Using Mentor Graphics ModelSim Simulator with Lattice iCEcube2

May 25, 2015(1.6) Application Note AN006

Summary

The Lattice iCEcube2 development software provides a complete FPGA implementation environment for today's FPGA designers. This application note details the basic design and simulation flow using Mentor Graphics ModelSim simulator.

Introduction

The sample design used in this application note is a simple 4-bit binary up-counter with an associated testbench. The counter design, counter.vhd is presented first:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity counter is port (
     clk   : in   std_logic;
     reset : in   std_logic;
     count : out  std_logic_vector (3 downto 0));
end counter;

architecture behavioral of counter is
     signal q : std_logic_vector (3 downto 0);
begin

process(clk, reset)
begin
     if(reset = '1') then
          q <= (others=>'0');
     elsif(clk'event and clk = '1') then
          q <= q + 1;
     end if;
end process;

count <= q;

end behavioral;
```

The testbench design, count_tb.vhd, is presented next:

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity count_tb is
end count_tb;
```

© 2007-2015 Lattice Semiconductor Corp.

```
architecture testbench_arch of count_tb is

      component counter
      port (
            clk   : in std_logic;
            reset : in std_logic;
            count : out std_logic_vector (3 downto 0));
      end component;

      signal clk : std_logic := '0';
      signal reset : std_logic := '0';
      signal count : std_logic_vector (3 downto 0) := "0000";

      constant period : time := 100 ns;
      constant duty_cycle : real := 0.5;
      constant offset : time := 100 ns;
begin

      uut : counter
      port map (
            clk => clk,
            reset => reset,
            count => count);

      process -- clock generation
      begin
      wait for offset;
      clock_loop : loop
            clk <= '0';
            wait for (period - (period * duty_cycle));
            clk <= '1';
            wait for (period * duty_cycle);
      end loop clock_loop;
      end process;

      process -- reset generation
      begin
            reset <= '0';
            -- -------------  Current Time:  0ns
            wait for 100 ns;
            reset <= '1';
            -- -------------  Current Time:  100ns
            wait for 35 ns;
            reset <= '0';
            -- -------------  Current Time:  135ns
            wait for 1865 ns;
            -- -------------  Current Time:  2000ns
      end process;

end testbench_arch;
```

© 2007-2015 Lattice Semiconductor Corp.

This document explains the following scenarios:

1. Pre-Synthesis Simulation.
2. Post-Synthesis Functional Simulation (Verilog/VHDL).
3. Place-n-Route Functional Simulation (Verilog).
4. Place-n-Route Timing Simulation (Verilog).
5. Place-n-Route Functional Simulation (VHDL).
6. Place-n-Route Timing Simulation (VHDL)

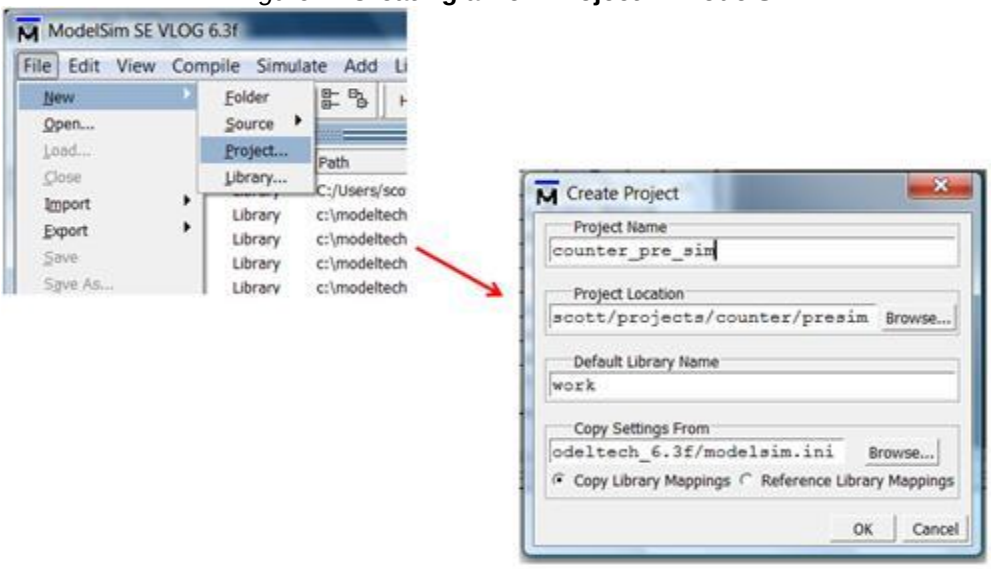
© 2007-2015 Lattice Semiconductor Corp.

Pre-Synthesis Simulation

This section details the steps required for pre-synthesis simulation.

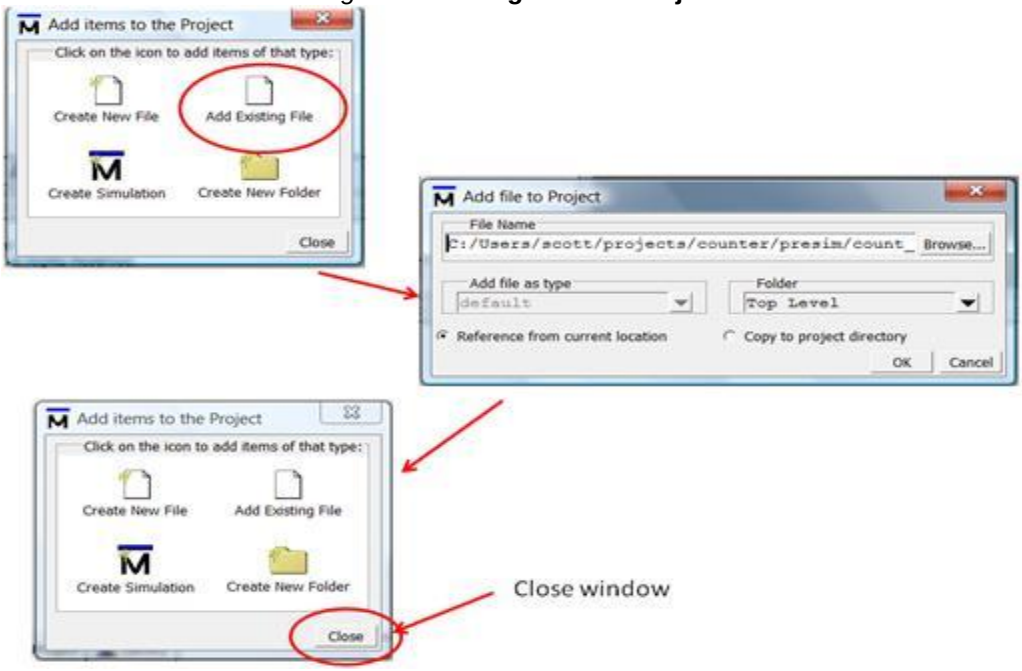
1. Create a new folder and copy the design (counter.vhd) and testbench (count_tb.vhd) shown above.
2. Start ModelSim and create a new project by clicking on File→New→Project and browse to the newly created project folder and enter a project name and click "OK".

Figure 1: **Creating a New Project in ModelSim**



3. Click on "add existing files" and add counter.vhd and count_tb.vhd to the project. Close the "Add items to the Project" window.

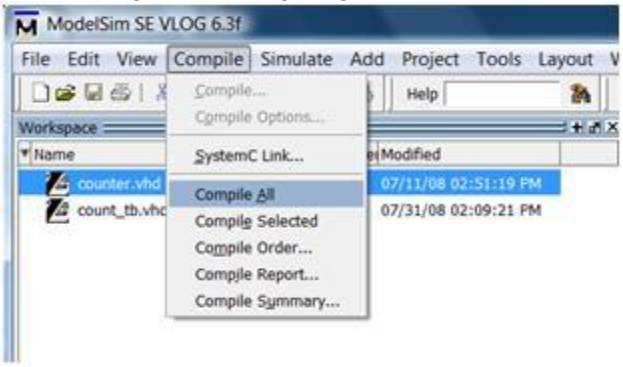
Figure 2: **Adding Files to Project**



4. Click on Compile→Compile All.

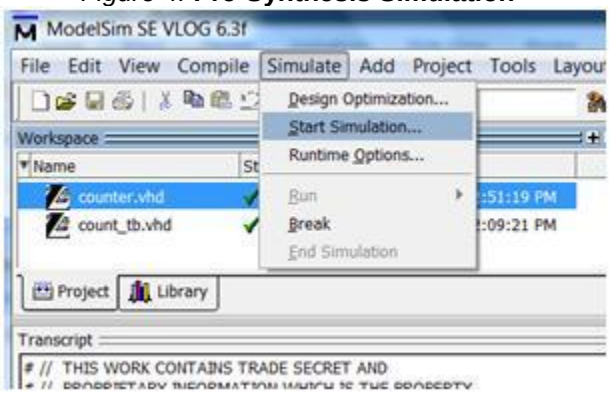
© 2007-2015 Lattice Semiconductor Corp.

