INGENIERÍA DE COMPUTADORES 3

Solución al examen de Junio 2015, 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` entre los instantes 0 y 80 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono is
end entity crono;
architecture crono of crono is
  signal x1, x2, x3, x4 : std_logic;
begin
  x2 <= '0', '1' after 25 ns,
        '0' after 35 ns,'1' after 50 ns;
  Proc1: process
  begin
    x1 <= '0';
    wait for 10 ns;
    x1 <= '1';
    wait for 15 ns;
    x1 <= '0';
  end process Proc1;
  Proc2: process
  begin
    x1 <= '0';
    wait for 15 ns;
    x1 <= '1';
    wait for 10 ns;
    x1 <= '0';
    wait;
  end process Proc2;
  Proc3: process (x2)
  begin
    x3 <= x2;
    x4 <= x3;
  end process Proc3;
end architecture crono;
```

Solución a la Pregunta 1

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

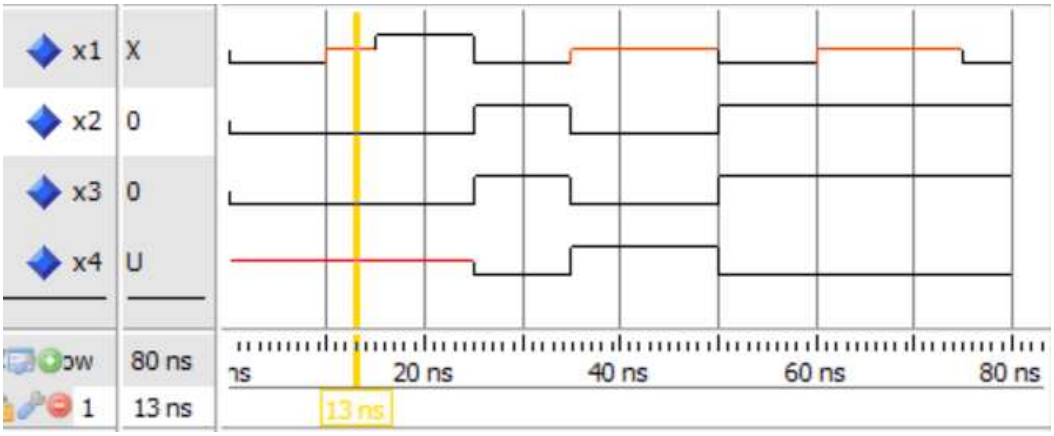

Figura 1.1: Cronograma de evolución de las señales.

PREGUNTA 2 (2.5 puntos)

Diseñe un circuito combinacional que tenga una señal de entrada de 8 bits llamada `data` y una señal de salida de 4 bits llamada `zeros`. La señal `zeros` se interpreta como un número binario sin signo. Esta señal ha de tener como valor el número de ceros existentes en `data` antes de encontrar el primer '1', empezando a contar por el extremo izquierdo (bit más significativo) de `data` . Por ejemplo, el valor de la señal de entrada `data` "00110110" da como resultado un valor de la señal `zeros` de "0010" (valor 2 en notación decimal).

El diseño ha de tener la siguiente **entity**:

```
entity numceros is
   port( zeros   : out std_logic_vector (3 downto 0);
         data    : in  std_logic_vector (7 downto 0) );
end entity numceros;
```

Solución a la Pregunta 2

La **architecture** que describe el comportamiento del circuito combinacional se muestra en Código VHDL 1.1.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity numceros is
    port( zeros    : out std_logic_vector ( 3 downto 0);
          data     : in std_logic_vector(7 downto 0));
end entity numceros;
architecture numceros of numceros is
begin
    process (data)
       variable count: unsigned (3 downto 0);
    begin
       for i in data'range loop
          case data(i) is
             when '0' => count := count + 1;
             when others => exit;
          end case;
       end loop;
       zeros <= std_logic_vector(count);
    end process;
end architecture numceros;
```

Código VHDL 1.1: Descripción del comportamiento del circuito combinacional solución del Ejercicio 2.

