VHDL.pdf

VHDL

4° Electrónica Digital

Grado en Ingeniería en Tecnologías Industriales

Escuela Técnica Superior de Ingenieros Industriales
Universidad Politécnica de Madrid

KEEP CALM AND ESTUDIA UN POQUITO

WUOLAH + #QuédateEnCasa

PEC VHDL - 14 DICIEMBRE 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.

a)

```
entity secuencia is
  Port (clk:    in std_logic;
        reset:  in std_logic;
        din:    in std_logic;
        deteccion: out std_logic
       ); led
end secuencia;

architecture Behavioral of secuencia is
  signal sec : std_logic_vector (3 downto 0);
                std_logic_vector (4 downto 0);
begin
  process (clk, reset)
  begin
    if reset = '1' then
      sec <= (others => '0');
    elsif clk'event and clk = '1' then
      sec <= sec (2 downto 0) & din;
                 sec (3 downto 0) & din;
    end if;
  end process;
  deteccion <= '1' when sec = "1110" and din = '1' else '0';
  led                    sec = "11101"
end Behavioral;
```

b) Los cambios serían los representados en verde

c) Sin utilizar un componente jerárquico, es muy simple cambiar la señal "deteccion" por una nueva señal "led" que tenga el mismo funcionamiento.

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.

```
entity modulo is
  Port ( clk:      in std_logic;
         reset:    in std_logic;
         Selector: out std_logic_vector (5 downto 0);
         segmento: out std_logic_vector (6 downto 0)
       );
end modulo;
architecture Behavioral of modulo is
    --Divisor frecuencia 1s
      constant MAX1S : integer := 50 * 10**6;
      signal cont1s :  integer range 0 to MAX1S-1;
      signal ovf1s :   std_logic;
    --Contadores
      signal enables : std_logic;
      signal conts :   unsigned(3 downto 0);
      signal ovfs :    std_logic;

      signal enableds: std_logic;
      signal contds:   unsigned(3 downto 0);
      signal ovfds:    std_logic;

      signal enablem:  std_logic;
      signal cont_m:   unsigned(3 downto 0);
      signal ovfm:     std_logic;

      signal enabledm: std_logic;
      signal contdm:   unsigned(3 downto 0);
      signal ovfdm:    std_logic;

      signal enableh:  std_logic;
      signal conth:    unsigned(3 downto 0);
      signal ovfh:     std_logic;

      signal enabledh: std_logic;
      signal contdh:   unsigned(3 downto 0);
      signal ovfdh:    std_logic;

    -- Refresco display (1KHz - 0,1ms)
      constant MAXREF : integer := 50*10**3;
      signal contREF: integer range 0 to MAXREF-1;
      signal refresco: std_logic;
    -- Contador display (de 0 a 5)
      signal contdisplay : unsigned (2 downto 0);
    -- Multiplexo
      signal interno : std_logic_vector (3 downto 0);
```

```
begin
--Divisor frecuencia 1s
  process (clk, reset)
  begin
    if reset = '1' then
      cont1s <= 0;
    elsif clk'event and clk='1' then
      if cont1s = MAX1S-1 then
        cont1s <= 0;
      else
        cont1s <= cont1s + 1;
      end if;
    end if;
  end process;

  ovf1s <= '1' when cont1s = MAX1S-1
           else '0';

--Contador segundos
  process (clk, reset)
  begin
    if reset = '1' then
      conts <= (others => '0');
    elsif clk'event and clk='1' then
      if enables = '1' then
        if conts = 9 then
          conts <= (others => '0');
        else
          conts <= conts + 1;
        end if;
      end if;
    end if;
  end process;

  enables <= ovf1s;
  ovfs <= '1' when conts = 9 and
          enables = '1' else '0';

--Contador decenas segundo
  process (clk, reset)
  begin
    if reset = '1' then
      contds <= (others => '0');
    elsif clk'event and clk = '1' then
      if enableds = '1' then
        if contds = 5 then
          contds <= (others => '0');
        else
          contds = contds + 1;
        end if;
      end if;
    end if;
  end process;
  enableds <= ovfs;
  ovfds <= '1' when contds = 5 and enableds = '1' else '0';

--Contador minutos
  process (clk, reset)
  begin
    if reset = '1' then
      contm <= (others => '0');
    elsif clk'event and clk = '1' then
      if enablem = '1' then
        if contm = 9 then
          contm <= (others => '0');
        else
          contm <= contm + 1;
        end if;
      end if;
    end if;
  end process;
  enablem <= ovfds;
  ovfm <= '1' when contm = 9 and enablem = '1' else '0';
--Contador decena minutos
  process (clk, reset)
  begin
    if reset = '1' then
      contdm <= (others => '0');
    elsif clk'event and clk = '1' then
      if enabledm = '1' then
        if contdm = 5 then
          contdm <= (others => '0');
        else
          contdm <= contdm + 1;
        end if;
      end if;
    end if;
  end process;
  enabledm <= ovfm;
  ovfdm <= '1' when contdm = 5 and enabledm = '1' else '0';
--Contador horas
--Contador decenas horas
```


```
-- Refresco cada 0,1ms (1KHz)
process (clk, reset)
begin
  if reset = '1' then
    contREF = 0;
  elsif clk'event and clk = '1' then
    if contREF = MAXREF-1 then
      contREF = 0;
    else
      contREF = contREF + 1;
    end if;
  end if;
end process;
refresco <= '1' when contREF = maxREF-1
            else '0';

-- Contador de 0 a 5 para displays
process (clk, reset)
begin
  if reset = '1' then
    contdisplay <= (others => '0');
  elsif clk'event and clk = '1' then
    if refresco = '1' then
      if contdisplay = 5 then
        contdisplay <= (others => '0');
      else
        contdisplay <= contdisplay + 1;
      end if;
    end if;
  end if;
end process;

-- Multiplexor
with contdisplay select
interna <= std_logic_vector (conts)   when "000",
           std_logic_vector (contds)  when "001",
           std_logic_vector (contm)   when "010",
           std_logic_vector (contdm)  when "011",
           std_logic_vector (conth)   when "100",
           std_logic_vector (contdh)  when "101",
           "----"                     when others;

-- Selector
with contdisplay select
selector <= "000001" when "000",
            "000010" when "001",
            "000100" when "010",
            "001000" when "011",
            "010000" when "100",
            "100000" when "101",
            "------" when others;

-- BCD - 7 segmentos
with interna select
segmentos <= "0000001" when "0000",
             "1001111" when "0001",
             "0010010" when "0010",
             "0000110" when "0011",
             "1001100" when "0100",
             "0100100" when "0101",
             "0100000" when "0110",
             "0001111" when "0111",
             "0000000" when "1000",
             "0000100" when "1001",
             "-------" when others;
```

PEC VHDL - 19 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.

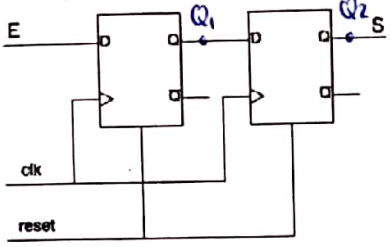


Figura 1

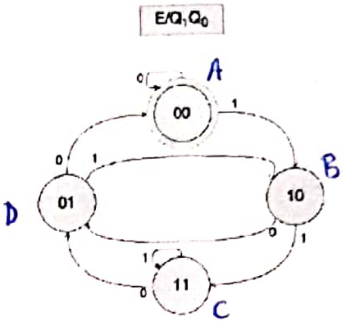


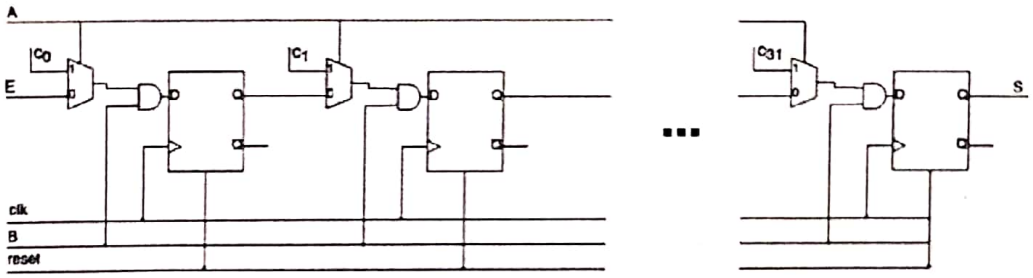
Figura 2

a)
```
entity Figura1 is
  Port ( clk:   in std_logic;
         reset: in std_logic;
         E :    in std_logic;
         S :    out std_logic
       );
end Figura1;

architecture Behavioral of Figura1 is
  signal Q1 : std_logic;
  signal Q2 : std_logic;

begin
  process (clk, reset)
  begin
    if reset = '1' then
      Q1 <= '0';
      Q2 <= '0';
    elsif clk'event and clk='1' then
      Q1 <= E;
      Q2 <= Q1;
    end if;
  end process;

  S <= Q2;
end Behavioral;
```

