ModelSim Simulation Tutorial

Josh Brake
8/31/22

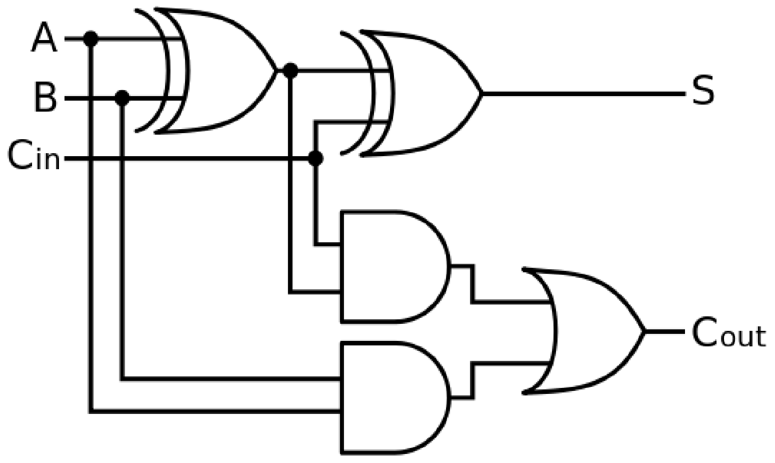
Introduction

To verify that designs are working, it is useful to simulate them using ModelSim. This document will provide a quick demo of how to perform a simulation in ModelSim and how to include iCE40UP library files in the simulation.

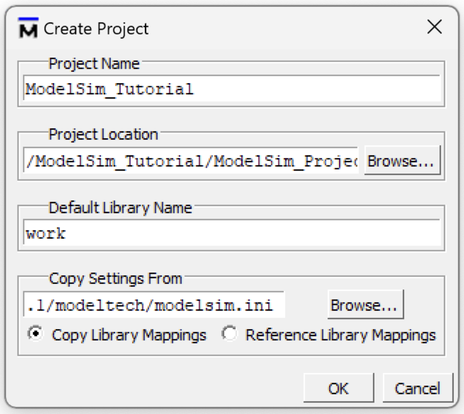
Simple Simulations by Forcing Signals

If you have a simple design and want to quickly test it out to see that it is working, you can use the force function in the ModelSim command window to apply test inputs and measure the outputs.

We will use a full adder circuit as a demonstration.



To code up the full adder, first, open ModelSim and create a new ModelSim project titled "ModelSim_Tutorial" and save it in a location where the path name has no spaces. Then create a new file named "fulladder.sv", making sure the file type is SystemVerilog. You should now have a blank file. Add the Verilog code below to create a structural Verilog model of the full adder circuit show above.



