

ECE275: Sequential Logic Circuits

Lab 1: Install Quartus and Program FPGA

Vikas Dhiman
Zafaryab Haider, Shihab Uddin Ahamad

August 26, 2023

Contents

1	Lab Overview	2
2	Installing Quartus	2
2.1	Downloading	2
2.2	Setup the USB Blaster Driver	3
2.3	Creating a New Project with the Quartus Software	3
2.4	Create a New Verilog File and Copy the Code	5
2.5	Run Compilation	7
2.6	Set Pin Assignments	8
2.7	Rerun Compilation and Program the FPGA	10
2.8	Completing the Lab	13
3	Part 1: Analyzing a Verilog Program Utilizing the ModelSim Software	13
3.1	Installing the ModelSim software	13
3.2	Create a Simple Verilog Program	13
3.3	Create a Testbench	13
3.4	Set Path to ModelSim	15
3.5	Run Simulation	15
3.6	Interact with the Simulation	15
3.7	Part 1 Completion	16
3.8	Part 2 Debugging	16
4	Questions	16
4.1	Question 1	16
4.2	Question 2	17
4.3	Question 3	17
4.4	Question 4	17
4.5	Question 5	17

1 Lab Overview

There are two part in this lab. For the first part, the goal of will be to install the Intel Quartus software, and then utilize it to program the provided Altera DE0 development FPGAs with an example Verilog program.

The installation can be somewhat finicky, so make sure to perform all of the steps and get the correct software version.

While there is a Linux version of Quartus, it has not been tested for this class, so use that at your own risk. There is no native Mac OS support.

For the second part, The goal is to utilize the ModelSim simulator to simulate your Verilog code without the use of the FPGA. This is an incredibly important part of FPGA design, as it can be very difficult to diagnose issues with your code when running on the FPGA. The simulator allows you to see specific states of logic with very tight timing constraints on inputs and outputs.

2 Installing Quartus

2.1 Downloading

We will be utilizing the Quartus II 13.1 software for this class. This version has been tested to be compatible with DE0 FPGA boards. You can download it from here: <https://www.intel.com/content/www/us/en/software-kit/666221/intel-quartus-ii-web-edition-design-software-version-13-1-for-windows.html>

You will need to select the combined tab to download the combined software so you will not have to download device support for the FPGA separately. Make sure you are downloading the 13.1 Web Edition. You will need to create a free account to download the software.

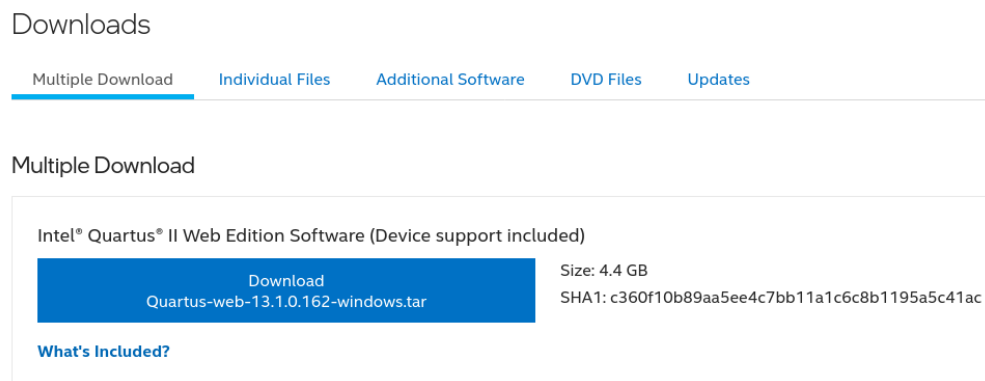


Figure 1: Select the Combined Installation and Quartus Version to Download

Important Note

The Intel website is very finicky, and may try to change the version you have selected after you login, or if you click around on the page. Double check your selection before downloading.

Once the download is finished, install the Quartus II software, using the default options on each page. Take note of your installation path, you will need it for the next step.

2.2 Setup the USB Blaster Driver

Instructions from http://www.terasic.com.tw/wiki/Altera_USB_Blaster_Driver_Installation_Instructions

For Windows 10:

1. Use the provided USB cable to connect the DE0 board to your computer
2. Open your control panel, and then go to devices and printers
3. Under Unspecified, USB Blaster should be listed. Right click on this and then select Properties
4. Select the Hardware tab and select Properties
5. A new window should pop up with the General tab already selected. Select Change Settings.
6. Again a new window should pop up with the General tab already selected. Select Update.
7. Select Browse my computer for driver software.
8. Find <Path to Quartus II installation>\quartus\drivers\ Check notes below for additional path information
9. Select OK. Make sure the proper path was selected then select Next.
10. If the Windows security window pops up check the Always trust software from “Altera Corporation” box and select Install.
 - Note 1: Your Altera file is located at the location you selected when you first installed Quartus. The location listed in this document is the default location) C:\altera\13.0sp1
 - Note 2: Stop at the drivers folder, i.e., do NOT go deeper by opening a folder within the drivers folder)

2.3 Creating a New Project with the Quartus Software

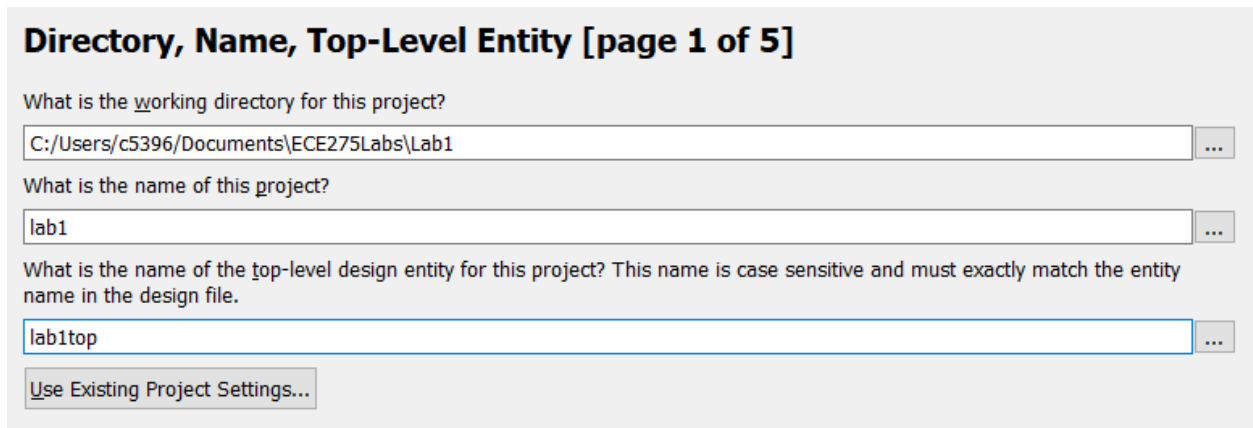
The Quartus software should be in your start menu programs list in a folder named Altera 13.1 Web Edition. Run the Quartus II 13.1 software in this folder.

