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

Alumnos:

Nelson Eduardo Coronado Gamez

Sergio Adad Bernal Adame

Clara Veronica Guerrero Correa

Hilda Berenice Espinosa Herrera

Guillermo Cruz Fernandez

7“B”

Fecha: 17/09/2020

Resumen.

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

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 a partir del diseño de 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 3.

Índice.

Introducción

Para completar la formación en sistemas digitales es importante no cubrir exclusivamente la parte teórica, sino que es necesario complementar con un conocimiento de implementación real de dichos sistemas digitales.

Uno de los objetivos generales de este proyecto es llegar a conocer una metodología de diseño de sistemas digitales.

Algunos objetivos de este proyecto son:

1. Implementar y programar la FPGA con el diseño completo para visualizar por medio de cuatro display 7 segmentos los caracteres propuestos en la tabla de Verdad

2. Poner en práctica los conocimientos teóricos sobre diseño de circuitos digitales combinacionales.

3. Diseñar una arquitectura clara y específica, con su correcta función y descripción

Requerimientos

Para realizar este proyecto se necesita tener algunos conocimientos previos :

Estar familiarizado con el lenguaje de descripción de hardware Verilog-HDL.

Conocer el entorno de diseño de XILINX, ya que se usará Vivado Design Suite (suite de software para síntesis y análisis de diseños HDL).

Diseño de circuitos digitales combinacionales (multiplexor, demultiplexor, codificador, decodificador, conversor de código, etc).

Saber Implementar y programar la FPGA (basys 3) con el diseño según sea el proyecto.

Tabla de la verdad, se necesitará para realizar el decodificador.

Conocer las herramientas de verificación del diseño desarrollado (simulaciones, diseño esquemático, etc).

Sistema hexadecimal.

Diagramas de implementación para representar la arquitectura física de un sistema.

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.

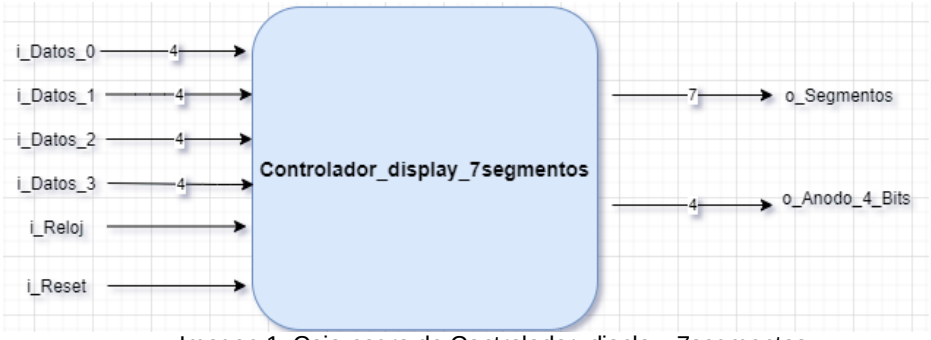

Imagen 1. Caja negra de Controlador_display_7segmentos.

