INGENIERÍA DE COMPUTADORES III

Examen de convocatoria Junio 2021, en el Aula virtual de Examen UNED (AvEx)

INSTRUCCIONES

- El examen debe realizarse de manera individual.
- No está permitido el uso de ningún material.
- El examen se compone de tres preguntas de desarrollo:

  1 pregunta del bloque 1, 1 pregunta del bloque 2 y 1 pregunta del bloque 3.

  La puntuación de la pregunta del bloques 1 es de 3 puntos.

  La puntuación de la pregunta del bloque 2 es de 3 puntos.

  La puntuación de la pregunta del bloque 3 es de 4 puntos.

  La aplicación mostrará aleatoriamente una pregunta de cada uno de los bloques, de modo que el examen a desarrollar tiene 3 preguntas.
- Para aprobar el examen debe obtener una puntuación igual a superior a 5 puntos.
- Dispone de 1 hora para realizar el examen.

Preguntas Bloque 1 (3 puntos)

Pregunta 1.1

Dado el siguiente fragmento de código VHDL, indique cuál es el valor de las señales `p1`, `p2`, `s1`, `s2`, `s3`, `s4` en los instantes 0, delta (δ), 2 delta (2δ), 3 delta (3δ), 5 ns y 5ns+ delta ($5ns + \delta$).

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono is
end entity crono;
architecture crono of crono is
  signal p1, p2, s1, s2, s3, s4: std_logic;
begin
  proc1: process
  begin
    p1 <= '1';
    p2 <= '0';
    wait;
  end process;
  s1<= p1 or p2;
  s2 <= s1 after 5 ns;
  s3 <= s2 xor s4;
  s4<= not s1;
end architecture crono;
```

Solución

- Señal `p1`: 'U' en 0ns y '1' a partir de delta.
- Señal `p2`: 'U' en 0ns y '0' a partir de delta.
- Señal `s1`: 'U' en 0ns y '1' a partir de 2delta.
- Señal `s2`: 'U' en 0ns y '1' a partir de 5 ns.
- Señal `s3` 'U' en 0ns y '1' a partir de 5 ns + delta.
- Señal `s4` 'U' en 0ns y '0' a partir de 3 delta.

Pregunta 1.2

Dado el siguiente fragmento de código VHDL indique cuál es el valor de las señales a, b, c, d en los instantes 0ns, 10ns, 10ns + delta (10ns+ δ), 15 ns, 15 ns+delta (15ns+ δ), 20 ns y 20 ns+delta (20ns+ δ).

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono is
end entity crono;
architecture crono of crono is
  signal p1, p2, p3, a, b, c, d : std_logic;
begin
  p1 <= '0',
        '1' after 10 ns,
        '0' after 15 ns;
  p2 <= '1',
        '0' after 20 ns;
  p3 <= '1',
        '0' after 10 ns,
        '1' after 20 ns;
  a <= p1 after 5 ns;
  Proc1: process (p1, p2)
  begin
    b <= p1 or a;
    c <= b xor p3;
    d <= a and p2;
  end process;
end architecture crono;
```

Solución

- Señal a: 'U' en 0ns, '0' en 5ns, '1' en 15ns y '0' en 20 ns.
- Señal b: 'U' en 0ns, '1' en 10ns+delta y '0' a partir de 20ns+delta.
- Señal c: 'U' en 0ns, '1' en 15ns+delta, y '0' a partir de 20ns+delta.
- Señal d: 'U' en 0ns, '0' en 10ns+delta, '1' en 15ns+delta y '0' a partir de 20ns+delta.

Pregunta 1.3

Dado el siguiente fragmento de código VHDL, indique cuál es el valor de las señales `a`, `b`, `c`, `d` en los instantes 0, delta (δ), 2 delta (2δ), 5ns, 5ns+ delta ($5ns + \delta$), 10ns, 15ns, 20ns, 25ns y 25ns+ delta ($25ns + \delta$).

```
architecture crono of crono is
  signal p1, p2, p3, a, b, c, d : std_logic;
begin
  p1 <= '0',
        '1' after 5 ns,
        '0' after 15 ns;
  p2 <= '1',
        '0' after 10 ns;
  p3 <= '1',
        '0' after 15 ns,
        '1' after 25 ns;
  a <= p1 after 5 ns;
  Proc1: process (p1, p3)
  begin
    b <= p1 xor a;
    c <= b nor p3;
    d <= a and p2;
  end process;
end architecture crono;
```

Solución

- Señal `a`: 'U' en 0ns, '0' en 5ns, '1' en 10ns y '0' a partir de 20ns.
- Señal `b`: 'U' en 0ns, '1' en 5ns+delta y '0' a partir de 25ns+delta.
- Señal `c`: 'U' en 0ns y '0' a partir 2delta.
- Señal `d`: 'U' en 0ns y '0' a partir 5ns+delta.

