iCEcube2 Tutorial

v1.0 – May 2010

Copyright Information

Notice of Disclaimer: This software is provided to you "as-is," without any express or implied warranty.

Contact Information

Lattice Semiconductor Corporation
5555 N.E. Moore Court
Hillsboro, Oregon 97124-6421
United States of America
Tel: +1 503 268 8000
Fax: +1 503 268 8347
www.latticesemi.com

Revision History

The following table lists the revision history of this document.

Version	Revision
1.0	Release iCEcube2 2010.03

Preface

About this Document

The *iCEcube2 Tutorial* provides iCE FPGA designers with an overview of the software tool and the design process using iCEcube2. This document covers the iCEcube2 tools for Project Setup, Navigation, and Physical Implementation on the iCE40 FGPA device.

For information on the Synopsys Synplify Pro software, please refer to the Synplify Pro documentation provided in the synpro/doc directory in the iCEcube2 software installation (<icecube2_install_dir>/synpro/doc), and on the Lattice website.

For information on the Aldec Active-HDL design tool, please refer to the Active-HDL documentations available at <icecube2_install_dir>/Aldec/Active-HDL/BOOKS or Active-HDL product help which can be invoked from the help menu in the software tool.

For detailed information of the iCEcube2 development tools, please refer to the iCEcube2 User Guide.

Software Version

This tutorial is based on iCEcube2 Software Version 2012.06

For more information about acquiring the iCEcube2 software, please visit the Lattice Semiconductor website: http://www.latticesemi.com.

Platform Requirements

The iCEcube2 software can be installed on a platform satisfying the following minimum requirements.
A Pentium 4 computer (500 MHz) with 256 MB of RAM, 256MB of Virtual Memory, and running one of the following Operating Systems :

- Windows 7 OS, 32-bit / 64-bit
- Windows XP Professional
- Red Hat Enterprise Linux WS v4.0

For installation help, refer to iCEcube2 Install overview document, please visit the software documentation webpage.

Programming Hardware

Here are the following ways to program iCE FPGA devices:

- A third party programmer, using the programming files generated by the iCecube2 Physical Implementation Tools. Consult the third party programmer user manual for instructions.
- The iCEblink and iCEman evaluation Boards, which not only serves as a vehicle to evaluate iCE FPGAs, but also includes an integrated device programmer. This programmer can be used to program devices on the evaluation boards, or it can be used

to program devices in a target system. Please visit Lattice Semiconductor website: http://www.latticesemi.com for additional information on the Evaluation Boards.

- Digilent USB cables
- The iCE Programming hardware : iCEcable, iCEprog (Programmer base module) and iCEsab(socket adaptor). Refer to lattice website: http://www.latticesemi.com for more details on programming hardware.

Overview

iCEcube2 Tool Suite

The iCEcube2 Tool Suite is comprised of several integrated components, running under either the Microsoft Windows or the Red Hat Linux environments. Please refer to Platform Requirements for additional information on supported operating systems.

The Figure 1 below depicts the design flow using the iCEcube2 Tool Suite. The components in blue signify functionality supported by Lattice Semiconductor's proprietary iCEcube2 software, and the components in purple indicate the functionality supported by Synopsys' Synplify Pro synthesis tools and the Aldec Active-HDL simulation tool. The iCEcube2 software, Synopsys Synplify Pro and the Aldec Active-HDL software constitute the iCEcube2 Tool Suite.

Note: The Aldec Active-HDL tool is available only in Windows environments.

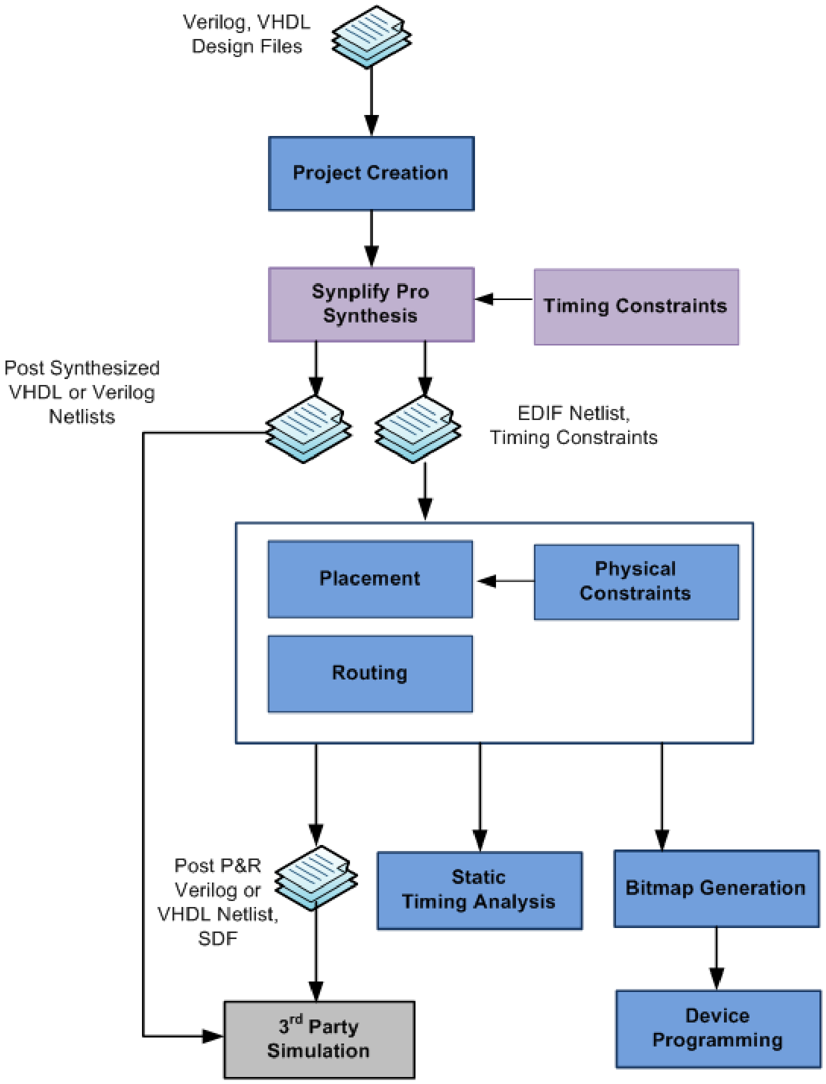


Figure 1: The iCEcube2 Design Flow

Design Flow

The following steps provide an overview of the design flow using the iCEcube2 Tool Suite.

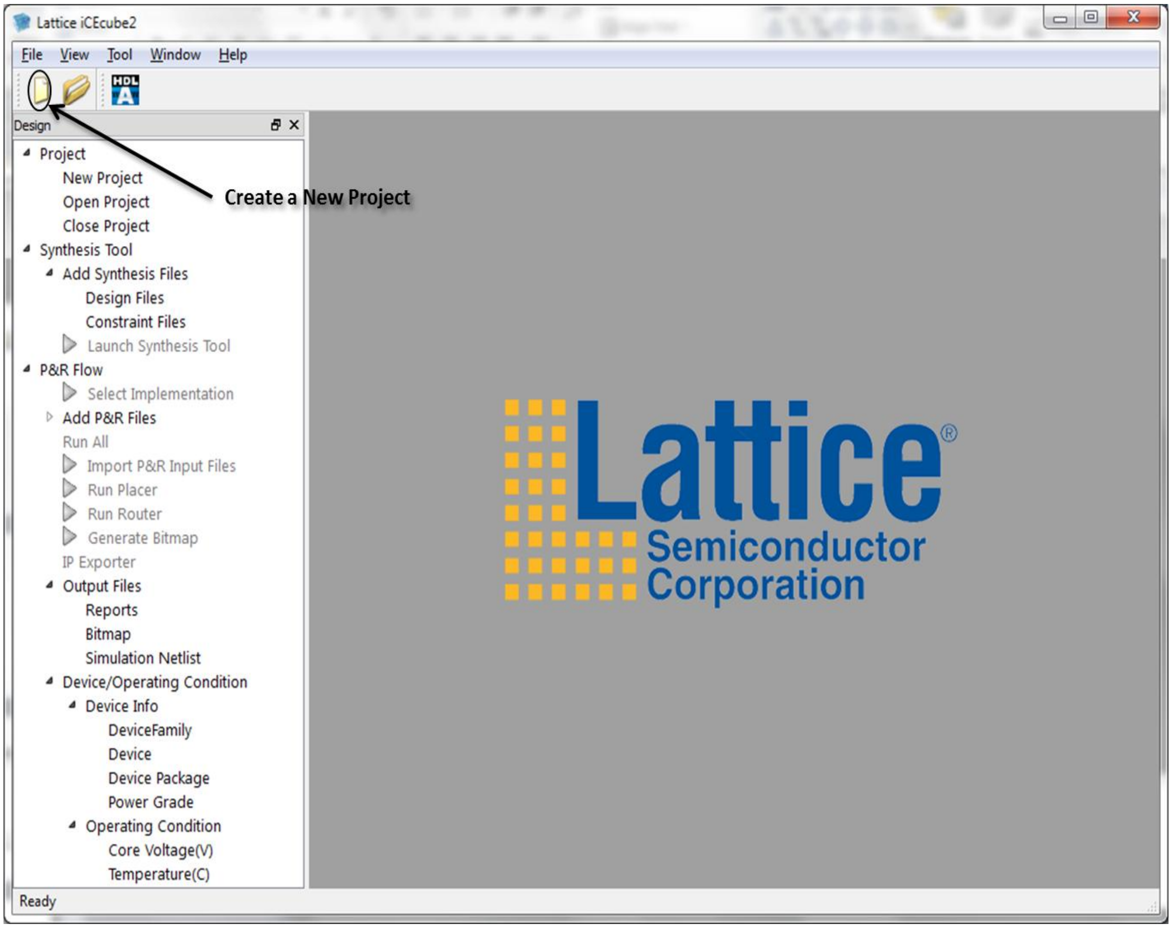
1. Create a new project in the iCEcube2 Project Navigator and specify a target device and its operating conditions. Add your HDL (Verilog or VHDL) design files and your Constraint files to the project.
2. Synthesize your design using the Synplify Pro design software. This software has been provided as part of the iCEcube2 Tool Suite, and can be invoked from the iCEcube2 Project Navigator. Within the Synopsys design environment, assign your Logic Synthesis, Timing and Pin constraints.
3. Perform Placement and Routing using the iCEcube2 place and route tools. iCEcube2 also supports physical implementation tools such as floor planning, allowing users to manually place logic cells and IOs.
4. Perform timing simulation of your design using the Aldec Active-HDL simulation tool or any industry-standard HDL simulation tool. The files necessary for simulation are automatically generated by the iCEcube2 Physical Implementation tools, after the routing phase.
5. Perform Static Timing Analysis using the iCEcube2 static timing analyzer.
6. Generate the device programming and configuration files from the iCEcube2 Physical Implementation tools.
7. Program your device using the device programming hardware provided by Lattice.

Tutorial

This chapter provides a brief introduction to the iCEcube2 design flow. The goal of this chapter is to familiarize the user with the fundamental steps needed to create a design project, synthesize and implement the design, generate the necessary device configuration files, and program the target device.

Creating a Project

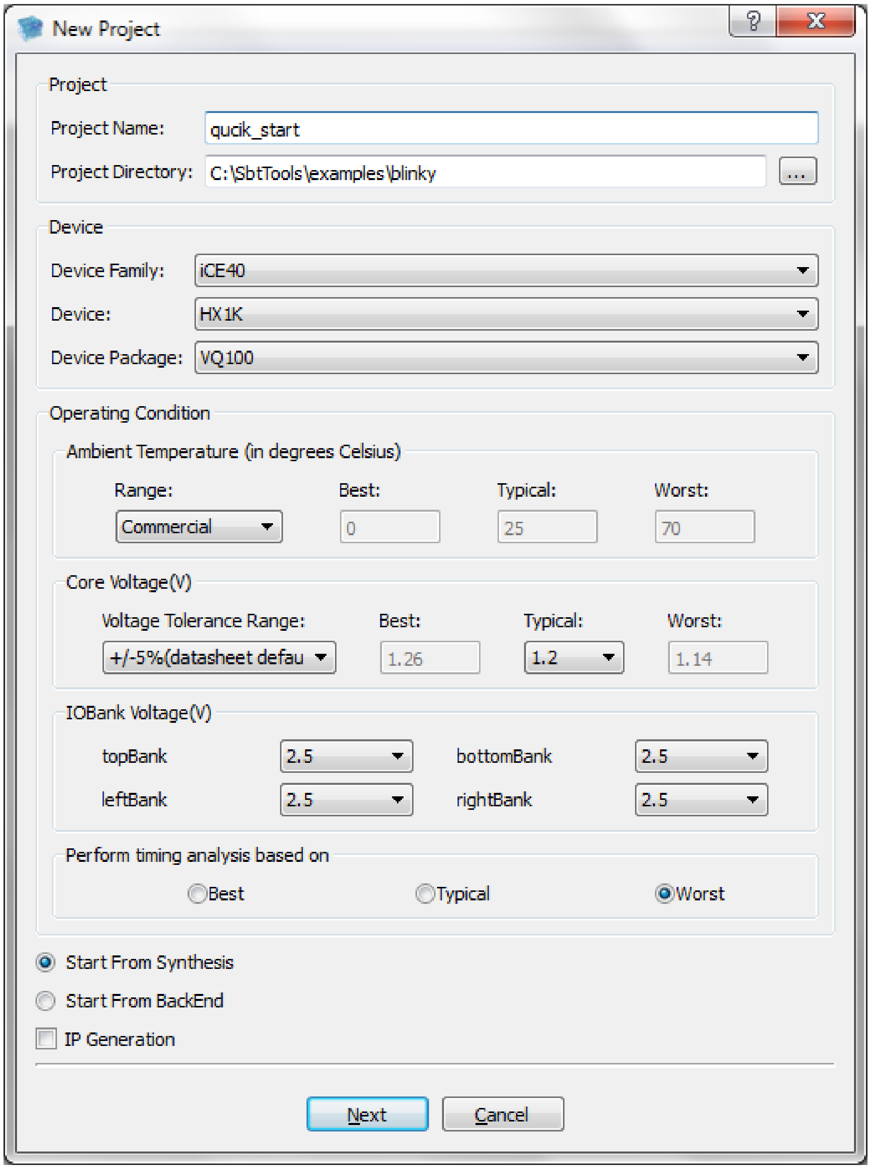
Starting the iCEcube2 software for the first time, you will see the following interface shown in Figure 2.



