INGENIERÍA DE COMPUTADORES 3

Solución al examen de Septiembre 2019

PREGUNTA 1 (2 puntos)

1.a) (1 punto) Dibuje el diagrama conceptual correspondiente al fragmento de código *Fragmento 1*.

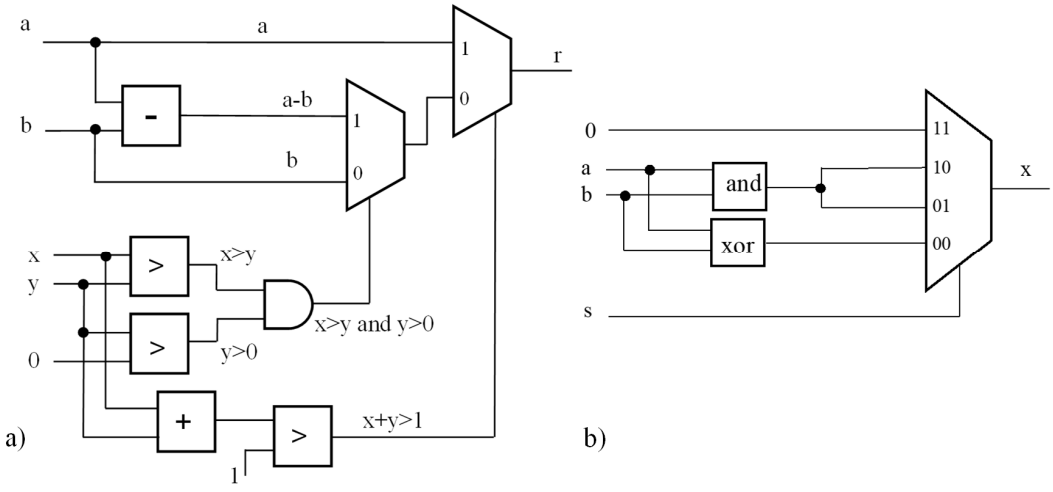
1.b) (1 punto) Dibuje el diagrama conceptual correspondiente al fragmento de código *Fragmento 2*.

```
---- Fragmento 1-------------
signal a, b, r : unsigned (7 downto 0);
signal x, y    : unsigned (3 downto 0);
...
r <=  a when x+y>1 else
      a-b when x>y and y>0 else
      b;
```

```
 ---- Fragmento 2-------------
signal s: std_logic_vector (1 downto 0);
signal a, b, x : std_logic;
...
with s select
   x <= (a xor b) when "00",
        (a and b) when "01"|"10",
         '0' when others;
```

Solución a la Pregunta 1

En la siguiente figura se muestran los diagramas conceptuales correspondientes a Fragmento 1 (a) y a Fragmento 2 (b).



PREGUNTA 2 (3 puntos)

Diseñe en VHDL los circuitos combinacionales indicados a continuación.

2.a) (1 punto) Un circuito con tres entradas y una salida, tal que la salida valga '1' si dos o más de las entradas valen '1'. Realice el diseño empleando un bloque **process** con una o varias sentencias **if**. La **entity** del circuito se muestra a continuación.

```
entity majority is
   port ( Y       : out std_logic;
          A, B, C : in std_logic
         );
end entity majority;
```

2.b) (1 punto) Un circuito con siete entradas y tres salidas, tal que los tres bits de salida indiquen la entrada con mayor prioridad que está a '1'. La **entity** del circuito se muestra a continuación. La entrada `Y7` es la de mayor prioridad y la `Y1` la de menor. Si ninguna de las entradas está a '1', la salida deberá ser "000". Realice el diseño empleando una asignación concurrente de selección.

```
entity priority is
   port ( DOUT : out std_logic_vector(2 downto 0);
          Y1,Y2,Y3,Y4,Y5,Y6,Y7 : in std_logic
         );
end entity priority;
```

2.c) (1 punto) Diseñe el circuito anterior empleando un bloque **process** con una o varias sentencias **if**.

Solución a la Pregunta 2

Los diseños solución de los apartados 2.a, 2.b y 2.c se muestran respectivamente en Código VHDL 1.1– 1.3.

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

entity majority is
    port ( Y       : out std_logic;
           A, B, C : in std_logic
         );
end entity majority;

architecture majority1 of majority is
begin
    process (A,B,C)
    begin
       Y <= '0';
       if ((A='1') and (B='1')) then
          Y <= '1';
       end if;
       if ((A='1') and (C='1')) then
          Y <= '1';
       end if;
       if ((B='1') and (C='1')) then
          Y <= '1';
       end if;
    end process;
end architecture majority1;
```

Código VHDL 1.1: Diseño solución apartado 2.a.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
architecture priority4 of priority is
 signal aux :unsigned(6 downto 0);
begin
   aux <= Y7&Y6&Y5&Y4&Y3&Y2&Y1;
   with to_integer(aux) select
    DOUT <=  "000" when 0,
             "001" when 1,
             "010" when 2|3,
             "011" when 4|5|6|7,
             "100" when 8 to 15,
             "101" when 16 to 31,
             "110" when 32 to 63,
             "111" when  others;
end architecture priority4;
```

Código VHDL 1.2: Diseño solución apartado 2.b.

