INGENIERÍA DE COMPUTADORES 3

Solución al examen de Junio 2017, Primera Semana

PREGUNTA 1 (2 puntos)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `x1`, `x2`, `x3`, `x4` y `x5` entre los instantes 0 y 100 ns.

```
entity crono1 is
end entity crono1;

architecture crono1 of crono1 is
  signal x1, x2, x3, x4, x5 : std_logic;
begin
  x1 <= '1', '0' after 10 ns,
        '1' after 25 ns,'0' after 30 ns,
        '1' after 40 ns;
  Proc1: process
  begin
    x2 <= '1';
    wait for 10 ns;
    x2 <= '0';
    wait for 15 ns;
    x2 <= '1';
    wait for 20 ns;
    x2 <= '0';
  end process;
   x3 <= (x1 xor x2) after 15 ns;
   x4 <= x1 xor x2;
   x5 <= transport (x1 xor x2) after 15 ns;
end architecture crono1;
```

Solución a la Pregunta 1

En la Figura 1.1 se muestra el cronograma de evolución de las señales.

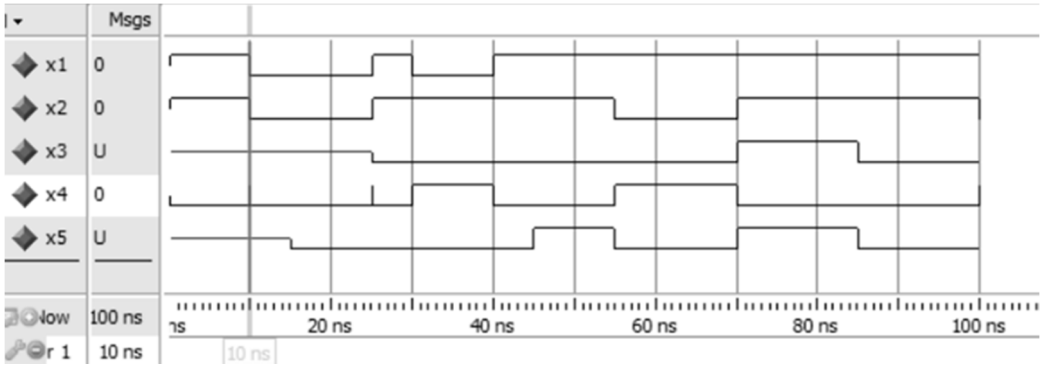


Figura 1.1: Cronograma de evolución de las señales.

PREGUNTA 2 (2 puntos)

Escriba en VHDL, de las formas que se detallan a continuación, la **architecture** que describe el comportamiento de un circuito combinacional codificador de 4 a 2 con prioridad. A continuación, se muestran la **entity** del circuito y su tabla de la verdad.

```
entity codificadorPrioridad4a2 is
   port (  codigo     : out std_logic_vector(1 downto 0);
           activo     : out std_logic;
           x          : in  std_logic_vector(3 downto 0) );
end entity codificadorPrioridad4a2;
```

x	codigo	activo
1 – – –	11	1
0 1 – –	10	1
0 0 1 –	01	1
0 0 0 1	00	1
0 0 0 0	00	0

En el código VHDL de la **architecture**, emplee para evaluar la señal `codigo`:

2.a) (0.5 puntos) Una asignación concurrente condicional (**when** - **else**).

2.b) (0.5 puntos) Una asignación concurrente de selección (**with** - **select**).

2.c) (0.5 puntos) Una sentencia **if**.

2.d) (0.5 puntos) Una sentencia **case**.

Solución a la Pregunta 2

La **architecture** que describe el circuito mediante la asignación concurrente condicional, la asignación concurrente de selección, la sentencia **if** y la sentencia **case** se muestran en Código VHDL 1.1–Código VHDL 1.4, respectivamente.

```
library IEEE;
use IEEE.std_logic_1164.all;

architecture codPrior4a2WHEN of codificadorPrioridad4a2 is
begin
   codigo <= "11" when ( x(3) = '1' ) else
             "10" when ( x(2) = '1' ) else
             "01" when ( x(1) = '1' ) else
             "00";
   activo <= x(3) or x(2) or x(1) or x(0);
end architecture codPrior4a2WHEN;
```

