Universidad Politécnica de Madrid
ETSI de Telecomunicación
Departamento de Ingeniería Electrónica

Circuitos Electrónicos (CELT)

Curso 2017-2018

Memoria final

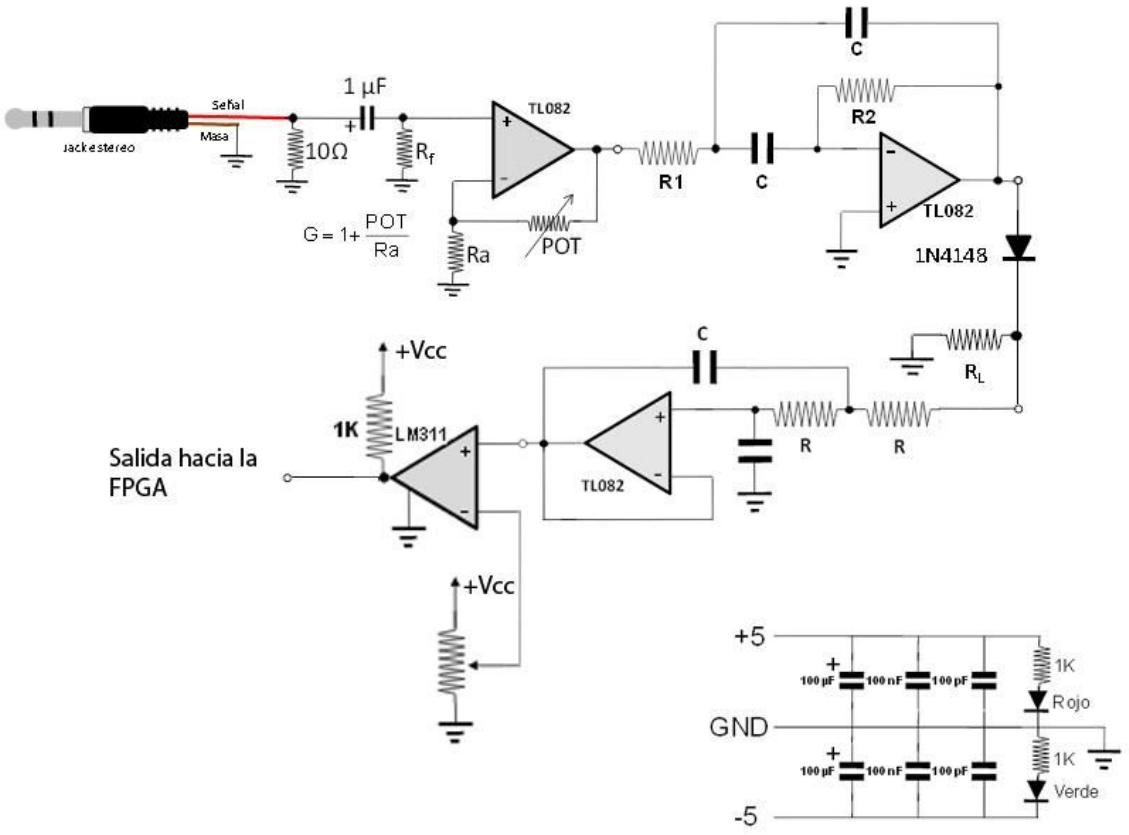
	Apellidos	Nombre
Alumno 1	Hernández Nogueira	Patricia
Alumno 2	Andrés Bruna	Camilo

Código de pareja	MT14

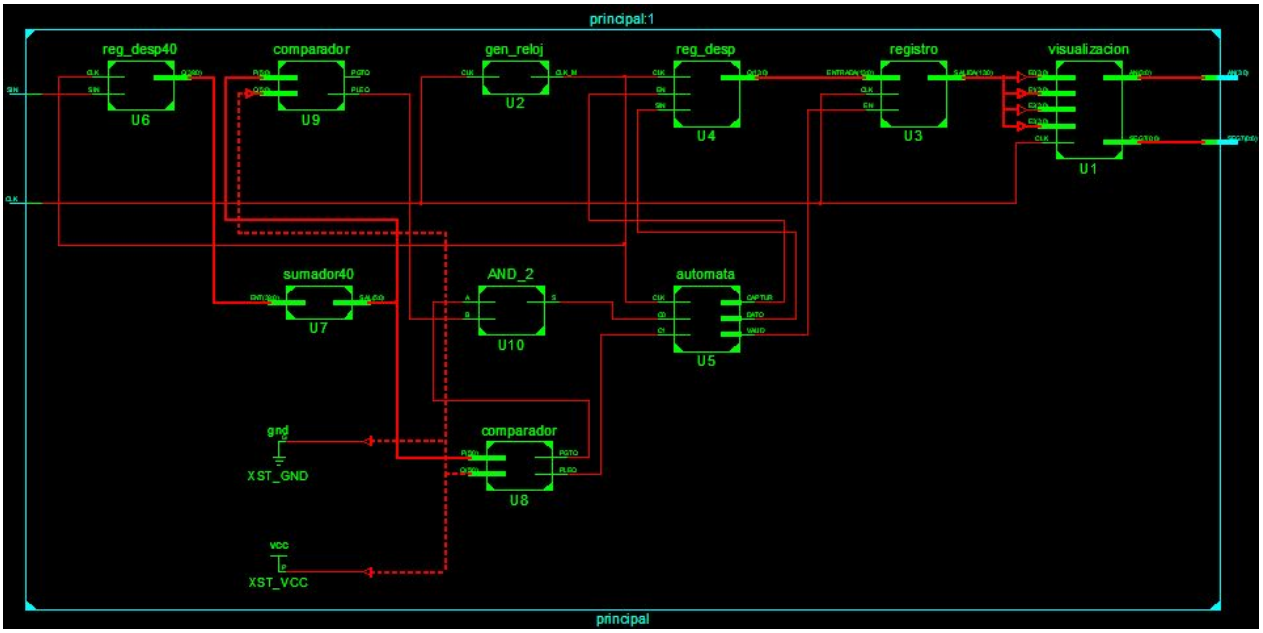
1. ESQUEMA COMPLETO DEL CIRCUITO ANALÓGICO

Todos los A.O están conectados a +Vcc = 5 V y -Vcc = -5 V.

Los valores de cada componente los detallamos, junto con cómo los hemos obtenido,más adelante.



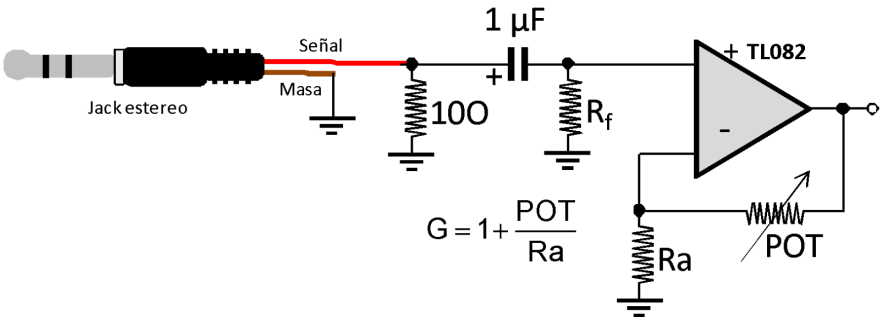
2. ESQUEMA COMPLETO DEL CIRCUITO DIGITAL



3. DISEÑO DETALLADO DEL CIRCUITO ANALÓGICO

3.1 Acondicionador de señal

3.1.1 Amplificador y eliminación de posible componente continua



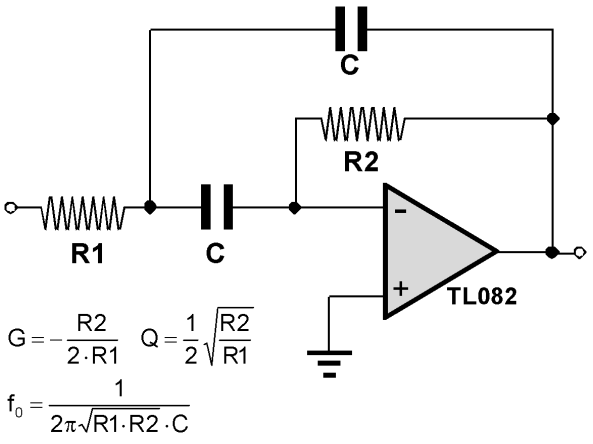
3.1.1.1 Diseño del filtro paso alto

En el manual se indica que la frecuencia de corte debe ser menor de 20 Hz y estar en un valor entre 10 y 15 Hz. Con este filtro se elimina la posible componente continua procedente del mp3. Nosotros tomamos en un primer momento una fc=12,5 Hz para ello necesitaríamos una Rf=13 KΩ. Puesto que conseguimos una resistencia de 12 KΩ , reajustamos la fc= 13,26 Hz.

3.1.1.2 Diseño del amplificador

La condición que se nos da es que la ganancia permita obtener 1Vpp a la salida del operacional para una señal de entrada a un volumen concreto desde el dispositivo de reproducción. De esta manera, POT y Ra pueden tomar cualquier valor siempre que cumplan esta relación. A partir de la expresión que se ve en el esquema del circuito hemos tomado Ra=2 KΩ y un potenciometro de 10 KΩ que hemos ajustado para obtener una ganancia razonable.

3.1.2 Filtro Paso Banda



El filtro limita en banda la señal que pasa al demodulador, reduciendo de este modo la posible interferencia producida por señales diferentes a la señal modulada de 1KHz, con lo que su fc=1kHz y su ganancia en el centro de su banda será 6. Empleando las ecuaciones del manual (que aparecen en la imagen de arriba) sabemos que debemos emplear unos valores determinados de R1, R2 y C que cumplan:

$$-2 \bullet G = \frac{R2}{R1} \rightarrow -\frac{1}{12} = \frac{R1}{R2}$$

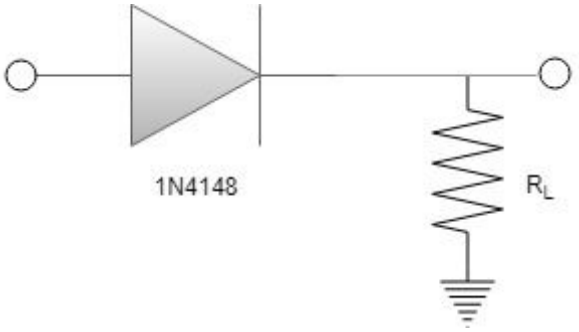
$$\sqrt{R1 \bullet R2} \bullet C = \frac{1}{2000 \bullet \pi} \rightarrow C = \frac{1}{2000 \bullet \pi \bullet \sqrt{R1 \bullet R2}}$$

Debemos elegir los valores de R1 y R2, los cuales determinarán el valor de C. Podemos emplear cualquier relación de estos que cumpla las igualdades anteriores.

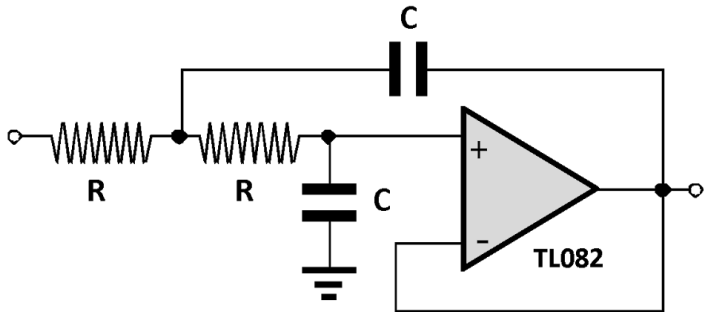
Nosotros hemos usado R1=1 KΩ y R2=12 KΩ, obteniendo con las fórmulas anteriores un valor de 45,94nF para el condensador, que aproximamos con un condensador de 47nF.

3.2 Rectificador

Usamos el rectificador del manual con RL= 1 KΩ como se indica.



