ARQUITECTURA DE COMPUTADORAS

TRABAJO PRÁCTICO N° 1

ALU

Docentes:

- Micolini, Orlando
- Rodríguez, Santiago

Integrantes:

- Abratte, Diego
- Moral, Ramiro

Desarrollo:

El desarrollo del trabajo práctico se basa en la implementación en FPGA de una Unidad aritmética lógica (ALU, por sus siglas en inglés). Para la descripción de dicho bloque, se utilizó Vivado 2018 y será presentado en una FPGA Basys III.

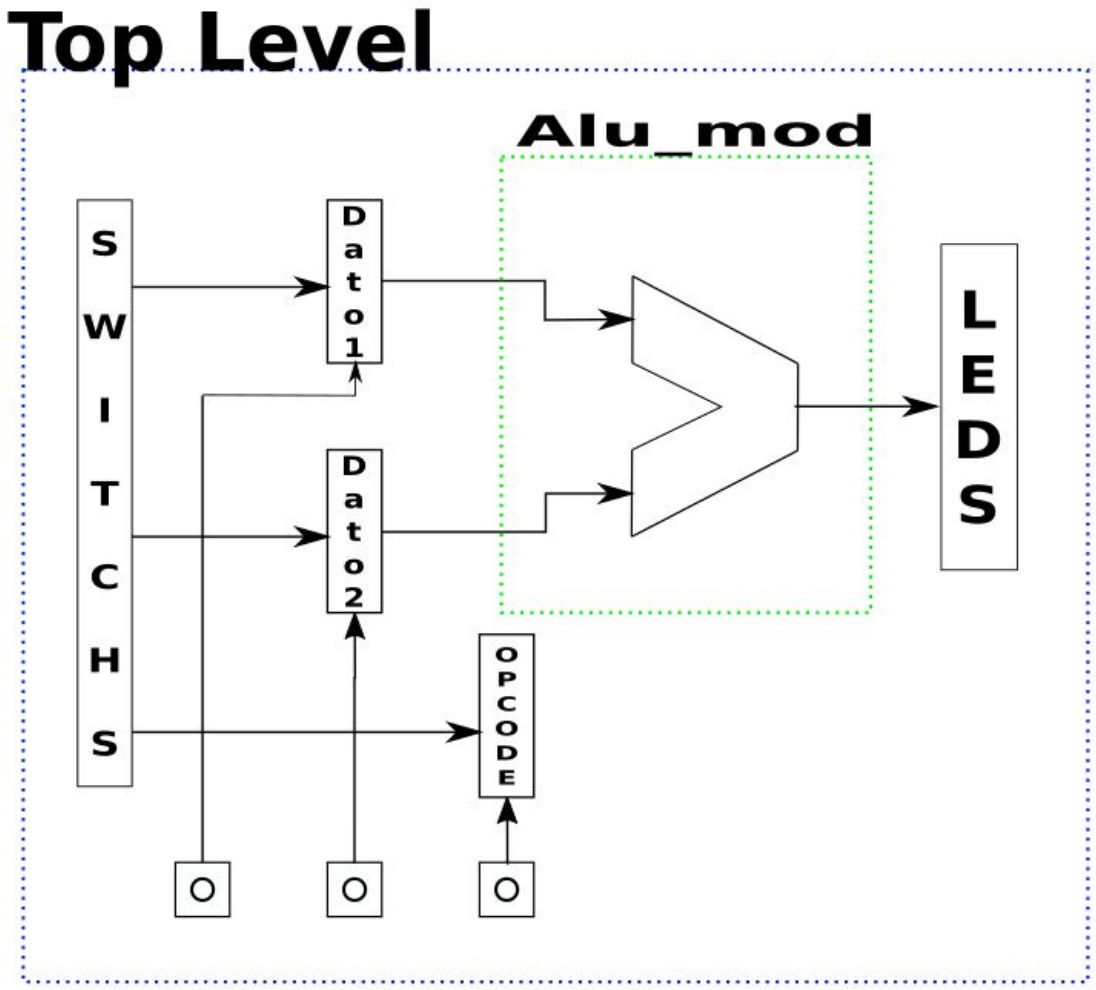
Una unidad aritmética lógica, es un circuito digital encargado de realizar las operaciones de suma, resta, operaciones lógicas (and, or, nor y xor) y operaciones de desplazamiento lógico y aritmético.

Para realizar esto, se conviene en un código de 6 bits para cada una de las operaciones a realizar en la ALU. Dicho código es el siguiente:

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

Tanto los operandos como el código de operación deben ser ingresados por los switchs de la FPGA. Los operandos y el resultado de la ALU son palabras de 8 bits de largo. El resultado de la ALU será representado por los LEDs de la FPGA.

De este modo, el diagrama de arquitectura queda como lo siguiente:



Por lo que se puede observar, dividimos la implementación en 2 módulos; El top level y de la ALU.