After the software is started, you will need to create a new project. You can do this by selecting “Create a New Project” in the initial pop up window, or by going to the file menu at the top left, going to new, and then selecting the new project option.

In the project setup wizard, you will need to set up the project name hierarchy, and select the correct FPGA model.

The first window should ask for the project working directory, project name, and top-level name. Try to use underscores instead of spaces in the project name and path, and avoid use of any special characters. The project save path will be a folder that contains multiple files that relate to the project, so it is a good idea to create a unique folder in a larger folder structure for each lab section. For example if a lab had multiple parts with unique code, the path could be <Path to Your Documents Folder>\ECE275Labs\Lab1\Lab1Part1 for the first part, and then <Path to Your Documents Folder>\ECE275Labs\Lab1\Lab1Part2 for the second part. The next box asks for the name of the project. Give it a short relevant name. The last box asks for the top level, generally an easy name to use is the project name with top added at the end (ex. Project: lab2part1, Top Level: lab2part1top). This makes it easy to keep track of your top level module name (you

will use the top level name similar to how main is used for the main function in C).



Directory, Name, Top-Level Entity [page 1 of 5]

What is the working directory for this project?

C:/Users/c5396/Documents\ECE275Labs\Lab1 ...

What is the name of this project?

lab1 ...

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.

lab1top ...

[Use Existing Project Settings...](#)

Figure 2: Set up the project naming hierarchy

The next step will ask you to add files, you can just hit next here.

Next you will need to select the correct FPGA model. You can use the family dropdown to select Cyclone III. Then, in the available devices window, find the model name:

FPGA Model Name: EP3C16F484C6

This information is available on the chip of the FPGA board. After selecting the correct model name, you can hit Finish on the New Project Wizard.

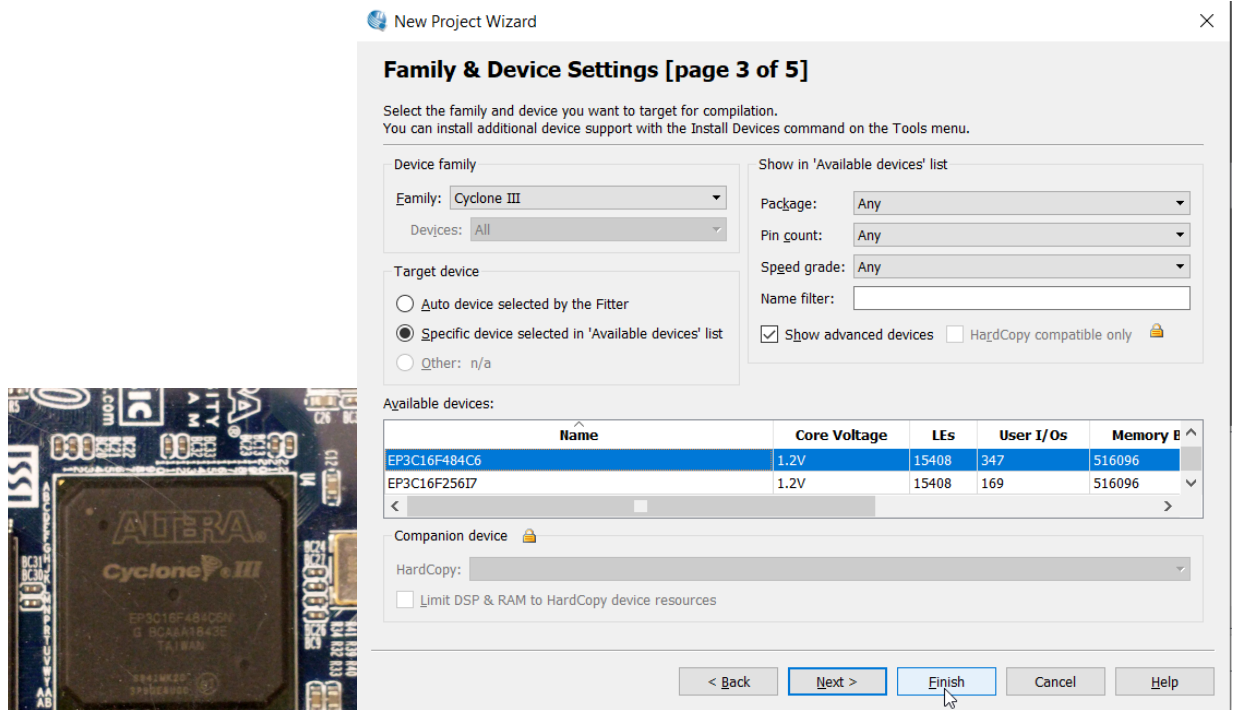


Figure 3: Select the correct FPGA model, EP3C16F484C6

2.4 Create a New Verilog File and Copy the Code

Verilog is known as a Hardware Description Language (HDL). In future labs, you will be writing your own code in Verilog, but for today's lab it is being provided for you.

To create a new Verilog file, go to the file menu in the top left, go to new, and then in the Window that pops up, select Verilog HDL File

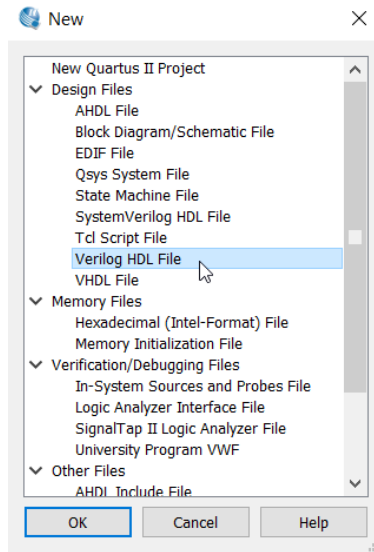


Figure 4: Create a new Verilog HDL File

This will create a new blank file, where you can copy and paste the following code. You will need to change the module name at the top to the name you chose for your top level in the wizard.

```

1 // module in verilog is like a function in C
2 module lab1top(
3     // input specifies the input wires to a circuit
4     input [9:0] SW,
5     // output specifies the output wires from a circuit
6     // In C language these will be return variable
7     output [9:0] LEDG
8 );
9     // assign keyword creates continuous assignment.
10    // It connects the two variables with a wire.
11    // Whenever SW is updated, then LEDG is updated (instantly).
12    assign LEDG[9:0] = SW[9:0];
13 endmodule // Verilog

```

After pasting the code, hit control+s to save the file. You can leave the file name as your top-level name.

