# INGENIERÍA DE COMPUTADORES III

Examen semana 2 de convocatoria Junio 2020, 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.

  La puntuación de cada pregunta de los bloques 1 y 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 `x1`, `x2`, `x3`, `x4` y `x5` en los instantes 0, delta (δ), 5ns, 10ns+delta ($10ns + \delta$), 15 ns, 20 ns, 20 ns + delta ($20ns + \delta$) y 25 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono4 is
end entity crono4;
architecture crono4 of crono4 is
  signal x1, x2, x3, x4, x5 : std_logic;
begin
  x1 <= '1',
        '0' after 5 ns,
        '1' after 15 ns,
        '0' after 20 ns;
  x2 <= '0',
        '1' after 15 ns,
        '0' after 25 ns;
  x3 <= x1 after 10 ns;
  Proc1: process
  begin
    for i in 0 to 2 loop
       x4 <= x1 or x2;
       x5 <= x4;
       wait for 10 ns;
     end loop;
     wait;
  end process;
end architecture crono4;
```

**Solución**

- Señal `x1`: 'U' en 0ns, '1' en delta, '0' en 5ns, '0' en 10ns+ delta, '1' en 15 ns y '0' a partir de 20ns.
- Señal `x2`: 'U' en 0ns, '0' en delta, '0' en 5ns, '0' en 10ns+ delta, '1' en 15 ns, '1' en 20 ns, '1' en 20 ns + delta y '0' en 25 ns.
- Señal `x3`: 'U' en 0ns, 'U' en delta, 'U' en 5ns, 'U' en 10ns+ delta y '0' a partir de 15 ns.
- Señal `x4`: 'U' en 0ns, 'U' en delta, 'U' en 5ns, '0' en 10ns+ delta, '0' en 15 ns, '0' en 20 ns y '1' a partir de 20 ns + delta.
- Señal `x5` 'U' en 0ns, 'U' en delta, 'U' en 5ns, 'U' en 10ns+ delta, 'U' en 15 ns, 'U' en 20 ns, '0' en 20 ns + delta y '0' en 25 ns.

## Pregunta 1.2

Dado el siguiente fragmento de código VHDL, indique cuál es el valor de las señales `x1`, `x2`, `x3`, `x4` y `x5` en los instantes 0, delta (δ), 5ns, 10ns+delta ($10ns + \delta$), 15 ns, 20 ns, 20 ns + delta ($20ns + \delta$) y 25 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono4 is
end entity crono4;
architecture crono4 of crono4 is
  signal x1, x2, x3, x4, x5 : std_logic;
begin
  x1 <= '0',
        '1' after 5 ns,
        '0' after 15 ns,
        '1' after 20 ns;
  x2 <= '0',
        '1' after 15 ns,
        '0' after 25 ns;
  x3 <= x1 after 10 ns;
  Proc1: process
  begin
    for i in 0 to 2 loop
       x4 <= x1 xor x2;
       x5 <= x4;
       wait for 10 ns;
     end loop;
     wait;
  end process;
end architecture crono4;
```

**Solución**

- Señal `x1`: 'U' en 0ns, '0' en delta, '1' en 5ns, '1' en 10ns+ delta y '0' en 15 ns, y '1' a partir de 20ns.
- Señal `x2`: 'U' en 0ns, '0' en delta, '0' en 5ns, '0' en 10ns+ delta, '1' en 15 ns, '1' en 20 ns, '1' en 20 ns + delta y '0' en 25 ns.
- Señal `x3`: 'U' en 0ns,'U' en delta, 'U' en 5ns, 'U' en 10ns+ delta y '1' a partir de 15 ns.
- Señal `x4`: 'U' en 0ns, 'U' en delta, 'U' en 5ns, '1' en 10ns+ delta, '1' en 15 ns, '1' en 20 ns y '0' a partir de 20 ns + delta.
- Señal `x5` 'U' en 0ns, 'U' en delta, 'U' en 5ns, 'U' en 10ns+ delta, 'U' en 15 ns, 'U' en 20 ns, '1' en 20 ns + delta y '1' en 25 ns.