Pregunta 1.4

Dado el siguiente fragmento de código VHDL, indique cuál es el valor de las señales `p1`, `p2`, `s1`, `s2`, `s3`, `s4` en los instantes 0, delta (δ), 2 delta (2δ), 3 delta (3δ), 5 ns y 5ns+ delta ($5ns + \delta$).

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono is
end entity crono;
architecture crono of crono is
  signal p1 : std_logic:= '1';
  signal p2, s1, s2, s3, s4: std_logic;
begin
  proc1: process
  begin
    p1 <= '0';
    p2 <= '1';
    wait;
  end process;
  s1<= p1 xor p2;
  s2 <= s1 after 7 ns;
  s3 <= s1 xor s4;
  s4<= not s2;
end architecture crono;
```

Solución

- Señal `p1`: '1' en 0ns y '0' a partir de delta.
- Señal `p2`: 'U' en 0ns y '1' a partir de delta.
- Señal `s1`: 'U' en 0ns y '1' a partir de 2delta.
- Señal `s2`: 'U' en 0ns y '1' a partir de 7 ns.
- Señal `s3` 'U' en 0ns y '1' a partir de 7 ns + 2delta.
- Señal `s4` 'U' en 0ns y '0' a partir de 7ns+delta.

Pregunta 1.5

Dado el siguiente fragmento de código VHDL, indique cuál es el valor de las señales a, b, c y d en los instantes 0, 10ns, 10ns+delta ($10ns + \delta$), 15 ns, 20ns, 20ns+delta ($20ns + \delta$), 30 ns y 35 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono is
end entity crono;
architecture crono of crono is
  signal s1, s2, a, b, c, d : std_logic;
begin
  s1 <= '0',
        '1' after 10 ns,
        '0' after 15 ns,
        '1' after 20 ns;
  s2 <= '0',
        '1' after 5 ns,
        '0' after 25 ns;
  a <= s1 after 10 ns;
  b <= s2 after 10 ns;
  Proc1: process
  begin
    for i in 0 to 2 loop
       c <= s1 xor s2;
       d <= c;
       wait for 10 ns;
     end loop;
     wait;
  end process;
end architecture crono;
```

Solución

- Señal a: 'U' en 0ns, '0' en 10ns y '1' a partir de 30ns.
- Señal b: 'U' en 0ns, '1' en 15ns y '0' a partir de 35ns.
- Señal c: 'U' en 0ns y '0' a partir de 10ns+ delta.
- Señal d: 'U' en 0ns y '0' a partir de 20 ns + delta.

Pregunta 1.6

Dado el siguiente fragmento de código VHDL, indique cuál es el valor de las señales a, b, c y d en los instantes 0, 5ns, 5ns+delta ($5ns + \delta$), 10ns, 10ns+delta ($10ns + \delta$), 20ns y 40 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono is
end entity crono;
architecture crono of crono is
  signal s1, s2, a, b, c, d : std_logic;
begin
  s1 <= '1',
        '0' after 10 ns,
        '1' after 30 ns;
  s2 <= '0',
        '1' after 10 ns,
        '0' after 15 ns;
  a <= s1 after 10 ns;
  b <= s2 after 10 ns;
  Proc1: process
  begin
    for i in 0 to 2 loop
       c <= s1 and s2;
       d <= c;
       wait for 5 ns;
     end loop;
     wait;
  end process;
end architecture crono;
```

Solución

- Señal a: 'U' en 0ns, '1' en 10ns, '0' en 20ns, y '1' a partir de 40ns.
- Señal b: 'U' en 0ns y '0' a partir de 10ns.
- Señal c: 'U' en 0ns y '0' a partir de 5ns+delta.
- Señal d: 'U' en 0ns y '0' a partir de 10ns+delta.

Pregunta 1.7

Dado el siguiente fragmento de código VHDL, indique cuál es el valor de las señales a, b, c y d en los instantes 0, delta (δ), 2delta (2δ),5ns, 5ns+delta ($5ns+\delta$),10 ns, 15ns, 15ns+delta ($15ns+\delta$), 20 ns, 20 ns + delta ($20ns+\delta$) y 30 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono is
end entity crono;
architecture crono of crono is
  signal x1, x2, x3, a, b, d : std_logic;
  signal c  : std_logic := '0';
begin
  x1 <= '1',
        '0' after 5 ns,
        '1' after 15 ns;
  x2 <= '0',
        '1' after 10 ns;
  x3 <= '0',
        '1' after 20 ns,
        '0' after 30 ns;
  a <= x1 after 5 ns;
  d <= b after 5 ns;
  Proc1: process (x1, x2, x3)
  begin
    b <= x1 and x2;
    c <= b or x3;
  end process;
end architecture crono;
```

Solución

- Señal a: 'U' en 0ns, '1' en 5ns, '0' en 10ns, y '1' a partir de 20ns.
- Señal b: 'U' en 0ns, '0' en 2delta y '1' a partir de 15 ns + delta.
- Señal c: '0' en 0ns, 'U' en delta, '0' en 5ns+ delta y '1' a partir de 20 ns + delta.
- Señal d: 'U' en 0ns, '0' en 5ns y '1' a partir de 20 ns.

