Universidad Tecnológica de la Mixteca
Cómputo Reconfigurable
Profesor: M.C. Arturo Pablo Sandoval García
Tercer Parcial
Ingeniería en Computación || Grupo: 402-B

17 de junio de 2023

Integrantes:

Elorza Velásquez Jorge Aurelio
González Martínez Ruben Eduardo
Martínez Lorenzo Frida Ximena
Robles Jimenez José Guadalupe

- Nota:Implementar en ISE DESING a nivel de VHDL y descarga en la tarjeta Nexys 2.

Actividad 1.- SELECTOR DE FUNCIONES EN VHDL.

1. Representar, simular e implementar en ISE-Design el siguiente circuito considerando que (A,B,C) son entradas del módulo de ***funciones (Y0, Y1,Y2, Y3)*** y (D0, D1) corresponde a las entradas del decodificador de 2 a 4 donde la línea (D0) corresponde al LSB. La implementación del circuito será de manera combinada en un archivo las compuertas ***en esquemático y el decodificador y el módulo de funciones (Y0, Y1,Y2, Y3) en un módulo en VHDL.***

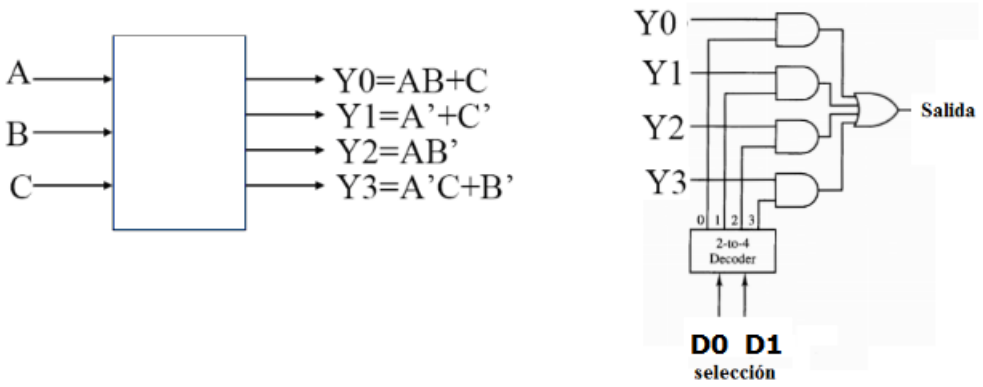
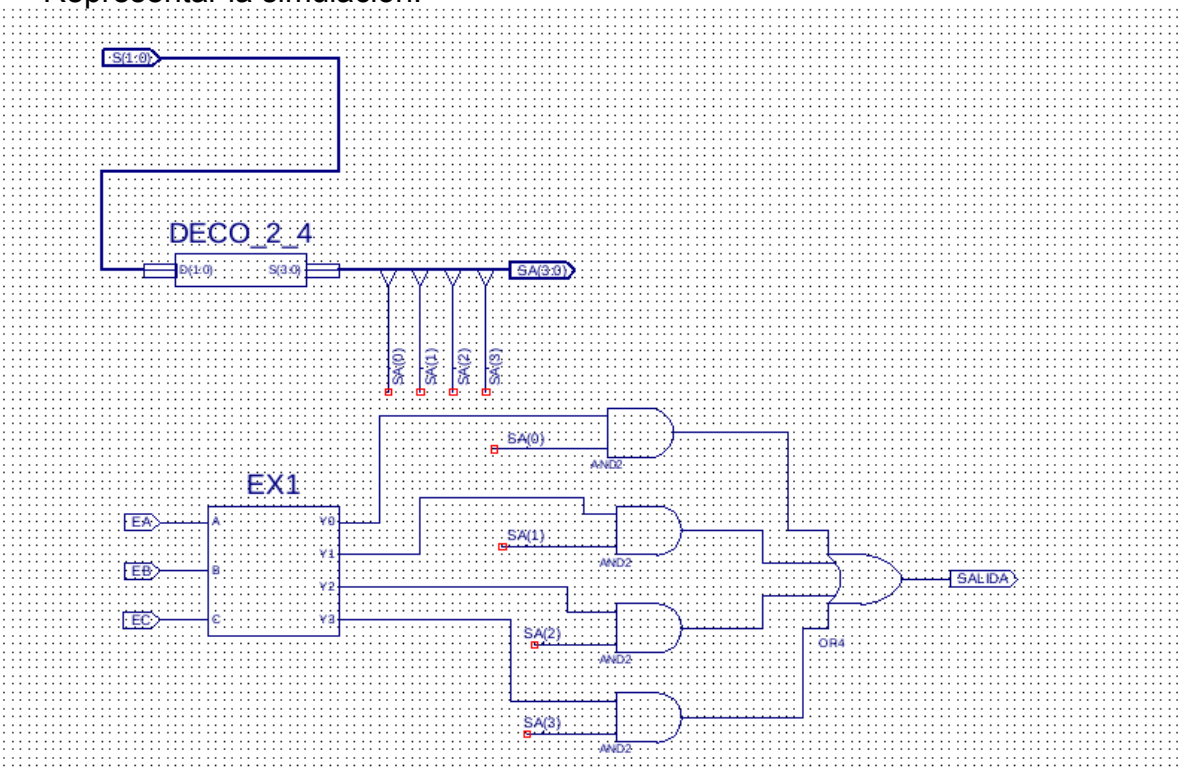