Create Project

Project Name: ModelSim_Tutorial

Project Location: /ModelSim_Tutorial/ModelSim_Projec

Default Library Name: work

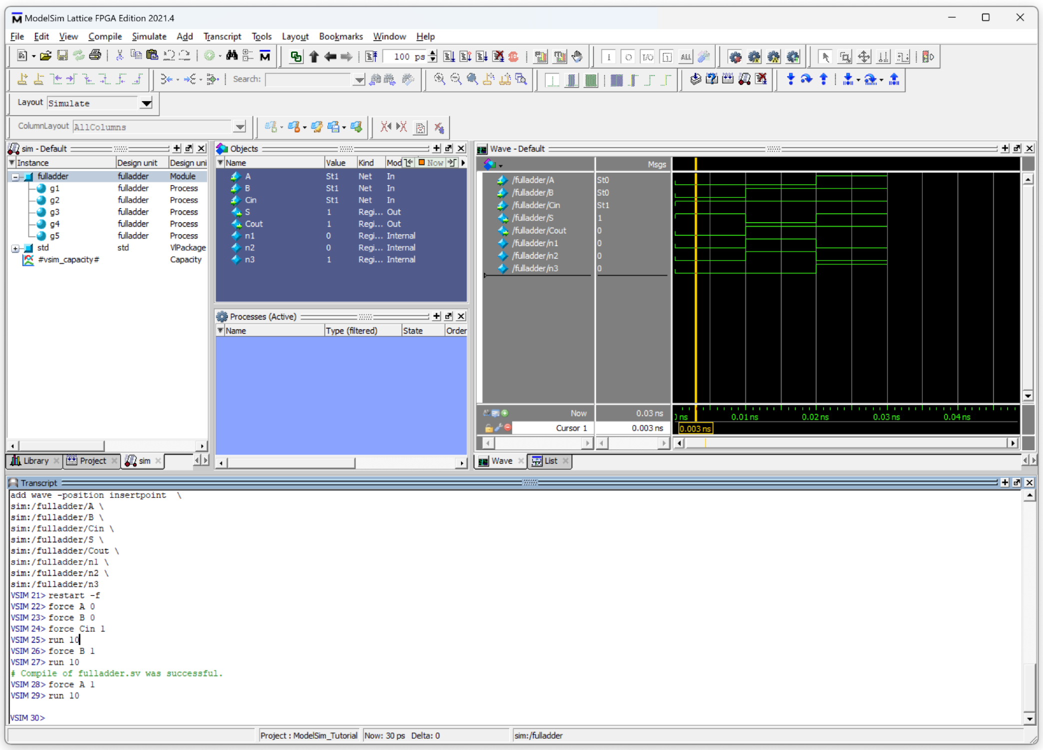
Copy Settings From: .1/modeltech/modelsim.ini

[✓ Copy Library Mappings] Reference Library Mappings

```
// fulladder.sv
// Structural Verilog full adder
// 5/26/22
// Josh Brake
// jbrake@hmc.edu

module fulladder(
 input   logic A, B, Cin,
 output  logic S, Cout
);

 logic n1, n2, n3;

 xor g1(n1, A, B);
 xor g2(S, n1, Cin);

 and g3(n2, n1, Cin);
 and g4(n3, A, B);

 or  g5(Cout, n2, n3);

endmodule
```

Next, we will simulate the file. First, compile your file via the Compile menu. Then, start the simulation using “Simulate > Start Simulation…”. You will see the GUI change and open up the Objects and Wave panes. You can always toggle windows on or off as desired using the View toolbar menu.

Next, add the selected symbols by selecting them in the Objects pane and adding them to the Wave pane either by using the right-click menu or dragging them. Next, to run the simulation, we need to initialize the input values which are floating (HiZ) to begin. We can do this by typing the command “force <Signal Name> <Signal Value>” in the Transcript pane. Force A, B, and Cin to some desired values of your choosing and then run the simulation for 10 time units by typing “run 10”. When you run the simulation, you will see the waves window advance the appropriate number of time points, displaying the values of the signals.



```
add wave -position insertpoint  \
sim:/fulladder/A \
sim:/fulladder/B \
sim:/fulladder/Cin \
sim:/fulladder/S \
sim:/fulladder/Cout \
sim:/fulladder/n1 \
sim:/fulladder/n2 \
sim:/fulladder/n3
VSIM 21> restart -f
VSIM 22> force A 0
VSIM 23> force B 0
VSIM 24> force Cin 1
VSIM 25> run 10
VSIM 26> force B 1
VSIM 27> run 10
# Compile of fulladder.sv was successful.
VSIM 28> force A 1
VSIM 29> run 10

VSIM 30>
```

Fill out a truth table for a full adder by evaluating the circuit schematic shown above for all combinations of A, B, and Cin and verify a few values by forcing them to get the hang of it.

Testbench Simulation

After testing about only two different inputs by manually forcing values you are probably already frustrated with the tediousness of running a simulation this way. Luckily, there is a much more efficient way to run simulations using testbenches.

A testbench is a separate module which generates simulated inputs and enables the generated output signals to be traced versus time. You've likely seen testbenches before (e.g., in E85) but you may have never written one from scratch by yourself before. While this is a bit daunting at first, the basic architecture of a testbench is straightforward. Before you proceed, you should review the HDL testbench section in Digital Design and Computer Architecture (Section 4.9 in the ARM edition) which covers several different styles of testbenches.