3.3 Filtro Paso Bajo



$$G = 1 \quad Q = \frac{1}{2} \quad f_0 = \frac{1}{2\pi RC} \quad \omega_0 = \frac{1}{RC}$$

$$H(j\omega) = \frac{G \cdot \omega_0^2}{-\omega^2 + j\omega\frac{\omega_0}{Q} + \omega_0^2}$$

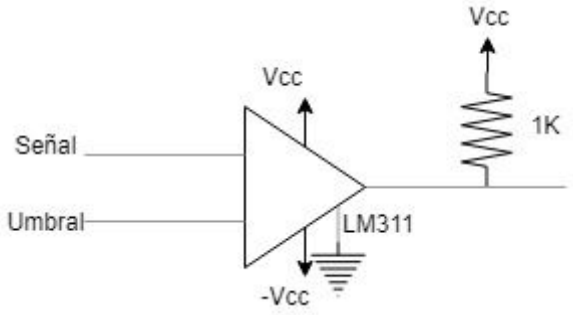
El filtro se emplea para atenuar la portadora sinusoidal de 1KHz y sus armónicos, dejando pasar la banda base correspondiente a la señal de datos.
Para calcular R y C, calculamos en primer lugar el módulo de la función de transferencia.

Los valores que hemos utilizado han sido: R = 1kΩ C = 1μF y G=1 (fijado en el enunciado)

Cálculo de R y C, calculamos en primer lugar el módulo de la función de transferencia, sustituyendo en este la frecuencia de corte entorno a los 100 Hz e igualando el módulo a 1/(2^1/2). De este modo podemos despejar Wo ,obteniendo de este modo 4 valores (2 positivos y 2 negativos), empleando uno de estos para despejar los valores de R y C.

En nuestro caso hemos empleado para C un valor de 1μF porque considerábamos que tenía una magnitud adecuada, que con el valor obtenido de Wo, podemos despejar R, que en nuestro caso es de 1 KΩ.

3.4 Comparador



Para fijar el umbral hemos analizado la señal a la salida del filtro paso bajo y nos hemos fijado en el valor medio entre las variaciones de flancos. Ese es el valor sobre el que hemos fijado este.

Para ello hemos empleado un potenciómetro de 10kΩ, puesto que con este tengamos precisión suficiente para modificar el umbral y ajustarlo a la señal. En el diseño final el valor del potenciómetro está reajustado en función del dispositivo de entrada de audio que se use y que la señal a volumen medio oscile por encima y por debajo de este según tome un valor u otro.

4. DISEÑO DETALLADO DEL CIRCUITO DIGITAL

4.1 Reg_desp40

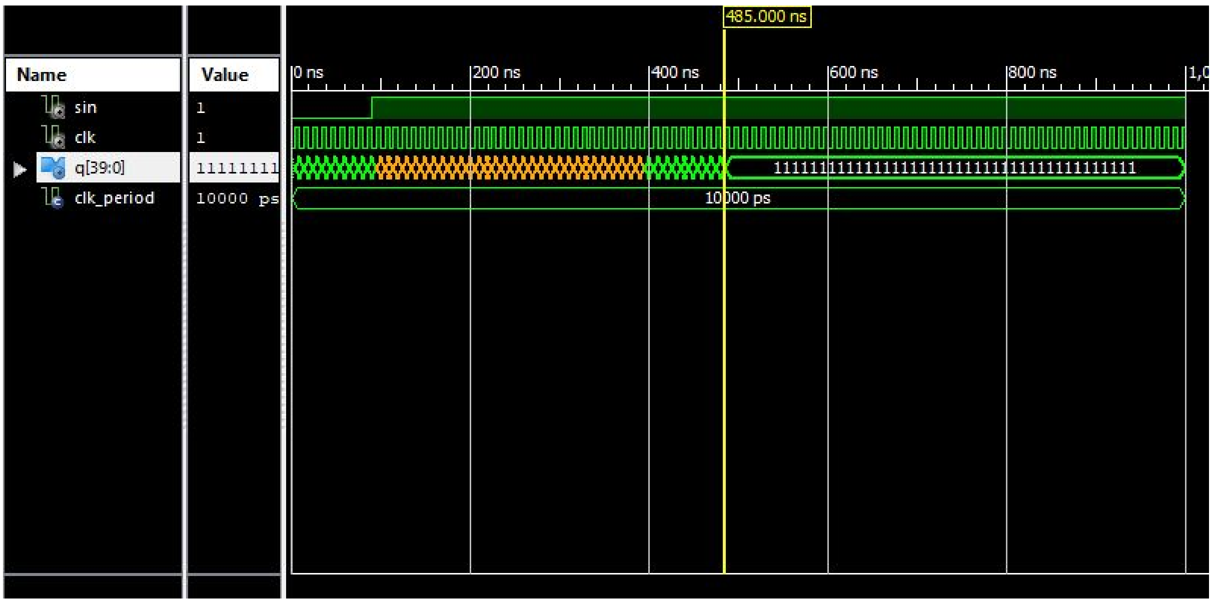
```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

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

architecture a_reg_desp40 of reg_desp40 is

signal content: std_logic_vector(39 downto 0); -- Señal que almacena el valor de Q

begin
       process(CLK)
              begin
                     if (CLK'event and CLK='1') then

                            -- content <= SIN & content(39 downto 1);-- desplazamiento
derecha con cada flanco activo
                            content(39 downto 1) <= content(38 downto 0) ;
                            content(0) <= SIN    ;


                     end if;
       end process;

Q <= content;    -- actualización de las salidas

end a_reg_desp40;
```

4.1.1 tb_reg_desp40

```
--------------------------------------------------------------------------------
-- Test Bench para prueba del registro de deplazamiento de 40 bits
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY test_reg_desp40 IS
END test_reg_desp40;

ARCHITECTURE behavior OF test_reg_desp40 IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT reg_desp40
    PORT(
         SIN : IN  std_logic;
         CLK : IN  std_logic;
         Q : OUT  std_logic_vector(39 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal SIN : std_logic := '0';
   signal CLK : std_logic := '0';
```

```
        --Outputs
   signal Q : std_logic_vector(39 downto 0);

   -- Clock period definitions
   constant CLK_period : time := 10 ns;

BEGIN

        -- Instantiate the Unit Under Test (UUT)
   uut: reg_desp40 PORT MAP (
          SIN => SIN,
          CLK => CLK,
          Q => Q
        );

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


   -- Stimulus process
   stim_proc: process
   begin
      -- Periodo de reloj de 10 ns
      wait for 10 ns;

               -- Vamos introduciendo a través de la línea de entrada serie
               -- 8 '0' y 32 '1' de forma sícrona con el reloj

      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<='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';
        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;
                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;
```



Como no se inicializa q, hasta que no se ha terminado de cargar tiene valores por definir, funciona correctamente según el código del testbench.

4.2 Sumador40

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity sumador40 is
 Port ( ENT : in STD_LOGIC_VECTOR (39 downto 0); -- entradas (40 bits)
 SAL : out STD_LOGIC_VECTOR (5 downto 0)); -- salida (6 bits)
end sumador40;
architecture a_sumador40 of sumador40 is
signal E0 : STD_LOGIC_VECTOR (5 downto 0);
signal E1 : STD_LOGIC_VECTOR (5 downto 0);
signal E2 : STD_LOGIC_VECTOR (5 downto 0);
signal E3 : STD_LOGIC_VECTOR (5 downto 0);
signal E4 : STD_LOGIC_VECTOR (5 downto 0);
signal E5 : STD_LOGIC_VECTOR (5 downto 0);
signal E6 : STD_LOGIC_VECTOR (5 downto 0);
signal E7 : STD_LOGIC_VECTOR (5 downto 0);
signal E8 : STD_LOGIC_VECTOR (5 downto 0);
signal E9 : STD_LOGIC_VECTOR (5 downto 0);
signal E10 : STD_LOGIC_VECTOR (5 downto 0);
signal E11 : STD_LOGIC_VECTOR (5 downto 0);
signal E12 : STD_LOGIC_VECTOR (5 downto 0);
signal E13 : STD_LOGIC_VECTOR (5 downto 0);
signal E14 : STD_LOGIC_VECTOR (5 downto 0);
signal E15 : STD_LOGIC_VECTOR (5 downto 0);
signal E16 : STD_LOGIC_VECTOR (5 downto 0);
signal E17 : STD_LOGIC_VECTOR (5 downto 0);
signal E18 : STD_LOGIC_VECTOR (5 downto 0);
signal E19 : STD_LOGIC_VECTOR (5 downto 0);
signal E20 : STD_LOGIC_VECTOR (5 downto 0);
signal E21 : STD_LOGIC_VECTOR (5 downto 0);
signal E22 : STD_LOGIC_VECTOR (5 downto 0);
signal E23 : STD_LOGIC_VECTOR (5 downto 0);
signal E24 : STD_LOGIC_VECTOR (5 downto 0);
signal E25 : STD_LOGIC_VECTOR (5 downto 0);
signal E26 : STD_LOGIC_VECTOR (5 downto 0);
signal E27 : STD_LOGIC_VECTOR (5 downto 0);
signal E28 : STD_LOGIC_VECTOR (5 downto 0);
signal E29 : STD_LOGIC_VECTOR (5 downto 0);
signal E30 : STD_LOGIC_VECTOR (5 downto 0);
signal E31 : STD_LOGIC_VECTOR (5 downto 0);
signal E32 : STD_LOGIC_VECTOR (5 downto 0);
signal E33 : STD_LOGIC_VECTOR (5 downto 0);
signal E34 : STD_LOGIC_VECTOR (5 downto 0);
signal E35 : STD_LOGIC_VECTOR (5 downto 0);
signal E36 : STD_LOGIC_VECTOR (5 downto 0);
signal E37 : STD_LOGIC_VECTOR (5 downto 0);
signal E38 : STD_LOGIC_VECTOR (5 downto 0);
signal E39 : STD_LOGIC_VECTOR (5 downto 0);
begin
 E39<="00000"&ENT(39);
 E38<="00000"&ENT(38);
 E37<="00000"&ENT(37);
 E36<="00000"&ENT(36);
 E35<="00000"&ENT(35);
 E34<="00000"&ENT(34);
 E33<="00000"&ENT(33);
 E32<="00000"&ENT(32);
 E31<="00000"&ENT(31);
 E30<="00000"&ENT(30);
 E29<="00000"&ENT(29);
 E28<="00000"&ENT(28);
 E27<="00000"&ENT(27);
```

```
 E26<="00000"&ENT(26);
 E25<="00000"&ENT(25);
 E24<="00000"&ENT(24);
 E23<="00000"&ENT(23);
 E22<="00000"&ENT(22);
 E21<="00000"&ENT(21);
 E20<="00000"&ENT(20);
 E19<="00000"&ENT(19);
 E18<="00000"&ENT(18);
 E17<="00000"&ENT(17);
 E16<="00000"&ENT(16);
 E15<="00000"&ENT(15);
 E14<="00000"&ENT(14);
 E13<="00000"&ENT(13);
 E12<="00000"&ENT(12);
 E11<="00000"&ENT(11);
 E10<="00000"&ENT(10);
 E9<="00000"&ENT(9);
 E8<="00000"&ENT(8);
 E7<="00000"&ENT(7);
 E6<="00000"&ENT(6);
 E5<="00000"&ENT(5);
 E4<="00000"&ENT(4);
 E3<="00000"&ENT(3);
 E2<="00000"&ENT(2);
 E1<="00000"&ENT(1);
 E0<="00000"&ENT(0);
 SAL<=E39+E38+E37+E36+E35+E34+E33+E32+E31+E30+
 E29+E28+E27+E26+E25+E24+E23+E22+E21+E20+
E19+E18+E17+E16+E15+E14+E13+E12+E11+E10+
E9+E8+E7+E6+E5+E4+E3+E2+E1+E0;
end a_sumador40;
```

4.2.1 tb_sumador40

```
--------------------------------------------------------------------------------
-- Test Bench para sumador de 40 bits de entrada
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY test_sumador40 IS
END test_sumador40;

