Display 7 segmentos

Antes de diseñar el convertidor a 7 segmentos, consultamos el manual de referencia de la tarjeta para saber con qué niveles lógicos se activa cada uno de los "displays", cuántos tiene y qué niveles se utilizan para encender los segmentos. A partir de la Figura 1 (tomada de [1]) podemos determinar que cada "display" se habilita con un nivel bajo en su correspondiente transistor ***AN(7:0)*** y que todos los dígitos comparten las mismas señales para los segmentos ***CA-DP***(7:0). Si ***AN(7:0)*** = x"00", todos los dígitos están habilitados y desplegarán el mismo valor dado en ***CA-DP***(7:0). Para desplegar un dígito diferente en cada "display", es necesario encender y apagar de manera alternada los 8 dígitos, uno a la vez, tan rápido que nuestros ojos perciban los 8 encendidos. Este proceso consiste en multiplexar en el tiempo el encendido de cada "display".

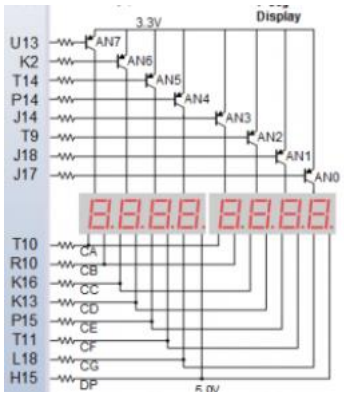
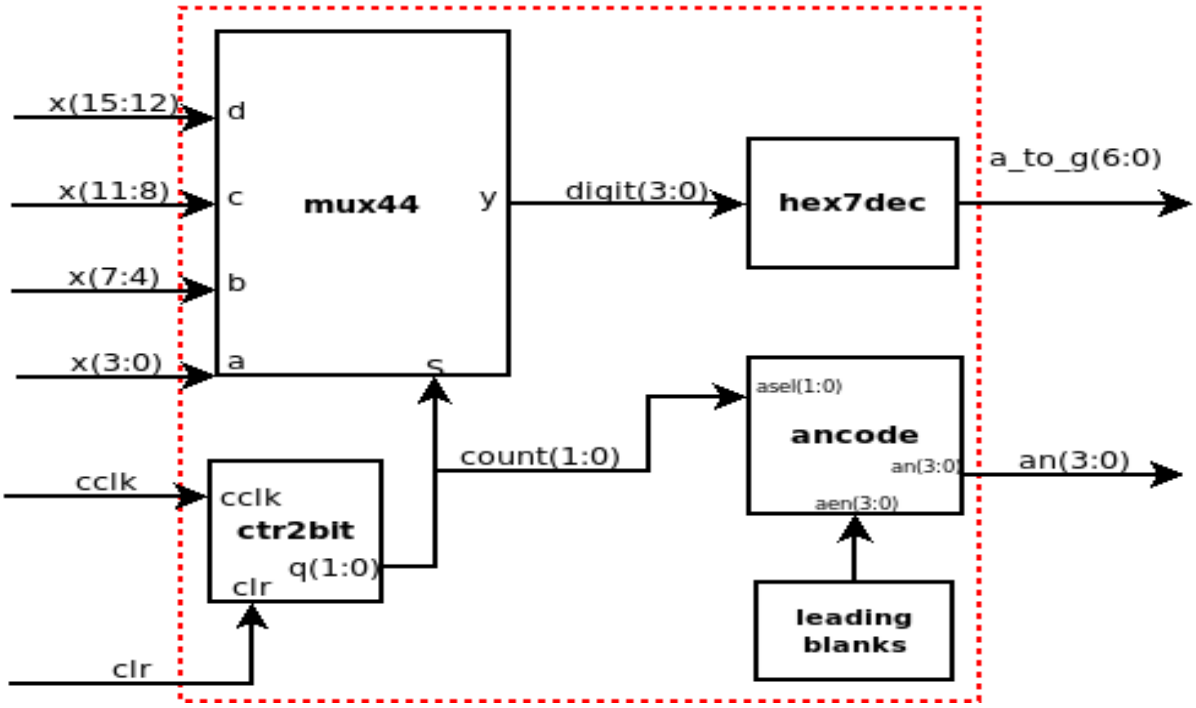


Figura 1.Módulo 7 seg.

