Desarrollo de un Turnero Digital Usando FPGA's Laboratorio de Electrónica Digital Módulo: 2

Héctor F. Jiménez S.
hfjimenez@utp.edu.co
PGP KEY ID: 0xB05AD7B8

Sebastian Zapata
sebastzapata93@gmail.com
PGP KEY ID: 0xfffff

Francisco Gallego
juanfran16@utp.edu.co

Fecha de Entrega: **22** Septiembre[1], 2016
Profesor: Ing.Msc(c) Ramiro Andres Barrios Valencia

1 Objetivos

- Desarrollar la Unidad de Control para la utilizacion del arreglo de displays de siete segmentos.
- Desarrollar modulo para la conversion para pasar a BCD.

2 Marco Teórico

Para desarrollar el modulo número dos del proyecto lo primero que realizamos es verificar en la hoja del fabricante las conexiones y el tipo de configuracion utilizada en la placa de desarrollo **Nexys2**. Al revisar el Datasheet y manual encontramos que la placa contiene 4 displays de 7 segmentos y punto decimal(dp) en la configuracion de anodo común lo cual implica que todos los anodos de los leds que conforman el display este conectados a un punto en común como se aprecia en la figura 1 , para encender un solo led necesitaremos un valor logico de ***0*** para cerrar el circuito.

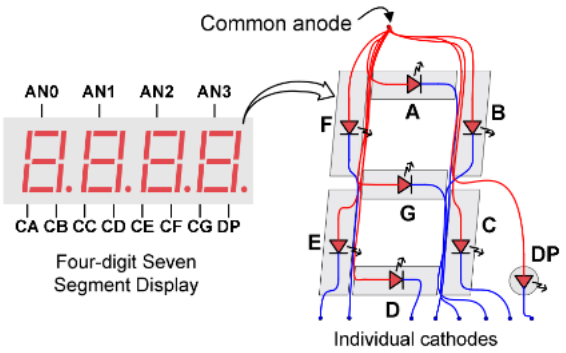

Figure 1: Esquema Catodo Común.

Revisando el esquematico de la tarjeta de desarrollo sabemos que la activacion de los displays se da por la saturación del transistores pnp como se observa en la siguiente figura 2 :

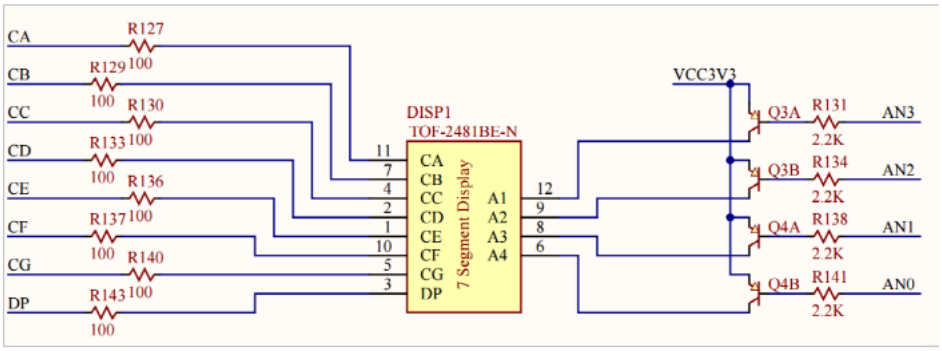

Figure 2: Esquema provisto en el esquemático, 4 pulsadores.

Los displays de 7 segmentos se encuentran en la configuración de ánodo común, un **0** se utiliza para encendido y un **1** para apagado (lógica negativa). Al representar los números usaremos una codificacion de 4 bits que permiten los 10 digitos (del *0 al 9*),y asi poder mostrar el digito en un visualizador de siete segmentos siendo necesario decodificar el valor numérico según el patrón de LEDs indicado, se tiene un carácter al cual se le asigna un valor numérico (codificación) para tratar con el digitalmente, posteriormente se transforma al patrón de LEDs correspondiente (decodificación) como se puede observar en el listing 1. El número que se recibe en 4 bits codificado se decodifica a su valor correspondiente donde prenderemos o apagaremos los 8 leds del display.

```
entity decoder_arch is
    Port ( numero : in  STD_LOGIC_VECTOR(3 downto 0);
           segment : out  STD_LOGIC_VECTOR(7 downto 0));
end decoder_arch;
architecture Behavioral of decoder_arch is
begin
        process(numero) begin
        case numero is
                when "0000" => segment <= "00000011"; --0
                when "0001" => segment <= "10011111"; --1
                when "0010" => segment <= "00100101"; --2
                when "0011" => segment <= "00001101"; --3
                when "0100" => segment <= "10011001"; --4
                when "0101" => segment <= "01001001"; --5
                when "0110" => segment <= "01000001"; --6
                when "0111" => segment <= "00011111"; --7
                when "1000" => segment <= "00000001"; --8
                when "1001" => segment <= "00001001"; --9
                when others => segment <= "00000011"; -- Nada
        end case;
end process;
end Behavioral;
```

Listing 1: Decodificador de 4 bits.