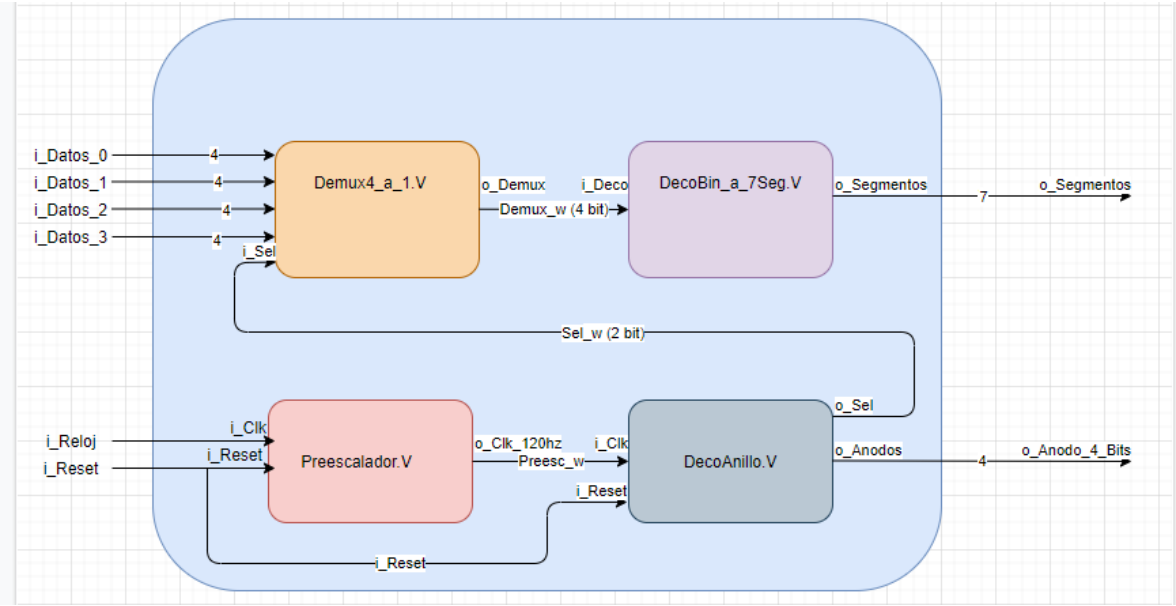

Imagen 2. Caja blanca de Controlador_display_7segmentos.

Para el correcto funcionamiento del controlador es necesario utilizar los bloques (submódulos) que se muestran en la Imagen 2, también se observa las conexiones entre ellos. En la siguiente tabla 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: Es un multiplexor de 4 a 1 que se encarga de seleccionar el dígito que se mostrará según corresponda a su ánodo activo en cero.

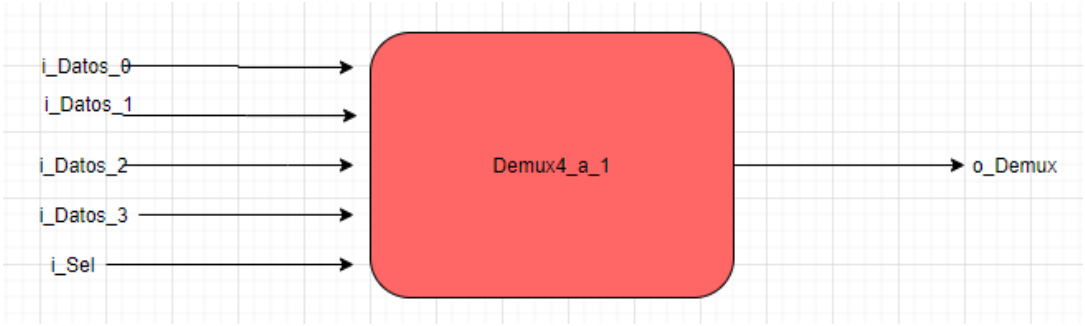

Imagen 3. Caja negra de Demux4_a_1.

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: 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.

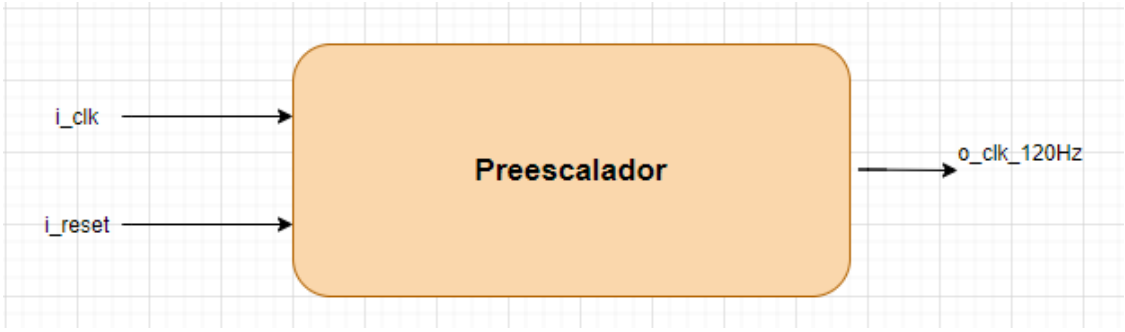

Imagen 4. Caja negra Preescalador.

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: 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.

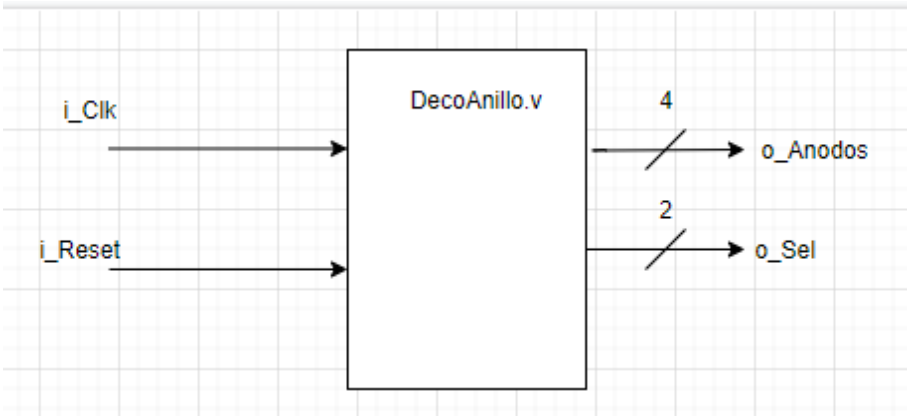

Imagen 5. Caja negra de DecoAnillo.