2.5 Run Compilation

The next step is to run compilation. You will have to run this before downloading to the FPGA anytime you make any code updates, or pin assignment changes. The compilation creates the files that are downloaded to the FPGA from your project files.

You can run compilation by hitting the purple play icon in the center top of the toolbar, or by going to the processing menu at the top, and selecting Start Compilation.

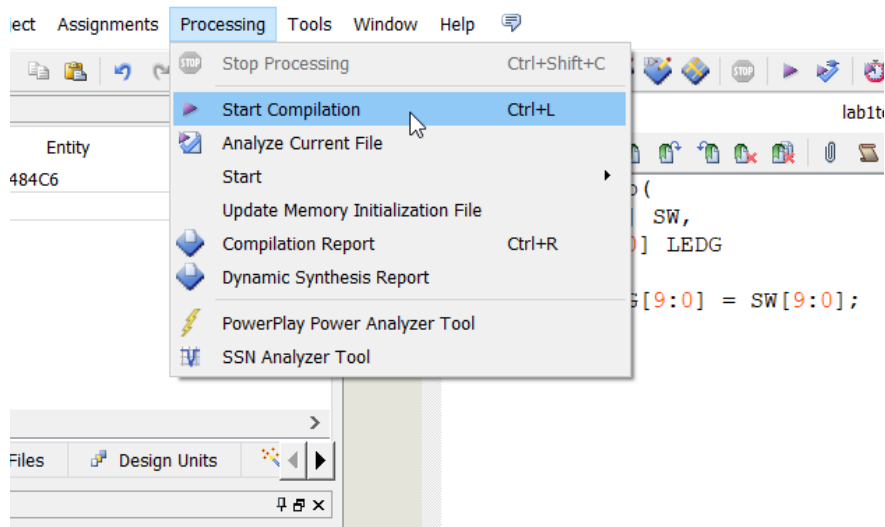


Figure 5: Start Project Compilation

When the compilation finished, a box should pop up indicating there were 12 warnings. This is ok, as long as you do not have any errors. Errors are higher level than warnings, and would indicate you did not copy something correctly with the code, or you forgot to change the module name to your top-level name.

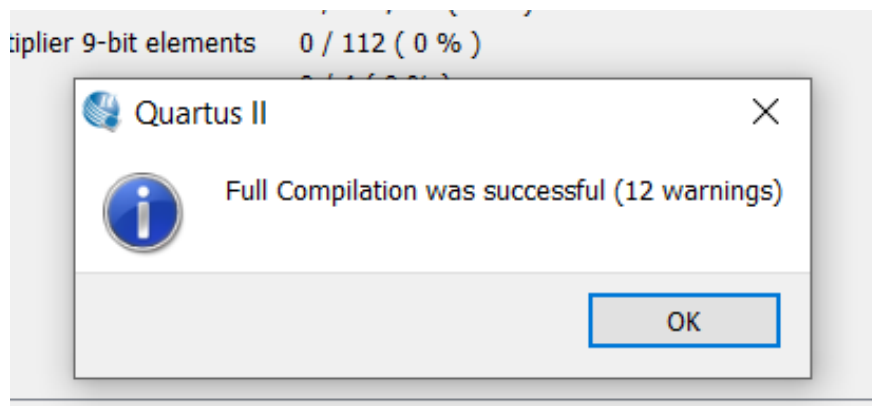


Figure 6: Compilation was successful, even with warnings

2.6 Set Pin Assignments

The last step before downloading to the FPGA is to set the pin assignments. This essentially references pins on the FPGA to their real world device connection. The pins are laid out in a grid on the FPGA, so are referenced with names such as AA12. This step will associate the switches and LED's that are utilized in the Verilog program to pins they are physically connected to on the FPGA. Detailed information on the pins can be found in the user manual of Altera DE0 User Manual <https://www.terasic.com.tw/cgi-bin/page/archive.pl?Language=English&CategoryNo=56&No=364&PartNo=4>

The easiest way to set pin assignments is to use their unique names as the variable names in the verilog code, and then use pre-made qsf files to create the references. The .qsf file for the board in this class will be provided to you. For this lab, you only need a few lines from the qsf file, and those are provided below:

```
set_location_assignment PIN_B1 -to LEDG[9]
set_location_assignment PIN_B2 -to LEDG[8]
set_location_assignment PIN_C2 -to LEDG[7]
set_location_assignment PIN_C1 -to LEDG[6]
set_location_assignment PIN_E1 -to LEDG[5]
set_location_assignment PIN_F2 -to LEDG[4]
set_location_assignment PIN_H1 -to LEDG[3]
set_location_assignment PIN_J3 -to LEDG[2]
set_location_assignment PIN_J2 -to LEDG[1]
set_location_assignment PIN_J1 -to LEDG[0]
set_location_assignment PIN_D2 -to SW[9]
set_location_assignment PIN_E4 -to SW[8]
set_location_assignment PIN_E3 -to SW[7]
set_location_assignment PIN_H7 -to SW[6]
set_location_assignment PIN_J7 -to SW[5]
set_location_assignment PIN_G5 -to SW[4]
set_location_assignment PIN_G4 -to SW[3]
set_location_assignment PIN_H6 -to SW[2]
set_location_assignment PIN_H5 -to SW[1]
set_location_assignment PIN_J6 -to SW[0]
```

You will need to copy and paste these lines into the Quartus Console for your project, and then hit enter to run them. The Quartus console can be found below your verilog file in the Quartus window. If it is not showing, you may need to go to the view file menu, go to Utility Windows, and check if the Tcl Console is activated.