Código VHDL 1.1: Architecture empleando la asignación concurrente condicional.

```
library IEEE;
use IEEE.std_logic_1164.all;

architecture codPrior4a2WITH of codificadorPrioridad4a2 is
begin
   with x select
      codigo <= "11" when "1000" | "1001" | "1010" | "1011" |
                          "1100" | "1101" | "1110" | "1111",
                "10" when "0100" | "0101" | "0110" | "0111",
                "01" when "0010" | "0011",
                "00" when others;
   activo <= x(3) or x(2) or x(1) or x(0);
end architecture codPrior4a2WITH;
```

Código VHDL 1.2: Architecture empleando la asignación concurrente de selección.

```
library IEEE;
use IEEE.std_logic_1164.all;

architecture codPrior4a2IF of codificadorPrioridad4a2 is
begin
    process ( x )
    begin
        if ( x(3) = '1' ) then
            codigo <= "11";
        elsif ( x(2) = '1' ) then
            codigo <= "10";
        elsif ( x(1) = '1' ) then
            codigo <= "01";
        else
            codigo <= "00";
        end if;
        activo <= x(3) or x(2) or x(1) or x(0);
    end process;
end architecture codPrior4a2IF;
```

Código VHDL 1.3: Architecture empleando la sentencia **if**.

```
library IEEE;
use IEEE.std_logic_1164.all;

architecture codPrior4a2CASE of codificadorPrioridad4a2 is
begin
    process ( x )
    begin
        case x is
            when "1000" | "1001" | "1010" | "1011" |
                 "1100" | "1101" | "1110" | "1111"   =>
               codigo <= "11";
            when "0100" | "0101" | "0110" | "0111"   =>
               codigo <= "10";
            when "0010" | "0011"                     =>
               codigo <= "01";
            when others                             =>
               codigo <= "00";
        end case;
        activo <= x(3) or x(2) or x(1) or x(0);
    end process;
end architecture codPrior4a2CASE;
```

Código VHDL 1.4: Architecture empleando la sentencia **case**.

PREGUNTA 3 (2 puntos)

Programe el banco de pruebas del circuito combinacional que ha diseñado en la Pregunta 2. Explique detalladamente cómo el programa de test comprueba de manera sistemática el funcionamiento del circuito. El banco de pruebas debe comprobar que los valores obtenidos de la UUT coinciden con los esperados, mostrando el correspondiente mensaje en caso de que no coincidan. Al final del test, debe mostrarse un mensaje indicando el número total de errores.

Solución a la Pregunta 3

El código VHDL del banco de pruebas se muestra en Código VHDL 1.5–1.6.

```
-------------------------------------------------
-- Banco de pruebas del codificador 4:2 con prioridad
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity bp_codificador_4_2 is
    constant MAX_COMB : integer := 16;       -- Num. combinaciones entrada
    constant DELAY    : time    := 10 ns;    -- Retardo usado en el test
end entity bp_codificador_4_2;

architecture bp_codificador_4_2 of bp_codificador_4_2 is
    -- Salidas UUT
    signal activo           : std_logic;
    signal codigo       : std_logic_vector(1 downto 0);
    -- Entradas UUT
    signal x  : std_logic_vector(3 downto 0);
  component codificadorPrioridad4a2 is
      port ( codigo       : out std_logic_vector(1 downto 0);
             activo       : out std_logic;
             x            : in  std_logic_vector(3 downto 0));
    end component codificadorPrioridad4a2;
begin   -- Cuerpo de la arquitectura
  UUT : component codificadorPrioridad4a2 port map
        (codigo, activo, x);
  main : process is
    variable temp : unsigned (3 downto 0); -- Usado en los cálculos
    variable esperado_activo       : std_logic;
    variable esperado_codigo : std_logic_vector(1 downto 0);
    variable error_count           : integer     := 0;
  begin
```

