INGENIERÍA DE COMPUTADORES 3

Solución al examen de Septiembre 2022

PREGUNTA 1 (2 puntos)

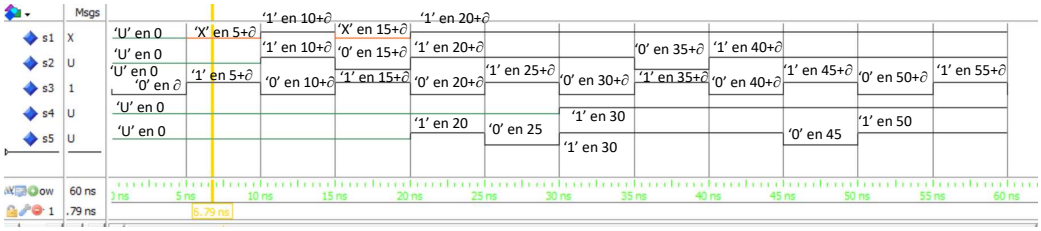
Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `s1`, `s2`, `s3`, `s4` y `s5` entre los instantes 0 y 60 ns. En el cronograma se debe indicar para cada una de estas señales el instante de tiempo en que la señal cambia de valor así como su nuevo valor.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono1 is
end entity crono1;
architecture crono1 of crono1 is
   signal s1, s2, s3, s4, s5 : std_logic;
begin
    bloque1 : process
    begin
       s1 <= '0';
       wait for 10 ns;
       s1 <= '1';  s2 <= '1';
       wait for 5 ns;
       s1 <= '0';  s2 <= '0';
       wait for 5 ns;
       s1 <= '1';  s2 <= '1';
       wait for 15 ns;
       s2 <= '0';
       wait for 5 ns;
       s1<= '1';  s2<='1';
       wait;
    end process bloque1;
    bloque2 : process
```

```
    begin
       s3 <= '0';
       wait for 5 ns;
       s1 <= '1';  s3 <= '1';
       wait for 5 ns;
    end process bloque2;
    s4 <= s2 after 10 ns;
    s5 <= transport s2 after 10  ns;
end architecture crono1;
```

Solución a la Pregunta 1

En la siguiente figura se muestra el cronograma de evolución de las señales.



PREGUNTA 2 (3 puntos)

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

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

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

2.c) (2.5 puntos) La estructura de un circuito combinacional detector de paridad de números de n bits, con $n >= 2$. La salida del circuito es 1 si la entrada tiene un número par de unos. En cualquier otro caso, la salida del circuito es 0. La **architecture** debe describir la estructura del circuito combinacional, instanciando y conectando adecuadamente las puertas lógicas NOT y XOR necesarias. Emplee las sentencias **generic**, **generate** y las puertas lógicas cuyo diseño ha realizado al contestar los dos apartados anteriores.

Solución a la Pregunta 2

El Código VHDL 1.1 es una posible forma de diseñar la puerta NOT de una entrada y la puerta XOR de dos entradas.

```
-------------------------------------
-- 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;
-------------------------------------
-- XOR de 2 entradas
library IEEE; use IEEE.std_logic_1164.all;

entity xor2 is
   port ( y0      : out std_logic;
          x0, x1 : in  std_logic );
end entity xor2;

architecture xor2 of xor2 is
begin
   y0 <= x0 xor x1;
end architecture xor2;
-------------------------------------
```

Código VHDL 1.1: Diseño de las puertas NOT y XOR de 2 entradas.

El Código VHDL 1.2 es un posible diseño del detector de paridad de números de n bits. Con el fin de ilustrar el tipo de circuito diseñado, en la Figura 1.1 se muestra un diagrama esquemático del circuito detector para $n = 3$.

```
-----------------------------------------------
-- Detector de paridad usando sentencia GENERATE condicional
library IEEE;
use IEEE.std_logic_1164.all;

entity paridad is
    generic ( n : integer := 3);
    port ( parity      : out   std_logic;
           parity_IN   : in    std_logic_vector(n-1 downto 0));
end entity paridad;

architecture paridad of paridad is
  signal temp: std_logic_vector(n-2 downto 0);
  signal tempn : std_logic;
  component xor2 is
    port ( y0      : out std_logic;
           x0, x1 : in   std_logic );
  end component xor2;

  component not1 is
    port ( y      : out std_logic;
           x      : in   std_logic );
  end component not1;
begin
    gen_array: for I in 0 to n-2 generate
       inicial: if I = 0 generate
         xor_0: xor2
         port map(temp(I), parity_IN(0), parity_IN(I+1));
       end generate inicial;

