Decodificador Hexadecimal a 7 Segmentos con Multiplexación

Camilo Andres Garzón Código:
David Ricardo Martínez Hernández Código: 261931
Juan Pablo Rodríguez Rojas Código: 261744

***Resumen*— Una de la aplicaciones mas importantes de la herramienta de lógica combinacional es su relativa facilidad para la construcción de elementos de funcionamiento básico, tal es el caso que tenemos en esta practica, en el cual se necesita construir un codificador para un numero hexadecimal, que en el sistema binario se necesitan 4 bits par su representación, y así al pasarlo por el proceso de codificación sea posible utilizar un display led de 7 segmentos, y así poder todos los numero hexadecimales de un solo dígito, de 0 a F. Vale la pena mencionar que se requiere una explicación adicional de lo que significa un display led de 7 segmentos, ya que basado en si su funcionamiento es ánodo o cátodo común, es que se construyo la tabla de verdad que describió el decodificador.**

***Palabras clave*— 7 Segmentos, Ánodo común, Cátodo común, Decodificador, FPGA, Led, reducción, Salida, Verilog, Xilinx.**

I. OBJETIVOS

- Comprender el lenguaje Verilog HDL.
- Entender la estructura, escritura en VHDL y el funcionamiento de un decodificador Hexadecimal a 7 segmentos.
- Entender la estructura, escritura en VHDL y el funcionamiento de un decodificador Hexadecimal a 7 segmentos con Multiplexación.
- Identificar todos los pasos de diseño de un sistema que en este caso es digital, desde la descripción del problema hasta la implementación.
- Observar como la dependencia de la tecnología con la cual se trabaja, en este caso display 7 segmentos con ánodo común, establecerá las bases a la solución que se debe obtener.
- Aprender los niveles de abstracción que pueden ser utilizados en el lenguaje Verilog con el fin de poder solucionar el problema planteado.

II. INTRODUCCIÓN

El ***display de*** 7 ***segmentos*** es uno de los elementos más utilizados a nivel mundial, aunque ha sido dejado de lado por las pantallas en ***LCD***, un 7 **segmentos** es la unión de 7 u 8 leds sobre una base de manera que se puedan representar todos los numeros y algunas letras (Código Decimal y Hexadecimal). Los 7 primeros leds son lo encargados de formar el símbolo y el octavo es el encargado de controlar el punto decimal.
Existen dos tipos de 7 segmentos, de **Ánodo común** y **Cátodo común**.
Ánodo común: Todos los Ánodos se encuentran conectados entre si, para dejar libre el Cátodo, los Ánodos son puestos a V_{CC} y es controlado por tierra, cada vez que se quiera prender un led su pin es llevado a tierra.
Cátodo común: Todos los Cátodos se encuentran conectados entre si, para dejar libre el Ánodo, los Cátodo son puestos a GND y es controlado por V_{CC}, cada vez que se quiera prender un led su pin es llevado a V_{CC}.
El diseño físico de un 7 segmentos representado en la fig 1 [5]

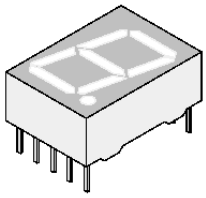

Fig. 1: Display 7 Segmentos [5]

El diseño esquemático de un 7 segmentos representado en la fig 2 [5]

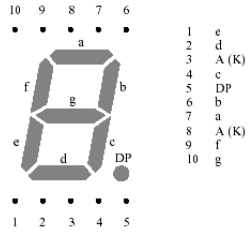

Fig. 2: Display 7 Segmentos esquemático [5]

Diagrama funcionamiento de los 7 segmentos en ánodo y cátodo común

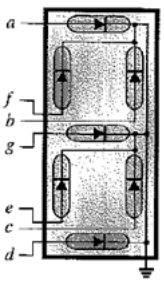

Fig. 3: Display 7 Segmentos funcional Cátodo Común [4]

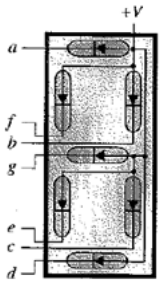

Fig. 4: Display 7 Segmentos funcional Ánodo Común [4]

Las **FPGA** (*Field Programmable Gate Array*), introducidas por **Xilinx** en 1985, son el dispositivo programable por el usuario de más general espectro. También se denominan **LCA** (*Logic Cell Array*). Consisten en una matriz bidimensional de bloques configurables que se pueden conectar mediante recursos generales de interconexión. Estos recursos incluyen segmentos de pista de diferentes longitudes, más unos conmutadores programables para enlazar bloques a pistas o pistas entre sí. En realidad, lo que se programa en una **FPGA** son los conmutadores que sirven para realizar las conexiones entre los diferentes bloques, más la configuración de los bloques.
Un ***decodificador*** es un circuito lógico con variables de varias estradas y salidas que convierte las entradas codificadas en salidas codificadas, donde los códigos de entrada son diferentes, en donde el codigo de entrada tiene generalmente menos bits que el de salida.[1]

