INGENIERÍA DE COMPUTADORES 3

Solución al examen de Junio 2022, Primera Semana

PREGUNTA 1 (2 puntos)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `s1`, `s2`, `s3` y `s4` 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 crono is
end entity crono;
architecture crono of crono is
  signal x1, x2, x3, s1, s2, s3, s4: std_logic;
begin
  x1 <= '1',
        '0' after 15 ns,
        '1' after 20 ns,
        '0' after 35 ns,
        '1' after 50 ns;
  x2 <= '0',
        '1' after 25 ns,
        '0' after 30 ns;
  x3 <= '0',
        '1' after 10 ns,
        '0' after 20 ns;
  s1 <= x1 after 5 ns;
  Proc1: process (x1, x2, x3)
    variable v1: std_logic;
  begin
    s2 <= x1 xor x2;
    v1 := x1 xor x2;
    s3 <= s2;
    s4 <= v1;
  end process;
end architecture crono;
```

Solución a la Pregunta 1

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

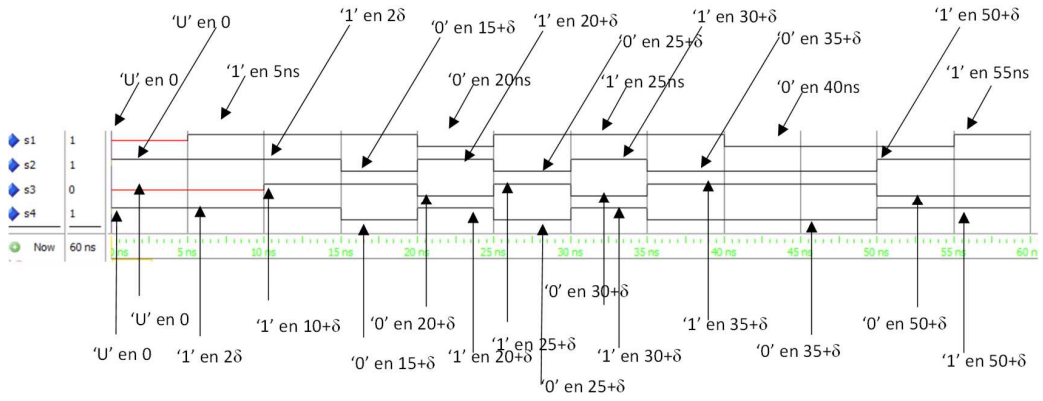


Figura 1.1: Cronograma de evolución de las señales.

PREGUNTA 2 (3 puntos)

Se quiere programar usando VHDL un circuito combinacional desplazador. El circuito tiene dos señales de entrada: x de 4 bits y la señal de selección s de dos bits. Y tiene una señal de salida y de 4 bits. El comportamiento del circuito se indica en la siguiente tabla de operación.

s	x	y	Operación
"00"	$x_3x_2x_1x_0$	$x_3x_2x_1x_0$	Pasa el dato
"01"	$x_3x_2x_1x_0$	$x_2x_1x_0$0	Desplaza a la izquierda y rellena con ceros
"10"	$x_3x_2x_1x_0$	0$x_3x_2x_1$	Desplaza a la derecha y rellena con ceros
"11"	$x_3x_2x_1x_0$	$x_0x_3x_2x_1$	Rota a la derecha

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

```
entity Desplazador is
   port( y: out std_logic_vector ( 3 downto 0);
         s  : in std_logic_vector (1 downto 0);
         x  : in std_logic_vector ( 3 downto 0));
end entity Desplazador;
```

2.a) (1.5 puntos) Escriba en VHDL la **architecture** que describe el comportamiento del circuito combinacional empleando sólo un bloque **process** y una sentencia **case**.

2.b) (0.5 puntos) Escriba en VHDL la **architecture** de un multiplexor 4 a 1 de un bit que describa su comportamiento empleando una sentencia **when–else**. El multiplexor ha de tener la **entity** siguiente:

```
entity mux4a1 is
   port( y: out std_logic;
         s: in std_logic_vector (1 downto 0);
         x: in std_logic_vector ( 3 downto 0));
end entity mux4a1;
```

2.c) (1 punto) Diseñe el circuito combinacional mediante conexión de multiplexores 4 a 1 como el descrito en el anterior apartado. Escriba en VHDL una **architecture** que describa la *estructura* del circuito que ha dibujado, instanciando y conectando los multiplexores 4 a 1 que ha diseñado anteriormente.

Solución a la Pregunta 2

El código VHDL solución del apartado 2.a se muestra en Codigo VHDL 1.1.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
architecture Comportamiento OF Desplazador is
begin
   process(s,x)
   begin
      case s is
          when "00" => -- pasa sin modificar
              y <= x;
          when "01" => -- despl. izqda con 0
              y <= x(2 downto 0) & '0';
          when "10" => -- despl. drcha. con 0
              y <= '0' & x(3 downto 1);
          when others => -- rota drcha.
              y <= x(0) & x(3 DOWNTO 1);
    end case;
end process;
end Comportamiento;
```

Código VHDL 1.1: Architecture del circuito descrito en la Pregunta 2.a.

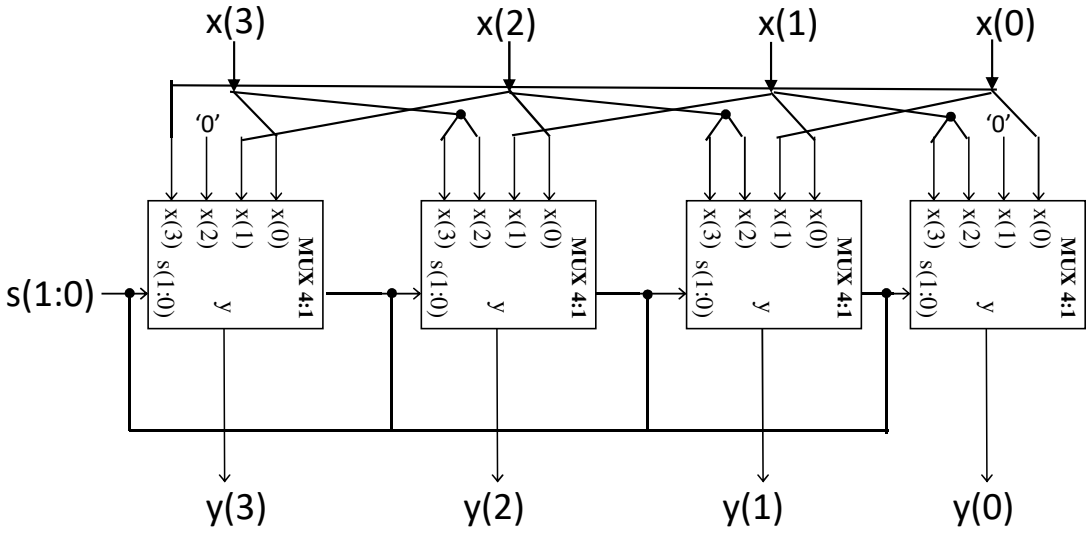
La **architecture** del multiplexor 4 a 1 se muestra en Codigo VHDL 1.2.

```
-- MUX 4x1
library IEEE;
use IEEE.std_logic_1164.all;

architecture mux4a1 of mux4a1 is
begin
   y <= x(0) when (s="00") else
        x(1) when (s ="01") else
        x(2) when (s= "10") else
        x(3);
end architecture mux4a1;
```

Código VHDL 1.2: Architecture del multiplexor 4 a 1.

El diagrama circuital se muestra en la siguiente figura.



