INGENIERÍA DE COMPUTADORES 3

Solución al examen de Junio 2016, 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 80 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity cronoW is
end entity cronoW;
architecture cronoW of cronoW is
  signal x1, x2, x3, x4, x5 : std_logic;
begin
  x1 <= '1', '0' after 5 ns,
        '1' after 15 ns,'0' after 20 ns,
        '1' after 40 ns;
  Proc1: process
  begin
    x2 <= '1';
    wait for 5 ns;
    x2 <= '0';
    wait for 15 ns;
    x2 <= '1';
    wait for 10 ns;
    x2 <= '0';
  end process;
   x3 <= x1 after 10 ns;
  Proc2: process
     variable valor  : std_logic;
  begin
    for i in 0 to 3 loop
       valor := x1 or x2;
       x4 <= valor;
       x5 <= x4;
       wait for 10 ns;
     end loop;
     wait;
  end process;
end architecture cronoW;
```

Solución a la Pregunta 1

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

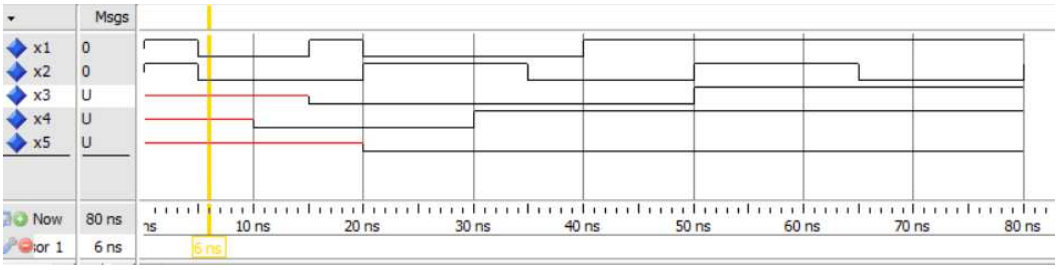
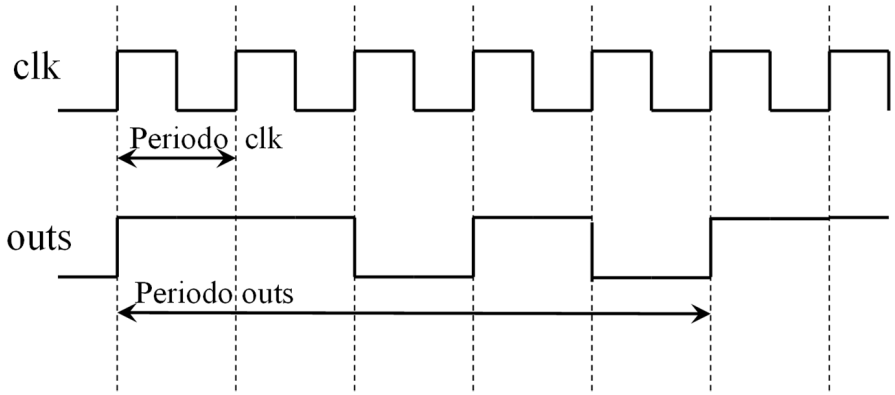


Figura 1.1: Cronograma de evolución de las señales.

PREGUNTA 2 (2.5 puntos)

Diseñe un generador de señales que obtenga la forma de onda mostrada en la parte inferior de la figura siguiente (señal `outs`) a partir de una señal de reloj `clk`. Describa su comportamiento como una máquina de estado finito de Moore.



Solución a la Pregunta 2

El diseño del generador de señales se muestra en Código VHDL 1.1–1.2.

```
------------------------------------------------------
-- Paquete con la definición de las constantes globales
library IEEE;
use IEEE.std_logic_1164.all;
package STATE_CONST is
   constant STATE_BITS: integer := 3;      -- Bits codifican estado
   constant S0: std_logic_vector(2 downto 0) := "000"; -- Estados
   constant S1: std_logic_vector(2 downto 0) := "001";
   constant S2: std_logic_vector(2 downto 0) := "010";
   constant S3: std_logic_vector(2 downto 0) := "011";
   constant S4: std_logic_vector(2 downto 0) := "100";
end package;
------------------------------------------------------
```

Código VHDL 1.1: Paquete de definición de las constantes globales del enedador de señales.