Figura 1

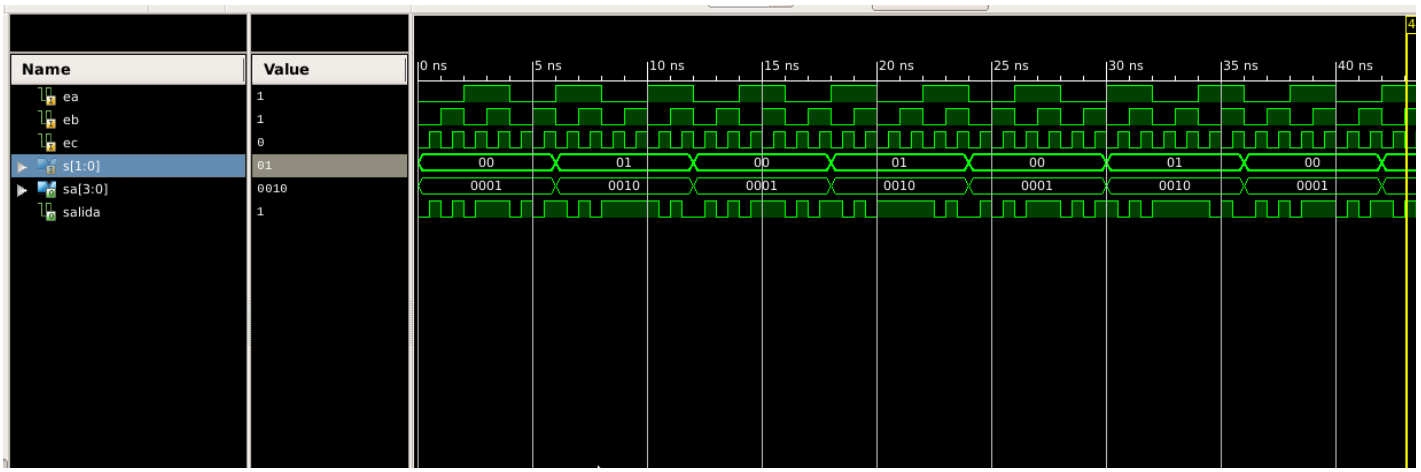
- Adicionar los códigos utilizados en VHDL
- Representar la simulación.



```
1
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  ----------------------------------------------------------------------------------
5  entity DECO_2_4 is
6      Port ( D : in  STD_LOGIC_VECTOR (1 downto 0);
7             S : out  STD_LOGIC_VECTOR (3 downto 0));
8  end DECO_2_4;
9  ----------------------------------------------------------------------------------
10 architecture Behavioral of DECO_2_4 is
11 begin
12
13 WITH D SELECT
14 S <= "0001" WHEN "00",
15       "0010" WHEN "01",
16       "0100" WHEN "10",
17       "1000" WHEN OTHERS;
18
19 end Behavioral;
20
```

```
1  ----------------------------------------------------------------------------------
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  ----------------------------------------------------------------------------------
5  entity EX1 is
6      Port ( A,B,C : in  STD_LOGIC;
7             Y0,Y1,Y2,Y3 : out  STD_LOGIC);
8  end EX1;
9  ----------------------------------------------------------------------------------
10 architecture Behavioral of EX1 is
11 begin
12
13 Y0 <= (A AND B) OR C;
14 Y1 <= (NOT A) OR (NOT C);
15 Y2 <= A AND (NOT B);
16 Y3 <= ((NOT A)AND C) OR (NOT B);
17 end Behavioral;
18
```

```
1  NET "EA" LOC = G18;
2  NET "EB" LOC = H18;
3  NET "EC" LOC = K18;
4
5  NET "S(1)" LOC = K17;
6  NET "S(0)" LOC = L14;
7
8  NET "SALIDA" LOC = J14;
```



Actividad 2.- ALU (Unidad lógica Aritmética)

Utilizando ISE DESIGN en VHDL y la tarjeta NEXYS 2. Represente una ALU, la cual contiene dos números de dos entradas de 2 bits, dos líneas de selección y 4 bits de salida. Las operaciones a realizar son : AND, OR , suma, multiplicación .. Esté circuito debe aceptar dos números decimales (a y b). Cada uno de los números decimales está limitado a los valores (0, 1, 2, 3). El circuito realizará la multiplicación (Res = a * b)y la suma (Res = a+b) adicionalmente las operaciones booleanas AND, OR. El resultado de **la salida se visualiza en un display de la tarjeta**.

- Los datos de salida se visualizarán en los leds de la tarjeta Nexys 2.
- Adicionar el códigos utilizados en VHDL
- Representar la simulación.

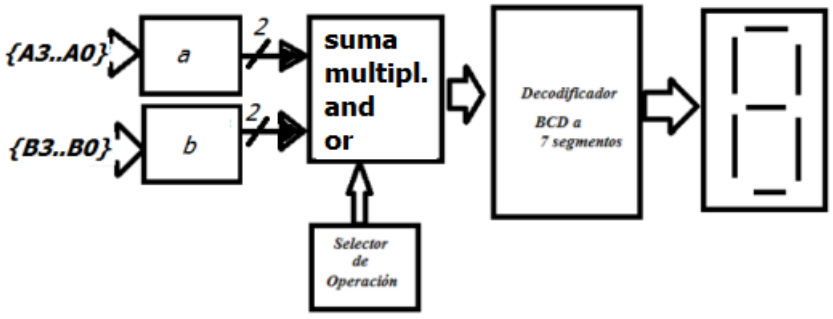
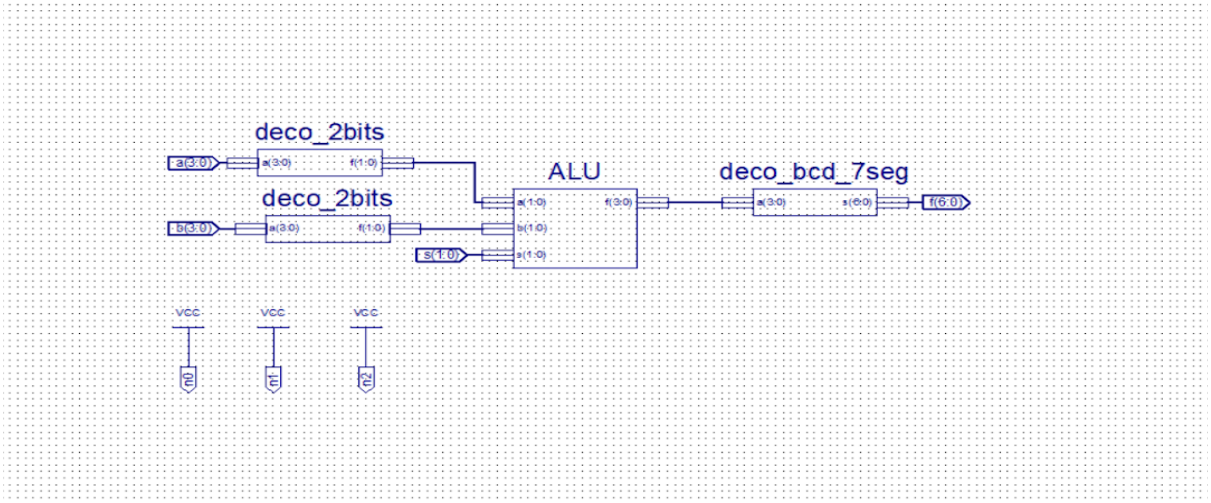


