Arquitectura de Computadoras

Ingeniería en Computación FCEFyN - UNC

Trabajo Práctico 1 - ALU

Izquierdo Agustina Mat: 37729473

Salvatierra Andrés Mat: 39611008

11 de septiembre de 2019

1. Introducción

Para el primer trabajo práctico de la materia se realizó la implementación en FPGA de una Unidad Aritmética Lógica (ALU). Se desarrolló para la placa Basys 3 utilizando el software Vivado 2018.2. El módulo posee un bus de datos parametrizable.

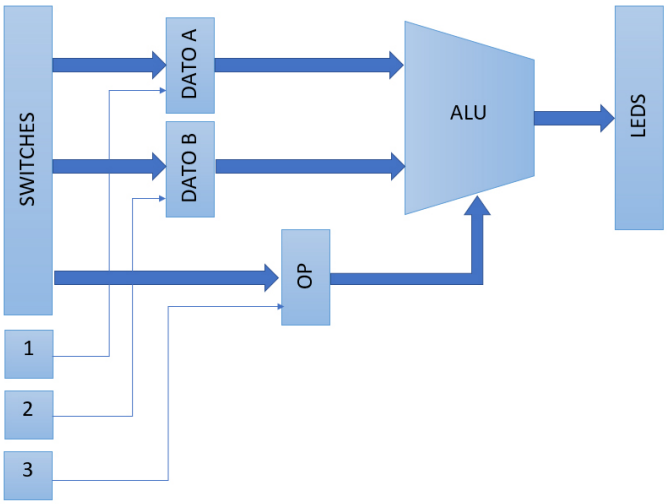

Figura 1: Diagrama de la ALU

La ALU soporta las siguientes operaciones, con los siguientes códigos:

Operación	Código
ADD	100000
SUB	100010
AND	100100
OR	100101
XOR	100110
NOR	100111
SRA	000011
SRL	000010

Cuadro 1: Operaciones de la ALU

2. Desarrollo

Para el desarrollo del módulo se definió un bloque llamado ALU, con los siguientes parámetros externos:

1. NB_DATA: numero de bits de los registros de entrada
2. NB_OPERADOR: numero de bits utilizado para representar las operaciones ya mencionadas

Las siguientes entradas y salidas del modulo:

1. i_dato_a [NB_DATA -1 : 0]: entrada del modulo, dato a
2. i_dato_b [NB_DATA -1 : 0]: entrada del modulo, dato b
3. i_operador [NB_OPERADOR -1 : 0]: entrada del modulo, representa la operación a realizar
4. o_resultado [NB_DATA -1 : 0]: salida del modulo, representa el resultado de la operación

Se desarrolló un módulo que se definió ALU_toplevel, con los siguientes entradas y salidas:

1. sw [NB_DATA -1 : 0]: switches de entrada de datos por el bus
2. btnL: botón izquierdo entrada dato a
3. btnC: botón central entrada dato b
4. btnR: botón derecho código de operación
5. clk: entrada de clock
6. led [NB_DATA-1 : 0]: salida del resultado por los leds de la placa

Para almacenar los valores ingresados se definieron dos registros A y B, que mantienen los números con los que se quiere operar. El resultado se almacena en el registro o_resultado.