b)
```
entity of Figura2 is
  Port ( clk:   in std_logic;
         reset: in std_logic;
         E :    in std_logic;
         Q :    out std_logic_vector (1 downto 0)
       );
end Figura2;

architecture Behavioral of Figura2 is
  type state_t is (A, B, C, D);
  signal state : state_t;

begin
  process (clk, reset)
  begin
    if reset = '1' then
      state <= A;
    elsif clk'event and clk = '1' then
      case state is

when A =>
  if E='1' then
    state <= B;
  end if;

when B =>
  if E='1' then
    state <= C;
  elsif E='0' then
    state <= D;
  end if;

when C =>
  if E='0' then
    state <= D;
  end if;

when D =>
  if E='1' then
    state <= B;
  elsif E='0' then
    state <= A;
  end if;

      end case;
    end if;
  end process;

  Q <= "00" when state=A else
       "10" when state=B else
       "11" when state=C else
       "01" when state=D;
end Behavioral;
```

CUESTIÓN 2 (3 puntos)

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




```
entity Registro is
  Port (clk: in std_logic;
        reset: in std_logic;
        A : in std_logic;
        B : in std_logic;
        C : in std_logic_vector (31 downto 0);
        E : in std_logic;
        S : out std_logic
  );
end Registro;

architecture Behavioral of Registro is
    signal reg : std_logic_vector (31 downto 0);
begin
  process (clk, reset)
  begin
    if reset = '1' then
       reg <= (others => '0');
    elsif clk'event and clk = '1' then
        if (A='1' and B='1') then
            reg <= C;
        elsif (A='0' and B='1' and E='1') then
            reg <= reg (30 downto 0) & E;
        end if;
    end if;
  end process;

  S <= reg(31);
end Behavioral;
```

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 $2^1 2^2 2^1 2^0$, 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.

Para ambas conversiones utilizaremos las siguientes librerías:

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

Conversión BCD a BCD exceso tres

```
entity conversion1 is
Port(entrada: in std_logic_vector (3 downto 0);
     salida: out std_logic_vector (3 downto 0)
    );
end conversion1;
architecture Behavioral of conversion1 is
    signal interna: unsigned (3 downto 0);
begin
  interna <= unsigned (entrada) +3;
  salida <= std_logic_vector (interna);
end Behavioral;
```

Conversión BCD a BCD Aiken

```
entity conversion2 is
  Port ( entrada: in std_logic_vector (3 downto 0);
         salida: out std_logic_vector (3 downto 0)
       );
end conversion2;
architecture Behavioral of conversion2 is
    signal interna: unsigned (3 downto 0);
    if (unsigned (entrada) < 5) then
        interna <= unsigned (entrada);
    else
      interna <= unsigned (not (9 - unsigned (entrada)))
    end if;
     salida <= std_logic_vector (interna);
end Behavioral;
```

PEC VHDL – 18 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 entre entre 8 y 15 vueltas, y así sucesivamente hasta los 8 LEDs que componen la tira.

a)

```
entity Velocimetro is
  Port ( clk    : in std_logic;
         reset  : in std_logic;
         sensor : in std_logic;
         Vueltas: out unsigned (5 downto 0)
       );
end Velocimetro;

architecture Behavioral of Velocimetro is
  -- Divisor frecuencia 1s
  constant maximo : integer := 50*10**6;
  signal cont1s   : integer range 0 to maximo-1;
  signal ovf1s    : std_logic;
  -- Variable interna
  signal num      : unsigned(5 downto 0);
begin
  --Contador 1s
  process (clk, reset)
  begin
    if reset = '1' then
      cont1s <= 0;
    elsif clk'event and clk = '1' then
      if cont1s = maximo-1 then
        cont1s <= 0;
      else
        cont1s <= cont1s + 1;
      end if;
    end if;
  end process;
  ovf1s <= '1' when cont1s = maximo-1 else '0';

  -- Contador de las vueltas
  process (clk, reset)
  begin
    if reset = '1' then
      num <= (others => '0');
    elsif clk'event and clk = '1' then
      if sensor = '1' then
        num <= num + 1;
      end if;
    end if;
  end process;

  -- Actualización de las vueltas
  process (clk, reset)
  begin
    if reset = '1' then
      Vueltas <= (others => '0');
    elsif clk'event and clk = '1' then
      if ovf1s = '1' then
        Vueltas <= num;
      end if;
    end if;
  end process;
```

b)

```
entity traLEDs is
  Port (clk: in std_logic;
        reset: in std_logic;
        sensor: in std_logic;
        leds : out std_logic_vector (7 downto 0)
       );
end traLEDs;

Architecture Behavioral of traLEDs is
    Component Velocimetro is
      Port (clk:     in std_logic;
            reset:   in std_logic;
            sensor:  in std_logic;
            Vueltas: out unsigned (5 downto 0)
           );
      end component
      signal numVueltas: unsigned (5 downto 0);
begin
    -- Instancia
     port map (clk     => clk,
               reset   => reset,
               sensor  => sensor,
               Vueltas => numVueltas
              );

    -- Funcionamiento leds

    leds <= "00000001" when numVueltas < 8  else
            "00000010" when numVueltas < 16 else
            "00000100" when numVueltas < 24 else
            "00001000" when numVueltas < 32 else
            "00010000" when numVueltas < 40 else
            "00100000" when numVueltas < 48 else
            "01000000" when numVueltas < 56 else
            "10000000" when numVueltas < 64 else
            "--------";

end Behavioral;
```

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:

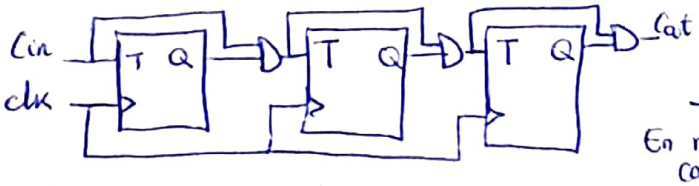
a) Usando como base un contador binario síncrono

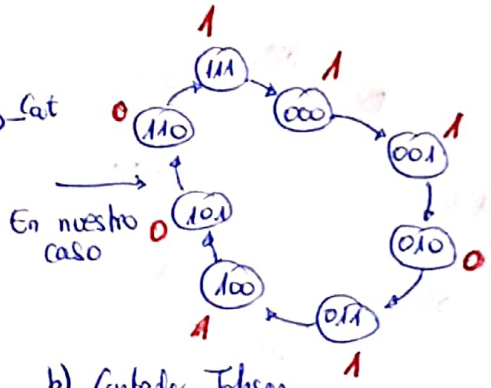
b) Usando como base un contador Johnson

c) Usando como base un contador en anillo

CONTADORES CON REGISTRO DE DESPLAZAMIENTO

a) Contador binario síncrono



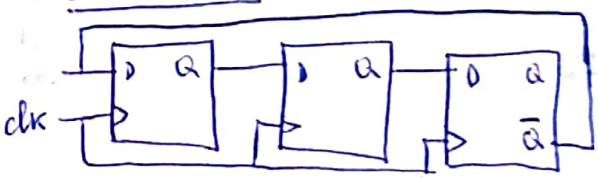


* Secuencia de n números: $2^{\text{x bits}}$

8 números → 3 bits

```
entity ContadorBinario is
  Port(clk: in std_logic;
       reset: in std_logic;
       Secuencia: out std_logic
  );
end ContadorBinario;

architecture Behavioral of ContadorBinario is
  Signal Contador: unsigned (2 downto 0);
  Signal ContadorJohson: unsigned (3 downto 0);
  Signal ContadorAnillo: unsigned (7 downto 0);
begin
  --Contador
  process (clk, reset)
  begin
    if reset='1' then
      Contador <= (others => '0');
      ContadorAnillo <= "00000001";
    elsif clk'event and clk='1' then
      Contador <= Contador + 1;
      ContadorJohson <= not (ContadorJohson(0)) & ContadorJohson (3 downto 1);
      ContadorAnillo <= ContadorAnillo(0) & ContadorAnillo (7 downto 1);
    end if;
  end process;
  --Secuencia
  with Contador select
  Secuencia <= '1' when "000", '1' when "001", '0' when "010", '1' when "011",
                "00001001"     "10001000"      "01100000"      "00110000"
               '1' when "100", '0' when "101", '0' when "110", '1' when "111", '-' when others;
                "00011110"     "00001110"      "00000110"      "00000001"
end Behavioral;
```

b) Contador Johson

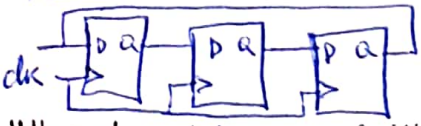


* Secuencia de n números: n/2 bits

8 números → 8 bits

"He quedo con todos menos el último y se lo añado negado a la izq."

c) Contador Anillo



"He quedo con todos menos el último y se lo añado a la izq."

