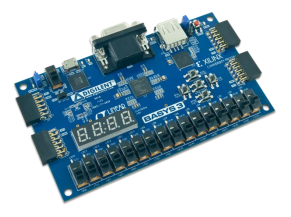
PRACTICA LIBRE: VOLTIMETRO DIGITAL IMPLEMENTADO CON UNA FPGA

APARTADO 1



Centro:

Universidad de Alcalá

Grado:

Grado en Ingeniería de Computadores

Asignatura:

Modelado de Sistemas Computacionales

Curso:

Curso Académico 2020/2020

Curso 3º - Cuatrimestre 2º

Grupo:

A3

Alumnos:

Sergio Sanz Cacho – 03201575K

Carlos Tejeda Martínez – 03148129G

INDICE

[DISEÑO DEL MÓDULO CNT\_DISPLAY 2](#_Toc35799928)

[PLANTEAMIENTO 2](#_Toc35799929)

[DISEÑO 2](#_Toc35799930)

[SIMULACIÓN FUNCIONAL 6](#_Toc35799931)

[SIMULACIÓN TEMPORAL 9](#_Toc35799932)

[CONCLUSIÓN 10](#_Toc35799933)

# DISEÑO DEL MÓDULO CNT\_DISPLAY

## PLANTEAMIENTO

De acuerdo con el desarrollo de la práctica, hemos tenido que desarrollar un controlador para el display del voltímetro que consiste en cuatro displays y un punto para la representación del valor medido que puede comprender desde 0 hasta 9999.

Este controlador se basa en la adquisición del valor a representar en código binario y mostrarlo en código decimal mediante los displays, tras el tratamiento de la señal adquirida. Para ello el módulo tendrá como entrada las señales de reloj (CLK), reset (RST), valor en binario (BCD), indicador de nuevo valor (BCD\_OK); y como salidas el valor de la cifra a representar en el display (SEG\_AG), el display al que va dirigido el valor (AND\_30) y la señal de encendido del punto del display (DP).

## DISEÑO

En primer lugar, a partir del diagrama expuesto en clase, estudiamos como ha de conformarse el interior del controlador a través de los componentes mostrados en el esquema, los cuales su función será implementada como procesos. Estos componentes son:

* Registro de 16bits:

Este componente recibe por entrada el valor a representar en formato binario y cuando la señal de validación del nuevo valor (BCD\_OK) llega, se divide el valor de entrada de BCD en las cuatro cifras para cada display que se darán por las salidas BCD\_U, BCD\_D, BCD\_C y BCD\_M.

En caso de que llegue la señal activa de reset (RST) se procede a poner las cuatro salidas de las cifras a cero.

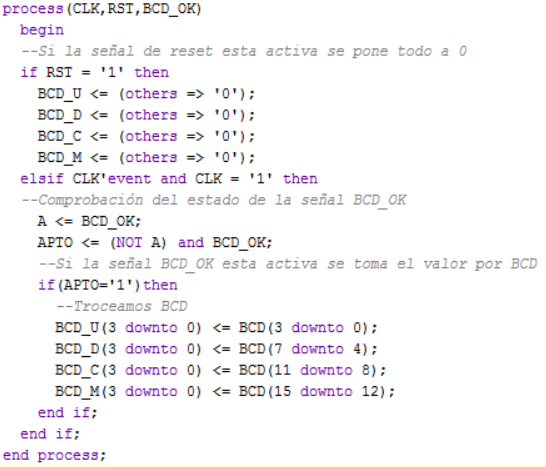


Ilustración 1-Implementación en VHDL del registro de 16bits

* Multiplexor:

Este componente es regulado por la señal S proveniente del contador de 2bits la cual cada 1 milisegundo ira variando, dando por salida del multiplexor la diferente configuración a mostrar en cada uno de los displays dependiendo del tipo de cifra a mostrar que es. Las salidas que tiene consisten en la señal para tener activo o desactivado el punto del display (DP), la señal que indica que display se esta configurando (AND\_30) y finalmente la señal en binario de la cifra a mostrar que pasará previamente a su visionado en el display por un conversor del número a binario a BCD para activar los segmentos apropiados del display (S\_multiplexor).

