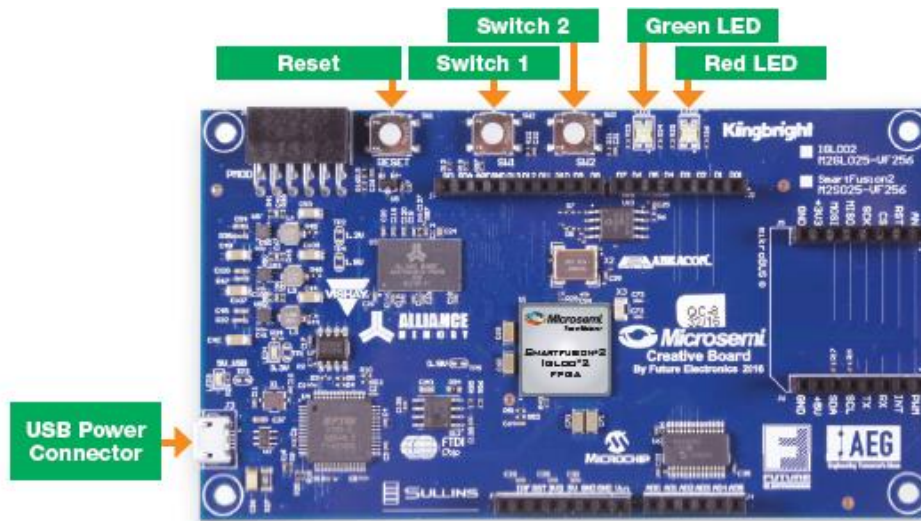


Future Electronics – Microsemi
IGLOO2 Creative Development Board

LAB 1

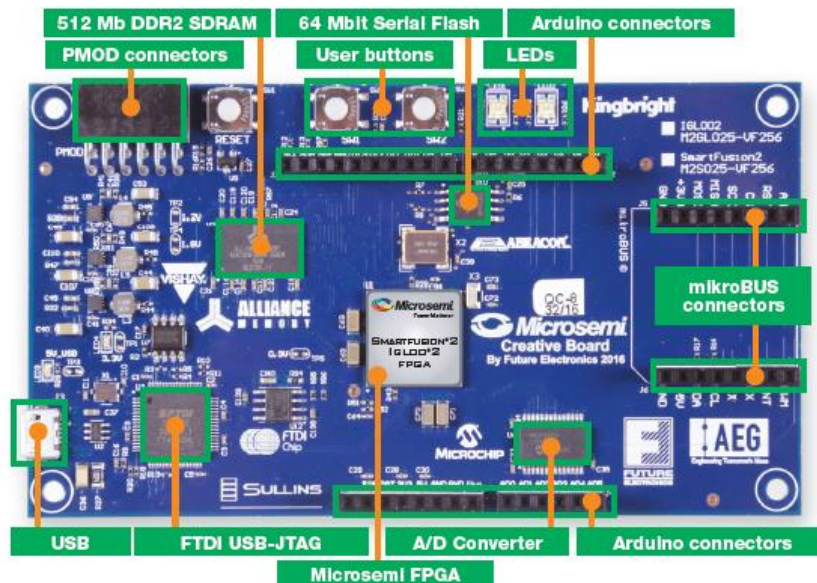
Release - Version A2

The IGLOO2 Board



SW1, SW2, Green LED and Red LED located on top board edge

USB connector
(for power and programming located on left board edge)



Hello and welcome to the Future Electronics Creative Board Lab 1 Instructions.

In order to complete this series of Labs you will first have had to download the Microsemi SoC Libero Development tools and installed them onto your laptop (or desktop).

You will also have to obtain a free license from Microsemi and install it on your machine.

This lab was originally written to work with Version 11.7 SP1 of the Libero tool suite.

Refer to the Revision table at the end of this document for updates and improvements to this Lab set.

INSTALL THE LAB

If you have not already obtained a copy of the Labs via other sources, please copy the **Future** folder from the USB stick onto the Root of your C: (or D:) drive.

When installed, you will have the following folder paths on your computer:

C:\Future\FPGA\Microsemi\CreativeBoard\IGLOO2\Source

C:\Future\FPGA\Microsemi\CreativeBoard\IGLOO2\Lab2_VHDL

C:\Future\FPGA\Microsemi\CreativeBoard\IGLOO2\Lab2_ver

C:\Future\FPGA\Microsemi\CreativeBoard\IGLOO2\Lab3

LAB 1 GOALS

This is what you will learn in Lab1:

- Create a project in Libero SoC
 - Import source files, IO, and Timing constraints
- Run Synthesis
- Verify functionality with the Post-Synthesis Simulation
- Compile the Design
- Run Place and Route
- Generate a Bitstream and Program the Device

Here is a list of the expected results for Lab 1:

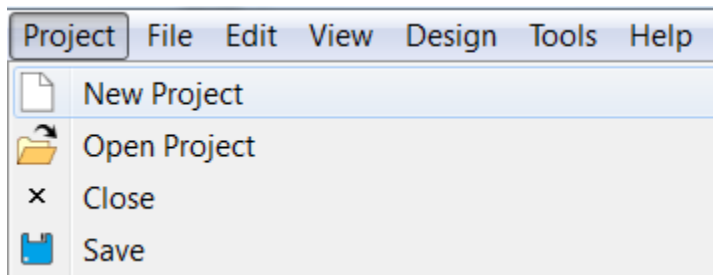
- At power up --> LED 1(green) blinks every 1 sec and LED 2(red) blinks every 2 sec
- Press and hold SW1 --> LED 1(green) blinks every 0.5 sec and LED2(red) blinks every 2 sec
- Press and hold SW2 --> LED 2(red) blinks every 1 sec and LED1(green) blinks every 1 sec
- Press and hold SW1 and SW2 --> LED 1(green) and LED 2(red) turn off

CREATING A PROJECT

Start the Libero SoC tool set by double clicking the ICON on your desktop or pathing to the installed folder and double clicking on the file called **Libero SoC v11.7**.

If this is the first time you have started the tools on your machine, it could take up to 45 seconds for the tools to start. If the tools ask you to check for updates, select NO.

Like many Windows tools, there are often several ways to accomplish the same results; in this lab we will be showing you how to use the Menu system to select and run things.



Select **Project... New Project**

Project Details

New project

Project details
Specify project details

Project Details

Project name:

Project location:

Description:

Preferred HDL type:

☐ Enable block creation

Libero
System-on-Chip

Enter the *Project name*: as **Lab1**

Set the *Project location*: to **C:/Future/FPGA/Microsemi/CreativeBoard/IGLOO2**

(This is the bottom of the folder path that you copied to your hard drive, where you will create Lab1 and also contains the Source folder for Lab1. There will also be solutions folders for Lab 2 in case you wish to start from there without completing Lab1. *Make sure that you enter the project location to where you actually placed the image if not where we recommend.*)

Enter a *Description*: if you desire to properly document the project in the provided box.

Note: This is where you decide if you are going to do the Lab in VHDL or Verilog

Select the *Preferred HDL type*: as either **VHDL** or **Verilog** using the pull down.

Confirm that the check box is **NOT** checked (Enable block creation).

Press the **NEXT** button.

Device Selection

New project

Device selection
Select a part for your project from the part number list

Selected part: M2GL025-VF256

Part filter

Family: IGLOO2 Die: M2GL025 Package: 256 VF
Speed: STD Core voltage: All Range: COM

Reset filters

Search part:

Part Number	4LUT	DFF	User I/Os	uSRAM 1K	LSRAM 18K	Math (18x18)
M2GL025-VF256	27696	27696	138	34	31	34

Help < Back Next > Finish Cancel

We will be using the following device for this lab: M2GL025-VF256

There are a few ways to set this; for this lab, we will use the **Part Filter** method.

Set Family: IGLOO2 Die: M2GL025 Package: 256 VF

Speed: STD Core voltage: All Range: COM

This will reduce your choice to one part called M2GL025-VF256.

Press the **NEXT** button.

Device Settings

New project

Device settings
Choose device settings for your project

Selected part: M26L025-VF256

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

Power on Reset delay : 100ms

☐ System controller suspended mode

Help < Back Next > Finish Cancel

Libero
System-on-Chip

The Libero tool set will default to **LVCMOS 2.5V** – we will be changing that for this Lab.

Set *I/O Settings* -> *Default I/O technology*: to **LVCMOS 3.3V**

Confirm the *Reserve pins for probes* is **NOT** checked.

Leave (or change) the *Power supplies*

PLL supply voltage (V): as **3.3**

Power on Reset delay: as **100ms**

Confirm the *System controller suspended mode* is **NOT** checked.

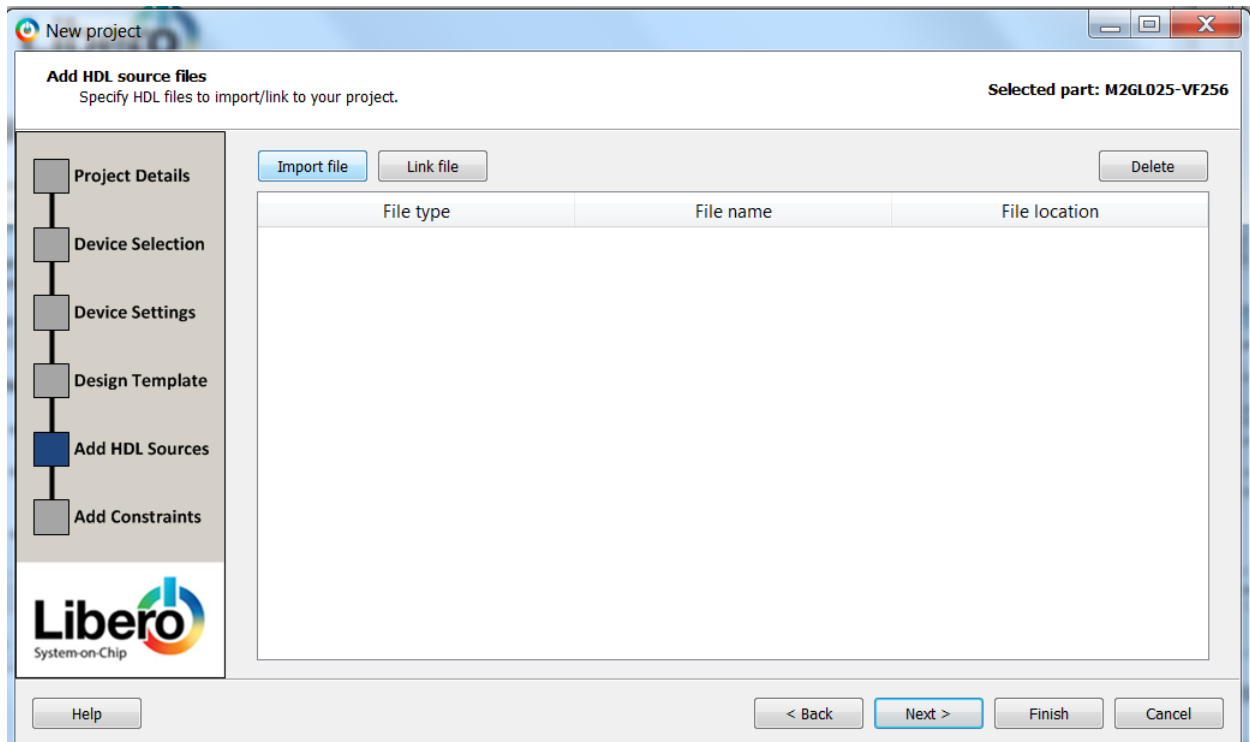
Press the **NEXT** button.

