INGENIERÍA DE COMPUTADORES 3

Solución al examen de Junio 2024, Segunda Semana

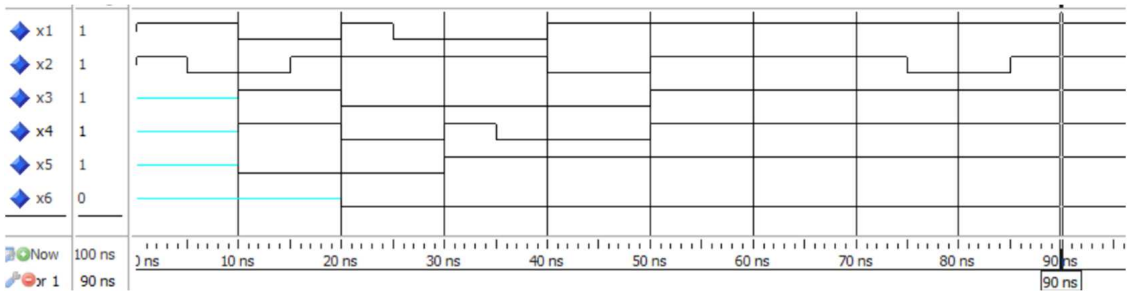
PREGUNTA 1 (2 puntos)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `x1`, `x2`, `x3`, `x4`, `x5` y `x6` entre los instantes 0 y 90 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 crono2 is
end entity crono2;
architecture crono2 of crono2 is
  signal x1, x2, x3, x4, x5, x6 : std_logic;
begin
  x1 <= '1', '0' after 10 ns, '1' after 20 ns,
        '0' after 25 ns, '1' after 40 ns;
  Proc1: process
  begin
    x2 <= '1';  wait for 5 ns;
    x2 <= '0';  wait for 10 ns;
    x2 <= '1';  wait for 20 ns;
    x2 <= '0';
  end process;
  x3 <= x1 after 10 ns;
  x4 <= transport x1 after 10 ns;
  Proc2: process
    variable v  : std_logic;
  begin
    for i in 0 to 3 loop
       v := x1 xor x2; x5 <= v;
       x6 <= x5; wait for 10 ns;
    end loop;
    wait;
  end process;
end architecture crono2;
```

Solución a la Pregunta 1

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



PREGUNTA 2 (3 puntos)

Programe en VHDL un sumador de dos números BCD de 8 bits con 1 bit de acarreo de entrada cuya salida es otro número BCD de 8 bits y 1 bit de acarreo de salida. El circuito tiene como entradas las señales `a` y `b`, que representan los dos números BCD de 8 bits, y una señal de acarreo de 1 bit llamada `cin`. Tiene como salidas la señal `sum` de 8 bits, que contiene el resultado de la suma, y la señal `cout` de 1 bit con el valor del acarreo de salida. Por ejemplo, si `a` es `00010001` entonces esta señal representa al número decimal 11. Si `b` es `00110001` entonces esta señal representa al número decimal 31. El resultado de la suma si la señal de acarreo de entrada es `0` será el número decimal 42, cuyo código BCD es `01000010`. Por tanto, la señal `sum` tendrá el valor `01000010` y la señal `cout` tendrá el valor `0`.

Se debe emplear el siguiente paquete:

```
package BCD_CONSTANTS is
   constant WIDTH     :  integer := 8;
                         -- Núm. bits de los operandos
end package BCD_CONSTANTS;
```

En el diseño únicamente pueden emplearse los dos siguientes paquetes de la librería IEEE:

```
   IEEE.std_logic_1164
   IEEE.numeric_std
```

El circuito sumador ha de tener la siguiente **entity**:

```
entity BCDSum is
 port(    sum  : out std_logic_vector (WIDTH-1 downto 0);
          cout : out std_logic;
          a, b : in std_logic_vector (WIDTH-1 downto 0);
          cin  : in std_logic );
end entity BCDSum;
```

Solución a la Pregunta 2

El Código VHDL 1.1 es el código del sumador BCD.

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4  use work.BCD_CONSTANTS.all;
5
6  architecture bcdSum of bcdSum is
7  begin
8    process(a, b, cin) is
9      variable sumv : unsigned(WIDTH downto 0);
10     variable acarreo : unsigned(4 downto 0);
11   begin
12     acarreo := "00000";
13     acarreo(0) := cin;
14     sumv(4 downto 0) :=
15           unsigned('0' & a(3 downto 0))
16         + unsigned('0' & b(3 downto 0))
17         + acarreo;
18     if (sumv(4 downto 0) > 9) then
19          sumv(4 downto 0) := sumv(4 downto 0) + 6;
20     end if;
21     acarreo(0) := sumv(4);
22     sumv(8 downto 4) :=
23           unsigned('0' & a(7 downto 4))
24         + unsigned('0' & b(7 downto 4))
25         + acarreo;
26     if (sumv(8 downto 4) > 9) then
27          sumv(8 downto 4) := sumv(8 downto 4) + 6;
28       end if;
29
30     sum <= std_logic_vector(sumv(WIDTH-1 downto 0));
31     cout <= sumv(WIDTH);
32    end process;
33 end architecture bcdSum;
```

Código VHDL 1.1: Architecture del sumador BCD.