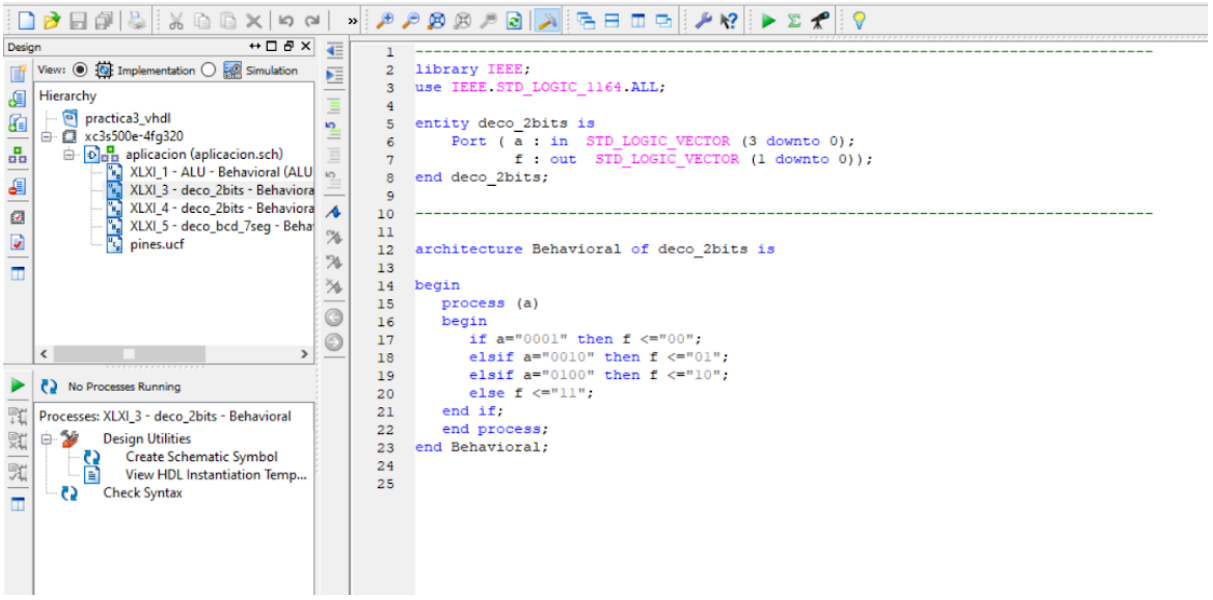
Figura 2. Representación en bloques de la operación de suma y multiplicación

- Adicionar los códigos utilizados en VHDL
- Representar la simulación.

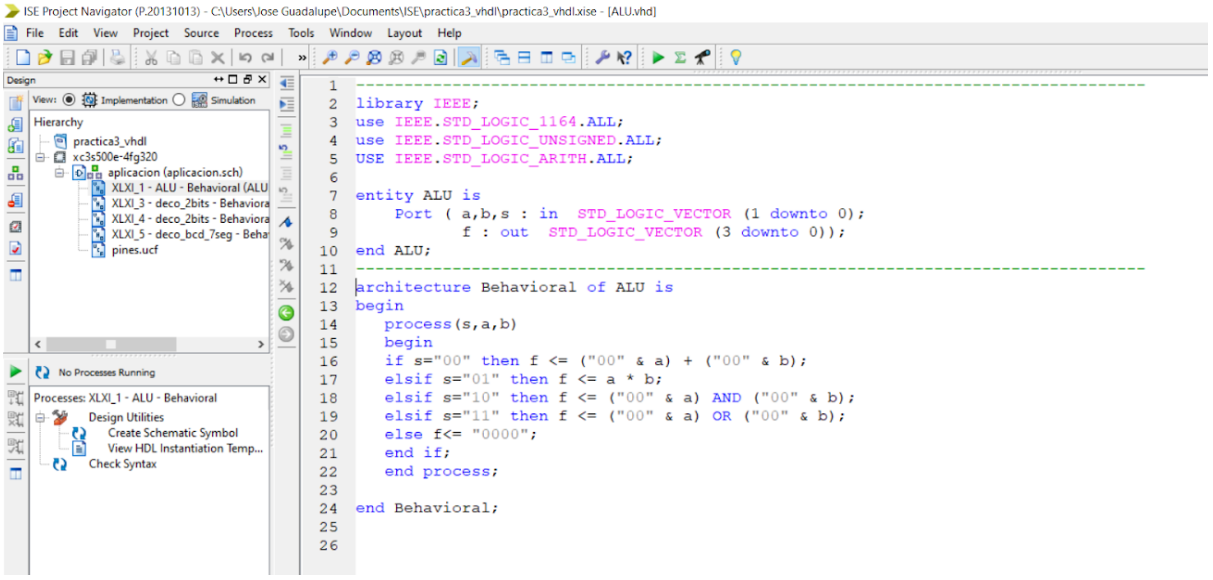
Implementación en forma esquemática en ISE DESIGN:



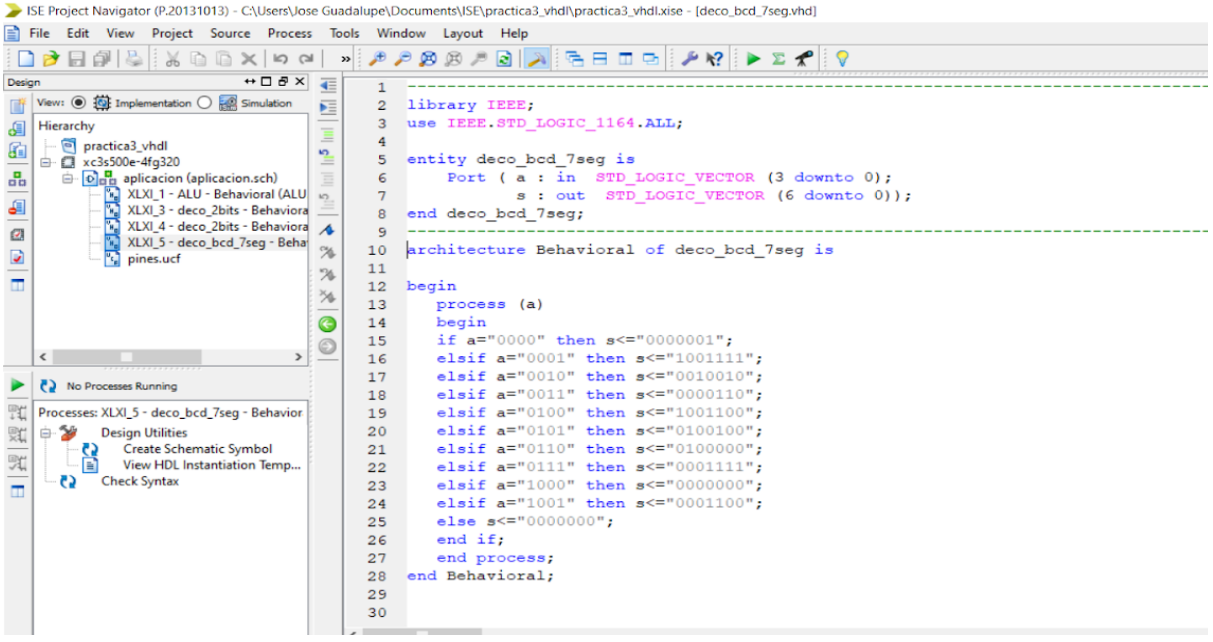
Código en VHDL del decodificador (cajitas pequeñas) :



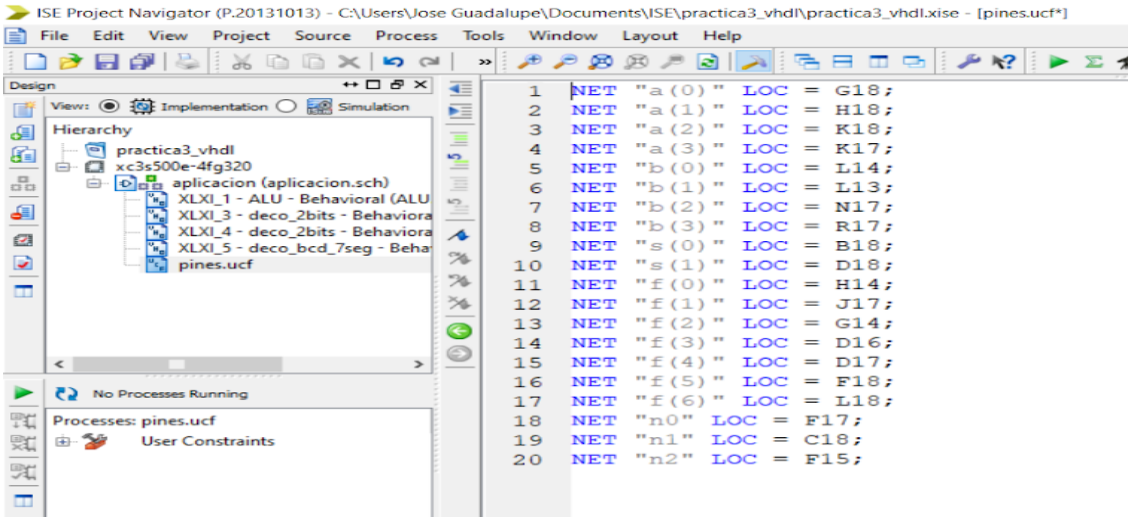
Código de la ALU:



Código del codificador BCD 7 segmentos:

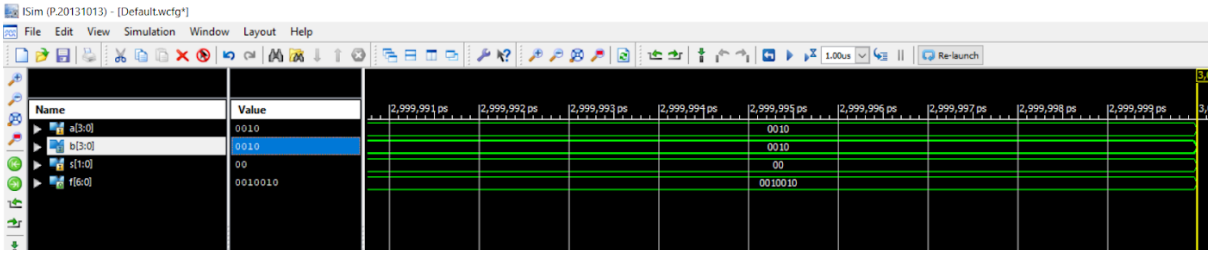


Pines:

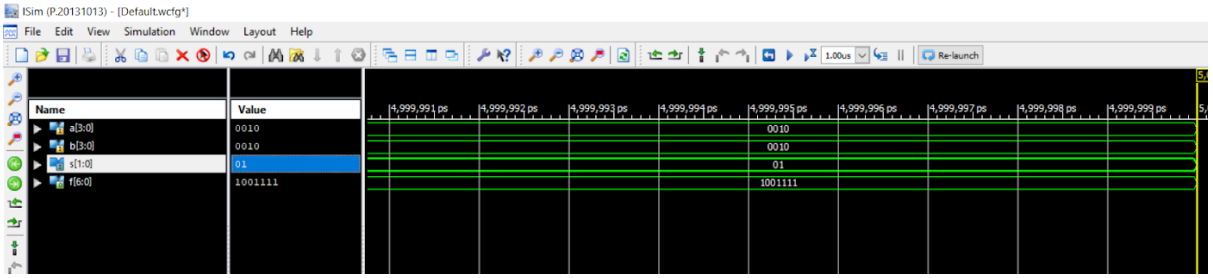


Simulación:

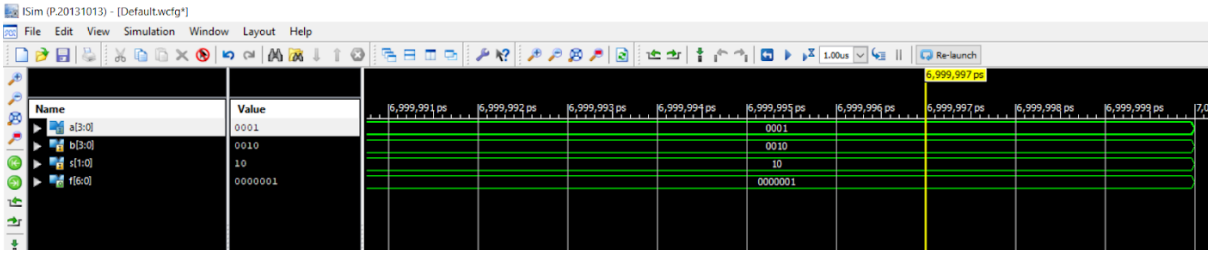
suma 1+1=2



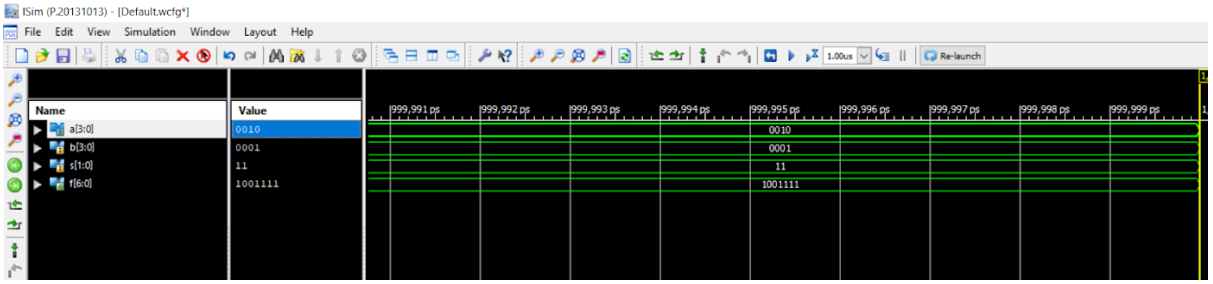
multiplicación 1*1=1



AND (0 AND 1)=0



OR (0 OR 1=1)



Actividad 3.MODULADOR PWM DIGITAL

La función PWM como abreviatura de la modulación por ancho de pulsos, La modulación por ancho de pulsos de una señal o fuente de energía es una técnica en la que se modifica el ciclo de trabajo de una señal periódica. Este circuito está conformado por los interruptores {D0..D7}, como entradas que ingresan a un comparador de 8 bits, así mismo un contador de 8 bits. La salida se visualiza en un led de la tarjeta Nexys2.

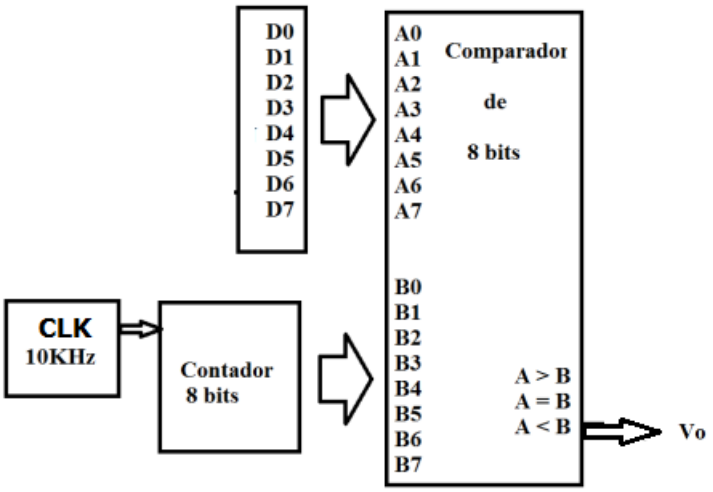


Figura 5. Modulador PWM digital.

Figura 3.- Modulador PWM

- Adicionar los códigos utilizados en VHDL

```
entity proyec3 is
    Port (clk,reset : in  STD_LOGIC;
          ent : in  STD_LOGIC_VECTOR (7 downto 0);
          sal : out  STD_LOGIC);
end proyec3;

architecture Behavioral of proyec3 is
COMPONENT comparador8b
   PORT(a : IN std_logic_vector(7 downto 0);
      b : IN std_logic_vector(7 downto 0);
      s : OUT std_logic
      );
   END COMPONENT;
   COMPONENT c8bist
   PORT(
      reloj : IN std_logic;
      r : IN std_logic;
      salida : OUT std_logic_vector(7 downto 0)
      );
   END COMPONENT;
   signal aux:std_logic_vector(7 downto 0);

begin

   Inst_comparador8b: comparador8b PORT MAP(
      a => ent,
      b =>aux,
      s =>sal
   );
   Inst_c8bist: c8bist PORT MAP(
      reloj =>clk ,
      r =>reset,
      salida => aux
   );
```

