Universidad Politécnica de Madrid
ETSI de Telecomunicación
Departamento de Ingeniería Electrónica

Circuitos Electrónicos (CELT)

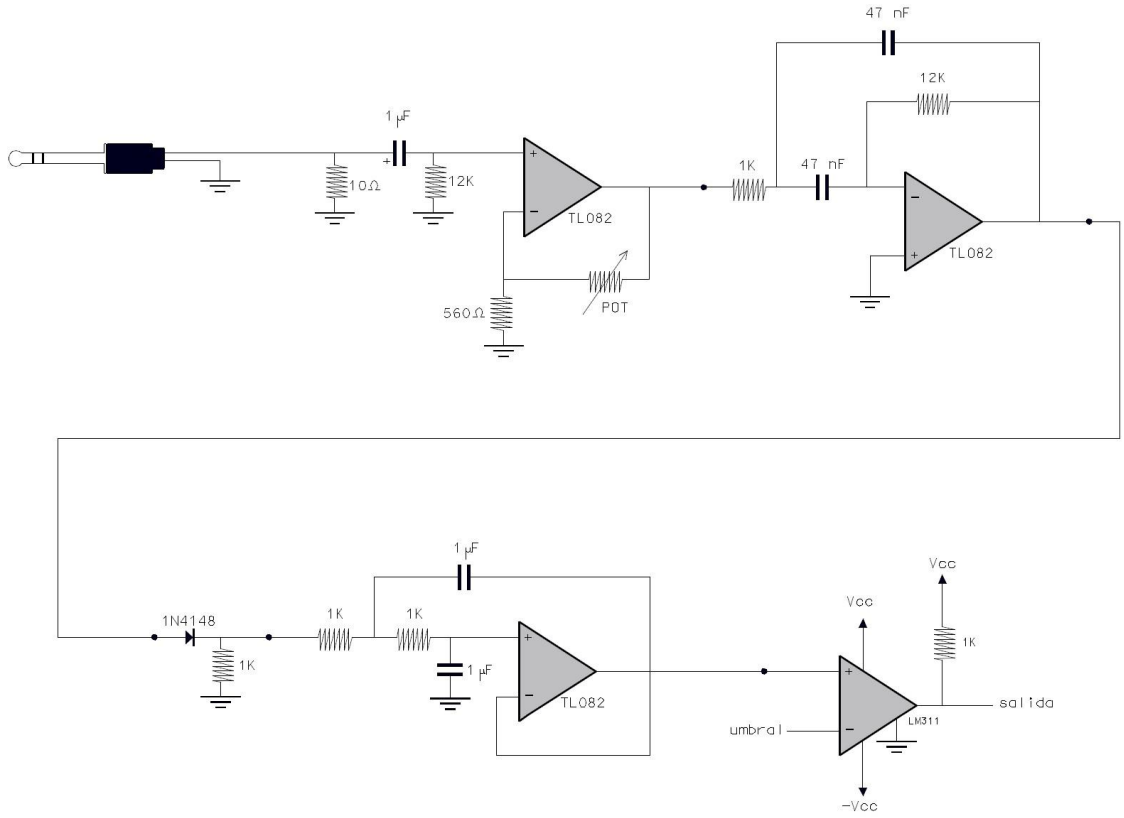
Curso 2017-2018

Memoria final

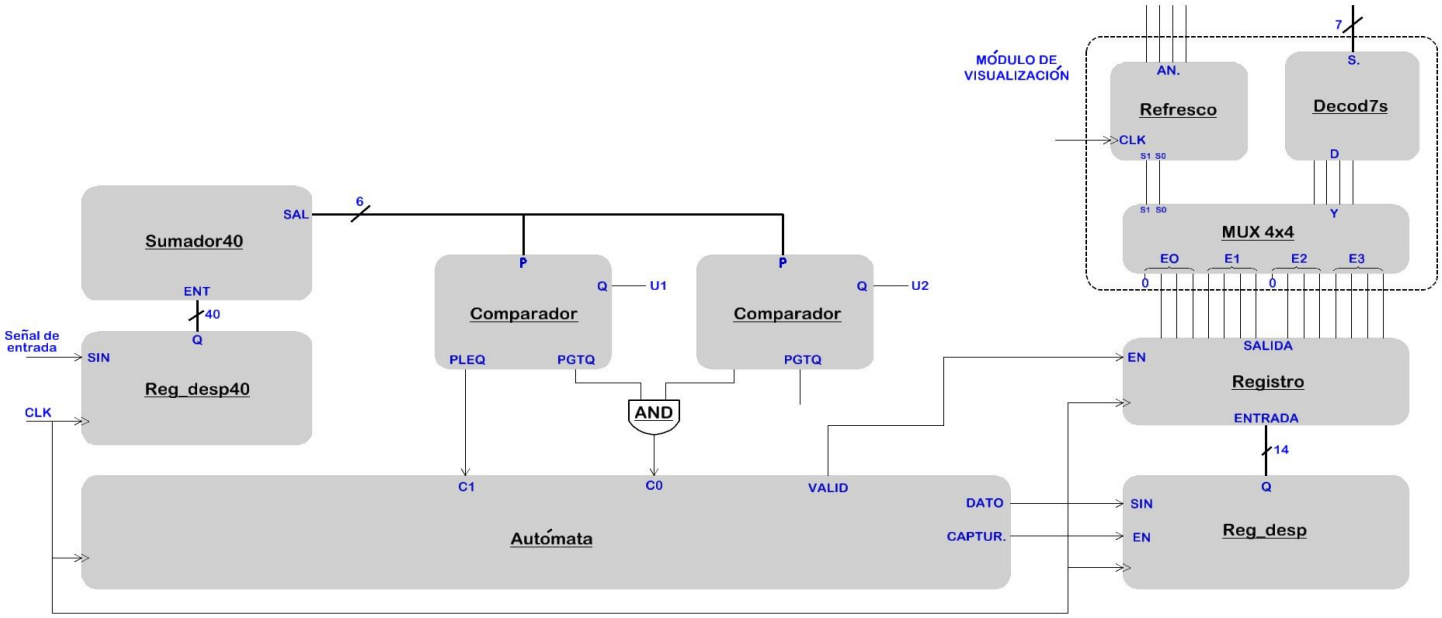
	Apellidos	Nombre
Alumno 1	Moreno Miguel	Andrés
Alumno 2	Escat Juanes	Javier

Código de pareja	MT-33

1. ESQUEMA COMPLETO DEL CIRCUITO ANALÓGICO



2. ESQUEMA COMPLETO DEL CIRCUITO DIGITAL

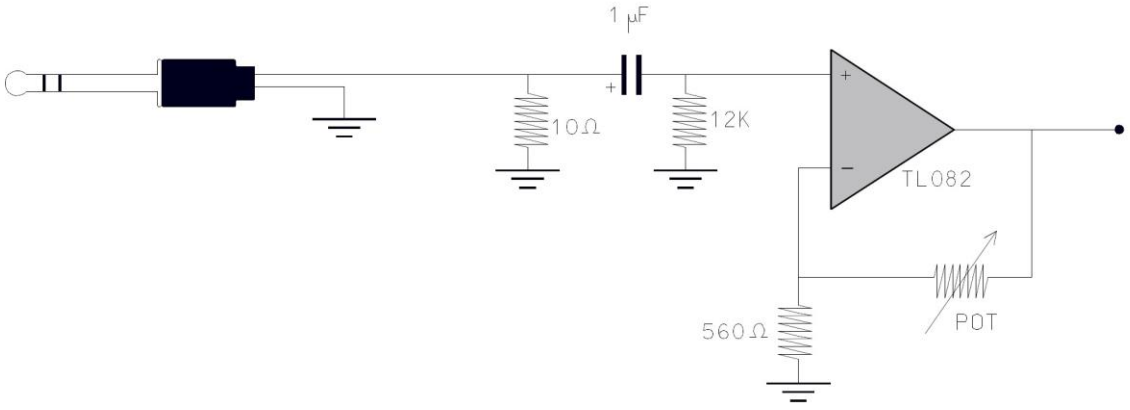


3. DISEÑO DETALLADO DEL CIRCUITO ANALÓGICO

3.1 Etapa 1: Acondicionador de señal

*Esta etapa consta de un **filtro paso alto** y un **amplificador** de la señal MP3. El filtro se compone de un condensador de 1µF y una resistencia R_f.*

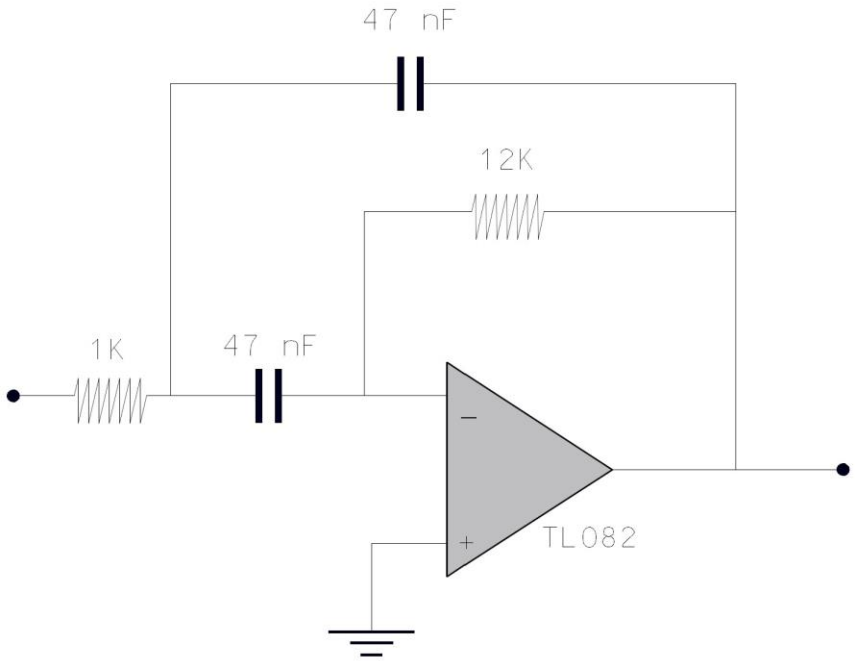
La frecuencia de corte ($f_C = \frac{1}{2\pi \cdot Rf \cdot C}$) debe estar entre 10 y 15 Hz, elegimos una R_f = 12kΩ que nos da una f_C = 13'2629Hz. Para el amplificador utilizamos una R_a = 560Ω y un POT = 10kΩ, con lo que conseguimos una señal de 1Vpp a la salida ajustando el potenciómetro.



3.2 Etapa 2: Filtro paso banda

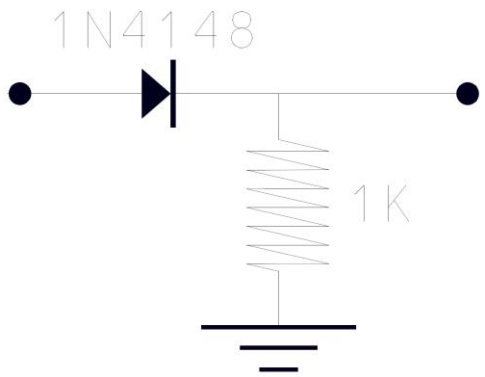
*Tenemos un **filtro paso banda** con una frecuencia central ($f_0 = \frac{1}{2\pi\sqrt{R1 \cdot R2 \cdot C}}$) igual a 1kHz y una ganancia ($G = \frac{-R2}{2 \cdot R1}$) de valor 6. Con estas incógnitas, elegimos resistencias R1 = 1kΩ y R2 = 12kΩ, valores dentro del rango de resistencias recomendadas, y el condensador C = 47nF.*

La función de transferencia es H(jw)= $\frac{-jw\cdot\frac{1}{R1C}}{-w^2+jw\frac{2}{R2C}+\frac{1}{R1R2C^2}}$



3.3 Etapa 3: Rectificador

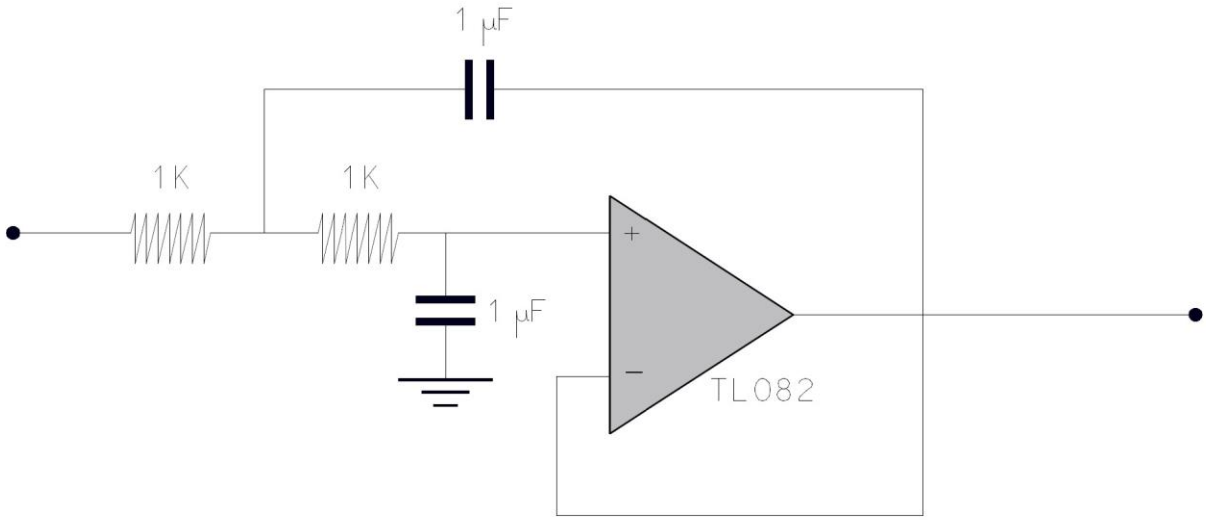
Para quedarnos con la mitad de voltaje positivo de la señal a la salida del paso banda utilizamos un diodo acorde a la frecuencia de la portadora.



3.4 Etapa 4: Filtro paso bajo

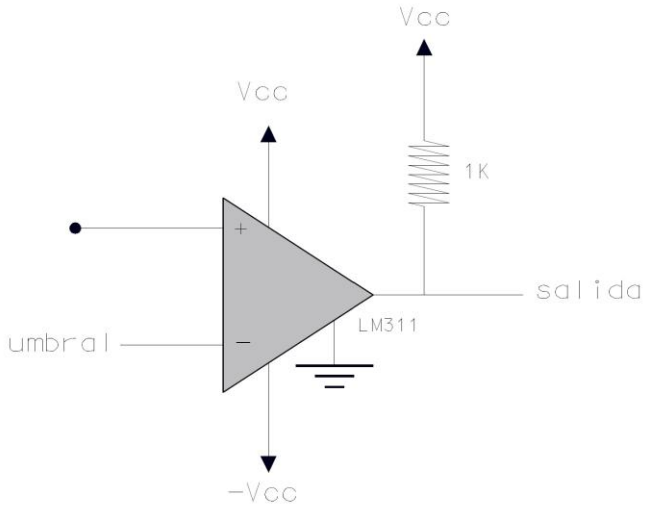
Para este filtro tenemos una ganancia (G=1) y un factor de calidad (Q=0.5) para que la frecuencia de corte coincida con $f_0 = 2\pi w_0$ *, siendo* $w_0=1/RC=1000$. *Con estos datos hemos escogido una* $R=1k\Omega$ *y un* $C = 1\mu F$.

