close** to close the Debug FPGA dialog box. Click **Cancel** when prompted about saving the active probes.
14. Close SmartDebug (**File > Exit**) and Libero SoC (**Project > Exit**).

End of SmartFusion2 Fabric lab

Record answers to questions below

Name: _____

Page 31

Record the number of combinatorial and sequential cells used in the design below.

LUTs (4LUT) _____

Flip-flops (DFF) _____

Record the maximum frequency for the clock FCCC_0/ GL0 below:

FCCC_0/ GL0: _____

Answers to Questions (results can slightly vary on your system)

Page 31

Record the number of sequential and combinatorial cells used in the design below.

4LUT 25

DFF 24

Record the maximum frequency for the clock FCCC_0/GL0 below:

FCCC_0/ GL0: 430.108 MHz

List of Figures

Figure 1 - Libero SoC Project Manager	5
Figure 2 - Libero SoC New Project dialog box	6
Figure 3 - Project Details	6
Figure 4 - Device selection (settings for SMF2000 board shown)	7
Figure 5 - Device Settings (settings for SMF2000 board shown)	8
Figure 6 - Design Template settings (all kits)	8
Figure 7 - Imported HDL source files	9
Figure 8 - Imported I/O Constraint file	10
Figure 9 - HDL source file and I/O constraint file in Libero SoC	10
Figure 10 - Opening the SmartDesign canvas	11
Figure 11 - Entering SmartDesign name	11
Figure 12 - Component pasting	11
Figure 13 - Clock & Management category of the Libero SoC IO Catalog	12
Figure 14 - Component naming	12
Figure 15 - Component naming	13
Figure 16 - Configuring the Chip Oscillators (SmartFusion2 target)	14
Figure 17 - Macro Library category of the Libero SoC IP Catalog	14
Figure 18 - IP Catalog Search field	15
Figure 19 - SmartDesign canvas after adding components	15
Figure 20 - Connection creating	16
Figure 21 - Adding a slice to the Q[23:0] output	17
Figure 22 - Edit Slices dialog box	17
Figure 23 - Adding a slice to the count24_0 output	17
Figure 24 - Marking the PROBE_B port unused	18
Figure 25 - Adding a port in Design Canvas	18
Figure 26 - Adding LED[1:8] output port in Design Canvas	18
Figure 27 - SmartDesign canvas after making connections	19
Figure 28 - Setting Fabric_Top as the Top Level (Root)	19
Figure 29 - Design Hierarchy with Fabric_Top as the Top Level	20
Figure 30 - Importing Simulation files	20
Figure 31 - Importing the ModelSim simulation files	21
Figure 32 - Importing the testbench file	21
Figure 33 - Active Testbench in the project	22
Figure 34 - Testbench and simulation files	22
Figure 35 - Location of Libero SoC project	22
Figure 36 - Libero SoC Project Settings dialog box	23
Figure 37 - Simulation options	23
Figure 38 - Tool Profiles	24
Figure 39 - ModelSim ME Profile	24
Figure 40 - Launching pre-synthesis simulation	25
Figure 41 - ModelSim Wave window	25
Figure 42 - Libero SoC Constraint Manager	26
Figure 43 - PDC file in the Libero SoC constraint editor	26
Figure 44 - Selecting the I/O PDC constraint file in the Design Flow window	27
Figure 45 - Deriving Timing Constraints	28
Figure 46 - Message Window	28

Figure 47 - Derived Timing Constraints.....	28
Figure 48 - Derived timing constraints.....	28
Figure 49 - Generate Bitstream.....	29
Figure 50 – Successful completion of design implementation	30
Figure 51 - Reports tab after implementing the design.....	30
Figure 52 - Timing reports configuration	31
Figure 53 - Generating the post-layout timing report	31
Figure 54 - Synthesis and Designer files on Libero SoC Files tab	32
Figure 55 – Device Manager	33
Figure 56 - Launching Programming software from Design Flow window	33
Figure 57 - Programming messages in Libero SoC log window	34
Figure 58 - Design Flow window after programming	34
Figure 59 - Launching SmartDebug	36
Figure 60 - SmartDebug GUI	36
Figure 61 - SmartDebug Live Probes tab.....	37
Figure 62 - count24_0 signals in the Live Probes tab.....	37
Figure 63 - SmartDebug Active Probes tab	38
Figure 64 - Adding the count24_0 signals to the Active Probes tab	38
Figure 65 - Reading the counter with the Active Probes	39