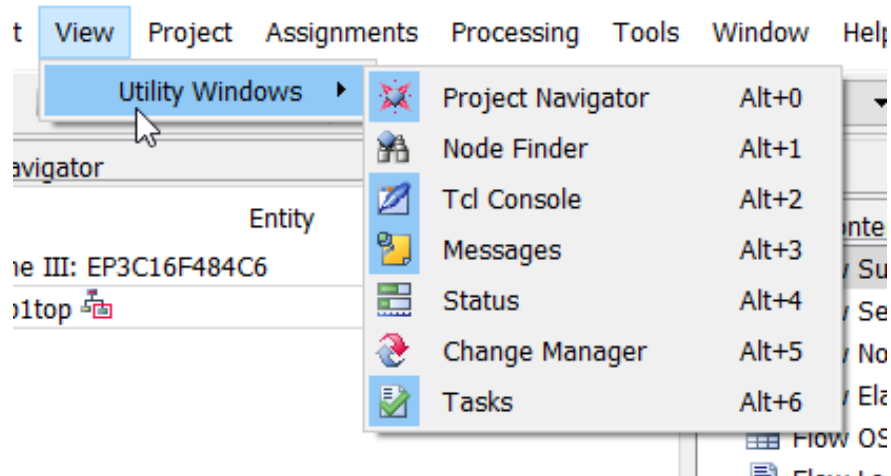


Figure 7: Check if the Tcl Console is Activated

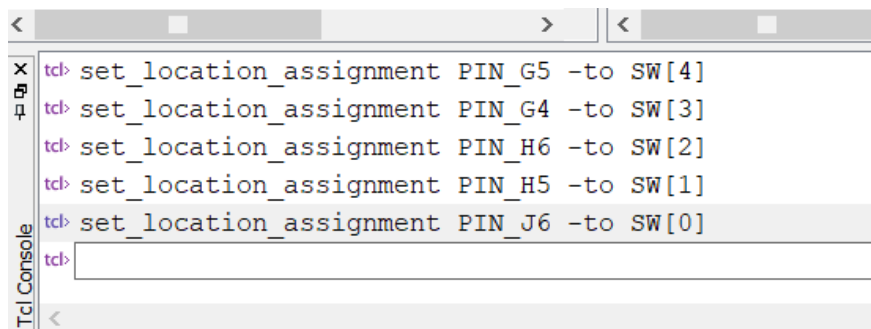


Figure 8: Tcl Console After Running Pin Assignments

You can check if the pin assignments were correctly run by opening the pin planner. This can be found in the assignments menu at the top. Your pins should match the pin layout shown in the picture.

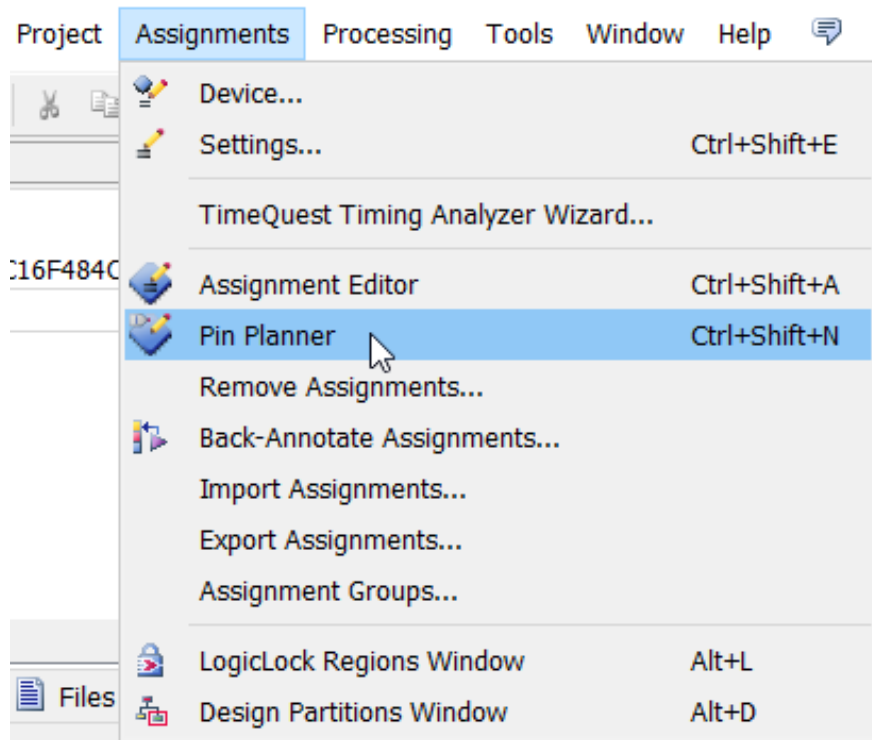


Figure 9: Open the Pin Planner

out	LEDG[9]	Output	PIN_B1	1	B1_N0	PIN_F10	2.5 V (default)	8mA (default)	2 (default)
out	LEDG[8]	Output	PIN_B2	1	B1_N0	PIN_V12	2.5 V (default)	8mA (default)	2 (default)
out	LEDG[7]	Output	PIN_C2	1	B1_N0	PIN_N22	2.5 V (default)	8mA (default)	2 (default)
out	LEDG[6]	Output	PIN_C1	1	B1_N0	PIN_E13	2.5 V (default)	8mA (default)	2 (default)
out	LEDG[5]	Output	PIN_E1	1	B1_N0	PIN_A8	2.5 V (default)	8mA (default)	2 (default)
out	LEDG[4]	Output	PIN_F2	1	B1_N0	PIN_L6	2.5 V (default)	8mA (default)	2 (default)
out	LEDG[3]	Output	PIN_H1	1	B1_N1	PIN_AB20	2.5 V (default)	8mA (default)	2 (default)
out	LEDG[2]	Output	PIN_J3	1	B1_N1	PIN_AB16	2.5 V (default)	8mA (default)	2 (default)
out	LEDG[1]	Output	PIN_J2	1	B1_N1	PIN_U14	2.5 V (default)	8mA (default)	2 (default)
out	LEDG[0]	Output	PIN_J1	1	B1_N1	PIN_H12	2.5 V (default)	8mA (default)	2 (default)
in	SW[9]	Input	PIN_D2	1	B1_N0	PIN_A5	2.5 V (default)	8mA (default)	
in	SW[8]	Input	PIN_E4	1	B1_N0	PIN_AB14	2.5 V (default)	8mA (default)	
in	SW[7]	Input	PIN_E3	1	B1_N0	PIN_N21	2.5 V (default)	8mA (default)	
in	SW[6]	Input	PIN_H7	1	B1_N0	PIN_B14	2.5 V (default)	8mA (default)	
in	SW[5]	Input	PIN_J7	1	B1_N1	PIN_C10	2.5 V (default)	8mA (default)	
in	SW[4]	Input	PIN_G5	1	B1_N0	PIN_M6	2.5 V (default)	8mA (default)	
in	SW[3]	Input	PIN_G4	1	B1_N0	PIN_AA20	2.5 V (default)	8mA (default)	
in	SW[2]	Input	PIN_H6	1	B1_N0	PIN_AA14	2.5 V (default)	8mA (default)	
in	SW[1]	Input	PIN_H5	1	B1_N0	PIN_R15	2.5 V (default)	8mA (default)	
in	SW[0]	Input	PIN_J6	1	B1_N0	PIN_C15	2.5 V (default)	8mA (default)	

Figure 10: Pin layout in Quartus Pin Planner

2.7 Rerun Compilation and Program the FPGA

Make sure to first rerun the compilation using the purple play icon or start compilation in the processing menu. After that is finished, go to the tools menu, and select the programmer option.