ARCHITECTURE behavior OF test_sumador40 IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT sumador40
    PORT(
         ENT : IN  std_logic_vector(39 downto 0);
         SAL : OUT  std_logic_vector(5 downto 0)
        );
    END COMPONENT;


   --Inputs
   signal ENT : std_logic_vector(39 downto 0) := (others => '0');

        --Outputs
   signal SAL : std_logic_vector(5 downto 0);

        -- No necesita señal de reloj al tratarse de un circuito combinacional


BEGIN
```

```
        -- Instantiate the Unit Under Test (UUT)
   uut: sumador40 PORT MAP (
          ENT => ENT,
          SAL => SAL
        );



   -- Stimulus process
   stim_proc: process
   begin
      -- Probamos diferentes valores en las entradas.
               -- Sería imposible probar todos los valores
               -- Puede cambiar y probar otros valores

      wait for 10 ns;
      ENT<="0000000000000000000000000000000000000000";
      wait for 10 ns;
               ENT<="0000000000000000000011111111111111111111";
      wait for 10 ns;
               ENT<="1111111111111111111100000000000000000000";
      wait for 10 ns;
               ENT<="1010101010101010101010101010101010101010";
      wait for 10 ns;
      ENT<="0101010101010101010101010101010101010101";
      wait for 10 ns;
               ENT<="1111111111111111111111111111111111111111";
      wait for 10 ns;

      wait;
   end process;

END;
```

Es el sumador que ya viene dado, aun así se ha comprobado su correcto funcionamiento

4.3 Comparador

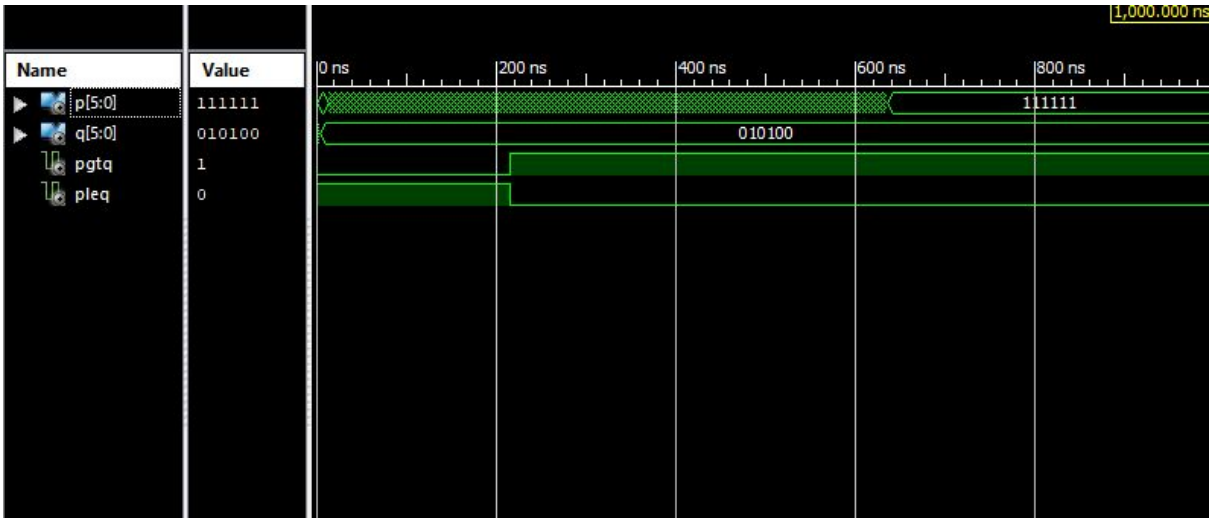
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity comparador is
 Port ( P : in STD_LOGIC_VECTOR (5 downto 0); -- Entrada P
 Q : in STD_LOGIC_VECTOR (5 downto 0); -- Entrada Q
 PGTQ : out STD_LOGIC; -- Salida P>Q
 PLEQ : out STD_LOGIC); -- Salida P=Q
end comparador;

architecture a_comparador of comparador is

begin
```

```
  PGTQ <= '1' when P > Q else '0';

  PLEQ <= '1' when P <= Q else '0';


end a_comparador;
```

4.3.1 tb_comparador

```
--------------------------------------------------------------------------------
-- Test Bench para comparador
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY test_comparador IS
END test_comparador;

ARCHITECTURE behavior OF test_comparador IS

    -- Component Declaration for the Unit Under Test (UUT)

    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;


   --Inputs
   signal P : std_logic_vector(5 downto 0) := (others => '0');
   signal Q : std_logic_vector(5 downto 0) := (others => '0');

        --Outputs
   signal PGTQ : std_logic;
   signal PLEQ : std_logic;

   -- No necesita señal de reloj al tratarse de un circuito combinacional


BEGIN

        -- Instantiate the Unit Under Test (UUT)
   uut: comparador PORT MAP (
          P => P,
          Q => Q,
          PGTQ => PGTQ,
          PLEQ => PLEQ
        );


   -- Stimulus process
   stim_proc: process
   begin
      wait for 5 ns;
               Q<="010100";  -- valor de Q=20 decimal
                                              -- Probar con otros valores

               -- P va tomado todos los valores desde 000000 hasta 111111
               P<="000000"; wait for 10 ns;
               P<="000001"; wait for 10 ns;
               P<="000010"; wait for 10 ns;
               P<="000011"; wait for 10 ns;
               P<="000100"; wait for 10 ns;
               P<="000101"; wait for 10 ns;
```

```
                P<="000110"; wait for 10 ns;
                P<="000111"; wait for 10 ns;
                P<="001000"; wait for 10 ns;
                P<="001001"; wait for 10 ns;
                P<="001010"; wait for 10 ns;
                P<="001011"; wait for 10 ns;
                P<="001100"; wait for 10 ns;
                P<="001101"; wait for 10 ns;
                P<="001110"; wait for 10 ns;
                P<="001111"; wait for 10 ns;
                P<="010000"; wait for 10 ns;
                P<="010001"; wait for 10 ns;
                P<="010010"; wait for 10 ns;
                P<="010011"; wait for 10 ns;
                P<="010100"; wait for 10 ns;
                P<="010101"; wait for 10 ns;
                P<="010110"; wait for 10 ns;
                P<="010111"; wait for 10 ns;
                P<="011000"; wait for 10 ns;
                P<="011001"; wait for 10 ns;
                P<="011010"; wait for 10 ns;
                P<="011011"; wait for 10 ns;
                P<="011100"; wait for 10 ns;
                P<="011101"; wait for 10 ns;
                P<="011110"; wait for 10 ns;
                P<="011111"; wait for 10 ns;
                P<="100000"; wait for 10 ns;
                P<="100001"; wait for 10 ns;
                P<="100010"; wait for 10 ns;
                P<="100011"; wait for 10 ns;
                P<="100100"; wait for 10 ns;
                P<="100101"; wait for 10 ns;
                P<="100110"; wait for 10 ns;
                P<="100111"; wait for 10 ns;
                P<="101000"; wait for 10 ns;
                P<="101001"; wait for 10 ns;
                P<="101010"; wait for 10 ns;
                P<="101011"; wait for 10 ns;
                P<="101100"; wait for 10 ns;
                P<="101101"; wait for 10 ns;
                P<="101110"; wait for 10 ns;
                P<="101111"; wait for 10 ns;
                P<="110000"; wait for 10 ns;
                P<="110001"; wait for 10 ns;
                P<="110010"; wait for 10 ns;
                P<="110011"; wait for 10 ns;
                P<="110100"; wait for 10 ns;
                P<="110101"; wait for 10 ns;
                P<="110110"; wait for 10 ns;
                P<="110111"; wait for 10 ns;
                P<="111000"; wait for 10 ns;
                P<="111001"; wait for 10 ns;
                P<="111010"; wait for 10 ns;
                P<="111011"; wait for 10 ns;
                P<="111100"; wait for 10 ns;
                P<="111101"; wait for 10 ns;
                P<="111110"; wait for 10 ns;
                P<="111111"; wait for 10 ns;

      wait;
   end process;

END;
```



Funciona acorde al código descrito por el testbench.

4.4 AND_2

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

entity AND_2 is
 Port ( A : in STD_LOGIC; -- Entrada A
 B : in STD_LOGIC; -- Entrada B
 S : out STD_LOGIC); -- Salida
end AND_2;

architecture a_AND_2 of AND_2 is

begin

 S <= A AND B ;

end a_AND_2;
```

4.5 Reg_desp

```
----------------------------------------
--REGISTRO DE DESPLAZAMIENTO
--
--Es el encargado de realizar la trama que se emplea para
--representar la hora(trama de 14 bits, siendo los 3
--mas significativos las decenas de hora,los 4 siguientes
--las unidades de hora,los siguientes 3 las decenas de
--minuto y los 4 ultimos las unidades de minuto.
--Esto lo hace mediante el automata, del que recibe
--la señal DATO(bit que forma parte de la
--trama) y la señal CAPTUR(indica cuando puede cargar DATO
--en la trama).
----------------------------------------



library IEEE;

use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

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

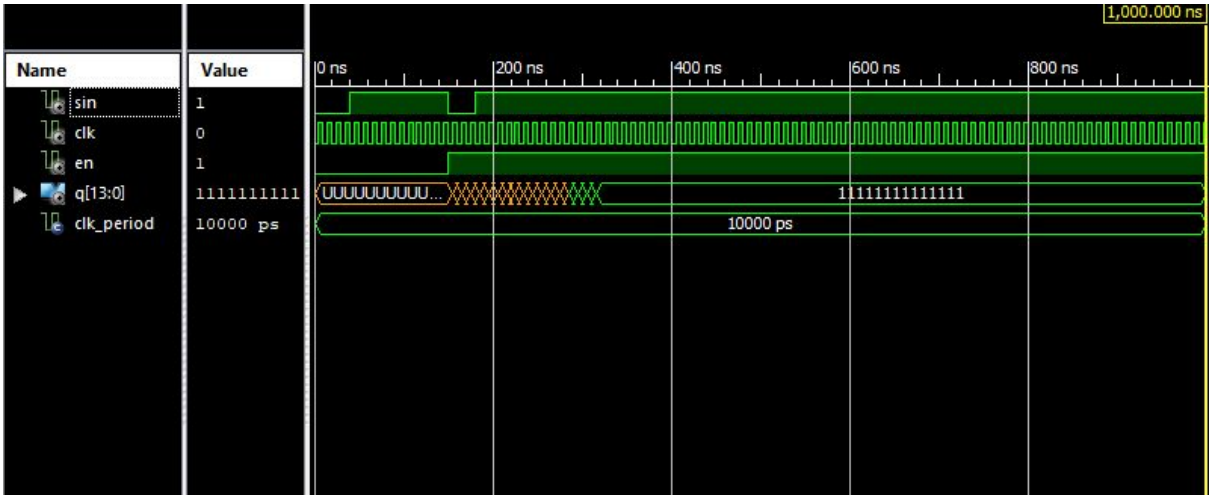
```
 CLK : in STD_LOGIC; -- Reloj
 EN : in STD_LOGIC; -- Enable
 Q : out STD_LOGIC_VECTOR (13 downto 0)); -- Salida paralelo
end reg_desp;

architecture a_reg_desp of reg_desp is

signal content: std_logic_vector(13 downto 0); --Señal auxiliar
--que se emplea para desplazar los datos cuando lo indica
--el enable.
begin
        process(CLK)
                begin
                        if (CLK'event and CLK='1' and EN = '1') then

                         content(13 downto 1) <= content(12 downto 0);
           content(0) <= SIN;


                        end if;
                                --Con cada flanco activo y cuando el enable este activo,
        --realiza el desplazamiento de datos.
        end process;

Q <= content;--Pasa los datos desplazados a la salida del registro
--de desplazamiento.
end a_reg_desp;
```

