UNIVERSIDAD AUTÓNOMA DE ZACATECAS

“Francisco García Salinas”

UNIDAD ACADÉMICA DE INGENIERÍA ELÉCTRICA

PROGRAMA ACADÉMICO: INGENIERÍA EN ROBÓTICA Y MECATRÓNICA

SISTEMAS DIGITALES III

Docente: Dr. Remberto Sandoval Arechiga

Reporte Controlador de display 7 segmentos

Equipo 4, Robóticos:

Nelson Eduardo Coronado Gamez

Sergio Adad Bernal Adame

Clara Veronica Guerrero Correa

Hilda Berenice Espinosa Herrera

Guillermo Cruz Fernandez

7“B”

Fecha: 27/09/2020

Resumen.

El proyecto consiste en crear un driver para cuatro display de 7 segmentos en verilog para poder implementarlo en la tarjeta basys 3.

Driver: Es un controlador de dispositivo o manejador de dispositivo (en inglés: driver). Es un programa informático que permite al sistema operativo interactuar con un periférico, haciendo una abstracción del hardware y proporcionando una interfaz (posiblemente estandarizada) para utilizar el dispositivo. Es una pieza esencial del software, sin la cual el hardware sería inutilizable.

El driver es capaz de mostrar un dígito (hexadecimal) en cada display, el dígito lo ingresamos por medio de los switches de la basys, cada cuatro switchs corresponden a uno de los cuatro display, así podremos representar cada una de las 16 combinaciones, además cuenta con un reset (botón).

El proyecto se creó en base a la metodología de diseño, primero se diseñó la arquitectura (caja negra, caja blanca, submódulos) del driver, posteriormente se realizó la descripción (en lenguaje verilog) en Vivado, se realizaron las simulaciones para observar el correcto funcionamiento de cada parte y finalmente se llevó a cabo la implementación en la basys.

Índice.

Introducción

Para poder observar los números en hexadecimal se usaron cuatro displays de 7 segmentos incluidos en la tarjeta basys 3 (plataforma de desarrollo).

El display de 7 segmentos es un dispositivo electrónico que se utiliza para representar visualmente números y algunos caracteres. Existen dos tipos de display de 7 segmentos, su principal diferencia es la conexión que debemos implementar para encenderlos, estos dos tipos se conocen como Ánodo común y Cátodo común.

En la basys 3 se usan los de tipo ánodo común (imagen 1), todos los ánodos de los ledes o segmentos (a-g) están unidos internamente a una patilla común que debe ser conectada a potencial positivo (1). El encendido de cada segmento individual se realiza aplicando potencial negativo (0) por la patilla correspondiente a través de una resistencia que limite el paso de la corriente.

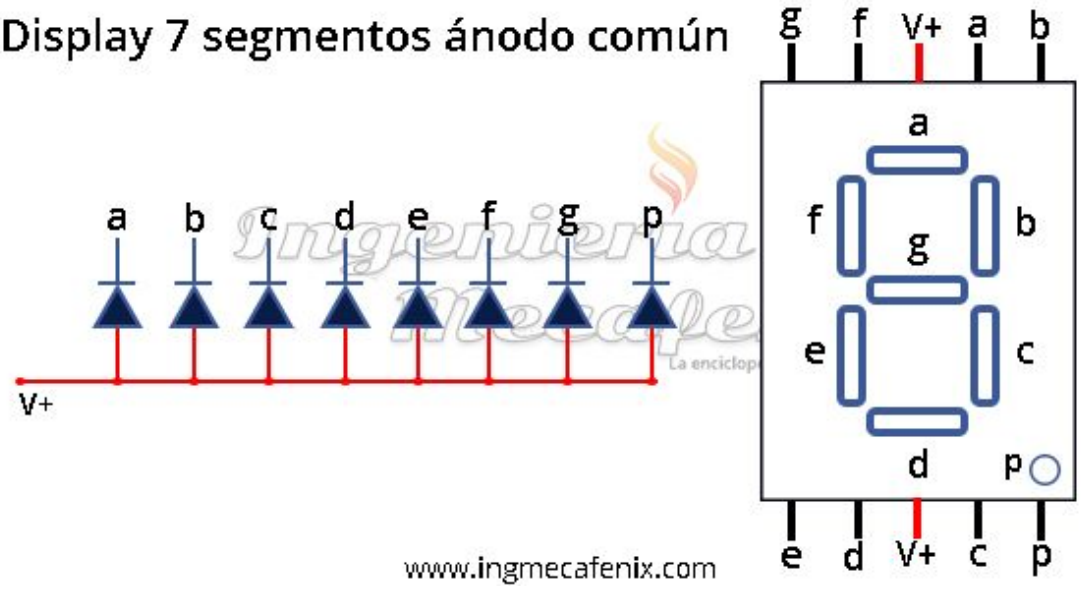

Imagen 1. Display de 7 segmentos anodo comun.

para el controlador de display se necesitan 6 entradas y 2 salidas para poder mostrar los números en hexadecimal. a continuación se describen en la (Tabla 1).

Nombre de la señal	Dirección	Tamaño (Bits)	Descripción
i_Datos_0	Entrada	4	Datos de entrada para el display 0. Se emplea un formato binario donde se representan los números del 0 al 15 (del 0x0 al 0xF).
i_Datos_1	Entrada	4	Datos de entrada para el display 1. Se emplea un formato binario donde se representan los números del 0 al 15 (del 0x0 al 0xF).
i_Datos_2	Entrada	4	Datos de entrada para el display 2. Se emplea un formato binario donde se representan los números del 0 al 15 (del 0x0 al 0xF).
i_Datos_3	Entrada	4	Datos de entrada para el display 3. Se emplea un formato binario donde se representan los números del 0 al 15 (del 0x0 al 0xF).
i_Reloj	Entrada	1	Señal de referencia temporal con una frecuencia de 100 MHz. De esta señal se deriva la frecuencia de selección de escritura en cada display (30Hz), en un total de 120Hz, para los 4 displays en conjunto.
i_Reset	Entrada	1	Señal que establece en el sistema un estado inicial.
o_Segmentos	Salida	7	Señal para controlar el encendido o apagado de los segmentos de cada display.
o_Anodo_4_Bits	Salida	4	Señal para controlar el display a escribir los datos. Con una frecuencia de 120 HzDonde para seleccionar cada display el bit correspondiente se coloca en bajo.

Tabla 1. Entradas y salidas de Controlador_display_7segmentos.

Uno de los objetivos generales de esta práctica es conocer la metodología de diseño de sistemas digitales.

Para tener éxito en la realización de este proyecto o en cualquier otro proyecto de descripción de hardware se tiene que seguir una metodología bastante sencilla, pero de suma importancia para evitar errores (sobre todo en las declaraciones de entradas, salidas y cables internos).

Primer paso: Diseñar la caja negra de nuestro proyecto (nombre del proyecto, entradas, salidas y tamaño en bits de cada una de ellas).

Es de gran utilidad realizar una tabla donde coloquemos los datos anteriores y además una descripción clara de cada elemento para facilitarnos la existencia a la hora de realizar los submódulos. Esto aplica para toda la arquitectura del proyecto (caja negra, caja blanca y submódulos).

