Lab 6 : Introduction to Verilog

Name:

Khanh

Sign the following statement:

On my honor, as an Aggie, I have neither given nor received unauthorized aid on this academic work

Nguyen

1 Objective

The main objective of this lab is to give hands on experience with Verilog HDL and the Xilinx ISE.

2 Pre-requisite

For this lab you are expected to know some basic Verilog programming and understand the Xilinx ISE. You are advised to go through the Verilog tutorial posted on your course website.

3 Using Xilinx ISE

Xilinx ISE is a software development environment designed by Xilinx Inc. To date it supports synthesis for different FPGA boards. In our lab we will be using the simulator only and will not use the synthesis tools. Xilinx ISE has its own built-in simulator, ISim. In addition, it can support third party simulators such as ModelSim. In our labs, however, we will be using ISim.

Before you begin, ensure Xilinx ISE is installed on the workstation that you will be working on (For windows go to start → programs → Xilinx ISE Design Suite 11 → ISE → ISE 32-bit). Click on Project Navigator, to launch the application and start working.

Please be sure you only select the 32-bit version, as the 64-bit version does not contain ISim.

Note: If you are planning to install Xilinx ISE on your personal computer, you can refer to the Xilinx Installation guidelines posted on the course website.

4 Short Tutorial

1. Launch Xilinx ISE
2. Create your project
   (a) Select "File → New Project," and you should see the New Project Wizard as shown in Figure 1.

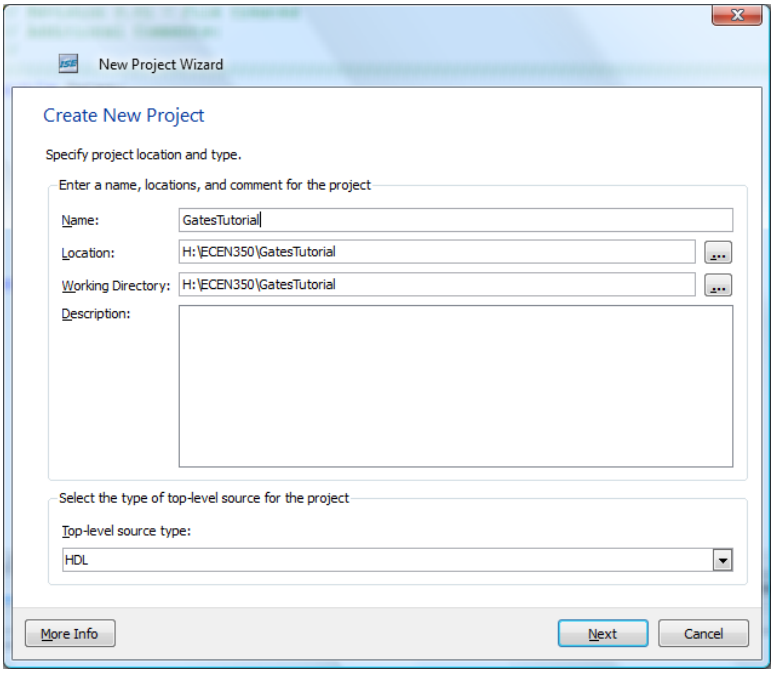

Fig. 1: New Project

   (b) Name the project *GatesTutorial* and tell it to save within a directory on your home drive. Then hit "Next."
   (c) You will see a large list of project options in the Device Properties window shown in Figure 2. Ensure your settings match those in the provided figure and hit "Next."