```
-------------------------------------------------------
---Generador de ondas implementado como
---máquina de estados
library IEEE;
use IEEE.std_logic_1164.all;
use work.STATE_CONST.all;

entity generador is
   port(outs : out std_logic;
        clk  : in std_logic);
end entity generador;

architecture fsm of generador is
       signal state : std_logic_vector(STATE_BITS-1 downto 0);
 begin
 --Cálculo del próximo estado
 proximo_estado: process (clk)
 begin
      if (rising_edge(clk)) then
           case state is
                when S0 => state <= S1;
                when S1 => state <= S2;
                when S2 => state <= S3;
                when S3 => state <= S4;
                when S4 => state <= S0;
                when others => state <= S0;
             end case;
        end if;
end process proximo_estado;
--Cálculo de la salida
 salida: process (state)
 begin
    case state is
      when S0 =>  outs <= '1';
      when S1 =>  outs <= '1';
      when S2 =>  outs <= '0';
      when S3 =>  outs <= '1';
      when S4 =>  outs <= '0';
      when others =>  outs <= '0';
    end case;
end process salida;

end architecture fsm;
-----------------------------------------
```

Código VHDL 1.2: Generador de señales.

PREGUNTA 3 (3.5 puntos)

Diseñe usando VHDL un desplazador de nBits bits que opere en el flanco de subida de la señal de reloj, con carga de datos síncrona, reset síncrono activo a nivel alto y con una señal para seleccionar desplazamiento a la izquierda o a la derecha. El desplazador tiene la siguiente **entity**.

```
entity desplazador is
   generic (nBits: integer:=8);
   port (  dout : out std_logic_vector( nBits-1 downto 0);
           data: in std_logic_vector( nBits-1 downto 0);
           clk, load, izqda, reset  : in  std_logic );
end desplazador;
```

Las entradas al circuito son la señal de reloj (`clk`), la señal de carga (`load`), la señal de reset (`reset`), la señal que selecciona el tipo de desplazamiento (`izqda`) y la señal `data`. El circuito tiene una única señal de salida llamada `dout` y una constante *generic* que indica el número de bits de las señales de entrada y salida de datos.

La señal de carga tiene prioridad sobre la señal de reset, y la señal de reset tiene prioridad sobre la operación de desplazamiento. Cuando las señales `load` y `reset` valen '0', se realiza en el flanco de subida de la señal de reloj el desplazamiento de un bit a la izquierda o a la derecha dependiendo del valor de la señal `izqda`. Cuando la señal `izqda` vale '0', se realiza el desplazamiento de un bit a la derecha introduciendo un '0' a la izquierda y cuando vale '1' se realiza el desplazamiento de un bit a la izquierda introduciendo un '0' a la derecha.

Solución a la Pregunta 3

El código VHDL del circuito desplazador se muestra en Código VHDL 1.3.