En la figura de abajo se muestra el diseño a realizar:



Un contador puede ser usado para dividir la frecuencia f de un reloj, donde la frecuencia de salida $q_{(i)}$ es:

$$f_i = \frac{f}{2^{i+1}}$$

Por ejemplo, en esta tarjeta la frecuencia base es de 100 Mhz, la frecuencia de salida en $q_{(8)}$ será:

$$f_8 = \frac{f}{2^{8+1}} = \frac{100{,}000{,}000}{2^9} = 195.3125Khz$$

El contador de 2 bits ***ctr2bit***, funcionando a 190 Hz, proporciona la señal de selección *S(1:0)* al ***mux44*** y la señal de selección *asel(1:0)* del componente ***ancode***, cuyo propósito es activar un dígito a la vez, dependiendo del valor:

Count(1:0)=”00”	An(3:0)=”1110”	Primer dígito de la derecha
Count(1:0)=”01”	An(3:0)=”1101”	Segundo dígito de la derecha
Count(1:0)=”10”	An(3:0)=”1011”	Tercero dígito de la derecha
Count(1:0)=”11”	An(3:0)=”0111”	Cuarto dígito de la derecha

Bibliografía

[1] “Nexys A7 Reference Manual - Digilent Reference”. https://digilent.com/reference/programmable-logic/nexys-a7/reference-manual (consultado el 1 de marzo de 2023).

Incluimos el código con el fin de relacionarlo con el diagrama de bloques.

```
        library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.std_logic_unsigned.all;
---------------------------------
entity X7segb is
    Port ( x : in  STD_LOGIC_VECTOR (15 downto 0);
           cclk,clr : in  STD_LOGIC;
           a_to_g : out  STD_LOGIC_VECTOR (6 downto 0);
           an : out  STD_LOGIC_VECTOR (3 downto 0));
end X7segb;

architecture Behavioral of X7segb is
        --señales intermedias para conexión interna
        signal digit : std_logic_vector (3 downto 0);
        signal count : std_logic_vector (1 downto 0);
        signal aen : std_logic_vector (3 downto 0);
        --señales para el divisor de frecuencia
        signal count_1 : std_logic_vector(17 downto 0);
        signal clk190 : std_logic;
        ----------------------------
        begin
        --Divisor de frecuencia
        count_1 <= count_1+1 when rising_edge(cclk);
        clk190 <= count_1(17);
        --control de display en blanco
        aen(3) <= (x(15) or x(14) or x(13) or x(12));          --Activa 3er dígito
                                                               --si hay datos a la entrada
        aen(2) <= (x(15) or x(14) or x(13) or x(12) or         --Activa 3er dígito
                          x(11) or x(10) or x(9) or x(8));     --si hay datos a la entrada
                                                               --o si hay datos para dígito 4
        aen(1) <= (x(15) or x(14) or x(13) or x(12) or         --Activa 2o dígito
                          x(11) or x(10) or x(9) or x(8)) or   --si hay datos a la entrada
                          x(7) or x(6) or x(5) or x(4);        --o si hay datos para dígito 3 ó 4
        aen(0) <= '1'; --Primer dígito (de derecha a izquierda) siempre encendido

        --contador de 2 bits
        cont_2bit: process (clk190,clr)
                begin
                        if (clr = '1') then
                                count <= "00";
                        elsif (rising_edge(clk190)) then
                                count <= count + 1;
                        end if;
                end process cont_2bit;
        --Multiplexor de 4x1
```

```
--Selecciona (con la señal count) 1 de 4 entradas (de 4 bits cada una)
        with count select
                digit <= x(3 downto 0) when "00",
                                x(7 downto 4) when "01",
                                x(11 downto 8) when "10",
                                x(15 downto 12) when others;
--seg7dec
--Convierte un número binario de 4 bits a 7 segmentos
        with digit select
                a_to_g <= "1001111" when "0001",        --1
                                "0010010" when "0010",  --2
                                "0000110" when "0011",  --3
                                "1001100" when "0100",  --4
                                "0100100" when "0101",  --5
                                "0100000" when "0110",  --6
                                "0001111" when "0111",  --7
                                "0000000" when "1000",  --8
                                "0000100" when "1001",  --9
                                "0001000" when "1010",  --A
                                "1100000" when "1011",  --B
                                "0110001" when "1100",  --C
                                "1000010" when "1101",  --D
                                "0110000" when "1110",  --E
                                "0111000" when "1111",  --F
                                "0000001" when others;  --0
--selección del dígito
ancode: process (count)
        begin
                if (aen(conv_integer(count)) = '1') then --convierte el valor de count(1:0) a entero
                        an <= (others => '1');          --asigna '1s' a toda la señal an(3:0) = "1111"
                        an(conv_integer(count)) <= '0';
                else
                        an <= "1111";
                end if;
        end process ancode;
end Behavioral;
```