Segundo paso: Diseñar la caja blanca, es decir, todo lo que llevará dentro la caja negra (bloques (entradas y salidas) y conexiones internas)

Al realizar un proyecto dividido en sub módulos, permite que la realización de este se vuelva más sencilla, ya que cada uno tiene una función específica, logrando en conjunto la función principal del proyecto.

Requerimientos

Como este proyecto consiste en el control de displays en la tarjeta basys 3, es necesario tener dicha tarjeta; esta tarjeta tiene 4 displays con los cuales podemos trabajar, para controlar el display que estemos utilizando, es necesaria una salida de 4 bits, un bit por cada display, que controle en cual display estamos mostrando el dato. A su vez necesitamos una salida de 7 bits, la cual cada bit será un segmento de display, estos segmentos dependiendo de si se encuentran encendidos o apagados podremos mostrar desde el dígito 0 hasta el 15 en hexadecimal.

Así como se tiene 4 displays también se necesita la entrada de 4 datos, estos datos serán de 4 bits para así poder representar desde el dígito 0 hasta el 15; una señal de reset que me permita reiniciar a cero el proyecto, y una señal de reloj que controle el cambio en los displays.

Internamente se necesita de un submódulo que adecue la señal de reloj a la frecuencia que trabaja la basys (Preescalador), también se necesita un bloque que me permita la salida del dato que será mostrado en el display correspondiente. Un bloque que controle el encendido y apagado de los displays y por último un decodificador que me codifique el dato seleccionado a la salida de 7 segmentos para su correcta representación.

Arquitectura

Controlador_display_7segmentos.

El controlador de display de 7 segmentos consiste en traducir datos de formato binario a código de 7 segmentos, selecciona el display en el cual se va a mostrar su dato correspondiente, esto quiere decir que el i_Datos_0 corresponde al display 0.

En la imagen 2 se puede observar todas las entradas y salidas necesarias de nuestro proyecto, estas señales son necesarias para obtener los resultados esperados (controlar el display y mostrar datos en él).

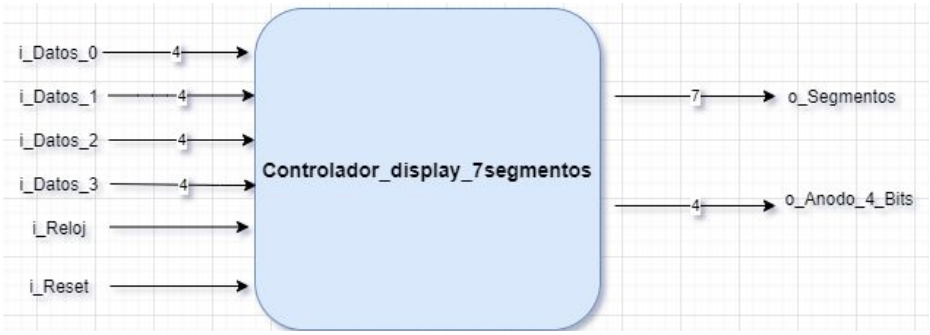

Imagen 2. Caja negra de Controlador_display_7segmentos.

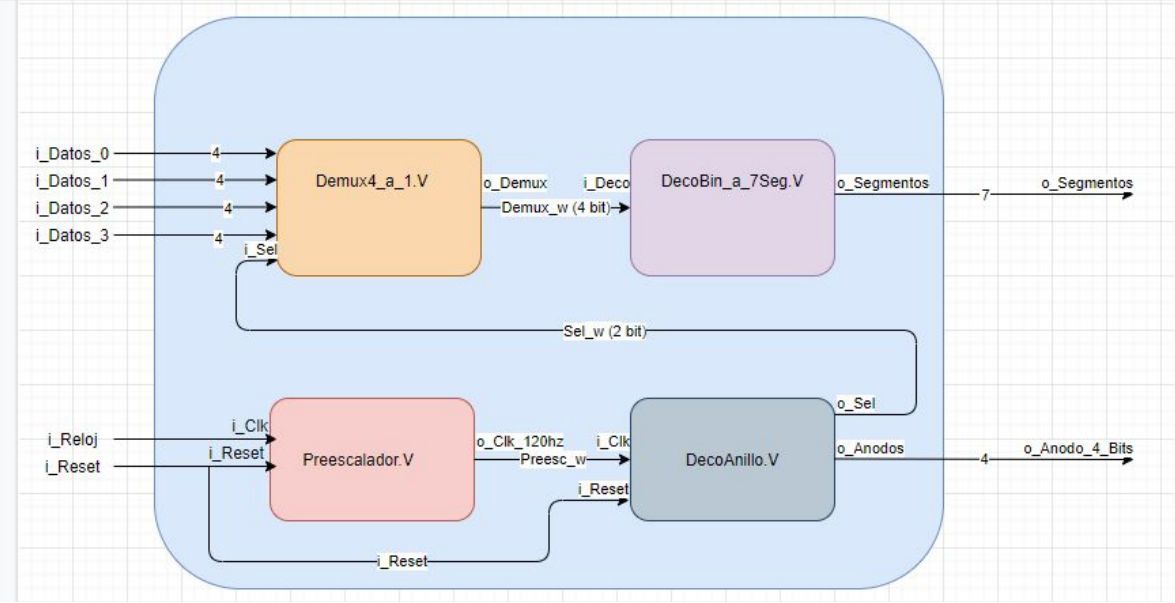

Imagen 3. Caja blanca de Controlador_display_7segmentos.

Para el correcto funcionamiento del controlador es necesario utilizar los bloques (submódulos) que se muestran en la Imagen 3 ya que cada uno de ellos desarrolla una funcionalidad específica para que el controlador opere de la manera adecuada, también se observa las conexiones necesarias entre ellos.

En la siguiente tabla (Tabla 1)se describe las entradas y salidas de nuestro controlador, así como una explicación de su funcionamiento o para que se utiliza.

Nombre de la señal	Dirección	Tamaño (Bits)	Descripción
i_Datos_0	Entrada	4	Datos de entrada para el display 0. Se emplea un formato binario donde se representan los números del 0 al 15 (del 0x0 al 0xF).
i_Datos_1	Entrada	4	Datos de entrada para el display 1. Se emplea un formato binario donde se representan los números del 0 al 15 (del 0x0 al 0xF).
i_Datos_2	Entrada	4	Datos de entrada para el display 2. Se emplea un formato binario donde se representan los números del 0 al 15 (del 0x0 al 0xF).
i_Datos_3	Entrada	4	Datos de entrada para el display 3. Se emplea un formato binario donde se representan los números del 0 al 15 (del 0x0 al 0xF).
i_Reloj	Entrada	1	Señal de referencia temporal con una frecuencia de 100 MHz. De esta señal se deriva la frecuencia de selección de escritura en cada display (30Hz), en un total de 120Hz, para los 4 displays en conjunto.
i_Reset	Entrada	1	Señal que establece en el sistema un estado inicial.
o_Segmentos	Salida	7	Señal para controlar el encendido o apagado de los segmentos de cada display.
o_Anodo_4_Bits	Salida	4	Señal para controlar el display a escribir los datos. Con una frecuencia de 120 HzDonde para seleccionar cada display el bit correspondiente se coloca en bajo.