Note that testbenches include Verilog statements that are non-synthesizable (in other words, they cannot be mapped onto hardware). One example is the initial block, which runs only once at the start of a simulation. In this sense, testbenches use SystemVerilog a bit like a scripting language to enable designs to be easily tested and verified.

In this example we will create a testbench with automatic checking against a test vector file. This is the most powerful type of testbench and will be a good foundation for all the testbenches you will write for this class.

A testbench is composed of the following elements:

- Create and initialize the signals for the inputs, outputs, and internal signals of your modules.
- Instantiate the device under test (DUT)
- Generate a simulated clock signal. Even if your module is purely combinational, you will need this clock signal to control when the input test vectors are applied and the outputs checked.
- Load the test vectors and initialize the signals for the simulation (e.g., pulsing the reset line)
- Apply the inputs and check the resulting outputs using the test vectors. One simple way to do this is to apply the test vector on the rising edge of the clock and then check the outputs against the expect outputs on the negative edge of the clock.

Every HDL module that you write should have a testbench to accompany it. While this will cost you some time up front, it will save you loads of time and potential head injuries from banging your head against the wall wondering why your circuit is not working as you expect.

Creating the Testbench

Next, we will create the testbench. Create a new SystemVerilog file "fulladder_tb.sv" and add it to your project. Then, type the following code into the testbench.

```
`timescale 1ns/1ns
`default_nettype none
`define N_TV 8

module fulladder_tb();
 // Set up test signals
 logic clk, reset;
 logic a, b, cin, s, cout, s_expected, cout_expected;
 logic [31:0] vectornum, errors;
 logic [10:0] testvectors[10000:0]; // Vectors of format s[3:0]_seg[6:0]

 // Instantiate the device under test
 fulladder dut(.A(a), .B(b), .Cin(cin), .S(s), .Cout(cout));

 // Generate clock signal with a period of 10 timesteps.
 always
   begin
     clk = 1; #5;
     clk = 0; #5;
   end

 // At the start of the simulation:
 //  - Load the testvectors
 //  - Pulse the reset line (if applicable)
 initial
   begin
     $readmemb("fulladder_testvectors.tv", testvectors, 0, `N_TV - 1);
     vectornum = 0; errors = 0;
     reset = 1; #27; reset = 0;
   end
  // Apply test vector on the rising edge of clk
 always @(posedge clk)
   begin
       #1; {a, b, cin, s_expected, cout_expected} = testvectors[vectornum];
   end
  initial
 begin
   // Create dumpfile for signals
   $dumpfile("fulladder_tb.vcd");
   $dumpvars(0, fulladder_tb);
 end
  // Check results on the falling edge of clk
 always @(negedge clk)
   begin
     if (~reset) // skip during reset
       begin
         if (cout != cout_expected || s != s_expected)
           begin
             $display("Error: inputs: a=%b, b=%b, cin=%b", a, b, cin);
             $display(" outputs: s=%b (%b expected), cout=%b (%b expected)", s,
s_expected, cout, cout_expected);
             errors = errors + 1;
           end


       vectornum = vectornum + 1;

       if (testvectors[vectornum] === 11'bx)
         begin
           $display("%d tests completed with %d errors.", vectornum, errors);
           $finish;
         end
     end
   end
endmodule
```

Notes

- Functions preceded by a dollar sign (e.g., $readmemb, $display, $finish, etc.) are system functions.

The Testvector (.tv) File

The test vector file is a plaintext file with a single test vector on each line. Typically these are organized in the format <inputs>_<expected_outputs> (underscores are ignored but help to provide visual organization) but the specific format is up to you. You just need to make sure to assign the bits properly in your testbench module.