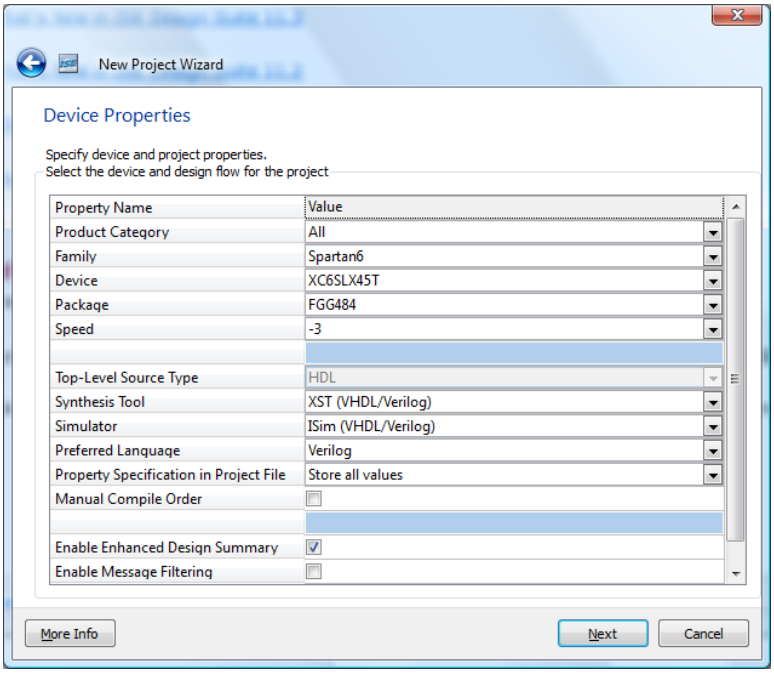

Fig. 2: Device Properties

(d) For the next two screens, hit "Next". We will not be adding or creating any source files just yet.

(e) Finally, complete the wizard by hitting "Next" twice and then "Finish." Do note the content of these screens.

3. From within ISE, select File → New and click on Text File. Type the text below into the new file and name it "Gates.v".

```
module Gates(in, out);
    input [0:1] in;
    output [0:2] out;

    and and0(out[0], in[0], in[1]);
    or or0(out[1], in[0], in[1]);
    xor xor0(out[2], in[0], in[1]);

endmodule
```

This creates a module which has one input comprising 2 bits, and one output comprising 3 bits. The first bit of the output is set to the AND of the two input bits, the second to the OR, and the third to the Exclusive OR.

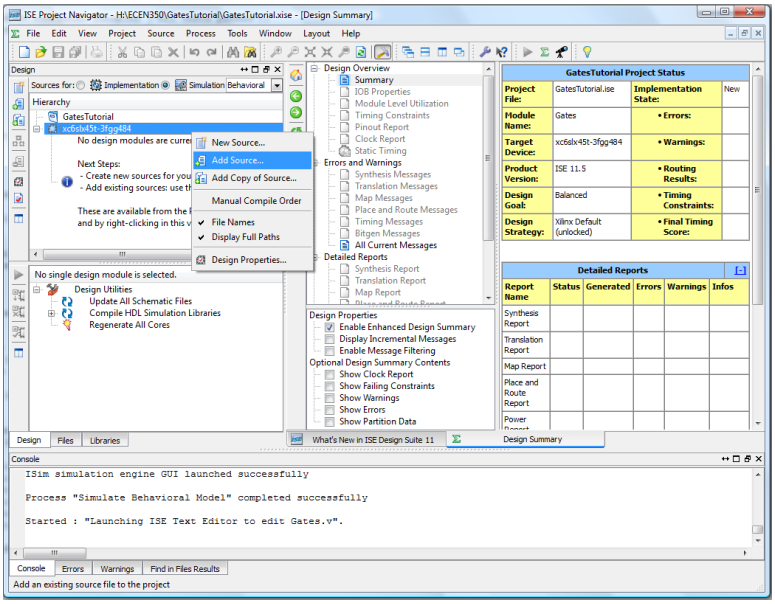

Fig. 3: Add Source

4. Right click on the FPGA part name and select Add Source (see Figure 3). Locate the "Gates.v" file you just created.

5. Test your circuit

   (a) Right click on the FPGA part name.

   (b) Click on "New Source".

   (c) Select Verilog Test Fixture and name it "GatesTest". Hit "Next."

   (d) Make sure Gates is selected and hit "Next," then "Finish."

   (e) Change the "Sources for" setting from "Implementation" to "Simulation".

   (f) Open GatesTest.v if it isn't already open in the editor

   (g) Add this code below where it says to add stimulus

```
#0      in=0;
#10     in=2'b01;
#10     in=2'b10;
#10     in=2'b11;
```

   (h) Save your work, and Double click on "Simulate Behavioral Model"

6. View the Results

   (a) You may need to zoom out to find the point where the data changed

(b) Zoom in so you can see the data as it changes

(c) Expand the output so you can see individual bits. The 0th bit should behave like an and gate, the 1st as an or gate, and the 2nd as an xor gate like in Figure 4:

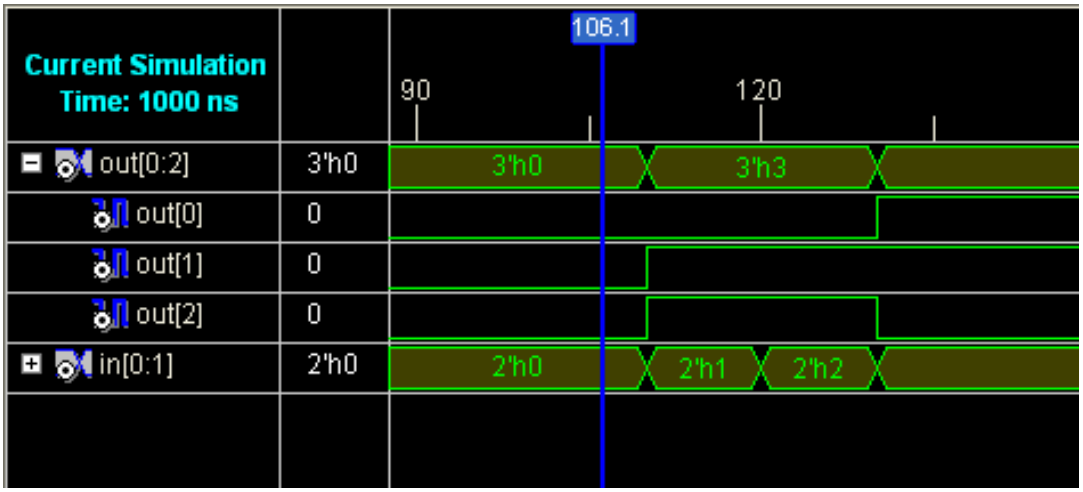

Fig. 4: Simulation

7. At this point, you must demo the simulation waveform to the TA.

5 Testing Verilog Designs

For all modules listed below:

- Write down a Verilog implementation (if you not done so in the prelab).
- Test your code using the provided testbenches and verify that your code is correct.
- Demonstrate the correct functionality to the TA.

Note: Comment your code properly, in order to get the credit you deserve.

5.1 JK-flip-flop

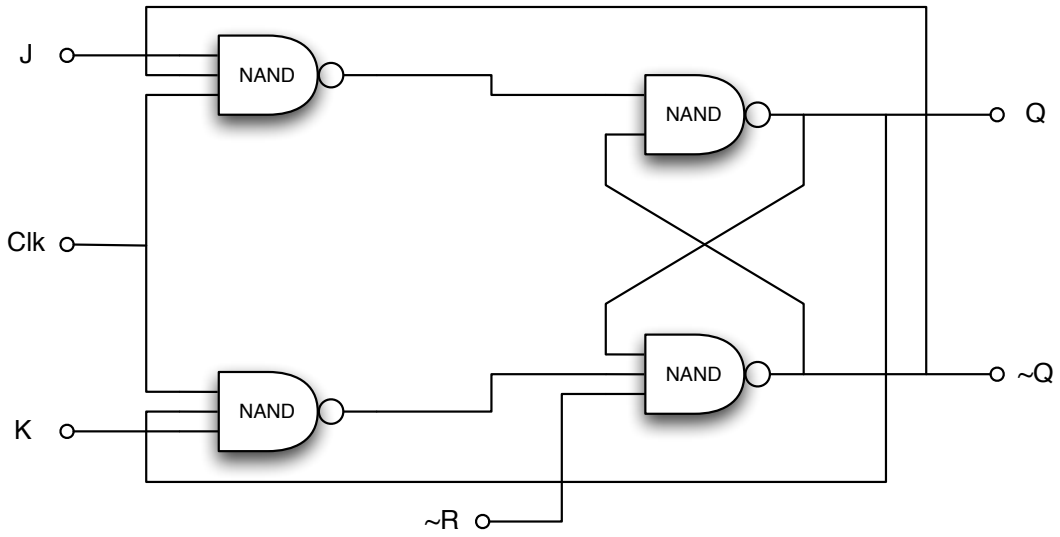

Fig. 5: JK-Flip-Flop With Reset

Use the structural model designed in the prelab with the following module definition. Give all your gates a delay of 2 and make sure that if reset is set to 1, the output is 0. See the designed in Figure 5 to see how to implement the reset.

```
module JK(out, j, k, clk, reset);
input j, k, clk, reset;
output out;
```

5.2 2-to-1 MUX designed (structural model)

Use the structural model designed in the prelab with the following module definition:

```
module Mux21(out, in, sel);
input [1:0] in;
input sel;
output out;
```

5.3 2-to-4 decoder

Use behavioral model with the following module definition:

```
module Decode24(out, in);
input [1:0] in;
output [3:0] out;
```

6 Deliverables

At the end of the lab you are supposed to turn in:

- Your Verilog HDL for each module