```
----------------------------------------
-- Desplazador
library IEEE;
use IEEE.std_logic_1164.all;
entity desplazador is
    generic (nBits: integer:=8);
    port (   dout : out std_logic_vector( nBits-1 downto 0);
             data: in std_logic_vector( nBits-1 downto 0);
             clk, load, izqda, reset  : in   std_logic );
end desplazador;
architecture desplazador of desplazador is
    signal R: std_logic_vector(nBits-1 downto 0);
begin
    process (clk)
    begin
        if rising_edge(clk) then
            if (load = '1') then
                R <= data;
            elsif (reset = '1') then
                R <= (others => '0');
            elsif (izqda = '0') then
                R(nBits-1) <= '0';
                for index in nBits-2 downto 0 loop
                    R(index) <= R(index+1);
                end loop;
            elsif (izqda = '1') then
                R(0) <= '0';
                for index in 0 to nBits-2 loop
                      R(index+1) <= R(index);
                end loop;
            end if;
        end if;
    end process;
    dout <= R;
end architecture desplazador;
----------------------------------------
```

Código VHDL 1.3: Diseño del circuito desplazador.

PREGUNTA 4 (2 puntos)

Programe en VHDL un banco de pruebas para el circuito que ha diseñado al contestar a la Pregunta 3 para un valor de la constante `nBits` igual a 8. La señal de reloj (`clk`) debe tener un periodo de 20 ns e inicialmente valer '0'. El programa de test debe realizar consecutivamente las acciones siguientes:

1. *Reset*. La señal `reset` ha de tener el valor '1' durante 15 ns.
2. *Cargar el valor "11111100"*.
3. *Realizar 4 desplazamientos a la izquierda*. Comprobar que el circuito pasa consecutivamente por los valores "11111000", "11110000", "11100000" y "11000000".
4. *Cargar el valor "10111100"*.
5. *Realizar 4 desplazamientos a la derecha*. Comprobar que el circuito pasa por los valores "01011110", "00101111", "00010111" y "00001011".

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 desplazador se muestra en Código VHDL 1.4–1.5.

```
-------------------------------------------------
-- Banco de pruebas del registro de 4 bits
library IEEE;
use IEEE.std_logic_1164.all;

entity bp_desplazador is
end entity bp_desplazador;

architecture bp_desplazador of bp_desplazador is
   constant PERIODO          : time         := 20 ns;   -- Reloj
   constant nBits : integer := 8;
   signal     data, dout      : std_logic_vector(nBits-1 downto 0);     -- Salidas UUT
   signal     clk   : std_logic   := '0';                -- Entradas UUT
   signal     load, izqda, reset      : std_logic;

   component desplazador is
      generic    (nBits: integer:=8);
      port (    dout : out std_logic_vector( nBits-1 downto 0);
                 data: in std_logic_vector( nBits-1 downto 0);
                 clk, load, izqda, reset   : in    std_logic );
   end component desplazador;
begin
   -- Instanciar y conectar UUT
   uut : component desplazador
            generic map (nBits)
            port map (dout, data, clk, load, izqda, reset);

    clk <= not clk after (PERIODO/2);
    reset <= '1' ,'0' after 15 ns;

   gen_vec_test : process is
      variable error_count: integer:= 0;
      variable temp: std_logic_vector (nBits-1 downto 0);
   begin
```

Código VHDL 1.4: Banco de pruebas del desplazador.

```
      report "Comienza la simulación";
      load <= '1';
      data <= "11111100";
      wait for (2*PERIODO);
      if (dout /= data) then
        report "ERROR";
        error_count := error_count +1;
      end if;
      load <= '0';
      izqda <= '1';
      temp := data;
      for i in 0 to 3 loop
       wait until rising_edge(clk);
       wait for (PERIODO/2);
       temp := temp(6 downto 0)&'0';
       if (dout /= temp) then
          report "ERROR DESP IZQDA";
          error_count := error_count +1;
       end if;
      end loop;
      load <= '1';
      data <= "10111100";
      wait for (2*PERIODO);
      if (dout /= data) then
        report "ERROR CARGA DATOS";
        error_count := error_count +1;
      end if;
      load <= '0';
      izqda <= '0';
      temp := data;
      for i in 0 to 3 loop
       wait for (PERIODO);
       temp := '0'&temp(7 downto 1);
       if (dout /= temp) then
          report "ERROR DESP DRCHA";
          error_count := error_count +1;
       end if;
      end loop;

      report "Finaliza la simulacion: "
       &integer'image(error_count) & " errores";
      wait;        -- Final del bloque process
   end process gen_vec_test;
end architecture bp_desplazador;
-------------------------------------------------
```

Código VHDL 1.5: Continuación del banco de pruebas del desplazador.