INGENIERÍA DE COMPUTADORES III

Solución al Ejercicio de Autocomprobación 4

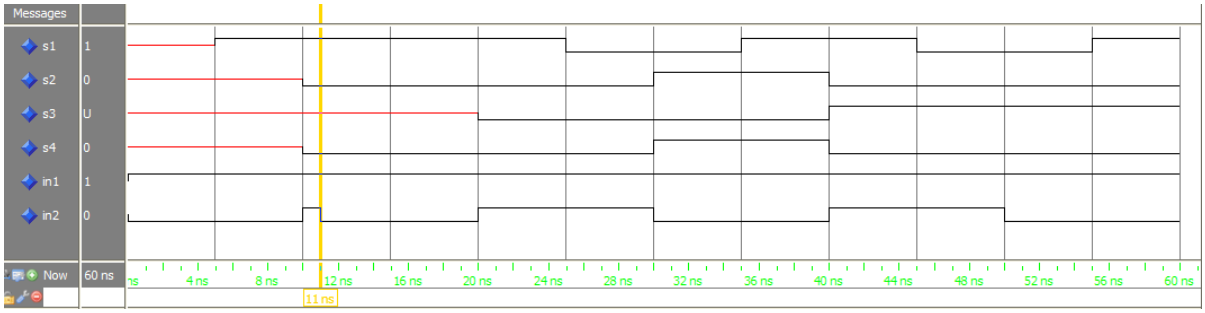
PREGUNTA 1 (2 puntos)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `in1`, `in2`, `s1`, `s2`, `s3` y `s4` entre los instantes 0 y 60 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity codigo is
end entity codigo;
architecture codigo of codigo is
   signal s1, s2, s3, s4 :  std_logic;
   signal in1, in2       :  std_logic;
begin
    s1 <= in1 nand in2 after 5 ns;
    process
        variable var1, var2 : std_logic;
    begin
        for i in 0 to 4 loop
           var1 := in1 nand in2;
           var2 := var1;
           s2   <= in1 nand in2;
           s3   <= s2;
           s4 <= var2;
           wait for 10 ns;
        end loop;
      wait;
    end process;
    in1 <= '1';
    in2 <= '0',
           '1' after 10 ns, '0' after 11 ns,
           '1' after 20 ns, '0' after 30 ns,
           '1' after 40 ns, '0' after 50 ns;
end architecture codigo;
```

Solución a la Pregunta 1

A continuación, se muestra el cronograma de evolución de las señales `s1`, `s2`, `s3`, `s4`, `in1` e `in2` entre los instantes 0 y 60 ns.



Debido al retardo de 5 ns, el valor de la señal `s1` no está definido hasta el instante 5 ns. Por ello, esta señal tiene inicialmente el valor 'U'. Debido a que el retardo de 5 ns es de tipo inercial, el pulso de 1 ns en `in2` no produce cambios en el valor de la señal `s1`.

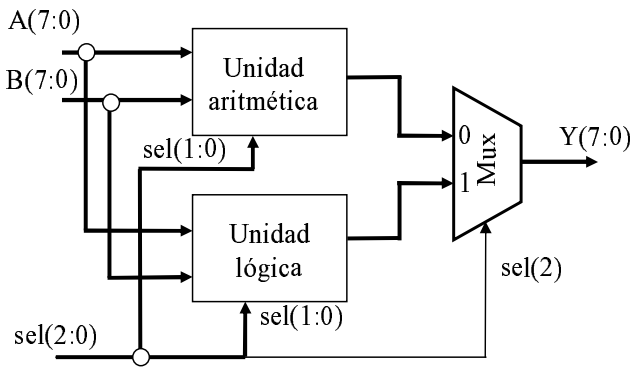
La señal `s2` no tiene un valor definido hasta el instante $(10+\delta)$ ns. En dicho instante la señal `s2` toma el valor '0'. Toma el valor '1' en el instante $(30+\delta)$ ns. Finalmente, la señal `s2` toma el valor '0' en el instante $(40+\delta)$ ns.

La señal `s3` está retrasada 10 ns respecto a la señal `s2`.

La señal `s4` toma los mismos valores que la señal `s2`.

PREGUNTA 2 (3 puntos)

A continuación, se muestra el circuito, la tabla de operaciones y la **entity** de una ALU. La ALU realiza operaciones sobre dos operandos de 8 bits, denominados `A` y `B`. La salida de la ALU se selecciona mediante el bit más significativo de la señal `sel`, mientras que la operación que realiza se especifica mediante los otros dos bits de esta señal.



Unidad	sel	Operación
Aritmética	000	A
	001	A+B
	010	A+1
	011	A-1
Lógica	100	not A
	101	A and B
	110	A or B
	111	A xor B