Device Template

We will accept the default setting (**None**) for this lab.

Press the **NEXT** button.

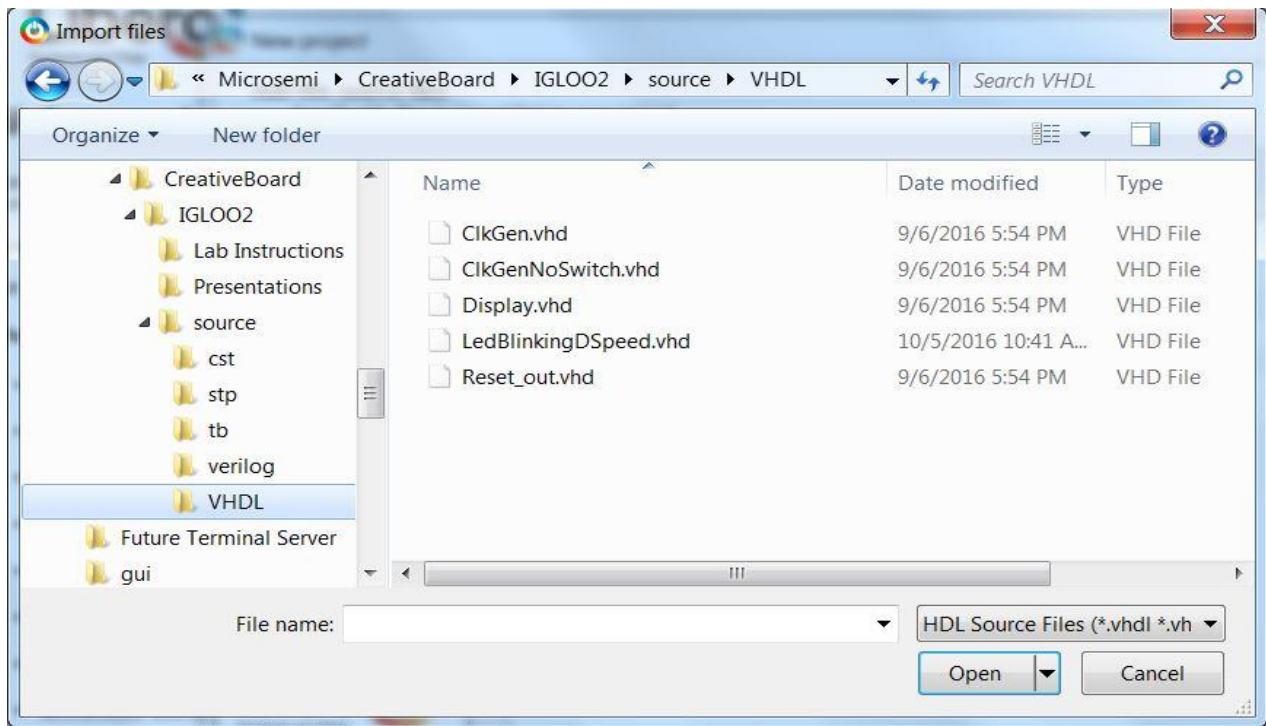
Add HDL Sources



We will now add some HDL source files to the design (after all, who wants to type a bunch of stuff?).

Select the **Import file** button at the top of the screen.

An overlay window will appear.



Going forward, you either use VHDL or Verilog based on the select you made a few screens back.

I use the following notation for describing the file extension: filename.v(hd)

You either select the .v for Verilog or .vhd for VHDL

Select either VHDL or Verilog by pathing to the

Future\FPGA\Microsemi\CreativeBoard\IGLOO2\Source then into either the **VHDL** or **verilog** folder.

Double click the top file: **LedBlinkingDSpeed.v(hd)**

(We select this first so that the tool will set this as the top file for the design.)

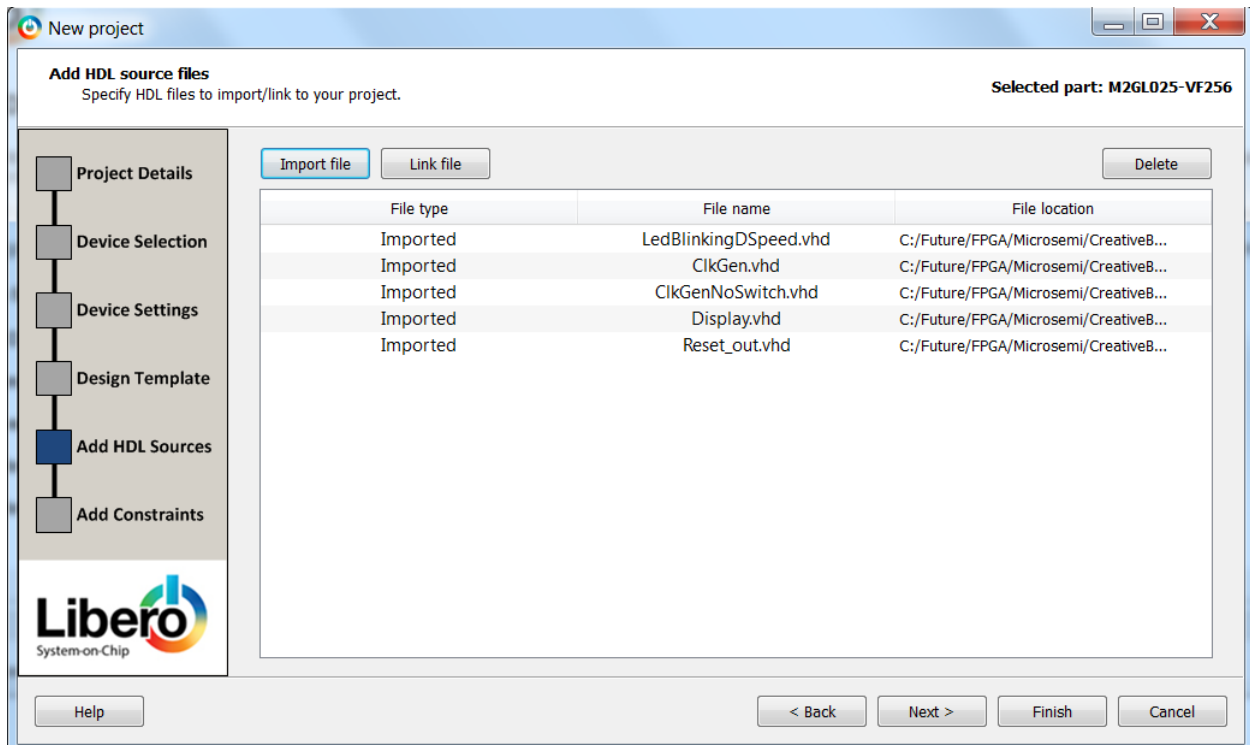
Select the **Import file** button again and select the remaining 4 files to add to the project.

You can use the windows *ctrl-click* function to select multiple files or add them one at a time.

add ClkGen.v(hd) ClkGenNoSwitch.v(hd) Display.v(hd) Reset_out.v(hd)

Press the **OPEN** button.

Note: The screen images show VHDL files, will look very similar with Verilog files.

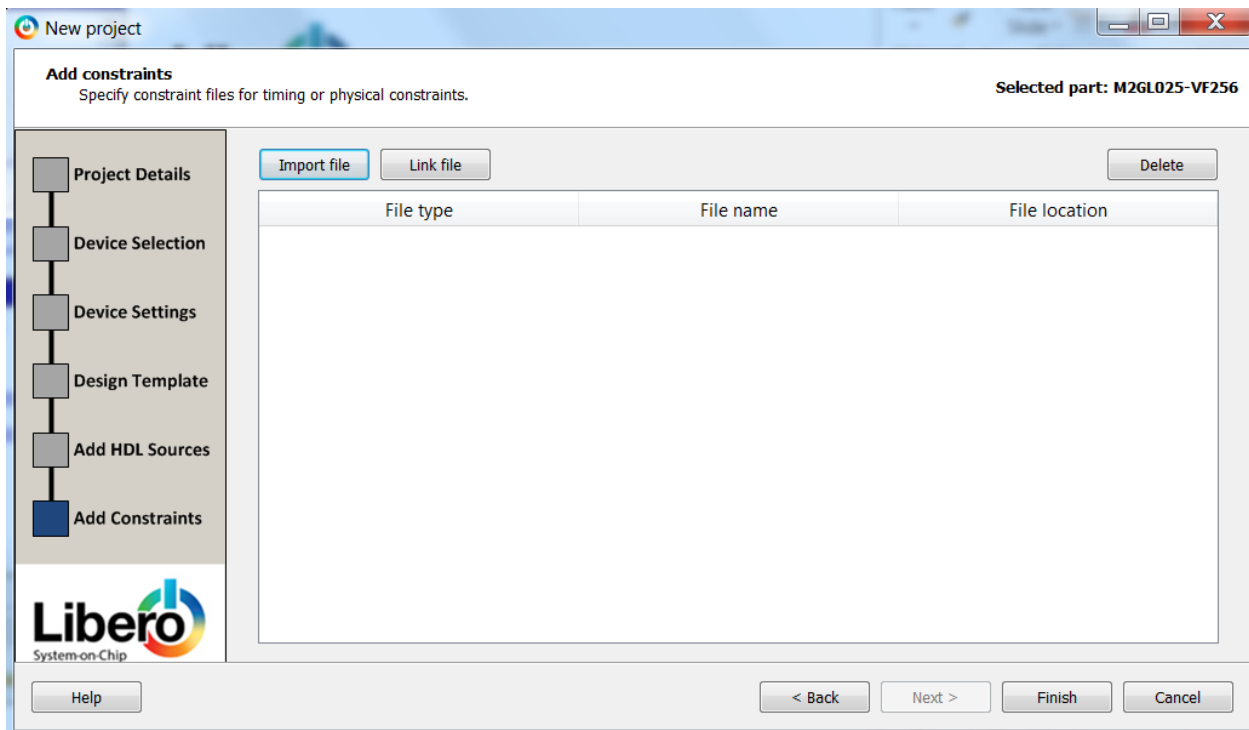


If you have successfully added all 5 VHDL files, your screen should look like this (or with Verilog files).

