

Arquitectura de Computadoras

TP1: ALU

Tomás Martín - 39326227

https://github.com/tomasmartin27/TP1_ALU.git

16 de septiembre del 2021

Consigna

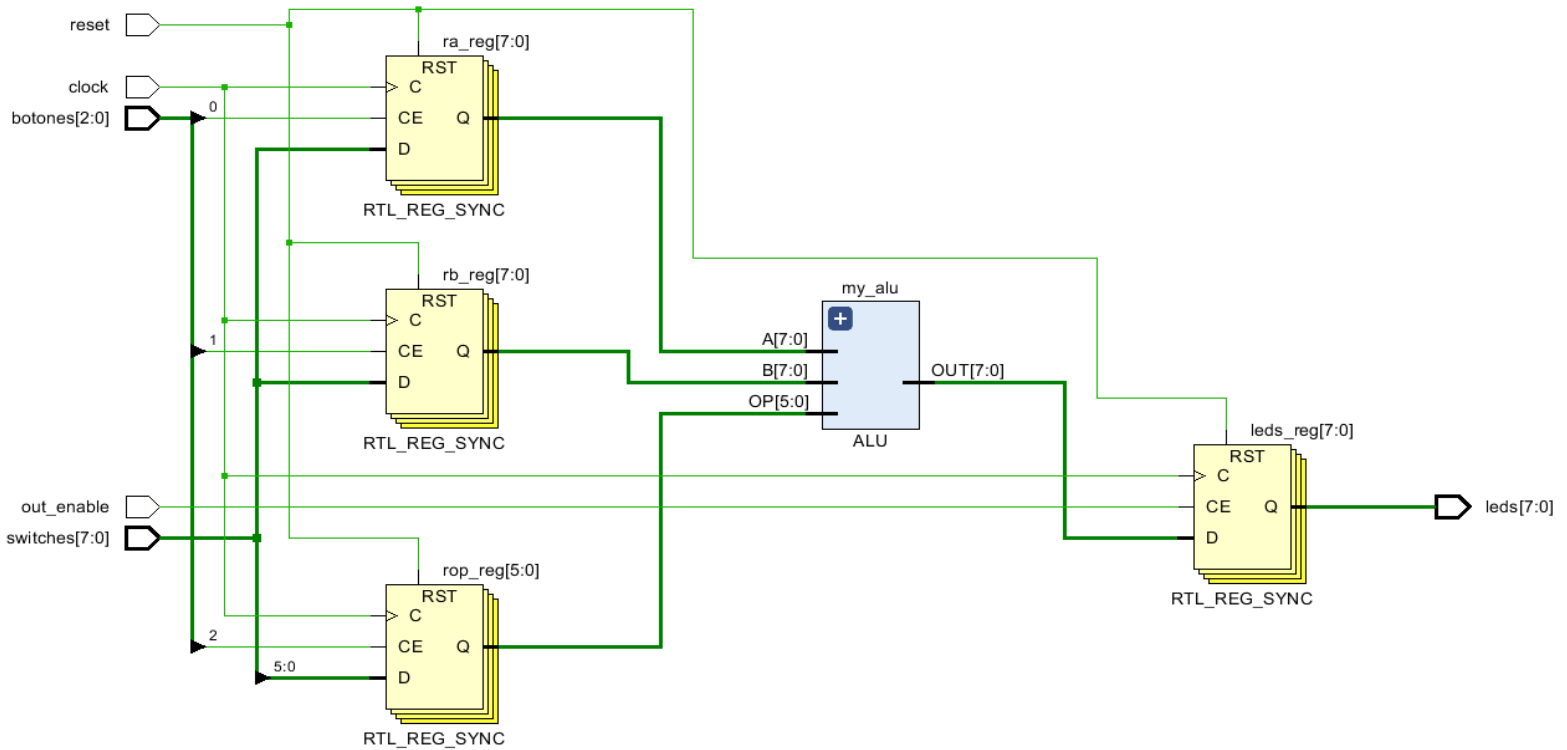
- Implementar en FPGA una ALU.
- La ALU debe ser parametrizable (bus de datos) para poder ser utilizada posteriormente en el trabajo final.
- Validar el desarrollo por medio de Test Bench.
 - El testbench debe incluir generación de entradas aleatorias y código de chequeo automático.
- Simular el diseño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