## Pregunta 1.3

Dado el siguiente fragmento de código VHDL, indique cuál es el valor de las señales `x1`, `x2`, `x3`, `x4` y `x5` en los instantes 0, delta (δ), 5ns, 10ns+delta ($10ns + \delta$), 15 ns, 20 ns, 20 ns + delta ($20ns + \delta$), 25 ns y 35 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono4 is
end entity crono4;
architecture crono4 of crono4 is
  signal x1, x2, x3, x4, x5 : std_logic;
begin
  x1 <= '1',
        '0' after 5 ns,
        '1' after 15 ns,
        '0' after 20 ns;
  x2 <= '1',
        '0' after 15 ns,
        '1' after 25 ns;
  x3 <= x1 after 15 ns;
  Proc1: process
  begin
    for i in 0 to 2 loop
       x4 <= x1 or x2;
       x5 <= x4;
       wait for 10 ns;
     end loop;
     wait;
  end process;
end architecture crono4;
```

**Solución**

- Señal `x1`: 'U' en 0ns, '1' en delta, '0' en 5ns, '0' en 10ns+ delta, '1' en 15 ns y '0' a partir de 20ns.
- Señal `x2`: 'U' en 0ns, '1' en delta, '1' en 5ns, '1' en 10ns+ delta, '0' en 15 ns, '0' en 20 ns, '0' en 20 ns + delta y '1' a partir de 25 ns.
- Señal `x3`: 'U' entre 0ns y 25 ns y '0' a partir de 35 ns.
- Señal `x4`: 'U' en 0ns, 'U' en delta, 'U' en 5ns, '1' en 10ns+ delta, '1' en 15 ns, '1' en 20 ns y '0' a partir de 20 ns + delta.
- Señal `x5` 'U' en 0ns, 'U' en delta, 'U' en 5ns, 'U' en 10ns+ delta, 'U' en 15 ns, 'U' en 20 ns y '1' a partir de 20 ns + delta.

## Pregunta 1.4

Dado el siguiente fragmento de código VHDL, indique cuál es el valor de las señales `x1`, `x2`, `x3`, `x4` y `x5` en los instantes 0, delta (δ), 5ns, 10 ns, 10ns+delta ($10ns + \delta$), 15 ns, 20 ns, 20 ns + delta ($20ns + \delta$) y 25 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono4 is
end entity crono4;
architecture crono4 of crono4 is
  signal x1, x2, x3, x4, x5 : std_logic;
begin
  x1 <= '1',
        '0' after 5 ns,
        '1' after 15 ns,
        '0' after 20 ns;
  x2 <= '1',
        '0' after 15 ns,
        '1' after 25 ns;
  x3 <= x1 after 5 ns;
  Proc1: process
  begin
    for i in 0 to 2 loop
       x4 <= x1 xor x2;
       x5 <= x4;
       wait for 10 ns;
     end loop;
     wait;
  end process;
end architecture crono4;
```

**Solución**

- Señal `x1`: 'U' en 0ns, '1' en delta, '0' en 5ns, '0' en 10 ns, '0' en 10ns+ delta, '1' en 15 ns y '0' a partir de 20ns.
- Señal `x2`: 'U' en 0ns, '1' en delta, '1' en 5ns, '1' en 10 ns, '1' en 10ns+ delta, '0' en 15 ns, '0' en 20 ns, '0' en 20 ns + delta y '1' en 25 ns.
- Señal `x3`: 'U' en 0ns, 'U' en delta, '1' en 5ns, '0' en 10 ns, '0' en 10ns+ delta, '0' en 15 ns, '1' en 20 ns, '1' en 20 ns + delta y '0' en 25 ns.
- Señal `x4`: 'U' en 0ns, 'U' en delta, 'U' en 5ns, 'U' en 10 ns, '1' en 10ns+ delta, '1' en 15 ns, '1' en 20 ns y '0' a partir de 20 ns + delta.
- Señal `x5` 'U' en 0ns, 'U' en delta, 'U' en 5ns, 'U' en 10 ns, 'U' en 10ns+ delta, 'U' en 15 ns, 'U' en 20 ns, '1' en 20 ns + delta y '1' en 25 ns.