The order may be different depending on the order in which you selected and added them.

Press the **NEXT** button.

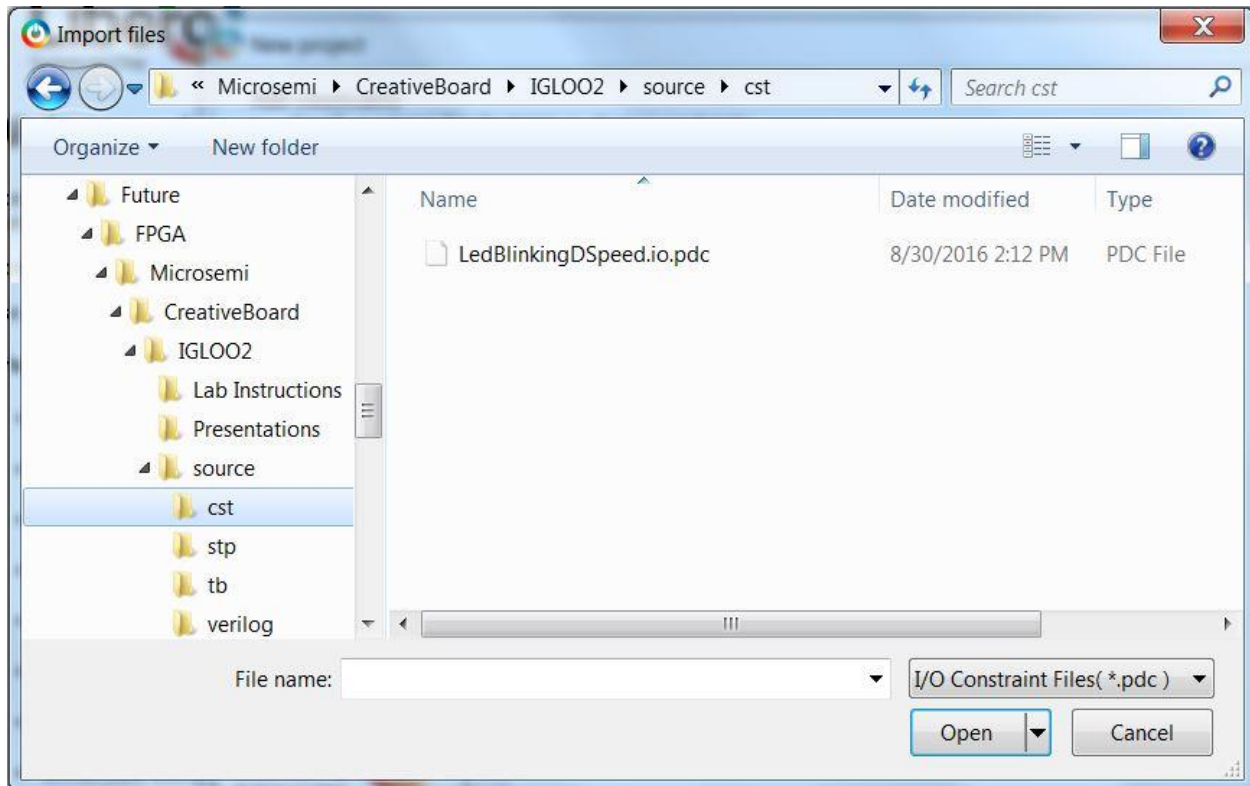
Add Constraints



We will now add some constraints files to the design.

Select the **Import file** button.

An overlay window will appear.



We will be adding both **.pdc** (IO) and **.sdc** (timing) constraints.

Path into the **<lab install folder>** -> **Source** -> **cst** folder.

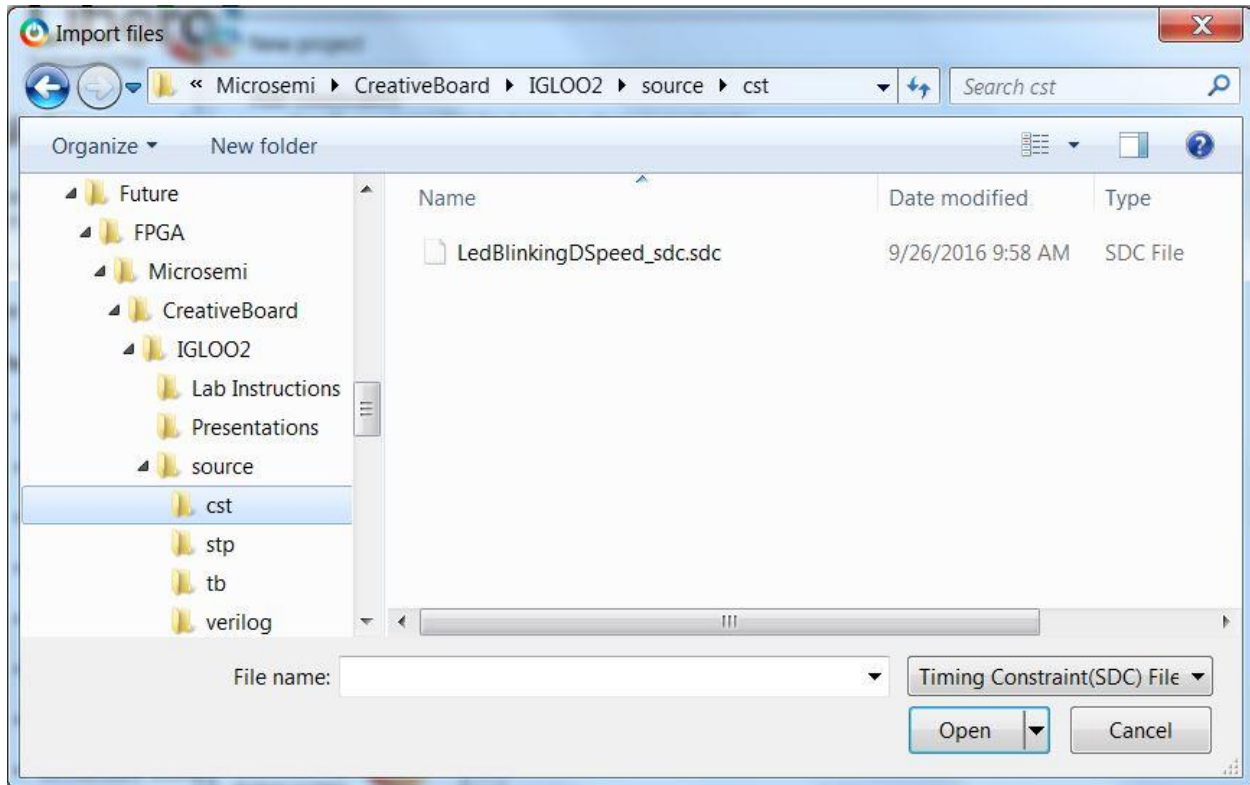
Verify that the filter selection just above the OPEN button is set to **I/O Constraint Files(*.pdc)**.

Select the **LedBlinkingDSpeed.io.pdc** file.

Press the **OPEN** button.

Select the **Import file** button again.

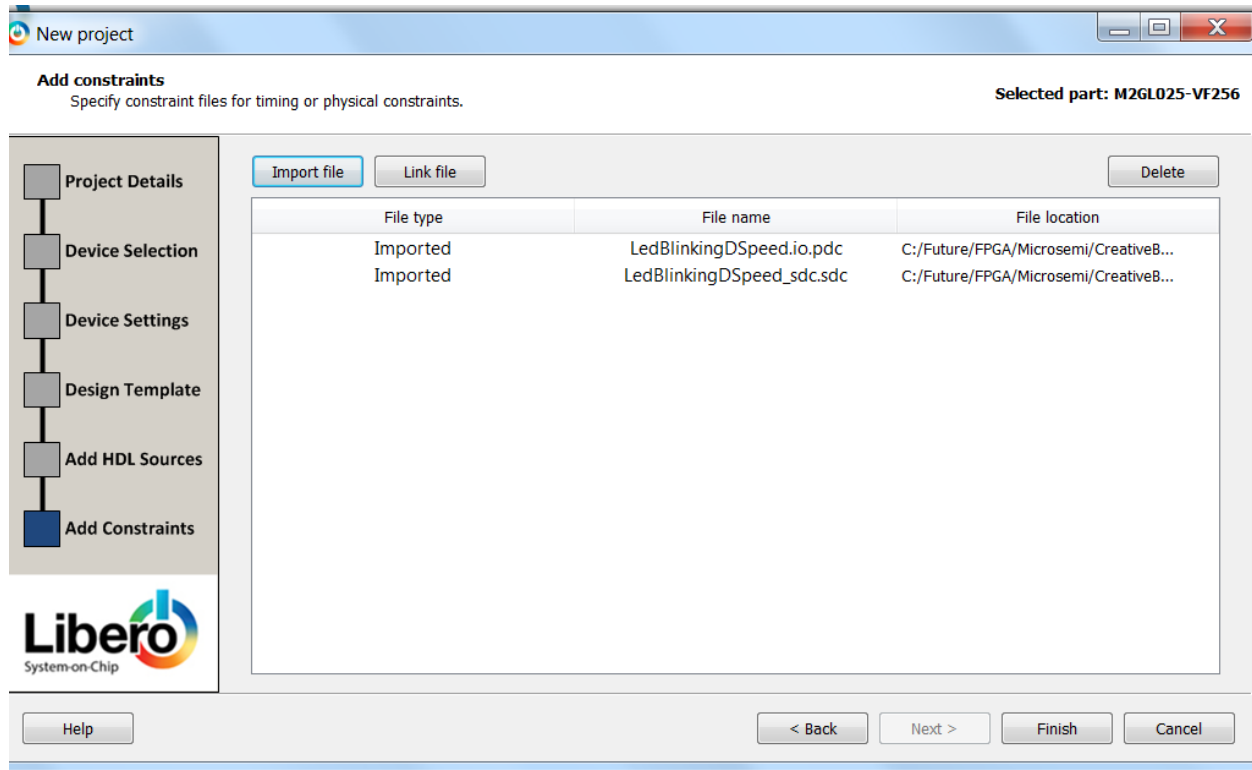
The overlay window will appear again.



Change the filter selection just above the OPEN button is set to **SDC Files(*.sdc)**.

Select the **LedBlinkingDSpeed_sdc.sdc** file.

