Código asignatura	Nombre asignatura
71012018	**Ingeniería de Computadores III**
Fecha alta y origen	Convocatoria
04/11/2015	Junio 2013 (2ª Semana)
Página Asignatura	

INGENIERÍA DE COMPUTADORES 3

Solución al examen de Junio 2013, Segunda Semana

PREGUNTA 1 (2 puntos)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `x1`, `x2`, `x3`, `x4`, `x5` entre los instantes 0 y 60 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity crono2 is
end entity crono2;
architecture crono2 of crono2 is
   signal x1  : unsigned (2 downto 0);
   signal x2, x3, x4, x5 : std_logic;
begin
 process is
  begin
    for i in 0 to 3 loop
      x1 <= to_unsigned(i,3);
      x2 <= std_logic(x1(2));
      x3 <= std_logic(x1(1));
      x4 <= std_logic(x1(0));
      wait for 10 ns;
    end loop;
    wait;
  end process;
  x5 <= x3 after 15 ns;
end architecture crono2;
```

Solución a la Pregunta 1

En la Figura 1.1 se muestra el cronograma de evolución de las señales.

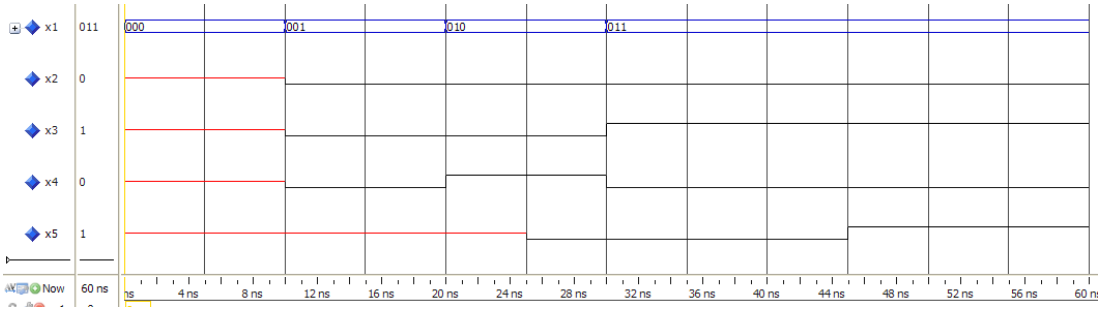
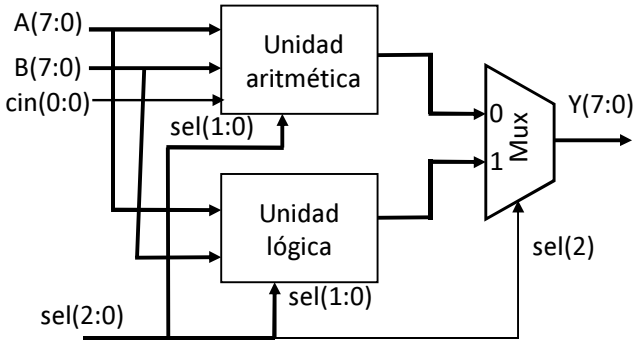


Figura 1.1: Cronograma de evolución de las señales.

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 por los otros dos bits de esta señal. Escriba en VHDL la **architecture** que describe el comportamiento de la ALU, empleando para ello únicamente tres sentencias de asignación concurrente de selección. Asimismo, en el diseño únicamente pueden emplearse los dos siguientes paquetes de la librería IEEE:

```
IEEE.std_logic_1164
IEEE.numeric_std
```



Unidad	sel	Operación
Aritmética	000	A+1
	001	A+B
	010	A+B+cin
	011	B-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);
         cin  : in std_logic_vector(0 downto 0));
end entity ALU;
```