Tabla 1. Entradas y salidas de Controlador_display_7segmentos.

a) Demux4_a_1: En la Imagen 4 se observa un multiplexor de 4 a 1, este bloque se encarga de seleccionar el dígito que se mostrará en el ánodo seleccionado en decodificador de anillo.

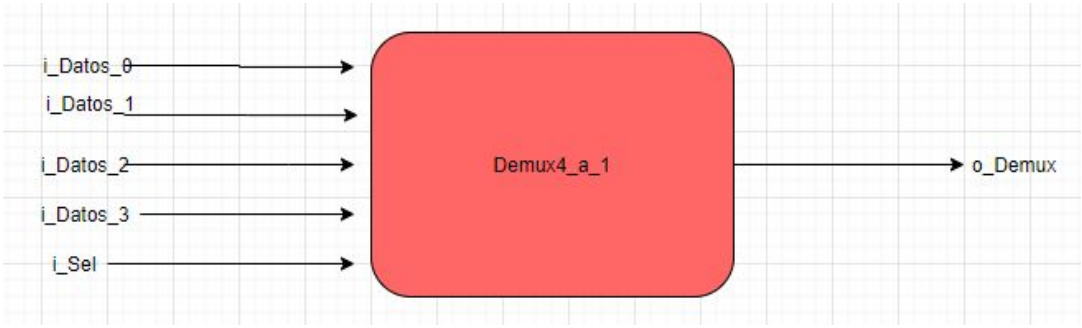

Imagen 4. Caja negra de Demux4_a_1.

En la Tabla 2 se encuentra una breve explicación de las entradas y salidas utilizadas en este bloque.

Nombre de la señal	Direccion	Tamaño(bits)	Descripción
i_Datos_0	entrada	4	Datosde entrada del display 0. Se emplea en formato binario, donde se representanlos números del 0 al 15(0x0 al 0xF)
i_Datos_1	entrada	4	Datosde entrada del display 1. Se emplea en formato binario, donde se representanlos números del 0 al 15(0x0 al 0xF)
i_Datos_2	entrada	4	Datos de entrada del display 2. Se emplea en formato binario, donde se representanlos números del 0 al 15(0x0 al 0xF)
i_Datos3	entrada	4	Datos de entrada del display 3. Se emplea en formato binario, donde se representanlos números del 0 al 15(0x0 al 0xF)
i_Sel	entrada	2	Señal que se encarga de seleccionar cual de los 4 datos de entrada se mandara a la salida
o_Demux	salida	4	Entregara un dato de entre los 4 datos de entrada,que será seleccionado por la entrada de selector

Tabla 2. Entradas y salidas de Demux4_a_1.

b) Preescalador: En la imagen 5 se puede observar la caja negra del preescalador, este bloque convierte la señal de referencia tiempo Clk de 100MHz a 120 Hz, esto se logra cambiando el estado de la señal cada 41666 ciclos positivos y negativos. Y en la Tabla 3, se encuentran las entradas y salidas de este bloque, así como una breve explicación de éstas.

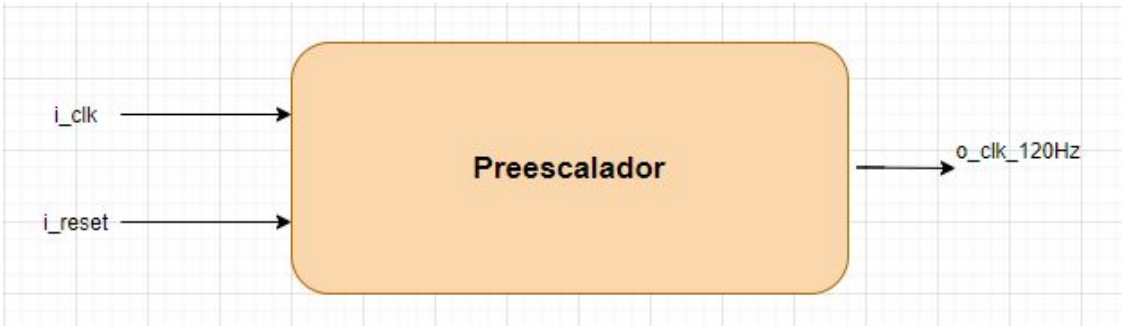

Imagen 5. Caja negra Preescalador.

Nombre de la señal	Dirección	Número de bits	Descripción
i_Clk	Entrada	1	Señal de referencia de tiempo de 100MHz
i_Reset	Entrada	1	Señal que reinicia el prescalador al estado inicial 0
o_Clk_120Hz	Salida	1	Señal de salida modificada a 120Hz

Tabla 3. Entradas y salidas de Preescalador.

c) **DecoAnillo:** La Imagen 6 nos muestra la caja negra de el Decodificador de anillo, su funcionalidad se basa en que cada haya un cambio en la señal i_Clk, cambiará la salida o_Anodos, esto quiere decir que que i_Clk cambie va a cambiar el display en el que se va a mostrar el dato. De la misma señal i_Clk se obtendrá una segunda salida que al igual que la de 0_Anodos, cambiará su valor en cada cambio de i_Clk. En la Tabla 4 se muestra las entradas y salidas de este bloque y una breve explicación de éstas.

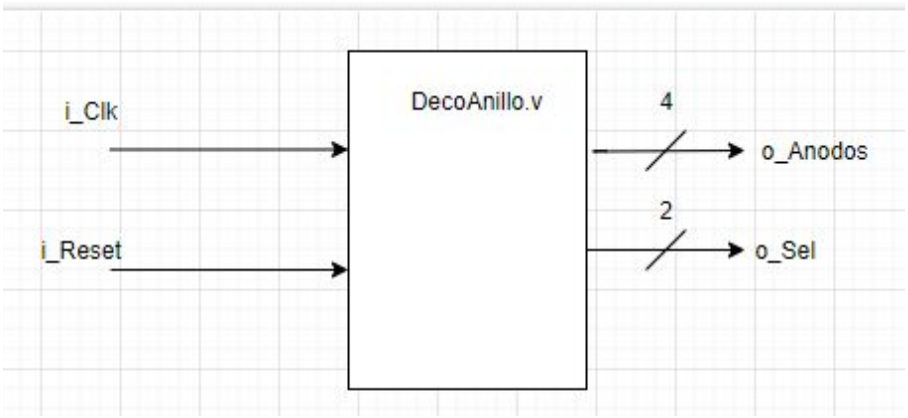

Imagen 6. Caja negra de DecoAnillo.

Nombre de la señal	Direccion	Tamaño (bits)	Descripción
i_Reset	Entrada	1	Señal que reinicia el proceso a su estado 0
o_Anodos	Salida	4	Señal de salida que selecciona el display en el cual se va a mostrar nuestro dato. Con una frecuencia de 120 HzDonde para seleccionar cada display el bit correspondiente se coloca en bajo.
o_Sel	Salida	2	Señal de salida que será conectada a la entrada del multiplexor para poder selccionar un dato.
i_Clk	Entrada	1	Señal de referencia de tiempo de 100 MHz

Tabla 4. Entradas y salidas de DecoAnillo.

d) **Deco_Bin_7seg:** Este bloque que se muestra en la Imagen 7, recibe una entrada de bits que representa un número BCD y a la salida se obtiene la representación en 7 bits, cada uno de esos bits representa un segmento del display. La señal de salida o_segementos será la encargada de representar en el display el formato hexadecimal de nuestro dato de entrada. En la Tabla 5, que se encuentra enseguida de la imagen 7, se encuentran las descripciones necesarias para este bloque.

