Lab 2 Part 1 –

Part 1A: IDE Tutorial

Complete the tutorial (Altera-Quartus) for the Integrated Design Environment (IDE) you will be using.

Part 1B: Gates in Verilog

Create a Verilog hardware description for the 7408 AND gate. When this is written, do a SYNTAX check by compiling this file. IF you get errors, go back and fix them. Usually it's best to fix the first error and re-compile. When you get a successful compilation, you then assign pins to the two inputs and one output. (See the ISE tutorial in this lab manual) You assign the pins according to function. Inputs of the gates should be assigned a switch, and the output should be assigned an LED. Deciding which switch and LED is up to you!

Once the pin assignment is done, you move on to downloading your design to the Amani board. (Again, see the tutorial in this lab manual) If that is done successfully, you can now TEST your circuit against the truth table you have prepared for lab. Write down your OBSERVED results next to your PREDICTED results. Create 3 more files (one file for each remaining gates) using structural Verilog programming for the 74LS00, 74LS04, and 74LS32. After you have done this, compile, download, and test each of them as you did the 7408 AND gate. Report your results.

Remember we are only mimicking the function of the 7400 Series TTL gates using Verilog and our Programmable Logic.

Verilog has built in primitives that can be used to describe a circuit using **structural** modeling.
The following primitives are for combinational logic circuits.
and, nand, or, nor, xor, xnor, not, buf.
Use four of these primitives to describe four 7400 Series TTL integrated circuits (ICs):

Remember there are 4 independent gates in each integrated circuit (7400 NAND, 7408 AND, 7432 OR) and 6 independent gates in the 7404 (NOT IC).

The following Verilog examples are available for you to type in the Verilog editor, and compile, download, and test to be sure they follow the truth table correctly. In your lab report, make the truth tables and logic diagrams for each Verilog Module.

```
module andgate(in1,in2,out);
      input in1,in2;
      output out;

      wire in1,in2,out;

      and g1(out,in1,in2);
endmodule
```

```
module orgate(in1,in2,out);
      input in1,in2;
      output out;

      wire in1,in2,out;

      or g1(out,in1,in2);
endmodule
```

```
module notgate(in1,out);
      input in1;
      output out;

      wire in1, out;

      not g1(out,in1);
endmodule
```

```
module nandgate(in1,in2,out);
      input in1,in2;
      output out;

      wire in1,in2,out;

      nand g1(out,in1,in2);
endmodule
```

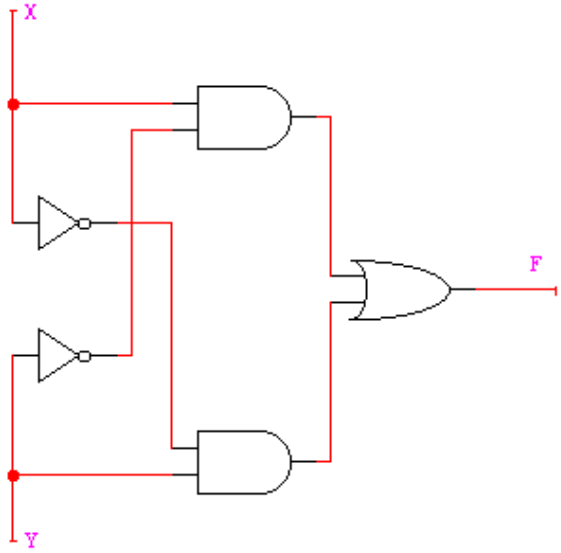
* *

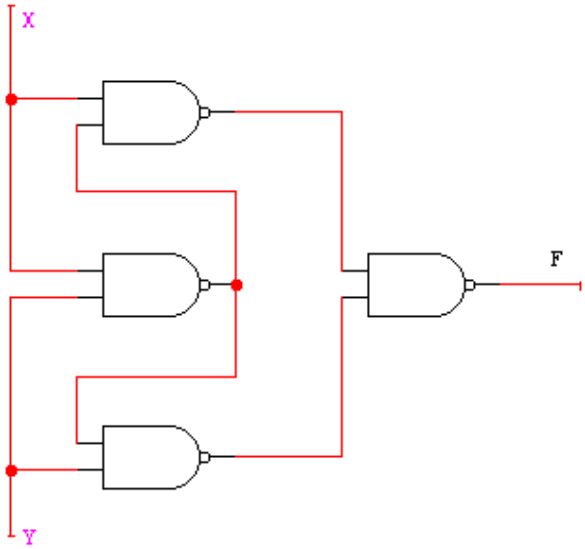
Once you have tested each gate as a module, create a single module instead of four independent modules. This module should have 2 inputs and 4 outputs. There is an output for each function.