* Secuencia de n números → n bits

8 números → 8 bits

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 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

PEC VHDL – 10 DICIEMBRE 2018

a)

```
entity conversorSerieParalelo is
  Port (clk:  in  std_logic;
        reset: in  std_logic;
        DIN :  in  std_logic;
        CT :   in  std_logic;
        Conversor: out std_logic_vector (7 downto 0)
       );
end conversorSerieParalelo;

architecture Behavioral of conversorSerieParalelo is
  -- Importante saber cuando almacenar el
     dato completo (tras 8 bits), cuando se
     reciben datos ...
type state_t is (STOP, WORKING);
signal state : state_t;
signal contDatos : unsigned (3 downto 0);
Signal almacenar : std_logic;
Signal numero : std_logic_vector (7 downto 0);
begin
  -- Maquina estados para funcionamiento
  process (clk, reset)
  begin
    if reset='1' then
      state <= STOP;
    elsif clk'event and clk='1' then
      case state is
      when STOP =>
        if CT='1' then
          state <= WORKING;
        end if;
      when WORKING =>
        if contDatos = 7 then
          state <= STOP;
        end if;
      end case;
    end if;
  end process;
almacenar <='1' when state <= WORKING
        and contDatos = 7 else '0';

-- Contador de datos a recibir
process (clk,reset)
begin
  if reset='1' then
    contDatos <= (others => '0');
  elsif clk'event and clk='1' then
    if state = STOP then
      contDatos <= (others => '0');
    else
      contDatos <= contDatos + 1;
    end if;
  end if;
end process;

-- Conversor (Registro desplazamiento)
process (clk,reset)
begin
  if reset='1' then
    numero <= (others => '0');
  elsif clk'event and clk='1' then
    if state = WORKING then
      numero <= numero(6 downto 0) & DIN;
    end if;
  end if;
end process;

-- Almacenamiento en el momento correcto
process (clk,reset)
begin
  if reset='1' then
    conversion <= (others => '0');
  elsif clk'event and clk='1' then
    if almacenar='1' then
      conversion <= numero;
    end if;
  end if;
end process;
end Behavioral;
```

b)

```
entity circuitoCompleto is
  Port (clk:    in std_logic;
        reset:  in std_logic;
        DIN:    in std_logic;
        CT:     in std_logic;
        SelD:   out std_logic;
        SelU:   out std_logic;
        SEG:    out std_logic_vector (6 downto 0)
  );
end circuitoCompleto;

architecture Behavioral of circuitoCompleto is
  Component conversorSerieParalelo is
    Port (clk:    in std_logic;
          reset:  in std_logic;
          DIN:    in std_logic;
          CT:     in std_logic;
          Conversion: out std_logic_vector (7 downto 0)
    );
  end component
  -- Refresco Display
  constant MAXREF : integer := 1000;
  signal contREF: integer range 0 to MAXREF-1;
  signal refresco : std_logic;
  -- Maquina estados para unidades o decenas
  type state_t is (NADA, DEC, UNID);
  signal state : state_t;
  -- Variables internas
  signal numero : std_logic_vector (7 downto 0);
  signal BCD : std_logic_vector (3 downto 0);
begin
  -- Instancia del componente
  port map (clk        => clk,
            reset      => reset,
            DIN        => DIN,
            CT         => CT,
            Conversion => numero
  );
```

```
SelD <= '1' when state = DEC else '0';
SelU <= '1' when state = UNID else '0';
```

```
  -- Maquina estados
  process (clk, reset)
  begin
    if reset = '1' then
      state <= NADA;
    elsif clk'event and clk='1' then
      case state is
        when NADA =>
          if refresco = '1' then
            state <= UNID;
          end if;
        when UNID =>
          if refresco = '1' then
            state <= DEC;
          end if;
        when DEC =>
          if refresco = '1' then
            state <= UNID;
          end if;
      end case;
    end if;
  end process;

  BCD <= numero (7 downto 4) when state<=DEC else
         numero (3 downto 0) when state<=UNID else
         "----";

  -- Refresco Display
  process (clk, reset)
  begin
    if reset='1' then
      contREF <= 0;
    elsif clk 'event and clk ='1' then
      if contREF = MAXREF -1 then
        contREF <= 0;
      else
        contREF <= contREF +1;
      end if;
    end if;
  end process;
  refresco <= '1' when contREF = MAXREF-1 else '0';
--Conversor BCD - 7 segments
with BCD select
  SEG <= "0000001" when "0000",
         "1001111" when "0001",
         "0010010" when "0010",
         "0000110" when "0011",
         "1001100" when "0100",
         "0100100" when "0101",
         "0100000" when "0110",
         "0001111" when "0111",
         "0000000" when "1000",
         "0000100" when "1001",
         "0110000" when others;
end Behavioral;
```

c)
```
entity LEDs is
  Port (clk: in std_logic;
        reset: in std_logic;
        UP : out std_logic;
        DOWN: out std_logic;
        conversor: in std_logic_vector (7 downto 0)
       );
end LEDs;

architecture Behavioral of LEDs is
  --Cargo el valor anterior del conversorSP
  signal convAnt : std_logic_vector (7 downto 0);
  --Maquina estados para subir o bajar
  type state_t is (NORMAL, SUBIR, BAJAR)
  signal state : state_t;
begin
  --Cargo el valor anterior del conversorSP
  process (clk, reset)
  begin
    if reset = '1' then
      convAnt <= (others => '0');
    elsif clk'event and clk = '1' then
      convAnt <= conversor;
    end if;
  end process;
  --Maquina estados
  process (clk, reset)
  begin
    if reset='1' then
      state <= NORMAL;
    elsif clk'event and clk = '1' then
      case state is
      when NORMAL =>
        if (unsigned(convAnt) > unsigned(conversor)) then
          state <= BAJAR;
        elsif (unsigned(conversor) > unsigned(convAnt)) then
          state <= SUBIR;
        end if;
      when SUBIR =>
        if (unsigned(convAnt) > unsigned(conversor)) then
          state <= BAJAR;
        elsif (unsigned(convAnt) = unsigned(conversor)) then
          state <= NORMAL;
        end if;
      when BAJAR =>
        if (unsigned(conversor) > unsigned(convAnt)) then
          state <= SUBIR;
        elsif (unsigned(conversor) = unsigned(convAnt)) then
          state <= NORMAL;
        end if;
      end case;
    end if;
  end process;

  UP <= '1' when state = SUBIR else '0';
  DOWN <= '1' when state = BAJAR else '0';
end Behavioral;
```

d)

```
entity testbench is
end testbench;

architecture Behavioral of testbench is
--Declaro el programa como component
  component programa is
    Port ( clk:   in std_logic;
           reset: in std_logic;
           DIN:   in std_logic;
           CT:    in std_logic;
           SELD:  out std_logic;
           SELU:  out std_logic;
           UP:    out std_logic;
           DOWN:  out std_logic;
           SEG:   out std_logic_vector (6 downto 0));
  end component;
-- Declaro las señales de mi testbench
  signal t_clk   : std_logic;
  signal t_reset : std_logic;
  signal t_DIN   : std_logic;
  signal t_CT    : std_logic;
  signal t_SELD  : std_logic;
  signal t_SELU  : std_logic;
  signal t_UP    : std_logic;
  signal t_DOWN  : std_logic;
  signal t_SEG   : std_logic_vector (6 downto 0);
  constant clk_period : time := 1000ns;
begin
  --Generacion del reloj
    proceso
    begin
      t_clk <= '1';
      wait for clk_period/2;
      t_clk <= '0';
      wait for clk_period/2;
    end process;

--Instanciacion
  uut: programa
    port map ( clk   => t_clk,
               reset => t_reset,
               DIN   => t_DIN,
               CT    => t_CT,
               SELD  => t_SELD,
               SELU  => t_SELU,
               UP    => t_UP,
               DOWN  => t_DOWN,
               SEG   => t_SEG
             );
end Behavioral;
```

```
-Generacion de los estimulos
proces
begin
  t_reset <= '1';
  t_CT <= '0';
  t_DIN <= '0';
  wait for clk_period;

  t_reset <= '0';
  wait for clk_period;
  t_CT <= '1';
  t_DIN <= '1';
  wait for clk_period;
  t_CT <= '0';
  t_DIN <= '0';
  wait for clk_period;
  t_DIN <= '0';
  wait for clk_period;
  t_DIN <= '1';
  wait for clk_period;
  t_DIN <= '1';
  wait for clk_period;
  t_DIN <= '0';
  wait for clk_period;
  t_DIN <= '0';
  wait for clk_period;
  t_DIN <= '0';
  wait for clk_period;
  t_DIN <= '0';  → Prueba para que se pueda cargar bits aunque se envíen 0.
  wait for 1000000 * clk_period;
  --Compruebo reset sistema
  -- Piso 1001 1000 (98)
  -- Veo el resultado 1s

  t_CT <= '1';
  t_DIN <= '1';
  wait for clk_period;
  t_CT <= '0';
  t_DIN <= '0';
  wait for clk_period;
  t_DIN <= '0';
  wait for clk_period;
  t_DIN <= '0';
  wait for clk_period;
  t_DIN <= '0';
  wait for clk_period;
  t_DIN <= '0';
  wait for clk_period;
  t_DIN <= '1';
  wait for clk_period;
  t_DIN <= '0';
  wait for clk_period;
  t_DIN <= '0';  → Prueba igual que antes
  wait for 1000000 * clk_period;
  -- Piso 1000 0010 (82)
  -- Led DOWN
```

