INGENIERÍA DE COMPUTADORES 3

Solución al examen de Septiembre 2025

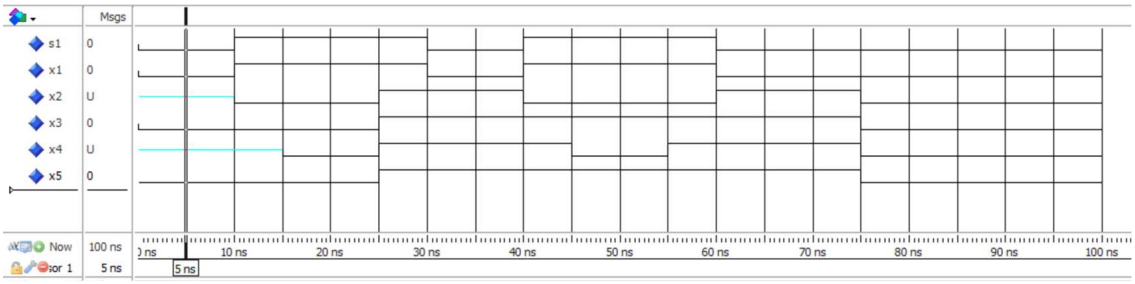
PREGUNTA 1 (2 puntos)

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. Indique el valor inicial de cada una de las señales, así como el instante en que cambian de valor y el nuevo valor.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono1 is
end entity crono1;
architecture crono1 of crono1 is
  signal s1, x1, x2, x3, x4 : std_logic;
  signal x5 : std_logic := '0';
begin
  process (s1, x5)
    variable temp : std_logic;
  begin
    temp := x5;
    x1 <= s1;
    x2 <= x1;
    x3 <= temp;
  end process;
  s1 <= '0', '1' after 10 ns, '0' after 30 ns,
        '1' after 40 ns, '0' after 60 ns;
  x4 <= transport s1 after 15 ns;
  x5 <= s1 after 15 ns;
end architecture crono1;
```

Solución a la Pregunta 1

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



En la siguiente figura se muestra el valor de las señales teniendo en cuenta el retardo delta.

ps	delta	/cronol/s1	/cronol/x1	/cronol/x2	/cronol/x3	/cronol/x4	/cronol/x5
0	+0	U	U	U	U	U	0
0	+1	0	U	U	0	U	0
0	+2	0	0	U	0	U	0
10000	+0	1	0	U	0	U	0
10000	+1	1	1	0	0	U	0
15000	+0	1	1	0	0	0	0
25000	+0	1	1	0	0	1	1
25000	+1	1	1	1	1	1	1
30000	+0	0	1	1	1	1	1
30000	+1	0	0	1	1	1	1
40000	+0	1	0	1	1	1	1
40000	+1	1	1	0	1	1	1
45000	+0	1	1	0	1	0	1
55000	+0	1	1	0	1	1	1
60000	+0	0	1	0	1	1	1
60000	+1	0	0	1	1	1	1
75000	+0	0	0	1	1	0	0
75000	+1	0	0	0	0	0	0

PREGUNTA 2 (3 puntos)

Diseñe un circuito secuencial síncrono que permita controlar el funcionamiento de una máquina expendedora sencilla. El usuario de la máquina puede introducir únicamente monedas de 5 céntimos y 10 céntimos. Cuando se introducen exactamente 15 céntimos, la máquina expendedora saca un chicle. Este sistema de control debe emitir una señal de apertura (señal `abrir`) cuando ha detectado que se han introducido 15 céntimos (una moneda de 5 céntimos y una moneda de 10 céntimos, o tres monedas de 5 céntimos). No hace falta que considere el caso en el cual el importe introducido sea superior a 15 céntimos. El sistema de control recibe dos señales de entrada: `cinco` y `diez`. La señal `cinco` es una señal de entrada del circuito que tiene valor `'1'` durante un tiempo sólo si el usuario introduce una moneda de 5 céntimos. La señal `diez` es una señal de entrada del circuito que tiene valor `'1'` durante un tiempo sólo si el usuario introduce una moneda de 10 céntimos.

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

