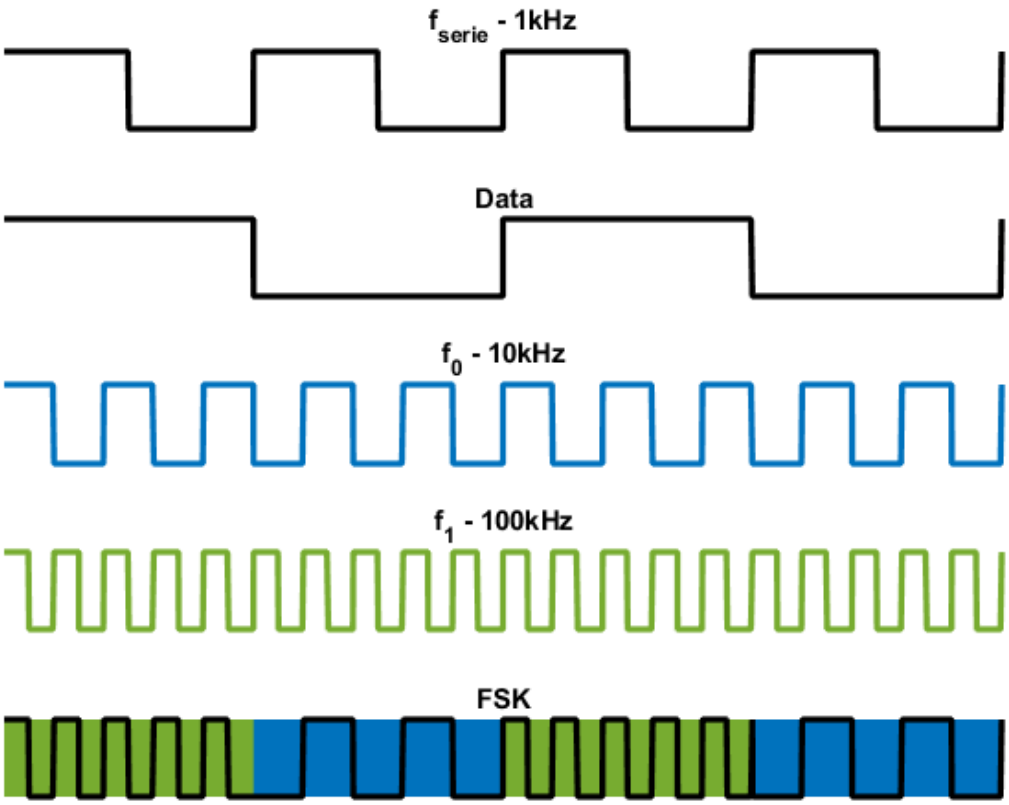
Asignatura: Electrónica Digital (GITI)
Parte 2 – VHDL
Publicación de preactas: 24/07/2020

Fecha: 10/07/2020
Convocatoria: Julio
Revisión: 27/07/2020, 9h

CUESTIÓN ÚNICA (10 puntos)

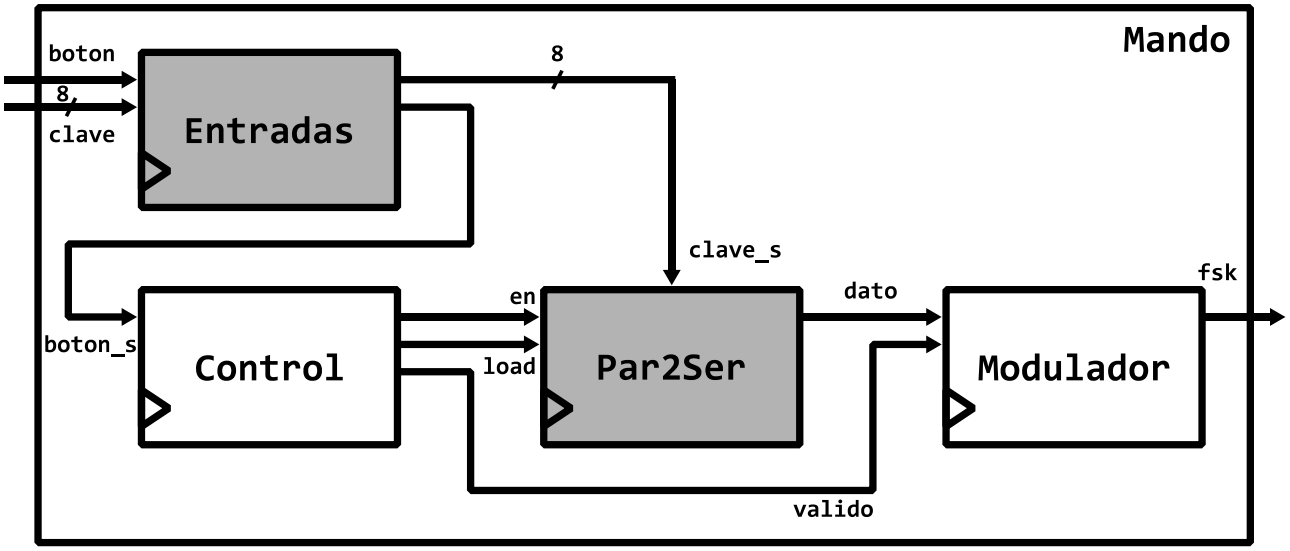
Se pretende diseñar el circuito de control de un mando a distancia universal para puertas de garaje. La funcionalidad del circuito es la que sigue:

- Cuando se pulsa el botón del mando, se deberá almacenar el valor actual de la clave, que a su vez estará indicado por el valor de un *switch* múltiple de 8 bits.
- Una vez almacenado, se deberá transmitir el valor de la clave a través de una línea serie y con una frecuencia base de 1 kHz.
- Dicha línea serie será modulada usando FSK (*Frequency-Shift Keying*), un tipo de modulación de señales que asigna una frecuencia f_0 (10 kHz) al valor '0' y otra frecuencia f_1 (100 kHz) al valor '1' (ver gráficas más abajo).
- Todas las señales de entrada deberán estar debidamente sincronizadas, y se añadirá un antirrebotes (> 2 ms) al botón, con el fin de generar una señal de 1 ciclo de reloj de duración.



En la figura anterior se puede apreciar cómo funciona la modulación FSK: la señal de datos (segunda gráfica), transmitida con una frecuencia base de 1 kHz (primera gráfica), es empleada para seleccionar una de las dos frecuencias portadoras (10 kHz, tercera gráfica; 100 kHz, cuarta gráfica). De esta manera se obtiene como salida la señal modulada (quinta gráfica).

Tras un primer análisis, se ha diseñado el siguiente diagrama de bloques para el circuito:



- El **bloque de entrada** se encarga de sincronizar las señales externas y de filtrar rebotes.
- El **bloque de conversión paralelo a serie** almacena la clave durante la transmisión en un registro de desplazamiento con una señal de habilitación y otra señal de carga en paralelo.
- El **bloque modulador** aplica la modulación FSK a una entrada de datos, pero sólo cuando una señal de control indica que el dato es válido (si no lo es, la salida tiene que valer '0').
- El **bloque de control**, que incluye una máquina de estados y cierta lógica adicional, se usa para gobernar el funcionamiento del circuito.

Se pide:

a) Código VHDL (sólo arquitectura) del **bloque modulador (4 puntos)**.
b) Código VHDL (sólo arquitectura) del **bloque de control (4 puntos)**.
c) Código VHDL (sólo arquitectura) del **bloque *top*** (descripción estructural) **(2 puntos)**.

Notas:

- Aunque no se ha representado, **todos los bloques** del diagrama son secuenciales y síncronos, por lo que **comparten las mismas señales de reloj** (`clk`) **y *reset*** (`reset`). La frecuencia de reloj del sistema es de 100 MHz.
- Se asume que los bloques resaltados en gris en el diagrama han sido diseñados previamente y se dispone de su código VHDL.
- Para que la máquina de estados del bloque de control no sea excesivamente compleja de diseñar y describir, se recomienda utilizar un contador de 0 a 7 con señal de *overflow* como parte de la lógica adicional.