Imagen 7. Caja negra de Deco_Bin_7seg.

Nombre de la señal	Dirección	Tamaño	Descripción
i_deco	Entrada	4	La señal de entrada i_deco es un dato de entrada proveniente del Demux4_a_1.V, Se emplea en formato binario, donde se representan los numeros del 0 al 15 (0 a F)
o_segmentos	Salida	7	La señal de salida es la encargada de representar los datos decodificados de formato hexadecimal. Se emplea en formato binario,donde se representan los numeros del 0 al 15(0 a F
o_display	Salida	4	La señal de salida es la encargada de elegir el display segun el tamaño que se ocupe el digito para representarlo.

Tabla 5. Entradas y salidas de Deco_Bin_7seg.

Implementación.

En la Imagen 8 se puede observar la tarjeta basys 3 que se utilizó en este proyecto, los switches con los que se pueden ingresar los datos que se van a mostrar en los displays, y los botones que utilizamos.

Para el correcto funcionamiento de los switches 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 en la basys.

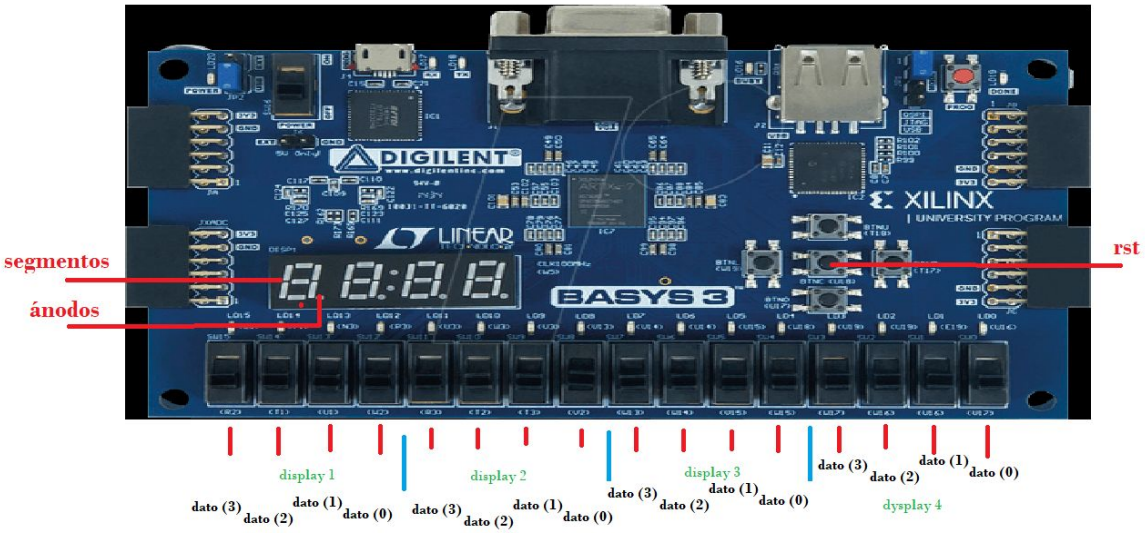

Imagen 8. Implementación en la tarjeta basys.

Pruebas

En las simulaciones de Controlador Diaplay_7segmentos (Imagen 9 y 10) desde el tiempo cero hasta el tiempo dos los valores de los datos son cero, después cuando t=2 cambian los valores, y los segmentos de los segementos se encuentran encendidos o apagados según el valor que se va a representar.

Imagen 9. Simulación de Controlador_display_7segmentos en hexadecimal.

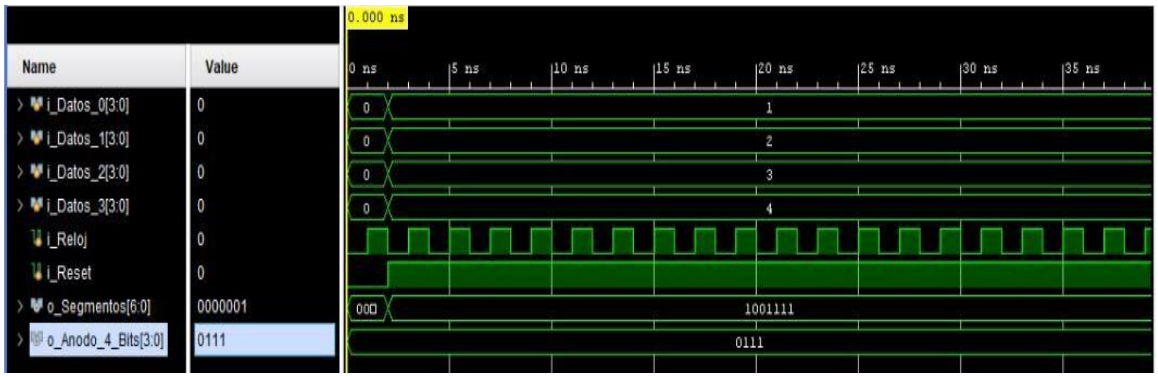

Imagen 10. Simulación de Controlador_display_7segementos en binario.

En la simulación de Demux4_a_1 (imagen 11), del tiempo 0 al 4 el valor de i_Sel es 00, por lo tanto el dato seleccionado y que se va a mostrar en la salida será el i_Dato1; de t=4 a t=6 i_Sel=01 por lo tanto el dato mostrado en la salida es el valor de i_Dato2; de t=6 a t=8 i_Sel=10 por lo tanto el dato mostrado en la salida es el valor de i_Dato3; de t=8 en adelante i_Sel=11 por lo tanto el dato mostrado en la salida es el valor de i_Dato4.