Solución a la Pregunta 2

La **architecture** que describe el comportamiento de la ALU se muestra en Código VHDL 1.1.

```
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);
          cin   : in std_logic_vector(0 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 <= std_logic_vector(signed(A)+1) when "00",
               std_logic_vector(signed(A)+signed(B)) when "01",
               std_logic_vector(signed(A)+signed(B)+signed("0"&cin)) when "10",
               std_logic_vector(signed(B)-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: Descripción del comportamiento de la ALU.

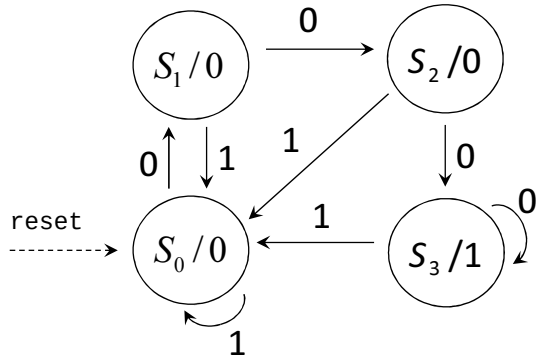
PREGUNTA 3 (2.5 puntos)

Diseñe un circuito secuencial síncrono capaz de detectar cuando le llegan al menos tres ceros consecutivos por su entrada. La **entity** del circuito se muestra a continuación. El circuito tiene una señal de reloj (`clk`), un entrada serie de un bit (`X`), una señal de reset asíncrona activa en '0' (`reset`), una señal que indica el estado en que se encuentra el circuito (`state`) y una señal de salida de un bit (`Y`). La señal `Y` se pone a '1' si por la entrada `X` se han recibido tres o más ceros consecutivos. La señal `reset` pone el circuito en su estado inicial. Todos los cambios tienen lugar en el flanco de subida de la señal de reloj. 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.

```
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;
```

Solución a la Pregunta 3

El diagrama de estados correspondiente al detector de secuencia se muestra en la figura mostrada a continuación.



El código VHDL del circuito detector de secuencia se muestra en Código VHDL 1.2.

```
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;

--Calculo 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 = '0') then --reset asíncrono
           internal_state <= "00";
      elsif (rising_edge(clk)) then
           case internal_state is
                when "00" =>
                     if X = '0' then
                          internal_state <= "01";
                     else
                          internal_state <= "00";
                    end if;
                when "01" =>
                     if X = '0' then
                          internal_state <= "10";
                     else
                          internal_state <= "00";
                     end if;
                when "10" =>
                     if X = '0' then
                           internal_state <= "11";
                     else
                           internal_state <= "00";
                    end if;
                 when "11" =>
                      if X = '0' 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;
```

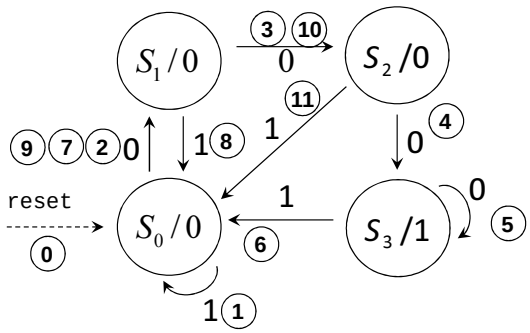
Código VHDL 1.2: Diseño del circuito detector de secuencia.

PREGUNTA 4 (2.5 puntos)

Programe el banco de pruebas del circuito secuencial que ha diseñado. 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 código VHDL correspondiente al banco de pruebas del detector se muestra en Código VHDL 1.3–1.4. Este banco de pruebas comprueba todas las transiciones de estados del circuito en el orden indicado en la figura mostrada a continuación.



```
--------------------------------------------------
-- Banco de pruebas del detector de 3 ceros 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: Banco de pruebas del detector.

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

   reset <=    '1','0' after (PERIODO/4),
               '1' 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 <= '1';    wait for PERIODO;    -- 1
         comprueba_salidas("00",state,'0',Y,error_count);
         X <= '0';    wait for PERIODO;    -- 2
         comprueba_salidas("01",state,'0',Y,error_count);
         X <= '0';    wait for PERIODO;    -- 3
         comprueba_salidas("10",state,'0',Y,error_count);
         X <= '0';    wait for PERIODO;    -- 4
         comprueba_salidas("11",state,'1',Y,error_count);
         X <= '0';    wait for PERIODO;    -- 5
         comprueba_salidas("11" ,state,'1',Y,error_count);
         X <= '1';    wait for PERIODO;    -- 6
         comprueba_salidas("00",state,'0',Y,error_count);
         X <= '0';    wait for PERIODO;    -- 7
         comprueba_salidas("01",state,'0',Y,error_count);
         X <= '1';    wait for PERIODO;    -- 8
         comprueba_salidas("00",state,'0',Y,error_count);
         X <= '0';    wait for PERIODO;    -- 9
         comprueba_salidas("01",state,'0',Y,error_count);
         X <= '0';    wait for PERIODO;    -- 10
         comprueba_salidas("10",state,'0',Y,error_count);
         X <= '1';    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 banco de pruebas del detector.