Figure 3: **Compiling HDL Files**



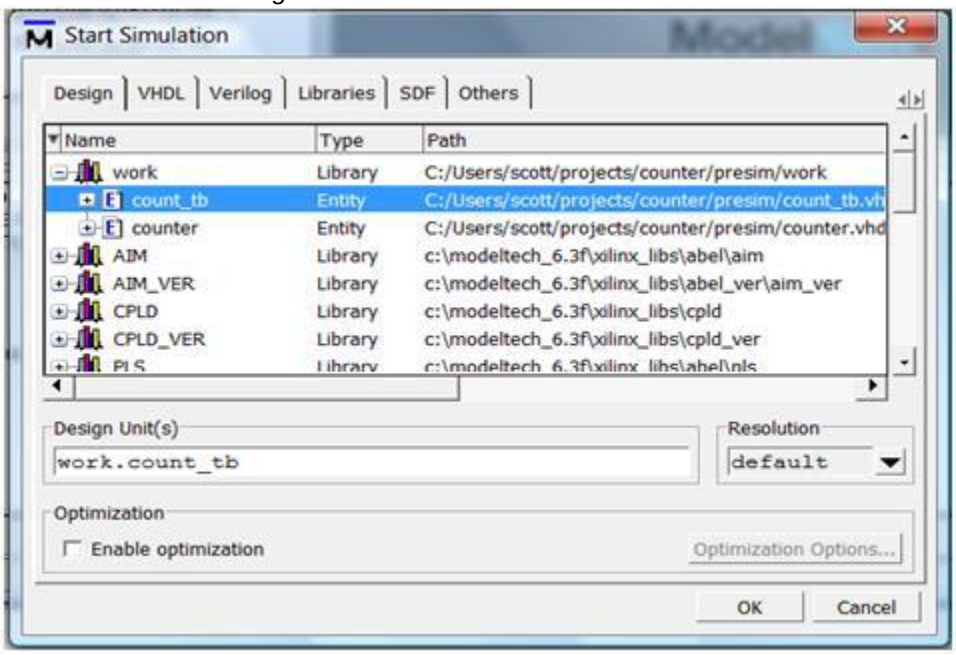
5. Click on Simulate→Start Simulation.

Figure 4: **Pre-Synthesis Simulation**



6. Expand the work folder and highlight count_tb.

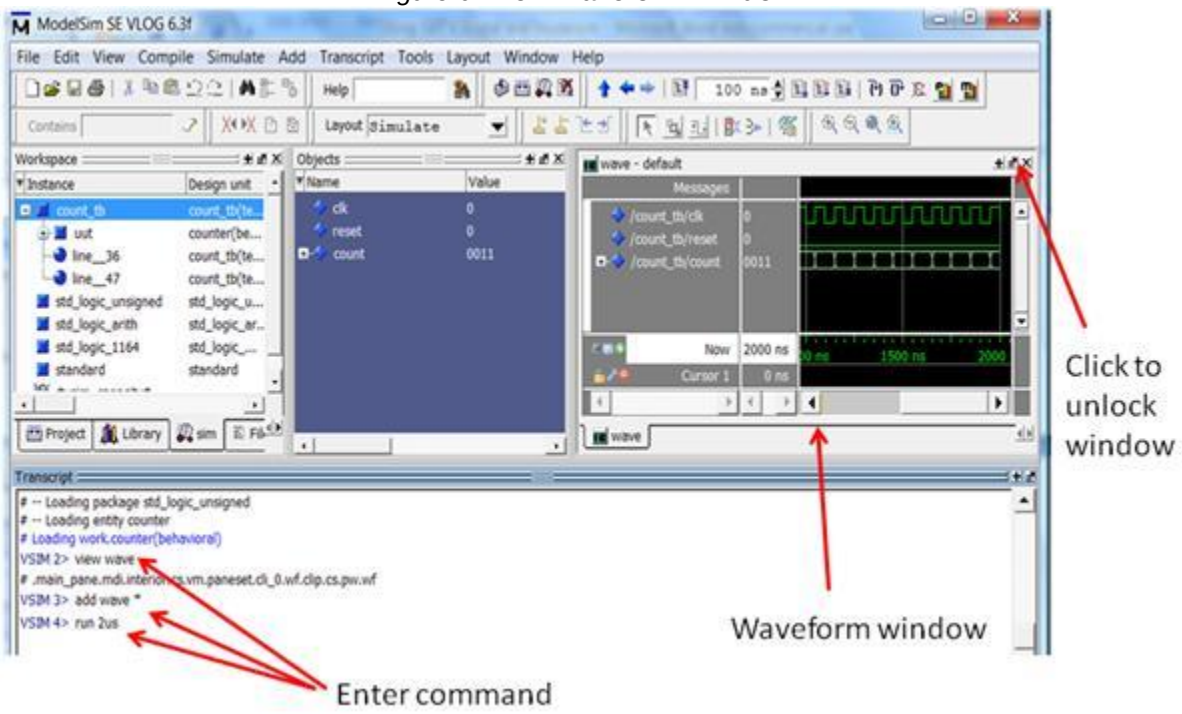
Figure 5: **Start Simulation Window**



7. Type the command “View wave” and press the enter key; a new waveform window will show up.
8. Type the command “Add wave *” and press the enter key. All I/O signals in the design will show up in the waveform window.
9. Type the command “run 2us” and press enter. The 2uS simulation waveform will show up in the waveform window.
10. Click on the “unlock” button to unlock the waveform window for a better view.

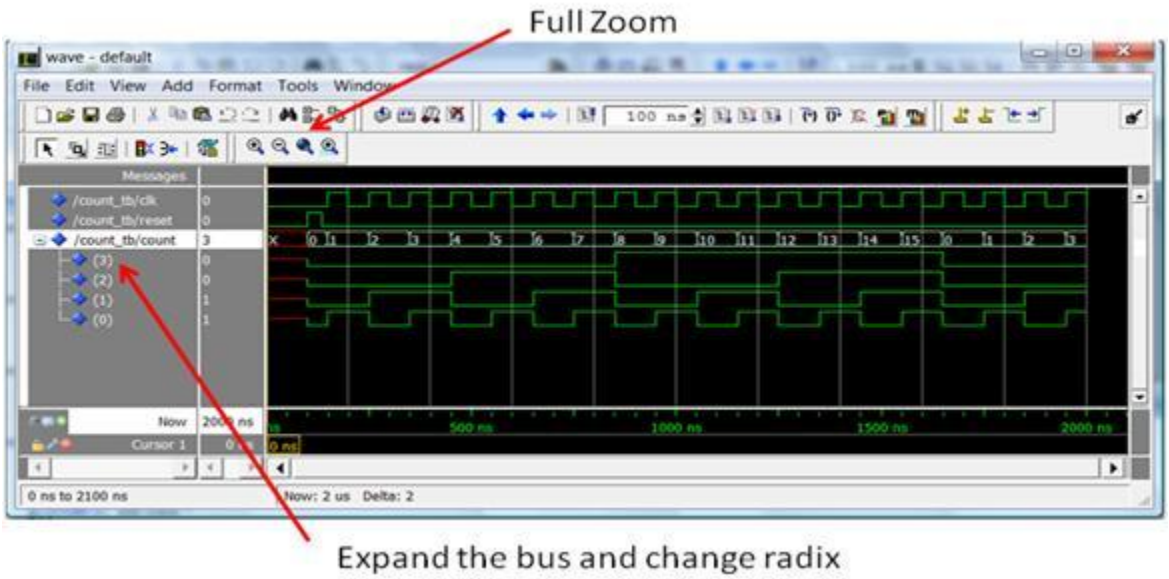
© 2007-2015 Lattice Semiconductor Corp.

Figure 6: **View Waveform Window**



11. Click on “count” to expand this bus and right click on “count” and select Radix→unsigned to change the bus number radix and click on “Zoom Full” to have a better view.

Figure 7: **Waveform Window**



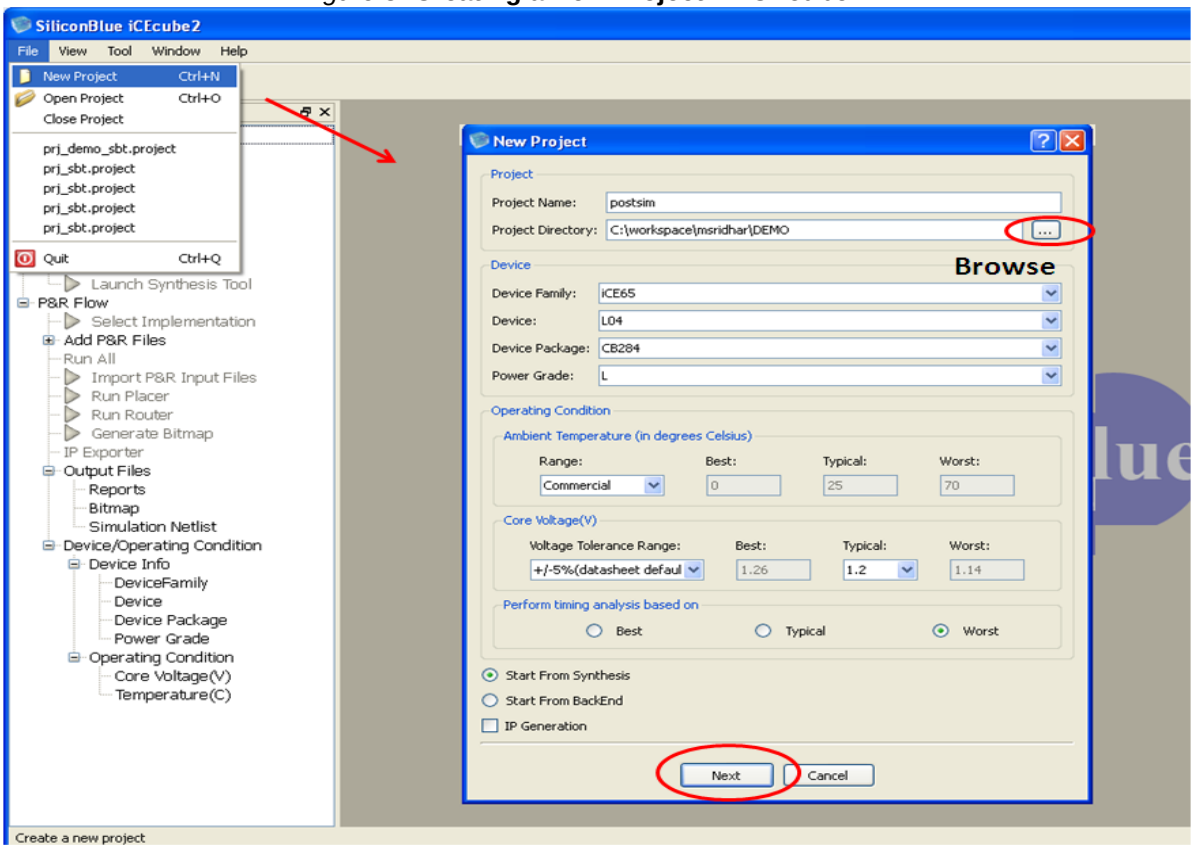
Generating Files required for Post-Synthesis and Post-Route Simulations

This section details the steps required for generating the files that are required for performing post-synthesis and post-route simulations.