Duración del examen: 1 hora

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity modulador is
    port (
        clk    : in  std_logic;
        reset  : in  std_logic;
        dato   : in  std_logic;
        valido : in  std_logic;
        fsk    : out std_logic
    );
end modulador;

architecture behavioral of modulador is

    -- Divisores de frecuencia
    constant C_MAX_100kHz : integer := (100*10**6)/(100*10**3);
    constant C_MAX_10kHz  : integer := 10;
    signal cnt_100kHz  : integer range 0 to C_MAX_100kHz-1;
    signal ovf_100kHz  : std_logic;
    signal cnt_10kHz   : integer range 0 to C_MAX_10kHz-1;
    signal ovf_10kHz   : std_logic;

    -- Generación de portadoras
    signal wave_100kHz : std_logic;
    signal wave_10kHz  : std_logic;

begin

    ---------------------------------------------
    -- Divisores de frecuencia (10kHz, 100kHz) --
    ---------------------------------------------

    process(clk, reset)
    begin
        if reset = '1' then
            cnt_100kHz <= 0;
        elsif clk'event and clk = '1' then
            if cnt_100kHz = C_MAX_100kHz-1 then
                cnt_100kHz <= 0;
            else
                cnt_100kHz <= cnt_100kHz + 1;
            end if;
        end if;
    end process;

    ovf_100kHz <= '1' when cnt_100kHz = C_MAX_100kHz-1 else '0';

    process(clk, reset)
    begin
        if reset = '1' then
            cnt_10kHz <= 0;
        elsif clk'event and clk = '1' then
            if ovf_100kHz = '1' then
                if cnt_10kHz = C_MAX_10kHz-1 then
                    cnt_10kHz <= 0;
                else
                    cnt_10kHz <= cnt_10kHz + 1;
                end if;
            end if;
        end if;
    end process;

    ovf_10kHz <= '1' when ovf_100kHz = '1' and cnt_10kHz = C_MAX_10kHz-1 else '0';

    ------------------------------
    -- Generación de portadoras --
    ------------------------------

    process(clk, reset)
    begin
        if reset = '1' then
```

```
74              wave_100kHz <= '0';
75              wave_10kHz <= '0';
76          elsif clk'event and clk = '1' then
77              if ovf_100kHz = '1' then
78                  wave_100kHz <= not wave_100kHz;
79              end if;
80              if ovf_10kHz = '1' then
81                  wave_10kHz <= not wave_10kHz;
82              end if;
83          end if;
84      end process;
85
86      --------------------------
87      -- Generación de salida --
88      --------------------------
89
90      process(clk, reset)
91      begin
92          if reset = '1' then
93              fsk <= '0';
94          elsif clk'event and clk = '1' then
95              if valido = '1' then
96                  if dato = '1' then
97                      fsk <= wave_100kHz;
98                  else
99                      fsk <= wave_10kHz;
100                 end if;
101             else
102                 fsk <= '0';
103             end if;
104         end if;
105     end process;
106
107     -- NOTA: esto podría hacerse con lógica combinacional;
108     --       al ser salida final del circuito la registramos.
109     --
110     -- fsk <= wave_100kHz when valido = '1' and dato = '1' else
111     --        wave_10kHz when valido = '1' and dato = '0' else
112     --        '0';
113
114 end behavioral;
115
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity control is
    port (
        clk    : in  std_logic;
        reset  : in  std_logic;
        boton  : in  std_logic;
        load   : out std_logic;
        en     : out std_logic;
        valido : out std_logic
    );
end control;

architecture behavioral of control is

    -- Divisor de frecuencia (1kHz)
    constant C_MAX_1kHz : integer := (100*10**6)/(10**3);
    signal cnt_1kHz : integer range 0 to C_MAX_1kHz-1;
    signal en_1kHz  : std_logic;
    signal ovf_1kHz : std_logic;

    -- Contador de dígitos
    signal digito     : unsigned(2 downto 0); -- Hay 8 dígitos, cuento de 0 a 7
    signal ovf_digito : std_logic;

    -- FSM de control
    type state_t is (S_WAIT, S_SEND);
    signal state : state_t;

begin

    ----------------------------------
    -- Divisor de frecuencia (1kHz) --
    ----------------------------------

    process(clk, reset)
    begin
        if reset = '1' then
            cnt_1kHz <= 0;
        elsif clk'event and clk = '1' then
            if en_1kHz = '1' then
                if cnt_1kHz = C_MAX_1kHz-1 then
                    cnt_1kHz <= 0;
                else
                    cnt_1kHz <= cnt_1kHz + 1;
                end if;
            end if;
        end if;
    end process;

    ovf_1kHz <= '1' when en_1kHz = '1' and cnt_1kHz = C_MAX_1kHz-1 else '0';

    -------------------------
    -- Contador de dígitos --
    -------------------------

    process(clk, reset)
    begin
        if reset = '1' then
            digito <= (others => '0');
        elsif clk'event and clk = '1' then
            if ovf_1kHz = '1' then
                if digito = 7 then
                    digito <= (others => '0');
                else
                    digito <= digito + 1;
                end if;
            end if;
        end if;
    end process;

```

```
74        ovf_digito <= '1' when ovf_1kHz = '1' and digito = 7 else '0';
75
76        --------------------
77        -- FSM de control --
78        --------------------
79
80        process(clk, reset)
81        begin
82            if reset = '1' then
83                state <= S_WAIT;
84            elsif clk'event and clk = '1' then
85                case state is
86                    when S_WAIT =>
87                        if boton = '1' then
88                            state <= S_SEND;
89                        end if;
90                    when S_SEND =>
91                        if ovf_digito = '1' then
92                            state <= S_WAIT;
93                        end if;
94                end case;
95            end if;
96        end process;
97
98        -- Control del temporizador (Moore)
99        en_1kHz <= '1' when state = S_SEND else '0';
100       -- Control de habilitación del registro de desplazamiento (Mealy)
101       en <= '1' when (state = S_WAIT and boton = '1') or (state = S_SEND and ovf_1kHz =
           '1') else '0';
102       -- Control de la carga del registro de desplazamiento (Mealy)
103       load <= '1' when state = S_WAIT and boton = '1' else '0';
104       -- Control del modulador (Moore)
105       valido <= en_1kHz;
106
107   end behavioral;
108
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity mando is
    port (
        clk   : in  std_logic;
        reset : in  std_logic;
        boton : in  std_logic;
        clave : in  std_logic_vector(7 downto 0);
        fsk   : out std_logic
    );
end mando;

architecture structural of mando is

    component entradas is
        port (
            clk     : in  std_logic;
            reset   : in  std_logic;
            boton   : in  std_logic;
            clave   : in  std_logic_vector(7 downto 0);
            boton_s : out std_logic;
            clave_s : out std_logic_vector(7 downto 0)
        );
    end component;

    component par2ser is
        port (
            clk   : in  std_logic;
            reset : in  std_logic;
            load  : in  std_logic;
            en    : in  std_logic;
            din   : in  std_logic_vector(7 downto 0);
            dout  : out std_logic
        );
    end component;

    component control is
        port (
            clk    : in  std_logic;
            reset  : in  std_logic;
            boton  : in  std_logic;
            load   : out std_logic;
            en     : out std_logic;
            valido : out std_logic
        );
    end component;

    component modulador is
        port (
            clk    : in  std_logic;
            reset  : in  std_logic;
            dato   : in  std_logic;
            valido : in  std_logic;
            fsk    : out std_logic
        );
    end component;

    -- Señales auxiliares
    signal boton_s : std_logic;
    signal clave_s : std_logic_vector(7 downto 0);
    signal load    : std_logic;
    signal enable  : std_logic;
    signal dato    : std_logic;
    signal valido  : std_logic;