```
entity ALU is
   port( Y    : out std_logic_vector ( 7 downto 0 );
         A, B : in  std_logic_vector ( 7 downto 0 );
         sel  : in  std_logic_vector ( 2 downto 0 )  );
end entity ALU;
```

Escriba en VHDL la **architecture** que describe el comportamiento de la ALU, empleando para ello tres sentencias de asignación concurrente de selección.

Puede realizar las hipótesis de diseño adicionales que estime convenientes, siempre y cuando las explique detalladamente y no estén en contradicción con las especificaciones anteriores.

Solución a la Pregunta 2

El diseño de la ALU se muestra en Código VHDL 1.1.

```
-------------------------------------------------------------
----Diseño de la ALU empleando sentencias concurrentes de seleccion
-------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity ALU  is
     port( Y       : out std_logic_vector ( 7  downto 0);
           A, B  : in std_logic_vector (7  downto 0);
           sel    : in std_logic_vector ( 2  downto 0) );
end entity ALU;

architecture ALU_concurrente  of ALU  is
   signal arith, logic: std_logic_vector(7  downto 0);
begin
   --Unidad Aritmética
   with sel(1  downto 0) select
        arith  <= A  when "00",
                  std_logic_vector(signed(A)+signed(B)) when "01",
                  std_logic_vector(signed(A)+1) when "10",
                  std_logic_vector(signed(A)-1) when    others;

   --Unidad Lógica
    with sel(1  downto 0) select
        logic  <= not A  when "00",
                  A  and B  when "01",
                  A  or B  when "10",
                  A  xor B  when others;

   --Multiplexor
    with sel(2) select
        Y  <= arith  when '0',
              logic  when others;

end architecture ALU_concurrente;
-------------------------------------------------------------
```

Código VHDL 1.1: Diseño de la ALU.

PREGUNTA 3 (3 puntos)

Diseñe un circuito secuencial síncrono del tipo *máquina de Moore*, con una entrada serie de un bit, que detecte cuando recibe por dicha entrada tres o mas bits consecutivos de valor '1'. La **entity** del circuito se muestra a continuación.

```
entity detector is
   port( Y     : out std_logic;
         state : out std_logic_vector(1 downto 0);
         X     : in  std_logic;
         reset : in  std_logic;
         clk   : in  std_logic);
end entity detector;
```

Como puede verse en la definición anterior de la **entity**, el circuito tiene una señal de reloj (`clk`), un entrada serie de un bit (`X`), una señal de reset asíncrona activa en '1' (`reset`), una señal que indica el estado en el que se encuentra el circuito (`state`) y una señal de salida de un bit (`Y`).

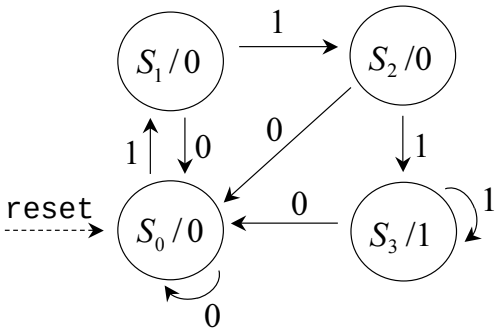
El funcionamiento del circuito es el siguiente. La señal `Y` se pone a '1' cuando por la entrada `X` se reciben tres o más bits consecutivos de valor '1'. Todos los cambios tienen lugar en el flanco de subida de la señal de reloj. Por otra parte, la señal `reset` pone asíncronamente el circuito en su estado inicial.

Escriba en VHDL la **architecture** que describe el comportamiento del circuito en términos de una máquina de Moore. Dibuje el diagrama de estados correspondiente al circuito que ha diseñado.

Puede realizar las hipótesis de diseño adicionales que estime convenientes, siempre y cuando las explique detalladamente y no estén en contradicción con las especificaciones anteriores.

Solución a la Pregunta 3

A continuación, se muestra el diagrama de estados del circuito implementado como una máquina de Moore.



El código VHDL que describe el comportamiento del circuito en términos de una máquina de Moore se muestra en Código VHDL 1.2.

