Lab 1 - Arithmetic and Logic Unit (ALU)

Introduction

Welcome to cs161L. In this class you are allowed to work on all labs alone, or in groups of two. This lab will be an introduction to microprocessors and the Verilog language. You will also be getting familiar with the Xilinx design environment. The goal of this lab is to implement a simple Arithmetic and Logic Unit (ALU) in Verilog.

ALUs are hardware circuits that perform the arithmetic computations within a processor. They support multiple operations like addition, subtraction, multiplication, division, square roots, etc. The hardware logic to perform these operations can vary widely based on the approach used (Carry-Lookahead vs Ripple-Carry adder) or the data types supported (Integer, Float, Double). An ALU could even supply multiple version of the same operation. They are not limited to only arithmetic operations. They can support bit-wise operations, like AND OR and NOT, as well. Input data and control bits are sent to the ALU. The control bits specify an operation, and the ALU redirects the inputs to the corresponding functional circuit. When the computation completes the result is output along with extra data about the operation (overflow, underflow, carryouts, etc.)

Before starting this lab you should be familiar with:

- Two's complement representation
- The Xilinx ISE Tutorial here. (For this lab you only need to go up to step 4, but the other steps will be useful for later labs.)
- Verilog Examples here

Deliverables

For this lab you are expected to build an ALU that supports 8 arithmetic operations. The ALU should be designed in such a way that the user can specify the operation's width without modifying the source code. In addition to the ALU you are also expected to build a test-bench that sufficiently verifies it's correctness.

- The module name should be named "my_alu"
- The module should use a parameter, called "NUMBITS", that specifies the operation's width
- The module should register all outputs

- The module should have input/output ports with the **EXACT** names listed below
- The module should support the operations listed below

Port Name	**Size**
A	N-bit input
B	N-bit input
opcode	3-bit input
result	N-bit output
carryout	1-bit Output
overflow	1-bit Output
zero	1-bit Output

Operation	**Opcode**
unsigned add	000
signed add	001
unsigned sub	010
signed sub	011
bit-wise AND	100
bit-wise OR	101
bit-wise XOR	110
Divide A by 2	111

The **carryout** port is the MSb's (Most Significant Bit's) carry out. The **zero** port should be '1' when the **result** port is all zeros.

The specification for the overflow and the carry out signals is as follows.

	A	***B***	***Result***
signed add	>= 0 < 0	>= 0 < 0	< 0 (Overflow) >= 0 (Overflow)
signed sub	>= 0 < 0	< 0 >= 0	< 0 (Overflow) >= (Overflow)
unsigned add	MSb's carryout is '1'		
unsigned sub	MSb's carryout is '0'		

Turn-In:

Each group should turn in one tar file to iLearn. The contents of which should be:

- A README file with the group members names, and any incomplete or incorrect functionality
- A Verilog file(s) with the ALU design.
- A Verilog file(s) with the test cases.