Embedded Systems Lab 3: Signals and Simulation

Introduction

This lab is intended to introduce the student to two useful tools in the development of FPGA designs.

SignalTap is an Altera tool that emulates a Logic Analyser that's able to hook in to any signal inside the FPGA. The student may set up trigger events, sample rates and signals of interest in order to get an inside look at how their code is actually performing inside the device. This is much like one would do in simulation, but allows real world inputs and outputs to still operate as expected.

ModelSim is the simulator tool that is bundled with Quartus. It provides the usual suite of simulation abilities that the student may be familiar with from the prerequisite course. Like other simulators, the student may write a special Verilog program called a Testbench file whose purpose is to run a target piece of code through a sequence of inputs and check the corresponding outputs. In ModelSim, the student may also manually enter input conditions and immediately see the outputs; for example, one may change a switch input state then see how the change propagates through the design. It is this second mode that we will explore in today's lab.

The student is likely to benefit from using ModelSim in the development of their major project as it allows rapid checking of the correctness of logic without a full compile/download/test cycle. In a big project, this cycle may take as much as 30 minutes whereas a simulation run may take only 1-2.

The student may also find SignalTap useful in the major project, as that project is very likely to interface with hardware such as the VGA card, cameras, Audio I/O etc. These peripherals are very hard to check in simulation, so simply inserting a SignalTap instance watching the signals and monitoring what does on allows accurate detection of bugs in the code.

Part 1: SignalTap

SignalTap is a piece of software from Altera that allows you to monitor the state of your code as it's actually executing on the board. Much like a Logic Analyser or Oscilloscope, SignalTap has a concept of "triggers". A trigger is a condition that must be met for SignalTap to update its display. In the exercise below, we will trigger on user-initiated events such as flicking the switches. In a more complex project, one would choose a more relevant trigger condition to the problem at hand.

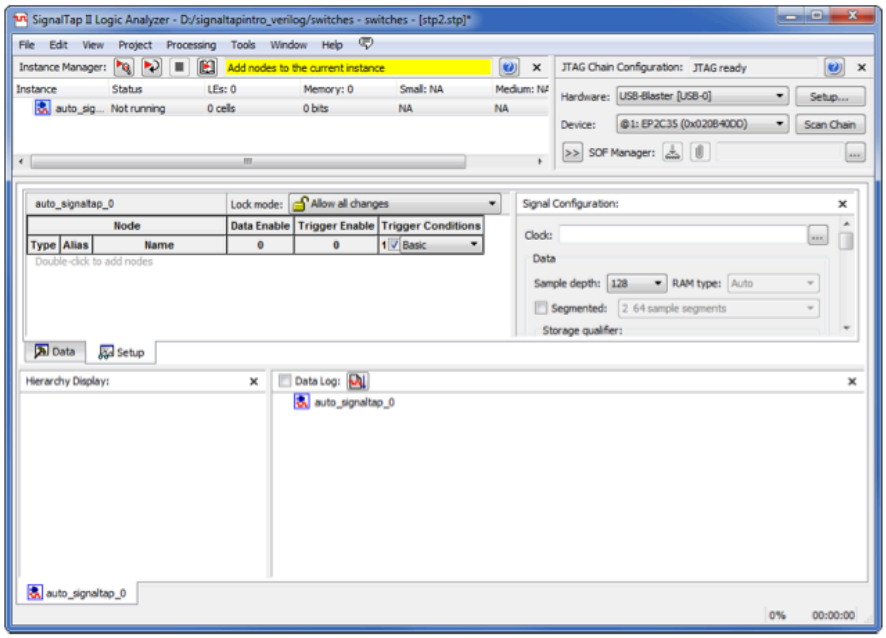
We will illustrate its functionality with a very simple example. Create a new Quartus Project with the usual settings for the DE2 board and enter the following code:

```
module signaltap(SW, CLOCK_50, LEDR);
input [7:0] SW;
input CLOCK_50;
output reg [7:0] LEDR;

always @(posedge CLOCK_50)
      LEDR <= SW;

endmodule
```

Import the pin assignments from the standard CSV file and compile. Verify the functionality on the board.

