INGENIERÍA DE COMPUTADORES 3

Solución al examen de Junio 2025, Segunda Semana

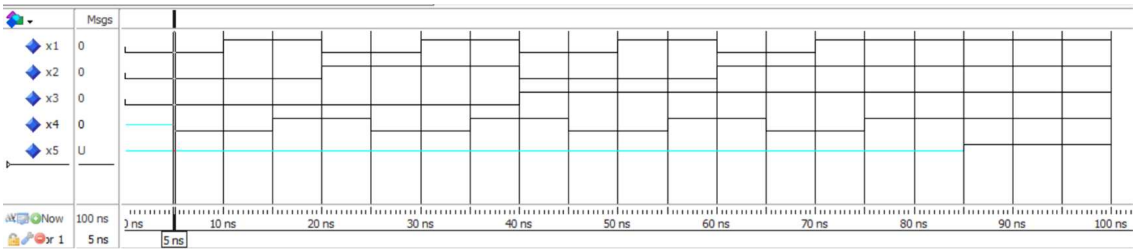
PREGUNTA 1 (2 puntos)

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 100 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;
use IEEE.numeric_std.all;
entity crono2 is
end entity crono2;
architecture crono2 of crono2 is
  signal x1, x2, x3, x4, x5 : std_logic;
begin
  process is
    variable temp : unsigned (2 downto 0);
  begin
    for i in 0 to 7 loop
      temp := TO_UNSIGNED(i,3);
      x1 <= std_logic(temp(0));
      x2 <= std_logic(temp(1));
      x3 <= std_logic(temp(2));
      wait for 10 ns;
    end loop;
    wait;
  end process;
  x4 <= x1 after 5 ns;
  x5 <= x1 after 15 ns;
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)

Diseñe un circuito digital combinacional que tenga como entrada una señal de 3 bits y como salida una señal de un bit que se ponga a `'1'` si la señal de entrada es `"000"`, `"001"`, o `"110"` y se ponga a `'0'` en cualquier otro caso.

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

```
entity circ is
  port(   y: out std_logic;
          x: in std_logic_vector(2 downto 0));
end entity circ;
```

2.a) (0.25 puntos) Escriba en VHDL la **entity** y la **architecture** que describe el comportamiento de una puerta NOT.

2.b) (0.25 puntos) Escriba en VHDL la **entity** y la **architecture** que describe el comportamiento de una puerta AND de 3 entradas.

2.c) (0.25 puntos) Escriba en VHDL la **entity** y la **architecture** que describe el comportamiento de una puerta OR de 3 entradas.

2.d) (0.25 puntos) Dibuje el diagrama a nivel de puertas lógicas del circuito combinacional descrito. Emplee para ello únicamente puertas NOT, puertas AND de 3 entradas y puertas OR de 3 entradas.

2.e) (2 puntos) Escriba en VHDL la **architecture** que describe la estructura del circuito combinacional siguiendo el diagrama dibujado en el apartado anterior

y empleando las puertas lógicas cuyo diseño ha realizado al contestar los tres primeros apartados.

Solución a la Pregunta 2

El Código VHDL 1.1–1.3 es el código de las puertas lógicas NOT, AND y OR de tres entradas.

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

entity not1 is
  port ( y0  : out std_logic;
         x0  : in std_logic );
end entity not1;

architecture not1 of not1 is
begin
  y0 <= not x0;
end architecture not1;
```

Código VHDL 1.1: Puerta NOT.

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

entity and3 is
  port ( y0 : out std_logic;
     x0, x1, x2 : in std_logic );
end entity and3;

architecture and3 of and3 is
begin
  y0 <= x0 and x1 and x2;
end architecture and3;
```

Código VHDL 1.2: Puerta AND de tres entradas.

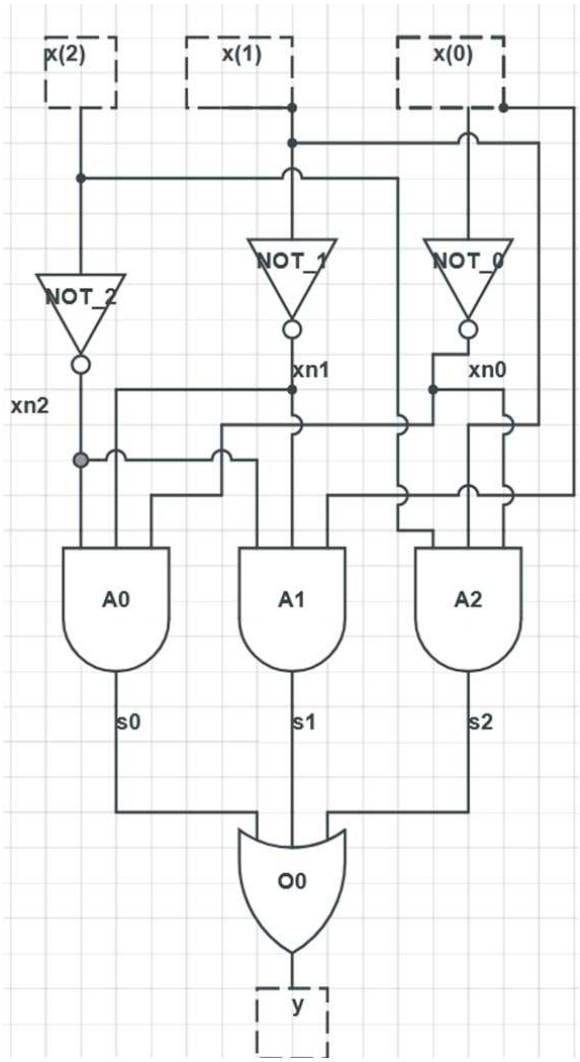
```vhdl
library IEEE;
use IEEE.std_logic_1164.all;

entity or3 is
  port ( y0      : out std_logic;
     x0, x1, x2 : in std_logic );
end entity or3;

architecture or3 of or3 is
begin
  y0 <= x0 or x1 or x2;
end architecture or3;
```