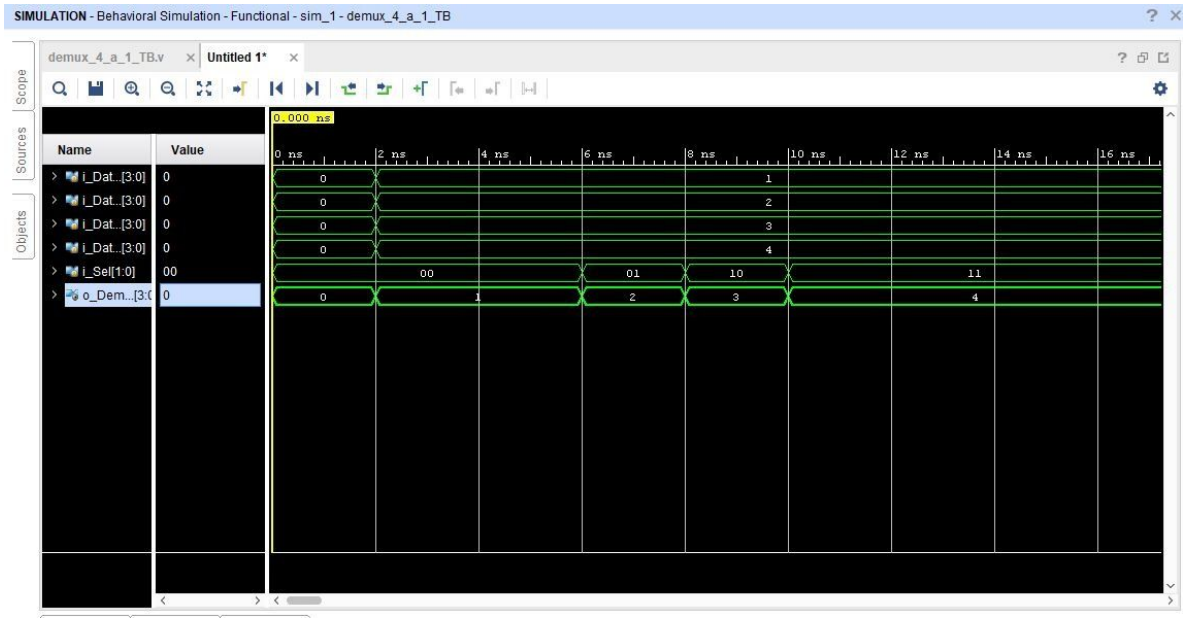

Imagen 11. Simulación de Demux4_a_1..

En el Preescalador (imagen 12) toma la frecuencia del reloj temporizador y la acondiciona antes de alimentar el temporizador, en la imagen se muestra como la señal clk es un pulso de reloj constante mientras que la de clk_20hz es un pulso de señal de reloj que se pone en alto cada 20hz.Reset es utilizado para activar/ desactivar el reloj.

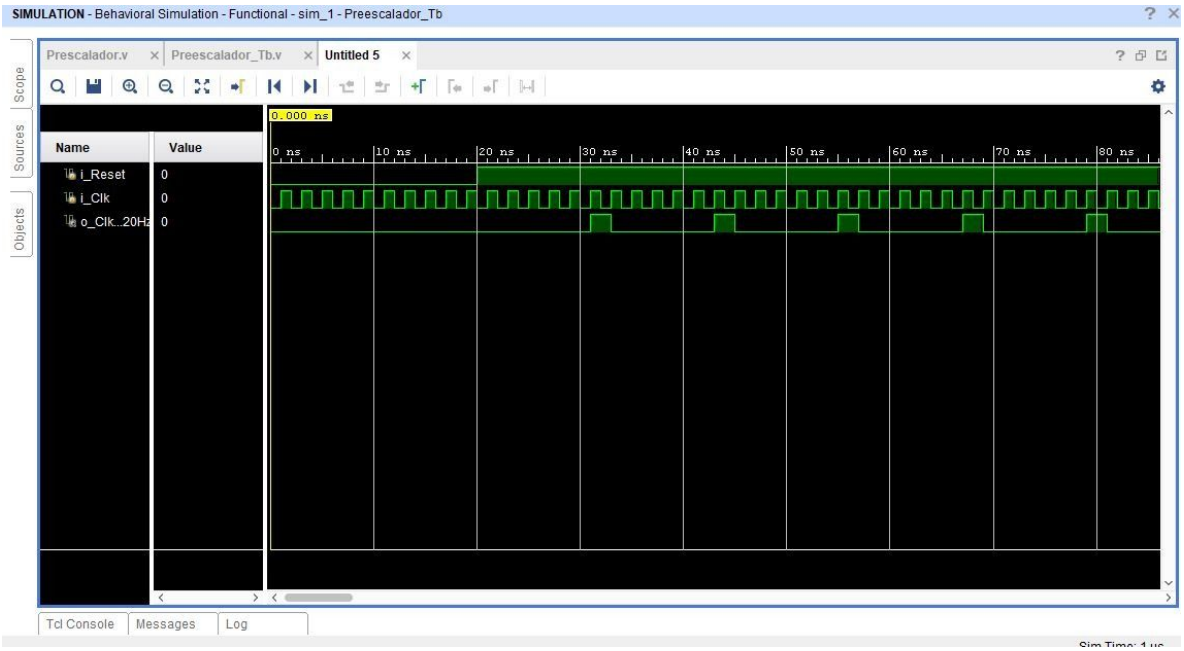

Imagen 12. Simulación de Preescalador.

En la simulación de DecoAnillo (imagen 13) se muestra como la señal de salida de los ánodos y el selector cambian con respecto a los pulsos del reloj y cuando el reset se activa el proceso vuelve al estado inicial.

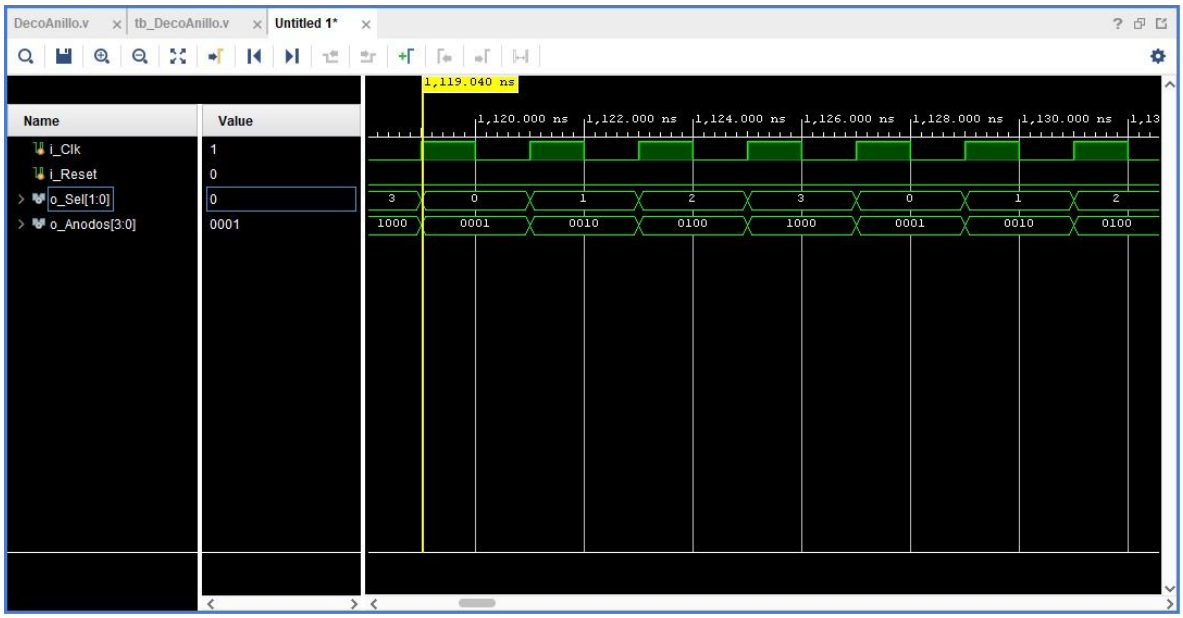

Imagen 13. Simulación de DecoAnillo.

En la simulacion de Deco_Bin_7Seg (imagen 14) se recibe una señal en decimal la cual la convierte en número binario y la combinación de varias señales de entrada activa varias señales de salida.

