INGENIERÍA DE COMPUTADORES III

Solución al Ejercicio de Autocomprobación 3

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---------------------
with sel select
 tmp <= ‘1’ when ‘‘01’’|‘‘11’’,
        ‘0’ when others;
 y <= ‘Z’ when sel = ‘‘00’’ else tmp;
```

```
---- Fragmento 2--------
if boolean_expr_1 then
  if boolean_expr_2 then
     a <= b;
  else
     a<= c;
  end if;
else
  a<= d;
end if;
```

Solución a la Pregunta 1

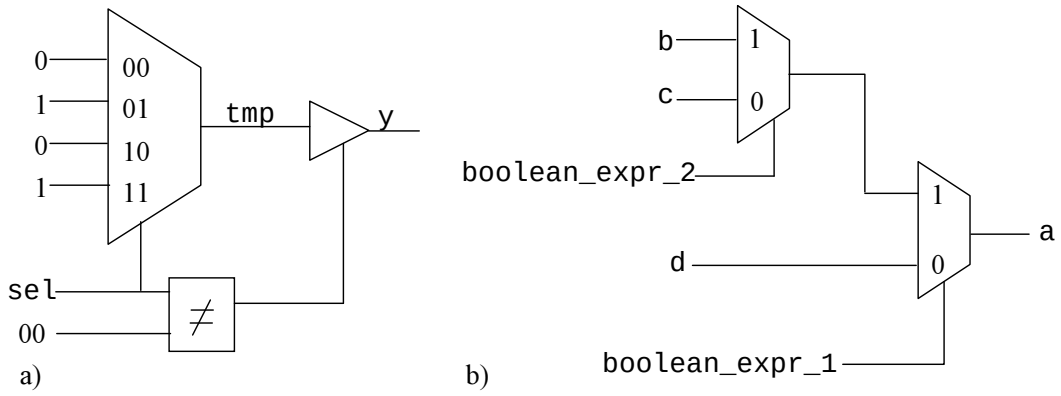


Figura 1.1: Diagramas conceptuales de: a) Fragmento 1; y b) Fragmento 2.

PREGUNTA 2 (2 puntos)

Escriba en VHDL la **entity** y la **architecture** que describe:

2.a) (0.25 puntos) El comportamiento de una puerta NOT de 1 entrada.

2.b) (0.25 puntos) El comportamiento de una puerta AND de 4 entradas.

2.c) (1.5 puntos) La estructura de un circuito combinacional decodificador 3 a 8 con entrada enable. La **architecture** debe describir la estructura del circuito combinacional, instanciando y conectando adecuadamente las puertas lógicas cuyo diseño ha realizado al contestar los dos apartados anteriores.

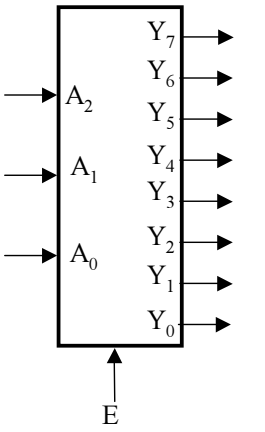
Solución a la Pregunta 2

Los diseños de la puerta NOT de 1 entrada y la puerta AND de 4 entradas, pedidos en los apartados 2.a y 2.b, se muestran en el Código VHDL 1.1.

```
-------------------------------------
-- NOT de 1 entrada
library IEEE; use IEEE.std_logic_1164.all;

entity not1 is
   port ( y        : out std_logic;
          x        : in    std_logic );
end entity not1;

architecture not1 of not1 is
begin
   y <= not x;
end architecture not1;
-------------------------------------
--AND de 4 entradas
library IEEE; use IEEE.std_logic_1164.all;

entity and4 is
   port ( y                    : out std_logic;
          x0, x1, x2, x3   : in    std_logic );
end entity and4;