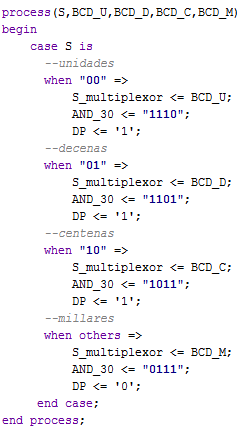


Ilustración 2-Implementación en VHDL del multiplexor

Tabla de combinaciones del multiplexor:

|  |  |  |  |
| --- | --- | --- | --- |
| S | S\_multiplexor | AND\_30 | DP |
| 00 | BCD\_U | 1110 | 1 |
| 01 | BCD\_D | 1101 | 1 |
| 10 | BCD\_C | 1011 | 1 |
| 11 | BCD\_M | 0111 | 0 |

* Preescaler:

Este componente produce la señal CE\_preescaler a partir de la señal de reloj (CLK), pasando de recibir pulsos cada 10 nanosegundos a producir una señal de salida con pulsos cada 1 milisegundo para activar el funcionamiento del contador de 2bits.

En caso de que llegue la señal activa de reset (RST) se procede a reiniciar la emisión de pulsos de 1ms.

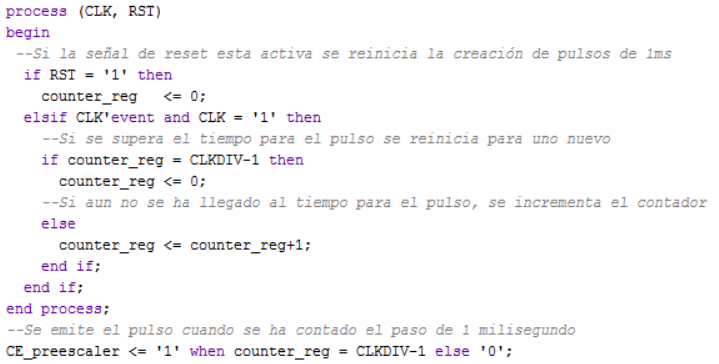


Ilustración 3-Implementación en VHDL del preescaler

* Contador de 2bits:

Este componente es regulado por la señal CE\_preescaler la cuál porta un pulso cada 1 milisegundo que activará al contador para incrementarse en uno su valor de salida S. Cuando llega a 3 (“11”) vuelve a 0 (“00”) por tanto podemos indicar las salida de las 4 cifras en el multiplexor en bucle, refrescando los datos mostrados cada 4 milisegundos.

En caso de que llegue la señal activa de reset (RST) se procede a iniciar desde 0 el contador.

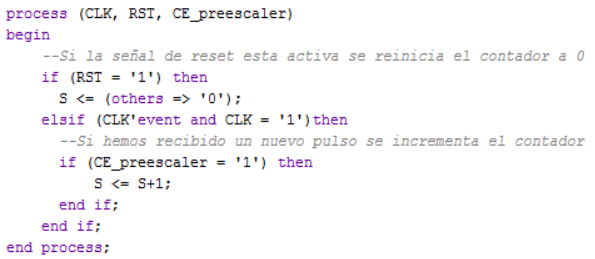


Ilustración 4-Implementación en VHDL del contador de 2bits

* Conversor BCD-7segmentos:

Este componente recibe la cifra que se ha de mostrar en formato binario (S\_multiplexor) y en función del número le asigna su código BCD de acuerdo con los segmentos que han de encenderse para la representación de la cifra, que se trasmitirá a través de la salida SEG\_AG al display correspondiente.

