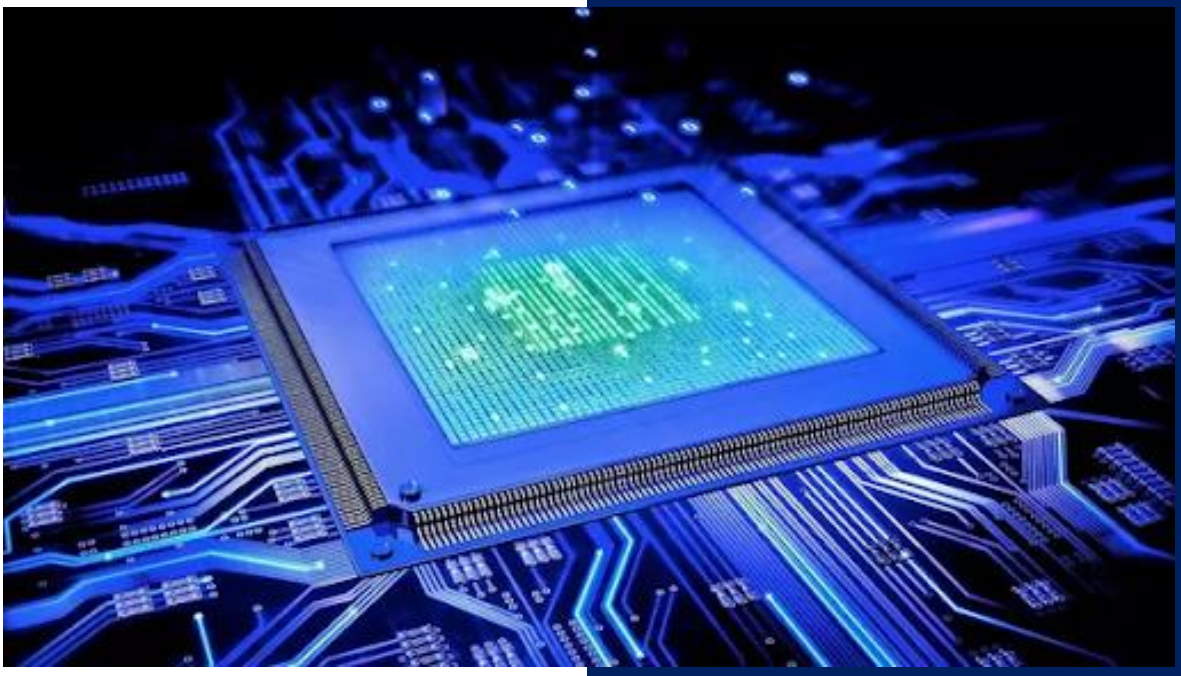
2020

Implementación del Controlador de Display



Universidad Autónoma de Zacatecas

Unidad Académica de Ingeniería Eléctrica

Programa de Ingeniería Robótica y Mecatrónica

Materia: Sistemas Digitales III

Docente: Remberto Sandoval Aréchiga

Alumnos:

José Alfredo Hernández Dueñas

Julio Ángel Pérez Dávila

Agustín Antonio Palafox Molina

José Roberto Novoa López

Jesús Francisco Villaseñor Correa

Equipo “5”

“7-B”

27 de Septiembre del 2020

Resumen

El proyecto está basado en el diseño e implementación de un controlador de una pantalla tipo display de siete segmentos ubicada en la tarjeta de desarrollo Basys 3, para ello se encontrará la descripción de hardware en VHDL para posteriormente implementarlo.

El diseño consiste en convertir un digito decimal del 0 al 15 y mostrarlo en un digito hexadecimal representado en el display. Para ello utilizaremos cuatro switch de la tarjeta como entradas cada una de un bit y un display de 7 segmentos como salida cada una de un bit para que serán las encargadas de representar las 16 combinaciones propuestas. Contará con un botón de reset o botón de reinicio del sistema.

Para diseñar el controlador del display se requirió de analizar la estructura interna de un display para realizar un control en las salidas que nos permite representar de manera correcta los dígitos, también comenzamos a partir de la realización de una caja negra con las entradas y las salidas que esperamos obtener, y para ser más precisos con el diseño de la caja blanca conformamos la arquitectura de nuestro proyecto.

Utilizamos la descripción de Hardware en VHDL para realizar la codificación de nuestra arquitectura y probarla con ayuda de simulaciones que nos permiten analizar el correcto funcionamiento del planteamiento del proyecto y de cada parte de el mismo.

Índice

Introducción

El display 7 Segmentos es un dispositivo opto-electrónico que permite visualizar números del 0 al 9. Existen dos tipos de display, de cátodo común y de ánodo común. Este tipo de elemento de salida digital o display, se utilizaba en los primeros dispositivos electrónicos de la década de los 70's y 80's. Hoy en día es muy utilizado en proyectos educativos o en sistemas clásicos. También debido a su facilidad de uso, mantenimiento y costo, son utilizados en relojes gigantes o incluso como marcadores en algunos tipos de canchas deportivas.

El objetivo del proyecto es diseñar un controlador que nos permita por medio de cuatro entradas representar los números

Para estructurar el display de siete segmentos se divide el proyecto en varias secciones como se comentó anteriormente, comenzando por el análisis del circuito interno del display, para posteriormente determinar su conexión y los ciertos requerimientos que el controlador debe cumplir para accionar el digito deseado.

Display 7 segmentos

Un display 7 segmentos consiste de 7 LED (light-emitting diode) organizados de manera tal que controlando cuales LED están encendidos y cuales, apagados, se pueden visualizar números decimales del 0 hasta el 9 sobre el dispositivo y algunas letras del alfabeto. Cada uno de los dígitos es nombrado con una letra, desde el dígito A hasta el G. Existen dos tipos de display 7 segmentos. Los que tienen cátodo común y los que tienen ánodo común.

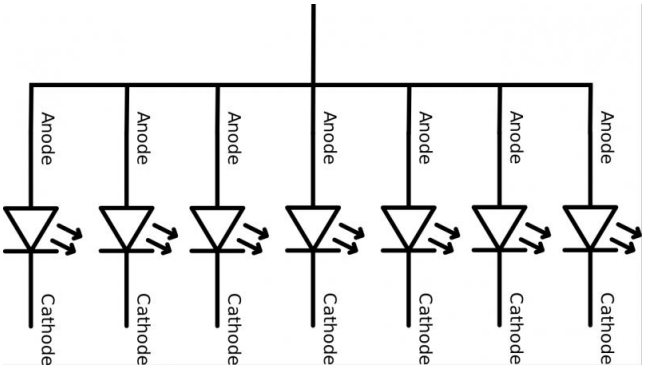


Imagen 1. Diagrama Display 7 segmentos

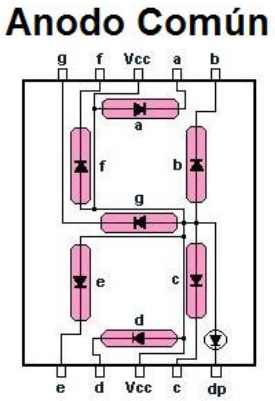


Imagen 2. Conexión interna ánodo común

La implementación de un controlador para un display de siete segmentos es algo avanzado que engloba una serie de secciones que van desde un análisis de circuitos hasta una descripción de hardware.

Las principales aplicaciones de los display 7 segmentos son como contadores, relojes de tiempo real, calculadoras, para desplegar marcadores o algún tipo de cuenta regresiva o incremental.

Imagen 3. Marcador con display

Sistema Hexadecimal

El sistema hexadecimal (abreviado como 'Hex', no confundir con sistema sexagesimal) es el sistema de numeración posicional que tiene como base el 16. Su uso actual está muy vinculado a la informática y ciencias de la computación donde las operaciones de la CPU suelen usar el byte u octeto como unidad básica de memoria.

En principio, dado que el sistema usual de numeración es de base decimal y, por ello, solo se dispone de diez dígitos, se adoptó la convención de usar las seis primeras letras del alfabeto latino para suplir los dígitos que nos faltan. El conjunto de símbolos es el siguiente:

Decimal	Binario	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Imagen 4. Tabla de representación Decimal, Binario y Hexadecimal

Diagrama de bloques

En la siguiente figura se muestra el diagrama de caja negra, la cual representa las entradas y salidas que debe tener el sistema.

Imagen 5. Representación de una caja negra básica display 7 segmentos

Tabla de verdad

La tabla d verdad nos ayuda a determinar la lógica de encendido de cada segmento, así como saber con qué combinaciones generamos la representación de algún digito o letra, sirve como un sistema de coordenadas donde el “off” indica que se apaga un segmento y el “on” que se enciende.

Common Anode Seven Segment Truth-Table

	a	b	c	d	e	f	g
0	Off	Off	Off	Off	Off	Off	On
1	On	Off	Off	On	On	On	On
2	Off	Off	On	Off	Off	On	Off
3	Off	Off	Off	Off	On	On	Off
4	On	Off	Off	On	On	Off	Off
5	Off	On	Off	Off	On	Off	Off
6	Off	On	Off	Off	Off	Off	Off
7	Off	Off	Off	On	On	On	On
8	Off	Off	Off	Off	Off	Off	Off
9	Off	Off	Off	On	On	Off	Off
A	Off	Off	Off	On	Off	Off	Off
B	On	On	Off	Off	Off	Off	Off
C	Off	On	On	Off	Off	Off	On
D	On	Off	Off	Off	Off	On	Off
E	Off	On	On	Off	Off	Off	Off
F	Off	On	On	On	Off	Off	Off

Imagen 6. Tabla de verdad ejemplo con 16 combinaciones posibles

Representación de dígitos en el display de segmentos

Estas serían las 16 combinaciones posibles que se determinaron para representar los números del 0 al 9 y las letras de la A al F, lo que permite representar los numero en hexadecimal de 0 al 15.

Imagen 7. Combinaciones de un display de 7 segmentos

VHDL

VHDL es un lenguaje de especificación definido por el IEEE (Institute of Electrical and Electronics Engineers) (ANSI/IEEE 1076-1993) utilizado para describir circuitos digitales y para la automatización de diseño electrónico. VHDL es acrónimo proveniente de la combinación de dos acrónimos: VHSIC (Very High Speed Integrated Circuit) y HDL (Hardware Description Language). Aunque puede ser usado de forma general para describir cualquier circuito digital se usa principalmente para programar PLD (Programable Logic Device - Dispositivo Lógico Programable), FPGA (Field Programmable Gate Array), ASIC y similares.