- Project
  - New Project
  - Open Project
  - Close Project
- Synthesis Tool
  - Add Synthesis Files
    - Design Files
    - Constraint Files
  - Launch Synthesis Tool
- P&R Flow
  - Select Implementation
  - Add P&R Files
  - Run All
  - Import P&R Input Files
  - Run Placer
  - Run Router
  - Generate Bitmap
  - IP Exporter
  - Output Files
    - Reports
    - Bitmap
    - Simulation Netlist
  - Device/Operating Condition
    - Device Info
      - DeviceFamily
      - Device
      - Device Package
      - Power Grade
    - Operating Condition
      - Core Voltage(V)
      - Temperature(C)

Figure 2: Create a New Project

The first step is to create a new design project and add the appropriate design files to your project. You can create a new project by either selecting **File>New Project** from the iCEcube2 menu, or by clicking the **Create a New Project** icon as seen in Figure 2. The New Project Wizard GUI is displayed in Figure 3.



New Project

Project

Project Name: qucik_start

Project Directory: C:\SbtTools\examples\blinky

Device

Device Family: iCE40

Device: HX1K

Device Package: VQ100

Operating Condition

Ambient Temperature (in degrees Celsius)

Range:	Best:	Typical:	Worst:
Commercial	0	25	70

Core Voltage(V)

Voltage Tolerance Range:	Best:	Typical:	Worst:
+/-5%(datasheet defau	1.26	1.2	1.14

IOBank Voltage(V)

topBank	2.5	bottomBank	2.5
leftBank	2.5	rightBank	2.5

Perform timing analysis based on

○ Best ○ Typical ● Worst

● Start From Synthesis

○ Start From BackEnd

☐ IP Generation

Figure 3: New Project Setup Form

This example is targeted for iCE40 family device. Follow the following steps to setup the project properties.

1. Project Name Field: Specify a project name (*quick_start*) in the Project Name field.

2. **Project Directory** Field: Specify any directory where you want to place the project directory in the Project Directory field.

3. **Device Family** Fields: This section allows you to specify the Lattice iCE device family you are targeting. For this example, change the Device Family to **iCE40**.

4. **Device** Fields**:** This section allows you to specify the Lattice device and package you are targeting. For this example, change the Device to **HX1K** and change the device package to the **VQ100**.

5. **Operating Condition** Fields: This section allows you to specify the operating conditions of the device which will be used for timing and power analysis. The **IO Bank Voltage** option shown in Figure 3 is not available for iCE65 family devices.

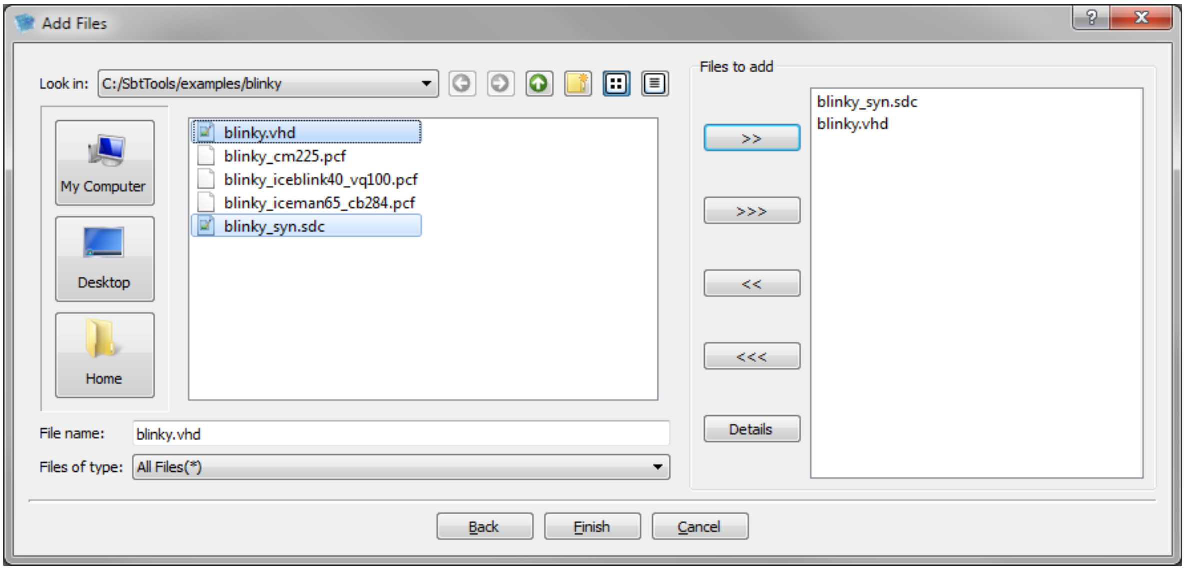
6. **Start From Synthesis:** This option allows you to start the flow from Synthesis using Synopsys Synplify Pro tool. For current example, select this option.

7. **Start From BackEnd:** This option allows the user to start from Post Synthesis flow.

8. Click Next to go to the Add Files dialog box shown in Figure 4. You will be prompted to create a new project directory. Click Yes.
9. In the Add Files dialog box, navigate to: <iCEcube2 installation directory>/examples/blinky
Highlight the following files:
blinky.vhd
blinky_syn.sdc*

Select each file and click **>>** to add the selected file, or click **>>>** to add all the files in the open directory (files can be removed using **<<** and **<<<**) to your project. Click **Finish** to create the project.

* The SDC file is a Synopsys constraint file, which contains timing constraint information.



Add Files

Look in: C:/SbtTools/examples/blinky

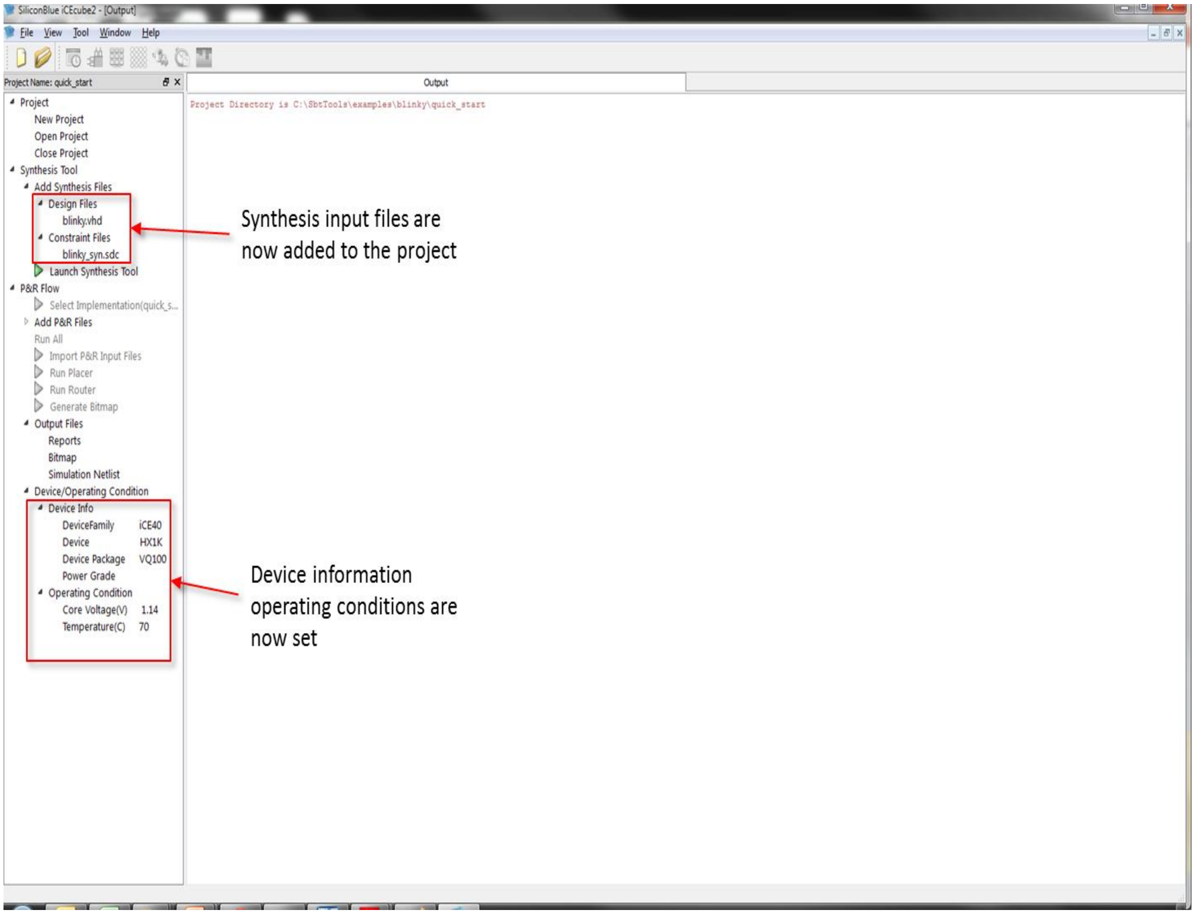
blinky.vhd
blinky_cm225.pcf
blinky_iceblink40_vq100.pcf
blinky_iceman65_cb284.pcf
blinky_syn.sdc

Files to add
blinky_syn.sdc
blinky.vhd

File name: blinky.vhd
Files of type: All Files(*)

Figure 4: New Project Wizard – Add Files dialog box

After successfully setting-up your project, you will return to the following iCEcube2 Project Navigator screen shown in Figure 5.



```
Project Directory is C:\SbtTools\examples\blinky\quick_start
```

DeviceFamily iCE40
Device HX1K
Device Package VQ100
Power Grade
Core Voltage(V) 1.14
Temperature(C) 70

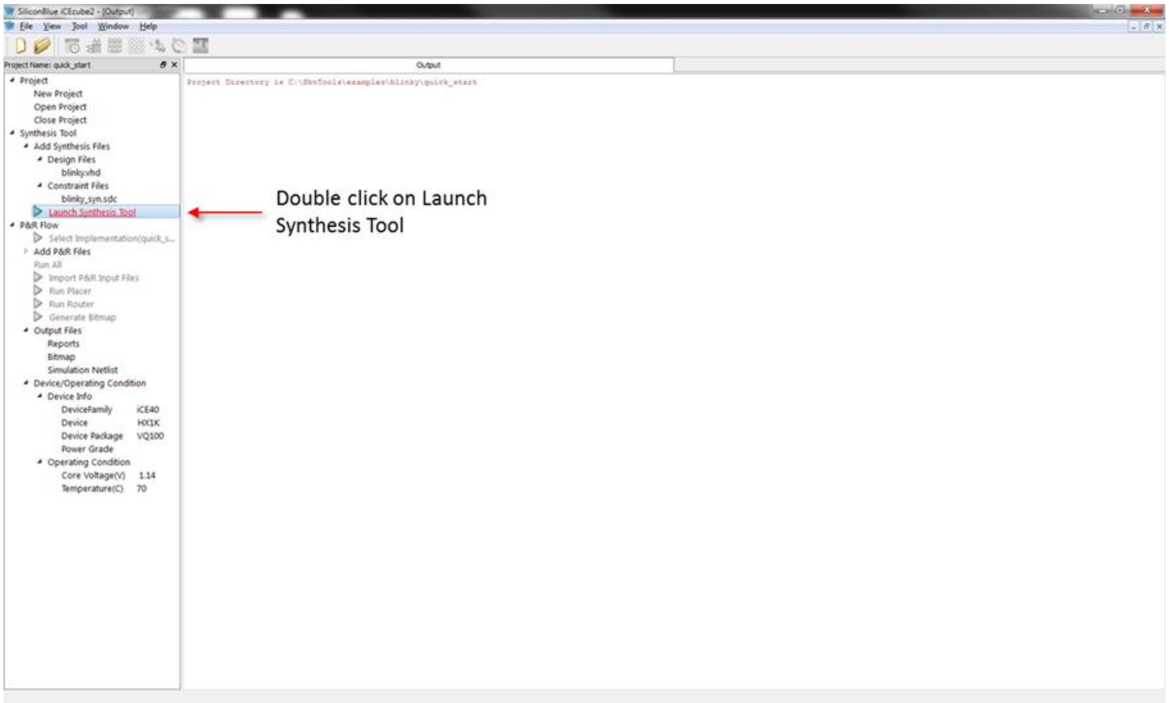
Synthesis input files are now added to the project

Device information operating conditions are now set

Figure 5: iCEcube2 Project Navigator View after Completing Project Set-up

Synthesizing the design

After a successful project setup, **Double-Click on the Synthesis Launch Synthesis Tool** icon in the project navigator window. See Figure 6. This will bring-up the Synopsys Synplify Pro synthesis tool's graphical user interface. See Figure 7.

