Universidad Politécnica de Madrid
ETSI de Telecomunicación
Departamento de Ingeniería Electrónica

Circuitos Electrónicos (CELT)

Curso 2018-2019

Memoria final

	Apellidos	Nombre
Alumno 1	Arias Cuadrado	David
Alumno 2	Gómez Rodríguez	Carlos

Código de pareja	

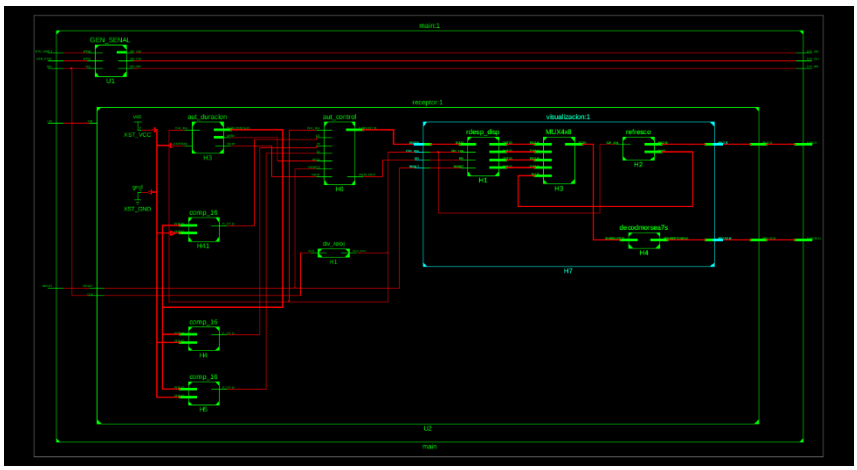
(El contenido de esta memoria debe ajustarse exactamente a los bloques analógicos y digitales de la práctica que se presenta. ***IMPORTANTE: Los comentarios en cursiva se han de eliminar en la versión final, incluyendo este****)*

1. ESQUEMA COMPLETO DEL CIRCUITO ANALÓGICO

Incluya en esta página un diagrama completo del circuito analógico con los valores de todos los componentes. No copie el diagrama de otra fuente, como pueda ser el enunciado del proyecto.

2. ESQUEMA COMPLETO DEL CIRCUITO DIGITAL

Incluya en esta página un diagrama completo del circuito digital. Represente cada bloque con sus entradas y salidas, y las interconexiones entre ellos. No copie el diagrama de otra fuente, como pueda ser el enunciado del proyecto.



3. DISEÑO DETALLADO DEL CIRCUITO ANALÓGICO

Crear un subapartado para cada etapa del circuito. Por cada etapa debe justificar el diseño de los componentes y los criterios de diseño empleados.

3.1 Etapa 1

Describa aquí la primera etapa. Incluya esquemáticos propios, no copias de otras fuentes.

....

3.n Etapa n

Describa aquí la etapa n. Incluya esquemáticos propios, no copias de otras fuentes.

4. DISEÑO DETALLADO DEL CIRCUITO DIGITAL

4.1 Main

```
----------------------------------------------------------------------------------
-- MAIN
--
-- Modulo principal que contiene como modulos
-- secundarios el Generador de senal y el receptor.
--
----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

use IEEE.STD_LOGIC_UNSIGNED.ALL;


entity main is
Port( CLK : in STD_LOGIC;
              BTN_START : in STD_LOGIC; -- Acciona la reproduccin del mensaje automtico.
              BTN_STOP : in STD_LOGIC; -- Para la reproduccin del mensaje automtico
              RESET0 : in STD_LOGIC; --Resetea los displays
              PULSADOR : in STD_LOGIC;--Reactiva la recepcion de un mensaje
              SELECTOR: in STD_LOGIC; --Mostrar mensaje o numero de palabras
              SPI_CLK : out STD_LOGIC;
          SPI_DIN : out STD_LOGIC;
              SPI_CS1 : out STD_LOGIC;
              LIN : in STD_LOGIC; -- Lnea de entrada de datos
              AN : out STD_LOGIC_VECTOR (3 downto 0); -- Activacion individual displays
              SEG7 : out STD_LOGIC_VECTOR (0 to 6); -- Salida para los displays
              LED : out STD_LOGIC); --Indica el fin del mensaje
end main;

architecture a_main of main is

component GEN_SENAL is

 Port( CLK : in  STD_LOGIC;

               BTN0 : in  STD_LOGIC;-- acciona la reproduccin del mensaje automtico.

               BTN1 : in  STD_LOGIC;-- para la reproduccin del mensaje automtico.

       SPI_CLK : out STD_LOGIC;

       SPI_DIN : out STD_LOGIC;     -- DATA_IN

                SPI_CS1 : out STD_LOGIC);    -- CHIP_SELECT



end component;


component receptor
 Port( CLK : in STD_LOGIC; -- reloj de la FPGA
               LIN : in STD_LOGIC; -- Lnea de entrada de datos
               RESET0 : in STD_LOGIC;
               PULSADOR: in STD_LOGIC;
               SELECTOR: in STD_LOGIC;
               AN : out STD_LOGIC_VECTOR (3 downto 0); -- Activacin individual
               SEG7 : out STD_LOGIC_VECTOR (0 to 6); -- Salida para los displays
               LED :out STD_LOGIC);
end component;

begin
U1 : gen_senal port map
 (CLK => CLK,
  BTN0 => BTN_START,
  BTN1 => BTN_STOP,
  SPI_CLK => SPI_CLK,
  SPI_DIN => SPI_DIN,
  SPI_CS1 => SPI_CS1);
```

```
U2 : receptor port map
 (CLK => CLK,
  LIN => LIN,
  RESET0 => RESET0,
  PULSADOR => PULSADOR,
  SELECTOR=> SELECTOR,
  AN => AN,
  SEG7 => SEG7,
  LED => LED);
end a_main;
```

4.2 Receptor

```
----------------------------------------------------------------------------------
-- RECEPTOR
-- Contiene diferentes modulos
-- que transforman la seal binaria
-- en simbolos morse y un modulo de
-- visualizacion que transforma cada simblolo
-- en un digito del display
--
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_arith.ALL;

use IEEE.STD_LOGIC_unsigned.ALL;


entity receptor is
Port ( CLK : in  STD_LOGIC; --Reloj con periodo de 20 ns
       LIN : in  STD_LOGIC; --Entrada de Datos
                RESET0 : in STD_LOGIC; --Reset asincrono
                PULSADOR : in STD_LOGIC; -- Pone de nuevo el display en funcionamiento
                SELECTOR: in STD_LOGIC; --Mostrar mensaje o numero de palabras
       AN : out  STD_LOGIC_VECTOR (3 downto 0); -- Activacion individual displays
       SEG7 : out  STD_LOGIC_VECTOR (0 to 6); -- Salida para los displays
                LED : out STD_LOGIC); --Indica el fin del mensaje
end receptor;

architecture Behavioral of receptor is
constant UMBRAL0 : STD_LOGIC_VECTOR (15 downto 0) := "0000000011001000"; -- 200 umbral ceros
constant UMBRAL1 : STD_LOGIC_VECTOR (15 downto 0) := "0000000011001000"; -- 200 umbral unos
constant UMBRALS : STD_LOGIC_VECTOR (15 downto 0) := "0000000111110100"; -- 500 umbral ceros
(SEPARADOR)
       SIGNAL CLK_1ms : STD_LOGIC;
       SIGNAL LIN2 : STD_LOGIC;
       SIGNAL VALID : STD_LOGIC;
       SIGNAL DATO : STD_LOGIC;
       SIGNAL DURACION : STD_LOGIC_VECTOR (15 downto 0);
       SIGNAL C0 : STD_LOGIC;
       SIGNAL C1 : STD_LOGIC;
       SIGNAL CS : STD_LOGIC;
       SIGNAL CODIGO : STD_LOGIC_VECTOR (7 downto 0);
       SIGNAL VALID_DISP : STD_LOGIC;
       SIGNAL FINAL : STD_LOGIC;
       SIGNAL CONTADOR : STD_LOGIC_VECTOR (8 downto 0);
       SIGNAL CENTENAS : STD_LOGIC_VECTOR (7 downto 0);
       SIGNAL DECENAS : STD_LOGIC_VECTOR (7 downto 0);
       SIGNAL UNIDADES : STD_LOGIC_VECTOR (7 downto 0);

component div_reloj is
Port ( CLK : in  STD_LOGIC;
       CLK_1ms : out  STD_LOGIC);
end component;

component detector_flanco is
Port ( CLK_1ms : in STD_LOGIC; -- reloj
                LIN : in STD_LOGIC; -- Lnea de datos
                VALOR : out STD_LOGIC); -- Valor detectado en el flanco
end component;

component aut_duracion is
```

```
Port ( CLK_1ms : in  STD_LOGIC;
       ENTRADA : in  STD_LOGIC;
       VALID : out  STD_LOGIC;
       DATO : out  STD_LOGIC;
       DURACION : out  STD_LOGIC_VECTOR (15 downto 0));
end component;

component comp_16 is
Port ( P : in  STD_LOGIC_VECTOR (15 downto 0);
       Q : in  STD_LOGIC_VECTOR (15 downto 0);
       P_GT_Q : out  STD_LOGIC);
end component;

component aut_control is
Port( CLK_1ms : in STD_LOGIC; -- reloj
              VALID : in STD_LOGIC; -- entrada de dato vlido
              DATO : in STD_LOGIC; -- dato (0 o 1)
              C0 : in STD_LOGIC; -- resultado comparador de ceros
              C1 : in STD_LOGIC; -- resultado comparador de unos
              CS : in STD_LOGIC; -- resultado comparador de espacios
              RESET0 : in STD_LOGIC;
              PULSADOR : in STD_LOGIc;
              FINAL : in STD_LOGIc;
              CODIGO : out STD_LOGIC_VECTOR (7 downto 0); -- cdigo morse obtenido
              VALID_DISP : out STD_LOGIC;-- validacin del display
              LED : out STD_LOGIC;
              CONTADOR: out STD_LOGIC_VECTOR(8 downto 0));
end component;

component visualizacion is
 Port( E0 : in STD_LOGIC_VECTOR (7 downto 0); -- Entrada sig. carcter
               EN : in STD_LOGIC; -- Activacin para desplazamiento
               CLK_1ms : in STD_LOGIC; -- Entrada de reloj
               RESET0 : in STD_LOGIC;
               SELECTOR: in STD_LOGIC;
               CENTENAS: in STD_LOGIC_VECTOR(7 downto 0);
               DECENAS: in STD_LOGIC_VECTOR(7 downto 0);
               UNIDADES: in STD_LOGIC_VECTOR(7 downto 0);
               SEG7 : out STD_LOGIC_VECTOR (0 to 6); -- Segmentos displays
               AN : out STD_LOGIC_VECTOR (3 downto 0)); -- Activacin displays
end component;

component Deteccionfin is
 Port( CLK_1ms : in STD_LOGIC;
               CODIGO : in STD_LOGIC_VECTOR (7 downto 0); -- cdigo morse obtenido
               VALID_DISP : in STD_LOGIC;
               FINAL : out STD_LOGIC );
end component;

component bintobcd is
 Port( num_bin: in  STD_LOGIC_VECTOR(8 downto 0);
               centenas: out STD_LOGIC_VECTOR(7 downto 0);
               decenas: out STD_LOGIC_VECTOR(7 downto 0);
               unidades: out STD_LOGIC_VECTOR(7 downto 0));
end component;

begin
H1: div_reloj port map (CLK,CLK_1ms);
H2: detector_flanco port map (CLK_1ms, LIN,LIN2);
H3: aut_duracion port map (CLK_1ms, LIN2, VALID, DATO, DURACION);
H4: comp_16 port map (DURACION,UMBRAL0,C0);
H5: comp_16 port map (DURACION,UMBRAL1,C1);
HS: comp_16 port map (DURACION,UMBRALS,CS);
H6: aut_control port map
(CLK_1ms,VALID,DATO,C0,C1,CS,RESET0,PULSADOR,FINAL,CODIGO,VALID_DISP,LED,CONTADOR);
M4: bintobcd port map (CONTADOR,CENTENAS,DECENAS,UNIDADES);
HF: Deteccionfin port map (CLK_1ms, CODIGO, VALID_DISP,FINAL);
H7: visualizacion port
map(CODIGO,VALID_DISP,CLK_1ms,RESET0,SELECTOR,CENTENAS,DECENAS,UNIDADES,SEG7,AN);
end Behavioral;
```