La función de transferencia es H(jw)= $\frac{G\cdot\frac{1}{R^2C^2}}{-w^2+jw\frac{3-G}{RC}+\frac{1}{R^2C^2}}$



3.5 Etapa 5: Comparador

*Por último, encontramos el **comparador**, que produce una señal a Vcc (5V) o masa (0V) en función de si la señal de entrada se encuentra por encima o por debajo de un umbral, definido por un potenciómetro, que debe situarse entre ambos picos de la señal de entrada.*



4. DISEÑO DETALLADO DEL CIRCUITO DIGITAL

4.1 Módulo 1: Generador de reloj

```
--------------------------------------------------------------------------------
-- GENERADOR DE RELOJ A 40 HZ
--
-- Este generador se encarga de producir una
-- señal de reloj a 40 Hz a partir de la señal
-- de entrada de la FPGA a 50 MHz, a través de
-- cuentas sucesivas de los flancos de reloj
-- de esta señal.
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity gen_reloj is
    Port ( CLK : in  STD_LOGIC;             -- Reloj de la FPGA
           CLK_OUT : out  STD_LOGIC);               -- Reloj de muestreo
end gen_reloj;

architecture Behavioral of gen_reloj is

        signal cont_M : STD_LOGIC_VECTOR (31 downto 0):= (others=>'0');    -- contador 1
        signal S_M : STD_LOGIC :='0';                                      -- señal auxiliar para
                                                                  -- generar reloj de salida


begin

PROC_CONT : process (CLK)                   -- proceso que genera una señal de reloj en función
                                            -- de la de entrada

begin
        if CLK'event and CLK='1' then       -- cada flanco positivo de reloj el contador aumenta
                cont_M <= cont_M + 1;
                if cont_M >= 625000 then    -- división de frecuencia a 40 Hz
                        S_M <=not S_M;      -- flanco en señal de reloj auxiliar
                        cont_M <=(others=>'0');      -- contador vuelve a cero
                end if;
        end if;
end process;

CLK_OUT<=S_M;                               -- la señal auxiliar de reloj se carga a la salida

end Behavioral;
```

4.2 Módulo 2: Registro de desplazamiento de 40 bits

```
--------------------------------------------------------------------------------
-- REGISTRO DE DESPLAZAMIENTO DE 40 BITS
--
-- Este registro se encarga de recoger y
-- almacenar los últimos 40 bits que le
-- han llegado por su entrada, y los
-- devuelve por otra salida de 40 bits.
--------------------------------------------------------------------------------
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;


entity reg_desp40 is
    Port ( SIN : in  STD_LOGIC;                              -- Señal de entrada
           CLK : in  STD_LOGIC;                              -- Señal de reloj de muestreo
           Q : out  STD_LOGIC_VECTOR (39 downto 0));               -- Salida del registro
end reg_desp40;

architecture Behavioral of reg_desp40 is
       signal Qs : STD_LOGIC_VECTOR (39 downto 0):= (others=>'0');-- Señal auxiliar del registro

begin
PROC_REG : process (CLK)             -- en este proceso se carga cada bit en el espacio
                            -- adyacente y por último la entrada en último lugar

begin
       if (CLK'event and CLK='1') then
              Qs(0)<=Qs(1);
              Qs(1)<=Qs(2);
              Qs(2)<=Qs(3);
              Qs(3)<=Qs(4);
              Qs(4)<=Qs(5);
              Qs(5)<=Qs(6);
              Qs(6)<=Qs(7);
              Qs(7)<=Qs(8);
              Qs(8)<=Qs(9);
              Qs(9)<=Qs(10);
              Qs(10)<=Qs(11);
              Qs(11)<=Qs(12);
              Qs(12)<=Qs(13);
              Qs(13)<=Qs(14);
              Qs(14)<=Qs(15);
              Qs(15)<=Qs(16);
              Qs(16)<=Qs(17);
              Qs(17)<=Qs(18);
              Qs(18)<=Qs(19);
              Qs(19)<=Qs(20);
              Qs(20)<=Qs(21);
              Qs(21)<=Qs(22);
              Qs(22)<=Qs(23);
              Qs(23)<=Qs(24);
              Qs(24)<=Qs(25);
              Qs(25)<=Qs(26);
              Qs(26)<=Qs(27);
              Qs(27)<=Qs(28);
              Qs(28)<=Qs(29);
              Qs(29)<=Qs(30);
              Qs(30)<=Qs(31);
              Qs(31)<=Qs(32);
              Qs(32)<=Qs(33);
              Qs(33)<=Qs(34);
              Qs(34)<=Qs(35);
              Qs(35)<=Qs(36);
              Qs(36)<=Qs(37);
              Qs(37)<=Qs(38);
              Qs(38)<=Qs(39);
              Qs(39)<= SIN;
       end if;
end process;

Q<=Qs;                        -- se carga la señal auxiliar en la salida
end Behavioral;
```

4.3 Módulo 3: Comparador

```
-- COMPARADOR
--
-- Este módulo se encarga de comparar las dos señales
-- que tiene a su entrada y determinar si la segunda es
-- menor que la primera o si, por el contrario, la
-- primera es menor o igual que la segunda.
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity comparador is
    Port ( P : in  STD_LOGIC_VECTOR (5 downto 0);          -- Señal que nos interesa medir
           Q : in  STD_LOGIC_VECTOR (5 downto 0);          -- Umbral de referencia de comparación
           PGTQ : out  STD_LOGIC;                  -- Salida que representa P>Q
           PLEQ : out  STD_LOGIC);                 -- Salida que representa P≤Q
end comparador;

architecture Behavioral of comparador is

begin

process(P,Q)                  -- proceso que compara las señales P y Q y determina cuál es mayor
begin
       if(P>Q) then           -- cuando P>Q, se activa PGTQ y se anula PLEQ
              PGTQ<='1';
              PLEQ<='0';
       else                   -- en caso contrario, se activa PLEQ y se anula PGTQ
              PGTQ<='0';
              PLEQ<='1';
       end if;

end process;

end Behavioral;
```

4.4 Módulo 4: Puerta AND

```
----------------------------------------------------------------------------------
-- PUERTA AND DE DOS ENTRADAS
--
-- Este módulo se encarga de realizar la operación
-- lógica AND entre las dos señales de su entrada.
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity AND_2 is
    Port ( A : in  STD_LOGIC;               -- Señal de entrada 1
           B : in  STD_LOGIC;               -- Señal de entrada 2
           S : out  STD_LOGIC);             -- Señal de salida A AND B
end AND_2;

architecture Behavioral of AND_2 is

begin
       S<=A and B;           -- a la salida se conecta el resultado de A AND B

end Behavioral;
```

4.5 Módulo 5: Autómata

```
----------------------------------------------------------------------------------
-- AUTÓMATA DE CONTROL DE MOORE
```

```
--
-- Este autómata tiene varias funciones
-- la primera es sincronizar el circuito
-- con la señal de entrada cada vez que se
-- recibe un dato de tipo SYNC, además
-- captura los datos de entrada en un registro
-- y los valida cada vez que llega una SYNC.
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity automata is
    Port ( CLK : in  STD_LOGIC;                    -- Señal de reloj del Sistema
           C0 : in  STD_LOGIC;                     -- Señal de entrada 1
           C1 : in  STD_LOGIC;                     -- Señal de entrada 2
           DATO : out  STD_LOGIC;           -- Señal de salida que lleva el dato obtenido
           CAPTUR : out  STD_LOGIC;         -- Señal de activación del registro de desplazamiento
           VALID : out  STD_LOGIC);         -- Señal de activación del registro de validación
end automata;

architecture Behavioral of automata is
       type TIPO_ESTADO is (ESP_SYNC,AVAN_ZM,MUESTREO,DATO0,DATO1,DATOSYNC);     -- 6 estados
       signal ST : TIPO_ESTADO:= ESP_SYNC ;               -- señal que recoge el estado actual
       signal salidas : STD_LOGIC_VECTOR (2 downto 0) :="000";    -- señal auxiliar de salida

begin
process (CLK)                -- el proceso evalúa el estado del autómata y actúa en consecuencia
       variable cont : STD_LOGIC_VECTOR (7 downto 0):="00000000"; -- contador para los estados
                                                                   -- recursivos del autómata
begin
       if (CLK'event and CLK = '1') then
              case ST is
                     when ESP_SYNC =>                    -- esperamos una SYNC
                            if(C0='0' and C1='0') then   -- si llega una SYNC pasamos al
                            ST<=AVAN_ZM;                 -- siguente estado
                            else ST<=ESP_SYNC;           -- sino nos mantenemos en el mismo
                            end if;
                     when AVAN_ZM=>               -- en este estado sólo esperamos 20 ciclos
                            cont:=cont+1;
                            if(cont=20) then      -- después pasaremos al estado siguiente
                                   cont:=(others=>'0');
                                   ST<=MUESTREO;
                            else
                                   ST<=AVAN_ZM;
                            end if;
                     when MUESTREO=>              -- en el estado de MUESTREO, el autómata
                            cont:=cont+1;         -- espera a que se estabilicen los módulos
                                                  -- anteriores del circuito y después evalúa
                            if(cont=39) then      -- si le ha llegado un 1, un 0 o una SYNC
                                   cont:=(others=>'0');
                                   if(C0='1' and C1='0') then    -- si le llega un 0, DATO0
                                          ST<=DATO0;
                                   end if;
                                   if(C0='0' and C1='1') then    -- si le llega un 1, DATO1
                                          ST<=DATO1;
                                   end if;
                                   if(C0='0' and C1='0') then    -- si le llega SYNC, DATOSYNC
                                          ST<=DATOSYNC;
                                   end if;
                            else
                                   ST<=MUESTREO;
                            end if;

                     when DATO0 =>                -- volvemos a MUESTREO en el siguiente ciclo
                            ST<=MUESTREO;
                     when DATO1 =>
```

```
                        ST<=MUESTREO;
                    when DATOSYNC =>
                        ST<=MUESTREO;
                end case;
        end if;
end process;

with ST select
        salidas<=                   -- el primer bit corresponde al dato que llega (0, 1), el
        "000" when ESP_SYNC,        -- segundo lo transmite al registro de desplazamiento y el
        "000" when AVAN_ZM,         -- tercero (activo cuando llega una SYNC) transmite los
        "000" when MUESTREO,        -- valores del registro de desplazamiento al
        "010" when DATO0,           -- registro de validación
        "110" when DATO1,
        "001" when DATOSYNC,
        "000" when others;

DATO<=salidas(2);           -- cada parte de la salida auxiliar se conecta donde corresponde
CAPTUR<=salidas(1);
VALID<=salidas(0);

end Behavioral;
```

4.6 Módulo 6: Registro de desplazamiento de 14 bits

```
----------------------------------------------------------------------------------
-- REGISTRO DE DESPLAZAMIENTO DE 14 BITS
--
-- Este registro se encarga de recoger y
-- almacenar los últimos 14 bits que le
-- han llegado por su entrada, y los
-- devuelve por otra salida de 14 bits.
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity reg_desp is
    Port ( SIN : in  STD_LOGIC;                             -- Señal de entrada
           CLK : in  STD_LOGIC;                             -- Señal de reloj de muestreo
           EN : in  STD_LOGIC;                              -- Enable
           Q : out  STD_LOGIC_VECTOR (13 downto 0));        -- Señal de salida
end reg_desp;

architecture Behavioral of reg_desp is

        signal Qs : STD_LOGIC_VECTOR (13 downto 0):= (others=>'0');-- señal auxiliar del registro
begin
PROC_REG : process (CLK)                    -- en este proceso se carga cada bit en el espacio
                                            -- adyacente y por último la entrada en último lugar
begin
        if (CLK'event and CLK='1' and EN='1') then
                Qs(0)<=SIN;
                Qs(1)<=Qs(0);
                Qs(2)<=Qs(1);
                Qs(3)<=Qs(2);
                Qs(4)<=Qs(3);
                Qs(5)<=Qs(4);
                Qs(6)<=Qs(5);
                Qs(7)<=Qs(6);
                Qs(8)<=Qs(7);
                Qs(9)<=Qs(8);
                Qs(10)<=Qs(9);
                Qs(11)<=Qs(10);
                Qs(12)<=Qs(11);
                Qs(13)<=Qs(12);
        end if;

Q<=Qs;                      -- se carga la señal auxiliar en la salida
```

```
end process;

end Behavioral;
```

4.7 Módulo 7: Registro de validación

```
----------------------------------------------------------------------------------
-- REGISTRO DE VALIDACIÓN
--
-- Este registro carga su entrada a la salida
-- cada vez que se activa su señal de enable.
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity registro is
    Port ( ENTRADA : in  STD_LOGIC_VECTOR (13 downto 0):= (others=>'0');  -- Señal de entrada
           SALIDA : out  STD_LOGIC_VECTOR (13 downto 0);                  -- Señal de salida
           EN : in  STD_LOGIC;                                            -- Enable
           CLK : in  STD_LOGIC);                                          -- Señal de reloj
end registro;

architecture Behavioral of registro is

begin

PROC_REG : process (CLK)
begin
       if (CLK'event and CLK='1' and EN='1') then   -- si el enable está activo y hay un flanco
              SALIDA<=ENTRADA;                      -- de reloj se carga la entrada en la salida
       end if;
end process;

end Behavioral;
```

4.8 Módulo 8: MUX 4x4

```
----------------------------------------------------------------------------------
-- MULTIPLEXOR DE 4 ENTRADAS
--
-- El multiplexor envía una de las 4 entradas a la
-- salida en función de la entrada de control S.
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity MUX4x4 is
    Port ( E0 : in  STD_LOGIC_VECTOR (3 downto 0);          -- Entrada 1
           E1 : in  STD_LOGIC_VECTOR (3 downto 0);          -- Entrada 2
           E2 : in  STD_LOGIC_VECTOR (3 downto 0);          -- Entrada 3
           E3 : in  STD_LOGIC_VECTOR (3 downto 0);          -- Entrada 4
           S : in  STD_LOGIC_VECTOR (1 downto 0);           -- Entrada de control
           Y : out  STD_LOGIC_VECTOR (3 downto 0));         -- Salida
end MUX4x4;

architecture Behavioral of MUX4x4 is

begin

with S select Y<=             -- según la señal de control, se conecta cada entrada a la salida
       E0 when "00",
       E1 when "01",
       E2 when "10",
       E3 when "11",
       E3 when others;

end Behavioral;
```

4.9 Módulo 9: Refresco

```
----------------------------------------------------------------------------------
```

```
-- MÓDULO DE REFRESCO
--
-- Este módulo se encarga de generar una señal de
-- reloj lo suficientemente rápida para que no sea
-- apreciable por el ojo (elegimos una a 500 Hz) y
-- generar una señal para la activación de los
-- displays y otra para el control del multiplexor.
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity refresco is
    Port ( CLK : in  STD_LOGIC;                              -- Señal de reloj de la FPGA
           S : out  STD_LOGIC_VECTOR (1 downto 0);           -- Señal de control del MUX4x4
           AN : out  STD_LOGIC_VECTOR (3 downto 0));         -- Señal de control de displays
end refresco;

architecture Behavioral of refresco is

       signal SS : STD_LOGIC_VECTOR (1 downto 0):= (others=>'0');        -- señal auxiliar de S
       signal div : STD_LOGIC_VECTOR (31 downto 0):= (others=>'0');      -- contador auxiliar
       signal CLK_v : STD_LOGIC :='0';                                   -- señal reloj auxiliar


begin

process (CLK)                                -- este proceso genera una señal de reloj a 500 Hz
begin
       if(CLK'event and CLK='1') then
              div<=div+1;
              if div = 50000 then            -- A 500 Hz
                     CLK_v <= not CLK_v;
                     div<=(others=>'0');
              end if;
       end if;

end process;

process(CLK_v)                         -- este proceso aumenta la señal SS cada ciclo auxiliar
begin
       if(CLK_v'event and CLK_v='1') then
              SS<=SS+1;
       end if;
end process;

S<=SS;
AN<=   "0111" when SS="00" else              -- se generan las salidas AN en función de SS
       "1011" when SS="01" else
       "1101" when SS="10" else
       "1110" when SS="11";

end Behavioral;
```

