INGENIERÍA DE COMPUTADORES 3

Solución al examen de Junio 2014, 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` entre los instantes 0 y 80 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono2 is
end entity crono2;
architecture crono2 of crono2 is
   signal x1 : std_logic := '0';
   signal x2, x3, x4, x5 : std_logic;
begin
 process
  begin
    for i in 0 to 4 loop
      x3 <= x1;
      x4 <= x5;
      wait for 15 ns;
    end loop;
    wait;
 end process;
 x1 <= '0', '1' after 20 ns, '0' after 40 ns,
       '1' after 50 ns;
 x5 <= x1 after 15 ns;
 x2 <= x1;
end architecture crono2;
```

Solución a la Pregunta 1

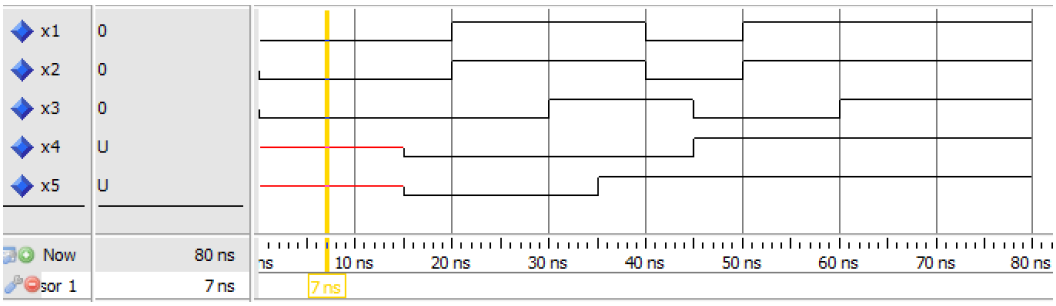


Figura 1.1: Cronograma de evolución de las señales.

PREGUNTA 2 (2.5 puntos)

Escriba en VHDL, de las dos formas que se detallan a continuación, la **architecture** que describe el comportamiento de un circuito codificador 4 a 2 con prioridad. La entrada 3 tiene mayor prioridad que la 2, ésta mayor que la 1 y finalmente la entrada 0 es la de menor prioridad. La **entity** del circuito se muestra a continuación.

```
entity codificador4a2 is
   port ( D  : out std_logic_vector(1 downto 0);
          Z  : out  std_logic;
          A  : in  std_logic_vector(3 downto 0) );
end entity codificador4a2;
```

El circuito tiene una señal de salida Z que ha de valer '0' sólo cuando todos los componentes de la señal A están a '0'.

2.a) (1.25 puntos) Empleando un único bloque **process**.

2.b) (1.25 puntos) Describiendo la estructura del circuito. Siga para ello los siguientes pasos. Primero, dibuje el diagrama circuital del codificador con prioridad empleando únicamente las siguientes puertas lógicas: AND de

dos entradas, OR de 2 entradas y NOT. Segundo, diseñe en VHDL dichas puertas lógicas. Finalmente, escriba la **architecture** del codificador siguiendo el diagrama circuital que ha dibujado previamente y usando las puertas lógicas que ha diseñado en el segundo paso.

Solución a la Pregunta 2

Las dos **architecture** que describen el comportamiento de un circuito combinacional codificador de prioridad se muestran en Código VHDL 1.1–1.2. En Figura 1.2 se muestra el diagrama circuital del codificador de prioridad.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity codificador4a2 is
       port ( D  : out std_logic_vector(1 downto 0);
              Z  : out  std_logic;
              A  : in  std_logic_vector(3 downto 0) );
end entity codificador4a2;
architecture codComport of codificador4a2 is
begin
   process (A) is -- Activado cuando cambia alguna entrada
   begin
     if ( (A(3)='1') or (A(2)='1') or (A(1)='1') or (A(0)='1') ) then
         Z <= '1';   -- Indica que la salida es válida
     else
         Z <= '0';   -- Salida no válida, puesto que no hay entrada
     end if;

     if    (A(3)='1') then  D <= "11";
     elsif (A(2)='1') then  D <= "10";
     elsif (A(1)='1') then  D <= "01";
     else                   D <= "00";
     end if;
   end process;
end architecture codComport;
```