3. Schematic

3.1. General

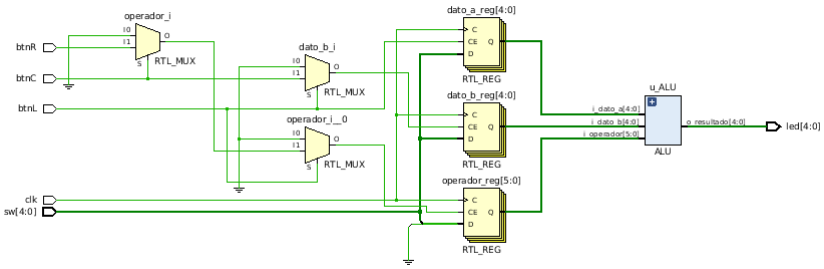

Figura 2: Diagrama general de la implementación

3.2. ALU

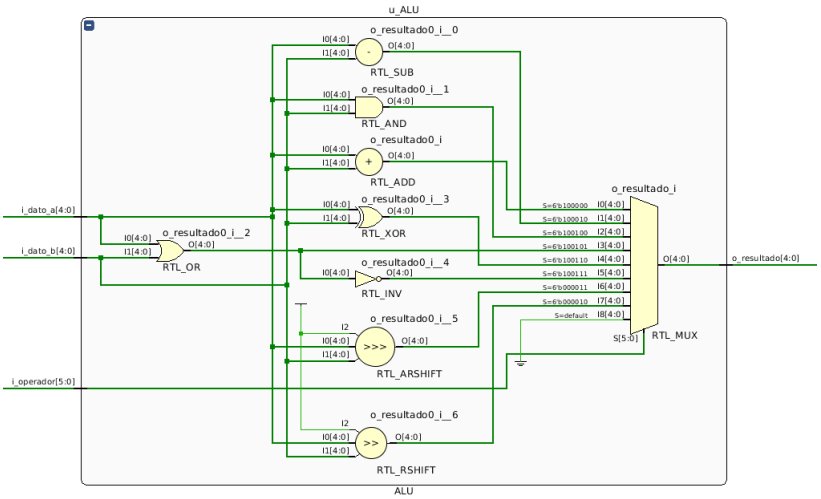

Figura 3: Diagrama modulo ALU

4. Test Bench

A la hora de realizar los tests del módulo se programó un test bench que compruebe el funcionamiento de cada una de las operaciones que debe realizar nuestra ALU, las simulaciones se realizaron con 5 bits.

1. Suma: 2 + 10 = 12
2. Suma con overflow: 10 + 10 = -12
3. Resta: 6 - 7 = -1
4. AND: 10101 & 00111 = 00101
5. OR: 11001 | 01011 = 11011
6. XOR: 11111 Ô1101 = 10010
7. NOR: 00110 ⌈ 10001 = 01000
8. SRA: 10110 »>11 = 11110
9. SRL: 11010 »11 = 00011

El procedimiento fue codificar un clock simulado mediante una sentencia always, y dentro de un bloque initial escribir uno por uno los tests anteriores, simulando la entrada manual de los valores.

La siguiente captura de pantalla muestra los resultados de las anteriores pruebas de funcionamiento.

La primera fila representa el dato a, la segunda el dato b, la tercera la operación y la última el resultado obtenido.

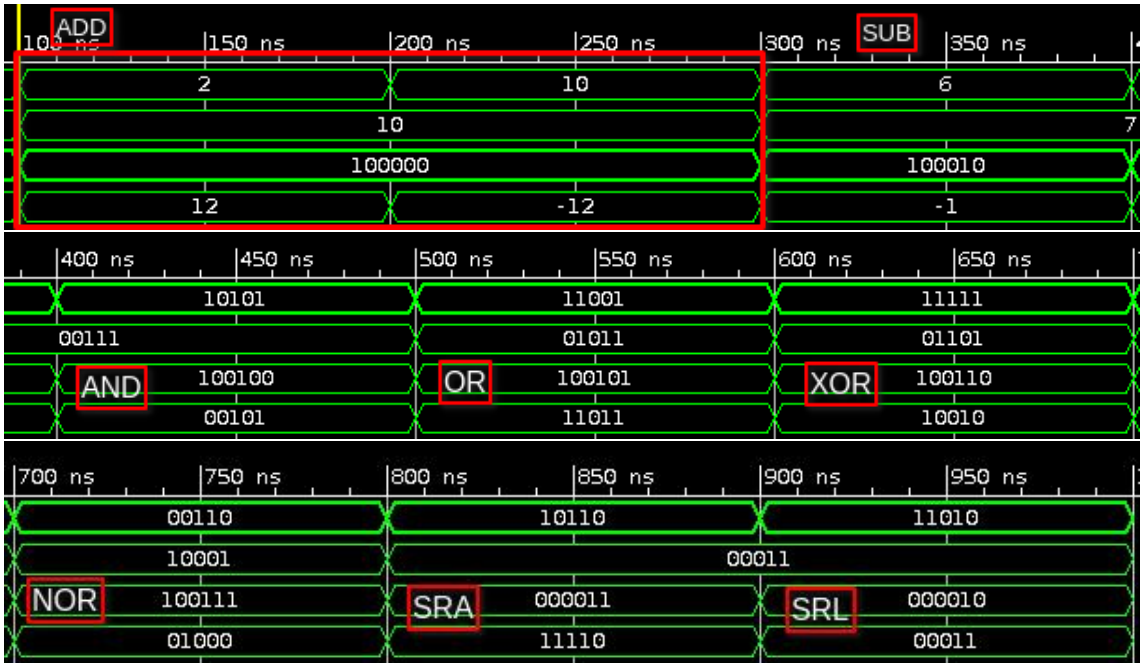

Figura 4: Resultado del Test Bench