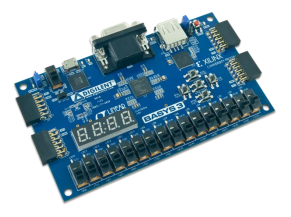
PRACTICA LIBRE: VOLTIMETRO DIGITAL IMPLEMENTADO CON UNA FPGA

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](#_Toc39308168)

[PLANTEAMIENTO 2](#_Toc39308169)

[DISEÑO 2](#_Toc39308170)

[SIMULACIÓN FUNCIONAL 8](#_Toc39308171)

[SIMULACIÓN TEMPORAL 13](#_Toc39308172)

[CONCLUSIÓN 15](#_Toc39308173)

[DISEÑO DEL MÓDULO BIN2BCD 16](#_Toc39308174)

[PLANTEAMIENTO 16](#_Toc39308175)

[DISEÑO 16](#_Toc39308176)

[SIMULACIÓN FUNCIONAL 18](#_Toc39308177)

[SIMULACIÓN TEMPORAL 18](#_Toc39308178)

[CONCLUSIÓN 18](#_Toc39308179)

[DISEÑO DEL MÓDULO CNT\_AD7476A 18](#_Toc39308180)

[PLANTEAMIENTO 18](#_Toc39308181)

[DISEÑO 19](#_Toc39308182)

[SIMULACIÓN FUNCIONAL 19](#_Toc39308183)

[SIMULACIÓN TEMPORAL 19](#_Toc39308184)

[CONCLUSIÓN 19](#_Toc39308185)

# DISEÑO DEL MÓDULO CNT\_DISPLAY

## PLANTEAMIENTO

De acuerdo con el desarrollo de la práctica, hemos tenido que desarrollar un controlador para el display de un 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 BCD 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.

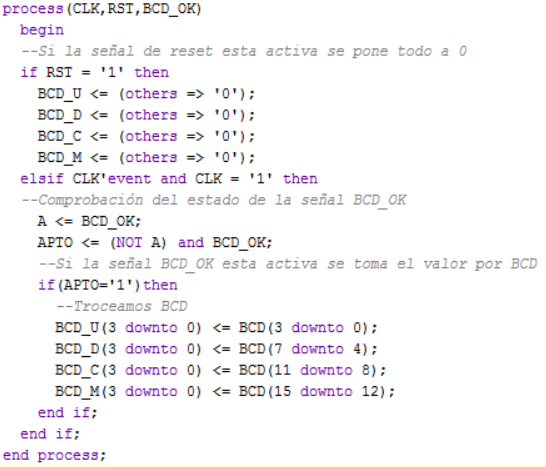


Ilustración -Implementación en VHDL del registro de 16bits

* 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 está 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).

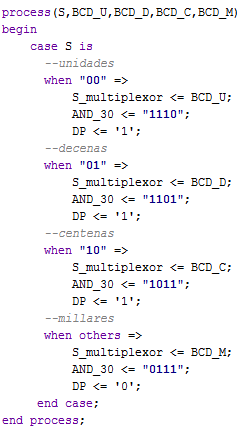


Ilustración -Implementación en VHDL del multiplexor

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.

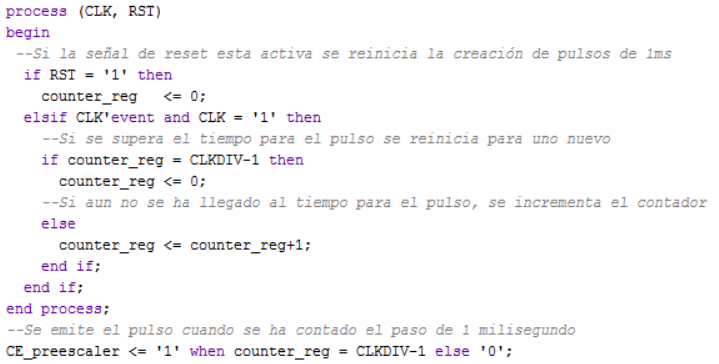


Ilustración -Implementación en VHDL del preescaler

* 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.

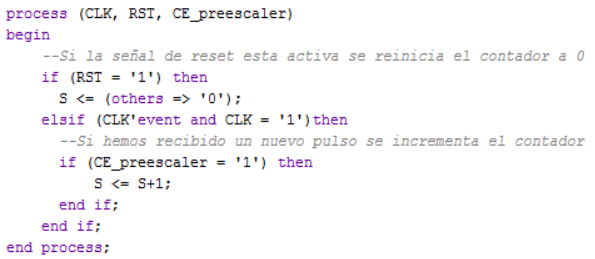


Ilustración -Implementación en VHDL del contador de 2bits

* Conversor BCD-7segmentos:

Este componente recibe la cifra que se ha de mostrar en formato BCD (S\_multiplexor) y en función del número le asigna su combinación para la visualización en 7 segmentos, 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.

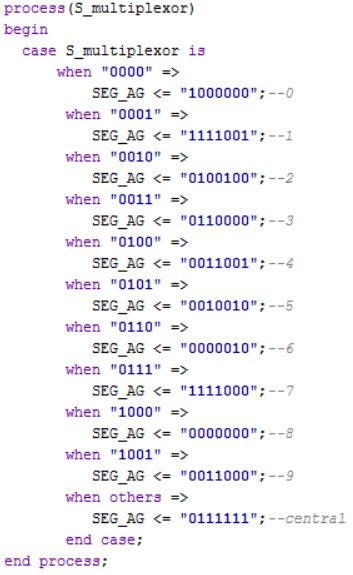
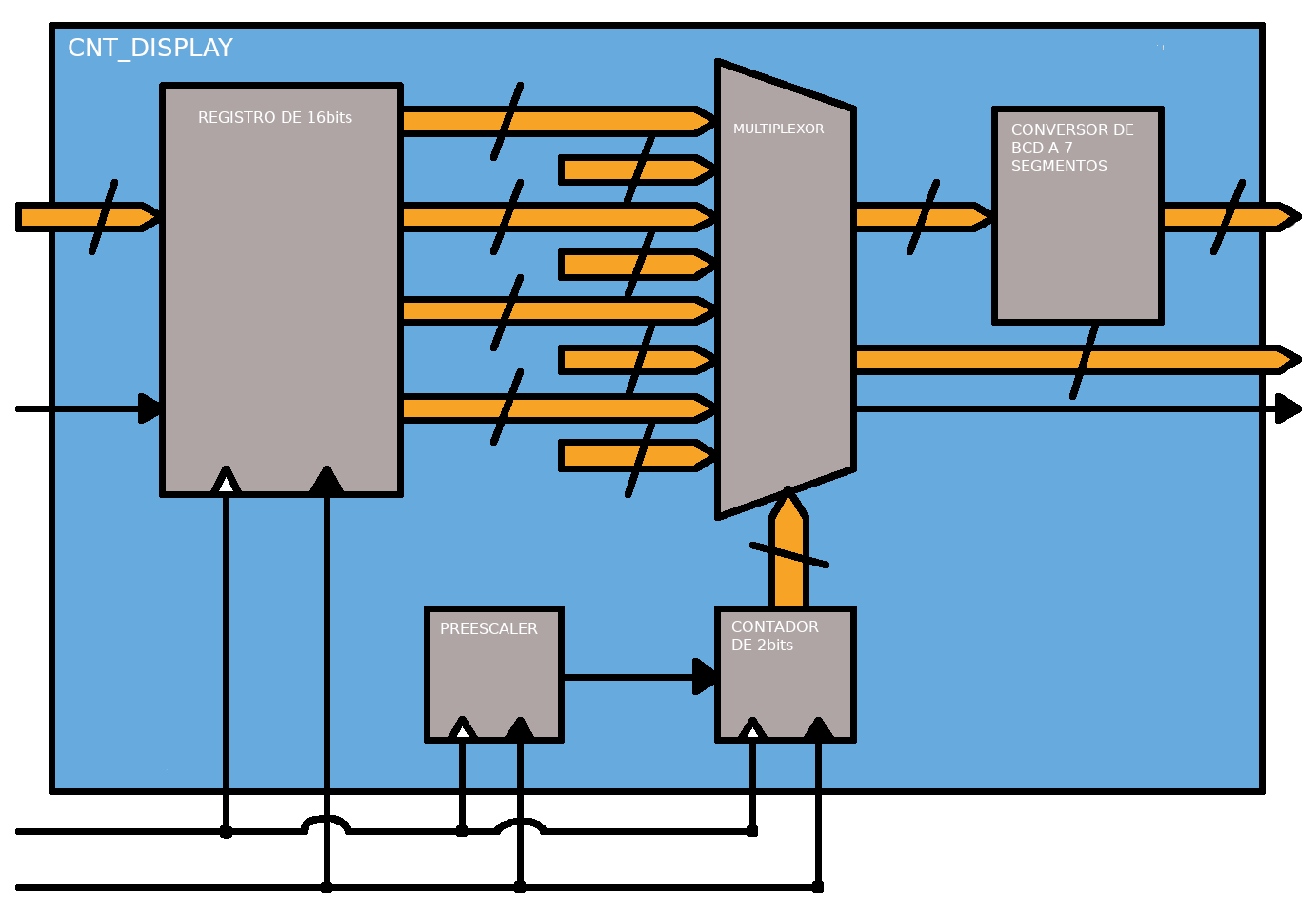


Ilustración -Implementación en VHDL del conversor de BCD a 7 segmentos

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.