Originalmente, el lenguaje VHDL fue desarrollado por el departamento de defensa de los Estados Unidos a inicios de los años 80 basado en el lenguaje de programación ADA con el fin de simular circuitos eléctricos digitales. Posteriormente se desarrollaron herramientas de síntesis e implementación en hardware a partir de los archivos VHD.

FPGA Artix 7

Los FPGA Xilinx Artix -7 ofrecen un rendimiento de coste optimizado en categorías que incluyen lógica, procesamiento de señales, memoria integrada, E / S LVDS, interfaces de memoria y, en particular, transceptores. Los FPGA Artix-7 son ideales para aplicaciones sensibles a los costos que necesitan características de alta gama.

Optimizada para FPGA de Xilinx, la CPU MicroBlaze es un procesador RISC de 32 bits altamente configurable que ofrece una implementación rápida y ajustes preestablecidos para casos de uso de microcontroladores, procesadores en tiempo real y procesadores de aplicaciones. Los FPGA Artix-7 ofrecen otras capacidades de integración de sistemas, como la tecnología de señal mixta analógica (AMS) avanzada e integrada. Analógico es el siguiente nivel de integración que se logra de

manera eficiente con los convertidores analógicos a digitales duales independientes de 12 bits, 1MSPS y 17 canales en los FPGA Artix-7.

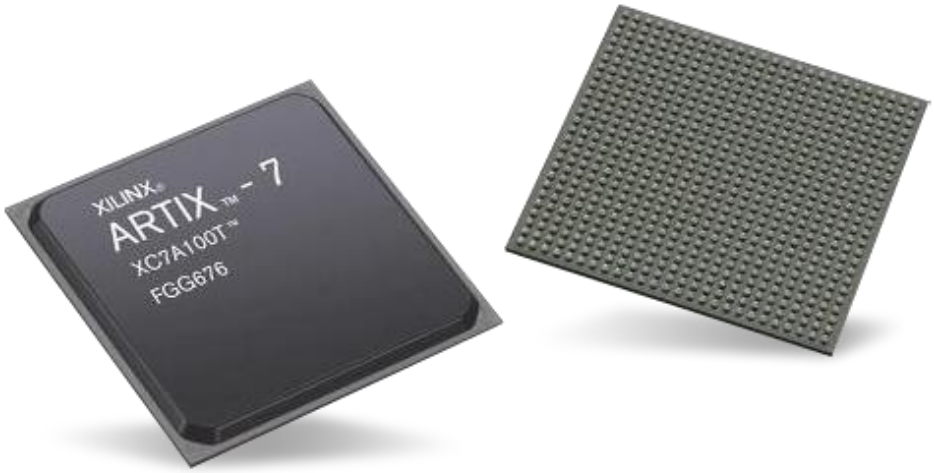


Imagen 8. Artix 7

Placa de desarrollo Basys 3

El Basys 3 es una placa de desarrollo FPGA de nivel de entrada diseñada exclusivamente para Vivado Design Suite, con arquitectura FPGA Xilinx Artix-7. Basys 3 es la última incorporación a la popular línea de placas de desarrollo FPGA, y se adapta perfectamente a estudiantes o principiantes que recién comienzan a utilizar la tecnología FPGA. Incluye las características estándar que se encuentran en todas las placas Basys: hardware completo listo para usar, una gran colección de dispositivos de E / S integrados, todos los circuitos de soporte FPGA requeridos.

Estadísticas:

Dispositivo / IC: Xilinx Artix-7 FPGA (XC7A35T-1CPG236C)

Conector (s):

- USB A
- USB micro-B
- Cuatro puertos Pmod de 12 pines
- VGA

Características:

- Presenta la FPGA Xilinx Artix-7: XC7A35T-1CPG236C
- 33,280 celdas lógicas en 5200 cortes (cada corte contiene cuatro LUT de 6 entradas y 8 flip-flops)
- 1.800 Kbits de RAM de bloque rápido
- Cinco mosaicos de gestión de reloj, cada uno con un bucle de bloqueo de fase (PLL)
- 90 porciones DSP
- Velocidades de reloj interno superiores a 450 MHz

- Convertidor de analógico a digital en chip (XADC)
- Puerto Digilent USB-JTAG para programación y comunicación FPGA
- Diseñado exclusivamente para Vivado Design Suite
- Descarga gratuita de WebPACK ™ para uso estándar
- Cable USB Micro-B
- Flash en serie
- Puente USB-UART
- Salida VGA de 12 bits
- Host USB HID para ratones, teclados y tarjetas de memoria
- 16 interruptores de usuario
- 16 LED de usuario
- 5 pulsadores de usuario
- Pantalla de 4 dígitos y 7 segmentos
- 4 puertos Pmod: 3 puertos Pmod estándar de 12 pines, 1 puerto de señal XADC de doble propósito / puerto Pmod estándar

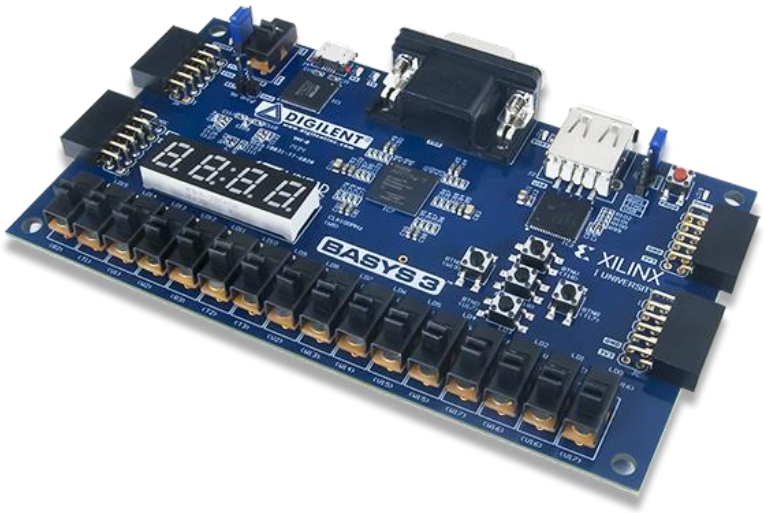


Imagen 9. Tarjeta de desarrollo Basys 3

Requerimientos

Diagrama de caja negra

Como se puede ver en el diagrama de caja negra se compone por diferentes componentes como lo son 4 datos de entrada que son los encargados de enviar la señal a su display correspondiente en un formato binario creando así números del 0 al 15 en formato hexadecimal.

El nombre que se le dio a estos datos de entrada (siendo todos de 4 bits) son los siguientes:

- i_Datos_0
- i_Datos_1
- i_Datos_2
- i_Datos_3

De igual manera hay otros dos valores de entrada:

i_Reloj

Esta entrada es la encargada de dar al controlador una referencia temporal, su uso es para generar una frecuencia de selección entre los cuatro displays, siendo que para cada display dispone de 30hz (120hz en total por los 4 displays).

i_Reset

La única función de esta entrada es restaurar el sistema a un estado inicial.

Este sistema también tiene 2 salidas:

o_Segmentos

Esta señal representa los segmentos que encenderán del display seleccionado en formato hexadecimal como ya se mencionó anteriormente.

o_Anodo_4_Bits

Es la señal que controla el display que será encendido con una frecuencia de 120hz, el método para controlar esto es poniendo en bajo el bit correspondiente del display que se desea utilizar.

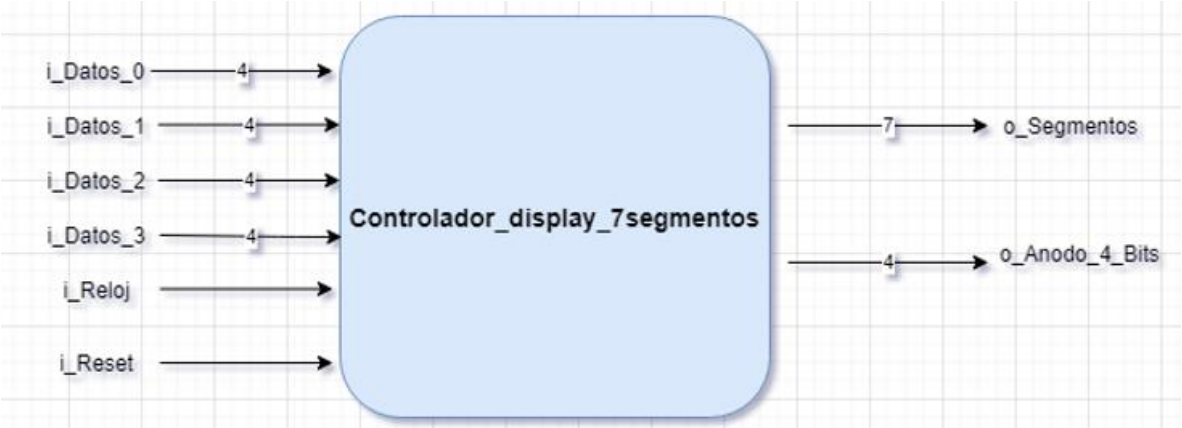


Imagen 10. Diagrama de caja negra

Diagrama de caja blanca

Interiormente el controlador está compuesto por 4 elementos:

- Demultiplexor
- Pre escalador
- Decodificador de anillo
- Decodificador binario

Demultiplexor

Circuito combinacional con entradas de datos y entradas de control de datos.

Pre escalador

Circuito combinacional encargado de tomar la frecuencia de reloj y dividirla para posteriormente alimentar al temporizador.

Decodificador de anillo

Circuito combinacional que constituye una cadena cerrada por medio de un registro de desplazamiento en el cual la entrada del 1er flip-flop está condicionada por la salida del último.

Decodificador binario

Circuito combinacional encargado de convertir un código binario de N bits de entrada.