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: 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.

Imagen 6. 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.

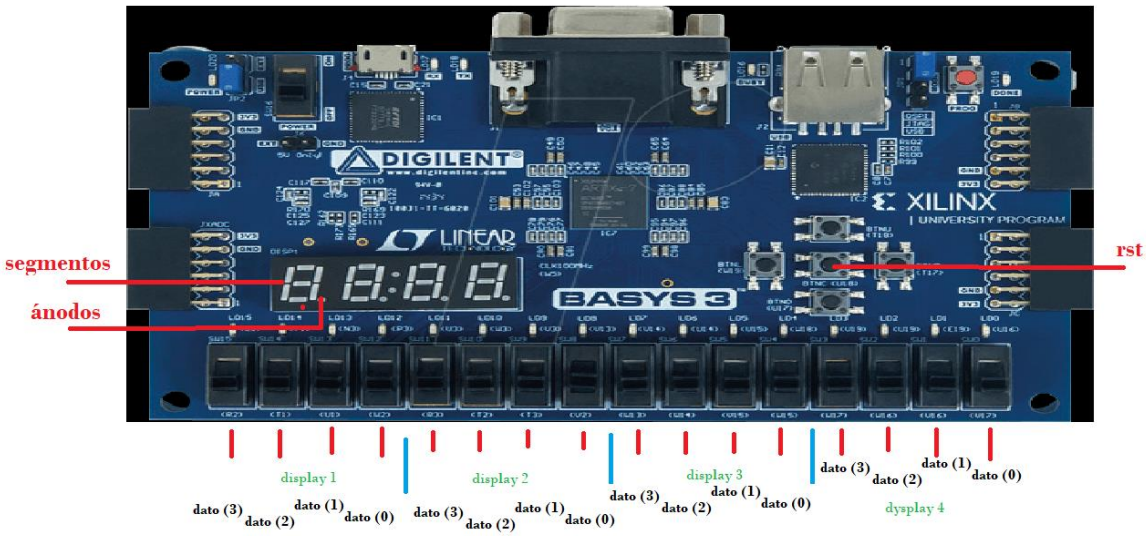

Imagen 7. Implementación en la tarjeta basys.

Pruebas

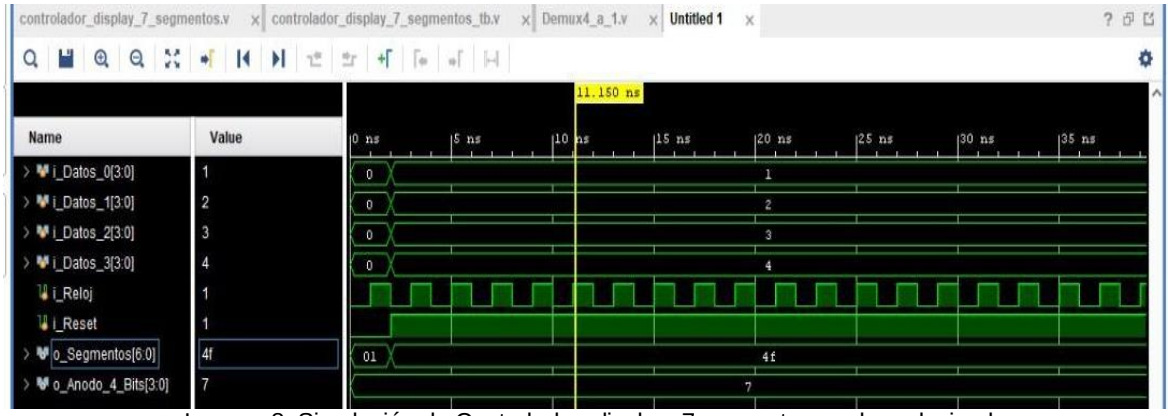

Imagen 8. Simulación de Controlador_display_7segmentos en hexadecimal.