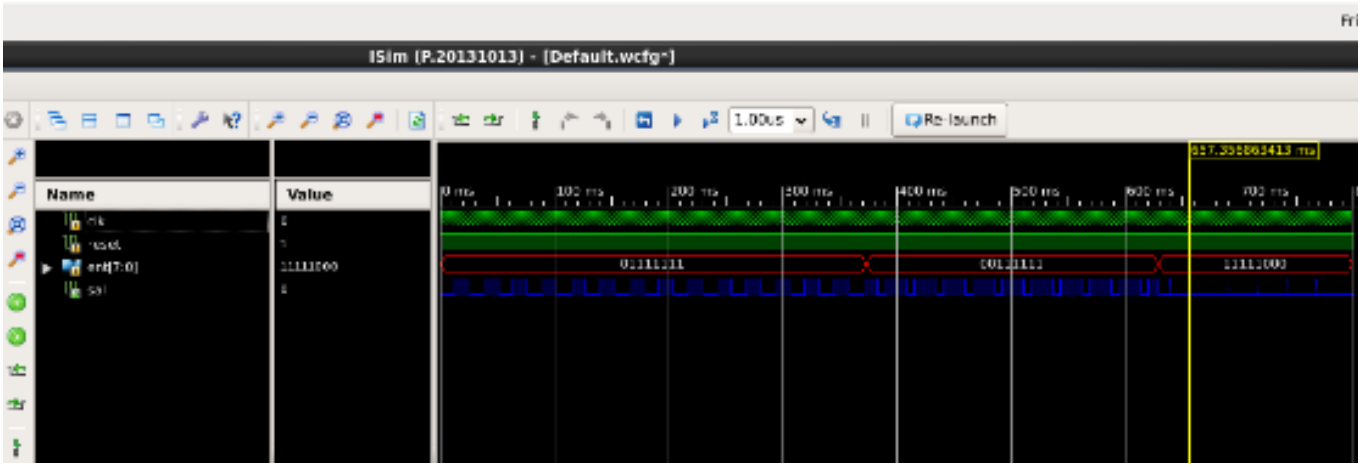
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;


entity comparador8b is
    Port ( a,b : in  STD_LOGIC_VECTOR (7 downto 0);
           s : out  STD_LOGIC);
end comparador8b;

architecture Behavioral of comparador8b is begin
   process (a,b) begin
      if a=b then s<='0';
         elsif (a<b) then s<='1';
         else s<= '0';
         end if;
         end process;
end Behavioral;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- ------------------------
entity c8bist is
    Port ( reloj,r : in  STD_LOGIC;
           salida : out  STD_LOGIC_vector(7 downto 0));
end c8bist;
-- ----------------------------------------
architecture Behavioral of c8bist is
COMPONENT F10khz
   PORT(
      clk : IN std_logic;
      reset : IN std_logic;
      led : OUT std_logic
      );
   END COMPONENT;
cOMPONENT contadorde8bits
   PORT(
      c : IN std_logic;
      q : OUT std_logic_vector(7 downto 0)
      );
   END COMPONENT;
signal aux1:std_logic;
begin
   Inst_F10khz: F10khz PORT MAP(
      clk => reloj,
      reset => r,
      led =>aux1
   );
   Inst_contadorde8bits: contadorde8bits PORT MAP(
      c =>aux1 ,
      q =>salida
   );
end Behavioral;
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity F10khz is
    Port (clk : in  STD_LOGIC;
          reset : in  STD_LOGIC;
          led : out  STD_LOGIC
          );
end F10khz;

architecture contador of F10khz is
   signal contar:integer range 0 to 2499:=0;
   signal Salida:std_logic;
begin
   Divisor_frecuencia: process (reset,clk) begin
   if reset='0' then
      Salida <='0';
      contar <=0;
      elsif rising_edge(clk) then
         if contar = 2499 then
            Salida <= not Salida;
            contar<=0;
         else
            contar<=contar+1;
         end if;
      end if;
   end process;
led<=Salida;
end contador ;
```

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_unsigned.all;

entity contadorde8bits is
    Port ( c : in  STD_LOGIC;

           q : out  STD_LOGIC_vector(7 downto 0));
end contadorde8bits;

architecture Behavioral of contadorde8bits is
   signal aux:std_logic_vector (7 downto 0):="00000000";
begin
conta:process (aux,c) begin
   if (c'event and c ='1') then
      if (aux="11111111") then
          aux<="00000000";
      else
         aux<=aux+1;
      end if;
   end if;
q<=aux;
end process;
end Behavioral;
```

- Representar la simulación.



ACTIVIDAD 4.- Circuito Secuencial en VHDL

Realizar un circuito secuencial en VHDL, el cual está conformado por un divisor de frecuencia, un contador en BCD y un decodificador de BCD a 7 segmentos como se muestra en la siguiente representación a bloques.

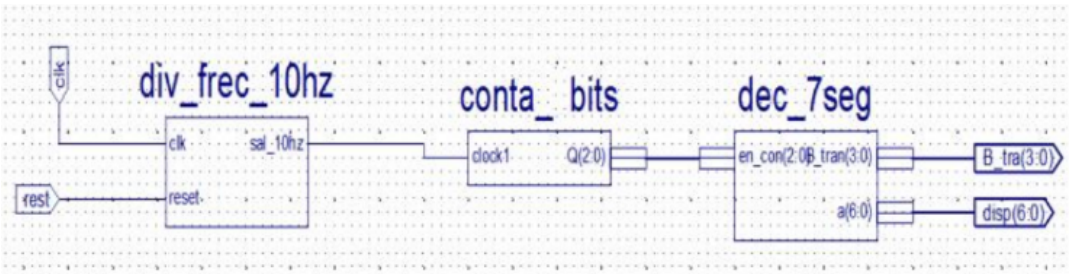


Fig. 4 Contador BCD.