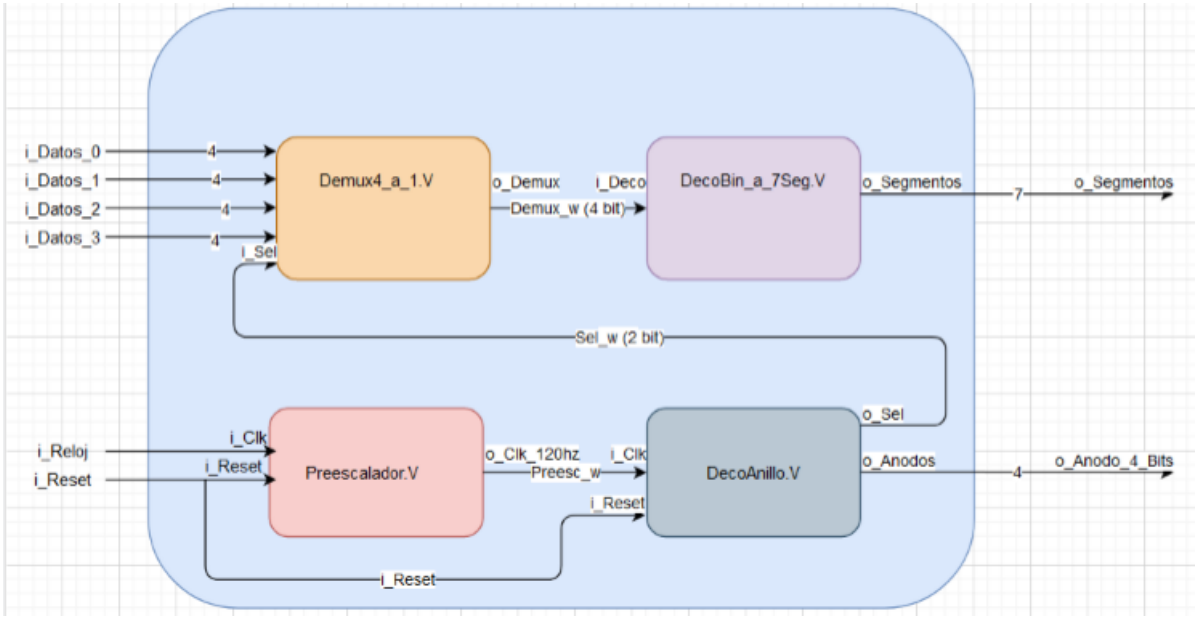


Imagen 11. Diagrama de caja blanca

Arquitectura

Caja Negra de Controlador Display 7 Segmentos

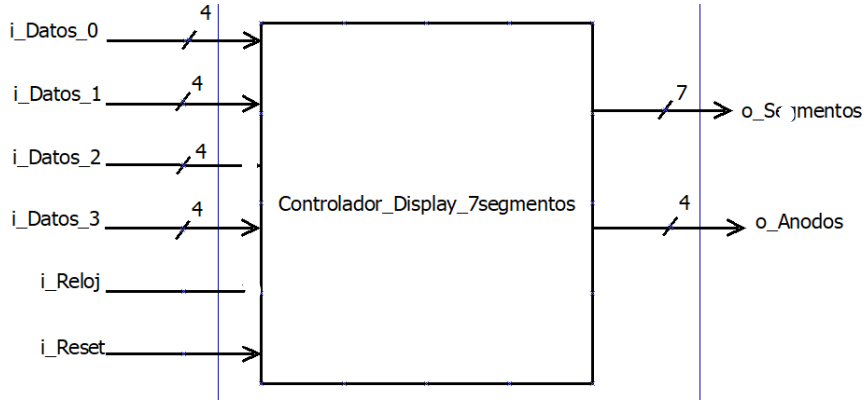


Imagen 12.Caja negra controlador_Display_7segmentos

La tabla 1 indica como nombramos cada señal del controlador display 7 segmentos, tanto numero de bits, que tipo de dato es entrada o salida y una pequeña descripcion de la funcion o accion realiza la señal.

Tabla 1.

Nombre de la señal	Dirección	Tamaño	Descripción
i_Datos_0	Entrada	4	Datos de entrada para display 0.En forma binaria del 0-15 (0x0 -F)
i_Datos_1	Entrada	4	Datos de entrada para display 1.En forma binaria del 0-15 (0x0 -F)
i_Datos_2	Entrada	4	Datos de entrada para display 2.En forma binaria del 0-15 (0x0 -F)
i_Datos_3	Entrada	4	Datos de entrada para display 3.En forma binaria del 0-15 (0x0 -F)
i_Reloj	Entrada	1	Señal de referencia temporal con frecuencia de 100 MHz, cada display maneja 30 Hz dando en total 120 Hz por los 4 displays
i_Reset	Entrada	1	Señal que establece en el sistema un estado inicial.
o_Segmentos	Salida	7	Señal para controlar el encendido y el apagado de los segmentos del display.
o_Anodos	Salida	4	Señal para controlar el display a escribir los datos. Con frecuencia a de 120Hz. Para la selección de los displays con bit que le corresponde colocado en bajo.

Función:

1.- Traducir los datos de formato binario a 7 segmentos.

2.- Seleccionar los datos que se mostraran en el display.

3.- Genera una señal de activación de los ánodos a partir de frecuencia de 120 Hz.

4.- Genera una señal de reloj de 120 Hz a partir de una de 100 MHz.

Caja Blanca de Controlador Display 7 Segmentos

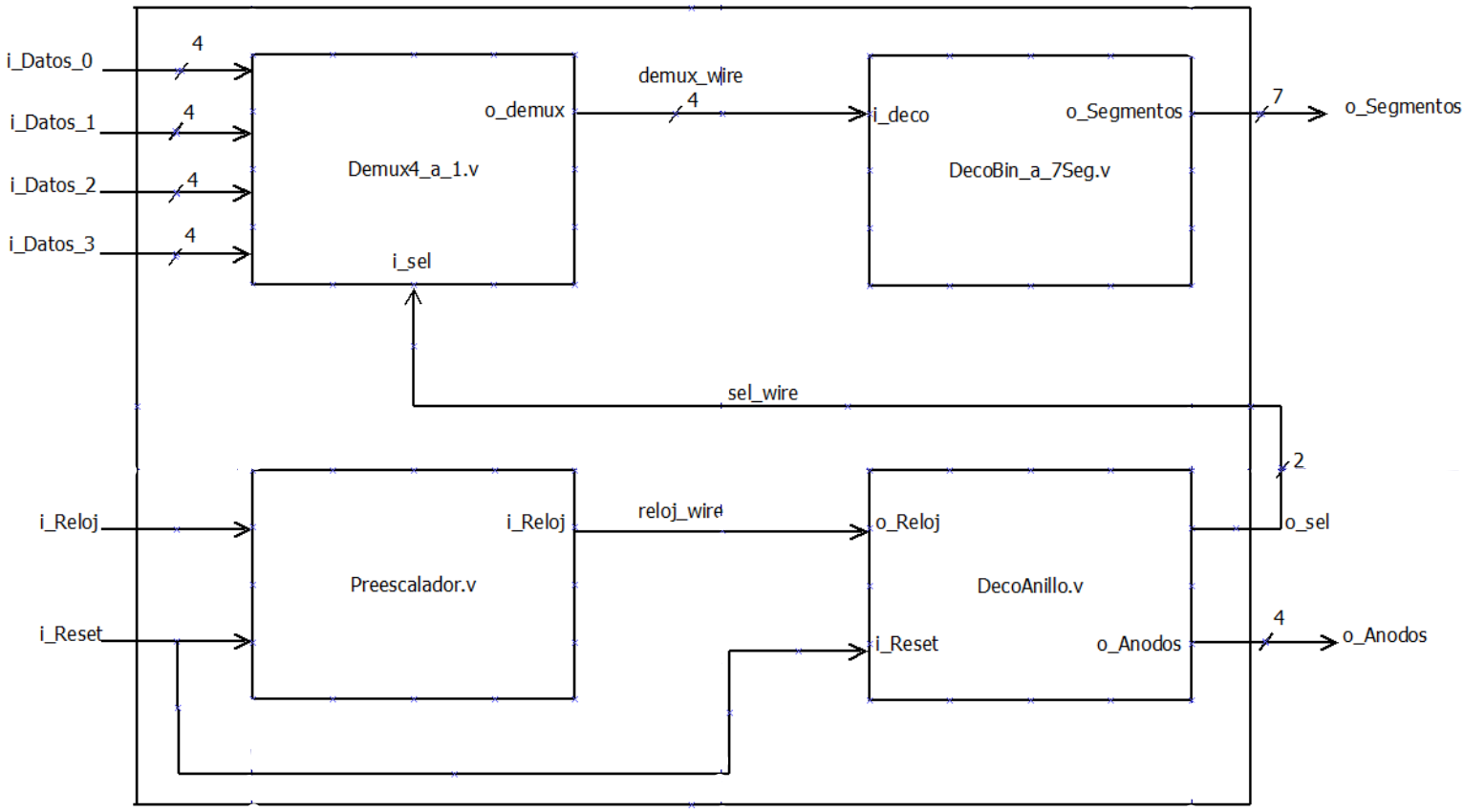


Imagen 13.Caja blanca controlador_Display_7segmentos

La tabla 2 muestra cada submodulo del Controlador de display 7 segmenos para hacer su funcionamiento y nos dice nombre de cada submodulo y el trabajo que desempeñara cada uno de ellos de fomra individual.

Tabla 2.

Bloque	Descripción
Demux4_a_1.v	Seleccionar los datos que se mostraran en el display.
Decobin_a_7Seg.v	Traducir los datos de formato binario a 7 segmentos.
Preescalador.v	Genera un señal de reloj de 120 Hz a partir de una de 100 MHz.
DecoAnillo.v	Genera una señal de activación de los ánodos a partir de frecuencia de 120 Hz.

Función:

Muestra bloque a bloque cómo funcionan los submódulos interactuando entre si por medio de conexiones internas.