Código VHDL 1.1: Descripción del comportamiento del circuito codificador de prioridad empleando un único bloque **process**.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity codificador4a2 is
       port ( D   : out std_logic_vector(1 downto 0);
              Z   : out  std_logic;
              A   : in  std_logic_vector(3 downto 0) );
end entity codificador4a2;
architecture codEst of codificador4a2 is
  signal notA2, andout, or1out, or3out: std_logic;
-- Declaración de las clases de los componentes
  component and2 is
    port (  y0 : out std_logic ;
         x0, x1 : in std_logic);
  end component and2;
  component not1 is
    port ( y0 : out std_logic;
           x0 : in std_logic );
  end component not1;
  component or2 is
    port ( y0 : out std_logic;
       x0, x1 : in std_logic );
  end component or2;
begin
  inv_1 : component not1 port map (notA2, A(2));
  or_1 : component or2 port map (or1out, A(3), A(2));
  and_1 : component and2 port map (andout, notA2, A(1));
  or_2 : component or2 port map (D(0), A(3), andout);
  or_3 : component or2 port map (or3out, A(1), A(0));
  or_4 : component or2 port map (Z, or3out, or1out);
  D(1) <= or1out;
end architecture codEst;
```

Código VHDL 1.2: Descripción de la estructura del circuito codificador de prioridad.

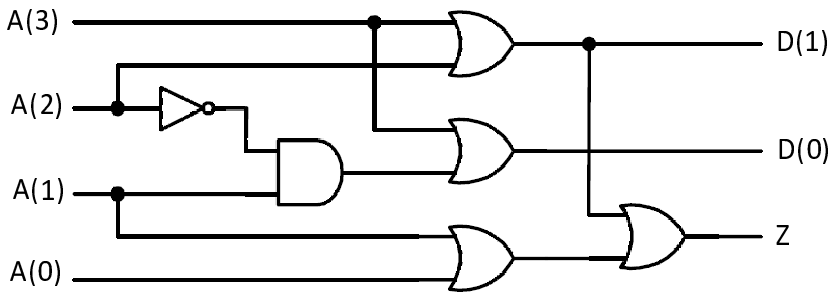
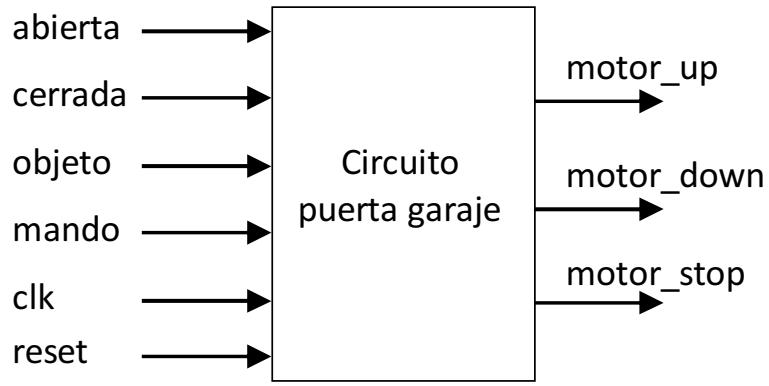


Figura 1.2: Diagrama circuital del circuito codificador de prioridad.

PREGUNTA 3 (3.5 puntos)

Escriba la **architecture** que describe el comportamiento de un circuito para controlar una puerta de un garaje como una máquina de Moore síncrona, con transiciones en el flanco de subida de la señal de reloj.

La puerta del garaje tiene un motor para subir/bajar la puerta, un sensor para indicar que la puerta está completamente abierta, un sensor para indicar que la puerta está completamente cerrada, un mando remoto con un botón para accionar la puerta y una fotocélula para detectar si existe o no un objeto en la misma. El símbolo lógico del circuito y la **entity** del circuito se muestran a continuación.



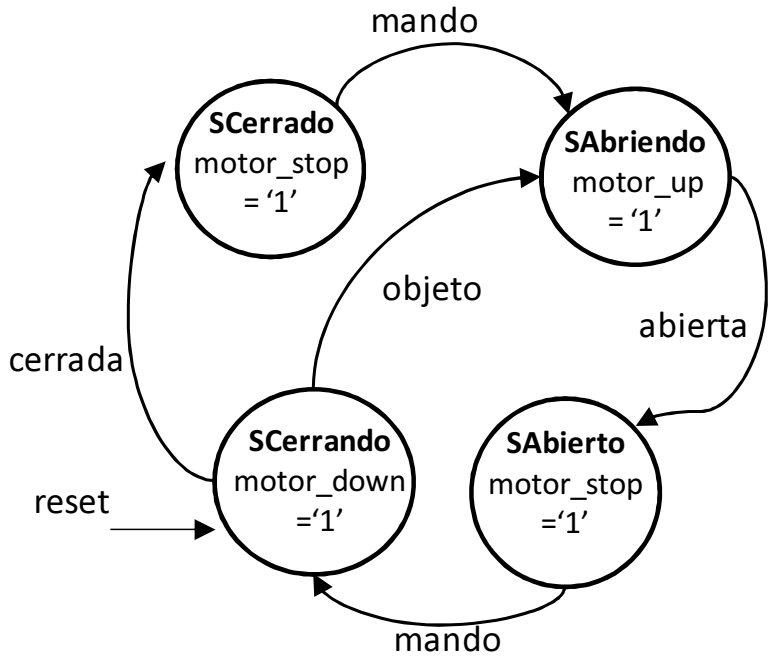