III. Materiales y Métodos

Para desarrollar este laboratorio fue necesaria la utilización de:

- Computador
- FPGA
- Software Xilinx, Inc

IV. Análisis y Resultados

IV-A. Decodificador Hexadecimal a 7 segmentos

Las tabla tabla de verdad para poder realizar esta guia es la TAB. I:

Hexadecimal	Binario	a	b	c	d	e	f	g
0	0000	0	0	0	0	0	0	1
1	0001	1	0	0	1	1	1	1
2	0010	0	0	1	0	0	1	0
3	1011	0	0	0	0	1	1	0
4	1100	1	0	0	1	1	0	0
5	1101	0	1	0	0	1	0	0
6	1110	0	1	0	0	0	0	0
7	1111	0	0	0	1	1	1	1
8	1000	0	0	0	0	0	0	0
9	1001	0	0	0	1	1	0	0
A	1010	0	0	0	1	0	0	0
B	1011	1	1	0	0	0	0	0
C	1100	0	1	1	0	0	0	1
D	1101	1	0	0	0	0	1	0
E	1110	0	1	1	0	0	0	0
F	1111	0	1	1	1	0	0	0

TABLA I: Tabla de Verdad Decodificador Hexadecimal a 7 segmentos

[1]Definición tomada de [3], *Pag 351*

EL código utilizado para lograr esta práctica fue:

```
module conhex(i0,i1,i2,i3,a,b,c,d,e,f,g);
input i0,i1,i2,i3;
output a,b,c,d,e,f,g;
assign a=(~i0&~i1&~i2&i3)|(~i0&i1&~i2&~i3)|(i0&i1&~i2&i3)|(i0&~i1&i2&i3);
assign b=(i0&i2&i3)|(i1&i2&~i3)|(i0&i1&~i3)|(~i0&i1&~i2&i3);
assign c=(i0&i1&i2)|(i0&i1&~i3)|(~i0&~i1&i2&~i3);
assign d=(i1&i2&i3)|(~i1&~i2&i3)|(~i0&i1&~i2&~i3)|(i0&~i1&i2&~i3);
assign e=(~i0&i3)|(~i1&~i2&i3)|(~i0&i1&~i2);
assign f=(~i0&~i1&i3)|(~i0&~i1&i2)|(~i0&i2&i3)|(i0&i1&~i2&i3);
assign g=(~i0&~i1&~i2)|(~i0&i1&i2&i3)|(i0&i1&~i2&~i3);
endmodule
```

La salida de la simulación se encuentra en la fig 4

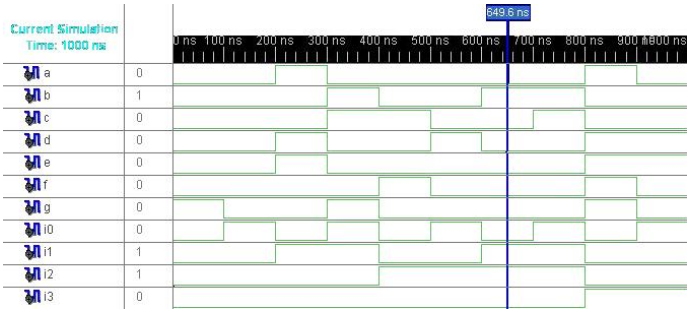

Fig. 5: Salida Simulación

Los resultados obtenidos en la práctica con la FPGA fueron los mismos resultados que los obtenidos en las simulaciones.

IV-B. Decodificador Hexadecimal a 7 Segmentos con Multiplexación

Para esta parte del laboratorio se utilizo TAB I, agregando una tabla adicional para controlar la posición del display TAB IV-B:

S_1 S_0 1	O_0	O_1	O_2	O_3
0 0 0	0	X	X	X
0 0 1	1	X	X	X
0 1 0	X	0	X	X
0 1 1	X	1	X	X
1 0 0	X	X	0	X
1 0 1	X	X	1	X
1 1 0	X	X	X	0
1 1 1	X	X	X	1

