ECE 443/543

Project #2

**General Guideline**

1. **Due on November 8, 5PM**
2. **You will receive 20 points deduction for every 12 hours late submission**
3. **You need to put the hard copy of your write-up report in my mail box (see the Section of Deliverable for details)**
4. **You need to submit your 7 VHDL Workspaces via Blackboard (see the Section of Deliverable for details). You have to zip them into one file and clearly name and organize them**
5. **You need to put the comments to your code**
6. **This is an individual lab assignment**
7. **ALDEC has to be used, not Quartus.**

**Objective**

For this lab, you will be implementing an ALU. **This ALU must be implemented structurally using VHDL.** The ALU must take two 16-bit input arguments (A and B) and produce a 16-bit output argument (R). A, B and R are **signed** number. The instruction will be selected using three signal lines (S2, S1 and S0). There will also be a 3 bit status output (status). The status output is described in Table 2. Table 1 list the name of the I/O lines, their size and direction.

Table 1 – I/O Lines

|  |  |  |
| --- | --- | --- |
| Name | Direction | Size |
| A | I | 16 |
| B | I | 16 |
| R | O | 16 |
| S2 | I | 1 |
| S1 | I | 1 |
| S0 | I | 1 |
| status | O | 3 |

Table 2 – status output description

|  |  |
| --- | --- |
| Bit Position | Meaning when high |
| 2 | overflow condition |
| 1 | result is zero |
| 0 | result is negative |

The instruction set is shown in Table 3. All undefined operations should result in R being zero.

Table 3 - Opcodes

|  |  |  |  |
| --- | --- | --- | --- |
| S2 | S1 | S0 | Operation |
| 0 | 0 | 0 | Signed Addition |
| 0 | 0 | 1 | Signed Multiplication |
| 0 | 1 | 0 | Passthrough A |
| 0 | 1 | 1 | Passthrough B |
| 1 | 0 | 0 | Signed Subtraction |
| 1 | 0 | 1 | Undefined |
| 1 | 1 | X | Undefined |

You must show that all of the parts of the ALU (adder, subtractor, multiplier and multiplexer) and the ALU itself work by showing the functional simulation diagrams. You should come up and show enough test cases to prove that it is working. For example, if you were testing the adder you could do 2+2, 2+(-2) and (-2)+(-2).

Hint:

The multiplexer should be an 8-input multiplexer. This selects the correct output for the instruction and there are 5 possible outputs.

This ALU works by executing all instructions at the same time. This means that no matter what the op-code is, it adds and multiplies the numbers. Take for instance this example where we have two 4-bit inputs and a 4-bit output.

    R1[0 .. 3] = add(A, B);

    R2[0 .. 3] = mult(A, B);

 R1 is the result of the addition of A and B. R2 is the result of the multiplication of A and B. If we want the output to be R, we can use a multiplexer to select the correct output. If the op-code is determined by S where if S = 0, the output is R1 and if S = 1, the output is R2.

   Multiplexer(R1(0), R2(0), S, R(0))

   Multiplexer(R1(1), R2(1), S, R(1))

   Multiplexer(R1(2), R2(2), S, R(2))

   Multiplexer(R1(3), R2(3), S, R(3))

  The multiplier should multiply two 16-bit numbers. That means that the result could require 32 bits. In order to take care of this problem, the status register should indicate an overflow condition if the result is greater than 16 bits.

In this project, the propagation delay for logic gates needs to be taken into consideration. Please assign 10 ns for each logic operation, AND, OR, NOT,…etc, as the propagation delay so that you can estimate how long it will take for the arithmetic operations. However, the multiplexer will be considered to work ideally without having propagation delay.

**Deliverable:**

1. Each VHDL Workspace including all codes and waveforms
   * Implement a half adder using gates (structural model) – (VHDL Workspace #1)
   * Implement a full adder using half adder and gates (structural model)– (VHDL Workspace #2)
   * Implement a 16-bit adder using full adder and gates (structural model) – (VHDL Workspace #3)
   * Implement a 16-bit subtractor using 16 bit adder and gates (structural model) – (VHDL Workspace #4)
   * Implement a 16-bit multiplier **using structural or behavioral model**– (VHDL Workspace #5)
   * Implement a 16-bit multiplexer using gates (structural model) - (VHDL Workspace #6)
   * Implement ALU using 16-bit adder, 16-bit multiplier, 16-bit multiplexer and gates (structural model)- (VHDL Workspace #7)
2. Reports
   * Your test results with your test cases for each VHDL Workspace above.
   * Answer following questions for ALU using 16 bit adder, 16 bit multiplier, 16 bit multiplexer and gates
     + How long does it take to add 1000 to 2000? You should show the timing diagram and explain it.
     + How long does it take to multiply 128 \* 128? Show the timing diagram\* and explain it.
     + Using the results from1 and 2, how many additions and multiplications per second can your ALU perform?
     + Describe some ways that you could optimize the design to make it faster.

**Grading**

* Implement half adder using gates –5 points
* Implement full adder using half adder and gates – 5 points
* Implement 16-bit adder using full adder and gates – 10 points
* Implement 16-bit subtractor using 16 bit adder and gates – 10 points
* Implement 16-bit multiplier using 16 bit adder and gates –10 points
* Implement 16-bit multiplexer using gates 10 points
* Implement ALU using 16-bit adder, 16-bit multiplier, 16-bit multiplexer and gates –20 points
* Write Up Report – 30 points