```
entity reguladorPuertaG is
 port (  motor_up, motor_down, motor_stop : out std_logic;
                 abierta, cerrada, objeto: in std_logic;
                      mando, clk, reset  : in  std_logic );
end reguladorPuertaG;
```

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

- Mientras la señal `motor_up` está a '1' el motor sube la puerta del garaje.
- Mientras la señal `motor_down` está a '1' el motor baja la puerta del garaje.
- El motor está parado mientras la señal `motor_stop` está a '1'.
- La señal `abierta` está a '1' mientras la puerta del garaje está completamente abierta.

– La señal `cerrada` está a '1' mientras la puerta del garaje está completamente cerrada.
– La señal `objeto` está a '1' mientras la fotocélula detecta un objeto.
– La señal `mando` está a '1' mientras está pulsado el botón del mando.
– El circuito tiene una señal de reset (`reset`) asíncrona activa a nivel alto.

Este circuito puede ser descrito con el diagrama de estados mostrado en la siguiente figura. Este diagrama tiene 4 estados: SCerrado, SAbriendo, SAbierto y SCerrando. Las transiciones entre estados se producen cuando la señal indicada en el arco de transición toma el valor '1'. En cada estado se indican las únicas señales de salida que tienen el valor '1' en dicho estado.



Solución a la Pregunta 3

El código VHDL del circuito regulador se muestra en Código VHDL 1.3.