Operaciones

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

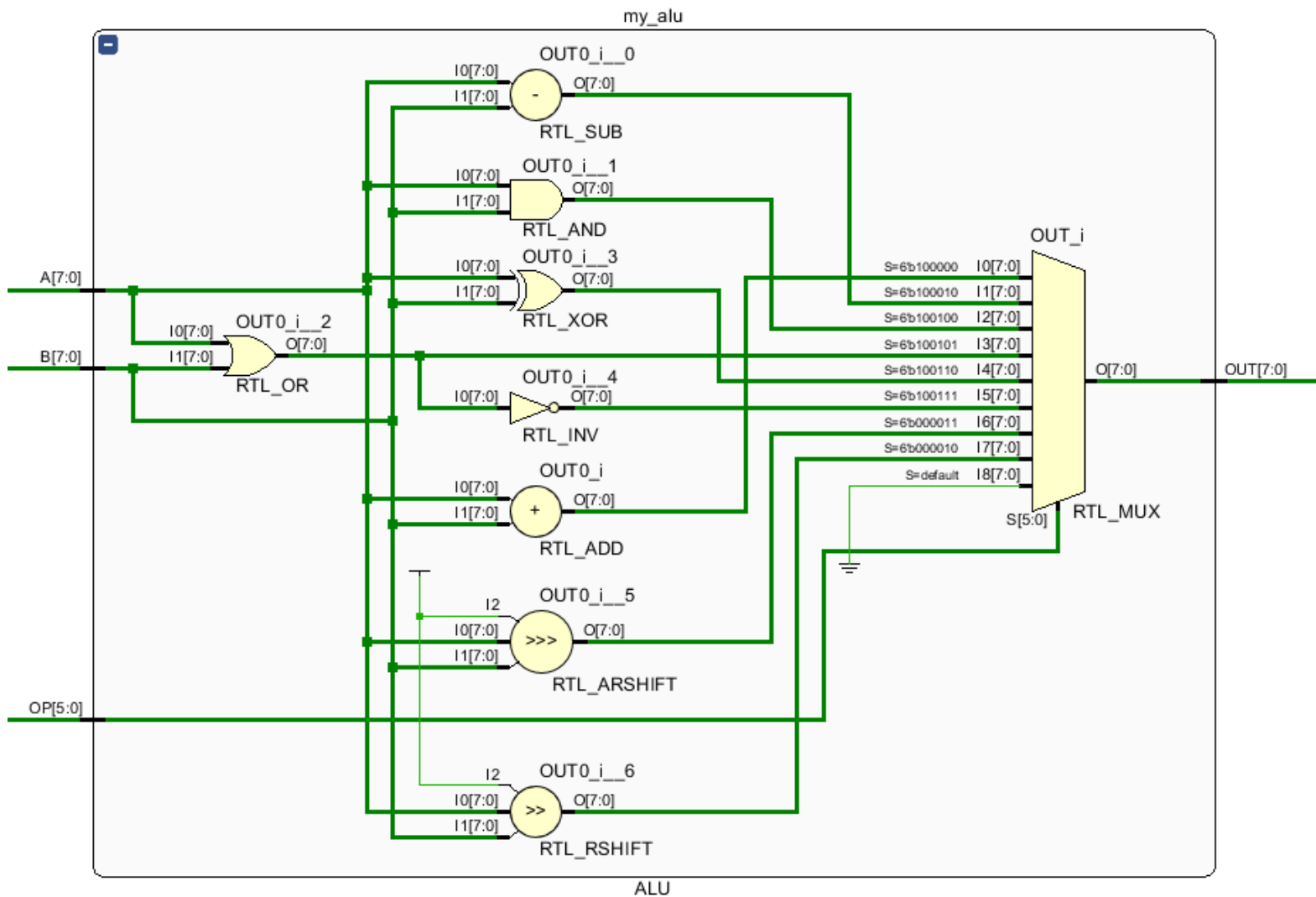
Schematic RTL

Control (Módulo Top)



En el esquemático se pueden observar las entradas y las salidas del módulo top y del módulo ALU. Las entradas del módulo top son: la señal de clock, un reset para los registros de entrada y salida, los switches para ingresar los valores, los botones para controlar la carga de los registros con los switches y un botón llamado `out_enable` que habilita la salida hacia los leds. La salida de este módulo son los leds para visualizar el valor obtenido.

ALU

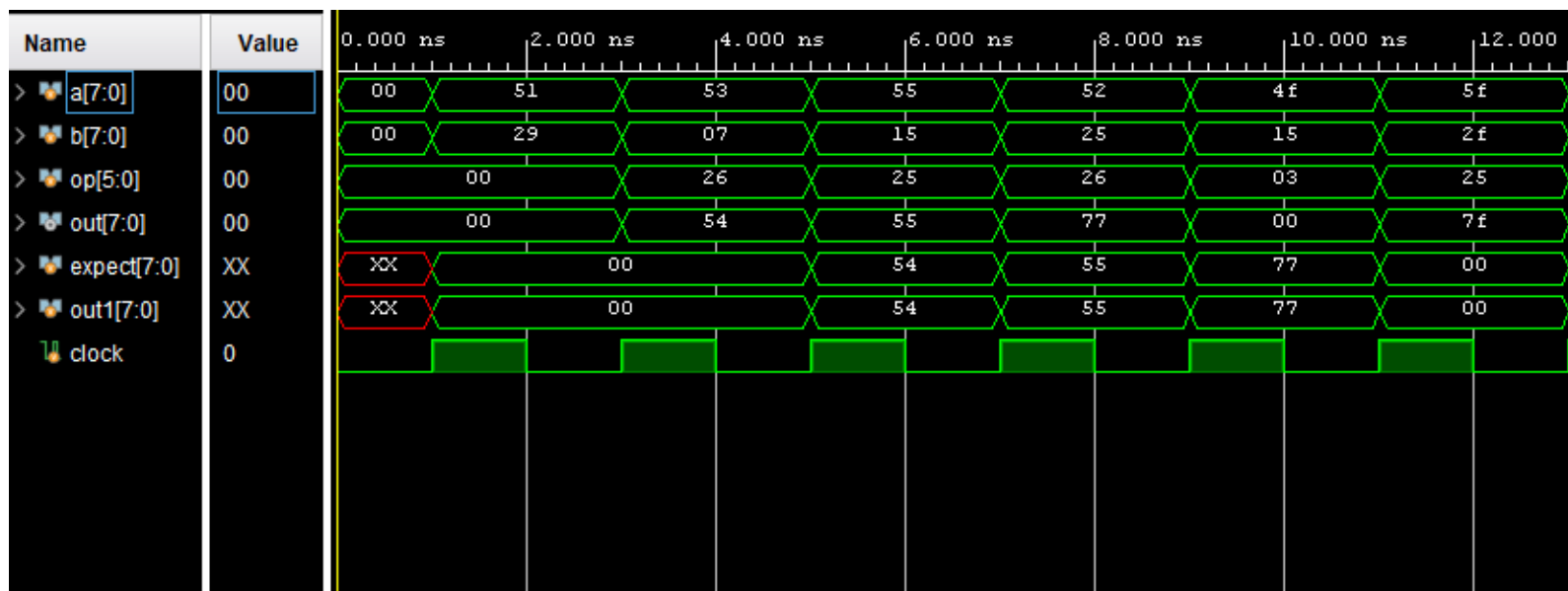


El módulo ALU tiene como entrada el dato A y B y la operación que se quiere realizar. Este módulo es puramente combinacional, a diferencia del top, se puede comprobar que no se utiliza la señal del clock. El módulo top es síncrono ya que se debe sincronizar las señales obtenidas por los botones y los switches ya que estas, al ser señales del mundo real, son asincrónicas por lo cual se las debe sincronizar, pasarlas por un Flip Flop y aceptar su entrada cuando el flanco del clock la autoriza para poder trabajar con ellas.