4.3 Divisor del Reloj

```
-- DIVISOR DEL RELOJ
--
-- Este modulo tiene como senal de entrada un reloj
-- CLK con un periodo de 20 ns, negando el valor de la salida
-- cada 1ms/(20ns*2) conseguimos crear un CLK que cambia
-- (de 0 a 1 o de 1 a 0) cada 0.5 ms y por tanto es un
-- reloj de perido un 1 ms.
----------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;

use IEEE.STD_LOGIC_unsigned.ALL;
entity div_reloj is
 Port ( CLK : in STD_LOGIC; -- Entrada reloj de la FPGA 50 MHz (Periodo de 20 ns)
                CLK_1ms : out STD_LOGIC); -- Salida reloj a 1 KHz (Periodo de 1 ms)
end div_reloj;

architecture a_div_reloj of div_reloj is
signal contador : STD_LOGIC_VECTOR (15 downto 0):=”0000000000000000”;
signal flag : STD_LOGIC:=’0’;

begin
process(CLK)
 begin
 if (CLK'event and CLK='1') then
       contador<=contador+1;
       if (contador=25000) then --Cuando la cuenta llegue  a medio periodo
              contador<=(others=>'0');
              flag<=not flag; -- Se niega el valor anterior de la salida
       end if;
 end if;
end process;

CLK_1ms<=flag;
end a_div_reloj;
```

4.4 Detector de Flancos

```
--------------------------------------------------------------------------------
-- DETECTOR DE FLANCOS
--
-- Corrige perturbaciones de la parte analogica (LIN)
-- debidas al ruido, el modulo evita y corrige la falsa
-- alarma (Tener '1' cuando el mensaje es '0') y pequeñas
-- variaciones de la señal que son errores.
-- La variable suma tiene 20 muestras de los últmos
-- 20 ms de entrada.
--------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;

entity detector_flanco is
 Port ( CLK_1ms : in STD_LOGIC; -- Reloj
                LIN : in STD_LOGIC; -- Linea de datos
                VALOR : out STD_LOGIC); -- Valor detectado en el flanco
end detector_flanco;

architecture a_detector_flanco of detector_flanco is
constant UMBRAL0 : STD_LOGIC_VECTOR (7 downto 0) := "00000101"; -- 5 umbral para el 0
constant UMBRAL1 : STD_LOGIC_VECTOR (7 downto 0) := "00001111"; -- 15 umbral para el 1
signal reg_desp : STD_LOGIC_VECTOR (19 downto 0):="00000000000000000000"; -- Registro de los
ultimos 20 ciclos de reloj
signal suma : STD_LOGIC_VECTOR (7 downto 0) :="00000000"; -- Suma de todos los '1' de reg_desp
signal s_valor : STD_LOGIC :='0';

begin
process (CLK_1ms)
 begin
 if (CLK_1ms'event and CLK_1ms='1') then
       suma <= suma +LIN -reg_desp(19); --Resta el ultimo valor en el registro
```

```
                                                                                --y
suma el futuro primer valor (LIN)

       reg_desp (19 downto 1)<=reg_desp(18 downto 0); --Desplaza todos los bits a la
izquierda
       reg_desp (0) <= LIN;                                                      --
Anade el nuevo valor al registro

   if(s_valor='1' and suma<UMBRAL0) then  -- Si la salida es '1' y la suma de '1'
              s_valor<='0';                                              -- de las
ultimas 20 muestras no supera

              -- el umbral de comparacion, la salida pasa a ser '0'

       elsif(s_valor='0' and suma>UMBRAL1) then -- Si la salida es '0' y la suma de '1' de

              s_valor<='1';                        -- las ultimas 20 muestras supera el
umbral
       end if;                                                                   --
de comparacion, la salida pasa a ser '1'
 end if;
end process;

VALOR<=s_valor;
end a_detector_flanco;
```

4.5 Autómata de Duración

```
--------------------------------------------------------------------------------
-- AUTÓMATA DURACIÓN
--
-- Este automata consigue mediante la entrada ('0' o '1') y el tiempo que esta dura
-- distinguir entre las cuatro posibles señales (RAYA, PUNTO, PAUSA O ESPACIO).
-- Si DATO vale 1 se trata de PUNTO o RAYA y si vale 0 se trata de PAUSA o ESPACIO
-- Si DURACIÓN es mayor que un umbral estarmos mandando RAYA O ESPACIO
-- y si es menor estaremos mandando PAUSA O PUNTO.
-- Las salidas solo se usar cuando se valide: VALID='1'.
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;

entity aut_duracion is
    Port (CLK_1ms : in  STD_LOGIC; -- Reloj con periodo 1 ms
          ENTRADA : in  STD_LOGIC; -- 1 cuando SÍMBOLO; 0 cuando PAUSA o ESPACIO
          VALID : out  STD_LOGIC;  -- Valida el uso de DATO y DURACION
          DATO : out  STD_LOGIC;     -- 1 cuando SÍMBOLO; 0 cuando PAUSA o ESPACIO
          DURACION : out  STD_LOGIC_VECTOR (15 downto 0)); --Tiempo de duracion en ms del dato
end aut_duracion;

architecture a_aut_duracion of aut_duracion is
type STATE_TYPE is (CERO,ALM_CERO,VALID_CERO,UNO,ALM_UNO,VALID_UNO,VALID_FIN);
signal ST : STATE_TYPE := CERO;
signal cont : STD_LOGIC_VECTOR (15 downto 0):="0000000000000000";
signal reg : STD_LOGIC_VECTOR (15 downto 0) :="0000000000000000";
begin
process (CLK_1ms)
 begin
 if (CLK_1ms'event and CLK_1ms='1') then
       case ST is
       when CERO =>
              cont<=cont+1;
              if (cont>600) then
                     ST<=VALID_FIN;
              elsif (ENTRADA='0') then
                     ST<=CERO;
              else
                     ST<=ALM_CERO;
              end if;

       when ALM_CERO =>
              reg<=cont;
              cont<=(others=>'0');
              ST<= VALID_CERO;
```

```
        when VALID_CERO =>
                ST<= UNO;

        when UNO=>
                cont<=cont+1;
                if (ENTRADA='0') then
                        ST<=ALM_UNO;
                else
                        ST<=UNO;
                end if;

        when ALM_UNO =>
                reg<=cont;
                cont<=(others=>'0');
                ST<= VALID_UNO;

        when VALID_UNO =>
                ST<= CERO;

        when VALID_FIN =>
                reg<=cont;
                cont<=(others=>'0');
                ST<= CERO;
        end case;
  end if;
 end process;

 -- PARTE COMBINACIONAL
 VALID<='1' when (ST=VALID_CERO or ST=VALID_UNO or ST=VALID_FIN) else '0';
 DATO <='1' when (ST=UNO or ST=VALID_UNO or ST=ALM_UNO) else '0';
 DURACION<= reg;

end a_aut_duracion;
```

4.6 Comparador

```
--------------------------------------------------------------------------------
-- COMPARADOR
--
-- Si P es mayor que un umbral la salida es '1',
-- en caso contrario la salida es '0'.
-- Sirve para comparar la duracion de los simbolos
-- y distinguir entre RAYA Y PUNTO y entre
-- ESPACIO Y PAUSA (Y PALABRA).
--------------------------------------------------------------------------------

library IEEE;
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;

use IEEE.STD_LOGIC_unsigned.ALL;

entity comp_16 is
  Port ( P : in  STD_LOGIC_VECTOR (15 downto 0);
         Q : in  STD_LOGIC_VECTOR (15 downto 0);
         P_GT_Q : out  STD_LOGIC);
end comp_16;

architecture Behavioral of comp_16 is
SIGNAL M : STD_LOGIC;

begin
        M<= '1' when (P>Q) else '0';
        P_GT_Q<=M;
end Behavioral;
```

4.7 Autómata de Control

```
--------------------------------------------------------------------------------
-- AUTOMATA DE CONTROL
--
-- Genera una seal con 8 bits
-- Los primeros 3 bits numeran la cantidad de PUNTOS y RAYAS de cada palabra
```

```
-- Los ultimos 5 bits representan de izquierda a derecha cada simbolo,
-- un '0' sera un PUNTO y un '1' sera una RAYA.
-- Cada palabra acaba cuando el automata alcanza el estado ESPACIO,
-- que es cuando se valida.
----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;

use IEEE.STD_LOGIC_unsigned.ALL;

entity aut_control is
Port( CLK_1ms : in STD_LOGIC; -- reloj
              VALID : in STD_LOGIC; -- entrada de dato vlido
              DATO : in STD_LOGIC; -- dato (0 o 1)
              C0 : in STD_LOGIC; -- resultado comparador de ceros
              C1 : in STD_LOGIC; -- resultado comparador de unos
              CS : in STD_LOGIC; -- resultado comparador de espacios
              RESET0 : in STD_LOGIC;
              PULSADOR : in STD_LOGIC;
              FINAL : in STD_LOGIc;
              CODIGO : out STD_LOGIC_VECTOR (7 downto 0); -- codigo morse obtenido
              VALID_DISP : out STD_LOGIC;
              LED : out STD_LOGIC;
              CONTADOR : out STD_LOGIC_VECTOR(8 downto 0));
end aut_control;

architecture a_aut_control of aut_control is
type STATE_TYPE is (ESPACIO,RESET,SIMBOLO,ESPERA, SEPARADOR,SEPARADOR_VALID,FIN);
signal ST : STATE_TYPE := RESET;
signal s_ncod : STD_LOGIC_VECTOR (2 downto 0):= "000";
signal s_cod : STD_LOGIC_VECTOR (4 downto 0):= "00000";
signal contador0: STD_LOGIC_VECTOR(8 downto 0) := "000000000";
signal n : INTEGER range 0 to 4;

begin
process (CLK_1ms, RESET0)
 begin
 if (RESET0 = '1') then --RESET asincrono
       contador0<= "000000000"; --Resetea el contador de palabras
       ST<=RESET; -- Pone el automata listo para recibir un nuevo simbolo

 elsif (CLK_1ms'event and CLK_1ms='1') then
 case ST is
 when SIMBOLO => --Se ha recibido un punto o una raya
       s_ncod<=s_ncod+1; --Suma 1 al numero de datos recibidos dentro de la mismo simbolo
       s_cod(n)<=C1; -- El resultado del comparador indica si el simbolo punto o raya
       n<=n-1; --Todo simbolo no puede tener mas de 5 datos
       ST<=ESPERA;

 when ESPERA => -- Esperando un nuevo simbolo
       if (valid='1' and dato='1') then -- Si se ha recibido un dato
              ST<=SIMBOLO;
       elsif(valid='1' and dato='0' and C0='0') then -- Si no se ha recibido nada
              ST<=ESPERA;
       elsif(valid='1' and dato='0' and C0='1') then -- Si el simbolo ha terminado
              ST<=ESPACIO;
       end if;

 when ESPACIO => -- VALIDA EL SIMBOLO
       -- Detector de FIN de mensaje basico
       -- if(s_ncod="011" and s_cod="10100") then
       --      ST<=FIN;
       if (CS='1') then -- Si el espacio que ha acabado el dato es de 700ms
              ST<=SEPARADOR;
       else
              ST<=RESET; -- Preparacion para recibir un nuevo simbolo
       end if;

 when SEPARADOR => -- Se ha acabado la palabra
       s_ncod<="010"; -- Letra M (Display apagado)
       s_cod<="11000";
       contador0<= contador0 + 1; -- Suma 1 al numero de palabras
       ST<= SEPARADOR_VALID;

 when SEPARADOR_VALID=> -- Valida el simbolo de SEPARADOR
```

```
       ST<= RESET;

 when RESET => --Prepara el automata para recibir un nuevo simbolo
       n <= 4;
       s_ncod<="000";
       s_cod<="00000";

       if(FINAL='1') then -- Si el mensaje ha terminado
              ST<= FIN;
       elsif (VALID='1' and dato='1') then ---Si se ha recibido un simbolo
              ST<=SIMBOLO;
       else
              ST<=RESET;
 end if;

 when FIN =>
       n <= 4;
       s_ncod<="000";
       s_cod<="00000";
       if (PULSADOR='1') then --Cuando el punsador valga '1'
              ST<=ESPERA;             -- se reactiva el automata.
       else
              ST <= FIN;
       end if;
 end case;
 end if;
end process;

-- PARTE COMBINACIONAL
LED <='1' when (ST=FIN) else '0';
VALID_DISP<='1' when (ST=ESPACIO or ST=SEPARADOR_VALID)  else '0';
CODIGO(4 downto 0)<= s_cod;
CODIGO(7 downto 5)<= s_ncod;
CONTADOR <= contador0;
end a_aut_control;
```