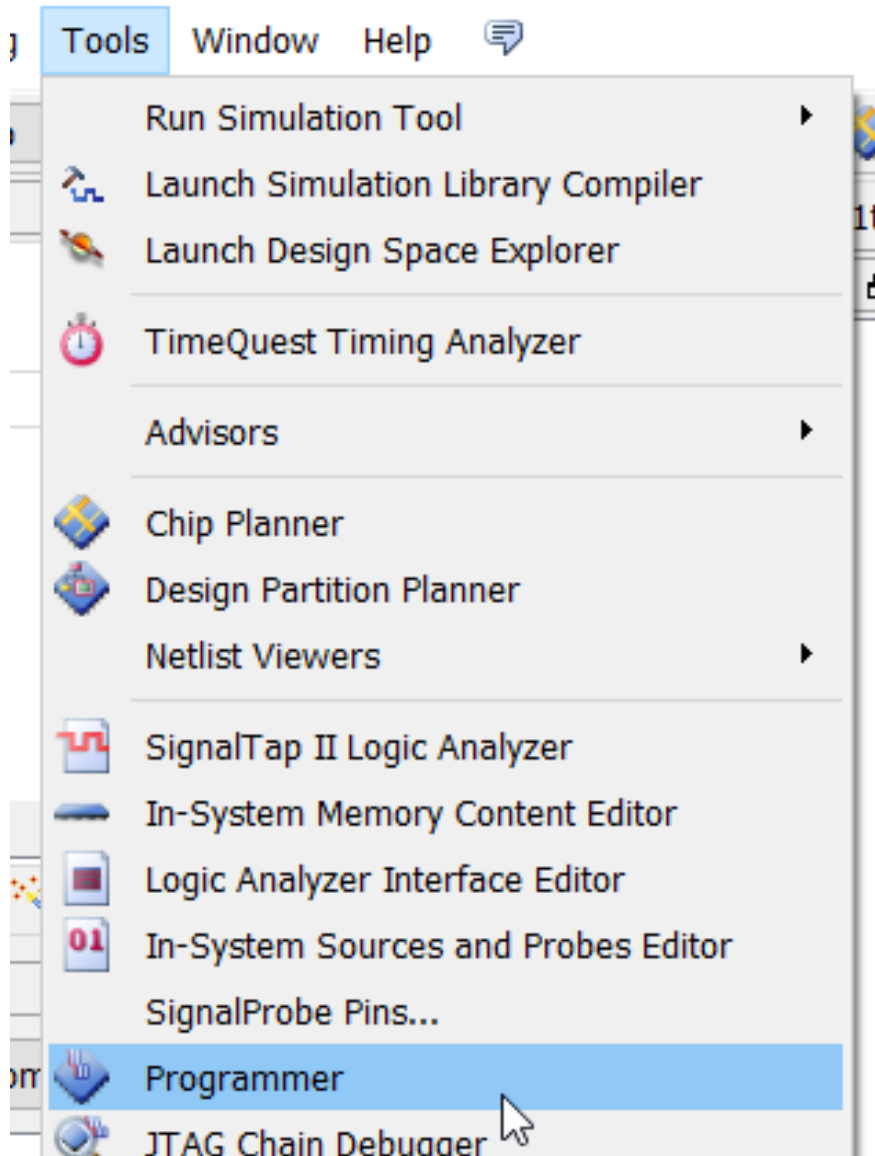


Figure 11: Open the Quartus Programmer

In the programmer window, there is an option for hardware setup towards the top left. Click on it to open the hardware setup window. Here you will have to select the USB Blaster option. If that is not available, make sure you properly completed the step of installing the USB-Blaster driver, and that the FPGA is on and connected to your computer through USB. Leave the JTAG programming mode selected in the main programmer window.

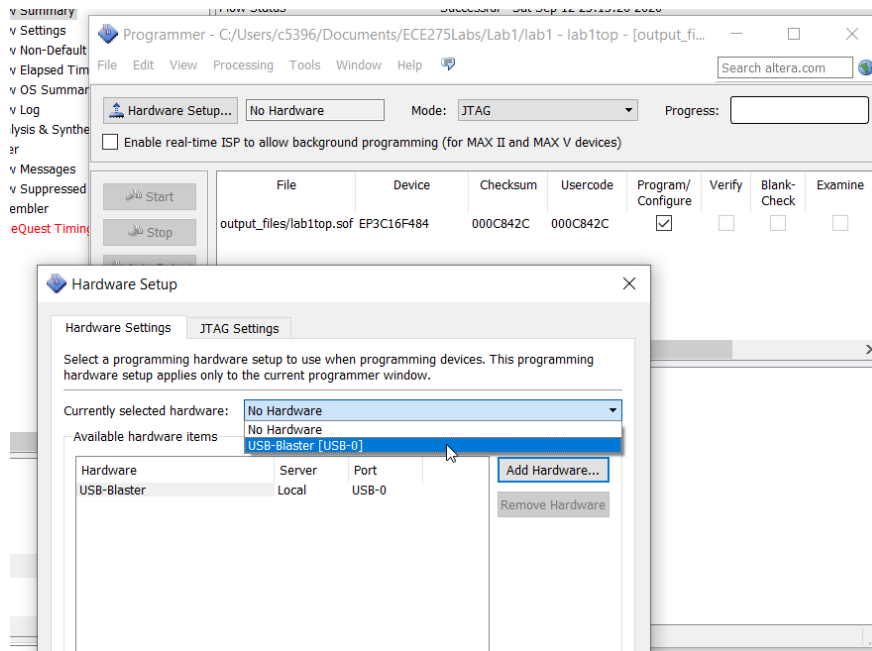


Figure 12: Select the USB-Blaster

At this point, you should be able to program your FPGA by selecting the output files .sof file in the programmer window, and then hit Start to program the FPGA. There is a confusing switch on the FPGA labeled RUN and PROG next to the 7-Segment LEDs. This must be in the RUN position to program the board when using JTAG. If the programmer says programming failed, check to see if that is in the RUN position.

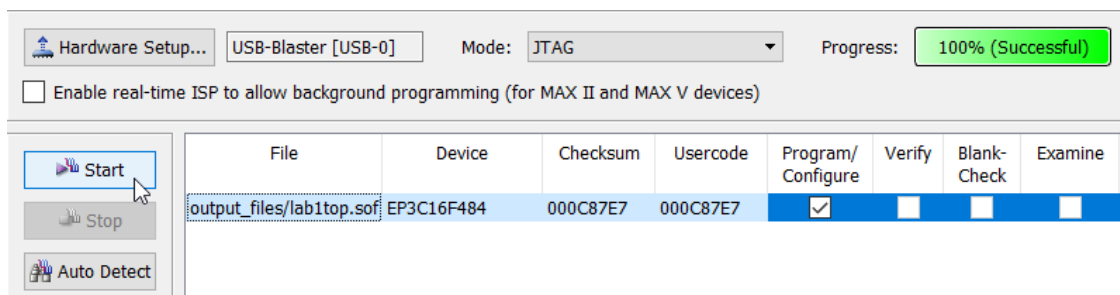


Figure 13: Program the FPGA

If your programmer window does not show any file in the middle to select, you will have to add your compiled project file. You can do this by selecting Add File, and then in the window the pops up double click on the output files folder, and then double clicking on your project's .sof file inside.