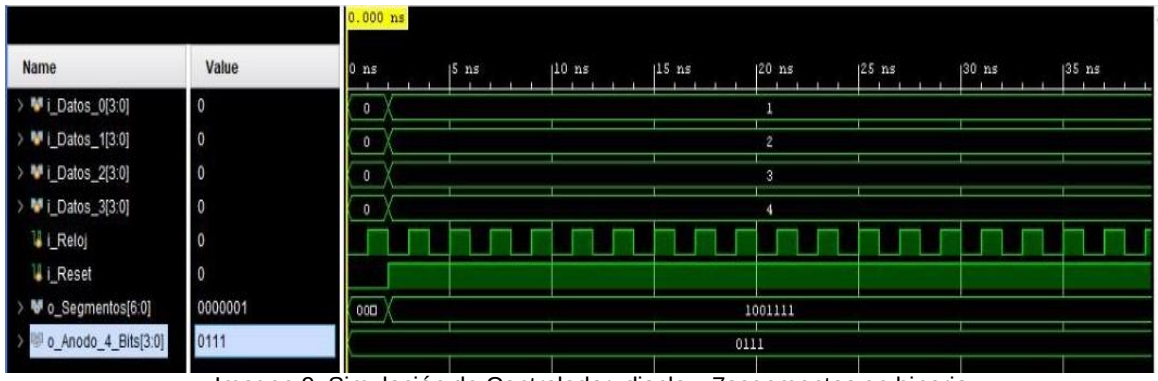

Imagen 9. Simulación de Controlador_display_7segementos en binario.

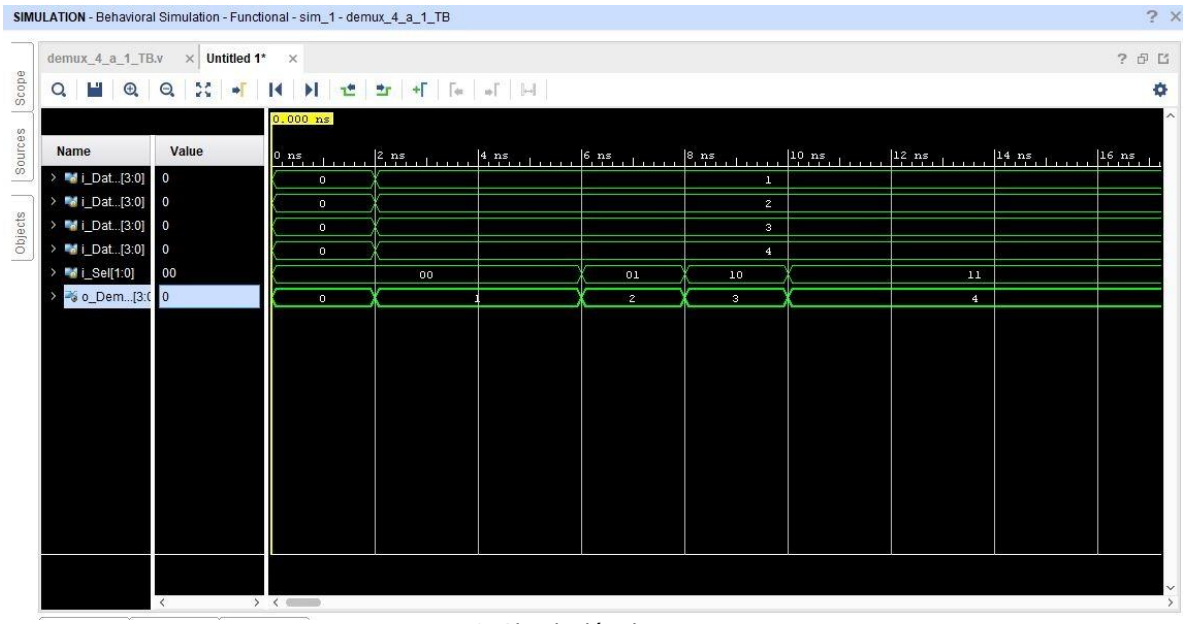

Imagen 10. Simulación de Demux4_a_1..