Pregunta 1.8

Dado el siguiente fragmento de código VHDL, indique cuál es el valor de las señales a, b, c y d en los instantes 0, delta (δ), 5ns, 10ns, 10ns+delta ($10ns + \delta$), 30ns, 30ns+delta ($30ns + \delta$), 40ns, 50ns y 50ns+delta ($50ns + \delta$).

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono is
end entity crono;
architecture crono of crono is
  signal a, b, c, d : std_logic;
  signal clk : std_logic:='0';
begin
   process (clk)
    variable v1, v2, v3: std_logic;
   begin
      if ( rising_edge(clk) ) then
         v1 := a;
         v2 := v1;
         v3 := v2;
         a <= c;
         b <= a;
      end if;
   end process;
  clk <= not clk after 10 ns;
  process is
  begin
    c<='0'; wait until falling_edge(clk);
    c<='1'; wait until falling_edge(clk);
    c<='1'; wait until falling_edge(clk);
    c<='0'; wait until falling_edge(clk);
    wait;
  end process;
  d <= c after 5 ns;
end architecture crono;
```

Solución

- Señal a: 'U' en 0ns, '0' en 10ns+delta, '0' en 30ns, '1' en 30ns+delta y '0' a partir de 70ns+delta.
- Señal b: 'U' en 0ns, '0' en 30ns+delta y '1' en 50ns+ delta.
- Señal c: 'U' en 0ns, '0' en delta, '1' en 20 ns+delta y '0' a partir de 60ns+delta.
- Señal d: 'U' en 0ns, '0' en 5ns, '1' en 25 y '0' a partir de 65ns.

Bloque 2 (3 puntos)

Pregunta 2.1

Escriba en VHDL la **architecture** de un circuito combinacional que tiene dos señales de entrada y una señal de salida. Las señales de entrada son: la señal de 1 bit `En` y la señal de 4 bits `xnum`. La señal de salida tiene 1 bit y se llama `y`.

Si la señal de entrada `En` tiene valor '1', entonces la señal de salida `y` toma siempre el valor '0'. Si la señal de entrada `En` tiene valor '0', entonces la señal de salida `y` toma el valor '1' si y solo si la señal de entrada `xnum` tiene el valor decimal 0, 4, 8, 9, 12 o 13. En cualquier otro caso, la señal de salida `y` tiene valor '0'. La señal `xnum(3:0)` representa un número binario sin signo siendo el bit 3 el más significativo.

La **entity** de este circuito se muestra a continuación.

```
entity Circuito2 is
   port(     y: out std_logic;
             En: in std_logic;
             xnum: in std_logic_vector ( 3 downto 0));
end entity Circuito2;
```

Emplee en el diseño un bloque **process** con una sentencia **case**.

Solución

```
---------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
architecture ejer2a of circuito2 is
begin
   process(En,xnum)
        variable vnum : std_logic_vector (4 downto 0);
     begin
       vnum := En&xnum;
       case vnum is
           when   "00000" | "00100" | "01100" | "01000"
           | "01101" | "01001" => y<='1';
           when others => y<='0';
       end case;
   end process;
end architecture ejer2a;
---------------------------------
```

Pregunta 2.2

Escriba en VHDL la **architecture** de un circuito combinacional que tiene dos señales de entrada y una señal de salida. Las señales de entrada son: la señal de 1 bit `En` y la señal de 4 bits `xnum`. La señal de salida tiene 1 bit y se llama `y`.

Si la señal de entrada `En` tiene valor '1', entonces la señal de salida `y` toma siempre el valor '0'. Si la señal de entrada `En` tiene valor '0', entonces la señal de salida `y` toma el valor '1' si y solo si la señal de entrada `xnum` tiene el valor decimal 2, 6, 7, 10, 11, 14 o 15. En cualquier otro caso, la señal de salida `y` tiene valor '0'. La señal `xnum(3:0)` representa un número binario sin signo siendo el bit 3 el más significativo.

La **entity** de este circuito se muestra a continuación.

```
entity Circuito2 is
   port(     y: out std_logic;
            En: in std_logic;
            xnum: in std_logic_vector ( 3 downto 0));
end entity Circuito2;
```

Emplee en el diseño un bloque **process** con una sentencia **case**.

Solución

```
---------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
architecture ejer2a of circuito2 is
begin
   process(En,xnum)
        variable vnum : std_logic_vector (4 downto 0);
     begin
       vnum := En&xnum;
       case vnum is
           when  "00010" | "00110" | "00111" | "01010"
           | "01110" | "01111" => y<='1';
           when others => y<='0';
       end case;
   end process;
end architecture ejer2a;
---------------------------------
```

Pregunta 2.3

Escriba en VHDL la **architecture** de un circuito combinacional que tiene dos señales de entrada y una señal de salida. Las señales de entrada son: la señal de 1 bit `En` y la señal de 4 bits `xnum`. La señal de salida tiene 1 bit y se llama `y`.

Si la señal de entrada `En` tiene valor '1', entonces la señal de salida `y` toma siempre el valor '0'. Si la señal de entrada `En` tiene valor '0', entonces la señal de salida `y` toma el valor '1' si y solo si la señal de entrada `xnum` tiene el valor decimal 0, 4, 8, 9, 12 o 13. En cualquier otro caso, la señal de salida `y` tiene valor '0'. La señal `xnum(3:0)` representa un número binario sin signo siendo el bit 3 el más significativo.

La **entity** de este circuito se muestra a continuación.