Behavioral Simulation del Módulo ALU

A través de un test bench se comprobó el funcionamiento del módulo ALU. Las entradas a y b se generan aleatoriamente, al igual que la elección de la operación a realizar.

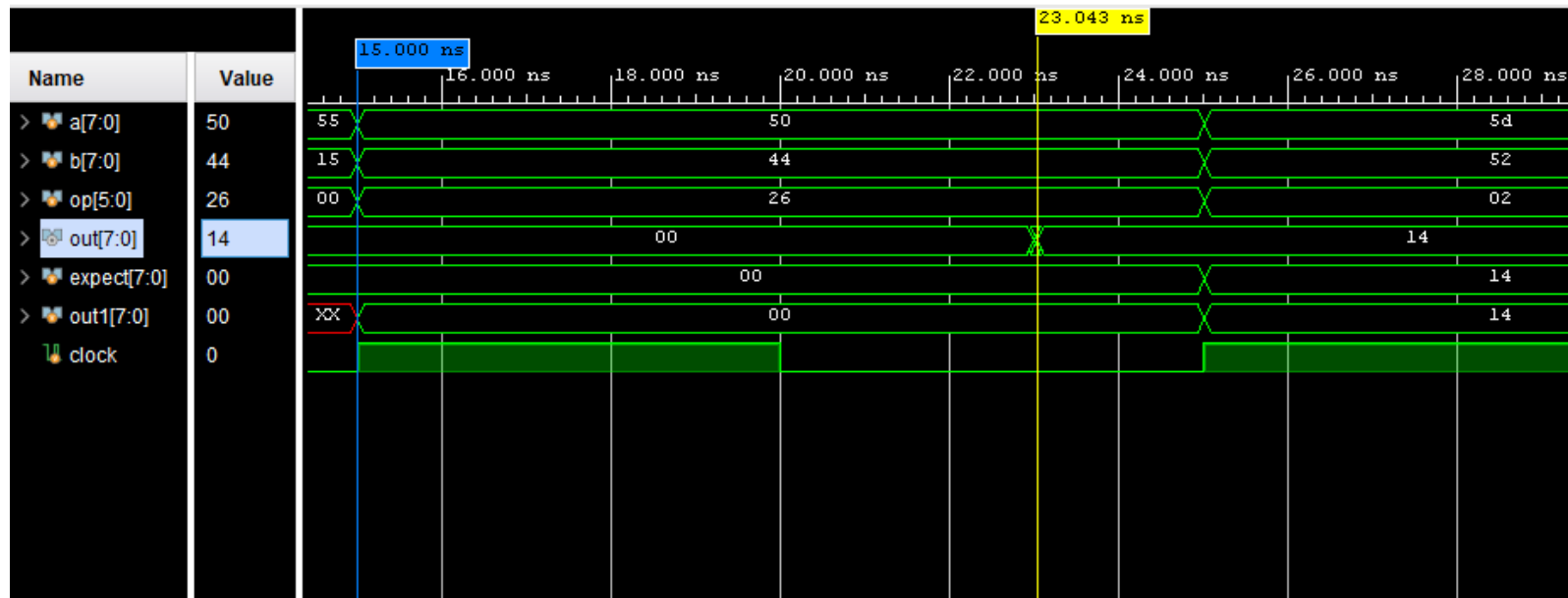
Por medio de un bloque assert se comprueba que el valor obtenido a la salida (out), guardado en el registro out1, sea el valor esperado (expect). Esto se comprueba en el siguiente ciclo de clock.



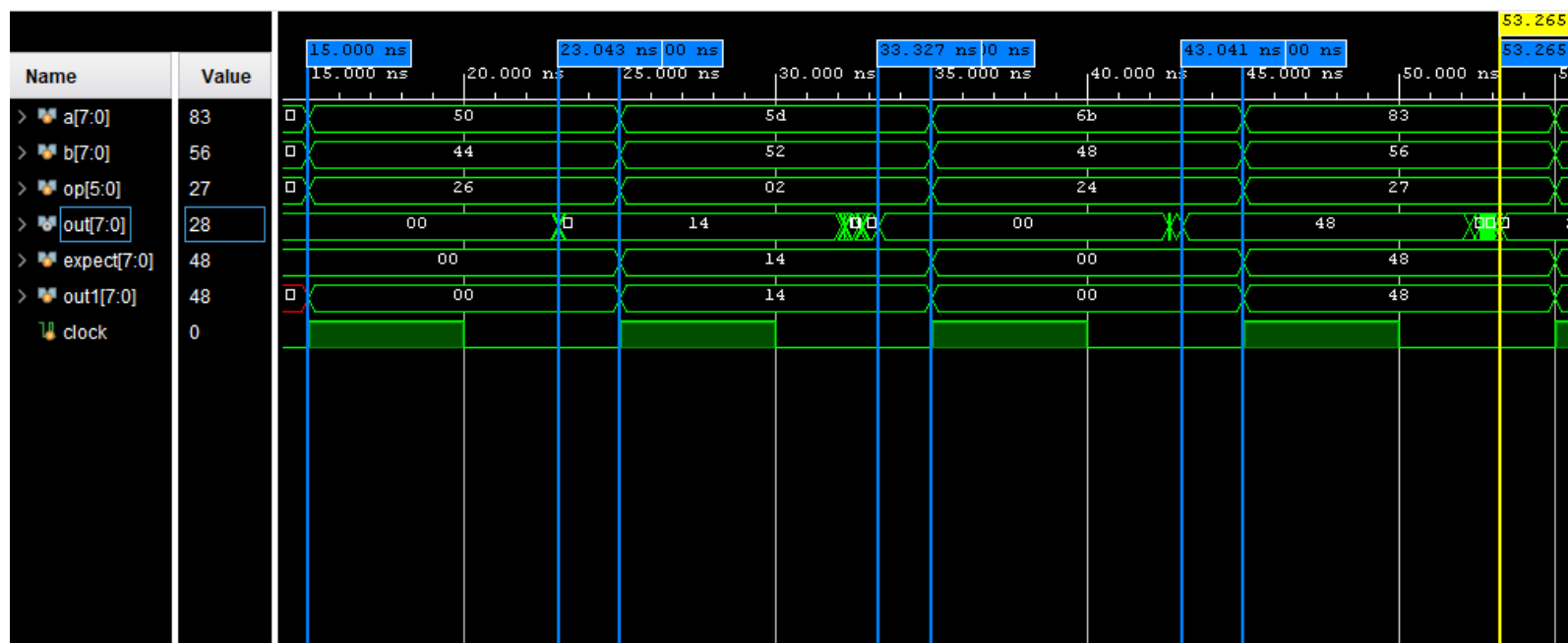
En cada flanco positivo del clock se generan valores de a y b, también se elige una operación (op), obteniendo así un valor de la operación (out). Luego, en el siguiente ciclo, se comprueba si este valor de salida es correcto.