4.5.1 tb_reg_desp

```
--------------------------------------------------------------------------------
-- Test Bench para registro de desplazamiento de 14 bits
--------------------------------------------------------------------------------
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 Declaration for the Unit Under Test (UUT)

    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;


   --Inputs
   signal SIN : std_logic := '0';
   signal CLK : std_logic := '0';
   signal EN : std_logic := '0';

        --Outputs
   signal Q : std_logic_vector(13 downto 0);

   -- Clock period definitions
   constant CLK_period : time := 10 ns;

BEGIN

        -- Instantiate the Unit Under Test (UUT)
```

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

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


-- Stimulus process
stim_proc: process
begin
   -- El periodo de reloj es de 10 ns
   wait for 10 ns;
            --Desactivamos el ENABLE y cargamos 3 '0' y 11 '1' en sincronía con el reloj
            EN<='0';
   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;

            --Ahora activamos el ENABLE y cargamos 3 '0' y 11 '1' en sincronía con el reloj
            EN<='1';
   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;
                wait;
    end process;

END;
```



La salida no está inicializada pero según van entrando los valores del testbench se va actualizando sin problema dando en todo momento el resultado esperado.

4.6 gen_reloj

```
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_M : out STD_LOGIC ); -- Reloj de muestreo

end gen_reloj;

architecture a_gen_reloj of gen_reloj is

signal cont_M : STD_LOGIC_VECTOR (31 downto 0):= (others=>'0');
signal S_M : STD_LOGIC :='0';

begin
PROC_CONT : process (CLK)
begin
if CLK'event and CLK='1' then
        cont_M <= cont_M + 1;
        if cont_M >= 625000 then -- división de frecuencia a 40 Hz      50Mhz/40Hz = 1.250.000
1.250.000/2 = 625.000
                S_M <=not S_M;
                cont_M <=(others=>'0');
        end if;
```

```
end if;
end process;
       CLK_M<=S_M;
end a_gen_reloj;
```

4.7 Autómata

```
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; -- 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); -- Enable del reg de validación
end automata;
architecture a_automata of automata is
type TIPO_ESTADO is (ESP_SYNC,AVAN_ZM,MUESTREO,DATO0,DATO1,DATOSYNC);    --TIPOS DE ESTADO
signal ST : TIPO_ESTADO:= ESP_SYNC ; -- Estado inicial en que arranca
       -- signal siguiente : TIPO_ESTADO:= ESP_SYNC ;
signal salidas : STD_LOGIC_VECTOR (2 downto 0) :="000";

begin

 process (CLK)
       variable cont : STD_LOGIC_VECTOR (7 downto 0):="00000000"; -- contador
       -- para contar ciclos en un estado, iniciado a 0
 begin
       if (CLK'event and CLK = '1') then
 -- process(ST, C0, C1)
 case ST is
 when ESP_SYNC => -- Estado normal, dura 1 ciclo de reloj
       if C1='0' and C0='0'   then
       ST<= AVAN_ZM;
       else
       ST <= ESP_SYNC;
       end if;
 when AVAN_ZM => -- Estado que dura 20 ciclos de reloj
       cont:= cont+1; -- Se incrementa el contador.
       if (cont=20) then -- Si llega a 20
       cont:=(others=>'0'); -- Poner el contador a 0
       ST<=MUESTREO; -- Y cambiar de estado
       else
       ST<=AVAN_ZM; -- Si no ha llegado a 20 permanecer
       end if; -- en el mismo estado
 when MUESTREO =>
 cont:= cont+1; -- Se incrementa el contador.
 if (cont=39) then -- Si llega a 39
       cont:=(others=>'0'); -- Poner el contador a 0
       if C0='1' and C1='1'   then
       ST<= MUESTREO;
       elsif C0='0' and C1='0' then
       ST <= DATOSYNC;
       elsif C0='0' and C1='1' then
       ST <= DATO1;
       elsif C0='1' and C1='0' then
       ST <= DATO0;
       end if;
 else
       ST <= MUESTREO;
 end if;
 when DATO0 =>
 cont:= cont+1; -- Se incrementa el contador.
 if (cont=1) then -- Si llega a 1
       cont:=(others=>'0'); -- Poner el contador a 0
       ST <= MUESTREO;
       else
```

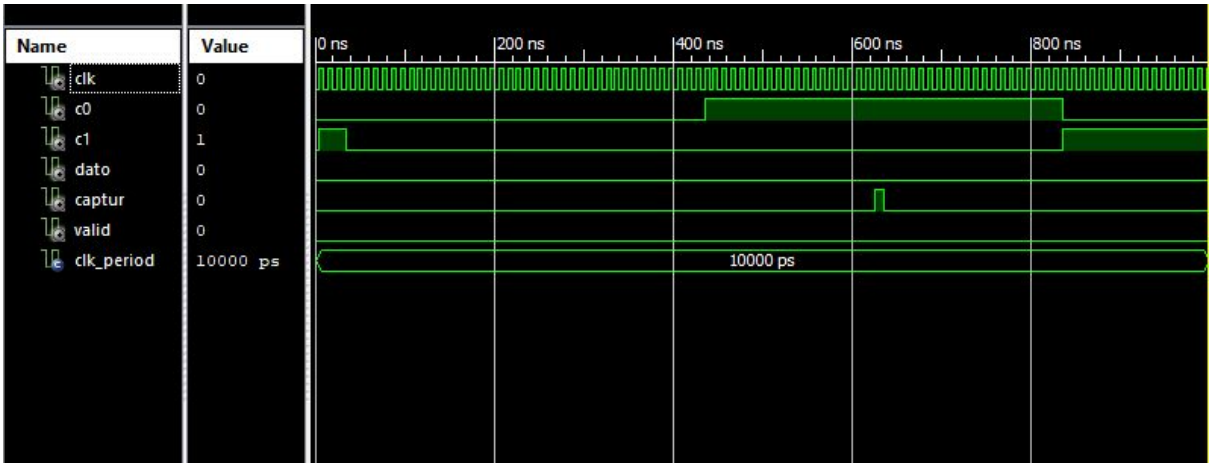
```
       ST <= DATO0;
 end if;
 when DATO1 =>
 cont:= cont+1; -- Se incrementa el contador.
 if (cont=1) then -- Si llega a 1
       cont:=(others=>'0'); -- Poner el contador a 0
       ST <= MUESTREO;
 else
       ST <= DATO1;
 end if;
 when DATOSYNC =>
 cont:= cont+1; -- Se incrementa el contador.
 if (cont=1) then -- Si llega a 1
       cont:=(others=>'0'); -- Poner el contador a 0
       ST <= MUESTREO;
 else
       ST <= DATOSYNC;
 end if;
 end case;
 end if;
 end process; --process del ST

 with ST select
 salidas<=
       "000" when ESP_SYNC,
       "000" when AVAN_ZM,
       "000" when MUESTREO,
       "010" when DATO0,
       "110" when DATO1,
       "001" when DATOSYNC,
       "000" when others;

DATO <= salidas(2);
CAPTUR <= salidas(1);
VALID <= salidas(0);

end a_automata;
```

4.7 tb_automata

```
--------------------------------------------------------------------------------
-- Test Bench para simulación de autómata
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY test_automata IS
END test_automata;

ARCHITECTURE behavior OF test_automata IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT automata
    PORT(
         CLK : IN  std_logic;
         C0 : IN  std_logic;
         C1 : IN  std_logic;
         DATO : OUT  std_logic;
         CAPTUR : OUT  std_logic;
         VALID : OUT  std_logic
        );
    END COMPONENT;


   --Inputs
   signal CLK : std_logic := '0';
   signal C0 : std_logic := '0';
   signal C1 : std_logic := '0';
```

```
        --Outputs
   signal DATO : std_logic;
   signal CAPTUR : std_logic;
   signal VALID : std_logic;

   -- Clock period definitions
   constant CLK_period : time := 10 ns;

BEGIN

        -- Instantiate the Unit Under Test (UUT)
   uut: automata PORT MAP (
          CLK => CLK,
          C0 => C0,
          C1 => C1,
          DATO => DATO,
          CAPTUR => CAPTUR,
          VALID => VALID
        );

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


   -- Stimulus process
   stim_proc: process
   begin
      -- Periodo del reloj = 10 ns.
      wait for 5 ns;

               -- Comenzamos en el estado inicial ESP_SYNC

               C0<='0';         -- Llegada de un 1
               C1<='1';
      wait for 30 ns;  -- Durante 3 ciclos de reloj
                                                    --  El  autómata  debe  quedarse  en
ESP_SYNC

               C0<='0';         -- Llegada de un SYNC
               C1<='0';
      wait for 400 ns;
                                                   -- El autómata debe pasar al estado
                                                    --  AVAN_ZM  y  permanecer  durante  20
ciclos.
                                                   -- Luego debe pasar al estado MUESTREO

                                                   -- Durante 40 ciclos

               C0<='1';         -- Llegada de un 0
               C1<='0';
      wait for 400 ns;
                                                    --  El  autómata  debe  pasar  al  estado
DATO0 1 ciclo de reloj
                                                   -- y volver a MUESTREO
                                                   -- Durante 40 ciclos

               C0<='0';         -- Llegada de un 1
               C1<='1';
      wait for 400 ns;
                                                    --  El  autómata  debe  pasar  al  estado
DATO1  1 ciclo de reloj
                                                   -- y volver a MUESTREO
                                                   -- Durante 40 ciclos
```

```
      C0<='0';          -- Llegada de un SYNC
                C1<='0';
      wait for 400 ns;
                                                    -- El autómata debe pasar al estado
DATOSYNC
                                                   -- y volver a MUESTREO

      wait;
   end process;

END;
```



Cambia a los estados que se indica en el testbench correctamente.

4.8 Registro

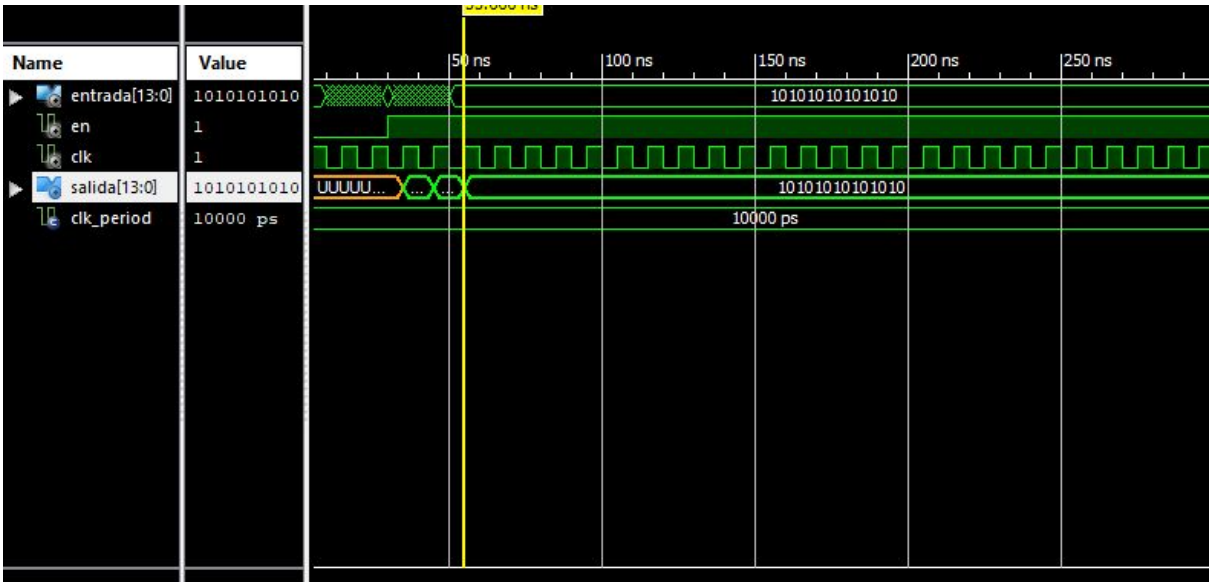
```
-----------------------------------------
--REGISTRO
--
--Este dispositivo se encarga de pasar los datos que
--tiene a su entrada a su salida cuando recibe un
--flanco positivo de la señal VALID del automata,
--es decir, la que controla la informacion que debe
--capturar el registro. Envia la informacion que
--tiene que representar el modulo de visualizacion,
--por tanto su informacion cambiara cuando cambie la hora,
--es decir, cada minuto.
-----------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity registro is
 Port ( ENTRADA : in STD_LOGIC_VECTOR (13 downto 0); -- Entradas
 SALIDA : out STD_LOGIC_VECTOR (13 downto 0); -- Salidas
 EN : in STD_LOGIC; -- Enable
 CLK : in STD_LOGIC); -- Reloj
end registro;
architecture a_registro of registro is

  signal S : std_logic_vector(13 downto 0);
--Señal auxiliar para pasar la señal de la entrada a la
--salida.
begin

       process(CLK)
```

```
            begin

                        if CLK'event and CLK='1' and EN = '1' then

                              S <= ENTRADA;

                        end if;

                        --Con cada flanco activo introducimos la señal de la
                    --entrada en la señal auxiliar S.

      end process;

      SALIDA <= S;  --Introduce la señal auxiliar en la señal de
--para pasarsela al modulo de visualizacion.

end a_registro;
```

4.8 tb_registro

```
--------------------------------------------------------------------------------
-- Test Bench para simulación de registro de captura de datos
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY test_registro IS
END test_registro;

ARCHITECTURE behavior OF test_registro IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT registro
    PORT(
         ENTRADA : IN  std_logic_vector(13 downto 0);
         SALIDA : OUT  std_logic_vector(13 downto 0);
                    EN: in STD_LOGIC;
         CLK : IN  std_logic
        );
    END COMPONENT;


   --Inputs
   signal ENTRADA : std_logic_vector(13 downto 0) := (others => '0');
        signal EN : std_logic := '0';
   signal CLK : std_logic := '0';

        --Outputs
   signal SALIDA : std_logic_vector(13 downto 0);

   -- Clock period definitions
   constant CLK_period : time := 10 ns;

BEGIN

        -- Instantiate the Unit Under Test (UUT)
   uut: registro PORT MAP (
          ENTRADA => ENTRADA,
          SALIDA => SALIDA,
                      EN => EN,
          CLK => CLK
        );

   -- Clock process definitions
   CLK_process :process
   begin
               CLK <= '0';
```

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


   -- El periodo de reloj es de 10 ns

   -- Stimulus process
   stim_proc: process
   begin
      -- Probamos diferentes valores de las entradas.
               -- Dichos valores solamente deben ser capturados con el flanco positivo del
reloj.
               -- En otro caso, la salida debe permanecer en el ultimo valor capturado

      wait for 10 ns;
      ENTRADA<="01010101010101";
      wait for 2 ns;
               ENTRADA<="10101010101010";
      wait for 2 ns;
               ENTRADA<="01010101010101";
      wait for 2 ns;
               ENTRADA<="10101010101010";
      wait for 2 ns;
               ENTRADA<="01010101010101";
      wait for 2 ns;
               ENTRADA<="10101010101010";
      wait for 2 ns;
               ENTRADA<="01010101010101";
      wait for 2 ns;
               ENTRADA<="10101010101010";
      wait for 2 ns;
               ENTRADA<="01010101010101";
      wait for 2 ns;
               ENTRADA<="10101010101010";
      wait for 2 ns;
               EN<='1';
               wait for 2 ns;
               ENTRADA<="01010101010101";
      wait for 2 ns;
               ENTRADA<="10101010101010";
      wait for 2 ns;
               ENTRADA<="01010101010101";
      wait for 2 ns;
               ENTRADA<="10101010101010";
      wait for 2 ns;
               ENTRADA<="01010101010101";
      wait for 2 ns;
               ENTRADA<="10101010101010";
      wait for 2 ns;
               ENTRADA<="01010101010101";
      wait for 2 ns;
               ENTRADA<="10101010101010";
      wait for 2 ns;
               ENTRADA<="01010101010101";
      wait for 2 ns;
               ENTRADA<="10101010101010";
      wait for 2 ns;

      wait;
   end process;

END;
```



La salida no está inicializada pero en cuanto empieza a recibir datos de salida los saca conforme a lo esperado.

4.9 Visualización

```
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); -- 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
               SEG7 : out STD_LOGIC_VECTOR (0 to 6); -- Salida para los displays
               AN : out STD_LOGIC_VECTOR (3 downto 0)); -- Activación individual
end visualizacion;

       architecture a_visualizacion of visualizacion is


----------------------------------------------------
--------------- SEÑALES AUXILIARES ---------------------

signal aux: std_logic_vector(3 downto 0);
signal s: std_logic_vector(1 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,E1,E2,E3,s,aux);
--U2 : decod7s port map (aux,SEG7);
--U3 : refresco port map (CLK,s, AN);
----------------------------------------------------
----------------------------------------------------
U1 : decod7s
      port map (
               D=> aux ,
               S=> SEG7
               );

U2 : MUX4x4
      port map (
                                    E0=>E0,
                                    E1=>E1,
```

```
                                        E2=>E2,
                                        E3=>E3,
                                        Y=>aux,
                                        S=>s
                                        );

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

end a_visualizacion;
```

4.9.1 MUX4x4

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity MUX4x4 is
 Port ( E0 : in STD_LOGIC_VECTOR (3 downto 0); -- entrada E0
 E1 : in STD_LOGIC_VECTOR (3 downto 0); -- entrada E1
 E2 : in STD_LOGIC_VECTOR (3 downto 0); -- entrada E2
 E3 : in STD_LOGIC_VECTOR (3 downto 0); -- entrada E3
 Y : out STD_LOGIC_VECTOR (3 downto 0); -- salida Y
 S : in STD_LOGIC_VECTOR (1 downto 0)); -- entradas de control

end MUX4x4;
architecture a_mux4 of MUX4x4 is
begin
Y <=    E0 when S="00" else -- se selecciona la salida en función de las entradas
                E1 when S="01" else -- de control
                E2 when S="10" else
                E3 when S="11";
end a_mux4;
```

4.9.2 Decod7s

```
----------------------------------------------------
-- DECODIFICADOR DE 7 SEGMENTOS
--
-- Este dispositivo recibe como entrada de datos la
-- salida del multiplexor. Con estos bits de entrada
-- controlamos las salidas, que a su vez están conectadas
-- con los diplay de 7 segmentos, activando el que corresponda.
----------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity decod7s is
 port ( D : in STD_LOGIC_VECTOR (3 downto 0); -- entrada de datos
 S : out STD_LOGIC_VECTOR (0 to 6)); -- salidas 7seg (abcdefg)
end decod7s;
architecture a_decod7s of decod7s is
begin
   with D select S <= -- con las entradas seleccionamos las salidas correspondientes.
                "0000001" when "0000",
                "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 a_decod7s;
```

4.9.3 Refresco

```
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;                       -- entrada de reloj
           AN : out  STD_LOGIC_VECTOR (3 downto 0); -- activación displays
           S : out  STD_LOGIC_VECTOR (1 downto 0)); -- selección en el MUX
end refresco;

architecture a_control of refresco is

signal SS  : STD_LOGIC_VECTOR (1 downto 0):="00";
signal aux : STD_LOGIC_VECTOR (31 downto 0):=(others=>'0'); -- contador 2

begin

process (CLK)
  begin
  if (CLK'event and CLK='1') then
       aux <= aux + 1;
       if (aux = 50000) then
    SS<=SS+1;                        -- genera la secuencia 00,01,10 y 11
       aux <=(others=>'0');
       end if;
 end if;
  end process;

S<=SS;
AN<="0111" when SS="00" else   -- activa cada display en function del valor de SS
    "1011" when SS="01" else
    "1101" when SS="10" else
    "1110" when SS="11";

end a_control;
```

4.10 Fichero de asociaciones

```
# Reloj principal del sistema
NET "CLK" LOC = "M6"; # Señal de reloj del sistema  FREC-->50Mhz

# Conexiones de los DISPLAYS
NET "SEG7<0>" LOC = "L14"; # señal = CA
NET "SEG7<1>" LOC = "H12"; # Señal = CB
NET "SEG7<2>" LOC = "N14"; # Señal = CC
NET "SEG7<3>" LOC = "N11"; # Señal = CD
NET "SEG7<4>" LOC = "P12"; # Señal = CE
NET "SEG7<5>" LOC = "L13"; # Señal = CF
NET "SEG7<6>" LOC = "M12"; # Señal = CG

# Señales de activación de los displays
NET "AN<0>" LOC = "K14"; # Activación del display 0 = AN0
NET "AN<1>" LOC = "M13"; # Activación del display 1 = AN1
NET "AN<2>" LOC = "J12"; # Activación del display 2 = AN2
NET "AN<3>" LOC = "F12"; # Activación del display 3 = AN3

#Entrada externa donde se conecta la señal entrante
NET "SIN" LOC = "B2"; #Entrada terminal 1
```

4.11 Entidad jerárquica TOP: Principal

```
----------------------------------------------
--CIRCUITO PRINCIPAL
--
--Este elemento se encarga de la descripcion
--arquitectural de todo el hardware digital.
--Relaciona todos los elementos del circuito
--para conseguir ver la hora en los displays
--de 4 segmentos.
----------------------------------------------

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; --entrada de reloj
  SIN : in STD_LOGIC; --entrada de datos
  AN: out STD_LOGIC_VECTOR (3 downto 0); --control de displays
  SEG7 : out STD_LOGIC_VECTOR (0 to 6)); --segmentos de displays



end principal;

architecture a_principal of principal is


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

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

```
		component sumador40
				Port ( ENT : in STD_LOGIC_VECTOR (39 downto 0);
				--Datos de la salida en paralelo del registro de desplazamiento
				--de 40.
				SAL : out STD_LOGIC_VECTOR (5 downto 0));
				--Datos de salida de 6 bits.
		end component;

		component comparador
				Port ( P : in STD_LOGIC_VECTOR (5 downto 0); --Señal a comparar
con el umbral.
				Q : in STD_LOGIC_VECTOR (5 downto 0);  --Señal del umbral.
				PGTQ : out STD_LOGIC;  --Señal que indica que la señal es mayor
que el umbral.
				PLEQ : out STD_LOGIC); --Señal que indica que la señal es menor
o igual que el umbral.
		end component;

		component AND_2 --Puerta and.
				Port ( A : in STD_LOGIC; --Entrada A.
				B : in STD_LOGIC; --Entrada B.
				S : out STD_LOGIC); --Salida S.
		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); -- Entradas
				SALIDA : out STD_LOGIC_VECTOR (13 downto 0); -- Salidas
				EN : in STD_LOGIC; -- Enable
				CLK : in STD_LOGIC); -- Reloj
		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 de refresco
				SEG7 : out STD_LOGIC_VECTOR (0 to 6); -- Salida para los
displays
				AN : out STD_LOGIC_VECTOR (3 downto 0)); -- Activación
		--individual
		end component;


	--Señales auxiliares para realizar las interconexiones entre los
	--diferentes componentes del circuito.
	signal CLK_V : STD_LOGIC; --señal auxiliar de la señal de reloj
	--para refresco periodico de los displays.
	signal CLK_M : STD_LOGIC; --señal auxiliar de la señal de reloj
	--muestreada a 40 Hz.
	signal Qq : STD_LOGIC_VECTOR (39 downto 0); --señal auxiliar de
	--la salida en paralelo del registro de desplazamiento de
	--40 bits.
	signal SALc : STD_LOGIC_VECTOR (5 downto 0); --señal auxiliar
	--de la salida del sumador de 40 bits de entrada.
```

```
signal PGTQ1 : STD_LOGIC; --señal auxiliar de la señal del
--primer comparador que indica que la salida del integrador
--que recibe es mayor que el umbral de decision de 34.
signal PLEQ1 : STD_LOGIC; --señal auxiliar de la señal del
--primer comparador que indica que la salida del integrador
--que recibe es menor o igual que el umbral de decision de 34.
signal PGTQ2 : STD_LOGIC; --señal auxiliar de la señal del
--segundo comparador que indica que la salida del integrador
--que recibe es mayor que el umbral de decision de 38.
signal PLEQ2 : STD_LOGIC; --señal auxiliar de la señal del
--segundo comparador que indica que la salida del integrador
--que recibe es menor o igual que el umbral de decision de 38.
signal SALAND : STD_LOGIC; --señal auxiliar de la señal de
--salida de la puerta and.
signal DATOo : STD_LOGIC; --señal auxiliar de la señal DATO que
--sale del automata.
signal CAPTURr : STD_LOGIC; --señal auxiliar de la señal CAPTUR
--que sale del automata.
signal VALIDd : STD_LOGIC; --señal auxiliar de la señal VALID
--que sale del automata.
signal Qdespnormal : STD_LOGIC_VECTOR (13 downto 0); --señal
--auxiliar de la señal de entrada al registro y salida del
--registro de desplazamiento.
signal Qsaldespnormal : STD_LOGIC_VECTOR (13 downto 0);
--señal auxiliar de la señal de salida del registro y de
--de entrada de visualizacion.

 --Constantes del circuito (umbrales de decisión)
 constant UMBRAL1: STD_LOGIC_VECTOR (5 downto 0) := "100010";
 --Umbral de 34
 constant UMBRAL2 : STD_LOGIC_VECTOR (5 downto 0) := "100110";
 --Umbral de 38
```