1. Start iCEcube2 and create a new project by clicking on Project→New Project, browse to the project folder and enter a new project name and select an appropriate part and click “Next”.

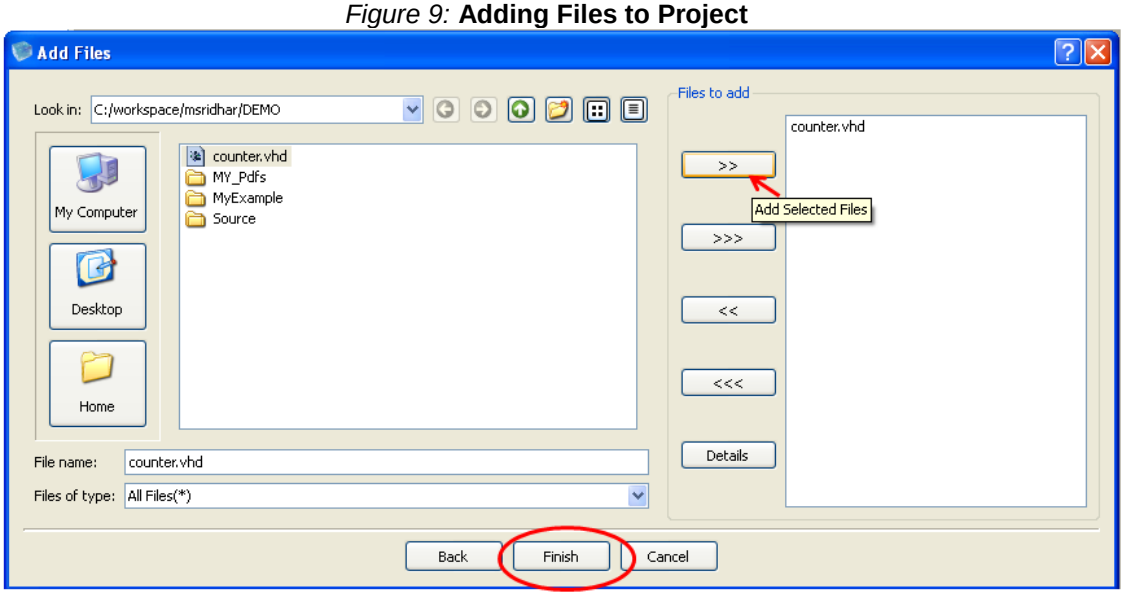
© 2007-2015 Lattice Semiconductor Corp.

Figure 8: **Creating a New Project in iCEcube2**



2. Highlight counter.vhd and click on “>>” to add counter.vhd to the right panel and click on “Finish”.

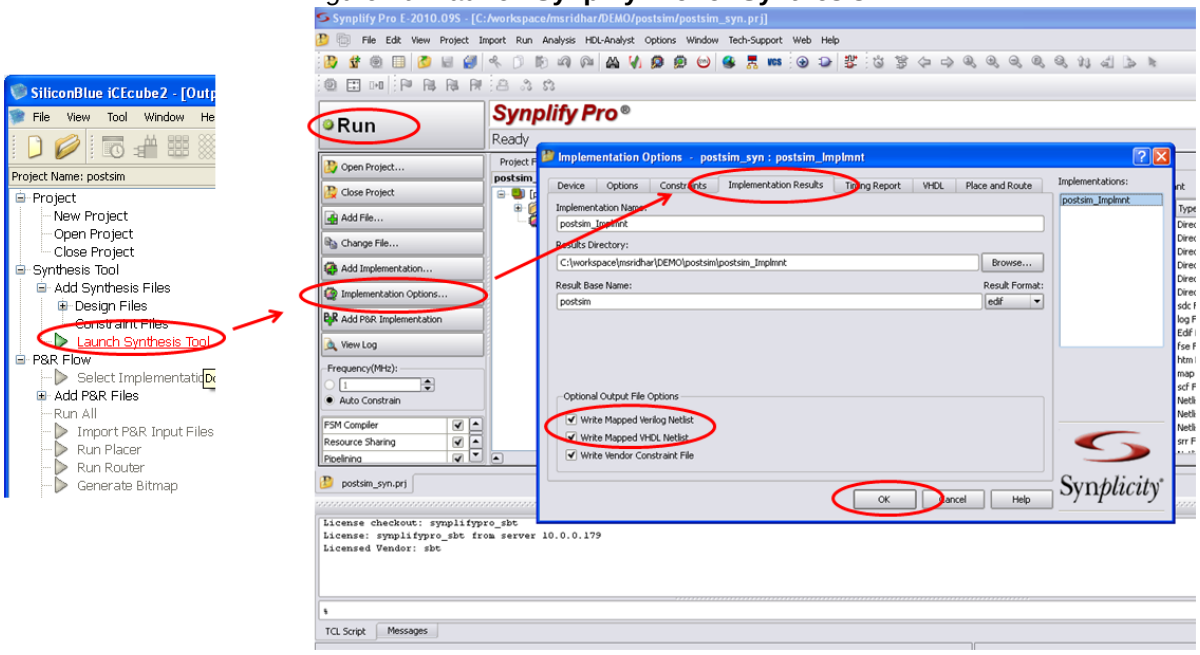
Figure 9: **Adding Files to Project**



3. Click on the “Launch Synthesis Tool” button to launch Synplify Pro for synthesis. In order to generate a post synthesis simulation model in verilog/vhdl, set the “Write Mapped Verilog Netlist/Write Mapped VHDL Netlist” option in synthesis tool by going through Implementation Options → Implementation Results, before doing synthesis. Click on the “RUN” icon in Synplify pro to synthesis the design.

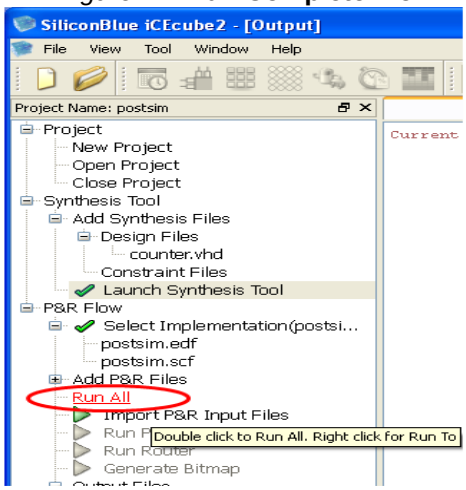
© 2007-2015 Lattice Semiconductor Corp.

Figure 10: **Launch Synplify Pro for Synthesis**



4. Close the Synplify Pro tool after Synthesis. This will bring you back to iCEcube2 tool window. The Synthesis outputs “PRJNAME.edf” and “PRJNAME.scf” would be automatically added to “Select Implementation” in “P&R Flow” of iCEcube2. Click on the “Run All” icon to run placement, routing and Bitmap Generation.

Figure 11: **Run Complete Flow**



After running the complete flow, the following outputs will be generated.

- PRJNAME.vm, post-synthesis simulation Verilog model
- PRJNAME.vhm, post-synthesis simulation VHDL model
- counter_sbt.v, post-route timing simulation Verilog model
- counter_sbt.vhd, post-route timing simulation VHDL model
- counter_sbt.sdf, for post-route timing Verilog model simulation
- counter_sbt_vital.sdf, for post-route timing VHDL model simulation.

Post Synthesis simulation models can be found typically in *PRJNAME/PRJNAME_Implmnt/.* These would be generated only if we keep the settings mentioned in Step 3, during synthesis.

© 2007-2015 Lattice Semiconductor Corp.

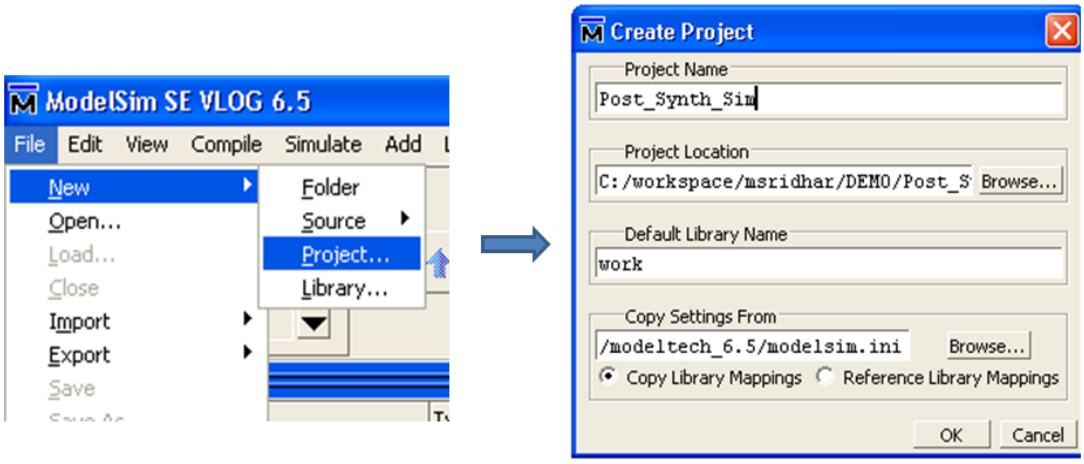
Post route timing models can be found typically in
PRJNAME/PRJNAME_Implmnt/sbt/outputs/simulation_netlist

SDF is a standardized representation of timing data commonly used when exchanging timing information between design and simulation tools.