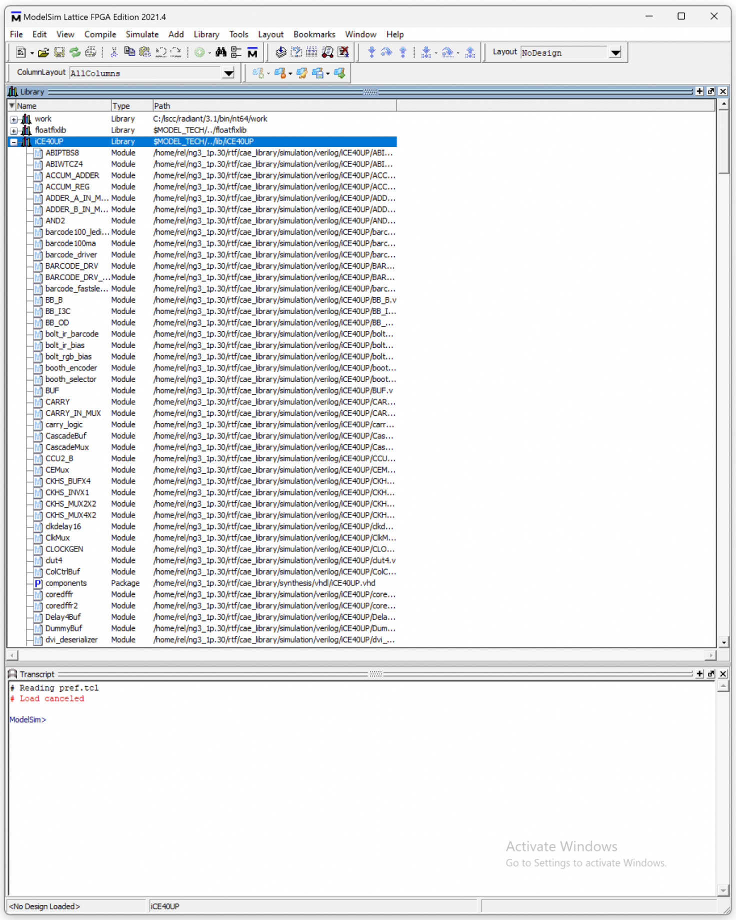
```
// fulladder_testvectors.tv
// Josh Brake
// jbrake@hmc.edu
// 5/26/22
//
// 5-bit vectors in binary.
// underscore is ignored
// // indicates comment
// a b cin _ s cout
000_00
001_10
010_10
011_01
100_10
101_01
110_01
111_11
```

Libraries

In many instances we want to leverage the iCE40 Technology Library. However, when we go to simulate these designs, we need to make sure to include the appropriate libraries so that ModelSim can find the correct design units when it goes to do the simulation.

When you install Lattice Radiant with ModelSim Lattice FPGA Edition, it comes with a set of installed libraries. For our purposes, we are most interested in the iCE40UP library and will need to include this in our simulation.

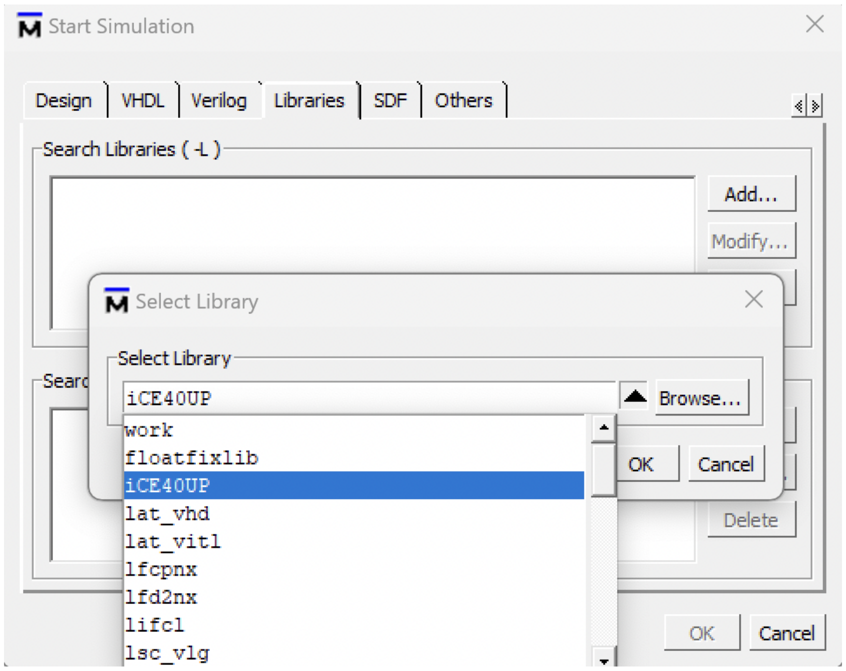
To do this, start the simulation by clicking Simulate > Start Simulation. Then navigate to the Libraries tab and add iCE40UP to the Search Libraries window using the "Add..." button.