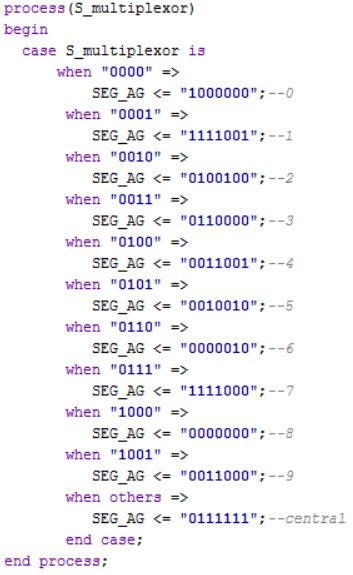
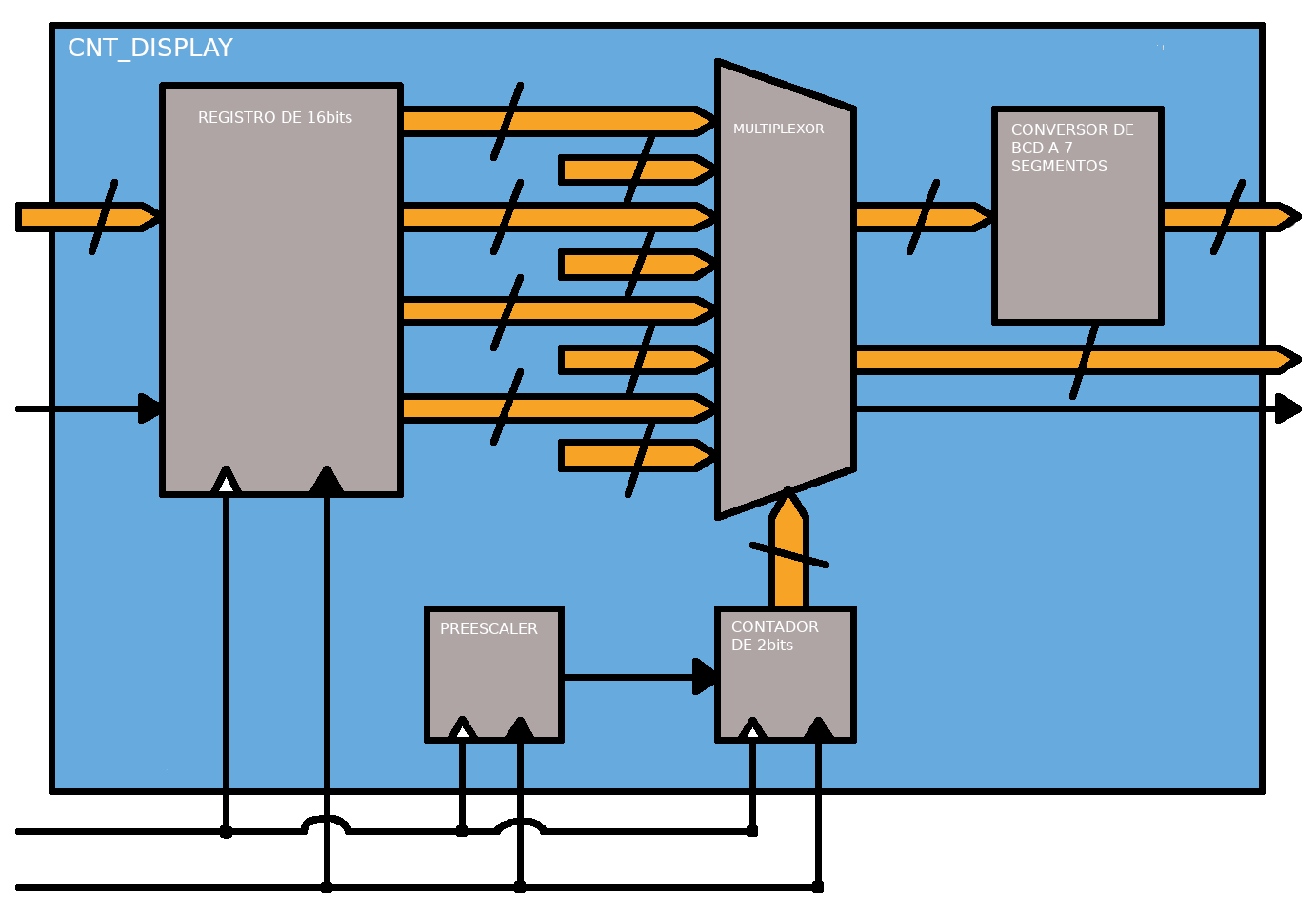


