

INGENIERÍA DE COMPUTADORES 3

Solución al examen de Junio 2026, Segunda Semana

PREGUNTA 1 (2 puntos)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales x2, x3, x4, x5 y x6 entre los instantes 0 y 50 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.

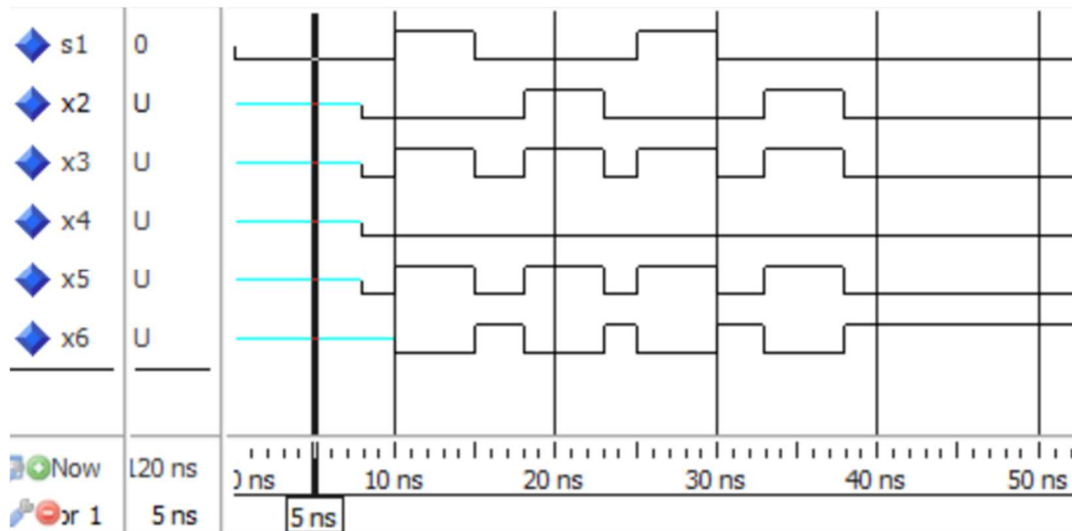
```
library IEEE;
use IEEE.std_logic_1164.all;

entity crono2 is
end entity crono2;

architecture crono2 of crono2 is
    signal s1, x2, x3, x4, x5, x6 : std_logic;
begin
    s1 <= '0', '1' after 10 ns,
          '0' after 15 ns, '1' after 25 ns,
          '0' after 30 ns;
    x2 <= transport s1 after 8 ns;
    Proc1: process(s1, x2)
    begin
        x3 <= s1 xor x2;
        x6 <= x3;
    end process;
    x4 <= s1 after 8 ns;
    x5 <= s1 xor x2;
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)

Escriba la **architecture** que describe el comportamiento de un circuito contador módulo m programable, con reset síncrono activo a nivel alto. Es decir, que cuenta desde 0 hasta el valor $m-1$, incrementando en cada ciclo de reloj su valor en 1. El circuito ha de tener la **entity** siguiente:

```
entity contadorProg is
    port( q : out std_logic_vector (3 downto 0);
          clk, reset : in std_logic ;
          m : in std_logic_vector (3 downto 0) );
end entity contadorProg;
```

Cuando la señal `reset` tiene el valor '1', pone la señal `q` a "0000". La señal `q` se interpreta como un número binario sin signo. Si la señal `reset` tiene el valor '0', en cada flanco de subida de la señal de reloj la señal `q` incrementa su valor en uno, pasando cíclicamente del valor 0 al valor $m-1$. Por ejemplo, si la señal `m` es "0100" la señal `q` toma cíclicamente los valores "0000", "0001", "0010", "0011", "0000", ...

Solución a la Pregunta 2

El código VHDL del contador programable se muestra en Código VHDL 1.1.

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3 use IEEE.numeric_std.all;
4 entity contadorProg is
5     port( q: out std_logic_vector (3 downto 0);
6           clk, reset : in std_logic ;
7           m : in std_logic_vector (3 downto 0));
8 end entity contadorProg;
9 architecture contadorProg of contadorProg is
10     signal r_reg: unsigned (3 downto 0);
11     signal r_next: unsigned (3 downto 0);
12 begin
13     process(clk)
14     begin
15         if rising_edge(clk) then
16             if (reset = '1') then
17                 r_reg <= (others => '0');
18             else
19                 r_reg <= r_next;
20             end if;
21         end if;
22     end process;
23     r_next <= (others=>'0') when r_reg=(unsigned(m)-1) else
24         r_reg + 1;
25     q <= std_logic_vector(r_reg);
26 end contadorProg;

```

Código VHDL 1.1: Architecture del circuito contador programable módulo m.

PREGUNTA 3 (2 puntos)

Programa en VHDL un banco de pruebas para el circuito que ha diseñado al contestar a la Pregunta 3. La señal de reloj (clk) debe tener un periodo de 20 ns e inicialmente valer '0'. El programa de test debe realizar consecutivamente las acciones siguientes:

1. *Reset y m con valor 5.* La señal de reset ha de tener el valor '1' durante 20 ns. La señal m ha de tener el valor "0101".
2. *Empezar la cuenta.* Comprobar que el circuito pasa por los valores "0000", "0001", ..., "0100" y "0000".
3. *Reset y m con valor 15.* La señal de reset ha de tener el valor '1' durante 20 ns. La señal m ha de tener el valor "1111".

4. *Empezar la cuenta.* Comprobar que el circuito pasa por los valores "0000", "0001", ..., "1110" y "0000".