```
entity Circuito2 is
   port(     y: out std_logic;
            En: in std_logic;
            xnum: in std_logic_vector ( 3 downto 0));
end entity Circuito2;
```

Realice el diseño estructural del circuito empleando para ello el siguiente paquete con las puertas lógicas not, and de 2 entradas y or de dos entradas.

```
-----------------------------------
-- package de componentes
-- Puertas lógicas
library IEEE;
use IEEE.std_logic_1164.all;
package puertasLogicas_package is
  component or2 is
    port ( y0         : out std_logic;
           x0, x1     : in  std_logic );
  end component or2;
  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;
end package puertasLogicas_package;
```

Solución

```
---------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
use work.puertasLogicas_package.all;
architecture ejer2a of circuito2 is
   signal sa0, sa1, so1, xn1, xn0, sE: std_logic;
begin -- (xnum1)'(xnum0)'+xnum(3)(xnum(1))'
   -- Instanciación y conexión de los componentes
   NOT1_1 : component not1 port map
            (xn1,      xnum(1));
   NOT1_2 : component not1 port map
            (xn0, xnum(0));
   AND2_1 : component and2 port map
            (sa0, xnum(3), xn1);
   AND2_2 : component and2 port map
            (sa1, xn1, xn0);
   OR2_1  : component or2  port map
            (so1, sa0, sa1);
   NOT1_3 : component not1 port map
            (sE,      En);
   AND2_3 : component and2 port map
            (y, so1, sE);
end architecture ejer2a;
---------------------------------
```

Pregunta 2.4

Escriba en VHDL la **architecture** de un circuito combinacional que tiene dos señales de entrada y una señal de salida. Las señales de entrada son: la señal de 1 bit `En` y la señal de 4 bits `xnum`. La señal de salida tiene 1 bit y se llama `y`.

Si la señal de entrada `En` tiene valor '1', entonces la señal de salida `y` toma siempre el valor '0'. Si la señal de entrada `En` tiene valor '0', entonces la señal de salida `y` toma el valor '1' si y solo si la señal de entrada `xnum` tiene el valor decimal 2, 6, 7, 10, 11, 14 o 15. En cualquier otro caso, la señal de salida `y` tiene el valor '0'. La señal `xnum(3:0)` representa un número binario sin signo siendo el bit 3 el más significativo.

La **entity** de este circuito se muestra a continuación.

```
entity Circuito2 is
   port(     y: out std_logic;
            En: in std_logic;
            xnum: in std_logic_vector ( 3 downto 0));
end entity Circuito2;
```

Realice el diseño estructural del circuito empleando para ello el siguiente paquete con las puertas lógicas not, and de 2 entradas y or de dos entradas.

```
-----------------------------------
-- package de componentes
-- Puertas lógicas
library IEEE;
use IEEE.std_logic_1164.all;
package puertasLogicas_package is
  component or2 is
    port ( y0         : out std_logic;
           x0, x1     : in  std_logic );
  end component or2;
  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;
end package puertasLogicas_package;
```

Solución

```
---------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
use work.puertasLogicas_package.all;
architecture ejer2a of circuito2 is
   signal sa0, sa1, so1, xn0, sE: std_logic;
begin-- xnum1(xnum0)'+xnum(2)xnum(1)
   -- Instanciación y conexión de los componentes
   NOT1_1 : component not1 port map
            (xn0,      xnum(0));
   AND2_1 : component and2 port map
            (sa0, xnum(1), xn0);
   AND2_2 : component and2 port map
            (sa1, xnum(2), xnum(1));
   OR2_1  : component or2  port map
            (so1, sa0, sa1);
   NOT1_3 : component not1 port map
            (sE,      En);
   AND2_3 : component and2 port map
            (y, so1, sE);
end architecture ejer2a;
---------------------------------
```

Pregunta 2.5

Escriba en VHDL la **architecture** que describe el comportamiento de un circuito combinacional rotador a la izquierda/derecha. El circuito ha de rotar la señal de entrada de 0 a 3 bits. En la descripción del circuito no se pueden emplear ni operadores ni funciones de desplazamiento lógico.

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

