

**ARQUITECTURA DE COMPUTADORAS**

**Trabajo Práctico 1**

Profesores:

* Santiago Rodriguez
* Martin Pereyra

Alumnos:

* Sebastian Klincovitzky 41158451
* Ivo Ferrari 40326730

[**Objetivos**](#_awxh1qn0bacb) **3**

[Sistema a desarrollar](#_hip5h3c1x7fw) 3

[Operaciones](#_cvopyf7a1vh8) 4

[**Módulos usados**](#_oi0gpxowlrlb) **4**

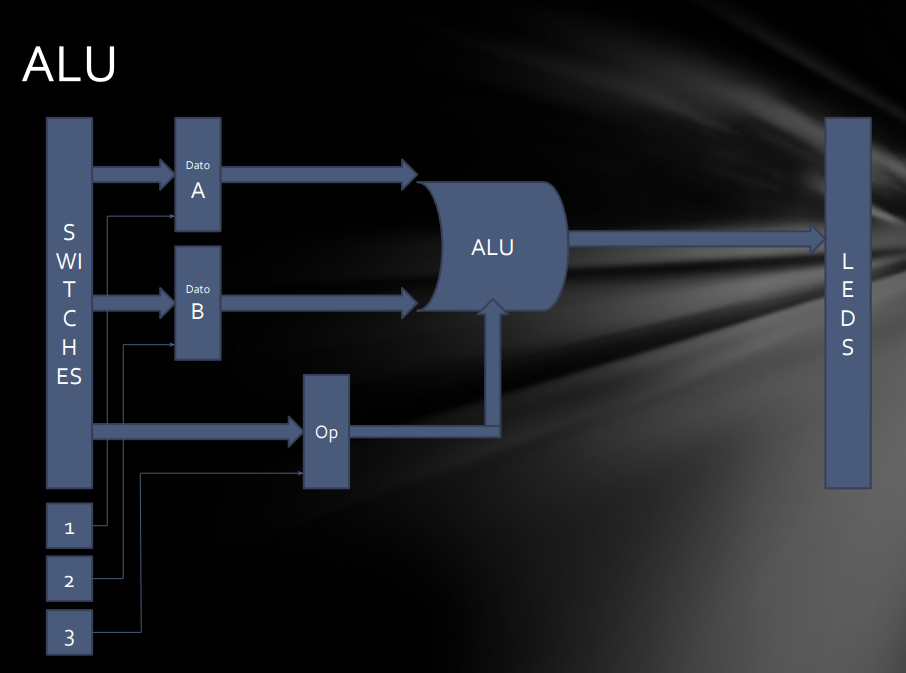
[**TestBench**](#_3c4w0fdojp15) **4**

[**Simulación**](#_ke51dcrg05zl) **4**

## Objetivos

1. Implementar en FPGA una ALU
2. La ALU debe ser parametrizable (bus de datos) para poder ser utilizada posteriormente en el trabajo final.
3. Validar el desarrollo por medio de Test Bench.
4. El testbench debe incluir generación de entradas aleatorias y código de chequeo automático.
5. Simular el diseño usando las herramientas de simulación de vivado incluyendo análisis de tiempo.

### Sistema a desarrollar



### Operaciones



## Módulos usados

Para el desarrollo de este trabajo creamos 2 módulos:

### ALU

Este módulo se compone de un bloque always basado en la señal posedge de un clock para la actualización de registros dependiendo la operación realizada, dentro del mismo. Se cargan también las operaciones a utilizar, ADD, SUB, AND, OR, XOR, SRA, SRL y NOR.

Se instancian también los parámetros LEDS, como una salida y cable, y también Data\_A, Data\_B, Op y clock pero a diferencia que estas fueron como input, todas de 6 bits excepto por supuesto el clock. Las entradas y salidas se rigen bajo un mismo parámetro, de forma que en caso de querer modificar el bus se puede realizar fácilmente con el cambio de un solo valor, diferente del parámetro usado para las operaciones, dado que se rigen por un código de 6 bits si o si.

