# ARQUITECTURA DE COMPUTADORAS

## Trabajo Práctico 1 - ALU

Casabella Martin, 39694763
martin.casabella@alumnos.unc.edu.ar

Morales Julian, 35046503
jmorales@unc.edu.ar

## I. Introducción

Para el primer trabajo práctico de la materia se realizó la implementación en FPGA de una Unidad Aritmética Lógica (ALU). Se desarrolló trabajando con la placa Basys 3 de digilent, y su FPGA Artix, utilizando el software Vivado 2018.2. El módulo posee un bus de datos parametrizable, junto con la longitud del código de operación, a fines de escalabilidad en el diseño y uso posterior en los siguientes trabajos.

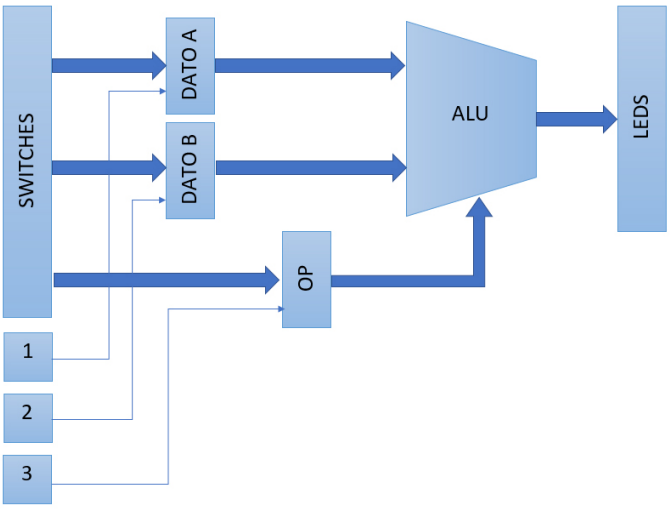


Figure 1: Diagrama de la ALU

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

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

Table 1: Operaciones de la ALU

## II. Desarrollo

Para el desarrollo del módulo se definió una jerarquía de modulos conformada por el modulo de la ALU, integrado en un top level, como se muestra en la figura:

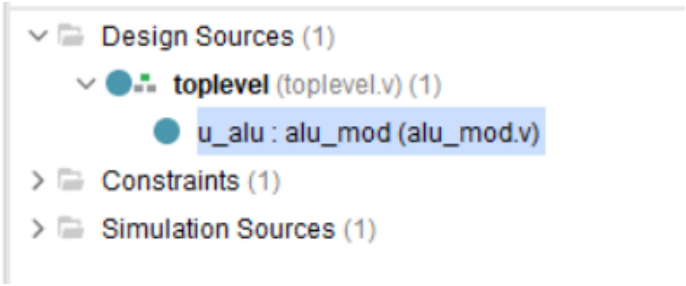


Figure 2: Jerarquía de módulos

El módulo de la toplevel, esta conformado por los siguientes puertos y parametros:

- parameter $[BUS\_LEN - 1 : 0]$ **BUS_LEN** : Parámetro que define el tamaño del bus
- input $[BUS\_LEN - 1 : 0]$ **i_sw** : Bus por donde el usuario ingresa ya sea un operando, o el opcode (switches físicos)
- input **i_btnC** : Botón central que se pulsa para almacenar el opcode
- input **i_btnL** : Botón izquierdo, que se pulsa para almacenar el operando 1
- input **i_btnR** : Botón derecho, que se pulsa para almacenar el operando 2
- input **i_btnU** : Botón superior, asignado al reset del sistema
- input **i_clk** : Clock de referencia sintetizado
- output $[BUS\_LEN - 1 : 0]$ **o_led** : Puerto de salida donde se mostrara el resultado (cableado a LEDs de la placa)

Para almacenar los valores ingresados se definieron dos registros *datoA* y *datoB*, que mantienen los operandos, mientras que otro registro introducido *opcode*, almacena el código de operación ingresado. A su vez, se define un wire que se le asigna a la instancia de la ALU para cablear el módulo top al registro del módulo de la ALU que almacena el resultado de la operación hecha. El mismo, va a los LEDs, como se mencionó anteriormente.