Asignatura: Electrónica Digital
Convocatoria: Julio 2019 Fecha: 27/06/2019

Prueba de Evaluación Continua (PEC)
Duración: 1 hora y 30 minutos

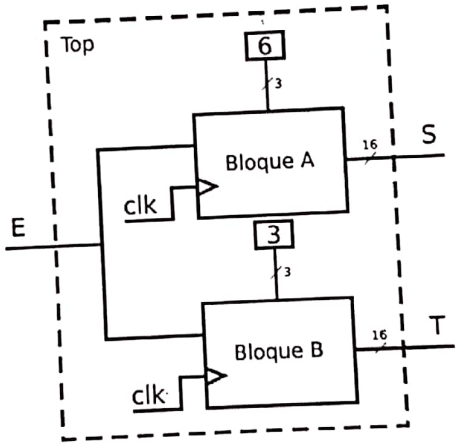
Ejercicio 1 (6 Puntos)

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)

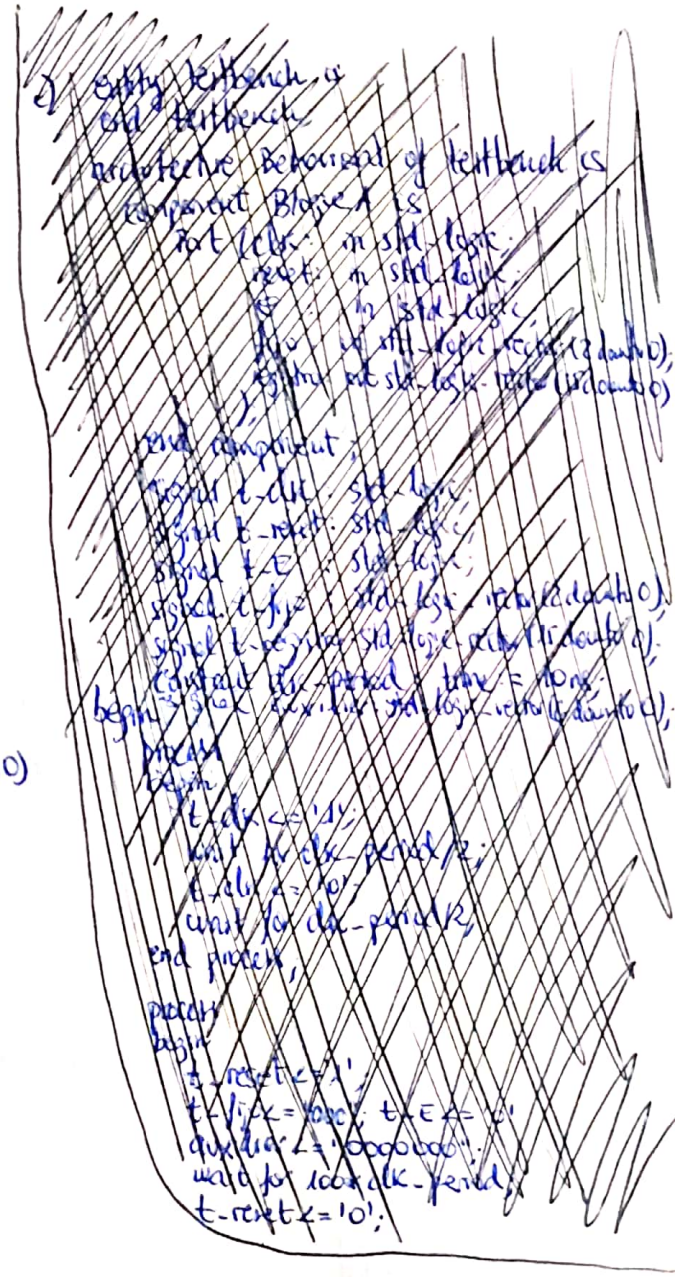
PEC VHDL – 27 Junio 2019

a)
```
entity bloqueA is
  Port ( clk:      in std_logic;
         reset:    in std_logic;
         E :       in std_logic;
         fijo:     in std_logic_vector (2 downto 0);
         registro: out std_logic_vector (15 downto 0)
       );
end bloqueA;

architecture Behavioral of bloqueA is
  --Maquina estados funcionamiento
  type state_t is (STOP, WORKING);
  signal state : state_t;
  signal resetsinc: std_logic;
  signal cargar : std_logic;
  --Contador de 0 a 7
  signal contador : integer range 0 to 7;
  --Variables internas
  signal serie: std_logic_vector (6 downto 0);
  signal interno: unsigned (15 downto 0);
begin
  --Maquina estados funcionamiento
  process (clk, reset)
  begin
    if reset = '1' then
      state <= STOP;
    elsif clk'event and clk = '1' then
      case state is
      when STOP =>
        if E = '1' then
          state <= WORKING;
        end if;
      when WORKING =>
        if contador = 7 then
          state <= STOP;
        end if;
      end case;
    end if;
  end process;

cargar <= '1' when serie (6 downto 4) = direccion
          and contador = 7 else '0';
reset <= '1' when serie (6 downto 4) = "111" and
          contador = 7 else '0';
```

```
--Contador 0 a 7
process (clk, reset)
begin
  if reset = '1' then
    contador <= 0;
  elsif clk'event and clk = '1' then
    if state = STOP then
      contador <= 0;
    else
      contador <= contador + 1;
    end if;
  end if;
end process;
--Registro desplazamiento
process (clk, reset)
begin
  if reset = '1' then
    serie <= (others => '0');
  elsif clk'event and clk = '1' then
    serie <= serie (5 downto 0) & E;
  end if;
end process;
--Cargar el valor numérico o reset síncrono
process (clk, reset)
begin
  if reset = '1' then
    interno <= (others => '0');
  elsif clk'event and clk = '1' then
    if reset = '1' then
      interno <= (others => '0');
    elsif cargar = '1' then
      interno <= interno +
        unsigned(serie(3 downto 0));
    end if;
  end if;
end process;
registro <= std_logic_vector(interno);
end Behavioral;
```

b)