```
-------------------------------------------------
---Detector de 3 o mas unos consecutivos
-------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

entity detector is
    port(   Y      : out std_logic;
            state  : out std_logic_vector(1 downto 0);
            X      : in std_logic;
            reset  : in std_logic;
            clk    : in std_logic);
end entity detector;

architecture detector of detector is
       signal internal_state: std_logic_vector(1 downto 0);
begin
 state <= internal_state;

   --Cálculo salida
 salida: process (internal_state) is
 begin
     case internal_state is
          when "00" => Y <= '0';
          when "01" => Y <= '0';
          when "10" => Y <= '0';
          when others => Y <= '1';
      end case;
 end process salida;
--Calculo del proximo estado
 proximo_estado: process (clk, reset)
 begin
      if (reset = '1') then --reset asíncrono
           internal_state <= "00";
      elsif (rising_edge(clk)) then
           case internal_state is
                when "00" =>     -- Estado actual: 00
                     if X = '1' then
                          internal_state <= "01";
                     else
                          internal_state <= "00";
                    end if;
                when "01" =>    --Estado actual: 01
                     if X = '1' then
                          internal_state <= "10";
                     else
                          internal_state <= "00";
                     end if;
                when "10" =>    --Estado actual: 10
                     if X = '1' then
                           internal_state <= "11";
                     else
                          internal_state <= "00";
                    end if;
                 when "11" =>    -- Estado actual: 11
                      if X = '1' then
                          internal_state <= "11";
                      else
                          internal_state <= "00";
                     end if;
                 when others=>    -- Por completitud
                      internal_state <= "00";
                end case;
           end if;
end process proximo_estado;
end architecture detector;
-----------------------------------------
```

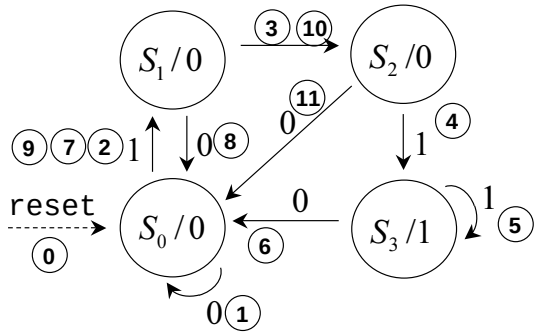
Código VHDL 1.2: Diseño del detector de una secuencia 3 unos o más consecutivos.

PREGUNTA 4 (2 puntos)

Programe en VHDL el banco de pruebas del circuito secuencial que ha diseñado al contestar a la Pregunta 3. Explique detalladamente cómo el programa de test comprueba de manera sistemática el funcionamiento del circuito. El banco de pruebas debe comprobar que los valores obtenidos 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 4

El banco de pruebas del detector de secuencias diseñado en la Pregunta 3 se muestra en Código VHDL 1.3 y 1.4. Este banco de pruebas comprueba todas las transiciones de estados del circuito en el orden indicado en la figura siguiente.