4.8 Visualización

```
--------------------------------------------------------------------------------
-- VISUALIZACION
--
-- Tiene como entradas los simbolos
-- en morse y como salidas la activacion
-- del display y el segmento correspondiente a estos.
--
--------------------------------------------------------------------------------
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 (7 downto 0); -- Entrada sig. carcter
              EN : in STD_LOGIC; -- Activacin para desplazamiento
              CLK_1ms : in STD_LOGIC; -- Entrada de reloj
              RESET0 : in STD_LOGIC;
              SELECTOR: in STD_LOGIC;
              CENTENAS: in STD_LOGIC_VECTOR(7 downto 0);
              DECENAS: in STD_LOGIC_VECTOR(7 downto 0);
              UNIDADES: in STD_LOGIC_VECTOR(7 downto 0);
              SEG7 : out STD_LOGIC_VECTOR (0 to 6); -- Segmentos displays
              AN : out STD_LOGIC_VECTOR (3 downto 0)); -- Activacion displays
end visualizacion;

architecture a_visualizacion of visualizacion is
component MUX4x8
 Port ( E0 : in STD_LOGIC_VECTOR (7 downto 0); -- Entrada 0
 E1 : in STD_LOGIC_VECTOR (7 downto 0); -- Entrada 1
 E2 : in STD_LOGIC_VECTOR (7 downto 0); -- Entrada 2
 E3 : in STD_LOGIC_VECTOR (7 downto 0); -- Entrada 3
 S : in STD_LOGIC_VECTOR (1 downto 0); -- Seal de control
 Y : out STD_LOGIC_VECTOR (7 downto 0)); -- Salida
end component;
```

```
component decodmorsea7s
 Port ( SIMBOLO : in STD_LOGIC_VECTOR (7 downto 0);
 SEGMENTOS : out STD_LOGIC_VECTOR (0 to 6));
end component
;
component refresco
 Port ( CLK_1ms : 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
end component;

component rdesp_disp
 Port ( CLK_1ms : in STD_LOGIC; -- entrada de reloj
 EN : in STD_LOGIC; -- enable
 E : in STD_LOGIC_VECTOR (7 downto 0); -- entrada de datos
 RESET0 : in STD_LOGIC;
 SELECTOR: in STD_LOGIC;
 CENTENAS: in STD_LOGIC_VECTOR(7 downto 0);
 DECENAS: in STD_LOGIC_VECTOR(7 downto 0);
 UNIDADES: in STD_LOGIC_VECTOR(7 downto 0);
 Q0 : out STD_LOGIC_VECTOR (7 downto 0); -- salida Q0
 Q1 : out STD_LOGIC_VECTOR (7 downto 0); -- salida Q1
 Q2 : out STD_LOGIC_VECTOR (7 downto 0); -- salida Q2
 Q3 : out STD_LOGIC_VECTOR (7 downto 0)); -- salida Q3
end component;

SIGNAL Q0: STD_LOGIC_VECTOR (7 downto 0);
SIGNAL Q1: STD_LOGIC_VECTOR (7 downto 0);
SIGNAL Q2: STD_LOGIC_VECTOR (7 downto 0);
SIGNAL Q3: STD_LOGIC_VECTOR (7 downto 0);
SIGNAL S0: STD_LOGIC_VECTOR (1 DOWNTO 0);
SIGNAL Y: STD_LOGIC_VECTOR (7 DOWNTO 0);

begin

H1: rdesp_disp port map (CLK_1ms,EN,E0,RESET0,SELECTOR,CENTENAS,DECENAS,UNIDADES,Q0,Q1,Q2,Q3);
H2: refresco port map(CLK_1ms, S0,AN);
H3: MUX4x8 port map(Q0,Q1,Q2,Q3,S0,Y);
H4: decodmorsea7s port map(Y,SEG7);

end a_visualizacion;
```

4.9 Registro de Desplazamiento

```
----------------------------------------------------------------------------------
-- REGISTRO DESPLAZAMIENTO
--
-- Este modulo desplaza las senales
-- eliminando la mas antigua y andiendo
-- la nueva senal cada vez que EN vale '1'
-- que es cuando ha llegado un simbolo nuevo
--
----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;

use IEEE.STD_LOGIC_unsigned.ALL;

entity rdesp_disp is
Port( CLK_1ms : in STD_LOGIC; -- entrada de reloj
               EN : in STD_LOGIC; -- enable
               E : in STD_LOGIC_VECTOR (7 downto 0); -- entrada de datos
               RESET0 : in STD_LOGIC;
               SELECTOR : in STD_LOGIC; --Selecciona entre el mensaje y el numero de palabras
               CENTENAS : in STD_LOGIC_VECTOR(7 downto 0);
               DECENAS : in STD_LOGIC_VECTOR(7 downto 0);
               UNIDADES : in STD_LOGIC_VECTOR(7 downto 0);
               Q0 : out STD_LOGIC_VECTOR (7 downto 0); -- salida Q0
               Q1 : out STD_LOGIC_VECTOR (7 downto 0); -- salida Q1
               Q2 : out STD_LOGIC_VECTOR (7 downto 0); -- salida Q2
               Q3 : out STD_LOGIC_VECTOR (7 downto 0)); -- salida Q3
end rdesp_disp;
```

```
architecture rdesp_disp of rdesp_disp is
SIGNAL QS0: STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL QS1: STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL QS2: STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL QS3: STD_LOGIC_VECTOR(7 DOWNTO 0);

begin
process(CLK_1ms,RESET0) --Reset asincrono
begin
 if(RESET0='1') then
  QS0<="01110100";
  QS1<="01110100";
  QS2<="01110100";
  QS3<="01110100";
 elsif (CLK_1ms'event and CLK_1ms='1') then
       if(EN='1') then -- Cuando llega un nuevo simbolo
              QS0<=QS1;    -- se hace el desplazamiento
              QS1<=QS2;
              QS2<=QS3;
              QS3<=E;   --Introduce el nuevo simbolo
       end if;
 end if;
end process;
-- MUX con SELECTOR como bit de control
Q0<= QS0 when (SELECTOR='1') else "01110100"; -- Si no esta mostrando el mensaje esta apagado
Q1<= QS1 when (SELECTOR='1') else CENTENAS;
Q2<= QS2 when (SELECTOR='1') else DECENAS;
Q3<= QS3 when (SELECTOR='1') else UNIDADES;
end rdesp_disp;
```

4.10 Refresco

```
----------------------------------------------------------------------------------
-- REFRESCO
--
-- Este modulo tiene solo como entrada el Clk
-- Y en cada ciclo de este (cada milisegundo)
-- activa uno de los 4 displays (salida AN)
-- y indica que display esta activado usando
-- el vector S.
-- De esta forma cada display actualiza su valor
-- durante un ciclo de reloj (1 ms) y se mantiene
-- sin cambios durante 3 ciclos (3 ms).
--
----------------------------------------------------------------------------------
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_1ms : in STD_LOGIC; -- reloj de refresco
              S : out STD_LOGIC_VECTOR (1 downto 0); -- Control para el mux
              AN : out STD_LOGIC_VECTOR (3 downto 0)); -- Control displays
end refresco;

architecture Behavioral of refresco is
SIGNAL M: STD_LOGIC_VECTOR(1 DOWNTO 0):="00"; --Inicializacion de M

begin
process(CLK_1ms)
begin
if (CLK_1ms'event and CLK_1ms='1') then
       if(M="11") then -- Va actualizando el valor de M de
              M<="00";     -- 0 a 3 en cada ciclo de reloj
       else
              M <= M+1; --Paso al siguiente display
       end if;
 end if;
end process;
```

```
WITH M SELECT AN <=
 "1110" WHEN "00", --Activo el cuarto display
 "1101" WHEN "01", --Activo el tercer display
 "1011" WHEN "10", --Activo el segundo display
 "0111" WHEN others;--Activo el primer display

S<=M;
end Behavioral;
```

4.11 Multiplexor

```
----------------------------------------------------------------------------------
-- MULTIPLEXOR
--
-- La senales de entrada E0-E3 son las 4 seales
-- que deberan salir en los displays mientras
-- que S es el display que en ese momento esta activo
-- Por lo que la salida sera la senal correspondiente
-- al display que en ese momento este activo.
-- El objetivo es que este modulo este sincronizado con refresco
-- Para que a la vez que se activa un dispay se le pase la senal
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;

use IEEE.STD_LOGIC_unsigned.ALL;

entity MUX4x8 is
 Port ( E0 : in STD_LOGIC_VECTOR (7 downto 0); -- Entrada 0
                E1 : in STD_LOGIC_VECTOR (7 downto 0); -- Entrada 1
                E2 : in STD_LOGIC_VECTOR (7 downto 0); -- Entrada 2
                E3 : in STD_LOGIC_VECTOR (7 downto 0); -- Entrada 3
                S : in STD_LOGIC_VECTOR (1 downto 0); -- Senal de control
                Y : out STD_LOGIC_VECTOR (7 downto 0)); -- Salida
end MUX4x8;


architecture MUX4x8 of MUX4x8 is
begin
with S select Y<= -- S es la senal de control
       E0 when "00", -- Cuarto Display
       E1 when "01", -- Tercer Display
       E2 when "10", -- Segundo Display
       E3 when others;-- Primer Display


end MUX4x8;
```