Se puede observar en la captura que una de las operaciones es una XOR (0x26) entre 0x53 (01010011) y 0x07 (00000111). El resultado de esta operación es 0x54 (01010100), el cual es el resultado obtenido y se comprueba que es el valor esperado. Lo mismo ocurre con las demás operaciones.

Post-Synthesis Timing Simulation del Módulo ALU



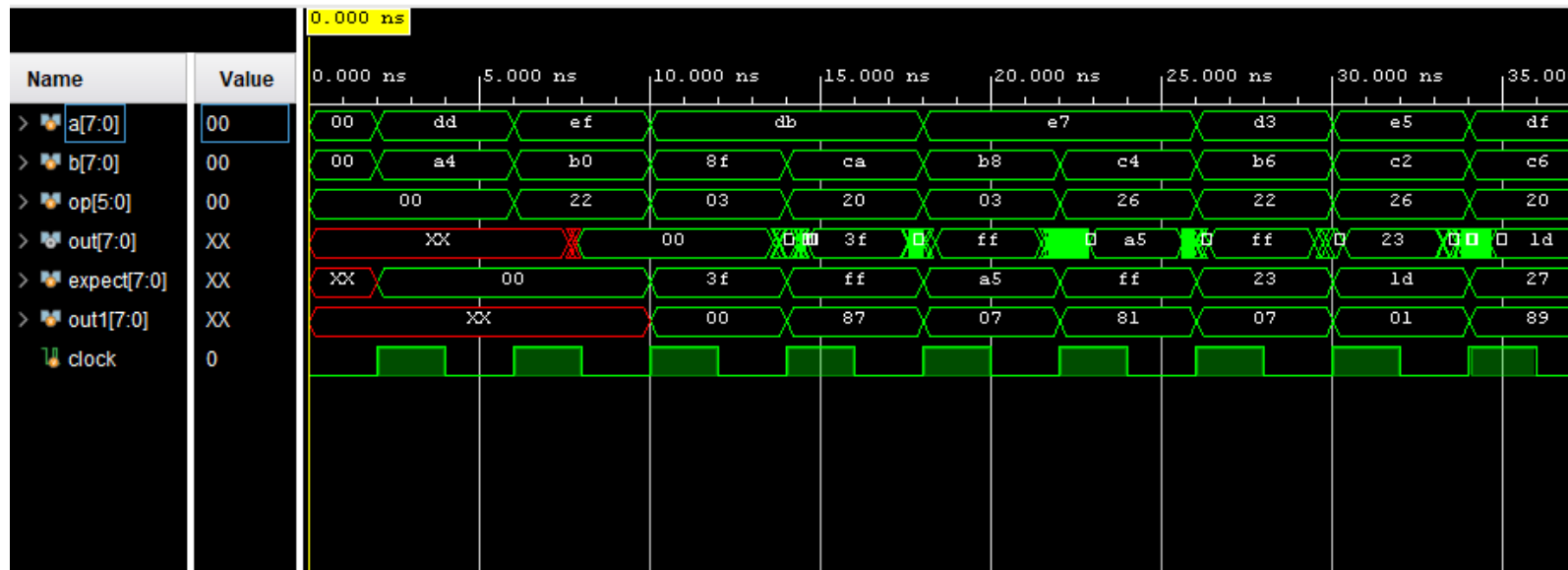
El tiempo que tarda el combinacional en realizar una operación (en este caso una XOR) y obtener un dato estable es igual a 8,043 ns.



El retardo es casi igual en todas las operaciones, entre 8,043 ns y 8,327 ns. En algunas operaciones es menor ya que varían menos que otras para llegar al valor estable porque tienen que cambiar menos bits en la salida. Por ejemplo, en la imagen de arriba, se puede ver que la operación SRL tiene que cambiar muchos bits en la salida, por lo que tarda más en llegar al estable y tiene más variaciones.

El análisis se realizó con una un periodo de clock de 10 ns, es decir, una frecuencia de clock de 100 MHz. Esta frecuencia es la máxima a la que puede funcionar la ALU ya que si ponemos un periodo menor a 8 ns (tiempo que tarda en realizarse el combinacional) esta no funcionará correctamente, como veremos a continuación.

Se configura un clock con un periodo de 4 nanosegundos (250 MHz).



Como el retardo es mayor al periodo del clock, se obtiene el resultado incorrecto o atrasado, es decir, luego de que ya se cambiaron las entradas para la siguiente operación. El valor esperado nunca es igual al valor de salida que debería tener en ese momento (out1).