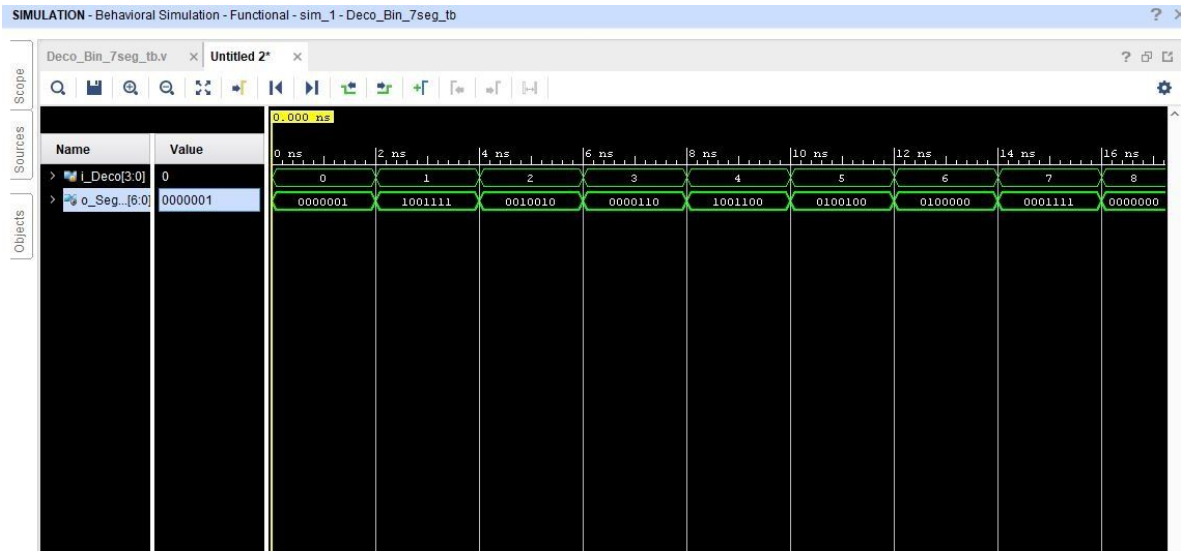

Imagen 14. Simulación de Deco_Bin_7seg.

Análisis de resultados.

La imagen 15 muestra la cantidad de recursos que el controlador está usando y los disponibles, también muestra una gráfica donde se puede observar el porcentaje de recursos que se están utilizando, el mayor porcentaje es el de IO (entradas/salidas), aun así nos quedan bastantes recursos para hacer un proyecto mucho más grande o con más capacidades.

Resource	Utilization	Available	Utilization %
LUT	13	20800	0.06
FF	6	41600	0.01
IO	29	106	27.36

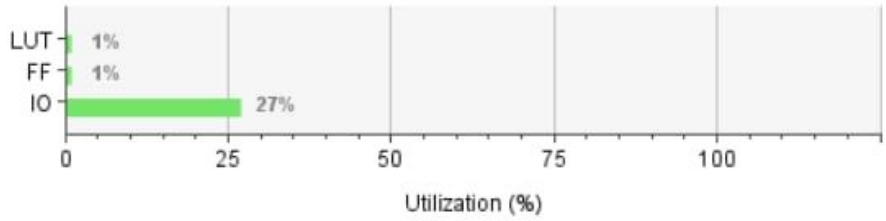

Imagen 15. Recursos y porcentajes de utilización del controlador en el FPGA Artix-7.

La imagen 16 muestra el reporte de utilización, como se puede observar se están utilizando un mínimo de registros y LUTs, respecto a las entradas y salidas (IO) se utilizan 29 de las 106 disponibles.

Name	Slice LUTs (20800)	Slice Registers (41600)	Bonded IOB (106)	BUFGCTRL (32)
Controlador_display_7se...	13	6	29	1
preesc (Preescalador)	2	4	0	0
dec_anillo (DecoAnillo)	11	2	0	0

Imagen 16. Reporte de utilización del controlador en el FPGA Artix-7.

Conclusiones

Si una persona normal sin conocimientos sobre programación o descripción de hardware observa el funcionamiento del proyecto Controlador de Display le parecerá bastante sencillo, ya que lo único que hace es mostrar la secuencia de números hexadecimales (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F), sin embargo, este sencillo funcionamiento es más complejo de lo que parece, siempre se debe seguir la metodologia de diseño para facilitarnos su desarrollo e implementación.

Por otra parte, algunos detalles importantes que se deben tener en cuenta en este proyecto son:

Se debe decidir si los displays van a activarse por cátodo o ánodo común, ya que si usamos una configuración de encendido en los segmentos por cátodo y estamos usando ánodos nos causará un error en la visualización de los números en el display.

Error en la frecuencia. Según el datasheet de la tarjeta basys 3 opera a una frecuencia de 100MHz, si se trabaja con esta frecuencia el ojo humano no alcanza a percibir los números en el display, esto es porque varía tan rápido que el segmento no alcanza a encender por completo. Para corregir esto se usa un preescalador, su función será cambiar 100MHz a 120Hz, esto se logra cambiando el estado de la señal cada 41666 ciclos positivos y 41666 ciclos negativos

Referencias

[1] Pertenece Samir Palnitkar (2003, February 21), “Verilog HDL: A Guide to Digital Design and Synthesis” Second Edition.

[2] Pertenece Stephen Brown and Zvonko Vranesic (-), “Fundamentals of Digital Logic with Verilog Design” Third edition.

[3] Pertenece JAMES E. PALMER DAVID E. PERLMAN (-), “INTRODUCCIÓN A LOS SISTEMAS DIGITALES”.

Apéndices.

Apéndice A.

Tablas de verdad.

Las tablas de verdad, son tablas que nos muestran la relación entre las entradas de un sistema y sus salidas, nos muestran las posibles combinaciones de entrada para obtener determinada salida.

i_Sel_bit1	i_Sel_bit2	datos
0	0	i_Datos_0
0	1	i_Datos_1
1	0	i_Datos_2
1	1	i_Datos_3

Tabla 6. Tabla de verdad de Demux4_a_1.

Apéndice B.

Tablas con información extra.

Tanto en la tabla 7 como en la 8, se muestra relación de pulsos de reloj con las salidas o_Anodos y o_Sel, según el número de pulso la salida tendrá un valor diferente, y al llegar al pulso tres la secuencia de salida vuelve a iniciar.

Pulso de reloj	o_Anodos
0	0001
1	0010
2	0100
3	1000

Tabla 7. Relación de pulsos con salida o_Anodos.

Pulsos de reloj	o_Sel
0	00
1	01
2	10
3	11

Tabla 8. Relación de pulsos y salida o_Sel.

En la Tabla 9 nos muestra la relación número- segmento de display, esto quiere decir que dependiendo el número que se quiera mostrar los segmentos se encuentran encendidos o apagados para poder mostrar el dato. A su vez en la Imagen se muestra el acomodo en el que se encuentran los segmentos del display.

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	1
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

Tabla 9. Combinaciones de código hexadecimal.