PREGUNTA 3 (3.5 puntos)

Diseñe usando VHDL un contador binario ascendente de 8 bits que opere en el flanco de subida de la señal de reloj, con carga de datos síncrona y reset asíncrono. El contador tiene la siguiente **entity**.

```
entity contador is
   port (  count : out std_logic_vector(7 downto 0);
           data  : in  std_logic_vector(7 downto 0);
           clk, reset, load  : in  std_logic );
end contador;
```

Las entradas al circuito son la señal de reloj (`clk`), la señal de reset (`reset`) asíncrona y activa a nivel bajo, la señal de carga (`load`) y la señal `data`. El circuito tiene una única señal de salida llamada `count`.

Cuando la señal `reset` vale '0', todos los bits de la señal `count` se ponen a 0. En caso contrario, en cada flanco de subida de la señal de reloj, si la señal `load` vale '0' se incrementa en '1' el valor de la señal de salida y si `load` vale '1' se asigna a la señal de salida el valor de la señal `data`.

Solución a la Pregunta 3

El código VHDL del circuito contador se muestra en Código VHDL 1.2.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
entity contador is
    port (   count : out std_logic_vector( 7  downto 0);
             data: in std_logic_vector( 7  downto 0);
             clk, reset, load   : in    std_logic );
end contador;
architecture contador of contador is
  signal count_i: unsigned (7  downto 0);
begin
   process(clk, reset)
   begin
      if (reset = '0') then
         count_i <= (others => '0');
      elsif rising_edge(clk) then
         if (load = '1') then
            count_i <= unsigned(data);
         else
            count_i <= count_i + '1';
         end if;
      end if;
   end process;
   count <= std_logic_vector(count_i);
end contador;
```

Código VHDL 1.2: Diseño del circuito 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 '1'. El programa de test debe realizar consecutivamente las acciones siguientes:

1. *Reset*. La señal de reset ha de tener el valor '0' durante 15 ns.
2. *Empezar la cuenta*. Comprobar que el circuito pasa consecutivamente por los valores "00000000" a "00000011".

3. *Cargar el valor "11111100"*. La señal `load` ha de valer '1' . La señal `data` ha de tener el valor "11111100".

4. *Seguir la cuenta*. Comprobar que el circuito pasa por los valores "11111100", "11111101", "11111110", "11111111" y "00000000".

El banco de pruebas debe comprobar que los valores de las señales de salida 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 4

El código VHDL correspondiente al banco de pruebas del regulador se muestra en Código VHDL 1.3–1.4.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity bp_contador is
    constant PERIODO     : time      := 20 ns; -- PERIODO clk

end entity bp_contador;

architecture bp_contador of bp_contador is
    signal count   : std_logic_vector(7 downto 0); --Salida UUT
    signal data    : std_logic_vector(7 downto 0); --Entradas UUT
    signal clk : std_logic := '0';
    signal reset, load: std_logic;

    component contador is
    port (   count : out std_logic_vector( 7 downto 0);
             data: in std_logic_vector( 7 downto 0);
             clk, reset, load   : in   std_logic );
    end component contador;

begin
 UUT : component contador port map
        (count, data, clk, reset, load);
 clk <= not clk after (PERIODO/2);
 reset <= '0' ,'1' after 15 ns;
 vec_test : process is
     variable temp   : std_logic_vector (7 downto 0);
     variable error_count : integer := 0;
  begin
```

Código VHDL 1.3: Banco de pruebas del contador.

```
      report "Comienza la simulación";
      -- Generar todos los posibles valores de entrada
      wait for PERIODO/2;
      wait until falling_edge(clk);
      for i in 0 to 3 loop
         temp := std_logic_vector(to_unsigned(i,8));
         if(temp /= count) then
             report "ERROR";
             error_count := error_count +1;
         end if;
         wait for PERIODO;
      end loop;
      load <= '1';
      data <= "11111100";
      wait for PERIODO;
      load <= '0';
      for i in 0 to 3 loop
         temp := std_logic_vector(to_unsigned(i,8)+unsigned(
                  data));
         if(temp /= count) then
             report "ERROR";
             error_count := error_count +1;
         end if;
         wait for PERIODO;
      end loop;
     temp := "00000000";
     if(temp /= count) then
             report "ERROR";
             error_count := error_count +1;
         end if;
        -- Informe mostrando el resultado del test
      report "Finaliza la simulación: " & integer'image(error_count)
                                                        & "errores";
      wait;    -- Termina la simulación
   end process vec_test;
end architecture bp_contador;
```

Código VHDL 1.4: Continuación del banco de pruebas del contador.