INGENIERÍA DE COMPUTADORES 3

Solución al examen de Septiembre 2023

PREGUNTA 1 (2 puntos)

1.a) (1 punto) Dibuje el diagrama conceptual correspondiente al fragmento de código *Fragmento 1*.

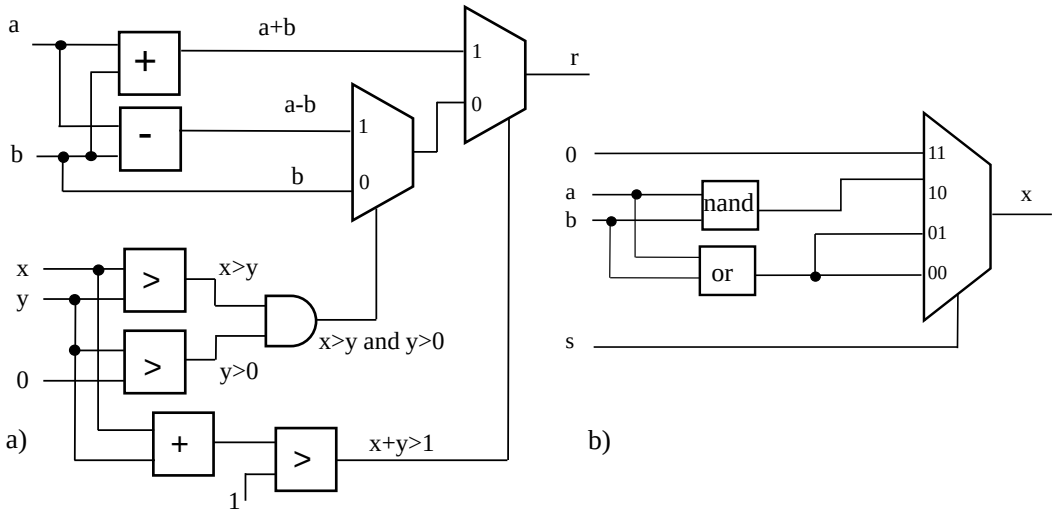
1.b) (1 punto) Dibuje el diagrama conceptual correspondiente al fragmento de código *Fragmento 2*.

```
---- Fragmento 1-------------
signal a, b, r : unsigned (7 downto 0);
signal x, y    : unsigned (3 downto 0);
...
r <=  a+b when x+y>1 else
      a-b when x>y and y>0 else
      b;
```

```
 ---- Fragmento 2-------------
signal s: std_logic_vector (1 downto 0);
signal a, b, x : std_logic;
...
with s select
   x <= (a or b) when "00"|"01",
        (a nand b) when "10",
         '0' when others;
```

Solución a la Pregunta 1

Los diagramas conceptuales correspondientes a Fragmento 1 y a Fragmento 2 se muestran en las Fig. 1a y 1b, respectivamente.



PREGUNTA 2 (3 puntos)

Escriba en VHDL, de las formas que se detallan a continuación, la **architecture** que describe el comportamiento de un circuito combinacional codificador de 4 a 2 con prioridad. A continuación, se muestran la **entity** del circuito y su tabla de la verdad.

```
entity codificadorPrioridad4a2 is
   port (  codigo     : out std_logic_vector(1 downto 0);
           activo     : out std_logic;
           x          : in  std_logic_vector(3 downto 0) );
end entity codificadorPrioridad4a2;
```

x	codigo	activo
1 – – –	11	1
0 1 – –	10	1
0 0 1 –	01	1
0 0 0 1	00	1
0 0 0 0	00	0

En el código VHDL de la **architecture**, emplee para evaluar la señal `codigo`:

2.a) (0.75 puntos) Una asignación concurrente condicional (**when** - **else**).

2.b) (0.75 puntos) Una asignación concurrente de selección (**with** - **select**).

2.c) (0.75 puntos) Una sentencia **if**.

2.d) (0.75 puntos) Una sentencia **case**.

Solución a la Pregunta 2

El Código VHDL 1.1–1.4 son las soluciones de los Apartados 2.a–2.d, respectivamente.