Post-Synthesis Functional Simulation (Verilog/VHDL)

1. Generate the files that are required for Post-Synthesis Functional simulation model using the steps described in the section “Generating Files required for Post-Synthesis and Post-Route Simulations”.
2. Open ModelSim. Create new project using File → New → Project. In the New Project window, give a project name and browse to a folder where the project needs to be saved. Click ok.

Figure 12: **Create New Project**



3. Click on “Add Existing Files” and add the following files:
   a. PRJNAME.vm, sb_ice_syn.v, counter_tb.vhd for verilog post-synthesis simulation
   b. PRJNAME.vhm, sb_ice_syn.vhd, counter_tb.vhd for VHDL post-synthesis simulation

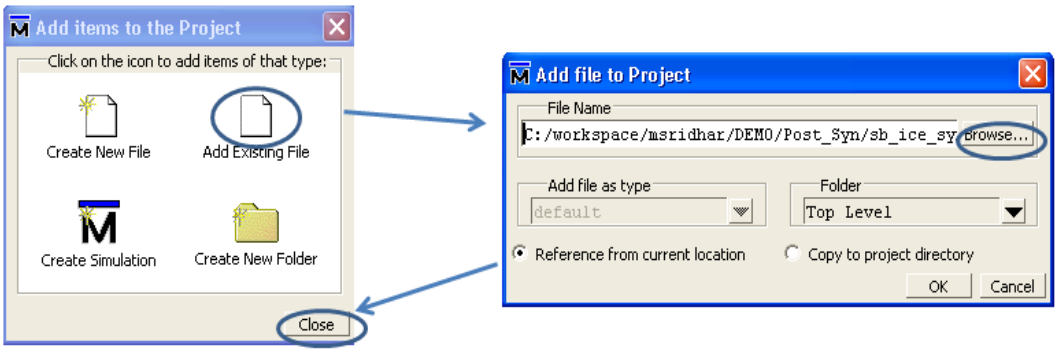
sb_ice_syn.v can be found in $INST_DIR/verilog and sb_ice_syn.vhd can be found in $INST_DIR/VHDL.

If your design contains **PLL,** add **ABIPTBS8.v** and **ABIWTCZ4.v** in $INST_DIR/verilog. For performing Post-Synth simulation for a VHDL design having PLL, you will require a mixed-language simulator, since the PLL model (ABIPTBS8.v) is available only in verilog format.
If the design contains **Hardened IP** primitives, add the encrypted Verilog simulation library **sb_ice_ipenc_modelsim.v** available in $INST_DIR/Verilog.

Close the “Add items to project” window once all the files are added.

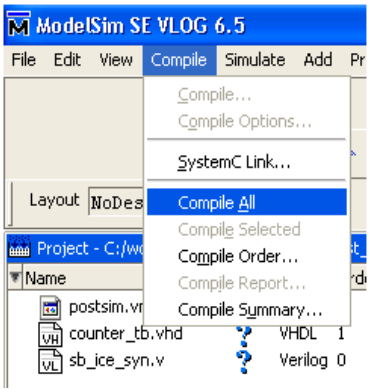
© 2007-2015 Lattice Semiconductor Corp.

Figure 13: **Add Files to Project**



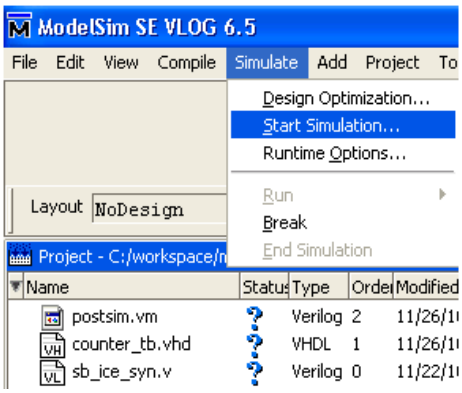
4. Click on Compile → compile all

Figure 14: **Compile All**



5. Start simulation from Simulate → Start Simulation

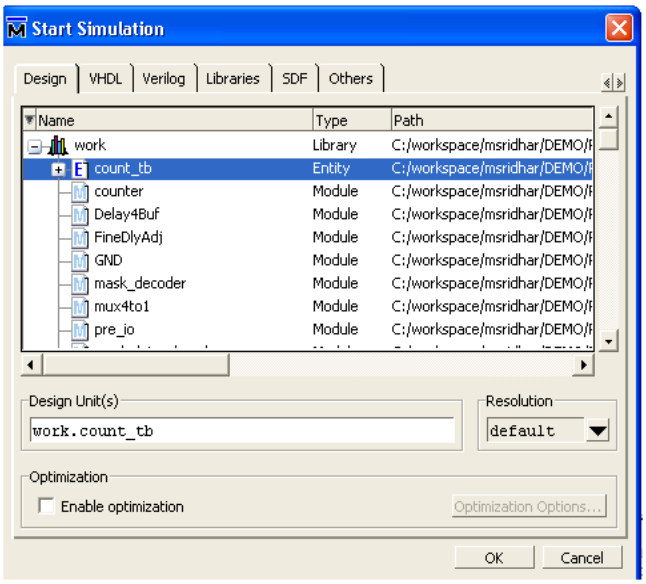
Figure 15: **Start Simulation**



6. Expand the work folder in “Start Simulation” window and highlight “counter_tb”. Click OK.

© 2007-2015 Lattice Semiconductor Corp.

Figure 16: **Run Simulation**

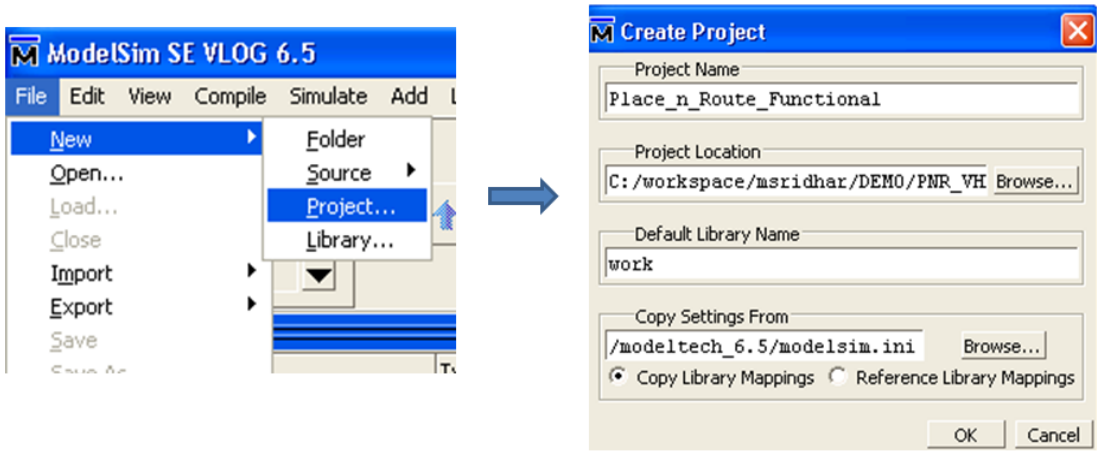


7. This will open simulation window. Add signals wave window and perform simulation as explained in the “Pre-Synthesis Simulation” section from point 7 onwards.
8.

Place-n-Route Functional Simulation (Verilog)

1. Generate the files that are required for Post-Route Functional simulation model using the steps described in the section “Generating Files required for Post-Synthesis and Post-Route Simulations”.
2. Open ModelSim. Create new project using File → New → Project. In the New Project window, give a project name and browse to a folder where the project needs to be saved. Click ok.

Figure 17: **Create New Project**



3. Click on “Add Existing Files” and add the following files:
   a. counter_sbt.v
   b. counter_tb.vhd

   Add the following verilog simulation libraries available in **$INST_DIR/verilog**

   a. sb_ice_syn.v
   b. sb_ice_lc.v

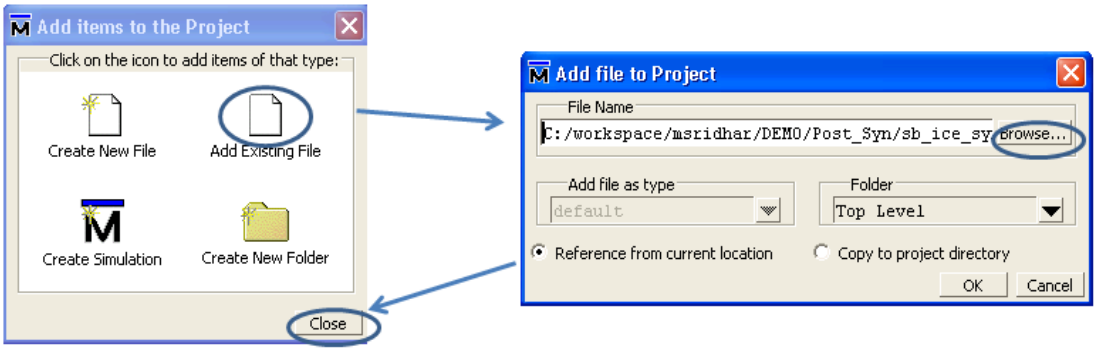
   If your design contains **PLL,** add **ABIPTBS8.v** & **ABIWTCZ4.v** in $INST_DIR/verilog.

© 2007-2015 Lattice Semiconductor Corp.

If the design contains **Hardened IP** primitives, add the encrypted simulation library **sb_ice_ipenc_modelsim.v** available in $INST_DIR/Verilog.

