1 Excercise 6: ALU Implementation

For this exercise we were asked to implement a 4 bit Aritmetic Logic Unit (ALU). The operations we had to develop were SUM, SUBSTRACTION, AND, OR, NOT, XOR, two's complement and shift left. For this, we decided to create a module responsible of adding 2 bits, and as output, it returned 2 bits, one bit for the answer, and another for the carry bit. By using this module, we decided now, to create a secondary module, responsible for adding 3 bits. This decision gave us al lot of simplification in the development of the module SUM for 4 bits. As you can see in the code "sum.v" found in the folder src, we commented the previous development without the module sum3Bits, and the new development with the module sum3Bits.

For the substaction, we decided to re-use the module created on exercise 4 of two's complement, and utilizing it correctly with the module SUM, we had our SUBSTRACTION module. For the operations AND, OR, NOT, XOR we chosen to use the predefined modules provided by verilog and utilize them bitwise.

1.1 Definitions

In this Aritmetic Logic Unit, we implemented with two accumulators (that we will call A and B), each one of four bits, three operational bits, four bits for the output accumulator (that we will call accumulator C) and one carry bit, ordered in the way they were mentioned.

To select the operation you want to make, you should turn the three operational bits in the following way:

	Bit 0	Bit 1	Bit 2
AND	0	0	0
NOT	0	0	1
OR	0	1	0
XOR	0	1	1
SHIFT LEFT	1	0	0
SUM	1	0	1
SUBSTRACTION	1	1	0
TWO'S COMPLEMENT	1	1	1

Table 1: Representation of operational bits

- AND: Performs an AND operation bitwise between acummulators A and B and returns it on accumulator C, meanwhile, the carry bit is left to zero.
- NOT: Performs a logic NOT operation bitwise between acummulators A and B, and returns the answer in the accumulator C. The carry bit stays as zero
- OR: Performs a logic OR operation bitwise between accumulators A and B, and returns the answer in the accumulator C. The carry bit stays as zero.
- XOR: Performs a logic XOR operation bitwise between accumulators A and B, and returns the answer in the accumulator C. The carry bit stays as zero.
- SHIFT LEFT: It manages to move every bit in accumulator A, one space left, and insterts a logic zero to the less significant bit. The answer is given in the C acccumulator and the carry bit will become 0 or 1 dependig on the most significant bit of A.
- SUM: Performs a numeric sum of the binary values of accumulator A and B and the answer is given in accumulator C. Depending on the overflow, the carry bit will become 1 or 0.

- SUBSTRACTION: Performs a numeric substraction of the binary values of accumulator A and B and the answer is given in accumulator C. The carry bit will become 1.
- TWO'S COMPLEMENT: Performs a two's complement of the binary value of accumulator A and the answer is given in accumulator C. The carry bit will become 0.