INGENIERÍA DE COMPUTADORES 3

Solución al examen de Junio 2018, Segunda 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 50 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono1 is
end entity crono1;
architecture crono1 of crono1 is
  signal x1, x2, x3, x4, x5 : std_logic;
begin
  x1 <= '0', '1' after 10 ns,
        '0' after 15 ns,'1' after 25 ns,
        '0' after 30 ns;
  Proc1: process
  begin
    x2 <= '1';
    wait for 5 ns;
    x2 <= '0';
    wait for 10 ns;
    x2 <= '1';
    wait for 5 ns;
    x2 <= '0';
  end process;
   x3 <= x1 after 5 ns;
  Proc2: process
     variable v1  : std_logic;
  begin
    for i in 0 to 3 loop
       v1 := x1 xor x2;
       x4 <= v1;
       x5 <= x4;
       wait for 5 ns;
     end loop;
     wait;
  end process;
end architecture crono1;
```

Solución a la Pregunta 1

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

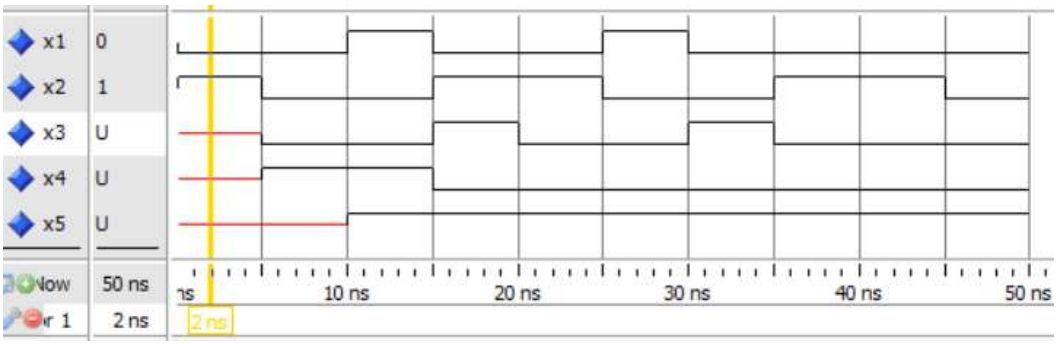
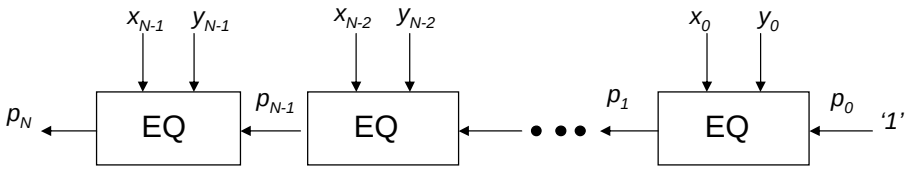


Figura 1.1: Cronograma de evolución de las señales.

PREGUNTA 2 (3 puntos)

Diseñe un circuito que compare si dos números de N bits tienen igual valor. Construya el circuito de manera iterativa, tal como se muestra en la siguiente figura.



Diseñe inicialmente un comparador de 1 bit, que tenga las tres entradas siguientes: los dos operandos de un bit (x_i e y_i) y el resultado de la comparación de la etapa anterior (p_i). p_i es un bit cuyo valor es '1' sólo si todas las parejas de bits comparadas hasta esa etapa son iguales. La salida del comparador de 1 bit, p_{i+1}, es el resultado de la comparación llevada a cabo en esa etapa, teniendo en cuenta el valor de las tres entradas. Finalmente, encadene N comparadores de 1 bit para

implementar el comparador de N bits, usando para ello la sentencia **generate**. N ha de ser una constante de tipo **generic**.

Solución a la Pregunta 2

El comparador de dos números de 1 bit satisface la siguiente función lógica: $p_{i+1} = (x_i \text{ xnor } y_i)$ and p_i. El diseño de este circuito, así como el diseño del circuito comparador de N bits, se muestra en Código VHDL 1.1–1.2.