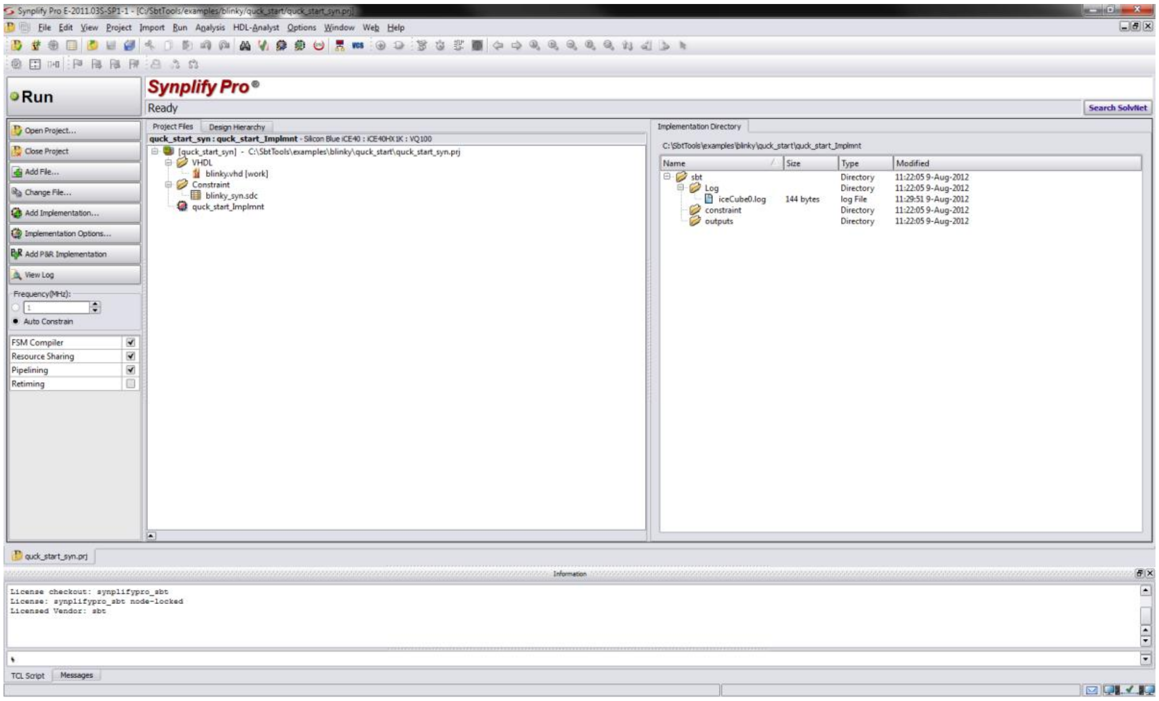


```
Project Directory is C:\SbtTools\examples\blinky\quick_start
```

DeviceFamily iCE40
Device HX1K
Device Package VQ100
Power Grade
Core Voltage(V) 1.14
Temperature(C) 70

Double click on Launch Synthesis Tool

Figure 6: Launch Synthesis Tool



Ready

quck_start_syn : quck_start_Implmnt - Silicon Blue iCE40 : iCE40HX1K : VQ100

[quck_start_syn] - C:\SbtTools\examples\blinky\quck_start\quck_start_syn.prj
blinky.vhd [work]
blinky_syn.sdc
quck_start_Implmnt

C:\SbtTools\examples\blinky\quck_start\quck_start_Implmnt

Name	Size	Type	Modified
sbt		Directory	11:22:05 9-Aug-2012
Log		Directory	11:22:05 9-Aug-2012
iceCube0.log	144 bytes	log File	11:29:51 9-Aug-2012
constraint		Directory	11:22:05 9-Aug-2012
outputs		Directory	11:22:05 9-Aug-2012

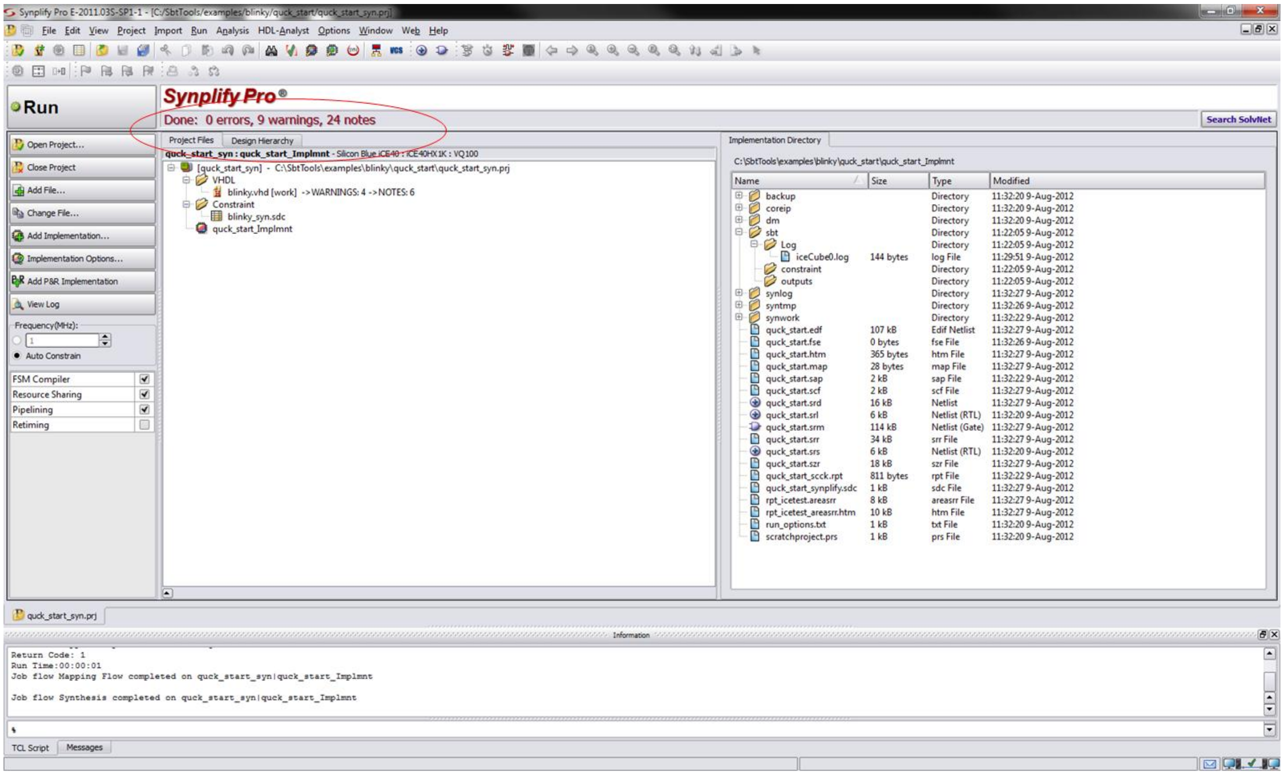
Frequency(MHz):
○ 1
● Auto Constrain

FSM Compiler	☑
Resource Sharing	☑
Pipelining	☑
Retiming	□

```
License checkout: synplifypro_sbt
License: synplifypro_sbt node-locked
Licensed Vendor: sbt
```

Figure 7: Synplify Pro Graphical User Interface

Hit the Run Button to synthesis your design. Once synthesis is complete, you will see a Done message. See Figure 8.



Done: 0 errors, 9 warnings, 24 notes

Frequency(MHz): ○ 1

● Auto Constrain

FSM Compiler	☑
Resource Sharing	☑
Pipelining	☑
Retiming	☐

quck_start_syn : quck_start_Implmnt - Silicon Blue iCE40 : iCE40HX1K : VQ100

```
[quck_start_syn] - C:\SbtTools\examples\blinky\quck_start\quck_start_syn.prj
  VHDL
    blinky.vhd [work] ->WARNINGS: 4 ->NOTES: 6
  Constraint
    blinky_syn.sdc
  quck_start_Implmnt
```

C:\SbtTools\examples\blinky\quck_start\quck_start_Implmnt

Name	Size	Type	Modified
backup		Directory	11:32:20 9-Aug-2012
coreip		Directory	11:32:20 9-Aug-2012
dm		Directory	11:32:20 9-Aug-2012
sbt		Directory	11:22:05 9-Aug-2012
Log		Directory	11:22:05 9-Aug-2012
iceCube0.log	144 bytes	log File	11:29:51 9-Aug-2012
constraint		Directory	11:22:05 9-Aug-2012
outputs		Directory	11:22:05 9-Aug-2012
synlog		Directory	11:32:27 9-Aug-2012
syntmp		Directory	11:32:26 9-Aug-2012
synwork		Directory	11:32:22 9-Aug-2012
quck_start.edf	107 kB	Edif Netlist	11:32:27 9-Aug-2012
quck_start.fse	0 bytes	fse File	11:32:26 9-Aug-2012
quck_start.htm	365 bytes	htm File	11:32:27 9-Aug-2012
quck_start.map	28 bytes	map File	11:32:27 9-Aug-2012
quck_start.sap	2 kB	sap File	11:32:22 9-Aug-2012
quck_start.scf	2 kB	scf File	11:32:27 9-Aug-2012
quck_start.srd	16 kB	Netlist	11:32:27 9-Aug-2012
quck_start.srl	6 kB	Netlist (RTL)	11:32:20 9-Aug-2012
quck_start.srm	114 kB	Netlist (Gate)	11:32:27 9-Aug-2012
quck_start.srr	34 kB	srr File	11:32:27 9-Aug-2012
quck_start.srs	6 kB	Netlist (RTL)	11:32:20 9-Aug-2012
quck_start.szr	18 kB	szr File	11:32:27 9-Aug-2012
quck_start_scck.rpt	811 bytes	rpt File	11:32:22 9-Aug-2012
quck_start_synplify.sdc	1 kB	sdc File	11:32:27 9-Aug-2012
rpt_icetest.areasrr	8 kB	areasrr File	11:32:27 9-Aug-2012
rpt_icetest_areasrr.htm	10 kB	htm File	11:32:27 9-Aug-2012
run_options.txt	1 kB	txt File	11:32:20 9-Aug-2012
scratchproject.prs	1 kB	prs File	11:32:20 9-Aug-2012

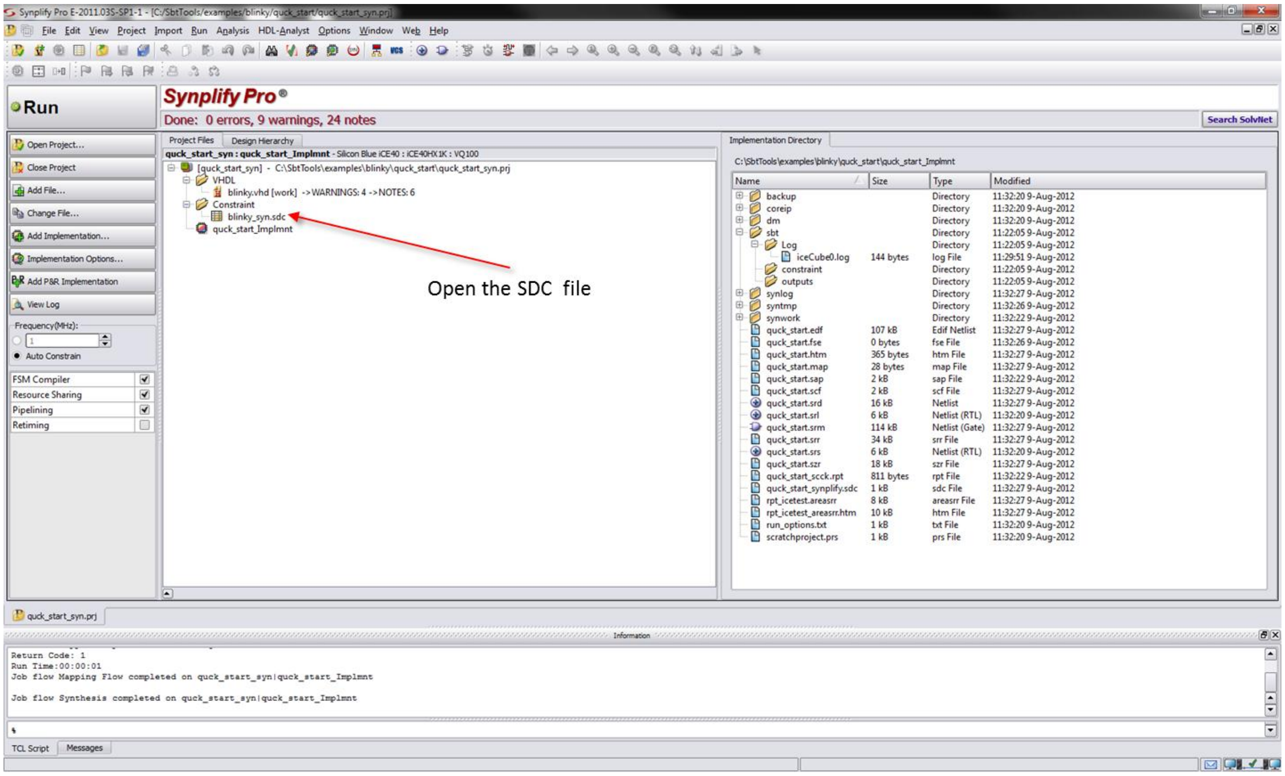
```
Return Code: 1
Run Time:00:00:01
Job flow Mapping Flow completed on quck_start_syn|quck_start_Implmnt

Job flow Synthesis completed on quck_start_syn|quck_start_Implmnt
```

```
%
```

Figure 8: Status showing synthesis has been completed

View Timing Constraints

Double Click on t**he** **blinky_syn.sdc** file under the Constraint folder. See Figure 9. It will open the timing constraints for the project shown in Figure 10.



