Class:	CPE-300L	Semester:	Spring 2020

Points		Document author:	Brysen Kokubun
		Author's email:	Brysenkokubun96@gmail.com
		Document topic:	Postlab 2

Instructor's comments:

1. Introduction / Theory of Operation

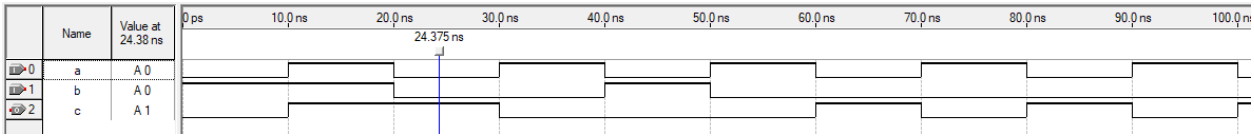
- For experiment-1 we obtained the waveform for a 2-input Xnor gate, using normal Verilog style, and ANSI-C style. After obtaining the waveform we implemented the logic to perform its logic on the DE2 board. Implementing the logic on the DE2 board was the most difficult due to driver/device issues. For experiment-2 we used gate primitives in order to implement the given schematic on the DE2 board as well as waveforms. For experiment-3 we created a 2-to-1 multiplexer, using continuous assignments and conditional statements. We simulated the waveform for the multiplexer in ModelSim and compared it to the truth table. For experiment-4 we modeled an OR gate having a 5ns delay and changed the input state to switch every 10ns & 2ns. We verified the difference between the two waveforms, and it showed that the logic for the gate, was "delayed" until the 5ns was over, then the OR gate logic was performed. For experiment-5 we created an 8-bit adder

a. Describe what the primitive is in Verilog
   - Primitives in Verilog are frequently used components which are built into Verilog's library. When using a gate primitive, the output goes on the left and the input goes on the right. Some Verilog primitives include, OR, XOR, NAND, AND, Nor, Not, etc.
b. Define the structural model in Verilog
   - Structural model in Verilog uses logic gates in order to create schematic/circuit diagrams. Verilog uses primitive gates to compile and synthesize code.
c. Define the dataflow in Verilog
   - Dataflow model in Verilog uses combinational logic as statements and functions rather than a logic gate structure. Dataflow uses various operators on operands to get the code to function correctly.