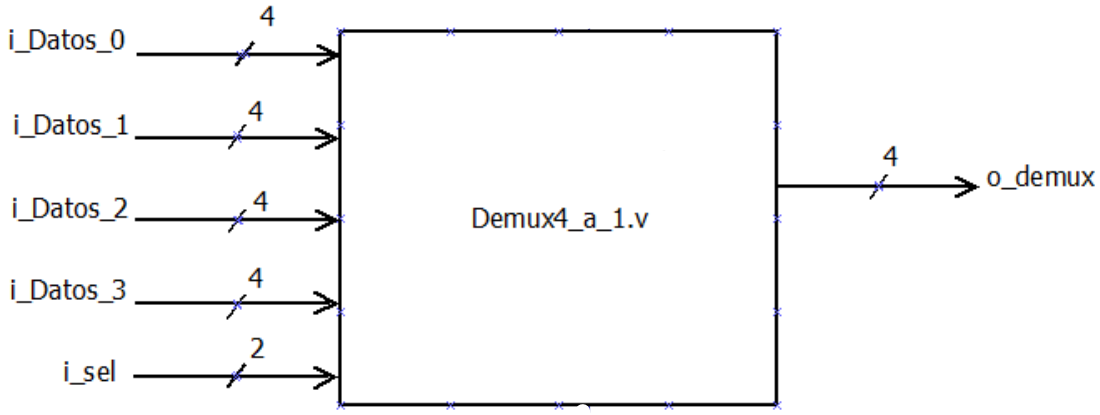


Imagen 14. Demux4_a_1

La tabla 3 indica como nombramos cada señal del mulriplexor 4 a 1, tanto numero de bits, indica que es dato de entrada y una pequeña descripcion de la funcion o accion realiza la señal.

Tabla 3.

Nombre de la señal	Dirección	Tamaño	Descripción
i_Datos_0	Entrada	4	Datos de entrada para display 0.En forma binaria del 0-15 (0x0 -F)
i_Datos_1	Entrada	4	Datos de entrada para display 1.En forma binaria del 0-15 (0x0 -F)
i_Datos_2	Entrada	4	Datos de entrada para display 2.En forma binaria del 0-15 (0x0 -F)
i_Datos_3	Entrada	4	Datos de entrada para display 3.En forma binaria del 0-15 (0x0 -F)
i_Sel	Entrada	2	Se encarga de seleccionar cual dato de entrada se mandara a la salida
o_Demux	Entrada	4	Entrega el dato seleccionado por el selector.

Función:

Se encarga de seleccionar el digito que se mostrará de acuerdo al ánodo correspondiente donde 0 es encendido y 1 apagado.

Tabla del Sel

La tabla 4 muestra que dato de entrada pasara a la salida si es 00 seria i_Dato_0, si es 01 seria i_Dato_1 y asi sucesibamente.

Tabla 4.

i_Sel bit1	i_Sel bit2	Datos
0	0	i_dato_0
0	1	i_dato_1
1	0	i_dato_2
1	1	i_dato_3

Caja Negra del Preescalador

Imagen 15.Preescalador

La tabla 5 nos dice si la señal es de entrada o salida, nombre, tamaño y su funcion

Tabla 5.

Nombre de la señal	Dirección	Tamaño	Descripción
i_Reloj	Entrada	1	Señal de referencia de tiempo a 100 MHz
i_Reset	Entrada	1	Señal que reinicia el preescalador al estado inicial
o_Reloj	Salida	1	Señal modificada a 120 Hz

Función:

1.- Convierte la señal de referencia al tiempo reloj de 100 MHz a una de 120 Hz, obteniendo una señal cada 41666 ciclos positivos y negativos.

2.- La señal i_Reset reinicia el preeescalador a su estado inicial.

Caja Negra del Decodificador Anillo

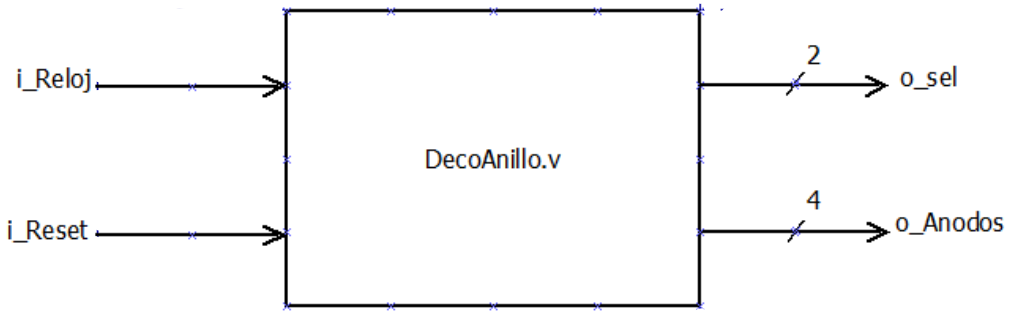


Imagen 16. DecoAnillo

La tabla 6 indica el nombre de la señal, que tipo es entrada o salida tamaño de bits y la funcion que realizara cada uno de ellos.

Tabla 6.

Nombre de la señal	Dirección	Tamaño	Descripción
i_Reloj	Entrada	1	Señal de referencia de tiempo a 100 MHz.
i_Reset	Entrada	1	Señal que reinicia el preescalador al estado inicial.
o_Sel	Salida	2	Señal de salida que esta conectada ala entrada del multiplexor para la selección de un dato.
o_Anodos	Salida	4	Señal de salida que selecciona el display en el cual se muestra nuestros datos con frecuencia de 120 Hz , donde para la selección de cada display el bit correspondiente se coloca abajo.

Función:

1.- Cada que se obtenga un flanco positivo en el reloj, la salida o_Anodos cambiara su valor. Cada que el valor llegue a 3 el conteo se reiniciara.

Tabla 7.

Pulsaciones del reloj	o_Anodos
0	0001
1	0010
2	0100
3	1000

2.- La señal de salida nos permite seleccionar el dato en el multiplexor a su vez también depende de los pulsos del reloj y se reinicia cuando los pulsos llegan a 3

Tabla 8.

Pulsaciones del reloj	o_Sel
0	00
1	01
2	10
3	11

Caja Negra del Decodificador Binario de 7 Segmentos

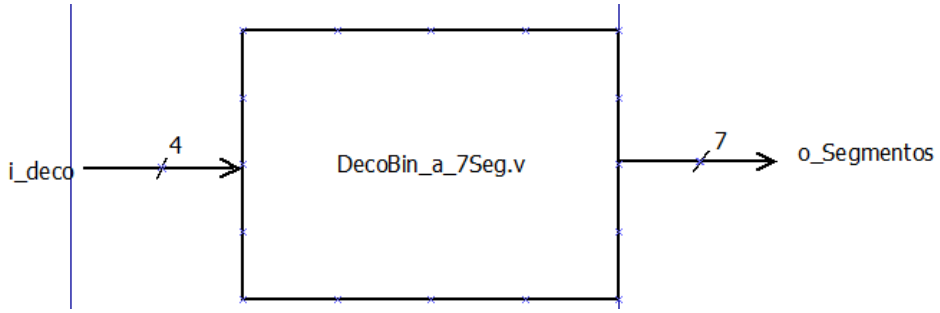


Imagen 17. DecoBin_a_7seg

La tabla 9 muestra el nombre de la señal, si es entrada o salida, tamaño de bits y su funcionamiento

Tabla 9.

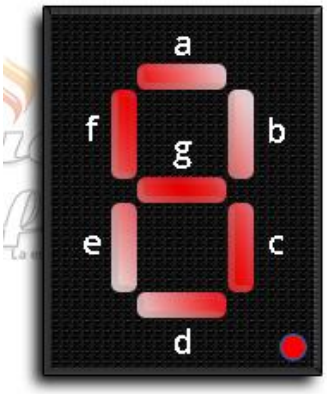
Nombre de la señal	Dirección	Tamaño	Descripción
i_deco	Entrada	4	Señal proviene del multiplexor y se emplea en formato binario donde se representan los numero del 0 al 15 (0x0xF)
o_Segmentos	Salida	7	La señal se encarga de representar datos decodificados de formato hexadecimal y se emplean en formato binario y representan los números del 0 al 15 (0x0xF)

La tabla 10 muestra la representación de que segmentos se encuentran prendidos o apagados para formar un numero de 0 a 9 y/o las letras de la A hasta la F.

Tabla 10.

HEX	A	B	C	D	E	F	G	
0	0	0	0	0	0	0	1	
1	1	0	0	1	1	1	1	
2	0	0	1	0	0	1	0	
3	0	0	0	0	1	1	0	
4	1	0	0	1	1	0	0	
5	0	1	0	0	1	0	0	
6	0	1	0	0	0	0	0	
7	0	0	0	1	1	1	1	
8	0	0	0	0	0	0	0	
9	0	0	0	0	1	0	0	
A	0	0	0	1	0	0	0	
B	1	1	0	0	0	0	0	
C	0	1	1	0	0	0	1	
D	1	0	0	0	0	1	0	
E	0	1	1	0	0	0	0	
F	0	1	1	1	0	0	0	

Diagrama de Ubicación:



Fución:

Recibe entradas de bits que representa un numero BCD y a la salida tiene 7 bits para representar dicho número en el display de 7 segmentos, por lo cual solo se pueden representar los numero del 0 al 9, por conveniencia este componente puede recibir números del 0 al 15 y la salida se encuentra en su representación hexadecimal en 7 segmentos por lo cual la salida es representada en el display.

Pruebas

Preescalador

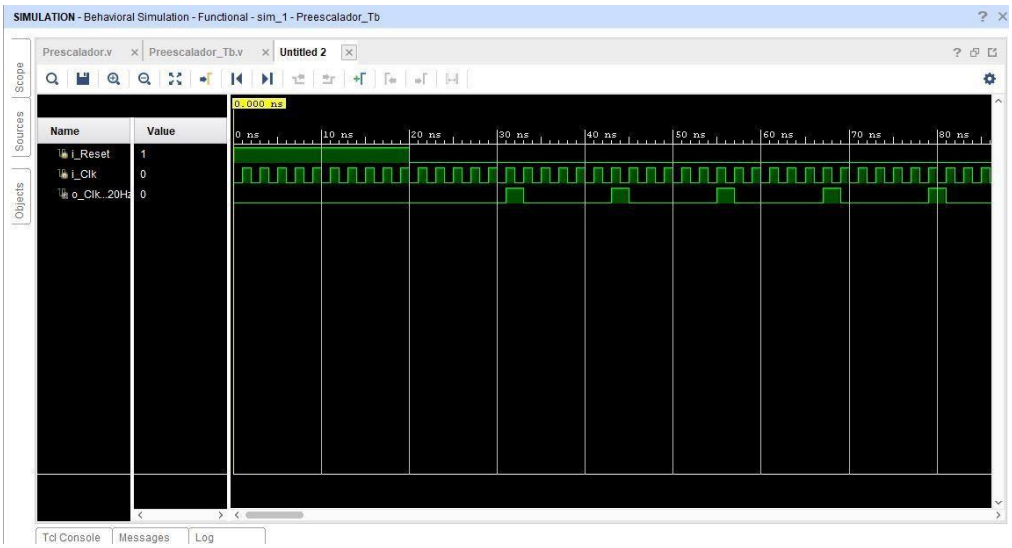


Imagen 18. Simulación del Prees calador.