```
  ------------------------------------
-- Comparador p1 = ((X=Y)and(p0=1))
--       p1 = (x xnor y) and p0

library IEEE;
use IEEE.std_logic_1164.all;

entity comparadorEQ1bit is port
   ( p1          : out std_logic;
     p0, x, y : in    std_logic);
end entity comparadorEQ1bit;

architecture comparadorEQ1bit of comparadorEQ1bit is
begin
    p1 <= (x xnor y) and p0;
end architecture comparadorEQ1bit;
```

Código VHDL 1.1: Diseño del circuito comparador de igualdad de dos números de 1 bit.

```
-------------------------------------------
-- Comparador igualdad de 2 números de N bits
library IEEE;
use IEEE.std_logic_1164.all;

entity comparadorEQNbits is
   generic( N: integer :=4);
   port     ( s       : out std_logic;
              x, y    : in std_logic_vector(N-1 downto 0));
end entity comparadorEQNbits;

architecture comparadorEQNbits of comparadorEQNbits is
    signal p: std_logic_vector(N downto 0);

    component comparadorEQ1bit is
       port ( p1          : out std_logic;
              p0, x, y  : in    std_logic);
    end component comparadorEQ1bit;
begin
     p(0) <= '1';

--Instanciación y conexión de componentes usando GENERATE
     conexion: for k in N-1 downto 0 generate --GENERATE iterativo
         comparadorNbits: comparadorEQ1bit
         port map (p(k+1), p(k), x(k), y(k));
     end generate conexion;

     s <= p(N);

end architecture comparadorEQNbits;
-----------------------------------
```

Código VHDL 1.2: Diseño del circuito comparador de igualdad de dos números de N bits.

PREGUNTA 3 (2 puntos)

Programe el banco de pruebas del circuito combinacional que ha diseñado en la Pregunta 2 para un valor de N igual a 6. 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 1.3–1.4.

```
---------------------------------------
-- Banco de pruebas comparador de igualdad de N bits
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity bp_comparadorEQNbits is
end entity bp_comparadorEQNbits;

architecture bp_comparadorEQNbits of bp_comparadorEQNbits is
   signal s       : std_logic;   -- Conectar salidas UUT
   signal x, y  : std_logic_vector(5 downto 0);   -- Conectar entradas UUT

   component comparadorEQNbits is
      generic( N: integer :=6);
      port     ( s       : out std_logic;
                 x, y  : in std_logic_vector(N-1 downto 0));
   end component comparadorEQNbits;
```

Código VHDL 1.3: Diseño del banco de pruebas del comparador (1/2).