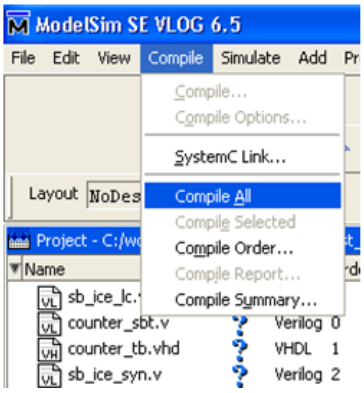
Close the “Add items to project” window once all the files are added.

Figure 18: **Add Files to Project**



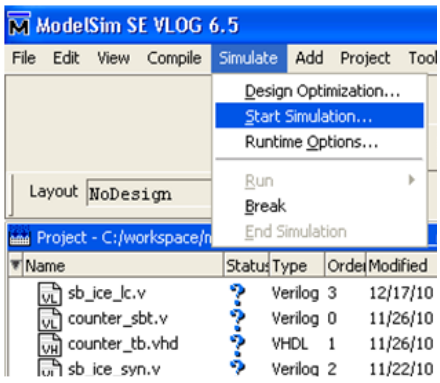
4. Click on Compile → Compile All.

Figure 19: **Compile All**



5. Start simulation from Simulate → Start Simulation.

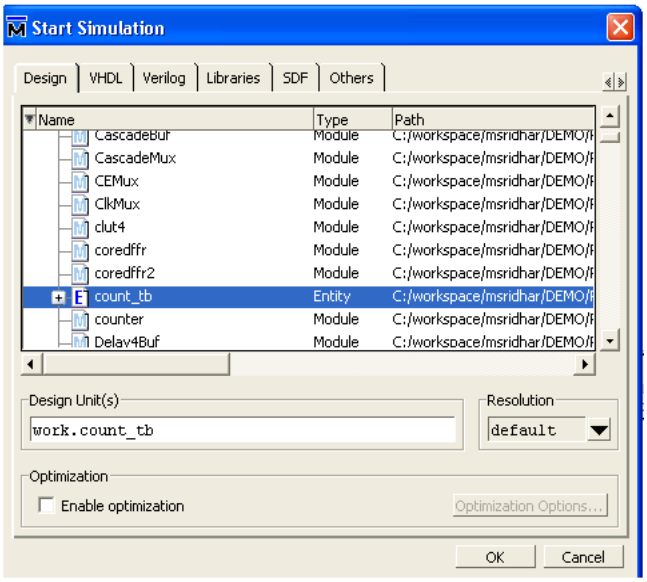
Figure 20: **Start Simulation**



6. Expand the work folder in “Start Simulation” window and highlight “counter_tb”. Click OK.

© 2007-2015 Lattice Semiconductor Corp.

Figure 21: **Run Simulation**



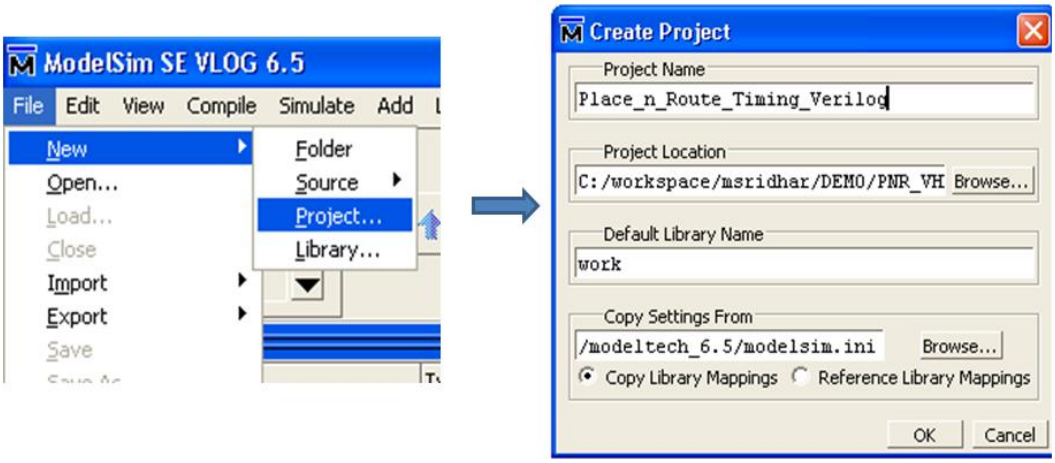
7. This will open simulation window. Add signals wave window and perform simulation as explained in the “Pre-Synthesis Simulation” section from point 7 onwards.

Place-n-Route Timing Simulation (Verilog)

1. Generate the files that are required for Post-Route Timing simulation model using the steps described in the section “Generating Files required for Post-Synthesis and Post-Route Simulations”.

2. Open ModelSim. Create new project using File → New → Project. In the New Project window, give a project name and browse to a folder where the project need to be saved. Click ok.

Figure 22: **Create New Project**



3. Click on “Add Existing Files” and add the following files:
   a. counter_sbt.v, sb_ice_syn.v, sb_ice_lc.v, counter_tb.vhd for verilog Post-Route timing simulation

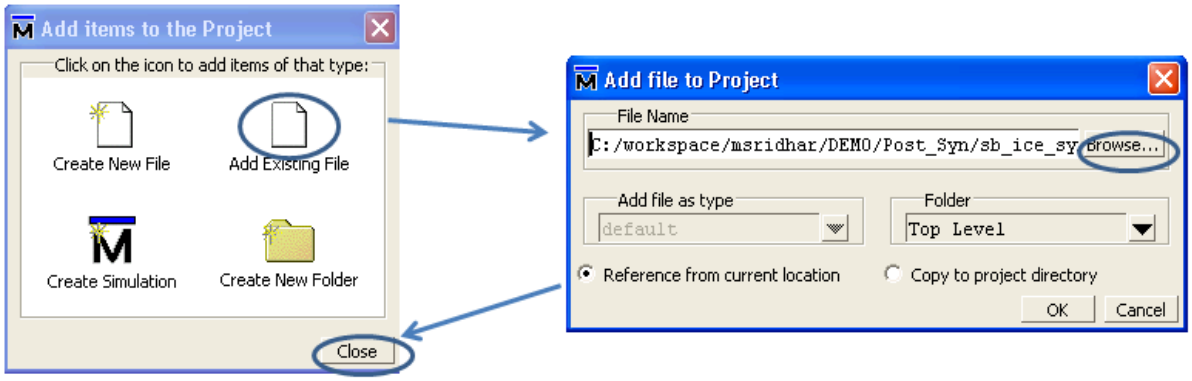
   sb_ice_syn.v, sb_ice_lc.v verilog files can be found in $INST_DIR/verilog.

   If your design contains **PLL,** add **ABIPTBS8.v** in $INST_DIR/verilog.
   If the design contains **Hardened IP** primitives, add the encrypted simulation library **sb_ice_ipenc_modelsim.v** available in $INST_DIR/Verilog.

© 2007-2015 Lattice Semiconductor Corp.

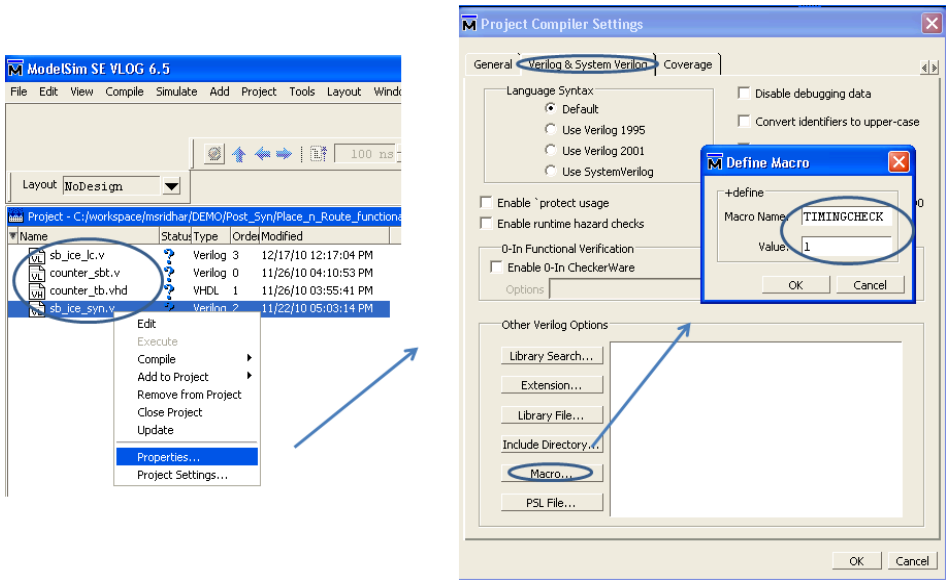
Close the “Add items to project” window once all the files are added.

Figure 23: **Add Files to Project**



4. In the project settings window, select the “verilog & system verilog” tab and click on “Macro”. In the define macro window set the macro name to “TIMINGCHECK” and set the value to 1. Click OK.

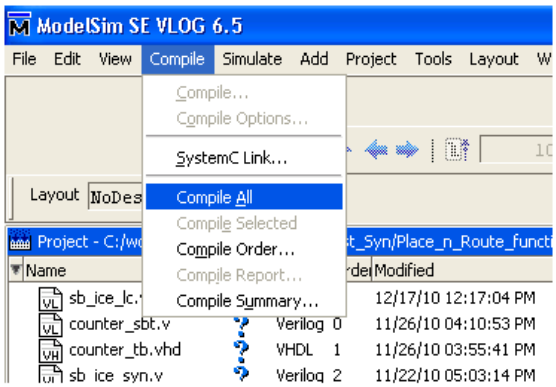
Figure 24: **Set TIMINGCHECK Macro**



5. Click on Compile → Compile All.

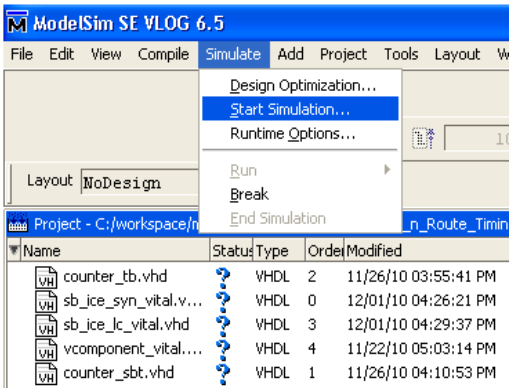
© 2007-2015 Lattice Semiconductor Corp.

