6.6 Experiment 6 (Arithmetic Logic Unit)

6.6.1 Aim

In this experiment, your knowledge to implement an Arithmetic Logic Unit (ALU) which is explained in section 4.9 of the course book will be tested.

6.6.2 Problem

You will implement verilog code for a 5-bit arithmetic logic unit (ALU). In ALU, there will be 2-bit select input (S), 5-bit data inputs (X, Y), 5-bit output (F), 1-bit output (C_{out}), 1-bit output ($Overflow$), and the functional table will be as follows:

S	Negative	Overflow	Zero	C_{out}	**F**	**Representation**
00	0	0	1	X ! = Y	00000	Unsigned
01	0	0	X[4:2] × Y[3:1]			Unsigned
10	X + (Y[3:0] / 2)					2's Complement
11	X + Y + 1					2's Complement

Note:

- Let's say Y = 10110; if the representation is unsigned, Y[2:0] will be 6 in decimal; if the representation is 2's complement signed, Y[2:0] will be -2 in decimal.
- Overflow output is only valid for the operations performed in 2's Complement representation (namely third and fourth operation shown in the above table). For other cases, it is always 0.
- Zero output will be 1 when the F value become 0.
- Negative output will be 1 when the F value become negative for 2's complement operations.
- Consider overflow only for addition and subtraction.
- For the S = 10, use Y[3:0] as 4 bit an unsigned number.

6.6.3 Preliminary Work

Before the experiment, you should prepare the following materials:

1. Design circuits with minimum number of components for each function of the ALU.
2. Merge all operations with select inputs and organize outputs.
3. Try to minimize your implementation by using repetitions (see the hint below).
4. Draw final circuit as the final design of the ALU with corresponding. functional tables and calculate the number of chips in your design.

5. Write the behavioral level verilog code of the ALU.
6. Write the verilog code for the testbench waveform in order to test all possible input combinations.
7. Verify the functionality of your implementation.

Then, follow the steps described in Section 3.1 to submit your preliminary work.

Hint: You can use one operation to implement another operation. For example, you can use subtraction for the comparison operation or addition for the multiplication operation.

6.6.4 Lab Work

1. Load your submitted code given by the assistants and demonstrate that the verilog code works with the same principles as the ALU assigned for your lab.
2. Do the same procedure as in preliminary work for the problem definition given in the lab section and demonstrate your implementation