4.10 Módulo 10: Decodificador

```
----------------------------------------------------------------------------------
-- DECODIFICADOR
--
-- Se encarga de convertir la señal numérica binaria
-- en una representación numérica decimal a través
-- de los displays de 7 segmentos.
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity decod7s is
```

```
    Port ( D : in  STD_LOGIC_VECTOR (3 downto 0);          -- Señal de entrada
           S : out  STD_LOGIC_VECTOR (0 to 6));            -- Señal de salida
end decod7s;

architecture Behavioral of decod7s is

begin
       with D select S<=                                  -- en este caso cubrimos todo el espectro de
              "0000001" when "0000",                      -- posibilidades para detectar errores
              "1001111" when "0001",
              "0010010" when "0010",
              "0000110" when "0011",
              "1001100" when "0100",
              "0100100" when "0101",
              "0100000" when "0110",
              "0001111" when "0111",
              "0000000" when "1000",
              "0001100" when "1001",
              "0001000" when "1010",
              "1100000" when "1011",
              "0110001" when "1100",
              "1000010" when "1101",
              "0110000" when "1110",
              "0111000" when "1111",
              "1111111" when others;

end Behavioral;
```

4.11 Módulo 11: Visualización

```
----------------------------------------------------------------------------------
-- MÓDULO TOP DE VISUALIZACIÓN
--
-- Conecta las entradas y salidas de MUX4x4, refresco y decod7s
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity visualizacion is
    Port ( E0 : in  STD_LOGIC_VECTOR (3 downto 0);
           E1 : in  STD_LOGIC_VECTOR (3 downto 0);
           E2 : in  STD_LOGIC_VECTOR (3 downto 0);
           E3 : in  STD_LOGIC_VECTOR (3 downto 0);
           CLK : in  STD_LOGIC;
           SEG7 : out  STD_LOGIC_VECTOR (0 to 6);
           AN : out  STD_LOGIC_VECTOR (3 downto 0));
end visualizacion;

architecture Behavioral of visualizacion is
       signal n_s: STD_LOGIC_VECTOR (1 downto 0);
       signal n_y: STD_LOGIC_VECTOR (3 downto 0);

component MUX4x4
       Port ( E0 : in STD_LOGIC_VECTOR (3 downto 0);       -- Entrada 0
              E1 : in STD_LOGIC_VECTOR (3 downto 0);       -- Entrada 1
              E2 : in STD_LOGIC_VECTOR (3 downto 0);       -- Entrada 2
              E3 : in STD_LOGIC_VECTOR (3 downto 0);       -- Entrada 3
              S : in STD_LOGIC_VECTOR (1 downto 0);        -- Señal de control
              Y : out STD_LOGIC_VECTOR (3 downto 0));      -- Salida
end component;

component decod7s
       Port ( D : in STD_LOGIC_VECTOR (3 downto 0);        -- Entrada BCD
              S: out STD_LOGIC_VECTOR (0 to 6));           -- Salida para excitar los displays
end component;

component refresco
```

```
        Port ( CLK : in STD_LOGIC;                          -- reloj
               S : out STD_LOGIC_VECTOR (1 downto 0);       -- Control para el mux
               AN : out STD_LOGIC_VECTOR (3 downto 0));     -- Control displays individuales
end component;

begin

U1: MUX4x4
        port map(E0=>E0,
                 E1=>E1,
                 E2=>E2,
                 E3=>E3,
                 S=>n_s,
                 Y=>n_y);

U2: decod7s
        port map(D=>n_y,
                 S=>SEG7);

U3: refresco
        port map(CLK=>CLK,
                 S=>n_s,
                 AN=> AN);

end Behavioral;
```

4.12 Entidad jerárquica TOP: Principal

```
----------------------------------------------------------------------------------
-- MÓDULO TOP PRINCIPAL
--
-- Conecta las entradas y salidas de todos los
-- componentes del circuito digital
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity principal is
    Port ( CLK : in  STD_LOGIC;                             -- Señal de reloj
           SIN : in  STD_LOGIC;                             -- Señal de entrada
           AN : out  STD_LOGIC_VECTOR (3 downto 0);         -- Control de displays
           SEG7 : out  STD_LOGIC_VECTOR (6 downto 0));      -- Salida de display
end principal;

architecture Behavioral of principal is

        constant UMBRAL1 : STD_LOGIC_VECTOR (5 downto 0) := "100010"; -- 34
        constant UMBRAL2 : STD_LOGIC_VECTOR (5 downto 0) := "100110"; -- 38

        signal CLK_M: STD_LOGIC;
        signal REG40: STD_LOGIC_VECTOR (39 downto 0);
        signal SUM: STD_LOGIC_VECTOR (5 downto 0);
        signal COMP1_MOORE: STD_LOGIC;
        signal COMP1_AND: STD_LOGIC;
        signal COMP2_AND: STD_LOGIC;
        signal AND_AUX: STD_LOGIC;
        signal SIN_REG: STD_LOGIC;
        signal EN_REG: STD_LOGIC;
        signal EN_VALID: STD_LOGIC;
        signal REG_DESPL: STD_LOGIC_VECTOR (13 downto 0);
        signal REG_VALID: STD_LOGIC_VECTOR (13 downto 0);
        signal Seg7_aux: STD_LOGIC_VECTOR (0 to 6);


component gen_reloj
```

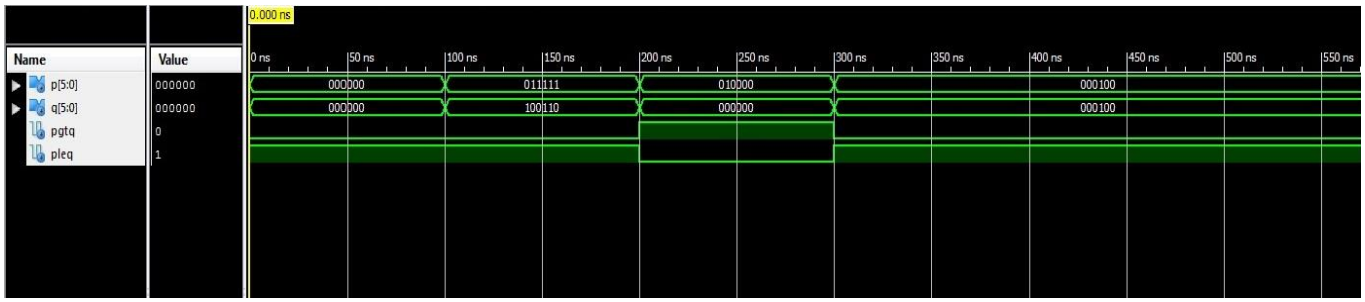
```
        Port ( CLK : in STD_LOGIC;           -- Reloj de la FPGA
               CLK_OUT : out STD_LOGIC);     -- Reloj de frecuencia dividida
end component;

component reg_desp40
        Port ( SIN : in STD_LOGIC;           -- Datos de entrada serie
               CLK : in STD_LOGIC;           -- Reloj de muestreo
               Q : out STD_LOGIC_VECTOR (39 downto 0)); -- Salida paralelo
end component;

component sumador40
        Port ( ENT : in STD_LOGIC_VECTOR (39 downto 0);
               SAL : out STD_LOGIC_VECTOR (5 downto 0));
end component;

component comparador
        Port ( P : in STD_LOGIC_VECTOR (5 downto 0);
               Q : in STD_LOGIC_VECTOR (5 downto 0);
               PGTQ : out STD_LOGIC;
               PLEQ : out STD_LOGIC);
end component;

component AND_2
        Port ( A : in STD_LOGIC;
               B : in STD_LOGIC;
               S : out STD_LOGIC);
end component;

component reg_desp
        Port ( SIN : in STD_LOGIC;           -- Datos de entrada serie
               CLK : in STD_LOGIC;           -- Reloj
               EN : in STD_LOGIC;            -- Enable
               Q : out STD_LOGIC_VECTOR (13 downto 0)); -- Salida paralelo
end component;

component registro
        Port ( ENTRADA : in STD_LOGIC_VECTOR (13 downto 0);
               SALIDA : out STD_LOGIC_VECTOR (13 downto 0);
               EN : in STD_LOGIC;            -- Enable
               CLK : in STD_LOGIC);
end component;

component automata
        Port ( CLK : in STD_LOGIC;           -- Reloj del autómata
               C0 : in STD_LOGIC;            -- Condición de decision para "0"
               C1 : in STD_LOGIC;            -- Condición de decisión para "1"
               DATO : out STD_LOGIC;         -- Datos a cargar
               CAPTUR : out STD_LOGIC;       -- Enable del reg. de desplaz.
               VALID : out STD_LOGIC);       -- Activación registro
end component;

component visualizacion
        Port ( E0 : in STD_LOGIC_VECTOR (3 downto 0);       -- Entrada MUX 0
               E1 : in STD_LOGIC_VECTOR (3 downto 0);       -- Entrada MUX 1
               E2 : in STD_LOGIC_VECTOR (3 downto 0);       -- Entrada MUX 2
               E3 : in STD_LOGIC_VECTOR (3 downto 0);       -- Entrada MUX 3
               CLK : in STD_LOGIC;                          -- Entrada de reloj FPGA
               SEG7 : out STD_LOGIC_VECTOR (0 to 6);        -- Salida para los displays
               AN : out STD_LOGIC_VECTOR (3 downto 0));     -- Activación individual
end component;

begin
        U1: gen_reloj port map(
        CLK=>CLK,
        CLK_OUT=>CLK_M
        );

        U2: reg_desp40 port map(
```

```
CLK=>CLK_M,
SIN=>SIN,
Q=>REG40
);

U3: sumador40 port map(
ENT=>REG40,
SAL=>SUM
);

U4: comparador port map(
P=>SUM,
Q=>UMBRAL1,
PGTQ=>COMP1_AND,
PLEQ=>COMP1_MOORE
);

U5: comparador port map(
P=>SUM,
Q=>UMBRAL2,
PLEQ=>COMP2_AND
);

U6: AND_2 port map(
A=>COMP1_AND,
B=>COMP2_AND,
S=>AND_AUX
);

U7: automata port map(
CLK=>CLK_M,
C0=>AND_AUX,
C1=>COMP1_MOORE,
DATO=>SIN_REG,
CAPTUR=>EN_REG,
VALID=>EN_VALID
);

U8: reg_desp port map(
SIN=>SIN_REG,
CLK=>CLK_M,
EN=>EN_REG,
Q=>REG_DESPL
);

U9: registro port map(
ENTRADA=>REG_DESPL,
SALIDA=>REG_VALID,
EN=>EN_VALID,
CLK=>CLK_M
);

U10: visualizacion port map(
E3(0)=>REG_VALID(0),
E3(1)=>REG_VALID(1),
E3(2)=>REG_VALID(2),
E3(3)=>REG_VALID(3),
E2(0)=>REG_VALID(4),
E2(1)=>REG_VALID(5),
E2(2)=>REG_VALID(6),
E2(3)=>'0',
E1(0)=>REG_VALID(7),
E1(1)=>REG_VALID(8),
E1(2)=>REG_VALID(9),
E1(3)=>REG_VALID(10),
E0(0)=>REG_VALID(11),
```

```
        E0(1)=>REG_VALID(12),
        E0(2)=>REG_VALID(12),
        E0(3)=>'0',
        CLK=>CLK,
        SEG7=>Seg7_aux,
        AN=>AN
        );

        SEG7<=Seg7_aux;


end Behavioral;
```

4.13 Testbench comparador

```
--------------------------------------------------------------------------------
-- TESTBENCH COMPARADOR
--
-- Se simulan valores de entrada y se comprueba que
-- la salida es la esperada.
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY tb_comparador IS
END tb_comparador;

ARCHITECTURE behavior OF tb_comparador IS

    COMPONENT comparador
    PORT(
         P : IN  std_logic_vector(5 downto 0);
         Q : IN  std_logic_vector(5 downto 0);
         PGTQ : OUT  std_logic;
         PLEQ : OUT  std_logic
        );
    END COMPONENT;

   signal P : std_logic_vector(5 downto 0) := (others => '0');
   signal Q : std_logic_vector(5 downto 0) := (others => '0');
   signal PGTQ : std_logic;
   signal PLEQ : std_logic;

BEGIN
   uut: comparador PORT MAP (
          P => P,
          Q => Q,
          PGTQ => PGTQ,
          PLEQ => PLEQ
        );

   stim_proc: process
   begin
      wait for 100 ns;              -- probamos entradas 0/0 (P=Q)
               P<="011111";
               Q<="100110";
      wait for 100 ns;              -- probamos entradas 31/38 (P<Q)
               P<="010000";
               Q<="000000";
      wait for 100 ns;              -- probamos entradas 16/0 (P>Q)
               P<="000100";
               Q<="000100";
      wait;                         -- probamos entradas 4/4 (P=Q)
   end process;

END;
```

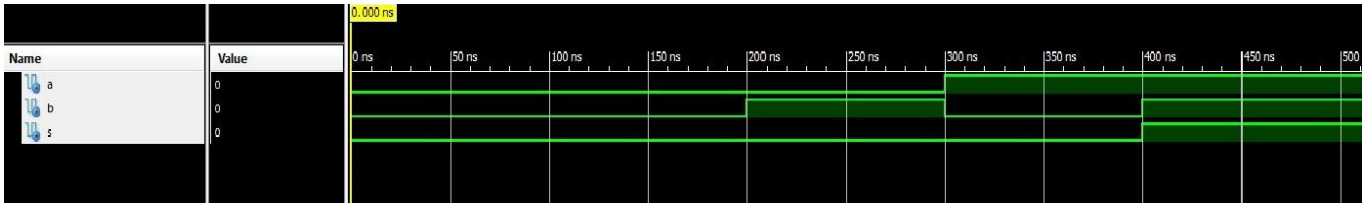


Comentario: Vemos que cuando la entrada “p” es menor o igual que “q” la salida “pleq” (p lower equal q) es 1 y “pgtq” (p greater than q) es 0. En el caso contrario (p>q) “pleq” es 0 y “pgtq” es 1, como cabía esperar.