architecture and4 of and4 is
begin
   y <= x0 and x1 and x2 and x3;
end architecture and4;
-------------------------------------
```

Código VHDL 1.1: Diseño de puerta NOT de 1 entrada y de puerta AND de 4 entradas.

Nombrando las entradas y salidas del circuito tal como se muestra en la Figura 1.2a, la tabla de la verdad del circuito decodificador 3 a 8 con entrada enable es la mostrada en la Figura 1.2b.



b)

E	A_2	A_1	A_0	Y_7	Y_6	Y_5	Y_4	Y_3	Y_2	Y_1	Y_0
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1	0	0
1	0	1	1	0	0	0	0	1	0	0	0
1	1	0	0	0	0	0	1	0	0	0	0
1	1	0	1	0	0	1	0	0	0	0	0
1	1	1	0	0	1	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0

Figura 1.2: Decodificador 3 a 8 con entrada enable: a) interfaz; y b) tabla de verdad.

En la página siguiente se muestra la Figura 1.3, en la cual se representa la estructura del circuito, compuesto de la conexión de puertas NOT de una entrada y puertas AND de cuatro entradas.

En la página siguiente a la figura, se encuentra la descripción estructural del circuito: el Código VHDL 1.2.

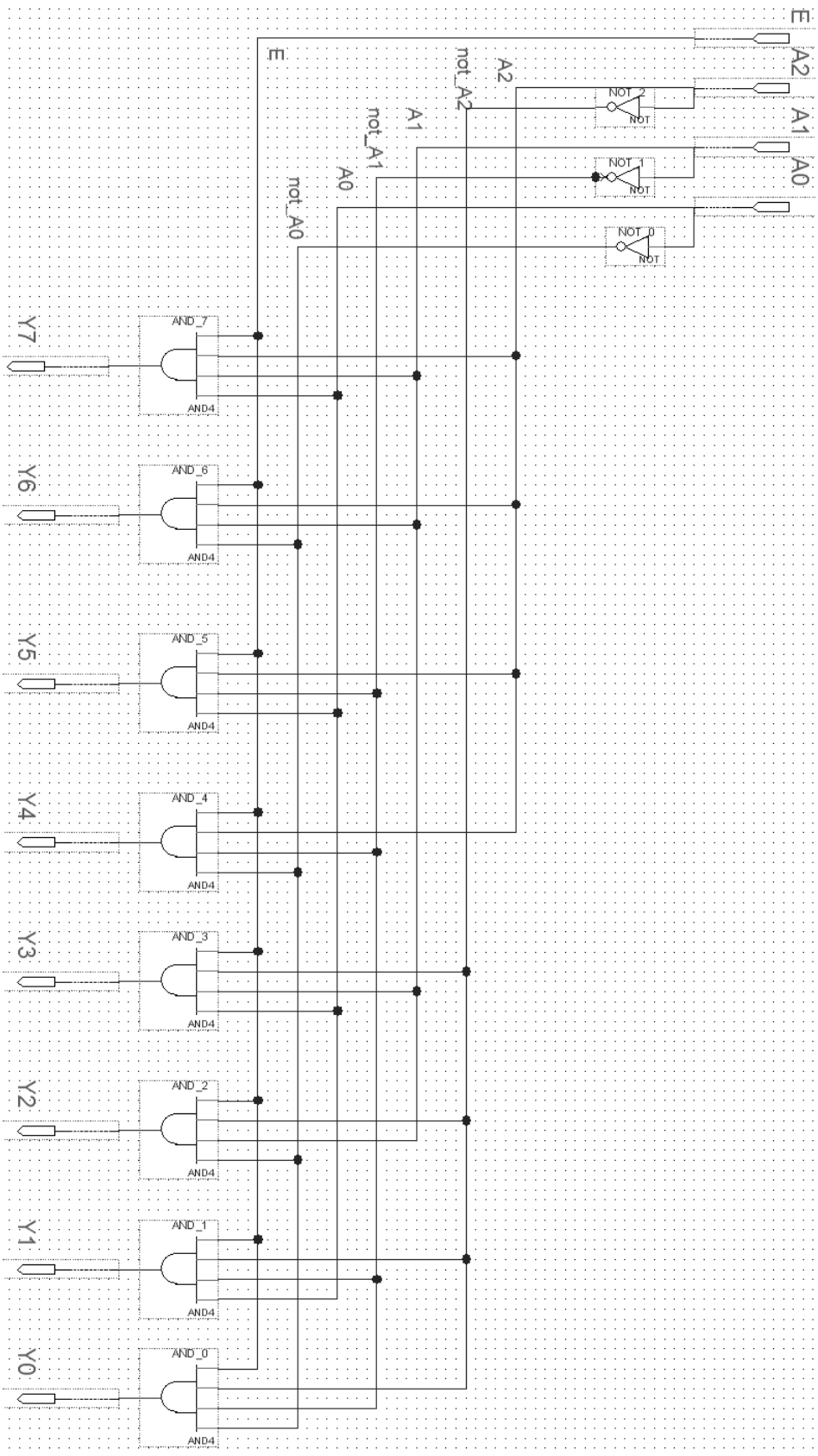


Figura 1.3: Diagrama de conexiones del circuito decodificador 3 a 8 con entrada enable.