TABLA II: Tabla de Verdad Multiplexación

Los valores de X son valores indefinidos y fueron escogidos de acuerdo a la necesidad y de acuerdo al planteamiento del problema, para lograr una reducción máxima en un circuito.
EL código utilizado para lograr esta práctica fue:

```
module convBCDmulti(i0,i1,i2,i3,a,b,c,d,e,f,g,s0,s1,mu);

input i0,i1,i2,i3,s0,s1;//Entradas

output a,b,c,d,e,f,g;//Leds del 7-segmentos

output reg [3:0] mu;//Cada 7 Segmentos

//COdificador Hexadecimal
assign a=(~i0&~i1&~i2&i3)|(~i0&i1&~i2&~i3)|(i0&i1&~i2&i3)|(i0&~i1&i2&i3);

assign b=(i0&i2&i3)|(i1&i2&~i3)|(i0&i1&~i3)|(~i0&i1&~i2&i3);

assign c=(i0&i1&i2)|(i0&i1&~i3)|(~i0&~i1&i2&~i3);

assign d=(i1&i2&i3)|(~i1&~i2&i3)|(~i0&i1&~i2&~i3)|(i0&~i1&i2&~i3);

assign e=(~i0&i3)|(~i0&i1&~i2)|(~i1&~i2&i3);

assign f=(~i0&~i1&i3)|(~i0&~i1&i2)|(~i0&i2&i3)|(i0&i1&~i2&i3);

assign g=(~i0&~i1&~i2)|(~i0&i1&i2&i3)|(i0&i1&~i2&~i3);

// Variacion del Demultiplexor
always @(s1 or s0)
begin
case({s1,s0})
2'b00:mu='b1110;
2'b01:mu='b1101;
2'b10:mu='b1011;
2'b11:mu='b0111;
endcase
end
endmodule
```

La salida de la simulación se encuentra en la fig **??**

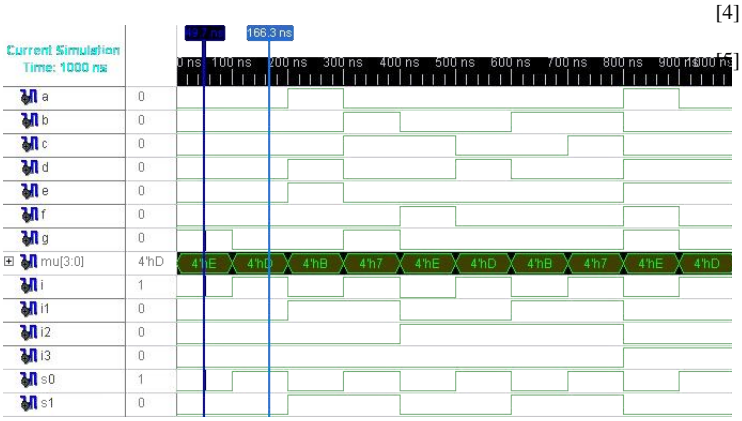

Fig. 6: Salida Simulación

Los resultados obtenidos en la práctica con la FPGA fueron los mismos resultados que los obtenidos en las simulaciones.

V. Conclusiones

- Una de las conclusiones más importantes recae en que al observar el lenguaje vemos que se pudo haber trabajado la etapa de multiplexación y decodificación como bloques fundamentales de la solución, es decir pudimos haberlos implementados en módulos diferentes, y haber creado un modulo principal, Top module, y en este llamar o hacer referencia de estos submódulos, ya que de hacerlo así, podemos reutilizar posteriormente en el diseño de otras soluciones estos bloques principales sin tener la necesidad de tener que escribirlos o diseñarlos de nuevo.
- Aunque en el informe no aparecen las simulaciones previas que se hicieron del sistema antes de su implementación final, vale la pena mencionar que antes de poder llegar a la solución definitiva se hicieron simulaciones después de implementar una solución previa y observamos que una de las funciones assign que aparecen en el código estaba mal implementada, de allí tuvimos que volver a la etapa de descripción del funcionamiento en Verilog y corregir la función que se encontraba defectuosa, de allí la importancia de seguir un sistema de pasos de diseño, ya que de otra manera cabe la posibilidad de presentar una solución del problema defectuosa.
- Vale la pena recalcar, que aunque la etapa de decodificación esta bien desarrollada, se pudo haber escrito de una manera más abstracta y su funcionamiento aunque para efectos de la presente practica es igual, resulta que entre más abstracto sea el nivel de la descripción del sistema, presenta una mayor facilidad de movimiento de información, y de implementación, al hacer que los sistemas sean menos dependientes de la tecnología necesaria para su implementación.

Referencias

[1] Dorf Svoboda. «*Circuitos Eléctricos*». Alfaomega, 2006.
[2] C. J. Savant. «*Diseños Electrónicos: Circuitos de Sistema*». Prentice-Hall, 2006.
[3] John F. Wakerly. «*Diseño Digital: Principios y Prácticas*». Prentice-Hall, 2001.
[4] Thomas L. Floyd. «*Fundamentos de Sistemas Digitales*». Prentice-Hall, 2000.
[5] Sitio Web: http://www.electronguia.com.ar/Principiante/Informacion/Displays%20de%207%20segmentos/Displays%20de%207%20segmentos.htm