```
entity rotador is
   port ( y  : out std_logic_vector(7 downto 0);
          dir: in std_logic;
          a  : in  std_logic_vector(7 downto 0);
          ctrl  : in  std_logic_vector(1 downto 0) );
end entity rotador;
```

La señal de entrada `a` se rota un determinado número de bits. El valor de la señal `ctrl` indica el número de bits a rotar. Cuando la señal `dir` tiene valor '1' la rotación se produce a la derecha, y cuando tiene valor '0' a la izquierda.

Emplee en el diseño un bloque **process** con una sentencia **if**.

Solución

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

architecture rotador of rotador is
begin
process(a, dir, ctrl)
        variable v1 : std_logic_vector (2 downto 0);
     begin
     v1 := dir&ctrl;
      if   (v1 = "001") then --1 bit izqda
         y <= a(6 downto 0) & a(7);
      elsif (v1 = "010") then --2 bits izqda
         y <= a(5 downto 0)&a(7)&a(6);
      elsif (v1 = "011") then --3 bits izqda
         y <= a(4 downto 0)&a(7 downto 5);
      elsif (v1 = "101") then --1 bit derecha
         y <= a(0)&a(7 downto 1);
      elsif (v1 = "110") then -- 2 bit derecha
         y <= a(1)&a(0)&a(7 downto 2);
       elsif (v1 = "111") then --3 bits derecha
         y <= a(2 downto 0) & a(7 downto 3);
      else -- "000" or "100" no rota
         y <= a;
       end if;
end process;
end architecture rotador;
```

Pregunta 2.6

Escriba en VHDL la **architecture** que describe el comportamiento de un circuito combinacional rotador a la izquierda/derecha. El circuito ha de rotar la señal de entrada de 0 a 3 bits. En la descripción del circuito no se pueden emplear ni operadores ni funciones de desplazamiento lógico.

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

```
entity rotador is
   port ( y  : out std_logic_vector(7 downto 0);
          dir: in std_logic;
          a  : in  std_logic_vector(7 downto 0);
          ctrl  : in  std_logic_vector(1 downto 0) );
end entity rotador;
```

La señal de entrada `a` se rota un determinado número de bits. El valor de la señal `ctrl` indica el número de bits a rotar. Cuando la señal `dir` tiene valor '1' la rotación se produce a la derecha, y cuando tiene valor '0' a la izquierda.

Emplee en el diseño un bloque **process** con una sentencia **case**.

Solución

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

architecture rotador of rotador is
begin
process(a, dir, ctrl)
      variable v1 : std_logic_vector (2 downto 0);
    begin
    v1 := dir&ctrl;
     case v1 is
         when  "001" => y <= a(6 downto 0) & a(7);
         when "010"=> y<=  a(5 downto 0)&a(7)&a(6);
         when  "011" => y <= a(4 downto 0)&a(7 downto 5);
         when "101" => y<=  a(0)&a(7 downto 1);
         when  "110" => y <= a(1)&a(0)&a(7 downto 2);
         when "111" => y<=  a(2 downto 0) & a(7 downto 3);
         when others => y<=a;
     end case;
end process;
end architecture rotador;
```

Pregunta 2.7

Escriba en VHDL la **architecture** que describe el comportamiento de un circuito combinacional rotador a la izquierda/derecha. El circuito ha de rotar la señal de entrada de 0 a 3 bits. En la descripción del circuito no se pueden emplear ni operadores ni funciones de desplazamiento lógico.

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

```
entity rotador is
   port ( y  : out std_logic_vector(7 downto 0);
          dir: in std_logic;
          a  : in  std_logic_vector(7 downto 0);
          ctrl  : in  std_logic_vector(1 downto 0) );
end entity rotador;
```

La señal de entrada `a` se rota un determinado número de bits. El valor de la señal `ctrl` indica el número de bits a rotar. Cuando la señal `dir` tiene valor '1' la rotación se produce a la derecha, y cuando tiene valor '0' a la izquierda.

Emplee en el diseño una sentencia concurrente **when**–**else**.

Solución

```
library IEEE;
use IEEE.std_logic_1164.all;
architecture rotador of rotador is
   signal v1: std_logic_vector(2 downto 0);
begin
   v1 <= dir&ctrl;
   y <= a(6 downto 0) & a(7) when (v1 = "001")
        else   a(5 downto 0)&a(7)&a(6) when (v1 = "010")
        else a(4 downto 0)&a(7 downto 5) when (v1 = "011")
        else a(0)&a(7 downto 1) when (v1 = "101")
        else a(1)&a(0)&a(7 downto 2) when (v1 = "110")
        else a(2 downto 0) & a(7 downto 3) when (v1 = "111")
        else a;
end architecture rotador;
```

Pregunta 2.8

Escriba en VHDL la **architecture** que describe el comportamiento de un circuito combinacional rotador a la izquierda/derecha. El circuito ha de rotar la señal de entrada de 0 a 3 bits. En la descripción del circuito no se pueden emplear ni operadores ni funciones de desplazamiento lógico.

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

```
entity rotador is
   port ( y  : out std_logic_vector(7 downto 0);
          dir: in std_logic;
          a  : in  std_logic_vector(7 downto 0);
          ctrl  : in  std_logic_vector(1 downto 0) );
end entity rotador;
```

La señal de entrada `a` se rota un determinado número de bits. El valor de la señal `ctrl` indica el número de bits a rotar. Cuando la señal `dir` tiene valor '1' la rotación se produce a la derecha, y cuando tiene valor '0' a la izquierda.

Emplee en el diseño una sentencia concurrente **with–select**.

Solución