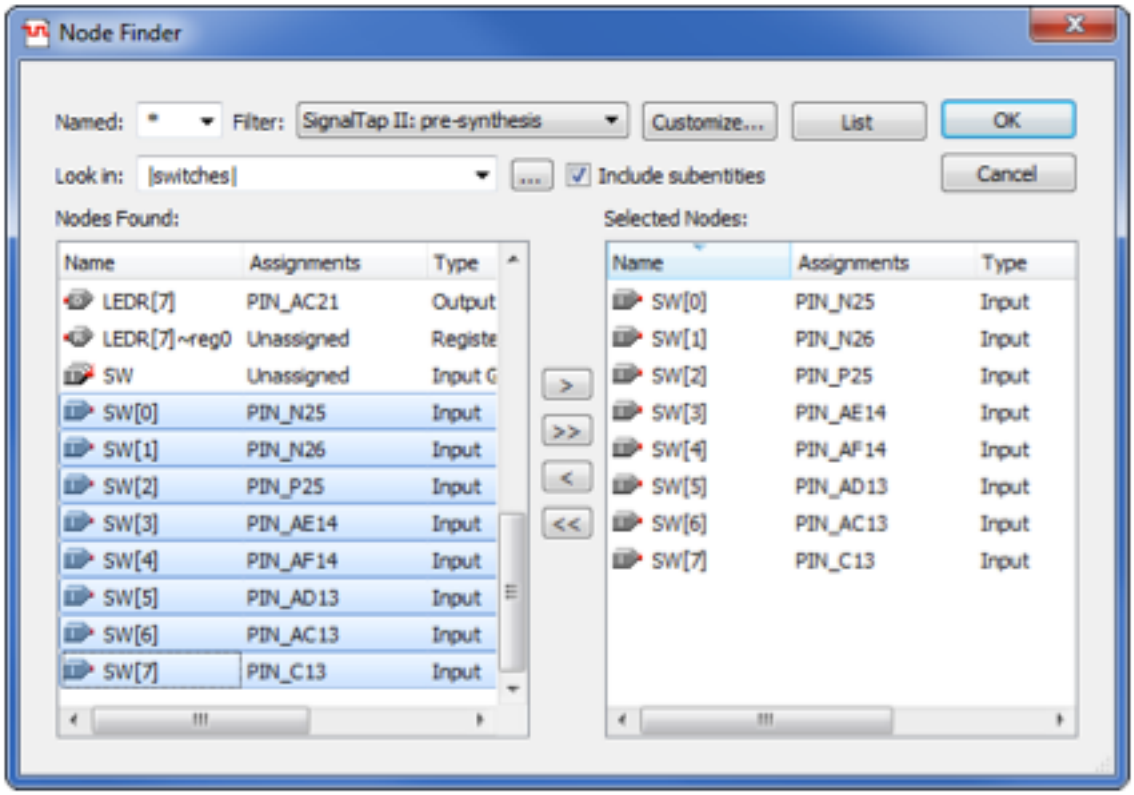
Set Up

Open the SignalTap II window by selecting File > New and choose SignalTapII Logic Analyser File and click OK. Save this file as signaltap.stp and click "Yes" when prompted whether to enable signaltap.stp in this project. Note: If you want to disable this file later, you can find the Enable button under Assignments > Settings > SignalTap II Logic Analyser.



We now need to add the nodes to connect to the analyser. In the Setup tab of the SignalTap II window, double-click in the area labelled "Double-click to add nodes". Select "SignalTap II: pre-

synthesis" in the filter box and click "List". Select SW[7] to SW[0] and click the right arrow ">". Note that as the design is synthesised, Quartus may have to add signals that aren't present in your HDL code. These are typically annotated with a Tilde character (~). By changing the filter, you can view none, some or all of these automatically generated signals.



Next we need to specify which clock is going to run the SignalTap module. In the Clock box of the Signal Configuration window, type "CLOCK_50". This is the 50MHz oscillator on the DE2 board. Note that the clock must be twice as fast as the fastest signal that is to be used as a trigger, or observed. This means for example, that using the 50MHz clock, you could not observe any event that is faster than 25MHz.

Finally we set up trigger conditions. These are the events that will cause the SignalTap module running in the FPGA to update the PC's screen. Right-click in the Trigger Conditions column on the SW[0] row and select "Rising Edge".

To connect to the board, ensure the DE2-115 is plugged in and turned on then click the Setup button in the Hardware section of the SignalTap II window, in the top right. Double-click "USB-Blaster" then click close.