Done: 0 errors, 9 warnings, 24 notes

○ 1

● Auto Constrain

FSM Compiler	☑
Resource Sharing	☑
Pipelining	☑
Retiming	□

```
quck_start_syn : quck_start_Implmnt - Silicon Blue iCE40 : iCE40HX1K : VQ100
[quck_start_syn] - C:\SbtTools\examples\blinky\quck_start\quck_start_syn.prj
    VHDL
        blinky.vhd [work] ->WARNINGS: 4 ->NOTES: 6
    Constraint
        blinky_syn.sdc
    quck_start_Implmnt
```

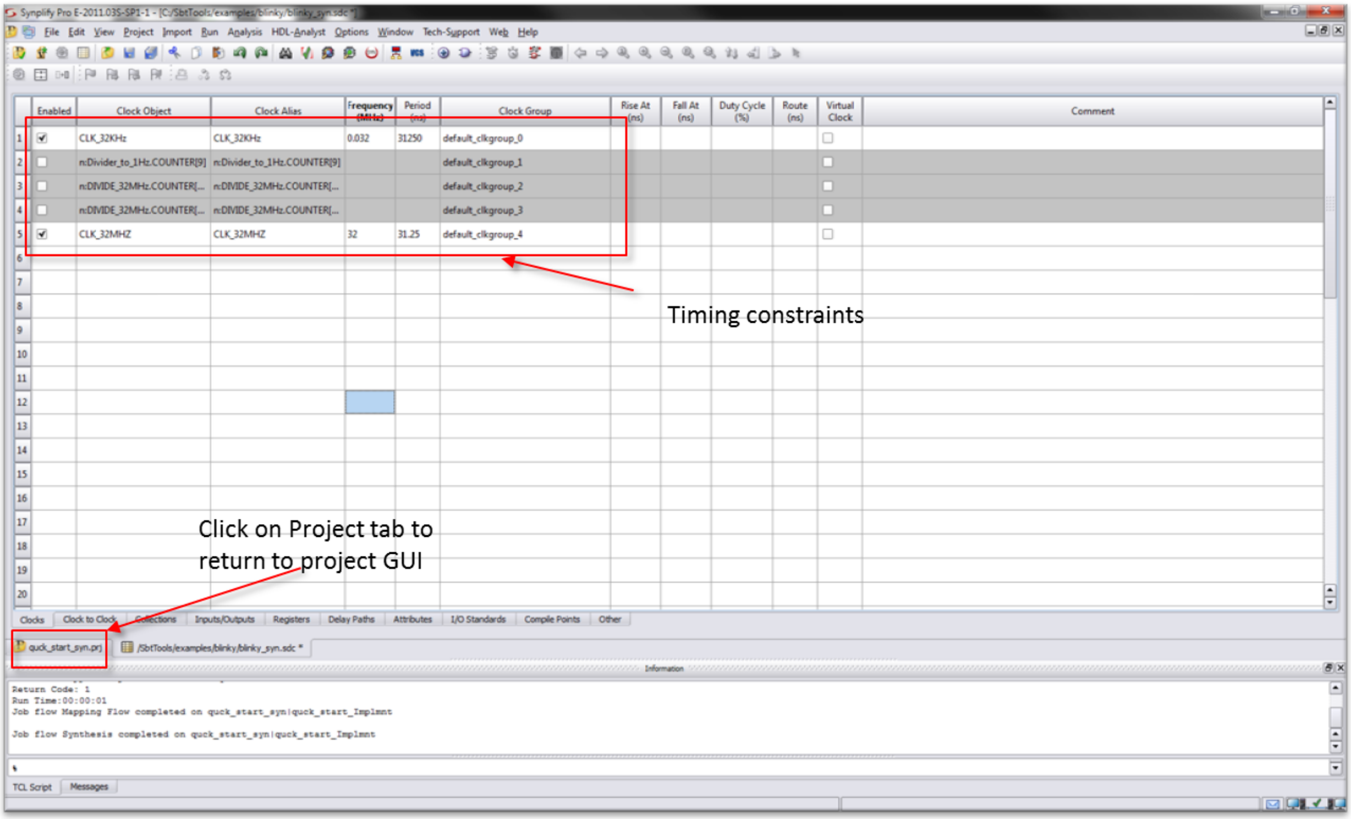
Open the SDC file

C:\SbtTools\examples\blinky\quck_start\quck_start_Implmnt

Name	Size	Type	Modified
backup		Directory	11:32:20 9-Aug-2012
coreip		Directory	11:32:20 9-Aug-2012
dm		Directory	11:32:20 9-Aug-2012
sbt		Directory	11:22:05 9-Aug-2012
Log		Directory	11:22:05 9-Aug-2012
iceCube0.log	144 bytes	log File	11:29:51 9-Aug-2012
constraint		Directory	11:22:05 9-Aug-2012
outputs		Directory	11:22:05 9-Aug-2012
synlog		Directory	11:32:27 9-Aug-2012
syntmp		Directory	11:32:26 9-Aug-2012
synwork		Directory	11:32:22 9-Aug-2012
quck_start.edf	107 kB	Edif Netlist	11:32:27 9-Aug-2012
quck_start.fse	0 bytes	fse File	11:32:26 9-Aug-2012
quck_start.htm	365 bytes	htm File	11:32:27 9-Aug-2012
quck_start.map	28 bytes	map File	11:32:27 9-Aug-2012
quck_start.sap	2 kB	sap File	11:32:22 9-Aug-2012
quck_start.scf	2 kB	scf File	11:32:27 9-Aug-2012
quck_start.srd	16 kB	Netlist	11:32:27 9-Aug-2012
quck_start.srl	6 kB	Netlist (RTL)	11:32:20 9-Aug-2012
quck_start.srm	114 kB	Netlist (Gate)	11:32:27 9-Aug-2012
quck_start.srr	34 kB	srr File	11:32:27 9-Aug-2012
quck_start.srs	6 kB	Netlist (RTL)	11:32:20 9-Aug-2012
quck_start.szr	18 kB	szr File	11:32:27 9-Aug-2012
quck_start_scck.rpt	811 bytes	rpt File	11:32:22 9-Aug-2012
quck_start_synplify.sdc	1 kB	sdc File	11:32:27 9-Aug-2012
rpt_icetest.areasrr	8 kB	areasrr File	11:32:27 9-Aug-2012
rpt_icetest_areasrr.htm	10 kB	htm File	11:32:27 9-Aug-2012
run_options.txt	1 kB	txt File	11:32:20 9-Aug-2012
scratchproject.prs	1 kB	prs File	11:32:20 9-Aug-2012

```
Return Code: 1
Run Time:00:00:01
Job flow Mapping Flow completed on quck_start_syn|quck_start_Implmnt

Job flow Synthesis completed on quck_start_syn|quck_start_Implmnt
```

Figure 9: Open the SDC file to View Timing Constraints



Synplify Pro E-2011.03S-SP1-1 - [C:/SbtTools/examples/blinky/blinky_syn.sdc *]
File Edit View Project Import Run Analysis HDL-Analyst Options Window Tech-Support Web Help
Clocks
Clock to Clock
Collections
Inputs/Outputs
Registers
Delay Paths
Attributes
I/O Standards
Compile Points
Other
quck_start_syn.prj
/SbtTools/examples/blinky/blinky_syn.sdc *
Information
TCL Script
Messages

	Enabled	Clock Object	Clock Alias	Frequency (MHz)	Period (ns)	Clock Group	Rise At (ns)	Fall At (ns)	Duty Cycle (%)	Route (ns)	Virtual Clock	Comment
1	☑	CLK_32KHz	CLK_32KHz	0.032	31250	default_clkgroup_0					□	
2	□	n:Divider_to_1Hz.COUNTER[9]	n:Divider_to_1Hz.COUNTER[9]			default_clkgroup_1					□	
3	□	n:DIVIDE_32MHz.COUNTER[...	n:DIVIDE_32MHz.COUNTER[...			default_clkgroup_2					□	
4	□	n:DIVIDE_32MHz.COUNTER[...	n:DIVIDE_32MHz.COUNTER[...			default_clkgroup_3					□	
5	☑	CLK_32MHZ	CLK_32MHZ	32	31.25	default_clkgroup_4					□	
6												
7												
8												
9												
10												
11												
12												
13												
14												
15												
16												
17												
18												
19												
20												

Timing constraints

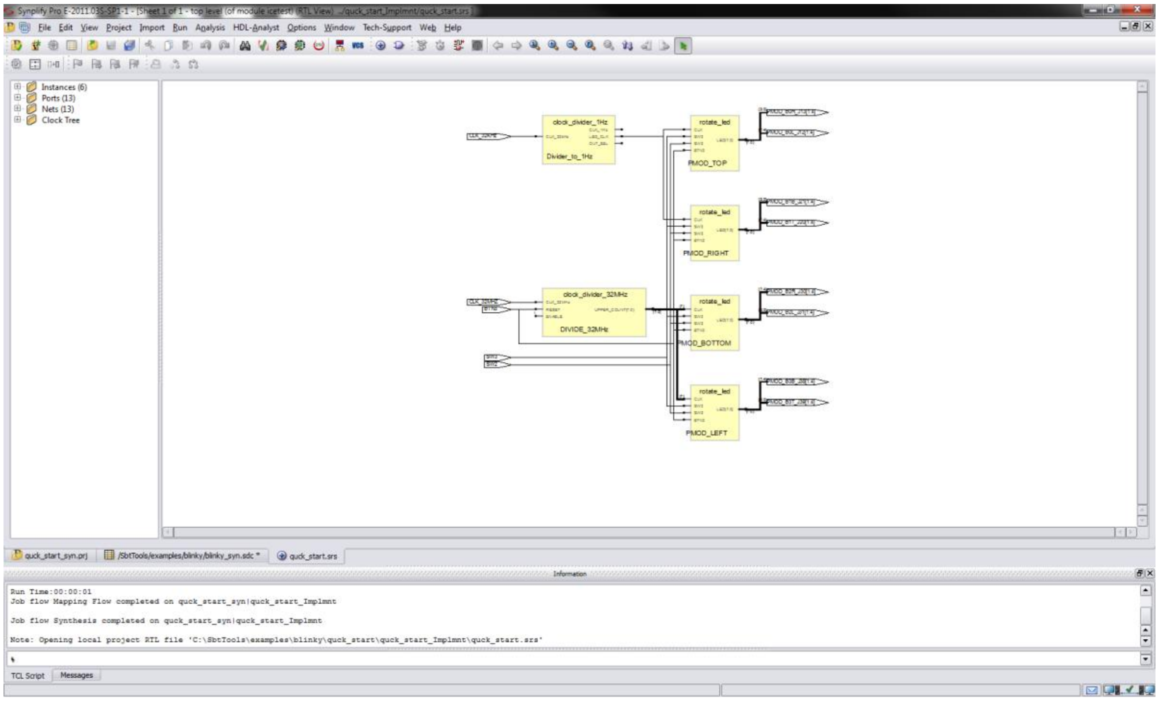
Click on Project tab to
return to project GUI

```
Return Code: 1
Run Time:00:00:01
Job flow Mapping Flow completed on quck_start_syn|quck_start_Implmnt

Job flow Synthesis completed on quck_start_syn|quck_start_Implmnt
```