begin

    entradas_i: entradas
    port map (
        clk     => clk,
        reset   => reset,
        boton   => boton,
```

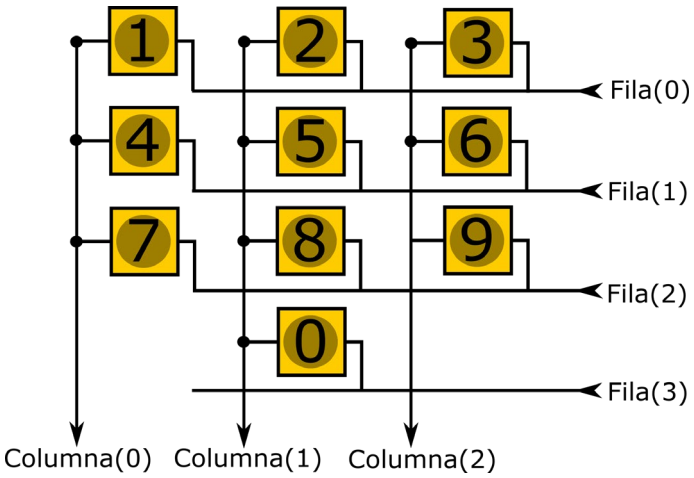
```
74          clave   => clave,
75          boton_s => boton_s,
76          clave_s => clave_s
77      );
78
79      par2ser_i: par2ser
80      port map (
81          clk   => clk,
82          reset => reset,
83          load  => load,
84          en    => enable,
85          din   => clave_s,
86          dout  => dato
87      );
88
89      control_i: control
90      port map (
91          clk    => clk,
92          reset  => reset,
93          boton  => boton_s,
94          load   => load,
95          en     => enable,
96          valido => valido
97      );
98
99      modulador_i: modulador
100     port map (
101         clk    => clk,
102         reset  => reset,
103         dato   => dato,
104         valido => valido,
105         fsk    => fsk
106     );
107
108 end structural;
109
```

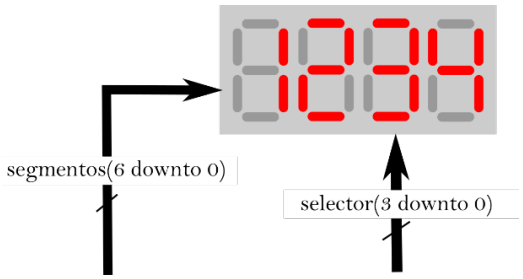
Asignatura: Electrónica Digital (GITI)
(**Prueba de Evaluación Continua**)

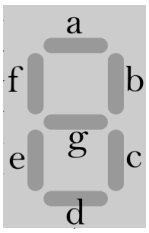
Fecha: 2/11/2019
Convocatoria: Diciembre

CUESTIÓN ÚNICA (10 puntos)

Se pretende diseñar el sistema de control digital de un nuevo terminal de comunicaciones. El sistema consta de un teclado numérico, con el que el usuario podrá introducir el número al que desea marcar, y un sistema de visualización, en el que se mostrará el número marcado.







El teclado numérico está dispuesto como una matriz de 4 filas y 3 columnas, sobre el que se disponen de los números del 0 al 9 (ver figura). Para la lectura del teclado es necesario poner un '1' en la señal del vector ***Fila*** correspondiente, de tal manera que si alguna de las teclas de la fila seleccionada está pulsada, el teclado devolverá un '1' en la salida ***Columna*** a la que está conectada la tecla. Es decir, cada tecla actúa como un interruptor que conecta una línea de fila con su línea de columna***.***

El sistema de visualización está formado por cuatro *displays* de siete segmentos, multiplexados en el tiempo. Todos los números estarán compuestos por 4 dígitos. El control del *display* se realizará mediante un vector de siete señales ***segmentos,*** donde ***segmentos***(6) corresponde con el segmento ***a***, hasta ***segmentos***(0), que corresponde con el segmento ***g*** (ver imagen)***.*** Se cuenta además con una señal ***selector*** de 4 bits que permite seleccionar el *display* activo, siendo ***selector(3)*** el correspondiente al display que mostrará el primer número introducido, hasta ***selector(0)***, que mostrará el último número.

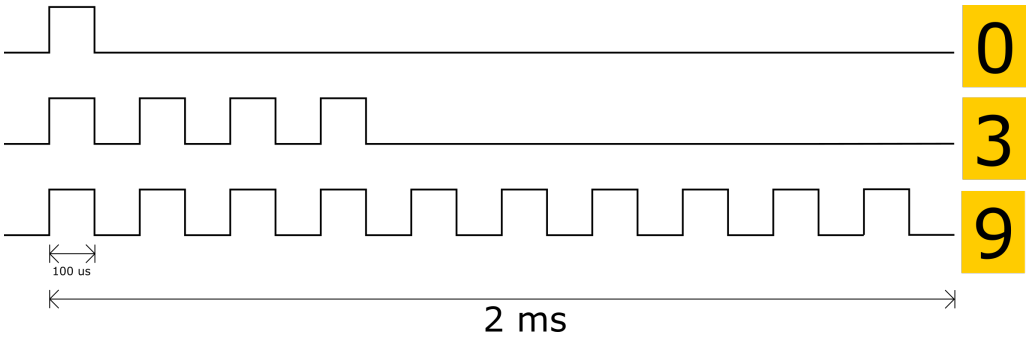
El sistema de control digital que se pretende diseñar constará de los siguientes bloques:

- **Bloque de Lectura de Teclado**: Deberá controlar la señal ***Fila*** realizando un barrido continuo sobre la misma, para detectar con la lectura de la señal ***Columna*** si alguna de las teclas está pulsada. Cuando detecte una pulsación, enviará el número correspondiente codificado en formato BCD por una salida de 4 bits denominada ***Digito***. Además, el módulo generará un pulso de duración un ciclo de reloj sobre una salida denominada ***Valido***, que indicará que se trata de un nuevo Dígito.

- **Bloque de Almacenamiento y Visualización**: Este bloque recibirá como entradas las señales ***Digito*** y ***Valido*** procedentes del bloque de lectura de teclado. Deberá ir almacenando en su interior los cuatro dígitos codificados en BCD. A la vez, los irá mostrando en el segmento

correspondiente del display. Se considerará una frecuencia de refresco de 4 KHz para los displays. Los displays se borrarán cuando se introduzca el primer dígito de un nuevo número.

- **Bloque de Generación de la Señal de llamada**: Recibirá también como entradas las señales ***Dígito*** y ***Válido*** procedentes del bloque de lectura de teclado. A partir de estas señales generará una señal **Marcado**, de un solo bit, sobre la que se irá codificando la señal de llamada, tal y como sigue. Cada número a enviar empezará siempre por un pulso a '1', seguido de un número de pulsos correspondiente al número a enviar. Cada pulso tendrá una duración de 100 µs, e irá seguido de un tiempo igual en el que la señal permanecerá necesariamente a '0'. El tiempo total para enviar cada dígito será por lo tanto de 2 ms, tal y como se muestra en los ejemplos de la imagen inferior. El número se va enviando sobre la línea, dígito a dígito, conforme se va recibiendo desde el bloque de lectura del teclado.



Se asume que las señales de los pulsadores son ideales, por lo que no tienen rebotes. Su duración se corresponderá con el tiempo que el usuario mantenga pulsada la tecla. Asumiremos que siempre se introducirán números de 4 dígitos, que el usuario no va a cometer errores (p.ej. pulsar dos botones a la vez), y que la frecuencia de reloj del sistema es de 125MHz.

Se pide:

a) Código VHDL (entidad y arquitectura) del bloque de lectura de teclado. **(3 puntos)**
b) Código VHDL (entidad y arquitectura) del bloque de Almacenamiento y visualización. **(2 puntos)**
c) Arquitectura del bloque de Generación de Señal de Llamada. **(3 puntos)**
d) Código del banco de pruebas (*testbench*) en el que se verifique un procedimiento completo de llamada sobre el sistema completo, incluido la introducción de los cuatro dígitos, de forma que permita comprobar la visualización y la generación de la señal de llamada. **(2puntos)**

Duración del examen: 2 horas

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity lectura_teclado is
    port (
        clk     : in  std_logic;
        reset   : in  std_logic;
        Fila    : out std_logic_vector(3 downto 0);
        Columna : in  std_logic_vector(2 downto 0);
        Digito  : out std_logic_vector(3 downto 0);
        Valido  : out std_logic
    );
end lectura_teclado;

architecture behavioral of lectura_teclado is

    -- Generacion de barrido de filas
    signal index : integer range 0 to 3;

    -- Registro del digito pulsado
    signal col_f0 : std_logic_vector(2 downto 0);
    signal col_f1 : std_logic_vector(2 downto 0);
    signal col_f2 : std_logic_vector(2 downto 0);
    signal col_f3 : std_logic_vector(2 downto 0);

begin

    -------------------------------------
    -- Generacion de barrido de filas --
    -------------------------------------

    process(clk,reset)
    begin
        if reset = '1' then
            index <= 0;
        elsif clk'event and clk = '1' then
            if index = 3 then
                index <= 0;
            else
                index <= index + 1;
            end if;
        end if;
    end process;

    with index select
        Fila <= "0001" when 0,
                "0010" when 1,
                "0100" when 2,
                "1000" when 3,
                "----" when others;

    ---------------------------------
    -- Registro del digito pulsado --
    ---------------------------------

    process(clk, reset)
    begin
        if reset = '1' then
            col_f0 <= (others => '0');
            col_f1 <= (others => '0');
            col_f2 <= (others => '0');
            col_f3 <= (others => '0');
        elsif clk'event and clk = '1' then
            case index is
                when 0 =>
                    col_f0 <= Columna;
                when 1 =>
                    col_f1 <= Columna;
                when 2 =>
                    col_f2 <= Columna;
```

```
                when 3 =>
                    col_f3 <= Columna;
                when others =>
                    -- No deberia suceder
            end case;
        end if;
    end process;

    ---------------------------
    -- Generacion de salidas --
    ---------------------------

    Digito <= "0001" when index = 0 and Columna = "100" else
              "0010" when index = 0 and Columna = "010" else
              "0011" when index = 0 and Columna = "001" else
              "0100" when index = 1 and Columna = "100" else
              "0101" when index = 1 and Columna = "010" else
              "0110" when index = 1 and Columna = "001" else
              "0111" when index = 2 and Columna = "100" else
              "1000" when index = 2 and Columna = "010" else
              "1001" when index = 2 and Columna = "001" else
              "0000" when index = 3 and Columna = "010" else
              "1111"; -- La simulacion se rompe con "----"

    Valido <= '1' when Columna /= "000" and ((index = 0 and Columna /= col_f0) or
    (index = 1 and Columna /= col_f1) or (index = 2 and Columna /= col_f2) or (index
    = 3 and Columna /= col_f3)) else '0';

end behavioral;