Tcl Console



```

a = 11011011, b = 10001111, op = 000011, out = 10000101      14
a = 11011011, b = 10001111, op = 000011, out = 10000111      14
Error en operacion 000011 en tiempo          14 ns
a = 11011011, b = 11001010, op = 100000, out = 10000111      14
a = 11011011, b = 11001010, op = 100000, out = 10010111      14
a = 11011011, b = 11001010, op = 100000, out = 10000111      14
a = 11011011, b = 11001010, op = 100000, out = 10010111      14
a = 11011011, b = 11001010, op = 100000, out = 10110101      14
a = 11011011, b = 11001010, op = 100000, out = 10010111      14
a = 11011011, b = 11001010, op = 100000, out = 10110111      14
a = 11011011, b = 11001010, op = 100000, out = 10110011      14
a = 11011011, b = 11001010, op = 100000, out = 10110111      14
a = 11011011, b = 11001010, op = 100000, out = 11110111      14
a = 11011011, b = 11001010, op = 100000, out = 10110111      14
a = 11011011, b = 11001010, op = 100000, out = 00110111      14
a = 11011011, b = 11001010, op = 100000, out = 00111111      15
a = 11011011, b = 11001010, op = 100000, out = 00011111      18
a = 11011011, b = 11001010, op = 100000, out = 00011110      18
a = 11011011, b = 11001010, op = 100000, out = 01011110      18
a = 11011011, b = 11001010, op = 100000, out = 01011010      18
a = 11011011, b = 11001010, op = 100000, out = 00000000      18
a = 11011011, b = 11001010, op = 100000, out = 00000001      18
a = 11011011, b = 11001010, op = 100000, out = 00000111      18
Error en operacion 100000 en tiempo          18 ns
a = 11100111, b = 10111000, op = 000011, out = 00000111      18
a = 11100111, b = 10111000, op = 000011, out = 01011111      18
a = 11100111, b = 10111000, op = 000011, out = 11011111      18
a = 11100111, b = 10111000, op = 000011, out = 11111111      18
a = 11100111, b = 10111000, op = 000011, out = 10011111      21
a = 11100111, b = 10111000, op = 000011, out = 10011110      22
a = 11100111, b = 10111000, op = 000011, out = 10011111      22
a = 11100111, b = 10111000, op = 000011, out = 10011101      22
a = 11100111, b = 10111000, op = 000011, out = 10000001      22
Error en operacion 000011 en tiempo          22 ns
a = 11100111, b = 11000100, op = 100110, out = 10000001      22

```

Por consola se puede visualizar todas las variaciones de la salida y los instantes en los que se evaluó la salida con el valor esperado resultando en un error, ya que la salida no era correcta.

```

a = 00010101, b = 11100101, op = 000010, out = 11010000      791
a = 00010101, b = 11100101, op = 000010, out = 11010100      791
Error en operacion 000010 en tiempo          795 ns
a = 00010001, b = 11100001, op = 100101, out = 11010100      795
a = 00010001, b = 11100001, op = 100101, out = 11010110      797
a = 00010001, b = 11100001, op = 100101, out = 01010110      797
a = 00010001, b = 11100001, op = 100101, out = 00010110      797
a = 00010001, b = 11100001, op = 100101, out = 00010010      797
a = 00010001, b = 11100001, op = 100101, out = 00001010      797
a = 00010001, b = 11100001, op = 100101, out = 00000000      797
Error en operacion 100101 en tiempo          801 ns
Error

```

En estos casos se imprime un mensaje de “Error” al final porque el test no pasó.

De lo contrario, si no ocurre ningún error, se imprime un mensaje de “Test Pass”.

```

a = 11111010, b = 10111100, op = 100101, out = 11111110      765
a = 11100100, b = 11010011, op = 000011, out = 11111111      775
a = 00001001, b = 11100110, op = 100101, out = 11101111      785
a = 00010001, b = 11100001, op = 100110, out = 11110000      795
Test Pass

```

La señal tiene que transitar por una secuencia de compuertas por lo tanto va demorar un determinado tiempo (tiempo del combinacional). El tiempo que hay que tener en cuenta es el del camino crítico o del camino más largo, en este caso mayor a 8 ns. Para un funcionamiento correcto, el periodo del clock debe ser mayor que el tiempo del combinacional.

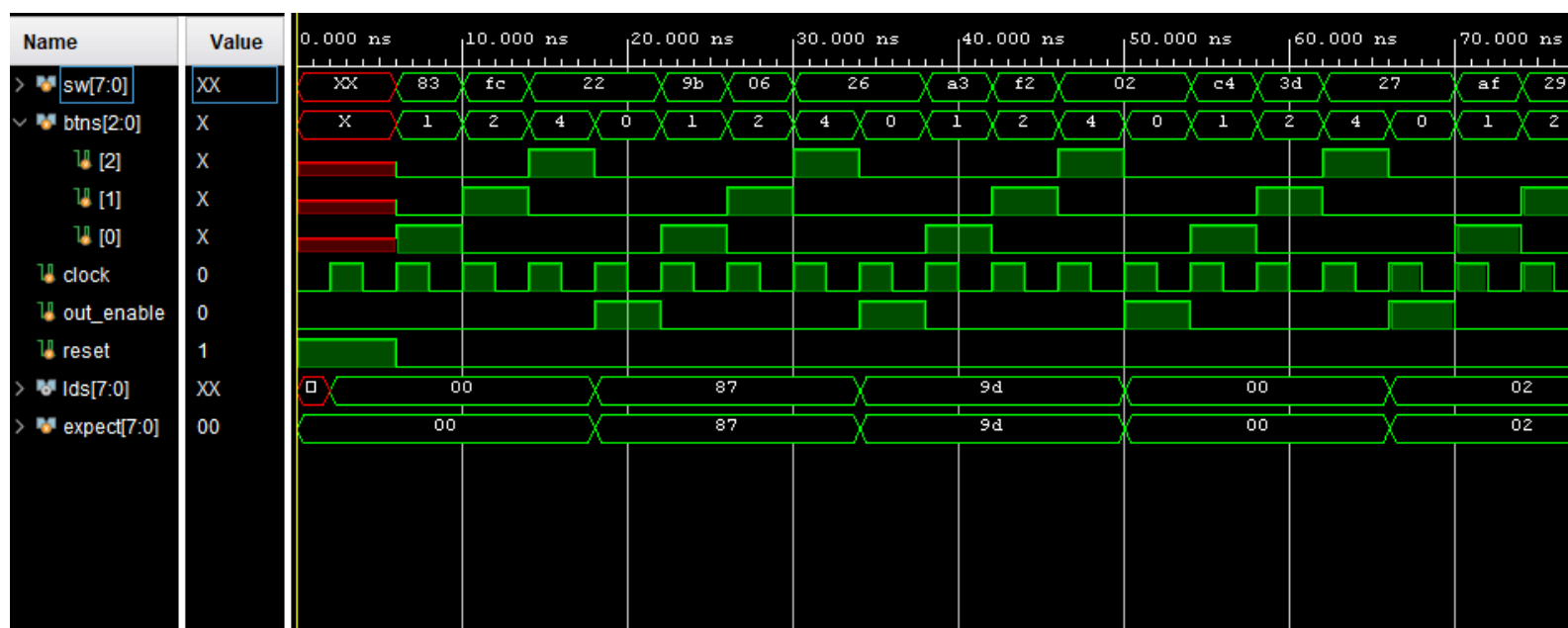
Cuando se hace overclocking, llega un momento en el que deja de funcionar el sistema porque no soporta los tiempos de setup y hold de los FF o se llegó al tiempo de un camino crítico, como sucede en este caso.