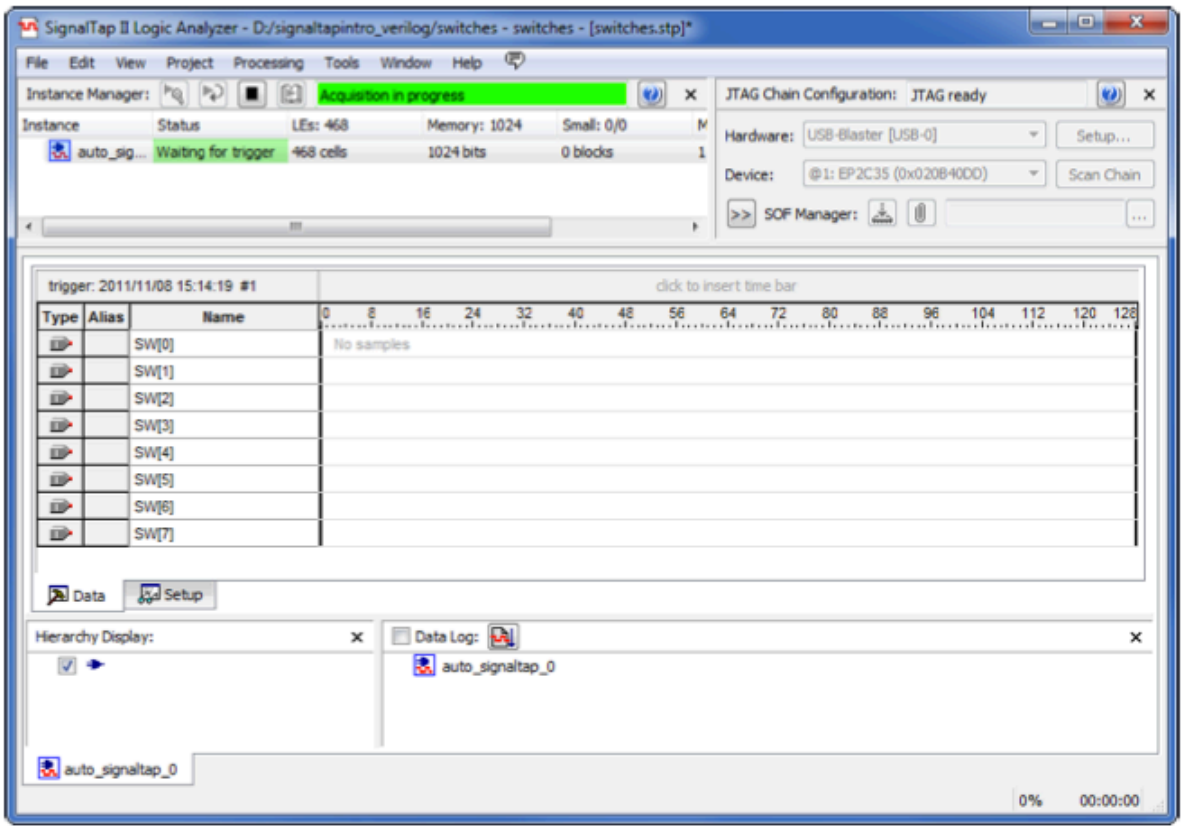
Return to the Quartus window, recompile your project and program the FPGA in the usual way. Once the programming is complete, you may return to the SignalTap window.