El banco de pruebas debe comprobar que los valores de las señales de salida de la 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 3

El código VHDL del banco de pruebas se muestra en Código VHDL 1.2.

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3 use IEEE.numeric_std.all;
4 entity bp_contadorProg is
5     constant PERIODO : time := 20 ns; -- PERIODO clk
6
7 end entity bp_contadorProg;
8
9 architecture bp_contadorProg of bp_contadorProg is
10     signal q : std_logic_vector(3 downto 0); --Salida UUT
11     signal m : std_logic_vector(3 downto 0); --Entradas UUT
12     signal clk : std_logic := '0';
13     signal reset: std_logic;
14     component contadorProg is
15     port( q: out std_logic_vector (3 downto 0);
16           clk, reset : in std_logic ;
17           m : in std_logic_vector (3 downto 0));
18     end component contadorProg;
19
20
21 begin
22     UUT : component contadorProg port map
23         (q, clk, reset, m);
24     clk <= not clk after (PERIODO/2);
25
26     vec_test : process is
27         variable temp : std_logic_vector (3 downto 0);
28         variable error_count : integer := 0;
29     begin
30         report "Comienza la simulación";
31         m <= "0101";
32         reset <= '1';
33         wait for PERIODO/2;
34         wait until falling_edge(clk);
35         reset <= '0';
36         for i in 0 to 4 loop
37             temp := std_logic_vector(to_unsigned(i,4));

```

```

38     if(temp /= q) then
39         report "ERROR";
40         error_count := error_count +1;
41     end if;
42     wait for PERIODO;
43 end loop;
44 temp := "0000";
45 if(temp /= q) then
46     report "ERROR";
47     error_count := error_count +1;
48 end if;
49 m <= "1111";
50 reset <= '1';
51 wait for PERIODO;
52 reset <= '0';
53 for i in 0 to 14 loop
54     temp := std_logic_vector(to_unsigned(i,4));
55     if(temp /= q) then
56         report "ERROR";
57         error_count := error_count +1;
58     end if;
59     wait for PERIODO;
60 end loop;
61 temp := "0000";
62 if(temp /= q) then
63     report "ERROR";
64     error_count := error_count +1;
65 end if;
66
67 -- Informe mostrando el resultado del test
68 report "Finaliza la simulación: " & integer'image(error_count) & "
errores";
69 wait; -- Termina la simulación
70 end process vec_test;
71 end architecture bp_contadorProg;

```

Código VHDL 1.2: Banco de pruebas del circuito contador.

PREGUNTA 4 (3 puntos)

Escriba el código VHDL de la **architecture** que describe el comportamiento del circuito secuencial descrito a continuación.

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

```

entity circuito3 is
    port ( y      : out std_logic_vector (3 downto 0);
           clk, reset, dir : in  std_logic );
end entity circuito3;

```

Las entradas al circuito son la señal de reloj (`clk`), la señal de reset asíncrono activo a nivel alto (`reset`) y la señal `dir`. La salida del circuito es la señal de 4 bits `y`.

El funcionamiento del circuito debe ser el siguiente. Cuando la señal `reset` pasa a valer '1', la señal de salida `y` toma el valor "0000". Cuando la señal de `reset` tiene valor '0' y la señal `dir` tiene valor '1', la señal de salida del circuito (`y`) toma cíclicamente los valores "0000", "0011", "0110", "0101" y "0111". Cuando la señal de `reset` tiene valor '0' y la señal `dir` tiene valor '0', la señal de salida del circuito (`y`) toma cíclicamente los valores "0000", "0110", "0111", "0101" y "0011". Todos los cambios, salvo el reset, tienen lugar en el flanco de subida de la señal de reloj.

El diseño debe realizarse describiendo el comportamiento del circuito en términos de una máquina de Moore.

Solución a la Pregunta 4

El Código VHDL 1.3 es el diseño del circuito secuencial.

```

1 library IEEE;
2 use IEEE.std_logic_1164.all;
3 architecture circuito3 of circuito3 is
4     signal state : std_logic_vector(2 downto 0);
5 begin
6     -- Calculo del proximo estado
7     proximo_estado: process (clk, reset) is
8     begin
9         if ( (reset = '1') ) then -- Reset asincrono
10             state <= "000";
11         elsif rising_edge(clk) then
12             case state is
13                 when "000" =>
14                     if (dir='1') then
15                         state <= "001"; --0011
16                     else
17                         state <= "010"; --0110
18                     end if;
19                 when "001" =>
20                     if (dir = '1') then
21                         state <= "010"; --0110
22                     else
23                         state <= "000"; --0000
24                     end if;
25                 when "010" =>
26                     if (dir = '1') then
27                         state <= "011"; --0101

```

```

28         else
29             state <= "100"; --0111
30         end if;
31     when "011" =>
32         if (dir = '1') then
33             state <= "100"; --0111
34         else
35             state <= "001"; --0011
36         end if;
37     when "100" =>
38         if (dir = '1') then
39             state <= "000"; --0000
40         else
41             state <= "011";--0101
42         end if;
43     when others =>
44         state <= "000";
45     end case;
46 end if;
47 end process proximo_estado;
48 Y <= "0000" when (state="000")
49     else "0011" when (state = "001")
50     else "0110" when (state = "010")
51     else "0101" when (state = "011")
52     else "0111" when (state = "100")
53     else "0000";
54
55 end architecture circuito3;

```

Código VHDL 1.3: Circuito solución de la Pregunta 4