2.8 Completing the Lab

After programming the FPGA, you should see that if you change the position of any of switches on the FPGA (SW0-SW9) it should illuminate or turn off the corresponding green LED. Please review the Verilog code so you can understand why this is accomplished. After this is working correctly, the lab instructor or TA will ask you the three questions in the questions section at the end of the lab. The answers to these questions are in the lab writeup, be prepared to answer them.

3 Part 1: Analyzing a Verilog Program Utilizing the ModelSim Software

3.1 Installing the ModelSim software

You will need to use the Altera Modelsim software for this lab. You should first check if you already have it installed from the initial installation of Quartus at the beginning of the semester. You can check in the Windows "Add or remove programs" section if Modelsim is installed. If it is not, you can download it from <https://www.intel.com/content/www/us/en/software-kit/666221/intel-quartus-ii-web-edition-design-software-version-13-1-for-windows.html>. After the download is finished, install the software. Pay attention to the installation directory question, as you will need this later. I left it as the default C:\altera\13.1 and will use that for an example later in this write-up. If you modify it, keep that in mind for specifying the Modelsim location later.

3.2 Create a Simple Verilog Program

Create a new Quartus project, and utilize the Verilog module shown below from your top level. This should be the same code from the end of the lab last week. Instantiate the module with SW[0] as your clock, SW[1] as R, SW[2] as S, and LED[0] as Q. Test the code on your FPGA board to make sure it is working before moving on to the next step. It should set the Q output on the rising edge of the clock pulse if S is set and reset the Q output on the rising edge of a clock pulse if R is set.

```
1 // module in verilog is like a function in C
2 module lab1top(
3     // input specifies the input wires to a circuit
4     input [9:0] SW,
5     // output specifies the output wires from a circuit
6     // In C language these will be return variable
7     output [9:0] LEDG
8 );
9     // assign keyword creates continuous assignment.
10    // It connects the two variables with a wire.
11    // Whenever SW is updated, then LEDG is updated (instantly).
12    assign LEDG[9:0] = SW[9:0];
13 endmodule // Verilog
```

3.3 Create a Testbench

A testbench is essentially a Verilog module that sets the timing for simulating the Verilog code. Normally you would create the testbench as a separate Verilog file (as we learned last week how to include external Verilog files/modules in your code), but in this case, feel free to keep it in the same file as your top-level for easy troubleshooting. If you are interested in learning more about test benches outside this lab or are trying to troubleshoot this lab, you can read more about Modelsim test benches in this Intel document: <https://community.intel.com/t5/Intel-Quartus-Prime-Software/ModelSim-testbench-tutorial/m-p/1295002/thread-id/69901>

```

1 // Timescale sets the duration of one clock with precision
2 // Here 1ns is the duration of a single timestep and 1ps is the precision
3 `timescale 1ns / 1ps
4 module testbench ( );
5     // Intialize a clock variable to keep track of time
6     reg Clk;
7     // Intialize a variable for switches using 10 bit binary notation.
8     reg [9:0] SW;
9     // Create wire for output LEDG
10    wire [9:0] LEDG;
11
12    // Connect the switch and LED to simulated module lab1top
13    lab1top lt1 (SW, LEDG);
14
15    // Intial block is executed when the circuit starts
16    initial begin
17        Clk <= 0; SW <= 0;
18        // After a delay of 5 timesteps set SW to the following value using 10 bit binary notation
19        #5 SW <= 10'b00_0001_0001;
20        // After another delay of 5 timesteps set SW to the following value 10 bit binary notation
21        #5 SW <= 10'b01_0011_0001;
22    end
23
24    // Always block is like an infinite while loop
25    always begin
26        // Every one 1 timestep invert the clock signal
27        #1 Clk <= ~Clk;
28    end
29 endmodule

```

I have written the testbench for you to use in the Verilog code shown above. The first line, “timescale 1ns / 1ps” sets the time deltas for changing input values at 1ns, and simulation precision at 1ps. Setting a lower simulation precision can give higher accuracy but increase simulation time. The inputs for simulation are then defined as registers, and outputs as wires. Next, the modules to be simulated are instantiated. In this case, that is the ‘lab1top’ module. The next section contains the initial, begin, and end lines. The code inside sets the initial values for the inputs and then the times for them to be modified based on the timescale value. In the code shown in the example, the Clk, SW inputs are all 0. 5ns later, the SW bit is modified to a 1. 5ns after that, the SW bit is also modified to a new value. This continues through a reset cycle as well. This means that when the simulation is run, the LEDG output should turn on after 5ns and then turn off 5ns later.

3.4 Set Path to ModelSim

To utilize your testbench, you will need to tell Quartus what simulator to use and its location. In Quartus, go to the tools menu, and then towards the bottom, select options. In the window that appears in the top left under general, select the option for EDA Tool Options. Here you will find ModelSim-Altera towards the bottom. Click the three-dot icon to the right of the ModelSim-Altera, and then browse to your ModelSim installation directory. You will need to go deeper into the directory for the path it is looking for. In my case, that was C:\altera\13.1\modelsim_ase\win32aloem. You would have to modify this path if you selected a different installation directory for your ModelSim installation.

3.5 Run Simulation

After you have selected your Modelsim installation, you can move on to the actual simulation. In Quartus, go into the tools menu, go to the Run Simulation Tool section, and choose the RTL Simulation option. If you get an error, you likely have not installed ModelSim or set your path correctly to your ModelSim installation.