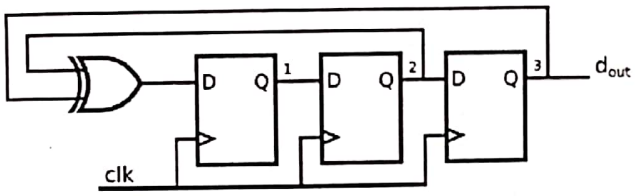
```
entity Top is
  Port ( clk:   in std_logic;
         reset: in std_logic;
         E :    in std_logic;
         S :    out std_logic_vector (15 downto 0);
         T :    out std_logic_vector (15 downto 0)
       );
end Top;

architecture Structural of Top is
  component BlogeA is
    Port ( clk:   in std_logic;
           reset: in std_logic;
           E :    in std_logic;
           fijo:  in std_logic_vector (2 downto 0);
           registro: out std_logic_vector (15 downto 0)
         );
  end component;
begin
  component_6 : BlogeA
    Port map ( clk      => clk,
               reset    => reset,
               E        => E,
               fijo     => "110",
               registro => S
             );
  Component_3 : Blogue B
    Port map ( clk => clk,
               reset => reset,
               E  => E,
               fijo => "011",
               registro => T
             );
end Structural;
```



c)
```
entity testbench is
end testbench;

architecture Behavioral of testbench is
  component BloqueA is
    Port (clk:  in std_logic;
          reset: in std_logic;
          E :  in std_logic;
          fijo: in std_logic_vector (2 downto 0);
          registro: out std_logic_vector (15 downto 0)
         );
  end component;

  signal t_clk    : std_logic;
  signal t_reset  : std_logic;
  signal t_E      : std_logic;
  signal t_fijo   : std_logic_vector (2 downto 0);
  signal t_registro : std_logic_vector (15 downto 0);
  signal t_auxiliar : std_logic_vector (6 downto 0);
  constant clk_period : time := 10 ns;
begin

  proc_clk: process
  begin
    t_clk <= '1';
    wait for clk_period/2;
    t_clk <= '0';
    wait for clk_period/2;
  end process;

  proc_stimuli: process
  begin
    t_reset <= '1';
    t_fijo <= "000";
    t_auxiliar <= "0000000";
    t_E <= '0';
    wait for 100 * clk_period;
    t_reset <= '0';
    wait for 100 * clk_period;

    -- Validacion de un numero
    t_fijo <= "011";
    t_auxiliar <= "0011001";
    t_E <= '1';
    wait for clk_period;
    for i in 6 downto 0 loop
      t_E <= t_auxiliar(i);
      wait for clk_period;
    end loop;
    t_E <= '0';
    wait for 100 * clk_period;

    -- Reset coincidente con "110"
    t_fijo <= "110";
    t_auxiliar <= "1100011";
    t_E <= '1';
    wait for clk_period;
    for i in 6 downto 0 loop
      t_E <= t_auxiliar(i);
      wait for clk_period;
    end loop;
    t_E <= '0';
    wait for 100 * clk_period;

  uut: BloqueA
    port map (clk    => t_clk,
              reset  => t_reset,
              E      => t_E,
              fijo   => t_fijo,
              registro => t_registro
             );
end Behavioral;
```

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. Cambios en rojo.

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.

a)
```
entity of circuito is
  Port (clk :   in std_logic;
        reset:  in std_logic;
        dout :  out std_logic
       );
end circuito;

architecture Behavioral of circuito is
   Signal nbits :  integer := 3 ; 65
   Signal inicioxor:  integer := 2; 47
   Signal finalxor:  integer := 3; 65
   Signal din : std_logic;
   Signal registro : std_logic_vector (nbits-1 downto 0);
begin
  -- Registro
   process (clk,reset)
   begin
      if reset='1' then
         registro <= (0 => '1', others => '0');
      elsif clk'event and clk='1' then
          if enable = '1' then
             registro <= registro (nbits-2 downto 0) & din;
      end if;
   end process;
   din <= registro (inicioxor-1) xor registro (finalxor-1);
   dout <= registro (nbits-1);
end Behavioral;
```

c) Se trata de hacer un contador de 1 segundo que genera una señal de enable cuando acabe.

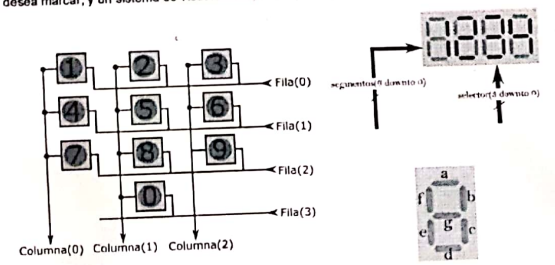
```
signal cont1s : integer range 0 to MAX-1;
constant MAX : integer := 8*10**6;
signal enable : std_logic;

process (clk,reset)
begin
  if reset='1' then
     cont1s <= 0;
  elsif clk'event and clk='1' then
     if cont1s = MAX-1 then
        cont1s <= 0;
     else
        cont1s <= cont1s + 1;
     end if;
  end if;
end process;
enable <= '1' when cont1s = MAX-1 else '0';
```

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.

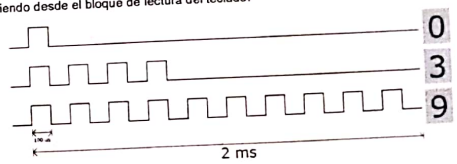
El sistema de control digital que se pretende diseñar constará de los siguientes bloques:

- **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 Digito.

- **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.

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)

- **Bloque de Generación de la Señal de llamada**: Recibirá también como entradas las señales ***Digito*** 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

PEC VHDL - 2 NOVIEMBRE 2019

a)
```
entity lecturaTeclado is
  Port (clk     : in std_logic;
        reset   : in std_logic;
        Columna : in std_logic_vector (2 downto 0);
        Fila    : out std_logic_vector (3 downto 0);
        Digito  : out std_logic_vector (3 downto 0);
        Valido  : out std_logic
       );
end lecturaTeclado;

architecture Behavioral of lecturaTeclado is
  -- Barrido de filas
  signal barrido : integer range 0 to 3;
  -- Detección botón pulsado
  signal colf0 : std_logic_vector (2 downto 0);
  signal colf1 : std_logic_vector (2 downto 0);
  signal colf2 : std_logic_vector (2 downto 0);
  signal colf3 : std_logic_vector (2 downto 0);

begin
  -- Barrido filas
  process (clk, reset)
  begin
    if reset = '1' then
      barrido <= 0;
    elsif clk'event and clk = '1' then
      if barrido = 3 then
        barrido <= 0;
      else
        barrido <= barrido + 1;
      end if;
    end if;
  end process;
  with barrido select
    Fila <= "0001" when 0,
            "0010" when 1,
            "0100" when 2,
            "1000" when 3,
            "----" when others;

  -- Detección del botón pulsado
  process (clk, reset)
  begin
    if reset = '1' then
      colf0 <= (others => '0');
      colf1 <= (others => '0');
      colf2 <= (others => '0');
      colf3 <= (others => '0');
    elsif clk'event and clk = '1' then
      case barrido is
        when 0 =>
          colf0 <= Columna;
        when 1 =>
          colf1 <= Columna;
        when 2 =>
          colf2 <= Columna;
        when 3 =>
          colf3 <= Columna;
      end case;
    end if;
  end process;
  -- Salida
  Digito <= "0001" when barrido = 0 and Columna = "001" else
            "0010" when barrido = 0 and Columna = "010" else
            "0011" when barrido = 0 and Columna = "100" else
            "0100" when barrido = 1 and Columna = "001" else
            "0101" when barrido = 1 and Columna = "010" else
            "0110" when barrido = 1 and Columna = "100" else
            "0111" when barrido = 2 and Columna = "001" else
            "1000" when barrido = 2 and Columna = "010" else
            "1001" when barrido = 2 and Columna = "100" else
            "0000" when barrido = 3 and Columna = "010" else
            "1111";
  -- Valido (nuevo dígito)
  Valido <= '1' when Columna /= "000" and
    ((barrido = 0 and Columna /= colf0) or
     (barrido = 1 and Columna /= colf1) or
     (barrido = 2 and Columna /= colf2) or
     (barrido = 3 and Columna /= colf3)) else '0';
end Behavioral;
```

b)
```
entity AlmacVis 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 AlmacVis;

architecture Behavioral of AlmacVis is
    -- Refresco displays
    constant MAX : integer := 31250;
    signal contREF : integer range 0 to MAX-1;
    signal refresco : std_logic;
    -- Multiplexor
    signal contMX : unsigned (1 downto 0);
    signal digitoMX: std_logic_vector (3 downto 0);
    -- Digitos
    signal index: integer range 0 to 3;
    signal digito0 : std_logic_vector (3 downto 0);
    signal digito1 : std_logic_vector (3 downto 0);
    signal digito2 : std_logic_vector (3 downto 0);
    signal digito3 : std_logic_vector (3 downto 0);
begin
  -- Actualizacion indice del digito
  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;

  case index is
    when 0 =>
      digito0 <= Digit;
    when 1 =>
      digito1 <= Digit;
    when 2 =>
      digito2 <= Digit;
    when 3 =>
      digito3 <= Digit;
  end case;

  -- Refresco displays
  process (clk, reset)
  begin
    if reset = '1' then
      contREF <= 0;
    elsif clk'event and clk = '1' then
      if contREF = MAX-1 then
        contREF <= 0;
      else
        contREF <= contREF + 1;
      end if;
    end if;
  end process;
  refresco <= '1' when contREF = MAX-1 else '0';
  -- Multiplexor
  process (clk, reset)
  begin
    if reset = '1' then
      contMX <= (others => '0');
    elsif clk'event and clk = '1' then
      if refresco = '1' then
        if contMX = 3 then
          contMX <= 0;
        else
          contMX <= contMX + 1;
        end if;
      end if;
    end if;
  end process;
  -- Selector
  with contMX select
    selector <= "0001" when "00",
                "0010" when "01",
                "0100" when "10",
                "1000" when "11",
                "----" when others;
  -- Multiplexor digitos
  with contMX select
    digitoMX <= digito0 when "00",
                digito1 when "01",
                digito2 when "10",
                digito3 when "11",
                "----" when others;
  -- BCD -> 7 segmentos
  with digitoMX select                    '1' - encendido
  Segmentos <= "1111110" when "0000",     '0' - apagado
               "0110000" when "0001",
               "1101101" when "0010",
               "1111001" when "0011",
               "0110011" when "0100",
               "1011011" when "0101",
               "1011111" when "0110",
               "1110000" when "0111",
               "1111111" when "1000",
               "1110011" when "1001",
               "0000000" when others;
end Behavioral;
```