       intermedio: if I /= 0 generate
         resto_xor: xor2
         port map(temp(I), temp(I-1), parity_IN(I+1));
       end generate intermedio;
  end generate gen_array;
  not1_0: not1
  port map(parity, temp(n-2));
end architecture paridad;
-----------------------------------------------
```

Código VHDL 1.2: Diseño del detector de paridad.

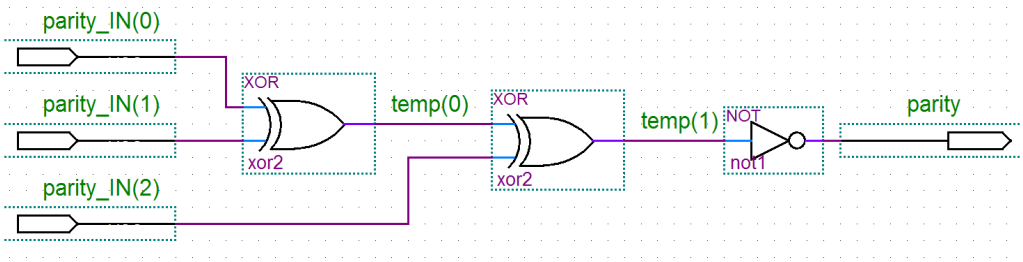


Figura 1.1: Circuito detector de paridad de números de tres bits.

PREGUNTA 3 (2 puntos)

Programe en VHDL el banco de pruebas del circuito combinacional que ha diseñado al contestar a la Pregunta 2c. Suponga que el número de bits que tiene como entrada el circuito es 3 (es decir, $n = 3$). 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

El Código VHDL 1.3, mostrado en la página siguiente, es un posible banco de pruebas del circuito detector de paridad de tres bits.

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

entity bp_detector_paridad is
end entity bp_detector_paridad;

architecture bp_detector_paridad of bp_detector_paridad is
  signal parity     : std_logic;   -- Conectar salidas UUT
  signal parity_IN : std_logic_vector( 2 downto 0);   -- Conectar entradas UUT

  component paridad is
     generic (n: integer := 3);
     port (  parity     : out   std_logic;
           parity_IN  : in    std_logic_vector(n-1 downto 0));
  end component paridad;

begin
  -- Instanciar y conectar UUT
  uut : component paridad
       port map( parity, parity_IN );
  gen_vec_test : process
    variable test_in     : unsigned (2 downto 0); -- Vector de test
    variable esperado_parity    : std_logic;
    variable error_count         : integer    := 0;
  begin
    test_in := B"000";
    report "Comienza la simulación";
    for count in 0 to 7 loop
      parity_IN <= std_logic_vector(test_in);
      if (count=0) then    esperado_parity   := '1'; --parity_IN = 000
      elsif (count=1) then  esperado_parity := '0'; --parity_IN = 001
      elsif (count=2) then  esperado_parity := '0'; --parity_IN = 010
      elsif (count=3) then  esperado_parity := '1'; --parity_IN = 011
      elsif (count=4) then  esperado_parity := '0'; --parity_IN = 100
      elsif (count=5) then  esperado_parity := '1'; --parity_IN = 101
      elsif (count=6) then  esperado_parity := '1'; --parity_IN = 110
      else  esperado_parity := '0';  --parity_IN = 111
      end if;

      wait for 10 ns;
      test_in := test_in + 1;
      if ( esperado_parity /= parity ) then
          report "ERROR en la salida valida. Valor esperado: "  &
                 std_logic'image(esperado_parity)              &
                 ", valor actual: "                               &
                 std_logic'image(parity)                       &
                 " 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_detector_paridad;
-------------------------------------
```

Código VHDL 1.3: Banco de pruebas del detector de paridad.

PREGUNTA 4 (3 puntos)

Diseñe usando VHDL un registro de desplazamiento de 4 bits conversor serie a paralelo. El circuito tiene las entradas siguientes: señal de reloj (`Clock`), señal de control de desplazamiento hacia la derecha (`Shift`), señal de reset asíncrono activo a nivel alto (`Reset`) y señal de entrada serie de datos (`Serial_in`). El circuito tiene una señal de 4 bits de salida paralelo de datos (`Q`). La **entity** del circuito es:

```
entity registro is port(
  Q : out std_logic_vector(3 downto 0);
  Clock, Shift, Serial_in, Reset : in std_logic);
end entity registro;
```

Cuando la señal `Reset` toma el valor '1', todos los bits del contenido del registro toman el valor 0. No se realiza ninguna operación mientras `Reset` vale '0' y la señal `Shift` vale '0'. Mientras la señal `Reset` vale '0' y la señal `Shift` vale '1', en cada flanco de subida de la señal de reloj se desplaza el contenido del registro un bit a la derecha y se carga `Serial_in` en el bit situado más a la izquierda (bit más significativo del registro). La señal de salida coincide con el contenido del registro.

El diseño del registro en VHDL debe realizarse describiendo el comportamiento del circuito, empleando para ello un único bloque **process**.

Solución a la Pregunta 4

El código VHDL que describe el comportamiento del circuito se muestra en Código VHDL 1.4.

```
-----------------------------------------------
-- Registro serie-a-paralelo
library IEEE;
use IEEE.std_logic_1164.all;

entity registro is port(
  Q : out std_logic_vector(3 downto 0);
  Clock, Shift, Serial_in, Reset : in std_logic);
end entity registro;

architecture registro of registro is
    signal content: std_logic_vector(3 downto 0);
begin
    process(Clock, Reset)
    begin
      if (Reset = '1') then
          content <= "0000";
        elsif(rising_edge(Clock)) then
              if (Shift = '1') then
                content <= Serial_in & content(3 downto 1);-- desplazamiento
derecha
                                    --y rellena a la izqda con bits
                                    --entrada Serial_in
                end if;
        end if;
    end process;

    Q <= content;

end architecture registro;
-----------------------------------------------
```

Código VHDL 1.4: Registro de desplazamiento de 4 bits conversor serie a paralelo.