Como podemos observar en la **imagen 18** el preescalador permite cambiar la señal de reloj a cualquier otra deseada, en este caso se cambia a 120 Hz. Las señales de reloj y reset se inicializan en 0 para poner a trabajar el mismo preescalador. Para la simulación se muestra cada 6 flancos positivos para su mejor apreciación.

Demux

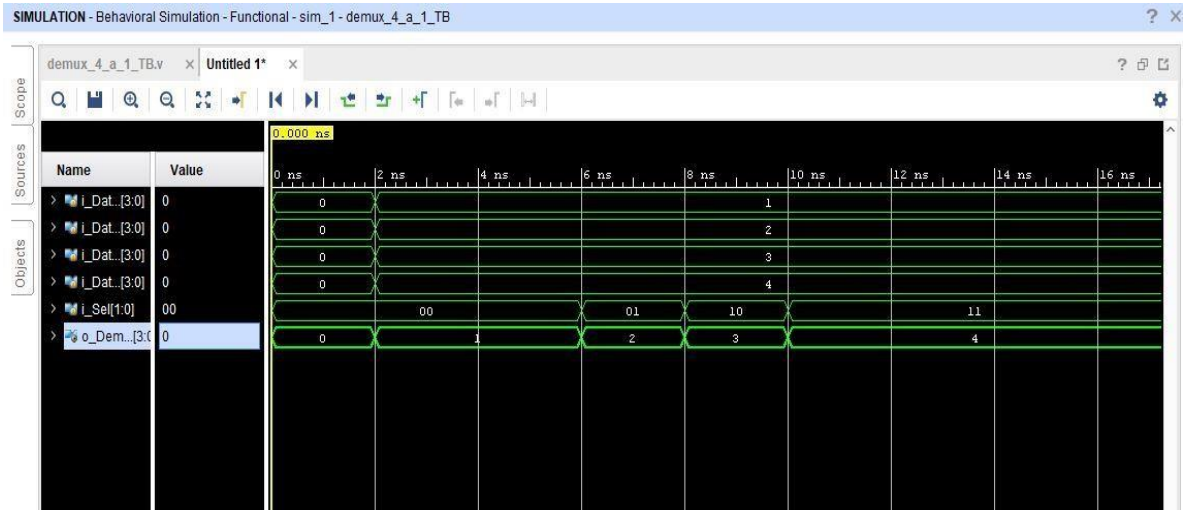


Imagen 19. Simulación Demux_4_a_1.

La simulación de la **imagen 19** del Demultiplexor utilizado en el controlador donde se inicializan todas las entradas 0 por costumbre, al cabo de 2ns se contempla que el selector continuo en la selección 00 indicando que estamos seleccionando en el Dato_0, así mismo se simula la selección de cada una de las opciones tenidas en el Demultiplexor.

DecoAnillo

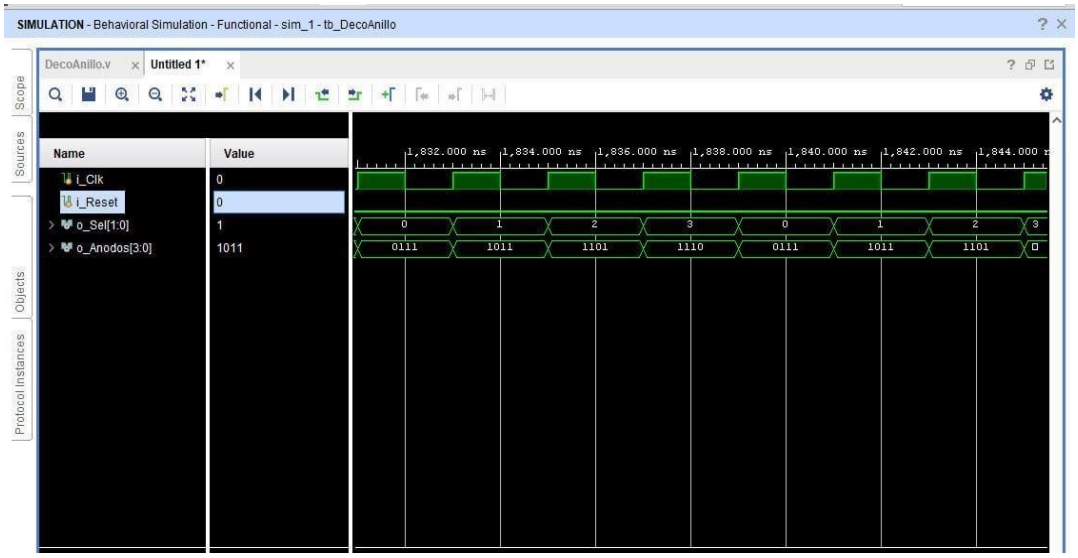


Figura 20. Simulación DecoAnillo.

Se puede observar en la **imagen 20** el funcionamiento del DecoAnillo donde aparecen los valores asignados en el testbench. En primer lugar, tenemos el reloj en bajo ya que en la codificación se negó para que este empezara a funcionar en 0, seguido a él se tiene el reloj en 0 ya que por el momento no queremos regresar al estado inicial todo el DecoAnillo, a continuación, le siguen el selector y la salida a los ánodos donde el selector nos permite seleccionar desde 0 a 3 de acuerdo al dato que se quiera utilizar y por último se observa el ánodo activo de acuerdo a la selección.

Deco_Bin_7seg

Imagen 21. Simulación Deco_Bin_7seg.

En la **imagen 21** podemos observar que en el decodificador binario se tiene que i_Deco se inicializa en 0 y sigue hasta 15 o F. En la salida se tiene la decodificación de estos números binarios donde cabe añadir que el 0 representa ON y 1 OFF esto bien por el tipo de display que utiliza la Basys.

Driver

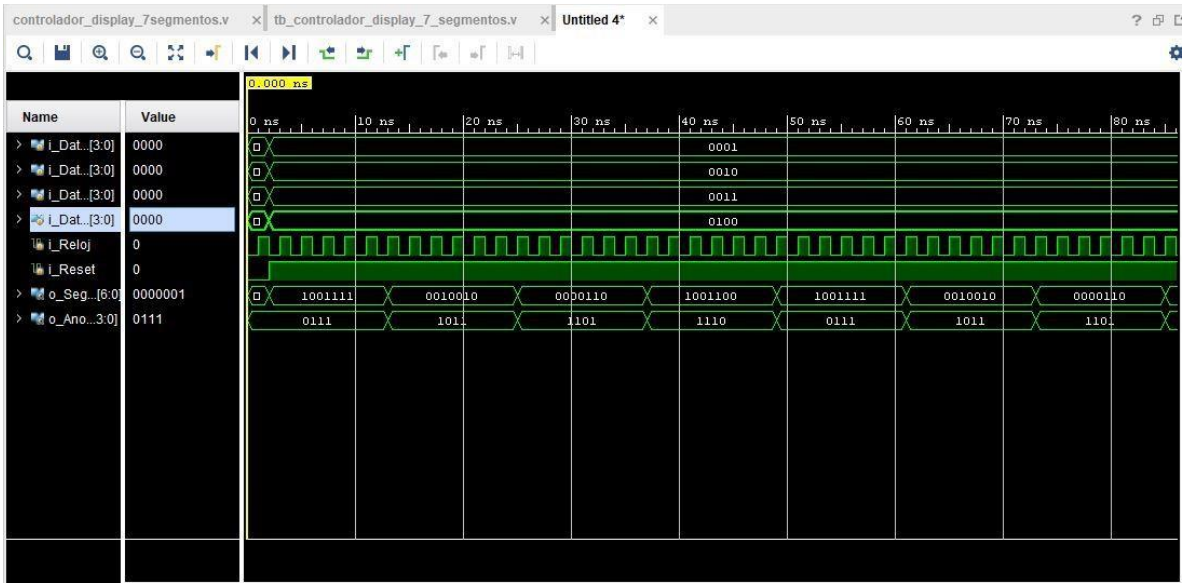


Imagen 22. Simulación del Controlador de display de 7 segmentos.

Por último, en la **imagen 22** se muestra la simulación del funcionamiento del Driver en su totalidad, teniendo en cuenta que se hizo la modificación del límite del preescalador a 6 pulsos para que así se pueda apreciar el cambio en un tiempo más reducido tanto del dato como del ánodo. Primeramente, se inicializan los cuatro datos y el reset en 1; después de 2ns se cambia el reset a 0 y los datos van cambiando a 1, 2, 3 y 4 respectivamente. El reloj cambia su valor cada 1ns indefinidamente, así cada 6 flancos positivos se realizan la multiplexión de dato y ánodo correspondientemente. Así mismo también se realiza la presentación de la simulación con valores binarios para complementar con mayor satisfacción la actividad.