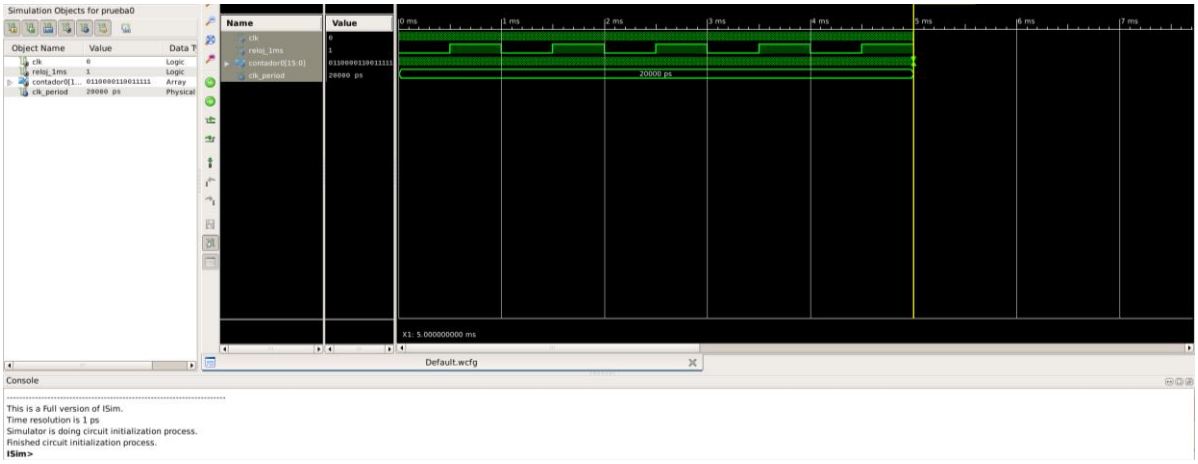
4.12 Asociaciones

```
# Reloj principal del sistema
NET "CLK" LOC = "M6"; # Seal de reloj del sistema
# Conexiones de los DISPLAYS
NET "SEG7<0>" LOC = "L14"; # seal = CA
NET "SEG7<1>" LOC = "H12"; # Seal = CB
NET "SEG7<2>" LOC = "N14"; # Seal = CC
NET "SEG7<3>" LOC = "N11"; # Seal = CD
NET "SEG7<4>" LOC = "P12"; # Seal = CE
NET "SEG7<5>" LOC = "L13"; # Seal = CF
NET "SEG7<6>" LOC = "M12"; # Seal = CG
# Seales de activacin de los displays
NET "AN<0>" LOC = "K14"; # Activacin del display 0 = AN0
NET "AN<1>" LOC = "M13"; # Activacin del display 1 = AN1
NET "AN<2>" LOC = "J12"; # Activacin del display 2 = AN2
NET "AN<3>" LOC = "F12"; # Activacin del display 3 = AN3
#Entrada externa donde se conecta la seal entrante
NET "LIN" LOC = "B2"; # Entrada de datos
#Salidas para generar la seal analgica (no tocar estas lneas)
NET "SPI_CLK" LOC = "A9"; # Salida externa terminal 14
NET "SPI_DIN" LOC = "B9"; # Salida externa terminal 15
NET "SPI_CS1" LOC = "C9"; # Salida externa terminal 17
#Entradas para arrancar y parar la seal analgica (no tocar estas lneas)
NET "BTN_START" LOC = "G12"; # Entrada externa BTN0
NET "BTN_STOP" LOC = "C11"; # Entrada externa BTN1
```

```
NET "RESET0" LOC = "M4"; #RESET
NET "PULSADOR" LOC = "A7"; #Reactivar recepcion de mensaje
NET "LED" LOC = "M5"; #Led que indica fin de mensaje

NET "SELECTOR" LOC = "P11"; #Recepcion del mensaje o mostrar numero de palabras
```

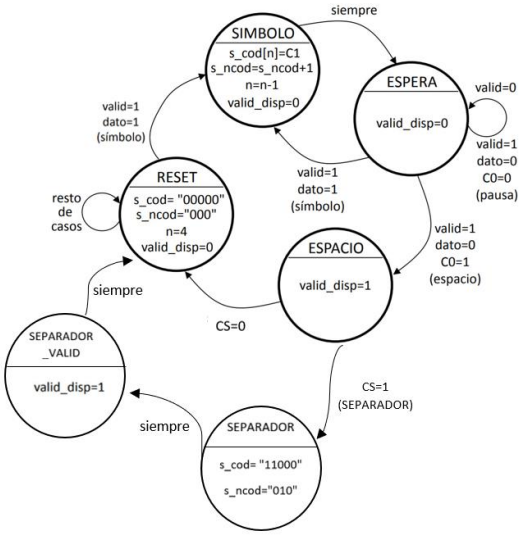
TestBench CLK

```
--------------------------------------------------------------------------------
-- Company:
--------------------------------------------------------------------------------
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY Prueba0 IS
END Prueba0;

ARCHITECTURE behavior OF Prueba0 IS

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

    COMPONENT div_reloj
    PORT(
         CLK : IN  std_logic;
         Reloj_1ms : OUT  std_logic;
         contador0 : OUT  std_logic_vector(15 downto 0)
        );
    END COMPONENT;


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

       --Outputs
   signal Reloj_1ms : std_logic;
   signal contador0 : std_logic_vector(15 downto 0);

   -- Clock period definitions
   constant CLK_period : time := 20 ns;

BEGIN

       -- Instantiate the Unit Under Test (UUT)
   uut: div_reloj PORT MAP (
          CLK => CLK,
          Reloj_1ms => Reloj_1ms,
          contador0 => contador0
        );

   -- 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;
```



5. MEJORAS

5.1 SEPARADOR



El objetivo es que cuando aparezca una pausa de 700 ms en vez de considerarlo como un espacio entre simbolos, considerarlo como un espacio entre palabras. Para ello primero en el autómata de duración tenemos que aumentar el tiempo máximo de recepción de '0' de 600ms a un tiempo mayor de 700ms (hemos usado 1400ms). Después tendremos que crear otro comparador (CS) que analiza si el tiempo transcurrido es menor que 500 (Pausa o Espacio) o mayor (Separador), elegimos 500 ms porque es el tiempo medio entre la separación de palabras (700ms) y la separación de simbolos(300ms). Finalmente habrá que cambiar el autómata de control para que sea como en de la imagen, en este nuevo autómata tras validar el último símbolo que ha entrado comprueba si el '0' que ha llevado el autómata a ESPACIO es un espacio entre símbolos (300 ms y CS='0') o se trata de un espacio entre palabras (700 ms y CS='1') , si se cumple esta segunda condición el autómata antes de volver a reset (Preparación para recibir un nuevo símbolo) valida un nuevo símbolo que su representación en el display sea dejarlo apagado, hemos usado "01011000" que se corresponde con la letra 'M' y cuya representación es un display apagado.

El código que hay que cambiar con respecto a la práctica básica es el siguiente (marcado en rojo):

-Autómata de Duración:

```
----------------------------------------------------------------------------
-- AUTOMATA DURACION
```

```
--
-- Este automata consigue mediante la entrada ('0' o '1') y el tiempo que esta dura
-- distinguir entre las cuatro posibles senales (RAYA, PUNTO, PAUSA O ESPACIO).
-- Si DATO vale 1 se trata de PUNTO o RAYA y si vale 0 se trata de PAUSA o ESPACIO
-- Si DURACION es mayor que un umbral estarmos mandando RAYA O ESPACIO
-- y si es menor estaremos mandando PAUSA O PUNTO.
-- Las salidas solo se usaran cuando se valide: VALID='1'.
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;

use IEEE.STD_LOGIC_unsigned.ALL;

entity aut_duracion is
  Port ( CLK_1ms : in  STD_LOGIC;
         ENTRADA : in  STD_LOGIC;
         VALID : out  STD_LOGIC;
         DATO : out  STD_LOGIC;
         DURACION : out  STD_LOGIC_VECTOR (15 downto 0));
end aut_duracion;

architecture a_aut_duracion of aut_duracion is
type STATE_TYPE is (CERO,ALM_CERO,VALID_CERO,UNO,ALM_UNO,VALID_UNO,VALID_FIN);
signal ST : STATE_TYPE := CERO;
signal cont : STD_LOGIC_VECTOR (15 downto 0):="0000000000000000";
signal reg : STD_LOGIC_VECTOR (15 downto 0) :="0000000000000000";

begin
process (CLK_1ms)
 begin
 if (CLK_1ms'event and CLK_1ms='1') then --Cada milisegundo
 case ST is
       when CERO =>
              cont<=cont+1; -- Almacena el tiempo en ms que lleva recibiendo '0'
              if (cont>1400) then --Si la cuenta sobrepasa este tiempo es que no se esta
recibiendo nada
                     ST<=VALID_FIN;
              elsif (ENTRADA='0') then
                     ST<=CERO;
              else
                     ST<=ALM_CERO; -- Cuando se recibe un '1' estado recibiendo '0'
              end if;                  -- el automata cambia de estado

       when ALM_CERO =>
              reg<=cont; --Guarda el tiempo que ha estado en '0'
              cont<=(others=>'0');
              ST<= VALID_CERO;

       when VALID_CERO =>
              ST<= UNO; -- Valida el '0'

       when UNO=>
              cont<=cont+1; -- Almacena el tiempo en ms que lleva recibiendo '0'
              if (ENTRADA='0') then
                     ST<=ALM_UNO; -- Cuando se recibe un '0' estado recibiendo '1'
              else                           -- el automata cambia de estado
                     ST<=UNO;
              end if;

       when ALM_UNO =>
              reg<=cont; --Actualiza el contador
              cont<=(others=>'0');
              ST<= VALID_UNO;

       when VALID_UNO =>
              ST<= CERO; --Valida el dato

       when VALID_FIN =>
              reg<=cont;
              cont<=(others=>'0'); --Resetea el contador
              ST<= CERO;
  end case;
 end if;
 end process;

 -- PARTE COMBINACIONAL
```

```
 VALID<='1' when (ST=VALID_CERO or ST=VALID_UNO or ST=VALID_FIN) else '0';
 DATO <='1' when (ST=UNO or ST=VALID_UNO or ST=ALM_UNO) else '0';
 DURACION<= reg;

end a_aut_duracion;
```