Ilustración 5-Implementación en VHDL del conversor de BCD a 7 segmentos

Tabla de combinaciones del conversor:

|  |  |
| --- | --- |
| S\_multiplexor | SEG\_AG |
| 0000 | 1000000 |
| 0001 | 1111001 |
| 0010 | 0100100 |
| 0011 | 0110000 |
| 0100 | 0110001 |
| 0101 | 0010010 |
| 0110 | 0000010 |
| 0111 | 1111000 |
| 1000 | 0000000 |
| 1001 | 0011000 |
| XXXX | 0111111 |

Combinando estos componentes el diagrama completo de la estructura del cnt\_display queda asi:

Las señales indicadas en la figura superior son detalladas a continuación:

18

17

16

15

14

13

12

10

11

9

4

3

8

7

6

5

2

1

Ilustración 6-Esquema completo de la estructura de la entidad cnt\_display con las señales utilizadas

|  |  |  |  |
| --- | --- | --- | --- |
| Número | Señal | Número | Señal |
| 1 | BCD (16bits) | 10 | Constante para AND\_30 relativa a BCD\_C (4bits) |
| 2 | BCD\_OK | 11 | Constante para AND\_30 relativa a BCD\_D (4bits) |
| 3 | CLK | 12 | Constante para AND\_30 relativa a BCD\_U (4bits) |
| 4 | RST | 13 | CE\_preescaler |
| 5 | BCD\_M (4bits) | 14 | S (2bits) |
| 6 | BCD\_C (4bits) | 15 | S\_multiplexor (4bits) |
| 7 | BCD\_D (4bits) | 16 | AND\_30 (4bits) |
| 8 | BCD\_U (4bits) | 17 | DP |
| 9 | Constante para AND\_30 relativa a BCD\_M (4bits) | 18 | SEG\_AG (7bits) |

Resumiendo, el desarrollo completo se basa en la llegada valores en codificación BCD por la señal BCD, con los cuales se empieza a trabajar en el registro de 16 bits una vez que llega un pulso a nivel alto de la señal BCD\_OK.

Cuando se valida el valor obtenido por BCD, se procede a dividir la señal recogida en los 4 segmentos relativos a cada cifra del valor en las señales de menor a mayor peso de BCD\_U, BCD\_D, BCD\_C y BCD\_M, llegando esta señal al multiplexor.

## SIMULACIÓN FUNCIONAL

En primer lugar, tenemos un proceso en el cuál se atribuye una secuencia de estimulaciones que consisten en la introducción de un valor al cnt\_display para posteriormente tras 6 milisegundos se pasa un nuevo dato. En esta secuencia se ha simulado el pulso del SW\_OK entre dos flancos de bajada para tener el pulso de 10 nanosegundos.

La metodología seguida es en cargar un nuevo valor en SW\_NUMERO, ocasionar el pulso SW\_OK como hemos detallado anteriormente y esperar un tiempo para repetir esta secuencia de instrucciones con otro valor.



Ilustración 6-Implementación en testbench en VHDL de una secuencia de actuación del cnt\_display

El siguiente conjunto de procesos su función comprende en convertir el valor mostrado por los displays al valor decimal para su comprobación del resultado en la señal D\_DISPLAY.

El siguiente proceso examina el dígito mostrado por el display para traducirlo a BCD

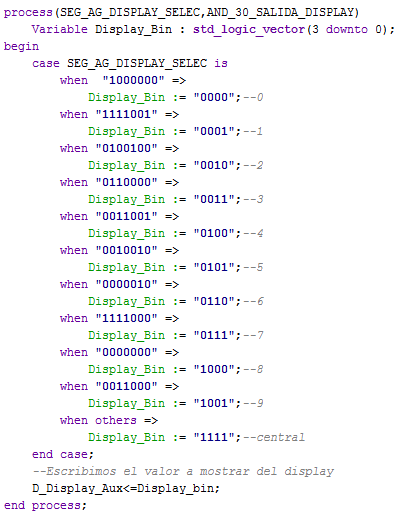


Ilustración 7 Conversor de 7 segmentos a BIN

Este proceso a partir de la obtención del valor en BCD y dependiendo del display del que se ha obtenido la información, recabamos los valores de cada posición.