Si el clock llega demasiado antes y el dato no terminó el camino combinacional, el valor que se va a estar muestreando va a ser un valor viejo del dato.

Behavioral Simulation del Módulo Top

Mediante un testbench se generan estímulos para testear el módulo top, se ingresan valores aleatorios al switch para cargar los registros que se pasan al módulo ALU para realizar las respectivas operaciones. Luego, se utiliza el botón out_enable para obtener la salida.

Con un bloque assert se comprueba que el valor obtenido a la salida (lds) sea el esperado (expect).



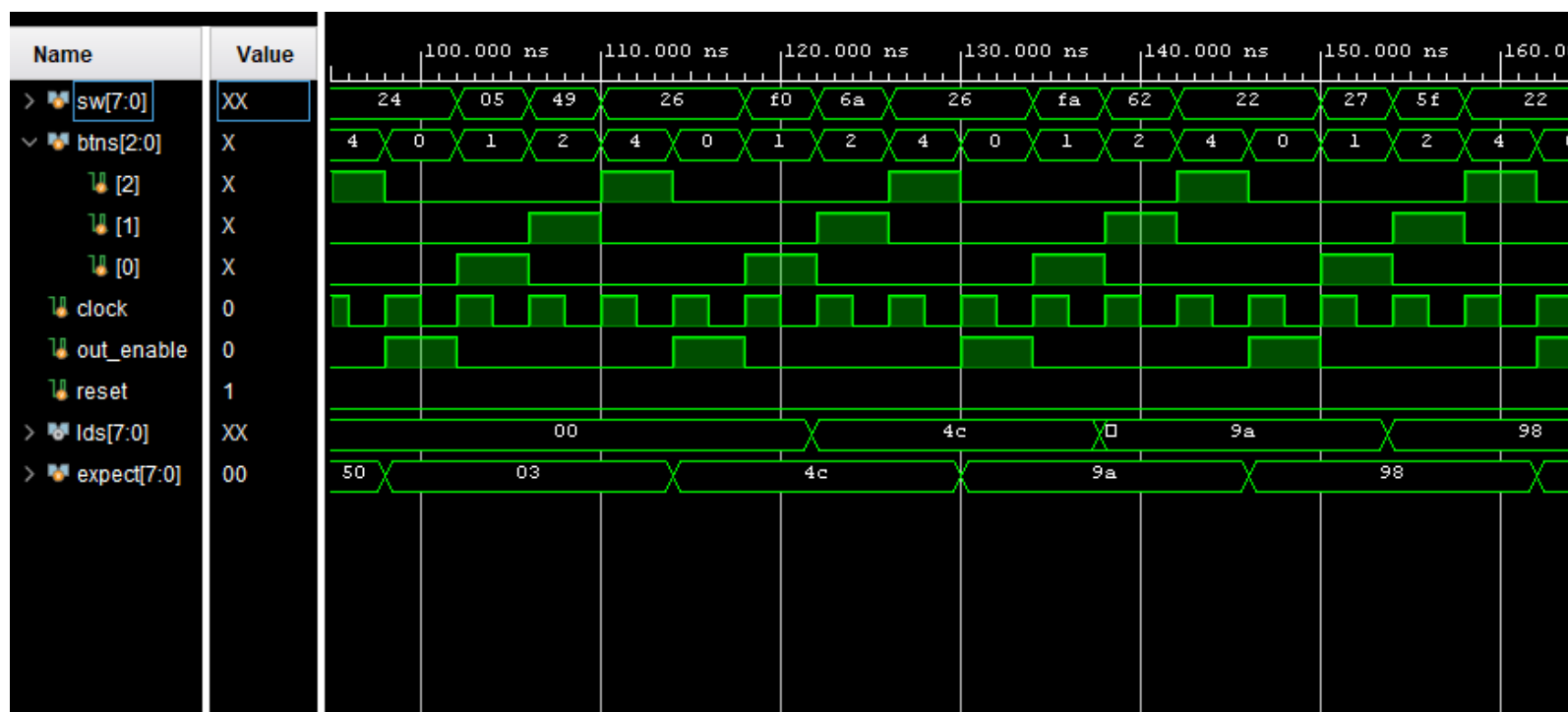
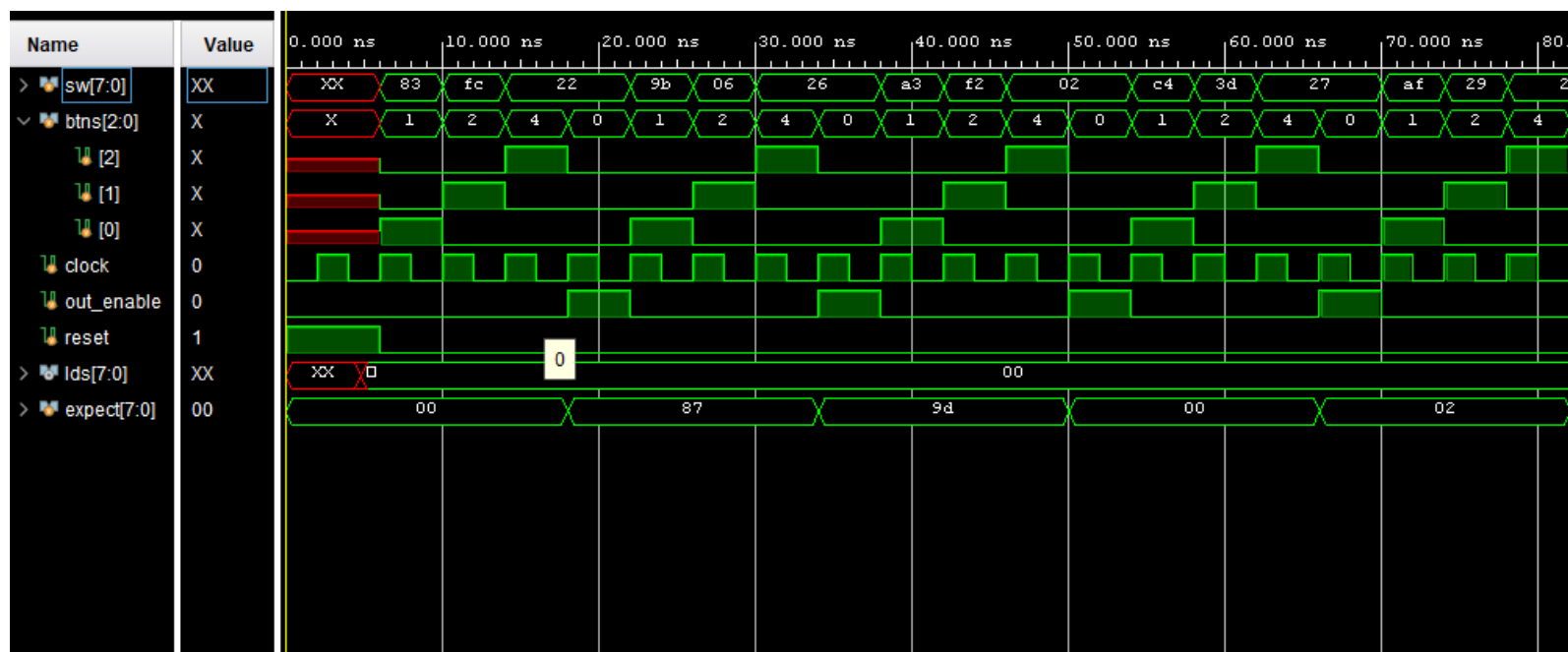
Lo primero que se realiza es un reset de los registros, se puede observar que la salida lds se pone en cero.