```

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity almacenamiento_visualizacion is
    port (
        clk       : in  std_logic;
        reset     : in  std_logic;
        Digito    : in  std_logic_vector(3 downto 0);
        Valido    : in  std_logic;
        segmentos : out std_logic_vector(6 downto 0);
        selector  : out std_logic_vector(3 downto 0)
    );
end almacenamiento_visualizacion;

architecture behavioral of almacenamiento_visualizacion is

    -- Almacenamiento de digitos
    signal index : integer range 0 to 3;
    signal digito_0 : std_logic_vector(3 downto 0);
    signal digito_1 : std_logic_vector(3 downto 0);
    signal digito_2 : std_logic_vector(3 downto 0);
    signal digito_3 : std_logic_vector(3 downto 0);

    -- Control 7 segmentos
    constant C_MAX_4kHz : integer := (125*10**6)/(4*10**3);
    signal cnt_4kHz   : integer range 0 to C_MAX_4kHz-1;
    signal ovf_4kHz   : std_logic;
    signal cnt_mux    : unsigned(1 downto 0);
    signal digito_mux : std_logic_vector(3 downto 0);

begin

    -------------------------------
    -- Almacenamiento de digitos --
    -------------------------------

    process(clk,reset)
    begin
        if reset = '1' then
            index <= 0;
            digito_0 <= (others => '1');
            digito_1 <= (others => '1');
            digito_2 <= (others => '1');
            digito_3 <= (others => '1');
        elsif clk'event and clk = '1' then
            if Valido = '1' then
                -- Actualizacion del indice del digito
                if index = 3 then
                    index <= 0;
                else
                    index <= index + 1;
                end if;
                -- Registro del digito
                case index is
                    when 0 =>
                        digito_0 <= Digito;
                        digito_1 <= (others => '1');
                        digito_2 <= (others => '1');
                        digito_3 <= (others => '1');
                    when 1 =>
                        digito_1 <= Digito;
                    when 2 =>
                        digito_2 <= Digito;
                    when 3 =>
                        digito_3 <= Digito;
                    when others =>
                        -- No deberia suceder
                end case;
            end if;
        end if;
```

```
72      end process;
73
74      -------------------------
75      -- Control 7 segmentos --
76      -------------------------
77
78      -- Divisor de frecuencia 4kHz (parte secuencial)
79      process(clk,reset)
80      begin
81          if reset = '1' then
82              cnt_4kHz <= 0;
83          elsif clk'event and clk = '1' then
84              if cnt_4kHz = C_MAX_4kHz-1 then
85                  cnt_4kHz <= 0;
86              else
87                  cnt_4kHz <= cnt_4kHz + 1;
88              end if;
89          end if;
90      end process;
91
92      -- Divisor de frecuencia 4kHz (parte combinacional)
93      ovf_4kHz <= '1' when cnt_4kHz = C_MAX_4kHz-1 else '0';
94
95      -- Contador auxiliar 0-3
96      process(clk,reset)
97      begin
98          if reset = '1' then
99              cnt_mux <= (others => '0');
100         elsif clk'event and clk = '1' then
101             if ovf_4kHz = '1' then
102                 if cnt_mux = 3 then
103                     cnt_mux <= (others => '0');
104                 else
105                     cnt_mux <= cnt_mux + 1;
106                 end if;
107             end if;
108         end if;
109     end process;
110
111     -- Selector de display
112     with cnt_mux select
113         selector <= "1000" when "00",
114                     "0100" when "01",
115                     "0010" when "10",
116                     "0001" when "11",
117                     "----" when others;
118
119     -- Multiplexor de digitos
120     with cnt_mux select
121         digito_mux <= digito_0 when "00",
122                       digito_1 when "01",
123                       digito_2 when "10",
124                       digito_3 when "11",
125                       "----" when others;
126
127     -- Decodificador BCD 7 segmentos
128     with digito_mux select
129         segmentos <= "1111110" when "0000",
130                      "0110000" when "0001",
131                      "1101101" when "0010",
132                      "1111001" when "0011",
133                      "0110011" when "0100",
134                      "1011011" when "0101",
135                      "1011111" when "0110",
136                      "1110000" when "0111",
137                      "1111111" when "1000",
138                      "1110011" when "1001",
139                      "0000000" when others;
140
141 end behavioral;
142
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity top is
    port (
        clk       : in  std_logic;
        reset     : in  std_logic;
        Fila      : out std_logic_vector(3 downto 0);
        Columna   : in  std_logic_vector(2 downto 0);
        segmentos : out std_logic_vector(6 downto 0);
        selector  : out std_logic_vector(3 downto 0);
        Marcado   : out std_logic
    );
end top;

architecture behavioral of top is

    -- Lectura de teclado
    component lectura_teclado is
        port (
            clk     : in  std_logic;
            reset   : in  std_logic;
            Fila    : out std_logic_vector(3 downto 0);
            Columna : in  std_logic_vector(2 downto 0);
            Digito  : out std_logic_vector(3 downto 0);
            Valido  : out std_logic
        );
    end component;

    -- Almacenamiento y visualizacion
    component almacenamiento_visualizacion is
        port (
            clk       : in  std_logic;
            reset     : in  std_logic;
            Digito    : in  std_logic_vector(3 downto 0);
            Valido    : in  std_logic;
            segmentos : out std_logic_vector(6 downto 0);
            selector  : out std_logic_vector(3 downto 0)
        );
    end component;

    -- Señales de conexion
    signal Digito : std_logic_vector(3 downto 0);
    signal Valido : std_logic;

    -- Temporizador 100us
    constant C_MAX_10kHz : integer := (125*10**6)/(10*10**3);
    signal cnt_10kHz : integer range 0 to C_MAX_10kHz-1;
    signal en_10kHz  : std_logic;
    signal ovf_10kHz : std_logic;

    -- FSM control
    type state_t is (S_WAIT, S_DIAL);
    signal state : state_t;
    signal cnt   : unsigned(4 downto 0);
    signal send  : std_logic_vector(3 downto 0);
    signal aux   : std_logic;

begin

    -- Lectura de teclado
    comp_1: lectura_teclado
    port map (
        clk     => clk,
        reset   => reset,
        Fila    => Fila,
        Columna => Columna,
        Digito  => Digito,
        Valido  => Valido
    );
```

```vhdl

    -- Almacenamiento y visualizacion
    comp_2: almacenamiento_visualizacion
    port map (
        clk       => clk,
        reset     => reset,
        Digito    => Digito,
        Valido    => Valido,
        segmentos => segmentos,
        selector  => selector
    );

    -----------------------------------
    -- Generacion de señal de marcado --
    -----------------------------------

    -- Temporizador 100us (parte secuencial)
    process(clk,reset)
    begin
        if reset = '1' then
            cnt_10kHz <= 0;
        elsif clk'event and clk = '1' then
            if en_10kHz = '1' then
                if cnt_10kHz = C_MAX_10kHz-1 then
                    cnt_10kHz <= 0;
                else
                    cnt_10kHz <= cnt_10kHz + 1;
                end if;
            end if;
        end if;
    end process;

    -- Temporizador 100us (parte combinacional)
    ovf_10kHz <= '1' when cnt_10kHz = C_MAX_10kHz-1 and en_10kHz = '1' else '0';

    -- Generador de tonos
    process(clk,reset)
    begin
        if reset = '1' then
            aux <= '0';
            cnt <= (others => '0');
        elsif clk'event and clk = '1' then
            if ovf_10kHz = '1' then
                -- Generacion de señal periodica
                aux <= not aux;
                -- Contador de ciclos
                if cnt = 19 then
                    cnt <= (others => '0');
                else
                    cnt <= cnt + 1;
                end if;
            end if;
        end if;
    end process;

    -- Asignacion de salida
    Marcado <= aux when cnt < (2*unsigned(send) + 2) else '0';

    -- FSM (ecuaciones de estado)
    process(clk,reset)
    begin
        if reset = '1' then
            state <= S_WAIT;
            send <= (others => '0');
        elsif clk'event and clk = '1' then
            case state is
                when S_WAIT =>
                    if Valido = '1' then
                        state <= S_DIAL;
                        send <= Digito;
                    end if;
```

```
143                 when S_DIAL =>
144                     if cnt = 19 and ovf_10kHz = '1' then
145                         state <= S_WAIT;
146                     end if;
147             end case;
148         end if;
149     end process;
150 
151     -- FSM (ecuaciones de salida)
152     en_10kHz <= '1' when state = S_DIAL else '0';
153 
154 end behavioral;
155 
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity top_tb is
end top_tb;

architecture testbench of top_tb is

    -- Unit Under Test
    component top is
        port (
            clk       : in  std_logic;
            reset     : in  std_logic;
            Fila      : out std_logic_vector(3 downto 0);
            Columna   : in  std_logic_vector(2 downto 0);
            segmentos : out std_logic_vector(6 downto 0);
            selector  : out std_logic_vector(3 downto 0);
            Marcado   : out std_logic
        );
    end component;