Figure 10: View Timing Constraints

Viewing Hierarchical View of Synthesis Results

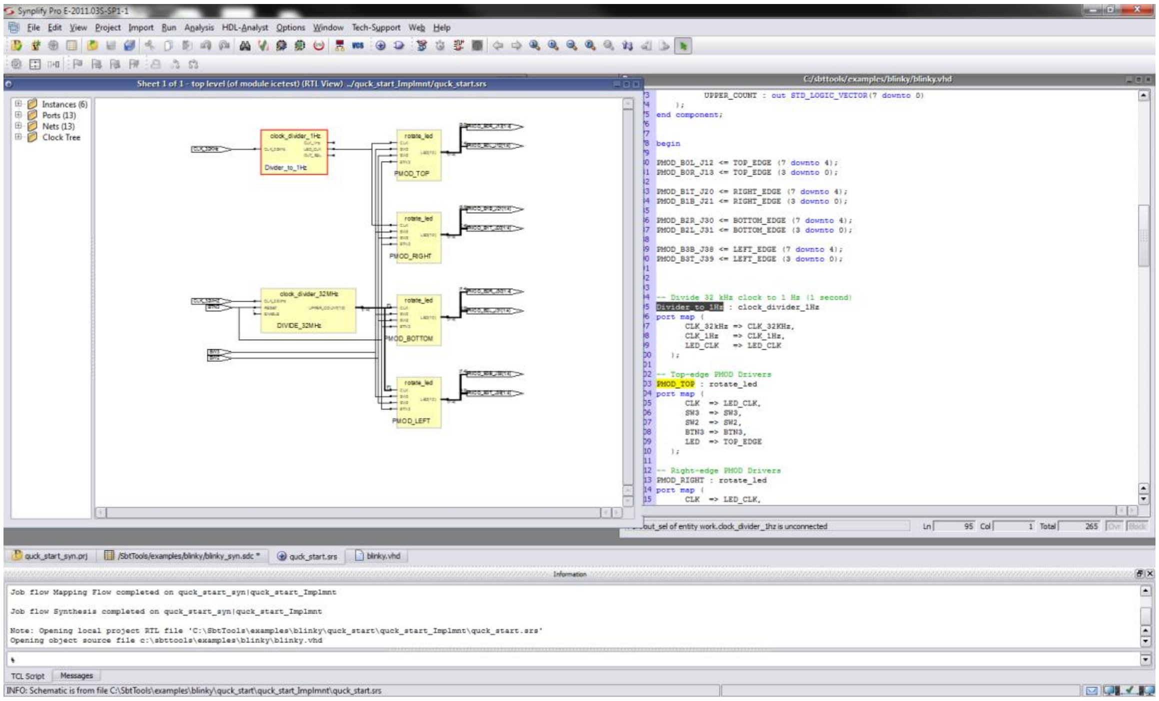
Under the **HDL-Analyst menu**, **Select RTL > Hierarchical View**. You will see a hierarchical RTL view of the design just synthesized. See Figure 11.



```
Run Time:00:00:01
Job flow Mapping Flow completed on quck_start_syn|quck_start_Implmnt

Job flow Synthesis completed on quck_start_syn|quck_start_Implmnt

Note: Opening local project RTL file 'C:\SbtTools\examples\blinky\quck_start\quck_start_Implmnt\quck_start.srs'
```

Figure 11: Hierarchical RTL View in HDL Analyst

If you double click on one of the blocks, it will take you to the RTL for that block. See Figure 12.


Synplify Pro E-2011.03S-SP1-1
File Edit View Project Import Run Analysis HDL-Analyst Options Window Tech-Support Web Help
Sheet 1 of 1 - top level (of module icetest) (RTL View) ../quck_start_Implmnt/quck_start.srs
Instances (6)
Ports (13)
Nets (13)
Clock Tree
clock_divider_1Hz
Divider_to_1Hz
rotate_led
PMOD_TOP
PMOD_RIGHT
clock_divider_32MHz
DIVIDE_32MHz
PMOD_BOTTOM
PMOD_LEFT
C:/sbttools/examples/blinky/blinky.vhd
quck_start_syn.prj
/SbtTools/examples/blinky/blinky_syn.sdc *
quck_start.srs
blinky.vhd
Information
TCL Script
Messages

```
        UPPER_COUNT : out STD_LOGIC_VECTOR(7 downto 0)
    );
end component;


begin

PMOD_B0L_J12 <= TOP_EDGE (7 downto 4);
PMOD_B0R_J13 <= TOP_EDGE (3 downto 0);

PMOD_B1T_J20 <= RIGHT_EDGE (7 downto 4);
PMOD_B1B_J21 <= RIGHT_EDGE (3 downto 0);

PMOD_B2R_J30 <= BOTTOM_EDGE (7 downto 4);
PMOD_B2L_J31 <= BOTTOM_EDGE (3 downto 0);

PMOD_B3B_J38 <= LEFT_EDGE (7 downto 4);
PMOD_B3T_J39 <= LEFT_EDGE (3 downto 0);



-- Divide 32 kHz clock to 1 Hz (1 second)
Divider_to_1Hz : clock_divider_1Hz
port map (
      CLK_32kHz => CLK_32KHz,
      CLK_1Hz   => CLK_1Hz,
      LED_CLK   => LED_CLK
   );

-- Top-edge PMOD Drivers
PMOD_TOP : rotate_led
port map (
      CLK  => LED_CLK,
      SW3  => SW3,
      SW2  => SW2,
      BTN3 => BTN3,
      LED  => TOP_EDGE
   );

-- Right-edge PMOD Drivers
PMOD_RIGHT : rotate_led
port map (
      CLK  => LED_CLK,
```

```
...out_sel of entity work.clock_divider_1hz is unconnected    Ln 95  Col 1  Total 265
```

```
Job flow Mapping Flow completed on quck_start_syn|quck_start_Implmnt

Job flow Synthesis completed on quck_start_syn|quck_start_Implmnt

Note: Opening local project RTL file 'C:\SbtTools\examples\blinky\quck_start\quck_start_Implmnt\quck_start.srs'
Opening object source file c:\sbttools\examples\blinky\blinky.vhd
```

```
INFO: Schematic is from file C:\SbtTools\examples\blinky\quck_start\quck_start_Implmnt\quck_start.srs
```

Figure 12: Double-clicking on a block will reveal its HDL code in HDL Analyst

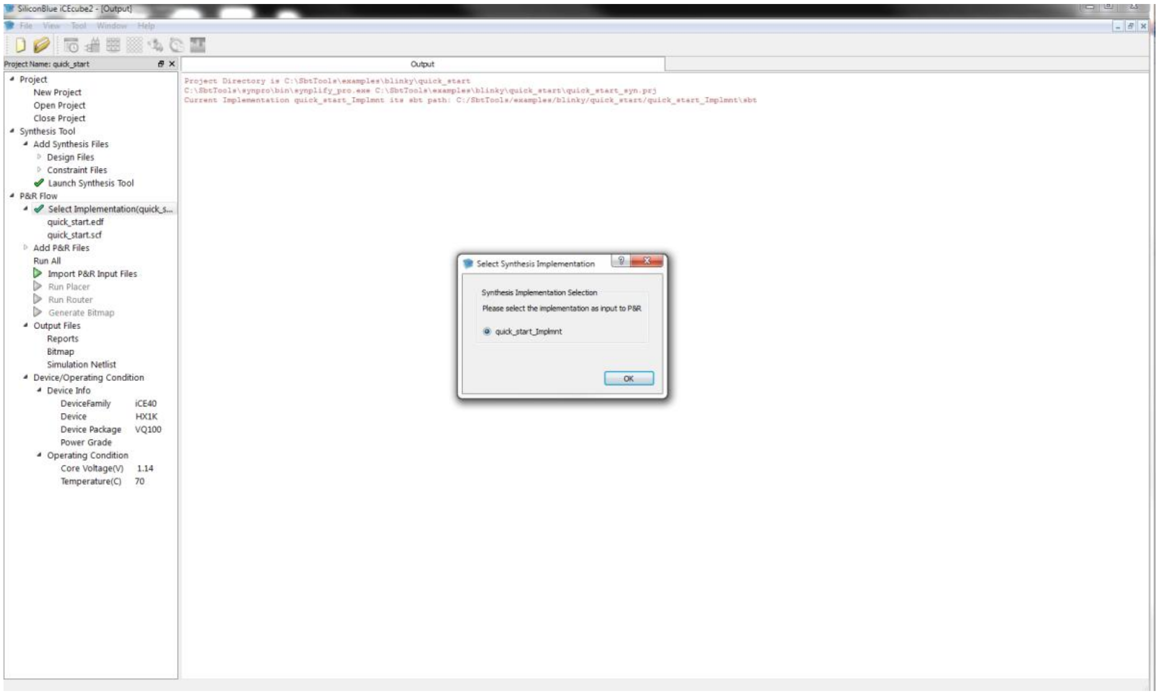
P&R Flow

This section goes through the post synthesis physical implementation flow.

Select Implementation

In order to ensure that the synthesized design can be successfully imported into iCEcube2, **exit** the Synplify Pro GUI.

Return to the iCEcube2 Navigator and **Double-click** on **Select Implementation**. See Figure 13. This will tell iCEcube2 which synthesis implementation to process for place and route. If you have different synthesis implementations, you will be able to select the synthesis implementation you wish to place and route. Since we only have one implementation, **select OK** when the Select Synthesis Implementation dialog box appears.



Project Name: quick_start

DeviceFamily	iCE40
Device	HX1K
Device Package	VQ100
Power Grade	
Core Voltage(V)	1.14
Temperature(C)	70

```
Project Directory is C:\SbtTools\examples\blinky\quick_start
C:\SbtTools\synpro\bin\synplify_pro.exe C:\SbtTools\examples\blinky\quick_start\quick_start_syn.prj
Current Implementation quick_start_Implmnt its sbt path: C:/SbtTools/examples/blinky/quick_start/quick_start_Implmnt\sbt
```

Synthesis Implementation Selection

Please select the implementation as input to P&R

● quick_start_Implmnt

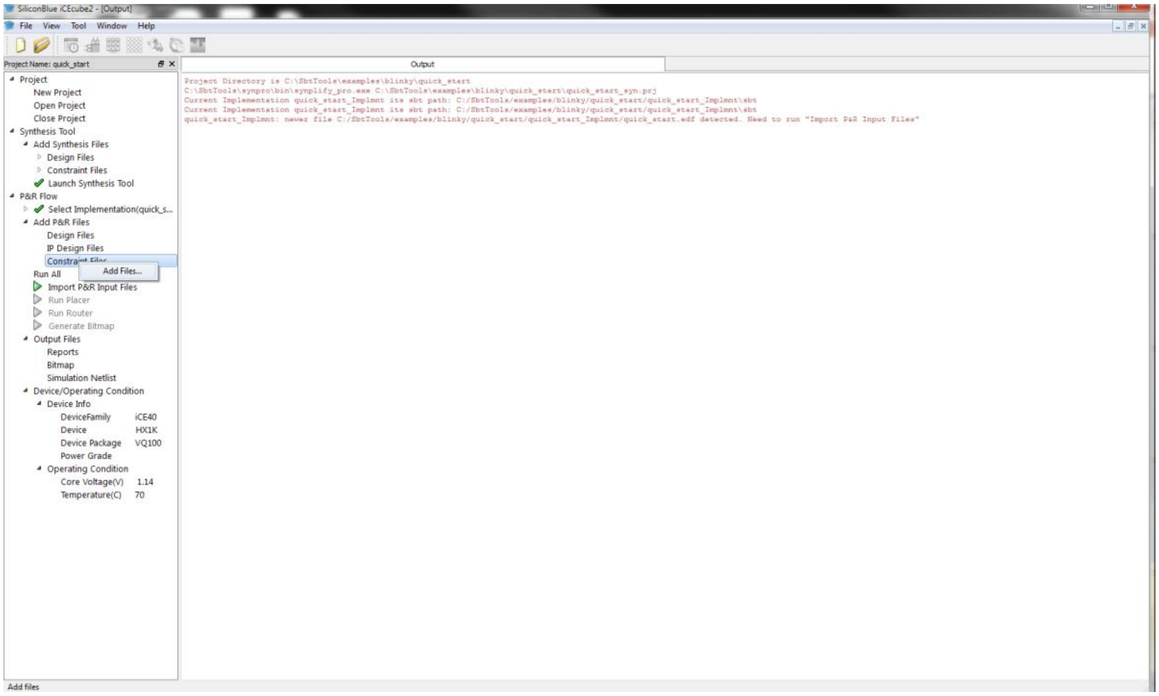
Figure 13: Select Synthesis Implementation

Importing Physical Constraints

Physical constraints such as pin assignments are stored in a .PCF file (Physical Constraint File). Add the .PCF file to your project.

In the iCEcube2 Project Navigator**, Right Click** on **Specify Additional Files**. Select **Add Files…** See Figure 14.

Note: For information on importing physical constraints from iCEcube to iCEcube2, please refer to the **Importing Physical Constraints from iCEcube to iCEcube2** section in the iCEcube2 software user guide.