Las entradas, salidas, registros y cables utilizados son los siguientes:

```
input   [ BUS_LEN - 1 : 0 ] i_sw      //Entradas para operadores y opcode
input                       i_btnC,   //Latch opcode
input                       i_btnL,   //Latch ope1
input                       i_btnR,   //Latch ope2
input                       i_btnU,   //Reset
input                       i_clk,    //Clock
output  [ BUS_LEN - 1 : 0 ] o_led,
output          [ 2 : 0 ] debug_led
```

//Registros de trabajo
reg [BUS_LEN - 1 : 0] datoA;
reg [BUS_LEN - 1 : 0] datoB;
reg [`OP_LEN - 1 : 0] opcode;

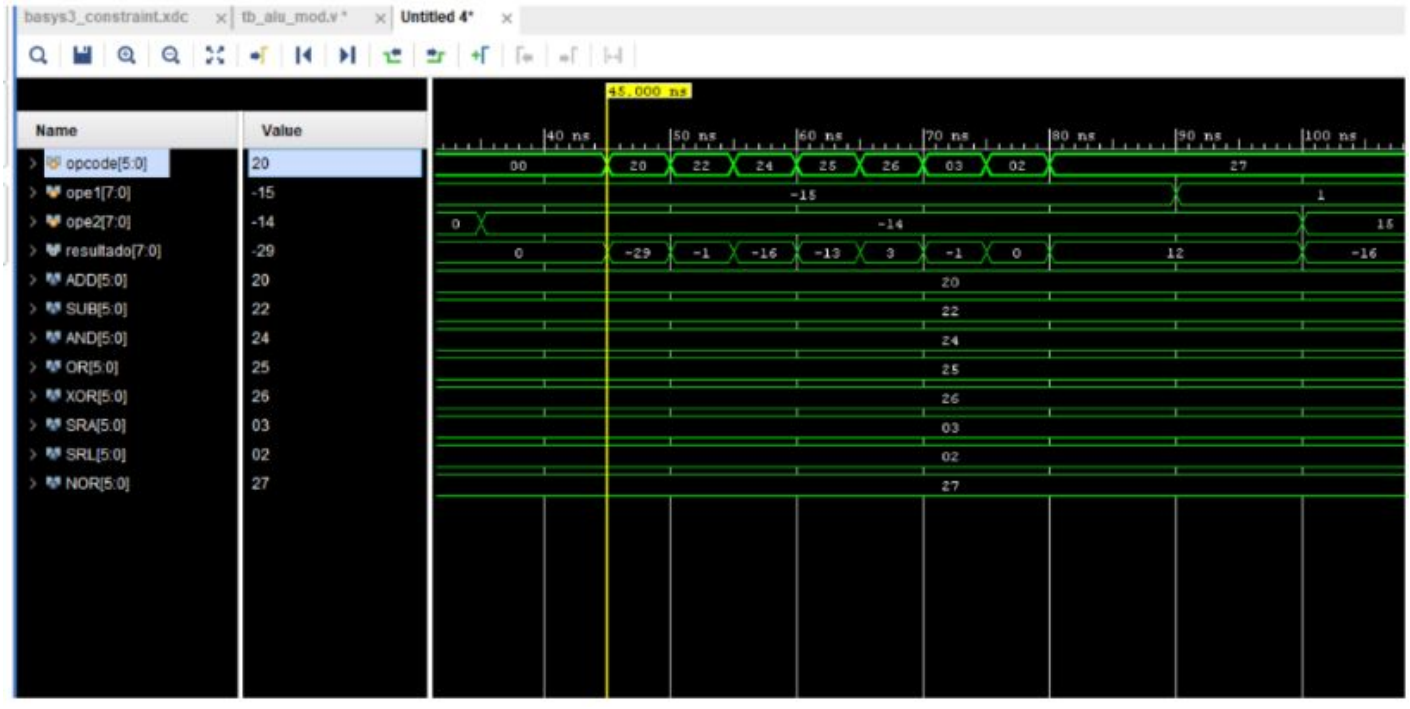
//Para deteccion de flanco. Estos registros son utilizados para comparar el estado actual de los botones con el estado anterior, de tal manera que sólo una transición de 0 a 1 en dichos botones latchean los registros.
reg latchA;
reg latchB;
reg latchOP;

//HW debug para indicar ingresos. Prenden un led dependiendo del operador/opcode ingresado.
reg [2:0] in_saved;

//Cables encargados de conectar el circuito combinacional de los módulos.
wire reset;
wire saveA;
wire saveB;
wire saveOP;
wire [BUS_LEN - 1 : 0] ope1;
wire [BUS_LEN - 1 : 0] ope2;
wire [`OP_LEN - 1 : 0] code;
wire [BUS_LEN - 1 : 0] resultado;

Test Bench:

Se realizó la simulación de los módulos descritos probando los diferentes valores de los opcode y su correspondiente resultado.



Aquí podemos ver una representación decimal signada de los valores de los operandos ingresados, siendo **ope1 = -15** y **ope2 = -14,** se puede comprobar rápidamente que la suma nos da -29, la resta nos da -1, etc..

Para verificar el funcionamiento, se realizó una segunda simulación cambiando los valores de los operandos. En este caso, **ope1 = 1** y **ope2 = 15**. Podemos comprobar viendo la imagen que da los valores esperables para los diferentes códigos de operación, la suma da 16, la resta da -1, etc..