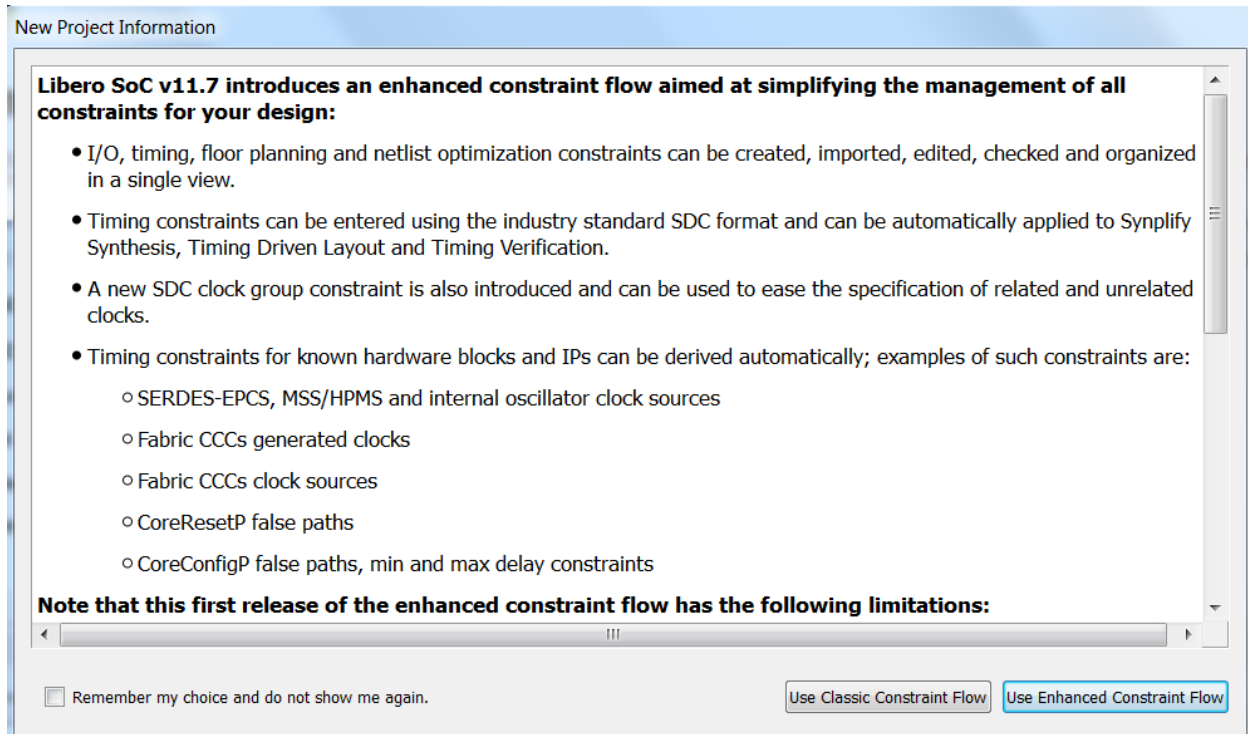
Press the **OPEN** button.



If you have successfully added both constraints files, your screen should look like this.

Press the **FINISH** button.

And overlay window will appear.



For this lab we will select the **Use Enhanced Constraint Flow**.

Press the **Use Enhanced Constraint Flow** button.

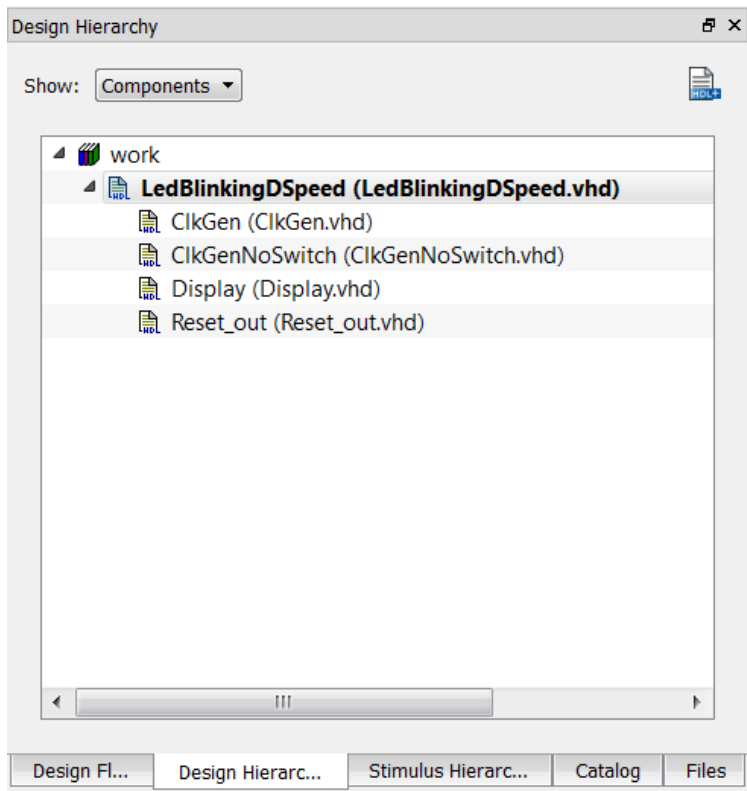
After a short amount of time, the main interface of the Libero tool set will appear.

We will now take a brief look at some of the windows that are in this interface.

At the bottom of the top-left window you will find several tabs with the following titles:

Design Flow Design Hierarchy Stimulus Hierarchy Catalog Files

Click on the **Design Hierarchy** tab.



You may need to expand to see all the files.

Verify that the file titled **LedBlinkingDSpeed.v(hd)** is the Bold File, indicating it is the top of the Design.

If it is NOT, then right click it and **Set as Root**.

Click on the various tabs to get familiar with the interface.

Can you tell us where the IO and Timing constraints files are located?

(Don't use Windows Explorer – use this tool.)

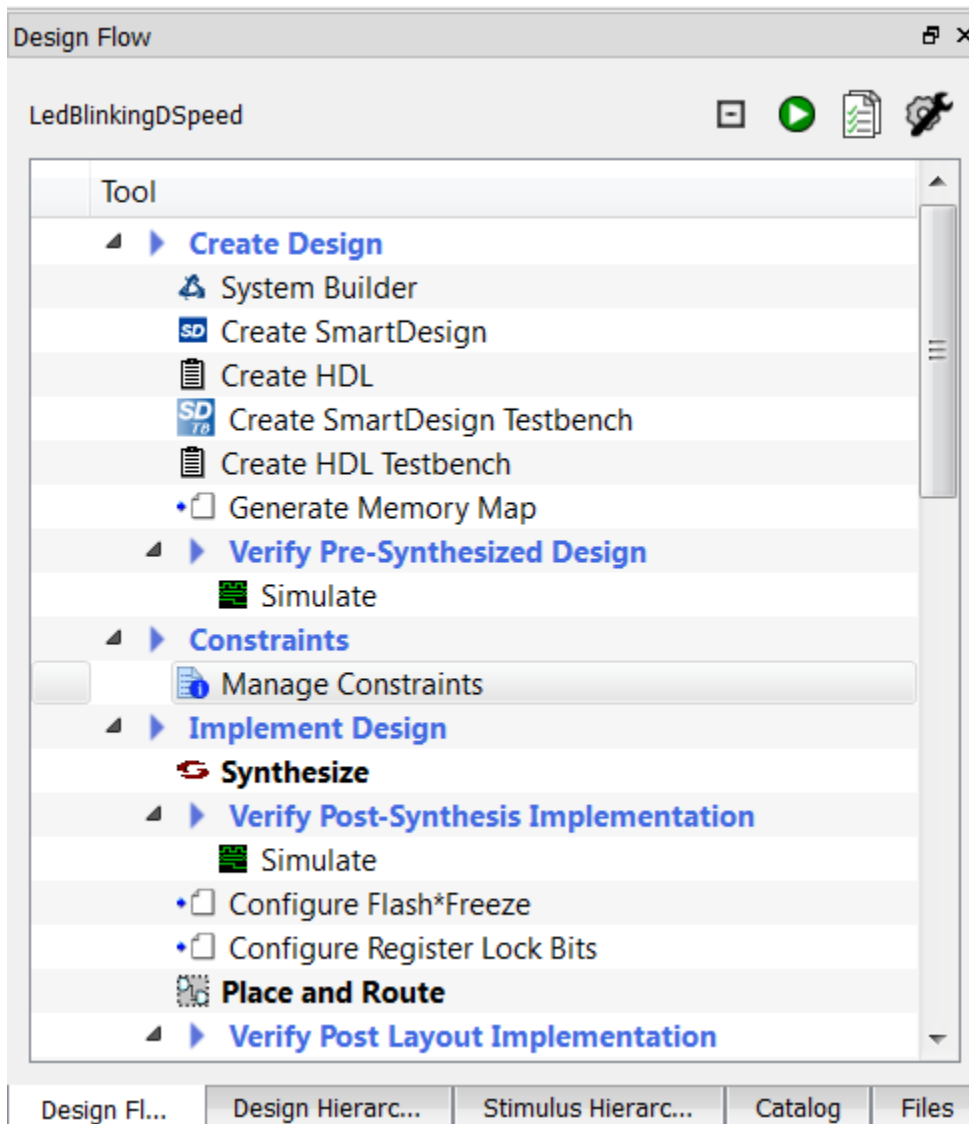
What is the path on your hard drive? (It is okay not to tell the entire path, but the last part of it.)

IO (.pdc) is located _____

Timing (.sdc) is located _____

When done, go to the next step.

Click on the **Design Flow** tab.



It should look something like above.

Before running Synthesize, you will need to have the timing constraints applied for Synthesize, Place and Route, and Timing Analysis.