Implementación

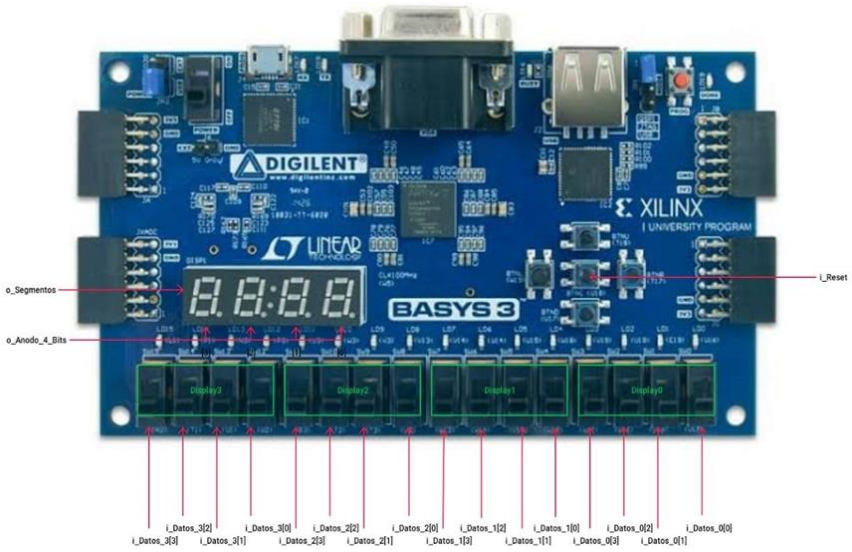


Imagen 23. Representación en la Basys 3.

En la **Imagen 23** podemos observar la tarjeta Basys 3 que se utilizó en este proyecto, los switch con los que se pueden ingresar los datos que se desean mostrar en los dígitos en el display, y los botones que utilizamos.

Cargamos el proyecto a la tarjeta por medio de un cable USB, para después implementar el correcto funcionamiento que las simulaciones arrojaron. Utilizando nuestra tabla 10 de representación de que segmentos se encuentran prendidos o apagados y la imagen 4 que es la representación Decimal, Binario y Hexadecimal nos apoyamos para que con los switch podamos ubicar cual es el que hay que activar para representar el digito deseado.Al final de las 16 representaciones presionamos el botón de reset para comenzar de nuevo.

Cabe resaltar que para el correcto funcionamiento de los switch y botones que utilizamos, es necesario utilizar el archivo de restricciones de la Basys 3 y modificarlo de acuerdo a los elementos que nosotros vamos a utilizar.

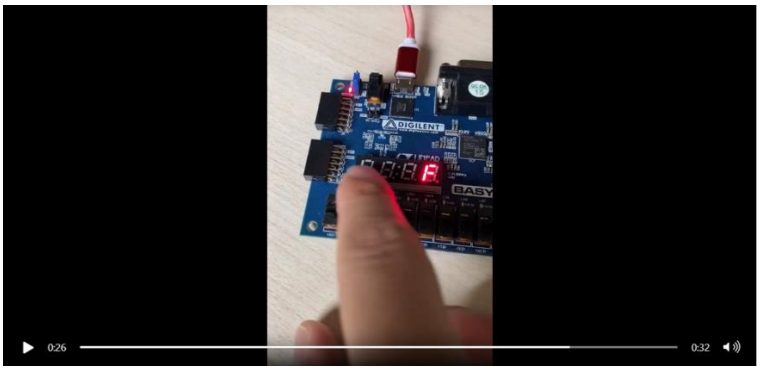


Imagen 24. Implementación en la tarjeta Basys 3.

Análisis de resultados

El sistema funcionó de manera a la cual las simulaciones que respectivamente habíamos especificado y trabajado por la unión de los equipos, quien cada uno fue encargado de una pequeña parte de lo que es el proyecto finalizado, mostrando en físico por medio de la placa Basys un desempeño óptimo para lo que se buscaba.

Conclusión:

Gracia a los datos obtenidos podemos ver que para hacer un controlador de display 7 segmentos se requiere cierto conocimiento, sobre la función principal, por ejemplo, como se compone internamente, que es lo que se muestra ya físicamente todo lo relacionado a lo que se quiera desarrollar, como vimos el controlador se conforma de señales de entrada y salidas e incluso con señales internas, cables que conecta una salida con una entrada, etc. Para entender mejor los componentes internos que se usaron fueron un multiplexor él te mostrara el dato solicitado, un preescalador que genera una señal de reloj de 120 Hz a partir de una de 100 MHz, también un decodificador de anillo es encargado de la selección de display de la basys3 y también y menos importante un decodificador binario de 7 segmentos que se encarga de representar con los segmentos un numero o letra. Y la unión de todo ellos forman el dichoso controlador de display 7 segmentos que se ve en los relojes de microondas, letreros led, etc.

Referencias:

[1] M. Morris Mano, *Diseño Digital 3ra Edición*. California State University, Los Ángeles. Pearson Educación, 2003.

[2] Thomas L. Floyd, Fundamentos de Sistemas Digitales 9na Edición. Madrid. Pearson Educación, 2006

[3] José Ignacio Martínez y Javier Castillo. (2019). ¿Qué es una FPGA?, Universidad Rey Juan Carlos de Madrid,

https://vhdl.es/fpga/

[4] Francisco Pineda. (2015). Basys 3 Artix-7 FPGA Board, https://sites.google.com/site/logicaprogramable/calculadoras/fpga/basys-3-artix-7-fpga-board

[5] Hetpro. Display 7 Segmentos ánodo y cátodo común, https://hetpro-store.com/TUTORIALES/display-7-segmentos-anodo-catodo-comun/

Apéndice.

Para este proyecto hubo un reto importante el cual en si no fue la realización del proyecto el cual si bien tubo su complejidad lo más complejo fue el hecho de trabajar en conjunto con varios equipos a la vez y a distancia con esto trayendo más de una vez los malos entendidos de cómo llevar a cabo el proyecto, generalmente más a errores gramaticales como que algunos usaran mayúsculas y otros minúsculas algunos usaron guiones y otros no que aunque parezcan errores simples si no se corrigen a tiempo al momento de integrar los trabajos el trabajo de todos se vuelve algo fastidioso el tener que corregirlos sobre tono al hacer las conexiones de este o al hacer la synthesis que generalmente cuando tedas cuenta de ellos pero en fin como experiencia y aprendizaje este tipo de trabajos son excelentes justamente para tener la experiencia de trabajar con múltiples grupos de trabajos en lo personal aunque a mi parece no fue tan mal este proyecto faltan cosas que mejorar lo más importante la comunicación entre equipos que desde mi punto de vista fue lo que más falto en este proyecto pero es algo que se puede mejorar para futuros proyectos.

Códigos

1.Codigo del controlador_display_7segmentos

```
1     `timescale 1ns / 1ps
2     //////////////////////////////////////////////////
3     // Company: UAZ
4     // Engineer: Alfa
5     //
6     // Create Date: 07.09.2020 10:37:39
7     // Design Name:controlador_display_7segmentos
8     // Module Name: controlador_display_7segmentos
9     // Project Name: controlador_display_7segmentos
10    // Target Devices: Basys 3
11    // Tool Versions:
12    // Description:
13    //
14    // Dependencies:
15    //
16    // Revision:
17    // Revision 0.01 - File Created
18    // Additional Comments:
19    //
20    //////////////////////////////////////////////////////////////////////////////////
21
22
23    module Controlador_display_7segmentos(
24        input [3:0] i_Datos_0,
25        input [3:0] i_Datos_1,
26        input [3:0] i_Datos_2,
27        input [3:0] i_Datos_3,
28        input i_Reloj,
29        input i_Reset,
```

```
30          output [6:0] o_Segmentos,
31          output [3:0] o_Anodo_4_Bits
32          );
33          wire [1:0]Sel_w;
34          wire [3:0]Demux_w;
35          wire Preesc_w;
36
37              Demux_4_a_1 mux(
38              .i_Datos_0(i_Datos_0),
39              .i_Datos_1(i_Datos_1),
40              .i_Datos_2(i_Datos_2),
41              .i_Datos_3(i_Datos_3),
42              .i_Sel(Sel_w),
43              .o_Demux(Demux_w)
44              );
45
46              DecoBin_7Seg dec_BCD (
47              .i_Deco(Demux_w),
48              .o_Segmentos(o_Segmentos)
49              );
50
51              DecoAnillo dec_anillo (
52              .i_Clk(Preesc_w),
53              .i_Reset(i_Reset),
54              .o_Anodos(o_Anodo_4_Bits),
55              .o_Sel(Sel_w)
56              );
57
58              Prescalador preesc (
59              .i_Reloj(i_Reloj),
60              .i_Reset(i_Reset),
61              .o_Clk_120Hz(Preesc_w)
62              );
63
64          endmodule
```

2.código del multiplexor

```
1       `timescale 1ns / 1ps
2
        //////////////////////////////////////////////////////////////////////////////////
3       // Company:Universidad Autónoma de Zacatecas
4       // Engineers:
5       //Agustin Antonio Palafox Molina
6       //Jose Alfredo Hernadez Dueñas
7       //Jesus Francisco Villaseñor Correa
8       //José Roberto Novoa López
9       //Julio Ángel Pérez Dávila
10      // Create Date: 07.09.2020 10:23:49
11      // Design Name:
12      //Deco_Bin_7Seg
13      // Module Name: Demux_4_a_1
```

```
14    // Project Name:
15    //Controlador_display_7segmentos
16    // Target Devices:
17    // Tool Versions:2018.1
18    // Description:
19    //El decodificador se encargara de transformar las variables
20    binarias en una serie de salidas que representan un segmento
21    que conforman un display numerico de siete de ellos.
22    //Al unir varios segmentos formamos o representamos un digito.
23    // Dependencies:
24    //
25    // Revision:
26    // Revision 0.01 - File Created
27    // Additional Comments:
28    //
29    //////////////////////////////////////////////////////////////////
30
31
32    module Demux_4_a_1
33        #(parameter n=4)
34        (
35        input [n-1:0] i_Datos_0,
36        input [n-1:0] i_Datos_1,
37        input [n-1:0] i_Datos_2,
38        input [n-1:0] i_Datos_3,
39        input [1:0] i_Sel, // entrada selectora del mux
40            output reg [n-1:0] o_Demux // salida
41        );
42
43        always @(i_Datos_0, i_Datos_1, i_Datos_2, i_Datos_3, i_Sel)
44        begin
45            case(i_Sel)
46                2'b00: begin o_Demux <= i_Datos_0; end
47                2'b01: begin o_Demux <= i_Datos_1; end
48                2'b10: begin o_Demux <= i_Datos_2; end
49                default : begin o_Demux <= i_Datos_3; end
50            endcase
51        end
52
53    endmodule
```