c)
```
entity llamada is
    Port (clk     : in std_logic;
          reset   : in std_logic;
          Digito  : in std_logic_vector (3 downto 0);
          Valido  : in std_logic;
          Marcado : out std_logic;
         );
end llamada;

architecture Behavioral of llamada is
    -- Temporizador 100 µs
    constant MAX: integer := 12500;
    signal cont_TEMP : integer range 0 to MAX-1;
    signal ovf_TEMP : std_logic;
    signal enable_TEMP : std_logic;
    -- Maquina estados funcionamiento
    type state_t is (STOP, WORKING);
    Signal state : state_t;
    Signal send : std_logic_vector (3 downto 0);
    -- Señales internas
    signal contador: integer range 0 to 19;
    signal aux: std_logic;
begin
    --Temporizador 100 µs
    process (clk, reset)
    begin
        if reset = '1' then
            cont_TEMP <= 0;
        elsif clk'event and clk='1' then
            if enable_TEMP = '1' then
                if cont_TEMP = MAX-1 then
                    cont_TEMP <= 0;
                else
                    cont_TEMP <= cont_TEMP+1;
                end if;
            end if;
        end if;
    end process;

    ovf_TEMP <= '1' when cont_TEMP = MAX-1
                and enable_TEMP = '1' else '0';

    --Maquina estados
    process (clk, reset)
    begin
        if reset = '1' then
            state <= STOP;
            Send <= (others => '0');
        elsif clk'event and clk='1' then
            case state is
                when STOP =>
                    if Valido = '1' then
                        state <= WORKING;
                        send <= Digito;
                    end if;
                when WORKING =>
                    if contador = 19 and ovf_TEMP = '1' then
                        state <= STOP;
                        send <= (others => '0');
                    end if;
            end case;
        end process;
        enable_TEMP <= '1' when state = WORKING else '0';
    --Generacion tono
    process (clk, reset)
    begin
        if reset = '1' then
            aux <= '0';
            contador <= 0;
        elsif clk'event and clk='1' then
            if ovf_TEMP = '1' then
                aux <= not aux;
                if contador = 19 then
                    contador <= 0;
                else
                    contador <= contador + 1;
                end if;
            end if;
        end if;
    end process;
    Marcado <= aux when
        contador < (2*unsigned(send)+2) else '0';
end Behavioral;
```

INDUSTRIALES
ETSII | UPM

1º Apellido
2º Apellido
Nombre
Nº de Matricula
Nº de Grupo
Asignatura
Especialidad
Año de carrera
Fecha

EJERCICIO
Hoja nº
CALIFICACIÓN

VHDL
- Entity → Conexiones (entradas y salidas del circuito).
- Architecture → Comportamiento o estructura del circuito.

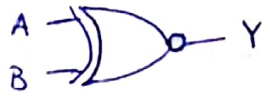
Puertas Lógicas

A, B, Y son de tipo bit ('0' ó '1')

Puerta NAND


```
entity puerta_NAND is
  Port ( A: in  bit;
         B: in  bit;
         Y: out bit
         );
end puerta_NAND;

architecture Behavioral of puerta_NAND is
begin
① [Y<= A nand B;]
② [Y<= '1' when A='0' and B='0' else
      '1' when A='0' and B='1' else
      '1' when A='1' and B='0' else
      '0';]
end Behavioral;
```

Puerta XNOR



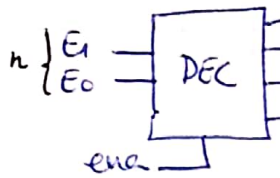

```
entity puerta_XNOR
  Port ( A: in  bit;
         B: in  bit;
         Y: out bit
         );
end puerta_XNOR;

architecture Behavioral of puerta_XNOR is
begin
① [Y<= ((not A) and (not B) or (A and B);]
② [Y<= '1' when A='0' and B='0' else
       '0' when A='0' and B='1' else
       '0' when A='1' and B='0' else
       '1';]
end Behavioral;
```

NOTA:
El tipo de dato 'bit' no cubre la realidad de la electrónica digital en todos sus aspectos → std_logic : '1', '0', 'Z', 'X', 'U', '-', 'L', 'H', 'W'

- Z: alta impedancia
- X: conflicto
- U: undefined
- -: don't care
- L: '0' debil
- H: '1' debil
- 'W': conflicto débil

DECODIFICADORES

→ DECODIFICADOR 2 a 4 CON ENABLE



en	E1	E0	S0	S1	S2	S3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

```
entity DEC2a4 is
  Port ( E : in std_logic_vector (1 downto 0);
         S : out std_logic_vector (3 downto 0);
         ena : in std_logic
       );
end DEC2a4;
```

std_logic_vector → Bus (Conjunto de señales numeradas que se usan generalmente juntas)

```
architecture Behavioral of DEC2a4 is
  Signal interna : std_logic_vector (3 downto 0);
```

Se pueden declarar señales internas dentro de la arquitectura

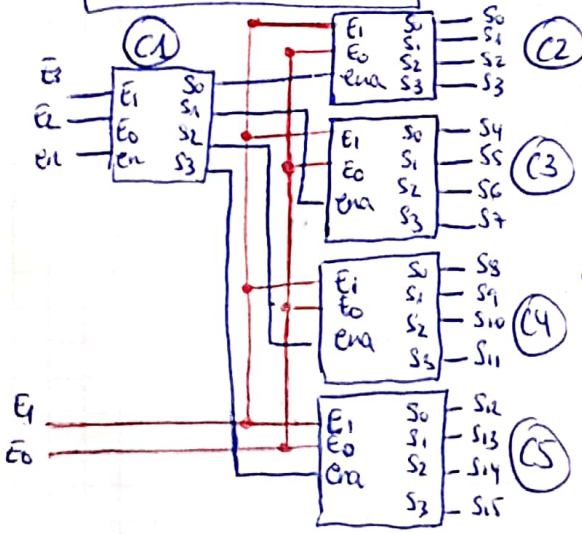
```
begin
interna <= "0001" when E = "00" else
           "0010" when E = "01" else
           "0100" when E = "10" else
           "1000" when E = "11" else
           "----" when others;

S <= interna when ena = '1' else "0000"
end Behavioral;
```

Asignaciones recurrentes (da igual el orden)

(2)

```
with E select
  S <= "0001" when "00",
       "0010" when "01",
       "0100" when "10",
       "1000" when "11",
       "----" when others;
```

→ DECODIFICADOR GRANDE




```
entity DEC4a16 is
  Port (E : in std_logic_vector (3 downto 0);
        S : out std_logic_vector (15 downto 0);
        en : in std_logic
        );
end DEC 4a16;

architecture Structural of DEC4a16 is
  component DEC2a4 is
    Port (E : in std_logic_vector (1 downto 0);
          S : out std_logic_vector (3 downto 0);
          ena : in std_logic
          );
  end component;
  signal enaint : std_logic_vector (3 downto 0);
```

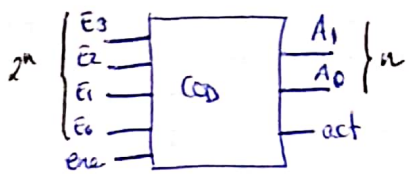
```
begin
C1 : DEC2a4 port map (E(3 downto 2), enaint, en);
C2 : DEC2a4 port map (E(1 downto 0), S(3 downto 0), enaint(0));
C3 : DEC2a4 port map (E(1 downto 0), S(7 downto 4), enaint(1));
C4 : DEC2a4 port map (E(1 downto 0), S(11 downto 8), enaint(2));
C5 : DEC2a4 port map (E(1 downto 0), S(15 downto 12), enaint(3));
end Structural;
```