-Receptor:

```
----------------------------------------------------------------------------------
-- RECEPTOR
-- Contiene diferentes modulos
-- que transforman la seal binaria
-- en simbolos morse y un modulo de
-- visualizacion que transforma cada simblolo
-- en un digito del display
--
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

use IEEE.STD_LOGIC_arith.ALL;

use IEEE.STD_LOGIC_unsigned.ALL;


entity receptor is
Port ( CLK : in  STD_LOGIC; --Reloj con periodo de 20 ns
       LIN : in  STD_LOGIC; --Entrada de Datos
                RESET0 : in STD_LOGIC; --Reset asincrono
                PULSADOR : in STD_LOGIC; -- Pone de nuevo el display en funcionamiento
                SELECTOR: in STD_LOGIC; --Mostrar mensaje o numero de palabras
       AN : out  STD_LOGIC_VECTOR (3 downto 0); -- Activacion individual displays
       SEG7 : out  STD_LOGIC_VECTOR (0 to 6); -- Salida para los displays
                LED : out STD_LOGIC); --Indica el fin del mensaje
end receptor;

architecture Behavioral of receptor is
constant UMBRAL0 : STD_LOGIC_VECTOR (15 downto 0) := "0000000011001000"; -- 200 umbral ceros
constant UMBRAL1 : STD_LOGIC_VECTOR (15 downto 0) := "0000000011001000"; -- 200 umbral unos
constant UMBRALS : STD_LOGIC_VECTOR (15 downto 0) := "0000000111110100"; -- 500 umbral ceros
(SEPARADOR)
       SIGNAL CLK_1ms : STD_LOGIC;
       SIGNAL LIN2 : STD_LOGIC;
       SIGNAL VALID : STD_LOGIC;
       SIGNAL DATO : STD_LOGIC;
       SIGNAL DURACION : STD_LOGIC_VECTOR (15 downto 0);
       SIGNAL C0 : STD_LOGIC;
       SIGNAL C1 : STD_LOGIC;
       SIGNAL CS : STD_LOGIC;
       SIGNAL CODIGO : STD_LOGIC_VECTOR (7 downto 0);
       SIGNAL VALID_DISP : STD_LOGIC;
       SIGNAL FINAL : STD_LOGIC;
       SIGNAL CONTADOR : STD_LOGIC_VECTOR (8 downto 0);
       SIGNAL CENTENAS : STD_LOGIC_VECTOR (7 downto 0);
       SIGNAL DECENAS : STD_LOGIC_VECTOR (7 downto 0);
       SIGNAL UNIDADES : STD_LOGIC_VECTOR (7 downto 0);

component div_reloj is
Port ( CLK : in  STD_LOGIC;
       CLK_1ms : out  STD_LOGIC);
end component;

component detector_flanco is
Port ( CLK_1ms : in STD_LOGIC; -- reloj
                LIN : in STD_LOGIC; -- Lnea de datos
                VALOR : out STD_LOGIC); -- Valor detectado en el flanco
end component;

component aut_duracion is
Port ( CLK_1ms : in  STD_LOGIC;
       ENTRADA : in  STD_LOGIC;
       VALID : out  STD_LOGIC;
       DATO : out  STD_LOGIC;
       DURACION : out  STD_LOGIC_VECTOR (15 downto 0));
end component;
```

```
component comp_16 is
Port ( P : in  STD_LOGIC_VECTOR (15 downto 0);
       Q : in  STD_LOGIC_VECTOR (15 downto 0);
       P_GT_Q : out  STD_LOGIC);
end component;

component aut_control is
Port( CLK_1ms : in STD_LOGIC; -- reloj
               VALID : in STD_LOGIC; -- entrada de dato vlido
               DATO : in STD_LOGIC; -- dato (0 o 1)
               C0 : in STD_LOGIC; -- resultado comparador de ceros
               C1 : in STD_LOGIC; -- resultado comparador de unos
               CS : in STD_LOGIC; -- resultado comparador de espacios
               RESET0 : in STD_LOGIC;
               PULSADOR : in STD_LOGIc;
               FINAL : in STD_LOGIc;
               CODIGO : out STD_LOGIC_VECTOR (7 downto 0); -- cdigo morse obtenido
               VALID_DISP : out STD_LOGIC;-- validacin del display
               LED : out STD_LOGIC;
               CONTADOR: out STD_LOGIC_VECTOR(8 downto 0));
end component;

component visualizacion is
 Port( E0 : in STD_LOGIC_VECTOR (7 downto 0); -- Entrada sig. carcter
                EN : in STD_LOGIC; -- Activacin para desplazamiento
                CLK_1ms : in STD_LOGIC; -- Entrada de reloj
                RESET0 : in STD_LOGIC;
                SELECTOR: in STD_LOGIC;
                CENTENAS: in STD_LOGIC_VECTOR(7 downto 0);
                DECENAS: in STD_LOGIC_VECTOR(7 downto 0);
                UNIDADES: in STD_LOGIC_VECTOR(7 downto 0);
                SEG7 : out STD_LOGIC_VECTOR (0 to 6); -- Segmentos displays
                AN : out STD_LOGIC_VECTOR (3 downto 0)); -- Activacin displays
end component;

component Deteccionfin is
 Port( CLK_1ms : in STD_LOGIC;
                CODIGO : in STD_LOGIC_VECTOR (7 downto 0); -- cdigo morse obtenido
                VALID_DISP : in STD_LOGIC;
                FINAL : out STD_LOGIC );
end component;

component bintobcd is
 Port( num_bin: in  STD_LOGIC_VECTOR(8 downto 0);
                centenas: out STD_LOGIC_VECTOR(7 downto 0);
                decenas: out STD_LOGIC_VECTOR(7 downto 0);
                unidades: out STD_LOGIC_VECTOR(7 downto 0));
end component;

begin
H1: div_reloj port map (CLK,CLK_1ms);
H2: detector_flanco port map (CLK_1ms, LIN,LIN2);
H3: aut_duracion port map (CLK_1ms, LIN2, VALID, DATO, DURACION);
H4: comp_16 port map (DURACION,UMBRAL0,C0);
H5: comp_16 port map (DURACION,UMBRAL1,C1);
HS: comp_16 port map (DURACION,UMBRALS,CS);
H6: aut_control port map
(CLK_1ms,VALID,DATO,C0,C1,CS,RESET0,PULSADOR,FINAL,CODIGO,VALID_DISP,LED,CONTADOR);
M4: bintobcd port map (CONTADOR,CENTENAS,DECENAS,UNIDADES);
HF: Deteccionfin port map (CLK_1ms, CODIGO, VALID_DISP,FINAL);
H7: visualizacion port
map(CODIGO,VALID_DISP,CLK_1ms,RESET0,SELECTOR,CENTENAS,DECENAS,UNIDADES,SEG7,AN);
end Behavioral;
```

-Autómata de Control

```
----------------------------------------------------------------------------------
-- AUTOMATA DE CONTROL
--
-- Genera una seal con 8 bits
-- Los primeros 3 bits numeran la cantidad de PUNTOS y RAYAS de cada palabra
-- Los ultimos 5 bits representan de izquierda a derecha cada simbolo,
-- un '0' sera un PUNTO y un '1' sera una RAYA.
-- Cada palabra acaba cuando el automata alcanza el estado ESPACIO,
-- que es cuando se valida.
```

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

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;

use IEEE.STD_LOGIC_unsigned.ALL;

entity aut_control is
Port( CLK_1ms : in STD_LOGIC; -- reloj
              VALID : in STD_LOGIC; -- entrada de dato vlido
              DATO : in STD_LOGIC; -- dato (0 o 1)
              C0 : in STD_LOGIC; -- resultado comparador de ceros
              C1 : in STD_LOGIC; -- resultado comparador de unos
              CS : in STD_LOGIC; -- resultado comparador de espacios
              RESET0 : in STD_LOGIC;
              PULSADOR : in STD_LOGIC;
              FINAL : in STD_LOGIc;
              CODIGO : out STD_LOGIC_VECTOR (7 downto 0); -- codigo morse obtenido
              VALID_DISP : out STD_LOGIC;
              LED : out STD_LOGIC;
              CONTADOR : out STD_LOGIC_VECTOR(8 downto 0));
end aut_control;

architecture a_aut_control of aut_control is
type STATE_TYPE is (ESPACIO,RESET,SIMBOLO,ESPERA, SEPARADOR,SEPARADOR_VALID,FIN);
signal ST : STATE_TYPE := RESET;
signal s_ncod : STD_LOGIC_VECTOR (2 downto 0):= "000";
signal s_cod : STD_LOGIC_VECTOR (4 downto 0):= "00000";
signal contador0: STD_LOGIC_VECTOR(8 downto 0) := "000000000";
signal n : INTEGER range 0 to 4;

begin
process (CLK_1ms, RESET0)
 begin
 if (RESET0 = '1') then --RESET asincrono
       contador0<= "000000000"; --Resetea el contador de palabras
       ST<=RESET; -- Pone el automata listo para recibir un nuevo simbolo

 elsif (CLK_1ms'event and CLK_1ms='1') then
 case ST is
 when SIMBOLO => --Se ha recibido un punto o una raya
       s_ncod<=s_ncod+1; --Suma 1 al numero de datos recibidos dentro de la mismo simbolo
       s_cod(n)<=C1; -- El resultado del comparador indica si el simbolo punto o raya
       n<=n-1; --Todo simbolo no puede tener mas de 5 datos
       ST<=ESPERA;

 when ESPERA => -- Esperando un nuevo simbolo
       if (valid='1' and dato='1') then -- Si se ha recibido un dato
              ST<=SIMBOLO;
       elsif(valid='1' and dato='0' and C0='0') then -- Si no se ha recibido nada
              ST<=ESPERA;
       elsif(valid='1' and dato='0' and C0='1') then -- Si el simbolo ha terminado
              ST<=ESPACIO;
       end if;

 when ESPACIO => -- VALIDA EL SIMBOLO
       -- Detector de FIN de mensaje basico
       -- if(s_ncod="011" and s_cod="10100") then
       --      ST<=FIN;
       if (CS='1') then -- Si el espacio que ha acabado el dato es de 700ms
              ST<=SEPARADOR;
       else
              ST<=RESET; -- Preparacion para recibir un nuevo simbolo
       end if;

 when SEPARADOR => -- Se ha acabado la palabra
       s_ncod<="010"; -- Letra M (Display apagado)
       s_cod<="11000";
       contador0<= contador0 + 1; -- Suma 1 al numero de palabras
       ST<= SEPARADOR_VALID;

 when SEPARADOR_VALID=> -- Valida el simbolo de SEPARADOR
       ST<= RESET;

 when RESET => --Prepara el automata para recibir un nuevo simbolo
       n <= 4;
```

```
        s_ncod<="000";
        s_cod<="00000";

        if(FINAL='1') then -- Si el mensaje ha terminado
                ST<= FIN;
        elsif (VALID='1' and dato='1') then ---Si se ha recibido un simbolo
                ST<=SIMBOLO;
        else
                ST<=RESET;
 end if;

 when FIN =>
        n <= 4;
        s_ncod<="000";
        s_cod<="00000";
        if (PULSADOR='1') then --Cuando el punsador valga '1'
                ST<=ESPERA;             -- se reactiva el automata.
        else
                ST <= FIN;
        end if;
 end case;
 end if;
end process;

-- PARTE COMBINACIONAL
LED <='1' when (ST=FIN) else '0';
VALID_DISP<='1' when (ST=ESPACIO or ST=SEPARADOR_VALID)  else '0';
CODIGO(4 downto 0)<= s_cod;
CODIGO(7 downto 5)<= s_ncod;
CONTADOR <= contador0;
end a_aut_control;
```

5.2 Reset Asíncrono

Esta mejora tiene dos etapas, la primera consiste preparar el receptor para la recepción de un nuevo símbolo y la segunda es borrar el contenido actual de los displays. Para lo primero habrá que cambiar el autómata de control mientras para lo segundo habrá que cambiar el módulo cuyas salidas se correspondan con los diferentes símbolos del display, el registro de desplazamiento. El primer paso será introducir un pulsador con el que poder resetear el receptor, por ello habrá que introducir esta señal en las asociaciones y en los módulos Main, Receptor, Autómata de Control, Visualización y Registro de Desplazamiento.
Para de la submejora 1 (preparar el autómata para recibir un nuevo símbolo) en el autómata de control habrá que introducir el reset en la lista de sensibilidades del proceso (reset asíncrono) y antes que el if del CLK introducir un if(Reset='1'){…} que manda el autómata al Estado RESET y cambiar el if del CLK por un elsif.
Para la sub mejora 2: Apagar los displays, dentro del registro de desplazamiento habrá que introducir también el reset en la lista de sensibilidades del proceso y antes del if del Clk poner otro condicional que cambie todas las señales por un símbolo cuyo equivalente sea un display apagado, en este caso hemos usado “01110100” que se corresponde con la letra ‘K’ el if del Clk habría que cambiarlo por un elsif.

-Asociaciones

```
# Reloj principal del sistema
NET "CLK" LOC = "M6"; # Seal de reloj del sistema
# Conexiones de los DISPLAYS
NET "SEG7<0>" LOC = "L14"; # seal = CA
NET "SEG7<1>" LOC = "H12"; # Seal = CB
NET "SEG7<2>" LOC = "N14"; # Seal = CC
NET "SEG7<3>" LOC = "N11"; # Seal = CD
NET "SEG7<4>" LOC = "P12"; # Seal = CE
NET "SEG7<5>" LOC = "L13"; # Seal = CF
NET "SEG7<6>" LOC = "M12"; # Seal = CG
# Seales de activacin de los displays
NET "AN<0>" LOC = "K14"; # Activacin del display 0 = AN0
NET "AN<1>" LOC = "M13"; # Activacin del display 1 = AN1
NET "AN<2>" LOC = "J12"; # Activacin del display 2 = AN2
NET "AN<3>" LOC = "F12"; # Activacin del display 3 = AN3
```

```
#Entrada externa donde se conecta la seal entrante
NET "LIN" LOC = "B2"; # Entrada de datos
#Salidas para generar la seal analgica (no tocar estas lneas)
NET "SPI_CLK" LOC = "A9"; # Salida externa terminal 14
NET "SPI_DIN" LOC = "B9"; # Salida externa terminal 15
NET "SPI_CS1" LOC = "C9"; # Salida externa terminal 17
#Entradas para arrancar y parar la seal analgica (no tocar estas lneas)
NET "BTN_START" LOC = "G12"; # Entrada externa BTN0
NET "BTN_STOP" LOC = "C11"; # Entrada externa BTN1

NET "RESET0" LOC = "M4"; #RESET
NET "PULSADOR" LOC = "A7"; #Reactivar recepcion de mensaje
NET "LED" LOC = "M5"; #Led que indica fin de mensaje