PREGUNTA 3 (3 puntos)

Programe un banco de pruebas del circuito combinacional que ha diseñado en la Pregunta 2. El banco de pruebas va a comprobar únicamente que el circuito genera correctamente los valores de las señales de salida en los siguientes cuatro casos:

- `a = "00010001", b = "00010001", cin = '0'`
- `a = "10011001", b = "00000001", cin = '0'`
- `a = "10011001", b = "00000001", cin = '1'`
- `a = "00110101", b = "10001000", cin = '1'`

El banco de pruebas debe permitir comprobar que los valores de las señales de salida del circuito combinacional a probar coinciden con los esperados, mostrando un mensaje de error en caso contrario. Al final del test, debe mostrarse un mensaje indicando que el test ha finalizado y el número total de errores detectados.

Solución a la Pregunta 3

El código VHDL del banco de pruebas se muestra en Código VHDL 1.2.

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4  use work.BCD_CONSTANTS.all;
5
6  entity bp_bcdSumador is
7    constant DELAY: time := 10 ns;
8  end entity bp_bcdSumador;
9
10 architecture bp_bcdSumador of bp_bcdSumador is
11   signal sum: std_logic_vector(WIDTH-1 downto 0);
12   signal cout: std_logic;
13   signal a, b: std_logic_vector(WIDTH-1 downto 0);
14   signal cin: std_logic;
15
16   component BCDSum is
17    port(    sum  : out std_logic_vector (WIDTH-1 downto 0);
18             cout : out std_logic;
19             a, b : in std_logic_vector (WIDTH-1 downto 0);
20             cin  : in std_logic );
21   end component BCDSum;
22
23   procedure check_bcd (actual_sum : in std_logic_vector (WIDTH-1 downto
       0);
```

```
                    actual_cout : in std_logic;
                    expected_sum : in std_logic_vector (WIDTH-1 downto
    0);
                    expected_cout : in std_logic;
                    error_count: inout integer) is
  begin
   assert ((expected_sum = actual_sum) and
           (expected_cout = actual_cout))
    report "ERROR: Expected expected (cout, sum) result" &
      integer'image(TO_INTEGER(unsigned(expected_cout&expected_sum)))
      & "/= actual result" &
      integer'image(TO_INTEGER(unsigned(actual_cout & actual_sum)))
      & "at time" & time'image(now);

   if ((expected_sum /= actual_sum) or
     (expected_cout /= actual_cout)) then
     error_count := error_count + 1;
   end if;
  end procedure check_bcd;

begin
  UUT:component BCDSum port map (sum, cout, a, b, cin);
  main: process is
    variable error_count: integer := 0;
    variable expected_sum: std_logic_vector(WIDTH-1 downto 0);
    variable expected_cout : std_logic;
    variable temp : integer;
  begin
    report "Start simulation";
    --Caso 1: a = "00010001", b = "00010001", cin = '0'
     a <= "00010001";
     b <= "00010001";
     cin <= '0';
     wait for DELAY;
     expected_sum := "00100010";
     expected_cout := '0';
     check_bcd(sum,cout,expected_sum, expected_cout, error_count);
     wait for DELAY;
     --Caso 2: a = "10011001", b = "00000001", cin = '0'
     a <= "10011001";
     b <= "00000001";
     cin <= '0';
     wait for DELAY;
     expected_sum := "00000000";
     expected_cout := '1';
     check_bcd(sum,cout,expected_sum, expected_cout, error_count);
     wait for DELAY;
     --Caso 3:   a = "10011001", b = "00000001", cin = '1'
     a <= "10011001";
     b <= "00000001";
     cin <= '1';
     wait for DELAY;
     expected_sum := "00000001";
     expected_cout := '1';
```

```
77      check_bcd(sum,cout,expected_sum, expected_cout, error_count);
78      wait for DELAY;
79     --Caso 4: a = "00110101", b = "10001000", cin = '1'
80      a <= "00110101";
81      b <= "10001000";
82      cin <= '1';
83      wait for DELAY;
84      expected_sum := "00100100";
85      expected_cout := '1';
86      check_bcd(sum,cout,expected_sum, expected_cout, error_count);
87      wait for DELAY;
88      assert (error_count = 0)
89        report "ERROR: Hay " &
90           integer'image(error_count) & "errores";
91      if (error_count = 0) then
92        report "Simulacion terminada SIN errores";
93      end if;
94      wait;
95   end process main;
96 end architecture bp_bcdSumador;
```

Código VHDL 1.2: Banco de pruebas del sumador BCD.

PREGUNTA 4 (2 puntos)

Programe en VHDL un registro de desplazamiento de 8 bits con entrada y salida serie. El registro realiza una operación de desplazamiento de 1 bit en cada flanco de subida de la señal de reloj (`clk`).

La **entity** del registro de desplazamiento se muestra a continuación:

```
entity registro8 is port
  ( Serial_out : out std_logic;
    Serial_in  : in  std_logic;
    clk        : in  std_logic );
end entity registro8;
```

Solución a la Pregunta 4

El Código VHDL 1.3 es el diseño del registro.

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

architecture registro8 of registro8 is
begin
   process(clk)
     variable R: std_logic_vector(6 downto 0);
   begin
     if (rising_edge(clk)) then
          Serial_out <= R(6);
         for index in 5 downto 0 loop
            R(index+1) := R(index);
         end loop;
         R(0) := Serial_in;
     end if;
   end process;
end architecture registro8;
```

Código VHDL 1.3: Registro de desplazamiento de 8 bits.