```
library IEEE;
use IEEE.std_logic_1164.all;
architecture rotador of rotador is
  signal v1: std_logic_vector(2 downto 0);

begin
v1 <= dir&ctrl;

with v1 select
y <= a(6 downto 0) & a(7) when "001",
       a(5 downto 0)&a(7)&a(6) when "010",
       a(4 downto 0)&a(7 downto 5) when "011",
         a(0)&a(7 downto 1) when    "101",
       a(1)&a(0)&a(7 downto 2) when "110",
       a(2 downto 0) & a(7 downto 3) when "111",
       a when others;
end architecture rotador;
```

Bloque 3 (4 puntos)

Pregunta 3.1

Escriba en VHDL la architecture de un circuito secuencial capaz de detectar cuando le llega la secuencia "00110" por su entrada X. El circuito detecta secuencias no solapadas.

El circuito tiene una señal de reloj (`clk`), una entrada serie de un bit (X), una señal de reset asíncrona activa en 0 (`reset`) y una señal de salida de un bit (Y). La señal Y vale '1' cuando se detecta la secuencia. En cualquier otro caso, la señal Y tiene valor '0'. La señal `reset` pone el circuito en su estado inicial (no se ha reconocido ningún elemento de la secuencia). Todos los cambios, salvo el reseteo del circuito, tienen lugar en el flanco de subida de la señal de reloj.

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

```
entity detector is
   port ( Y       : out std_logic;
          reset   : 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 Moore.

Solución

```
library IEEE;
use IEEE.std_logic_1164.all;
architecture detector of detector is
signal state : std_logic_vector(2 downto 0);
begin
-- Calculo del proximo estado
proximo_estado: process (clk, reset) is
begin
if ( (reset = '0') ) then -- Reset asincrono
   state <= "000";
elsif rising_edge(clk) then
    case state is
        when "000" =>
          if (x = '0') then
              state <= "001";
          end if;
        when "001" =>
           if (x = '1') then
              state <= "000";
           else
              state <= "010";
            end if;
         when "010" =>
          if (x = '1') then
              state <= "011";
          end if;
         when "011" =>
          if (x = '0') then
              state <= "001";
          else
              state <= "100";
          end if;
        when "100" =>
           if (x = '1') then
              state <= "000";
           else
              state <= "101";
            end if;
         when "101" =>
          if (x = '1') then
              state <= "000";
          else
              state <= "001";
          end if;
         when others =>
          state <= "000";
         end case;
      end if;
end process proximo_estado;
Y <= '1' when (state="101") else
'0';
end architecture detector;
-----------------------------------------------
```

Pregunta 3.2

Escriba en VHDL la architecture de un circuito secuencial capaz de detectar cuando le llega la secuencia "00110" por su entrada X. El circuito detecta secuencias no solapadas.

El circuito tiene una señal de reloj (clk), una entrada serie de un bit (X), una señal de reset asíncrona activa en 0 (reset) y una señal de salida de un bit (Y). La señal Y vale '1' cuando se detecta la secuencia. En cualquier otro caso, la señal Y tiene valor '0'. La señal reset pone el circuito en su estado inicial (no se ha reconocido ningún elemento de la secuencia). Todos los cambios, salvo el reseteo del circuito, tienen lugar en el flanco de subida de la señal de reloj.

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

```
entity detector is
   port ( Y       : out std_logic;
          reset   : 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.

Solución

```
library IEEE;
use IEEE.std_logic_1164.all;
architecture detector of detector is
signal state : std_logic_vector(2 downto 0);
begin
-- Calculo del proximo estado
proximo_estado: process (clk, reset) is
begin
   Y <= '0';
   if ( (reset = '0') ) then -- Reset asincrono
      state <= "000";
   elsif rising_edge(clk) then
      case state is
          when "000" =>
            if (x = '0') then
               state <= "001";
            end if;
          when "001" =>
            if (x = '1') then
               state <= "000";
            else
               state <= "010";
             end if;
          when "010" =>
            if (x = '1') then
                state <= "011";
            end if;
          when "011" =>
            if (x = '0') then
               state <= "001";
            else
               state <= "100";
            end if;
          when "100" =>
            if (x = '1') then
               state <= "000";
            else
               state <= "000";
               Y <= '1';
            end if;
          when others =>
            state <= "000";
      end case;
   end if;
end process proximo_estado;
end architecture detector;
-----------------------------------------------
```

Pregunta 3.3

Escriba en VHDL la architecture de un circuito secuencial capaz de detectar cuando le llega la secuencia "11001" por su entrada X. El circuito detecta secuencias no solapadas.

El circuito tiene una señal de reloj (clk), una entrada serie de un bit (X), una señal de reset síncrono activa en 1 (reset) y una señal de salida de un bit (Y). La señal Y vale '1' cuando se detecta la secuencia. En cualquier otro caso, la señal Y tiene valor '0'. La señal reset pone el circuito en su estado inicial (no se ha reconocido ningún elemento de la secuencia). Todos los cambios tienen lugar en el flanco de subida de la señal de reloj.

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

```
entity detector is
   port ( Y       : out std_logic;
          reset   : 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 Moore.