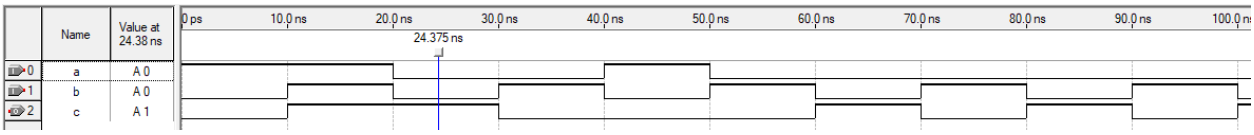
2. Experiment Results

Experiment 1: Prelab Question #3 Waveform/DE2 Board Implementation

a. Waveforms

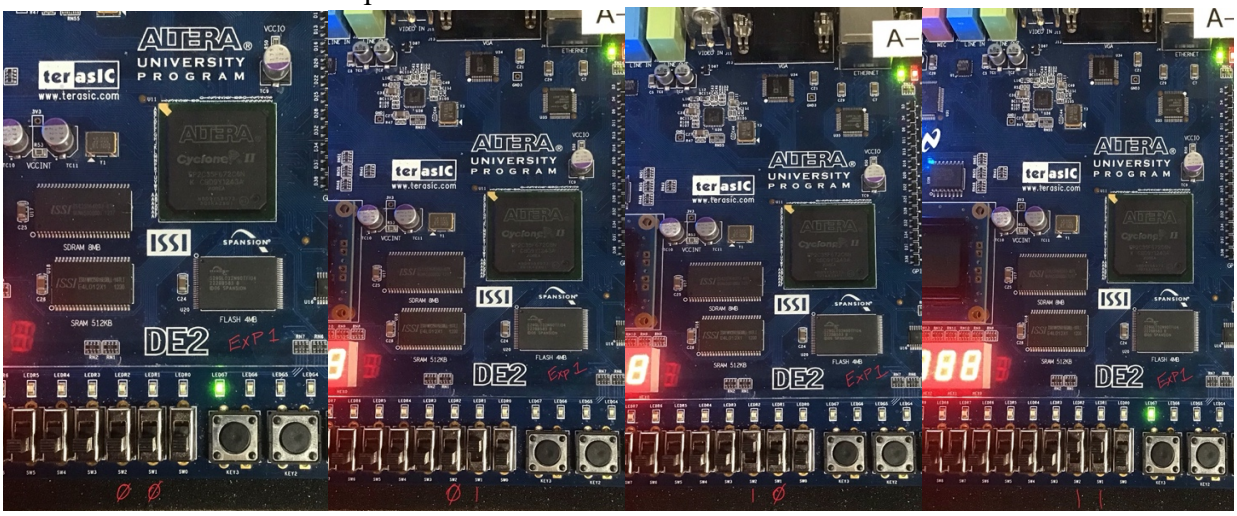


XNOR: Normal Verilog Style



XNOR: ANSI-C Style

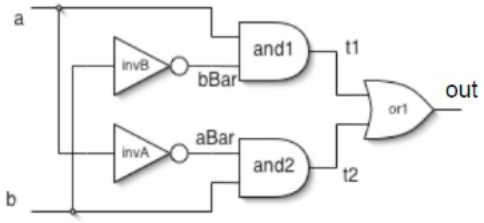
b. Picture of Implementation in DE2



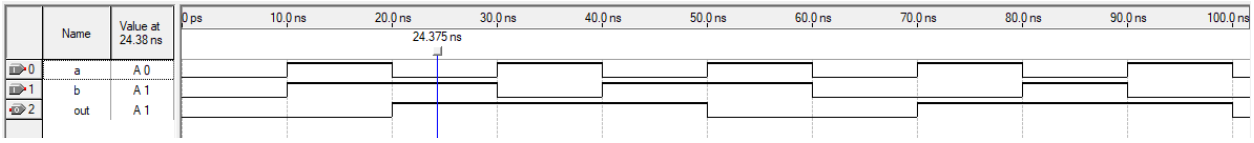
00 = 1 "LED ON" **01 = 0 "LED OFF"** **10 = 0 "LED OFF"** **11 = 1 "LED ON"**

Experiment 2: Verilog code with Primitives

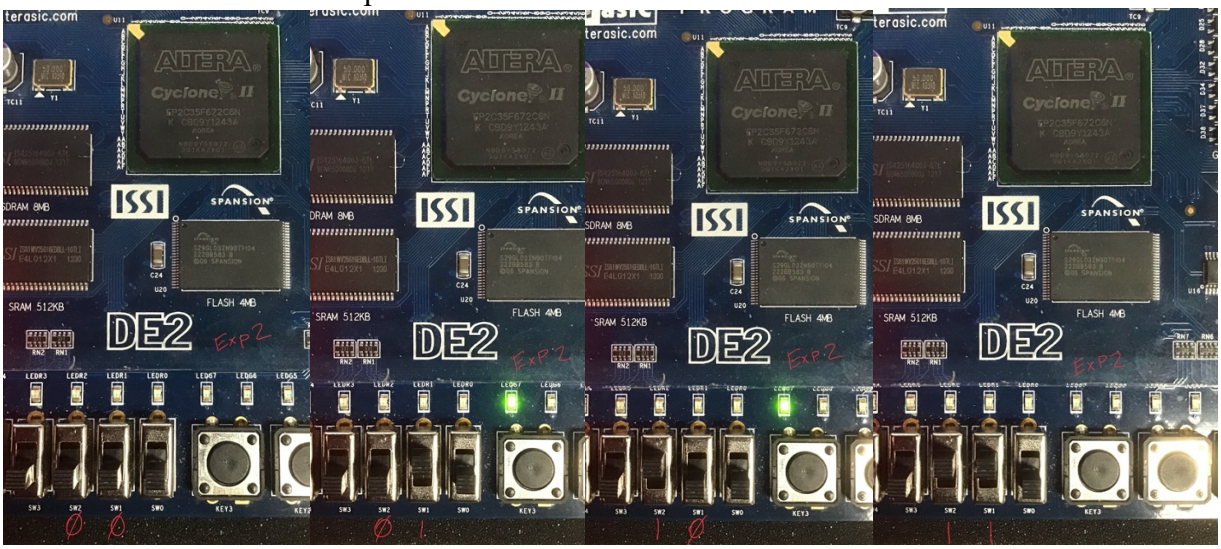
a. Verilog Code & Waveform




```
1  module Lab2 (output out, input a, b);
2      wire aBar, bBar, t1, t2;
3      not InvB (bBar, b);
4      not InvA (aBar, a);
5      and and1 (t1, a, bBar);
6      and and2 (t2, b, aBar);
7      or or1 (out, t1, t2);
8  endmodule
```