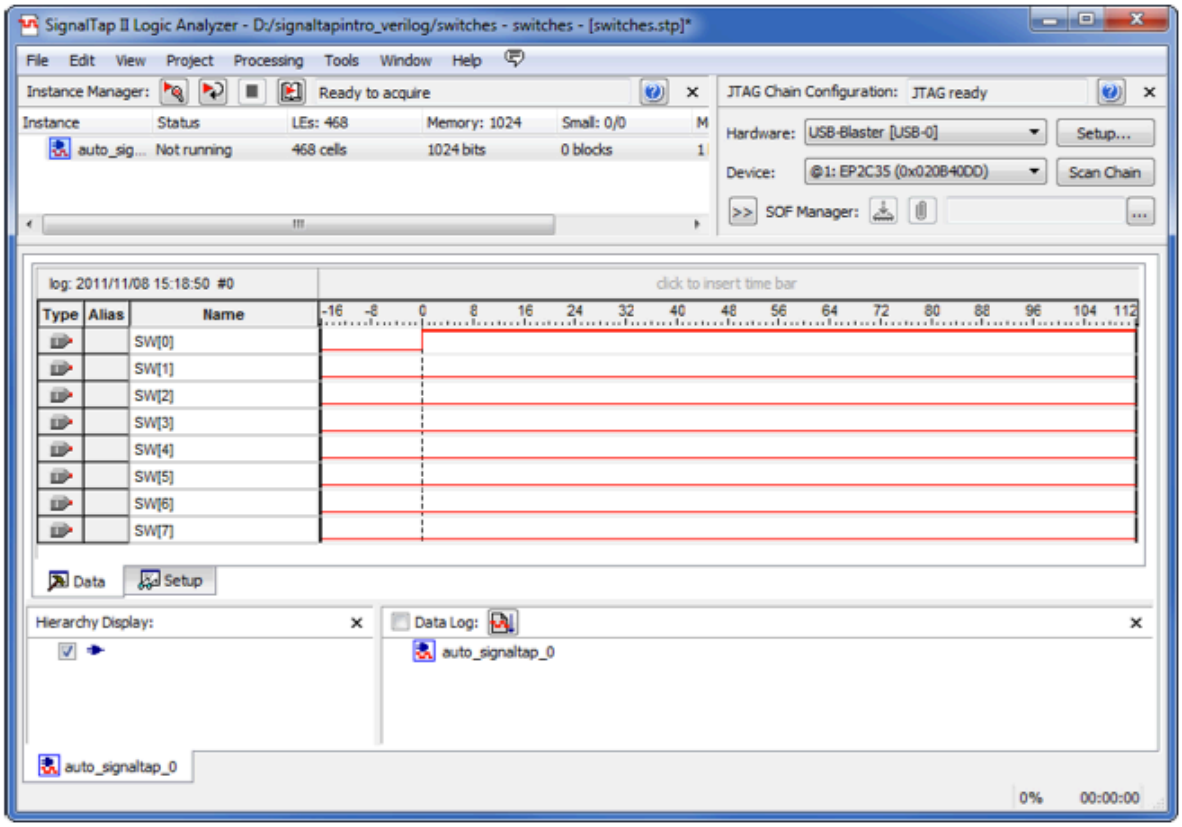
If the compile fails with an error message regarding Licence restrictions, you need to enable a feature called "Talkback". In order to do this, go to Tools > Options > Internet and Connectivity > Talkback Settings and check the "Enable Talkback" box. You should only have to do this once.

NOTE: Every change made in the SignalTap window will require that the main project is recompiled and programmed to the board. If you see any errors regarding to "Invalid JTAG Configuration", "Instance Not Found" or similar, ensure the board is correctly programmed from the main project window.

Simple Triggering

We're now ready to run. Click Processing > Run Analysis then click on the Data tab of the SignalTap II Window. Try setting SW[0] from off to on, this will cause a trigger condition and stop the analysis. Click Processing > Run Analysis again and this time, set the other switches to different positions before turning SW[0] from off to on





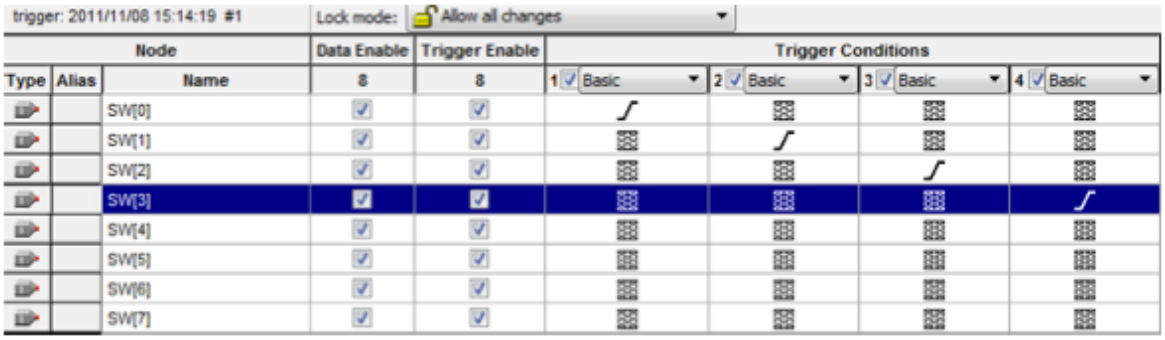
Try using Processing > Autorun Analysis. This works as above, except that the analyser keeps waiting for new triggers, even after it received the first one. That is, you don't have to click "Run Analysis" after each event.