Figure 25: **Compile All**



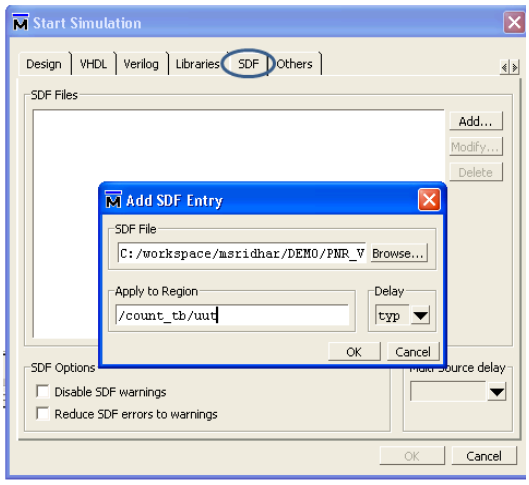
6. Start simulation using Simulate → Start Simulation

Figure 26: **Start Simulation**



7. In the Simulation window, select SDF tab and add the SDF file counter_sbt.sdf and type "/count_tb/uut" in "Apply to Region". Click OK.

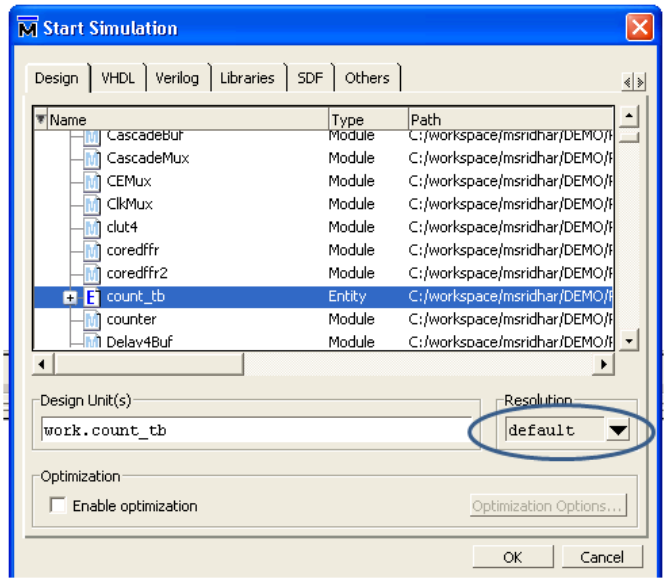
Figure 27: **Add SDF to project**



8. In the Design tab, select "count_tb" under work. Make sure that the "Time Resolution" was set to PS. Click OK.

© 2007-2015 Lattice Semiconductor Corp.

Figure 28: **Run Simulation**

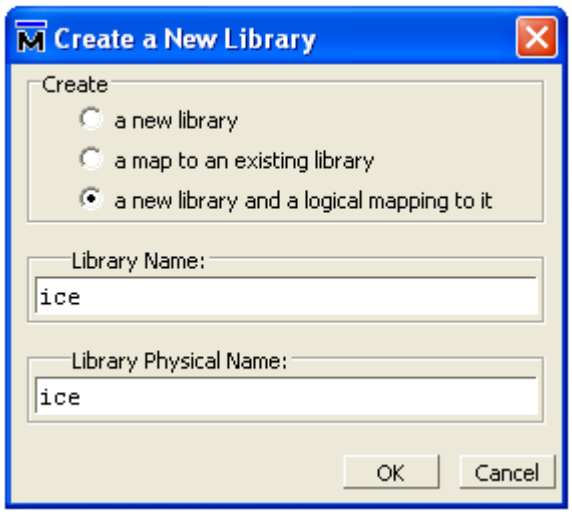


9. This will open simulation window. Add signals wave window and perform simulation as explained in the “Pre-Synthesis Simulation” section from point 7 onwards.

Place-n-Route Functional Simulation (VHDL)

1. Generate the files that are required for Post-Route Functional simulation model using the steps described in the section “Generating Files required for Post-Synthesis and Post-Route Simulations”.
2. Open Modelsim. Create vhdl resource library using File → New → Library. In the Create New Library window select “a new library and a logical mapping to it” option and specify the library name as ‘**ice**’.

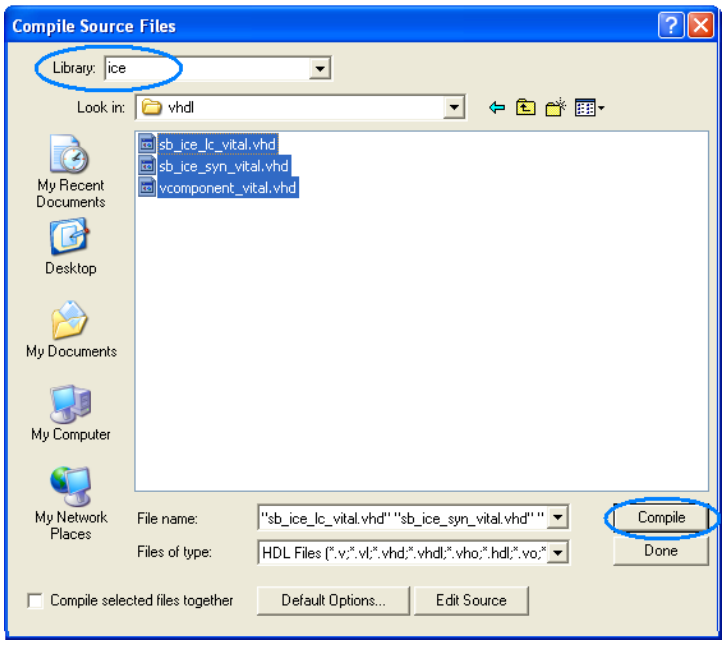
Figure 29 : Create ice resource library



3. Select Compile → Compile option. Change library to ‘**ice**’ and browse to path $INST_DIR/vhdl. Select the following vhdl simulation library files and click on “compile” button.
   a. vcomponent_vital.vhd
   b. sb_ice_syn_vital.vhd
   c. sb_ice_lc_vital.vhd

© 2007-2015 Lattice Semiconductor Corp.

Figure 30 : Create "ice" resource library

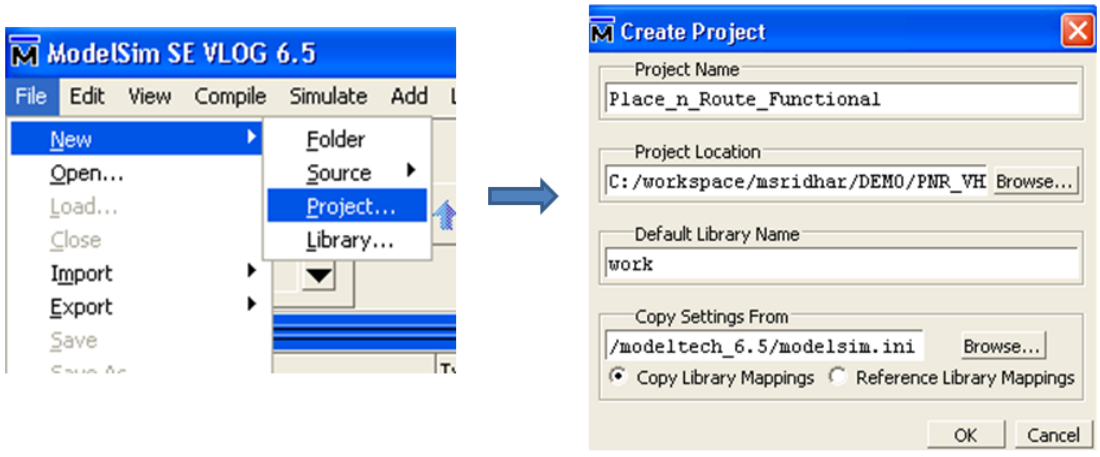


If the design contains PLL, Select Verilog PLL modules ABIPTBS8.v and ABIWTCZ4.v in $INST_DIR/verilog and compile into '**ice**' resource library.
If the design contains **Hardened IP** primitives, add the encrypted Verilog simulation library **sb_ice_ipenc_modelsim.v** available in $INST_DIR/Verilog.

Note: Mixed Language simulator support is required to perform PLL, Hard IP VHDL simulations.

4. Create new project using File → New → Project. In the New Project window, give a project name and browse to a folder where the project needs to be saved. Click ok.

Figure 31: **Create New Project**

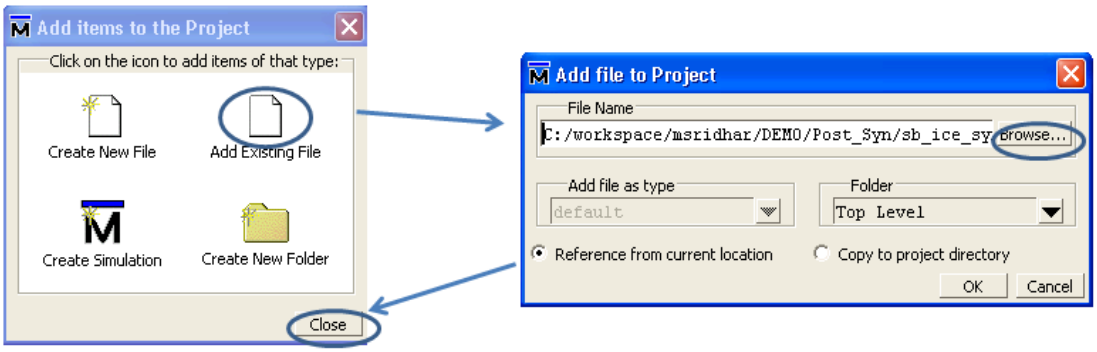


5. Click on "Add Existing Files" and add the following files:
   a. counter_sbt.vhd
   b. counter_tb.vhd

   Close the "Add items to project" window once all the files are added.