b. Picture of Implementation in DE2

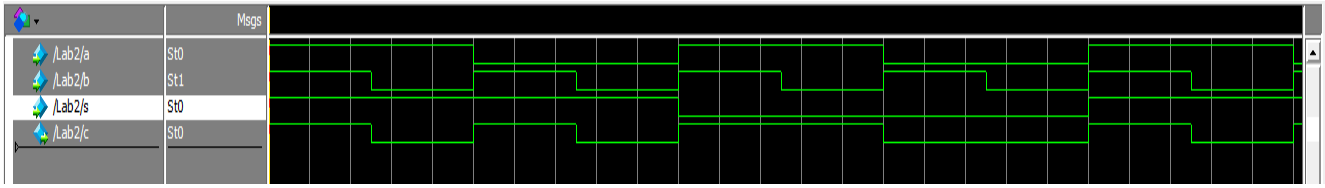


00 = 1 "LED OFF" **01 = 1 "LED ON"** **10 = 1 "LED ON"** **11 = 0 "LED OFF"**

Experiment 3: Multiplexer 2-to-1

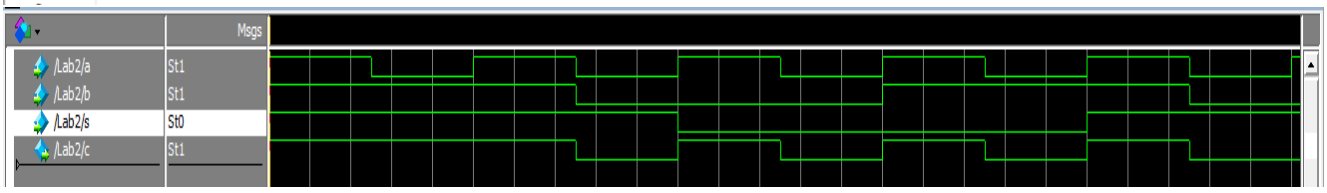
a. Using Continuous Assignment

```
1  module Lab2 (output c, input a, b, s);
2      wire t1, t2;
3      and and1 (t1, a, ~s);
4      and and2 (t2, b, s);
5      or or1 (c, t1, t2);
6  endmodule
```



b. Using Conditional Statement

```
Ln#
1  module Lab2 (output c, input a, b, s);
2    assign c = s ? b : a;
3  endmodule
```



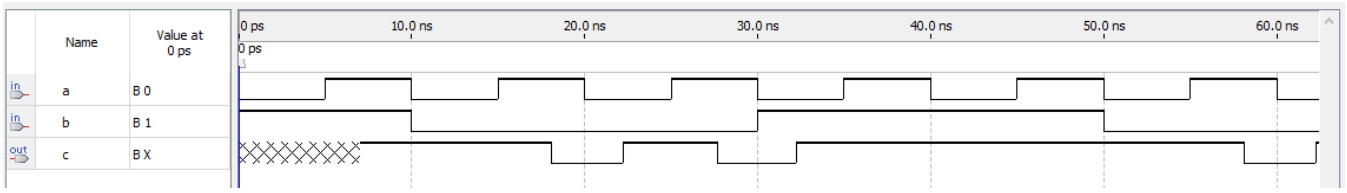
Experiment 4: Delays Model OR gate w/ 5ns Delay

- Gate input state changing every 10ns

a. Code

```
Ln#
1   module Lab2 (output c, input a, b);
2     assign #5 c = a|b;
3   endmodule
4
```

b. Waveform

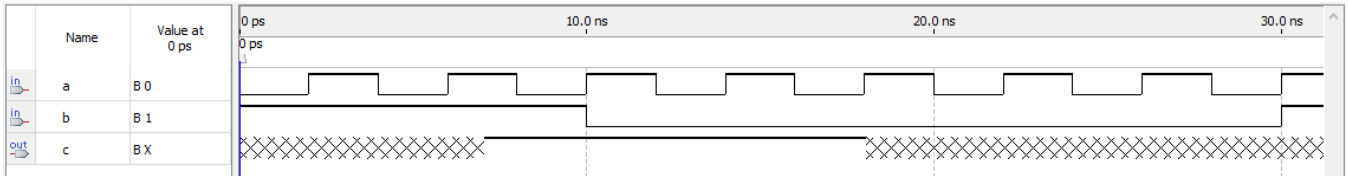


- Gate input state changing every 2ns

a. Code

```
Ln#
1   module Lab2 (output c, input a, b);
2     assign #5 c = a|b;
3   endmodule
4
```