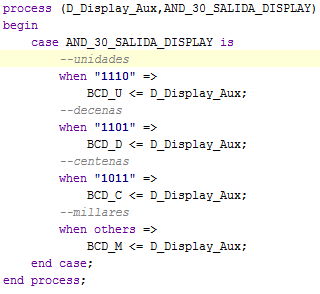


Ilustración 8

El último proceso coloca el conjunto de los valores de los displays, tratado por los anteriores procesos, acorde a sus pesos por el display que ocupan y obtenemos el valor completo mostrado en los displays

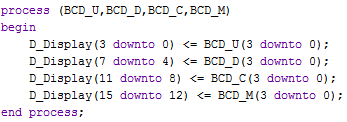


Ilustración 9

Con el proyecto sintetizado lanzamos la simulación funcional de duración total de 12 milisegundos, resumiéndose la ejecución completa en la siguiente imagen:



Ilustración 10-Diagrama de la ejecución completa del testbench del ap1

Las señales observadas en el diagrama son las siguientes:

* RELOJ
* RESET
* SW\_NUMERO
* SW\_OK\_NUMERO
* BCD\_OK
* AND\_30
* DP\_PUNTO
* SEG\_AG\_DISPLAY\_SELEC
* D\_Display\_Aux
* BCD\_U
* BCD\_D
* BCD\_C
* BCD\_M
* D\_DISPLAY

Ën este se muestran la estimulación de la entrada del cnt\_display con dos valores distintos, “1999” primero y después “0000”. Además, se estimula la entrada BCD\_OK en cada caso que se añade un nuevo valor para validarlo (10ns de pulso). La recepción y ejecución del cnt\_display acorde a la llevada de señal activa de BCD\_OK se comprueba en las otras imágenes.

Debido al funcionamiento de los procesos en el cuál las señales se actualizan al final del mismo proceso, podemos observar que la actualización del valor BCD se produce en el flanco de subida posterior al que detecta el BCD\_OK.

Simulacion del ap1

Igual que antes. Diferencia única es que se estimula la señal SW\_OK que simula la pulsación de un botón de la placa, y esta señal es la que genera el pulso de 10ns de BCD\_OK. De igual manera se puede observar que la actualización del valor BCD se produce en el flanco de subida posterior al que detecta el BCD\_OK