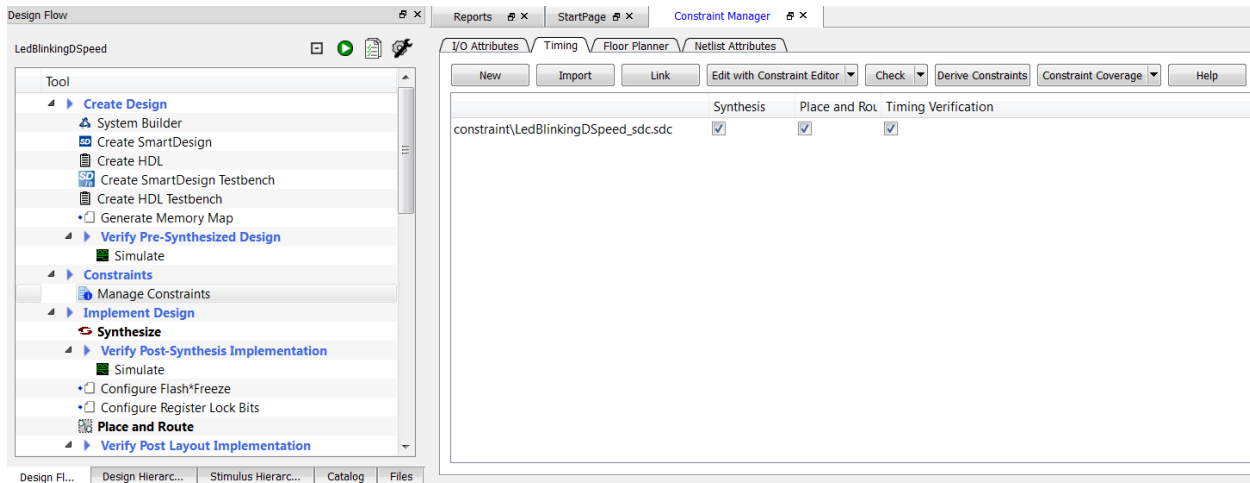
Double-click on **Manage Constraints** under **Constraints**, as shown above.

A new window will open in the right hand pane.

It is called the **Constraints Manager** window.

There are multiple sub-tabs within that window.

Select the **Timing** tab.



Check the **Synthesis**, **Place and Route** and **Timing Verification** boxes.

Select the **I/O Attributes** tab.

Check the **Place and Route** box.

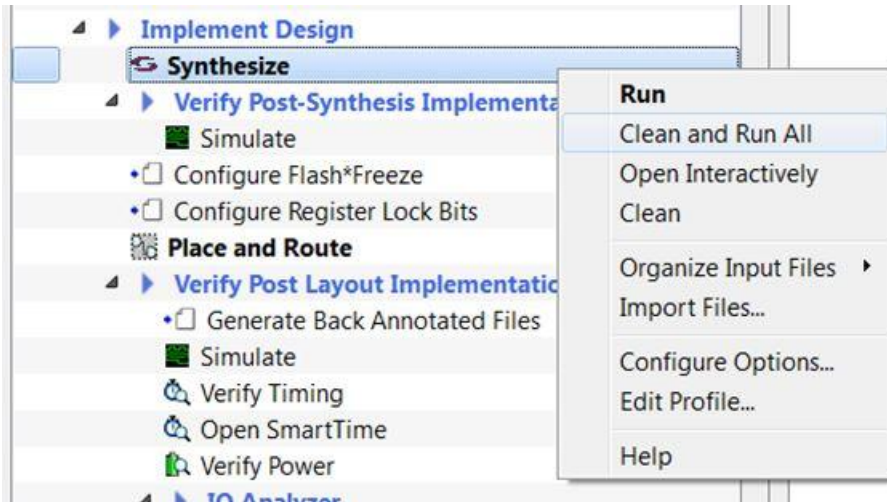
Explore these tabs a little bit. Notice that you can create or add additional constraints files with this window, and that you can apply different constraints to different aspects of the design flow when processing your designs. Also notice that you can inform the tool that you might be targeting different variants or sizes of target parts, and that the tool should help you not make conflicting selections.

When done exploring, close the **Constraint Manager** by selecting the **X** on the Constraint Manager window tab.

Next we will synthesize and simulate the design.

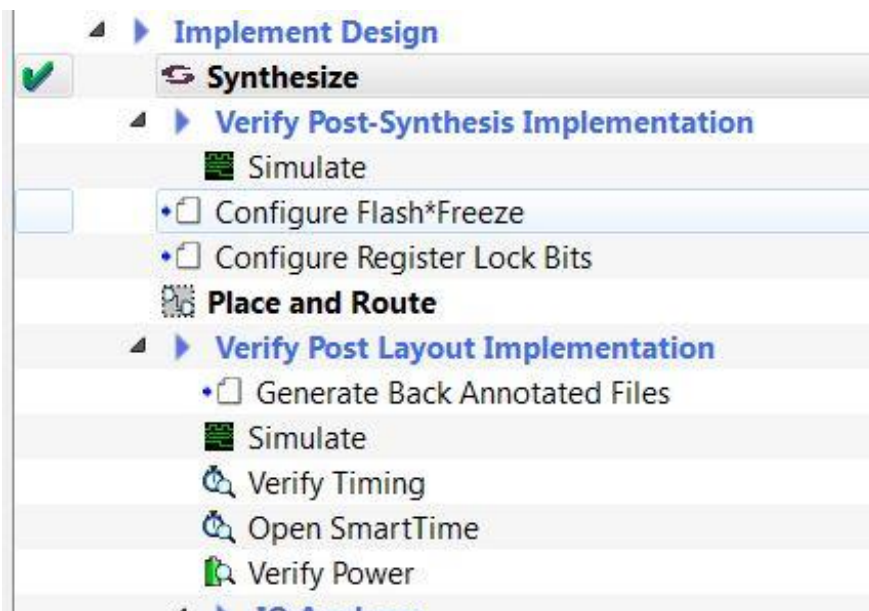
If all looks good, we will then Place & Route the design and Program the device.

Synthesize the Design



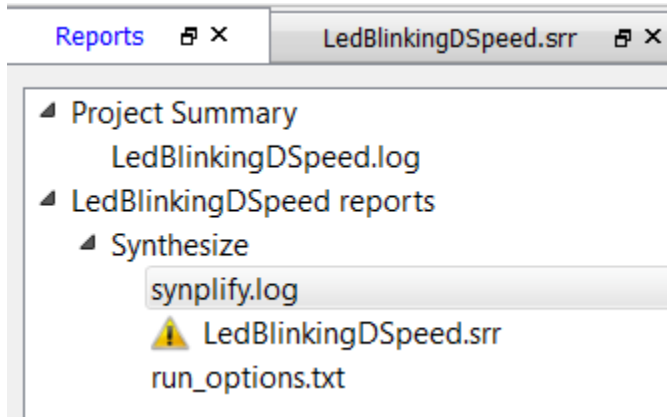
In the **Design Flow** window, under the **Implement Design** stage, *Right-Click Synthesize* to expose some options, then *Click Clean and Run All* (if it asks 'Are you sure...', *click Yes*).

If there are no syntax errors or other issues with your source files, a **GREEN Check Mark** will appear indicating a successful synthesis of your design.



Synthesis Reports

Find the **Reports** tab in the main window to the right of the Design Flow window. The Report tab was created when you ran the Synthesize step above.



Please take a few minutes to become familiar with the reports:

Synplify.log Synthesize flow instructions and associated return codes

<DesignTop>.srr Premapping Report, Clock Optimization Report
Performance Summary, Resource Usage Report

In the clock summary section of this report, what is the Requested Clock Speed? _____

Run_options.txt Commands run from Create Design through Synthesize

At the end of the Run_options report, what is the file extension of the resulting output file? _____

Post Synthesis Simulation

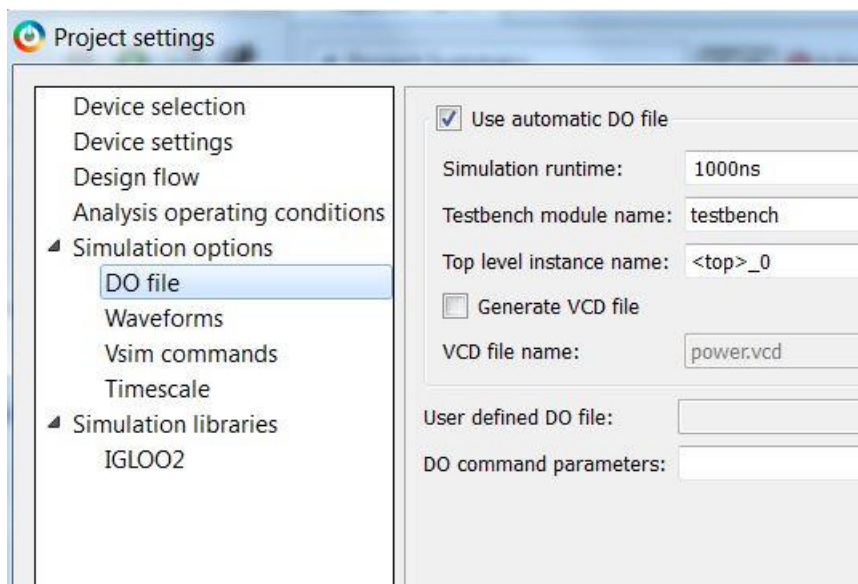
After we have successfully synthesized our design, we will take a look at the functional operation of the design to make sure it is doing what we expect. Note that you can and should be running Simulation before you Synthesize; but since we have provided all the code for you there should be no errors, so we have decided to show that you can also run Simulation post Synthesize as well.

We will add a pre-written simulation **stimulus file** to our design environment to save you some time.

In the Main GUI, click **Project**, then **Project Settings...**

An overlay window will open and should look familiar from when you set the target part.

Click on **DO file** under the **Simulation options**.



Your screen should look like this before you make the changes.

Verify **Use automatic DO file** is checked.

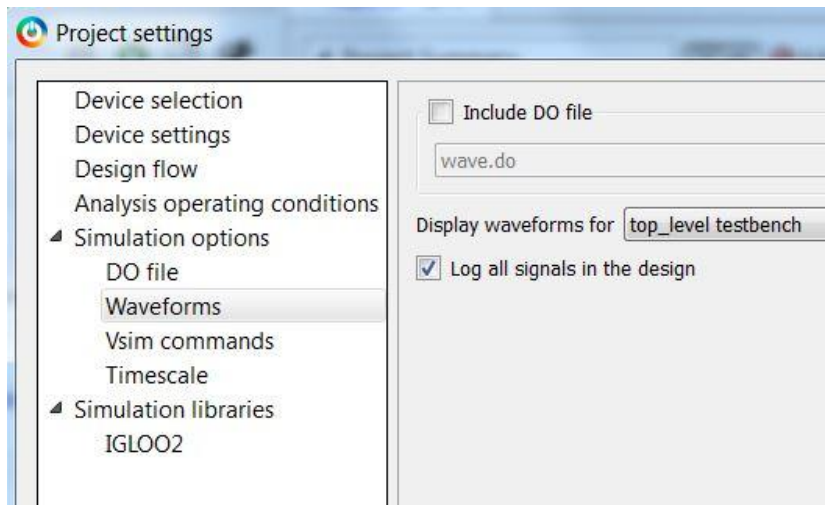
Change the *Simulation runtime*: to **650us**

Change the *Testbench module name*: to **LedBlinkingDSpeed_tb**

Note: You are doing this because we have provided you with a prewritten testbench file. You will still need to load the actual file, which we will do in a few more steps.

Click the **Save** button in the upper right.

Click on **Waveforms** under the **Simulation options**.

