INGENIERÍA DE COMPUTADORES 3

Solución al examen de Septiembre 2024

PREGUNTA 1 (2 puntos)

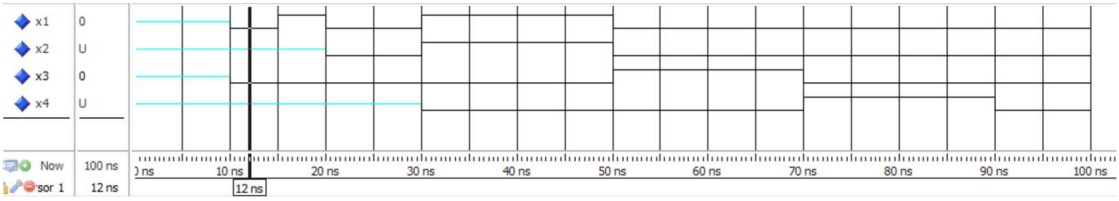
Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `x1`, `x2`, `x3` y `x4` entre los instantes 0 y 100 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono1 is
end entity crono1;

architecture crono1 of crono1 is
  signal s1, s2, s3: std_logic := '0';
  signal x1, x2, x3, x4 : std_logic;
begin
  process (s2)
  begin
    if (rising_edge(s2)) then
       s3 <= x1;
       x3 <= s3;
       x4 <= x3;
    end if;
  end process;
  s1 <= '0',  '1' after 5 ns, '0' after 10 ns,
        '1' after 20 ns, '0' after 40 ns;
  s2 <= not s2 after 10 ns;
  x1 <= transport s1 after 10 ns;
  x2 <= s1 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 cuatro **architecture** que describan lo siguiente:

2.a) (0.5 puntos) La primera **architecture** debe describir el comportamiento de un multiplexor con un entrada de selección (`sel`), dos entradas de 8 bits (`x1`, `x2`) y una salida de 8 bits (`y`). En el diseño de esta **architecture** emplee un bloque **process** con una sentencia **case**. La **entity** del circuito es la siguiente:

```
entity mux is port(
     y       :   out std_logic_vector(7 downto 0);
     x1, x2  :   in  std_logic_vector(7 downto 0);
     sel     :   in std_logic);
end entity mux;
```

2.b) (0.75 puntos) La segunda **architecture** debe describir el comportamiento de una unidad aritmética con dos entradas `A` y `B`. El tamaño de los dos operandos de entrada, `A` y `B`, es de 8 bits. La operación que realiza es especificada por la señal de entrada `sel`. Emplee en la programación del circuito una sentencia concurrente condicional (**when - else**). A continuación, se muestra la **entity** y la tabla de operaciones correspondiente al circuito.

```
entity uaritm is port(
     y       :   out std_logic_vector(7 downto 0);
     A, B    :   in  std_logic_vector(7 downto 0);
     sel     :   in  std_logic_vector(1 downto 0));
end entity uaritm;
```

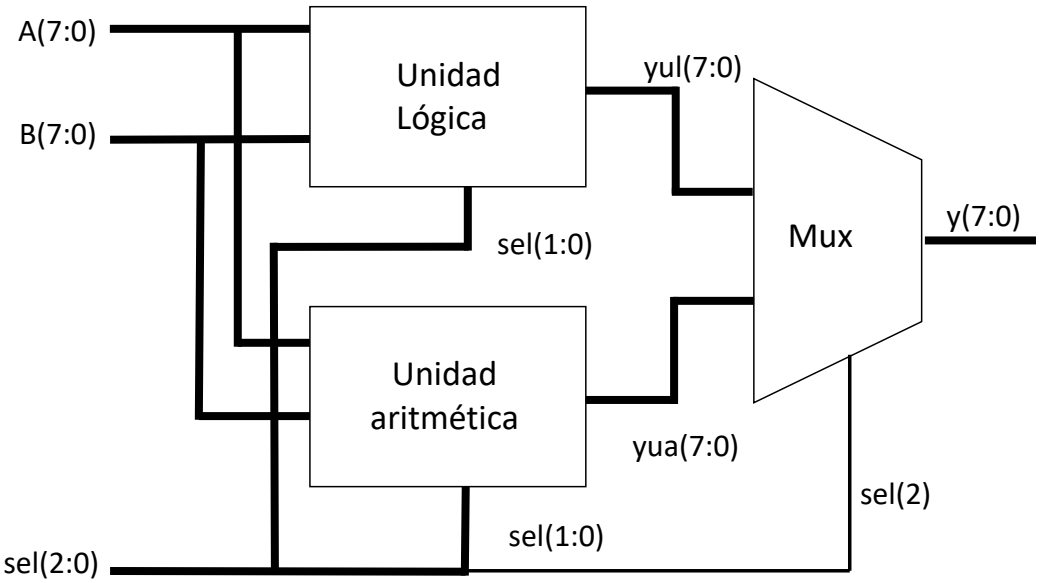
sel	Operación
0 0	A
0 1	$A + B$
1 0	A - B
1 1	-A