Estos valores serán pasados a los displays indicados de manera secuencial cada 1ms, refrescando cada display cada 4ms mediante el uso de la señal de control del multiplexor S, generada por el contador binario de 2bits alimentando su funcionamiento mediante una señal generada en el preescaler que ocasiona las variaciones cada 1ms.

Finalmente, como último paso al visionado del valor por el display se traduce el valor BCD a 7 segmentos para su representación en el display.

## SIMULACIÓN FUNCIONAL

En la creación del testbench para ambas simulaciones, declaramos los componentes y señales apropiadas del cnt\_display y asignamos una señal inicial de RESET activa durante 123ns, desactivándose después; además de declarar la señal de RELOJ de 10ns de período

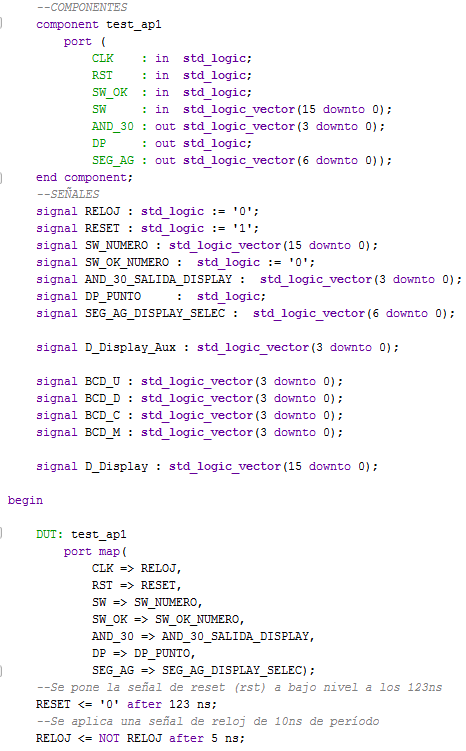


Ilustración -Asignación de señales en el testbench

En primer lugar, tenemos un proceso en el cual 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 -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

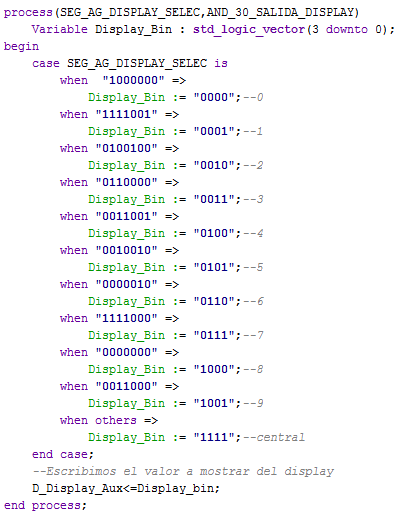


Ilustración Conversor de 7 segmentos a BCD

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.

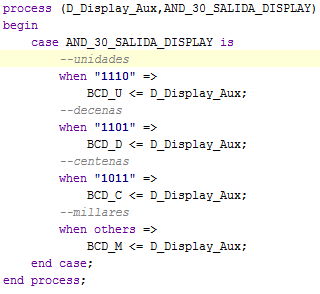


Ilustración -Proceso en VHDL para obtener valores BCD de los displays

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

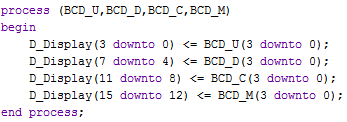


Ilustración -Proceso en VHDL para obtener el valor en BCD

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

Ilustración -Diagrama de la ejecución completa del testbench del ap1

Las señales del sistema, las cuales están divididas en diferentes grupos, son:

* 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 BCD 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 la imagen anterior observamos la estimulación de la SW\_NUMERO con dos valores distintos, “1999” primero y después “0000”. Además, se estimula la entrada BCD\_OK mediante un pulso de 10ns, para indicar la validación de un nuevo dato de entrada.

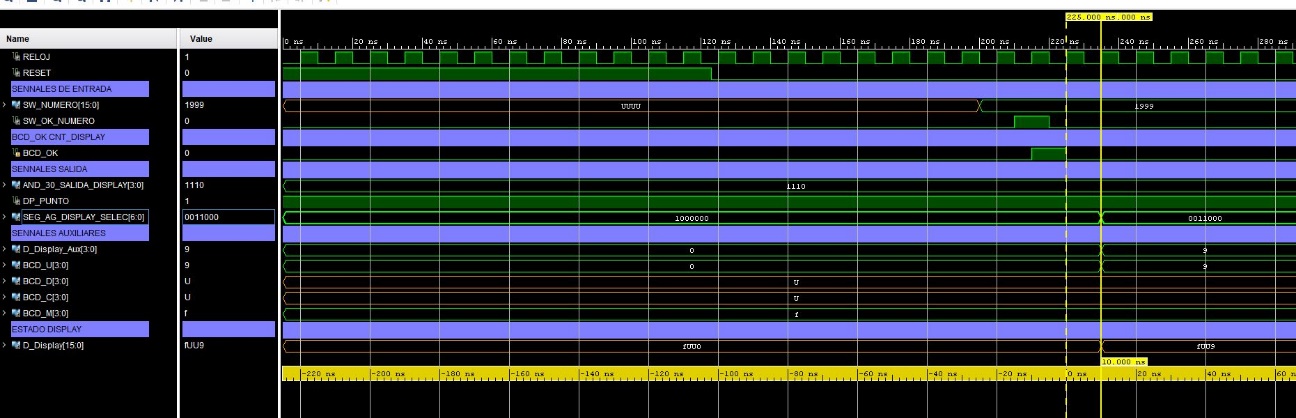
La recepción de los nuevos valores, acorde a la llegada de señal activa de BCD\_OK, y su posterior muestreo en los displays, se comprueba en las siguientes dos imágenes.

Ilustración -Estimulo inicial “1999”

Lo primero que observamos en estas dos imágenes, es que la señal SW\_OK no es la que desencadena la aceptación de un nuevo valor, sino que esta señal genera el pulso de 10ns(BCD\_OK) encargado de validar el nuevo valor de entrada.

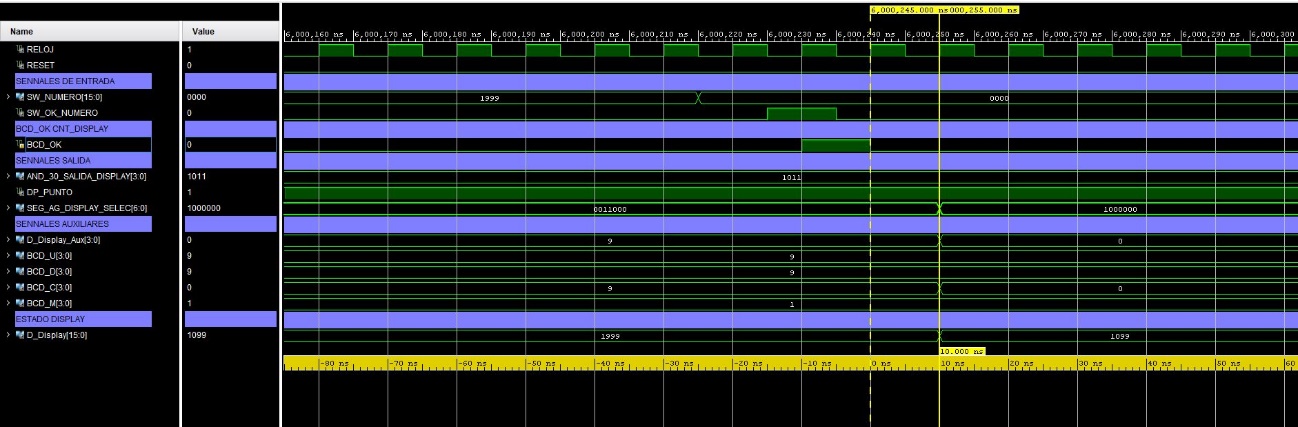


Ilustración -Segundo estimulo “0000”

También cabe destacar que, debido al funcionamiento de los procesos en el cuál las señales se actualizan al final de este, podemos observar que la actualización del valor BCD se produce en el flanco de subida del reloj posterior al que detecta el BCD\_OK.