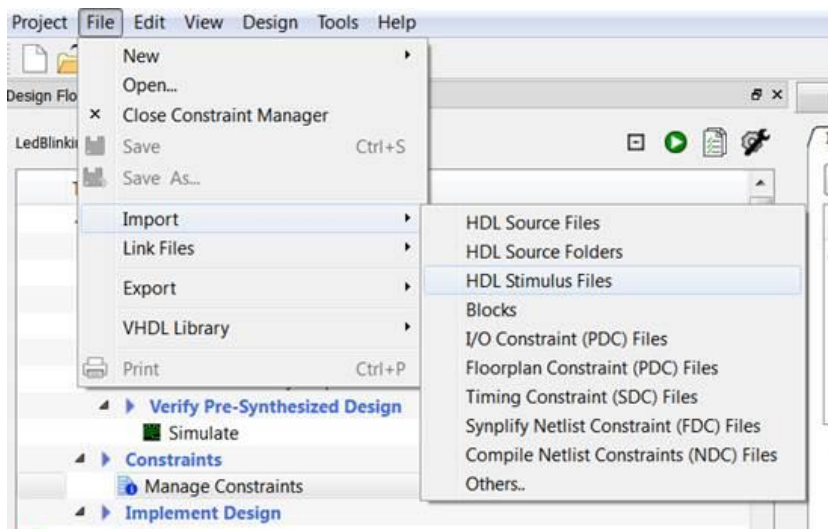


In Waveforms, *check* the box to enable **Log all signals in the design**.

Click the **Save** button in the upper right.

Click the **Close** button in the lower right.

In the Main GUI, click **File**, then **Import** > *Slide* over and click **HDL Stimulus Files**.



An overlay window will appear.

Browse to the **C:/Future/FPGA/Microsemi/CreativeBoard/IGLOO2/source/tb** folder.

Select the **LedBlinkingDSpeed_tb.v(hd)** file.

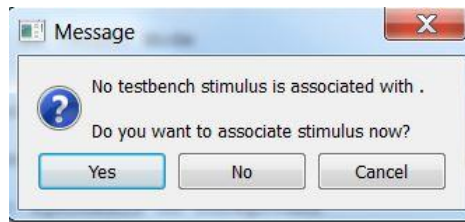
Click the **OPEN** button.

The file was imported into the ...\\IGLOO2\\Lab1\\stimulus folder.

Double-click on **Simulate** under **Verify Post-Synthesis Implementation**.

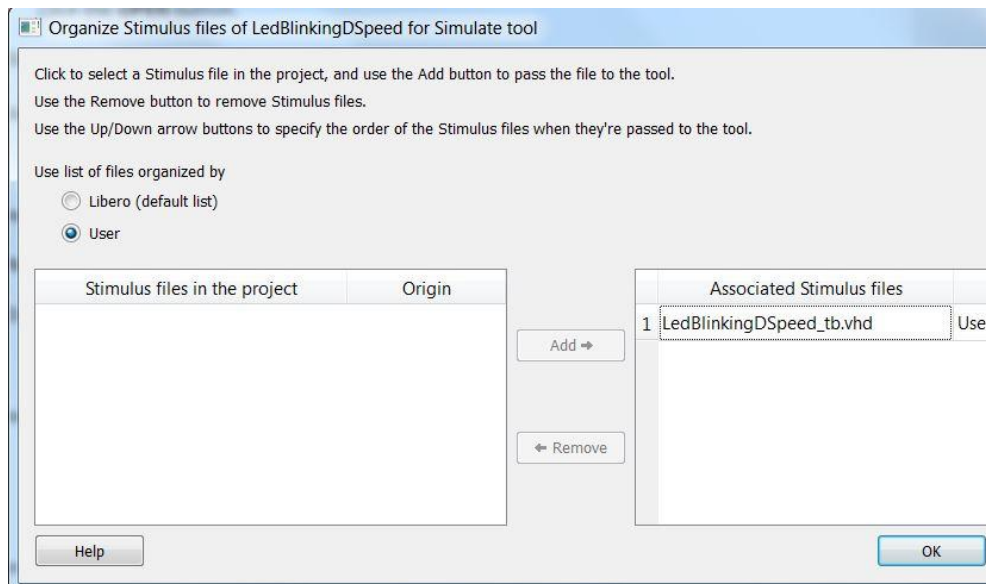
Note: Be careful **NOT** to click the one under Place and Route.

If a message window pops up saying



Click **Yes**

A window will open called **Organize stimulus files of LedBlinkingDSpeed for Simulate tool**.



Under **Use list of files organized by** Select the **User** radio button.

Select the **LedBlinkingDSpeed_tb.v(hd)** file and press the **Add →** button.

The selected file will move to the right window.

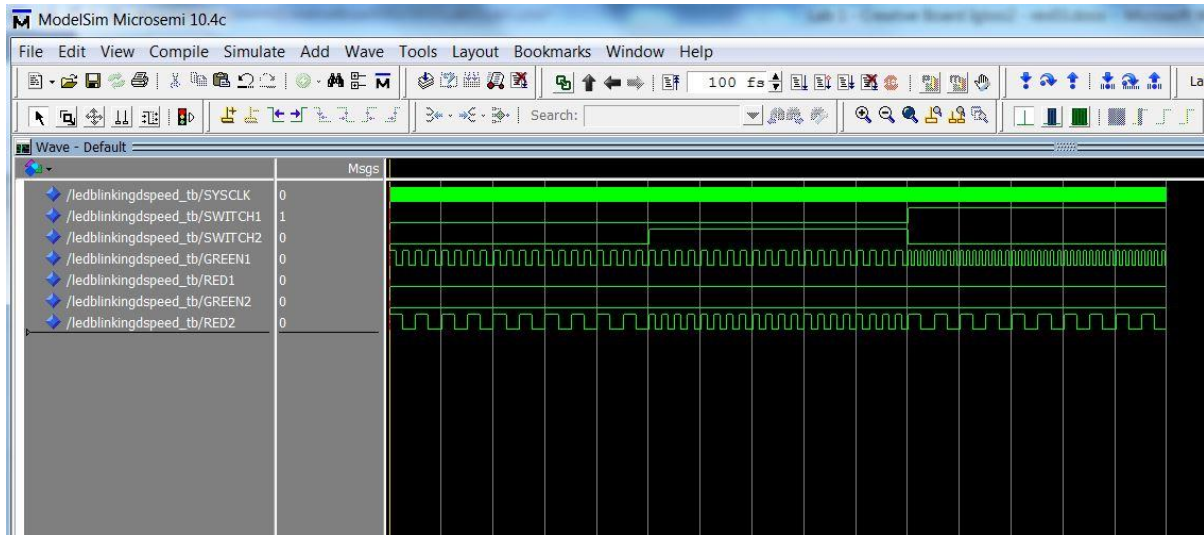
Click the **OK** button.

ModelSim will launch and a bunch of stuff will get executed automatically (that is what that .do file was for), and a series of windows (tabs across the bottom) will get opened and populated. You should be presented with the **Wave** tab. If not, *click* the **Wave** tab at the bottom of the window.

After a few more seconds, the Wave window will populate with a simulation of your design.

Right-click anywhere in the black waveform window, and select **Zoom Full**.

Simulation Results



Take a look around the ModelSim tool set.

Right-click in the Wave window to zoom in and zoom out.

Look at the other windows within ModelSim to get a feel for things.

The Transcript Window shows a list of all the tasks that were run via the .do file.

When you are done examining the interface, close ModelSim by either *clicking* **File -> Quit** or *clicking* the **X** in the upper right corner.

If prompted, click **Yes**.

Adjust the Blink Rate in the Code for FPGA Compiling

The code that was Synthesized and Simulated has some values set to make the simulation run faster.

We now need to modify the code by adjusting some values so that it will run at the desired speed in real hardware. The code fragments have already been written for you. All you will need to do is comment out certain lines and uncomment others.

In the left window, change from the **Design Flow** tab to the **Design Hierarchy** tab.

Double-click the **LEDBlinkingDSpeed** source file to open it for editing (this is VHDL code).

```
81 | -----
82 | -- Scale Factor used to simulate for Lab1
83 | -- They are shorter times to make the simulation go quicker
84 | -- Comment these if you want to make real hardware
85 | signal Scale_Factor0 : std_logic_vector(31 downto 0) := "0000000000000000000000001111101000"; --every 20 us
86 | signal Scale_Factor1 : std_logic_vector(31 downto 0) := "00000000000000000000000011111010"; --every 5 us
87 | signal Scale_Factor3 : std_logic_vector(31 downto 0) := "000000000000000000000000111110100"; --every 10 us
88 | -----
89 | -- Scale Factor used to program the board for Lab1
90 | -- They provide longer delays so you can see the LEDs blink
91 | -- Comment these factors if you want to simulate
92 | --signal Scale_Factor0 : std_logic_vector(31 downto 0) := "00000101111101011110000100000000"; --every 2 sec
93 | --signal Scale_Factor1 : std_logic_vector(31 downto 0) := "00000001011111010111100001000000"; --every 0.5 sec
94 | --signal Scale_Factor3 : std_logic_vector(31 downto 0) := "00000010111110101111000010000000"; --every 1 sec
95 | -----
96 | -- Scale Factor used to program the board for Lab 2
97 | -- These change the Blink rates from Lab1 for Lab2
98 | -- Comment these factors if you want to simulate
99 | --signal Scale_Factor0 : std_logic_vector(31 downto 0) := "00001011111010111100001000000000"; --every 4 sec
100 | --signal Scale_Factor1 : std_logic_vector(31 downto 0) := "00000000101111101011110000100000"; --every 0.25 sec
101 | --signal Scale_Factor3 : std_logic_vector(31 downto 0) := "00000001011111010111100001000000"; --every 0.5 sec
102 | -----
103 |
104 | -- component
105 | component Reset_out
106 |
```

Scroll down the file until you find the above code.