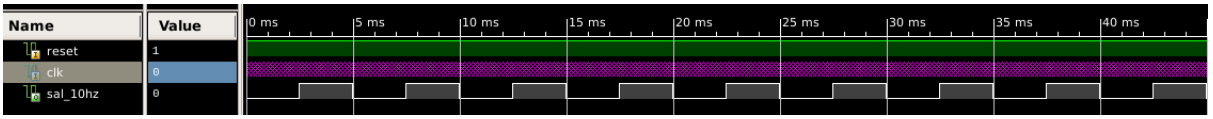
- Habrá que añadir los códigos de los bloques utilizados.
- Representar la simulación.
- En seguida se muestra el archivo de configuración de los pines para descargarlo en la tarjeta.

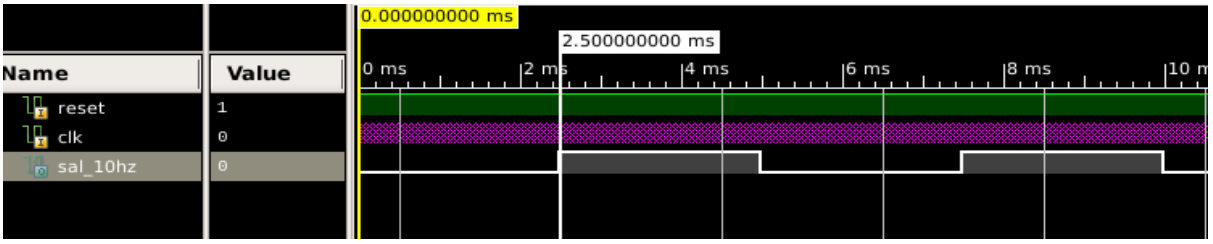
CONFIGURACIÓN EN PINES

```
NET "clk_in" LOC = B8;
NET "rst_in" LOC = G18;

NET "B(0)" LOC = F15;
NET "B(1)" LOC = F17;
NET "B(2)" LOC = H17;
NET "B(3)" LOC = C18;

NET "Dis(6)" LOC = L18;
NET "Dis(5)" LOC = F18;
NET "Dis(4)" LOC = D17;
NET "Dis(3)" LOC = D16;
NET "Dis(2)" LOC = G14;
NET "Dis(1)" LOC = J17;
NET "Dis(0)" LOC = H14;
```

Divisor de Frecuencia 10 Hz.

```vhdl
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
----------------------------------------------------------------------------------
entity DIVISOR2 is
    Port ( CLK,RESET : in  STD_LOGIC;
           SAL_10HZ : out  STD_LOGIC);
end DIVISOR2;
----------------------------------------------------------------------------------
architecture Behavioral of DIVISOR2 is
signal conta: integer range 0 to 24999999:=0;
signal temp: std_logic:='0';
begin
   process (RESET,CLK) is
begin
   if(RESET='0') then
      temp <= '0';
      conta <= 0;
   elsif (rising_edge(CLK)) then
      if (conta=24999999) then       -- Comparator
         temp <= not temp;
         conta <= 0;
      else
         conta <= conta+1;          --Contador
      end if;
   end if;
end process;

SAL_10HZ <= temp;

end Behavioral;
```





Como se aprecia en la simulación se necesitan un total de 2.499999ms para que cambie de estado la salida, esto debido al factor de escalamiento, con un CLK a 1ns.

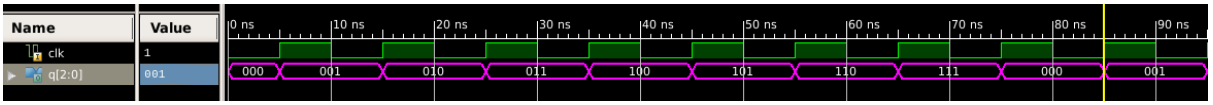
$$Escala = \frac{fentrada}{fdeseada} = \frac{50\ MHz}{10\ Hz} = 5M$$

Por lo tanto, el contador para el divisor de frecuencia tiene como función generar la señal de salida de 10Hz cada 2500000 ciclos.

$$Escala = \frac{5M}{2} - 1 = 2,499,999$$

Contador de 3 bits.

```
1  ----------------------------------------------------------------------------------
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4  use IEEE.STD_LOGIC_UNSIGNED.ALL;
5  ----------------------------------------------------------------------------------
6  entity CONTADOR is
7      Port ( CLK : in  STD_LOGIC;
8             Q : out  STD_LOGIC_VECTOR (2 downto 0));
9  end CONTADOR;
10 ----------------------------------------------------------------------------------
11 architecture Behavioral of CONTADOR is
12    signal aux : std_logic_vector (2 downto 0) := "000";
13 begin
14     conta: process (aux, CLK)
15     begin
16         if (rising_edge(CLK)) then
17             if (aux = "111") then
18                 aux <= "000";
19             else
20                 aux <= aux+1;
21             end if;
22         end if;
23         Q <= aux;
24     end process conta;
25 end Behavioral;
```



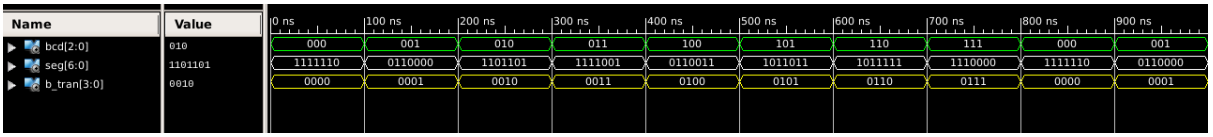
Esta entidad representa un contador de 3 bits sincronizado por un reloj. Toma una entrada de reloj clock y proporciona una salida de 3 bits llamada Q, que representa el estado actual del contador. En cada flanco de subida del reloj, el proceso dentro de la arquitectura incrementa el valor actual del contador en 1, siguiendo un patrón de conteo ascendente de 0 a 7. Si el contador alcanza el valor máximo (7), se reinicia a 0.

Display BCD 7 segmentos.