## 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, que es lanzada tras la implementación, 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, como muestra la siguiente imagen.

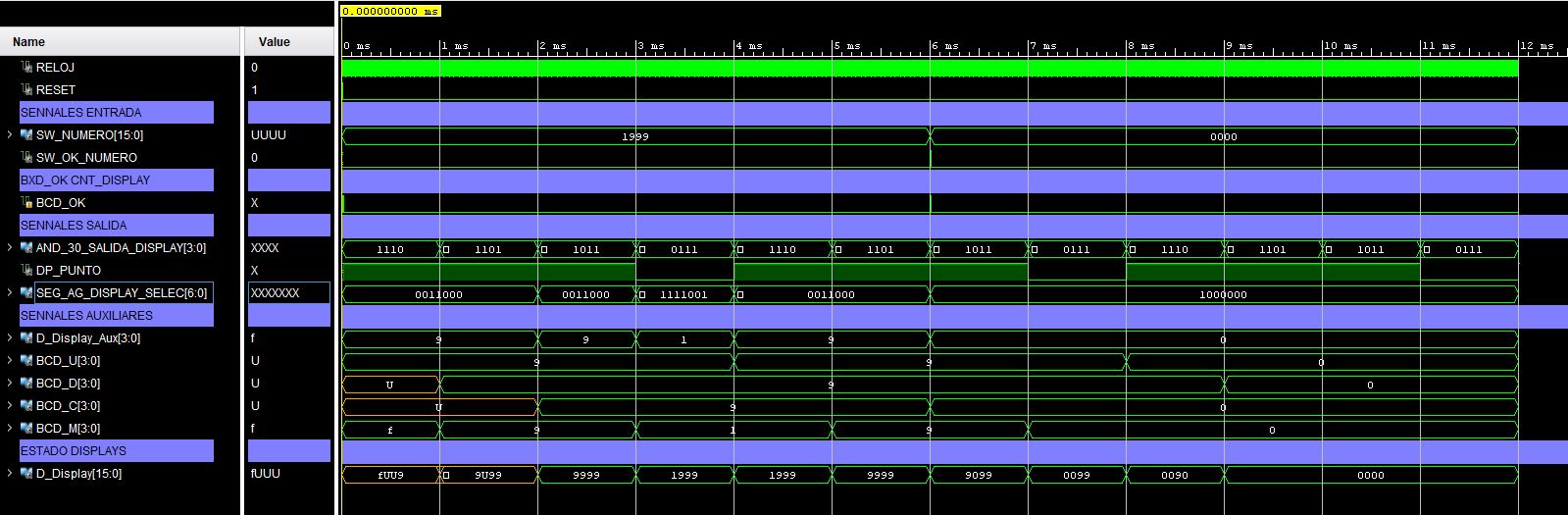
En la siguiente ilustración podemos observar todas las señales del sistema las cuales se encuentran explicadas en el apartado de la simulación funcional.

Ilustración -Simulación temporal completa

En las 2 imágenes siguientes 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 -Primer estimulo “1999” 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.

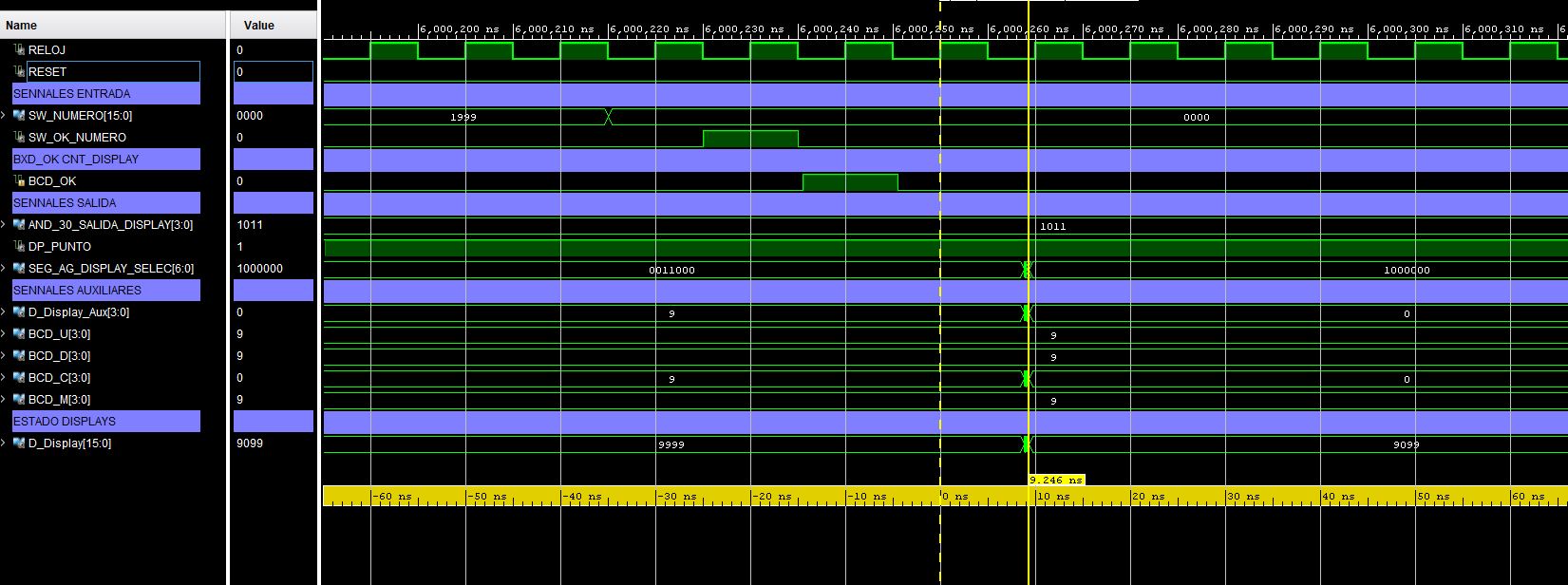
En la siguiente imagen observamos el segundo estimulo el cambio entre “1999” y “0000" y al igual que con el primer estimulo observamos un retardo entre el flanco de subida en el cual se debería haber producido el cambio en dicho valor BCD y el momento en el que ocurre, aunque esta vez dicho retardo es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

Ilustración -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.

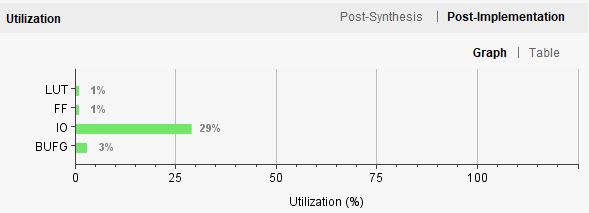


Ilustración - Grafica de utilización

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 más se usan, un 29% de los mismos.

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

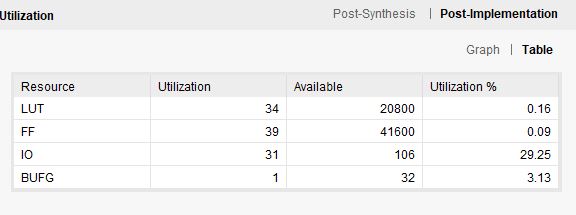


Ilustración -Tabla de utilización

## CONCLUSIÓN

Tras la realización de este apartado, hemos llegado a una serie de conclusiones principales.

La primera de ellas seria la de no pensar que estamos programando código y tener siempre presente que se trata de modelar un hardware que tiene que poder ser sintetizable e implementable.

La segunda conclusión es la de lo importante que es tener cuidado con el lugar y la forma de emplear las señales para así evitar la inferencia de latches.

