INGENIERÍA DE COMPUTADORES III

Examen semana 1 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 `a`, `b`, `c`, `d`, `s1`, `s2`, `s3` en los instantes 0, delta (δ), 2 delta (2δ), 3 delta (3δ) y 4 delta (4δ).

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono4 is
end entity crono4;
architecture crono4 of crono4 is
  signal a,b,c,d,s1,s2,s3: std_logic;
begin
  proc1: process
  begin
    a <= '0';
    b <= '1';
    c <= '0';
    wait;
  end process;
  s1<= a or b;
  s2 <= b and c;
  s3 <= s1 xor s2;
  d<= not s3;
end architecture crono4;
```

Solución

- Señal `a`: 'U' en 0ns, '0' en delta, '0' en 2delta, '0' en 3 delta y '0' en 4 delta.
- Señal `b`: 'U' en 0ns, '1' en delta, '1' en 2delta, '1' en 3 delta y '1' en 4 delta.
- Señal `c`: 'U' en 0ns, '0' en delta, '0' en 2delta, '0' en 3 delta y '0' en 4 delta.
- Señal `d`: 'U' en 0ns, 'U' en delta, 'U' en 2delta, 'U' en 3 delta y '0' en 4 delta.
- Señal `s1` 'U' en 0ns, 'U' en delta, '1' en 2delta, '1' en 3 delta y '1' en 4 delta.
- Señal `s2` 'U' en 0ns, 'U' en delta, '0' en 2delta, '0' en 3 delta y '0' en 4 delta.
- Señal `s3` 'U' en 0ns, 'U' en delta, 'U' en 2delta, '1' en 3 delta y '1' en 4 delta.

Pregunta 1.2

Dado el siguiente fragmento de código VHDL, indique cuál es el valor de las señales `a`, `b`, `c`, `d`, `s1`, `s2`, `s3` en los instantes 0, delta (δ), 2 delta (2δ) y 3 delta (3δ).

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono5 is
end entity crono5;
architecture crono5 of crono5 is
  signal a,b,c,d,s1,s2,s3: std_logic;
begin
  proc1: process
  begin
    a <= '1';
    b <= '0';
    c <= '1';
    wait;
  end process;
  s1<= a xor b;
  s2 <= b and c;
  s3 <= s1 xor s2;
  d<= not s2;
end architecture crono5;
```

Solución

- Señal `a`: 'U' en 0ns, '1' en delta, '1' en 2delta y '1' en 3 delta.
- Señal `b`: 'U' en 0ns, '0' en delta, '0' en 2delta y '0' en 3 delta.
- Señal `c`: 'U' en 0ns, '1' en delta, '1' en 2delta y '1' en 3 delta.
- Señal `d`: 'U' en 0ns, 'U' en delta, 'U' en 2delta y '1' en 3 delta.
- Señal `s1`: 'U' en 0ns, 'U' en delta, '1' en 2delta y '1' en 3 delta.
- Señal `s2`: 'U' en 0ns, 'U' en delta, '0' en 2delta y '0' en 3 delta.
- Señal `s3` 'U' en 0ns, 'U' en delta, 'U' en 2delta y '1' en 3 delta.

Pregunta 1.3

Dado el siguiente fragmento de código VHDL, indique cuál es el valor de las señales `a`, `b`, `c`, `d`, `s1`, `s2`, `s3` en los instantes 0, delta (δ), 2 delta (2δ), 10+ delta ($10+\delta$), 10+2 delta ($10+2\delta$), 10+3 delta ($10+3\delta$) y 10+4delta ($10+4\delta$).