    -- I/O ports
    signal clk       : std_logic;
    signal reset     : std_logic;
    signal Fila      : std_logic_vector(3 downto 0);
    signal Columna   : std_logic_vector(2 downto 0);
    signal segmentos : std_logic_vector(6 downto 0);
    signal selector  : std_logic_vector(3 downto 0);
    signal Marcado   : std_logic;

    -- Clock
    constant clk_period : time := 8ns;

begin

    uut: top
    port map (
        clk       => clk,
        reset     => reset,
        Fila      => Fila,
        Columna   => Columna,
        segmentos => segmentos,
        selector  => selector,
        Marcado   => Marcado
    );

    -- Generacion de reloj
    process
    begin
        clk <= '1';
        wait for clk_period/2;
        clk <= '0';
        wait for clk_period/2;
    end process;

    -- Generacion de estimulos
    process
    begin
        -- Reseteamos el sistema
        reset <= '1';
        Columna <= "000";
        wait for clk_period*100;
        reset <= '0';
        wait for clk_period*10;

        -- Sincronizamos con primera fila
        wait until Fila = "0001";
        wait for clk_period*0.001;

        -- 1
```

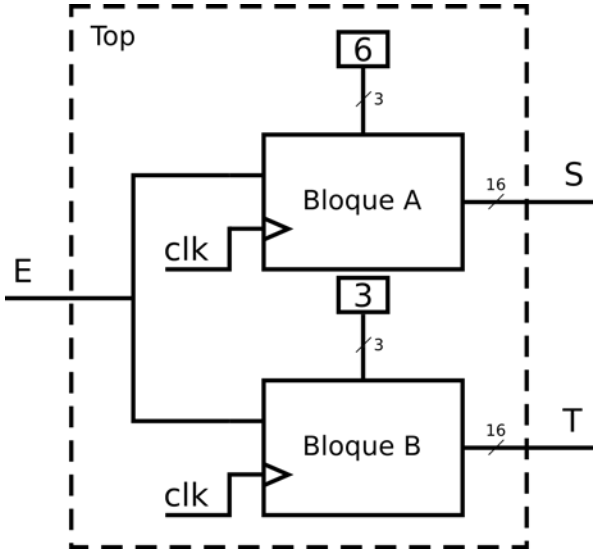
```
72          for i in 0 to 9 loop
73              Columna <= "100";
74              wait for clk_period;
75              Columna <= "000";
76              wait for 3*clk_period;
77          end loop;
78          Columna <= "000";
79          wait for 10ms;
80
81          -- 2
82          for i in 0 to 9 loop
83              Columna <= "010";
84              wait for clk_period;
85              Columna <= "000";
86              wait for clk_period*3;
87          end loop;
88          Columna <= "000";
89          wait for 10ms;
90
91          -- 9
92          for i in 0 to 9 loop
93              Columna <= "000";
94              wait for clk_period*2;
95              Columna <= "001";
96              wait for clk_period;
97              Columna <= "000";
98              wait for clk_period;
99          end loop;
100         Columna <= "000";
101         wait for 10ms;
102
103         -- 0
104         for i in 0 to 9 loop
105             Columna <= "000";
106             wait for clk_period*3;
107             Columna <= "010";
108             wait for clk_period;
109         end loop;
110         Columna <= "000";
111         wait for 10ms;
112
113         wait;
114     end process;
115
116 end testbench;
117
```

Asignatura: Electrónica Digital **Prueba de Evaluación Continua (PEC)**
Convocatoria: Julio 2019 Fecha: 27/06/2019 **Duración: 1 hora y 30 minutos**

Ejercicio 1 (6 Puntos)

Por una línea serie se reciben, de manera síncrona con el reloj, paquetes de 8 bits, en los que el primer bit recibido vale siempre '1', los tres siguientes indican una dirección y los cuatro restantes, un valor numérico. Mientras no hay mensaje, la línea permanece a cero. Los bits de la dirección y del valor numérico se reciben ordenados del más significativo al menos significativo. Se pide:

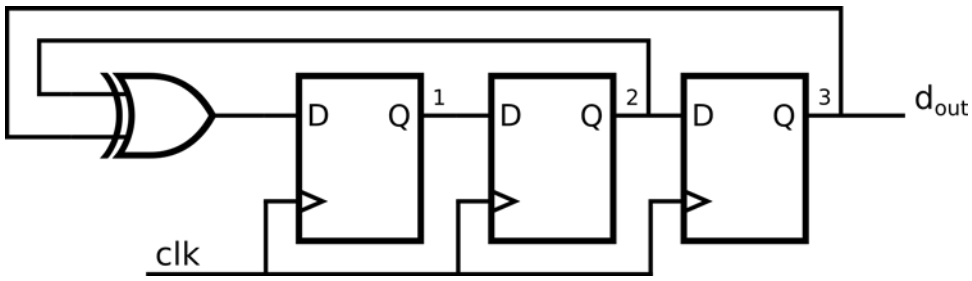
a) Descripción VHDL de un bloque (bloque A en la figura) que reciba la señal serie, interprete el paquete, y en caso de que los bits de dirección coincidan con un valor fijo, que se le indica al bloque por otra entrada (en paralelo, de tres bits), acumule el valor numérico recibido sobre un registro interno de 16 bits. Este valor interno se saca al exterior del módulo. De forma particular, si la dirección recibida es 111, no se atiende al valor numérico del paquete, sino que se resetea el contador de forma síncrona.
b) Con dos de los bloques arriba descritos y empleando para ello una descripción estructural, se desea implementar un sistema de acumuladores que muestre por salidas separadas el valor de un acumulador para las direcciones 3 y 6, tal como muestra la figura. La entrada serie es común a ambos bloques.
c) Implemente también en VHDL un *test bench* que permita validar la acumulación de más de un número en uno solo de los módulos y su función de reset con la dirección 111.



…/… (sigue en la cara posterior)

Ejercicio 2 (4 puntos)

Dado el circuito de la figura:



Se pide:

a) Implementación en VHDL del circuito de la figura, sabiendo que no puede inicializarse a 000 (el sistema no funcionaría).
b) Implementación en VHDL de un circuito similar, pero con longitud de 65 bits, aplicando la XOR entre los bits con posiciones 65 y 47. Nótese que la salida de este circuito puede considerarse un generador de números aleatorios de un bit.
c) Empleando un bloque como el de la figura, asumiendo que se le añade una señal de habilitación (un puerto *enable*), y sabiendo que tenemos un reloj de 8 MHz, diseñe un circuito que genere un número aleatorio (de un bit) nuevo cada segundo.

Nota: No está permitido el uso de calculadora en todo el examen.

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity accumulator is
    port(
        clk   : in  std_logic;
        reset : in  std_logic;
        addr  : in  std_logic_vector(2 downto 0);
        din   : in  std_logic;
        dout  : out std_logic_vector(15 downto 0)
    );
end accumulator;

architecture behavioral of accumulator is

    signal shift_reg : std_logic_vector(6 downto 0);
    signal data_cnt  : unsigned(2 downto 0);

    -- Simple FSM: '1' - operating, '0' - idle
    signal state     : std_logic;
    signal load      : std_logic;
    signal reset_s   : std_logic;

    signal acc       : unsigned(15 downto 0);

begin

    -- Input shift register
    process(clk,reset)
    begin
        if reset = '1' then
            shift_reg <= (others => '0');
        elsif clk'event and clk = '1' then
            shift_reg <= shift_reg(5 downto 0) & din;
        end if;
    end process;

    -- Input data counter
    process(clk,reset)
    begin
        if reset = '1' then
            data_cnt <= (others => '0');
        elsif clk'event and clk = '1' then
            if state = '0' then
                data_cnt <= (others => '0');
            else
                data_cnt <= data_cnt + 1;
            end if;
        end if;
    end process;

    -- Control logic (simple FSM)
    process(clk,reset)
    begin
        if reset = '1' then
            state <= '0';
        elsif clk'event and clk = '1' then
            if state = '0' and din = '1' then
                state <= '1';
            elsif state = '1' and data_cnt = 7 then
```

```
62                  state <= '0';
63              end if;
64          end if;
65      end process;
66
67      load    <= '1' when shift_reg(6 downto 4) = addr and data_cnt = 7 else '0';
68      reset_s <= '1' when shift_reg(6 downto 4) = "111" and data_cnt = 7 else '0';
69
70      -- Accumulator logic
71      process(clk,reset)
72      begin
73          if reset = '1' then
74              acc <= (others => '0');
75          elsif clk'event and clk = '1' then
76              if reset_s = '1' then
77                  acc <= (others => '0');
78              else
79                  if load = '1' then
80                      acc <= acc + unsigned(shift_reg(3 downto 0));
81                  end if;
82              end if;
83          end if;
84      end process;
85
86      -- Output connection
87      dout <= std_logic_vector(acc);
88
89  end behavioral;
90
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity top is
    port(
        clk   : in  std_logic;
        reset : in  std_logic;
        e     : in  std_logic;
        s     : out std_logic_vector(15 downto 0);
        t     : out std_logic_vector(15 downto 0)
    );
end top;

architecture structural of top is

    signal addr_a : std_logic_vector(2 downto 0);
    signal addr_b : std_logic_vector(2 downto 0);

begin

    addr_a <= "110";
    addr_b <= "011";

    bloque_a: entity work.accumulator
    port map(
        clk   => clk,
        reset => reset,
        addr  => addr_a,
        din   => e,
        dout  => s
    );

    bloque_b: entity work.accumulator
    port map(
        clk   => clk,
        reset => reset,
        addr  => addr_b,
        din   => e,
        dout  => t
    );

end structural;