Advanced Triggering

It's possible to get SignalTap II to trigger on more than a simple rising edge. Click the Setup tab of the SignalTap II window, find the Signal Configuration pane and select 4 from Trigger Conditions dropdown menu (you may have to scroll down in the Signal Configuration pane to see this menu). This modifies the node list window by creating three new Trigger Conditions columns.

Right click the Trigger Condition 1 cell for SW[0], and select Rising Edge. Do the same for the Trigger Condition 2 cell for SW[1], Trigger Condition 3 for SW[2], and Trigger Condition 4 for SW[3].

Select four rising edge triggers, one in each column as shown below.



This will now trigger when a rising edge is found on SW[0], SW[1], SW[2] and SW[3] in that order.

Recompile the design, download to the DE2 board and verify this trigger condition.

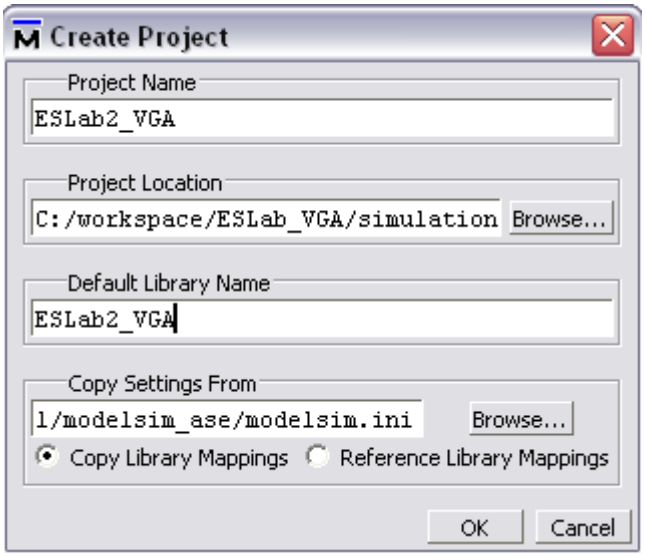
Part 2: ModelSim

ModelSim is a powerful simulation tool that can be used to test the functionality of your designs without going through a full Synthesis and Download cycle.

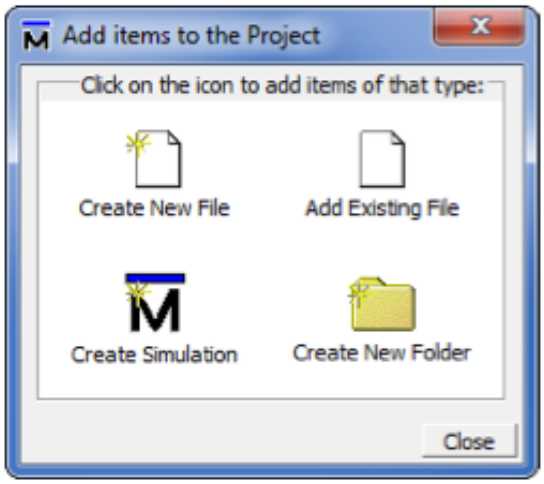
ModelSim projects exist separate from Quartus projects. This is done so that test bench files and other pieces of simulation-specific Verilog never end up being synthesised for the target device. Here we will simulate the VGA driver from last lab.

Download the **revised** ESLab2_VGA project from Wattle. This differs from the previous week's code simply by the addition of a reset signal wired through to a button. On an FPGA, registers start at some value, usually zero; however in simulation all registers must, at some stage, be explicitly set to an initial value.