```
----------------------------------
-- Decodificador 3 a 8 bits con entrada enable
library IEEE;
use IEEE.std_logic_1164.all;

entity decodificador_3_a_8 is port
   ( Y7, Y6, Y5, Y4, Y3, Y2, Y1, Y0 : out std_logic;
     E, A0, A1, A2                  : in  std_logic);
end entity decodificador_3_a_8;

architecture decodificador_3_a_8 of decodificador_3_a_8 is
   signal not_A2, not_A1, not_A0 :  std_logic;
   -- Declaración de las clases de los componentes
   component not1 is port
      ( y          : out std_logic;
        x          : in  std_logic );
   end component not1;
   component and4 is port
      ( y              : out std_logic;
        x0, x1, x2, x3 : in  std_logic );
   end component and4;
begin
   -- Instanciación y conexión de los componentes
   NOT_0 : component not1 port map (not_A0, A0);
   NOT_1 : component not1 port map (not_A1, A1);
   NOT_2 : component not1 port map (not_A2, A2);
   AND_7 : component and4 port map (Y7, E, A2, A1, A0);
   AND_6 : component and4 port map (Y6, E, A2, A1, not_A0);
   AND_5 : component and4 port map (Y5, E, A2, not_A1, A0);
   AND_4 : component and4 port map (Y4, E, A2, not_A1, not_A0);
   AND_3 : component and4 port map (Y3, E, not_A2, A1, A0);
   AND_2 : component and4 port map (Y2, E, not_A2, A1, not_A0);
   AND_1 : component and4 port map (Y1, E, not_A2, not_A1, A0);
   AND_0 : component and4 port map (Y0, E, not_A2, not_A1, not_A0);
end architecture decodificador_3_a_8;
----------------------------------
```

Código VHDL 1.2: Diseño del decodificador 3 a 8 con entrada enable, realizado mediante la descripción estructural del circuito al nivel de puertas lógicas.

PREGUNTA 3 (3 puntos)

Programe en VHDL el banco de pruebas del circuito combinacional que ha diseñado al contestar la Pregunta 2c. Explique detalladamente cómo el programa de test comprueba exhaustivamente el valor de la UUT para todos los posibles valores de la entrada. 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

Existen varias formas de realizar el diseño del banco de pruebas del decodificador 3 a 8 con entrada enable. A continuación, se muestra una de ellas.

```
--------------------------------------
-- Banco de pruebas
-- decodificador_3_a_8 con entrada enable
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity bp_decodificador_3_a_8 is
end entity bp_decodificador_3_a_8;

architecture bp_decodificador_3_a_8 of bp_decodificador_3_a_8 is
   signal Y                    : std_logic_vector( 7 downto 0);   -- Conectar salidas UUT
   signal E, A0, A1, A2    : std_logic;    -- Conectar entradas UUT

