INGENIERÍA DE COMPUTADORES 3

Solución al examen de Junio 2013, 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`, `x5` entre los instantes 0 y 60 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity crono1 is
end entity crono1;
architecture crono1 of crono1 is
   signal x1, x2, x3, x4, x5 : std_logic;
begin
 process is
    variable temp  : unsigned (2 downto 0);
  begin
    for i in 0 to 3 loop
      temp := TO_UNSIGNED(i,3);
      x1 <= std_logic(temp(2));
      x2 <= std_logic(temp(1));
      x3 <= std_logic(temp(0));
      wait for 10 ns;
    end loop;
    wait;
  end process;
  x4 <= x3 after 5 ns;
  x5 <= x3 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.

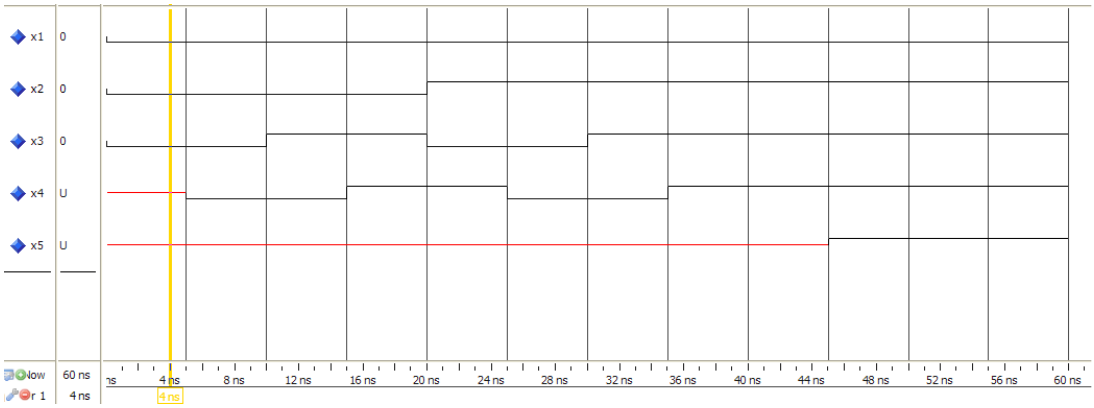

Figura 1.1: Cronograma de evolución de las señales.

PREGUNTA 2 (3 puntos)

Escriba en VHDL, de las cuatro formas que se detallan a continuación, la **architecture** que describe el comportamiento de un circuito combinacional multiplexor 8 a 1. La **entity** del circuito es:

```
entity mux is
   port ( y  : out std_logic;
          x  : in  std_logic_vector(7 downto 0);
          s  : in  std_logic_vector(2 downto 0) );
end entity mux;
```

2.a) (0.75 puntos) Empleando una sentencia concurrente condicional (**when - else**).

2.b) (0.75 puntos) Empleando una asignación concurrente de selección (**with - select**).

2.c) (0.75 puntos) Empleando un bloque **process** con una sentencia **if**.

2.d) (0.75 puntos) Empleando un bloque **process** con una sentencia **case**.

Solución a la Pregunta 2

Las cuatro **architecture** que describen el comportamiento de un circuito combinacional multiplexor 8 a 1 se muestran en Código VHDL 1.1–1.4.

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

architecture arch_cond of mux is
begin
    y <= x(0) when (s="000") else
         x(1) when (s="001") else
         x(2) when (s="010") else
         x(3) when (s="011") else
         x(4) when (s="100") else
         x(5) when (s="101") else
         x(6) when (s="110") else
         x(7);
end architecture arch_cond;
```

Código VHDL 1.1: Descripción del comportamiento del multiplexor empleando una sentencia concurrente condicional.

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

architecture arch_cond of mux is
begin
    with s select
        y <= x(0) when "000",
             x(1) when "001",
             x(2) when "010",
             x(3) when "011",
             x(4) when "100",
             x(5) when "101",
             x(6) when "110",
             x(7) when others;

end architecture arch_cond;
```

Código VHDL 1.2: Descripción del comportamiento del multiplexor empleando una sentencia concurrente de selección.

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

architecture arch_procIf of mux is
begin
    process ( x, s )
    begin
        if (s = "000") then
            y <= x(0);
        elsif (s = "001") then
            y <= x(1);
        elsif (s = "010") then
            y <= x(2);
        elsif (s = "011") then
            y <= x(3);
        elsif (s = "100") then
            y <= x(4);
        elsif (s = "101") then
            y <= x(5);
        elsif (s = "110") then
            y <= x(6);
        else
            y <= x(7);
        end if;
    end process;
end architecture arch_procIf;
```

Código VHDL 1.3: Descripción del comportamiento del multiplexor empleando un bloque **process** con una sentencia **if**.

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

architecture arch_procCase of mux is
begin
    process ( x, s )
    begin
        case s is
            when "000" =>
                y <= x(0);
            when "001" =>
                y <= x(1);
            when "010" =>
                y <= x(2);
            when "011" =>
                y <= x(3);
            when "100" =>
                y <= x(4);
            when "101" =>
                y <= x(5);
            when "110" =>
                y <= x(6);
            when others =>
                y <= x(7);
        end case;
    end process;
end architecture arch_procCase;
```

Código VHDL 1.4: Descripción del comportamiento del multiplexor empleando un bloque **process** con una sentencia **case**.

