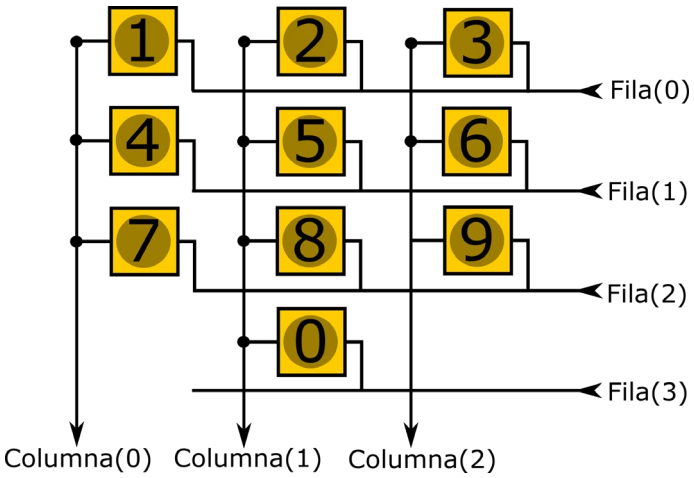
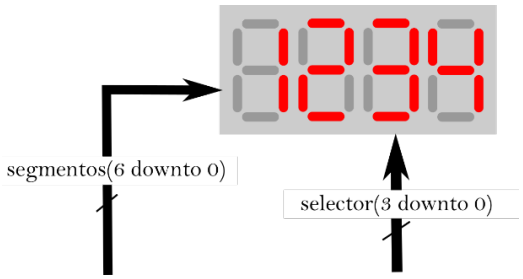
Asignatura: Electrónica Digital (GITI)
(**Prueba de Evaluación Continua**)

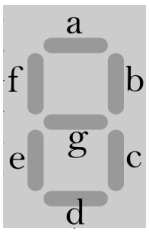
Fecha: 2/11/2019
Convocatoria: Diciembre

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.

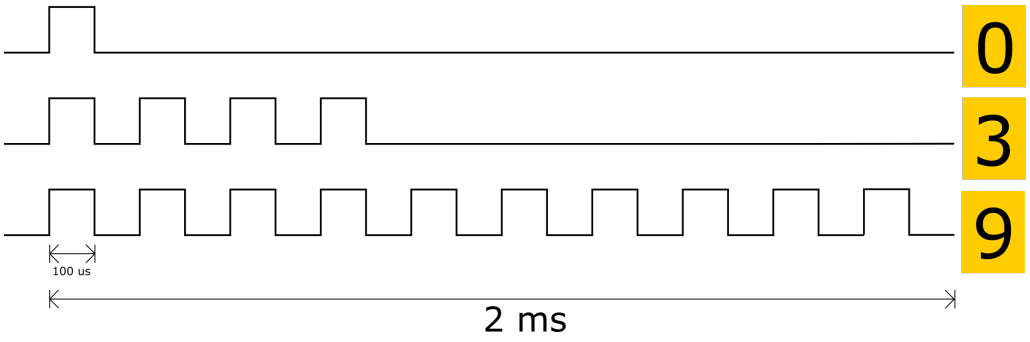
El sistema de control digital que se pretende diseñar constará de los siguientes bloques:

- **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
```