Y como última conclusión, pero no menos importante por ello, tendríamos la importancia de realizar un testbench robusto para así poder detectar cualquier pequeño error en nuestro diseño.

# DISEÑO DEL MÓDULO BIN2BCD

## PLANTEAMIENTO

Tras el diseño del módulo cnt\_display, ya disponemos del último apartado del proyecto el cuál mostrará por los displays el voltaje medido en el módulo del cnt\_AD7476A. Para disponer de la entrada a este último módulo en formato BCD debemos incluir un elemento intermedio entre estos dos capaz de convertir el valor en binario puro de salida del primer módulo a formato BCD para la entrada en el último módulo.

Por tanto, se diseñará un conversor de binario a BCD a partir de las señales de entrada al conversor con el valor en binario medido en el primer módulo (DATA) y su señal de control que indica la llegada de un nuevo valor (DATA\_OK), que una vez realizada la conversión se emitirá el valor en formato BCD (BCD) y su señal de control para notificar al módulo del cnt\_display que tiene un nuevo valor de entrada (BCD\_OK).

## DISEÑO

Para empezar a trabajar, a partir del esquema del pdf con el enunciado, planteamos como ha de conformarse el interior del módulo a través de los componentes mostrados en el diagrama, cuyas funcionalidades se implementarán como procesos. Siendo los componentes los enumerados a continuación:

* Contador binario descendente de 12 bits:

Este componente recibe por entrada el valor muestreado en formato binario (DATA) y cuando la señal de validación del nuevo valor (DATA\_OK) llega, comienza una cuenta regresiva desde el valor recibido hasta el cero, la cual iteración por iteración es emitida al resto de componentes (cnt\_bin\_out).

En caso de que llegue la señal activa de reset (RST) se procede a reiniciar el contador mediante el ajuste de la señal cnt\_bin\_out a cero, en espera de un nuevo valor.

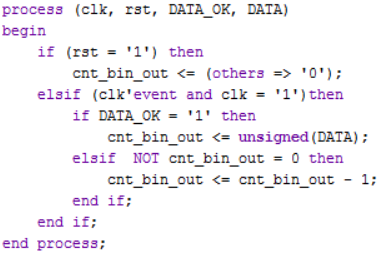


Ilustración -Implementación en VHDL del contador binario descendente de 12bits

* Contador BCD ascendente de 4 dígitos:

Este componente recibe por entrada el valor en decremento del primer contador (cnt\_bin\_out) y la señal que notifica la existencia de un nuevo dato para empezar el conteo (DATA\_OK). Este contador reflejará su conteo a través de las señales de salida hacía el registro de 16 bits (BCD\_U, BCD\_D, BCD\_C y BCD\_M) mientras que el primer contador no sea cero (cnt\_bin\_out) consiguiendo el valor binario conseguido por entrada por el primer contador (DATA) pero en formato BCD.

En caso de que llegue la señal activa de reset (RST) se procede a reiniciar el contador mediante el ajuste de las señales BCD que representan cada cifra a cero (BCD\_U, BCD\_D, BCD\_C y BCD\_M), en espera de un nuevo valor.

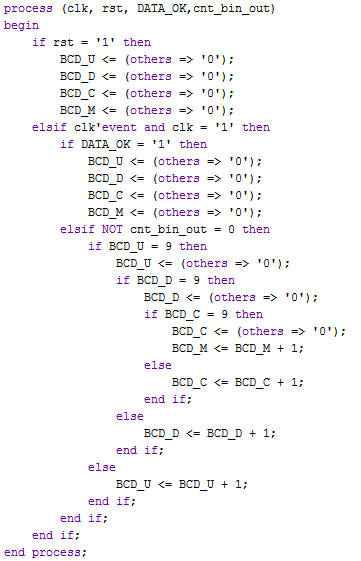


Ilustración -Implementación en VHDL del contador BCD ascendente de 4 dígitos

* Generador de BCD\_OK:

Este componente recibe por entrada el valor en decremento por cada iteración hasta llegar a 0 (cnt\_bin\_out), este hecho ocasionará que se emita un pulso por la salida (BCD\_OK) para que el módulo cnt\_display trabaje con el nuevo valor BCD que se le ofrece.

En caso de que llegue la señal activa de reset (RST) se procede a poner la señal de salida en estado bajo (BCD\_OK).

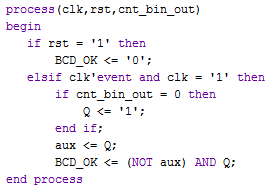


Ilustración -Implementación en VHDL del generador de BCD\_OK

* Registro de 16 bits:

Este componente recibe por entrada los valores del número en formato BCD generados por el contador de 4 dígitos en BCD (BCD\_U, BCD\_D, BCD\_C y BCD\_M) y en el momento que el primer contador a llegado a cero (cnt\_bin\_out), es el momento en el que el generador ocasionara un nuevo pulso por salida (BCD\_OK) por lo que se ensamblará todos los dígitos en una única señal que será el valor en binario que entro en el módulo convertido a formato BCD para el siguiente módulo (BCD).

En caso de que llegue la señal activa de reset (RST) se procede a poner la señal de salida igual a cero (BCD).

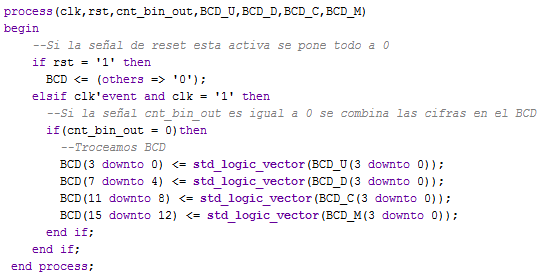
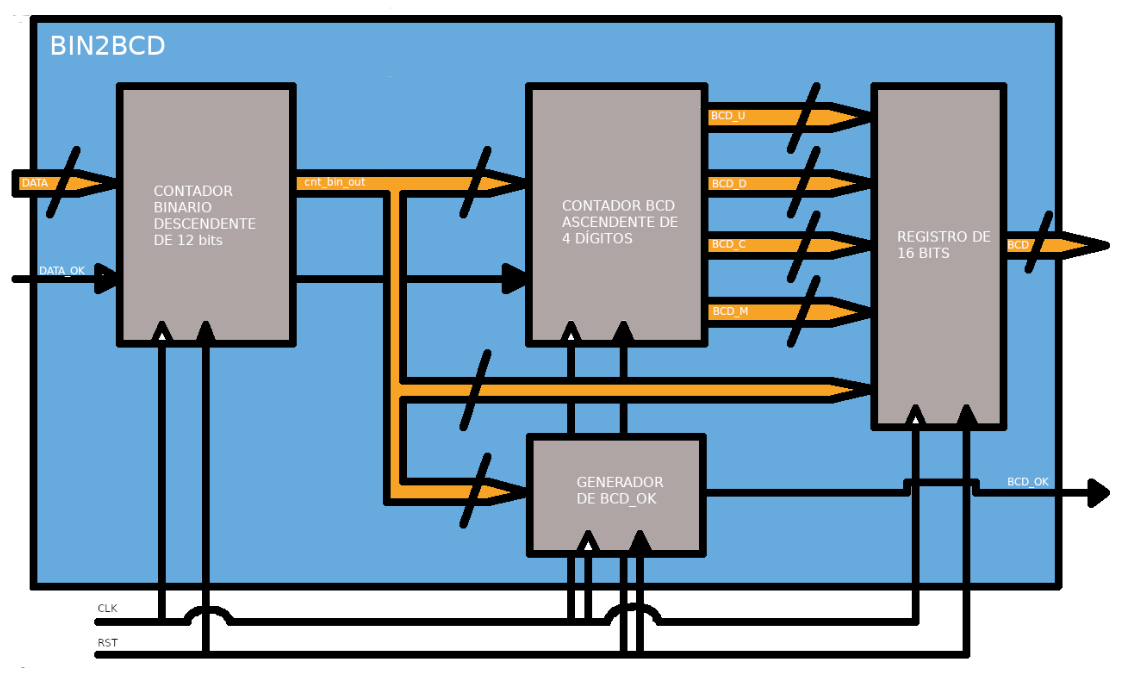


Ilustración -Implementación en VHDL del registro de 16 bits



## SIMULACIÓN FUNCIONAL

Para demostrar el funcionamiento del módulo desarrollamos un testbench para comprobar que el código escrito se comporta de manera igual a la funcionalidad que esperamos de cada componente.

Por tanto, el banco de pruebas consta de una duración de 12 ms en el cuál tras asignar el período del reloj en 10ns como precisa el enunciado, le planteamos al módulo la conversión de un dato inicial igual a “000001100100” (100) para que después de 5ms varíe al nuevo dato de “000011001000” (200) y también lo convierta a formato BCD.

Con esto comprobamos la funcionalidad básica de conversión de formato binario a decimal, además de una pequeña demostración de su comportamiento cuando DATA es cargado con otro valor. A continuación, se muestra el código del testbench:

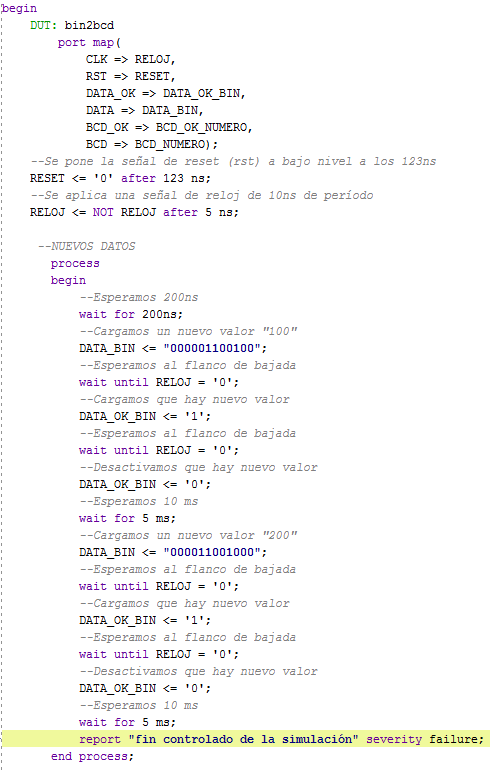


Ilustración -Testbench en VHDL para la simulación funcional del módulo

En el inicio de la simulación podemos observar como iniciamos con todos los valores iguales a 0 o en estado bajo y observamos el primer cambio cuando varía el valor de DATA en el instante 200ns después del inicio de la simulación, con el valor de “000001100100” (100). Un periodo de reloj más tarde se inicia el pulso de DATA\_OK comunicando la existencia de un nuevo dato en el módulo, hecho que cuando es detectado los contadores se ponen en funcionamiento. El contador binario parte desde el valor de entrada de DATA e inicia su decremento, mientras que el segundo contador BCD inicia desde 0. Los valores binarios en los cronogramas se muestran en base decimal para una mayor facilidad en la lectura de resultados.

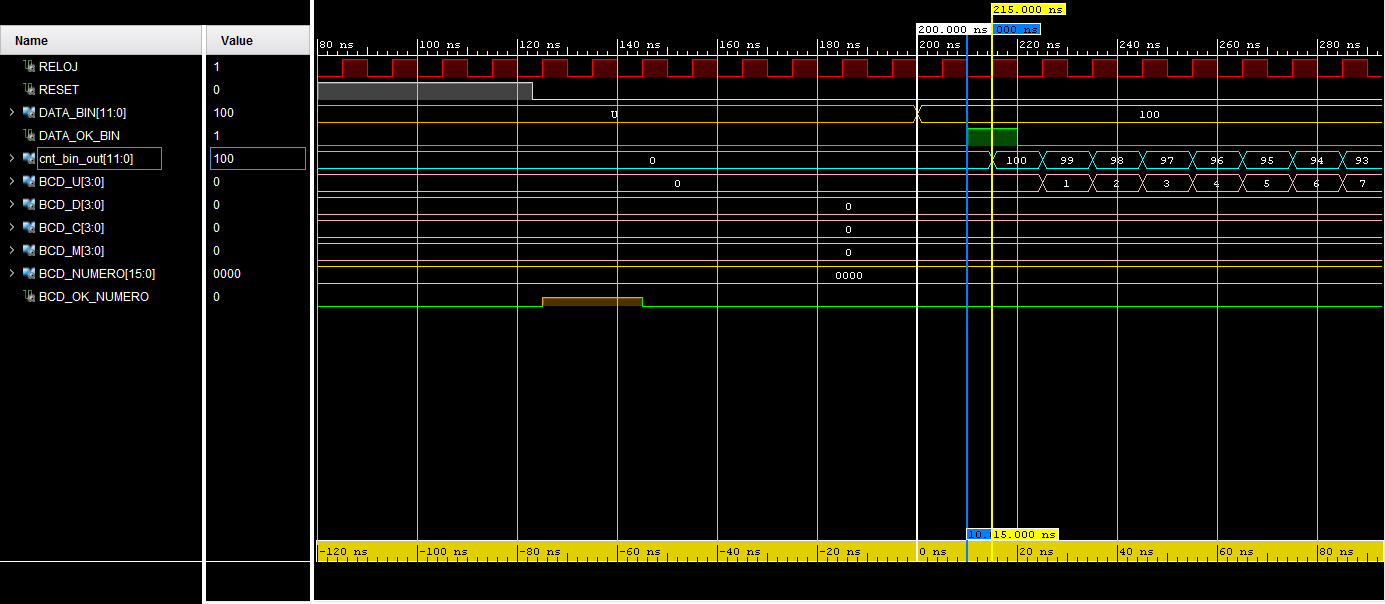
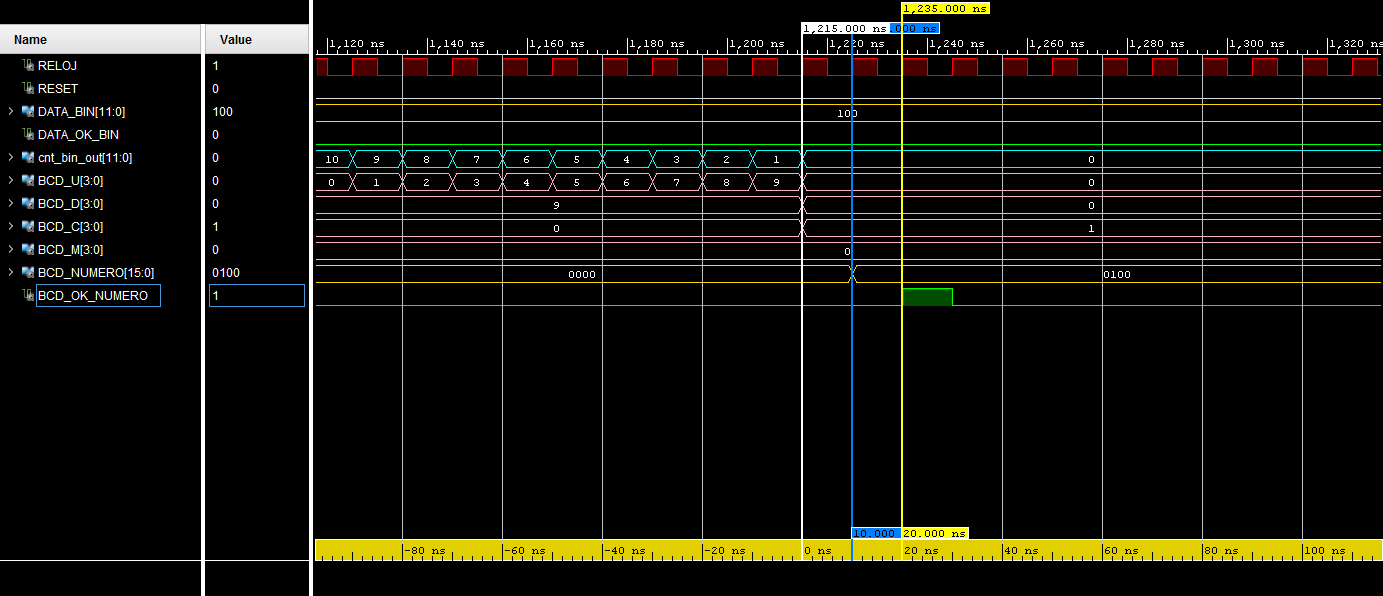
Los contadores continuarán con sus actividades hasta que el contador binario llega a 0, evento que genera un nuevo cambio en la simulación. Este evento indica que ya hemos obtenido la conversión del valor de entrada en binario a formato BCD, por tanto, se inicia la salida del dato convertido y su señal de notificación al siguiente módulo. Esto se comprueba en el cambio del valor de la señal BCD un periodo más tarde de que el contador binario llegue a cero, señal construida a partir de los fragmentos BCD que genera el contador BCD en sus iteraciones; seguida en un período más tarde por el pulso de la señal BCD\_OK que notifica al cnt\_display que tiene un nuevo valor a representar.

Ilustración -Cronograma del final de la conversión del valor DATA igual a 100 a BCD

Ilustración -Cronograma de inicio de la simulación con un nuevo valor de DATA igual a 100

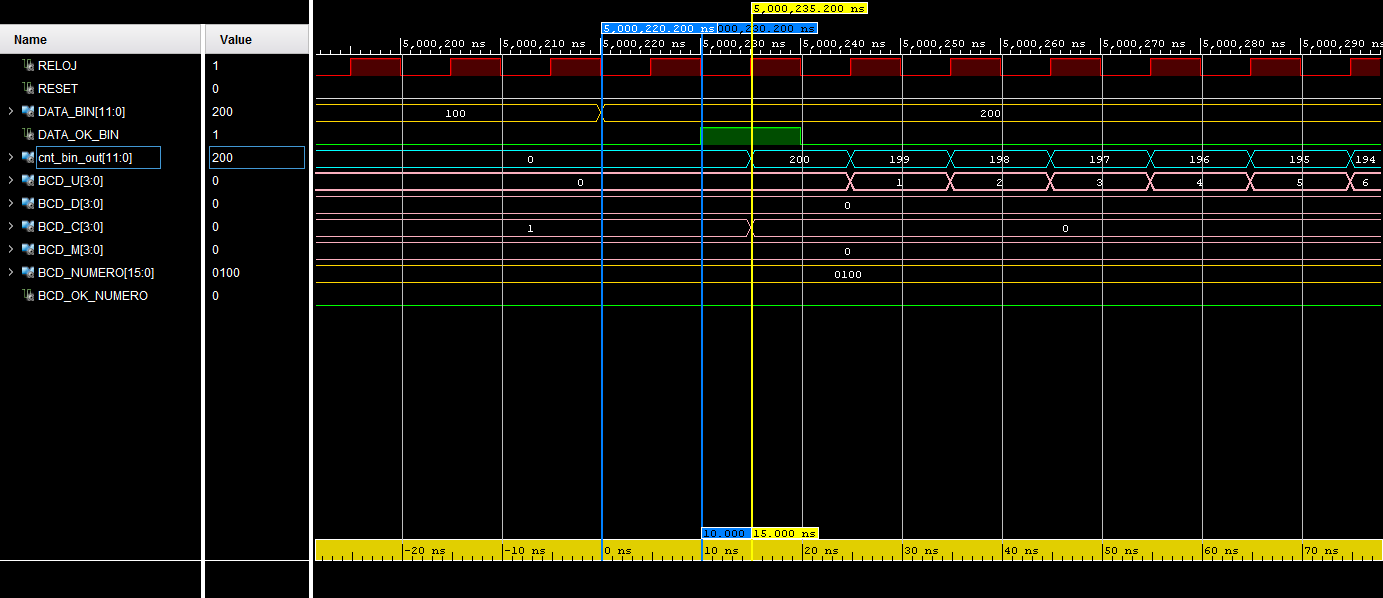
A continuación, llegamos al momento en el que el valor de DATA es variado a “000011001000” (200) acompañado tras un período del pulso en la señal DATA\_OK que confirma la variación en el dato de entrada al conversor y al igual que antes en cuanto se detecta en flanco de subida el pulso, se inicia el funcionamiento de los contadores. A diferencia que en el anterior caso el contador binario decrementará su valor a partir del 200 debido al valor de entrada del DATA.

Ilustración -Cronograma de variación del valor DATA de 100 a 200, iniciándose nueva conversión

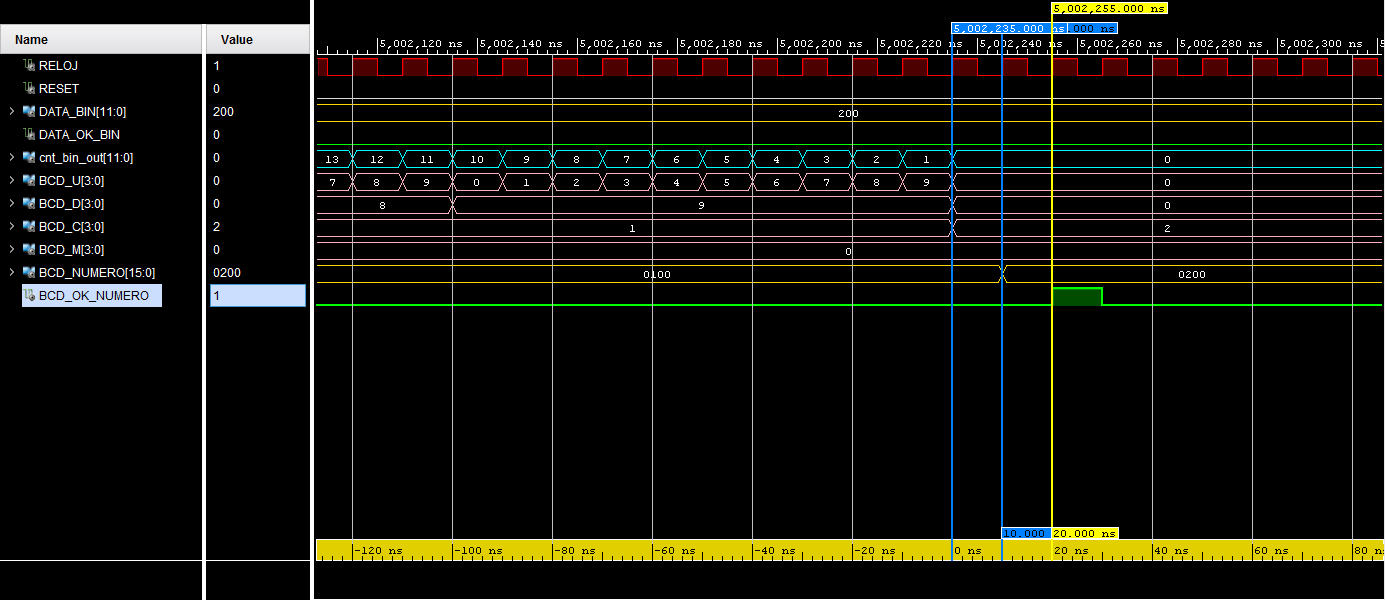
Finalmente se localiza el último punto destacado de la simulación cuando termina la conversión del valor 200 al formato BCD cuando al igual que con el valor 100, el contador binario llega a 0, reflejando los mismos cambios en los mismos tiempos de respuesta; estos son el cambio en el valor de BCD con el nuevo valor convertido tras un periodo de reloj tras la llegada del contador a 0 acompañado por un pulso de la señal BCD\_OK un periodo más tarde para la notificación al próximo módulo de un nuevo valor en BCD.

Ilustración -Cronograma del final de la conversión del valor de DATA igual a 200 a BCD

## SIMULACIÓN TEMPORAL

A continuación, y para acabar en las comprobaciones del buen funcionamiento del módulo se procede a realizar una simulación temporal sobre hardware para comprobar su comportamiento en un entorno más cercano al real.

Para esta última simulación usamos el mismo testbench de la anterior simulación, el cuál se muestra en el anterior apartado en la ilustración 23.

En el inicio de la simulación podemos observar cómo iniciamos con todos los valores iguales a 0 o en estado bajo y observamos el primer cambio cuando varía el valor de DATA en el instante 200ns después del inicio de la simulación, con el valor de “000001100100” (100). Un periodo de reloj más tarde se inicia el pulso de DATA\_OK comunicando la existencia de un nuevo dato en el módulo, hecho que cuando es detectado los contadores se ponen en funcionamiento. El contador binario parte desde el valor de entrada de DATA e inicia su decremento, mientras que el segundo contador BCD inicia desde 0. Los valores binarios en los cronogramas se muestran en base decimal para una mayor facilidad en la lectura de resultados.

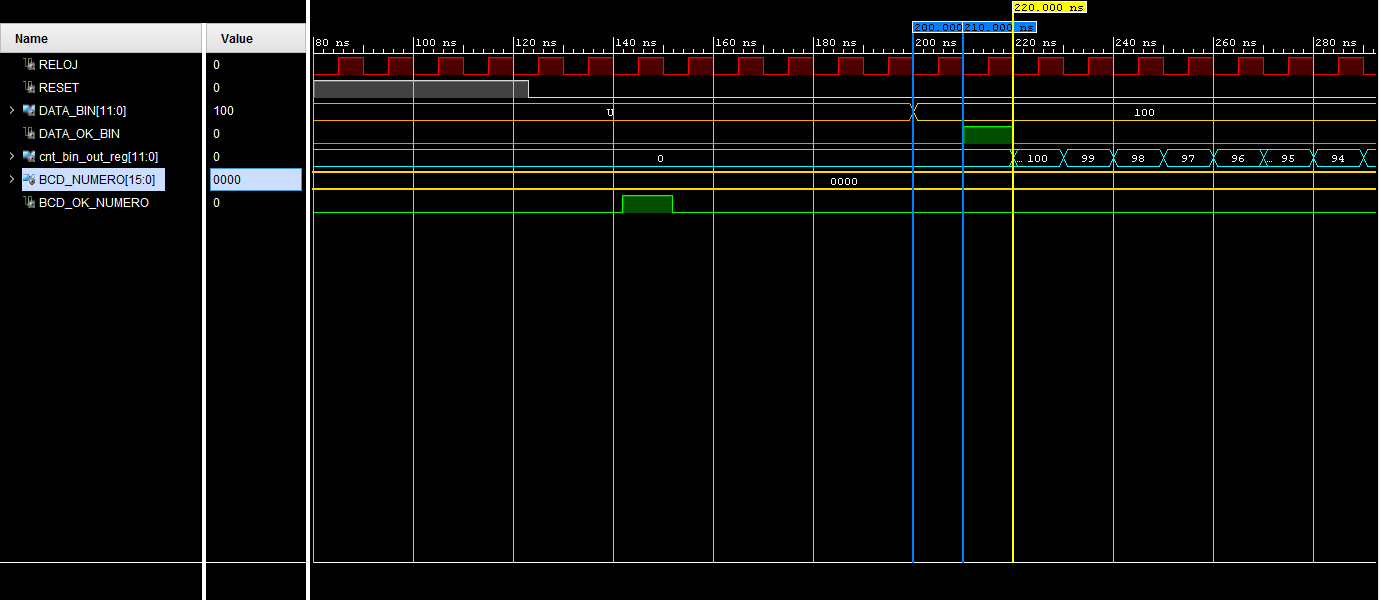
En primer lugar, se observa tras el reset y partir con todos los datos con valor 0 y las señales a nivel bajo, observamos el primer cambio en el momento que se carga el primer valor que le asignamos a DATA, siendo este “000001100100” (100), y tras 1 periodo de reloj como en la anterior simulación se produce el pulso en la señal DATA\_OK para notifica el nuevo valor en DATA. Seguido del pulso se aprecia el primer cambio respecto a la simulación funcional con la activación de los contadores medio período más tarde que en el primer caso.

Ilustración -Cronograma de inicio de la simulación con un nuevo valor de DATA igual a 100

Cuando llega el contador binario a 0 también podemos apreciar cambios en los tiempos de los cambios de valores y señales respecto al anterior caso, viéndose el primer cambio en el valor nuevo del BCD tardando este un poco más de un período, acompañado por el pulso en su señal de notificación BCD\_OK para el cnt\_display tras otros 10ns.

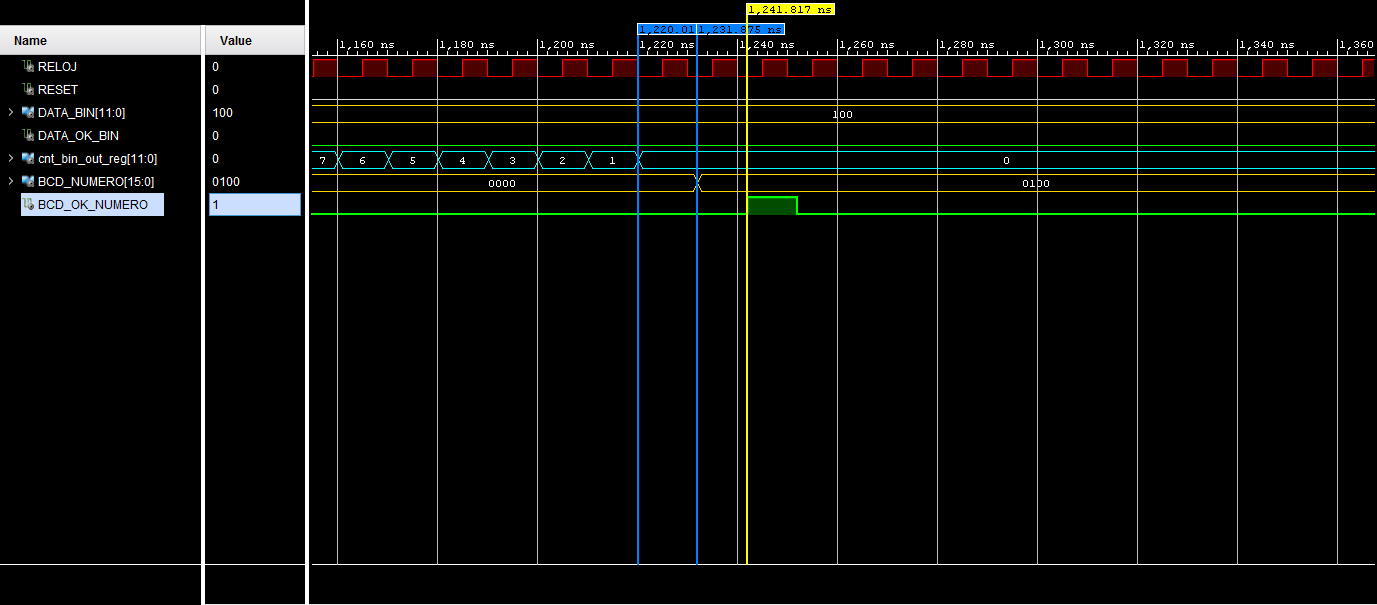


Ilustración -Cronograma del final de la conversión del valor DATA igual a 100 a BCD

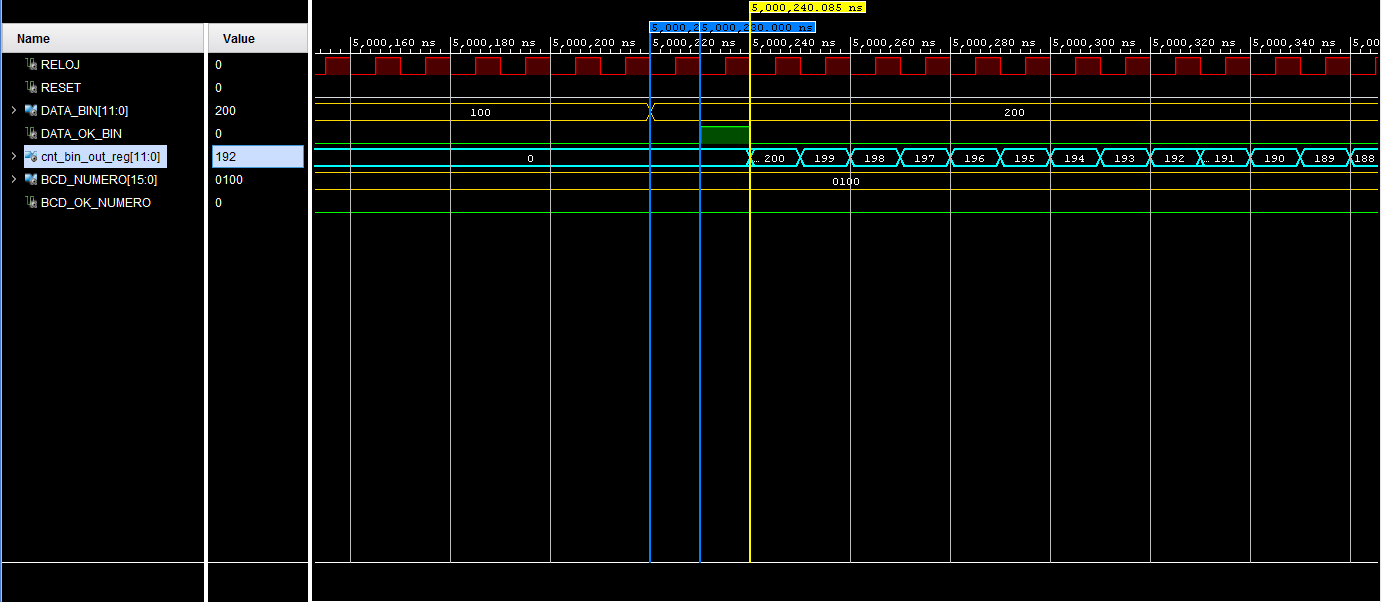
A continuación, llega el momento en el que el valor del DATA varía a “000011001000” (200) iniciándose el proceso análogo a cuando se puso el valor DATA a “000001100100” (100), iniciándose una nueva conversión. Produciéndose un nuevo pulso en DATA\_OK tras un período de la variación del valor de DATA y tras finalizar este se inicia el funcionamiento de los contadores.

