EE531 Advanced VLSI Design
Lab 1

Arithmetic Logic Unit (ALU) Design
In this lab you will design and implement an arithmetic logic unit. An ALU is controlled by an operational code (op code) used to select one of several possible arithmetic or logical operations for computation. Your ALU will operate on at most 2 8-bit, signed integer inputs which must be stored in registers A0 and B0. *The two registers are to be explicitly implemented in your SystemVerilog code.* Single input operations (e.g. shift) will operate on the contents of register A0, returning the result to register A0.

Operations to be implemented:

- Addition: A0 ← A0 + B0
- Subtraction: A0 ← A0 – B0
- Multiplication: A0,B0 ← A0 * B0
- Logical AND: A0 ← A0 and B0
- Logical OR: A0 ← A0 or B0
- Logical XOR: A0 ← A0 xor B0
- Logical NOT: A0 ← ~A0
- Logical Shift Left: A0 ← A0 lsl B0 ; B0 stores number of positions to shift
- Logical Shift Right: A0 ← A0 lsr B0 ; B0 stores number of positions to shift
- Arithmetic Shift Right: A0 ← A0 asr B0 ; B0 stores number of positions to shift
- Rotate Shift Left: A0 ← A0 rsl B0 ; B0 stores number of positions to shift
- Rotate Shift Right: A0 ← A0 rsr B0 ; B0 stores number of positions to shift

NOTE: For multiplication, A0,B0 acts as single 16-bit register with A0 storing the most significant digit (bits 15 – 8) of the result and B0 the lower digit.

In addition to the ALU operations above, include the following operations for loading and reading the registers:

- Load Register A0: A0 ← (contents of input[7:0])
- Load Register B0: B0 ← (contents of input[7:0])
- Read Register A0: (output_lower[7:0]) ← A0
- Read Register B0: (output_lower[7:0]) ← B0
- Read Register A0,B0: (output_upper[7:0]) ← A0, (output_lower[7:0]) ← B0

NOTE: Since the numbers being read are signed, use a 1-bit output port to display the sign bits for the three read operations. The MSB of inputs will be the sign bits for load operations.

Lab assignment tasks:

- **RTL Design & Verification:** Where reasonable, use operators and type cast functions to implement the ALU operations.
- Simulate original SystemVerilog design using Vivado
  * Make sure to clearly define a few test cases (for each operation) and expected results, simulation should clearly show correspondence with expected operation.

Deliverables:

- Original SystemVerilog Code for your ALU design – **include as Appendix A in report**
- Testbench SystemVerilog Code – **include as Appendix B in report**
- Simulation results of original SystemVerilog design **– include as snapshots in writeup**

NOTE: You are to create your own cover sheet for all labs and the final project report. The cover sheet should clearly indicate your name and lab/project title.

Reports are to be neat and clear. Make sure you have reasonable sections, indicating what you're doing, the approach taken and your results. When including simulation results, make sure you indicate clearly what your expectation was and also clearly indicate what the results tells you. Finally, make sure to include a conclusion that describes what was challenging and what you learned from this exercise.