4.14 Testbench AND

```
--------------------------------------------------------------------------------
-- TESTBENCH AND
--
-- Se simulan valores de entrada y se comprueba que
-- la salida es la esperada.
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY tb_AND2 IS
END tb_AND2;

ARCHITECTURE behavior OF tb_AND2 IS

    COMPONENT AND_2
    PORT(
         A : IN  std_logic;
         B : IN  std_logic;
         S : OUT  std_logic
        );
    END COMPONENT;

   signal A : std_logic := '0';
   signal B : std_logic := '0';
   signal S : std_logic;

BEGIN
   uut: AND_2 PORT MAP (
          A => A,
          B => B,
          S => S
        );
   stim_proc: process
   begin
      wait for 100 ns;
               A<='0';
               B<='0';
      wait for 100 ns;        -- probamos entradas 0/0
               A<='0';
               B<='1';
      wait for 100 ns;        -- probamos entradas 0/1
               A<='1';
               B<='0';
      wait for 100 ns;        -- probamos entradas 1/0
               A<='1';
               B<='1';
      wait;                   -- probamos entradas 1/1
   end process;
```

```
END;
```



Comentario: Aquí comprobamos la puerta AND con su propia tabla de verdad siendo las entradas “a” y ”b” y la salida “s”

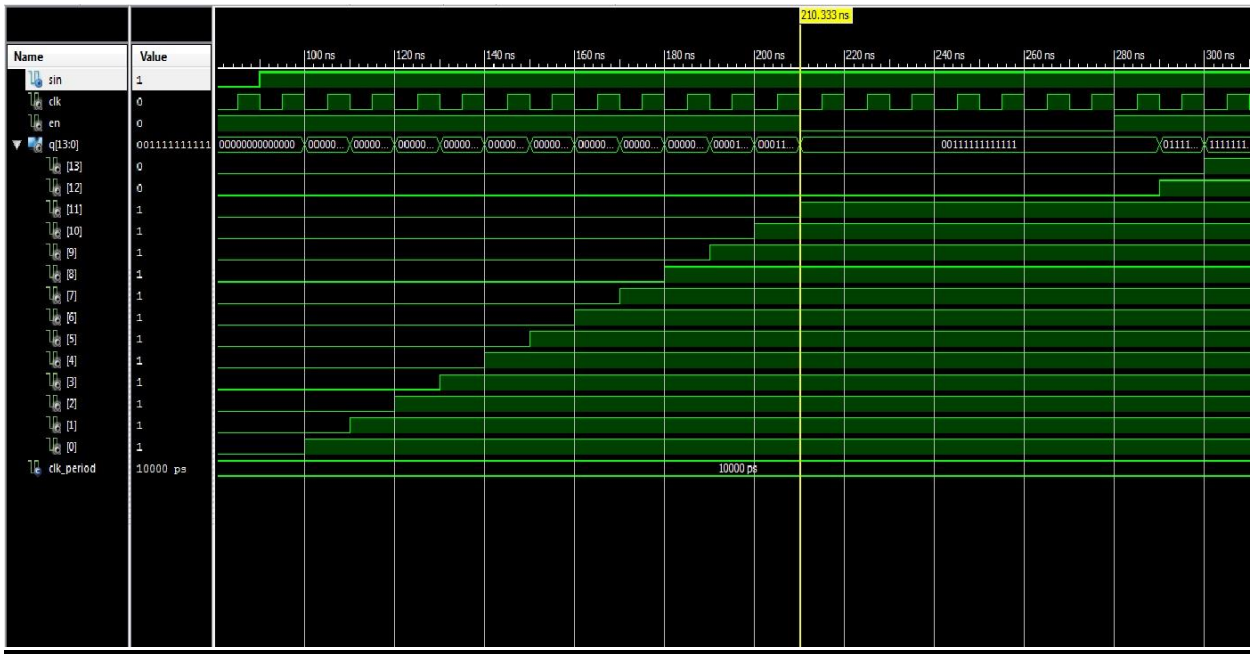
A	B	S
0	0	0
0	1	0
1	0	0
1	1	1

4.15 Testbench registro de desplazamiento de 14 bits

```
--------------------------------------------------------------------------------
-- TESTBENCH REGISTRO DE DESPLAZAMIENTO DE 14 BITS
--
-- Se simulan valores de entrada y se comprueba que
-- la salida es la esperada.
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY test_reg_desp IS
END test_reg_desp;

ARCHITECTURE behavior OF test_reg_desp IS

    COMPONENT reg_desp
    PORT(
         SIN : IN  std_logic;
         CLK : IN  std_logic;
         EN : IN std_logic;
         Q : OUT  std_logic_vector(13 downto 0)
        );
    END COMPONENT;

   signal SIN : std_logic := '0';
   signal CLK : std_logic := '0';
   signal EN : std_logic := '0';
   signal Q : std_logic_vector(13 downto 0);
   constant CLK_period : time := 10 ns;

BEGIN
   uut: reg_desp PORT MAP (
          SIN => SIN,
          CLK => CLK,
          EN => EN,
          Q => Q
        );

   CLK_process :process
   begin
               CLK <= '0';
               wait for CLK_period/2;
```

```
            CLK <= '1';
            wait for CLK_period/2;
end process;

stim_proc: process
begin
   wait for 10 ns;

            -- Vamos introduciendo a través de la línea de entrada serie
            -- 8 '0' y 32 '1' de forma síncrona con el reloj
            EN<='1';
   SIN<='0';
   wait for CLK_period;
            SIN<='0';
   wait for CLK_period;
            SIN<='0';
   wait for CLK_period;
            SIN<='0';
   wait for CLK_period;
            SIN<='0';
   wait for CLK_period;
            SIN<='0';
   wait for CLK_period;
            SIN<='0';
   wait for CLK_period;
            SIN<='0';
   wait for CLK_period;
            SIN<='1';
   wait for CLK_period;
            SIN<='1';
   wait for CLK_period;
            SIN<='1';
   wait for CLK_period;
            SIN<='1';
   wait for CLK_period;
            SIN<='1';
   wait for CLK_period;
            SIN<='1';
   wait for CLK_period;
            SIN<='1';
   wait for CLK_period;
            SIN<='1';
   wait for CLK_period;
            SIN<='1';
   wait for CLK_period;
            SIN<='1';
   wait for CLK_period;
            SIN<='1';
   wait for CLK_period;
            SIN<='1';
   wait for CLK_period;
            SIN<='1';
            EN<='0';
   wait for CLK_period;
            SIN<='1';
   wait for CLK_period;
            SIN<='1';
   wait for CLK_period;
            SIN<='1';
   wait for CLK_period;
            SIN<='1';
   wait for CLK_period;
            SIN<='1';
   wait for CLK_period;
            SIN<='1';
   wait for CLK_period;
            SIN<='1';
```

```
            EN<='1';
      wait for CLK_period;
            SIN<='1';
      wait for CLK_period;
            SIN<='1';
      wait for CLK_period;
            SIN<='1';
      wait for CLK_period;
            SIN<='1';
      wait for CLK_period;
            SIN<='1';
      wait for CLK_period;
            SIN<='1';
      wait for CLK_period;
            SIN<='1';
      wait for CLK_period;
            SIN<='1';
      wait for CLK_period;
            SIN<='1';
      wait for CLK_period;
            SIN<='1';
      wait for CLK_period;
            SIN<='1';
      wait for CLK_period;
            SIN<='1';
      wait for CLK_period;
            SIN<='1';
      wait for CLK_period;
      wait;

   end process;
END;
```



Comentario: El testbench del registro nos ofrece la posibilidad de ver cómo se van cargando los datos de la entrada “sin”, desde q(0) hasta q(13), en cada ciclo de reloj siempre y cuando esté activa a nivel alto la señal “EN” correspondiente al enable.

4.16 Testbench entidad TOP (principal)

```
--------------------------------------------------------------------------------
-- TESTBENCH PRINCIPAL
--
-- Se simulan valores de entrada y se comprueba que
```

```
-- la salida es la esperada en cada etapa del circuito
-- completo.
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY TB_Principal IS
END TB_Principal;

ARCHITECTURE behavior OF TB_Principal IS

    COMPONENT principal
    PORT(
         CLK : IN  std_logic;
         SIN : IN  std_logic;
         AN : OUT  std_logic_vector(3 downto 0);
         SEG7 : OUT  std_logic_vector(6 downto 0)
        );
    END COMPONENT;

   signal CLK : std_logic := '0';
   signal SIN : std_logic := '1';
   signal AN : std_logic_vector(3 downto 0);
   signal SEG7 : std_logic_vector(6 downto 0);
   constant CLK_period : time := 1 ns;

BEGIN
   uut: principal PORT MAP (
          CLK => CLK,
          SIN => SIN,
          AN => AN,
          SEG7 => SEG7
        );

   CLK_process :process
   begin
               CLK <= '0';
               wait for CLK_period/2;
               CLK <= '1';
               wait for CLK_period/2;
   end process;

   stim_proc: process
   begin
        wait for 50 ms;              --SYNC
        SIN<='0';
        wait for 5 ms;               --0
        SIN<='1';
        wait for 45 ms;
        SIN<='0';
        wait for 5 ms;               --0
        SIN<='1';
        wait for 45 ms;
        SIN<='0';
        wait for 10 ms;              --1
        SIN<='1';
        wait for 40 ms;
        SIN<='0';
        wait for 10 ms;              --1
        SIN<='1';
        wait for 40 ms;
        SIN<='0';
        wait for 5 ms;               --0
        SIN<='1';
        wait for 45 ms;
        SIN<='0';
        wait for 5 ms;               --0
```

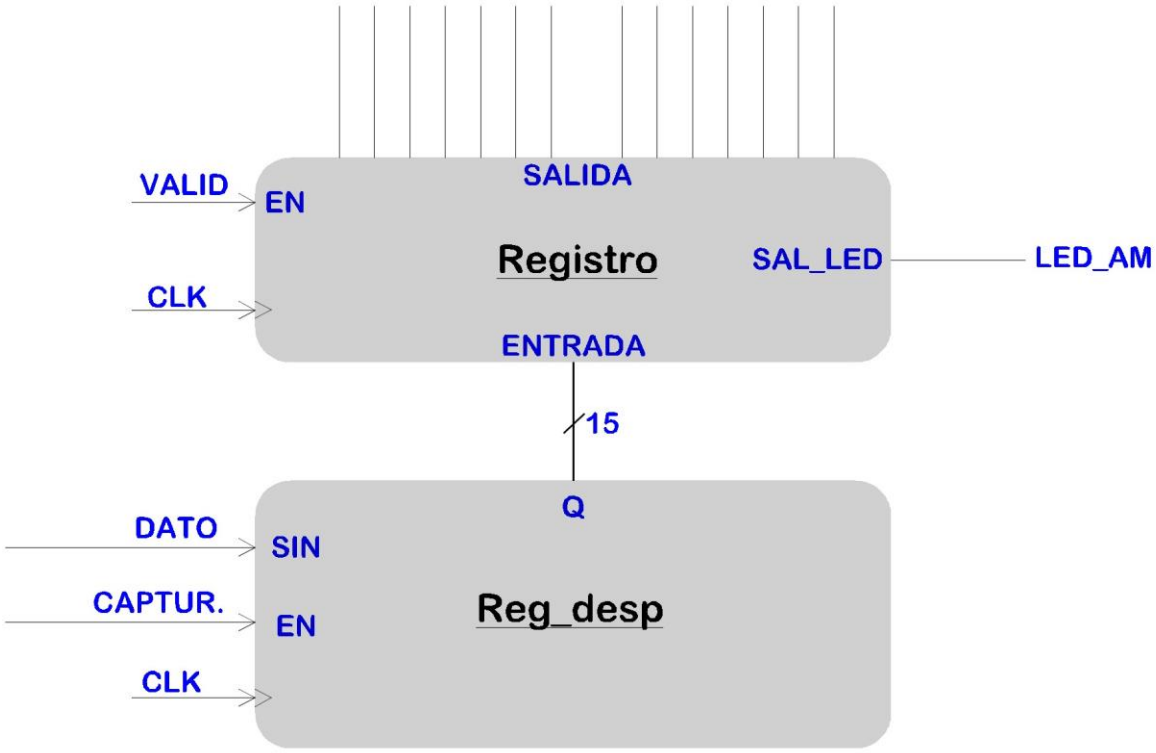
```
        SIN<='1';
        wait for 45 ms;
        SIN<='0';
        wait for 10 ms;             --1
        SIN<='1';
        wait for 40 ms;
        SIN<='0';
        wait for 5 ms;              --0
        SIN<='1';
        wait for 45 ms;
        SIN<='0'
        wait for 10 ms;             --1
        SIN<='1';
        wait for 40 ms;
        SIN<='0';
        wait for 10 ms;             --1
        SIN<='1';
        wait for 40 ms;
        SIN<='0';
        wait for 5 ms;              --0
        SIN<='1';
        wait for 45 ms;
        SIN<='0';
        wait for 5 ms;              --0
        SIN<='1';
        wait for 45 ms;
        SIN<='0';
        wait for 5 ms;              --0
        SIN<='1';
        wait for 45 ms;
        SIN<='0';
        wait for 5 ms;              --0
        SIN<='1';
        wait for 45 ms;             --001 1001 011 0000  19:30

        wait for 50 ms;             --SYNC
        SIN<='0';
      wait;
   end process;

END;
```

Comentario: Para este fichero testbench no tenemos captura de la simulación puesto que tarda mucho en hacerla y tiene tantos datos que no se ven bien en una captura.

5. MEJORAS

Se incluyen los códigos que han sido modificados. El resto de códigos se mantienen igual respecto al proyecto básico.

5.1 Mejora AM/PM



5.1.1 Módulo 1: Registro de desplazamiento de 15 bits

```
----------------------------------------------------------------------------------
-- REGISTRO DE DESPLAZAMIENTO DE 15 BITS
--
-- Este registro se encarga de recoger y
-- almacenar los últimos 15 bits que le
-- han llegado por su entrada, y los
-- devuelve por otra salida de 15 bits.
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity reg_desp is
    Port ( SIN : in  STD_LOGIC;                              -- Señal de entrada
           CLK : in  STD_LOGIC;                              -- Señal de reloj de muestreo
           EN : in  STD_LOGIC;                               -- Enable
           Q : out  STD_LOGIC_VECTOR (14 downto 0));         -- Señal de salida
end reg_desp;

architecture Behavioral of reg_desp is

       signal Qs : STD_LOGIC_VECTOR (14 downto 0):= (others=>'0');-- señal auxiliar del registro
begin
PROC_REG : process (CLK)                    -- en este proceso se carga cada bit en el espacio
                                            -- adyacente y por último la entrada en último lugar
```

```
begin
	if (CLK'event and CLK='1' and EN='1') then
		Qs(0)<=SIN;
		Qs(1)<=Qs(0);
		Qs(2)<=Qs(1);
		Qs(3)<=Qs(2);
		Qs(4)<=Qs(3);
		Qs(5)<=Qs(4);
		Qs(6)<=Qs(5);
		Qs(7)<=Qs(6);
		Qs(8)<=Qs(7);
		Qs(9)<=Qs(8);
		Qs(10)<=Qs(9);
		Qs(11)<=Qs(10);
		Qs(12)<=Qs(11);
		Qs(13)<=Qs(12);
		Qs(14)<=Qs(13);
	end if;

Q<=Qs;                          -- se carga la señal auxiliar en la salida
end process;

end Behavioral;
```

5.1.2 Módulo 2: Registro de validación

```
----------------------------------------------------------------------------------
-- REGISTRO DE VALIDACIÓN
--
-- Este registro carga su entrada a la salida
-- cada vez que se activa su señal de enable.
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity registro is
    Port ( ENTRADA : in  STD_LOGIC_VECTOR (14 downto 0):= (others=>'0');  -- Señal de entrada
           SALIDA : out  STD_LOGIC_VECTOR (14 downto 0);                  -- Señal de salida
           EN : in  STD_LOGIC;                                            -- Enable
           CLK : in  STD_LOGIC;                                           -- Señal de reloj
           SAL_LED : out  STD_LOGIC);                                     -- Salida del LED PM
end registro;

architecture Behavioral of registro is

begin

PROC_REG : process (CLK)
begin
	if (CLK'event and CLK='1' and EN='1') then   -- si el enable está activo y hay un flanco
		SALIDA<=ENTRADA;                     -- de reloj se carga la entrada en la salida
		SAL_LED<=ENTRADA(0);                 -- y el bit del LED a la salida LED
	end if;
end process;
end Behavioral;
```