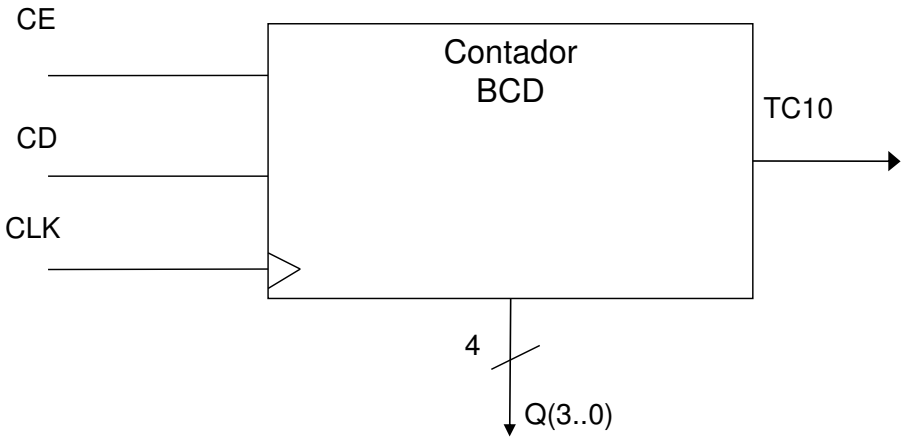
Código VHDL 1.5: Diseño del banco de pruebas del codificador de 4 a 2 con prioridad.

```
    report "Comienza la simulación";
    -- Generar todos los posibles valores de entrada
    for i in 0 to (MAX_COMB-1) loop
       temp := TO_UNSIGNED(i,4);
       x(3) <= std_logic(temp(3));
       x(2) <= std_logic(temp(2));
       x(1) <= std_logic(temp(1));
       x(0) <= std_logic(temp(0));
       -- Calcular el valor esperado
       if (i=0) then
            esperado_activo     := '0';
            esperado_codigo := "00";
       else
            esperado_activo     := '1';
            if     (i=1)   then  esperado_codigo := "00";
            elsif (i<=3) then  esperado_codigo := "01";
            elsif (i<=7) then  esperado_codigo := "10";
            else                  esperado_codigo := "11";
            end if;
       end if;
       wait for DELAY; -- Espera y compara con las salidas de UUT
       if ( esperado_activo /= activo ) then
            report "ERROR en la salida valida. Valor esperado: "  &
                    std_logic'image(esperado_activo)                &
                    ", valor actual: "                                &
                    std_logic'image(activo)                         &
                    " en el instante: "                                &
                    time'image(now);
            error_count := error_count + 1;
       end if;
       if ( esperado_codigo /= codigo ) then
            report "ERROR en la salida codificada. Valor esperado: " &
                    std_logic'image(esperado_codigo(1))           &
                    std_logic'image(esperado_codigo(0))           &
                    ", valor actual: "                                  &
                    std_logic'image(codigo(1))                    &
                    std_logic'image(codigo(0))                    &
                    " en el instante: "                                 &
                    time'image(now);
            error_count := error_count + 1;
       end if;
    end loop; -- Final del bucle for de posibles valores de entrada
    -- Informe del número total de errores
    if (error_count = 0) then
         report "Simulación finalizada sin errores";
    else
         report   "ERROR: Hay "                &
                   integer'image(error_count) &
                   " errores.";
    end if;
    wait;      -- Final de la simulación
  end process main;
end architecture bp_codificador_4_2;
```

Código VHDL 1.6: Continuación del banco de pruebas del codificador de 4 a 2 con prioridad.

PREGUNTA 4 (4 puntos)

Diseñe usando VHDL un circuito contador síncrono BCD cuyo símbolo se muestra en la siguiente figura. El circuito tiene las siguientes señales de entrada: una señal de reloj (`CLK`), una señal de un bit (`CE`) y una señal de un bit (`CD`). Tiene las dos siguientes señales de salida: una señal de 4 bits (`Q`) y una señal de un bit (`TC10`).



El circuito tiene el comportamiento descrito a continuación.

La señal `CD` tiene prioridad sobre la señal `CE`. Mientras la señal de entrada `CD` tiene el valor '1', la señal de salida `Q` tiene el valor "0000".