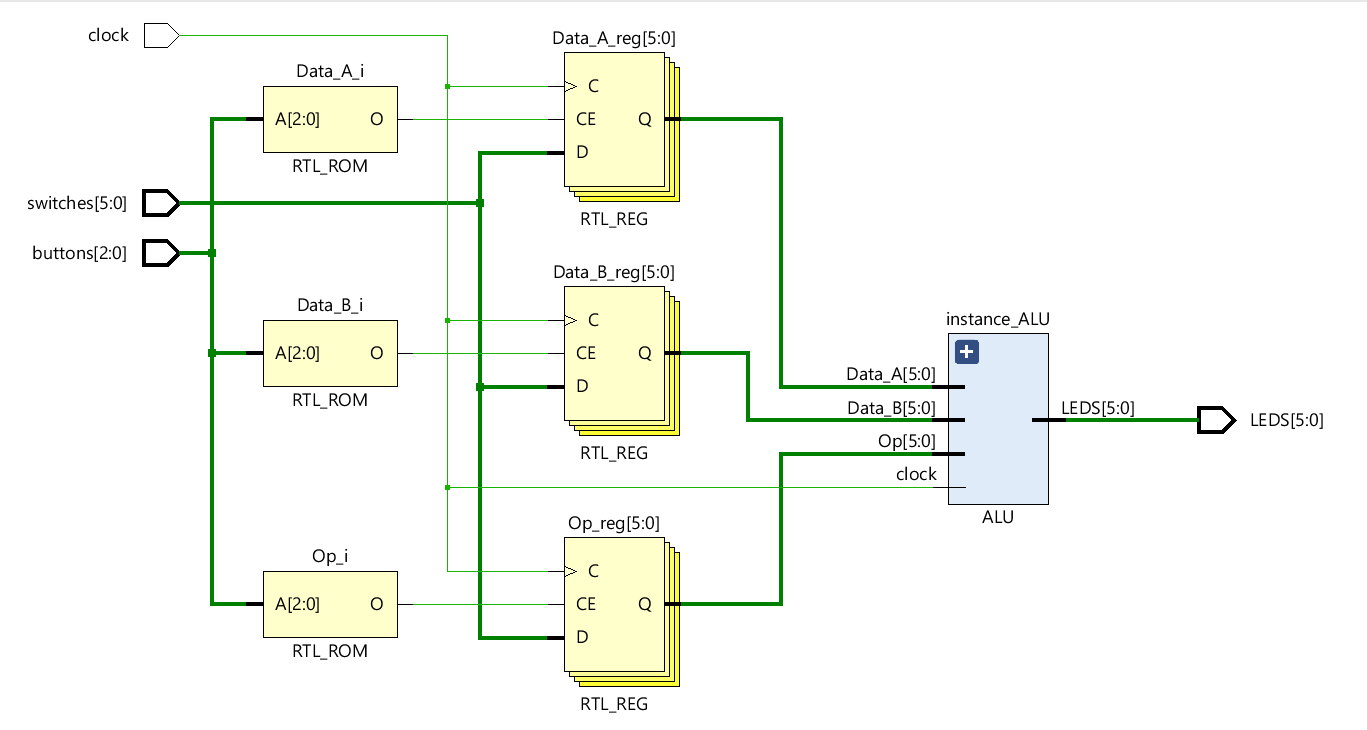
Este se implementa como el principal por lo que en el mismo se instancia el input options a partir de los registros necesarios para cada parámetro.

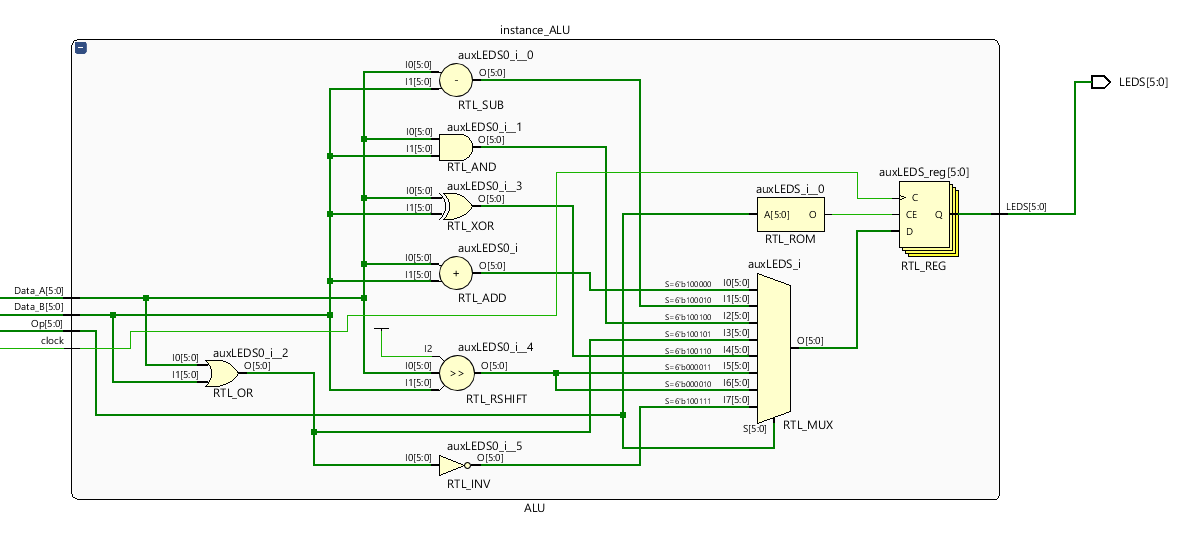
### INPUT OPTIONS ALU

Compuesto de un bloque always basado en la señal posedge de un clock para la carga de datos a través de un switch de 6 bits.

Instanciamos aquí como salida y wire el parámetro LEDS, y como entrada y wire también, switches con los 6 bits, buttons de 3 bits y clock.

En este módulo pasamos variables a nuestro módulo core, que es el módulo ALU.