NET "SELECTOR" LOC = "P11"; #Recepcion del mensaje o mostrar numero de palabras
```

-Autómata de Control

```
----------------------------------------------------------------------------------
-- AUTOMATA DE CONTROL
--
-- Genera una seal con 8 bits
-- Los primeros 3 bits numeran la cantidad de PUNTOS y RAYAS de cada palabra
-- Los ultimos 5 bits representan de izquierda a derecha cada simbolo,
-- un '0' sera un PUNTO y un '1' sera una RAYA.
-- Cada palabra acaba cuando el automata alcanza el estado ESPACIO,
-- que es cuando se valida.
----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;

use IEEE.STD_LOGIC_unsigned.ALL;

entity aut_control is
Port( CLK_1ms : in STD_LOGIC; -- reloj
               VALID : in STD_LOGIC; -- entrada de dato vlido
               DATO : in STD_LOGIC; -- dato (0 o 1)
               C0 : in STD_LOGIC; -- resultado comparador de ceros
               C1 : in STD_LOGIC; -- resultado comparador de unos
               CS : in STD_LOGIC; -- resultado comparador de espacios
               RESET0 : in STD_LOGIC;
               PULSADOR : in STD_LOGIC;
               FINAL : in STD_LOGIc;
               CODIGO : out STD_LOGIC_VECTOR (7 downto 0); -- codigo morse obtenido
               VALID_DISP : out STD_LOGIC;
               LED : out STD_LOGIC;
               CONTADOR : out STD_LOGIC_VECTOR(8 downto 0));
end aut_control;

architecture a_aut_control of aut_control is
type STATE_TYPE is (ESPACIO,RESET,SIMBOLO,ESPERA, SEPARADOR,SEPARADOR_VALID,FIN);
signal ST : STATE_TYPE := RESET;
signal s_ncod : STD_LOGIC_VECTOR (2 downto 0):= "000";
signal s_cod : STD_LOGIC_VECTOR (4 downto 0):= "00000";
signal contador0: STD_LOGIC_VECTOR(8 downto 0) := "000000000";
signal n : INTEGER range 0 to 4;

begin
process (CLK_1ms, RESET0)
 begin
 if (RESET0 = '1') then --RESET asincrono
       contador0<= "000000000"; --Resetea el contador de palabras
       ST<=RESET; -- Pone el automata listo para recibir un nuevo simbolo

 elsif (CLK_1ms'event and CLK_1ms='1') then
 case ST is
 when SIMBOLO => --Se ha recibido un punto o una raya
       s_ncod<=s_ncod+1; --Suma 1 al numero de datos recibidos dentro de la mismo simbolo
       s_cod(n)<=C1; -- El resultado del comparador indica si el simbolo punto o raya
       n<=n-1; --Todo simbolo no puede tener mas de 5 datos
       ST<=ESPERA;

 when ESPERA => -- Esperando un nuevo simbolo
```

```
       if (valid='1' and dato='1') then -- Si se ha recibido un dato
              ST<=SIMBOLO;
       elsif(valid='1' and dato='0' and C0='0') then -- Si no se ha recibido nada
              ST<=ESPERA;
       elsif(valid='1' and dato='0' and C0='1') then -- Si el simbolo ha terminado
              ST<=ESPACIO;
       end if;

 when ESPACIO => -- VALIDA EL SIMBOLO
       -- Detector de FIN de mensaje basico
       -- if(s_ncod="011" and s_cod="10100") then
       --      ST<=FIN;
       if (CS='1') then -- Si el espacio que ha acabado el dato es de 700ms
              ST<=SEPARADOR;
       else
              ST<=RESET; -- Preparacion para recibir un nuevo simbolo
       end if;

 when SEPARADOR => -- Se ha acabado la palabra
       s_ncod<="010"; -- Letra M (Display apagado)
       s_cod<="11000";
       contador0<= contador0 + 1; -- Suma 1 al numero de palabras
       ST<= SEPARADOR_VALID;

 when SEPARADOR_VALID=> -- Valida el simbolo de SEPARADOR
       ST<= RESET;

 when RESET => --Prepara el automata para recibir un nuevo simbolo
       n <= 4;
       s_ncod<="000";
       s_cod<="00000";

       if(FINAL='1') then -- Si el mensaje ha terminado
              ST<= FIN;
       elsif (VALID='1' and dato='1') then ---Si se ha recibido un simbolo
              ST<=SIMBOLO;
       else
              ST<=RESET;
 end if;

 when FIN =>
       n <= 4;
       s_ncod<="000";
       s_cod<="00000";
       if (PULSADOR='1') then --Cuando el punsador valga '1'
              ST<=ESPERA;             -- se reactiva el automata.
       else
              ST <= FIN;
       end if;
 end case;
 end if;
end process;

-- PARTE COMBINACIONAL
LED <='1' when (ST=FIN) else '0';
VALID_DISP<='1' when (ST=ESPACIO or ST=SEPARADOR_VALID)  else '0';
CODIGO(4 downto 0)<= s_cod;
CODIGO(7 downto 5)<= s_ncod;
CONTADOR <= contador0;
end a_aut_control;
```

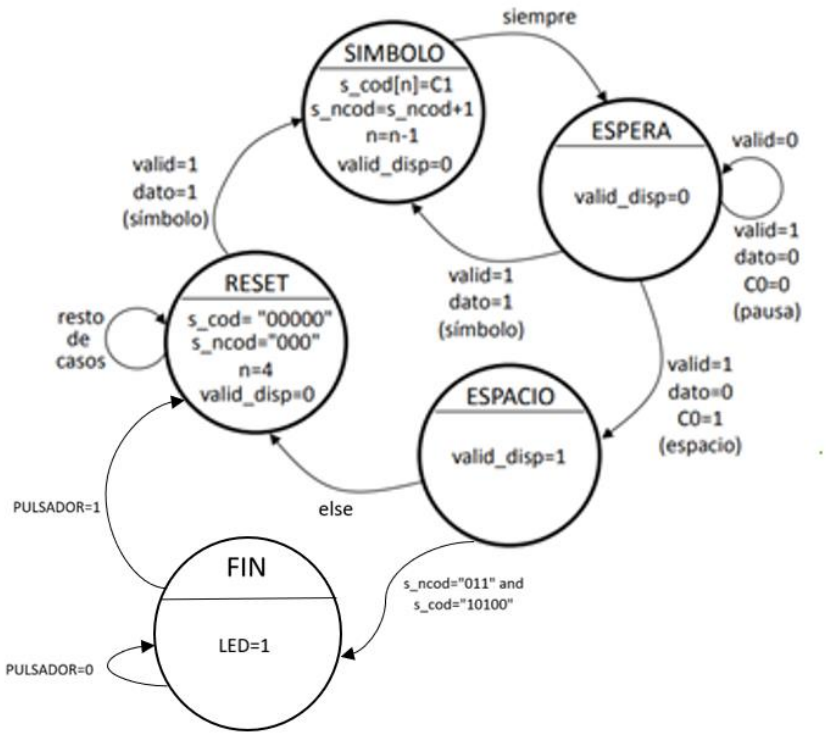
-Registro de Desplazamiento

```
--------------------------------------------------------------------------------
-- REGISTRO DESPLAZAMIENTO
--
-- Este modulo desplaza las senales
-- eliminando la mas antigua y andiendo
-- la nueva senal cada vez que EN vale '1'
-- que es cuando ha llegado un simbolo nuevo
--
--------------------------------------------------------------------------------

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

```
use IEEE.STD_LOGIC_unsigned.ALL;

entity rdesp_disp is
Port( CLK_1ms : in STD_LOGIC; -- entrada de reloj
               EN : in STD_LOGIC; -- enable
               E : in STD_LOGIC_VECTOR (7 downto 0); -- entrada de datos
               RESET0 : in STD_LOGIC;
               SELECTOR : in STD_LOGIC; --Selecciona entre el mensaje y el numero de palabras
               CENTENAS : in STD_LOGIC_VECTOR(7 downto 0);
               DECENAS : in STD_LOGIC_VECTOR(7 downto 0);
               UNIDADES : in STD_LOGIC_VECTOR(7 downto 0);
               Q0 : out STD_LOGIC_VECTOR (7 downto 0); -- salida Q0
               Q1 : out STD_LOGIC_VECTOR (7 downto 0); -- salida Q1
               Q2 : out STD_LOGIC_VECTOR (7 downto 0); -- salida Q2
               Q3 : out STD_LOGIC_VECTOR (7 downto 0)); -- salida Q3
end rdesp_disp;


architecture rdesp_disp of rdesp_disp is
SIGNAL QS0: STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL QS1: STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL QS2: STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL QS3: STD_LOGIC_VECTOR(7 DOWNTO 0);

begin
process(CLK_1ms,RESET0) -- Reset asincrono
begin
 if(RESET0='1') then -- Cuando el reset esta a '1'
  QS0<="01110100";   -- ponemos un simbolo cuya sepresentacion sean
  QS1<="01110100";   -- todos los displays apagados.
  QS2<="01110100";
  QS3<="01110100";
 elsif (CLK_1ms'event and CLK_1ms='1') then
       if(EN='1') then -- Cuando llega un nuevo simbolo
               QS0<=QS1;    -- se hace el desplazamiento
               QS1<=QS2;
               QS2<=QS3;
               QS3<=E;   --Introduce el nuevo simbolo
       end if;
 end if;
end process;
-- MUX con SELECTOR como bit de control
Q0<= QS0 when (SELECTOR='1') else "01110100"; -- Si no esta mostrando el mensaje esta apagado
Q1<= QS1 when (SELECTOR='1') else CENTENAS;
Q2<= QS2 when (SELECTOR='1') else DECENAS;
Q3<= QS3 when (SELECTOR='1') else UNIDADES;
end rdesp_disp;
```