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

entity priority is
    port ( DOUT                    : out std_logic_vector(2 downto 0);
           Y1,Y2,Y3,Y4,Y5,Y6,Y7 : in  std_logic
          );
end entity priority;


architecture priority2 of priority is
begin
    process (Y1,Y2,Y3,Y4,Y5,Y6,Y7)
    begin
      if    (Y7='1') then DOUT <= "111";
      elsif (Y6='1') then DOUT <= "110";
      elsif (Y5='1') then DOUT <= "101";
      elsif (Y4='1') then DOUT <= "100";
      elsif (Y3='1') then DOUT <= "011";
      elsif (Y2='1') then DOUT <= "010";
      elsif (Y1='1') then DOUT <= "001";
      else DOUT <= "000";
      end if;
    end process;
end architecture priority2;
```

Código VHDL 1.3: Diseño solución apartado 2.c.

PREGUNTA 3 (3 puntos)

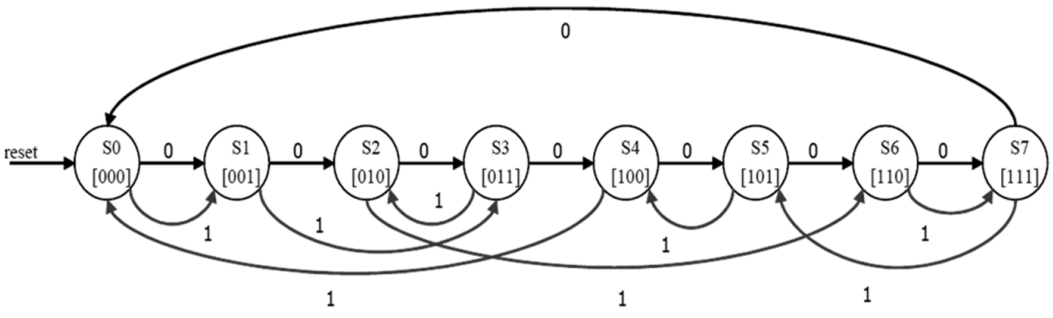
Se quiere diseñar un contador síncrono con señal de reset asíncrona y con control del modo de la cuenta (binaria ascendente o avance según el código de Gray). El circuito tiene las siguientes señales de entrada de un bit: la señal de reloj `clk`, la señal de cambio de modo `M` y señal de reset asíncrona `reset`. El circuito tiene la señal de salida `s` de tres bits.

Si la señal `M` tiene el valor '0', el contador es binario ascendente. Por el contrario, si la señal `M` tiene el valor '1', el contador avanza según el código de Gray. El código de Gray es el siguiente:"000", "001", "011", "010", "110", "111", "101", "100". Por ejemplo, si la salida anterior fue "011" y la señal `M` tiene el valor '1', la señal `s` toma el valor "010". Pero si la señal `M` tiene el valor '0', la señal `s` toma el valor "100".

Programe en VHDL el circuito contador describiendo su comportamiento como una máquina de estados finito de tipo Moore que tiene las transiciones en el flanco de subida de la señal de reloj. Dibuje el diagrama de estado correspondiente al diseño realizado.

Solución a la Pregunta 3

El diagrama de estados del contador se muestra en la siguiente figura.



El código VHDL del contador se muestra en Código VHDL 1.4.

