INGENIERÍA DE COMPUTADORES III

Solución al Ejercicio de Autocomprobación 8

PREGUNTA 1 (3 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`:

1.a) (0.75 puntos) Una asignación concurrente condicional (**when** - **else**).

1.b) (0.75 puntos) Una asignación concurrente de selección (**with** - **select**).

1.c) (0.75 puntos) Una sentencia **if**.

1.d) (0.75 puntos) Una sentencia **case**.

Solución a la Pregunta 1

La solución a los Apartados 1.a – 1.d se muestra en Código VHDL 1.1–1.4.

```
-------------------------------------------------
architecture codPrior4a2 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 codPrior4a2;
-------------------------------------------------
```

Código VHDL 1.1: Respuesta al Apartado 1.a: diseño de un codificador 4:2 con prioridad, empleando una asignación concurrente condicional (**when** - **else**) para evaluar la señal *codigo*.

```
-------------------------------------------------
architecture codPrior4a2_selec 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 codPrior4a2_selec;
-------------------------------------------------
```

Código VHDL 1.2: Respuesta al Apartado 1.b: diseño de un codificador 4:2 con prioridad, empleando una asignación concurrente de selección (**with** - **select**) para evaluar la señal *codigo*.

```
-------------------------------------------------
architecture codPrior4a2_procIf 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 codPrior4a2_procIf;
-------------------------------------------------
```

Código VHDL 1.3: Respuesta al Apartado 1.c: diseño de un codificador 4:2 con prioridad, empleando una sentencia **if** para evaluar la señal *codigo*.

```
-------------------------------------------------
architecture codPrior4a2_procCase 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 codPrior4a2_procCase;
-------------------------------------------------
```

Código VHDL 1.4: Respuesta al Apartado 1.d: diseño de un codificador 4:2 con prioridad, empleando una sentencia **case** para evaluar la señal *codigo*.

PREGUNTA 2 (2 puntos)

Diseñe en VHDL, usando una asignación concurrente condicional (**when-else**), la **architecture** de un circuito combinacional desplazador de 4 bits de entrada, que realice las operaciones indicadas en la tabla siguiente.

op	Operación
0 0	Desplaza a la izquierda rellenando con '0'
0 1	Desplaza a la derecha rellenando con '0'
1 0	Rota a la izquierda
1 1	Rota a la derecha

A continuación se muestra la **entity** del circuito desplazador de 4 bits:

```
entity desplazador is
   port (  salida     : out std_logic_vector(3 downto 0);
           op         : in  std_logic_vector(1 downto 0);
           entrada    : in  std_logic_vector(3 downto 0) );
end entity desplazador;
```

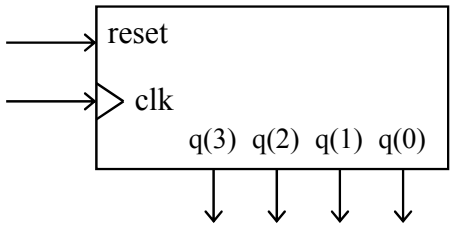
Solución a la Pregunta 2

```
--------------------------------
architecture desplazador of desplazador is
begin

    salida <=   entrada(2 downto 0) & '0'
                 when (op = "00")
                else '0' & entrada(3 downto 1)
                 when (op = "01")
                else entrada(2 downto 0) & entrada(3)
                 when (op = "10")
                else entrada(0) & entrada(3 downto 1);

end architecture desplazador;
--------------------------------------------------
```

Código VHDL 1.5: Diseño del circuito combinacional desplazador de 4 bits de entrada.

PREGUNTA 3 (3 puntos)

Diseñe en VHDL la **architecture** que describe el comportamiento de un circuito contador en anillo de 4 bits, con señal de reset asíncrona. En la figura se muestran los puertos del circuito. Debajo de la figura está la declaración de la **entity** del circuito.




```
entity ring_counter is
   port( q          :   out std_logic_vector(3 downto 0);
         clk, reset :   in  std_logic );
end entity ring_counter;
```

El circuito se comporta de la forma siguiente. Cuando la señal de reset se pone a `1`, la salida del contador se pone al valor `0001`. Mientras la señal de reset está a `0`, el contador funciona de manera síncrona, rotando circularmente hacia la derecha el patrón de bits: `0001`, `1000`, `0100`, `0010`, `0001`, `1000`, `0100`, y así sucesivamente. La operación de desplazamiento se produce en el flanco de subida de la señal de reloj.