2.c) (0.75 puntos) La tercera **architecture** debe describir el comportamiento de una unidad lógica con dos entradas `A` y `B`. El tamaño de los dos operandos de entrada, `A` y `B`, es de 8 bits. La operación que realiza es especificada por la señal de entrada `sel`. Emplee en la programación del circuito un bloque **process** con una sentencia **if**. A continuación, se muestra la **entity** y la tabla de operaciones correspondiente al circuito.

```
entity ulogic is port(
     y      :   out std_logic_vector(7 downto 0);
     A, B   :   in  std_logic_vector(7 downto 0);
     sel    :   in  std_logic_vector(1 downto 0));
end entity ulogic;
```

sel	Operación
0 0	A or B
0 1	A nand B
1 0	A nor B
1 1	A xor B

2.d) (1 punto) Por último, la cuarta **architecture** debe describir la estructura de una unidad aritmético lógica (ALU). La **architecture** debe describir la estructura del circuito combinacional mostrado en la figura, instanciando y conectando adecuadamente los circuitos cuyo diseño ha realizado al contestar los tres apartados anteriores. Los dos bits menos significativos de la señal `sel` seleccionan la operación a realizar por la unidad lógica y por la unidad aritmética. El bit más significativo de la señal `sel` selecciona si la salida de la ALU es la salida de la unidad aritmética o la salida de la unidad lógica.



La **entity** del circuito es:

```
entity alu is port(
     y      :   out std_logic_vector(7 downto 0);
     A, B   :   in  std_logic_vector(7 downto 0);
     sel    :   in  std_logic_vector(2 downto 0));
end entity alu;
```

Solución a la Pregunta 2

El Código VHDL 1.1–1.4 son las soluciones de los Apartados 2.a–2.d, respectivamente.

```vhdl
-- Multiplexor 2 a 1
library IEEE;
use IEEE.std_logic_1164.all;
architecture mux of mux is
begin
  process(x1, x2, sel) is
    begin
    case sel is
        when ('0') =>
          y <= x1;
        when others =>
          y <= x2;
      end case;
  end process;
end architecture mux;
```

Código VHDL 1.1: Multiplexor.

```vhdl
-- Unidad aritmética
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
architecture uaritm of uaritm is
begin
  y <=   A when (sel="00")
    else std_logic_vector (signed(A)+signed(B)) when (sel="01")
    else std_logic_vector (signed(A)-signed(B)) when (sel="10")
    else std_logic_vector (-signed(A)        ) ;
end architecture uaritm;
```

Código VHDL 1.2: Unidad aritmética.

```vhdl
-- Unidad lógica
library IEEE;
use IEEE.std_logic_1164.all;

architecture ulogic of ulogic is
begin
  process(A, B, sel) is
    begin
      if (sel = "00") then
         y <= A or B;
      elsif (sel = "01") then
         y <= A nand B;
      elsif (sel = "10") then
         y <= A nor B;
      elsif (sel = "11") then
         y <= A xor B;
      end if;
  end process;
end architecture ulogic;
```

Código VHDL 1.3: Unidad lógica.

```
-- Unidad aritmetico logica
library IEEE;
use IEEE.std_logic_1164.all;

architecture alu of alu is
  signal yul, yua: std_logic_vector(7 downto 0);

  component mux is
   port(  y      : out std_logic_vector(7 downto 0);
          x1, x2 : in std_logic_vector(7 downto 0);
          sel    : in std_logic);
  end component mux;

  component uaritm is
    port(  y      :   out std_logic_vector(7 downto 0);
           A, B   :   in  std_logic_vector(7 downto 0);
           sel    :   in std_logic_vector(1 downto 0));
  end component uaritm;

  component ulogic  is
     port(  y      :   out std_logic_vector(7 downto 0);
            A, B   :   in  std_logic_vector(7 downto 0);
            sel    :   in std_logic_vector(1 downto 0));
  end component ulogic;

begin
  -- Instanciación y conexión de los componentes
  ulogic_1 : component ulogic port map
          (y => yul, A => A, B => B, sel => sel(1 downto 0) );
  uaritm_1 : component uaritm port map
          (y => yua, A => A, B => B, sel => sel(1 downto 0) );
  mux_1 : component mux port map
          (y => y, x1 => yua, x2 => yul, sel => sel(2) );
end architecture alu;
```

Código VHDL 1.4: Unidad aritmético-lógica.