```
entity maquina is
   port( abrir : out std_logic;
         cinco : in std_logic;
         diez  : in std_logic;
         reset : in std_logic;
         clk   : in std_logic);
end entity maquina;
```

El circuito tiene una señal de reloj (`clk`), dos entradas de un bit (`cinco` y `diez`), una señal de reset asíncrona activa en `'1'` (`reset`) y una señal de salida de un bit (`abrir`).

La señal `reset` es la única señal asíncrona del circuito y pone el circuito en su estado inicial. Este estado inicial indica que no ha recibido ninguna moneda.

La señal `abrir` se pone a `'1'` sólo si la máquina ha recibido exactamente 15 céntimos. Esta señal ha de permanece con el valor `'1'` sólo durante un periodo de la señal de reloj `clk`.

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.

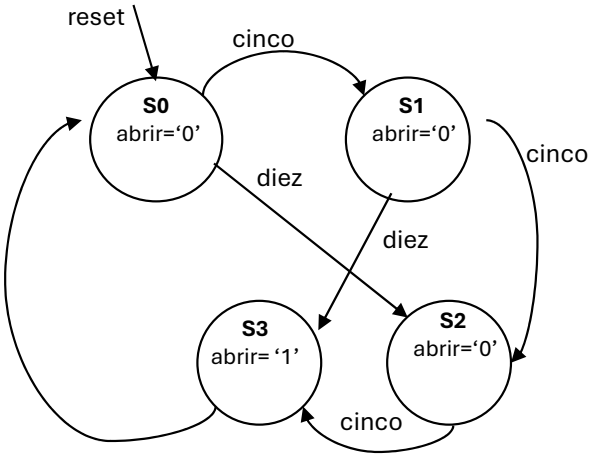
Solución a la Pregunta 2

El Código VHDL 1.1 es una posible solución a la Pregunta 2.

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

entity maquina is
   port( abrir     : out std_logic;
         cinco     : in std_logic;
         diez : in std_logic;
         reset : in std_logic;
         clk   : in std_logic);
end entity maquina;

architecture fsm of maquina is
 signal internal_state: std_logic_vector (1 downto 0);
begin
 --Cálculo del próximo estado
 proximo_estado: process (clk, reset)
 begin
  if (reset = '1') then
    internal_state <= "00";
  elsif (rising_edge(clk)) then
     case internal_state is
      when "00" =>
         if cinco= '1' then
            internal_state <= "01";
         elsif diez = '1' then
            internal_state <= "10";
         end if;
      when "01" =>
         if cinco = '1' then
             internal_state <= "10";
         elsif diez = '1' then
             internal_state <= "11";
          end if;
      when "10" =>
          if cinco = '1' then
             internal_state <="11";
          end if;
      when others=>
         internal_state <= "00";
     end case;
  end if;
 end process proximo_estado;
 salida: process (internal_state)
 begin
  abrir <= '0';
  if internal_state = "11" then
    abrir <= '1';
  end if;
 end process salida;

end architecture fsm;
```

Código VHDL 1.1: Máquina expendedora.

En la siguiente figura se muestra el diagrama de estados del circuito, que tiene cuatro estados (`S0`, `S1`, `S2` y `S3`). El estado `S0` es el inicial, donde aún no se ha recibido ninguna moneda, en el estado `S1` se han recibido 5 céntimos, en el estado `S2` se han recibido 10 céntimos y en el estado `S3` se han recibido los 15 céntimos.