*comment out **signal Scale_Factor0**, **signal Scale_Factor1** and **signal Scale_Factor3** under the comment "Scale Factor used to simulate for Lab1"*

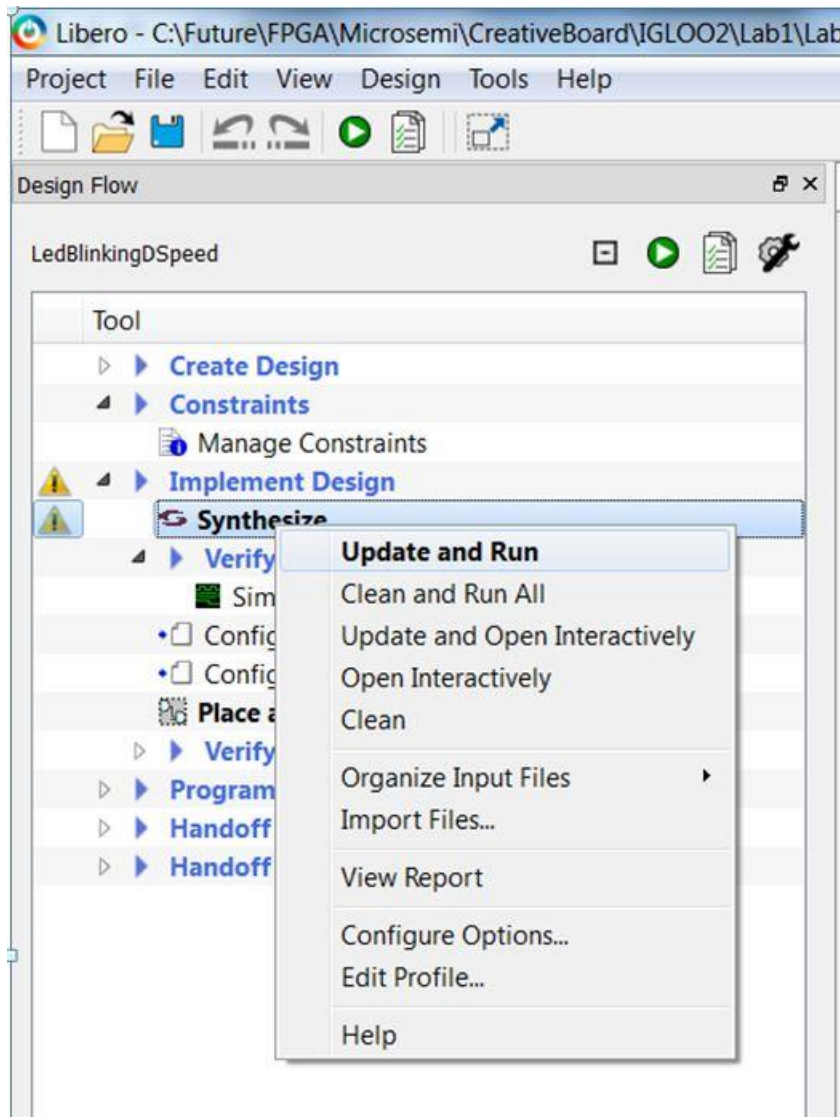
*uncomment **signal Scale_Factor0**, **signal Scale_Factor1** and **signal Scale_Factor3** under the comment "Scale Factor used to program the board for Lab1"*

There is a fast way to do this to multiple lines of code as follows:

Select multiple lines in the editor, *Right-click* and select **comment selection** or **uncomment selection** to make the edits quickly.

Save the file as the existing file name (overwrite it if asked).

Resynthesize the Design

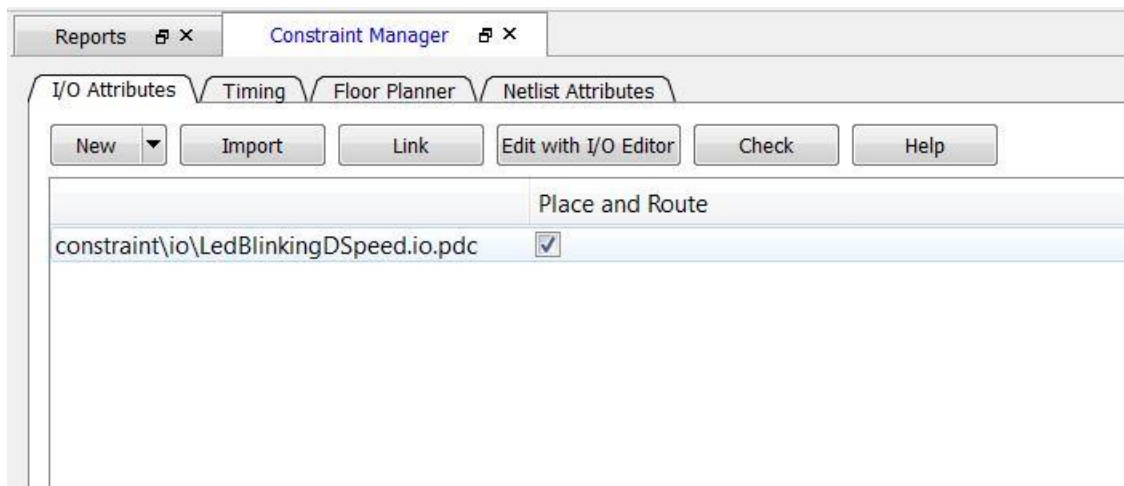


Back in the **Design Flow** tab, find and *Right-click* **Synthesize** and *select* **Update and Run**.

This will **rerun** the flow through Synthesis (due to updated source code).

Reviewing Constraints before running Place and Route

In the **Design Flow** tab, open the **Manage Constraints** window again and position yourself on the **I/O Attributes** tab.

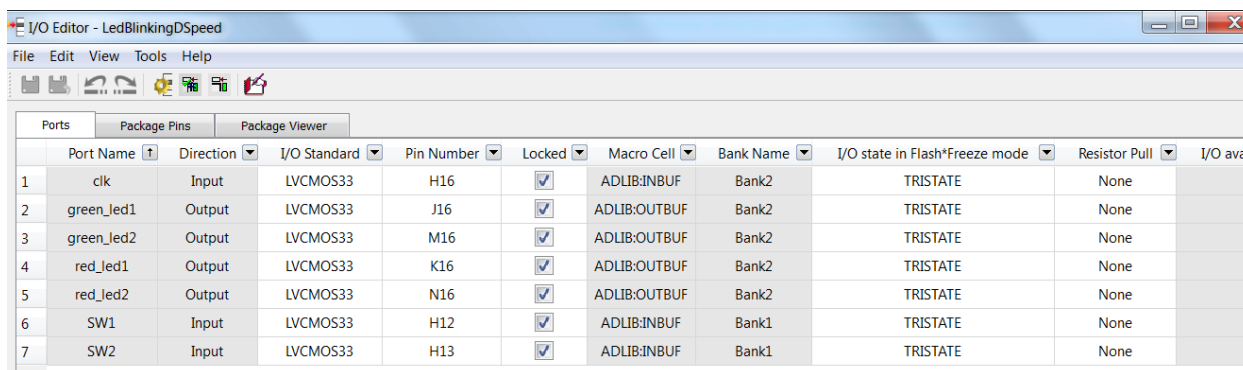


Click the **Edit with I/O Editor** button and the window will grey out while the I/O Editor is launching. This may take up to a minute depending on the size of the design. You may briefly see the Microsemi Splash Logo appear on your screen.



Look at your task bar, as sometimes it opens in the closed mode.

I/O Editor



	Port Name	Direction	I/O Standard	Pin Number	Locked	Macro Cell	Bank Name	I/O state in Flash*Freeze mode	Resistor Pull	I/O available
1	clk	Input	LVC MOS33	H16	<input checked="" type="checkbox"/>	ADLIB:INBUF	Bank2	TRISTATE	None	
2	green_led1	Output	LVC MOS33	J16	<input checked="" type="checkbox"/>	ADLIB:OUTBUF	Bank2	TRISTATE	None	
3	green_led2	Output	LVC MOS33	M16	<input checked="" type="checkbox"/>	ADLIB:OUTBUF	Bank2	TRISTATE	None	
4	red_led1	Output	LVC MOS33	K16	<input checked="" type="checkbox"/>	ADLIB:OUTBUF	Bank2	TRISTATE	None	
5	red_led2	Output	LVC MOS33	N16	<input checked="" type="checkbox"/>	ADLIB:OUTBUF	Bank2	TRISTATE	None	
6	SW1	Input	LVC MOS33	H12	<input checked="" type="checkbox"/>	ADLIB:INBUF	Bank1	TRISTATE	None	
7	SW2	Input	LVC MOS33	H13	<input checked="" type="checkbox"/>	ADLIB:INBUF	Bank1	TRISTATE	None	

Confirm all pins in the design are **Locked**. (A locked pin means that the placement is now fixed and the place and route tool must place that Port Name in the design on only that physical pin.)


Look around the tool a little bit but do NOT change any settings.

Close the I/O Editor when done.


Back in the **Constraints Manager** window, click the **Timing** tab.

Click on the **Edit with Constraints Editor** button, and then click **Edit Place and Route Constraints**.

Like before, the window will grey out, the logo will briefly appear, and then the **Constraints** window will

appear (or be minimized). 

Constraints Editor

	Syntax	Clock Name	Clock Source	Period (ns)	Frequency (MHz)	Dutycycle (%)	First Edge	Offset (ns)	Waveform
1	Click here to add a constraint								
2		LedBlinkingDSpeed clk	[get_ports { clk }]	20.000	50.000	50.0000	rising ▼	0.000	0 10

Confirm **50 MHz** clock constraint is properly set.

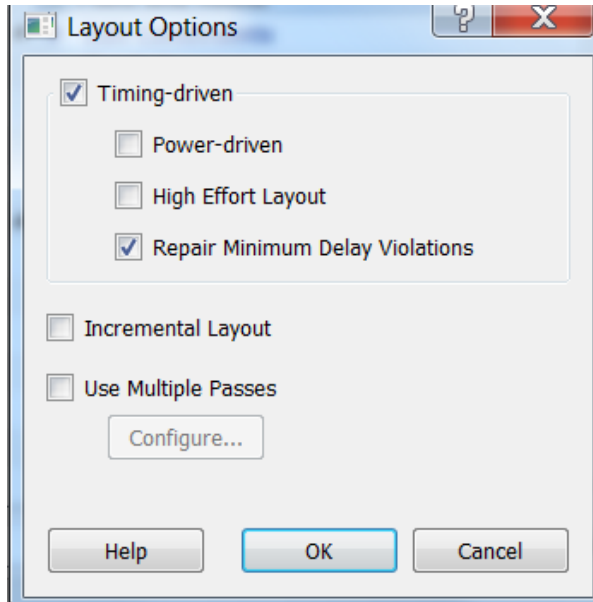
Click around a little bit and ask yourself what all these other setting might be used for.

When attacking larger designs, this tool gives you the ability to set constraints (objective) that the different tools use to help your design meet timing. This way, the tools only work hard on the places that need the extra help and can relax a little in the places that have additional timing margins.

Close the **Constraints Editor**.