5.1.3 Entidad jerárquica TOP: Principal

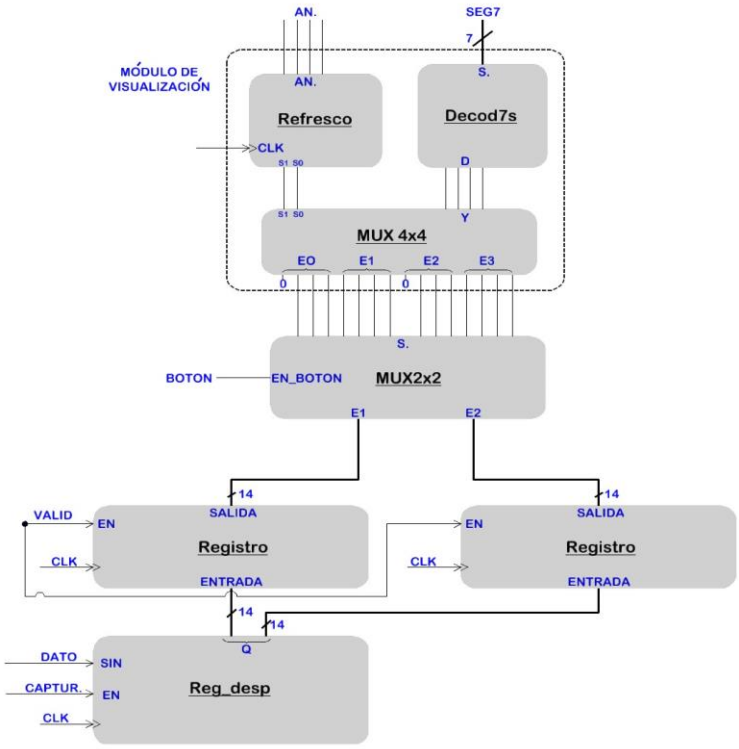
```
----------------------------------------------------------------------------------
-- MÓDULO TOP PRINCIPAL
--
-- Conecta las entradas y salidas de todos los
-- componentes del circuito digital
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
```

```
entity principal is
    Port ( CLK : in  STD_LOGIC;                              -- Señal de reloj
           SIN : in  STD_LOGIC;                              -- Señal de entrada
           AN : out  STD_LOGIC_VECTOR (3 downto 0);          -- Control de displays
           SEG7 : out  STD_LOGIC_VECTOR (6 downto 0);        -- Salida de display
           LED_AM : out  STD_LOGIC);                         -- Enable del LED PM
end principal;

architecture Behavioral of principal is

       constant UMBRAL1 : STD_LOGIC_VECTOR (5 downto 0) := "100010"; -- 34
       constant UMBRAL2 : STD_LOGIC_VECTOR (5 downto 0) := "100110"; -- 38

       signal CLK_M: STD_LOGIC;
       signal REG40: STD_LOGIC_VECTOR (39 downto 0);
       signal SUM: STD_LOGIC_VECTOR (5 downto 0);
       signal COMP1_MOORE: STD_LOGIC;
       signal COMP1_AND: STD_LOGIC;
       signal COMP2_AND: STD_LOGIC;
       signal AND_AUX: STD_LOGIC;
       signal SIN_REG: STD_LOGIC;
       signal EN_REG: STD_LOGIC;
       signal EN_VALID: STD_LOGIC;
       signal REG_DESPL: STD_LOGIC_VECTOR (14 downto 0);
       signal REG_VALID: STD_LOGIC_VECTOR (14 downto 0);
       signal Seg7_aux: STD_LOGIC_VECTOR (0 to 6)

component gen_reloj
       Port ( CLK : in STD_LOGIC;           -- Reloj de la FPGA
              CLK_OUT : out STD_LOGIC);     -- Reloj de frecuencia dividida
end component;

component reg_desp40
       Port ( SIN : in STD_LOGIC;           -- Datos de entrada serie
              CLK : in STD_LOGIC;           -- Reloj de muestreo
              Q : out STD_LOGIC_VECTOR (39 downto 0)); -- Salida paralelo
end component;

component sumador40
       Port ( ENT : in STD_LOGIC_VECTOR (39 downto 0);
              SAL : out STD_LOGIC_VECTOR (5 downto 0));
end component;

component comparador
       Port ( P : in STD_LOGIC_VECTOR (5 downto 0);
              Q : in STD_LOGIC_VECTOR (5 downto 0);
              PGTQ : out STD_LOGIC;
              PLEQ : out STD_LOGIC);
end component;

component AND_2
       Port ( A : in STD_LOGIC;
              B : in STD_LOGIC;
              S : out STD_LOGIC);
end component;

component reg_desp
       Port ( SIN : in STD_LOGIC;           -- Datos de entrada serie
              CLK : in STD_LOGIC;           -- Reloj
              EN : in STD_LOGIC;            -- Enable
              Q : out STD_LOGIC_VECTOR (14 downto 0)); -- Salida paralelo
end component;

component registro
       Port ( ENTRADA : in STD_LOGIC_VECTOR (14 downto 0);
              SALIDA : out STD_LOGIC_VECTOR (14 downto 0);
              EN : in STD_LOGIC;            -- Enable
```

```
		CLK : in STD_LOGIC;
		SAL_LED : out  STD_LOGIC);
end component;

component automata
	Port ( CLK : in STD_LOGIC;		-- Reloj del autómata
		C0 : in STD_LOGIC;		-- Condición de decision para "0"
		C1 : in STD_LOGIC;		-- Condición de decisión para "1"
		DATO : out STD_LOGIC;		-- Datos a cargar
		CAPTUR : out STD_LOGIC;		-- Enable del reg. de desplaz.
		VALID : out STD_LOGIC);		-- Activación registro
end component;

component visualizacion
	Port ( E0 : in STD_LOGIC_VECTOR (3 downto 0);	-- Entrada MUX 0
		E1 : in STD_LOGIC_VECTOR (3 downto 0);	-- Entrada MUX 1
		E2 : in STD_LOGIC_VECTOR (3 downto 0);	-- Entrada MUX 2
		E3 : in STD_LOGIC_VECTOR (3 downto 0);	-- Entrada MUX 3
		CLK : in STD_LOGIC;			-- Entrada de reloj FPGA
		SEG7 : out STD_LOGIC_VECTOR (0 to 6);	-- Salida para los displays
		AN : out STD_LOGIC_VECTOR (3 downto 0));	-- Activación individual
end component;

begin
	U1: gen_reloj port map(
	CLK=>CLK,
	CLK_OUT=>CLK_M
	);

	U2: reg_desp40 port map(
	CLK=>CLK_M,
	SIN=>SIN,
	Q=>REG40
	);

	U3: sumador40 port map(
	ENT=>REG40,
	SAL=>SUM
	);

	U4: comparador port map(
	P=>SUM,
	Q=>UMBRAL1,
	PGTQ=>COMP1_AND,
	PLEQ=>COMP1_MOORE
	);

	U5: comparador port map(
	P=>SUM,
	Q=>UMBRAL2,
	PLEQ=>COMP2_AND
	);

	U6: AND_2 port map(
	A=>COMP1_AND,
	B=>COMP2_AND,
	S=>AND_AUX
	);

	U7: automata port map(
	CLK=>CLK_M,
	C0=>AND_AUX,
	C1=>COMP1_MOORE,
	DATO=>SIN_REG,
	CAPTUR=>EN_REG,
	VALID=>EN_VALID
```

```
        );

        U8: reg_desp port map(
        SIN=>SIN_REG,
        CLK=>CLK_M,
        EN=>EN_REG,
        Q=>REG_DESPL
        );

        U9: registro port map(
        ENTRADA=>REG_DESPL,
        SALIDA=>REG_VALID,
        EN=>EN_VALID,
        CLK=>CLK_M,
        SAL_LED=>LED_AM
        );

        U10: visualizacion port map(
        E3(0)=>REG_VALID(0),
        E3(1)=>REG_VALID(1),
        E3(2)=>REG_VALID(2),
        E3(3)=>REG_VALID(3),
        E2(0)=>REG_VALID(4),
        E2(1)=>REG_VALID(5),
        E2(2)=>REG_VALID(6),
        E2(3)=>'0',
        E1(0)=>REG_VALID(7),
        E1(1)=>REG_VALID(8),
        E1(2)=>REG_VALID(9),
        E1(3)=>REG_VALID(10),
        E0(0)=>REG_VALID(11),
        E0(1)=>REG_VALID(12),
        E0(2)=>REG_VALID(13),
        E0(3)=>'0',
        CLK=>CLK,
        SEG7=>Seg7_aux,
        AN=>AN
        );
        SEG7<=Seg7_aux;
end Behavioral;
```

5.2 Mejora Fecha



5.2.1 Módulo 1: Registro de desplazamiento de 28 bits

```
----------------------------------------------------------------------------------
-- REGISTRO DE DESPLAZAMIENTO DE 28 BITS
--
-- Este registro se encarga de recoger y
-- almacenar los últimos 28 bits que le
-- han llegado por su entrada, y los
-- devuelve por otra salida de 28 bits.
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity reg_desp is
    Port ( SIN : in  STD_LOGIC;                              -- Señal de entrada
           CLK : in  STD_LOGIC;                              -- Señal de reloj de muestreo
           EN : in  STD_LOGIC;                               -- Enable
           Q : out  STD_LOGIC_VECTOR (27 downto 0));         -- Señal de salida
end reg_desp;

architecture Behavioral of reg_desp is

       signal Qs : STD_LOGIC_VECTOR (27 downto 0):= (others=>'0');-- señal auxiliar del registro
begin
PROC_REG : process (CLK)                        -- en este proceso se carga cada bit en el espacio
                                                -- adyacente y por último la entrada en último lugar
begin
       if (CLK'event and CLK='1' and EN='1') then
              Qs(0)<=SIN;
              Qs(1)<=Qs(0);
              Qs(2)<=Qs(1);
              Qs(3)<=Qs(2);
              Qs(4)<=Qs(3);
              Qs(5)<=Qs(4);
              Qs(6)<=Qs(5);
              Qs(7)<=Qs(6);
```

```
                Qs(8)<=Qs(7);
                Qs(9)<=Qs(8);
                Qs(10)<=Qs(9);
                Qs(11)<=Qs(10);
                Qs(12)<=Qs(11);
                Qs(13)<=Qs(12);
                Qs(14)<=Qs(13);
                Qs(15)<=Qs(14);
                Qs(16)<=Qs(15);
                Qs(17)<=Qs(16);
                Qs(18)<=Qs(17);
                Qs(19)<=Qs(18);
                Qs(20)<=Qs(19);
                Qs(21)<=Qs(20);
                Qs(22)<=Qs(21);
                Qs(23)<=Qs(22);
                Qs(24)<=Qs(23);
                Qs(25)<=Qs(24);
                Qs(26)<=Qs(25);
                Qs(27)<=Qs(26);
        end if;
Q<=Qs;                          -- se carga la señal auxiliar en la salida
end process;

end Behavioral;
```

5.2.2 Módulo 2: Registro de validación

El registro de validación vuelve a coincidir con el del proyecto original.

5.2.3 Módulo 3: Multiplexor 2 a 1

```
----------------------------------------------------------------------------------
-- MULTIPLEXOR 2 A 1
--
-- El multiplexor envía una de las 2 entradas a la
-- salida en función de la entrada de control EN_BOTON.
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity MUX2x2 is
    Port ( E1 : in  STD_LOGIC_VECTOR (13 downto 0);         -- Entrada 1
           E2 : in  STD_LOGIC_VECTOR (13 downto 0);         -- Entrada 2
           EN_BOTON: in STD_LOGIC;                          -- Señal de control
           S : out  STD_LOGIC_VECTOR (13 downto 0));        -- Salida
end MUX2x2;

architecture Behavioral of MUX2x2 is

begin
process (EN_BOTON)
begin
        if (EN_BOTON='0') then
                S<= E1 ;
        else
                S<= E2;
        end if;
end process;

end Behavioral;
```

5.2.4 Entidad jerárquica TOP: Principal

```
----------------------------------------------------------------------------------
-- MÓDULO TOP PRINCIPAL
--
-- Conecta las entradas y salidas de todos los
```

```
-- componentes del circuito digital
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity principal is
    Port ( CLK : in  STD_LOGIC;                              -- Señal de reloj
           SIN : in  STD_LOGIC;                              -- Señal de entrada
           AN : out  STD_LOGIC_VECTOR (3 downto 0);          -- Control de displays
           SEG7 : out  STD_LOGIC_VECTOR (6 downto 0);        -- Salida de display
           BOTON : in  STD_LOGIC);                           -- Botón de cambio a fecha
end principal;

architecture Behavioral of principal is

        constant UMBRAL1 : STD_LOGIC_VECTOR (5 downto 0) := "100010"; -- 34
        constant UMBRAL2 : STD_LOGIC_VECTOR (5 downto 0) := "100110"; -- 38

        signal CLK_M: STD_LOGIC;
        signal REG40: STD_LOGIC_VECTOR (39 downto 0);
        signal SUM: STD_LOGIC_VECTOR (5 downto 0);
        signal COMP1_MOORE: STD_LOGIC;
        signal COMP1_AND: STD_LOGIC;
        signal COMP2_AND: STD_LOGIC;
        signal AND_AUX: STD_LOGIC;
        signal SIN_REG: STD_LOGIC;
        signal EN_REG: STD_LOGIC;
        signal EN_VALID: STD_LOGIC;
        signal REG_DESPL1: STD_LOGIC_VECTOR (13 downto 0);
        signal REG_DESPL2: STD_LOGIC_VECTOR (13 downto 0);
        signal REG_VALID1: STD_LOGIC_VECTOR (13 downto 0);
        signal REG_VALID2: STD_LOGIC_VECTOR (13 downto 0);
        signal REG_VALID3: STD_LOGIC_VECTOR (13 downto 0);
        signal Seg7_aux: STD_LOGIC_VECTOR (0 to 6)

component gen_reloj
        Port ( CLK : in STD_LOGIC;           -- Reloj de la FPGA
               CLK_OUT : out STD_LOGIC);     -- Reloj de frecuencia dividida
end component;

component reg_desp40
        Port ( SIN : in STD_LOGIC;           -- Datos de entrada serie
               CLK : in STD_LOGIC;           -- Reloj de muestreo
               Q : out STD_LOGIC_VECTOR (39 downto 0)); -- Salida paralelo
end component;

component sumador40
        Port ( ENT : in STD_LOGIC_VECTOR (39 downto 0);
               SAL : out STD_LOGIC_VECTOR (5 downto 0));
end component;

component comparador
        Port ( P : in STD_LOGIC_VECTOR (5 downto 0);
               Q : in STD_LOGIC_VECTOR (5 downto 0);
               PGTQ : out STD_LOGIC;
               PLEQ : out STD_LOGIC);
end component;

component AND_2
        Port ( A : in STD_LOGIC;
               B : in STD_LOGIC;
               S : out STD_LOGIC);
end component;

component reg_desp
        Port ( SIN : in STD_LOGIC;           -- Datos de entrada serie
```

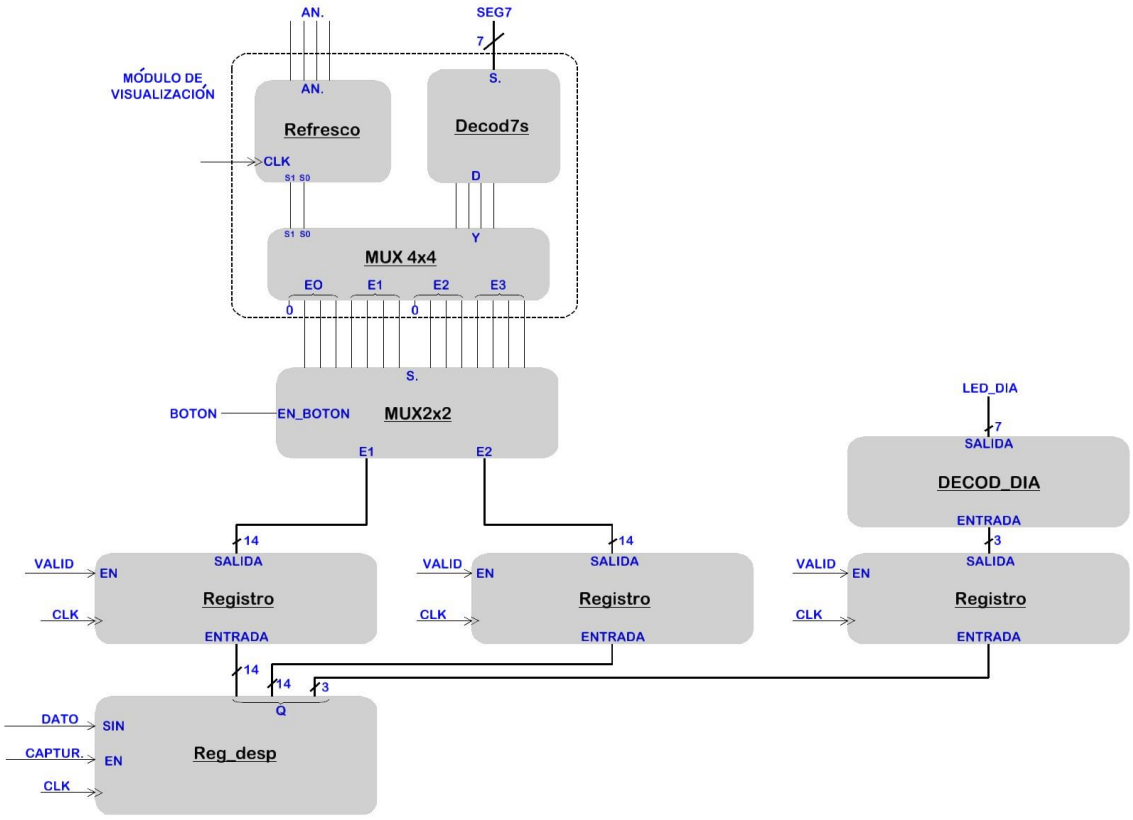
```
               CLK : in STD_LOGIC;           -- Reloj
               EN : in STD_LOGIC;            -- Enable
               Q : out STD_LOGIC_VECTOR (27 downto 0)); -- Salida paralelo
end component;

component registro
       Port ( ENTRADA : in STD_LOGIC_VECTOR (13 downto 0);
              SALIDA : out STD_LOGIC_VECTOR (13 downto 0);
              EN : in STD_LOGIC;            -- Enable
              CLK : in STD_LOGIC;
              SAL_LED : out  STD_LOGIC);
end component;

component automata
       Port ( CLK : in STD_LOGIC;           -- Reloj del autómata
              C0 : in STD_LOGIC;            -- Condición de decision para "0"
              C1 : in STD_LOGIC;            -- Condición de decisión para "1"
              DATO : out STD_LOGIC;         -- Datos a cargar
              CAPTUR : out STD_LOGIC;       -- Enable del reg. de desplaz.
              VALID : out STD_LOGIC);       -- Activación registro
end component;

component visualizacion
       Port ( E0 : in STD_LOGIC_VECTOR (3 downto 0);       -- Entrada MUX 0
              E1 : in STD_LOGIC_VECTOR (3 downto 0);       -- Entrada MUX 1
              E2 : in STD_LOGIC_VECTOR (3 downto 0);       -- Entrada MUX 2
              E3 : in STD_LOGIC_VECTOR (3 downto 0);       -- Entrada MUX 3
              CLK : in STD_LOGIC;                          -- Entrada de reloj FPGA
              SEG7 : out STD_LOGIC_VECTOR (0 to 6);        -- Salida para los displays
              AN : out STD_LOGIC_VECTOR (3 downto 0));     -- Activación individual
end component;

component MUX2x2
       Port ( E1 : in STD_LOGIC_VECTOR (13 downto 0);      -- Entrada 1
              E2 : in STD_LOGIC_VECTOR (13 downto 0);      -- Entrada 2
              S : out STD_LOGIC_VECTOR (13 downto 0);      -- Salida
              EN_BOTON : in STD_LOGIC);                    -- Enable
end component;

begin
       U1: gen_reloj port map(
       CLK=>CLK,
       CLK_OUT=>CLK_M
       );

       U2: reg_desp40 port map(
       CLK=>CLK_M,
       SIN=>SIN,
       Q=>REG40
       );

       U3: sumador40 port map(
       ENT=>REG40,
       SAL=>SUM
       );

       U4: comparador port map(
       P=>SUM,
       Q=>UMBRAL1,
       PGTQ=>COMP1_AND,
       PLEQ=>COMP1_MOORE
       );

       U5: comparador port map(
       P=>SUM,
       Q=>UMBRAL2,
       PLEQ=>COMP2_AND
```

```
);

U6: AND_2 port map(
A=>COMP1_AND,
B=>COMP2_AND,
S=>AND_AUX
);

U7: automata port map(
CLK=>CLK_M,
C0=>AND_AUX,
C1=>COMP1_MOORE,
DATO=>SIN_REG,
CAPTUR=>EN_REG,
VALID=>EN_VALID
);

U8: reg_desp port map(
SIN=>SIN_REG,
CLK=>CLK_M,
EN=>EN_REG,
Q(0)=>REG_DESPL2(0),
Q(1)=>REG_DESPL2(1),
Q(2)=>REG_DESPL2(2),
Q(3)=>REG_DESPL2(3),
Q(4)=>REG_DESPL2(4),
Q(5)=>REG_DESPL2(5),
Q(6)=>REG_DESPL2(6),
Q(7)=>REG_DESPL2(7),
Q(8)=>REG_DESPL2(8),
Q(9)=>REG_DESPL2(9),
Q(10)=>REG_DESPL2(10),
Q(11)=>REG_DESPL2(11),
Q(12)=>REG_DESPL2(12),
Q(13)=>REG_DESPL2(13),
Q(14)=>REG_DESPL1(0),
Q(15)=>REG_DESPL1(1),
Q(16)=>REG_DESPL1(2),
Q(17)=>REG_DESPL1(3),
Q(18)=>REG_DESPL1(4),
Q(19)=>REG_DESPL1(5),
Q(20)=>REG_DESPL1(6),
Q(21)=>REG_DESPL1(7),
Q(22)=>REG_DESPL1(8),
Q(23)=>REG_DESPL1(9),
Q(24)=>REG_DESPL1(10),
Q(25)=>REG_DESPL1(11),
Q(26)=>REG_DESPL1(12),
Q(27)=>REG_DESPL1(13)
);

U9: registro port map(
ENTRADA=>REG_DESPL1,
SALIDA=>REG_VALID1,
EN=>EN_VALID,
CLK=>CLK_M
);

U10: registro port map(
ENTRADA=>REG_DESPL2,
SALIDA=>REG_VALID2,
EN=>EN_VALID,
CLK=>CLK_M
);
U11: MUX2x2 port map(
E1=>REG_VALID1,
E2=>REG_VALID2,
```

```
        EN_BOTON=>BOTON,
        S=>REG_VALID3
        );

        U12: visualizacion port map(
        E3(0)=>REG_VALID3(0),
        E3(1)=>REG_VALID3(1),
        E3(2)=>REG_VALID3(2),
        E3(3)=>REG_VALID3(3),
        E2(0)=>REG_VALID3(4),
        E2(1)=>REG_VALID3(5),
        E2(2)=>REG_VALID3(6),
        E2(3)=>'0',
        E1(0)=>REG_VALID3(7),
        E1(1)=>REG_VALID3(8),
        E1(2)=>REG_VALID3(9),
        E1(3)=>REG_VALID3(10),
        E0(0)=>REG_VALID3(11),
        E0(1)=>REG_VALID3(12),
        E0(2)=>REG_VALID3(13),
        E0(3)=>'0',
        CLK=>CLK,
        SEG7=>Seg7_aux,
        AN=>AN
        );
        SEG7<=Seg7_aux;
end Behavioral;
```

5.3 Mejora Día

Se incluyen solo las modificaciones respecto a la mejora 2 (Mejora Fecha).



5.3.1 Módulo 1: Registro de desplazamiento de 31 bits

```
----------------------------------------------------------------------------------
-- REGISTRO DE DESPLAZAMIENTO DE 31 BITS
--
-- Este registro se encarga de recoger y
-- almacenar los últimos 31 bits que le
-- han llegado por su entrada, y los
-- devuelve por otra salida de 31 bits.
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity reg_desp is
    Port ( SIN : in  STD_LOGIC;                              -- Señal de entrada
           CLK : in  STD_LOGIC;                              -- Señal de reloj de muestreo
           EN : in  STD_LOGIC;                               -- Enable
           Q : out  STD_LOGIC_VECTOR (30 downto 0));         -- Señal de salida
end reg_desp;

architecture Behavioral of reg_desp is

       signal Qs : STD_LOGIC_VECTOR (30 downto 0):= (others=>'0');-- señal auxiliar del registro
begin
PROC_REG : process (CLK)                    -- en este proceso se carga cada bit en el espacio
                                            -- adyacente y por último la entrada en último lugar
begin
       if (CLK'event and CLK='1' and EN='1') then
              Qs(0)<=SIN;
              Qs(1)<=Qs(0);
              Qs(2)<=Qs(1);
              Qs(3)<=Qs(2);
              Qs(4)<=Qs(3);
              Qs(5)<=Qs(4);
              Qs(6)<=Qs(5);
              Qs(7)<=Qs(6);
              Qs(8)<=Qs(7);
              Qs(9)<=Qs(8);
              Qs(10)<=Qs(9);
              Qs(11)<=Qs(10);
              Qs(12)<=Qs(11);
              Qs(13)<=Qs(12);
              Qs(14)<=Qs(13);
              Qs(15)<=Qs(14);
              Qs(16)<=Qs(15);
              Qs(17)<=Qs(16);
              Qs(18)<=Qs(17);
              Qs(19)<=Qs(18);
              Qs(20)<=Qs(19);
              Qs(21)<=Qs(20);
              Qs(22)<=Qs(21);
              Qs(23)<=Qs(22);
              Qs(24)<=Qs(23);
              Qs(25)<=Qs(24);
              Qs(26)<=Qs(25);
              Qs(27)<=Qs(26);
              Qs(28)<=Qs(27);
              Qs(29)<=Qs(28);
              Qs(30)<=Qs(29);
       end if;

Q<=Qs;                        -- se carga la señal auxiliar en la salida
end process;

end Behavioral;
```

5.3.2 Módulo 2: Decodificador del día de la semana

```
----------------------------------------------------------------------------------
-- DECODIFICADOR DEL DÍA DE LA SEMANA
--
-- Este decodificador genera una señal de salida
-- que enciende el LED correspondiente al día de
-- la semana, en función de la entrada.
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity DECOD_DIA is
    Port ( ENTRADA : in  STD_LOGIC_VECTOR (2 downto 0):= (others=>'1');   -- Entrada
           SALIDA : out  STD_LOGIC_VECTOR (6 downto 0));                  -- Salida
end DECOD_DIA;

architecture Behavioral of DECOD_DIA is

begin
               with ENTRADA select SALIDA<=         --  la salida enciende el LED correspondiente
               "1000000" when "000",
               "0100000" when "001",
               "0010000" when "010",
               "0001000" when "011",
               "0000100" when "100",
               "0000010" when "101",
               "0000001" when "110",
               "0000000" when "111",
               "0000000" when others;


end Behavioral;
```

5.3.3 Entidad jerárquica TOP: Principal

```
----------------------------------------------------------------------------------
-- MÓDULO TOP PRINCIPAL
--
-- Conecta las entradas y salidas de todos los
-- componentes del circuito digital
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

entity principal is
    Port ( CLK : in  STD_LOGIC;                             -- Señal de reloj
           SIN : in  STD_LOGIC;                             -- Señal de entrada
           AN : out  STD_LOGIC_VECTOR (3 downto 0);         -- Control de displays
           SEG7 : out  STD_LOGIC_VECTOR (6 downto 0);       -- Salida de display
           BOTON : in  STD_LOGIC);                          -- Botón de cambio a fecha
end principal;

architecture Behavioral of principal is

       constant UMBRAL1 : STD_LOGIC_VECTOR (5 downto 0) := "100010"; -- 34
       constant UMBRAL2 : STD_LOGIC_VECTOR (5 downto 0) := "100110"; -- 38

       signal CLK_M: STD_LOGIC;
       signal REG40: STD_LOGIC_VECTOR (39 downto 0);
       signal SUM: STD_LOGIC_VECTOR (5 downto 0);
       signal COMP1_MOORE: STD_LOGIC;
       signal COMP1_AND: STD_LOGIC;
       signal COMP2_AND: STD_LOGIC;
       signal AND_AUX: STD_LOGIC;
       signal SIN_REG: STD_LOGIC;
```

```
	signal EN_REG: STD_LOGIC;
	signal EN_VALID: STD_LOGIC;
	signal REG_DESPL1: STD_LOGIC_VECTOR (13 downto 0);
	signal REG_DESPL2: STD_LOGIC_VECTOR (13 downto 0);
	signal REG_DESPL3: STD_LOGIC_VECTOR (2 downto 0);
	signal REG_VALID1: STD_LOGIC_VECTOR (13 downto 0);
	signal REG_VALID2: STD_LOGIC_VECTOR (13 downto 0);
	signal REG_VALID3: STD_LOGIC_VECTOR (13 downto 0);
	signal REG_VALID_DIA: STD_LOGIC_VECTOR (13 downto 0);
	signal Seg7_aux: STD_LOGIC_VECTOR (0 to 6)

component gen_reloj
	Port ( CLK : in STD_LOGIC;           -- Reloj de la FPGA
	       CLK_OUT : out STD_LOGIC);     -- Reloj de frecuencia dividida
end component;

component reg_desp40
	Port ( SIN : in STD_LOGIC;           -- Datos de entrada serie
	       CLK : in STD_LOGIC;           -- Reloj de muestreo
	       Q : out STD_LOGIC_VECTOR (39 downto 0)); -- Salida paralelo
end component;

component sumador40
	Port ( ENT : in STD_LOGIC_VECTOR (39 downto 0);
	       SAL : out STD_LOGIC_VECTOR (5 downto 0));
end component;

component comparador
	Port ( P : in STD_LOGIC_VECTOR (5 downto 0);
	       Q : in STD_LOGIC_VECTOR (5 downto 0);
	       PGTQ : out STD_LOGIC;
	       PLEQ : out STD_LOGIC);
end component;

component AND_2
	Port ( A : in STD_LOGIC;
	       B : in STD_LOGIC;
	       S : out STD_LOGIC);
end component;

component reg_desp
	Port ( SIN : in STD_LOGIC;           -- Datos de entrada serie
	       CLK : in STD_LOGIC;           -- Reloj
	       EN : in STD_LOGIC;            -- Enable
	       Q : out STD_LOGIC_VECTOR (30 downto 0)); -- Salida paralelo
end component;

component registro
	Port ( ENTRADA : in STD_LOGIC_VECTOR (13 downto 0);
	       SALIDA : out STD_LOGIC_VECTOR (13 downto 0);
	       EN : in STD_LOGIC;            -- Enable
	       CLK : in STD_LOGIC;
	       SAL_LED : out  STD_LOGIC);
end component;

component automata
	Port ( CLK : in STD_LOGIC;           -- Reloj del autómata
	       C0 : in STD_LOGIC;            -- Condición de decision para "0"
	       C1 : in STD_LOGIC;            -- Condición de decisión para "1"
	       DATO : out STD_LOGIC;         -- Datos a cargar
	       CAPTUR : out STD_LOGIC;       -- Enable del reg. de desplaz.
	       VALID : out STD_LOGIC);       -- Activación registro
end component;

component visualizacion
	Port ( E0 : in STD_LOGIC_VECTOR (3 downto 0);       -- Entrada MUX 0
	       E1 : in STD_LOGIC_VECTOR (3 downto 0);       -- Entrada MUX 1
	       E2 : in STD_LOGIC_VECTOR (3 downto 0);       -- Entrada MUX 2
```

```
                E3 : in STD_LOGIC_VECTOR (3 downto 0);       -- Entrada MUX 3
                CLK : in STD_LOGIC;                          -- Entrada de reloj FPGA
                SEG7 : out STD_LOGIC_VECTOR (0 to 6);        -- Salida para los displays
                AN : out STD_LOGIC_VECTOR (3 downto 0));     -- Activación individual
end component;

component MUX2x2
        Port ( E1 : in STD_LOGIC_VECTOR (13 downto 0);       -- Entrada 1
                E2 : in STD_LOGIC_VECTOR (13 downto 0);      -- Entrada 2
                S : out STD_LOGIC_VECTOR (13 downto 0);      -- Salida
                EN_BOTON : in STD_LOGIC);                    -- Enable
end component;

component DECOD_DIA
        Port ( ENTRADA : in STD_LOGIC_VECTOR (2 downto 0);  -- Entrada
                SALIDA : out STD_LOGIC_VECTOR (2 downto 0));-- Salida
end component;

begin
        U1: gen_reloj port map(
        CLK=>CLK,
        CLK_OUT=>CLK_M
        );

        U2: reg_desp40 port map(
        CLK=>CLK_M,
        SIN=>SIN,
        Q=>REG40
        );

        U3: sumador40 port map(
        ENT=>REG40,
        SAL=>SUM
        );

        U4: comparador port map(
        P=>SUM,
        Q=>UMBRAL1,
        PGTQ=>COMP1_AND,
        PLEQ=>COMP1_MOORE
        );

        U5: comparador port map(
        P=>SUM,
        Q=>UMBRAL2,
        PLEQ=>COMP2_AND
        );

        U6: AND_2 port map(
        A=>COMP1_AND,
        B=>COMP2_AND,
        S=>AND_AUX
        );

        U7: automata port map(
        CLK=>CLK_M,
        C0=>AND_AUX,
        C1=>COMP1_MOORE,
        DATO=>SIN_REG,
        CAPTUR=>EN_REG,
        VALID=>EN_VALID
        );

        U8: reg_desp port map(
        SIN=>SIN_REG,
        CLK=>CLK_M,
```

```
EN=>EN_REG,
Q(0)=>REG_DESPL3(0),
Q(1)=>REG_DESPL3(1),
Q(2)=>REG_DESPL3(2),
Q(3)=>REG_DESPL2(0),
Q(4)=>REG_DESPL2(1),
Q(5)=>REG_DESPL2(2),
Q(6)=>REG_DESPL2(3),
Q(7)=>REG_DESPL2(4),
Q(8)=>REG_DESPL2(5),
Q(9)=>REG_DESPL2(6),
Q(10)=>REG_DESPL2(7),
Q(11)=>REG_DESPL2(8),
Q(12)=>REG_DESPL2(9),
Q(13)=>REG_DESPL2(10),
Q(14)=>REG_DESPL2(11),
Q(15)=>REG_DESPL2(12),
Q(16)=>REG_DESPL2(13),
Q(17)=>REG_DESPL1(0),
Q(18)=>REG_DESPL1(1),
Q(19)=>REG_DESPL1(2),
Q(20)=>REG_DESPL1(3),
Q(21)=>REG_DESPL1(4),
Q(22)=>REG_DESPL1(5),
Q(23)=>REG_DESPL1(6),
Q(24)=>REG_DESPL1(7),
Q(25)=>REG_DESPL1(8),
Q(26)=>REG_DESPL1(9),
Q(27)=>REG_DESPL1(10),
Q(28)=>REG_DESPL1(11),
Q(29)=>REG_DESPL1(12),
Q(30)=>REG_DESPL1(13)
);

U9: registro port map(
ENTRADA=>REG_DESPL1,
SALIDA=>REG_VALID1,
EN=>EN_VALID,
CLK=>CLK_M
);

U10: registro port map(
ENTRADA=>REG_DESPL2,
SALIDA=>REG_VALID2,
EN=>EN_VALID,
CLK=>CLK_M
);

U11: MUX2x2 port map(
E1=>REG_VALID1,
E2=>REG_VALID2,
EN_BOTON=>BOTON,
S=>REG_VALID3
);

U12: visualizacion port map(
E3(0)=>REG_VALID3(0),
E3(1)=>REG_VALID3(1),
E3(2)=>REG_VALID3(2),
E3(3)=>REG_VALID3(3),
E2(0)=>REG_VALID3(4),
E2(1)=>REG_VALID3(5),
E2(2)=>REG_VALID3(6),
E2(3)=>'0',
E1(0)=>REG_VALID3(7),
E1(1)=>REG_VALID3(8),
E1(2)=>REG_VALID3(9),
E1(3)=>REG_VALID3(10),
```

```
        E0(0)=>REG_VALID3(11),
        E0(1)=>REG_VALID3(12),
        E0(2)=>REG_VALID3(13),
        E0(3)=>'0',
        CLK=>CLK,
        SEG7=>Seg7_aux,
        AN=>AN
        );

        U13: registro port map(
        ENTRADA(0)=>REG_DESPL3(0),
        ENTRADA(1)=>REG_DESPL3(1),
        ENTRADA(2)=>REG_DESPL3(2),
        ENTRADA(3)=>'0',
        ENTRADA(4)=>'0',
        ENTRADA(5)=>'0',
        ENTRADA(6)=>'0',
        ENTRADA(7)=>'0',
        ENTRADA(8)=>'0',
        ENTRADA(9)=>'0',
        ENTRADA(10)=>'0',
        ENTRADA(11)=>'0',
        ENTRADA(12)=>'0',
        ENTRADA(13)=>'0',
        SALIDA=>REG_VALID_DIA,
        EN=>EN_VALID,
        CLK=>CLK_M
        );

        SEG7<=Seg7_aux;
end Behavioral;
```

ANEXO I

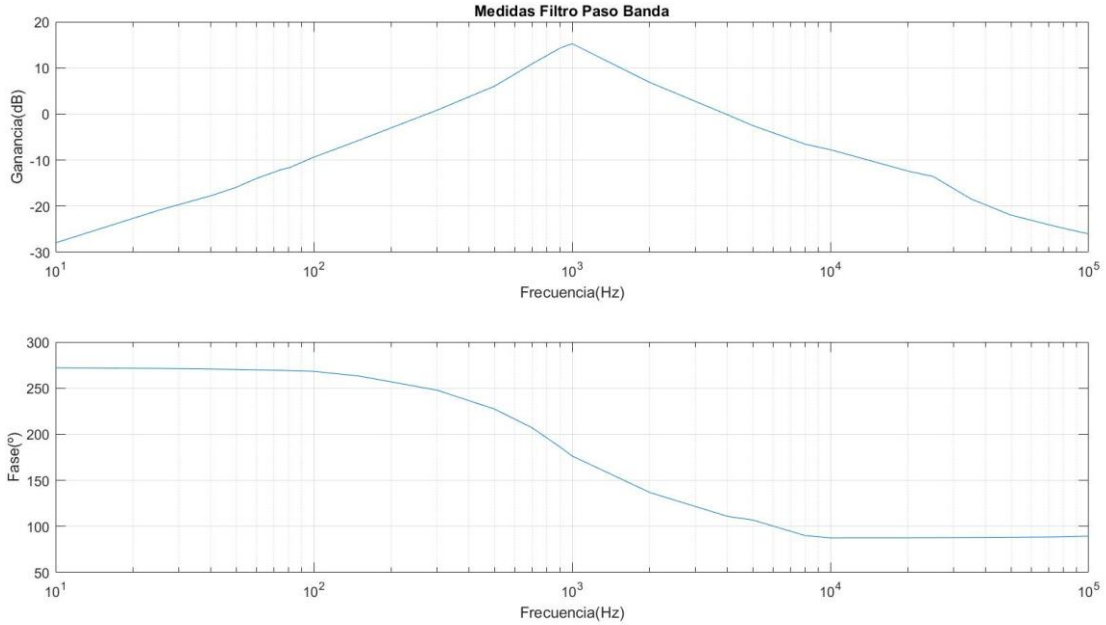
Entrega 1

CÁLCULOS Y MEDIDAS SOBRE EL FILTRO PASO BANDA

frecuencia (Hz)	entrada(Vpp)	salida(Vpp)	ganancia	ganancia(dB)	Variación T(s)	fase (º)
10	5	0,04	0,008	-41,93820026	0,075555556	272
25	5	0,09	0,018	-34,8945499	0,030155556	271,4
40	1	0,13	0,13	-17,72113295	0,018798611	270,7
50	1	0,16	0,16	-15,91760035	0,015016667	270,3
60	1	0,2	0,2	-13,97940009	0,01249537	269,9
75	1	0,25	0,25	-12,04119983	0,009974074	269,3
80	1	0,26	0,26	-11,70053304	0,009340278	269
100	1	0,34	0,34	-9,370421659	0,00745	268,2
150	1	0,52	0,52	-5,679933127	0,004872222	263,1
300	1	1,1	1,1	0,827853703	0,002294444	247,8
500	1	2	2	6,020599913	0,001263333	227,4
700	1	3,5	3,5	10,88136089	0,000821429	207
900	1	5,2	5,2	14,32006687	0,000574074	186
1000	1	5,92	5,92	15,44643413	0,00049	176,4
frecuencia (Hz)	entrada(Vpp)	salida(Vpp)	ganancia	ganancia(dB)	Variación T(s)	fase (º)
2000	1	2,32	2,32	7,309759698	0,00023125	166,5
4000	1	1	1	0	0,000101944	146,8
5000	1	0,75	0,75	-2,498774732	7,60556E-05	136,9
8000	1	0,47	0,47	-6,558042841	3,71528E-05	107
10000	1	0,41	0,41	-7,744322866	2,43333E-05	87,6
15000	1	0,3	0,3	-10,45757491	1,62407E-05	87,7
20000	1	0,24	0,24	-12,39577517	1,21806E-05	87,7
25000	1	0,21	0,21	-13,55561411	9,75556E-06	87,8
35000	1	0,12	0,12	-18,41637508	6,98413E-06	88
50000	1	0,08	0,08	-21,93820026	0,0000049	88,2
75000	1	0,06	0,06	-24,43697499	3,28148E-06	88,6
100000	1	0,05	0,05	-26,02059991	2,48333E-06	89,4

El valor del cero es 0, los polos son -1773.1 ± 5880.5j y sus módulos son 6142

$$H(jw)=\frac{-jw\cdot\frac{1}{R1C}}{-w^2+jw\frac{2}{R2C}+\frac{1}{R1R2C^2}} \qquad H(s)=\frac{-s\cdot\frac{1}{R1C}}{s^2+s\frac{2}{R2C}+\frac{1}{R1R2C^2}}$$



ANEXO II

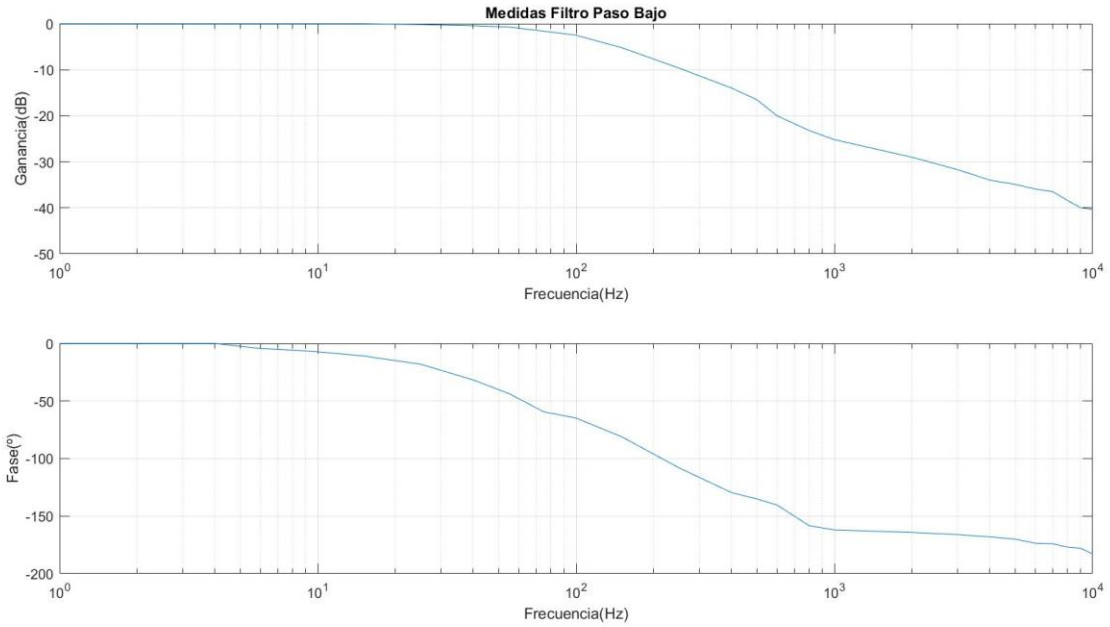
Entrega 2

CÁLCULOS Y MEDIDAS SOBRE EL FILTRO PASO BAJO

frecuencia (Hz)	entrada(Vpp)	salida(Vpp)	ganancia	ganancia(dB)	Variacion T(s)	fase (º)
1	1	1	1	0	0	0
4	1	1	1	0	0	0
6	1	1	1	0	-2,00E-03	-4,32
8	1	1	1	0	-2,00E-03	-5,76
10	1	1	1	0	-2,00E-03	-7,2
15	1	1	1	0	-2,00E-03	-10,8
25	1	0,98	0,98	-0,17547849	-2,00E-03	-18
40	1	0,95	0,95	-0,44552789	-2,20E-03	-31,68
55	1	0,92	0,92	-0,72424345	-2,20E-03	-43,56
75	1	0,83	0,83	-1,61843815	-2,20E-03	-59,4
100	1	0,75	0,75	-2,49877473	-1,80E-03	-64,8
150	1	0,55	0,55	-5,19274621	-1,50E-03	-81
250	1	0,33	0,33	-9,6297212	-1,20E-03	-108
400	1	0,18	0,18	-14,8945499	-9,00E-04	-129,6
500	1	0,13	0,13	-17,721133	-7,50E-04	-135
600	1	0,07	0,07	-23,0980392	-6,50E-04	-140,4
800	3,9	0,16	0,041026	-27,7388925	-5,50E-04	-158,4
1000	3,98	0,1	0,025126	-31,9976614	-4,50E-04	-162
2000	4	0,0936	0,0234	-32,6156829	-2,28E-04	-164,2
3000	4	0,084	0,021	-33,5556141	-1,54E-04	-166,1
4000	4	0,08	0,02	-33,9794001	-1,17E-04	-168,1
5000	4,5	0,081	0,018	-34,8945499	-9,44E-05	-170
6000	4,5	0,072	0,016	-35,9176003	-8,04E-05	-173,6
7000	4,5	0,0675	0,015	-36,4781748	-6,90E-05	-174
8000	5	0,06	0,012	-38,4163751	-6,15E-05	-177
9000	5	0,055	0,011	-39,1721463	-5,49E-05	-178
10000	5	0,055	0,011	-39,1721463	-5,09E-05	-183,4

$$H(jw)=\frac{G\cdot\frac{1}{R^2C^2}}{-w^2+jw\frac{3-G}{RC}+\frac{1}{R^2C^2}} \qquad H(s)=\frac{G\cdot\frac{1}{R^2C^2}}{s^2+s\frac{3-G}{RC}+\frac{1}{R^2C^2}}$$

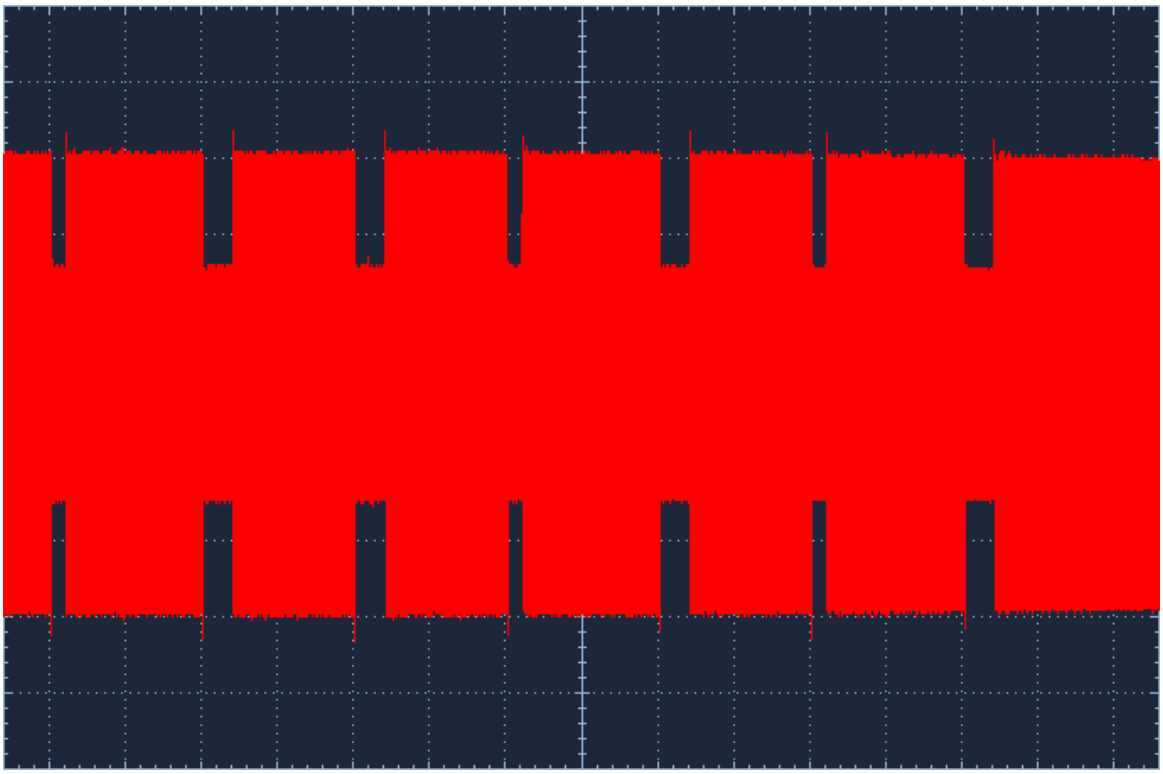
No hay ceros en esta función de transferencia, el valor del polo doble es -1000 rad/s y su módulo de 1000 rad/s o 157 Hz.



