PHYS 234 Lab 1

Ethan Yang

Part A. Logic Gates

3-Input XOR

Verilog Code:

```
`timescale 1ns / 1ps
module lab1a(
    input a,
    input b,
    input d,
    output c
    );

assign c = a^b^d;
endmodule
```

To implement the 3-input XOR, we use the arithmetic representation ⊕ to conjunct the three input variables. The XOR symbol is represented by the bitwise operator ^ in the code snippet above.

After programming the code into the FPGA, we would be able to perform simple XOR arithmetics. Flipping the switches at “L3”, “P11”, and “K3” should trigger the corresponding LEDs. When the LED is lit up it represents 1 and when it’s off it represents 0. The truth table below demonstrates all the possible outcomes on the circuit board.

A	B	D	A^B^D
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

When even numbers of switches are turned on (1/true, equivalently), the output returns 0/false, whereas when odd numbers of switches are on, the output always returns 1/true. Associativity and commutativity properties apply to the XOR operations, therefore no matter if our formula is (a^b)^d, a^(b^d), or d^a^b, the outcomes would be the same.

Part B. Binary Arithmetic

Addition

Verilog Code:

```
`timescale 1ns / 1ps
module lab1b(
    input [3:0] a,
    input [3:0] b,
    output [4:0] c
    );

assign c = a + b;
Endmodule
```

The verilog code was programmed into the FPGA and used to find out all possible ways to sum to 11110 and 00011. The results are carried out by testing and flipping the switches until we get our desired outputs.

- 11110

a	b
1111	1111

The decimal representation of 1111 + 1111 = 11110 is 15 + 15 = 30. This is the only way to do so because 30 can only be summed to be maxing out both our 4 digits inputs.

- 00011

a	b
0000	0011
0011	0000
0010	0001
0001	0010

There are 4 ways to calculate the sum 00011: 0000 + 0011, 0010 + 0001, and flipping the orders of the variables. In decimal, 00011 is 3, and the calculations represented in decimal are 0 + 3 and 2 + 1.

Subtraction
Simply swap out the + sign on the “assign” line in the verilog code for the - sign to perform subtractions. The results for 7-3, 7-1, 3-7, and 1-7 on the FPGA board are shown in the table below:

a	**b**	**a - b**
7	3	**7-3**
0111	0011	0100
7	1	**7-1**
0111	0001	0110
3	7	**3-7**
0011	0111	1100
1	7	**1-7**
0001	0111	1010

The calculations for 7-3 and 7-1 are very straightforward, the results are 4 and 6, which in binary are 0100 and 0110. On the other hand, the results for 3-7 and 1-7 are represented in two’s complement since they are negative numbers. In two’s complement, the rightmost digit would be negative while the rest would remain positive. As a result 3-7=-4 will be 1100 (-8+4=-4) and 1-7=-6 will be 1010 (-8+2=-6).