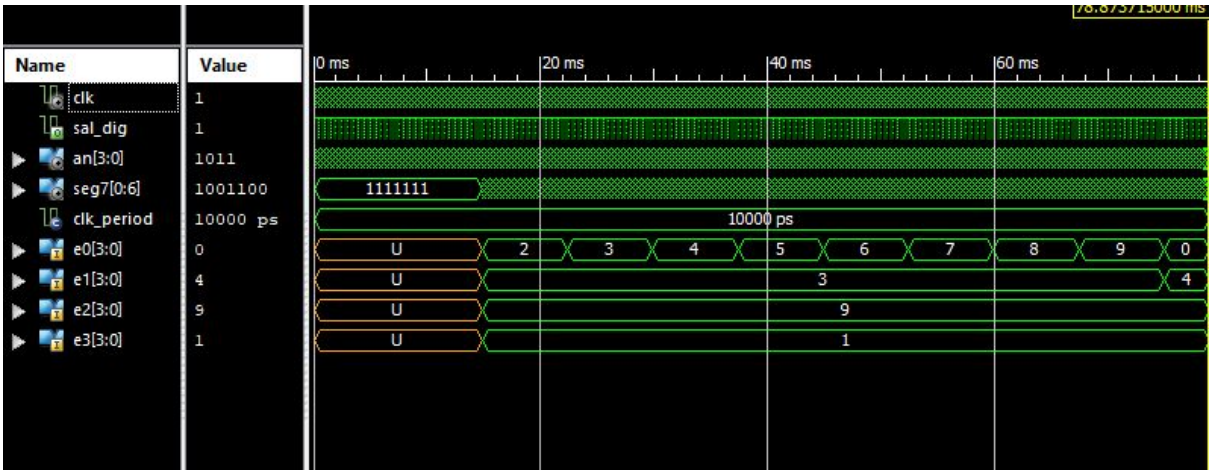
```
       begin

               U1 : visualizacion

     port map (
                          E0 => Qsaldespnormal(3 downto 0),
                                                                  E1(2      downto      0)=>
Qsaldespnormal(6 downto 4),
                                                                  E1(3) => '0',
                          E2=>Qsaldespnormal(10 downto 7),
                                                                  E3(3) => '0',
                                                                  E3(2      downto      0)=>
Qsaldespnormal(13 downto 11),

                                       CLK=>CLK,   -- no clk_v
                                       SEG7=>SEG7,
                                       AN=>AN
               );

 U2 : gen_reloj
       port map (
                CLK=>CLK,
                CLK_M=>CLK_M
               -- CLK_V=>CLK_V
                );
```

```
U3 : registro
      port map (
               ENTRADA=>Qdespnormal,
               SALIDA=>Qsaldespnormal,
               CLK=>CLK,  -- VALIDd, rclk
                                    EN => VALIDd
               );
U4 : reg_desp
      port map (
               SIN=>DATOo,
               CLK=>CLK_M,
                                    EN=>CAPTURr,
                                    Q=>Qdespnormal
                                    );
U5 : automata
      port map (
               CLK=>CLK_M,
               C0=>SALAND,
                                    C1=>PLEQ1,
                                    DATO=>DATOo,
                                    CAPTUR=>CAPTURr,
                                    VALID=>VALIDd
                                    );
U6 : reg_desp40
      port map (
               SIN=>SIN,
               CLK=>CLK_M,
               Q=>Qq
               );
U7 : sumador40
      port map (
               ENT=>Qq,
               SAL=>SALc
               );
U8 : comparador
      port map (
               P=>SALc,
               Q=>UMBRAL1,
               PGTQ=>PGTQ1,
               PLEQ=>PLEQ1
               );
U9 : comparador
      port map (
               P=>SALc,
               Q=>UMBRAL2,
               PGTQ=>PGTQ2,
               PLEQ=>PLEQ2
               );
U10 : AND_2
       port map (
                A=>PGTQ1,
                B=>PLEQ2,
                S=>SALAND
                );



end a_principal;
```

4.11.1 tb_principal

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;

ENTITY tb_aglutina IS
```

```
END tb_aglutina;

ARCHITECTURE behavior OF tb_aglutina IS

    -- Component Declaration for the Unit Under Test (UUT)

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


   --Inputs
   signal CLK : std_logic := '0';

        --Outputs
   signal AN : std_logic_vector(3 downto 0);
   signal SEG7 : std_logic_vector(0 to 6);

   -- Clock period definitions
   constant CLK_period : time := 10 ns;

BEGIN

        -- Instantiate the Unit Under Test (UUT)
   uut: SuperPrincipal PORT MAP (
          CLK => CLK,
          AN => AN,
          SEG7 => SEG7
        );

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


   -- Stimulus process
   stim_proc: process
   begin
      -- hold reset state for 100 ns.
      wait for 100 ns;

      wait for CLK_period*10;

      -- insert stimulus here

      wait;
   end process;