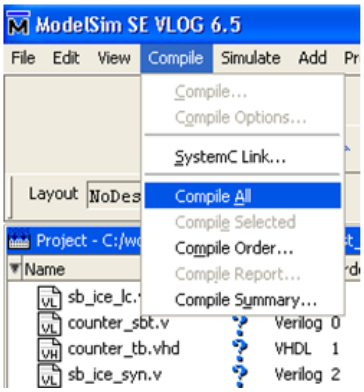
© 2007-2015 Lattice Semiconductor Corp.

Figure 32: **Add Files to Project**



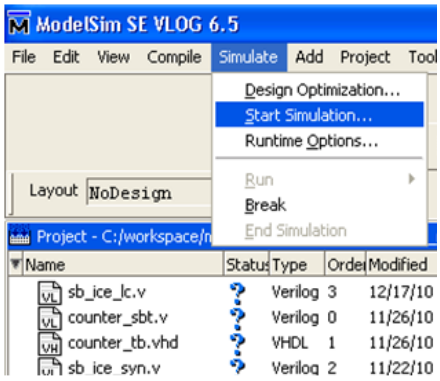
6. Click on Compile → Compile All.

Figure 33: **Compile All**



7. Start simulation from Simulate → Start Simulation.

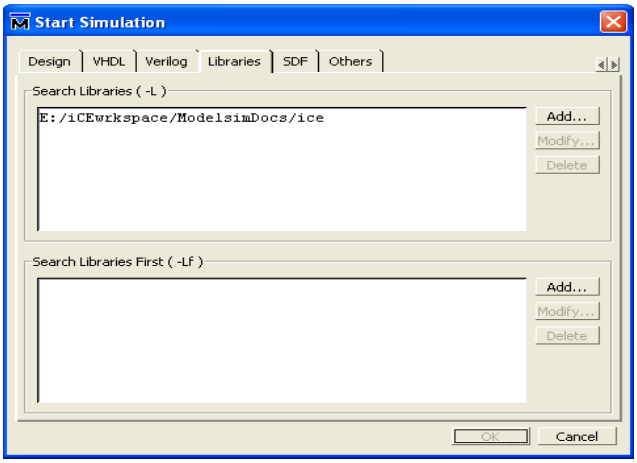
Figure 34: **Start Simulation**



8. In the simulation windows, select the tab “Libraries” and add the “**ice**” resource library in the search libraries path.

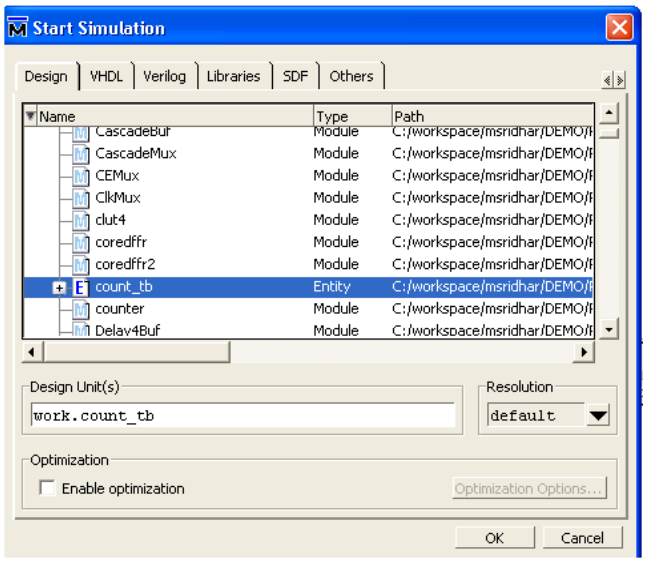
© 2007-2015 Lattice Semiconductor Corp.

Figure 35 : Add "ice" resource library to the project



9. Expand the work folder in "Start Simulation" window and highlight "counter_tb". Click OK.

Figure 36: **Run Simulation**



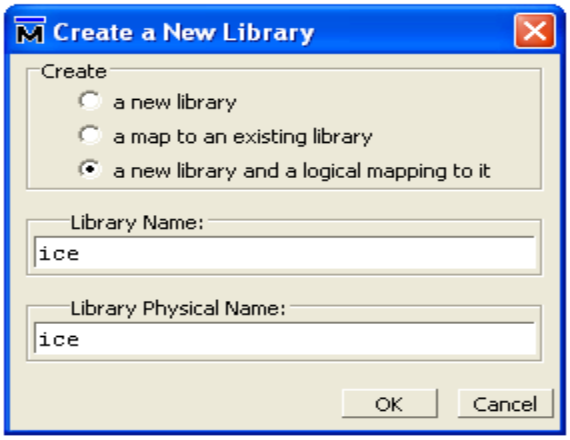
This will open simulation window. Add signals wave window and perform simulation as explained in the "Pre-Synthesis Simulation" section from point 7 onwards.

Place-n-Route Timing Simulation (VHDL)

1. Generate the files that are required for Post-Route Timing simulation model using the steps described in the section "Generating Files required for Post-Synthesis and Post-Route Simulations".
2. Open Modelsim. Create vhdl resource library using File → New → Library. In the Create New Library window select "a new library and a logical mapping to it" option and specify the library name as '**ice**'.

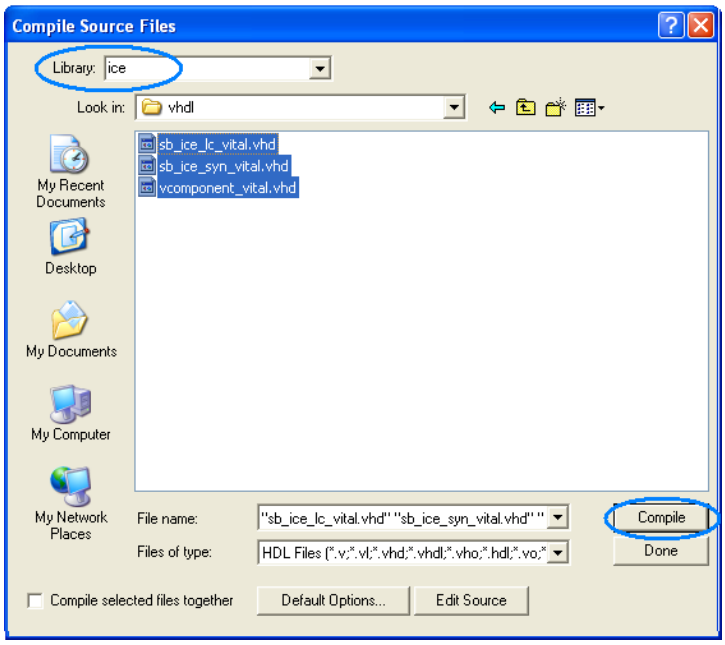
© 2007-2015 Lattice Semiconductor Corp.

Figure 37 : Create ice resource library



3. Select Compile → Compile option. Change library to '**ice**' and browse to path $INST_DIR/vhdl. Select the following vhdl simulation library files and click on "compile" button.
   a. vcomponent_vital.vhd
   b. sb_ice_syn_vital.vhd
   c. sb_ice_lc_vital.vhd

Figure 38 : Create "ice" resource library



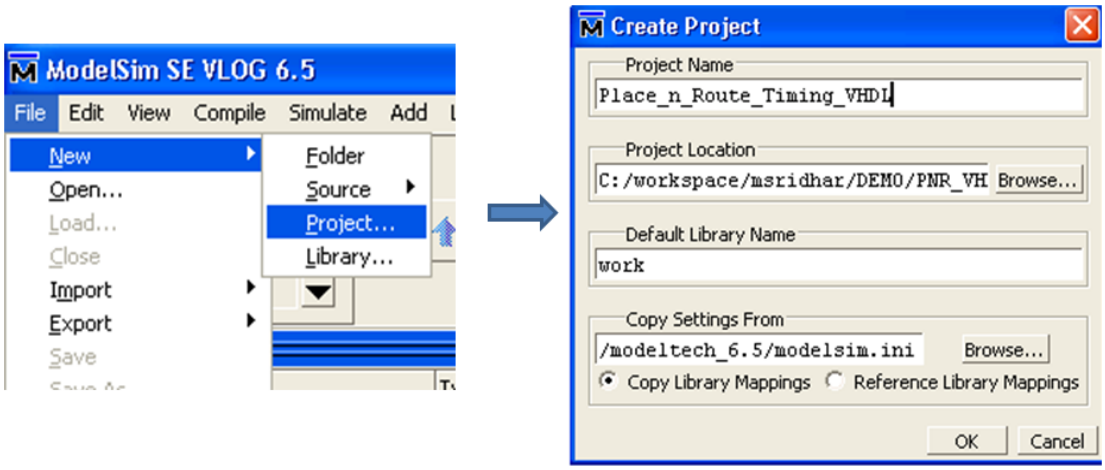
If the design contains PLL, Select Verilog PLL modules ABIPTBS8.v and ABIWTCZ4.v in $INST_DIR/verilog and compile into '**ice**' resource library.
If the design contains **Hardened IP** primitives, add the encrypted Verilog simulation library **sb_ice_ipenc_modelsim.v** available in $INST_DIR/Verilog.

Note: Mixed Language simulator support is required to perform PLL, Hard IP VHDL simulations.

4. Create new project using File → New → Project. In the New Project window, give a project name and browse to a folder where the project needs to be saved. Click ok.