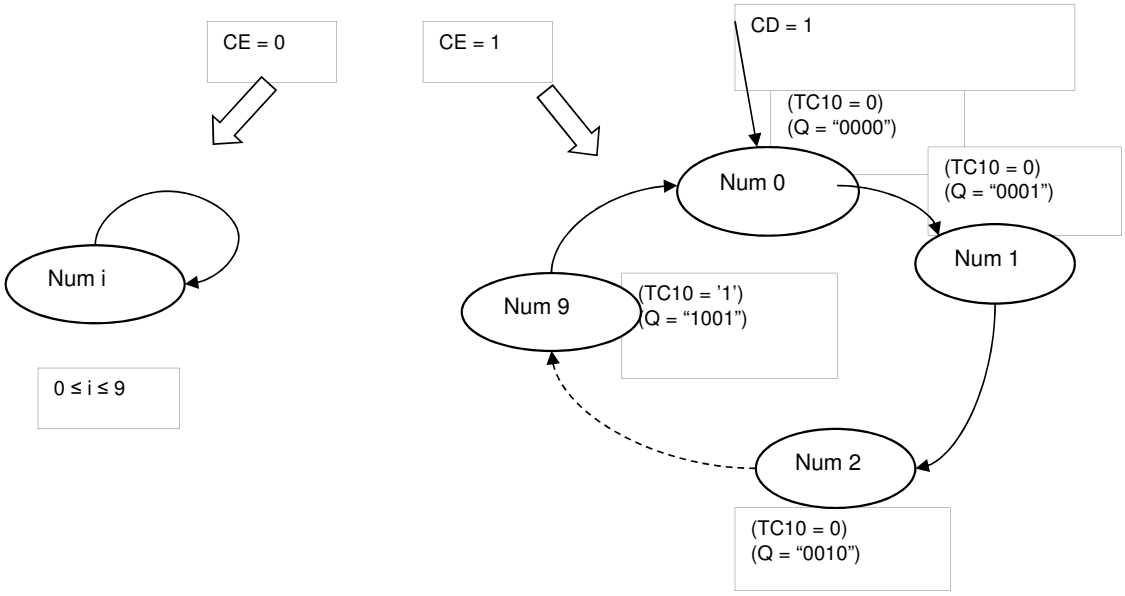
Cuando la señal de entrada `CE` tiene el valor '1', la señal de salida `Q` toma cíclicamente en el flanco de subida de la señal de reloj los valores "0000", "0001", "0010", "0011", "0100", "0101", "0110", "0111", "1000" y "1001". Si la señal de entrada `CE` tiene el valor '0', se mantiene el valor de la señal de salida `Q`.

La señal de salida `TC10` tiene valor '1' mientras la señal `CE` tiene el valor '1' y la señal `Q` tiene el valor "1001".

Escriba en VHDL la **architecture** que describe el comportamiento del circuito en términos de una máquina de Moore. Dibuje el diagrama de estados correspondiente al circuito que ha diseñado.

Solución a la Pregunta 4

El diagrama de estados correspondiente al circuito contador se muestra en la siguiente figura.



El código VHDL correspondiente al contador se muestra en Código VHDL 1.7–1.8.

```
----------------------------------------------------
---Contador BCD implementado como máquina de Moore
----------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

entity contador is port(
     Q       :    out std_logic_vector(3 downto 0);
     TC10    :    out std_logic;
     CLK, CD, CE :    in    std_logic);
end entity contador;

architecture contador of contador is
       signal internal_state: std_logic_vector(3 downto 0);
begin
 Q <= internal_state;
```

Código VHDL 1.7: Contador BCD.

```
--Cálculo del próximo estado
proximo_estado: process (CLK, CD)
begin
     if (CD = '1') then
          internal_state <= "0000";
     elsif (rising_edge(CLK)) then
        if (CE = '1') then
          case internal_state is
                when "0000" =>
                   internal_state <= "0001";
                when "0001" =>
                   internal_state <= "0010";
                when "0010" =>
                   internal_state <= "0011";
                when "0011" =>
                   internal_state <= "0100";
                when "0100" =>
                   internal_state <= "0101";
                when "0101" =>
                   internal_state <= "0110";
                when "0110" =>
                   internal_state <= "0111";
                when "0111" =>
                   internal_state <= "1000";
                when "1000" =>
                    internal_state <= "1001";
                when "1001"=>
                    internal_state <= "0000";
                when others =>
                    internal_state <= "0000";
              end case;
           end if;
     end if;
end process proximo_estado;
SalTC10: process(internal_state, CE)
begin
   if (internal_state = "1001" and CE ='1') then
      TC10 <= '1';
   else TC10 <= '0';
   end if;
end process SalTC10;
end architecture contador;
-----------------------------------------
```

Código VHDL 1.8: Continuación del contador BCD.