```
Library IEEE;
use IEEE.std_logic_1164.all;
entity reguladorPuertaG is
  port (  motor_up, motor_down, motor_stop : out std_logic;
          abierta, cerrada, objeto, mando, clk, reset  : in  std_logic
);
end reguladorPuertaG;
architecture reguladorPuertaG of reguladorPuertaG is
    type type_state is (SCerrado, SAbriendo, SAbierto, SCerrando);
    signal  internal_state: type_state;
begin
proximo_estado: process (clk, reset)
 begin
   if (reset = '1') then
        internal_state <= SCerrando;
   elsif (rising_edge(clk)) then
        case internal_state is
             when SCerrado =>
               if (mando = '1') then
                   internal_state <= SAbriendo;
              end if;
             when SAbriendo =>
               if (abierta = '1') then
                   internal_state <= SAbierto;
              end if;
             when SAbierto =>
               if (mando = '1') then
                    internal_state <= SCerrando;
               end if;
             when SCerrando =>
               if (objeto = '1') then
                    internal_state <= SAbriendo;
               elsif (cerrada = '1') then
                    internal_state <= SCerrado;
               end if;
       end case;
   end if;
end process proximo_estado;
salidas: process(internal_state)
begin
   motor_up <= '0'; motor_down <= '0'; motor_stop <= '0';
   case internal_state is
         when SCerrado =>
              motor_stop <= '1';
         when SAbriendo =>
              motor_up <= '1';
         when SAbierto =>
              motor_stop <= '1';
         when SCerrando =>
              motor_down <= '1';
    end case;
end process salidas;
end architecture reguladorPuertaG;
```

Código VHDL 1.3: Diseño del circuito regulador.

PREGUNTA 4 (2 puntos)

Programe en VHDL un banco de pruebas para el circuito que ha diseñado al contestar a la Pregunta 3. La señal de reloj (`clk`) debe ser de 10 Hz e inicialmente valer '0'. El programa de test debe realizar consecutivamente las acciones siguientes:

1. *Reset*. Resetear el circuito. Todas las señales de entrada al circuito, excepto la señal de reset, han de tener el valor '0'.
2. *En el instante 30 s la fotocélula detecta un objeto* . Darle a la señal `objeto` el valor '1' durante 10 segundos (hasta el instante 40 s). Comprobar que el circuito está generando la señal de salida adecuada para subir la puerta del garaje.
3. *En el instante 90 s la puerta está completamente abierta*. Darle a la señal `abierta` el valor '1'. Comprobar que el circuito está generando la señal de salida adecuada para parar la puerta del garaje.
4. *En el instante 120 s se aprieta el botón del mando*. Darle a la señal `mando` el valor '1' durante 5 segundos (hasta el instante 125 s). Comprobar que el circuito está generando la señal de salida adecuada para bajar la puerta del garaje.
5. *En el instante 180 s la puerta está completamente cerrada*. Darle a la señal `cerrada` el valor '1'. Comprobar que el circuito está generando la señal de salida adecuada para parar la puerta del garaje.

Solución a la Pregunta 4

El código VHDL correspondiente al banco de pruebas del regulador se muestra en Código VHDL 1.4.

```
-- Banco de pruebas del regulador de la puerta del garaje
library IEEE;
use IEEE.std_logic_1164.all;

entity bp_reguladorPG is
end entity bp_reguladorPG;
architecture bp_reguladorPG of bp_reguladorPG is
  signal   up              : std_logic;    -- Salidas UUT
  signal   down              : std_logic;    -- Salidas UUT
  signal   stop              : std_logic;    -- Salidas UUT

  signal   clk           : std_logic  := '0';     -- Entradas UUT
  signal   reset         : std_logic := '1';
  signal   abierta         : std_logic := '0';
  signal   cerrada         : std_logic := '0';
  signal   objeto         : std_logic := '0';
  signal   mando         : std_logic := '0';
  constant PERIODO : time       := 100000 us;  --10 Hz
  component reguladorPuertaG is
   port (  motor_up, motor_down, motor_stop : out std_logic;
          abierta, cerrada, objeto, mando, clk, reset  : in  std_logic
);
   end component reguladorPuertaG;
   -- Procedimiento para comprobar las salidas
begin
  -- Instanciar y conectar UUT
  uut : component reguladorPuertaG port map
                  (up, down, stop, abierta, cerrada, objeto, mando,
clk, reset);

  reset <=   '1',
             '0' after (PERIODO/4);
  clk <= not clk after (PERIODO/2);

  gen_vec_test : process is
    variable error_count : integer := 0; -- Núm. errores
  begin
    report "Comienza la simulación";
     -- Vectores de test y comprobación del resultado
      wait for 30 sec;    -- t = 30 sec
      objeto <= '1';
      wait for 10 sec;    -- t = 40 sec
      objeto <= '0';
      wait for 50 sec;    -- t = 90 sec
      abierta <= '1';
      wait for 30 sec;    -- t = 120 sec
      mando <= '1'; abierta <= '0';
      wait for 5 sec;     -- t = 125 sec
      mando <= '0';
      wait for 55 sec;    -- t = 180
      cerrada <= '1';
    wait;
  end process gen_vec_test;
end architecture bp_reguladorPG;
```

Código VHDL 1.4: Banco de pruebas del regulador.