El código VHDL solución del apartado 2.c se muestra en Codigo VHDL 1.3.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
architecture Estructura OF Desplazador is
   signal s0, s1, s2, s3: std_logic_vector(3 downto 0);
-- Declaración de las clases de los componentes
   component mux4a1 is
     port( y: out std_logic;
              s: in std_logic_vector (1 downto 0);
              x: in std_logic_vector ( 3 downto 0));
     end component mux4a1;
begin
  -- Instanciación y conexión de los componentes
   s0 <=  x(1)&x(1)&'0'&x(0);
   s1 <= x(2)&x(2)&x(0)&x(1);
   s2 <= x(3)&x(3)&x(1)&x(2);
   s3 <= x(0)&'0'&x(2)&x(3);
   Mux0 : component mux4a1 port map (y(0), s, s0);
   Mux1 : component mux4a1 port map (y(1), s, s1 );
   Mux2 : component mux4a1 port map (y(2), s, s2);
   Mux3 : component mux4a1 port map (y(3), s, s3);
end Estructura;
```

Código VHDL 1.3: Architecture del circuito descrito en la Pregunta 2.c.

PREGUNTA 3 (2 puntos)

Programe el banco de pruebas del circuito combinacional que ha diseñado en la Pregunta 2.c. Explique detalladamente cómo el programa de test comprueba de manera sistemática el funcionamiento del circuito. El banco de pruebas debe comprobar que los valores obtenidos de la UUT coinciden con los esperados, mostrando el correspondiente mensaje de error 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 del banco de pruebas se muestra en Código VHDL 1.4–1.5.

```
-- Simulacion finaliza a los 640 ns
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity bp_desplazador is
   constant DELAY : time := 10 ns;   -- Retardo usado en el test
end entity bp_desplazador;
architecture bp_desplazador of bp_desplazador is
   signal y : std_logic_vector(3 downto 0); -- Salida UUT
   signal x : std_logic_vector(3 downto 0); -- Entrada UUT
   signal s : std_logic_vector(1 downto 0); -- Entrada UUT
   component Desplazador is
     port( y   : out std_logic_vector ( 3 downto 0);
           s   : in std_logic_vector (1 downto 0);
           x   : in std_logic_vector ( 3 downto 0));
   end component Desplazador;
begin
```

Código VHDL 1.4: Diseño del banco de pruebas del desplazador (1/2).