b. Waveform

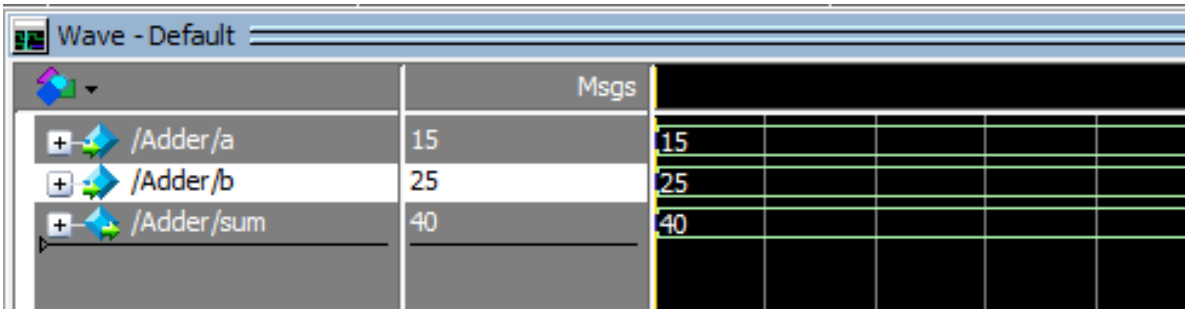


Experiment 5: Vectors Implementing 8-bit Adder

a. Code

```
1   module Adder(a, b, sum);
2     input [7:0] a, b;
3     output [7:0] sum;
4     assign sum = a+b;
5   endmodule
```

b. Waveform



3. Questions

1. What is the Verilog preprocessor directive?
   - Verilog provides sets of language preprocessing directives for macro definitions, conditional compilation of code, and file inclusion. Some preprocessor directives include define, else, ifdef, include, resetall, undef.
2. What is the difference between & and && operators?
   - '&' AND's each bit together, result is size of largest operand while '&&' AND's two expressions together, returns 1 bit Boolean true (1) or Boolean false (0).

3. Is Verilog case sensitive? If yes, what does it mean?
   - Verilog is case sensitive, when declaring a variable name, it has to be EXACTLY the SAME in order to be used later in the code. Ex. XOR1 and Xor1 and xOR1 are all different.

4. List types of nets in Verilog.
   - Nets variables represent physical connections between structural entities and do not store values, instead nets have value of their drivers which change continuously. Examples of types of nets in Verilog are: wire, tri, wor, trior, wand, triand, tri0, tri1, supply0, supply1, etc.

5. Label the structural and behavioral model in experiment 3
   - The structural model was part-a of experiment 3.

```
1  module Lab2 (output c, input a, b, s);
2      wire t1, t2;
3      and and1 (t1, a, ~s);
4      and and2 (t2, b, s);
5      or or1 (c, t1, t2);
6  endmodule
```

The behavioral model for experiment 3 was part-b.

```
Ln#
1   module Lab2 (output c, input a, b, s);
2     assign c = s ? b : a;
3   endmodule
```

4. Problems

- Throughout this lab there were many problems that occurred. Majority of the problems came from setting up the DE2 board, the proper drivers, usb-blaster, programmer tool, etc. After a little bit of trial and error we eventually got the DE2 board to implement the code that was created. Other than the troubles with the DE2 board, the other parts of the lab went smoothly.

5. Conclusions

- To conclude for lab 2, we were able to familiarize ourselves with Quartus, ModelSim and DE2 board. Within Quartus we created Verilog HDL codes to implement various schematics, to verify waveforms matching the truth tables. Within ModelSim we were able to create waveforms in order to conclude that the Verilog HDL code was correctly matching the truth tables. Implementing the Verilog HDL code on the DE2 board was by far the trickiest of the experiments. But after a few failed attempts we were able to verify that our Verilog HDL code of given schematics were

functioning properly by connecting inputs to toggle switches and the output to an LED, to signify either 1 or 0. This lab successfully helped us get more used to and comfortable with the various tools, syntax, and modes of the IDE's and DE2 board.