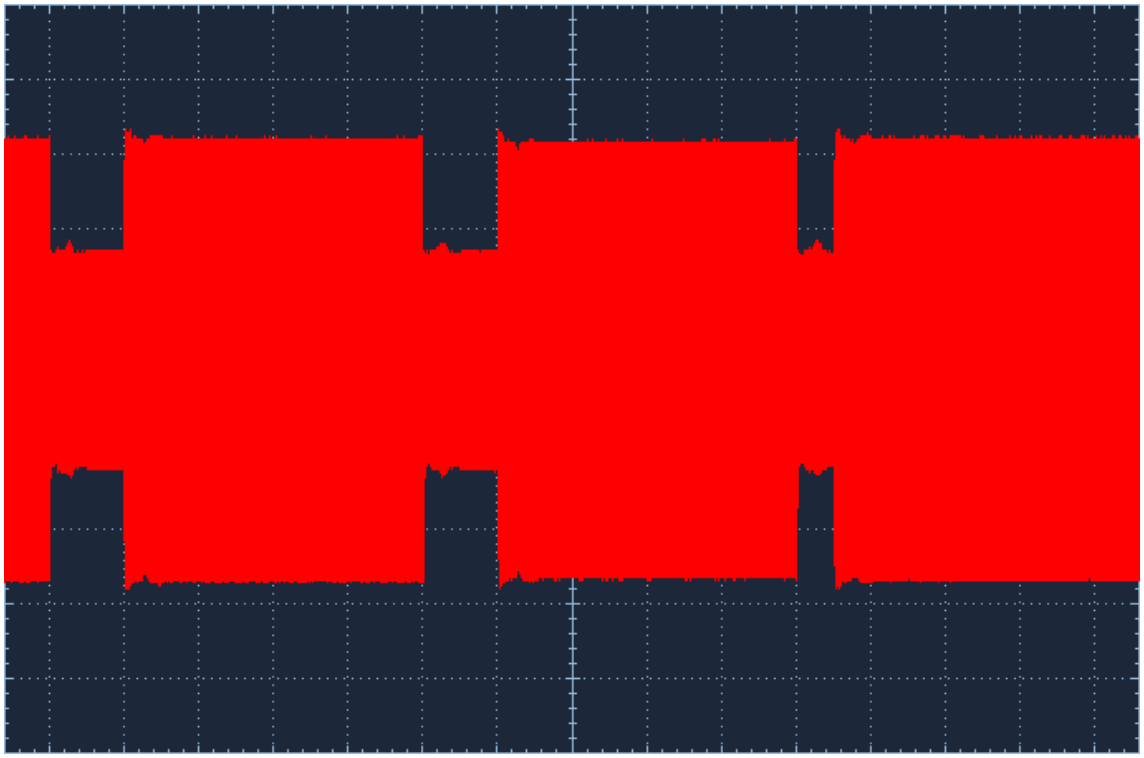
ANEXO III

Entrega 3

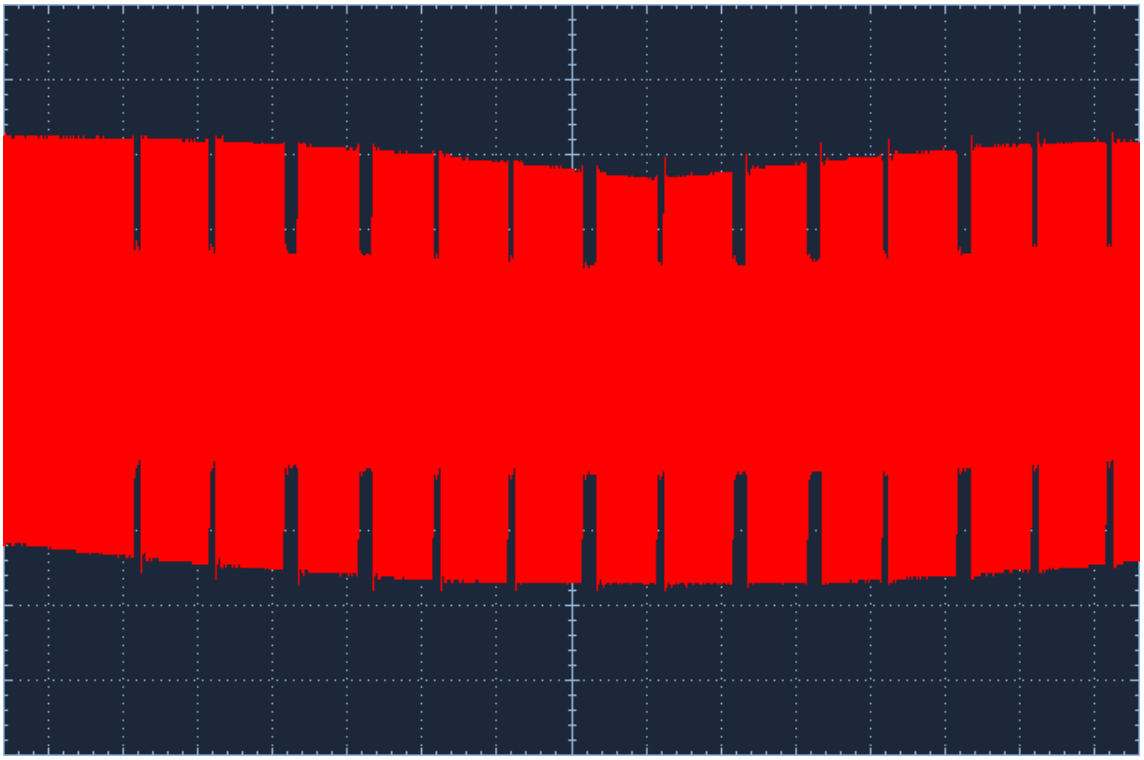
SALIDA MP3



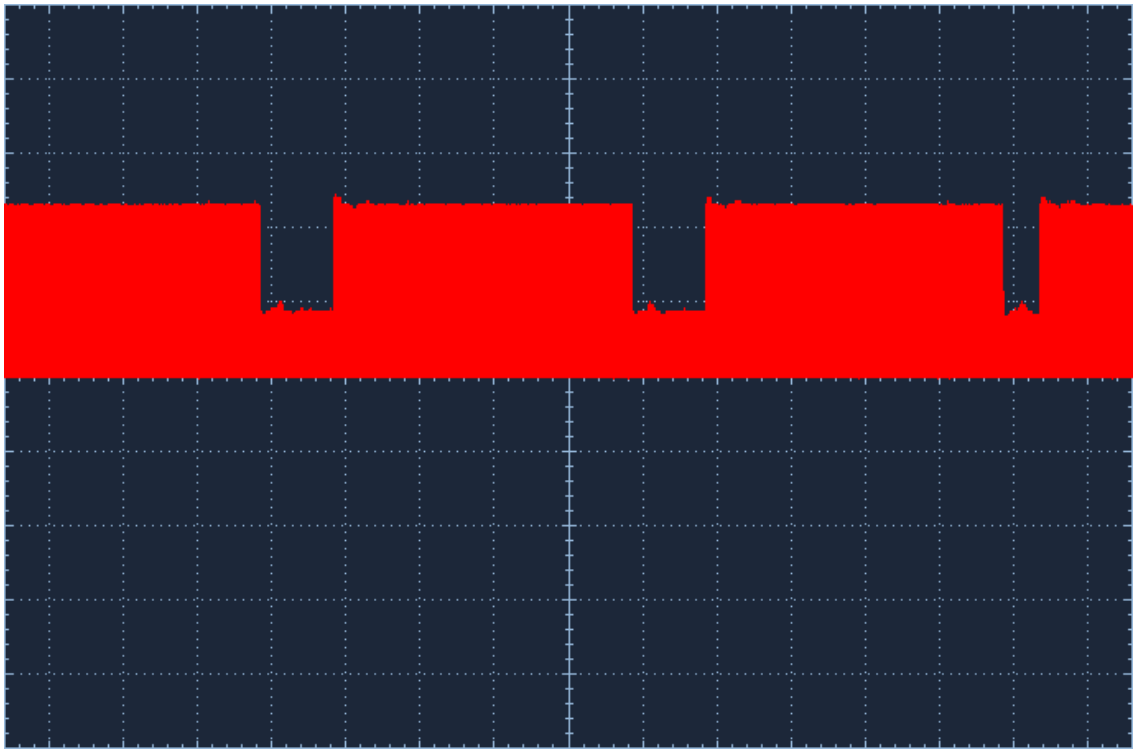
SEÑAL A LA SALIDA DEL PASO BANDA



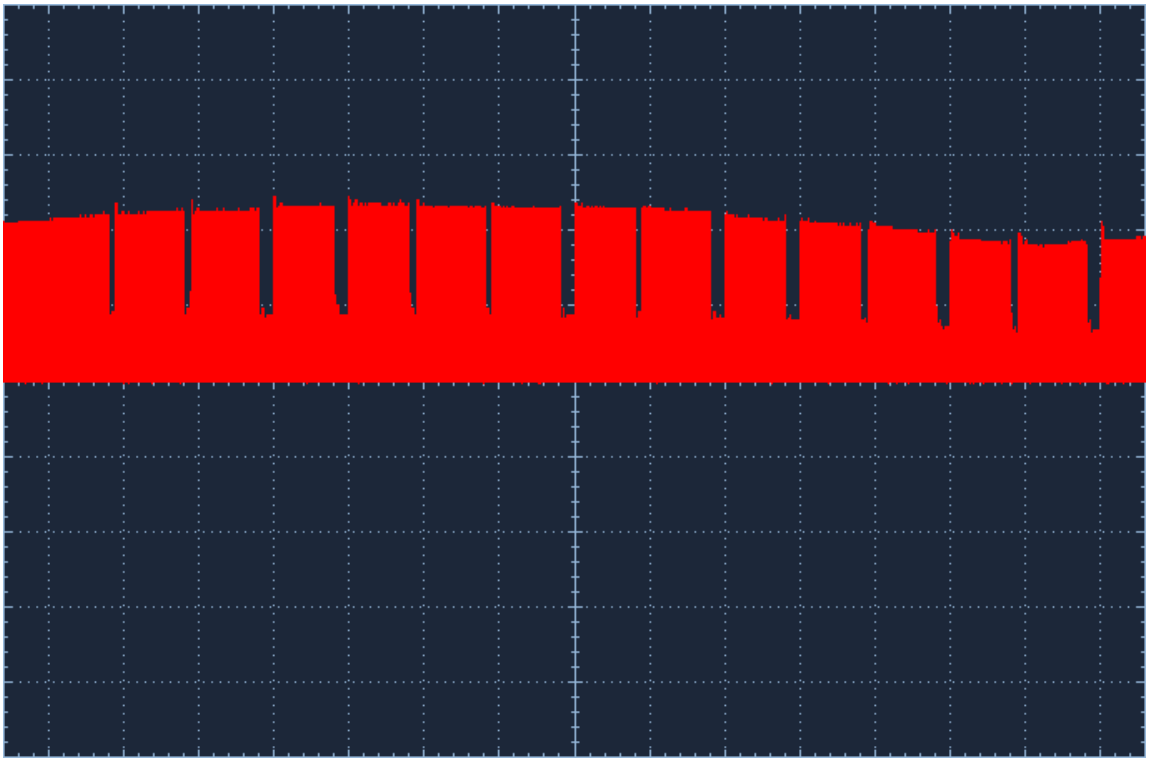
SEÑAL A LA SALIDA DEL PASO BANDA (TRAMA COMPLETA)



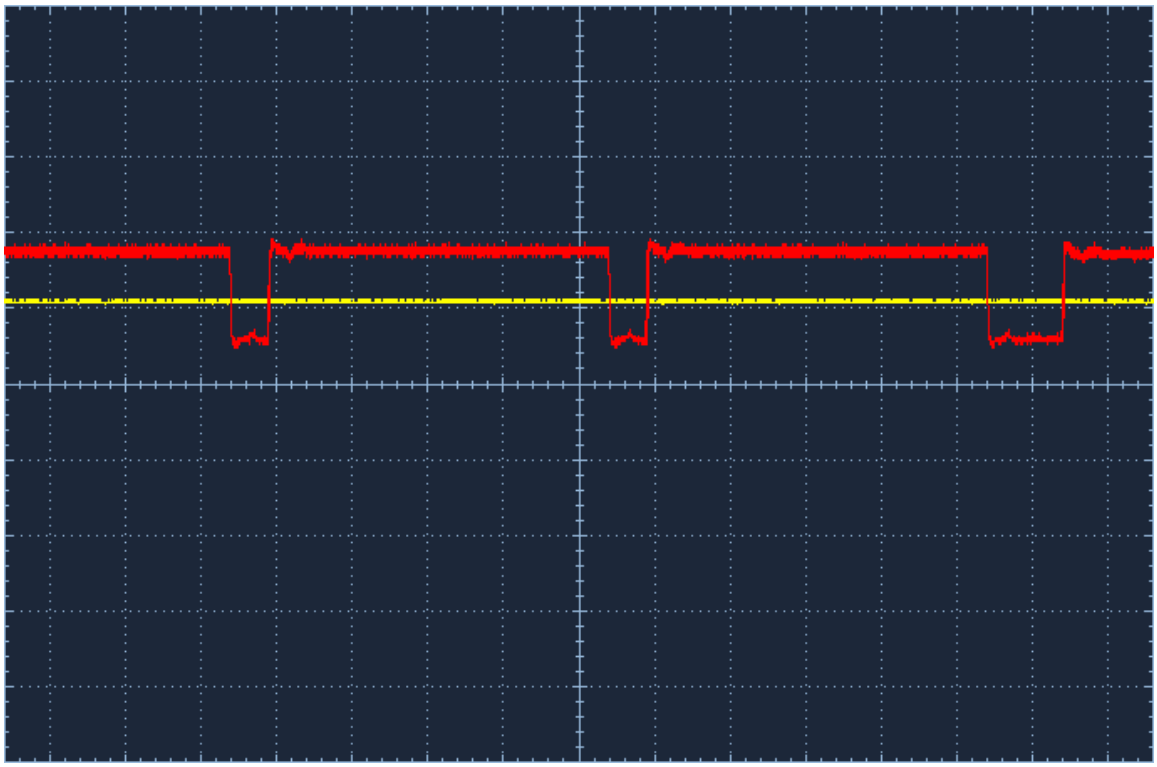
SEÑAL A LA SALIDA DEL RECTIFICADOR



SEÑAL A LA SALIDA DEL RECTIFICADOR (TRAMA COMPLETA)



SEÑAL A LA SALIDA DEL PASO BAJO



SEÑAL DE 40 Hz OBTENIDA MEDIANTE DIVISIÓN DE RELOJ