1 Exercise 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 Definiciones

Esta unidad aritmetica lógica se implemento con 2 acumuladores (que llamaremos A y B en este orden) de 4 bits cada uno, tres bits de operaciones, 4 bits del acumulador de salida (que llamaremos Acumulador C) y un bit de carry, ordenados en el orden en que fueron mencionados. Para seleccionar que operacion se desea hacer, se deben encender los 3 bits de operaciones de la siguiente manera:

	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: Representacion de operaciones en los bits.

- AND: Realiza una operacion AND bit a bit entre acumuladores A y B y la devuelve en el acumulador C, a su vez, el carry se devuelve en cero.
- NOT: Realiza una operacion logica NOT bit a bit del acumulador A y devuelve la respuesta en el acumulador C, a su vez, el carry queda en cero.
- OR: Realiza una operacion OR bit a bit entre acumuladores A y B y la devuelve en el acumulador C, a su vez, el carry se devuelve en cero.
- XOR: Realiza una operacion XOR bit a bit entre acumuladores A y B y la devuelve en el acumulador C, a su vez, el carry se devuelve en cero.
- SHIFT LEFT: Se encarga de mover cada bit del acumulador A un espacio a la derecha e inserta un cero al bit menos significativo, la respuesta la devuelve en el acumulador C y el carry valdra 0 o 1 dependiendo del bit mas significativo de A
- SUM: Se encarga de hacer una suma numerica de los valores decimales de los acumuladores A y B y devuelve el resultado (en codigo binario) en el acumulador C. Dependiendo si es representable el resultado de la suma en un nibble se encendera o no el bit Carry.

- SUBSTRACTION: Se encarga de hacer una resta numerica de los valores decimales de los acumuladores A y B y devuelve el resultado (en codigo binario) en el acumulador C. El carry se devuelve en 1.
- TWO'S COMPLEMENT: Se encarga de hacer el complemento a 2 del acumulador A y devuelve el resultado en el acumulador C. El carry se devuelve en cero.