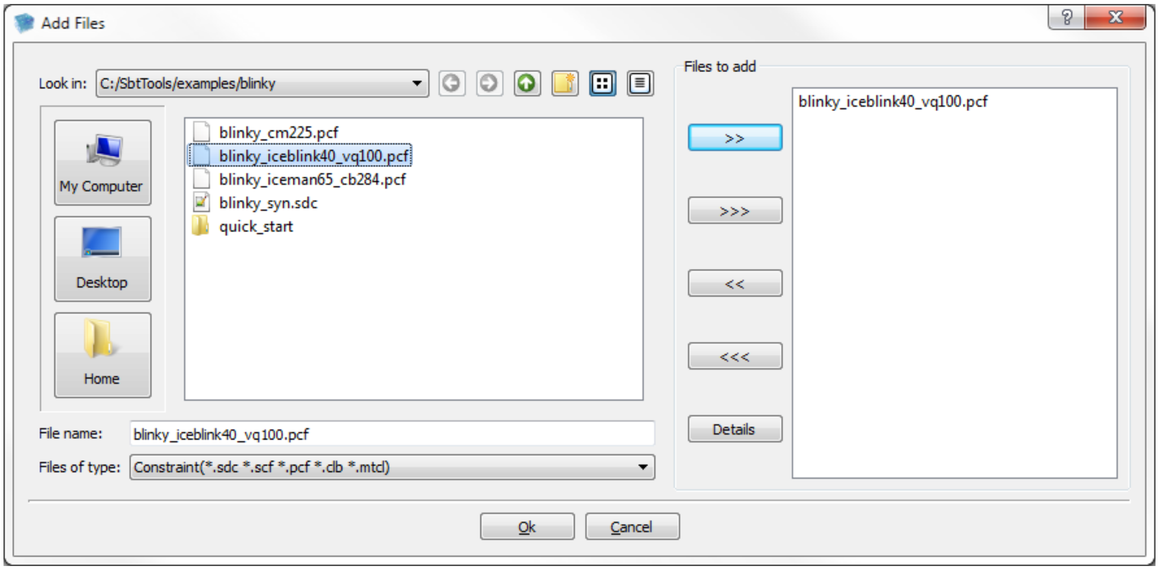


```
Project Directory is C:\SbtTools\examples\blinky\quick_start
C:\SbtTools\synpro\bin\synplify_pro.exe C:\SbtTools\examples\blinky\quick_start\quick_start_syn.prj
Current Implementation quick_start_Implmnt its sbt path: C:/SbtTools/examples/blinky/quick_start/quick_start_Implmnt\sbt
Current Implementation quick_start_Implmnt its sbt path: C:/SbtTools/examples/blinky/quick_start/quick_start_Implmnt\sbt
quick_start_Implmnt: newer file C:/SbtTools/examples/blinky/quick_start/quick_start_Implmnt/quick_start.edf detected. Need to run "Import P&R Input Files"
```

DeviceFamily iCE40
Device HX1K
Device Package VQ100
Power Grade
Core Voltage(V) 1.14
Temperature(C) 70

Figure 14: Specify Additional Files for Place and Route

Navigate to the <iCEcube2 Installation Directory>/examples/blinky and **Add blinky.pcf** file. See Figure 15.



Look in: C:/SbtTools/examples/blinky

File name: blinky_iceblink40_vq100.pcf

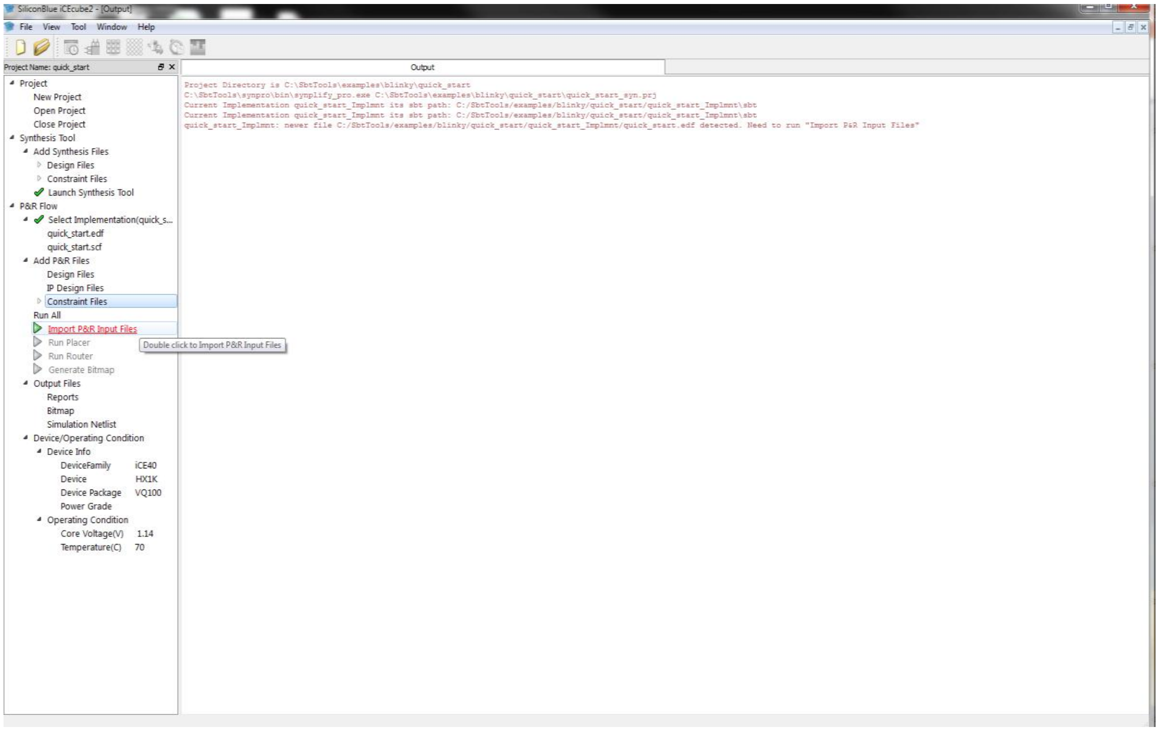
Files of type: Constraint(*.sdc *.scf *.pcf *.db *.mtcl)

Files to add: blinky_iceblink40_vq100.pcf

Figure 15: Add .pcf file

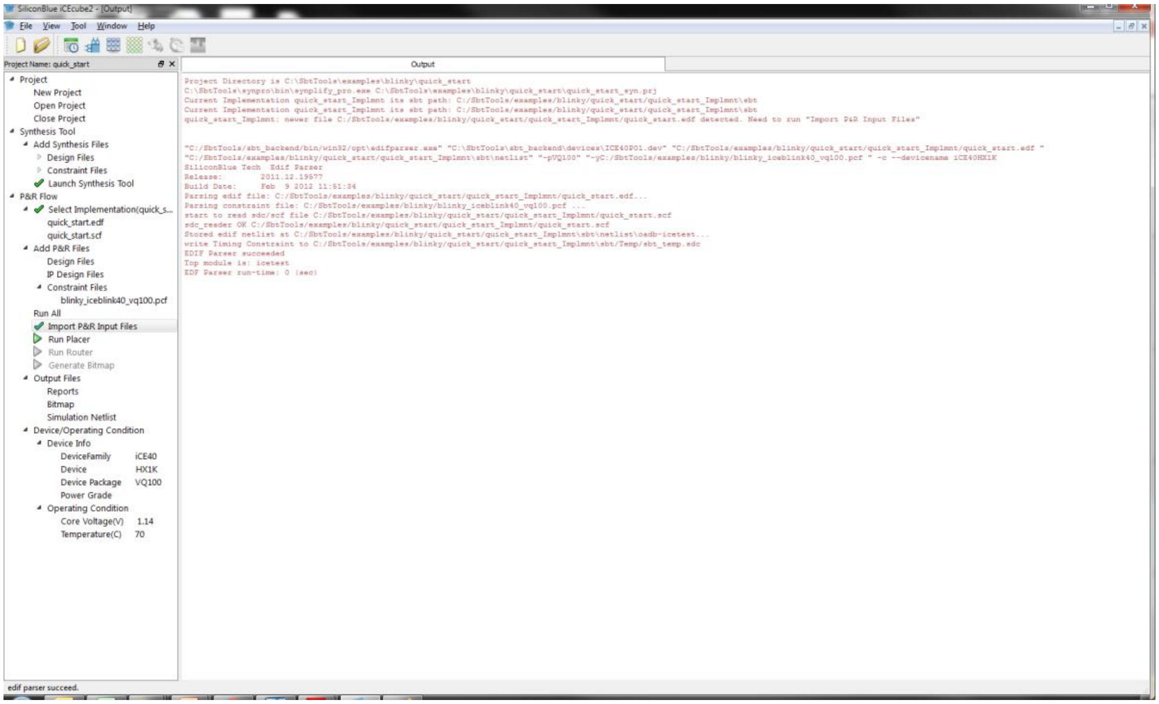
Import Place & Route Input Files

The next step is to import the files for Place and Route. **Double-click** on **Import P&R Input Files** in the Project Navigator. See Figure 16. Once completed you will see a green check next to Import P&R Input Files. See Figure 17.



```
Project Directory is C:\SbtTools\examples\blinky\quick_start
C:\SbtTools\synpro\bin\synplify_pro.exe C:\SbtTools\examples\blinky\quick_start\quick_start_syn.prj
Current Implementation quick_start_Implmnt its sbt path: C:/SbtTools/examples/blinky/quick_start/quick_start_Implmnt\sbt
Current Implementation quick_start_Implmnt its sbt path: C:/SbtTools/examples/blinky/quick_start/quick_start_Implmnt\sbt
quick_start_Implmnt: newer file C:/SbtTools/examples/blinky/quick_start/quick_start_Implmnt/quick_start.edf detected. Need to run "Import P&R Input Files"
```

Figure 16: Import P&R Input files



```
Project Directory is C:\SbtTools\examples\blinky\quick_start
C:\SbtTools\synpro\bin\synplify_pro.exe C:\SbtTools\examples\blinky\quick_start\quick_start_syn.prj
Current Implementation quick_start_Implmnt its sbt path: C:/SbtTools/examples/blinky/quick_start/quick_start_Implmnt\sbt
Current Implementation quick_start_Implmnt its sbt path: C:/SbtTools/examples/blinky/quick_start/quick_start_Implmnt\sbt
quick_start_Implmnt: newer file C:/SbtTools/examples/blinky/quick_start/quick_start_Implmnt/quick_start.edf detected. Need to run "Import P&R Input Files"


"C:/SbtTools/sbt_backend/bin/win32/opt\edifparser.exe" "C:\SbtTools\sbt_backend\devices\ICE40P01.dev" "C:/SbtTools/examples/blinky/quick_start/quick_start_Implmnt/quick_start.edf "
"C:/SbtTools/examples/blinky/quick_start/quick_start_Implmnt\sbt\netlist" "-pVQ100" "-yC:/SbtTools/examples/blinky/blinky_iceblink40_vq100.pcf " -c --devicename iCE40HX1K
SiliconBlue Tech  Edif Parser
Release:        2011.12.19577
Build Date:     Feb  9 2012 11:51:34
Parsing edif file: C:/SbtTools/examples/blinky/quick_start/quick_start_Implmnt/quick_start.edf...
Parsing constraint file: C:/SbtTools/examples/blinky/blinky_iceblink40_vq100.pcf ...
start to read sdc/scf file C:/SbtTools/examples/blinky/quick_start/quick_start_Implmnt/quick_start.scf
sdc_reader OK C:/SbtTools/examples/blinky/quick_start/quick_start_Implmnt/quick_start.scf
Stored edif netlist at C:/SbtTools/examples/blinky/quick_start/quick_start_Implmnt\sbt\netlist\oadb-icetest...
write Timing Constraint to C:/SbtTools/examples/blinky/quick_start/quick_start_Implmnt\sbt/Temp/sbt_temp.sdc
EDIF Parser succeeded
Top module is: icetest
EDF Parser run-time: 0 (sec)
```

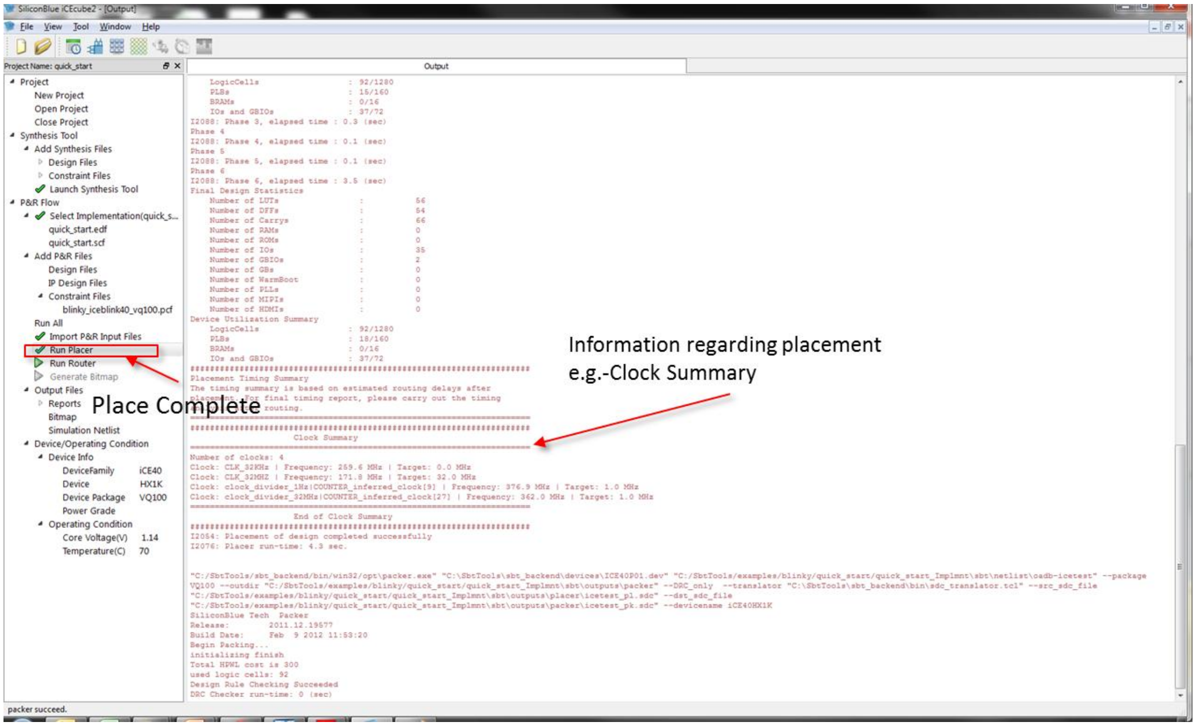
edif parser succeed.

