Lab Assignment #1

Guideline

This lab is to be done individually. Each person does his/her own assignment and turns it in.

Objective

To learn designing basic combinational circuits in Verilog and implementing them on an FPGA.

Problem 1: Subtractor Design

a. Write Verilog code for a 1-bit full subtractor using logic equations. You may not use the subtraction operator '-'. Rather, use the given 1-bit subtractor truth table given below to derive logic functions.

b. Write Verilog code for a 4-bit subtractor using the module defined in part (a) as a component. Test it for the following input combinations:
   1. A = 1001, B = 0011, Bin = 1
   2. A = 0011, B = 0110, Bin = 1

1-bit full subtractor truth table:

A	B	Bin	Diff	Bout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Verify that your design works correctly by using the "force" and "run" commands in the transcript window to provide inputs and observe outputs on the waveform window.

Problem 2: ALU Design

Design an Arithmetic and Logic Unit (ALU) that implements 8 functions as described in Table 1. Table 1 also illustrates the encoding of the control input.

The 4-bit ALU has the following inputs:

- A: 4-bit input
- B: 4-bit input
- Cin: 1-bit input
- Output: 4-bit output
- Cout: 1-bit output
- Control: 3-bit control input

Table 1: ALU Instructions

Control	Instruction	Operation Pseudocode Description
000	Add	Output <= A plus B plus Cin; Cout contains the carry
001	Sub	Output <= A minus B minus Cin; Cout contains the borrow
010	Bitwise Or	Output <= A or B
011	Bitwise And	Output <= A and B
100	Shl	Output <= A[2:0] concatenated with '0'
101	Shr	Output <= '0' concatenated with A[3:1]
110	Rol	Output <= A[2:0] concatenated with A[3]
111	Ror	Output <= A[0] concatenated with A [3:1]

The following points should be taken care of:

- Use a case statement (or a similar 'combinational' statement) that checks the input combination of "Code" and acts on A, B, and Cin as described in Table 1.
- The above circuit is completely combinational. The output should change as soon as the code combination or any of the input changes.
- You can use arithmetic and logical operators to realize your design for Problem 2 only.
- Cout should be '0' when the control is not 'Add' or 'Sub'

Simulate this circuit by using the "force" and "run" statements in the transcript window to provide inputs and observe outputs on the waveform window.

Problem 3: Synthesizing and implementing the subtractor on the FPGA

Create a new project in Vivado. Use the code for the 4-bit subtractor that you wrote in Problem 1. Synthesize and implement the design on the Artix 7 FPGA on Basys3 board. Use the following pin assignments for creating the XDC file:

A	Switches[7->4]
B	Switches[3->0]
Bin	BTNU
Diff	LED[3->0]
Bout	LED4

Download the design onto the board and make sure it works as expected. Include the *design_name.bit* file that you download to the board in your Canvas submission.

Useful Information

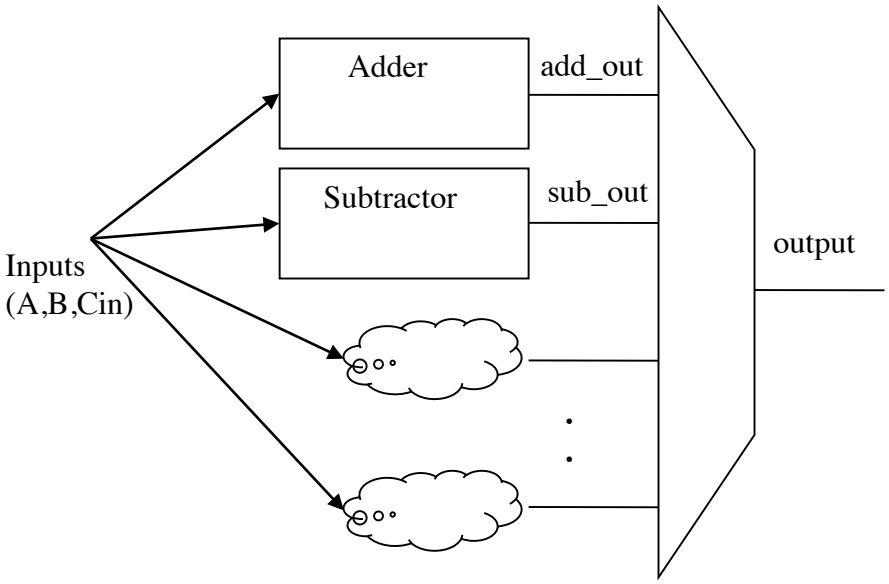
1. For problem 2, you can use the subtractor block from problem 1 for doing the subtraction (although just using the arithmetic operators will make your design easier). If you use the subtractor from problem 1, remember that we are designing hardware. So, doing something like the following is incorrect:

```
module xyz(…)
always @(…)
case (control)
0 : four_bit_sub sub_inst(in1,in2,bin,output);
……
```

First of all, it is important to realize that instantiating a module is not like 'calling' a function in C. Once instantiated, the module is always evaluating its inputs. Therefore, it cannot be conditional. It is always present. So, you should do something like this:

```
module xyz(…)
four_bit_sub sub_inst(in1,in2,bin,sub_out)
always @(…)
case (control)
0 : output <= sub_out;

……
```



2. Make sure that your designs work by testing them thoroughly. You should not just use the test inputs in the lab description. Also, it is always better to submit a do-file which has sufficient number of input combinations (not just the ones given in the lab description).

3. Do not use # statements in your design for providing delays.

   #15 X <= A or B;

   In fact, you should never use the delay statement in the lab during the semester.

Submission Details

All parts of this lab will be submitted on Canvas only. You will not need to submit anything as a hard copy. Please zip all relevant files into a single folder with the following naming scheme: ***Lastname_Lab#.zip***

Problem	Submission Requirements
1	• Verilog file(s) • Do-file
2	• Verilog file(s) • Do-file
3	• Bit-file • XDC File

Checkout Details

You will be expected to describe briefly the codes for problems 1 and 2, simulate and show waveforms in Modelsim, and answer verbal questions. Also, for the last problem you will have to demonstrate that your circuit works on the board.