5.3 Final del mensaje

Cuando se recibe una K se paraliza el receptor hasta que se accione un pulsador, por tanto lo primero será introducir una señal PULSADOR (entrada) y una señal LED (salida) en las asociaciones y en los módulos MAIN, RECEPTOR y AUTÓMATA DE CONTROL. Habrá que añadir un nuevo estado de absorción al autómata de control del cual solo se salga cuando la señal pulsador valga ‘1’, la condición para entrar en este estado es que el último símbolo validado sea una K (s_ncod=”011” y s_cod=”10100”), cuando estamos en este estado la salida LED vale ‘1’.




```
--------------------------------------------------------------------------------
-- AUTOMATA DE CONTROL
--
-- Genera una seal con 8 bits
-- Los primeros 3 bits numeran la cantidad de PUNTOS y RAYAS de cada palabra
-- Los ultimos 5 bits representan de izquierda a derecha cada simbolo,
-- un '0' sera un PUNTO y un '1' sera una RAYA.
-- Cada palabra acaba cuando el automata alcanza el estado ESPACIO,
-- que es cuando se valida.
--------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;

use IEEE.STD_LOGIC_unsigned.ALL;

entity aut_control is
Port( CLK_1ms : in STD_LOGIC; -- reloj
              VALID : in STD_LOGIC; -- entrada de dato vlido
              DATO : in STD_LOGIC; -- dato (0 o 1)
              C0 : in STD_LOGIC; -- resultado comparador de ceros
              C1 : in STD_LOGIC; -- resultado comparador de unos
              CS : in STD_LOGIC; -- resultado comparador de espacios
              RESET0 : in STD_LOGIC;
              PULSADOR : in STD_LOGIC;
              FINAL : in STD_LOGIc;
              CODIGO : out STD_LOGIC_VECTOR (7 downto 0); -- codigo morse obtenido
              VALID_DISP : out STD_LOGIC;
              LED : out STD_LOGIC;
              CONTADOR : out STD_LOGIC_VECTOR(8 downto 0));
end aut_control;

architecture a_aut_control of aut_control is
type STATE_TYPE is (ESPACIO,RESET,SIMBOLO,ESPERA, SEPARADOR,SEPARADOR_VALID,FIN);
signal ST : STATE_TYPE := RESET;
signal s_ncod : STD_LOGIC_VECTOR (2 downto 0):= "000";
signal s_cod : STD_LOGIC_VECTOR (4 downto 0):= "00000";
signal contador0: STD_LOGIC_VECTOR(8 downto 0) := "000000000";
signal n : INTEGER range 0 to 4;

begin
```

```
process (CLK_1ms, RESET0)
 begin
 if (RESET0 = '1') then --RESET asincrono
       contador0<= "000000000"; --Resetea el contador de palabras
       ST<=RESET; -- Pone el automata listo para recibir un nuevo simbolo

 elsif (CLK_1ms'event and CLK_1ms='1') then
 case ST is
 when SIMBOLO => --Se ha recibido un punto o una raya
       s_ncod<=s_ncod+1; --Suma 1 al numero de datos recibidos dentro de la mismo simbolo
       s_cod(n)<=C1; -- El resultado del comparador indica si el simbolo punto o raya
       n<=n-1; --Todo simbolo no puede tener mas de 5 datos
       ST<=ESPERA;

 when ESPERA => -- Esperando un nuevo simbolo
       if (valid='1' and dato='1') then -- Si se ha recibido un dato
              ST<=SIMBOLO;
       elsif(valid='1' and dato='0' and C0='0') then -- Si no se ha recibido nada
              ST<=ESPERA;
       elsif(valid='1' and dato='0' and C0='1') then -- Si el simbolo ha terminado
              ST<=ESPACIO;
       end if;

 when ESPACIO => -- VALIDA EL SIMBOLO
       -- Detector de FIN de mensaje basico
       -- if(s_ncod="011" and s_cod="10100") then
       --      ST<=FIN;
       if (CS='1') then -- Si el espacio que ha acabado el dato es de 700ms
              ST<=SEPARADOR;
       else
              ST<=RESET; -- Preparacion para recibir un nuevo simbolo
       end if;

 when SEPARADOR => -- Se ha acabado la palabra
       s_ncod<="010"; -- Letra M (Display apagado)
       s_cod<="11000";
       contador0<= contador0 + 1; -- Suma 1 al numero de palabras
       ST<= SEPARADOR_VALID;

 when SEPARADOR_VALID=> -- Valida el simbolo de SEPARADOR
       ST<= RESET;

 when RESET => --Prepara el automata para recibir un nuevo simbolo
       n <= 4;
       s_ncod<="000";
       s_cod<="00000";

       if(FINAL='1') then -- Si el mensaje ha terminado
              ST<= FIN;
       elsif (VALID='1' and dato='1') then ---Si se ha recibido un simbolo
              ST<=SIMBOLO;
       else
              ST<=RESET;
 end if;

 when FIN =>
       n <= 4;
       s_ncod<="000";
       s_cod<="00000";
       if (PULSADOR='1') then --Cuando el punsador valga '1'
              ST<=ESPERA;             -- se reactiva el automata.
       else
              ST <= FIN;
       end if;
 end case;
 end if;
end process;

-- PARTE COMBINACIONAL
LED <='1' when (ST=FIN) else '0';
VALID_DISP<='1' when (ST=ESPACIO or ST=SEPARADOR_VALID)  else '0';
CODIGO(4 downto 0)<= s_cod;
CODIGO(7 downto 5)<= s_ncod;
CONTADOR <= contador0;
end a_aut_control;
```

5.4 Final de Mensaje mejorado

El objetivo de esta mejora es que el receptor sea capaz de identificar la secuencia ESPACIO-K-ESPACIO, como hemos configurado el espacio como una M, la secuencia que habrá que identificar será M-K-M, identificar esta secuencia dentro del autómata de control requeriría hacer muchos cambios en este, por ello se ha optado por crear un nuevo módulo que se comporta como un buffer almacenando los últimos tres símbolos; si la secuencia de los últimos tres símbolos coincide con M-K-M, la señal de salida valdrá '1', esta señal será una entrada al autómata de control. El diagrama del autómata es igual al de antes solo que ahora la condición para entrar en el estado FIN es que la señal de entrada final valga '1'.

-Detector Fin

```
----------------------------------------------------------------------------------
-- DETECTORFIN
--
-- Tiene como entrada el codigo de cada simbolo que va saliendo,
-- Si la secuencia de los ultimos tres simbolos es
-- ESPACIO-K-ESPACIO
-- La salida FINAL se pone a '1'
--
----------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Deteccionfin is
 Port( CLK_1ms : in STD_LOGIC;
               CODIGO : in STD_LOGIC_VECTOR (7 downto 0); -- codigo morse obtenido
               VALID_DISP : in STD_LOGIC;
               FINAL : out STD_LOGIC );
end Deteccionfin;


architecture Deteccionfin of Deteccionfin is
SIGNAL QS0: STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL QS1: STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL QS2: STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL K: STD_LOGIC_VECTOR(23 DOWNTO 0);

begin
process(CLK_1ms)
begin
if (CLK_1ms'event and CLK_1ms='1') then
 if(VALID_DISP='1') then -- Se actualiza cuando llega un nuevo valor,
  QS0<=QS1;                                  -- al igual que el registro de desplazamiento
  QS1<=QS2;
  QS2<=CODIGO;
 end if;
end if;
end process;

K(23 downto 16) <= QS0;
K(15 downto 8) <= QS1;
K(7 downto 0) <=  QS2;

with K SELECT FINAL <= --Una ROM con una sola salida '1' y el resto '0'
'1' when "010110000111010001011000",-- Secuencia M-K-M
'0' when others;
end Deteccionfin;
```

-Autómata de Control

```
----------------------------------------------------------------------------------
-- AUTOMATA DE CONTROL
--
-- Genera una seal con 8 bits
```

```
-- Los primeros 3 bits numeran la cantidad de PUNTOS y RAYAS de cada palabra
-- Los ultimos 5 bits representan de izquierda a derecha cada simbolo,
-- un '0' sera un PUNTO y un '1' sera una RAYA.
-- Cada palabra acaba cuando el automata alcanza el estado ESPACIO,
-- que es cuando se valida.
----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;

use IEEE.STD_LOGIC_unsigned.ALL;

entity aut_control is
Port( CLK_1ms : in STD_LOGIC; -- reloj
              VALID : in STD_LOGIC; -- entrada de dato vlido
              DATO : in STD_LOGIC; -- dato (0 o 1)
              C0 : in STD_LOGIC; -- resultado comparador de ceros
              C1 : in STD_LOGIC; -- resultado comparador de unos
              CS : in STD_LOGIC; -- resultado comparador de espacios
              RESET0 : in STD_LOGIC;
              PULSADOR : in STD_LOGIC;
              FINAL : in STD_LOGIc;
              CODIGO : out STD_LOGIC_VECTOR (7 downto 0); -- codigo morse obtenido
              VALID_DISP : out STD_LOGIC;
              LED : out STD_LOGIC;
              CONTADOR : out STD_LOGIC_VECTOR(8 downto 0));
end aut_control;

architecture a_aut_control of aut_control is
type STATE_TYPE is (ESPACIO,RESET,SIMBOLO,ESPERA, SEPARADOR,SEPARADOR_VALID,FIN);
signal ST : STATE_TYPE := RESET;
signal s_ncod : STD_LOGIC_VECTOR (2 downto 0):= "000";
signal s_cod : STD_LOGIC_VECTOR (4 downto 0):= "00000";
signal contador0: STD_LOGIC_VECTOR(8 downto 0) := "000000000";
signal n : INTEGER range 0 to 4;

begin
process (CLK_1ms, RESET0)
 begin
 if (RESET0 = '1') then --RESET asincrono
       contador0<= "000000000"; --Resetea el contador de palabras
       ST<=RESET; -- Pone el automata listo para recibir un nuevo simbolo

 elsif (CLK_1ms'event and CLK_1ms='1') then
 case ST is
 when SIMBOLO => --Se ha recibido un punto o una raya
       s_ncod<=s_ncod+1; --Suma 1 al numero de datos recibidos dentro de la mismo simbolo
       s_cod(n)<=C1; -- El resultado del comparador indica si el simbolo punto o raya
       n<=n-1; --Todo simbolo no puede tener mas de 5 datos
       ST<=ESPERA;

 when ESPERA => -- Esperando un nuevo simbolo
       if (valid='1' and dato='1') then -- Si se ha recibido un dato
              ST<=SIMBOLO;
       elsif(valid='1' and dato='0' and C0='0') then -- Si no se ha recibido nada
              ST<=ESPERA;
       elsif(valid='1' and dato='0' and C0='1') then -- Si el simbolo ha terminado
              ST<=ESPACIO;
       end if;

 when ESPACIO => -- VALIDA EL SIMBOLO
       -- Detector de FIN de mensaje basico
       -- if(s_ncod="011" and s_cod="10100") then
       --      ST<=FIN;
       if (CS='1') then -- Si el espacio que ha acabado el dato es de 700ms
              ST<=SEPARADOR;
       else
              ST<=RESET; -- Preparacion para recibir un nuevo simbolo
       end if;

 when SEPARADOR => -- Se ha acabado la palabra
       s_ncod<="010"; -- Letra M (Display apagado)
       s_cod<="11000";
       contador0<= contador0 + 1; -- Suma 1 al numero de palabras
       ST<= SEPARADOR_VALID;
```

```
 when SEPARADOR_VALID=> -- Valida el simbolo de SEPARADOR
       ST<= RESET;

 when RESET => --Prepara el automata para recibir un nuevo simbolo
       n <= 4;
       s_ncod<="000";
       s_cod<="00000";

       if(FINAL='1') then -- Si el mensaje ha terminado
              ST<= FIN;
       elsif (VALID='1' and dato='1') then ---Si se ha recibido un simbolo
              ST<=SIMBOLO;
       else
              ST<=RESET;
 end if;

 when FIN =>
       n <= 4;
       s_ncod<="000";
       s_cod<="00000";
       if (PULSADOR='1') then --Cuando el punsador valga '1'
              ST<=ESPERA;             -- se reactiva el automata.
       else
              ST <= FIN;
       end if;
 end case;
 end if;
end process;

-- PARTE COMBINACIONAL
LED <='1' when (ST=FIN) else '0';
VALID_DISP<='1' when (ST=ESPACIO or ST=SEPARADOR_VALID)  else '0';
CODIGO(4 downto 0)<= s_cod;
CODIGO(7 downto 5)<= s_ncod;
CONTADOR <= contador0;
end a_aut_control;
```

-Asociaciones

```
# Reloj principal del sistema
NET "CLK" LOC = "M6"; # Seal de reloj del sistema
# Conexiones de los DISPLAYS
NET "SEG7<0>" LOC = "L14"; # seal = CA
NET "SEG7<1>" LOC = "H12"; # Seal = CB
NET "SEG7<2>" LOC = "N14"; # Seal = CC
NET "SEG7<3>" LOC = "N11"; # Seal = CD
NET "SEG7<4>" LOC = "P12"; # Seal = CE
NET "SEG7<5>" LOC = "L13"; # Seal = CF
NET "SEG7<6>" LOC = "M12"; # Seal = CG
# Seales de activacin de los displays
NET "AN<0>" LOC = "K14"; # Activacin del display 0 = AN0
NET "AN<1>" LOC = "M13"; # Activacin del display 1 = AN1
NET "AN<2>" LOC = "J12"; # Activacin del display 2 = AN2
NET "AN<3>" LOC = "F12"; # Activacin del display 3 = AN3
#Entrada externa donde se conecta la seal entrante
NET "LIN" LOC = "B2"; # Entrada de datos
#Salidas para generar la seal analgica (no tocar estas lneas)
NET "SPI_CLK" LOC = "A9"; # Salida externa terminal 14
NET "SPI_DIN" LOC = "B9"; # Salida externa terminal 15
NET "SPI_CS1" LOC = "C9"; # Salida externa terminal 17
#Entradas para arrancar y parar la seal analgica (no tocar estas lneas)
NET "BTN_START" LOC = "G12"; # Entrada externa BTN0
NET "BTN_STOP" LOC = "C11"; # Entrada externa BTN1

NET "RESET0" LOC = "M4"; #RESET
NET "PULSADOR" LOC = "A7"; #Reactivar recepcion de mensaje
NET "LED" LOC = "M5"; #Led que indica fin de mensaje

NET "SELECTOR" LOC = "P11"; #Recepcion del mensaje o mostrar numero de palabras
```