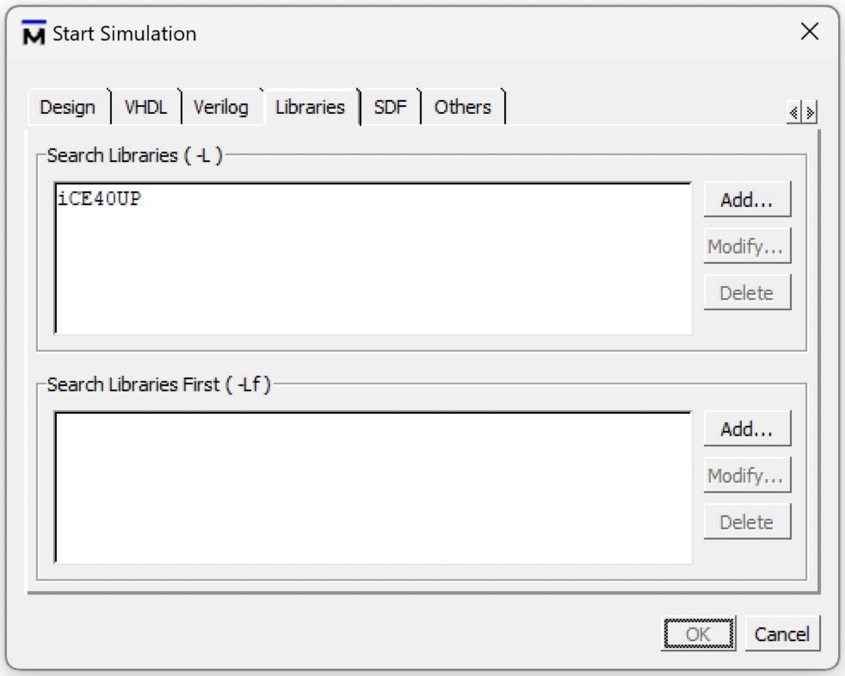

ModelSim Lattice FPGA Edition 2021.4
File Edit View Compile Simulate Add Library Tools Layout Bookmarks Window Help
Layout NoDesign
ColumnLayout AllColumns
Library
Transcript

Name	Type	Path
work	Library	C:/lscc/radiant/3.1/bin/nt64/work
floatfixlib	Library	$MODEL_TECH/../floatfixlib
iCE40UP	Library	$MODEL_TECH/../lib/iCE40UP
ABIPTBS8	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/ABI…
ABIWTCZ4	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/ABI…
ACCUM_ADDER	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/ACC…
ACCUM_REG	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/ACC…
ADDER_A_IN_M…	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/ADD…
ADDER_B_IN_M…	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/ADD…
AND2	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/AND…
barcode100_ledi…	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/barc…
barcode100ma	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/barc…
barcode_driver	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/barc…
BARCODE_DRV	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/BAR…
BARCODE_DRV_…	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/BAR…
barcode_fastsle…	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/barc…
BB_B	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/BB_B.v
BB_I3C	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/BB_I…
BB_OD	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/BB_…
bolt_ir_barcode	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/bolt…
bolt_ir_bias	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/bolt…
bolt_rgb_bias	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/bolt…
booth_encoder	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/boot…
booth_selector	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/boot…
BUF	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/BUF.v
CARRY	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/CAR…
CARRY_IN_MUX	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/CAR…
carry_logic	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/carr…
CascadeBuf	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/Cas…
CascadeMux	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/Cas…
CCU2_B	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/CCU…
CEMux	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/CEM…
CKHS_BUFX4	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/CKH…
CKHS_INVX1	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/CKH…
CKHS_MUX2X2	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/CKH…
CKHS_MUX4X2	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/CKH…
clkdelay16	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/clkd…
ClkMux	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/ClkM…
CLOCKGEN	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/CLO…
clut4	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/clut4.v
ColCtrlBuf	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/ColC…
components	Package	/home/rel/ng3_1p.30/rtf/cae_library/synthesis/vhdl/iCE40UP.vhd
coredffr	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/core…
coredffr2	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/core…
Delay4Buf	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/Dela…
DummyBuf	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/Dum…
dvi_deserializer	Module	/home/rel/ng3_1p.30/rtf/cae_library/simulation/verilog/iCE40UP/dvi_…