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono4 is
end entity crono4;
architecture crono4 of crono4 is
  signal a,b,c,d,s1,s2,s3: std_logic;
begin
  proc1: process
  begin
    a <= '0';
    b <= '1';
    wait for 10 ns;
    c <= '0';
    wait;
  end process;
  s1<= a or b;
  s2 <= b and c;
  s3 <= s1 xor s2;
  d<= not s3;
end architecture crono4;
```

Solución

- Señal `a`: 'U' en 0ns, '0' a partir de delta.
- Señal `b`: 'U' en 0ns, '1' a partir de delta.
- Señal `c`: 'U' en 0ns, 'U' en delta, 'U' en 2delta y '0' a partir 10+delta.
- Señal `d`: 'U' en 0ns, 'U' en delta, 'U' en 2delta, 'U' en 10+delta, 'U' en 10+2delta, 'U' en 10+3delta y '0' en 10+4delta.
- Señal `s1`: 'U' en 0ns, 'U' en delta, '1' a partir de 2delta.
- Señal `s2`: 'U' en 0ns, 'U' en delta, 'U' en 2delta, 'U' en 10 +delta y '0' a partir de 10+2delta.
- Señal `s3` 'U' en 0ns, 'U' en delta, 'U' en 2delta, 'U' en 10+delta, 'U' en 10+2delta y '1' a partir de 10+3delta.

Pregunta 1.4

Dado el siguiente fragmento de código VHDL, indique cuál es el valor de las señales `a`, `b`, `c`, `d`, `s1`, `s2`, `s3` en los instantes 0, delta (δ), 2 delta (2δ), 10+ delta ($10+\delta$), 10+2 delta ($10+2\delta$) y 10+3 delta ($10+3\delta$).

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono5 is
end entity crono5;
architecture crono5 of crono5 is
  signal a,b,c,d,s1,s2,s3: std_logic;
begin
  proc1: process
  begin
    a <= '1';
    wait for 10 ns;
    b <= '0';
    c <= '1';
    wait;
  end process;
  s1<= a and b;
  s2 <= b and c;
  s3 <= s1 or s2;
  d<= a or s2;
end architecture crono5;
```

Solución

- Señal `a`: 'U' en 0ns, '1' a partir de delta.
- Señal `b`: 'U' en 0ns, 'U' en delta, 'U' en 2delta y '0' a partir de 10+delta.
- Señal `c`: 'U' en 0ns, 'U' en delta, 'U' en 2delta y '1' a partir de 10+delta.
- Señal `d`: 'U' en 0ns, 'U' en delta, '1' a partir de 2delta.
- Señal `s1`: 'U' en 0ns, 'U' en delta, 'U' en 2delta, 'U' en 10+delta y '0' a partir de 10+2delta.
- Señal `s2`: 'U' en 0ns, 'U' en delta, 'U' en 2delta, 'U' en 10+delta y '0' a partir de 10+2delta.
- Señal `s3` 'U' en 0ns, 'U' en delta, 'U' en 2delta, 'U' en 10+delta, 'U' en 10+2delta y '0' a partir de 10+3delta.

Bloque 2 (3 puntos)

Pregunta 2.1

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

```
entity Desplazador is
   port( y: out std_logic_vector (7 downto 0);
             s: in std_logic_vector (1 downto 0);
             x: in std_logic_vector ( 7 downto 0));
end entity Desplazador;
```

s	Operación
"00"	Pasa el dato
"01"	Desplaza a la izquierda una posición rellenando con ceros
"10"	Desplaza a la derecha una posición rellenando con unos
"11"	Desplaza a la derecha dos posiciones rellenando con ceros

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

Solución

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
architecture DespCase OF Desplazador is
begin
   process(s,x)
   begin
      case s is
          when "00" => -- pasa sin modificar
              y <= x;
          when "01" => -- despl. izqda con 0
              y <= x(6 downto 0) & '0';
          when "10" => -- despl. drcha. con 1
              y <= '1' & x(7 downto 1);
          when others => -- desp. dercha 2 pos.
              y <="00" & x(7 DOWNTO 2);
     end case;
end process;
end DespCase;
```

Pregunta 2.2

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