```
  begin
    -- Instanciar y conectar UUT
    uut : component comparadorEQNbits port map
          ( s => s, x => x, y => y );
    gen_vec_test : process
      variable num_errores   : integer := 0;              -- Numero de errores
    begin
      for op_X in 0 to 2**6-1 loop
        for op_Y in 0 to 2**6-1 loop
            x     <= std_logic_vector(to_unsigned(op_x,6));
            y     <= std_logic_vector(to_unsigned(op_y,6));
            wait for 10 ns;
            -- Comprueba resultado
            if (x=y and s='0') or (x/=y and s='1') then
              report "Error. Valores " &
                       integer'image(op_x) &
                       "," &
                       integer'image(op_y);
              num_errores := num_errores + 1;
            end if;
        end loop;
      end loop;
      report "Test completo. Hay "      &
             integer'image(num_errores) &
             " errores.";
      wait; --Final simulación
    end process gen_vec_test;
end architecture bp_comparadorEQNbits;
--------------------------------
```

Código VHDL 1.4: Continuación del banco de pruebas del comparador (2/2).

PREGUNTA 4 (3 puntos)

Diseñe usando VHDL el siguiente circuito secuencial síncrono. El circuito opera en el flanco de subida de la señal de reloj (`clk`) y tiene como entradas la señal de reloj `clk`, una señal `reset` asíncrona activa a nivel alto y una señal de un bit `X`. El circuito tiene una señal de salida de un bit `Z`.

La señal de salida `Z` tiene valor '1' durante un ciclo de reloj cuando la secuencia de los tres últimos valores de la entrada sea "111", "110' ó "000", y tiene valor '0' en cualquier otro caso. La señal `reset` inicializa el circuito poniéndolo en un estado en que los últimos tres bits recibidos por la señal de entrada `X` son "001".

Diseñe el circuito como una máquina de Moore. Escriba en una tabla, cuya estructura se muestra a continuación, todos los estados de la máquina, indicando para cada estado cuál es el siguiente estado cuando la señal de entrada vale `X = '0'` y cuando la señal de entrada vale `X = '1'`. Especifique además el valor de la señal de salida `Z` correspondiente a cada estado.

Tabla de estados.

Estado	Estado siguiente si `X = '0'`	Estado siguiente si `X = '1'`	`Z`

Solución a la Pregunta 4

A continuación se muestra la tabla pedida en el enunciado de la Pregunta 4.

Tabla de estados.

Estado	Estado siguiente si `X = '0'`	Estado siguiente si `X = '1'`	`Z`
000	000	001	1
001	010	011	0
010	100	101	0
011	110	111	0
100	000	001	0
101	010	011	0
110	100	101	1
111	110	111	1

El código VHDL correspondiente al detector se muestra en Código VHDL 1.5–1.6.

```
----------------------------------------------------------
---Detector de secuencias 111, 110 o 000
----------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

entity detector is
    port( Z      : out std_logic;
          X      : in std_logic;
          reset  : in std_logic;
          clk    : in std_logic);
end entity detector;

architecture detector of detector is
       signal internal_state: std_logic_vector(2 downto 0);
begin

--Cálculo salida
 salida: process (internal_state) is
 begin
    case internal_state is
         when "000" => Z <= '1';
  when "110" => Z <= '1';
  when "111" => Z <= '1';
         when others => Z <= '0';
      end case;
 end process salida;

 --Cálculo del próximo estado
 proximo_estado: process (clk, reset)
 begin
     if (reset = '1') then --reset asíncrono
          internal_state <= "001";
```

Código VHDL 1.5: Circuito detector.

```
        elsif (rising_edge(clk)) then
            case internal_state is
                when "000" =>     -- Estado actual: 000
                    if X = '1' then
                        internal_state <= "001";
                    end if;
                when "001" =>   --Estado actual: 001
                    if X = '1' then
                        internal_state <= "011";
                    else
                        internal_state <= "010";
                    end if;
                when "010" =>   --Estado actual: 010
                    if X = '1' then
                         internal_state <= "101";
                    else
                         internal_state <= "100";
                   end if;
                 when "011" =>   -- Estado actual: 011
                     if X = '1' then
                         internal_state <= "111";
                     else
                         internal_state <= "110";
                    end if;
                 when "100" => --Estado actual: 100
                     if X = '1' then
                          internal_state <= "001";
                     else
                          internal_state <= "000";
                     end if;
                 when "101" => --Estado actual: 101
                      if X= '1' then
                          internal_state <= "011";
                      else
                          internal_state <= "010";
                      end if;
                 when "110" => --Estado actual: 110
                       if X = '1' then
                           internal_state <= "101";
                       else
                           internal_state <= "100";
                       end if;
                 when others =>   -- Estado actual: 111
                       if X = '1' then
                           internal_state <= "111";
                       else
                           internal_state <= "110";
                       end if;
                end case;
            end if;
end process proximo_estado;

end architecture detector;
-----------------------------------------
```

Código VHDL 1.6: Circuito detector.