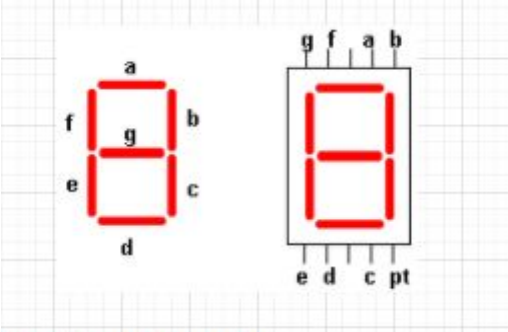

Imagen 17. Diagrama de display 7 segmentos.

Código de implementación

a) Código Controlador_display_7segementos.

```
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.      Demux4_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.       Preescalador preesc (
59.       .i_Clk(i_Reloj),
60.       .i_Reset(i_Reset),
61.       .o_Clk_120Hz(Preesc_w)
62.       );
63.
64.endmodule
```

b) Código Demux4_a_1.

```
1. `timescale 1ns / 1ps
2. //////////////////////////////////////////////////////////////////////////////
   //////////////
3. // Company:
4. // Engineer:
5. //
6. // Create Date: 14.09.2020 09:31:11
7. // Design Name:
8. // Module Name: demux_4_a_1
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 Demux4_a_1
24.#(parameter n=4)
```

```
(
input [n-1:0] i_Datos_0,
input [n-1:0] i_Datos_1,
input [n-1:0] i_Datos_2,
input [n-1:0] i_Datos_3,
input [1:0] i_Sel, // entrada selectora del mux
output reg [n-1:0] o_Demux // salida
);
always @*
begin
      case(i_Sel)
            2'b00: begin o_Demux <= i_Datos_0; end
            2'b01: begin o_Demux <= i_Datos_1; end
            2'b10: begin o_Demux <= i_Datos_2; end
            default : begin o_Demux <= i_Datos_3; end
      endcase
end
endmodule
```

c) Preescalador.

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 14.09.2020 09:34:00
// Design Name:
// Module Name: Prescalador
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Preescalador #(parameter lim=5) ( // lim=416666) (
input i_Clk,
input i_Reset,
output reg o_Clk_120Hz
```

```
27.);
28.reg [19:0] cuent; // registro para almacenar la cuenta interna del
   preescalador
29.
30.always @ ( posedge i_Clk, negedge i_Reset)
31.
32.if (~i_Reset)
33.begin
34.cuent <= 0;
35.o_Clk_120Hz <= 0;
36.end
37.else if (cuent < lim) //aumenta la cuenta en uno mintras no llegue al
   limite
38. begin
39.cuent <= cuent + 1'd1;
40.o_Clk_120Hz <= 0;
41.end
42.
43.else // manda un pulso a la salida cuando llega al limite y la cuenta
   regresa a cero
44.begin
45.cuent <= 0;
46.o_Clk_120Hz <= 1;
47.end
48.endmodule
```

d) DecoAnillo

```
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
```

e) Deco_Bin_7seg.

```
1. `timescale 1ns / 1ps
2. //////////////////////////////////////////////////////////////////////////////////
3. // Company:
4. // Engineer:
5. //
6. // Create Date: 14.09.2020 09:40:57
7. // Design Name:
8. // Module Name: Deco_Bin_7Seg
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 DecoBin_7Seg
24. (
25.input [3:0] i_Deco,
26.output reg [6:0] o_Segmentos
27.);
28.
29.always@*
30.      begin
31.case(i_Deco)
32.4'd0: begin  o_Segmentos <= 7'b0000001; end
33.4'd1: begin  o_Segmentos <= 7'b1001111; end
34.4'd2: begin  o_Segmentos <= 7'b0010010; end
35.4'd3: begin  o_Segmentos <= 7'b0000110; end
36.4'd4: begin  o_Segmentos <= 7'b1001100; end
37.4'd5: begin  o_Segmentos <= 7'b0100100; end
38.4'd6: begin  o_Segmentos <= 7'b0100000; end
39.4'd7: begin  o_Segmentos <= 7'b0001111; end
40.4'd8: begin  o_Segmentos <= 7'b0000000; end
41.4'd9: begin  o_Segmentos <= 7'b0000100; end
42.4'd10: begin o_Segmentos <= 7'b0001001; end
43.4'd11: begin o_Segmentos <= 7'b1100000; end
44.4'd12: begin o_Segmentos <= 7'b0110001; end
```

```
45.4'd13: begin o_Segmentos <= 7'b1000010; end
46.4'd14: begin o_Segmentos <= 7'b0110000; end
47.default : begin  o_Segmentos <= 7'b0111000; end
48.endcase
49.end
50.endmodule
```

f) Código de restricciones.

```
1 ## This file is a general .xdc for the Basys3 rev B board
2 ## To use it in a project:
3 ## - uncomment the lines corresponding to used pins
4 ## - rename the used ports (in each line, after get_ports) according
to the top level signal names in the project
5
6 ## Clock signal
7 set_property PACKAGE_PIN W5 [get_ports i_Reloj]
8 set_property IOSTANDARD LVCMOS33 [get_ports i_Reloj]
9 create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5}
[get_ports i_Reloj]
10
11 ## Switches
12 set_property PACKAGE_PIN V17 [get_ports {i_Datos_0[0]}]
13 set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_0[0]}]
14 set_property PACKAGE_PIN V16 [get_ports {i_Datos_0[1]}]
15 set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_0[1]}]
16 set_property PACKAGE_PIN W16 [get_ports {i_Datos_0[2]}]
17 set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_0[2]}]
18 set_property PACKAGE_PIN W17 [get_ports {i_Datos_0[3]}]
19 set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_0[3]}]
20 set_property PACKAGE_PIN W15 [get_ports {i_Datos_1[0]}]
21 set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_1[0]}]
22 set_property PACKAGE_PIN V15 [get_ports {i_Datos_1[1]}]
23 set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_1[1]}]
24 set_property PACKAGE_PIN W14 [get_ports {i_Datos_1[2]}]
25 set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_1[2]}]
26 set_property PACKAGE_PIN W13 [get_ports {i_Datos_1[3]}]
27 set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_1[3]}]
28 set_property PACKAGE_PIN V2 [get_ports {i_Datos_2[0]}]
29 set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_2[0]}]
30 set_property PACKAGE_PIN T3 [get_ports {i_Datos_2[1]}]
31 set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_2[1]}]
32 set_property PACKAGE_PIN T2 [get_ports {i_Datos_2[2]}]
33 set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_2[2]}]
34 set_property PACKAGE_PIN R3 [get_ports {i_Datos_2[3]}]
35 set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_2[3]}]
```