DECIMAL	BINARIO
0	0001
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Table 2.1: Valores Correspondientes de Binario a Bcd

Una vez hemos construido el modulo que se encarga de decodificar el valor que queremos mostrar en los displays necesitamos saber como seleccionar el display adecuado; para esto utilizamos un multiplexor de 4 bits. un multiplexor o selector de datos es un circuito lógico que acepta varias entradas y solamente permite a una de ellas alcanzar la salida. La figura 3 muestra el diagrama de

un multiplexor, donde se observa que la salida Z puede tomar el valor de A o B, pero no de ambos a la vez, en base al valor del parámetro de selección $S0$. Nosotros hemos escrito el codigo que se encarga de recibir cual sera el modulo que seleccionemos ***(selectorin)*** una vez hecho esto lo que nosotros tenemos que hacer es habilitar con un 0 el display requerido y pasarle el valor decodificado El listing 3

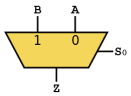

Figure 3: Multiplexor 2 a 1.

```
entity mux7s_arch is
    Port ( d0 : in  STD_LOGIC_VECTOR(3 downto 0);
           d1 : in  STD_LOGIC_VECTOR(3 downto 0);
           d2 : in  STD_LOGIC_VECTOR(3 downto 0);
           d3 : in  STD_LOGIC_VECTOR(3 downto 0);
           selectorin : in  STD_LOGIC_VECTOR(3 downto 0);
           numero : out  STD_LOGIC_VECTOR(3 downto 0));
end mux7s_arch;

architecture Behavioral of mux7s_arch is
begin
        process(selectorin,d0,d1,d2,d3) begin
                case selectorin is
                        when "1110" =>
                                numero<=d0;
                        when "1101" =>
                                numero<=d1;
                        when "1011" =>
                                numero<=d2;
                        when "0111" =>
                                numero<=d3;
                        when others =>
                                numero<=d0;
                end case;
        end process;
end Behavioral;
```

Listing 2: Decodificador de 4 bits.

Pero existe un problema y es que nosotros solo podremos visualizar un display a la vez, es decir tendremos que hacer un refresco por cada display de al menos **4ms** para engañar la retina del ojo humano utilizando la frecuencia correcta,

se llega a la ilusión de que hay cuatro dígitos distintos y encendidos al mismo tiempo si son 4 displays en total sera **16ms**. Este efecto es similar al comentado por el profe ramiro donde un vídeo esta compuesto de muchas imágenes sucesivas y a cierta frecuencia el ojo ve en movimiento. Una frecuencia de **250kHz** es más que suficiente para generar este efecto, por lo cual se utiliza el divisor creado en el modulo uno VHDL, que se presenta a continuación.

```
entity selector_arch is
    Port ( clk : in  STD_LOGIC;
           salida : out  STD_LOGIC_VECTOR(3 downto 0):="1110");
end selector_arch;

architecture Behavioral of selector_arch is

signal contador : integer range 0 to 250000 := 0;
signal ciclos : integer range 0 to 3 :=0;
begin
        process(clk) begin
                if rising_edge(clk) then
                        if contador = 250000 then
                                if ciclos = 0 then
                                        salida<="1101";
                                        ciclos<=ciclos+1;
                                        end if;
                                if ciclos = 1 then
                                        salida<="1011";
                                        ciclos<=ciclos+1;
                                        end if;
                                if ciclos = 2 then
                                        salida<="0111";
                                        ciclos<=ciclos+1;
                                        end if;
                                if ciclos = 3 then
                                        salida<="1110";
                                        ciclos<=0;
                                        end if;
                                contador<=0;
                        else
                                contador<= contador+1;
                        end if;
                end if;
        end process;
end Behavioral;
```

Listing 3: Divisor de Frecuencia 250kHz.

Físicamente para realizar el impacto sobre la fpga debemos especificar cuales son las conexiones físicas en la fpga[2], ademas también están especificadas en la tarjeta. véase figura 4 , figura 5.

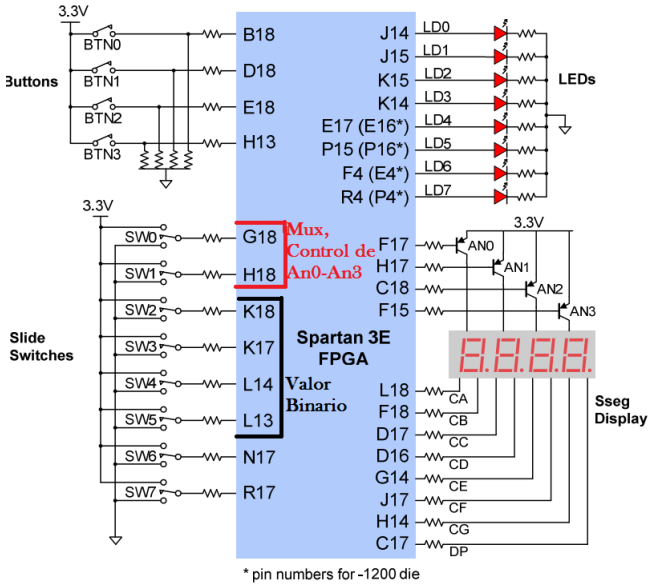

Figure 4: Esquema provisto en el esquemático, 4 pulsadores.

Figure 5: Esquema provisto en el esquemático, 4 pulsadores.

3 Errores Comunes

En este modulo tuvimos muchos problemas, apesar de ser sencillo en la finalizacion algunas veces siempre nos aparecia conexiones indefinidas y desconectadas. Toco realizar una revision exaustiva de todo el codigo, para determinar por que no se conectaban. Tiempo despues al querer impactar en la fpga olvidamos pinear la señal mas importante de todas que es la que sincroniza todas las operaciones, hablamos de la señal del CLK.

2. Este fue uno de nuestros problemas al impactar pues al momento de mapear los pines reales olvidamos pinear nuestra señal de reloj.

4 Anexos

los demás archivos son adicionales que resuelven la misma implementación, se encuentran en el repositorio[3] *Nota*: Las referencias utilizadas se encuentran en los pies de página. Si requiere de manera detallada estas contacte con *miembros del equipo.*

3. `https://github.com/heticor915/DigitalElectronicsLab/tree/master/Reports/Module2`