# Bloque 2 (3 puntos)

## Pregunta 2.1

Escriba en VHDL la **architecture** de una ALU de 8 bits. La **entity** y la tabla de operaciones de este circuito se muestran a continuación.

```
entity ALU is
   port( y: out std_logic_vector (7 downto 0);
             s: in std_logic_vector (1 downto 0);
             A: in std_logic_vector ( 7 downto 0);
             B: in std_logic_vector ( 7 downto 0));
end entity ALU;
```

| s | Operación |
|---|---|
| "00" | not B |
| "01" | A+B |
| "10" | B-A |
| "11" | B+1 |

Los números A y B se interpretan como enteros con signo para las operaciones aritméticas.

En el diseño sólo puede emplear las siguientes librerías:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;
```

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

**Solución**

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
architecture ALUCase of ALU is
begin
  alu: process (s, A, B)
  begin
     case s is
        when "00" => --not B
           y <=  not B;
        when "01" => --A+B
           y <= std_logic_vector(signed(A) + signed(B));
        when "10" => --B-A
           y <= std_logic_vector(signed(B) - signed(A));
        when others => --B+1
           y <= std_logic_vector (signed(B) + 1);
     end case;
   end process alu;
end architecture ALUCase;
```

## Pregunta 2.2

Escriba en VHDL la **architecture** de una ALU de 8 bits. La **entity** y la tabla de operaciones de este circuito se muestran a continuación.

```
entity ALU is
   port( y: out std_logic_vector (7 downto 0);
            s: in std_logic_vector (1 downto 0);
            A: in std_logic_vector ( 7 downto 0);
            B: in std_logic_vector ( 7 downto 0));
end entity ALU;
```

| s | Operación |
|---|---|
| "00" | not B |
| "01" | A+B |
| "10" | B-A |
| "11" | B+1 |

Los números A y B se interpretan como enteros con signo para las operaciones aritméticas.

En el diseño sólo puede emplear las siguientes librerías:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;
```

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

**Solución**

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
architecture ALUIf of ALU is
begin
  alu: process (s, A, B)
  begin
      if (s="00") then --not B
            y <=  not B;
      elsif (s="01") then --A+B
            y <= std_logic_vector(signed(A) + signed(B));
      elsif (s="10") then --B-A
            y <= std_logic_vector(signed(B) - signed(A));
      else --B+1
            y <= std_logic_vector (signed(B) + 1);
      end if;
    end process alu;
end architecture ALUIf;
```

## Pregunta 2.3

Escriba en VHDL la **architecture** de una ALU de 8 bits. La **entity** y la tabla de operaciones de este circuito se muestran a continuación.

```
entity ALU is
   port( y: out std_logic_vector (7 downto 0);
             s: in std_logic_vector (1 downto 0);
             A: in std_logic_vector ( 7 downto 0);
             B: in std_logic_vector ( 7 downto 0));
end entity ALU;
```

| s | Operación |
|---|---|
| "00" | not B |
| "01" | A+B |
| "10" | B-A |
| "11" | B+1 |

Los números A y B se interpretan como enteros con signo para las operaciones aritméticas.

En el diseño sólo puede emplear las siguientes librerías:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;
```

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

**Solución**

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
architecture ALUWhen of ALU is
begin
  y <= not(B) when (s = "00") else-- not B
         std_logic_vector(signed(A) + signed(B)) when (s= "01") else -- A+B
         std_logic_vector(signed(B) - signed(A)) when (s="10") else -- B-A
         std_logic_vector (signed(B) + 1); -- B+1
end architecture ALUWhen;
```

## Pregunta 2.4

Escriba en VHDL la **architecture** de una ALU de 8 bits. La **entity** y la tabla de operaciones de este circuito se muestran a continuación.

```
entity ALU is
   port( y: out std_logic_vector (7 downto 0);
             s: in std_logic_vector (1 downto 0);
             A: in std_logic_vector ( 7 downto 0);
             B: in std_logic_vector ( 7 downto 0));
end entity ALU;
```

| s | Operación |
|---|---|
| "00" | not B |
| "01" | A+B |
| "10" | B-A |
| "11" | B+1 |

Los números A y B se interpretan como enteros con signo para las operaciones aritméticas.