Código VHDL 1.3: Puerta OR de tres entradas.

El diagrama a nivel de puertas lógicas se muestra en la siguiente figura.



El Código VHDL 1.4 es el código del circuito solución de la Pregunta 2.e.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
architecture Estructura OF circ is
  signal s0, s1, s2: std_logic;
  signal xn0, xn1, xn2: std_logic;
-- Declaración de las clases de los componentes

  component not1 is
    port ( y0  : out std_logic;
           x0  : in std_logic );
  end component not1;
  component or3 is
    port ( y0         : out std_logic;
           x0, x1, x2 : in std_logic );
  end component or3;
  component and3 is
    port (y0         : out std_logic;
          x0, x1, x2 : in std_logic );
  end component and3;
begin

  Not_0 : component not1 port map (xn0, x(0));
  Not_1 : component not1 port map (xn1, x(1));
  Not_2 : component not1 port map (xn2, x(2));
  A0 : component and3 port map (s0, xn0, xn1, xn2 );
  A1 : component and3 port map (s1, x(0), xn1, xn2);
  A2 : component and3 port map (s2, xn0, x(1), x(2));
  O0 : component or3 port map (y, s0, s1, s2);

end Estructura;
```

Código VHDL 1.4: Architecture del circuito combinacional solución de la Pregunta 2.e.

PREGUNTA 3 (2 puntos)

Programe en VHDL el banco de pruebas del circuito combinacional que ha diseñado al contestar a la Pregunta 2.e. 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 en caso de que no coincidan.

Solución a la Pregunta 3

El código VHDL del banco de pruebas se muestra en Código VHDL 1.5.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity bp_circ is
end entity bp_circ;
architecture bp_circ of bp_circ is
  signal y : std_logic; -- Conectar salida UUT
  signal x : std_logic_vector(2 downto 0); -- Conectar entrada UUT
  component circ is port
  ( y : out std_logic;
    x : in std_logic_vector(2 downto 0));
  end component circ;
begin
  -- Instanciar y conectar UUT
  uut : component circ port map( y=> y, x => x);
  gen_vec_test : process
    variable test_in : unsigned (2 downto 0); -- Vector de test
    variable y2: std_logic;
  begin
    test_in := B"000";
    for count in 0 to 7 loop
      x(2) <= test_in(2);
      x(1) <= test_in(1);
      x(0) <= test_in(0);
      wait for 10 ns;
      if ( test_in = "000") or
         (test_in = "001") or (test_in = "110" ) then
        y2 := '1';
      else
        y2 := '0';
      end if;
      assert(y2 = y)
      report "ERROR. La salida del circuito no corresponde con la
    salida esperada";
      test_in := test_in + 1;
    end loop;
    wait;
  end process gen_vec_test;
end architecture bp_circ;
```

Código VHDL 1.5: Banco de pruebas del circuito combinacional de la Pregunta 2.e.

PREGUNTA 4 (3 puntos)

Escriba la **architecture** que describe el comportamiento de un circuito para controlar un ascensor de una vivienda de 4 pisos (numerados de 0 a 3) como una máquina de Moore síncrona, con transiciones en el flanco de subida de la señal de reloj. El ascensor tiene 4 botones, uno por cada piso y debe ir al piso indicado por sus 4 botones. Cuando llega al piso deseado abre las puertas, que permanecerán así hasta que reciba otra llamada. El ascensor no tiene memoria, por lo que no le afecta que se pulsen los botones mientras está subiendo/bajando.

Las entradas del circuito controlador son: la señal `piso` de dos bits que indica el piso en que se encuentra el ascensor (0, 1, 2 o 3 en binario), la señal de 4 bits `boton` que indica el piso al que el usuario desea ir, la señal de reloj `clk`, una señal de reset asíncrona (`reset`) y la señal `celula` que indica si existe un objeto en la puerta del ascensor. El circuito tiene dos señales de salida: `motor` y `puerta`.