   component decodificador_3_a_8 is
       port (    Y7, Y6, Y5, Y4, Y3, Y2, Y1, Y0  : out std_logic;
                    E, A0, A1, A2                              : in    std_logic);
   end component decodificador_3_a_8;
```

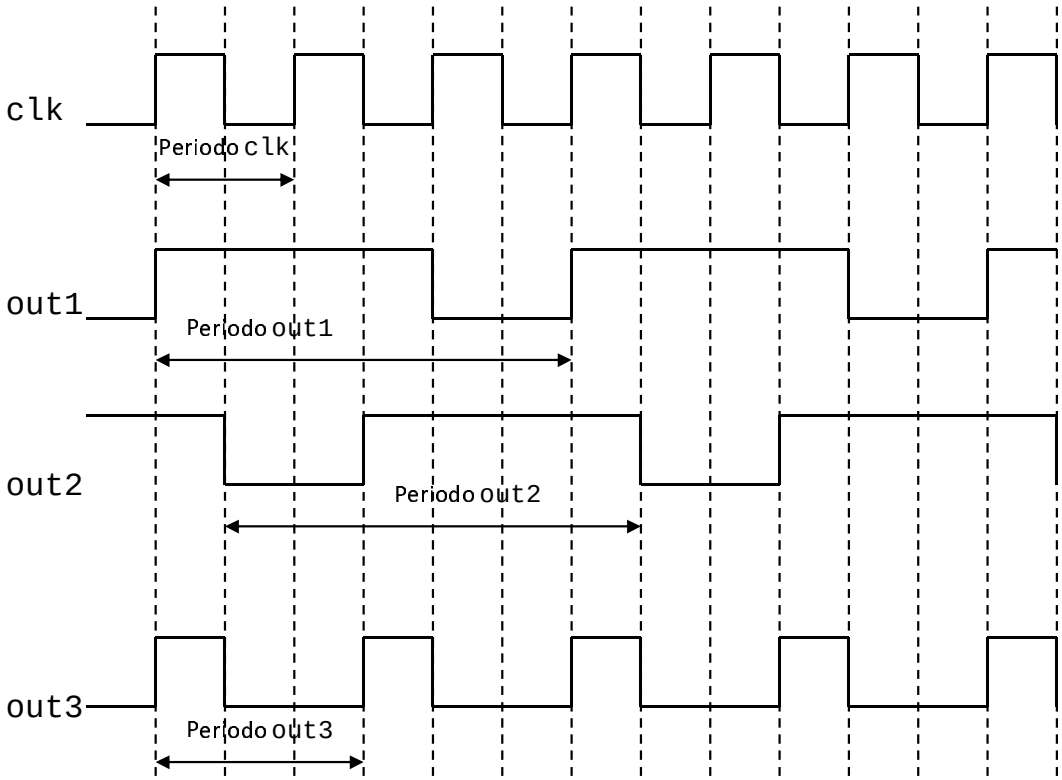
Código VHDL 1.3: Fragmento inicial del banco de pruebas.

```
begin
   -- Instanciar y conectar UUT
   uut : component decodificador_3_a_8
         port map( Y(7), Y(6), Y(5), Y(4), Y(3), Y(2), Y(1), Y(0), E, A0, A1, A2 );
   gen_vec_test : process
      variable test_in      : unsigned (3 downto 0); -- Vector de test
      variable esperado_Y     : std_logic_vector(7 downto 0);
      variable error_count          : integer     := 0;
   begin
      test_in := B"0000";
      report "Comienza la simulación";
      for count in 0 to 15 loop
         E <= test_in(3);
         A2 <= test_in(2);
         A1 <= test_in(1);
         A0 <= test_in(0);
         if (count<8) then    esperado_Y   := "00000000";
         elsif (count=8) then   esperado_Y := "00000001";
         elsif (count=9) then   esperado_Y := "00000010";
         elsif (count=10) then esperado_Y := "00000100";
         elsif (count=11) then esperado_Y := "00001000";
         elsif (count=12) then esperado_Y := "00010000";
         elsif (count=13) then esperado_Y := "00100000";
         elsif (count=14) then esperado_Y := "01000000";
         else   esperado_Y := "10000000";
         end if;