PREGUNTA 3 (3 puntos)

Escriba el código VHDL de la **architecture** que describe el comportamiento de un circuito contador binario de 3 bits cuya salida sigue cíclicamente la secuencia "000", "011", "110", 101" y "111", con señal de reset asíncrona activa a nivel alto. La **entity** del circuito se muestra a continuación.

```
entity contador is
  port (  count      : out std_logic_vector (2 downto 0);
          clk, reset : in  std_logic );
end contador;
```

Las entradas al circuito son la señal de reloj (`clk`) y la señal de reset asíncrono activo a nivel alto (`reset`). La salida del circuito contador es la señal de 3 bits `count`.

El funcionamiento del circuito debe ser el siguiente:

- *Reset asíncrono activo a nivel alto*. Cuando `reset` pasa a valer '1', el contador se pone al valor "000".
- *Cuenta síncrona*. Mientras `reset` vale '0', el contador pasa en cada flanco de subida de la señal de reloj por la secuencia "000", "011", "110", "101" y "111".

El diseño debe realizarse describiendo el comportamiento del circuito en términos de una máquina de Moore.

Solución a la Pregunta 3

El código VHDL del circuito contador se muestra en Código VHDL 1.5.

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

entity contador is
port ( count : out std_logic_vector (2 downto 0);
       clk, reset : in std_logic );
end contador;

architecture contador of contador is
signal state: std_logic_vector ( 2 downto 0) ;
signal internal_state : std_logic_vector (2 downto 0) ;
begin
 --Calculo del proximo estado
 proximo_estado: process(reset,clk) is
 begin
  if (reset = '1') then -- Reset asincrono
    internal_state <= "000";
  elsif rising_edge(clk) then -- Flanco subida del reloj
    case internal_state is
      when "000" =>
         internal_state <= "001";
      when "001" =>
         internal_state <= "010";
      when "010" =>
         internal_state <= "011";
      when "011" =>
         internal_state <= "100";
      when "100" =>
         internal_state <= "000";
      when others => -- Por completitud
         internal_state <= "XXX";
    end case;
  end if;
 end process proximo_estado;
 --Calculo salida
 salida: process (state)
 begin
    case state is
         when "000" => count <= "000";
         when "001" => count <= "011";
         when "010" => count <= "110";
         when "011" => count <= "101";
         when "100" => count <= "111";
         when others => count <= "XXX";
     end case;
 end process salida;
 state <= internal_state;
end contador ;
```

Código VHDL 1.5: Diseño del circuito contador.

PREGUNTA 4 (2 puntos)

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

1. *Reset*. Resetear el contador.
2. *Cuenta síncrona* . Incrementar el valor del contador hasta "111".
3. *Reset*. Resetear el contador.

El correcto funcionamiento del circuito debe comprobarse mediante inspección visual. No es necesario que el banco de prueba compruebe que las salidas de la UUT son las esperadas. Dibuje el cronograma de evolución que han de seguir las señales de entrada y salida de la UUT.

Solución a la Pregunta 4

El código VHDL correspondiente al banco de pruebas del contador se muestra en Código VHDL 1.6. El cronograma correspondiente a la simulación de este banco de pruebas se muestra en la Figura 1.2.

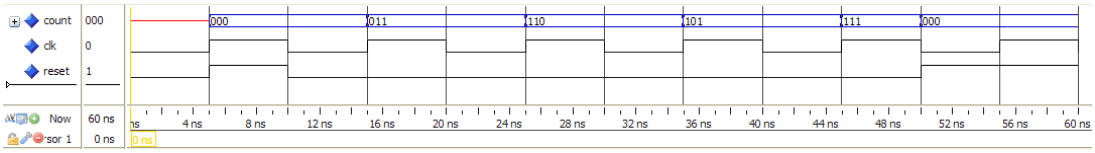

Figura 1.2: Cronograma resultado de la simulación del banco de pruebas del contador.

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

entity bp_contador is
   constant PERIODO : time := 10 ns; -- Periodo reloj
end entity bp_contador;

architecture bp_contador of bp_contador is
   signal count : std_logic_vector (2 downto 0);   -- Conectar salidas UUT
   signal clk: std_logic := '0';-- Conectar entrada UUT
   signal reset : std_logic := '0';   -- Conectar entradas UUT

   component contador is
   port ( count : out std_logic_vector (2 downto 0);
          clk, reset : in std_logic );
   end component contador;

begin
   -- Instanciar y conectar UUT
   uut : component contador port map
         ( count => count, clk => clk, reset => reset);
   clk <= not clk after PERIODO/2;
   reset <= '0',
               '1' after ( PERIODO/2 ),
               '0' after ( PERIODO),
               '1' after (5*PERIODO);
   end architecture bp_contador;
-------------------------------------
```

Código VHDL 1.6: Banco de pruebas del contador.