3.6 Interact with the Simulation

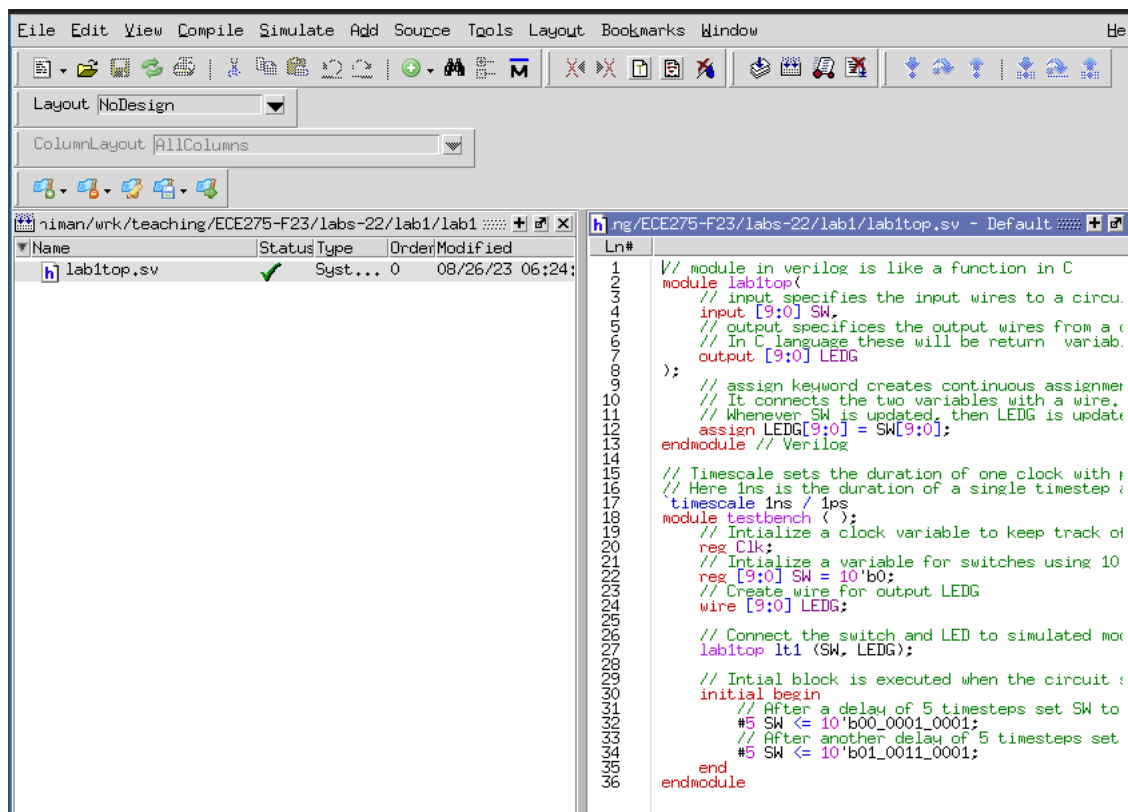


Figure 14: ModelSim view with testbench

The last step should have opened a separate window, ModelSim Altera.

By default, none of the waves you want to view will be selected, and your testbench will likely not be selected (Figure 16). To select your testbench, you must expand the work option under the libraries on the left. As long as you have run compilation on your top level, the testbench should show under the work section. Double-click on it to set it as active. Now under "Objects" and processes, you should see the inputs and outputs from the testbench. Click on any one of Clk, SW and LEDG. Use Ctrl+A to select all, right click, and choose the option to "Add Wave".

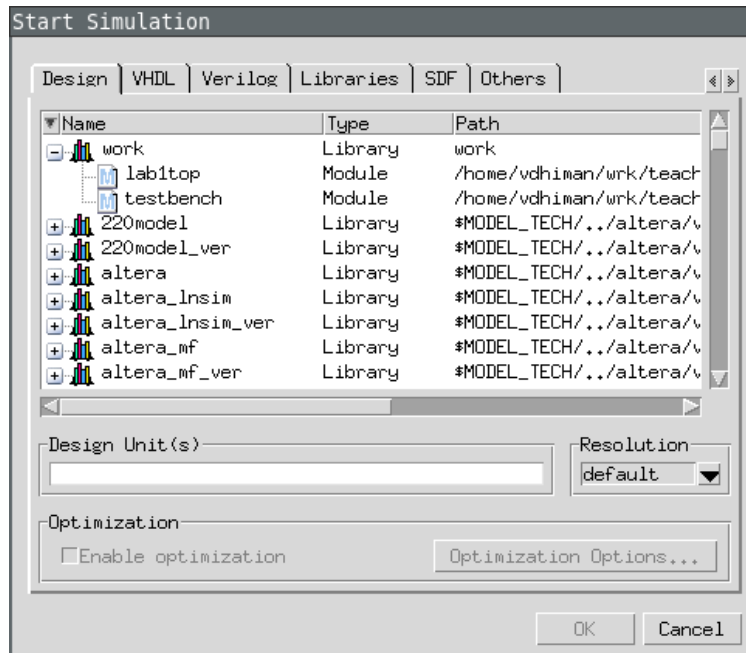


Figure 15: ModelSim Start Simulation interaction. Click on work, then testbench.

You should now see a wave graph on the right, but it will not contain data until you actually run the simulation (Figure 17).

The option to run-time steps of the simulation is at the top of the ModelSim program, next to a box that says "100ps". This is the time step value the ModelSim simulation will simulate each time you hit the run option ("Simulate-;Run-;Run 100") (Figure 18).

This is significantly shorter than the amount of time before the first input will change state (100ps vs. 10ns). You can either increase this value and use the run button next to the time (or "Simulate-;Run-;Run 100"). To see all the data easily, right-click on the graph and choose the zoom full option. With this, you should see the full waveforms that show a the waveforms as shown in Figure 19. Use the "Simulate-;Restart.." to restart the simulation from 0ns.

3.7 Part 1 Completion

Modify your testbench file to use different time values for the Clk, SW, and LEDG. Show your TA how this affects your waveform output in the simulation, and explain to them the changes you made.

3.8 Part 2 Debugging

Recall from your C-language class what is a debugger and a breakpoint? Find a way to add breakpoint to stop at the line 12 of the module lab1top. Use "Simulate-;Run-;Continue" to keep stopping at the breakpoint, every time SW changes. Look at the value of LEDG in the Objects pane. What do you observe? What does each button on the "Step toolbar" (Figure 20) do?

4 Questions

4.1 Question 1

When do you have to run or rerun compilation for a Quartus project?