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

entity codificadorPrioridad4a2 is
   port ( codigo   :  out std_logic_vector(1 downto 0);
          activo   :  out std_logic;
          x        :  in  std_logic_vector(3 downto 0) );
end entity codificadorPrioridad4a2;

architecture codPrior4a2 of codificadorPrioridad4a2 is
begin
   codigo <= "11" when ( x(3) = '1' ) else
             "10" when ( x(2) = '1' ) else
             "01" when ( x(1) = '1' ) else
             "00";
   activo <= x(3) or x(2) or x(1) or x(0);
end architecture codPrior4a2;
```

Código VHDL 1.1: Circuito descrito empleando una sentencia **when - else**.

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

entity codificadorPrioridad4a2 is
   port ( codigo   :  out std_logic_vector(1 downto 0);
          activo   :  out std_logic;
          x        :  in  std_logic_vector(3 downto 0) );
end entity codificadorPrioridad4a2;

architecture codPrior4a2_selec of codificadorPrioridad4a2 is
begin
   with x select
      codigo <= "11" when "1000" | "1001" | "1010" | "1011" |
                          "1100" | "1101" | "1110" | "1111",
                "10" when "0100" | "0101" | "0110" | "0111",
                "01" when "0010" | "0011",
                "00" when others;
   activo <= x(3) or x(2) or x(1) or x(0);
end architecture codPrior4a2_selec;
```

Código VHDL 1.2: Circuito descrito empleando una sentencia **with - select**.

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

entity codificadorPrioridad4a2 is
   port ( codigo   :  out std_logic_vector(1 downto 0);
          activo   :  out std_logic;
          x        :  in  std_logic_vector(3 downto 0) );
end entity codificadorPrioridad4a2;

architecture codPrior4a2_procIf of codificadorPrioridad4a2 is
begin
   process ( x )
   begin
      if ( x(3) = '1' ) then
          codigo <= "11";
      elsif ( x(2) = '1' ) then
          codigo <= "10";
      elsif ( x(1) = '1' ) then
          codigo <= "01";
      else
          codigo <= "00";
      end if;
      activo <= x(3) or x(2) or x(1) or x(0);
   end process;
end architecture codPrior4a2_procIf;
```

Código VHDL 1.3: Circuito descrito empleando una sentencia **if**.

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

entity codificadorPrioridad4a2 is
   port ( codigo   :  out std_logic_vector(1 downto 0);
          activo   :  out std_logic;
          x        :  in  std_logic_vector(3 downto 0) );
end entity codificadorPrioridad4a2;

architecture codPrior4a2_procCase of codificadorPrioridad4a2 is
begin
   process ( x )
   begin
      case x is
         when "1000" | "1001" | "1010" | "1011" |
              "1100" | "1101" | "1110" | "1111"   =>
            codigo <= "11";
         when "0100" | "0101" | "0110" | "0111"   =>
            codigo <= "10";
         when "0010" | "0011"                     =>
            codigo <= "01";
         when others                              =>
            codigo <= "00";
      end case;
      activo <= x(3) or x(2) or x(1) or x(0);
   end process;
end architecture codPrior4a2_procCase;
```

Código VHDL 1.4: Circuito descrito empleando una sentencia **case**.

PREGUNTA 3 (3 puntos)

Escriba en VHDL la **architecture** de un circuito secuencial síncrono que opera en el flanco de subida de la señal de reloj. Se supone que la señal de reloj de entrada tiene un periodo de 0.01 s. Este circuito tiene dos señales de entrada de 1 bit: la señal de reloj `clk` y la señal de reset asíncrona activa a nivel 0 `rst`. El circuito tiene una señal de salida de dos bits llamada `salida` cuyo valor es el resultado de la cuenta descrita a continuación.

El circuito realiza la cuenta cíclica de `"00"-"01"-"10"-"11"`. Se ha de diseñar el circuito como una máquina de estado de tipo Moore, de modo que cada estado corresponda a un valor diferente de la señal de salida del circuito, permaneciendo 1 s en cada estado. La señal de reset pasa al circuito al estado cuya señal de salida tiene el valor `"00"`.

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

```
entity contador is
  port ( salida   : out std_logic_vector(1 downto 0);
         clk, rst : in  std_logic);
end entity contador;
```

Para el diseño del circuito emplee únicamente las siguientes librerías:

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

Solución a la Pregunta 3

El código VHDL del circuito contador se muestra en Código VHDL 1.5.