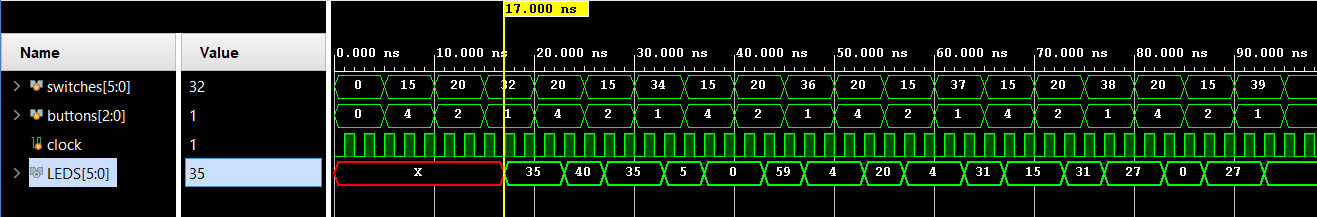


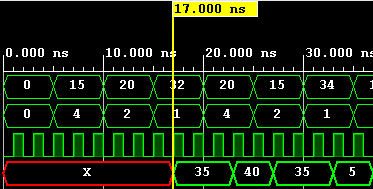
## TestBench

En nuestro trabajo, hemos realizado hasta tres TestBench:

1. TestBench hardcodeado para probar la funcionalidad de la ALU
2. TestBench hardcodeado para probar la funcionalidad de input options ALU
3. TestBench que genera pruebas aleatorias y luego se corrigen automáticamente.

## Simulación





Para mayor facilidad en la exposición, se adjuntan imágenes de la simulación del TestBench del input options ALU, pudiendo ver todas las operaciones programadas en la primera imagen.

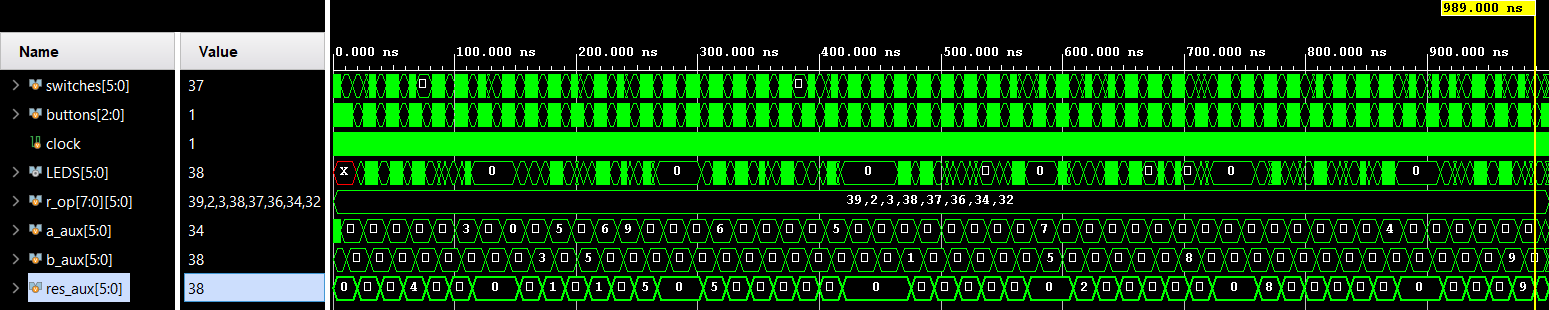
En la segunda imagen, se puede ver en mayor tamaño las primeras pruebas para poder ver en mayor detalle y facilitar la explicación:

De arriba para abajo se pueden ver los valores de

1. Switches (quien se encarga de cargar los valores)
2. buttons (quien se encarga de indicar a qué módulo queremos cargar el valor)
3. clock (configurado en el TestBench para que cambie constantemente de valor entre 1 y 0)
4. LEDS (que mostrarían el resultado de la operación)

Teniendo esto en cuenta, de izquierda a derecha:

1. Valores inicializados en 0.
2. Se carga el valor 15 en Data\_A (buttons 4 = 100, se está apretando el botón 1).
3. Se carga el valor 20 en Data\_B (buttons 2 = 010, se está apretando el botón 2).
4. Se carga el código para ADD en operations (d.32 = b.100000 y buttons 1 = 001, se está apretando el botón 3).
5. Inmediatamente, se puede ver que LEDS ahora muestra el valor de 35, que es la suma de 15 y 20.
6. Se carga un 20 en Data\_A, como Data\_B sigue teniendo un 20, y operations marcó por última vez ADD, el valor de LEDS vuelve a realizar la suma = 40.
7. Se carga un 15 en Data\_B, por las razones expresadas anteriormente, LEDS vuelve a cambiar.
8. Se carga el código para SUB en operations, en LEDS se puede ver el resultado de la resta entre 20 - 15 = 5.



Aquí se puede ver en el TestBench automático, quien realiza pruebas con valores aleatorios constantemente y que, dado que no ha fallado, se entiende que está todo funcionando correctamente.

Para mayor verificación, también se pueden ver mensajes impresos con cada prueba:

