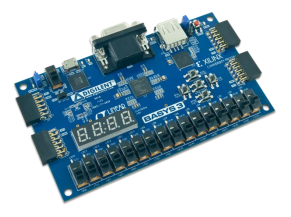
PRACTICA LIBRE: VOLTIMETRO DIGITAL IMPLEMENTADO CON UNA FPGA



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](#_Toc39594235)

[PLANTEAMIENTO 2](#_Toc39594236)

[DISEÑO 2](#_Toc39594237)

[SIMULACIÓN FUNCIONAL 7](#_Toc39594238)

[SIMULACIÓN TEMPORAL 12](#_Toc39594239)

[CONCLUSIÓN 14](#_Toc39594240)

[DISEÑO DEL MÓDULO BIN2BCD 15](#_Toc39594241)

[PLANTEAMIENTO 15](#_Toc39594242)

[DISEÑO 15](#_Toc39594243)

[SIMULACIÓN FUNCIONAL 18](#_Toc39594244)

[SIMULACIÓN TEMPORAL 21](#_Toc39594245)

[CONCLUSIÓN 24](#_Toc39594246)

[DISEÑO DEL MÓDULO CNT\_AD7476A 25](#_Toc39594247)

[PLANTEAMIENTO 25](#_Toc39594248)

[DISEÑO 26](#_Toc39594249)

[SIMULACIÓN FUNCIONAL 28](#_Toc39594250)

[SIMULACIÓN TEMPORAL 31](#_Toc39594251)

[CONCLUSIÓN 33](#_Toc39594252)

[VOLTIMETRO 34](#_Toc39594253)

[DISEÑO 34](#_Toc39594254)

[SIMULACIÓN FUNCIONAL 34](#_Toc39594255)

[SIMULACIÓN TEMPORAL 38](#_Toc39594256)

[CONCLUSIONES 42](#_Toc39594257)

# 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.

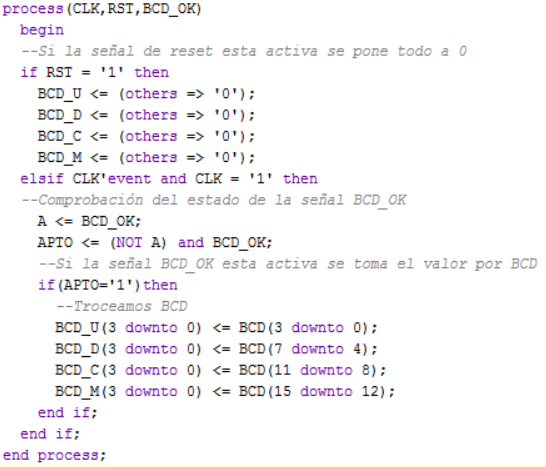


Ilustración -Implementación en VHDL del registro de 16bits

* 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).

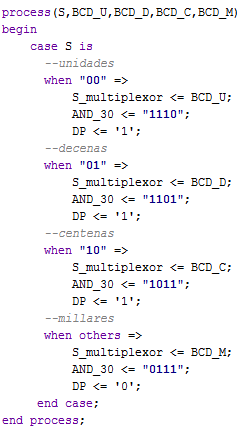


Ilustración -Implementación en VHDL del multiplexor

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.

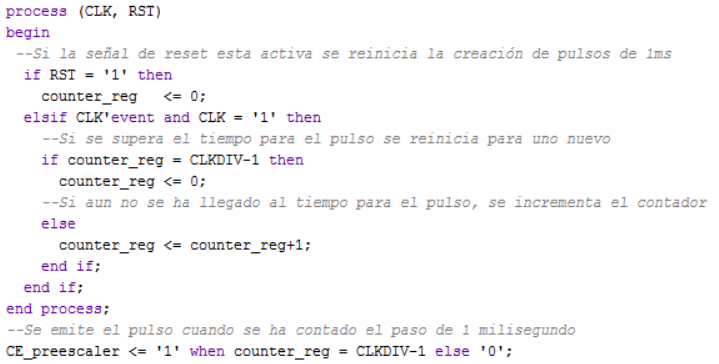


Ilustración -Implementación en VHDL del preescaler

* 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.

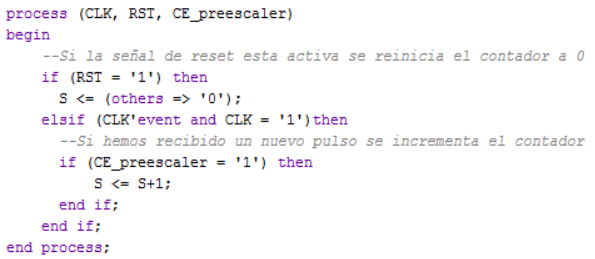


Ilustración -Implementación en VHDL del contador de 2bits

* 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.

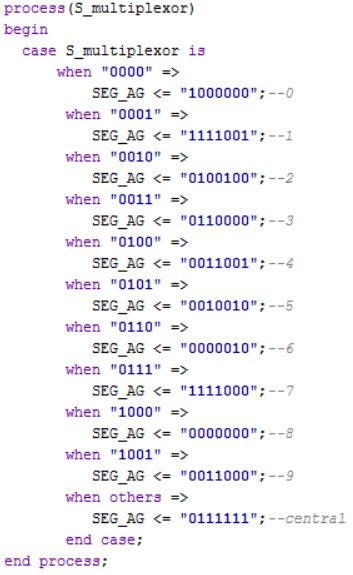
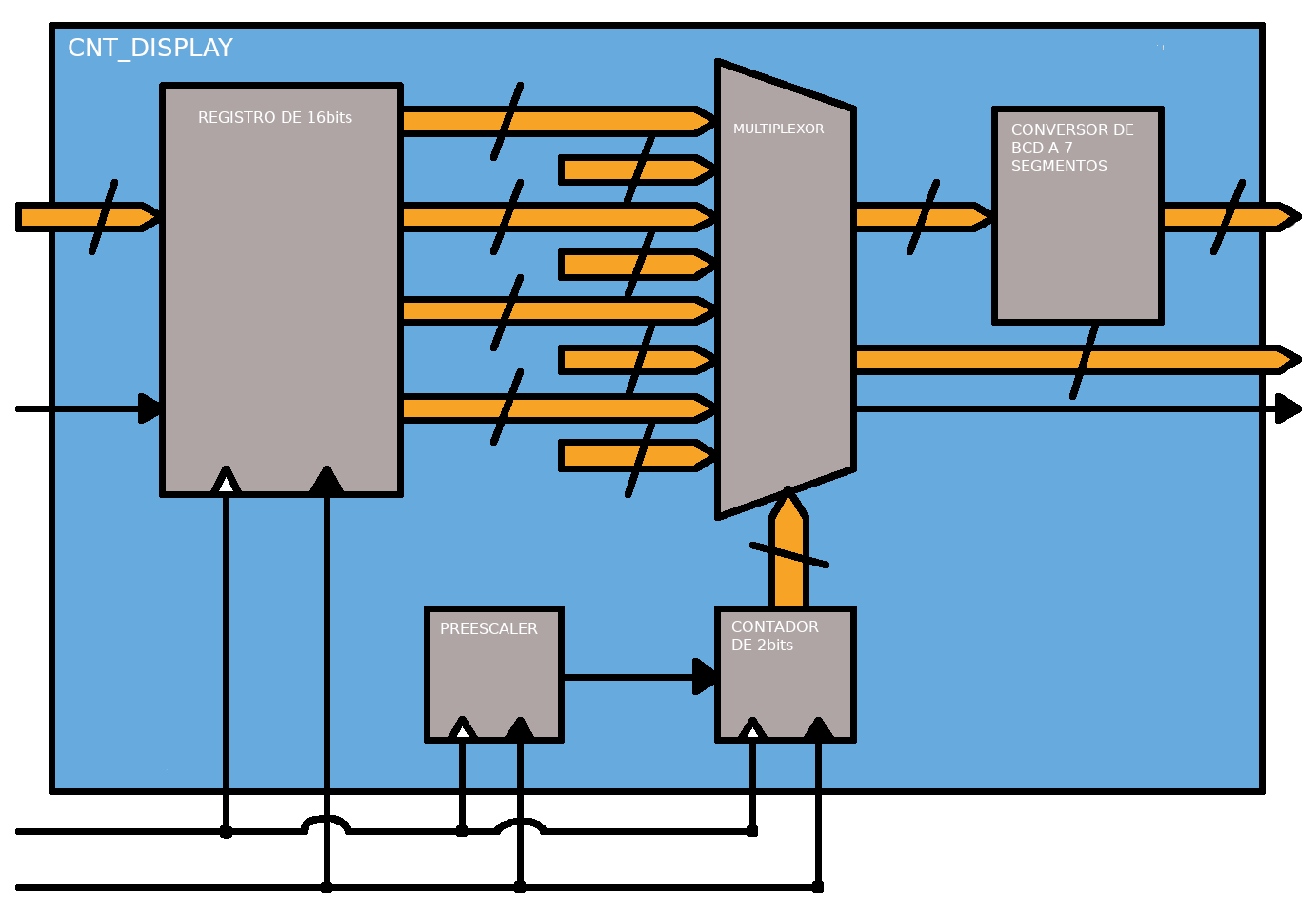


Ilustración -Implementación en VHDL del conversor de BCD a 7 segmentos

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

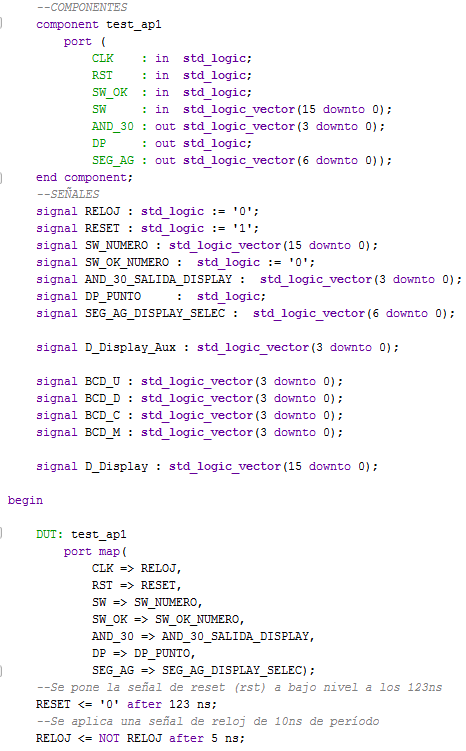


Ilustración -Asignación de señales en el testbench

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

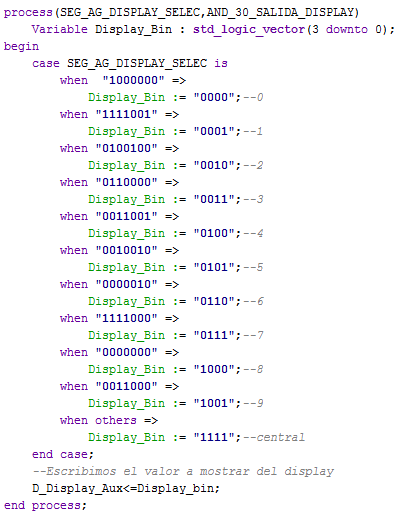


Ilustración Conversor de 7 segmentos a BCD

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.

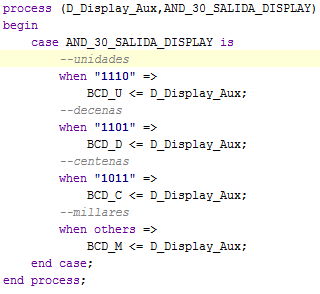


Ilustración -Proceso en VHDL para obtener valores BCD de los displays

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

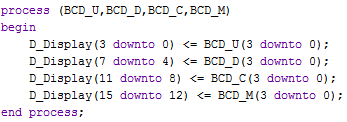


Ilustración -Proceso en VHDL para obtener el valor en BCD

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.

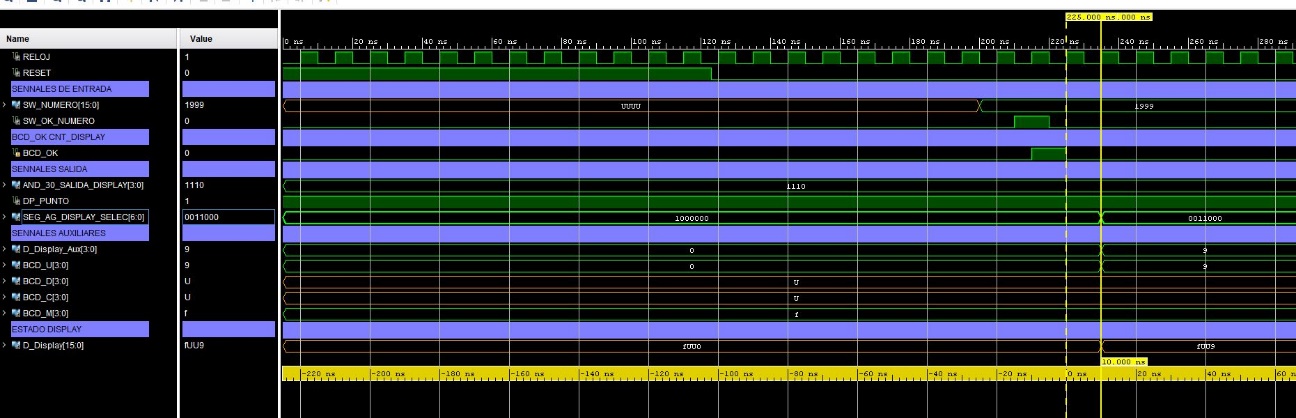
La recepción de los nuevos valores, acorde a la llegada de señal activa de BCD\_OK, y su posterior muestreo en los displays, se comprueba en las siguientes dos imágenes.

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.

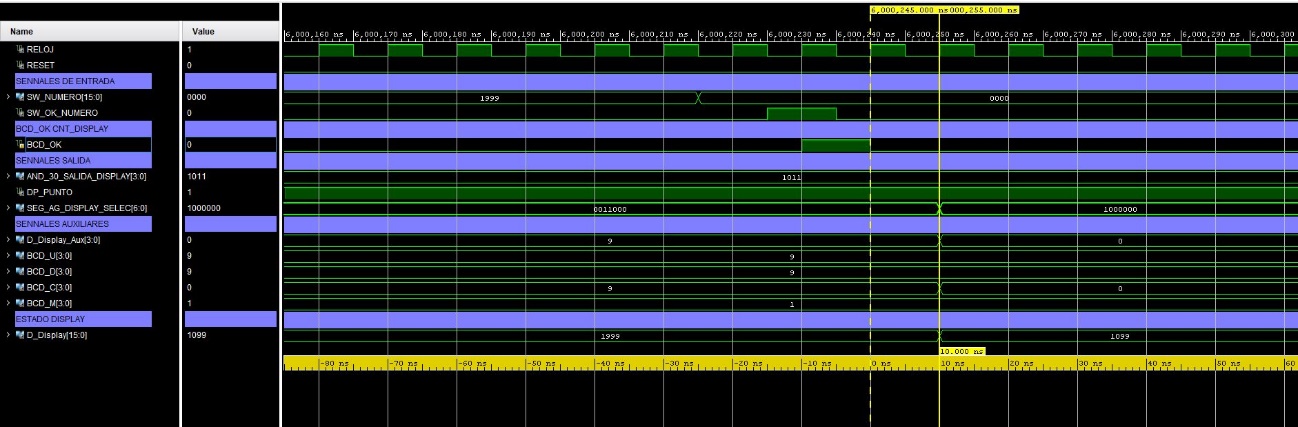


Ilustración -Segundo estimulo “0000”

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.

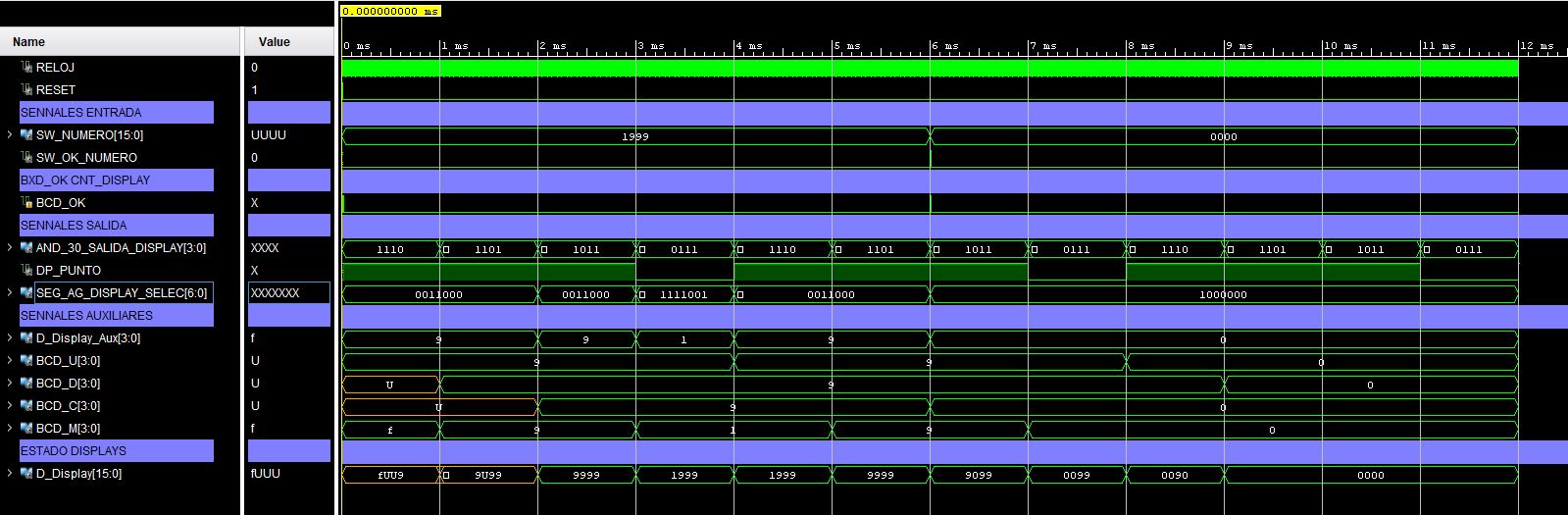
En la siguiente ilustración podemos observar todas las señales del sistema las cuales se encuentran explicadas en el apartado de la simulación funcional.

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.

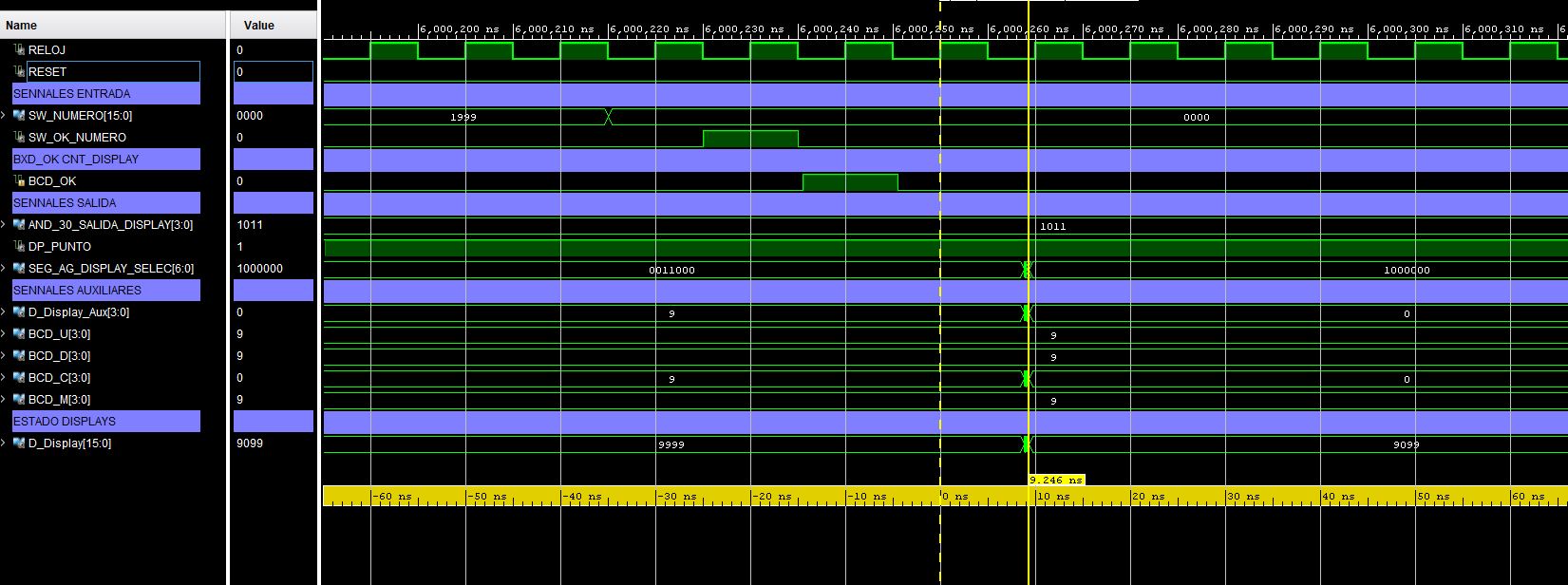
En la siguiente imagen observamos el segundo estimulo el cambio entre “1999” y “0000" y al igual que con el primer estimulo observamos un retardo entre el flanco de subida en el cual se debería haber producido el cambio en dicho valor BCD y el momento en el que ocurre, aunque esta vez dicho retardo es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

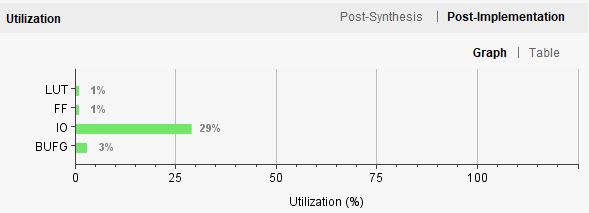


Ilustración - Grafica de utilización

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).

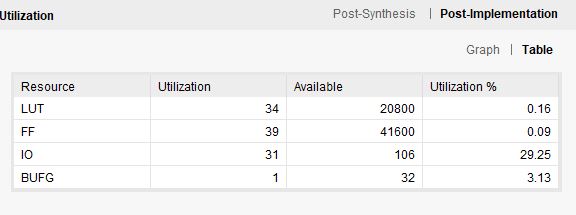


Ilustración -Tabla de utilización

## CONCLUSIÓN

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

La primera de ellas sería 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.

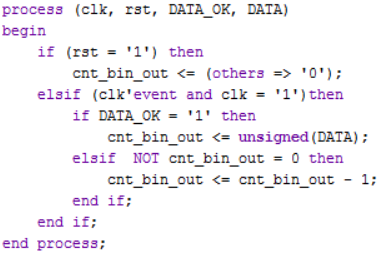


Ilustración -Implementación en VHDL del contador binario descendente de 12bits

* 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.

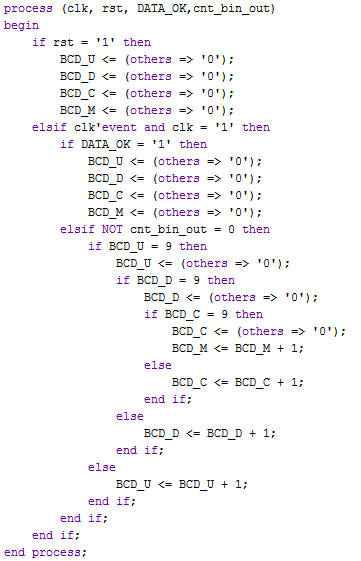


Ilustración -Implementación en VHDL del contador BCD ascendente de 4 dígitos

* 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).

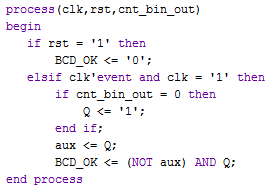


Ilustración -Implementación en VHDL del generador de BCD\_OK

* 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).

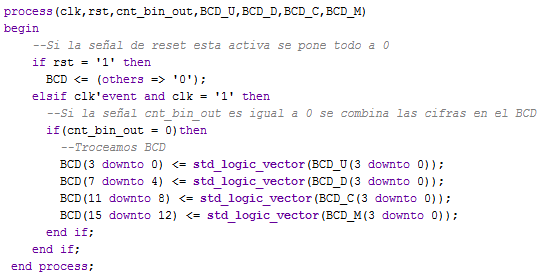
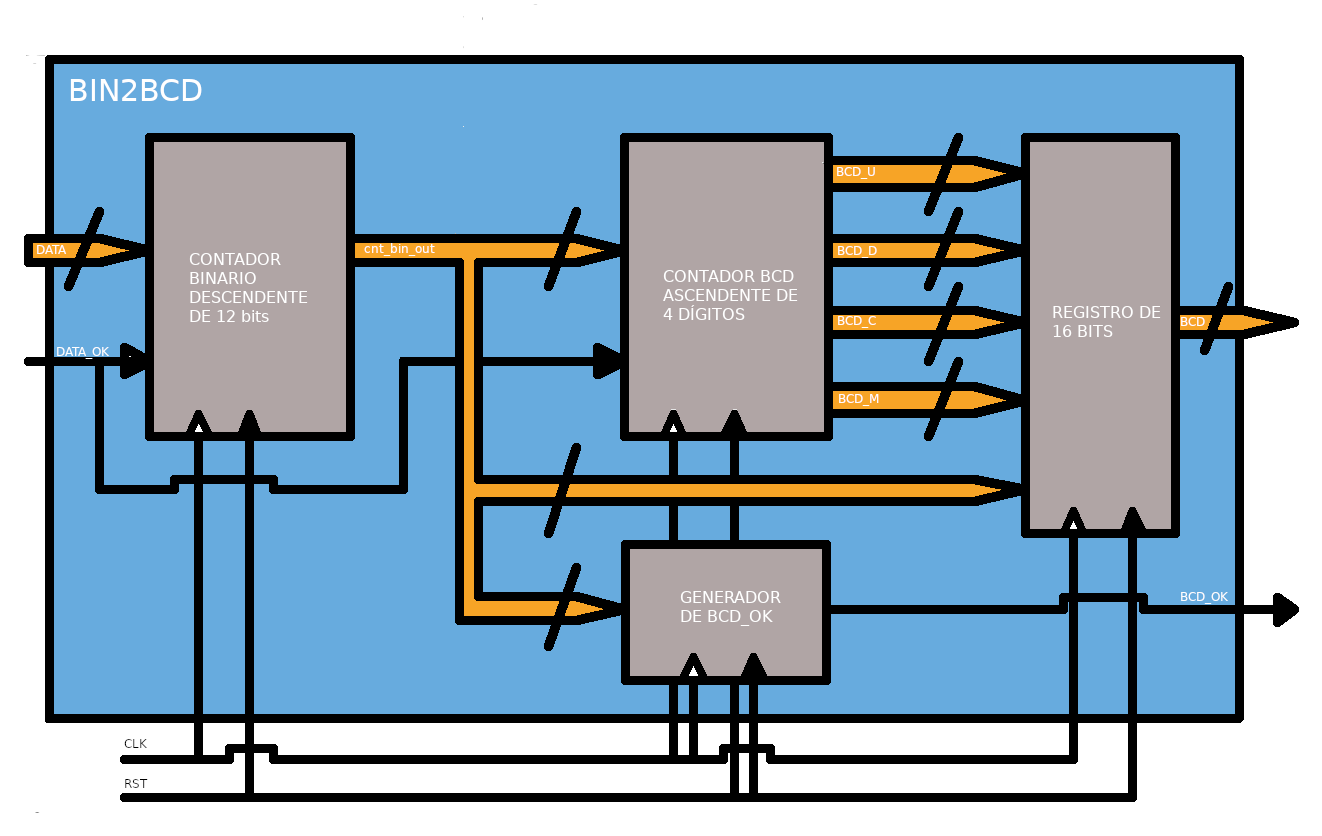


Ilustración -Implementación en VHDL del registro de 16bits

Ilustración -Implementación en VHDL del registro de 16 bits

11

10

9

8

7

6

5

4

3

2

1

Ilustración -Esquema completo de la estructura de la entidad bin2bcd con las señales utilizadas

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

|  |  |  |  |
| --- | --- | --- | --- |
| Número | Señal | Número | Señal |
| 1 | DATA (12bits) | 7 | BCD\_M (4bits) |
| 2 | DATA\_OK | 8 | BCD (16bits) |
| 3 | cnt\_bin\_out (12bits) | 9 | BCD\_OK |
| 4 | BCD\_U (4bits) | 10 | CLK |
| 5 | BCD\_D (4bits) | 11 | RST |
| 6 | BCD\_C (4bits) | - | - |

## 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:

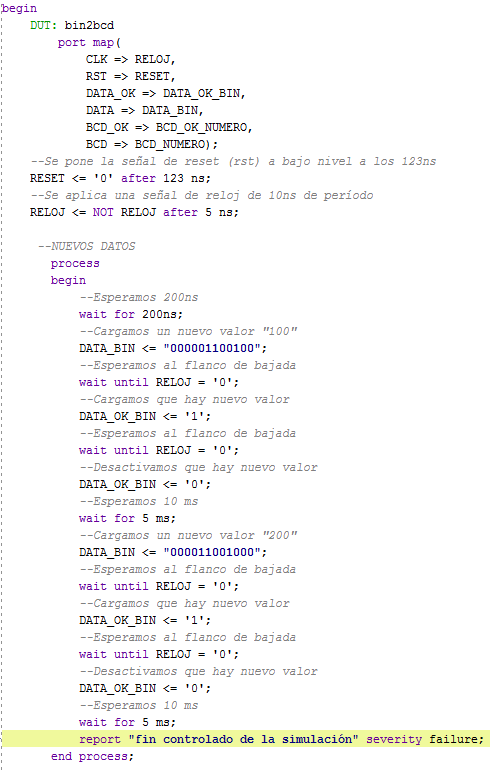


Ilustración -Testbench en VHDL para la simulación funcional del módulo

Antes de introducirnos en el análisis de las simulaciones, vamos a explicar en una tabla la correspondencia de las señales simuladas con las reales:

|  |  |  |
| --- | --- | --- |
| Señal simulada | Señal | Descripción |
| RELOJ | CLK | Señal de reloj que domina los tiempos de ejecución de los componentes de bin2bcd |
| RESET | RST | Señal para reiniciar a los valores iniciales a las señales mediante los componentes |
| DATA\_BIN | DATA | Señal que porta el valor en binario obtenido en el anterior módulo a convertir a formato bcd |
| DATA\_OK\_BIN | DATA\_OK | Señal que notifica la disposición de un nuevo valor en DATA para iniciar la conversión |
| cnt\_bin\_out | cnt\_bin\_out | Señal interna para notificar a los componentes el valor del contador binario, sobre todo para conocer cuando este llega a 0 |
| BCD\_U | BCD\_U | Señal que porta el valor de las unidades del valor en conversión |
| BCD\_D | BCD\_D | Señal que porta el valor de las decenas del valor en conversión |
| BCD\_C | BCD\_C | Señal que porta el valor de las centenas del valor en conversión |
| BCD\_M | BCD\_M | Señal que porta el valor de las unidades de millar del valor en conversión |
| BCD\_NUMERO | BCD | Señal que porta el valor del conjunto de las señales BCD antes mencionadas para su posterior representación en el display |
| BCD\_OK\_NUMERO | BCD\_OK | Señal que notifica al siguiente módulo de la disposición de un nuevo valor en BCD |

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.

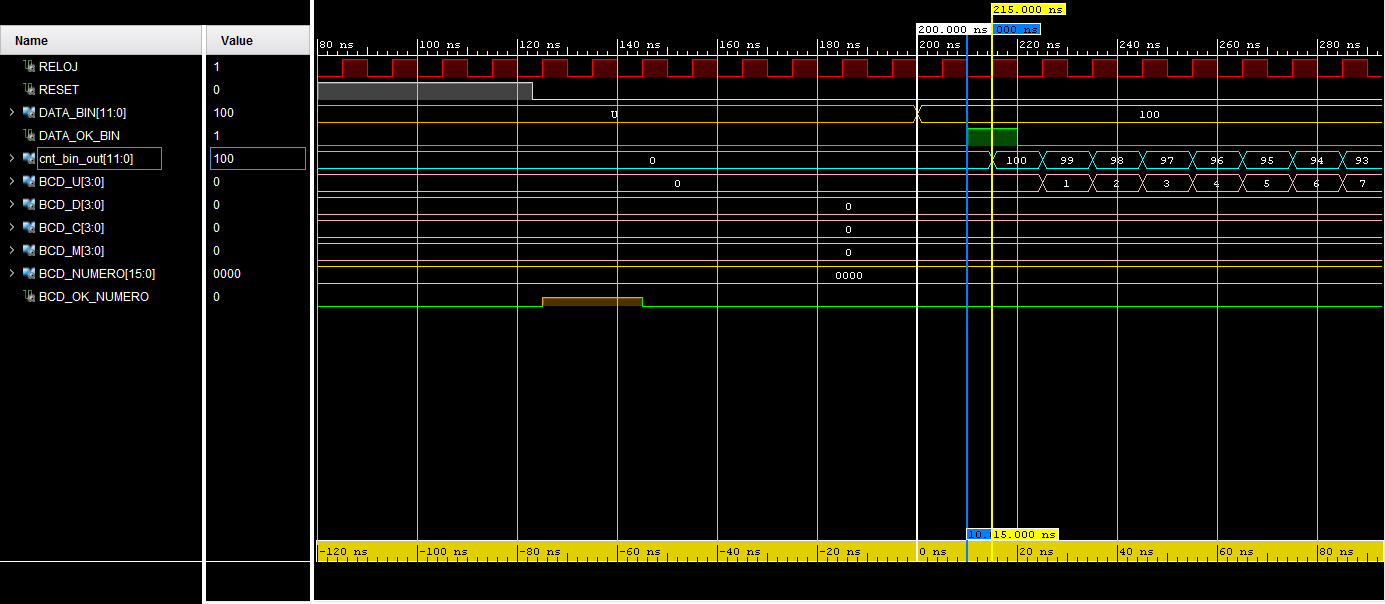


Ilustración 26-Cronograma de inicio de la simulación con un nuevo valor de DATA igual a 100

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.

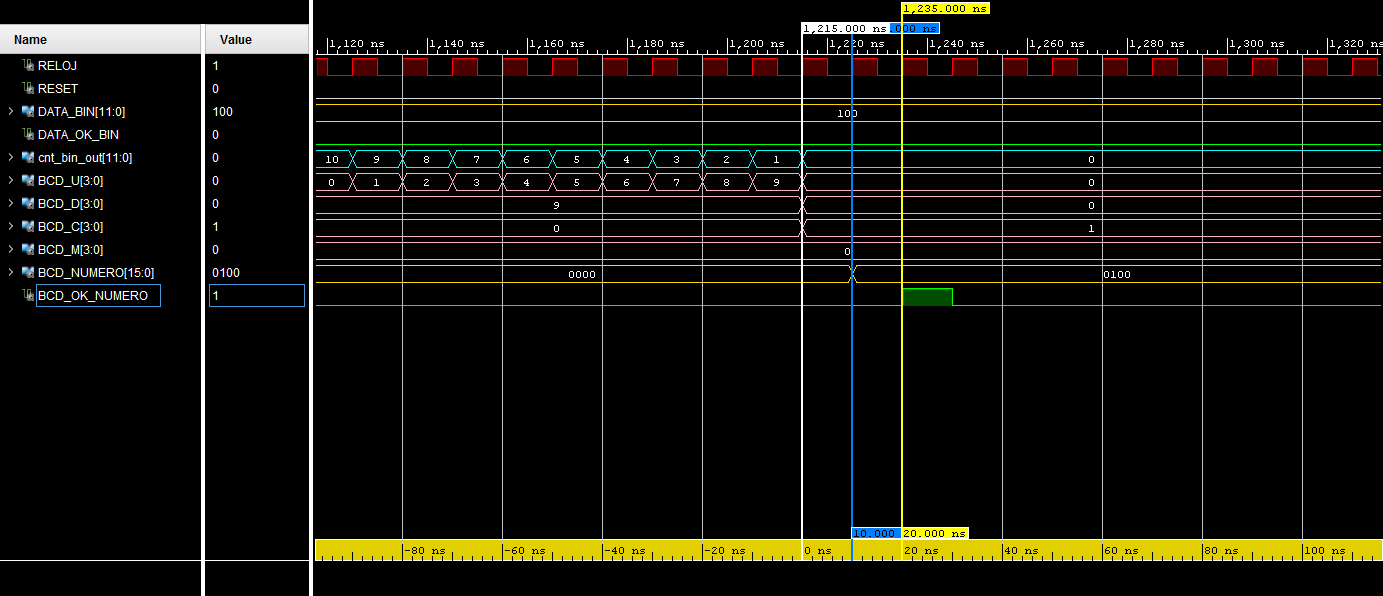


Ilustración -Cronograma del final de la conversión del valor DATA igual a 100 a BCD

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.

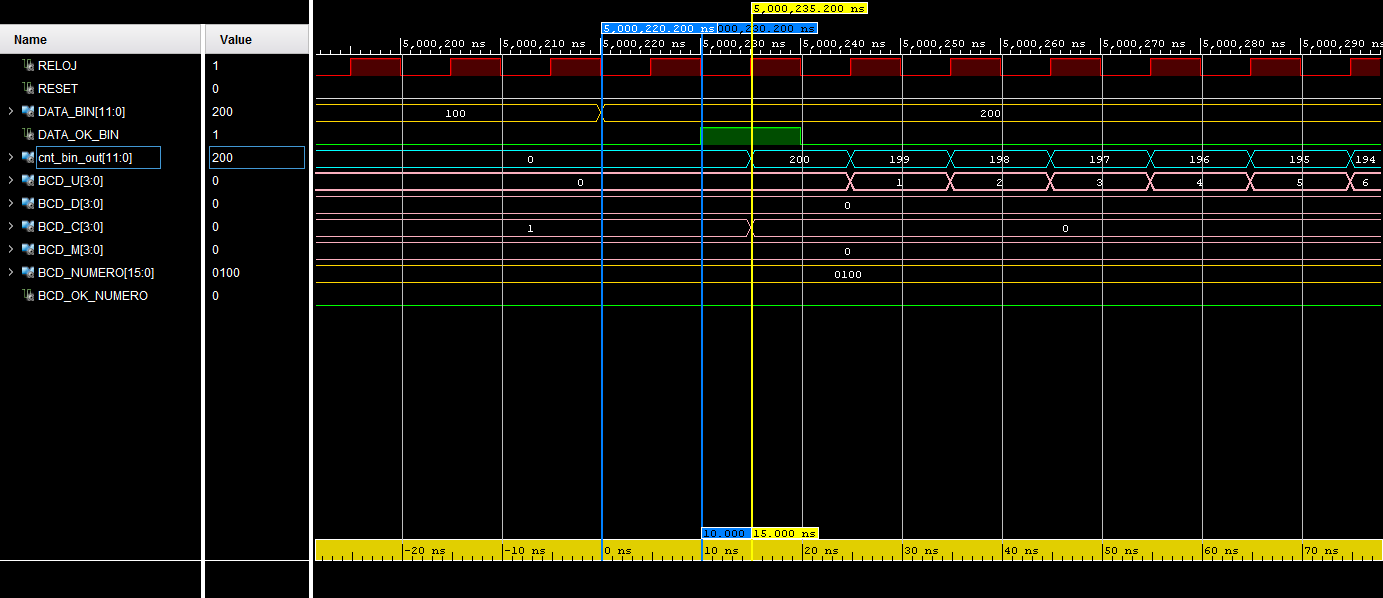


Ilustración -Cronograma de variación del valor DATA de 100 a 200, iniciándose nueva conversión

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 cual se muestra en el anterior apartado en la ilustración 23.

Antes de introducirnos en el análisis de las simulaciones, vamos a explicar en una tabla la correspondencia de las señales simuladas con las reales:

|  |  |  |
| --- | --- | --- |
| Señal simulada | Señal | Descripción |
| RELOJ | CLK | Señal de reloj que domina los tiempos de ejecución de los componentes de bin2bcd |
| RESET | RST | Señal para reiniciar a los valores iniciales a las señales mediante los componentes |
| DATA\_BIN | DATA | Señal que porta el valor en binario obtenido en el anterior módulo a convertir a formato bcd |
| DATA\_OK\_BIN | DATA\_OK | Señal que notifica la disposición de un nuevo valor en DATA para iniciar la conversión |
| cnt\_bin\_out | cnt\_bin\_out | Señal interna para notificar a los componentes el valor del contador binario, sobre todo para conocer cuando este llega a 0 |
| BCD\_NUMERO | BCD | Señal que porta el valor del conjunto de las señales BCD representantes de cada dígito para su posterior representación en el display |
| BCD\_OK\_NUMERO | BCD\_OK | Señal que notifica al siguiente módulo de la disposición de un nuevo valor en BCD |

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.

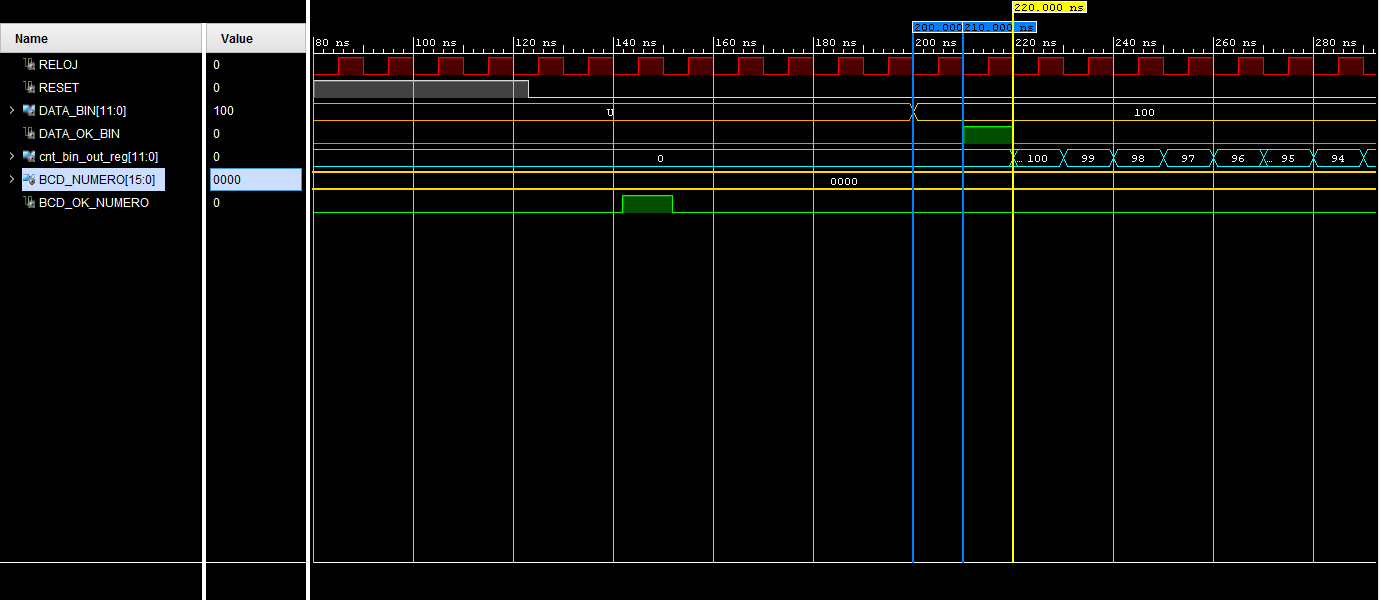


Ilustración -Cronograma de inicio de la simulación con un nuevo valor de DATA igual a 100

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.

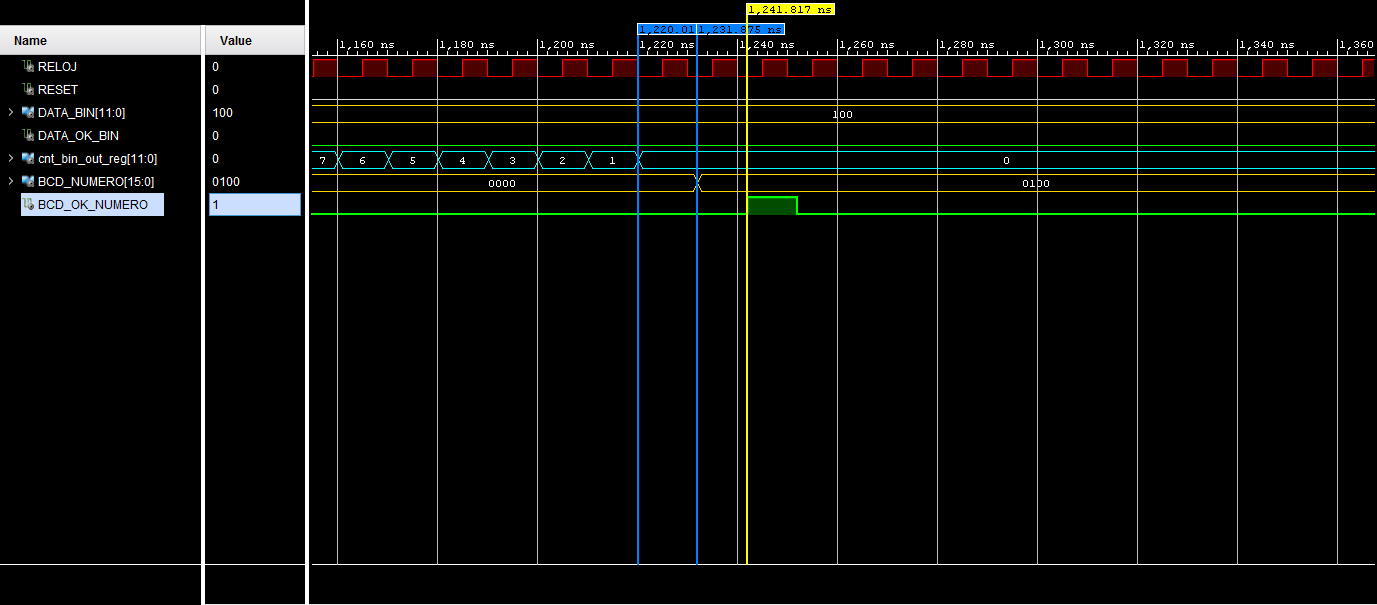


Ilustración -Cronograma del final de la conversión del valor DATA igual a 100 a BCD

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.

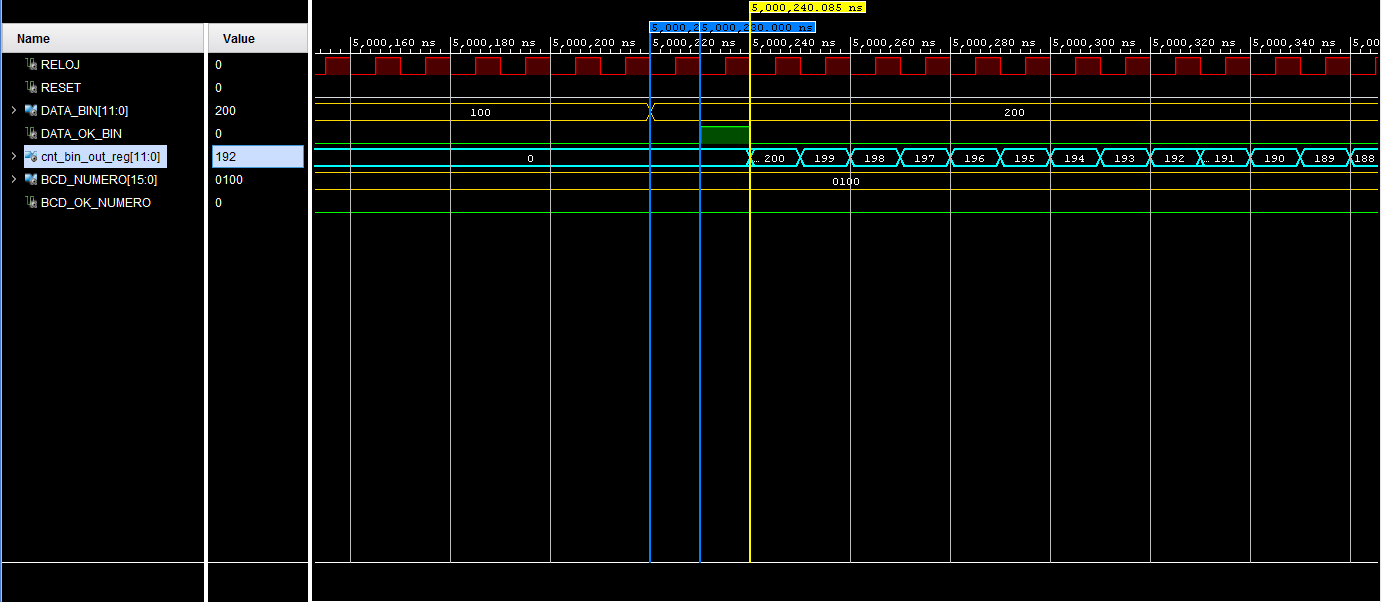


Ilustración -Cronograma de variación del valor DATA de 100 a 200, iniciándose nueva conversión

Finalmente se llegará al final de la conversión cuando el contador binario llega a 0, observándose el mismo comportamiento que cuando se finalizó 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.

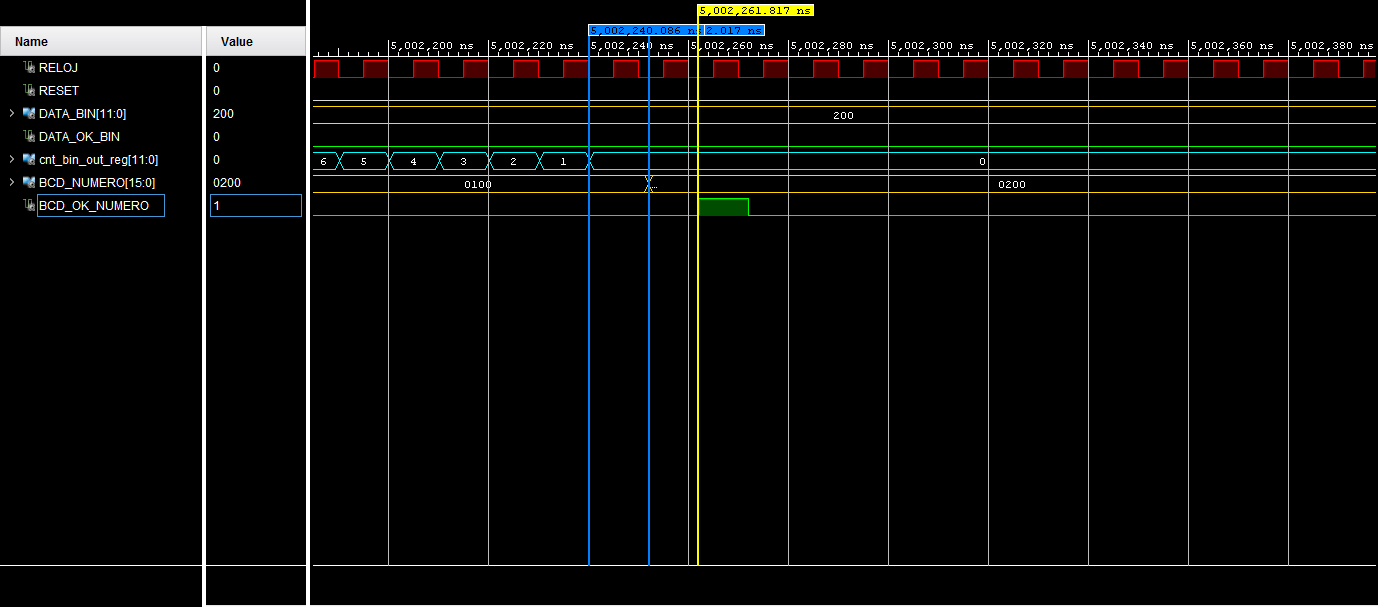


Ilustración -Cronograma del final de la conversión del valor de DATA igual a 200 a BCD

## CONCLUSIÓN

Finalmente, tras acabar las simulaciones de manera satisfactoria ya solo que consultar estadísticas y analizar los resultados obtenidos.

En la gráfica siguiente se puede comprobar la cantidad de recursos disponibles de cada tipo, siendo los recursos IO los más usados con un porcentaje igual a 38%.

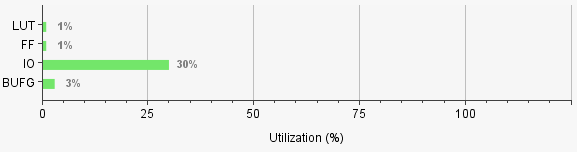


Ilustración -Gráfica de porcentaje de uso de los recursos

Siguiendo el estudio de la gráfica podemos comprobar los recursos empleados, su disponibilidad y utilización resultando en un uso mayoritario de los registros FF.

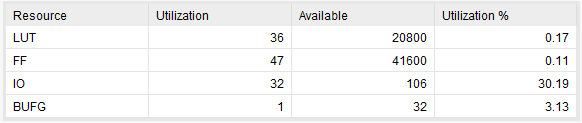


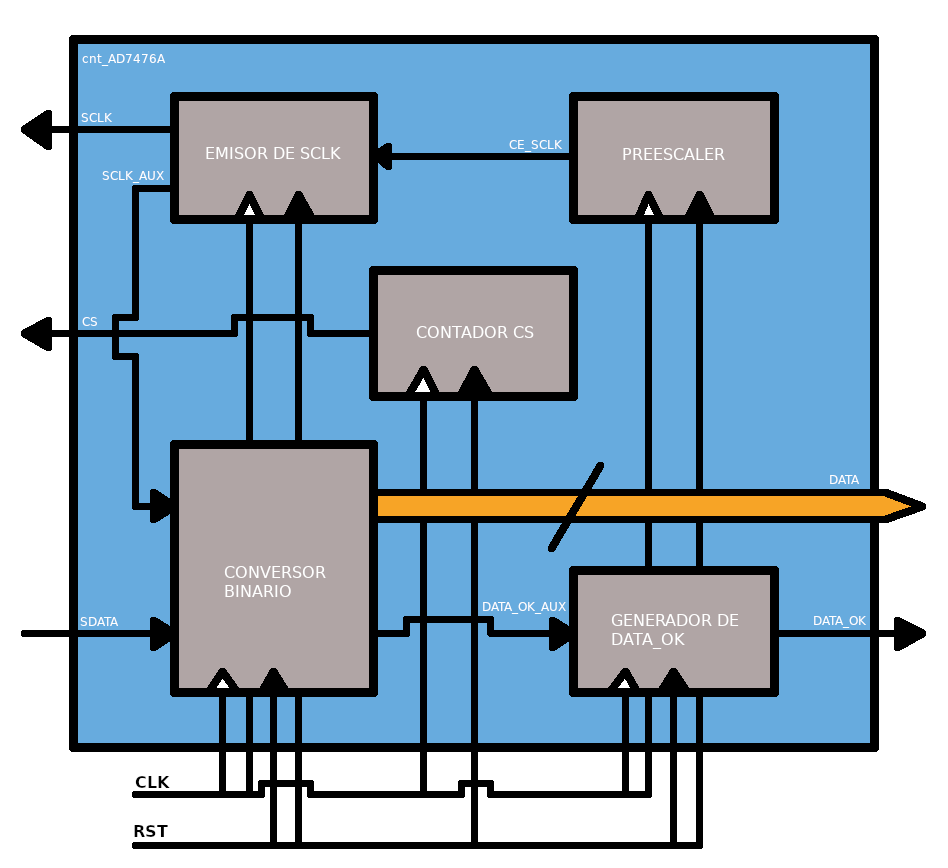
Ilustración -Tabla de información de los recursos

# 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 a cabo 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 a cabo 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).



10

9

8

7

6

5

4

3

1

2

Ilustración -Esquema del diseño del módulo cnt\_AD7476A

A continuación, detallamos el conjunto de señales que aparecen en el esquema:

|  |  |  |  |
| --- | --- | --- | --- |
| Número | Señal | Número | Señal |
| 1 | CE\_SCLK | 6 | DATA (12bits) |
| 2 | SCLK | 7 | DATA\_OK\_AUX |
| 3 | SCLK\_AUX | 8 | DATA\_OK |
| 4 | CS | 9 | CLK |
| 5 | SDATA | 10 | RST |

## 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.

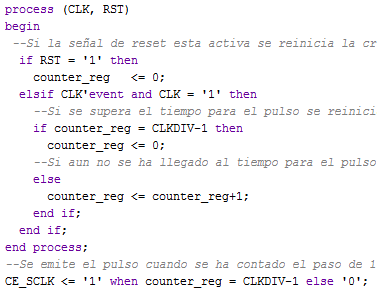


Ilustración -Implementación en VHDL del preescaler para SCLK

* 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.

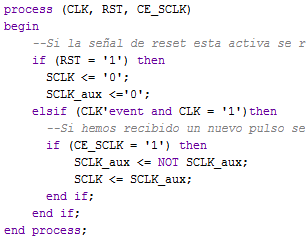


Ilustración -Implementación en VHDL del emisor de SCLK

* Contador CS:

Este componente se encarga de manejar la señal encargada de indicar los tiempos en el que se está llevando a cabo el muestreo y cuando no (CS). La señal CS a nivel bajo significa que se está realizando el muestreo, añadiéndose a este intervalo un período de SCLK. Esta última señal es dominada por un contador que comprueba el equivalente en ciclos de reloj de todo el tiempo de muestreo (CS\_L\_counter) y por la señal que indica el periodo que la señal se mantiene en nivel alto, tiempo que no se realiza muestreo (CS\_H\_counter).

En caso de que llegue la señal activa de reset (RST) se procede a poner la señal CS en nivel alto para iniciar a continuación un nuevo muestreo.

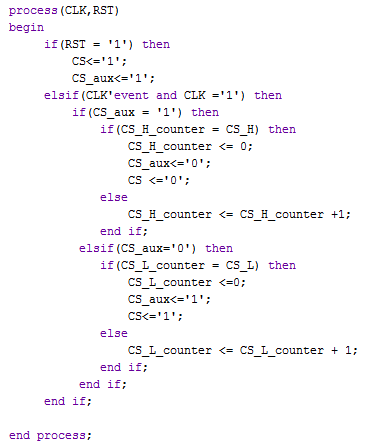


Ilustración -Implementación en VHDL del contador de CS

* 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). Para conocer el momento de final del muestreo se usa el contador de ciclos de SCLK que toma muestrear un dato (desp\_counter) para ir desplazando los valores llegados en cada ciclo y obtener el valor muestreado completo.

En caso de que llegue la señal activa de reset (RST) se procede a poner el contador de ciclos de muestreo a 0 (desp\_counter) y la señal que indica que hay un nuevo valor también se reinicia a estado bajo (DATA\_OK).

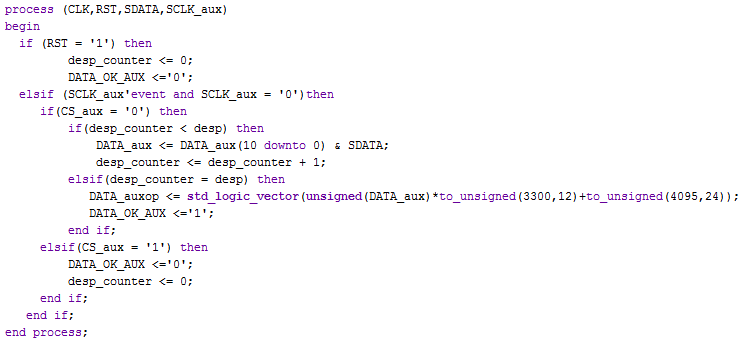


Ilustración -Implementación en VHDL del conversor a binario

* Generador DATA\_OK:

Este componente a partir de las señal de entrada de final de muestreo generado por el conversor a binario (DATA\_OK\_AUX) se encarga de emitir un pulso en la señal DATA\_OK para notificar al siguiente módulo la existencia de un nuevo valor en DATA a convertir en formato bcd.

EN caso de que llegue la señal activa de reset (RST) se procede poner en estado bajo la señal DATA\_OK para no notificar variaciones en DATA.

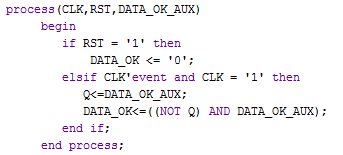


Ilustración -Implementación en VHDL del generador de DATA\_OK

## SIMULACIÓN FUNCIONAL

Para comprobar el funcionamiento del módulo desarrollado implementamos un testbench para verificar que el módulo recién escrito cumpla la funcionalidad por la cual se ha diseñado.

Por tanto, el banco de pruebas consta de una duración de 15ms en el cuál después de asignar la frecuencia del reloj (CLK) y la primera señal de reset durante los primeros 123ns, como en los anteriores testbench de los módulos anteriormente implementados. En la prueba consiste en la sencilla tarea de asignar a la entrada del muestreador la señal de 2.5V para comprobar por la salida del módulo si obtenemos el valor correspondiente en binario.

Con esto comprobamos la funcionalidad básica de la obtención en valor binario del valor del voltaje suministrado por entrada del muestreador, el cual se procesará a través de los módulos hasta su representación en el display.

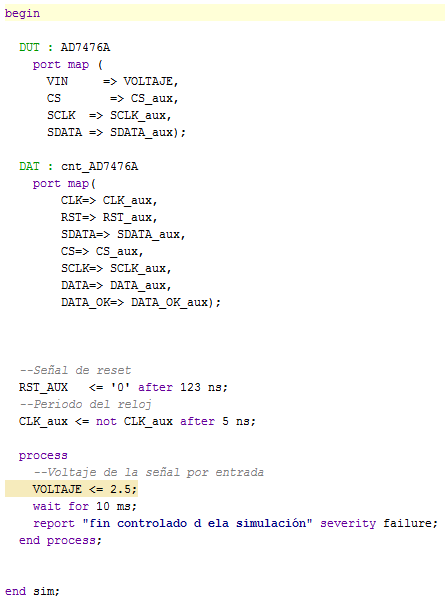


Ilustración -Testbench en VHDL para la simulación funcional del módulo

Antes de introducirnos en el análisis de las simulaciones, vamos a explicar en una tabla la correspondencia de las señales simuladas con las reales:

|  |  |  |
| --- | --- | --- |
| Señal simulada | Señal | Descripción |
| CLK\_aux | CLK | Señal de reloj que domina los tiempos de ejecución de los componentes de bin2bcd |
| RST\_aux | RST | Señal para reiniciar a los valores iniciales a las señales mediante los componentes |
| VOLTAJE | Vin | Señal de voltaje que se introduce mediante la entrada del ADC para su muestreo |
| SDATA\_aux | SDATA | Valores muestreados periódicamente en función de SCLK de la señal VOLTAJE |
| CS\_aux | CS | Señal que domina los tiempos de muestreo y de parada entre un muestreo y el siguiente |
| SCLK\_aux | SCLK | Señal de reloj que domina los tiempos de ejecución del ADC |
| DATA\_aux | DATA | Señal que porta el valor en binario muestreado a convertir a formato bcd |
| DATA\_OK\_aux | DATA\_OK | Señal que notifica la disposición de un nuevo valor en DATA para iniciar la conversión |

Para iniciar la simulación se empieza por declarar el periodo del reloj (CLK) y una primera señal de reset de 123ns de duración (RST), para iniciar el módulo desde valores iniciales. Comprobándose la carga del valor de 2.5V en el voltaje que pasa por el ADC y tras el cambio de CS a nivel bajo, significa que ha empezado el tiempo de muestreo, observándose tras 20ns la llegada del primer valor del muestreo que acaba de empezar (SDATA)

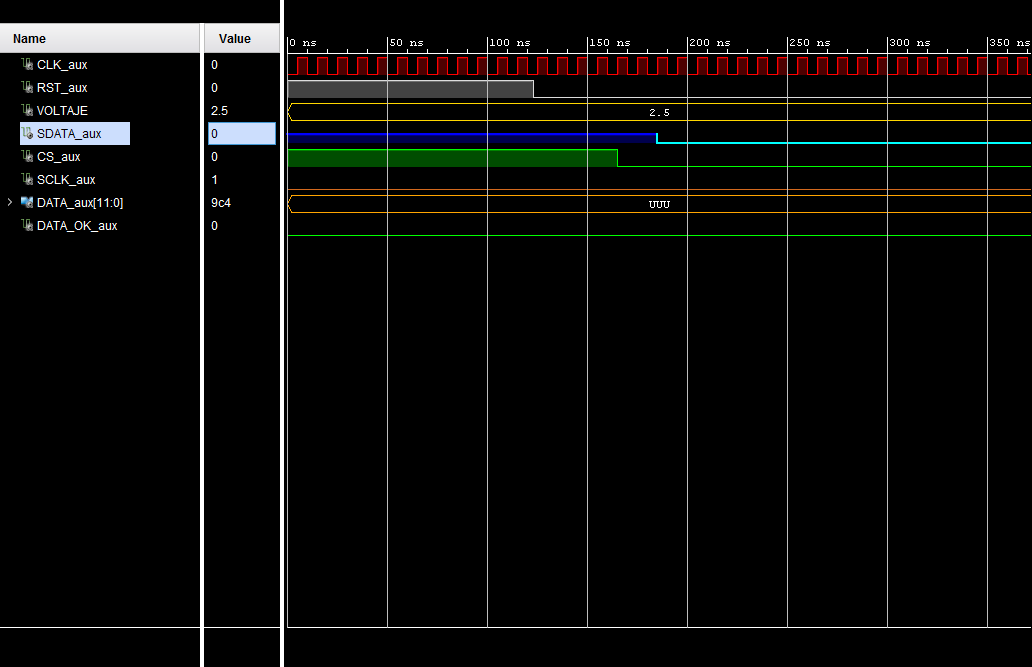


Ilustración -Cronograma de inicio de la simulación con voltaje igual a 2.5V

Tras la obtención de valores a través del muestreo, este llega a su fin como se puede observar en SDATA con un color azul oscuro, debido a que el tiempo de muestreo se le añade un ciclo extra del reloj SCLK. Significando que ya tenemos el dato muestreado por lo que se procede a modificar el valor de salida de DATA con el valor ensamblado a través de los desplazamientos de la llegada de valores de SDATA, procediendo de manera consecutiva en la emisión del pulso de DATA\_OK notificando al siguiente módulo la variación de un nuevo valor de DATA.

Además, tras el periodo extra que se le añade al tiempo de muestreo se puede observar la generación de un pulso en la señal CS con el tiempo prefijado entre muestreo y muestreo, para dar comienzo cuando vuelva a nivel bajo de un nuevo muestreo.

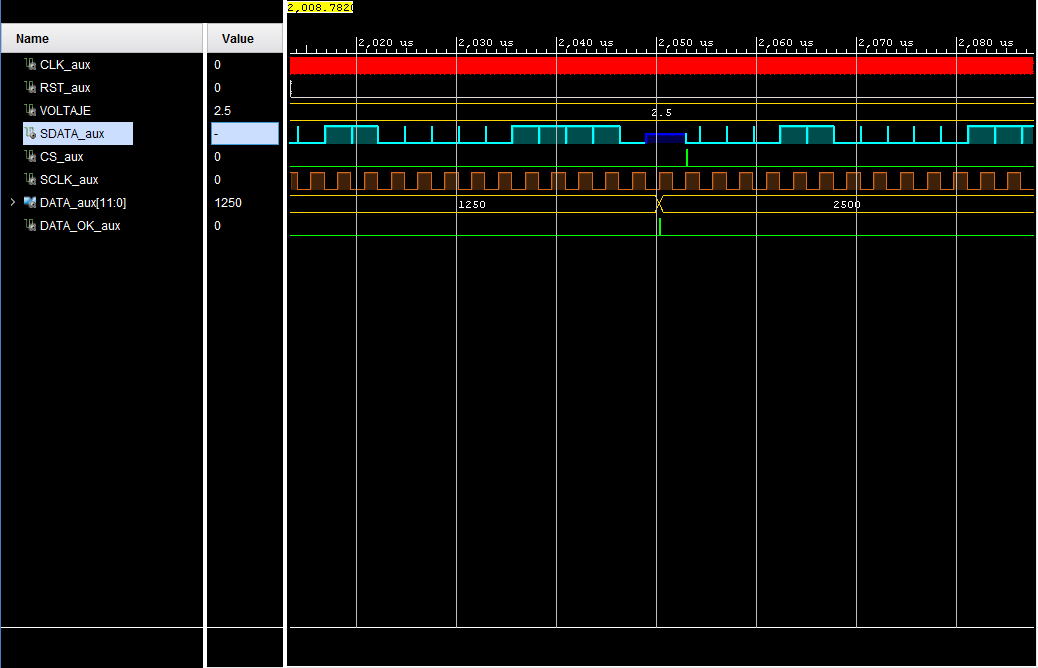


Ilustración -Cronograma de momento de estabilización de DATA tras el muestreo

## SIMULACIÓN TEMPORAL

Tras comprobar que la simulación funcional presenta la funcionalidad que esperamos, procedemos a determinar el correcto funcionamiento final mediante una simulación temporal. Para llevarla a cabo procedemos a reutilizar el testbench antes implementado para la simulación funcional, este se puede comprobar en la ilustración 38 de la simulación funcional de este módulo.

Antes de introducirnos en el análisis de las simulaciones, vamos a explicar en una tabla la correspondencia de las señales simuladas con las reales:

|  |  |  |
| --- | --- | --- |
| Señal simulada | Señal | Descripción |
| CLK\_aux | CLK | Señal de reloj que domina los tiempos de ejecución de los componentes de bin2bcd |
| RST\_aux | RST | Señal para reiniciar a los valores iniciales a las señales mediante los componentes |
| VOLTAJE | Vin | Señal de voltaje que se introduce mediante la entrada del ADC para su muestreo |
| SDATA\_aux | SDATA | Valores muestreados periódicamente en función de SCLK de la señal VOLTAJE |
| CS\_aux | CS | Señal que domina los tiempos de muestreo y de parada entre un muestreo y el siguiente |
| SCLK\_aux | SCLK | Señal de reloj que domina los tiempos de ejecución del ADC |
| DATA\_aux | DATA | Señal que porta el valor en binario muestreado a convertir a formato bcd |
| DATA\_OK\_aux | DATA\_OK | Señal que notifica la disposición de un nuevo valor en DATA para iniciar la conversión |

Para iniciar la simulación se empieza por declarar el periodo del reloj (CLK) y una primera señal de reset de 123ns de duración (RST), para iniciar el módulo desde valores iniciales. Comprobándose la carga del valor de 2.5V en el voltaje que pasa por el ADC y tras el cambio de CS a nivel bajo, significa que ha empezado el tiempo de muestreo, observándose tras 20ns la llegada del primer valor del muestreo que acaba de empezar (SDATA)

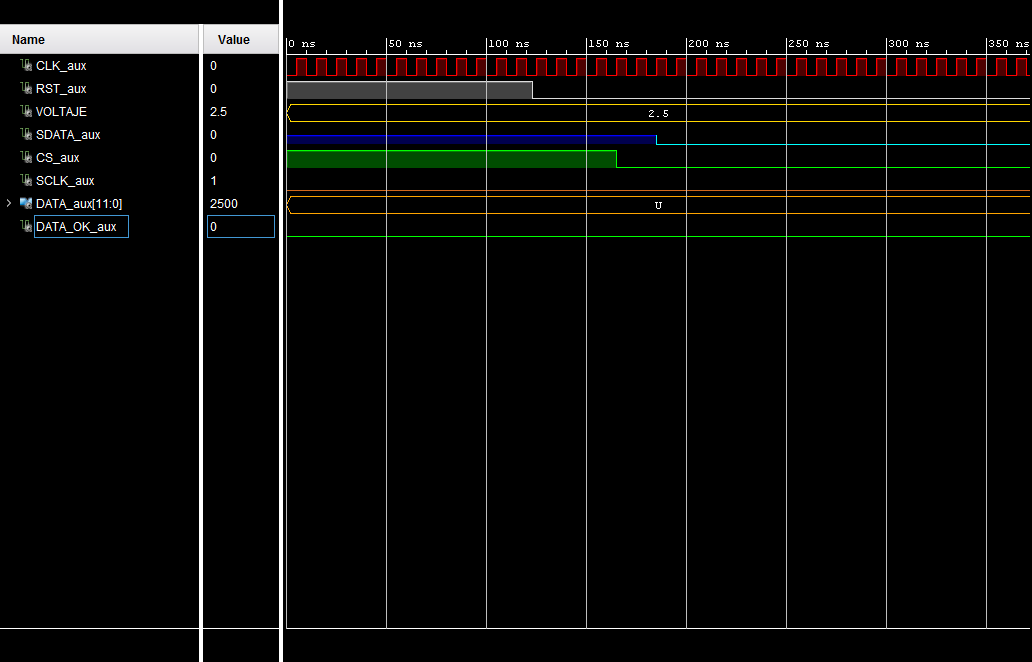


Ilustración -Cronograma de inicio de la simulación con voltaje igual a 2.5V

Tras la obtención de valores a través del muestreo, este llega a su fin como se puede observar en SDATA con un color azul oscuro, debido a que el tiempo de muestreo se le añade un ciclo extra del reloj SCLK. Significando que ya tenemos el dato muestreado por lo que se procede a modificar el valor de salida de DATA con el valor ensamblado a través de los desplazamientos de la llegada de valores de SDATA, procediendo de manera consecutiva en la emisión del pulso de DATA\_OK notificando al siguiente módulo la variación de un nuevo valor de DATA.

Además, tras el periodo extra que se le añade al tiempo de muestreo se puede observar la generación de un pulso en la señal CS con el tiempo prefijado entre muestreo y muestreo, para dar comienzo cuando vuelva a nivel bajo de un nuevo muestreo.



Ilustración -Cronograma de momento de estabilización de DATA tras el muestreo

## CONCLUSIÓN

Una vez acabada las simulaciones llega el momento de concluir este apartado con la muestra de las estadísticas y datos de relevancia que requiere el módulo para su correcto proceder.

Primero comprobamos la gráfica que genera indicando los porcentajes de utilización de recursos por parte del módulo, destacando sobre los demás el uso de los recursos IO con un porcentaje de un 17%.

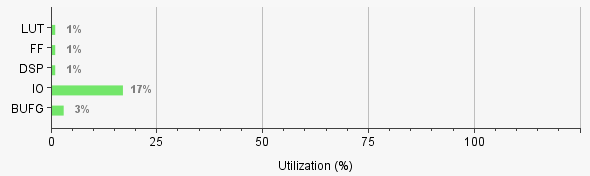


Ilustración -Gráfica de porcentaje de uso de los recursos

A continuación, a partir de la anterior gráfica observamos la tabla que enumera los recursos, su utilización y su disponibilidad, marcando como recurso predominante en el módulo el de los registros FF.

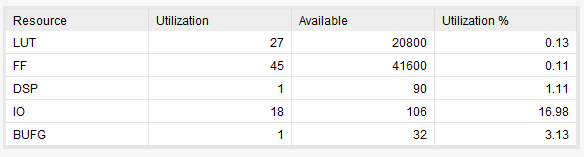


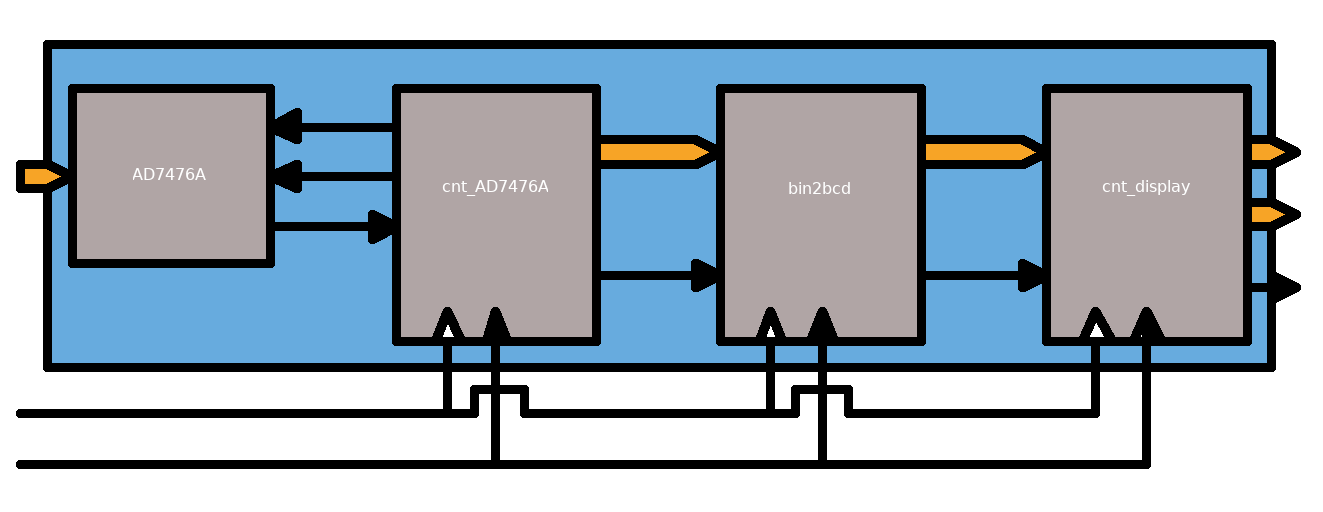
Ilustración -Tabla de información de los recursos

# VOLTIMETRO

## DISEÑO

Tras la implementación de los tres módulos requeridos, ya lo último que queda por hacer es ensamblar todos además del ADC a través de las señales de salida y entrada de los módulos, resultando en que las señales SDATA,SCLK,CS,DATA, DATA\_OK, BCD y BCD\_OK quedarán ocultas dentro del cuerpo del diseño solo siendo visibles la entrada de voltaje al ADC y la salida de la información del valor a representar por el display (DP, AND\_30 y SEG\_AG).

Con ello obtenemos la funcionalidad total de introducir una señal portadora de un voltaje en la entrada del sistema creado, valor que obtendremos tras su paso por el ADC para después ajustar el valor a formatos a través de su paso por los módulos hasta ajustarlo para la representación en el display.



13

12

11

10

9

8

7

6

5

4

3

2

1

Ilustración -Esquema del diseño del voltímetro

A continuación, se enumerarán las señales enumeradas:

|  |  |  |  |
| --- | --- | --- | --- |
| Número | Señal | Número | Señal |
| 1 | Vin | 8 | BCD\_OK |
| 2 | SCLK | 9 | SEG\_AG (7bits) |
| 3 | CS | 10 | AND\_30 (4bits) |
| 4 | SDATA | 11 | DP |
| 5 | DATA (12bits) | 12 | CLK |
| 6 | DATA\_OK | 13 | RST |
| 7 | BCD (16bits) | - | - |

## SIMULACIÓN FUNCIONAL

Anteriormente hemos ido comprobando el correcto funcionamiento en los módulos creados según los íbamos diseñando, aunque el resultado a sido correcto, no implica que ahora en conjunto funcionen de manera correcta. Por tanto, ahora debemos diseñar una nueva testbench para probar que todos los módulos acoplados en el mismo sistema funcionen correctamente de acuerdo con sus funcionalidades individuales que ofrecen. El diseño del banco de pruebas es el siguiente:

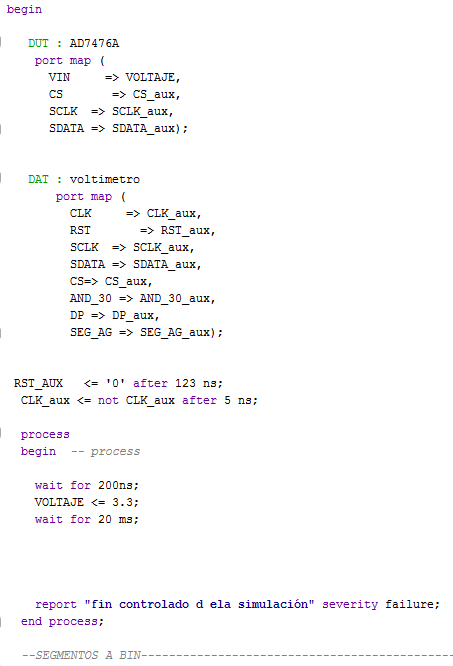


Ilustración -Testbench en VHDL para la simulación funcional del sistema

Debido a que hay señales que nos interesa comprobar en las simulaciones añadimos a este testbench procesos extra para comprobar datos del display:

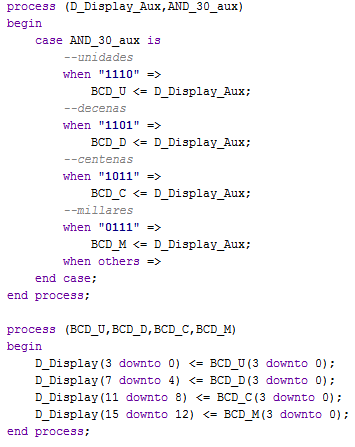


Ilustración -Implementación en VHDL de procesos auxiliares para la visualización de salida del display

Antes de introducirnos en el análisis de las simulaciones, vamos a explicar en una tabla la correspondencia de las señales simuladas con las reales:

|  |  |  |
| --- | --- | --- |
| Señal simulada | Señal | Descripción |
| CLK\_aux | CLK | Señal de reloj que domina los tiempos de ejecución de los componentes de bin2bcd |
| RST\_aux | RST | Señal para reiniciar a los valores iniciales a las señales mediante los componentes |
| cnt\_AD7476A | | |
| VOLTAJE | Vin | Señal de voltaje que se introduce mediante la entrada del ADC para su muestreo |
| SDATA\_aux | SDATA | Valores muestreados periódicamente en función de SCLK de la señal VOLTAJE |
| CS\_aux | CS | Señal que domina los tiempos de muestreo y de parada entre un muestreo y el siguiente |
| SCLK\_aux | SCLK | Señal de reloj que domina los tiempos de ejecución del ADC |
| bin2bcd | | |
| DATA | DATA | Señal que porta el valor en binario muestreado a convertir a formato bcd |
| DATA\_OK | DATA\_OK | Señal que notifica la disposición de un nuevo valor en DATA para iniciar la conversión |
| cnt\_display | | |
| BCD | BCD | Señal que porta el valor del conjunto de las señales BCD representantes de cada dígito para su posterior representación en el display |
| BCD\_OK | BCD\_OK | Señal que notifica la existencia de nuevo valor en la señal BCD para iniciar su representación por display |
| AND\_30\_aux | AND\_30 | Señal que domina la asignación del display que estamos configurando |
| DP\_aux | DP | Señal que indica si encendemos el punto del display o no |
| SEG\_AG\_aux | SEG\_AG | Señal que contiene la información de los segmentos del display que se han de iluminar |
| display | | |
| D\_Display\_Aux | D\_Display\_Aux | REVISAR |
| BCD\_U | BCD\_U | Señal que porta el valor de las unidades del valor en cnt\_display |
| BCD\_D | BCD\_D | Señal que porta el valor de las decenas del valor en cnt\_display |
| BCD\_C | BCD\_C | Señal que porta el valor de las centenas del valor en cnt\_display |
| BCD\_M | BCD\_M | Señal que porta el valor de las unidades de millar del valor en cnt\_display |
| D\_Display | D\_Display | Señal que representa el valor mostrado en los 4 displays |

Comenzando la simulación iniciamos tras declarar el periodo del reloj (CLK) y una primera señal de reset de 123ns de duración (RST) para iniciar el sistema desde valores iniciales. Comprobándose que las primeras señales en variar desde el inicio, aparte de la introducción del voltaje para el ADC igual a 3.3V, son las relativas al último módulo diseñado de cnt\_AD7476A lo que significa que iniciará el muestreo del voltaje.

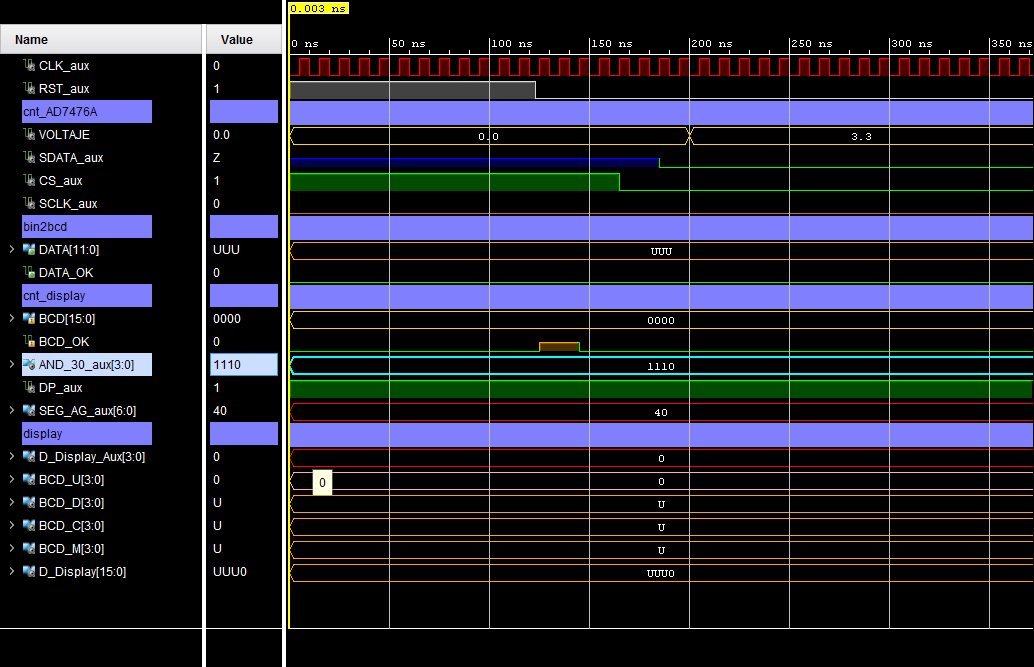


Ilustración -Cronograma de inicio de la simulación con voltaje igual a 3.3V

Tras finalizar el muestreo y la conversión del voltaje realizada en combinación con el ADC y el módulo cnt\_AD7476A, tenemos el valor del voltaje introducido convertido a formato binario el cual se indica en el cambio de valor DATA , para que a continuación se emita un pulso en DATA\_OK para notificar al módulo bin2bcd la variación de valor de DATA para iniciar la nueva conversión a formato bcd.

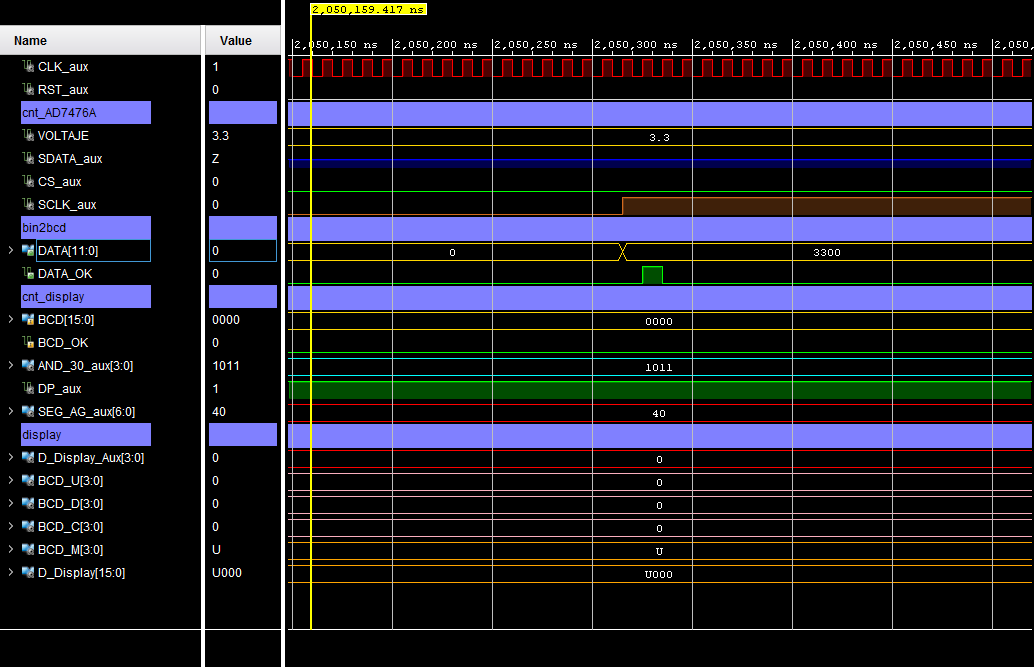


Ilustración -Cronograma de momento de estabilización de DATA tras el muestreo

En el momento que acaba la conversión del valor en formato binario a formato bcd, cuando el contador binario del módulo bin2bcd llega a cero, se modifica el valor de BCD para que en el periodo siguiente se emita su correspondiente pulso por BCD\_OK para notificar al módulo cnt\_display de la disponibilidad de un nuevo valor por BCD para mostrar por los displays.

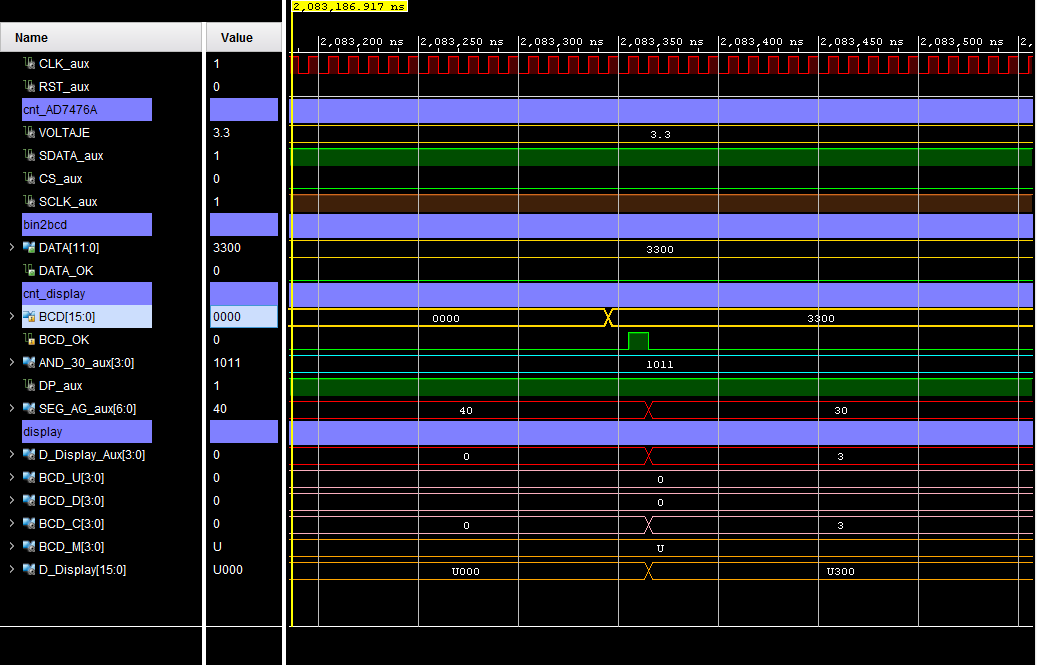


Ilustración -Cronograma de momento de estabilización de BCD tras la conversión de DATA

Como último momento destacable de la simulación representamos el momento en el que el último display se actualiza mediante la función que desempeña el módulo cnt\_display, mostrándose finalmente el valor completo del voltaje introducido en la entrada del sistema por los cuatro displays.

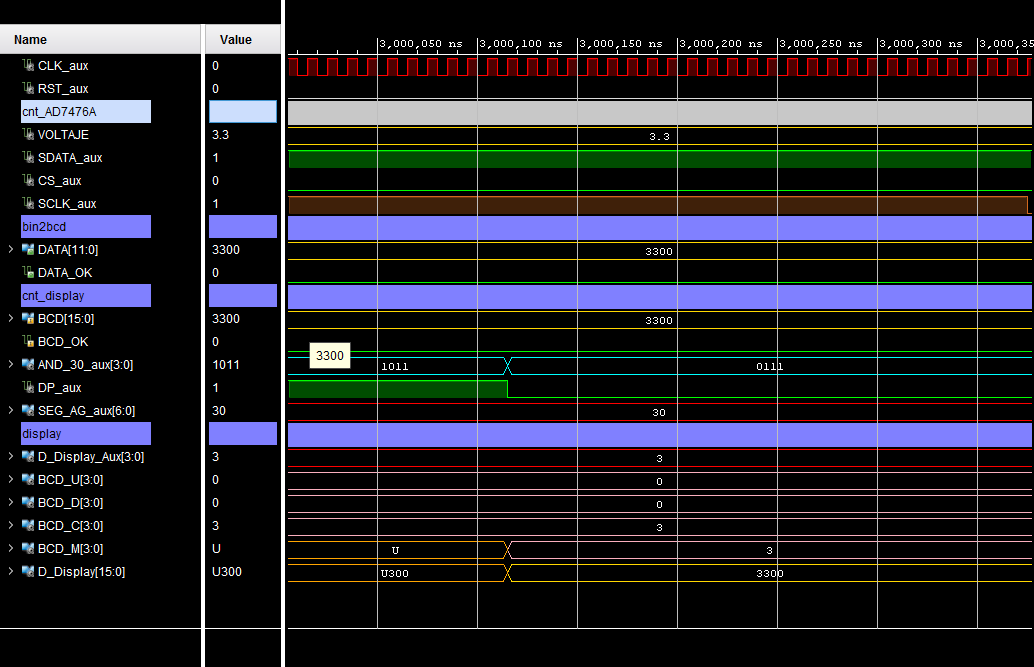


Ilustración -Cronograma de momento de estabilización de la salida del valor de voltaje por display

## SIMULACIÓN TEMPORAL

Una vez comprobado el correcto proceder del sistema en las simulaciones funcionales, llega el momento final de la última comprobación de la funcionalidad del sistema mediante el desarrollo de una simulación temporal. Para llevarla a cabo reutilizaremos el testbench diseñado en el apartado anterior para comprobar la eficiencia del sistema. La implementación del testbench se puede comprobar en la ilustración XX del apartado de la simulación funcional del sistema.