5.5 Contador de Palabras

Esta mejora tiene como objetivo que cuando se accione uno de los interruptores, en vez de mostrar la recepción del mensaje muestre el número de palabras que lleva el mensaje. Para ello, el primer paso será introducir una señal SELECTOR que actuará como bit de control de un MUX que tiene como entradas los diferentes dígitos y un número. Este MUX estará dentro del registro de desplazamiento, por ello la nueva señal SELECTOR se incluirá en las asociaciones y en los módulos Main, Receptor, Visualización y Registro de Desplazamiento. Lo primero que habrá que necesitar será una nueva señal de salida del autómata de control que sea un contador del número de palabras que llevamos, este contador aumenta su valor '1' cada vez que el autómata pasa por el estado SEPARADOR (que indica que una palabra a acabado, mejora 1) el siguiente paso es adaptar esta señal contador para que pueda usarse en los displays es decir dividir la señal contador en unidades, decenas y centenas o lo que es lo mismo, en código BCD, debido a la complejidad de este módulo se ha usado un módulo VHDL encontrado en internet y que se encuentra en referencias.
En el registro de desplazamiento habrá que usar el selector como bit de control de un MUX
En la ROM de descodificación de los displays habrá que incorporar la descodificación de los bits de 0 a 9.
Si cuando se está mostrando el número de palabras se pulsa el botón reset el contador pasa a valer 0.

-Autómata de Control

```
----------------------------------------------------------------------------------
-- AUTOMATA DE CONTROL
--
-- Genera una seal con 8 bits
-- Los primeros 3 bits numeran la cantidad de PUNTOS y RAYAS de cada palabra
-- Los ultimos 5 bits representan de izquierda a derecha cada simbolo,
-- un '0' sera un PUNTO y un '1' sera una RAYA.
-- Cada palabra acaba cuando el automata alcanza el estado ESPACIO,
-- que es cuando se valida.
----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;

use IEEE.STD_LOGIC_unsigned.ALL;

entity aut_control is
Port( CLK_1ms : in STD_LOGIC; -- reloj
               VALID : in STD_LOGIC; -- entrada de dato vlido
               DATO : in STD_LOGIC; -- dato (0 o 1)
               C0 : in STD_LOGIC; -- resultado comparador de ceros
               C1 : in STD_LOGIC; -- resultado comparador de unos
               CS : in STD_LOGIC; -- resultado comparador de espacios
               RESET0 : in STD_LOGIC;
               PULSADOR : in STD_LOGIC;
               FINAL : in STD_LOGIc;
               CODIGO : out STD_LOGIC_VECTOR (7 downto 0); -- codigo morse obtenido
               VALID_DISP : out STD_LOGIC;
               LED : out STD_LOGIC;
               CONTADOR : out STD_LOGIC_VECTOR(8 downto 0));
end aut_control;

architecture a_aut_control of aut_control is
type STATE_TYPE is (ESPACIO,RESET,SIMBOLO,ESPERA, SEPARADOR,SEPARADOR_VALID,FIN);
signal ST : STATE_TYPE := RESET;
signal s_ncod : STD_LOGIC_VECTOR (2 downto 0):= "000";
signal s_cod : STD_LOGIC_VECTOR (4 downto 0):= "00000";
signal contador0: STD_LOGIC_VECTOR(8 downto 0) := "000000000";
signal n : INTEGER range 0 to 4;

begin
process (CLK_1ms, RESET0)
 begin
 if (RESET0 = '1') then --RESET asincrono
       contador0<= "000000000"; --Resetea el contador de palabras
       ST<=RESET; -- Pone el automata listo para recibir un nuevo simbolo

 elsif (CLK_1ms'event and CLK_1ms='1') then
```

```
 case ST is
 when SIMBOLO => --Se ha recibido un punto o una raya
       s_ncod<=s_ncod+1; --Suma 1 al numero de datos recibidos dentro de la mismo simbolo
       s_cod(n)<=C1; -- El resultado del comparador indica si el simbolo punto o raya
       n<=n-1; --Todo simbolo no puede tener mas de 5 datos
       ST<=ESPERA;

 when ESPERA => -- Esperando un nuevo simbolo
       if (valid='1' and dato='1') then -- Si se ha recibido un dato
              ST<=SIMBOLO;
       elsif(valid='1' and dato='0' and C0='0') then -- Si no se ha recibido nada
              ST<=ESPERA;
       elsif(valid='1' and dato='0' and C0='1') then -- Si el simbolo ha terminado
              ST<=ESPACIO;
       end if;

 when ESPACIO => -- VALIDA EL SIMBOLO
       -- Detector de FIN de mensaje basico
       -- if(s_ncod="011" and s_cod="10100") then
       --      ST<=FIN;
       if (CS='1') then -- Si el espacio que ha acabado el dato es de 700ms
              ST<=SEPARADOR;
       else
              ST<=RESET; -- Preparacion para recibir un nuevo simbolo
       end if;

 when SEPARADOR => -- Se ha acabado la palabra
       s_ncod<="010"; -- Letra M (Display apagado)
       s_cod<="11000";
       contador0<= contador0 + 1; -- Suma 1 al numero de palabras
       ST<= SEPARADOR_VALID;

 when SEPARADOR_VALID=> -- Valida el simbolo de SEPARADOR
       ST<= RESET;

 when RESET => --Prepara el automata para recibir un nuevo simbolo
       n <= 4;
       s_ncod<="000";
       s_cod<="00000";

       if(FINAL='1') then -- Si el mensaje ha terminado
              ST<= FIN;
       elsif (VALID='1' and dato='1') then ---Si se ha recibido un simbolo
              ST<=SIMBOLO;
       else
              ST<=RESET;
 end if;

 when FIN =>
       n <= 4;
       s_ncod<="000";
       s_cod<="00000";
       if (PULSADOR='1') then --Cuando el punsador valga '1'
              ST<=ESPERA;             -- se reactiva el automata.
       else
              ST <= FIN;
       end if;
 end case;
 end if;
end process;

-- PARTE COMBINACIONAL
LED <='1' when (ST=FIN) else '0';
VALID_DISP<='1' when (ST=ESPACIO or ST=SEPARADOR_VALID)  else '0';
CODIGO(4 downto 0)<= s_cod;
CODIGO(7 downto 5)<= s_ncod;
CONTADOR <= contador0;
end a_aut_control;
```

-NumbertoBCD

```
------------------------------------------------
-- BCD TRANSFORMER
-- Transforma de 0 a 511 de binario a bcd
--
```

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

entity bintobcd is
  PORT( num_bin: in  STD_LOGIC_VECTOR(8 downto 0);
                centenas: out STD_LOGIC_VECTOR(7 downto 0);
                decenas: out STD_LOGIC_VECTOR(7 downto 0);
                unidades: out STD_LOGIC_VECTOR(7 downto 0));
end bintobcd;

architecture Behavioral of bintobcd is
SIGNAL num_bcd: STD_LOGIC_VECTOR(10 downto 0);
begin
   proceso_bcd: process(num_bin)
   variable z: STD_LOGIC_VECTOR(19 downto 0);
    begin
        -- Inicializacin de datos en cero.
        z := (others => '0');
        -- Se realizan los primeros tres corrimientos.
        z(11 downto 3) := num_bin;
        for i in 0 to 5 loop
            -- Unidades (4 bits).
            if z(12 downto 9) > 4 then
                z(12 downto 9) := z(12 downto 9) + 3;
            end if;
            -- Decenas (4 bits).
            if z(16 downto 13) > 4 then
                z(16 downto 13) := z(16 downto 13) + 3;
            end if;
            -- Centenas (3 bits).
            if z(19 downto 17) > 4 then
                z(19 downto 17) := z(19 downto 17) + 3;
            end if;
            -- Corrimiento a la izquierda.
            z(19 downto 1) := z(18 downto 0);
        end loop;
        -- Pasando datos de variable Z, correspondiente a BCD.
        num_bcd <= z(19 downto 9);
    end process;

 -- Parte combinacional
 centenas(7 downto 3)<="00000";
 decenas(7 downto 4)<="0000";
 unidades(7 downto 4)<="0000";
 centenas(2 downto 0)<=num_bcd(10 downto 8);
 decenas(3 downto 0)<=num_bcd(7 downto 4);
 unidades(3 downto 0)<=num_bcd(3 downto 0);
end Behavioral;
```

-Registro de Desplazamiento

```
----------------------------------------------------------------------------------
-- REGISTRO DESPLAZAMIENTO
--
-- Este modulo desplaza las senales
-- eliminando la mas antigua y andiendo
-- la nueva senal cada vez que EN vale '1'
-- que es cuando ha llegado un simbolo nuevo
--
----------------------------------------------------------------------------------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_arith.ALL;

use IEEE.STD_LOGIC_unsigned.ALL;

entity rdesp_disp is
Port( CLK_1ms : in STD_LOGIC; -- entrada de reloj
              EN : in STD_LOGIC; -- enable
              E : in STD_LOGIC_VECTOR (7 downto 0); -- entrada de datos
              RESET0 : in STD_LOGIC;
              SELECTOR : in STD_LOGIC; --Selecciona entre el mensaje y el numero de palabras
              CENTENAS : in STD_LOGIC_VECTOR(7 downto 0);
```

```
               DECENAS : in STD_LOGIC_VECTOR(7 downto 0);
               UNIDADES : in STD_LOGIC_VECTOR(7 downto 0);
               Q0 : out STD_LOGIC_VECTOR (7 downto 0); -- salida Q0
               Q1 : out STD_LOGIC_VECTOR (7 downto 0); -- salida Q1
               Q2 : out STD_LOGIC_VECTOR (7 downto 0); -- salida Q2
               Q3 : out STD_LOGIC_VECTOR (7 downto 0)); -- salida Q3
end rdesp_disp;


architecture rdesp_disp of rdesp_disp is
SIGNAL QS0: STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL QS1: STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL QS2: STD_LOGIC_VECTOR(7 DOWNTO 0);
SIGNAL QS3: STD_LOGIC_VECTOR(7 DOWNTO 0);

begin
process(CLK_1ms,RESET0) -- Reset asincrono
begin
 if(RESET0='1') then -- Cuando el reset esta a '1'
  QS0<="01110100";   -- ponemos un simbolo cuya sepresentacion sean
  QS1<="01110100";   -- todos los displays apagados.
  QS2<="01110100";
  QS3<="01110100";
 elsif (CLK_1ms'event and CLK_1ms='1') then
       if(EN='1') then -- Cuando llega un nuevo simbolo
               QS0<=QS1;    -- se hace el desplazamiento
               QS1<=QS2;
               QS2<=QS3;
               QS3<=E;   --Introduce el nuevo simbolo
       end if;
 end if;
end process;
-- MUX con SELECTOR como bit de control
Q0<= QS0 when (SELECTOR='1') else "01110100"; -- Si no esta mostrando el mensaje esta apagado
Q1<= QS1 when (SELECTOR='1') else CENTENAS;
Q2<= QS2 when (SELECTOR='1') else DECENAS;
Q3<= QS3 when (SELECTOR='1') else UNIDADES;
end rdesp_disp;
```

ANEXO I

Un anexo por cada contenido que no vaya en secciones anteriores

ANEXO II

...

REFERENCIAS

Módulo Binary to BCD
http://www.estadofinito.com/binario-bcd-7seg/