Para tarjetas de desarrollo con los displays de 7 segmentos conectados en paralelo, se

requiere adicionar un vector de salida y código extra. (Este vector es “*B_tran*”)

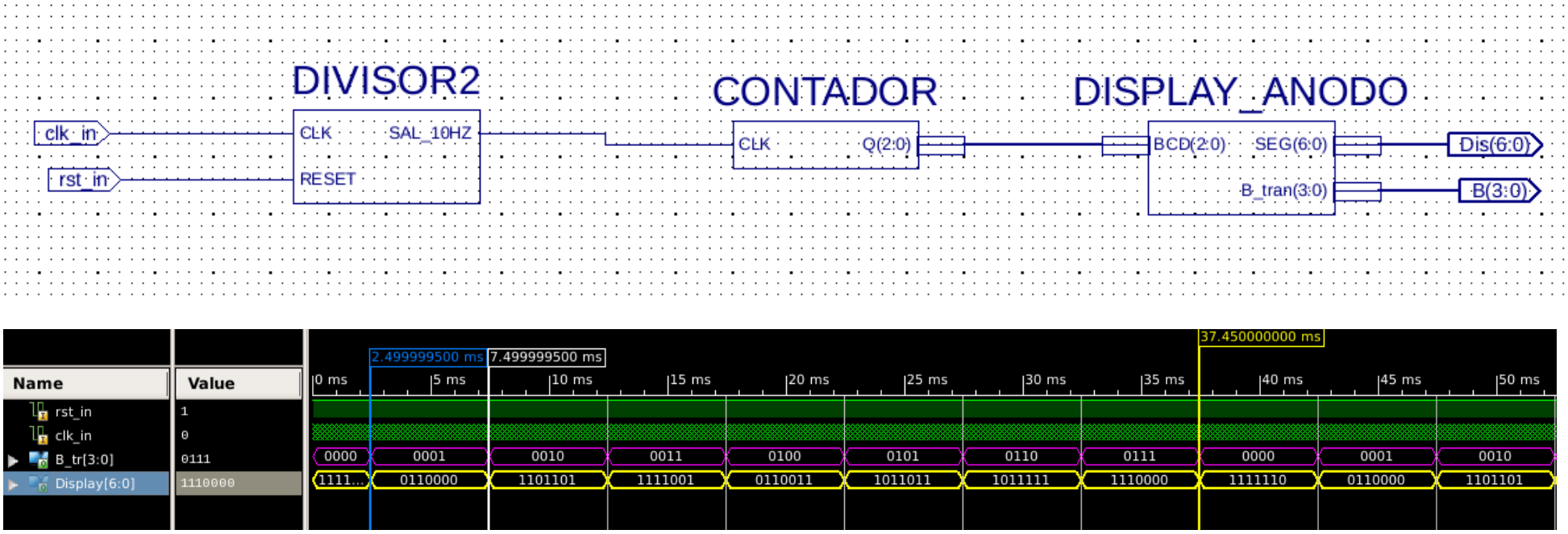
```
----------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
----------------------------------------------------------------------
entity BCD_to_7seg is
    Port ( BCD : in  STD_LOGIC_VECTOR (2 downto 0);
           SEG : out  STD_LOGIC_VECTOR (6 downto 0);
           B_tran : out  STD_LOGIC_VECTOR (3 downto 0));
end BCD_to_7seg;
----------------------------------------------------------------------
architecture Behavioral of BCD_to_7seg is
begin
    process (BCD)
    begin
        case BCD is
            when "000" =>
                SEG <= "1111110"; -- 0
                B_tran <= "0000";
            when "001" =>
                SEG <= "0110000"; -- 1
                B_tran <= "0001";
            when "010" =>
                    SEG <= "1101101"; -- 2
                    B_tran <= "0010";
                when "011" =>
                    SEG <= "1111001"; -- 3
                    B_tran <= "0011";
                when "100" =>
                    SEG <= "0110011"; -- 4
                    B_tran <= "0100";
                when "101" =>
                    SEG <= "1011011"; -- 5
                    B_tran <= "0101";
                when "110" =>
                    SEG <= "1011111"; -- 6
                    B_tran <= "0110";
                when "111" =>
                    SEG <= "1110000"; -- 7
                    B_tran <= "0111";
                when others =>
                    SEG <= "0000000"; -- Invalid input
                    B_tran <= "0000";
            end case;
        end process;
end Behavioral;
```



```
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
--------------------------------------------------------------------------------
ENTITY TB_DECO IS
END TB_DECO;

ARCHITECTURE behavior OF TB_DECO IS
    -- Component Declaration for the Unit Under Test (UUT)
    COMPONENT BCD_to_7seg
    PORT(
         BCD : IN  std_logic_vector(2 downto 0);
         SEG : OUT  std_logic_vector(6 downto 0);
         B_tran : OUT  std_logic_vector(3 downto 0)
        );
    END COMPONENT;

   --Inputs
   signal BCD : std_logic_vector(2 downto 0) := (others => '0');
   --Outputs
   signal SEG : std_logic_vector(6 downto 0);
   signal B_tran : std_logic_vector(3 downto 0);

BEGIN
   -- Instantiate the Unit Under Test (UUT)

   uut: BCD_to_7seg PORT MAP (
          BCD => BCD,
          SEG => SEG,
          B_tran => B_tran
        );

   -- Stimulus process
   stim_proc0: process
   begin
      wait for 100 ns;
         BCD(0) <= NOT BCD(0);
   end process;

      stim_proc1: process
   begin
      wait for 200 ns;
         BCD(1) <= NOT BCD(1);
   end process;

      stim_proc2: process
   begin
      wait for 400 ns;
         BCD(2) <= NOT BCD(2);
   end process;
END;
```

Para este bloque el “BCD” genera los números en decimal del 0 al 7 (Como se muestra en SEG(6:0)) y vuelve a empezar en 0, esto sin importar haber generado todas las combinaciones de BCD(2:0).

FINAL:



*Nota: Para el bloque del “DIVISOR2” se le incrementó el tiempo debido a que la salida generaba cambios muy rápidos, así que al aumentar el tiempo ya era visible en la realidad los números en el display de 7 segmento