Solución

```
library IEEE;
use IEEE.std_logic_1164.all;
architecture detector of detector is
signal state : std_logic_vector(2 downto 0);
begin
-- Calculo del proximo estado
proximo_estado: process (clk) is
begin
   if ( (reset = '1') and rising_edge(clk) ) then -- Reset sincrono
      state <= "000";
   elsif rising_edge(clk) then
      case state is
         when "000" =>
            if (x = '1') then
               state <= "001";
            end if;
         when "001" =>
             if (x = '0') then
               state <= "000";
             else
               state <= "010";
              end if;
          when "010" =>
             if (x = '0') then
                state <= "011";
             end if;
          when "011" =>
            if (x = '1') then
               state <= "001";
            else
               state <= "100";
            end if;
         when "100" =>
             if (x = '0') then
               state <= "000";
             else
               state <= "101";
              end if;
          when "101" =>
             if (x = '0') then
                state <= "000";
             else
                state <= "001";
             end if;
          when others =>
             state <= "000";
         end case;
      end if;
end process proximo_estado;
Y <= '1' when (state="101") else
'0';
end architecture detector;
-----------------------------------------------
```

Pregunta 3.4

Escriba en VHDL la architecture de un circuito secuencial capaz de detectar cuando le llega la secuencia "11001" por su entrada X. El circuito detecta secuencias no solapadas.

El circuito tiene una señal de reloj (clk), una entrada serie de un bit (X), una señal de reset síncrona activa en 1 (reset) y una señal de salida de un bit (Y). La señal Y vale '1' cuando se detecta la secuencia. En cualquier otro caso, la señal Y tiene valor '0'. La señal reset pone el circuito en su estado inicial (no se ha reconocido ningún elemento de la secuencia). Todos los cambios tienen lugar en el flanco de subida de la señal de reloj.

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

```
entity detector is
   port ( Y       : out std_logic;
          reset   : 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.

Solución

```
library IEEE;
use IEEE.std_logic_1164.all;
architecture detector of detector is
signal state : std_logic_vector(2 downto 0);
begin
-- Calculo del proximo estado
proximo_estado: process (clk) is
begin
   Y <= '0';
   if ( (reset = '1') and rising_edge(clk)) then -- Reset sincrono
      state <= "000";
   elsif rising_edge(clk) then
      case state is
          when "000" =>
            if (x = '1') then
               state <= "001";
            end if;
          when "001" =>
            if (x = '0') then
               state <= "000";
            else
               state <= "010";
             end if;
          when "010" =>
            if (x = '0') then
                state <= "011";
            end if;
          when "011" =>
            if (x = '1') then
               state <= "001";
            else
               state <= "100";
            end if;
          when "100" =>
            if (x = '0') then
               state <= "000";
            else
               state <= "000";
               Y <= '1';
            end if;
          when others =>
            state <= "000";
      end case;
   end if;
end process proximo_estado;
end architecture detector;
-----------------------------------------------
```

Pregunta 3.5

Escriba el código VHDL de la **architecture** que describe el comportamiento del circuito secuencial síncrono 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

```
library IEEE;
use IEEE.std_logic_1164.all;
architecture circuito3 of circuito3 is
signal state : std_logic_vector(2 downto 0);
begin
-- Calculo del proximo estado
proximo_estado: process (clk, reset) is
begin
if ( (reset = '1') ) then -- Reset asincrono
   state <= "000";
elsif rising_edge(clk) then
     case state is
         when "000" =>
            if (dir='1') then
                state <= "001"; --0011
            else
                state <= "010"; --0110
            end if;
         when "001" =>
             if (dir = '1') then
                state <= "010"; --0110
             else
                state <= "000"; --0000
              end if;
          when "010" =>
             if (dir = '1') then
                 state <= "011"; --0101
             else
                 state <= "100"; --0111
             end if;
          when "011" =>
           if (dir = '1') then
                state <= "100"; --0111
           else
                state <= "001"; --0011
           end if;
         when "100" =>
             if (dir = '1') then
                state <= "000"; --0000
             else
                state <= "011";--0101
              end if;
          when others =>
            state <= "000";
          end case;
      end if;
end process proximo_estado;
Y <= "0000" when (state="000")
     else  "0011" when (state = "001")
    else  "0110" when (state = "010")
    else "0101" when (state = "011")
    else "0111" when (state = "100")
    else "0000";
end architecture circuito3;
-------------------------------------------------
```

Pregunta 3.6

Escriba el código VHDL de la **architecture** que describe el comportamiento del circuito secuencial síncrono 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 Mealy.

Solución