Lab 2 Part 2 – Exclusive OR gate

Each circuit below is a two input EXCLUSIVE-OR gate. Recall that an EXCLUSIVE-OR has a high output if one or the other inputs are high. Write a structural model to implement this function. You can write two description programs, one for each logic diagram, or combine them into one module.
Compile, download, and test each circuit. Show that each circuit satisfies the truth table for an EXCLUSIVE-OR function. Build the Exclusive OR gates with the **74XX gates and the CPLD/FPGA**.

Re-draw each circuit for your lab report with the proper names of each internal node and inputs/outputs (I/O).





F = X exclusive-or Y

```
module xorgate(x,y,f);
      input x,y;
      output f;

      wire x,y,f;
      wire notxout,notyout,and1out,and2out;

      or or1(f,and1out,and2out);

      and and1(and1out,x,notyout),
      and2(and2out,y,notxout);

      not notx(notxout,x),
      noty(notyout,y);

endmodule
```

QUESTION #1: Is there an easier way to write an EXCLUSIVE-OR function still using primitives and structural modeling? Explain in the conclusion portion of your report.

QUESTION #2: If you really needed to use EXCLUSIVE-OR gates could you buy a 7400 Series TTL Integrated Circuit (IC)? If so, what is the TTL part number and explain and draw the pin out of the device.

LAB 2 Part 3

Consider a circuit with 4 inputs A, B, C, and D and one output F. Input A should be the most significant bit. Create a truth table to implement the equation below. Show all the work (Boolean Algebra) as part of the design to reduce the equation for F. Draw the logic diagram (with names of internal and external signals) for your final reduced equation. After completing your logic diagram, write the Verilog description using "**Structural Modeling**". Compile, download and test your design against the truth table you made from the original equation below.

F = /((/C + /D) · (/A + B + /D)) ⟵ (using standard AND OR NOT symbols)

Note: If you wrote the above equation in Verilog HDL using the "dataflow" modeling technique, the equation would look like this:

F = ~((~C | ~D) & (~A | B | ~D))

LAB 2 Part 4

This exercise is to improve your method of reduction techniques. Use only the **four** integrated circuits (ICs) packages: the 7400 (NAND), 7404(NOT), 7408(AND) and 7432(OR). **Wire your design in Multisim and Verilog,** and **verify** your designs with the **Truth Table**.

Description:

Consider a circuit with 4 inputs A, B, C, and D and two outputs F1 and F2. Input A shall be the most significant bit. Design a circuit to implement the given Truth Table. Show all the work (K-maps, Boolean Algebra, DeMorgan's) as part of the design to find the final equations for both F1 and F2. Also create the "**complete" schematic** diagram (and include in your lab report) for the circuit using the four (or less) integrated circuits (ICs). When you are happy with your design, build it in Multisim and Verilog and show both designs match the Truth Table.

Note: *Both F1 and F2 must function at the SAME time, so your schematic diagram will have both outputs and shall be realized with both outputs functioning at the same time. (Hint: Share circuits from each function)*

Truth Table

Inputs				Outputs	
A	**B**	**C**	**D**	**F1**	**F2**
0	0	0	0	0	1
0	0	0	1	1	1
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	0	0
0	1	0	1	1	1
0	1	1	0	0	1
0	1	1	1	0	0
1	0	0	0	1	0
1	0	0	1	1	1
1	0	1	0	0	1
1	0	1	1	0	1
1	1	0	0	0	0
1	1	0	1	1	1
1	1	1	0	0	1
1	1	1	1	0	0