END;
```



Se ha reducido la frecuencia de la salida del generador de reloj para evitar tiempos de espera excesivos en la simulación, así mismo se ha implementando el gen_signal de moodle para esta simulación.

ANEXO I : Entrega 1

ENTREGA 1: FILTRO PASO BANDA

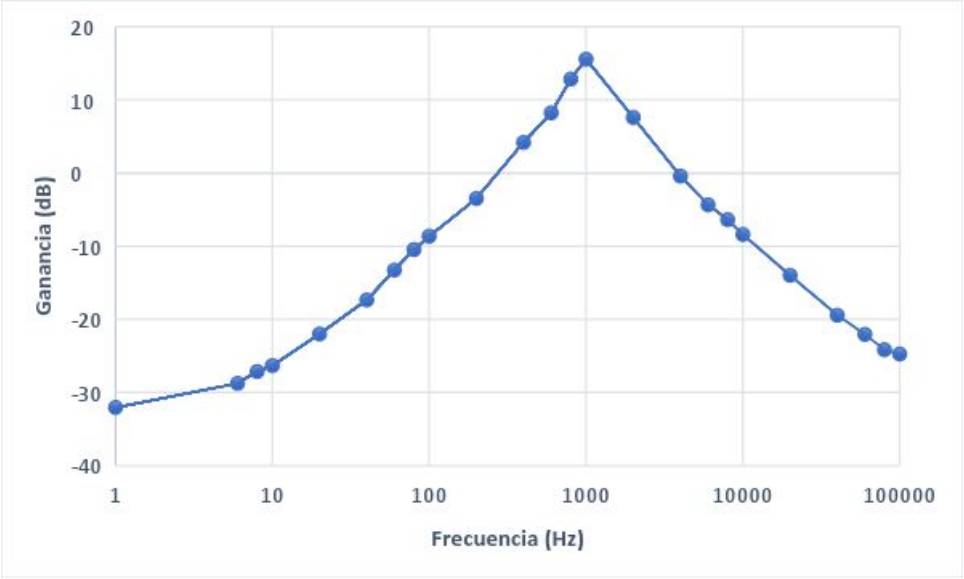
Frecuencia (Hz)	Entrada (vpp)	Salida (vpp)	Ganancia	Ganancia (dB)	Δt (s)	Fase °
1	1	0,025	0,025	-32,042		-90
6	1,04	0,038	0,036	-28,745		-90
8	1	0,044	0,044	-27,131	0,0312	-90
10	1,056	0,051	0,048	-26,322	0,0248	-90
20	1,12	0,089	0,079	-21,996	0,0124	-90
40	1,128	0,153	0,136	-17,352	0,0062	-90
60	1,04	0,226	0,217	-13,258	0,0042	-90,72
80	1	0,3	0,3	-10,457	0,00336	-96,768
100	1	0,37	0,37	-8,636	0,0027	-97,2
200	1,104	0,74	0,670	-3,474	0,00136	-97,92
400	1	1,62	1,62	4,190	0,00072	-103,68
600	1,108	2,86	2,581	8,236	0,000548	-118,368
800	1,056	4,64	4,394	12,857	0,00048	-138,24
1000	1	6	6	15,563	0,00048	-172,8
2000	1	2,4	2,4	7,604	0,000342	-246,24
4000	1,04	0,992	0,954	-0,410	0,00018	-259,2
6000	1,056	0,64	0,606	-4,349	0,000122	-263,52
8000	1	0,48	0,48	-6,375	0,000092	-264,96
10000	1	0,38	0,38	-8,404	0,000074	-266,4
20000	1	0,2	0,2	-13,979	0,0000371	-267,12
40000	1,012	0,108	0,107	-19,435	0,0000187	-269,28

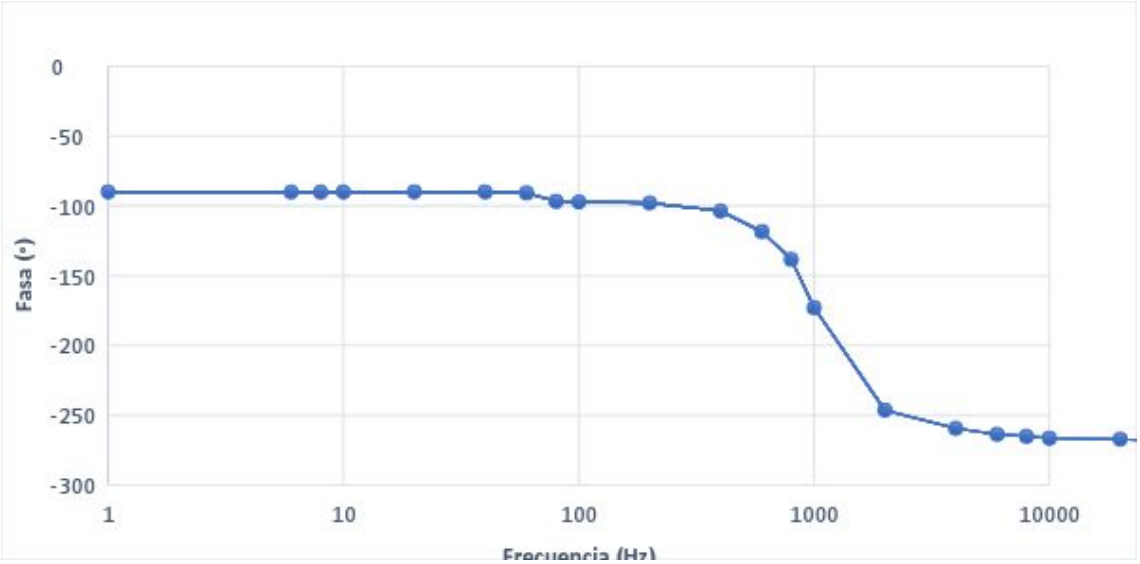
60000	1,064	0,084	0,0789	-22,053	0,0000125	-270
80000	1	0,062	0,062	-24,152	0,0000094	-270,72
100000	1	0,058	0,058	-24,731	0,0000075	-270

Cálculo de ceros y polos:

$$H(jw) = \frac{-jw\frac{1}{R_1 C}}{-w^2 + jw\frac{2}{R_2 C} + \frac{1}{R_1 R_2 C^2}}$$

Para $R_1 = 1K\Omega$, $R_2 = 12\ K\Omega$ y $C = 47\ nF$ calculamos que hay un cero en 0 Hz y un polo doble en 913 Hz.




0
-50
-100
-150
-200
-250
-300
Fasa (°)
1
10
100
1000
10000

ANEXO II : Entrega 2

...

ENTREGA 2: FILTRO PASO BAJO

Frecuencia (Hz)	Entrada (vpp)	Salida (vpp)	Ganancia	Ganancia (dB)	Δt (s)	Fase °
1	1,02	1,04	1,019607843	0,388361717	0	·
4	0,01	1,2	120	95,74983486	0	·
8	1	0,94	0,94	-1,237508074	0,0028	·
10	1	0,976	0,976	-0,485853851	0,0024	6,912
20	1,012	0,968	0,956521739	-0,889035251	0,002	7,2
40	0,997	0,92	0,922768305	-1,607541998	0,002	14,4
60	1	0,864	0,864	-2,923650204	0,00208	29,9759808
80	1,001	0,792	0,791208791	-4,68386775	0,00184	39,664707
100	0,998	0,72	0,721442886	-6,530041286	0,0018	51,84
120	0,996	0,624	0,626506024	-9,351937784	0,00164	60,244898
140	1,016	0,536	0,527559055	-12,78988934	0,0016	69,397590 4
160	1	0,496	0,496	-14,02358705	0,00176	88,739495 8
180	1,016	0,432	0,42519685	-17,1040608	0,00164	94,615384 6
200	1	0,392	0,392	-18,72986878	0,00144	93,237410 1
250	1,001	0,28	0,27972028	-25,47930352	0,0013	0,117
300	1	0,22	0,22	-30,28255465	0,00098	105,31343 3
400	1,007	0,13	0,129096326	-40,94392885	0,00101	150,55900 6
600	1	0,07	0,07	-53,18520074	0,00065	140,11976
800	1	0,0392	0,0392	-64,78157064	0,00053	227,95698 9
1000	1,014	0,0276	0,027218935	-72,07684823	0,000448	520,25806 5

2000	1,003	0,0106	0,010568295	-90,99793574	0,000224	311,351351