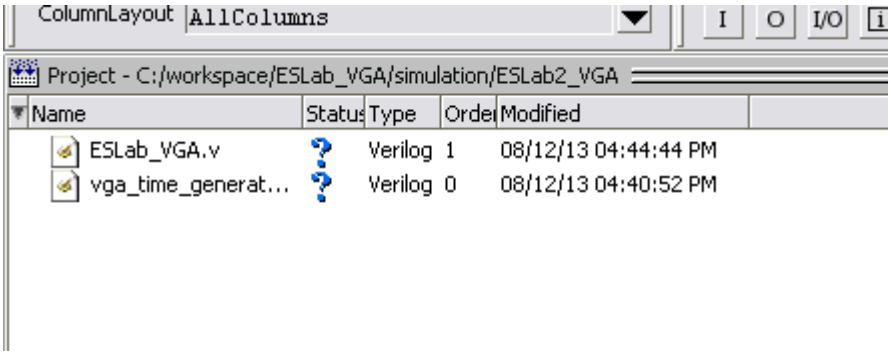
Open the ModelSim software and select File > New > Project.



Enter the Project Name and Default Library name to match the Quartus project, and select the Project Location to be in the Simulation subfolder of the Quartus project. Click OK.



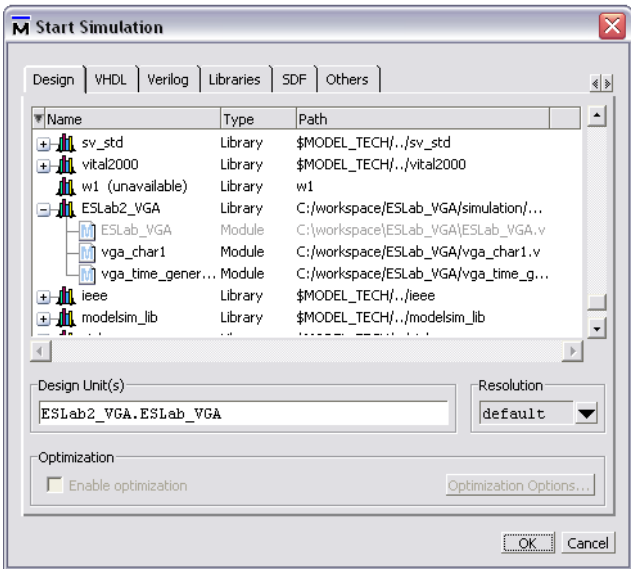
In the Add items to Project box, select Add Existing File and add all the Verilog files from the project. Click OK. Your project window should look something like the below.



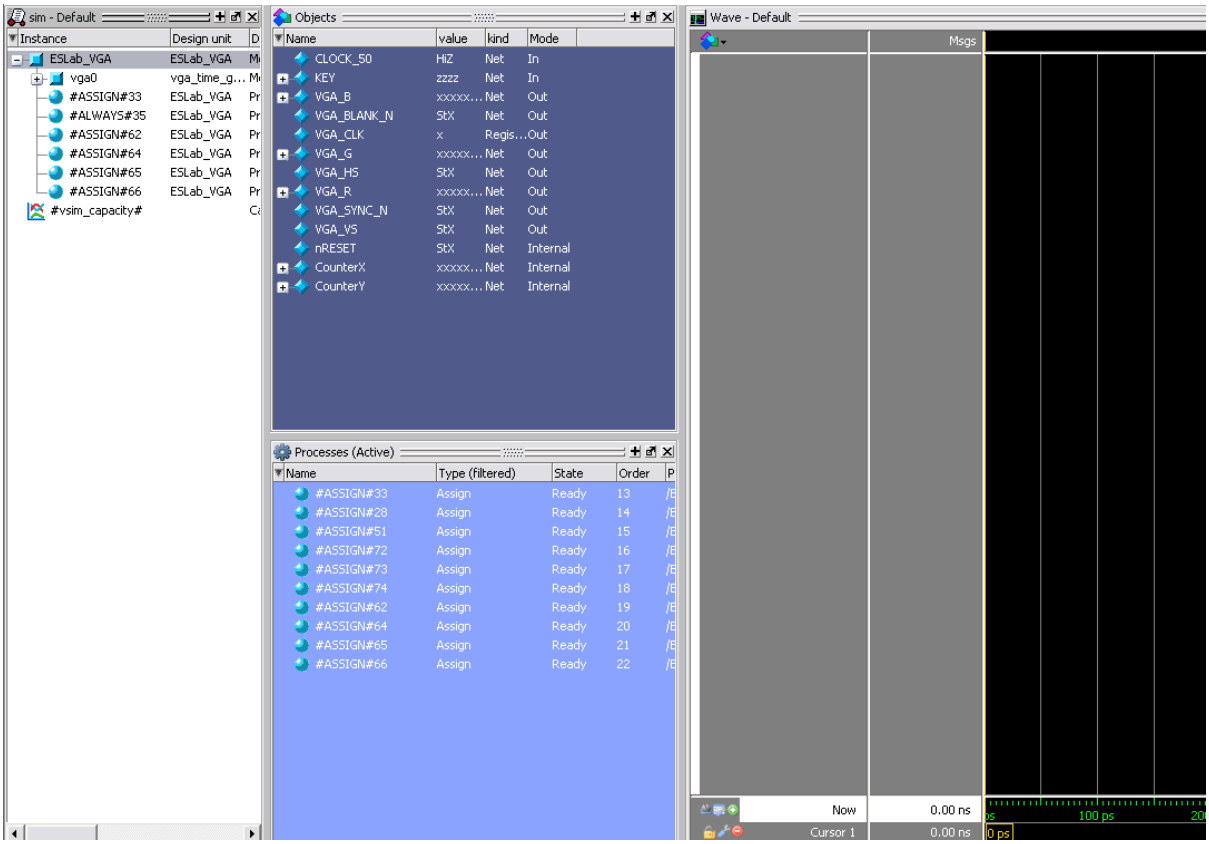
Select Compile > Compile All.