         wait for 10 ns;
         test_in := test_in + 1;
         if ( esperado_Y /= Y ) then
              report "ERROR en la salida valida. Valor esperado: "  &
                     std_logic'image(esperado_Y(7))                 &
                     std_logic'image(esperado_Y(6))                 &
                     std_logic'image(esperado_Y(5))                 &
                     std_logic'image(esperado_Y(4))                 &
                     std_logic'image(esperado_Y(3))                 &
                     std_logic'image(esperado_Y(2))                 &
                     std_logic'image(esperado_Y(1))                 &
                     std_logic'image(esperado_Y(0))                 &
                     ", valor actual: "                                 &
                     std_logic'image(Y(7))                          &
                     std_logic'image(Y(6))                          &
                     std_logic'image(Y(5))                          &
                     std_logic'image(Y(4))                          &
                     std_logic'image(Y(3))                          &
                     std_logic'image(Y(2))                          &
                     std_logic'image(Y(1))                          &
                     std_logic'image(Y(0))                          &
                     " en el instante: "                                &
                     time'image(now);
              error_count := error_count + 1;
         end if;

      end loop;
      report   "ERROR: Hay "              &
                integer'image(error_count) &
                " errores.";
      wait;
   end process gen_vec_test;
end architecture bp_decodificador_3_a_8;
-------------------------------------
```

Código VHDL 1.4: Fragmento final del banco de pruebas.

PREGUNTA 4 (3 puntos)

Escriba en VHDL la **architecture** que modela el comportamiento de un circuito digital que, a partir de una señal del reloj `clk`, genera las tres señales mostradas a continuación.



En el diseño del circuito, implemente dos máquinas de estado, una que opere exclusivamente en la transición positiva de la señal `clk` y otra que opere exclusivamente en la transición negativa. Estas máquinas de estado generan las señales `out1` y `out2`. Obtenga la señal `out3` mediante una AND lógica de las señales `out1` y `out2`.

A continuación, se muestra el código VHDL de la **entity** del circuito.

```
entity gen is port(
   out1, out2, out3 :  out std_logic;
   clk              :  in  std_logic );
end entity gen;
```

Solución a la Pregunta 4

El Código VHDL 1.5 y 1.6, mostrado en esta página y en la siguiente, es un posible diseño del circuito especificado en la Pregunta 4.

```
--------------------------------------
-- Circuito generador de señales
-- Pregunta 4
library IEEE; use IEEE.std_logic_1164.all;
entity gen is port(
   out1, out2, out3 : out std_logic;
   clk              : in std_logic );
end entity gen;
architecture gen of gen is
   type state is (one, two, three);
   signal internal_state1: state;
   signal internal_state2: state;
   signal temp_out1, temp_out2: std_logic;
begin
------Maquina de estados 1
proximo_estado1:process(clk)
begin
   if (rising_edge(clk)) then
      case internal_state1 is
         when one =>
            internal_state1 <= two;
         when two =>
            internal_state1 <= three;
         when three =>
            internal_state1 <= one;
      end case;
   end if;
end process proximo_estado1;
salida1: process (internal_state1)
begin
   case internal_state1 is
      when one =>
         temp_out1<= '1';
      when two =>
         temp_out1 <= '0';
      when three =>
         temp_out1 <= '1';
    end case;
end process salida1;
```

Código VHDL 1.5: Fragmento inicial del diseño del circuito especificado en la Pregunta 4.

```
--------Maquina de estados 2
proximo_estado2:process (clk)
begin
   if (falling_edge(clk)) then
      case internal_state2 is
         when one =>
            internal_state2 <= two;
         when two =>
            internal_state2 <= three;
         when three =>
            internal_state2 <= one;
      end case;
   end if;
end process proximo_estado2;
salida2:process (internal_state2)
begin
   case internal_state2 is
      when one =>
         temp_out2 <= '1';
      when two =>
         temp_out2 <= '1';
      when three =>
         temp_out2 <= '0';
   end case;
end process salida2;
out1 <= temp_out1;
out2 <= temp_out2;
out3 <= temp_out1 and temp_out2;
end gen;
```

Código VHDL 1.6: Fragmento final del diseño del circuito especificado en la Pregunta 4.