```

```
 1  library ieee;
 2  use ieee.std_logic_1164.all;
 3
 4  entity accumulator_tb is
 5  end accumulator_tb;
 6
 7  architecture testbench of accumulator_tb is
 8
 9      signal clk   : std_logic;
10      signal reset : std_logic;
11      signal addr  : std_logic_vector(2 downto 0);
12      signal din   : std_logic;
13      signal dout  : std_logic_vector(15 downto 0);
14
15      signal aux   : std_logic_vector(6 downto 0);
16
17      constant clk_period : time := 10 ns;
18
19  begin
20
21      uut: entity work.accumulator
22      port map(
23          clk   => clk,
24          reset => reset,
25          addr  => addr,
26          din   => din,
27          dout  => dout
28      );
29
30      proc_clk: process
31      begin
32          clk <= '1';
33          wait for clk_period/2;
34          clk <= '0';
35          wait for clk_period/2;
36      end process;
37
38      proc_stim: process
39      begin
40          reset <= '1';
41          addr <= "000";
42          aux <= "0000000";
43          din <= '0';
44          wait for clk_period*100.5;
45
46          reset <= '0';
47          wait for clk_period*100;
48
49          addr <= "011";
50          aux <= "0111010";
51
52          din <= '1';
53          wait for clk_period;
54          for i in 6 downto 0 loop
55              din <= aux(i);
56              wait for clk_period;
57          end loop;
58          din <= '0';
59          wait for clk_period*100;
60
61          addr <= "110";
```

```
62            aux <= "1100011";
63
64            din <= '1';
65            wait for clk_period;
66            for i in 6 downto 0 loop
67                din <= aux(i);
68                wait for clk_period;
69            end loop;
70            din <= '0';
71            wait for clk_period*100;
72
73            addr <= "110";
74            aux <= "1111111";
75
76            din <= '1';
77            wait for clk_period;
78            for i in 6 downto 0 loop
79                din <= aux(i);
80                wait for clk_period;
81            end loop;
82            din <= '0';
83            wait for clk_period*100;
84
85            wait;
86        end process;
87
88    end testbench;
89
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity lfsr is
    port(
        clk   : in  std_logic;
        reset : in  std_logic;
        dout  : out std_logic
    );
end lfsr;

architecture behavioral of lfsr is

    constant C_DEPTH : integer := 3; -- In b is 65
    constant C_TAP_1 : integer := 3; -- in b is 65
    constant C_TAP_2 : integer := 2; -- In b is 47

    signal shift_register : std_logic_vector(C_DEPTH-1 downto 0);
    signal din            : std_logic;

begin

    process(clk,reset)
    begin
        if reset = '1' then
            shift_register <= (0 => '1', others => '0');
        elsif clk'event and clk = '1' then
            shift_register <= shift_register(C_DEPTH-2 downto 0) & din;
        end if;
    end process;

    din  <= shift_register(C_TAP_1-1) xor shift_register(C_TAP_2-1);
    dout <= shift_register(C_DEPTH-1);

end behavioral;
```

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity random is
    port(
        clk   : in  std_logic;
        reset : in  std_logic;
        dout  : out std_logic
    );
end random;

architecture behavioral of random is

    constant C_DEPTH : integer := 3; -- In b is 65
    constant C_TAP_1 : integer := 3; -- in b is 65
    constant C_TAP_2 : integer := 2; -- In b is 47

    signal shift_register : std_logic_vector(C_DEPTH-1 downto 0);
    signal din            : std_logic;
    signal enable         : std_logic;

    constant C_MAX_CNT : integer := 8*10**6;
    signal cnt : integer range 0 to C_MAX_CNT-1;

begin

    process(clk,reset)
    begin
        if reset = '1' then
            cnt <= 0;
        elsif clk'event and clk = '1' then
            if cnt = C_MAX_CNT-1 then
                cnt <= 0;
            else
                cnt <= cnt + 1;
            end if;
        end if;
    end process;

    enable <= '1' when cnt = C_MAX_CNT-1 else '0';

    process(clk,reset)
    begin
        if reset = '1' then
            shift_register <= (0 => '1', others => '0');
        elsif clk'event and clk = '1' then
            if enable = '1' then
                shift_register <= shift_register(C_DEPTH-2 downto 0) & din;
            end if;
        end if;
    end process;

    din  <= shift_register(C_TAP_1-1) xor shift_register(C_TAP_2-1);
    dout <= shift_register(C_DEPTH-1);