```
 library IEEE;
use IEEE.std_logic_1164.all;
entity contadorBinGray is
    port( s       : out std_logic_vector(2 downto 0);
          M, clk, reset : in std_logic);
end entity contadorBinGray;
architecture contadorBinGray of contadorBinGray is
  signal state: std_logic_vector(2 downto 0);
begin
  proximo_estado : process (clk, reset)
  begin
    if (reset = '1') then
       state <= (others => '0');
    elsif (rising_edge(clk)) then
         case state is
               when "000" =>
                    state <= "001";
               when "001" =>
                    if M = '1' then
                       state <= "011";
                    else
                       state <= "010";
                    end if;
               when "010" =>
                    if M = '1' then
                       state <= "110";
                    else
                       state <= "011";
                    end if;
               when "011" =>
                    if M = '1' then
                       state <= "010";
                    else
                       state <= "100";
                    end if;
               when "100" =>
                    if M = '1' then
                       state <= "000";
                    else
                       state <= "101";
                    end if;
               when "101" =>
                    if M = '1' then
                       state <= "100";
                    else
                       state <= "110";
                    end if;
               when "110" =>
                    state <= "111";
                when others=>
                    if M = '1' then
                       state <= "101";
                    else
                       state <= "000";
                    end if;
               end case;
        end if;
     end process;
  -- logica de salida
    s <= state;
end contadorBinGray;
```

Código VHDL 1.4: Diseño del contador.

PREGUNTA 4 (2 puntos)

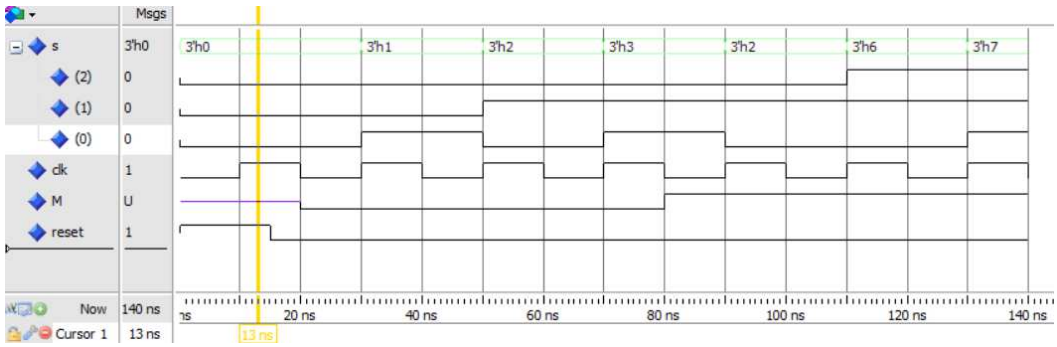
Programe en VHDL un banco de pruebas para el circuito que ha diseñado al contestar a la Pregunta 3. La señal de reloj (`clk`) debe tener un periodo de 20 ns e inicialmente valer '0'. El primer flanco de subida de la señal de reloj se ha de producir en el instante 10 ns. El programa de test debe realizar consecutivamente las acciones siguientes:

1. *Reset*. La señal de reset ha de tener el valor '1' durante los primeros 15 ns.
2. *Mantener la cuenta binaria ascendente durante tres periodos de la señal de reloj*.
3. *Cambiar la cuenta para que avance según el código de Gray. Mantener esta cuenta durante tres periodos de la señal de reloj*.

El programa de test debe mostrar mensajes de error en el caso de que la señal de salida no tome el valor esperado. Dibuje el cronograma de las señales aplicadas al circuito y las salidas esperadas.

Solución a la Pregunta 4

El código VHDL del banco de pruebas del contador se muestra en Código VHDL 1.5–1.6. En la figura siguiente se muestra el cronograma obtenido al simular el banco de pruebas.



```
  ---------------------------------------
-- Banco de pruebas del contador binario
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity bp_contador is
end entity bp_contador;
architecture bp_contador of bp_contador is
   constant PERIODO    : time          := 20 ns; -- Reloj
   signal     s           : std_logic_vector(2 downto 0);-- Salidas UUT
   signal     clk         : std_logic   := '0';      -- Entradas UUT
   signal     M, reset   : std_logic;
   component contadorBinGray is
    port(   s       : out std_logic_vector(2 downto 0);
            M, clk, reset : in std_logic);
   end component contadorBinGray;
      -- Procedimiento para comprobar las salidas
    procedure comprueba_salidas
       ( esperado_s              :          std_logic_vector(2 downto 0);
         actual_s                 :          std_logic_vector(2 downto 0);
         error_count              : inout integer) is
    begin
         -- Comprueba q
         if ( esperado_s /= actual_s ) then
             report "ERROR: Estado esperado ("      &
                     std_logic'image(esperado_s(2))    &
                     std_logic'image(esperado_s(1))    &
                     std_logic'image(esperado_s(0))    &
                     "), estado actual ("              &
                     std_logic'image(actual_s(2))      &
                     std_logic'image(actual_s(1))      &
                     std_logic'image(actual_s(0))      &
                     "), instante: "                   &
                     time'image(now);
             error_count := error_count + 1;
         end if;
     end procedure comprueba_salidas;
begin
   -- Instanciar y conectar UUT
   uut : component contadorBinGray port map
                    (s, M, clk, reset);
   reset <=   '1',
              '0' after 15 ns;
   clk <= not Clk after (PERIODO/2);
   gen_vec_test : process is
     variable temp   : unsigned (2 downto 0);
     variable error_count : integer := 0; -- Núm. errores
   begin
```

Código VHDL 1.5: Diseño del banco de pruebas del contador (1/2).

```
        report "Comienza la simulación";
        -- Vectores de test y comprobación del resultado
        wait for PERIODO;
        M <='0';
      for i in 0 to 2 loop
        temp := TO_UNSIGNED(i,3);
        comprueba_salidas( std_logic_vector(temp), s, error_count);
        wait for PERIODO;
      end loop;
      M <= '1';
       comprueba_salidas( "011", s, error_count);
      wait for PERIODO;     -- 1
      comprueba_salidas( "010", s, error_count);
       wait for PERIODO;
      comprueba_salidas( "110", s, error_count);
        wait for PERIODO;
      comprueba_salidas( "111", s, error_count);
-- Informe final
      if (error_count = 0) then
          report "Simulación finalizada sin errores";
      else
          report   "ERROR: Hay "               &
                    integer'image(error_count ) &
                    " errores.";
      end if;
      wait;        -- Final del bloque process
   end process gen_vec_test;
end architecture bp_contador;
-----------------------------------
```

Código VHDL 1.6: Diseño del banco de pruebas del contador (2/2).