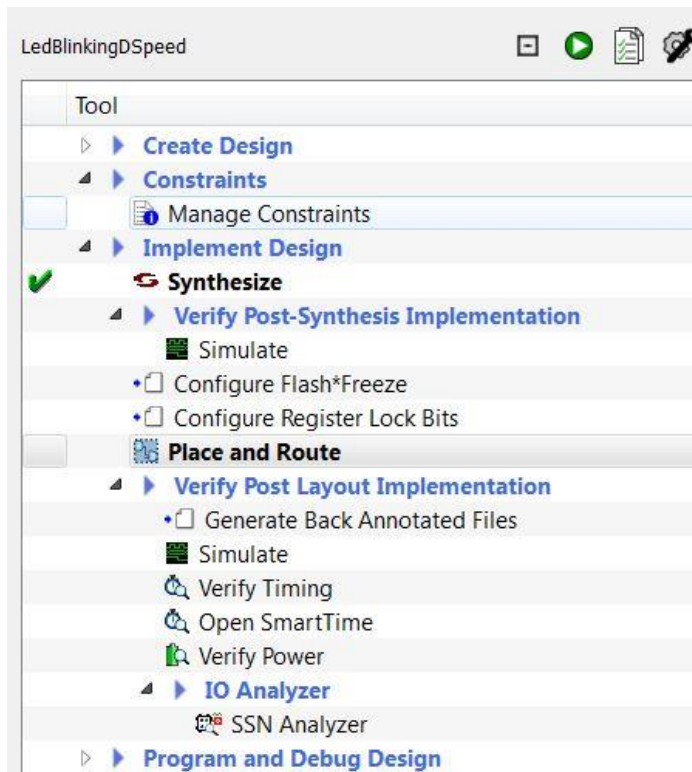
Place and Route the Design

Take a look at the additional Place and Route options by *Right-clicking* on **Place and Route** and selecting **Configure Options...**




These options allow you to tailor the design to your needs. Click **Cancel**.

Double-click on **Place and Route** or Right-click and select **Run**.



Place and Route Reports

Place and Route

LedBlinkingDSpeed_glb_net_report.xml
LedBlinkingDSpeed_mindelay_repair_report.rpt
 LedBlinkingDSpeed_layout_log.log

When **Place and Route** has completed, the Reports window will get populated with additional information.

Take a few minutes to inspect the new information presented in the reports:

LedBlinkingDSpeed_compile_netlist_resources.xml - from the Compile step

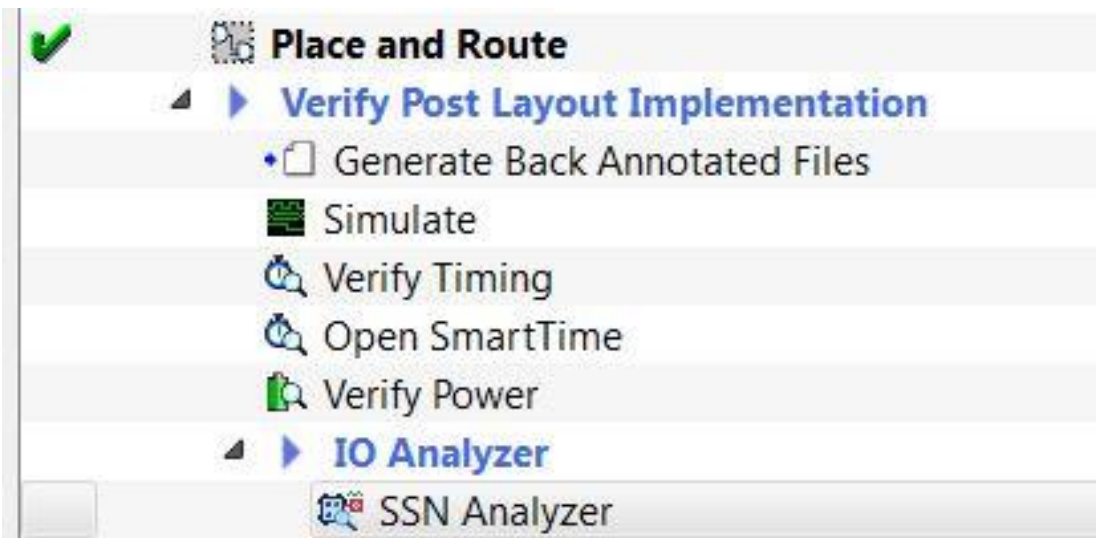
LedBlinkingDSpeed_compile_netlist_hier_resources.csv - from the Compile step

LedBlinkingDSpeed_glb_net_report.xml - from the Place and Route step

LedBlinkingDSpeed_mindelay_repair_report.rpt - from the Place and Route step

LedBlinkingDSpeed_layout_log.log (and more)

There are quite a few additional features available to better understand your design once you have run Place and Route.

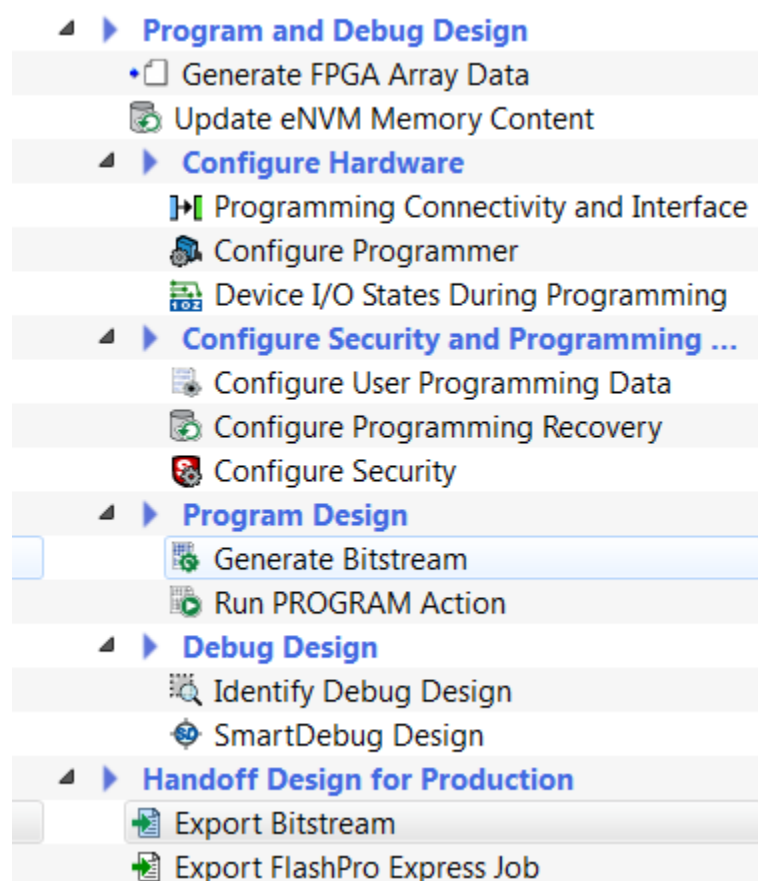


Post Route Simulation Timing Verification

Power Verification IO Analysis including SSN Analysis

Generate the Programming file (Generate Bitstream)

Under **Program Design**, Double-click on **Generate Bitstream** to generate the bitstream.



There will be another line of information in the reports section to the right.

We'll look at that later.

Program the Device

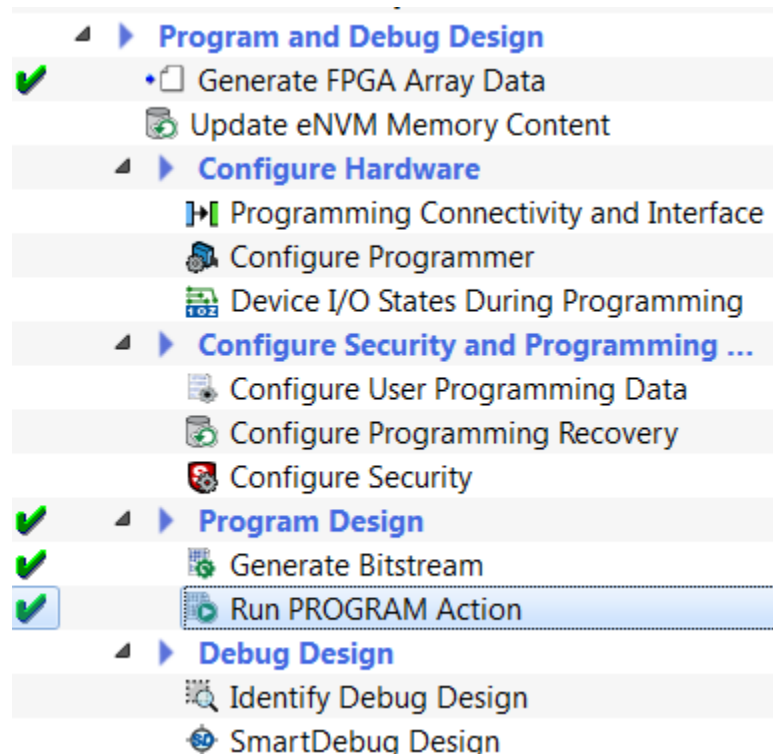
OK, it is finally time to get that board out and connect it to the laptop USB port.

This will power the board and make the needed hardware connection to the programming software.

(If you really need help connecting the board, ask your instructor)

Under **Program Design**, Double-click on **Run PROGRAM Action** to program the IGLOO2 device on the board.

(If you cannot find **Run PROGRAM Action** – Right-click on **Run xxxxx Action** and Click on **Configure**).



The Programmer will begin to verify that a cable is connected, Blank the Device, and Program it.

After about 55 seconds (give or take), you should notice a **Green check mark** will be placed on **Program Design** and associated 2 sub-actions.


The LEDs should now be blinking.

Go back and reread what you are expecting the LEDs to be doing, and verify your results.

Press some buttons – do you remember which one should do what?

Can you get both LEDs to blink at the same rate? In phase? Alternating?

Program and Debug Design Reports

- ▴ Generate FPGA Array Data
 LedBlinkingDSpeed_init_config.xml
- ▴ Generate Bitstream
 LedBlinkingDSpeed_generateBitstream.log
- ▴ Run PROGRAM Action
  LedBlinkingDSpeed_PROGRAM.log

View the reports generated from Program Design as shown on the Reports tab.

Generate FPGA Array Data

LedBlinkingDSpeed_init_config.xml

Other than the title and date, what does this report tell you? _____

Generate Bitstream

LedBlinkingDSpeed_generateBitstream.log

What is the CHECKSUM for this design? _____

Run PROGRAM Action

LedBlinkingDSpeed_PROGRAM.log

What was the total elapsed time for the Programmer Action? _____

Congratulations!!! Lab 1 is now complete.

Revision History:

First Release	A	11-Oct-16	Daryl Lee Specter
Corrections for typos and to add clarity (A1 skipped)			
pages 9, 12, 13, 15, 16, 20, 22, 24, 27, 28	A2	13-Nov-16	Daryl Lee Specter

[illegible]

[illegible]