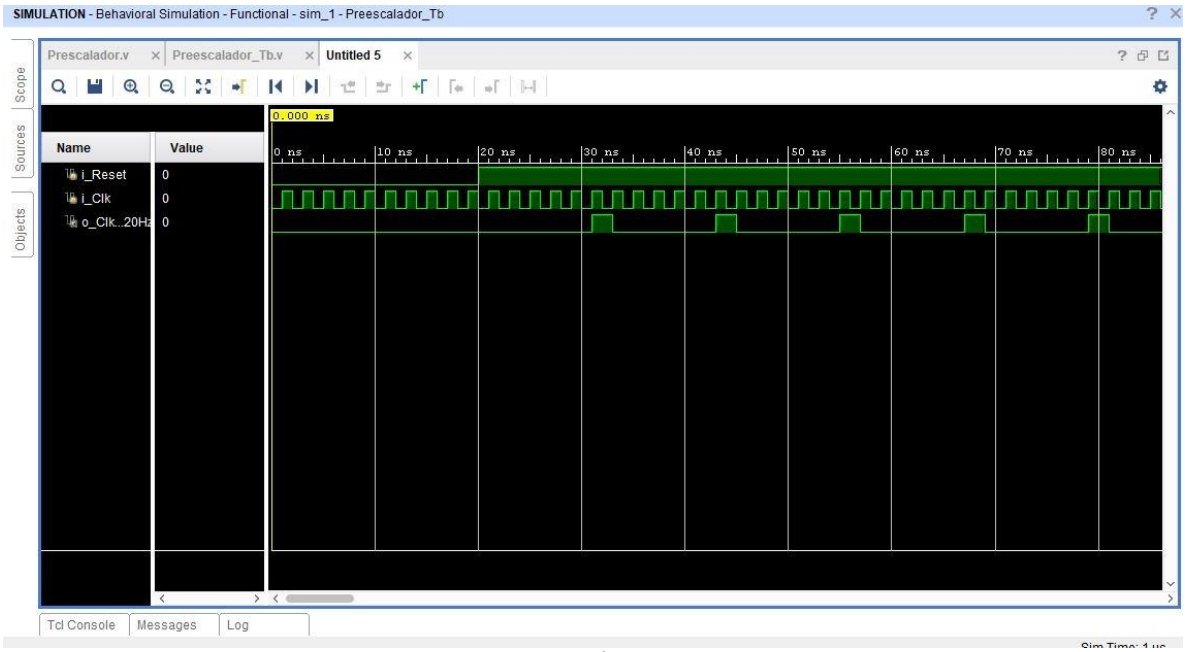

Imagen 11. Simulación de Preescalador.

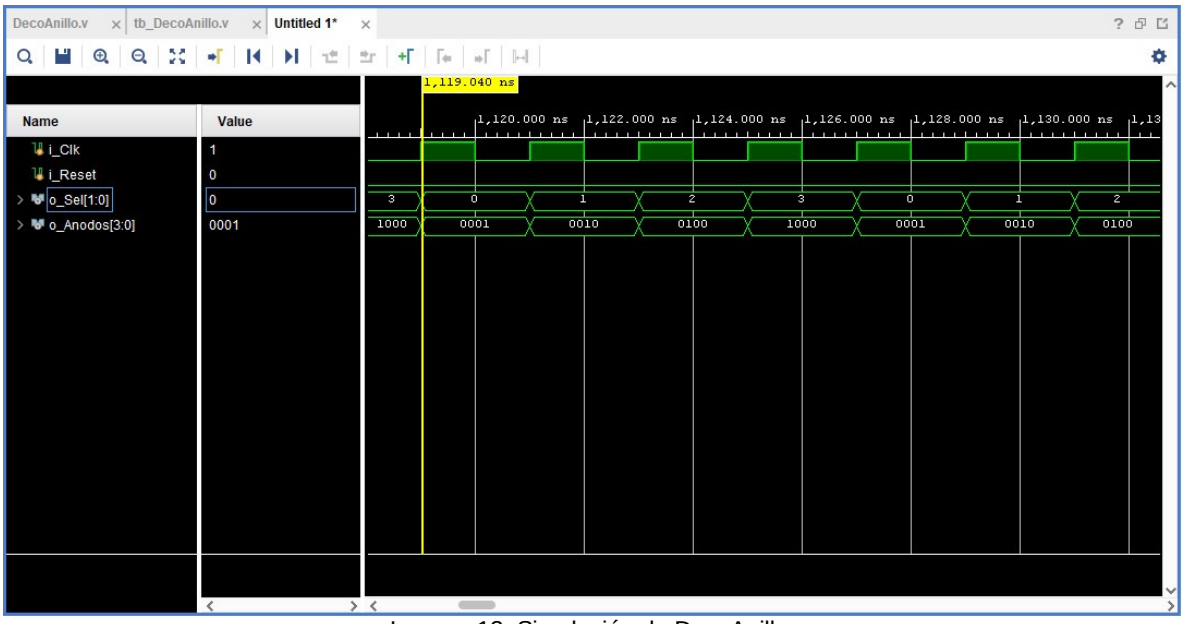

Imagen 12. Simulación de DecoAnillo.