PREGUNTA 3 (3 puntos)

Escriba en VHDL un circuito secuencial síncrono capaz de detectar si recibe la secuencia `"0100"` por su entrada `X`. La **entity** del circuito se muestra a continuación.

```
entity detector is
   port( Y     : out std_logic;
         state : out std_logic_vector(2 downto 0);
         X     : in std_logic;
         reset : in std_logic;
         clk   : in std_logic );
end entity detector;
```

El circuito tiene una señal de reloj (`clk`), una entrada serie de un bit (`X`), una señal de reset asíncrona activa en `'1'` (`reset`), una señal que indica el estado en que se encuentra el circuito (`state`) y una señal de salida de un bit (`Y`).

La señal `Y` se pone a `'1'` si los últimos 4 bits recibidos por la entrada `X` se corresponden con la secuencia `"0100"`. La máquina no vuelve al estado inicial tras haber reconocido la secuencia, sino que detecta secuencias solapadas. La señal `reset` pone el circuito en su estado inicial. Todos los cambios, excepto el reseteo del circuito, tienen lugar en el flanco de subida de la señal de reloj.

Escriba en VHDL la **architecture** que describe el comportamiento del circuito en términos de una máquina de Moore. Dibuje el diagrama de estados correspondiente al circuito que ha diseñado.

Solución a la Pregunta 3

El circuito diseñado tiene 5 estados (S0, S1, S2, S3 y S4). El circuito se encuentra en el estado S0 cuando no se ha detectado ningún bit de la secuencia. Está en los estados S1, S2, S3 o S4 cuando se han detectado respectivamente 1, 2, 3 ó 4 primeros bits de la secuencia. En la Figura 1.1, se muestra el diagrama de estados de dicho circuito.

El código VHDL que describe el comportamiento del circuito en términos de una máquina de Moore, se muestra en Código VHDL 1.5.

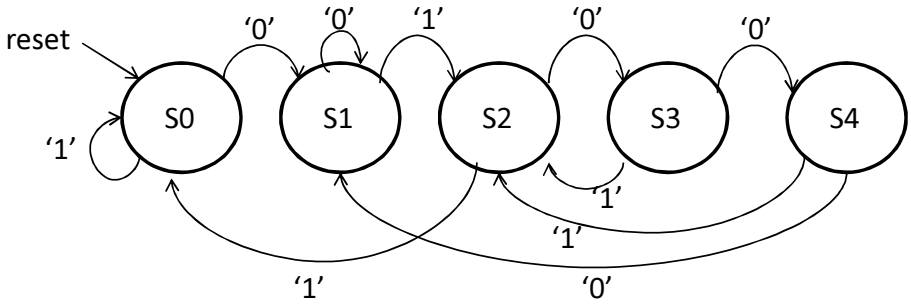


Figura 1.1: Diagrama de estados del circuito.

```
1  ---Detector de la secuencia 0100
2  library IEEE;
3  use IEEE.std_logic_1164.all;
4
5  architecture detector of detector is
6        signal internal_state: std_logic_vector(2 downto 0);
7  begin
8   state <= internal_state;
9
10   --Cálculo salida
11   salida: process (internal_state) is
12   begin
13      case internal_state is
14          when "100" => Y <= '1';
15          when others => Y <= '0';
16       end case;
17   end process salida;
18
19   --Cálculo del próximo estado
20   proximo_estado: process (clk, reset)
21   begin
22       if (reset = '1') then --reset asíncrono
23           internal_state <= "000";
24       elsif (rising_edge(clk)) then
25           case internal_state is
26               when "000" =>   -- Estado actual: S0
27                   if X = '1' then
28                       internal_state <= "000";
29                   else
30                       internal_state <= "001";
31                  end if;
32               when "001" =>  --Estado actual: S1
33                   if X = '1' then
34                       internal_state <= "010";
35                   else
36                       internal_state <= "001";
37                   end if;
38               when "010" =>  --Estado actual: S2
39                   if X = '1' then
```

```
40                      internal_state <= "000";
41                  else
42                      internal_state <= "011";
43                end if;
44              when "011" =>  -- Estado actual: S3
45                  if X = '1' then
46                      internal_state <= "010";
47                  else
48                      internal_state <= "100";
49                end if;
50              when "100" =>  -- Estado actual: S4
51                  if X = '1' then
52                      internal_state <= "010";
53                  else
54                      internal_state <= "001";
55                end if;
56              when others=>  -- Por completitud
57                  internal_state <= "000";
58            end case;
59        end if;
60 end process proximo_estado;
61
62 end architecture detector;
```

Código VHDL 1.5: Circuito detector.