```
library IEEE;
use IEEE.std_logic_1164.all;
architecture circuito3 of circuito3 is
signal state : std_logic_vector(2 downto 0);
begin
   proximo_estado: process (clk, reset) is
   begin
    if ( (reset = '1') ) then -- Reset asincrono
     state <= "000";  y <= "0000";
    elsif rising_edge(clk) then
       case state is
        when "000" =>
          if (dir='1') then
              state <= "001"; --0011
              y<= "0011";
          else
              state <= "010"; --0110
              y<="0110";
          end if;
        when "001" =>
            if (dir = '1') then
              state <= "010"; y <= "0110";
            else
              state <= "000"; y <= "0000";
            end if;
         when "010" =>
          if (dir = '1') then
              state <= "011"; y <= "0101";
          else
              state <= "100"; y <= "0111";
          end if;
         when "011" =>
          if (dir = '1') then
              state <= "100"; y <= "0111";
          else
              state <= "001"; y <= "0011";
          end if;
         when "100" =>
          if (dir = '1') then
              state <= "000"; y <= "0000";
          else
              state <= "011"; y <= "0101";
            end if;
          when others =>
           state <= "000"; y <= "0000";
        end case;
      end if;
   end process proximo_estado;
end architecture circuito3;
-------------------------------------------------
```

Pregunta 3.7

Escriba el código VHDL de la **architecture** que describe el comportamiento del circuito secuencial síncrono 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 síncrona activa a nivel bajo (`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 '0', la señal de salida `y` toma el valor `"0000"` en el flanco de subida de la señal de reloj. Cuando la señal de `reset` tiene valor '1' y la señal `dir` tiene valor '1', la señal de salida del circuito (y) toma cíclicamente los valores `"0000"`, `"0010"`, `"0100"` y `"1000"`. Cuando la señal de `reset` tiene valor '1' y la señal `dir` tiene valor '0', la señal de salida del circuito (y) toma cíclicamente los valores `"0000"`, `"0100"`, `"0010"` y `"1000"`. Todos los cambios 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

```
library IEEE;
use IEEE.std_logic_1164.all;
architecture circuito3 of circuito3 is
signal state : std_logic_vector(1 downto 0);
begin
-- Calculo del proximo estado
proximo_estado: process (clk) is
begin
if ( (reset = '0') and (rising_edge(clk)) ) then -- Reset sincrono
   state <= "00"; --0000
elsif rising_edge(clk) then
    case state is
        when "00" =>
           if (dir='1') then
               state <= "01"; --0010
           else
               state <= "10"; --0100
           end if;
        when "01" =>
            if (dir = '1') then
               state <= "10"; --0100
            else
               state <= "11"; --1000
             end if;
         when "10" =>
            if (dir = '1') then
                state <= "11"; --1000
            else
                state <= "01"; --0010
            end if;
        when others =>
            if (dir = '1') then
               state <= "00"; --0000
            else
               state <= "00";--0000
             end if;
       end case;
end if;
end process proximo_estado;
Y <= "0000" when (state="00")
    else  "0010" when (state = "01")
    else  "0100" when (state = "10")
    else "1000";
end architecture circuito3;
-------------------------------------------------
```

Pregunta 3.8

Escriba el código VHDL de la **architecture** que describe el comportamiento del circuito secuencial síncrono 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 síncrona activa a nivel bajo (`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 '0', la señal de salida `y` toma el valor `"0000"` en el flanco de subida de la señal de reloj. Cuando la señal de `reset` tiene valor '1' y la señal `dir` tiene valor '1', la señal de salida del circuito (y) toma cíclicamente los valores `"0000"`, `"0010"`, `"0100"` y `"1000"`. Cuando la señal de `reset` tiene valor '1' y la señal `dir` tiene valor '0', la señal de salida del circuito (y) toma cíclicamente los valores `"0000"`, `"0100"`, `"0010"` y `"1000"`. Todos los cambios 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 Mealy.

Solución

```
library IEEE;
use IEEE.std_logic_1164.all;
architecture circuito3 of circuito3 is
signal state : std_logic_vector(1 downto 0);
begin
proximo_estado: process (clk) is
begin
if ( (reset = '0') and (rising_edge(clk)) ) then -- Reset sincrono
   state <= "00"; --0000
   Y <= "0000";
elsif rising_edge(clk) then
     case state is
         when "00" =>
            if (dir='1') then
                state <= "01"; --0010
                Y <= "0010";
            else
                state <= "10"; --0100
                Y <= "0100";
            end if;
         when "01" =>
             if (dir = '1') then
                state <= "10"; --0100
                Y <= "0100";
             else
                state <= "11"; --1000
                Y <= "1000";
              end if;
          when "10" =>
            if (dir = '1') then
                state <= "11"; --1000
                Y <= "1000";
             else
                state <= "01"; --0010
                Y <= "0010";
             end if;
         when others =>
            state <= "00";  Y <= "0000";
        end case;
end if;
end process proximo_estado;
end architecture circuito3;
-----------------------------------------------
```