```
36 set_property PACKAGE_PIN W2 [get_ports {i_Datos_3[0]}]
37 set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_3[0]}]
38 set_property PACKAGE_PIN U1 [get_ports {i_Datos_3[1]}]
39 set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_3[1]}]
40 set_property PACKAGE_PIN T1 [get_ports {i_Datos_3[2]}]
41 set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_3[2]}]
42 set_property PACKAGE_PIN R2 [get_ports {i_Datos_3[3]}]
43 set_property IOSTANDARD LVCMOS33 [get_ports {i_Datos_3[3]}]
44
45
46 ## LEDs
47 #set_property PACKAGE_PIN U16 [get_ports {led[0]}]
48 #set_property IOSTANDARD LVCMOS33 [get_ports {led[0]}]
49 #set_property PACKAGE_PIN E19 [get_ports {led[1]}]
50 #set_property IOSTANDARD LVCMOS33 [get_ports {led[1]}]
42
51 #set_property PACKAGE_PIN U19 [get_ports {led[2]}]
52 #set_property IOSTANDARD LVCMOS33 [get_ports {led[2]}]
53 #set_property PACKAGE_PIN V19 [get_ports {led[3]}]
54 #set_property IOSTANDARD LVCMOS33 [get_ports {led[3]}]
55 #set_property PACKAGE_PIN W18 [get_ports {led[4]}]
56 #set_property IOSTANDARD LVCMOS33 [get_ports {led[4]}]
57 #set_property PACKAGE_PIN U15 [get_ports {led[5]}]
58 #set_property IOSTANDARD LVCMOS33 [get_ports {led[5]}]
59 #set_property PACKAGE_PIN U14 [get_ports {led[6]}]
60 #set_property IOSTANDARD LVCMOS33 [get_ports {led[6]}]
61 #set_property PACKAGE_PIN V14 [get_ports {led[7]}]
62 #set_property IOSTANDARD LVCMOS33 [get_ports {led[7]}]
63 #set_property PACKAGE_PIN V13 [get_ports {led[8]}]
64 #set_property IOSTANDARD LVCMOS33 [get_ports {led[8]}]
65 #set_property PACKAGE_PIN V3 [get_ports {led[9]}]
66 #set_property IOSTANDARD LVCMOS33 [get_ports {led[9]}]
67 #set_property PACKAGE_PIN W3 [get_ports {led[10]}]
68 #set_property IOSTANDARD LVCMOS33 [get_ports {led[10]}]
69 #set_property PACKAGE_PIN U3 [get_ports {led[11]}]
70 #set_property IOSTANDARD LVCMOS33 [get_ports {led[11]}]
71 #set_property PACKAGE_PIN P3 [get_ports {led[12]}]
72 #set_property IOSTANDARD LVCMOS33 [get_ports {led[12]}]
73 #set_property PACKAGE_PIN N3 [get_ports {led[13]}]
74 #set_property IOSTANDARD LVCMOS33 [get_ports {led[13]}]
75 #set_property PACKAGE_PIN P1 [get_ports {led[14]}]
76 #set_property IOSTANDARD LVCMOS33 [get_ports {led[14]}]
77 #set_property PACKAGE_PIN L1 [get_ports {led[15]}]
78 #set_property IOSTANDARD LVCMOS33 [get_ports {led[15]}]
79
80
81 ##7 segment display
82 set_property PACKAGE_PIN W7 [get_ports {o_Segmentos[6]}]
83 set_property IOSTANDARD LVCMOS33 [get_ports {o_Segmentos[6]}]
84 set_property PACKAGE_PIN W6 [get_ports {o_Segmentos[5]}]
85 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[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {o_Anodo_4_Bits[0]}]
set_property PACKAGE_PIN U4 [get_ports {o_Anodo_4_Bits[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {o_Anodo_4_Bits[1]}]
set_property PACKAGE_PIN V4 [get_ports {o_Anodo_4_Bits[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {o_Anodo_4_Bits[2]}]
set_property PACKAGE_PIN W4 [get_ports {o_Anodo_4_Bits[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {o_Anodo_4_Bits[3]}]
43


##Buttons
set_property PACKAGE_PIN U18 [i_Reset]
set_property IOSTANDARD LVCMOS33 [i_Reset]
#set_property PACKAGE_PIN T18 [get_ports btnU]
#set_property IOSTANDARD LVCMOS33 [get_ports btnU]
#set_property PACKAGE_PIN W19 [get_ports btnL]
#set_property IOSTANDARD LVCMOS33 [get_ports btnL]
#set_property PACKAGE_PIN T17 [get_ports btnR]
#set_property IOSTANDARD LVCMOS33 [get_ports btnR]
#set_property PACKAGE_PIN U17 [get_ports btnD]
#set_property IOSTANDARD LVCMOS33 [get_ports btnD]
```

Código de tests benches

a) Test bench 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_tb
8   // Module Name: Controlador_display_7Segmentos_tb
9   // Project Name: Controlador_display_7Segmentos_tb
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_tb;
24    reg [3:0] i_Datos_0;
25    reg [3:0] i_Datos_1;
26    reg [3:0] i_Datos_2;
27    reg [3:0] i_Datos_3;
28    reg i_Reloj;
29    reg i_Reset;
30    wire [6:0] o_Segmentos;
31    wire [3:0] o_Anodo_4_Bits;
32
33      Controlador_display_7segmentos uut(
34    .i_Datos_0(i_Datos_0),
35    .i_Datos_1(i_Datos_1),
36    .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    i_Reset= 1;
47      i_Reloj  = 0;
48      i_Datos_0= 4'd0;
49      i_Datos_1= 4'd0;
50      i_Datos_2= 4'd0;
51      i_Datos_3= 4'd0;
```

```
    #2
      i_Datos_0= 4'd1;
      i_Datos_1= 4'd2;
      i_Datos_2= 4'd3;
      i_Datos_3= 4'd4;
    i_Reset = 0;

  end
  always
    #1 i_Reloj  = ~i_Reloj ;
  endmodule
```

b) Test bench Demux4_a_1.

```
   `timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
   // Company:
   // Engineer:
   //
   // Create Date: 17.09.2020 19:35:32
   // Design Name:
   // Module Name: demux_4_a_1_TB
   // Project Name:
   // Target Devices:
   // Tool Versions:
   // Description:
   //
   // Dependencies:
   //
   // Revision:
   // Revision 0.01 - File Created
   // Additional Comments:
   //
//////////////////////////////////////////////////////////////////////////////////


  module demux_4_a_1_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 [1:0] i_Sel;
    wire [3:0] o_Demux;

     Demux4_a_1 uut (
    .i_Datos_0(i_Datos_0),
    .i_Datos_1(i_Datos_1),
    .i_Datos_2(i_Datos_2),
    .i_Datos_3(i_Datos_3),
    .i_Sel(i_Sel),
    .o_Demux(o_Demux)
    );
```

```
42
43    initial
44    begin
45          i_Datos_0=0;// inicialización de las entradas
46          i_Datos_1=0;
47          i_Datos_2=0;
48          i_Datos_3=0;
49          i_Sel=0;
50
51          #2
52          i_Datos_0= 4'd1; //Se asigna valor a las 4 entradas
53          i_Datos_1= 4'd2;
54          i_Datos_2= 4'd3;
55          i_Datos_3= 4'd4;
56
57          #2 i_Sel = 0; // Se actualiza el selector cada 2 tiempos
58          #2 i_Sel = 1;
59          #2 i_Sel = 2;
60          #2 i_Sel = 3;
61
62    end
63  endmodule
```

c) Test Becnch Preescalador.

```
1   `timescale 1ns / 1ps
2//////////////////////////////////////////////////////////////////////////////////
3   // Company:
4   // Engineer:
5   //
6   // Create Date: 17.09.2020 19:39:52
7   // Design Name:
8   // Module Name: Preescalador_Tb
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 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<=0;
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
```

d) Test bench DecoAnillo.

```
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=0;
39.       #2 i_Reset=1;
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
```

e) Test bench Deco_Bin_7seg.

```
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 Deco_Bin_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
```