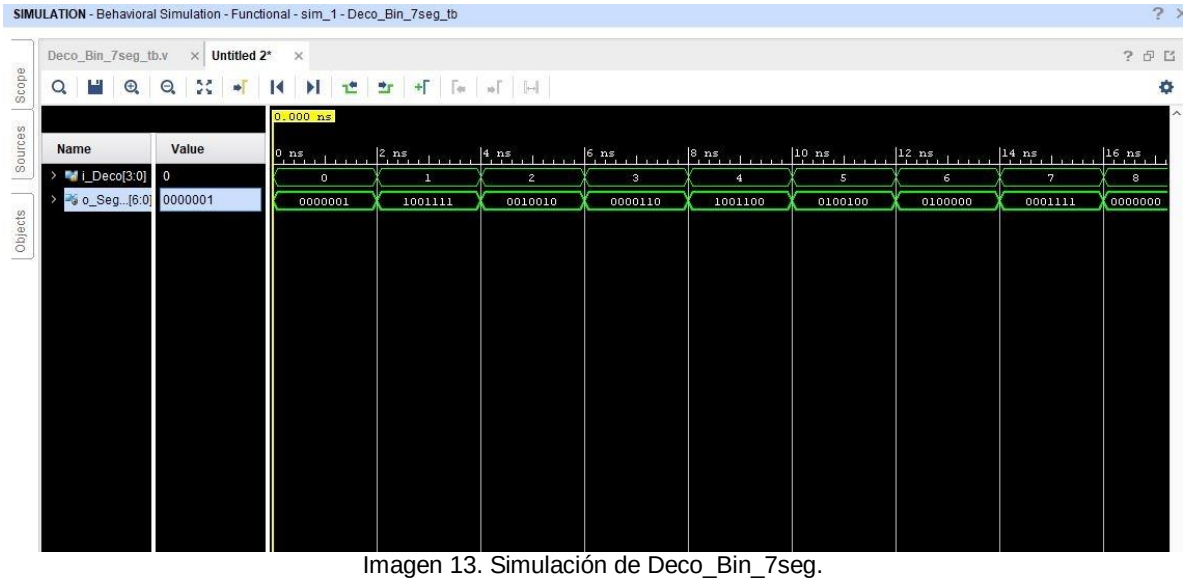

Imagen 13. Simulación de Deco_Bin_7seg.

Análisis de resultados

En las simulaciones de Controlador Diaplay_7segmentos (Imagen 8 y 9) 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.

En la simulación de Demux4_a_1 (imagen 10), 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.

En el Preescalador (imagen 11) 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.

En la simulación de DecoAnillo (imagen 12) 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.

En la simulacion de Deco_Bin_7Seg (imagen 12) 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.

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.

Para tener éxito en la realización de este proyecto o de cualquier otro proyecto de descripción de hardware se tiene que seguir una guía 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.

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] Apuntes de Sistemas Digitales I y II.

[2] Pertenece Equipos Sistemas Digitales lll, “Rob7_Controlador_Display” [Online]. Disponible en: https://app.diagrams.net/#G1vvJhz-PfJGxuvPBWbpXbfhC-Y3YjaSQ-

[3] Pertenece a Equipo 1 Alfa, “Controlador_display_7segmentos.v” [Online]. Disponible en: https://github.com/remberto-uaz/Rob7MicrosAgo2020/tree/master/Equipo_1

[4] Pertenece a Equipo 2 MicroBitos (2020, Septiembre 17), “Demux4_a_1.v” [Online]. Disponible en: https://github.com/remberto-uaz/Rob7MicrosAgo2020/tree/master/Equipo_2

[5] Pertenece a Equipo 3 Rocket, “Preescalador.v” [Online]. Disponible en: https://github.com/remberto-uaz/Rob7MicrosAgo2020/tree/master/Equipo_3

[6] Pertenece a Equipo 4 Roboticos, “DecoAnillo.v” [Online]. Disponible en: https://github.com/remberto-uaz/Rob7MicrosAgo2020/tree/master/Equipo_4

[7] Pertenece a Equipo 5 N, “Deco_Bin_7Seg.v” [Online]. Disponible en: https://github.com/remberto-uaz/Rob7MicrosAgo2020/tree/master/Equipo_5

Apéndice

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. Tabla de verdad de Demux4_a_1.

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

Tabla. Relación de pulsos con salida o_Anodos.

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

Tabla. Relación de pulsos y salida o_Sel.

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. Combinaciones de código hexadecimal.

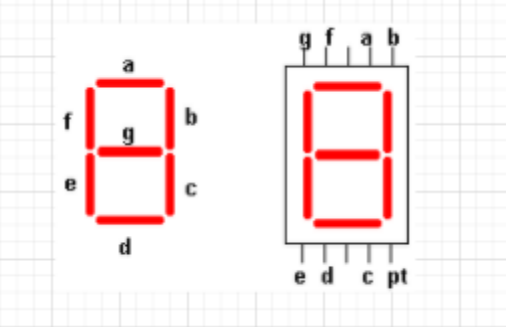

Imagen. Diagrama de display 7 segmentos.