En cada flanco positivo del clock se cargan, en los registros, el valor de los switches, donde cada botón carga el registro correspondiente. El primer registro que se carga es el valor del dato A con un valor generado aleatoriamente igual a 0x83 (10000011). En el siguiente flanco se carga el dato B con 0xFC (11111100). Luego, en el siguiente flanco se carga el valor de la operación, elegida aleatoriamente de las operaciones disponibles, con 0x22 correspondiente a una SUB. Para posteriormente, en el siguiente flanco, habilitar la salida y obtener el valor de la SUB que es igual a 0xFF87 (111111110000111). Al tener una salida de 8 bits, si el resultado de una operación es mayor a este valor, se obtienen solamente los 8 bits menos significativos del resultado. Se puede comprobar que es el mismo valor que el esperado. Lo mismo sucede con las demás operaciones.

Post-Synthesis Timing Simulation del Módulo Top

Para realizar un análisis del timing post-síntesis, se varió el clock y se fue viendo si las operaciones funcionaban con ese valor de clock.

- Clock = periodo de 4 nanosegundos (250 MHz).

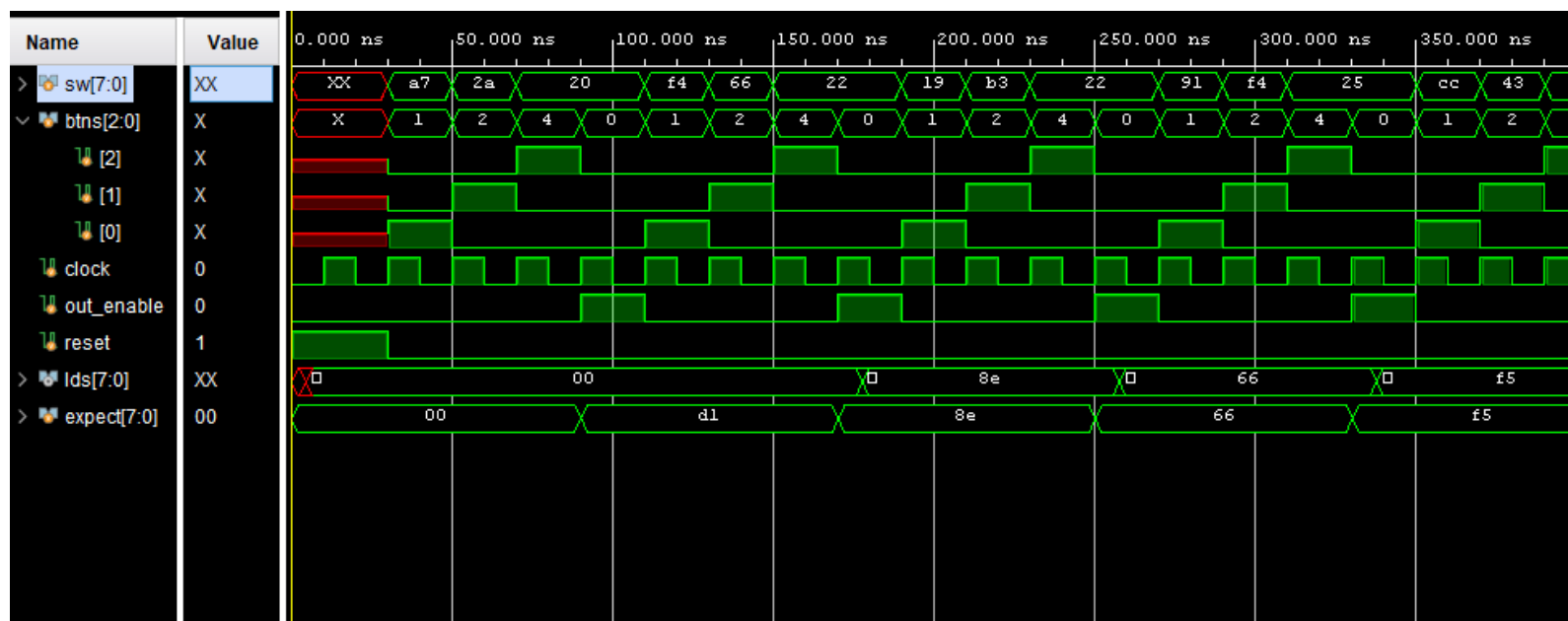


El registro de salida no contiene ningún valor, hasta que muchos ciclos después aparece el resultado, pero está disponible varios ciclos después de que se habilita la salida.