Una vez definido el tamaño de BUS deseado, el módulo se centra en un bloque **always** que se ejecuta con cada flanco positivo del clock. Si se detecta que alguno de los dos elementos de control están presionados (botón izquierdo o central) se almacena en el registro correspondiente el valor en los switchs. Se encienden 3 LEDS asignados para indicar el ingreso con su respectiva secuencia, a modo indicativo. Si el opcode es inválido, nada se visualiza en los LEDs. En cualquier otro momento, simplemente se copia el valor anterior de cada registro para mantenerlo guardado en el Flip-Flop.

Para el ingreso por pulsadores o botones, se realiza una detección de flanco, siendo normal cerrados los pulsadores de la placa. Para ello, se introduce un registro auxiliar, y se testea el estado del mismo en relación al pulsador, para saber si realmente el usuario pulsó el botón. Vemos la lógica de la detección de flanco en la siguiente porción de código:

```
assign saveA  = (latchA == 0 &&  i_btnL == 1) ? 1'b1 : 1'b0;
assign saveB  = (latchB == 0 &&  i_btnR == 1) ? 1'b1 : 1'b0;
assign saveOP = (latchOP == 0 &&  i_btnC == 1) ? 1'b1 : 1'b0;
```

## III. Test Bench

A la hora de realizar los tests del módulo se elaboró un test bench o archivo de simulación que compruebe el funcionamiento de cada una de las operaciones programadas en la ALU. Veamos el comportamiento del sistema en la siguiente figura:

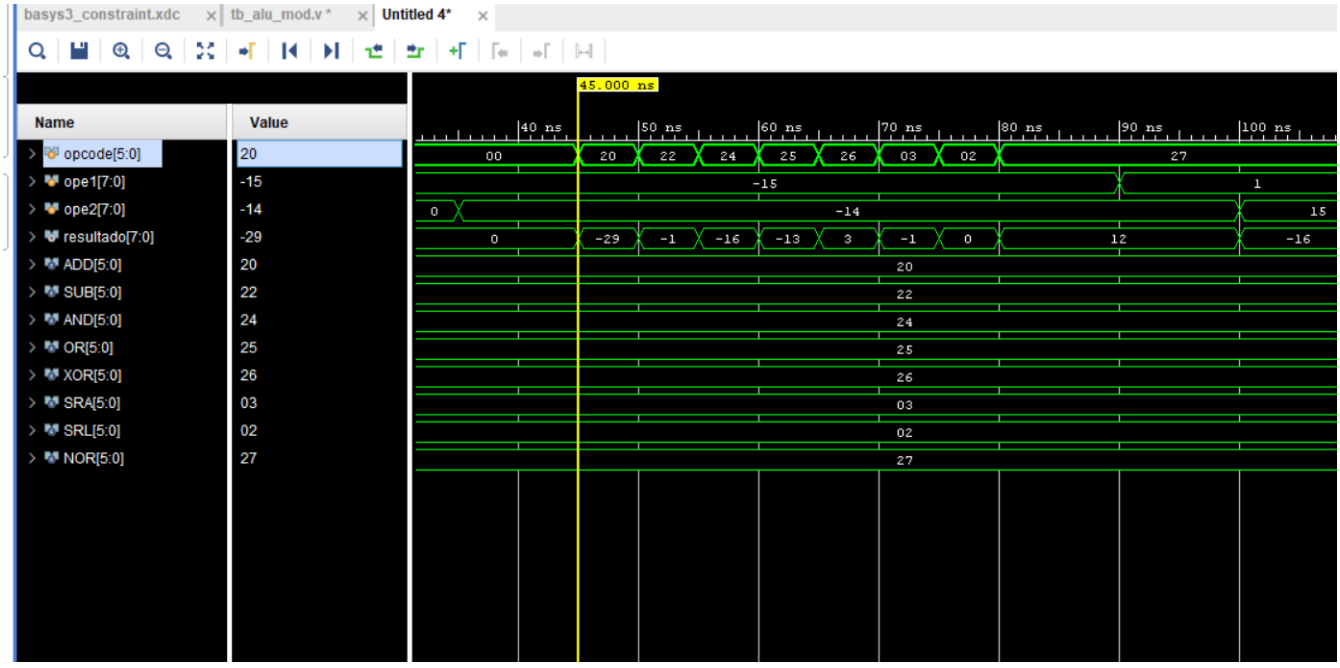


Figure 3: Primer test bench realizado

Se ve al costado izquierdo de la figura, con representación decimal signada (tal y como se declararon los puertos de trabajo) los dos operandos ingresados.
A su vez, los parametros *ADD*, *SUB*, *AND*, *OR*, *XOR*, *SRA*, *SRL*, y *NOR*, internos al archivo de simulación, nos facilitan con la misma notación, el código de operación ingresado y por ende, la operación que se esta realizando en el instante de simulación dado.

Veamos con otros operandos distintos, y la misma secuencia de operaciones:

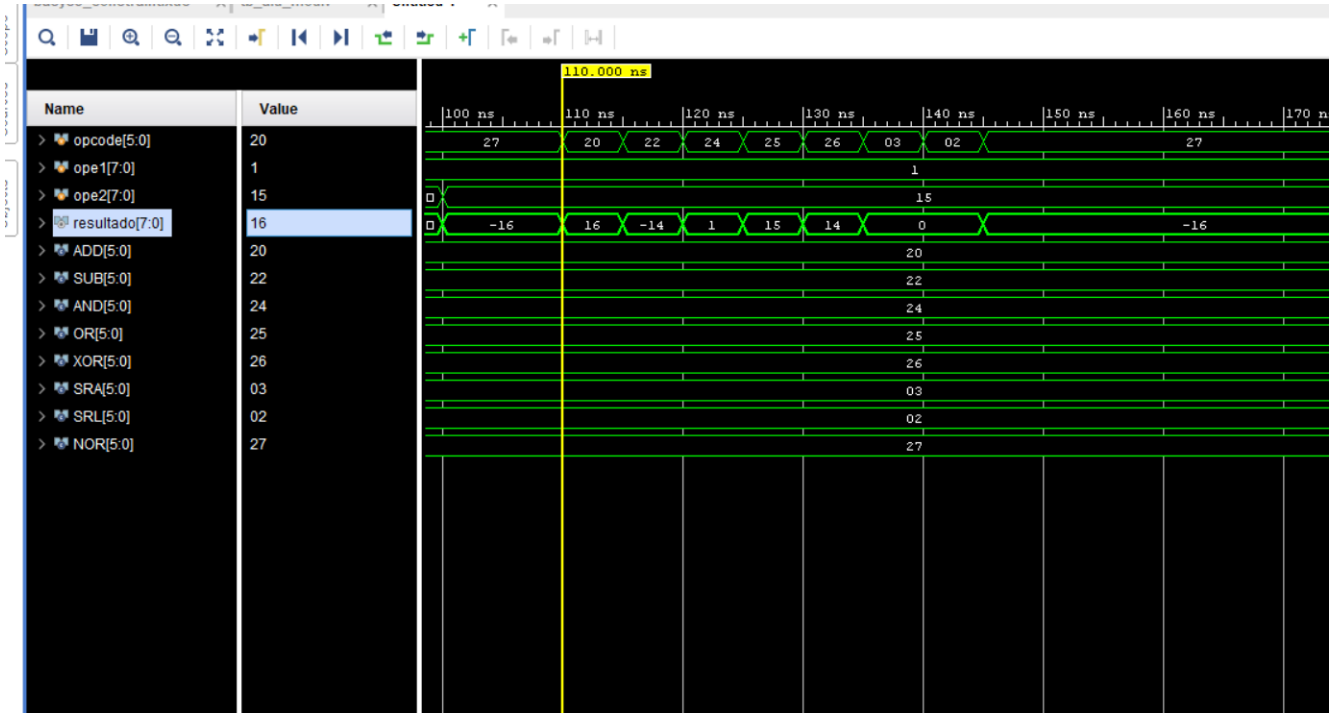


Figure 4: Segundo test bench realizado