```
entity Desplazador is
   port( y: out std_logic_vector (7 downto 0);
             s: in std_logic_vector (1 downto 0);
             x: in std_logic_vector ( 7 downto 0));
end entity Desplazador;
```

s	Operación
"00"	Pasa el dato
"01"	Desplaza a la izquierda una posición rellenando con ceros
"10"	Desplaza a la derecha una posición rellenando con unos
"11"	Desplaza a la derecha dos posiciones rellenando con ceros

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

Solución

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
architecture DespIf OF Desplazador is
begin
   process(s,x)
   begin
      if (s="00") then -- pasa sin modificar
             y <= x;
      elsif (s="01") then -- despl. izqda con 0
             y <= x(6 downto 0) & '0';
      elsif (s="10") then -- despl. drcha. con 1
             y <= '1' & x(7 downto 1);
      else -- desp. dercha 2 pos.
             y <="00" & x(7 DOWNTO 2);
      end if;
    end process;
end DespIf;
```

Pregunta 2.3

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

```
entity Desplazador is
   port( y: out std_logic_vector (7 downto 0);
         s: in std_logic_vector (1 downto 0);
         x: in std_logic_vector ( 7 downto 0));
end entity Desplazador;
```

s	Operación
"00"	Pasa el dato
"01"	Desplaza a la izquierda una posición rellenando con ceros
"10"	Desplaza a la derecha una posición rellenando con unos
"11"	Desplaza a la derecha dos posiciones rellenando con ceros

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

Solución

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
architecture DespWhen OF Desplazador is
begin
   y <= x when (s = "00") else-- pasa sin modificar
        x(6 downto 0) & '0' when (s= "01") else -- despl. izqda con 0
        '1' & x(7 downto 1) when (s="10") else -- despl. drcha. con 1
        "00" & x(7 DOWNTO 2); -- desp. dercha 2 pos.
end DespWhen;
```

Pregunta 2.4

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

```
entity Desplazador is
   port( y: out std_logic_vector (7 downto 0);
         s: in std_logic_vector (1 downto 0);
         x: in std_logic_vector ( 7 downto 0));
end entity Desplazador;
```

s	Operación
"00"	Pasa el dato
"01"	Desplaza a la izquierda una posición rellenando con ceros
"10"	Desplaza a la derecha una posición rellenando con unos
"11"	Desplaza a la derecha dos posiciones rellenando con ceros

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

Solución

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
architecture DespWith OF Desplazador is
begin
   with s select
      y <= x   when "00",
               -- pasa sin modificar
            x(6 downto 0) & '0' when "01",
               -- despl. izqda con 0
            '1' & x(7 downto 1) when "10",
               -- despl. drcha. con 1
            "00" & x(7 downto 2) when others;
               -- desp. dercha 2 pos.
end DespWith;
```

Bloque 3 (4 puntos)

Pregunta 3.1

Escriba en VHDL la architecture de un circuito contador de 4 bits con señal de reset asíncrona activa a nivel alto. El circuito se comporta de la forma siguiente. Cuando la señal de `reset` se pone a 1, la salida del contador se pone al valor "0001". Mientras la señal de `reset` está a 0, el contador funciona de manera síncrona, rotando circularmente hacia la derecha el patrón de bits: 0001, 1000, 0100, 0010, 0001, 1000, 0100, y así sucesivamente. La operación de desplazamiento se produce en el flanco de subida de la señal de reloj. La **entity** de este circuito se muestra a continuación.