Figure 17: Successful Import of P&R Input Files

Place the Design

Double-click on **Run Placer**

Once placement is complete, a green check will appear and the Output window will show information about the placement of the design. See Figure 18.



Place Complete

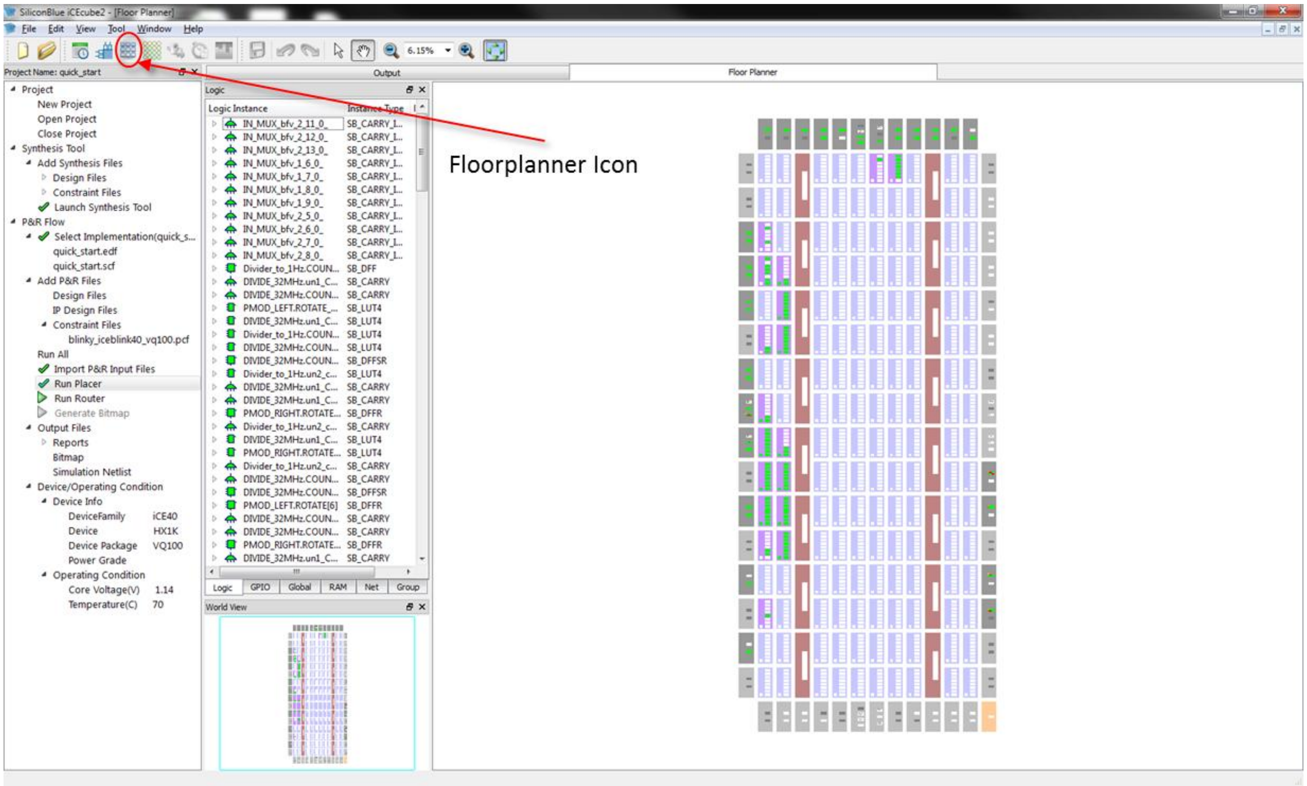
Information regarding placement e.g.-Clock Summary

```
    LogicCells                   : 92/1280
    PLBs                         : 15/160
    BRAMs                        : 0/16
    IOs and GBIOs                : 37/72
I2088: Phase 3, elapsed time : 0.3 (sec)
Phase 4
I2088: Phase 4, elapsed time : 0.1 (sec)
Phase 5
I2088: Phase 5, elapsed time : 0.1 (sec)
Phase 6
I2088: Phase 6, elapsed time : 3.5 (sec)
Final Design Statistics
    Number of LUTs                 :            56
    Number of DFFs                 :            54
    Number of Carrys               :            66
    Number of RAMs                 :            0
    Number of ROMs                 :            0
    Number of IOs                  :            35
    Number of GBIOs                :            2
    Number of GBs                  :            0
    Number of WarmBoot             :            0
    Number of PLLs                 :            0
    Number of MIPIs                :            0
    Number of HDMIs                :            0
Device Utilization Summary
    LogicCells                   : 92/1280
    PLBs                         : 18/160
    BRAMs                        : 0/16
    IOs and GBIOs                : 37/72
##########################################################################
Placement Timing Summary
The timing summary is based on estimated routing delays after
placement. For final timing report, please carry out the timing
analysis after routing.
==========================================================================
##########################################################################
                     Clock Summary
==========================================================================
Number of clocks: 4
Clock: CLK_32KHz | Frequency: 259.6 MHz | Target: 0.0 MHz
Clock: CLK_32MHZ | Frequency: 171.8 MHz | Target: 32.0 MHz
Clock: clock_divider_1Hz|COUNTER_inferred_clock[9] | Frequency: 376.9 MHz | Target: 1.0 MHz
Clock: clock_divider_32MHz|COUNTER_inferred_clock[27] | Frequency: 362.0 MHz | Target: 1.0 MHz
==========================================================================
                     End of Clock Summary
##########################################################################
I2054: Placement of design completed successfully
I2076: Placer run-time: 4.3 sec.


"C:/SbtTools/sbt_backend/bin/win32/opt\packer.exe" "C:\SbtTools\sbt_backend\devices\ICE40P01.dev" "C:/SbtTools/examples/blinky/quick_start/quick_start_Implmnt\sbt\netlist\oadb-icetest" --package
VQ100 --outdir "C:/SbtTools/examples/blinky/quick_start/quick_start_Implmnt\sbt\outputs\packer" --DRC_only  --translator "C:\SbtTools\sbt_backend\bin\sdc_translator.tcl" --src_sdc_file
"C:/SbtTools/examples/blinky/quick_start/quick_start_Implmnt\sbt\outputs\placer\icetest_pl.sdc" --dst_sdc_file
"C:/SbtTools/examples/blinky/quick_start/quick_start_Implmnt\sbt\outputs\packer\icetest_pk.sdc" --devicename iCE40HX1K
SiliconBlue Tech  Packer
Release:        2011.12.19577
Build Date:     Feb  9 2012 11:53:20
Begin Packing...
initializing finish
Total HPWL cost is 300
used logic cells: 92
Design Rule Checking Succeeded
DRC Checker run-time: 0 (sec)
```

Figure 18: Place complete

View Floorplanner

At this point, since placement has been completed, you can view the placement of the design by opening the Floorplanner. You can open the Floorplanner by going to the menu and selecting **Tool > Floorplanner** or you can also select the Floorplanner Icon. See Figure 19.



Device Info	
DeviceFamily	iCE40
Device	HX1K
Device Package	VQ100
Power Grade	

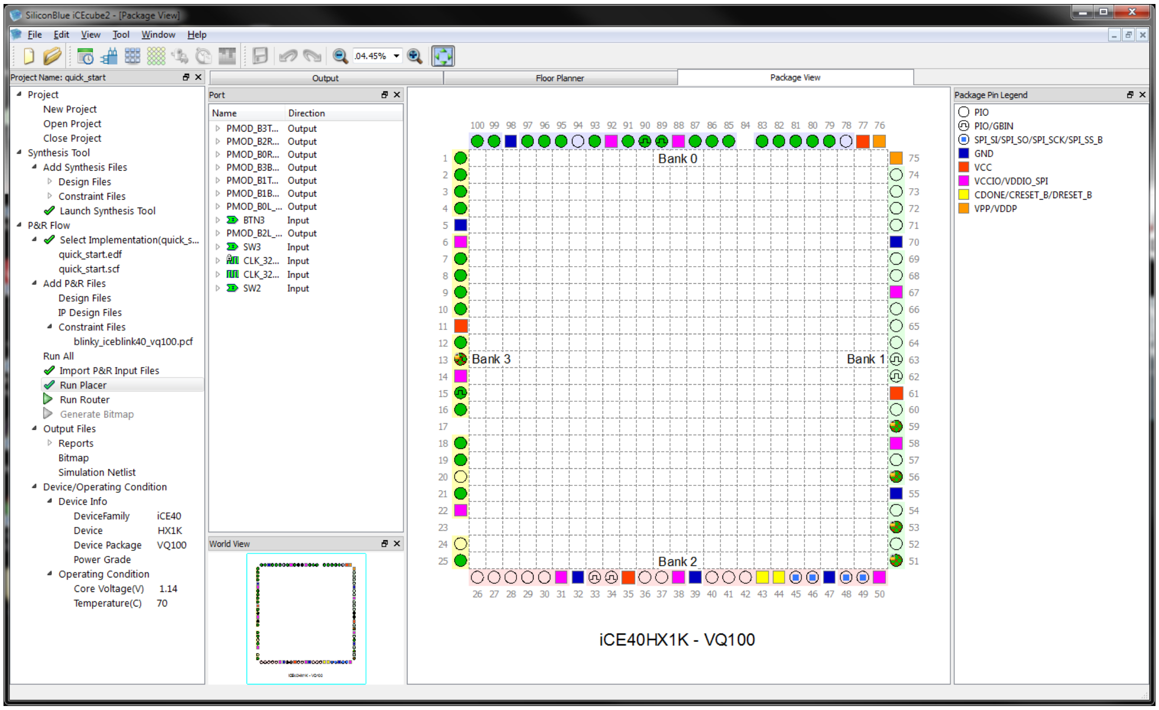
Operating Condition	
Core Voltage(V)	1.14
Temperature(C)	70

Logic Instance	Instance Type
IN_MUX_bfv_2_11_0_	SB_CARRY_L...
IN_MUX_bfv_2_12_0_	SB_CARRY_L...
IN_MUX_bfv_2_13_0_	SB_CARRY_L...
IN_MUX_bfv_1_6_0_	SB_CARRY_L...
IN_MUX_bfv_1_7_0_	SB_CARRY_L...
IN_MUX_bfv_1_8_0_	SB_CARRY_L...
IN_MUX_bfv_1_9_0_	SB_CARRY_L...
IN_MUX_bfv_2_5_0_	SB_CARRY_L...
IN_MUX_bfv_2_6_0_	SB_CARRY_L...
IN_MUX_bfv_2_7_0_	SB_CARRY_L...
IN_MUX_bfv_2_8_0_	SB_CARRY_L...
Divider_to_1Hz.COUN...	SB_DFF
DIVIDE_32MHz.un1_C...	SB_CARRY
DIVIDE_32MHz.COUN...	SB_CARRY
PMOD_LEFT.ROTATE_...	SB_LUT4
DIVIDE_32MHz.un1_C...	SB_LUT4
Divider_to_1Hz.COUN...	SB_LUT4
DIVIDE_32MHz.COUN...	SB_LUT4
DIVIDE_32MHz.COUN...	SB_DFFSR
Divider_to_1Hz.un2_c...	SB_LUT4
DIVIDE_32MHz.un1_C...	SB_CARRY
DIVIDE_32MHz.un1_C...	SB_CARRY
PMOD_RIGHT.ROTATE...	SB_DFFR
Divider_to_1Hz.un2_c...	SB_CARRY
DIVIDE_32MHz.un1_C...	SB_LUT4
PMOD_RIGHT.ROTATE...	SB_LUT4
Divider_to_1Hz.un2_c...	SB_CARRY
DIVIDE_32MHz.COUN...	SB_CARRY
DIVIDE_32MHz.COUN...	SB_DFFSR
PMOD_LEFT.ROTATE[6]	SB_DFFR
DIVIDE_32MHz.COUN...	SB_CARRY
DIVIDE_32MHz.COUN...	SB_CARRY
PMOD_RIGHT.ROTATE...	SB_DFFR
DIVIDE_32MHz.un1_C...	SB_CARRY

Figure 19: Floorplanner

View the Package Viewer

You can also see how pins were placed for your design by selecting the Package Viewer. You can select the package viewer by going to the menu and selecting **Tool >> Package Viewer** or you can also select the Package Vierwer Icon. See Figure 20.