INDUSTRIALES
ETSII | UPM

1º Apellido
2º Apellido
Nombre
Nº de Matrícula
Nº de Grupo
Asignatura
Especialidad
Año de carrera
Fecha

EJERCICIO
Hoja nº
CALIFICACIÓN

CODIFICADORES

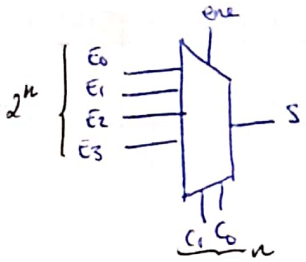


ena	E0	E1	E2	E3	A1	A0	act	
0	x	x	x	x	0	0	0	deshabilitado
1	0	0	0	0	0	0	0	inactivo
1	x	x	x	1	1	1	1	activo
1	x	x	1	0	1	0	1	activo
1	x	1	0	0	0	1	1	activo
1	1	0	0	0	0	0	1	activo

```
entity COD4a2 is
  Port (E: in  std_logic_vector (3 downto 0);
        A: out std_logic_vector (1 downto 0);
        ena: in std_logic;
        act: out std_logic
        );
end COD4a2

architecture Behavioral of COD4a2 is
    signal interna : std_logic_vector (1 downto 0);

begin
    interna <= "11" when E(3) = '1' else
               "10" when E(2) = '1' else
               "01" when E(1) = '1' else
               "00";

    S <= interna when ena = '1' else "00";
    act <= '1' when ena = '1' and E /= "0000" else '0';
end Behavioral;
```

→ la prioridad va marcada por el orden de interpolación de la else.

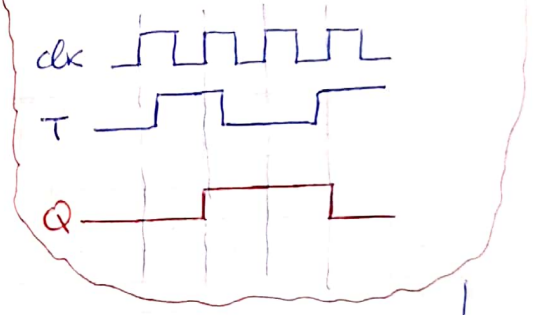
MULTIPLEXORES




```
entity MUX4a1 is
  Port ( E: in std_logic_vector (3 downto 0);
         C: in std_logic_vector (1 downto 0);
         ena: in std_logic;
         S: out std_logic
         );
end MUX4a1;
architecture Behavioral of MUX4a1 is
   Signal interna : std_logic;
begin
  with C select
     interna <= E(0) when "00",
                E(1) when "01",
                E(2) when "10",
                E(3) when "11",
                '_' when others;

     S<= interna when ena = '1' else '0';
end Behavioral;
```

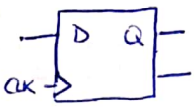
NOTA: Biestable T (Toggle)

Si T='1' → Q cambia su valor
Si T='0' → Q no cambia su valor

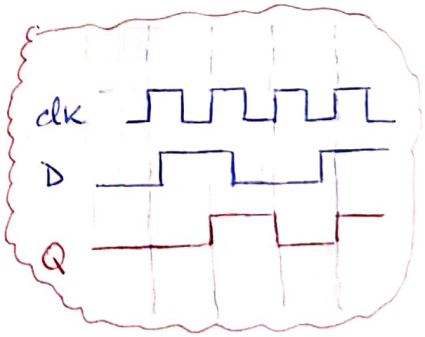


BIESTABLES

→ BIESTABLE D



Si D=0 → Q=0
Si D=1 → Q=1




```
/* Ya hay una entidad y estructura */
process (clk, reset)   lista sensibilidad
begin
  if (reset = '1') then
      Q<= 0;
  elsif (clk 'event and clk = '1') then
      Q<= D;
  end if;
end process;
```

Fuera de la parte síncrona (en la parte asíncrona solo hacemos la inicialización)

DIVISOR DE FRECUENCIA

Genera una señal dirigida al control de aquellas tareas periódicas que se realizan a distinta frecuencia de la de reloj del sistema.

Ejemplo: Quiero generar una señal de 1Hz partiendo de un reloj a 125 MHz.
(Es lo mismo que generar un contador de 1 segundo).

```
entity freqdiv is
    port ( clk : in std_logic;
           reset : in std_logic;
           Enable : in std_logic;
           Signal: out std_logic);
end freqdiv;
architecture Behavioral of freqdiv is
    Constant MAX_1Hz : integer := 125 000 000;
signal cnt_1Hz :      integer range 0 to MAX_1Hz-1;
signal en_1Hz :       std_logic;
signal ovf_1Hz :      std_logic;
begin
  process (clk, reset)
    begin
      if reset = '1' then
          cnt_1Hz <= 0;
      elsif clk'event and clk='1' then
          if en_1Hz = '1' then
              if cnt_1Hz = MAX_1Hz-1 then
                  cnt_1Hz <= 0;
              else
                  cnt_1Hz <= cnt_1Hz + 1;
              end if;
          end if;
      end if;
  end process;
```

$1 \cdot X = 125000000 \rightarrow X = 125000000$
Si yo quisiera una señal a 4KHz
$4000 \cdot X = 125000000 \rightarrow X = 31250$
Por lo tanto: integer := 31250

```
en_1Hz <= Enable;
ovf_1Hz <= '1' when
  (cnt_1Hz = MAX_1Hz-1) and
  en_1Hz = '1' else '0';
signal <= ovf_1Hz;
end Behavioral;
```

CONTADORES BCD

Útiles cuando después debemos implementar un multiplexor o un decodificador BCD (parecido a un prefetch).
Ejemplo: Contador de segundos (de 0 a 60).

```
signal cnt_s : unsigned (3 downto 0);
signal en_s  : std_logic;
signal ovf_s : std_logic;
```

```
cnt_ds : unsigned (3 downto 0);
en_ds  : std_logic;
ovf_ds : std_logic;
```

```
process (clk, reset)
begin
  if reset = '1' then
    cnt_s <= (others => 0);
  elsif en_s = '1' then
    if cnt_s = 9 then
      cnt_s <= 0;
    else
      cnt_s <= cnt_s + 1;
    end if;
  end if;
end process;
ovf_s <= '1' if cnt_s = 9 and
         en_s = '1' else '0';
en_ds <= ovf_s;
```

```
process (clk, reset)
begin
  if reset = '1' then
    cnt_ds <= (others => 0);
  elsif en_ds = '1' then
    if cnt_ds = 5 then
      cnt_ds <= 0;
    else
      cnt_ds <= cnt_ds + 1;
    end if;
  end if;
end process;
ovf_ds <= '1' if cnt_ds = 5 and
          en_ds = '1' else '0';
en_m <= ovf_ds
```

Si siguiera con los minutos

DECODIFICADOR BCD

segmentos (6 downto 0)

a → a
b → f | g | b
c → g
d →
e → e | | c
f →
g → d

'0' → encendido
'1' → apagado

```
with (variable que entra al decodificador) select
  salida <= "1001111" when "0001", (1)
            "0010010" when "0010", (2)
            "0000110" when "0011", (3)
            "1001100" when "0100", (4)
            "0100100" when "0101", (5)
            "0100000" when "0110", (6)
            "0001111" when "0111", (7)
            "0000000" when "1000", (8)
            "0001100" when "1001", (9)
            "0000001" when "0000", (0)
            "-------" when others;
```

a b c d e f g

INDUSTRIALES
ETSII | UPM

1º Apellido
2º Apellido
Nombre
Nº de Matrícula
Nº de Grupo
Asignatura
Especialidad
Año de carrera
Fecha

EJERCICIO
Hoja nº
CALIFICACIÓN

BOTÓN → Generalmente es una entrada total, y el le responsable de mandar el inicio del funcionamiento de un elemento.