© 2007-2015 Lattice Semiconductor Corp.

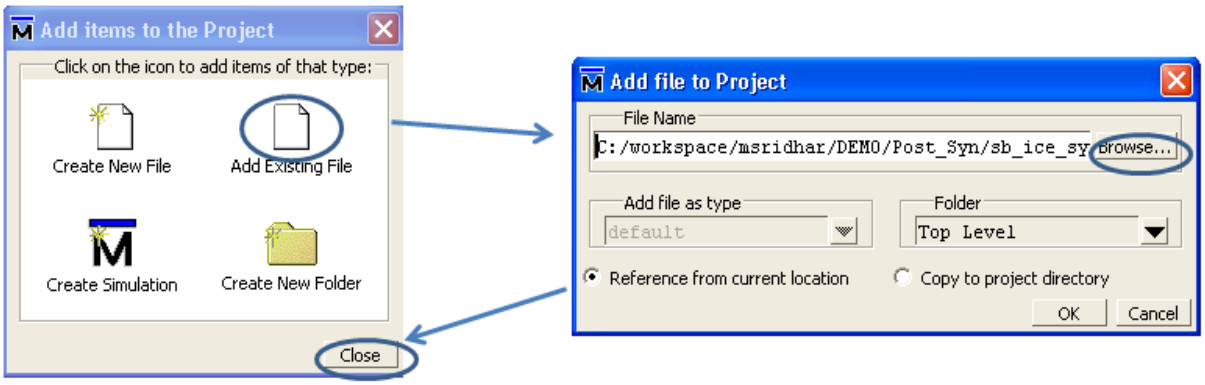
Figure 39: **Create New Project**



5. Click on “Add Existing Files” and add the following files:
   a. counter_sbt.vhd
   b. counter_tb.vhd

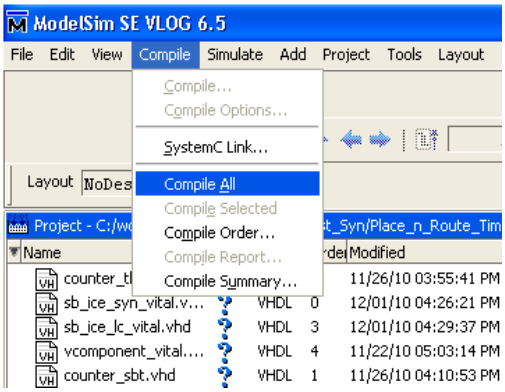
   Close the “Add items to project” window once all the files are added.

Figure 40: **Add Files to Project**



6. Click on Compile → Compile All.

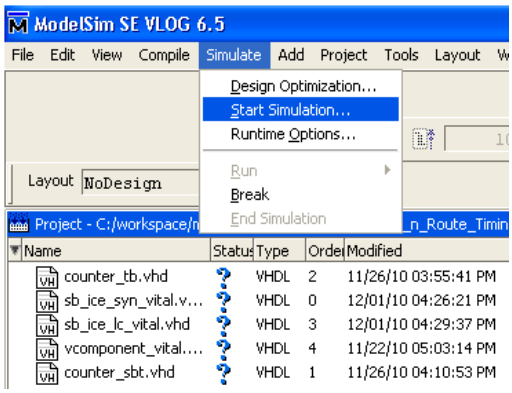
Figure 41: **Compile All**



7. Start simulation using Simulate → Start Simulation.

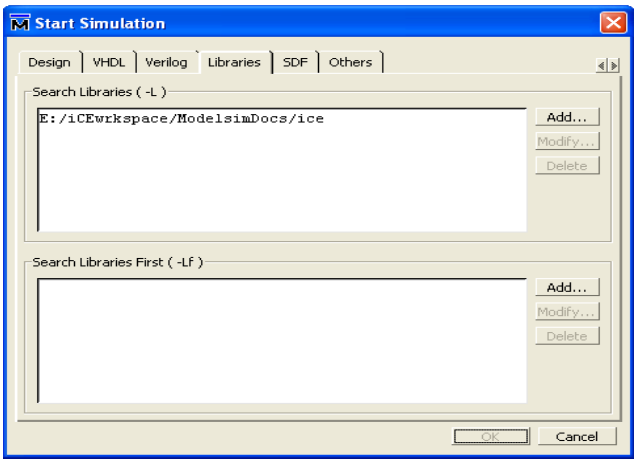
© 2007-2015 Lattice Semiconductor Corp.

Figure 42: **Start Simulation**



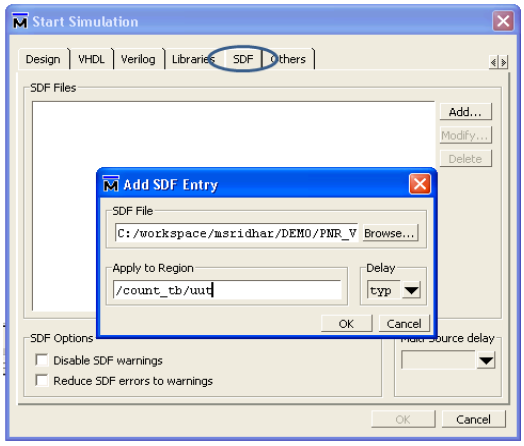
8. In the Simulation window , select “Libraries” tab and add the “**ice**” resource library in the search libraries path.

Figure 43 : Add “ice” resource library to the project



9. In the Simulation window, select SDF tab and add the SDF file counter_sbt_vital.sdf and type “/count_tb/uut” in “Apply to Region”. Click OK.

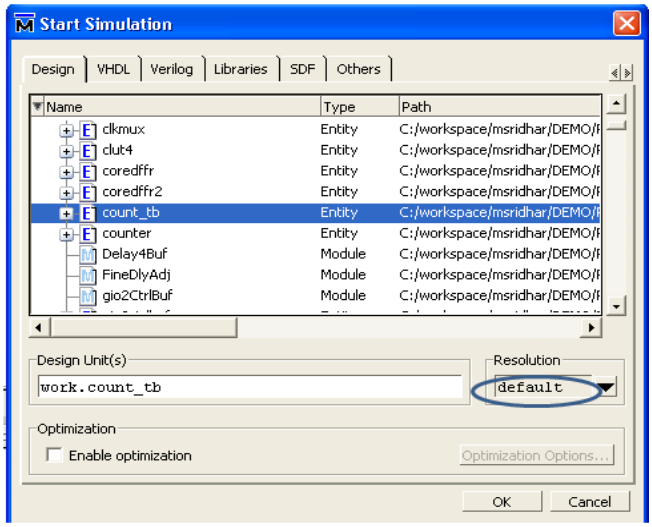
Figure 44: **Add SDF to project**



© 2007-2015 Lattice Semiconductor Corp.

10. In the Design tab, select “count_tb” under work. Make sure that the “Time Resolution” was set to PS. Click OK.

Figure 45: **Run Simulation**



11. This will open simulation window. Add signals wave window and perform simulation as explained in the “Pre-Synthesis Simulation” section from point 7 onwards.

Conclusion

The Lattice iCEcube2 development software provides a complete FPGA implementation environment for today’s FPGA designers. For simulation needs, Mentor Graphics ModelSim simulator complements the iCEcube2 development software and is a cost effective, easy to use simulation tool.

References

For more information on products, solutions, and applications enabled by Lattice Semiconductor Corporation, take the next step and visit www.latticesemi.com.

Revision History

Version	Date	Description
1.6	18-Dec-2013	Added resource library details for iCE40LM primitives.
1.5	26-APR-2013	Added resource library generation details for post route VHDL simulations.
1.4	24-DEC-2010	Added sections for pre/post synthesis, functional/timing, Verilog/VHDL simulations
1.3	23-DEC-2008	Updated corporate contact information.
1.2	20-OCT-2008	Initial release, updated iCEcube screen shots and timing simulation information.
1.1	06-AUG-2008	First draft, functional simulation only.

© 2007-2015 Lattice Semiconductor Corp.

Copyright

Copyright © 2007-2015 Lattice Semiconductor Corporation. All rights reserved. This document may not, in whole or part, be reproduced, modified, distributed, or publicly displayed without prior written consent from Lattice Semiconductor Corporation ("Lattice").

Trademarks

All Lattice trademarks are as listed at www.latticesemi.com/legal**.** Synopsys and Synplify Pro are trademarks of Synopsys, Inc. Aldec and Active-HDL are trademarks of Aldec, Inc. All other trademarks are the property of their respective owners.

Disclaimers

NO WARRANTIES: THE INFORMATION PROVIDED IN THIS DOCUMENT IS "AS IS" WITHOUT ANY EXPRESS OR IMPLIED WARRANTY OF ANY KIND INCLUDING WARRANTIES OF ACCURACY, COMPLETENESS, MERCHANTABILITY, NONINFRINGEMENT OF INTELLECTUAL PROPERTY, OR FITNESS FOR ANY PARTICULAR PURPOSE. IN NO EVENT WILL LATTICE OR ITS SUPPLIERS BE LIABLE FOR ANY DAMAGES WHATSOEVER (WHETHER DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OF OR INABILITY TO USE THE INFORMATION PROVIDED IN THIS DOCUMENT, EVEN IF LATTICE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. BECAUSE SOME JURISDICTIONS PROHIBIT THE EXCLUSION OR LIMITATION OF CERTAIN LIABILITY, SOME OF THE ABOVE LIMITATIONS MAY NOT APPLY TO YOU.

Lattice may make changes to these materials, specifications, or information, or to the products described herein, at any time without notice. Lattice makes no commitment to update this documentation. Lattice reserves the right to discontinue any product or service without notice and assumes no obligation to correct any errors contained herein or to advise any user of this document of any correction if such be made. Lattice recommends its customers obtain the latest version of the relevant information to establish that the information being relied upon is current and before ordering any products.

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