Select Simulate > Start Simulation

Find ESLab2_VGA in the list, expand it and select the ESLab2_VGA.v Verilog file as below



Once you click OK, a window will appear similar to the one below. It lists all the signals that can be driven or monitored from the simulator.



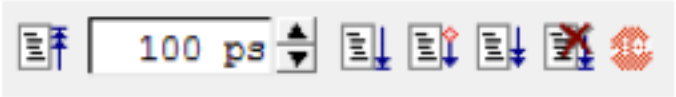
In the Objects window, select "CLOCK_50", "VGA_R", "VGA_G", VGA_B", "nRESET", "CounterX", "CounterY"; right click and select Add > To Wave > Selected Signals.

ModelSim can be used for scripted simulations or manual simulations. For this tutorial, we will manually drive inputs much as one would drive the switches and buttons on the board.

The two inputs to the system are the clock and the reset line; we must set these manually.

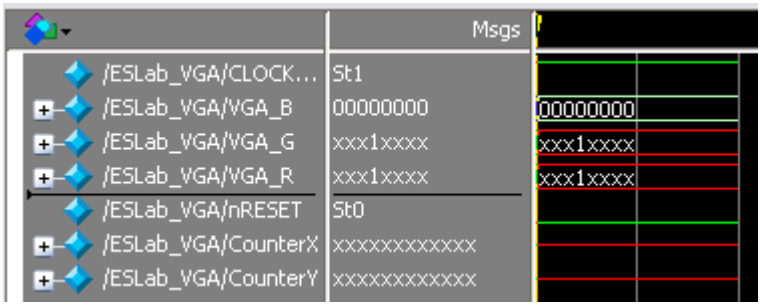
Right click on "CLOCK_50" in the Wave tab, select Clock and set the period to "20000". This is a 50MHz clock, just like on the board. Right click on "nRESET" and select Force. Enter a Value of 0 and click OK.

Locate the simulation controls box as shown below.



The value in the box specifies how long the simulation should run and the button immediately to the right is the run button. The others allow you to pause and restart the simulation; we won't use them in this tutorial.

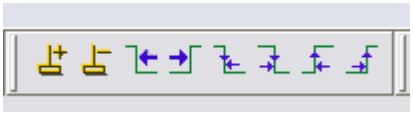
Click the Run button to advance the simulator 100ps. This will give the reset signal time to perform the reset inside the design. The wave window will look something like the below.



Now right-click on the "nRESET" signal, select Force and set value to 1.

Now enter "25ms" in the time box instead of 100ps and click Run. Use the Zoom tools to zoom in and out and view the results of the simulation.

To find the portion of the simulation that is drawing the green horizontal line, highlight the VGA_G signal in the Wave window and find the feature bar like below.



Click "Find Previous Transition" to move to the location where the VGA_G signal changed. Confirm that the display was green for the time that CounterY was equal to 0x30.