Ilustración -Cronograma de variación del valor DATA de 100 a 200, iniciándose nueva conversión

Finalmente se llegará al final de la conversión cuando el contador binario llega a 0, observándose el mismo comportamiento que cuando se finalizo la conversión de binario a BCD del valor 100. Tras la llegada del contador a cero, el nuevo valor BCD a partir de la conversión realizada se carga tras periodo y cuarto, para ser acompañado 10ns después por su correspondiente señal de notificación de nuevo valor cargado en BCD mediante un pulso en BCD\_OK.

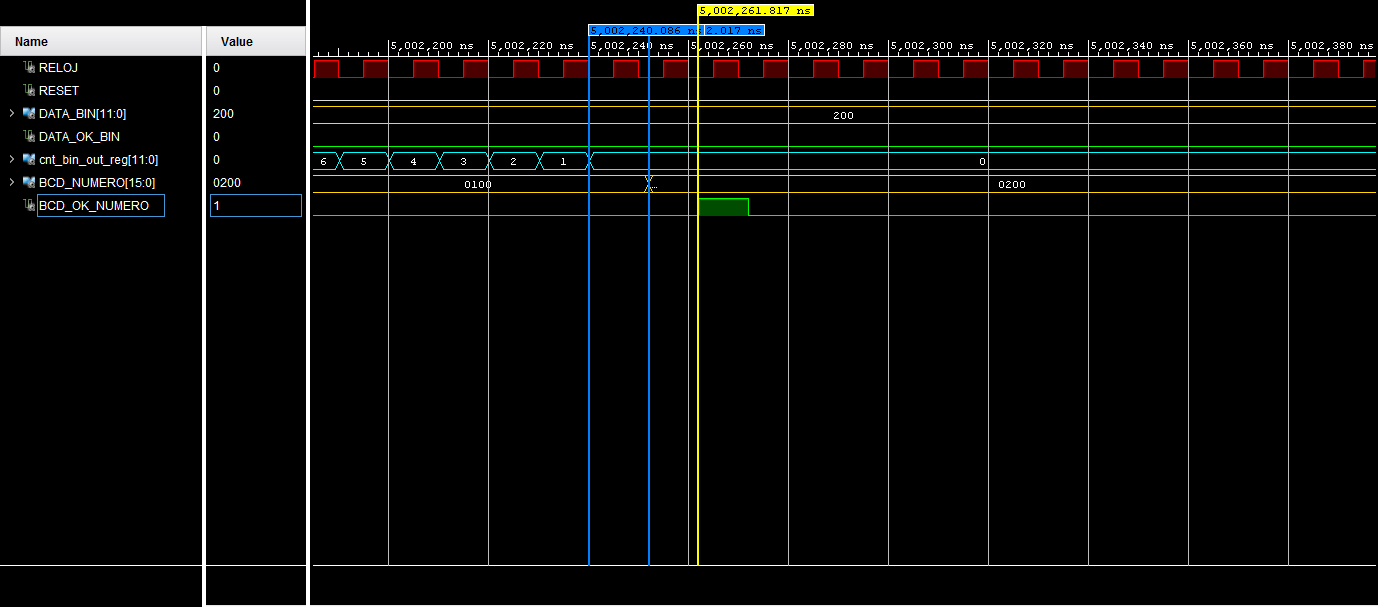


Ilustración Cronograma del final de la conversión del valor de DATA igual a 200 a BCD

## CONCLUSIÓN

# DISEÑO DEL MÓDULO CNT\_AD7476A

## PLANTEAMIENTO

Finalmente, tras el desarrollo de los otros dos módulos llega el momento de diseñar el módulo que dará sentido y utilidad a los módulos llevados acabo hasta ahora. Este módulo cumple la funcionalidad de medir la señal que se proporciona por entrada y que será comunicada hasta el último módulo para mostrar su valor en los displays.

Debido a la naturaleza analógica de la señal a medir esta ha de ser muestreada para obtener un valor digitalizado, esto se llevará acabo mediante la comunicación de la frecuencia de reloj en la que se llevará acabo el muestreo (SCLK) y la señal de control que indica que se puede muestrear la siguiente señal (CS) cuyo resultado será la obtención de valores discretos de la señal mediante la señal de entrada (S\_DATA). Tras el muestreo completo de la señal y conocer su valor este módulo envía el valor en binario al siguiente módulo (DATA) junto a la señal de control que indica al próximo módulo la existencia de un nuevo valor para representar (DATA\_OK).

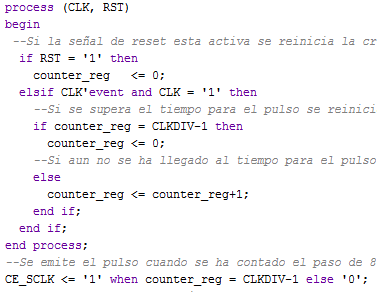
## DISEÑO

Iniciando en el diseño del último diseño partimos desde las señales indicadas en el enunciado que tiene el módulo para plantear los componentes que lo compondrán, cuyas funcionalidades se presentan mediante procesos. Siendo los componentes los siguientes:

* Preescaler:

Este componente se dedica a reajustar a partir de la señal de reloj (CLK) a ocasionar pulsos en la frecuencia que determinamos para el muestreo de la señal entrante.

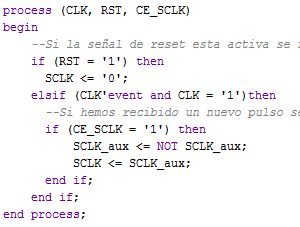
En caso de que llegue la señal activa de reset (RST) se procede a poner el contador interno del preescaler a 0 para reiniciar la generación de pulsos.



* Emisor de SCLK:

Este componente a partir de la señal por entrada del preescaler (CE\_SCLK) se emite una señal a modo de reloj para el muestreo con pulsos regulares con mismo tiempo en estado alto que en estado bajo para dominar al elemento muestreador que nos dará el valor de la señal medida.

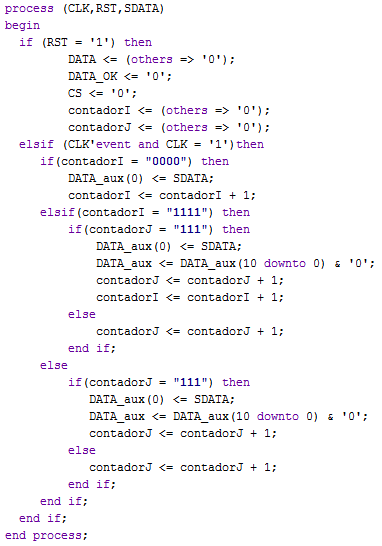
En caso de que llegue la señal activa de reset (RST) se procede a poner la señal de reloj emitida (SCLK) en estado bajo para reiniciar así la señal periódica para dominar el muestreador.



* Conversor a binario:

Este componente a partir de la señal con los valores de la señal muestreada (SDATA) recoge iteración por iteración los valores facilitados por el componente muestreador para que en última instancia cuando se ha acabado el muestreo, el resultado de los valores ensamblados que han ido llegando forman el número binario que representa el valor de la señal muestreada y que se trasmitirá a los siguiente módulos para su apropiada representación (DATA) acompañada de la señal que indica la existencia de un nuevo valor (DATA\_OK).

En caso de que llegue la señal activa de reset (RST) se procede a poner los datos del valor muestreado igual a cero (DATA) y la señal que indica que hay un nuevo valor también se reinicia a estado bajo (DATA\_OK).



## SIMULACIÓN FUNCIONAL

## SIMULACIÓN TEMPORAL

## CONCLUSIÓN