Cada vez que llega un flanco del clock, los flips flops del registro de salida van a mantener en D la salida de la ALU. Si el flanco llega muy rápido (frecuencia de clock alta) lo que puede pasar es que el dato no esté estable lo suficientemente antes del flanco (setup time) a la entrada de los flips flops o que el dato no haya llegado por un tiempo combinacional muy largo.

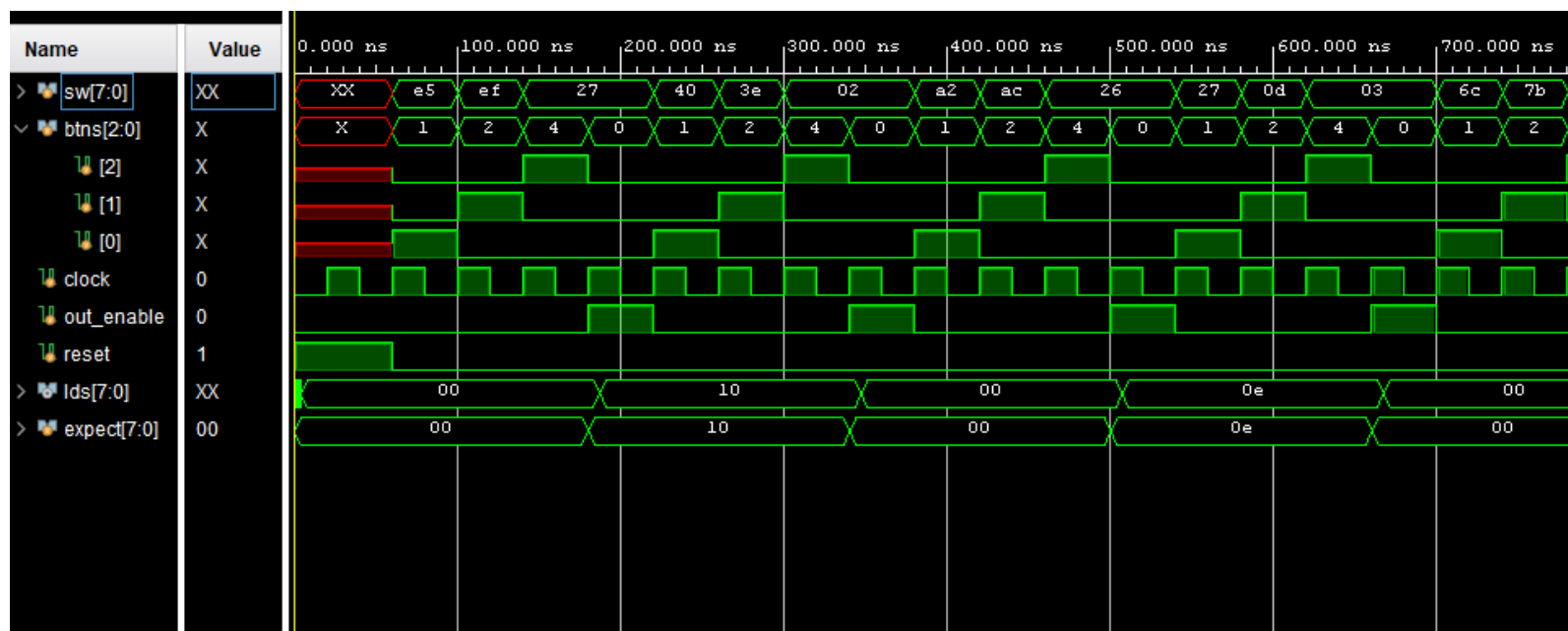
Se sabe que el tiempo del combinacional es de 8 nanosegundos por lo tanto no se está cumpliendo que el periodo de clock debe ser mayor por eso el mal funcionamiento.

- Clock = periodo de 20 nanosegundos (50MHz).



La primera operación no es correcta y las demás varían hasta llegar al valor estable.

- Clock = periodo de 40 nanosegundos (25MHz).



Se observa un funcionamiento correcto. También se puede apreciar que el retraso desde que se habilita la salida hasta que se obtiene el valor, es menor que en el resto.

Al igual que en el testbench para el módulo ALU si ocurrió un error se imprime “Error” al final y si no se imprime “Test Pass”.

```

switch = xxxxxxxx, botones = xxx, leds = 0xxxxxx0      5
switch = xxxxxxxx, botones = xxx, leds = 00000000      5
switch = 10000011, botones = 001, leds = 00000000      6
switch = 11111100, botones = 010, leds = 00000000     10
switch = 00100010, botones = 100, leds = 00000000     14
switch = 00100010, botones = 000, leds = 00000000     18
switch = 10011011, botones = 001, leds = 00000000     22
switch = 00000110, botones = 010, leds = 00000000     26
switch = 00100110, botones = 100, leds = 00000000     30
Error en operacion 00100110 en tiempo                  34 ns
switch = 00100110, botones = 000, leds = 00000000     34
switch = 10100011, botones = 001, leds = 00000000     38
switch = 11110010, botones = 010, leds = 00000000     42
switch = 00000010, botones = 100, leds = 00000000     46
Error en operacion 00000010 en tiempo                  50 ns
switch = 00000010, botones = 000, leds = 00000000     50

```

```
switch = 01001010, botones = 001, leds = 00000100      794
switch = 10110001, botones = 010, leds = 00000100      794
switch = 00000011, botones = 100, leds = 00000100      798
Error

switch = 01001010, botones = 001, leds = 00000100      790
switch = 10110001, botones = 010, leds = 00000100      794
switch = 00000011, botones = 100, leds = 00000100      798
Test Pass
```