```
   -- Instanciar y conectar UUT
   uut : component Desplazador port map
         ( y => y, x => x, s => s );
   gen_vec_test : process
      variable esperado_y : std_logic_vector(3 downto 0);
      variable error_count: integer     := 0;      -- Numero total de errores
   begin
    for i in 0 to 15 loop
        x  <= std_logic_vector(TO_UNSIGNED(i,4));
        for count in 0 to 3 loop
          s <= std_logic_vector(TO_UNSIGNED(count,2));
          if count=0 then
             esperado_y := std_logic_vector(TO_UNSIGNED(i,4));
          elsif count= 1 then
            esperado_y := std_logic_vector(TO_UNSIGNED(i,4) sll 1);
          elsif count = 2 then
            esperado_y := std_logic_vector(TO_UNSIGNED(i,4) srl 1);
          else
            esperado_y := std_logic_vector(TO_UNSIGNED(i,4) ror 1);
          end if;
          wait for 10 ns;
         if ( esperado_y /= y ) then
            report "ERROR en la salida valida. Valor esperado: "  &
                    std_logic'image(esperado_y(3))          &
                    std_logic'image(esperado_y(2))          &
                    std_logic'image(esperado_y(1))          &
                    std_logic'image(esperado_y(0))          &
                    ", valor actual: "                              &
                    std_logic'image(y(3))           &
                    std_logic'image(y(2))           &
                    std_logic'image(y(1))           &
                    std_logic'image(y(0))           &                  " en el
instante: "                         &
                    time'image(now);
            error_count := error_count + 1;
         end if;
        end loop;
     end loop;
     -- Informe del número total de errores
          report   "Hay "               &
                   integer'image(error_count) &
                   " errores.";
     wait;     -- Final de la simulación
   end process gen_vec_test;
end architecture bp_desplazador;
```

Código VHDL 1.5: Continuación del banco de pruebas del desplazador (2/2).

PREGUNTA 4 (3 puntos)

Diseñe usando VHDL el siguiente circuito secuencial síncrono. El circuito opera en el flanco de subida de la señal de reloj (`clk`) y tiene como entradas la señal de reloj `clk`, una señal `reset_n` síncrona activa a nivel bajo y una señal de un bit `x`. El circuito tiene una señal de salida: la señal de un bit `z`.

La señal de salida `z` tiene valor '1' cuando la secuencia de los cuatro últimos valores de la entrada `x` sea "0110", y tiene valor '0' en cualquier otro caso. El circuito detecta secuencias solapadas. El circuito se ha de describir como una máquina de estados de tipo Mealy.

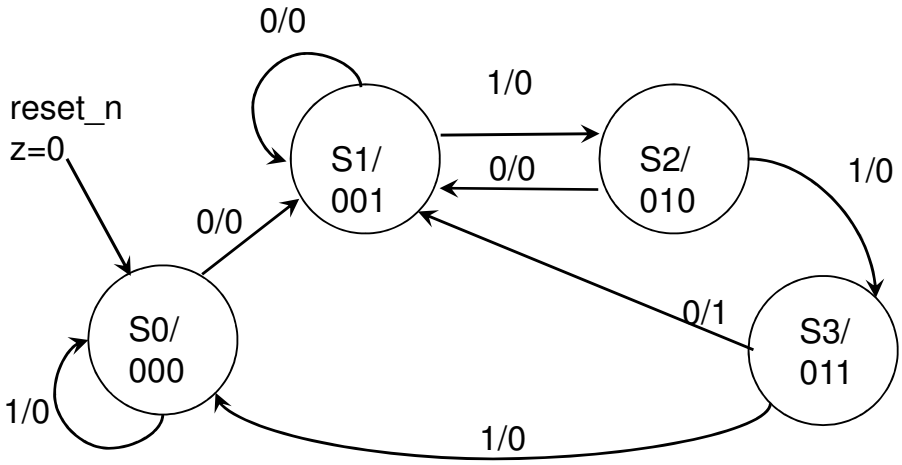
La **entity** del circuito se muestra a continuación.

```
entity detector is
   port ( z       : out std_logic;
          reset_n : in std_logic;
          clk     : in std_logic;
          x       : in std_logic );
end entity detector;
```

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

Solución a la Pregunta 4

El diagrama de estados correspondiente al circuito detector se muestra en la siguiente figura.



El código VHDL correspondiente al circuito detector se muestra en Código VHDL 1.6.

```
-------------------------------------------------
-- Máquina de Mealy detectora de la secuencia 0110

library IEEE;
use IEEE.std_logic_1164.all;

architecture detector of detector is
    signal state : std_logic_vector(1 downto 0);
begin
      -- Cálculo del próximo estado
      proximo_estado: process (clk) is
      begin
         if ( (reset_n = '0') and rising_edge(clk) ) then -- Reset síncrono
              state <= "00";
              z<='0';
         elsif rising_edge(clk) then                      -- Flanco de subida del reloj
             case state is
                 when "00" =>
                      z<='0';
                      if (x = '0') then
                           state <= "01";
                      end if;
                 when "01" =>
                      z<='0';
                      if (x = '1') then
                           state <= "10";
                      end if;
                 when "10" =>
                     z<='0';
                     if (x = '0') then
                           state <= "01";
                      else
                           state <= "11";
                      end if;
                 when others =>
                      if (x = '0') then
                           state <= "01";
                           z<='1';
                      else
                           state <= "00";
                           z<='0';
                      end if;
             end case;
          end if;
      end process proximo_estado;
end architecture detector;
-------------------------------------------------
```

Código VHDL 1.6: Circuito detector.