PREGUNTA 4 (2 puntos)

Programe en VHDL un banco de pruebas para el circuito secuencial que ha diseñado al contestar a la Pregunta 3. El programa de test debe primero resetear el circuito y a continuación cargar en el circuito, a través de la entrada `X`, los siete bits siguientes (en el orden indicado): `'0'`, `'1'`, `'0'`, `'0'`, `'1'`, `'0'`, `'0'`. Si los valores de las dos señales de salida de la UUT no coinciden con lo esperado, el programa de test debe mostrar el correspondiente mensaje.

Solución a la Pregunta 4

El Código VHDL 1.6 es el diseño del banco de pruebas solución de la Pregunta 4.

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

entity bp_detector is
end entity bp_detector;

architecture bp_detector of bp_detector is
  constant PERIODO   : time       := 100 ns; -- Reloj
  signal   state     : std_logic_vector(2 downto 0);-- Salidas UUT
  signal   Y         : std_logic;
  signal   clk       : std_logic  := '0';    -- Entradas UUT
  signal   reset, X  : std_logic;


  component detector is
   port( Y     : out std_logic;
         state : out std_logic_vector(2 downto 0);
         X     : in std_logic;
         reset : in std_logic;
         clk   : in std_logic);
  end component detector;

  -- Procedimiento para comprobar las salidas
  procedure comprueba_salidas
      ( esperado_state  :       std_logic_vector(2 downto 0);
        actual_state    :       std_logic_vector(2 downto 0);
        esperado_Y      :       std_logic;
        actual_Y        :       std_logic;
        error_count     :       inout integer) is
  begin
       -- Comprueba state
       if ( esperado_state /= actual_state ) then
          report "ERROR: Estado esperado ("     &
                 std_logic'image(esperado_state(2))    &
                 std_logic'image(esperado_state(1))    &
                 std_logic'image(esperado_state(0))    &
                 "), estado actual ("           &
                 std_logic'image(actual_state(2))      &
                 std_logic'image(actual_state(1))      &
                 std_logic'image(actual_state(0))      &
                 "), instante: "                &
                 time'image(now);
          error_count := error_count + 1;
       end if;        -- Comprueba Y
       if ( esperado_Y /= actual_Y ) then
          report "ERROR: Salida Y esperada ("     &
```

```
48                  std_logic'image(esperado_Y)     &
49                  "), salida actual ("            &
50                  std_logic'image(actual_Y)       &
51                  "), instante: "                 &
52                  time'image(now);
53          error_count := error_count + 1;
54       end if;
55   end procedure comprueba_salidas;
56 begin
57   -- Instanciar y conectar UUT
58   uut : component detector port map
59                   (Y, state, X, reset, clk);
60
61   reset <=   '0', '1' after (PERIODO/4),
62              '0' after (PERIODO/2);
63   clk <= not clk after (PERIODO/2);
64   gen_vec_test : process is
65     variable error_count : integer := 0; -- Núm. errores
66   begin
67     report "Comienza la simulación";
68      -- Vectores de test y comprobación del resultado
69       X <= '0';   wait for PERIODO;   -- 1
70       comprueba_salidas("001", state, '0', Y, error_count);
71       X <= '1';   wait for PERIODO;   -- 2
72       comprueba_salidas("010", state, '0', Y, error_count);
73       X <= '0';   wait for PERIODO;   -- 3
74       comprueba_salidas("011", state, '0', Y, error_count);
75       X <= '0';   wait for PERIODO;   -- 4
76       comprueba_salidas("100", state, '1', Y, error_count);
77       X <= '1';   wait for PERIODO;   -- 5
78       comprueba_salidas("010" ,state, '0', Y, error_count);
79       X <= '0';   wait for PERIODO;   -- 6
80       comprueba_salidas("011", state, '0', Y, error_count);
81       X <= '0';   wait for PERIODO;   -- 7
82       comprueba_salidas("100", state, '1', Y, error_count);
83     -- Informe final
84        report  "Hay "              &
85                integer'image(error_count ) &
86                " errores.";
87     wait;     -- Final del bloque process
88   end process gen_vec_test;
89 end architecture bp_detector;
```

Código VHDL 1.6: Banco de pruebas del circuito detector.