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;
architecture contador of contador is
  signal state : unsigned(1 downto 0);
begin
  process(clk,rst)
    variable temp1: integer range 0 to 100;
  begin
    if (rst='0') then
       temp1:= 0;
       state<= "00";
     elsif (rising_edge(clk)) then
       temp1 := temp1+1;
       if (temp1=100) then
          temp1:=0;
          state <= state+1;
        end if;
      end if;
  end process;
  salida <= std_logic_vector(state);
end contador;
```

Código VHDL 1.5: Circuito contador.

PREGUNTA 4 (2 puntos)

Programe el banco de pruebas del circuito secuencial que ha diseñado en la Pregunta 3. El banco de pruebas debe generar una señal de reloj de un periodo de 0.01 s, que ha de ser entrada del circuito secuencial. El programa de test debe realizar las siguientes comprobaciónes, mostrando en cada comprobación un mensaje de error sólo si la salida no tiene el valor esperado.

- Resetear el circuito. Comprobar que tras el reseteo la señal de salida toma el valor `"00"`.
- Esperar un segundo y después comprobar que la salida tiene el valor `"01"`.
- Esperar un segundo y después comprobar que la salida tiene el valor `"10"`.
- Esperar un segundo y después comprobar que la salida tiene el valor `"11"`.
- Esperar un segundo y después comprobar que la salida vuelve a tomar el valor `"00"`.
- Mostrar un mensaje con el número total de errores.

Solución a la Pregunta 4

El Código VHDL 1.6 es el diseño del banco de pruebas solución de la Pregunta 4.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity bp_contador is
end entity bp_contador;
architecture bp_contador of bp_contador is
  constant PERIODO   : time       := 0.01 sec; -- Reloj
  signal   salida : std_logic_vector (1 downto 0);
  signal   clk       : std_logic  := '0';    -- Entradas UUT
  signal   reset : std_logic;

  component contador is
    port(salida   : out std_logic_vector(1 downto 0);
         clk, rst : in  std_logic);
  end component contador;
  -- Procedimiento para comprobar las salidas
   procedure comprueba_salidas
      ( esperado_z             :       std_logic_vector(1 downto 0);
        actual_z               :       std_logic_vector(1 downto 0);
        error_count            : inout integer) is
   begin
       if ( esperado_z /= actual_z ) then
          report "ERROR: Salida esperada ("        &
                 std_logic'image(esperado_z(1))    &
                 std_logic'image(esperado_z(0))    &
                 "), salida actual ("              &
                  std_logic'image(actual_z(1))     &
                 std_logic'image(actual_z(0))      &
                 "), instante: "                   &
                 time'image(now);
          error_count := error_count + 1;
       end if;
    end procedure comprueba_salidas;
begin
  -- Instanciar y conectar UUT
  uut : component contador port map
                  (salida, clk, reset);

  clk <= not clk after (PERIODO/2);
  gen_vec_test : process is
    variable error_count : integer := 0; -- Núm. errores
  begin
      report "Comienza la simulación";
      -- Vectores de test y comprobación del resultado
      reset <= '1';
      wait for PERIODO/4;
      reset <= '0';
      wait for PERIODO;
```

```
48        comprueba_salidas("00", salida, error_count);
49       -- report time'image(now);
50        reset<='1';
51        wait for 100*PERIODO;   -- 1
52        comprueba_salidas("01", salida, error_count);
53        wait for 100*PERIODO; -- Accion tras pulsar P
54        comprueba_salidas("10", salida, error_count);
55        wait for 100*PERIODO; -- Accion tras pulsar P
56        comprueba_salidas("11", salida, error_count);
57        wait for 100*PERIODO; -- Accion tras pulsar P
58        comprueba_salidas("00", salida, error_count);
59         -- Informe final
60        if (error_count = 0) then
61          report "Simulación finalizada sin errores";
62        else
63          report  "ERROR: Hay "              &
64                  integer'image(error_count) &
65                  " errores.";
66        end if;
67        wait;
68    end process gen_vec_test;
69  end architecture bp_contador;
```

Código VHDL 1.6: Banco de pruebas del circuito contador.