Código de implementación

1. Código Controlador_display_7segementos

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: UAZ
// Engineer: Alfa
//
// Create Date: 07.09.2020 10:37:39
// Design Name:controlador_display_7segmentos
// Module Name: controlador_display_7segmentos
// Project Name: controlador_display_7segmentos
// Target Devices: Basys 3
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Controlador_display_7segmentos(
    input [3:0] i_Datos_0,
    input [3:0] i_Datos_1,
    input [3:0] i_Datos_2,
    input [3:0] i_Datos_3,
    input i_Reloj,
    input i_Reset,
    output [6:0] o_Segmentos,
    output [3:0] o_Anodo_4_Bits
    );
    wire [1:0]Sel_w;
    wire [3:0]Demux_w;
    wire Preesc_w;

        Demux4_a_1 mux(
        .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(Sel_w),
        .o_Demux(Demux_w)
        );

        DecoBin_7Seg dec_BCD (
```

```
    .i_Deco(Demux_w),
    .o_Segmentos(o_Segmentos)
    );

    DecoAnillo dec_anillo (
    .i_Clk(Preesc_w),
    .i_Reset(i_Reset),
    .o_Anodos(o_Anodo_4_Bits),
    .o_Sel(Sel_w)
    );

    Preescalador preesc (
    .i_Clk(i_Reloj),
    .i_Reset(i_Reset),
    .o_Clk_120Hz(Preesc_w)
    );

endmodule
```

2. Código Demux4_a_1.

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 14.09.2020 09:31:11
// Design Name:
// Module Name: demux_4_a_1
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Demux4_a_1
   #(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
```

3. 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
);
reg [19:0] cuent; // registro para almacenar la cuenta interna del preescalador

always @ ( posedge i_Clk, negedge i_Reset)

    if (~i_Reset)
```

```
    begin
        cuent <= 0;
        o_Clk_120Hz <= 0;
    end

 else if (cuent < lim) //aumenta la cuenta en uno mintras no llegue al limite
     begin
         cuent <= cuent + 1'd1;
         o_Clk_120Hz <= 0;
     end

 else // manda un pulso a la salida cuando llega al limite y la cuenta regresa a cero
     begin
         cuent <= 0;
         o_Clk_120Hz <= 1;
     end
endmodule
```

4. DecoAnillo

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 14.09.2020 09:36:26
// Design Name:
// Module Name: DecoAnillo
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module DecoAnillo(
    input i_Reset,
    input i_Clk,
    output [1:0] o_Sel,
    output [3:0] o_Anodos        //Declaramos el bus de salida an, para controlar los
andos.
    );
```

```
wire [3:0]in;  //Declaramos el bus auxiliar tipo wire de 4 bits in.
wire [15:0]x;
reg [1:0]Clk;  //Declaramos el bus auxiliar tipo reg de 2 bits sel.

//Contador de corrida libre de 2 bits.
always @(posedge i_Clk or negedge i_Reset)    //Siempre que ocurra el flanco positivo de i_Clk o i_Reset.
  if(~i_Reset)Clk<=0;    //Si se activa Clk el siguiente valor de sel sera 0.
  else Clk<=Clk+1;  //De lo contrario el siguiente valor de o_Sel sera sel+1.

//Multiplexor Quadruple de 4 entradas 1 salidas.
assign in=x[Clk*4+:4]; //Si sel=2 entonces in=x[2*4+3+:4] o in=x[11:8].
                       //Por lo tanto se seleccionan las 4 entradas correspondientes al display a encender.


//Decodificador de 2 a 4. Selecciona el bit de salida correspondiente.
  assign o_Anodos=(Clk==0)?4'b0001:   //Si sel es igual a 0 an sera igual a 4'b0001
         (Clk==1)?4'b0010:   //Si sel es igual a 1 an sera igual a 4'b0010
         (Clk==2)?4'b0100:   //Si sel es igual a 2 an sera igual a 4'b0100
             4'b1000;   //Si sel es igual a 3 an sera igual a 4'b1000
assign o_Sel=(Clk==0)?2'b00:
        (Clk==1)?2'b01:
        (Clk==2)?2'b10:
              2'b11;

 endmodule
```

5. Deco_Bin_7seg.

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 14.09.2020 09:40:57
// Design Name:
// Module Name: Deco_Bin_7Seg
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
```

```
module DecoBin_7Seg
 (
   input [3:0] i_Deco,
   output reg [6:0] o_Segmentos
   );

   always@*
      begin
         case(i_Deco)
            4'd0: begin  o_Segmentos <= 7'b0000001; end
            4'd1: begin  o_Segmentos <= 7'b1001111; end
            4'd2: begin  o_Segmentos <= 7'b0010010; end
            4'd3: begin  o_Segmentos <= 7'b0000110; end
            4'd4: begin  o_Segmentos <= 7'b1001100; end
            4'd5: begin  o_Segmentos <= 7'b0100100; end
            4'd6: begin  o_Segmentos <= 7'b0100000; end
            4'd7: begin  o_Segmentos <= 7'b0001111; end
            4'd8: begin  o_Segmentos <= 7'b0000000; end
            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 tests benches