4000	1	0,0045	0,0045	-108,0735576	·	·
6000	1	0,0034	0,0034	-113,6795969	·	·

Cálculo polos:

Conforme al apartado 4.3.3, la función del filtro paso bajo viene definida como:

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

Los valores utilizados han sido:

R = 1kΩ C = 1μF G=1

Calculamos los polos igualando a cero el denominador:

$$-w^2 + jw\frac{3-G}{RC} + \frac{1}{R^2C^2} = 0 \Rightarrow s^2 + s\frac{3-G}{RC} + \frac{1}{R^2C^2} = 0$$

Por tanto, despejando s:

$$s = \frac{\frac{G-3}{RC} \pm \sqrt{\left(\frac{3-G}{RC}\right)^2 - 4\frac{1}{R^2C^2}}}{2} = \frac{-2000 \pm \sqrt{(2000)^2 - 4*1000000}}{2} =$$

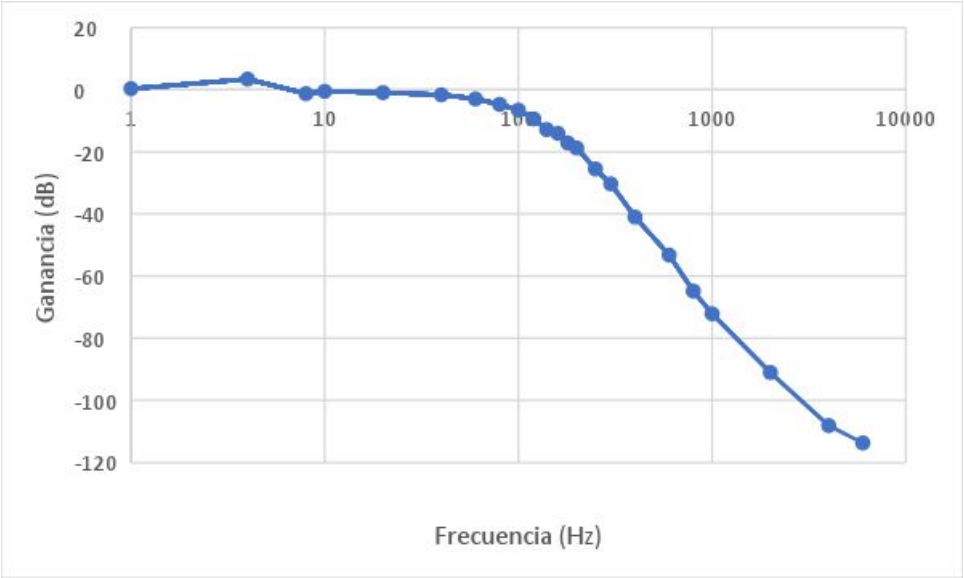
$$= \frac{-2000 \pm 0}{2} = -1000$$

Haciendo la conversión obtenemos:

$$s = jw = j * 2\pi f = -1000 \Rightarrow f = -\frac{1000}{j2\pi}$$

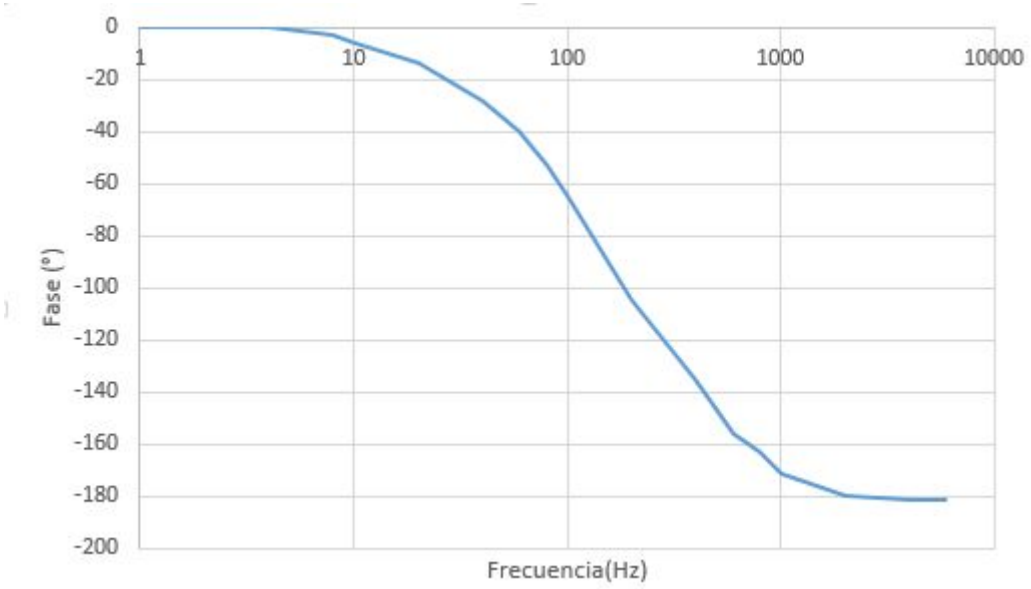
$$f_{p1,2} = 159,15$$

El módulo del polo doble será: $|fp| = 159,15$



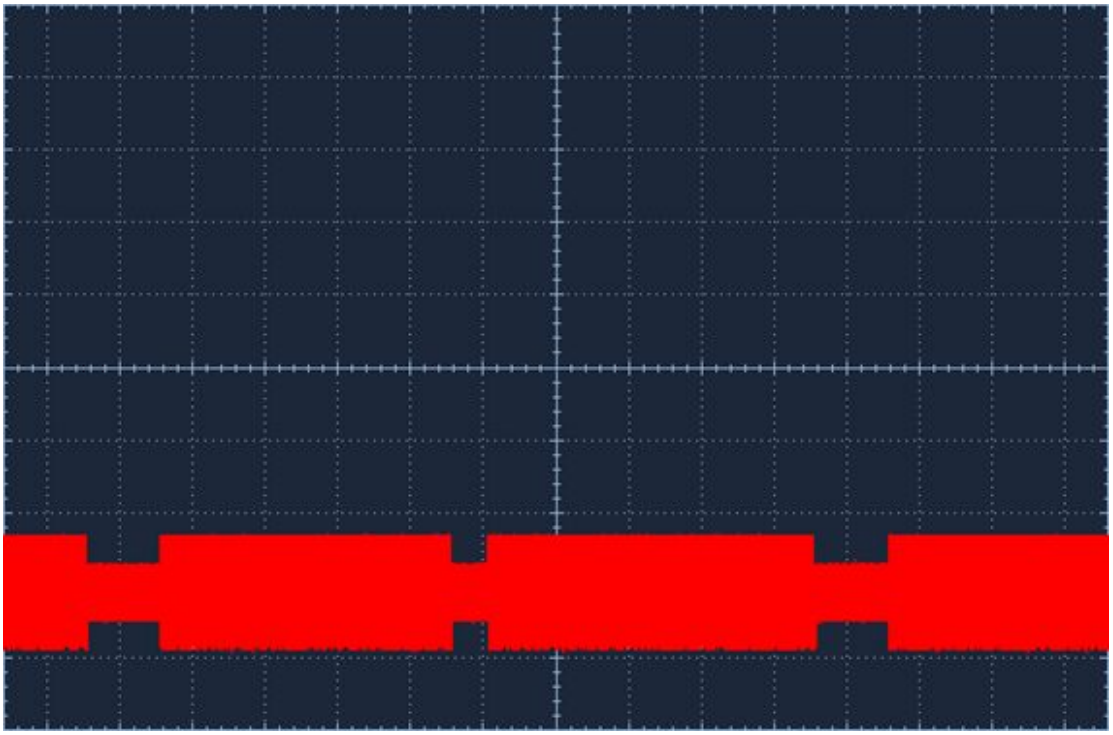
Puesto que habíamos medido mal las fases esta es la nueva tabla de medidas y la nueva gráfica

T	Δt	Fase	Frecuencia
-	-	0	1
0,25	0	0	4
0,125	0,00105	-3,024	8
0,1	0,00168	-6,048	10
0,05	0,00195	-14,04	20
0,025	0,00196	-28,224	40
0,01666667	0,00186	-40,176	60
0,0125	0,00184	-52,992	80
0,01	0,00179	-64,44	100
0,005	0,00145	-104,4	200
0,0025	0,00094	-135,36	400
0,00166667	0,000721	-155,736	600
0,00125	0,000567	-163,296	800
0,001	0,000476	-171,36	1000
0,0005	0,00025	-180	2000
0,00025	0,000126	-181,44	4000
0,00016667	0,000084	-181,44	6000

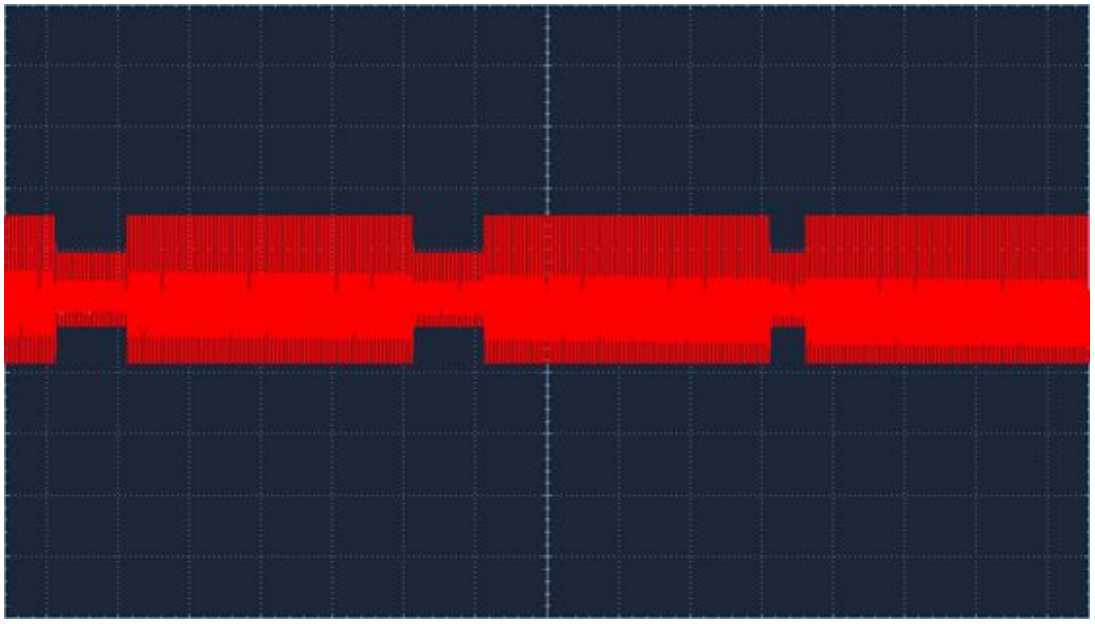

0
-20
-40
-60
-80
-100
-120
-140
-160
-180
-200
Fase (°)
1
10
100
1000
10000
Frecuencia(Hz)

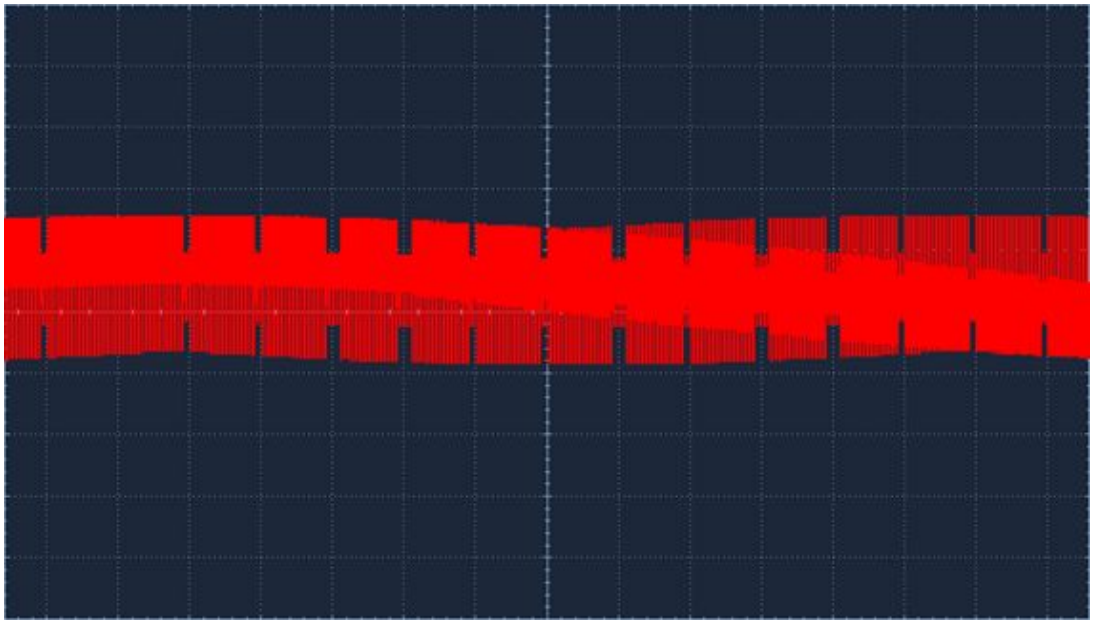
ANEXO III : Entrega 3

1. Señal a la salida del dispositivo MP3. Realice una captura donde se puedan apreciar al menos 2 bits de información. (un cero y un uno)

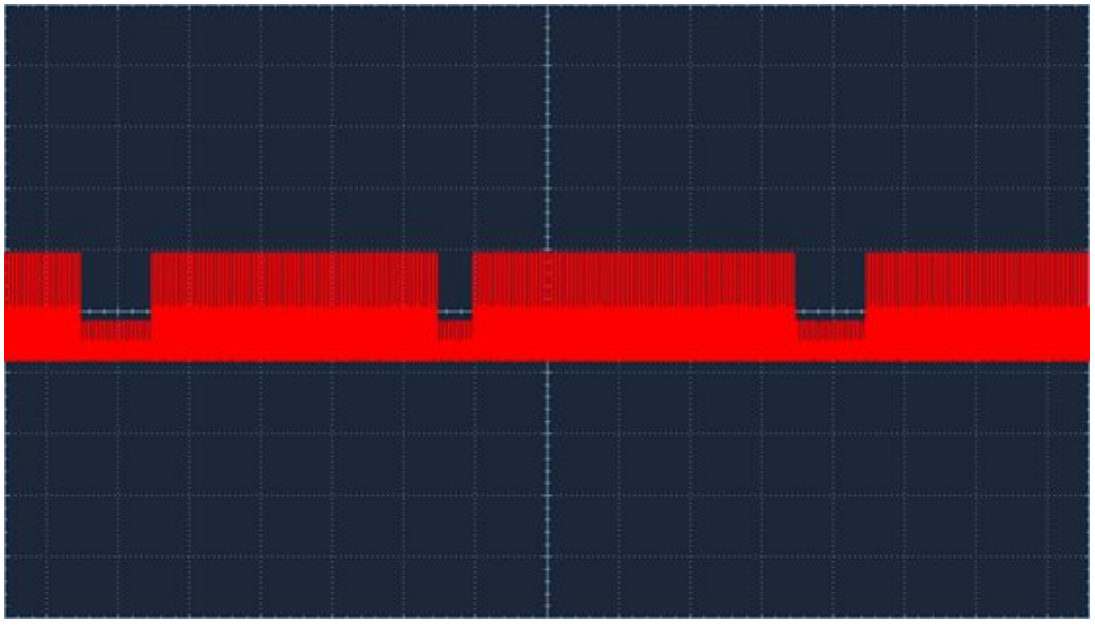


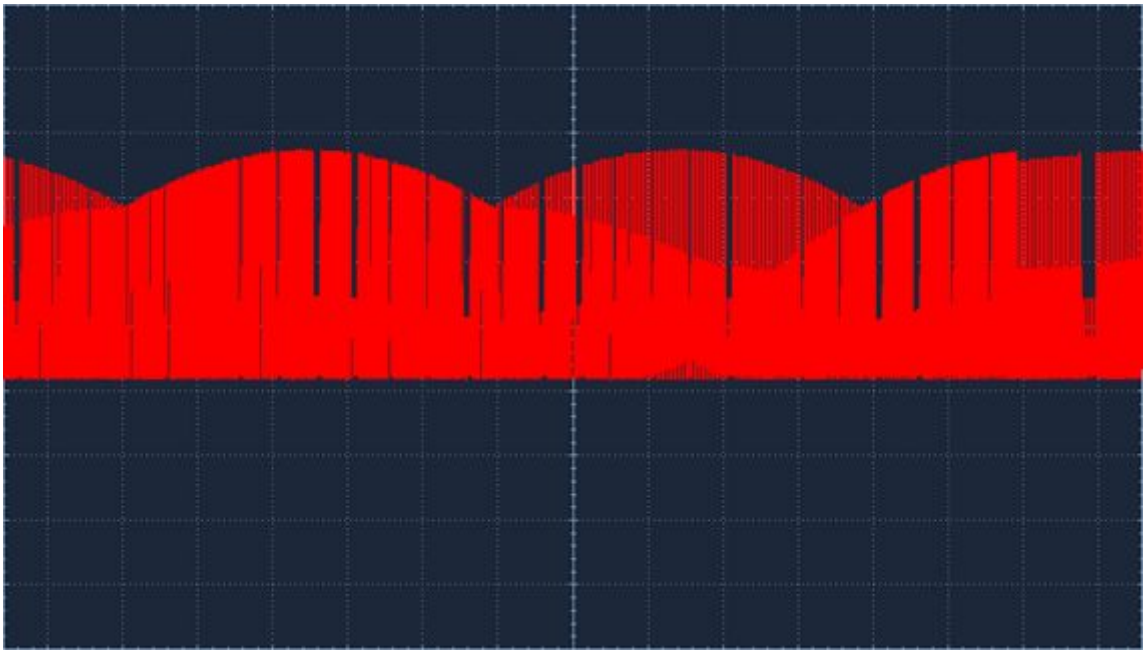
2. Señal a la salida del filtro paso banda. Realice dos capturas, una donde se pueda apreciar al menos 2 bits de información (un cero y un uno) y otra donde se pueda apreciar una trama completa (14 bits de información más el bit de sincronismo).



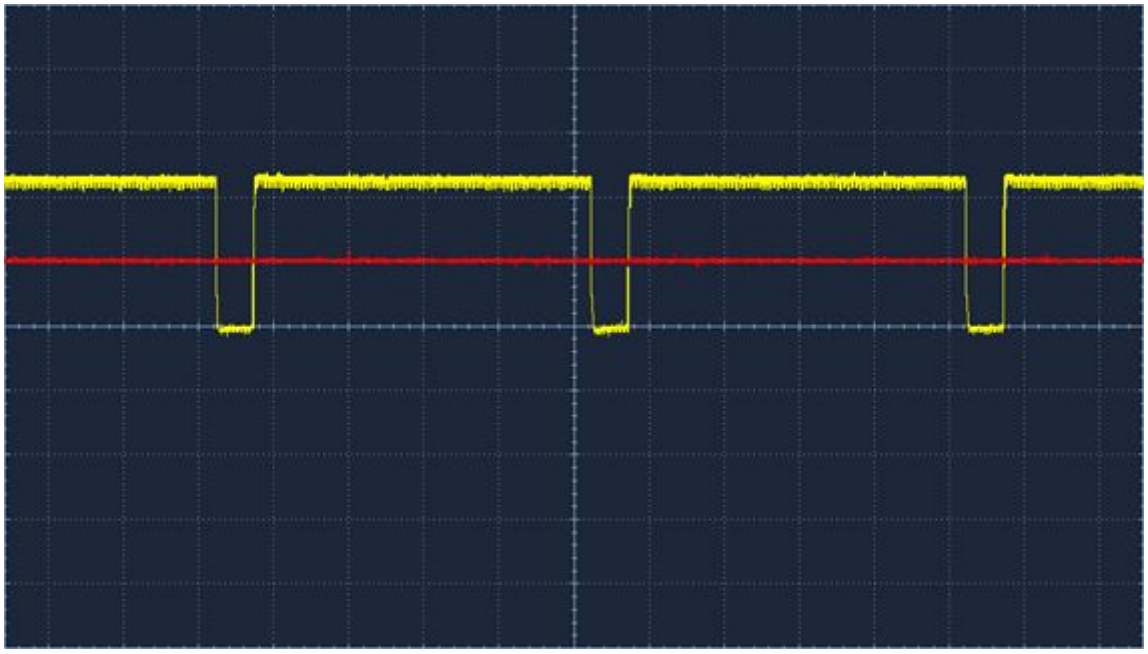


3. Señal a la salida del rectificador. Realice dos capturas, una donde se pueda apreciar al menos 2 bits de información (un cero y un uno) y otra donde se pueda apreciar una trama completa (14 bits de información más el bit de sincronismo).





4. Señal a la salida del filtro paso bajo. Realice una captura donde se puedan apreciar al menos 2 bits de información (un cero y un uno) y el nivel de referencia del comparador, utilizando los dos canales del osciloscopio.



5. Salida de 40 Hz obtenida mediante división de frecuencia de la señal de reloj de la FPGA (señal CLK_M). Capture la señal de tal manera que pueda apreciarse su frecuencia en la imagen.