end behavioral;
```

Asignatura: Electrónica Digital

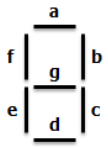
PEC diciembre 2018 **Fecha:** 10 de diciembre de 2018

Cuestión única (10 puntos)

Con objeto de reducir el cableado de un ascensor, se pretende realizar un sistema de visualización del piso en el que se encuentra dicho ascensor, que recibe los datos de forma serie por una interfaz muy sencilla y de pocos cables. La visualización se realiza sobre dos displays de siete segmentos.

La información con los dígitos a visualizar se recibe por una línea serie formada por dos líneas, una línea **DIN** (Dato In) por la que se reciben en forma serie durante ocho ciclos de reloj consecutivos, los dos dígitos a visualizar, en formato BCD, empezando por el bit más significativo del dígito más significativo hasta terminar por el bit menos significativo del dígito menos significativo. La otra señal que se recibe, denominada **CT** (Comienzo de Trama) marca el comienzo del primer bit que se transmite, activándose durante un solo ciclo de reloj coincidente con la recepción del primer bit de información de **DIN**. La información que pueda circular por DIN tras ocho ciclos se descarta hasta que llegue un nuevo flanco por **CT**.

La información recibida se debe mostrar sobre dos displays de siete segmentos, multiplexados en el tiempo, por lo que las señales de salida del circuito son un vector de siete señales SEG(6 downto 0) donde SEG(6) corresponde con el segmento a, hasta SEG(0), que corresponde con el segmento g. (ver imagen). Adicionalmente, dos señales **SelD** y **SelU** (Selección de decenas y unidades, respectivamente). El refresco para los displays será de 1kHz. Todo el circuito funcionará de manera síncrona, empleando para ello una señal externa de reloj (CLK) de 1MHz.



El sistema consta de los siguientes elementos:

- Un conversor de datos seria a paralelo, que transforma el dato en una señal de ocho bits, cuyo valor debe ser el dato a representar (los dos dígitos BCD) desde el momento en que se recibe el último bit de una transmisión, hasta que se recibe el último bit de la transmisión siguiente.

- Un sistema de multiplexación temporal en el que a la frecuencia de refresco mencionada anteriormente, se seleccione un dígito BCD o el otro, activando las señales **SELD** y **SELU** de manera acorde.

- Un único conversor de BCD a siete segmentos. Si el número recibido no fuera BCD, se debe sacar una E (indicador de error).

Se pide:

a) Código VHDL (entidad y arquitectura) del conversor de datos serie a paralelo. (2,5 puntos)

b) Código del circuito completo, en el que se instanciará el conversor serie del apartado a y se añadirán los procesos o sentencias concurrentes adicionales necesarias. (3 puntos)

c) Se quiere añadir dos LEDs adicionales, para indicar si el ascensor debe subir o bajar (señales **UP** y **DOWN**) respectivamente. Estas señales se deben obtener registrando el valor de los dos dígitos anteriores y compararlo con los valores actuales. Describir el código VHDL de la arquitectura que sería necesario añadir para tener esta funcionalidad adicional. (2,5 puntos)

d) Código de un banco de pruebas (testbench) en el que se verifique el reset del sistema, una transmisión correcta, su visualización durante 1 segundo, y otra incorrecta, visualizada durante otro segundo. (2 puntos)

Duración del examen: 1 hora y 30 minutos

Documento anónimo

Digpec20182.pdf

Dig_pec2018_2

4° Electrónica Digital

Grado en Ingeniería en Tecnologías Industriales

Escuela Técnica Superior de Ingenieros Industriales
Universidad Politécnica de Madrid

DESPUÉS DE LOS FINALES...
¡RELÁJATE!

- **480** cartas resultado de mucho amor y birra
- **80 cartas especiales** para animar tus fiestas
- **A partir de 16 años**, de 3 a 15 jugadores
- Perfecto para largas noches de risas con amigos

GUA TA FAC GUA TA FAC GUA TA FAC

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity multiplexor is
   port(
        clk: in std_logic;
        reset: in std_logic;
        Din: in std_logic;
        CT: in std_logic;
        segmentos: out std_logic_vector(6 downto 0)

   );
end multiplexor;

architecture behavioral of multiplexor is
--Refresh
   constant Max: integer:= 1000-1;
   signal cnt_1K: integer range 0 to Max;
   signal refresh: std_logic;

   signal BCD_digit    : unsigned (7 downto 0);
   signal BCD    :  std_logic_vector (7 downto 0);
  signal  BCD_4: unsigned (3 downto 0);
   signal ciclos: unsigned(2 downto 0);
   signal selector: unsigned(1 downto 0);
   signal SeID:  std_logic;
```

```
   signal SeIU:  std_logic;
begin



Cycles: process(clk,reset)
begin
 if reset='1' then
       ciclos<="000";
 elsif clk'event and clk='1' then
       if CT='1' then
         ciclos<=(others=>'0');
       elsif ciclos="111" then
         ciclos<="111";
       else ciclos<=ciclos+1;
       end if;
  end if;
end process;

Digits: process(clk,reset)
begin
 if reset='1' then
       BCD_digit<=(others=>'0');
 elsif clk'event and clk='1' then
       if ciclos<"111" then
         BCD_digit<=BCD_digit(6 downto 0) & Din;
       end if;
```

```
 end if;
end process;
BCD<=std_logic_vector(BCD_digit) when ciclos="111" else "00000000"  ;

--MULTIPLEXOR
Refresco: process(clk,reset)
begin
   if reset='1' then
      cnt_1K<=0;
   elsif clk'event and clk='1' then
      if cnt_1K=Max then
         cnt_1K<=0;
      else cnt_1K<=cnt_1K+1;
      end if;
    end if;
end process;

refresh<='1' when cnt_1K=Max else '0';



process (clk, reset)
          begin
             if reset = '1' then
             selector <= "01";
             elsif clk'event and clk='1' then
                if refresh = '1' then--refresco de 7 segmentos
                   if selector = "10" then
```

```
                selector <= "01";
                else
                selector <= "10";
                end if;
              end if;
            end if;
end process;
with selector select --multiplexor
   BCD_4 <= unsigned(BCD(7 downto 4)) when "10" ,
         unsigned(BCD(3 downto 0)) when "01" ,
         "-"   when others;

   segmentos <=  "0000001" when BCD_4= "0000" else
           "1001111" when BCD_4 = "0001" else
           "0010010" when BCD_4 = "0010" else
           "0000110" when BCD_4 = "0011" else
           "1001100" when BCD_4 = "0100" else
           "0100100" when BCD_4 = "0101" else
           "0100000" when BCD_4 = "0110" else
           "0001111" when BCD_4 = "0111" else
           "0000000" when BCD_4 = "1000" else
           "0001100" when BCD_4 = "1001";
end behavioral;
```

Documento anónimo

DigPEC201811.pdf

Dig_PEC20181_1

4° Electrónica Digital

Grado en Ingeniería en Tecnologías Industriales

Escuela Técnica Superior de Ingenieros Industriales
Universidad Politécnica de Madrid

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity sseg is
   port(
         clk: in std_logic;
         reset: in std_logic;
         Din: in std_logic;
         CT: in std_logic;
          BCD    :out  std_logic_vector (7 downto 0)
   );
end sseg;

architecture behavioral of sseg is

   signal BCD_digit    : unsigned (7 downto 0);
   signal ciclos: unsigned(2 downto 0);
begin

process(clk,reset)
begin
 if reset='1' then
       ciclos<="000";
 elsif clk'event and clk='1' then
       if CT='1' then
         ciclos<=(others=>'0');
```

```
      elsif ciclos="111" then
        ciclos<="111";
      else ciclos<=ciclos+1;
      end if;
 end if;
end process;

process(clk,reset)
begin
 if reset='1' then
      BCD_digit<=(others=>'0');
 elsif clk'event and clk='1' then
      if ciclos<"111" then
        BCD_digit<=BCD_digit(6 downto 0) & Din;
      end if;
 end if;
end process;
BCD<=std_logic_vector(BCD_digit) when ciclos="111" else "00000000"  ;

end behavioral;
```

Asignaturas:
Electrónica Digital (GITI)

Fecha: 18/12/2017
Examen: PEC diciembre 2017

CUESTIÓN 1 (5 puntos)

Se desea implementar un velocímetro para una motocicleta, empleando para ello un *sensor* (que generará un pulso de duración 1 ciclo de reloj cada vez que la rueda delantera de la motocicleta realice una vuelta completa) y una FPGA. En la FPGA se implementará un circuito que recibiendo los pulsos del *sensor* activará una tira de 8 LEDs, de manera proporcional a la velocidad de la motocicleta. La frecuencia de la señal de reloj que recibe la FPGA son 50 MHz. Se pide:

a) Describir un módulo VHDL que a partir de la salida del sensor, el reloj de la FPGA y una señal de reset, genere una señal de 6 bits con el número de vueltas (codificado en binario) que ha dado la rueda de la motocicleta en el último segundo. Esta salida deberá actualizarse al final de cada segundo, manteniéndose estable durante todo el segundo siguiente. El máximo número de vueltas que se considerará es 63.

b) Usando la descripción del apartado a como un componente jerárquico, describa la lógica que controla la activación de la tira de LEDs, de tal manera que el primer LED se activará cuando la velocidad proporcionada por el bloque anterior esté por debajo de 8 vueltas por segundo, el segundo se activará cuando la entrada entré entre 8 y 15 vueltas, y así sucesivamente hasta los 8 LEDs que componen la tira.

CUESTIÓN 2 (2 puntos)

Describir en VHDL (entidad y arquitectura) la siguiente máquina de estados:

CUESTIÓN 3 (3 puntos)

Describir un módulo VHDL que se corresponda con un circuito que genere la secuencia síncrona **11011001** de forma cíclica, de estas tres maneras posibles:

c) Usando como base un contador binario síncrono

d) Usando como base un contador Johnson

e) Usando como base un contador en anillo

***Duración del examen*: 1 hora y 30 minutos.**

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

entity velocimetro is
    Port ( clk          : in STD_LOGIC;
           reset        : in STD_LOGIC;
           sensor       : in STD_LOGIC;
           VueltasOut   : out std_logic_vector(5 downto 0));
end velocimetro;

architecture Behavioral of velocimetro is

    constant Conta1segMAX : integer := 50*(10**6) - 1;
    signal conta1seg : integer range 0 to Conta1segMAX;
    signal UnSeg : std_logic;

    signal Vueltas: unsigned(5 downto 0);

begin

    -- Divisor de Frecuencia a 1 segundo
    DivFreq: process(clk,reset)
    begin
        if reset = '1' then
            conta1seg <= 0;
        elsif clk'event and clk = '1' then
            if conta1seg < Conta1segMAX then
                conta1seg <= conta1seg + 1;
            else
                conta1seg <= 0;
            end if;
        end if;
    end process;
    UnSeg <= '1' when conta1seg = Conta1segMAX else '0';

    -- Proceso en el que se cuentan las vueltas y se actualiza la salida (1 vez por segundo)
    ContaVueltas: process(clk,reset)
    begin
        if reset = '1' then
            Vueltas     <= "000000";
            VueltasOut  <= "000000";
        elsif clk'event and clk = '1' then
            if Unseg = '1' then
                VueltasOut <= std_logic_vector(Vueltas);
                Vueltas <= "000000";
            elsif sensor = '1' then
                    Vueltas <= Vueltas + 1;
            end if;
        end if;
    end process;

end Behavioral;
```

```
 1  library IEEE;
 2  use IEEE.STD_LOGIC_1164.ALL;
 3  use IEEE.NUMERIC_STD.ALL;
 4
 5  entity TopVelocimetro is
 6      Port ( clk : in STD_LOGIC;
 7             reset : in STD_LOGIC;
 8             sensor : in STD_LOGIC;
 9             leds : out STD_LOGIC_VECTOR(7 DOWNTO 0)
10           );
11  end TopVelocimetro;
12
13  architecture Behavioral of TopVelocimetro is
14
15      component velocimetro is
16          Port ( clk : in STD_LOGIC;
17                 reset : in STD_LOGIC;
18                 sensor : in STD_LOGIC;
19                 VueltasOut   : out std_logic_vector(5 downto 0));
20      end component;
21
22      signal VueltasOut : std_logic_vector(5 downto 0);
23      signal uV : unsigned(5 downto 0);
24
25  begin
26
27      -- Instancia del componente velocímetro
28      instVelo: velocimetro
29                port map(
30                 clk => clk,
31                 reset => reset,
32                 sensor => sensor,
33                 VueltasOut => VueltasOut
34                 );
35
36      --Transformación a unsigned para facilitar la codificación
37      uV <= unsigned(VueltasOut);
38
39      leds <= "00000001" when uV < 8 else
40              "00000010" when uV < 16 else
41              "00000100" when uV < 24 else
42              "00001000" when uV < 32 else
43              "00010000" when uV < 40 else
44              "00100000" when uV < 48 else
45              "01000000" when uV < 56 else
46              "10000000";
47  end Behavioral;
48
```

```vhdl
------------------------
-- Electrónica Digital --
-- PEC diciembre 2017  --
-- Cuestión 3          --
------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity c3 is
    port (
        clk   : in  std_logic;
        reset : in  std_logic;
        seq   : out std_logic
    );
end c3;

-----------------------------------------
-- La solución comienza a partir de aquí --
-----------------------------------------

-- a) Base: contador binario síncrono
architecture cnt_binario of c3 is
    signal cnt : unsigned(2 downto 0);
begin

    process(clk,reset)
    begin
        if reset = '1' then
            cnt <= (others => '0');
        elsif clk'event and clk = '1' then
            cnt <= cnt + 1;
        end if;
    end process;

    with cnt select
        seq <= '1' when "000",
               '1' when "001",
               '0' when "010",
               '1' when "011",
               '1' when "100",
               '0' when "101",
               '0' when "110",
               '1' when "111",
               '-' when others;

end cnt_binario;

-- b) Base: contador Johnson
architecture cnt_johnson of c3 is
    signal shiftreg : std_logic_vector(3 downto 0);
begin

    process(clk,reset)
    begin
        if reset = '1' then
            shiftreg <= (others => '0');
        elsif clk'event and clk = '1' then
            shiftreg <= shiftreg(2 downto 0) & not shiftreg(3);
        end if;
    end process;

    with shiftreg select
        seq <= '1' when "0000",
               '1' when "0001",
               '0' when "0011",
               '1' when "0111",
               '1' when "1111",
               '0' when "1110",
               '0' when "1100",
               '1' when "1000",
               '-' when others;
```

```vhdl

end cnt_johnson;

-- c) Base: contador en anillo
architecture cnt_anillo of c3 is
    signal shiftreg : std_logic_vector(7 downto 0);
begin

    process(clk,reset)
    begin
        if reset = '1' then
            shiftreg <= (0 => '1', others => '0');
        elsif clk'event and clk = '1' then
            shiftreg <= shiftreg(6 downto 0) & shiftreg(7);
        end if;
    end process;

    with shiftreg select
        seq <= '1' when "00000001",
               '1' when "00000010",
               '0' when "00000100",
               '1' when "00001000",
               '1' when "00010000",
               '0' when "00100000",
               '0' when "01000000",
               '1' when "10000000",
               '-' when others;

end cnt_anillo;