SiliconBlue iCEcube2 - [Package View]

Project Name: quick_start

```
Project
    New Project
    Open Project
    Close Project
Synthesis Tool
    Add Synthesis Files
        Design Files
        Constraint Files
    ✓ Launch Synthesis Tool
P&R Flow
    ✓ Select Implementation(quick_s...
        quick_start.edf
        quick_start.scf
    Add P&R Files
        Design Files
        IP Design Files
        Constraint Files
            blinky_iceblink40_vq100.pcf
    Run All
    ✓ Import P&R Input Files
    ✓ Run Placer
    Run Router
    Generate Bitmap
    Output Files
        Reports
        Bitmap
        Simulation Netlist
    Device/Operating Condition
        Device Info
            DeviceFamily      iCE40
            Device            HX1K
            Device Package    VQ100
            Power Grade
        Operating Condition
            Core Voltage(V)   1.14
            Temperature(C)    70
```

Port

Name	Direction
PMOD_B3T...	Output
PMOD_B2R...	Output
PMOD_B0R...	Output
PMOD_B3B...	Output
PMOD_B1T...	Output
PMOD_B1B...	Output
PMOD_B0L_...	Output
BTN3	Input
PMOD_B2L_...	Output
SW3	Input
CLK_32...	Input
CLK_32...	Input
SW2	Input

Package Pin Legend

- ○ PIO
- ○ PIO/GBIN
- ◉ SPI_SI/SPI_SO/SPI_SCK/SPI_SS_B
- 🟦 GND
- 🟥 VCC
- 🟪 VCCIO/VDDIO_SPI
- 🟨 CDONE/CRESET_B/DRESET_B
- 🟧 VPP/VDDP

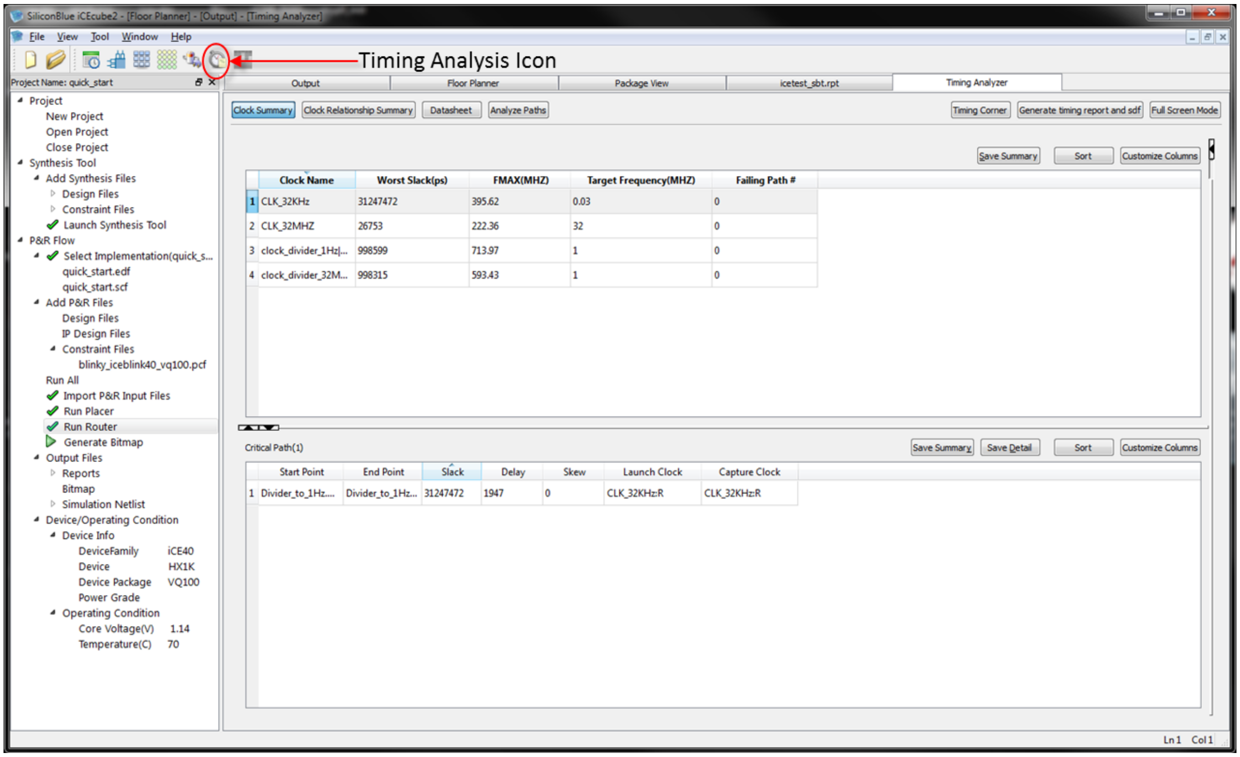
Figure 20: Package Viewer

Route the Design

Double-click on Route in the project navigation window. Place and Route have been separated into different steps as to allow you to re-route the design after making placement modifications in the floorplanner without having to re-run the placer.

Perform Static Timing Analysis

Now that you have routed the design, you can perform timing analysis to check to see if the design is meets your timing requirements. To launch the timing analyzer, go to the menu and **select Tool > Timing Analysis**. You can also select the Timing Analysis Icon. See Figure 21.



DeviceFamily	iCE40
Device	HX1K
Device Package	VQ100
Power Grade	
Core Voltage(V)	1.14
Temperature(C)	70

	Clock Name	Worst Slack(ps)	FMAX(MHZ)	Target Frequency(MHZ)	Failing Path #
1	CLK_32KHz	31247472	395.62	0.03	0
2	CLK_32MHZ	26753	222.36	32	0
3	clock_divider_1Hz|...	998599	713.97	1	0
4	clock_divider_32M...	998315	593.43	1	0

Critical Path(1)

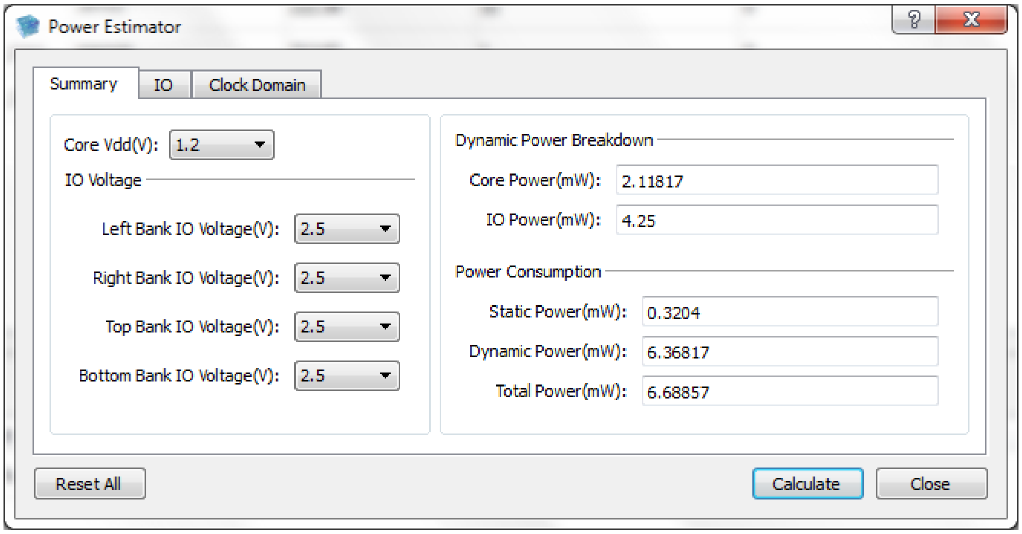
	Start Point	End Point	Slack	Delay	Skew	Launch Clock	Capture Clock
1	Divider_to_1Hz....	Divider_to_1Hz...	31247472	1947	0	CLK_32KHz:R	CLK_32KHz:R

Figure 21: Timing Analysis Summary

You can see from the timing analysis that our 32 KHz design is running at over 395.62 MHz and our 32 MHz clock is running at over 222.36 MHz (worst case timing). If we were not meeting timing, the timing analyzer will allow you to see your failing paths and do a more in-depth analysis. For this tutorial, we won't go into details on timing slack analysis. See iCEcube2 software usage guide for details on timing analysis.

Perform Power Analysis

iCEcube2 also comes with power estimator tool. To launch the power estimator, go to the menu and select **Tool >> Power Estimator**. You can alternatively select the power estimator icon. Figure 22. There are multiple tabs in the Power Estimator tool including Summary, IO, and Clock Domain. On the Summary tab, **change the Core Vdd to 1.2V** and make sure all **IO voltages are at 2.5V**. Then hit **calculate**. The estimator will update with power information for both static and dynamic power. For more information on using the IO and Clock Domain tabs, please refer to the detailed section on the Power Estimator tool in the iCEcube2 software userguide.



Core Vdd(V): 1.2

IO Voltage

Left Bank IO Voltage(V): 2.5

Right Bank IO Voltage(V): 2.5

Top Bank IO Voltage(V): 2.5

Bottom Bank IO Voltage(V): 2.5

Dynamic Power Breakdown

Core Power(mW): 2.11817

IO Power(mW): 4.25

Power Consumption

Static Power(mW): 0.3204

Dynamic Power(mW): 6.36817

Total Power(mW): 6.68857

Figure 22: Power Estimator

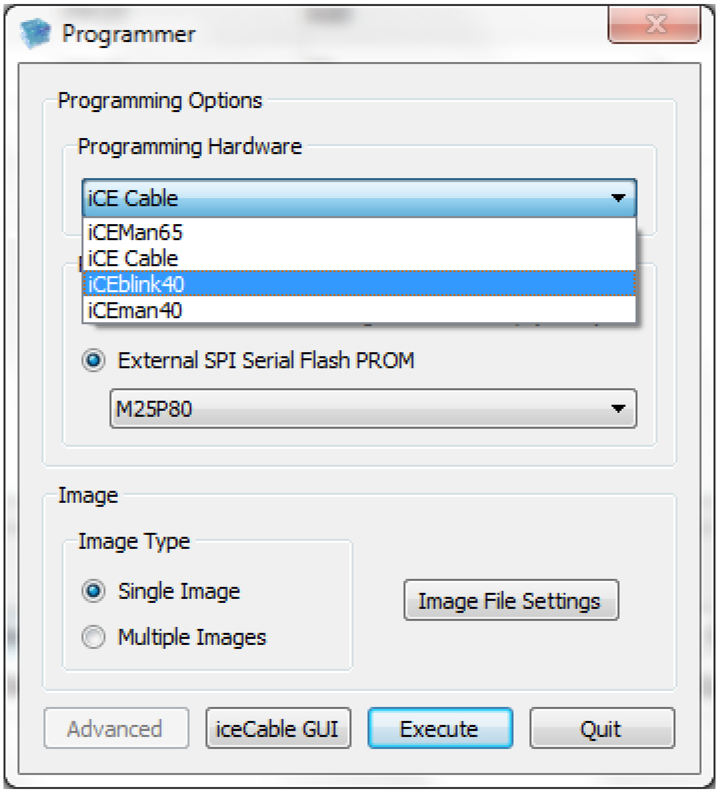
Programming the Device

In order to program a device, you will need to generate a programming file. In the project navigator, **double click on Bitmap**. Expand the Bitmap section in the Output Files. The section is now populated with .hex and .bin files. The .hex files are used for programming the external SPI Flash on Lattice iCE evaluation kit. The iCE40 configures itself from the SPI Flash.

You are now ready to program an iCE40 FPGA device with the generated bitmap. Invoke the programmer from the **Programming** icon which is now enabled in the Project Navigator. Alternately, you may invoke it from the **Tool>Programmer** menu item.

To program the Lattice iCEblink40 board, select iCEblink40 from the Programming Options dropdown menu and select M25P10A for the External Serial Flash PROM. See Figure 23. For more details on programming the iCEblink40 evaluation kit, refer to Evaluation Kit's User's Guide. Click on the image file settings button to ensure you latest .hex file is selected, as shown in Figure 24.

Additional details on programming a device are provided in a separate section *Programming the Device* in section *iCEcube2 Physical Implementation Tools* of the iCEcube2 software usage guide.



Programmer

Programming Options

Programming Hardware

iCE Cable ▾

- iCEMan65
- iCE Cable
- iCEblink40
- iCEman40

● External SPI Serial Flash PROM

M25P80 ▾

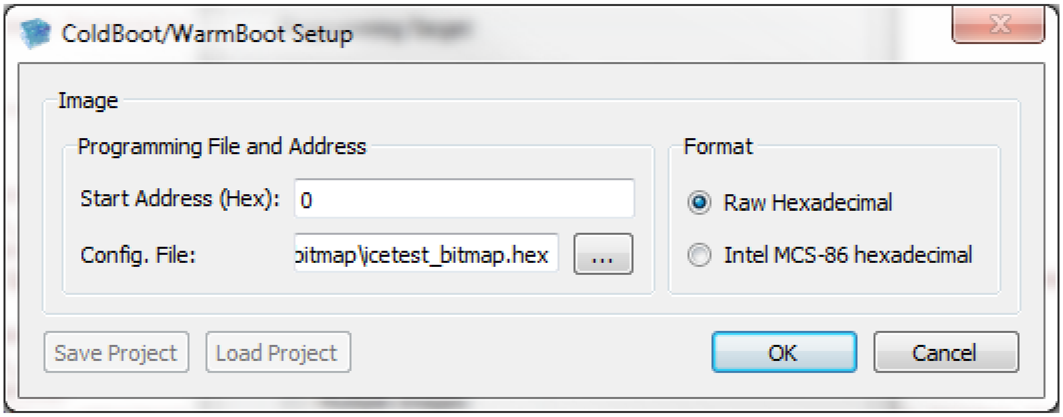
Image

Image Type

● Single Image

○ Multiple Images

Figure 23: Programmer Graphical User Interface



ColdBoot/WarmBoot Setup

Image

Programming File and Address

Start Address (Hex): 0

Config. File: ɔitmap\icetest_bitmap.hex ...

Format

● Raw Hexadecimal

○ Intel MCS-86 hexadecimal

Figure 24: Image File Settings