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

Solución

```
architecture counter of counter is
    signal r_reg  : std_logic_vector(3 downto 0);
    signal r_next : std_logic_vector(3 downto 0);
begin
    -- Registro
    process (clk, reset)
    begin
        if (reset='1') then
            r_reg <= "0001";
        elsif ( rising_edge(clk) ) then
            r_reg <= r_next;
        end if;
    end process;
    -- Lógica cálculo siguiente estado
    r_next <= r_reg(0) & r_reg(3 downto 1);
    -- Lógica de salida
    q <= r_reg;
end architecture counter;
```

Pregunta 3.2

Escriba en VHDL la **architecture** de un circuito contador de 4 bits con señal de reset asíncrona activa a nivel bajo. El circuito se comporta de la forma siguiente. Cuando la señal de `reset` se pone a 0, la salida del contador se pone al valor `"0001"`. Mientras la señal de `reset` está a 1, el contador funciona de manera síncrona, rotando circularmente hacia la derecha el patrón de bits: 0001, 1000, 0100, 0010, 0001, 1000, 0100, y así sucesivamente. La operación de desplazamiento se produce en el flanco de subida de la señal de reloj. La **entity** de este circuito se muestra a continuación.

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

Solución

```
architecture counter of counter is
    signal r_reg  : std_logic_vector(3 downto 0);
    signal r_next : std_logic_vector(3 downto 0);
begin
    -- Registro
    process (clk, reset)
    begin
        if (reset='0') then
            r_reg <= "0001";
        elsif ( rising_edge(clk) ) then
            r_reg <= r_next;
        end if;
    end process;
    -- Lógica cálculo siguiente estado
    r_next <= r_reg(0) & r_reg(3 downto 1);
    -- Lógica de salida
    q <= r_reg;
end architecture counter;
```

Pregunta 3.3

Escriba en VHDL la **architecture** de un circuito contador de 4 bits con señal de reset síncrona activa a nivel alto. El circuito se comporta de la forma siguiente. Cuando la señal de `reset` se pone a 1 y se está en el flanco de subida de la señal de reloj, la salida del contador se pone al valor `"0001"`. Mientras la señal de `reset` está a 0, el contador funciona de manera síncrona, rotando circularmente hacia la derecha el patrón de bits: 0001, 1000, 0100, 0010, 0001, 1000, 0100, y así sucesivamente. La operación de `reset` y de desplazamiento se producen en el flanco de subida de la señal de reloj. La **entity** de este circuito se muestra a continuación.

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

Solución

```
architecture counter of counter is
    signal r_reg  : std_logic_vector(3 downto 0);
    signal r_next : std_logic_vector(3 downto 0);
begin
    -- Registro
    process (clk)
    begin
        if (rising_edge(clk)) then
           if (reset = '1') then
             r_reg <= "0001";
             else
             r_reg <= r_next;
         end if;
       end if;
    end process;
    -- Lógica cálculo siguiente estado
    r_next <= r_reg(0) & r_reg(3 downto 1);
    -- Lógica de salida
    q <= r_reg;
end architecture counter;
```

Pregunta 3.4

Escriba en VHDL la architecture de un circuito contador de 4 bits con señal de reset síncrona activa a nivel bajo. El circuito se comporta de la forma siguiente. Cuando la señal de `reset` se pone a 0 y se está en el flanco de subida de la señal de reloj, la salida del contador se pone al valor `"0001"`. Mientras la señal de `reset` está a 1, el contador funciona de manera síncrona, rotando circularmente hacia la derecha el patrón de bits: 0001, 1000, 0100, 0010, 0001, 1000, 0100, y así sucesivamente. La operación de reset y de desplazamiento se producen en el flanco de subida de la señal de reloj.

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

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

Solución

```
architecture counter of counter is
    signal r_reg  : std_logic_vector(3 downto 0);
    signal r_next : std_logic_vector(3 downto 0);
begin
    -- Registro
    process (clk)
    begin
        if (rising_edge(clk)) then
           if (reset = '0') then
             r_reg <= "0001";
             else
             r_reg <= r_next;
         end if;
        end if;
    end process;
    -- Lógica cálculo siguiente estado
    r_next <= r_reg(0) & r_reg(3 downto 1);
    -- Lógica de salida
    q <= r_reg;
end architecture counter;
```