Antes de introducirnos en el análisis de las simulaciones, vamos a explicar en una tabla la correspondencia de las señales simuladas con las reales:

|  |  |  |
| --- | --- | --- |
| Señal simulada | Señal | Descripción |
| CLK\_aux | CLK | Señal de reloj que domina los tiempos de ejecución de los componentes de bin2bcd |
| RST\_aux | RST | Señal para reiniciar a los valores iniciales a las señales mediante los componentes |
| cnt\_AD7476A | | |
| VOLTAJE | Vin | Señal de voltaje que se introduce mediante la entrada del ADC para su muestreo |
| SDATA\_aux | SDATA | Valores muestreados periódicamente en función de SCLK de la señal VOLTAJE |
| CS\_aux | CS | Señal que domina los tiempos de muestreo y de parada entre un muestreo y el siguiente |
| SCLK\_aux | SCLK | Señal de reloj que domina los tiempos de ejecución del ADC |
| bin2bcd | | |
| DATA | DATA | Señal que porta el valor en binario muestreado a convertir a formato bcd |
| DATA\_OK | DATA\_OK | Señal que notifica la disposición de un nuevo valor en DATA para iniciar la conversión |
| cnt\_display | | |
| AND\_30\_aux | AND\_30 | Señal que domina la asignación del display que estamos configurando |
| DP\_aux | DP | Señal que indica si encendemos el punto del display o no |
| SEG\_AG\_aux | SEG\_AG | Señal que contiene la información de los segmentos del display que se han de iluminar |
| display | | |
| D\_Display\_Aux | D\_Display\_Aux | REVISAR |
| BCD\_U | BCD\_U | Señal que porta el valor de las unidades del valor en cnt\_display |
| BCD\_D | BCD\_D | Señal que porta el valor de las decenas del valor en cnt\_display |
| BCD\_C | BCD\_C | Señal que porta el valor de las centenas del valor en cnt\_display |
| BCD\_M | BCD\_M | Señal que porta el valor de las unidades de millar del valor en cnt\_display |
| D\_Display | D\_Display | Señal que representa el valor mostrado en los 4 displays |

Comenzando la simulación iniciamos tras declarar el periodo del reloj (CLK) y una primera señal de reset de 123ns de duración (RST) para iniciar el sistema desde valores iniciales. Comprobándose que las primeras señales en variar desde el inicio, aparte de la introducción del voltaje para el ADC igual a 3.3V, son las relativas al último módulo diseñado de cnt\_AD7476A lo que significa que iniciará el muestreo del voltaje. Debido a la simulación temporal encontramos al inicio de cronograma una serie de ruidos debido a la naturaleza de esta simulación, además de variaciones en tiempos de estimulaciones de señales respecto a la anterior simulación.

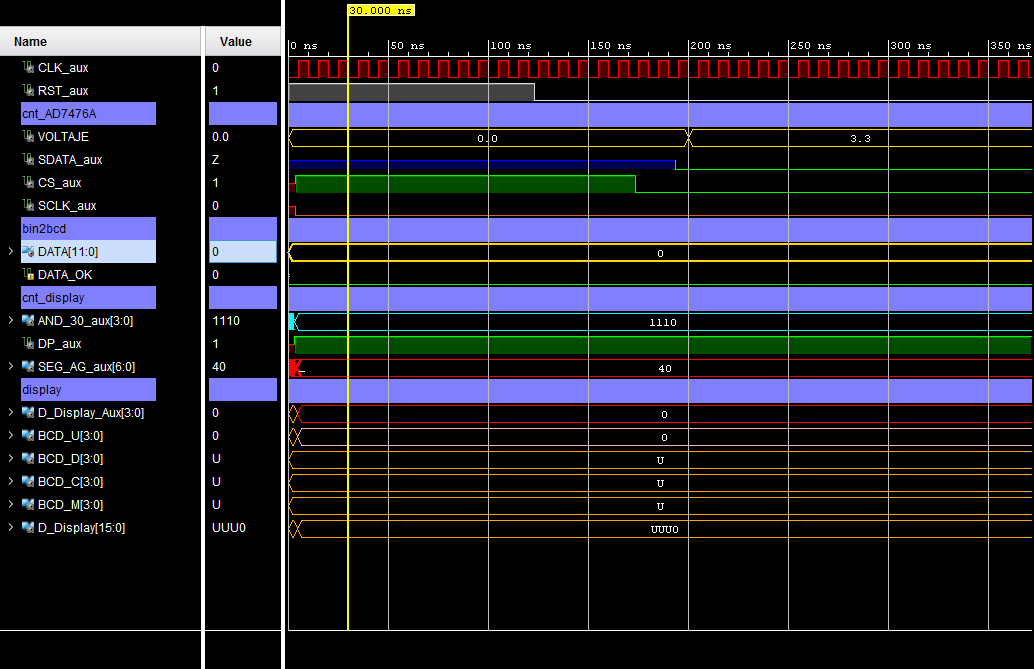


Ilustración -Cronograma de inicio de la simulación con voltaje igual a 3.3V

Tras finalizar el muestreo y la conversión del voltaje realizada en combinación con el ADC y el módulo cnt\_AD7476A, tenemos el valor del voltaje introducido convertido a formato binario el cual se indica en el cambio de valor DATA , para que a continuación se emita un pulso en DATA\_OK para notificar al módulo bin2bcd la variación de valor de DATA para iniciar la nueva conversión a formato bcd.

En comparación con la anterior simulación se puede observar que la variación del valor de DATA no es tan limpia, sino que presenta un ruido bastante imperceptible en la transición de un valor a otro.

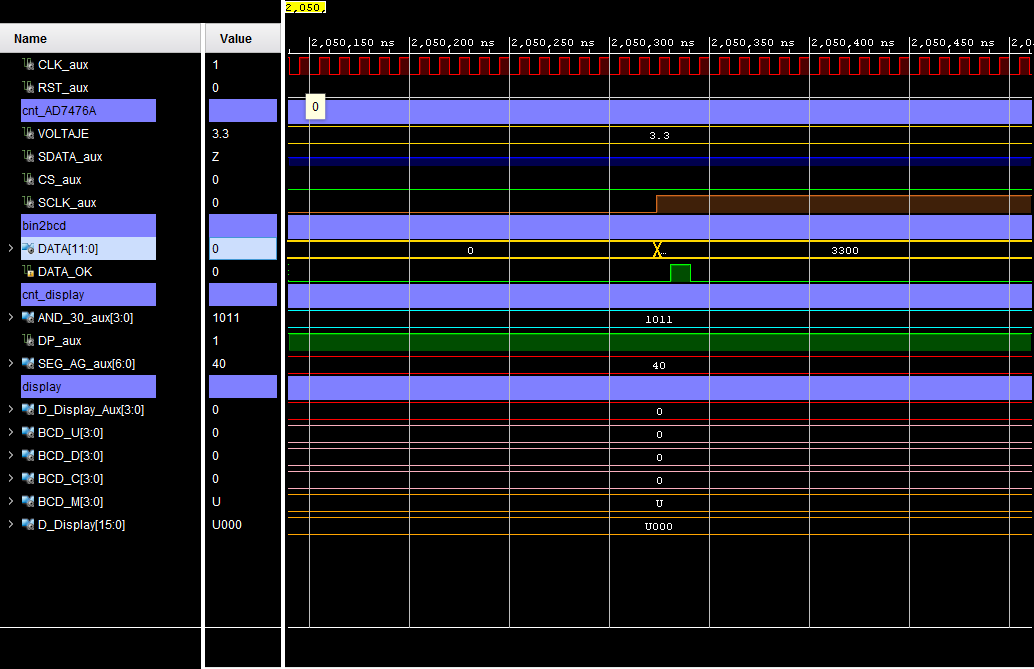


Ilustración -Cronograma de momento de estabilización de DATA tras el muestreo

En el momento que acaba la conversión del valor en formato binario a formato bcd, cuando el contador binario del módulo bin2bcd llega a cero, se modifica el valor de BCD para que en el periodo siguiente se emita su correspondiente pulso por BCD\_OK para notificar al módulo cnt\_display de la disponibilidad de un nuevo valor por BCD para mostrar por los displays.

A diferencia de la anterior simulación se contempla el suceso de ruido (glitch) en ciertas transiciones, como en AND\_30 en este caso, además de la dificultad añadida de no poder visualizar la señal BCD y BCD\_OK debido a que no encontramos en el testbench la posibilidad de añadir esa señal y guiarnos por el cambio en la salida SEG\_AG.

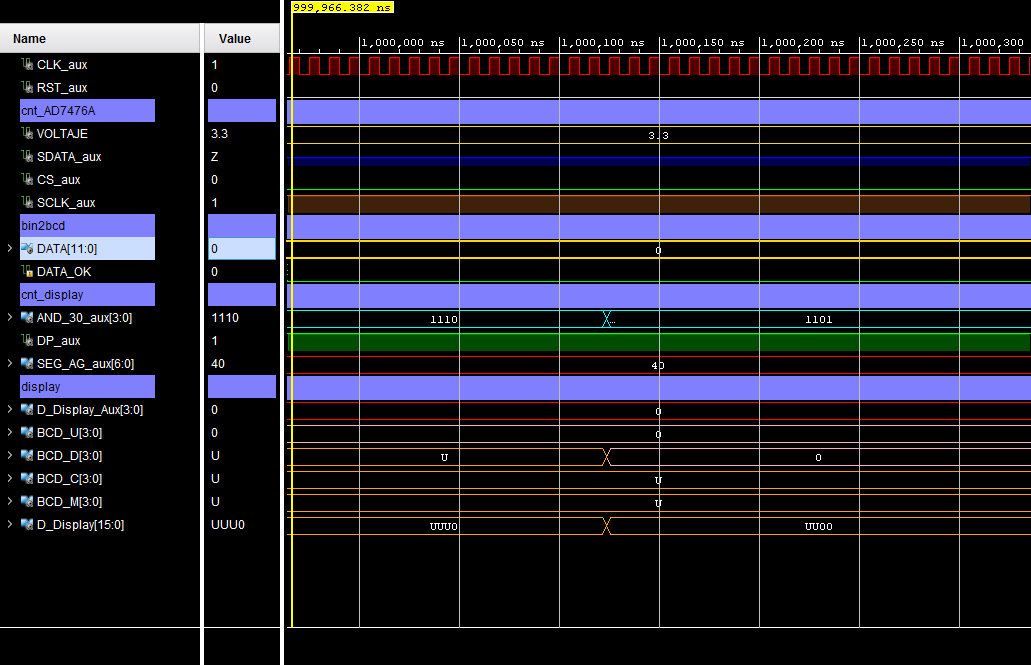


Ilustración -Cronograma de momento de estabilización de BCD tras la conversión de DATA

Como último momento destacable de la simulación representamos el momento en el que el último display se actualiza mediante la función que desempeña el módulo cnt\_display, mostrándose finalmente el valor completo del voltaje introducido en la entrada del sistema por los cuatro displays.

A diferencia de la anterior simulación y suceso que ocurre en la mayoría de estos últimos 3 cronogramas se puede observar que las transiciones no llegan a ser limpias, sino que presentan cierto pequeño ruido (glitch), como el mostrado en este caso en la variación de AND\_30.

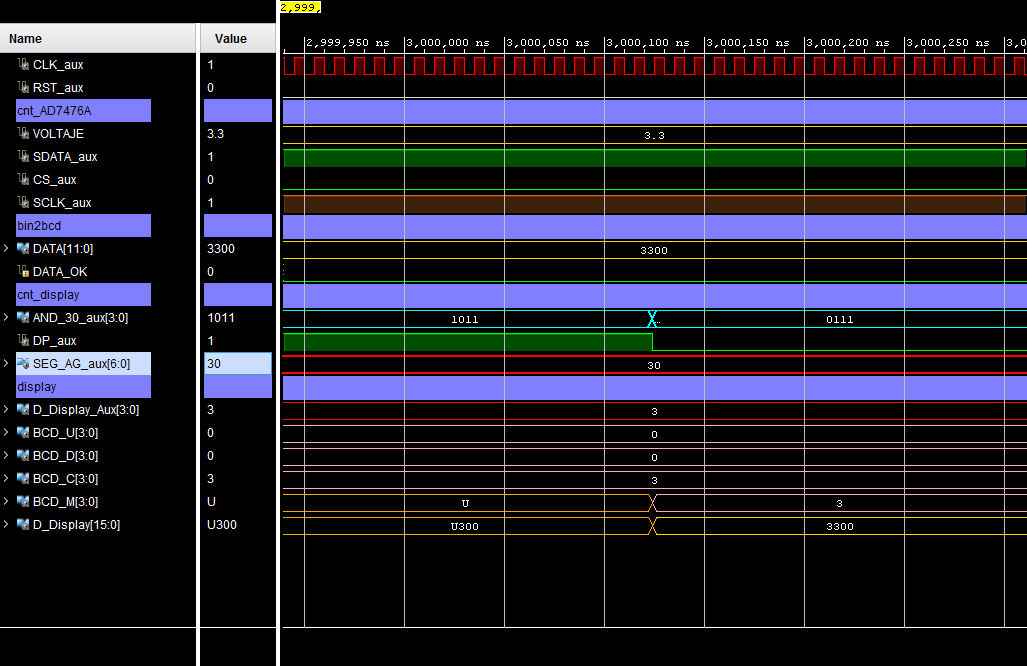


Ilustración -Cronograma de momento de estabilización de la salida del valor de voltaje por display

## CONCLUSIONES

Llegados a este punto ya solo resta examinar las estadísticas en relación a los recursos usados por parte del sistema, que se resumirá en el conjunto de las estadísticas de los módulos que lo componen.

Primero al examinar la gráfica podemos comprobar que, al igual que en los módulos, el mayor porcentaje de uso de un recurso es del recurso IO con un porcentaje de 16%, porcentaje medianamente mantenido en las estadísticas de los módulos.

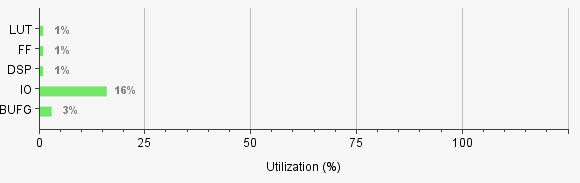


Ilustración -Gráfica de porcentajes de uso de los recursos

Después de este primer análisis, cuando comprobamos la tabla observamos, que al igual que antes, la tendencia presentada en los módulos se mantiene, representada en el uso predominante de los registros FF.

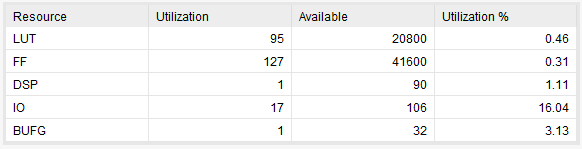


Ilustración -Tabla de información de los recursos