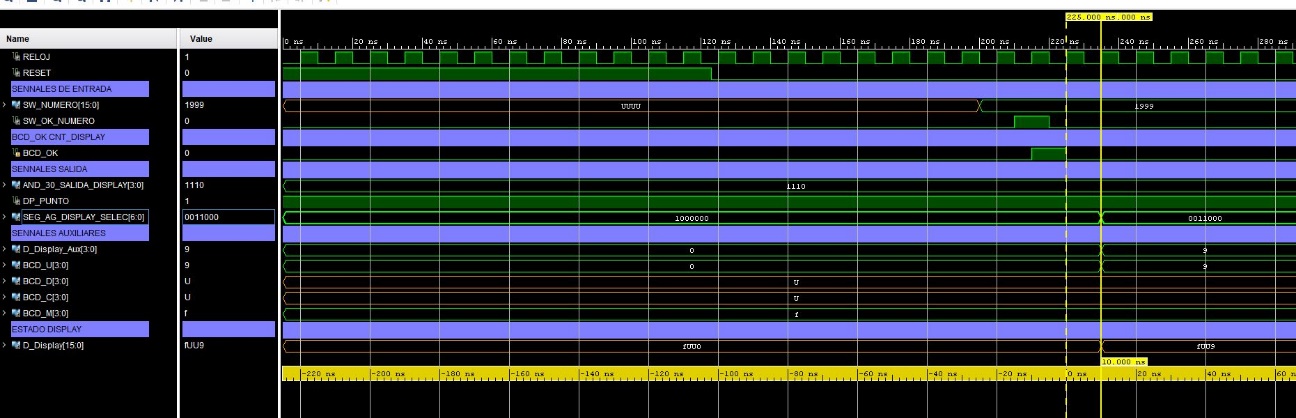


Ilustración 11

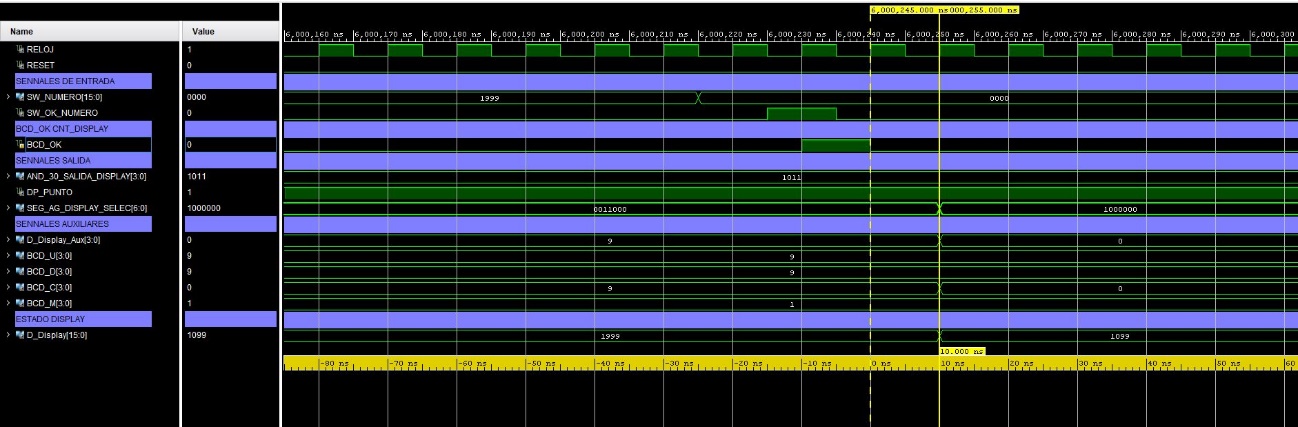


Ilustración 12

## SIMULACIÓN TEMPORAL

Para esta simulación se emplea el mismo banco de pruebas que para la simulación funcional lo que implica que el código es el mismo que el explicado en el apartado anterior.

La simulación temporal posee una duración total de 12 ns.

Durante los cuales podemos observar el efecto de los estímulos aplicados( Señales de entrada 1999 y 0000, y el indicador de nuevo valor SW\_OK) a las distintas entradas, en las diferentes salidas del sistema.

En la siguiente ilustración podemos observar todas las señales del sistema las cuales están divididas en diferentes grupos y son:

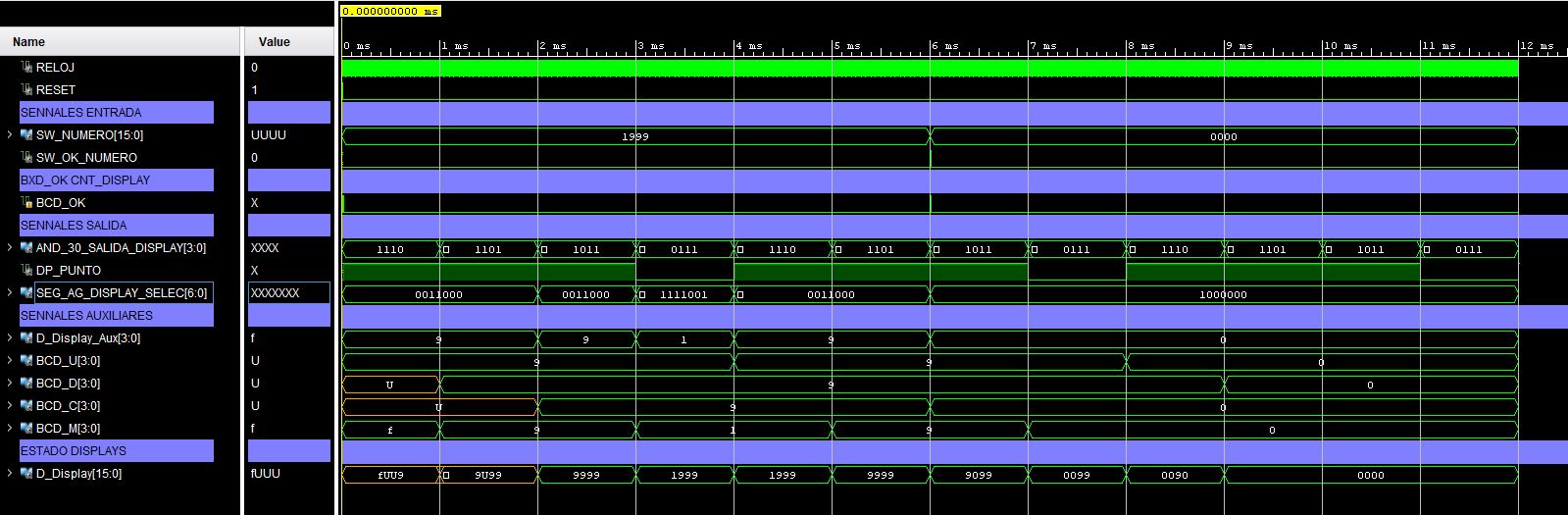


Ilustración simulación temporal completa

* Las dos primeras señales son:
  + RELOJ : Señal de reloj que sincroniza todos los eventos del sistema
  + RESET : Señal asíncrona encargada de reiniciar el sistema.
* Las señales de entrada
  + SW\_NUMERO : Valor de entrada en este caso (1999 y 0000), que será mostrado en los displays
  + SW\_OK\_NUMERO : Entrada por la que se comunica mediante un pulso de 10ns que tenemos un valor nuevo en la entrada
* Señal interna:
  + BCD\_OK: Entrada del CNT\_DISPLAY, que es excitada por la señal generada por SW\_OK
* Señales de salida:
  + AND\_30\_SALIDA\_DISPLAY : Señal encargada de indicar en cuál de los 4 displays nos encontramos
  + DP\_PUNTO : Señal que indica si se debe o no encender el punto decimal
  + SEG\_AG\_DISPLAY\_SELEC : Valor en 7 segmentos a escribir en el display indicado por AND\_30
* Señales auxiliares del testBench
  + D\_Display\_Aux : Señal que guarda el valor en Bin del display actual.
  + BCD\_U : Señal que guarda el valor en Bin del display correspondiente a las unidades.
  + BCD\_D : Señal que guarda el valor en Bin del display correspondiente a las decenas.
  + BCD\_C : Señal que guarda el valor en Bin del display correspondiente a las centenas.
  + BCD\_M : Señal que guarda el valor en Bin del display correspondiente a los millares.
* Señal que muestra el estado de los displays
  + D\_DISPLAY : Esta señal recoge el valor completo escrito por los 4 displays, para poder comprobar su correcto funcionamiento.

En estas 2 imágenes se observa el efecto de la activación del SW\_OK que a su vez desencadena la activación del BCD\_OK, cuya activación carga el nuevo valor de entrada en los displays.



Ilustración 14 Primer Estimulo y tiempo de retardo

En este primer estimulo podemos observar el retardo 10.422 ns entre el flanco de subida en el cuál debería haberse producido el cambio del valor BCD a escribir en los displays y el momento en el cual se produce.

En la siguiente imagen observamos el segundo estimulo el cambio entre 1999 y 0000 y al igual que con el primer estimulo observamos un retardo entre el flanco de subida en el cual se debería haber producido el cambio en dicho valor BCD y el momento en el que ocurre, aunque esta vez dicho retardo es de 9.246 ns.