Ejemplo: Botón que mande el inicio de un divisor de frecuencia (14Hz).

```
entity fantastics
    clk:   in std_logic;
    reset: in std_logic;
    start: in std_logic;
    led:   out std_logic_vector (7 downto 0);
end fantastic;
```

declarado en la arquitectura (parte del fregador)

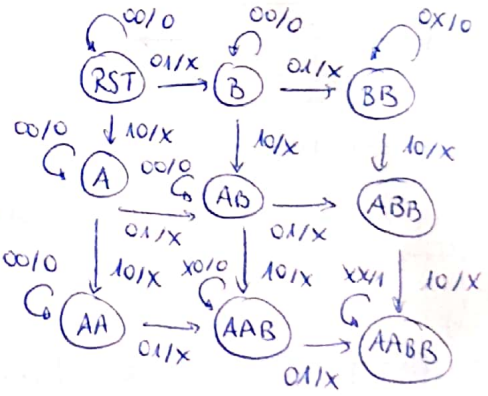
```
process (clk, reset)
  begin
  if reset = '1' then
     en_14Hz <= '0';
  elsif clk'event and clk = '1' then
     if start = '1' then
        en_14Hz <= '1';
     end if;
  end if;
end process;
```

MÁQUINA DE ESTADOS

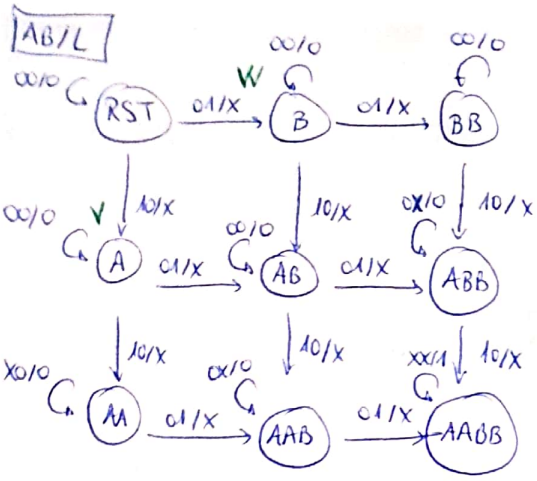
ABIL

```
entity FSM IS
  Port ( A   : in std_logic;
         B   : in std_logic;
         clk : in std_logic;
         reset : in std_logic;
         L   : out std_logic
       );
end FSM;
```

No las pedire usar como estados después, ya que el programa "se raya".




```
architecture Behavioral of FSM is
① /* Declaración de Tipo */
   type state_t is (RST, V, BB, W, AB, ABB, AA, AAB, AABB);
   signal state : state_t;
```




```
begin
/* Actualización del estado */
  procen (clk, reset)
  begin
  if reset = '1' then
     State <= RST;
  elsif clk'event and clk = '1' then
  case state is
→ when RST =>
     if B = '1' then
        state <= W;
     elsif A = '1' then
        state <= V;
     end if;

→ when W =>
     if B = '1' then
        state <= BB;
     elsif A = '1' then
        state <= AB;
     end if;

→ when BB =>
     if A = '1' then
        state <= ABB;
     end if;

→ when V =>
     if B = '1' then
        state <= AB;
     elsif A = '1' then
        state <= AA;
     end if;

→ when AB =>
     if B = '1' then
        state <= ABB;
     if A = '1' then
        state <= AAB;
     end if;

→ when ABB =>
     if A = '1' then
        state <= AABB;
     end if;

→ when AA =>
     if B = '1' then
        state <= AAB;
     end if;

→ when AAB =>
     if B = '1' then
        state <= AABB;
     end if;

→ when AABB =>
     end case;
     end if;
     end process;

L = '1' when state = AABB else '0';
end Behavioral;
```

INDUSTRIALES
ETSII | UPM

1º Apellido
2º Apellido
Nombre
Nº de Matrícula
Nº de Grupo
Asignatura
Especialidad
Año de carrera
Fecha

EJERCICIO

Hoja nº

CALIFICACIÓN

TESTBENCH

```
entity testbench is
end testbench;

architecture Behavioral of testbench is
① /* Declarar el programa del que se quiere hacer el testbench como component */
  component practica is
    Port (clk : in std_logic;
          reset : in std_logic;
          start : in std_logic;
          leds : out std_logic_vector (7 downto 0)
          );
  end component;
② /* Declarar las señales de mi testbench (las mismas que en el component)*/
signal test_clk :    std_logic;
signal test_reset :  std_logic;
signal test_start :  std_logic;
signal test_leds :   std_logic_vector (7 downto 0);

    constant clk_period :  time := 8ns;
begin
③ /* Generación del reloj */
    process
    begin
        test_clk <= '1';
        wait for clk_period /2;
        test_clk <= '0';
        wait for clk_period/2;
end process;
```

④ /* Generación de estímulos */

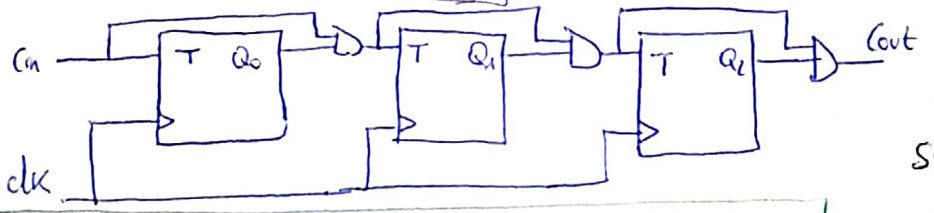
```
process
begin
    test_reset <= '1';
    test_start <= '0';
    wait for 10 * clk_period;
    test_reset <= '0';
    wait for 10 * clk_period;
    test_start <= '1';
    wait for 10 * clk_period;
    test_start <= '0';
end process;
```

⑤ /* Instanciar el componente */

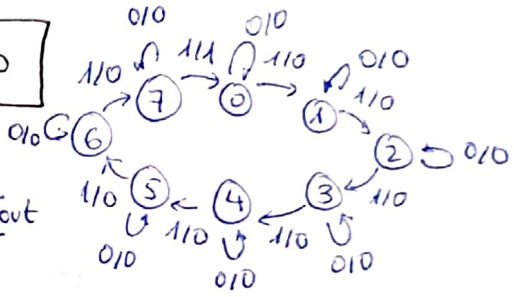
```
uut : practica
  Port map ( clk => test_clk,
             reset => test_reset,
             start => test_start,
             leds => test_leds,
           );

end Behavioral;
```

CONTADORES CON REGISTRO DE DESPLAZAMIENTO

→ Contador binario síncrono:



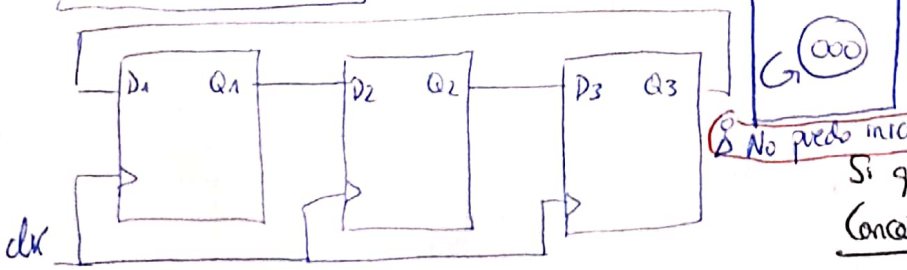
* Para generar una secuencia de 8 números → 3 bits *



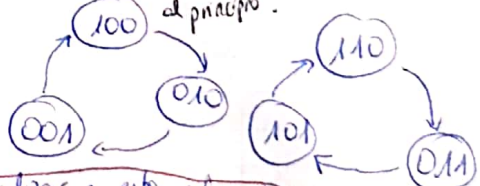
Si quiero n números: x bits

$n = 2^x$

→ Contador en anillo:



Cojo los dos de la izq. y le añado el de la derecha al principio.



⚠ No puedo inicializar a estos valores ⚠

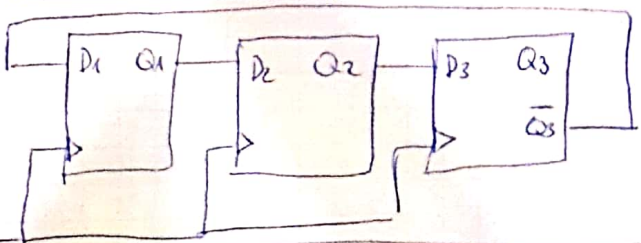
Si quiero n números: n bits

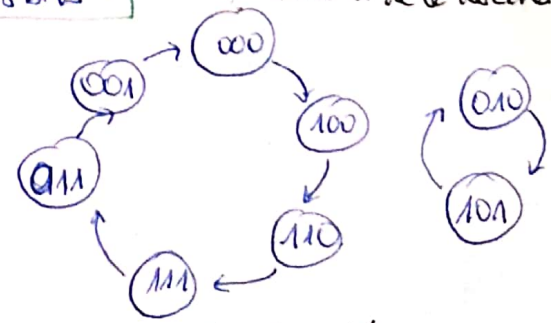
Concatenación.

* Para generar una secuencia de 8 números → 8 bits *

Cojo los dos de la izquierda y le añado el de la derecha negado.

→ Contador Johnson:





* Para generar una secuencia de 8 números → 4 bits *

Concatenación

Si quiero n números: n/2 bits