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

```
entity ctrlAscensor is
   port ( motor  : out std_logic_vector(1 downto 0);
          puerta : out std_logic;
          boton  : in std_logic_vector(3 downto 0);
          piso   : in std_logic_vector(1 downto 0);
          clk, reset, celula  : in  std_logic );
end ctrlAscensor;
```

El significado de las señales de salida y entrada al circuito es el siguiente.

- La señal `motor` tiene dos bits cuyo valor puede ser `"00"`, `"10"` o `"01"` para parar, accionar el motor para que suba o accionarlo para que baje el ascensor, respectivamente. Mientras la señal `motor` tiene el valor `"10"`, el motor sube el ascensor. Cuando la señal `motor` tiene el valor `"01"`, el motor baja el ascensor. Finalmente, el motor está parado si la señal `motor` tiene el valor `"00"`.
- La señal `puerta` está a `'1'` mientras la puerta del ascensor está abierta. Cuando el valor de la señal `puerta` está a `'0'`, la puerta del ascensor está cerrada.
- La señal `boton` tiene 4 bits, uno por cada piso. Sólo puede tener un bit el valor uno, siendo la posición de dicho bit el número del piso al que el usuario

desea ir. Por ejemplo, si `boton` tiene el valor `"1000"`, quiere decir que el usuario del ascensor quiere ir al piso 3. Por otro lado, si la señal `boton` tiene el valor `"0000"`, quiere decir que el usuario del ascensor no ha indicado el piso al que desea ir.

- La señal `piso` tiene 2 bits, indicando su valor en binario el piso en que está el ascensor. Por ejemplo, si `piso` tiene el valor `"10"`, quiere decir que el ascensor se encuentra en el piso 2.
- Señal de reloj `clk`.
- La señal `celula` está a `'1'` mientras la fotocélula de la puerta detecta un objeto. Si no se detecta ningún objeto, esta señal tiene valor `'0'`.
- El circuito tiene una señal de reset (`reset`) asíncrona activa a nivel alto. El reseteo del circuito provoca que el circuito vaya asíncronamente al estado Inicial, que se describe a continuación.

Este circuito puede ser descrito con 4 estados: Inicial, Cerrado, Subiendo, Bajando.

En el estado Inicial la puerta está abierta, el motor está parado y se guarda el piso al que el usuario desea ir. Si el piso al que el usuario quiere ir es distinto al piso en el que estamos, se pasa al estado Cerrado. Por el contrario, si el piso al que el usuario quiere ir es igual que el piso en el que estamos, no hay cambio de estado.

En el estado Cerrado, el motor está parado y la puerta está abierta. Se sale de este estado cuando la fotocélula no detecta objetos en la puerta. En este caso, se pasa al estado Subiendo si el piso al que el usuario quiere ir es mayor que el piso en el que estamos. En caso contrario, se pasa al estado Bajando.

En el estado Bajando, la señal `motor` acciona el motor para que baje y la puerta está cerrada. Cuando llegamos al piso deseado, se pasa al estado Inicial.

En el estado Subiendo, la señal `motor` acciona el motor para que suba y la puerta está cerrada. Cuando llegamos al piso deseado, se pasa al estado Inicial.

Solución a la Pregunta 4

El Código VHDL 1.6 es el diseño del circuito de control del ascensor.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
architecture fsm of ctrlAscensor is
  type state is (Inicial, Cerrado, Subiendo, Bajando);
  signal estado: state;
  function codifica ( pulso : std_logic_vector(3 downto 0) ) return
   std_logic_vector is
  begin
    case pulso is
      when "0001" => return "00";
      when "0010" => return "01";
      when "0100" => return "10";
      when "1000" => return "11";
      when others => return "00";
     end case;
  end codifica;
begin
  process (clk, reset) is
    variable bot: std_logic_vector(1 downto 0);
  begin
    if (reset='1') then
       estado <= Inicial;

    elsif rising_edge(clk) then
       case estado is
        when Inicial =>
         motor <= "00";
         puerta <= '1';
         bot := codifica(boton);
         if (boton/="0000" and bot/= piso) then
            estado <= Cerrado;
         end if;
        when Cerrado =>
         if (celula = '0') then
            if (bot>piso) then estado <= Subiendo;
            else estado <= Bajando;
            end if;
         end if;
        when Subiendo =>
          puerta <= '0';
          motor <= "10";
          if (bot=piso) then
            estado <= Inicial;
          end if;
        when Bajando =>
          puerta <= '0';
```

```
47            motor <= "01";
48            if (bot=piso) then
49              estado <= Inicial;
50            end if;
51          end case;
52       end if;
53    end process;
54 end fsm;
```

Código VHDL 1.6: Circuito de control del ascensor