En el diseño sólo puede emplear las siguientes librerías:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;
```

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

**Solución**

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
architecture ALUWith of ALU is
begin
  with s select
        y <=   not B when "00", --notB
        std_logic_vector(signed(A) + signed(B)) when "01",
                --A+B
        std_logic_vector(signed(B) - signed(A)) when "10",
                --B-A
        std_logic_vector (signed(B) + 1) when others;
                --B+1
end architecture ALUWith;
```

# Bloque 3 (4 puntos)

## Pregunta 3.1

Escriba en VHDL la architecture de un circuito secuencial capaz de detectar cuando le llega la secuencia `"011"` por su entrada `X`. 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**

```
-- Máquina de Moore detectora de la secuencia 011

library IEEE;
use IEEE.std_logic_1164.all;

architecture detector of detector is
    signal state : std_logic_vector(1 downto 0);
begin
      -- Cálculo del próximo estado
      proximo_estado: process (clk) is
      begin
          if ( (reset = '0') ) then -- Reset asíncrono
               state <= "00";
          elsif rising_edge(clk) then
              case state is
                  when "00" =>
                       if (x = '0') then
                            state <= "01";
                       end if;
                  when "01" =>
                       if (x = '1') then
                            state <= "10";
                       end if;
                  when "10" =>
                      if (x = '1') then
                            state <= "11";
                       else
                            state <= "01";
                       end if;
                  when others =>
                       if (x = '0') then
                            state <= "01";
                       else
                            state <= "00";
                        end if;
              end case;
          end if;
      end process proximo_estado;
      Y <= '1' when (state="11") 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 "011" por su entrada X. 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**

```
-- Máquina de Mealy detectora de la secuencia 011

library IEEE;
use IEEE.std_logic_1164.all;

architecture detector of detector is
    signal state : std_logic_vector(1 downto 0);
begin
      -- Cálculo del próximo estado
      proximo_estado: process (clk) is
      begin
          Y <= '0';
          if ( (reset = '0') ) then -- Reset asíncrono
               state <= "00";
          elsif rising_edge(clk) then
              case state is
                  when "00" =>
                       if (x = '0') then
                            state <= "01";
                       end if;
                  when "01" =>
                       if (x = '1') then
                            state <= "10";
                       end if;
                  when "10" =>
                      if (x = '1') then
                            state <= "00";
                            Y<='1';
                       else
                            state <= "01";
                       end if;
                  when others =>
                       state <= "00";
              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 "011" por su entrada X. 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 Moore.

**Solución**

```
-- Máquina de Moore detectora de la secuencia 011

library IEEE;
use IEEE.std_logic_1164.all;

architecture detector of detector is
    signal state : std_logic_vector(1 downto 0);
begin
      -- Cálculo del próximo estado
      proximo_estado: process (clk) is
      begin
          if ( (reset = '1') and rising_edge(clk) ) then
               state <= "00";
          elsif rising_edge(clk) then
              case state is
                  when "00" =>
                       if (x = '0') then
                            state <= "01";
                       end if;
                  when "01" =>
                       if (x = '1') then
                            state <= "10";
                       end if;
                  when "10" =>
                      if (x = '1') then
                            state <= "11";
                       else
                            state <= "01";
                       end if;
                  when others =>
                       if (x = '0') then
                            state <= "01";
                       else
                            state <= "00";
                        end if;
              end case;
          end if;
      end process proximo_estado;
      Y <= '1' when (state="11") 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 "011" por su entrada X. 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**

```
-- Máquina de Mealy detectora de la secuencia 011

library IEEE;
use IEEE.std_logic_1164.all;

architecture detector of detector is
    signal state : std_logic_vector(1 downto 0);
begin
      -- Cálculo del próximo estado
      proximo_estado: process (clk) is
      begin
          Y <= '0';
          if ( (reset = '1') and rising_edge(clk)) then
               state <= "00";
          elsif rising_edge(clk) then
              case state is
                  when "00" =>
                       if (x = '0') then
                            state <= "01";
                       end if;
                  when "01" =>
                       if (x = '1') then
                            state <= "10";
                       end if;
                  when "10" =>
                      if (x = '1') then
                            state <= "00";
                            Y<='1';
                       else
                            state <= "01";
                       end if;
                  when others =>
                       state <= "00";
              end case;
          end if;
      end process proximo_estado;
end architecture detector;
-----------------------------------------------
```