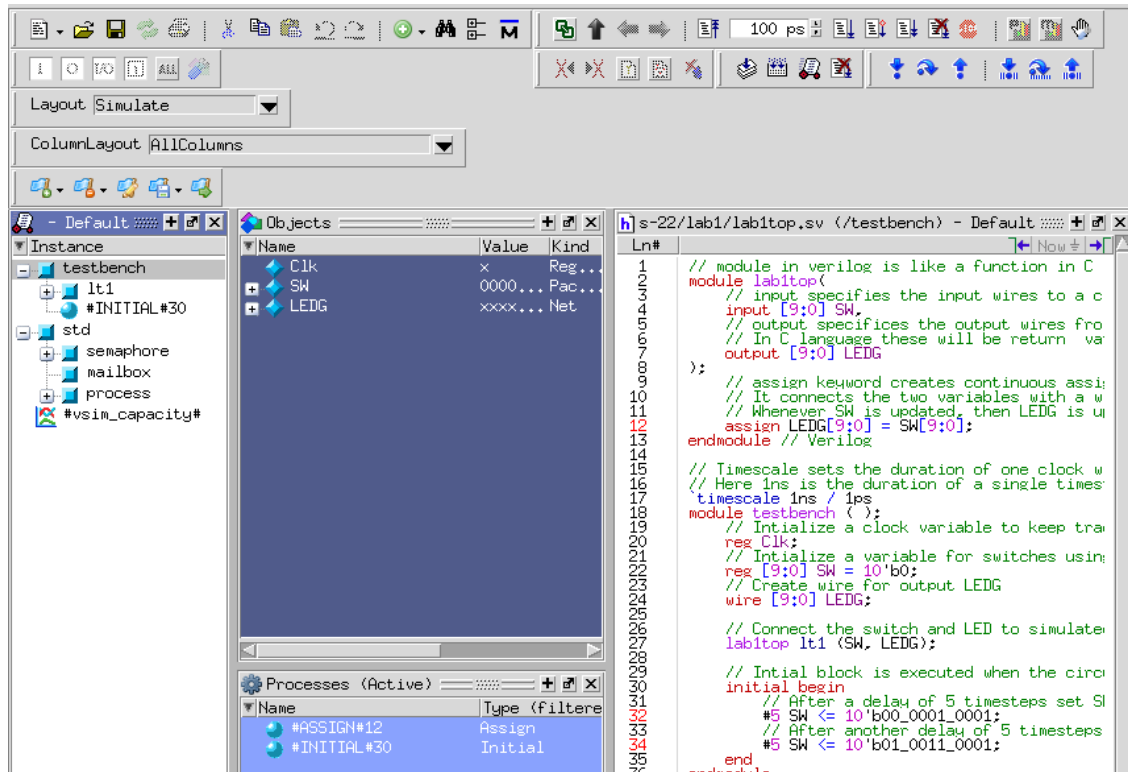


Figure 16: ModelSim before the waves are selected.

4.2 Question 2

After copying commands from a .qsf file to assign FPGA pins to the correct real world devices, where would you navigate in Quartus to check if the pins were assigned properly?

4.3 Question 3

What position should the RUN/PROG switch be in on the DE0 boards when programming with JTAG?

4.4 Question 4

What hardware description language will we be using for labs in this course?

4.5 Question 5

Why should you create separate folders for each Quartus project, instead of saving all of the projects in the same folder?

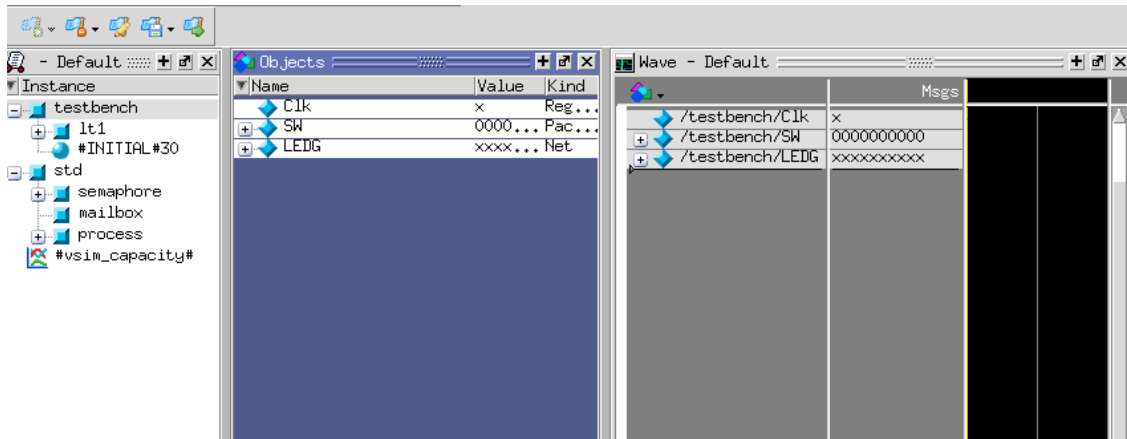


Figure 17: ModelSim after the waves are added.



Figure 18: The time duration of the simulation to run, by default 100ps

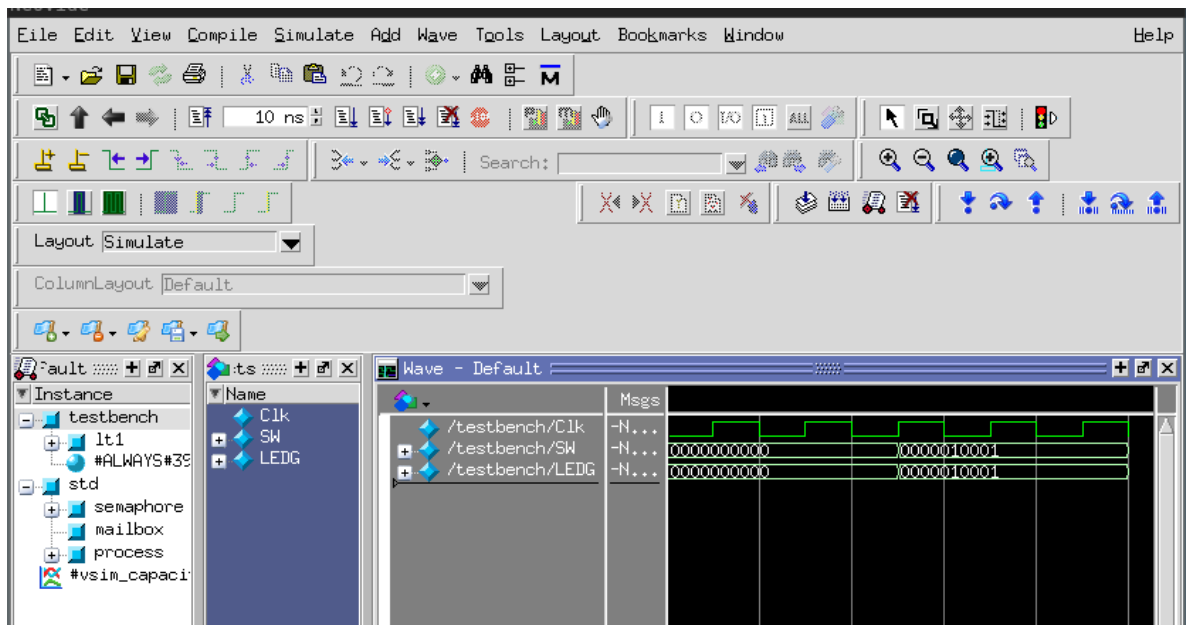


Figure 19: ModelSim after running the simulation for 10ns



Figure 20: ModelSim Step toolbar