```
-----------------------------------------------------
-- Banco de pruebas del detector de 3 unos consecutivos
library IEEE;
use IEEE.std_logic_1164.all;

entity bp_detector is
end entity bp_detector;

architecture bp_detector of bp_detector is
   constant PERIODO    : time         := 100 ns; -- Reloj
   signal    state      : std_logic_vector(1 downto 0);-- Salidas UUT
   signal    Y          : std_logic;
   signal    clk        : std_logic   := '0';      -- Entradas UUT
   signal    reset, X   : std_logic;


   component detector is
    port( Y       : out std_logic;
          state   : out std_logic_vector(1 downto 0);
          X       : in std_logic;
          reset   : in std_logic;
          clk     : in std_logic);
   end component detector;

    -- Procedimiento para comprobar las salidas
    procedure comprueba_salidas
        ( esperado_state  :        std_logic_vector(1 downto 0);
          actual_state    :         std_logic_vector(1 downto 0);
          esperado_Y      :         std_logic;
          actual_Y        :         std_logic;
          error_count     :         inout integer) is
    begin
         -- Comprueba state
         if ( esperado_state /= actual_state ) then
             report "ERROR: Estado esperado ("      &
                    std_logic'image(esperado_state(1))     &
                    std_logic'image(esperado_state(0))     &
                    "), estado actual ("            &
                    std_logic'image(actual_state(1))       &
                    std_logic'image(actual_state(0))       &
                    "), instante: "                 &
                    time'image(now);
             error_count := error_count + 1;
         end if;          -- Comprueba Y
         if ( esperado_Y /= actual_Y ) then
             report "ERROR: Salida Y esperada ("      &
                    std_logic'image(esperado_Y)     &
                    "), salida actual ("            &
                    std_logic'image(actual_Y)       &
                    "), instante: "                 &
                    time'image(now);
             error_count := error_count + 1;
         end if;
     end procedure comprueba_salidas;
```

Código VHDL 1.3: Diseño del banco de pruebas del detector de una secuencia 3 unos o más consecutivos.

```
begin
   -- Instanciar y conectar UUT
   uut : component detector port map
                    (Y, state, X, reset, clk);

   reset <=    '0','1' after (PERIODO/4),
               '0' after (PERIODO+PERIODO/4);
   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
        X <= '0';    wait for PERIODO;    -- 1
        comprueba_salidas("00",state,'0',Y,error_count);
        X <= '1';    wait for PERIODO;    -- 2
        comprueba_salidas("01",state,'0',Y,error_count);
        X <= '1';    wait for PERIODO;    -- 3
        comprueba_salidas("10",state,'0',Y,error_count);
        X <= '1';    wait for PERIODO;    -- 4
        comprueba_salidas("11",state,'1',Y,error_count);
        X <= '1';    wait for PERIODO;    -- 5
        comprueba_salidas("11" ,state,'1',Y,error_count);
        X <= '0';    wait for PERIODO;    -- 6
        comprueba_salidas("00",state,'0',Y,error_count);
        X <= '1';    wait for PERIODO;    -- 7
        comprueba_salidas("01",state,'0',Y,error_count);
        X <= '0';    wait for PERIODO;    -- 8
        comprueba_salidas("00",state,'0',Y,error_count);
        X <= '1';    wait for PERIODO;    -- 9
        comprueba_salidas("01",state,'0',Y,error_count);
        X <= '1';    wait for PERIODO;    -- 10
        comprueba_salidas("10",state,'0',Y,error_count);
        X <= '0';    wait for PERIODO;    -- 11
        comprueba_salidas("00",state,'0',Y,error_count);
     -- Informe final
         report  "Hay "                &
                  integer'image(error_count ) &
                  " errores.";
     wait;       -- Final del bloque process
   end process gen_vec_test;
end architecture bp_detector;
-------------------------------------
```

Código VHDL 1.4: Continuación del diseño del banco de pruebas del detector de una secuencia 3 unos o más consecutivos.