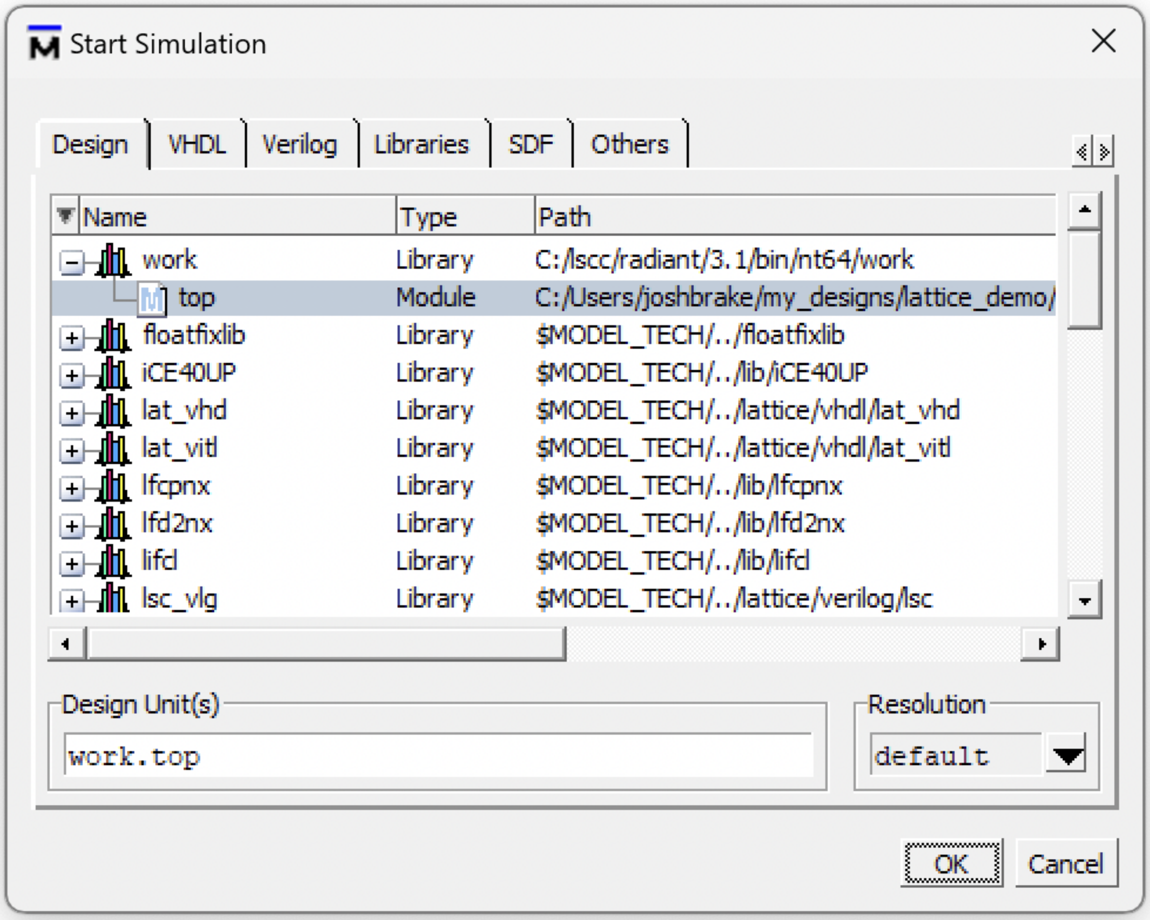
```
# Reading pref.tcl
# Load canceled

ModelSim>
```