3.código de el Preescalador.

```
1     timescale 1ns / 1ps
2     //////////////////////////////////////////////////////////////////
3     // Company: Universidad Autonoma Zacatecas
4     // Engineer: Jorge Manuel de Avila Reveles, Geovanni López Macias,
         Gerardo Frías Donlucas, César Rodolfo Salazar Rodríguez
5     //
6     // Create Date: 09.09.2020 10:38:22
7     // Design Name:
8     // Module Name: Preescalador
9     // Project Name: Controlador_Display_7Segmentos
10    // Target Devices: Basys 3
11    // Tool Versions: Vivado 2018.2
```

```
12    // Description: Modulo encargado de convertir la señal base de la
         tarjeta en una mas pequeña

13    //
14    // Dependencies:
15    //
16    // Revision: 0.2
17    // Revision 0.01 - File Created
18    // Additional Comments:
19    //
20    //////////////////////////////////////////////////////////////////////
21
22
23
24    module Prescalador #(parameter lim=416666) (
25        input i_Reloj,
26        input i_Reset,
27        output reg o_Clk_120Hz
28    );
29     reg  [19:0] cuent; // registro para almacenar la cuenta interna del
                             preescalador
30
31    always @ ( posedge i_Reloj, negedge i_Reset)
32
33        if (i_Reset)
34        begin
35            cuent <= 0;
36            o_Clk_120Hz <= 0;
37        end
38
39     else if (cuent < lim) //aumenta la cuenta en uno mintras no llegue
                               al limite
40         begin
41             cuent <= cuent + 1'd1;
42             o_Clk_120Hz <= 0;
43         end
44
45     else // manda un pulso a la salida cuando llega al limite y la
            cuenta regresa a cero

46         begin
47             cuent <= 0;
48             o_Clk_120Hz <= 1;
49         end
50    endmodule
```

4.código del decodificador anillo.

```
1     `timescale 1ns / 1ps
2     //////////////////////////////////////////////////////////////////////
3     // Company:
4     // Engineer:
5     //
6     // Create Date: 14.09.2020 09:36:26
7     // Design Name:
```

```
8     // Module Name: DecoAnillo
9     // Project Name:
10    // Target Devices:
11    // Tool Versions:
12    // Description:
13    //
14    // Dependencies:
15    //
16    // Revision:
17    // Revision 0.01 - File Created
18    // Additional Comments:
19    //
20    //////////////////////////////////////////////////////////////////////////////////
21
22
23    module DecoAnillo(
24       input i_Reset,
25       input i_Clk,
26       output [1:0] o_Sel,
27       output [3:0] o_Anodos      //Declaramos el bus de salida an,
                                      para controlar los andos.
28       );
29
30    wire [3:0]in;  //Declaramos el bus auxiliar tipo wire de 4 bits in.
31    wire [15:0]x;
32    reg [1:0]Clk;  //Declaramos el bus auxiliar tipo reg de 2 bits sel.
33
34    //Contador de corrida libre de 2 bits.
35    always @(posedge i_Clk or negedge i_Reset)   //Siempre que ocurra
                                    el flanco positivo de i_Clk o i_Reset.
36       if(i_Reset)Clk<=0;    //Si se activa Clk el siguiente valor de
                                    sel sera 0.

37       else Clk<=Clk+1;  //De lo contrario el siguiente valor de o_Sel
                               sera sel+1.
38
39    //Multiplexor Quadruple de 4 entradas 1 salidas.
40    assign in=x[Clk*4+:4]; //Si sel=2 entonces in=x[2*4+3+:4] o
                                 in=x[11:8].
41   //Por lo tanto se seleccionan las 4 entradas correspondientes al
       display a encender.
42
43
44    //Decodificador de 2 a 4. Selecciona el bit de salida
        correspondiente.
45    assign o_Anodos=(Clk==0)?4'b0111:   //Si sel es igual a 0 an sera
                                             igual a 4'b0111
46              (Clk==1)?4'b1011:   //Si sel es igual a 1 an sera igual a
                                       4'b1011

47              (Clk==2)?4'b1101:   //Si sel es igual a 2 an sera igual a
                                       4'b1101
48                       4'b1110;   //Si sel es igual a 3 an sera igual a
                                       4'b1110
49    assign o_Sel=(Clk==0)?2'b00:
50              (Clk==1)?2'b01:
51              (Clk==2)?2'b10:
```

```
52                          2'b11;
53
54      endmodule
```

5.código del decodificador bcd.

```
1       `timescale 1ns / 1ps
2       //////////////////////////////////////////////////////////////////
3       // Company:Universidad Autónoma de Zacatecas
4       // Engineers:
5       //Agustin Antonio Palafox Molina
6       //Jose Alfredo Hernadez Dueñas
7       //Jesus Francisco Villaseñor Correa
8       //José Roberto Novoa López
9       //Julio Ángel Pérez Dávila
10      // Create Date: 07.09.2020 10:23:49
11      // Design Name:
12      //Deco_Bin_7Seg
13      // Module Name: Deco_Bin_7Seg
14      // Project Name:
15      //Controlador_display_7segmentos
16      // Target Devices:
17      // Tool Versions:2018.1
18      // Description:
19      //El decodificador  se encargara de transformar  las variables
binarias en una serie de salidas que representan un segmento que
conforman un display numerico de siete de ellos.
20      //Al unir varios segmentos formamos o representamos un digito.
21      // Dependencies:
22      //
23      // Revision:
24      // Revision 0.01 - File Created
25      // Additional Comments:
26      //
27      //////////////////////////////////////////////////////////////////
28
29
30      module DecoBin_7Seg
31       (
32          input [3:0] i_Deco,
33          output reg [6:0] o_Segmentos
34          );
35
36          always@*
37              begin
38                  case(i_Deco)
39                      4'd0: begin  o_Segmentos <= 7'b0000001; end
40                      4'd1: begin  o_Segmentos <= 7'b1001111; end
41                      4'd2: begin  o_Segmentos <= 7'b0010010; end
42                      4'd3: begin  o_Segmentos <= 7'b0000110; end
43                      4'd4: begin  o_Segmentos <= 7'b1001100; end
44                      4'd5: begin  o_Segmentos <= 7'b0100100; end
45                      4'd6: begin  o_Segmentos <= 7'b0100000; end
46                      4'd7: begin  o_Segmentos <= 7'b0001111; end
47                      4'd8: begin  o_Segmentos <= 7'b0000000; end
48                      4'd9: begin  o_Segmentos <= 7'b0000100; end
```

```
                    4'd10: begin o_Segmentos <= 7'b0001001; end
                    4'd11: begin o_Segmentos <= 7'b1100000; end
                    4'd12: begin o_Segmentos <= 7'b0110001; end
                    4'd13: begin o_Segmentos <= 7'b1000010; end
                    4'd14: begin o_Segmentos <= 7'b0110000; end
                    default : begin  o_Segmentos <= 7'b0111000; end
                endcase
            end
    endmodule
```

Código de test benches

1. test benches del controlador_display_7segmentos

```
    `timescale 1ns / 1ps

    //////////////////////////////////////////////////////////////////////////////////
    // Company: UAZ
    // Engineer: Alfa
    //
    // Create Date: 07.09.2020 10:37:39
    // Design Name:Controlador_display_7Segmentos_tb
    // Module Name: Controlador_display_7Segmentos_tb
    // Project Name: Controlador_display_7Segmentos_tb
    // Target Devices: Basys 3
    // Tool Versions:
    // Description:
    //
    // Dependencies:
    //
    // Revision:
    // Revision 0.01 - File Created
    // Additional Comments:
    //

    //////////////////////////////////////////////////////////////////////////////////


    module Controlador_display_7Segmentos_tb;
        reg [3:0] i_Datos_0;
        reg [3:0] i_Datos_1;
        reg [3:0] i_Datos_2;
        reg [3:0] i_Datos_3;
        reg i_Reloj;
        reg i_Reset;
        wire [6:0] o_Segmentos;
        wire [3:0] o_Anodo_4_Bits;

        controlador_display_7segmentos uut(
         .i_Datos_0(i_Datos_0),
         .i_Datos_1(i_Datos_1),
         .i_Datos_2(i_Datos_2),
```

```
37          .i_Datos_3(i_Datos_3),
38          .i_Reloj(i_Reloj),
39          .i_Reset(i_Reset),
40          .o_Segmentos(o_Segmentos),
41          .o_Anodo_4_Bits(o_Anodo_4_Bits)
42        );
43
44      initial
45      begin
46
47          i_Reset= 1;
48          i_Reloj  = 0;
49          i_Datos_0= 4'd0;
50          i_Datos_1= 4'd0;
51          i_Datos_2= 4'd0;
52          i_Datos_3= 4'd0;
53          #2
54
55          i_Datos_0= 4'd4;
56          i_Datos_1= 4'd2;
57          i_Datos_2= 4'd3;
58          i_Datos_3= 4'd4;
59
60
61           i_Reset = 0;
62
63      end
64      always
65          #1 i_Reloj  = ~i_Reloj ;
66      endmodule
```