```

Asignaturas:
Electrónica Digital (GITI)
Complementos de Electrónica Digital (MEI)

Fecha: 19/12/2016
Examen: PEC diciembre 2016

CUESTIÓN 1 (4 puntos)

a) Escribir una arquitectura VHDL para el circuito de la figura 1.

b) Escribir la arquitectura VHDL para un circuito que responda al diagrama de estados representado en la figura 2.

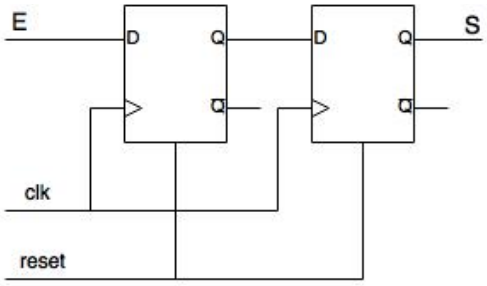


Figura 1

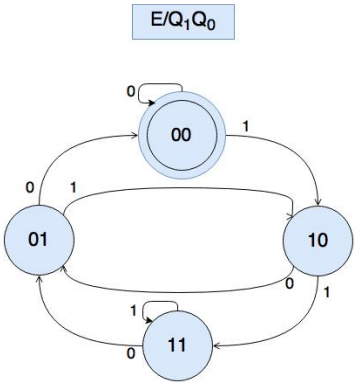
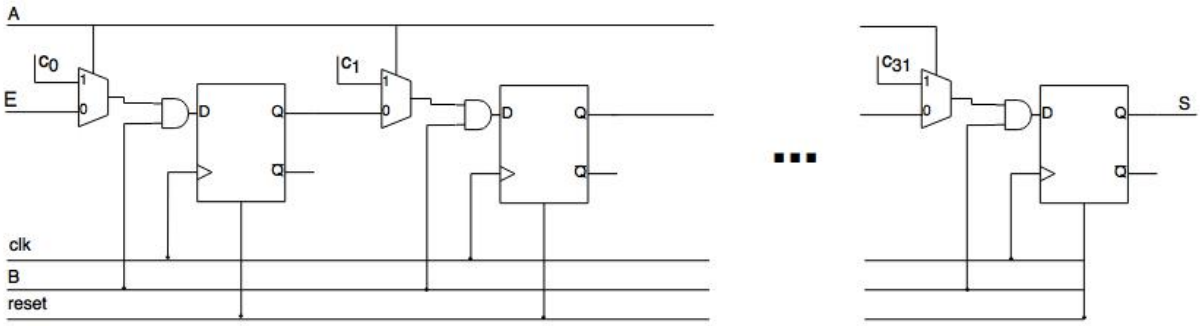


Figura 2

CUESTIÓN 2 (3 puntos)

Describir en VHDL (entidad y arquitectura) el circuito de la siguiente figura:



CUESTIÓN 3 (3 puntos)

El código BCD-exceso-tres y el código BCD Aiken son dos codificaciones BCD autocomplemetarias, es decir, dos dígitos que sumen 9 son uno el negado del otro. El BCD exceso tres se obtiene sumando tres al número en BCD natural. En BCD Aiken, los pesos de los dígitos binarios son $2^1 2^2 2^2 1^2 0^0$, siendo la codificación desde 0 hasta 4 igual que en BCD natural, y la autocomplementaria en los demás casos.

Se pide describir en VHDL (bibliotecas usadas, entidad y arquitectura) dos circuitos **combinacionales**, uno que convierta de BCD a BCD exceso tres, y otro, que convierta de BCD a BCD Aiken. En ambos casos, la entidad debe tener, tanto en la entrada como la salida, señales de tipo *std_logic_vector*. Realizar las arquitecturas correspondientes de la manera más sencilla posible.

***Duración del examen*: 1 hora y 30 minutos.**

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS INDUSTRIALES

UPM
DIE

DEPARTAMENTO DE AUTOMÁTICA,
INGENIERÍA ELECTRÓNICA
E INFORMÁTICA INDUSTRIAL

DIVISIÓN DE INGENIERÍA ELECTRÓNICA (DIE)

Asignatura: Electrónica Digital

PEC diciembre 2015 **Fecha:** 14 de diciembre de 2015

Cuestión 1 (5 puntos)

a) Describir un módulo en VHDL que detecte la secuencia **11101**, síncrona con el reloj y con repetición, en su versión Mealy, inspirándose en un registro de desplazamiento y la lógica adicional necesaria.

b) ¿Qué hubiera cambiado en la descripción anterior, si en lugar de ser un detector de tipo Mealy fuera uno de tipo Moore? Mencionar los cambios, pero no reescribir todo el código.

c) Usando la descripción del apartado a como un componente jerárquico, describir un sistema que encienda o apague un LED conectado a su salida, cada vez que se detecte la sentencia mencionada en el apartado a, por su entrada.

d) Realizar un banco de pruebas para simular el circuito del apartado b, verificando a la vez el reaprovechamiento de los bits de la secuencia anterior como bits correctos de la siguiente secuencia. Dibujar la forma de onda de la señal de entrada que se quiere probar, y la salida esperada.

Cuestión 2 (5 puntos)

Describir en VHDL un módulo que, usando como entradas un reloj de 50 MHz, un reset y seis señales BCD que codifican hora, minutos y segundos de un reloj (unidades y decenas para cada uno), se generen las seis señales de selección de cada display y los siete segmentos necesarios para un esquema de displays multiplexados en el tiempo. Las señales de selección de display y los segmentos son activos por nivel alto, y la velocidad de refresco de 0,1 ms por cada display.

Duración del examen: 1 hora 15 minutos