Lab 1 – Introduction to Verilog HDL & the Lab Design Flow

Name: Robert Bara

Section #: 003

Date: 1/23/2020

Summary/Abstract

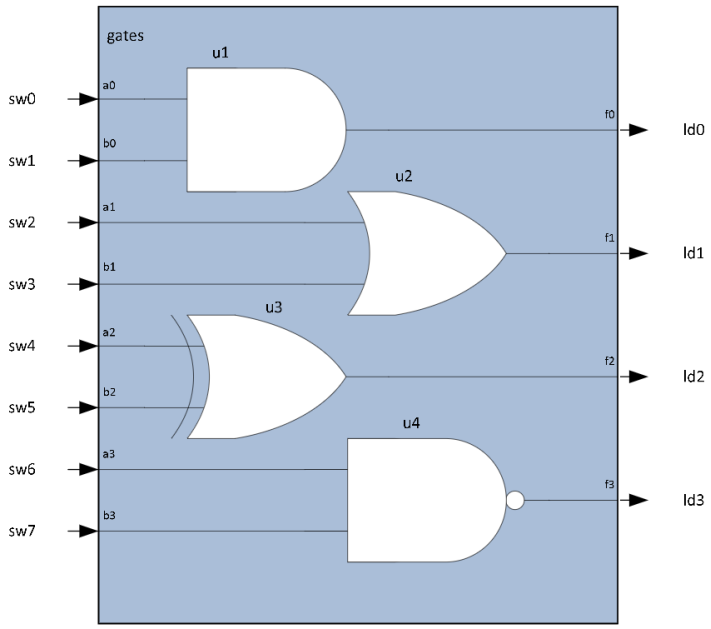
Lab 1 introduces the Cloud 9 IDE and Verilog language by coding four different kinds of gates: AND, OR, XOR, and NAND. Upon completion of the truth table, the schematic and truth table explains how to switch on/off the lights based on their respective gate.

Introduction

This lab applies a basic introduction to Verilog by creating various gates based on the schematic shown. Relevant coding information for this lab calls for the use of gate primitives in Verilog instead of using assign statements. Identifying that u1 is an AND gate, u2 is an OR, u3 is an XOR, and u4 is a NAND from the schematic is crucial to creating a truth table and code to implement these gates to the Basys3 circuit. Completing the lab in this order strengthens logic skills and serves as an introduction to how the IDE and Basys3 boards function, which will be used to design all labs moving forward.

Procedure

Given the schematic shown, the following gates are: u1=AND, u2=OR, u3=XOR, and u4=NAND.



From this, identifying that there is two inputs **a** and **b** and one output **f** for each gate allows the completion of the following truth table:

a0, 1, 2 or 3	b0, 1, 2 or 3	f0	f1	f2	f3
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	1
1	1	1	1	0	0

Now that the theoretical logic of the circuit is completed, it is a matter of translating the table into Verilog's language. Upon typing the code in the *gates.v* file under lab1 on Cloud9, the code is ran and then simulated using the file *gates.sim*. After the simulation finds 0 mismatches, the testbench can be ran to check to code and create a .bit file. The .bit file is then downloaded to a USB and uploaded to the Basys3 circuit board by entering the USB and connecting the USB to mini B cable into the board and pushing in the power button. Finally using the truth table and following the schematic, the switches sw0-sw7 control the inputs for each gate and by using the combinations shown in the truth table with 1 representing "switch on" and 0 representing "switch off" for the inputs, the gate will either turn on or off lights ld0-ld3 based on the type of gate they are and being that the code is correct.

Results

My results provided the necessary code for each gate with 100% success. Originally, I used assign statements before entering the lab, but quickly converted them to gate primitives by using the website from the manual and what was taught through the lecture videos.

Design Code

The following code is the code I ran in the lab by using gate primitives and generated a .bit file from with no mismatches:

```
1  //
2  //Robert Bara lab1 : version 01/15/2020
3  //
4  `timescale 1ns / 1ps
5  module gates(
6      output logic f0, output logic f1, output logic f2, output logic f3,
7      input logic a0, input logic b0, input logic a1, input logic b1,
8      input logic a2, input logic b2, input logic a3, input logic b3
9      );
10
11     // Write code starting here ...
12     and u1(f0,a0,b0);
13     or u2(f1,a1,b1);
14     xor u3(f2,a2,b2);
15     nand u4(f3,a3,b3);
16
17 endmodule
18
```

Simulation Results

The following is the screenshot of the confirmation that the simulation from *gates.sim* was a success, please let me know if I need to screenshot more of it for future labs.

```
source xsim.dir/work.tb_gates/xsim_script.tcl
# xsim {work.tb_gates} -autoloadwcfg -tclbatch {tb_gates.tcl} -onerror quit
Vivado Simulator 2018.2
Time resolution is 1 ps
source tb_gates.tcl
## run all
Simulation complete - no mismatches!!!
$finish called at time : 80 ns : File "/home/tuj22026/2613_2020s/lab1/tb_gates.sv" Line 62
## exit
INFO: [Common 17-206] Exiting xsim at Thu Jan 23 13:16:32 2020...


Process exited with code: 0
Pane is dead
```

Hardware Implementation

The hardest part of implementing the hardware was understanding which switch lines up with the lights being that the labels are small and a bit hard to read. After using the truth the table and testing all the gates work, my design was demonstrated to Franka on the afternoon of 1/23/2020 at 1:30pm during the lab period.

Conclusion

Using the material taught in lecture, I was able to understand the schematic and complete the truth table. Using the truth table and introduction to Verilog's syntax, I coded the gate primitives in four lines of code and simulated the program with success. After this I ran the testbench and created a .bit file from the code to be uploaded to the Basys3 board. Using the switches, I was able to create four different gates by instantiating the appropriate primitives in Verilog that matches the schematic and truth table needed for the design.