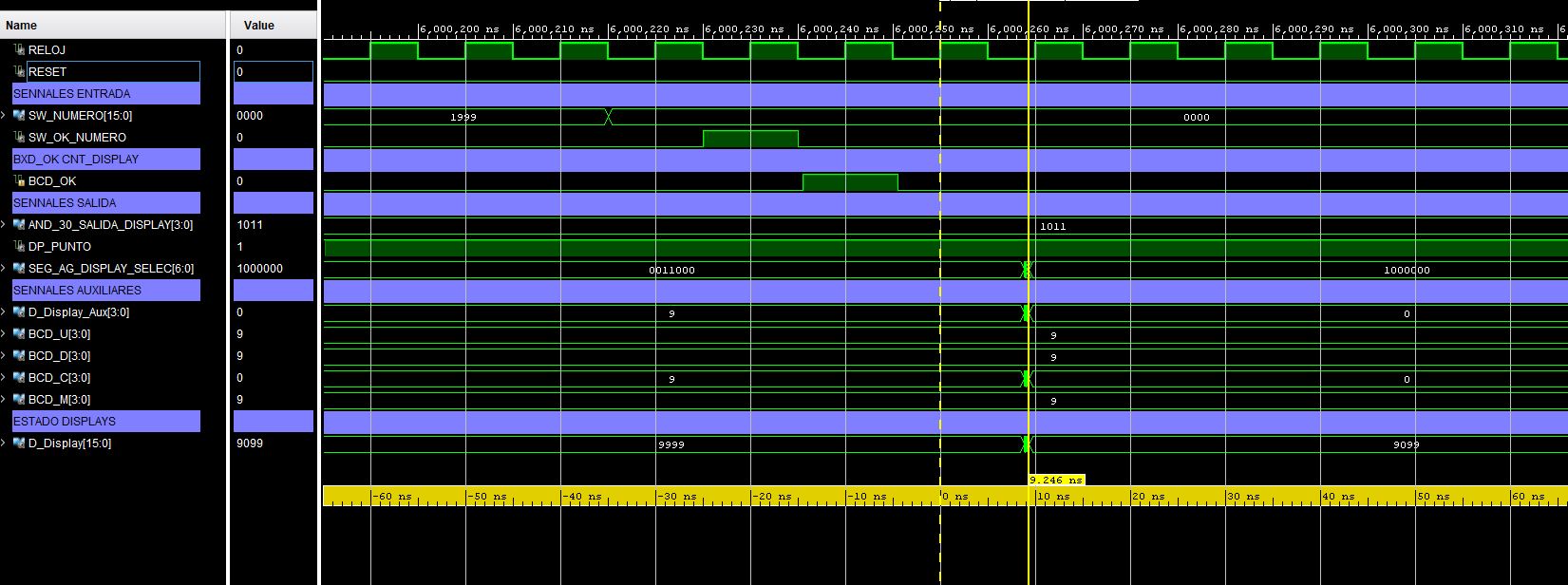


Ilustración 15 Segundo estimulo y tiempo de retardo

Tras esta simulación temporal para la cual hemos tenido que realizar la implantación del diseño creado, podemos obtener datos sobre la utilización de los recursos de la FPGA por nuestro diseño.

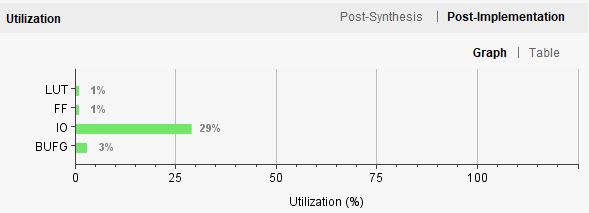


Ilustración Grafica de utilización

En la gráfica de la ilustración anterior, podemos observar que de la cantidad de recursos disponibles de cada tipo, son los recursos de tipo IO(Entrada y Salida) de los que mas se usan, un 29% de los mismos.

En la siguiente tabla observamos, que el recurso mas empleado por nuestro proyecto es el de FF(Registros).

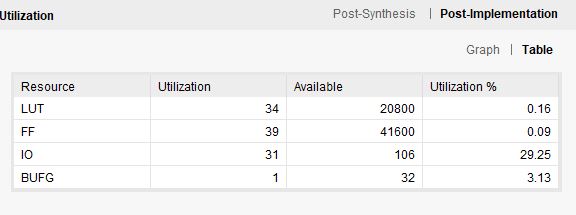


Ilustración Tabla de utilización

## CONCLUSIÓN

Miau