Solución a la Pregunta 3

```
----------------------------------------------------
-- Contador en anillo con inicialización asíncrona
library IEEE;
use IEEE.std_logic_1164.all;

entity ring_counter is
   port ( q             : out std_logic_vector(3 downto 0);
          clk, reset : in    std_logic );
end entity ring_counter;

architecture ringCounterReset of ring_counter is
    signal r_reg   : std_logic_vector(3 downto 0);
    signal r_next : std_logic_vector(3 downto 0);
begin
    -- Registro
    process (clk, reset)
    begin
        if (reset='1') then
            r_reg <= "0001";
        elsif  ( rising_edge(clk) ) then
            r_reg <= r_next;
        end if;
    end process;
    -- Lógica cálculo siguiente estado
    r_next <= r_reg(0) & r_reg(3 downto 1);
    -- Lógica de salida
    q <= r_reg;
end architecture ringCounterReset;
```

Código VHDL 1.6: Diseño del circuito contador en anillo de 4 bits, con señal de reset asíncrona.

PREGUNTA 4 (2 puntos)

Programe en VHDL un banco de pruebas para el contador en anillo de 4 bits que ha diseñado al contestar a la Pregunta 3. 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.

Solución a la Pregunta 4

Véase el Código VHDL 1.7. En la Figura 1.1 se muestra el resultado de la simulación del banco de pruebas.

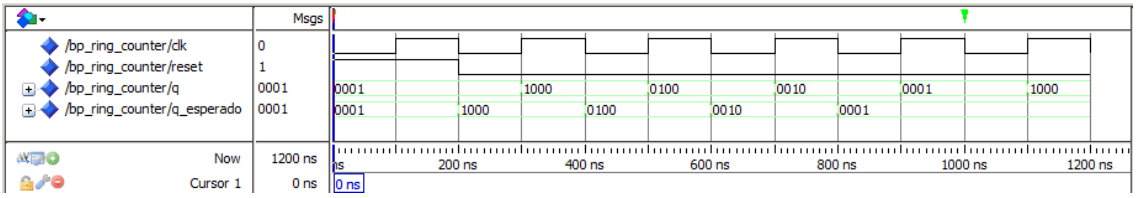


Figura 1.1: Resultado de la simulación del banco de pruebas.

```
-----------------------------------------
-- Banco de pruebas del contador en anillo
library IEEE;
use IEEE.std_logic_1164.all;

entity bp_ring_counter is
end entity bp_ring_counter;

architecture bp_ring_counter of bp_ring_counter is
   constant PERIODO : time := 200 ns;
   signal   q_esperado : std_logic_vector(3 downto 0);
   signal   q          : std_logic_vector(3 downto 0); -- Salida      UUT
   signal   clk        : std_logic := '0';              -- Entradas UUT
   signal   reset      : std_logic;

   component ring_counter is
      port ( q           : out std_logic_vector(3 downto 0);
             clk, reset : in   std_logic );
   end component ring_counter;

begin
   -- Instanciar y conectar UUT
   uut : component ring_counter port map (q=>q,clk=>clk,reset=>reset);

   clk <= not clk after (PERIODO/2);

   gen_vect_test : process is
      variable numTest : integer := 0;
   begin
      report "Comienza el test";
      reset <= '1';   -- Reset
      q_esperado <= "0001";
      wait until rising_edge(clk);
      wait for (PERIODO/2);
      numTest := numTest + 1;
      assert(q=q_esperado)
          report "Error vector " & integer'image(numTest);

      reset <= '0';   -- Activa cuenta síncrona

      for i in 3 downto 0 loop
         q_esperado <= (i=>'1', others =>'0');
         wait until rising_edge(clk);
         wait for (PERIODO/2);
         numTest := numTest + 1;
         assert( q = q_esperado )
             report "Error vector " & integer'image(numTest);
      end loop;

      report "Finaliza el test";
      wait;
   end process gen_vect_test;

end architecture bp_ring_counter;
```

Código VHDL 1.7: Banco de pruebas del circuito contador en anillo de 4 bits.