1. Test bench Controlador_display_7segmentos

```
4'd0;
   i_Datos_1= 4'd0;
   i_Datos_2= 4'd0;
   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 = 1;

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

2. 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
   #(parameter n=4);
   reg [n-1:0] i_Datos_0;
   reg [n-1:0] i_Datos_1;
   reg [n-1:0] i_Datos_2;
```

```
 reg [n-1:0] i_Datos_3;
 reg [1:0] i_sel;
 wire [n-1:0] o_demux;

demux_4x1 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)
);



 initial
   begin
     i_Datos_0=0;// inicialización de las entradas
     i_Datos_1=0;
     i_Datos_2=0;
     i_Datos_3=0;
     i_sel=0;

     #2
     i_Datos_0= 4'd1; //Se asigna valor a las 4 entradas
     i_Datos_1= 4'd2;
     i_Datos_2= 4'd3;
     i_Datos_3= 4'd4;

     #2 i_sel = 0; // Se actualiza el selector cada 2 tiempos
     #2 i_sel = 1;
     #2 i_sel = 2;
     #2 i_sel = 3;

   end
endmodule
```

3. Test Becnch Preescalador.

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 17.09.2020 19:39:52
// Design Name:
// Module Name: Preescalador_Tb
// Project Name:
// Target Devices:
// Tool Versions:
```

```
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module Preescalador_Tb;
    reg i_Reset;
    reg i_Clk;
    wire o_Clk_120Hz;
    Prescalador uut(
        .i_Reset(i_Reset),
        .i_Clk(i_Clk),
        .o_Clk_120Hz(o_Clk_120Hz)
    );
    initial
    begin
        i_Reset<=1;
        i_Clk<=0; //inicializa las entradas

        #20 i_Reset<=1; // cuando se activa el reset empieza el prescalador
    end
    always@(*)
    begin
        #1 i_Clk <= ~i_Clk; //clk invierte su valor para que el reloj este en constante
cambio
    end

endmodule
```

4. Test bench DecoAnillo.

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: Universidad Autonoma de Zacatecas
// Engineer: Sergio Adad Bernal Adame
//
// Create Date: 09.09.2020 11:26:32
// Design Name:
// Module Name: tb_DecoAnillo
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
```

```
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module DecoAnillo_TB;
    reg i_Clk;
    reg i_Reset;
    wire [1:0] o_Sel;
    wire [3:0] o_Anodos;

    DecoAnillo uut(
        .i_Clk(i_Clk),
        .i_Reset(i_Reset),
        .o_Sel(o_Sel),
        .o_Anodos(o_Anodos)
     );
    initial
        begin
         i_Clk=0;
         i_Reset=0;
        #2 i_Reset=1;
     end
   always
   #1 i_Clk=!i_Clk;
    initial
        begin
            #2 i_Clk=4'b0111;
            #2 i_Clk=4'b1011;
            #2 i_Clk=4'b1101;
            #2 i_Clk=4'b1110;
            #2 i_Clk=2'b00;
            #2 i_Clk=2'b01;
            #2 i_Clk=2'b10;
            #2 i_Clk=2'b11;
     end
endmodule
```

5. Test bench Deco_Bin_7seg.

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company: Universidad Automona de Zacatecas
// Engineers:
```

```
//Agustín Antonio Palafox Molina
//José Alfredo Hernadez Dueñas
//Jesús Francisco Villaseñor Correa
//José Roberto Novoa López
//Julio Angel Pérez Dávila
// Create Date: 09.09.2020 10:37:24
// Design Name:
// Module Name:deco_bin_7seg
// Project Name:
//Controlador_display_7segmentos
// Target Devices:
// Tool Versions:
// Description:
//Testbench
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module DecoBin_7seg_tb;

    reg [3:0] i_Deco;
    wire [6:0] o_Segmentos;

    DecoBin_7seg uut(
        .i_Deco(i_Deco),
        .o_Segmentos(o_Segmentos)
    );

    initial
        begin
            i_Deco = 0;

            #2 i_Deco = 1;
            #2 i_Deco = 2;
            #2 i_Deco = 3;
            #2 i_Deco = 4;
            #2 i_Deco = 5;
            #2 i_Deco = 6;
            #2 i_Deco = 7;
            #2 i_Deco = 8;
            #2 i_Deco = 9;
            #2 i_Deco = 10;
            #2 i_Deco = 11;
            #2 i_Deco = 12;
            #2 i_Deco = 13;
            #2 i_Deco = 14;
```

```
            #2 i_Deco = 15;
        end
endmodule
```