<No Design Loaded> iCE40UP





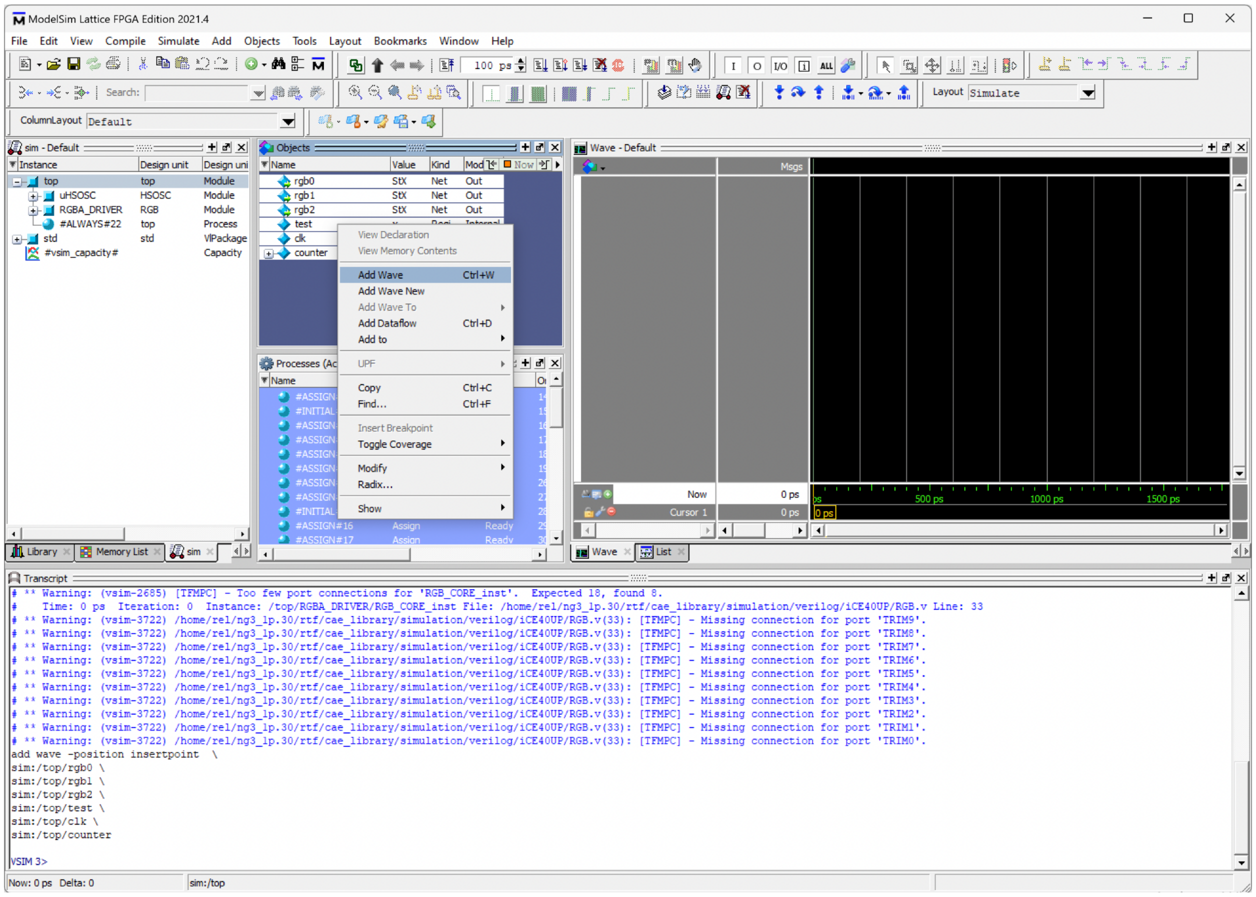
Then navigate to the "Design" tab, select the "top" module and then click OK. Your simulation should now start and open up the Wave window.