2.test benches del multiplexor

```
1       `timescale 1ns / 1ps
2
        //////////////////////////////////////////////////////////////////////////////////
3       // Company:Universidad Autónoma de Zacatecas
4       // Engineers:
5       //Agustin Antonio Palafox Molina
6       //Jose Alfredo Hernadez Dueñas
7       //Jesús Francisco Villaseñor Correa
8       //José Roberto Novoa López
9       //Julio Ángel Pérez Dávila
10      // Create Date: 07.09.2020 10:23:49
11      // Design Name:
12      //Deco_Bin_7Seg
13      // Module Name: Demux4_a_1_tb
14      // Project Name:
15      //Controlador_display_7segmentos
16      // Target Devices:
17      // Tool Versions:2018.1
18      // Description:
```

```
19    //El decodificador  se encargara de transformar  las variables
binarias en una serie de salidas que representan un segmento que
conforman un display numerico de siete de ellos.
20    //Al unir varios segmentos formamos o representamos un digito.
21    // Dependencies:
22    //
23    // Revision:
24    // Revision 0.01 - File Created
25    // Additional Comments:
26    //
27
      //////////////////////////////////////////////////////////////////
////////////////
28    module Demux4_a_1_tb;
29        #(parameter n=4);
30        reg [n-1:0] i_Datos_0;
31        reg [n-1:0] i_Datos_1;
32        reg [n-1:0] i_Datos_2;
33        reg [n-1:0] i_Datos_3;
34        reg [1:0] i_Sel;
35        wire [n-1:0] o_Demux;
36
37       Demux4x1 uut (
38         .i_Datos_0(i_Datos_0),
39         .i_Datos_1(i_Datos_1),
40         .i_Datos_2(i_Datos_2),
41         .i_Datos_3(i_Datos_3),
42         .i_sel(i_sel),
43         .o_Demux(o_Demux)
44       );
45
46
47
48        initial
49          begin
50            i_Datos_0=0;// inicialización de las entradas
51            i_Datos_1=0;
52            i_Datos_2=0;
53            i_Datos_3=0;
54            i_Sel=0;
55
56            #2
57            i_Datos_0= 4'd1; //Se asigna valor a las 4 entradas
58            i_Datos_1= 4'd2;
59            i_Datos_2= 4'd3;
60            i_Datos_3= 4'd4;
61
62            #2 i_Sel = 0; // Se actualiza el selector cada 2 tiempos
63            #2 i_Sel = 1;
64            #2 i_Sel = 2;
65            #2 i_Sel = 3;
66
67         end
68    endmodule
```

3.test benches del Preescalador

```
1     timescale 1ns / 1ps
2
      //////////////////////////////////////////////////////////////////
////////////////
3     // Company: Universidad Autonoma Zacatecas
4     // Engineer: Jorge Manuel de Avila Reveles, Geovanni López Macias,
Gerardo Frías Donlucas, César Rodolfo Salazar Rodríguez
5     //
6     // Create Date: 09.09.2020 10:38:22
7     // Design Name:
8     // Module Name: Preescalador
9     // Project Name: Controlador_Display_7Segmentos
10    // Target Devices: Basys 3
11    // Tool Versions: Vivado 2018.2
12    // Description: Modulo encargado de convertir la señal base de la
tarjeta en una mas pequeña
13    //
14    // Dependencies:
15    //
16    // Revision: 0.2
17    // Revision 0.01 - File Created
18    // Additional Comments:
19    //
20
      //////////////////////////////////////////////////////////////////
////////////////
21
22
23    module Preescalador_Tb;
24        reg i_Reset;
25        reg i_Clk;
26        wire o_Clk_120Hz;
27        Preescalador uut(
28            .i_Reset(i_Reset),
29            .i_Clk(i_Clk),
30            .o_Clk_120Hz(o_Clk_120Hz)
31        );
32        initial
33        begin
34            i_Reset<=1;
35            i_Clk<=0; //inicializa las entradas
36
37            #20 i_Reset<=0; // cuando se activa el reset empieza el
prescalador
38        end
39        always@(*)
40        begin
41            #1 i_Clk <= ~i_Clk; //clk invierte su valor para que el
reloj este en constante cambio
42        end
43
44    endmodule
```

4.test benches del decodificador anillo.

```
1     `timescale 1ns / 1ps
2
      //////////////////////////////////////////////////////////////////////////////////
3     // Company: Universidad Autonoma de Zacatecas
4     // Engineer: Sergio Adad Bernal Adame
5     //
6     // Create Date: 09.09.2020 11:26:32
7     // Design Name:
8     // Module Name: tb_DecoAnillo
9     // Project Name:
10    // Target Devices:
11    // Tool Versions:
12    // Description:
13    //
14    // Dependencies:
15    //
16    // Revision:
17    // Revision 0.01 - File Created
18    // Additional Comments:
19    //
20
      //////////////////////////////////////////////////////////////////////////////////
21
22
23    module DecoAnillo_tb;
24        reg i_Clk;
25        reg i_Reset;
26        wire [1:0] o_Sel;
27        wire [3:0] o_Anodos;
28
29        DecoAnillo uut(
30            .i_Clk(i_Clk),
31            .i_Reset(i_Reset),
32            .o_Sel(o_Sel),
33            .o_Anodos(o_Anodos)
34         );
35        initial
36            begin
37             i_Clk=0;
38             i_Reset=1;
39            #2 i_Reset=0;
40         end
41       always
42       #1 i_Clk=!i_Clk;
43        initial
44            begin
45                #2 i_Clk=4'b0111;
46                #2 i_Clk=4'b1011;
47                #2 i_Clk=4'b1101;
48                #2 i_Clk=4'b1110;
49                #2 i_Clk=2'b00;
50                #2 i_Clk=2'b01;
```

```
51                  #2 i_Clk=2'b10;
52                  #2 i_Clk=2'b11;
53          end
54      endmodule
```

5.test benches del decodificador bcd.

```
1       `timescale 1ns / 1ps
2
        //////////////////////////////////////////////////////////////////////////////////
3       // Company: Universidad Automona de Zacatecas
4       // Engineers:
5       //Agustín Antonio Palafox Molina
6       //José Alfredo Hernadez Dueñas
7       //Jesús Francisco Villaseñor Correa
8       //José Roberto Novoa López
9       //Julio Angel Pérez Dávila
10      // Create Date: 09.09.2020 10:37:24
11      // Design Name:
12      // Module Name:deco_bin_7seg
13      // Project Name:
14      //Controlador_display_7segmentos
15      // Target Devices:
16      // Tool Versions:
17      // Description:
18      //Testbench
19      // Dependencies:
20      //
21      // Revision:
22      // Revision 0.01 - File Created
23      // Additional Comments:
24      //
25
        //////////////////////////////////////////////////////////////////////////////////
26
27
28      module DecoBin_7seg_tb;
29
30          reg [3:0] i_Deco;
31          wire [6:0] o_Segmentos;
32
33          DecoBin_7seg uut(
34              .i_Deco(i_Deco),
35              .o_Segmentos(o_Segmentos)
36          );
37
38          initial
39              begin
40                  i_Deco = 0;
41
42                  #2 i_Deco = 1;
43                  #2 i_Deco = 2;
44                  #2 i_Deco = 3;
```

```
45                  #2 i_Deco = 4;
46                  #2 i_Deco = 5;
47                  #2 i_Deco = 6;
48                  #2 i_Deco = 7;
49                  #2 i_Deco = 8;
50                  #2 i_Deco = 9;
51                  #2 i_Deco = 10;
52                  #2 i_Deco = 11;
53                  #2 i_Deco = 12;
54                  #2 i_Deco = 13;
55                  #2 i_Deco = 14;
56                  #2 i_Deco = 15;
57              end
58      endmodule
```

RESTRINCIONES.

```
## Clock signal
set_property PACKAGE_PIN W5 [get_ports i_Reloj]

      set_property IOSTANDARD LVCMOS33 [get_ports i_Reloj]
      create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5}
[get_ports i_Reloj]

 ## Switches
set_property PACKAGE_PIN V17 [get_ports {i_Datos_0[0]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_0[0]}]
set_property PACKAGE_PIN V16 [get_ports {i_Datos_0[1]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_0[1]}]
set_property PACKAGE_PIN W16 [get_ports {i_Datos_0[2]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_0[2]}]
set_property PACKAGE_PIN W17 [get_ports {i_Datos_0[3]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_0[3]}]
set_property PACKAGE_PIN W15 [get_ports {i_Datos_1[0]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_1[0]}]
set_property PACKAGE_PIN V15 [get_ports {i_Datos_1[1]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_1[1]}]
set_property PACKAGE_PIN W14 [get_ports {i_Datos_1[2]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_1[2]}]
set_property PACKAGE_PIN W13 [get_ports {i_Datos_1[3]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_1[3]}]
set_property PACKAGE_PIN V2 [get_ports {i_Datos_2[0]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_2[0]}]
set_property PACKAGE_PIN T3 [get_ports {i_Datos_2[1]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_2[1]}]
```

```
set_property PACKAGE_PIN T2 [get_ports {i_Datos_2[2]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_2[2]}]
set_property PACKAGE_PIN R3 [get_ports {i_Datos_2[3]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_2[3]}]
set_property PACKAGE_PIN W2 [get_ports {i_Datos_3[0]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_3[0]}]
set_property PACKAGE_PIN U1 [get_ports {i_Datos_3[1]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_3[1]}]
set_property PACKAGE_PIN T1 [get_ports {i_Datos_3[2]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_3[2]}]
set_property PACKAGE_PIN R2 [get_ports {i_Datos_3[3]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_3[3]}]
##7 segment display
set_property PACKAGE_PIN W7 [get_ports {o_Segmentos[6]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {o_Segmentos[6]}]
set_property PACKAGE_PIN W6 [get_ports {o_Segmentos[5]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {o_Segmentos[5]}]
set_property PACKAGE_PIN U8 [get_ports {o_Segmentos[4]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {o_Segmentos[4]}]
set_property PACKAGE_PIN V8 [get_ports {o_Segmentos[3]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {o_Segmentos[3]}]
set_property PACKAGE_PIN U5 [get_ports {o_Segmentos[2]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {o_Segmentos[2]}]
set_property PACKAGE_PIN V5 [get_ports {o_Segmentos[1]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {o_Segmentos[1]}]
set_property PACKAGE_PIN U7 [get_ports {o_Segmentos[0]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {o_Segmentos[0]}]

#set_property PACKAGE_PIN V7 [get_ports dp]

      #set_property IOSTANDARD LVCMOS33 [get_ports dp]

set_property PACKAGE_PIN U2 [get_ports {o_Anodo_4_Bits[3]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {o_Anodo_4_Bits[3]}]
set_property PACKAGE_PIN U4 [get_ports {o_Anodo_4_Bits[2]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {o_Anodo_4_Bits[2]}]
set_property PACKAGE_PIN V4 [get_ports {o_Anodo_4_Bits[1]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {o_Anodo_4_Bits[1]}]
set_property PACKAGE_PIN W4 [get_ports {o_Anodo_4_Bits[0]}]

      set_property IOSTANDARD LVCMOS33 [get_ports {o_Anodo_4_Bits[0]}]
```

```
##Buttons
set_property PACKAGE_PIN U18 [get_ports i_Reset]


      set_property IOSTANDARD LVCMOS33 [get_ports i_Reset ]
```