PREGUNTA 3 (2 puntos)

Escriba en VHDL la **architecture** que describe el comportamiento de un circuito combinacional que tiene como señal de salida el número de ceros que contiene la señal de entrada de izquierda a derecha hasta que se encuentra el primer uno. La **entity** del circuito se muestra a continuación.

```
entity numCeros is
   port( Y     : out std_logic_vector(3 downto 0);
         X     : in std_logic_vector(7 downto 0) );
end entity numCeros;
```

El circuito tiene una señal de entrada de ocho bits (`X`) y una señal de salida de cuatro bits (`Y`).

La señal `Y` indica en binario sin signo el número de bits que tiene la entrada `X` con valor `'0'` hasta que se encuentra el primer bit con valor `'1'`, empezando a revisar los bits de la señal `X` de izquierda a derecha. Por ejemplo, cuando la señal `X` tiene el valor `"00100101"`, la señal `Y` tiene el valor `"0010"` que se corresponde con el valor decimal 2.

Solución a la Pregunta 3

El código VHDL 1.2 describe el comportamiento del circuito.

```vhdl
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
architecture numCeros of numCeros is
begin
   process (x)
     variable count: unsigned (3 downto 0);
   begin
     count := "0000";
     for i in 7 downto 0 loop
       case x(i) is
          when '0' => count := count + 1;
          when others => exit;
       end case;
     end loop;
     y <= std_logic_vector(count);
   end process;
end numCeros;
```

Código VHDL 1.2: Circuito detector.

PREGUNTA 4 (3 puntos)

Programe en VHDL un banco de pruebas que compruebe, para todos los posibles valores de la señal de entrada, si el circuito que ha diseñado al contestar a la Pregunta 3 se comporta correctamente.

Explique detalladamente cómo el programa de test comprueba exhaustivamente el valor del UUT para todos los posibles valores de la entrada.

El banco de pruebas debe comprobar que los valores obtenidos del UUT coinciden con los esperados, mostrando el correspondiente mensaje en caso de que no coincidan. Al final del test, debe mostrarse un mensaje indicando el número total de errores.

Solución a la Pregunta 4

El Código VHDL 1.3 es el diseño del banco de pruebas solución de la Pregunta 4.

```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4  entity bp_circ is
5  end entity bp_circ;
6  architecture bp_circ of bp_circ is
7    signal y : std_logic_vector(3 downto 0); -- Conectar salida UUT
8    signal x : std_logic_vector(7 downto 0); -- Conectar entrada UUT
9    component numCeros is port
10   ( y : out std_logic_vector(3 downto 0);
11     x : in std_logic_vector(7 downto 0));
12   end component numCeros;
13 begin
14   -- Instanciar y conectar UUT
15   uut : component numCeros port map( y=> y, x => x);
16   gen_vec_test : process
17     variable test_in : unsigned (7 downto 0); -- Vector de test
18     variable y2: std_logic_vector(3 downto 0);
19     variable num_errores: integer:=0;
20   begin
21     test_in := B"00000000";
22     for count in 0 to 2**8-1 loop
23       x <= std_logic_vector(test_in);
24       wait for 10 ns;
25       if std_match (x,"00000000") then y2 := "1000";
26       elsif std_match(x,"00000001") then y2 := "0111";
27       elsif std_match(x,"0000001-") then y2 := "0110";
28       elsif std_match(x,"000001--") then y2 := "0101";
29       elsif std_match(x,"00001---") then y2 := "0100";
30       elsif std_match(x,"0001----") then y2 := "0011";
31       elsif std_match(x,"001-----") then y2 := "0010";
32       elsif std_match(x,"01------") then y2 := "0001";
33       else y2 := "0000";
34       end if;
35       test_in := test_in + 1;
36       assert(y2 = y)
37       report "ERROR. La salida del circuito no corresponde con la
     salida esperada";
38       if (y2/=y) then num_errores := num_errores+1;
39       end if;
40     end loop;
41     report "Hay " & integer'image(num_errores) & " errores.";
42     wait;
43   end process gen_vec_test;
44 end architecture bp_circ;
```

Código VHDL 1.3: Banco de pruebas.