Name	Type	Path
work	Library	C:/lscc/radiant/3.1/bin/nt64/work
top	Module	C:/Users/joshbrake/my_designs/lattice_demo/
floatfixlib	Library	$MODEL_TECH/../floatfixlib
iCE40UP	Library	$MODEL_TECH/../lib/iCE40UP
lat_vhd	Library	$MODEL_TECH/../lattice/vhdl/lat_vhd
lat_vitl	Library	$MODEL_TECH/../lattice/vhdl/lat_vitl
lfcpnx	Library	$MODEL_TECH/../lib/lfcpnx
lfd2nx	Library	$MODEL_TECH/../lib/lfd2nx
lifcl	Library	$MODEL_TECH/../lib/lifcl
lsc_vlg	Library	$MODEL_TECH/../lattice/verilog/lsc

Design Unit(s): `work.top`

Resolution: `default`



```
# ** Warning: (vsim-2685) [TFMPC] - Too few port connections for 'RGB_CORE_inst'.  Expected 18, found 8.
#    Time: 0 ps  Iteration: 0  Instance: /top/RGBA_DRIVER/RGB_CORE_inst File: /home/rel/ng3_lp.30/rtf/cae_library/simulation/verilog/iCE40UP/RGB.v Line: 33
# ** Warning: (vsim-3722) /home/rel/ng3_lp.30/rtf/cae_library/simulation/verilog/iCE40UP/RGB.v(33): [TFMPC] - Missing connection for port 'TRIM9'.
# ** Warning: (vsim-3722) /home/rel/ng3_lp.30/rtf/cae_library/simulation/verilog/iCE40UP/RGB.v(33): [TFMPC] - Missing connection for port 'TRIM8'.
# ** Warning: (vsim-3722) /home/rel/ng3_lp.30/rtf/cae_library/simulation/verilog/iCE40UP/RGB.v(33): [TFMPC] - Missing connection for port 'TRIM7'.
# ** Warning: (vsim-3722) /home/rel/ng3_lp.30/rtf/cae_library/simulation/verilog/iCE40UP/RGB.v(33): [TFMPC] - Missing connection for port 'TRIM6'.
# ** Warning: (vsim-3722) /home/rel/ng3_lp.30/rtf/cae_library/simulation/verilog/iCE40UP/RGB.v(33): [TFMPC] - Missing connection for port 'TRIM5'.
# ** Warning: (vsim-3722) /home/rel/ng3_lp.30/rtf/cae_library/simulation/verilog/iCE40UP/RGB.v(33): [TFMPC] - Missing connection for port 'TRIM4'.
# ** Warning: (vsim-3722) /home/rel/ng3_lp.30/rtf/cae_library/simulation/verilog/iCE40UP/RGB.v(33): [TFMPC] - Missing connection for port 'TRIM3'.
# ** Warning: (vsim-3722) /home/rel/ng3_lp.30/rtf/cae_library/simulation/verilog/iCE40UP/RGB.v(33): [TFMPC] - Missing connection for port 'TRIM2'.
# ** Warning: (vsim-3722) /home/rel/ng3_lp.30/rtf/cae_library/simulation/verilog/iCE40UP/RGB.v(33): [TFMPC] - Missing connection for port 'TRIM1'.
# ** Warning: (vsim-3722) /home/rel/ng3_lp.30/rtf/cae_library/simulation/verilog/iCE40UP/RGB.v(33): [TFMPC] - Missing connection for port 'TRIM0'.
add wave -position insertpoint  \
sim:/top/rgb0 \
sim:/top/rgb1 \
sim:/top/rgb2 \
sim:/top/test \
sim:/top/clk \
sim:/top/counter

VSIM 3>
```

Note that you can also achieve the steps above by running "vsim work.top -L iCE40UP" in the command window.