INGENIERÍA DE COMPUTADORES 3

Solución al examen de Junio 2016, Primera Semana

PREGUNTA 1 (2 puntos)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales a, b, c, d entre los instantes 0 y 100 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono1 is
end entity crono1;
architecture crono1 of crono1 is
   signal a, b, c, d : std_logic;
begin
    bloque1 : process
    begin
       b <= '0';
       wait for 5 ns;
       a <= '1';
       b <= '1';
       wait for 10 ns;
       a <= '0';
       b <= '0';
       wait for 5 ns;
       a <= '1';
       wait for 15 ns;
       b <= '1';
       wait for 5 ns;
       b<= '1';
    end process bloque1;
    bloque2 : process
    begin
       wait for 20 ns;
       a <= '1';
       wait for 20 ns;
       a <= '0';
    end process bloque2;
    c <= b after 10 ns;
    d <= transport b after 10 ns;
end architecture crono1;
```

Solución a la Pregunta 1

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

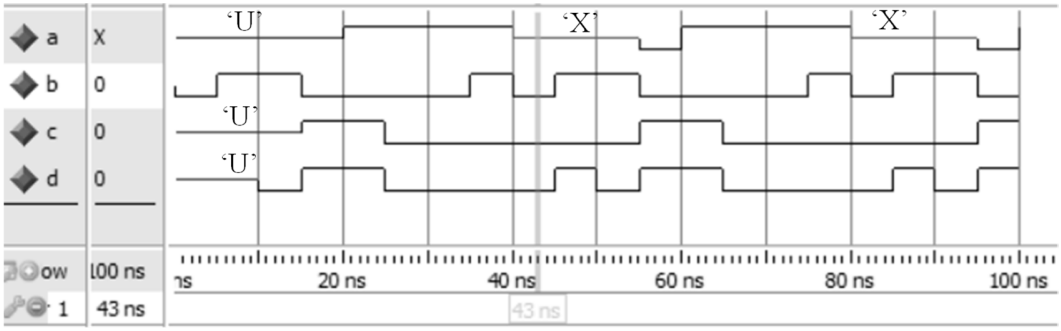

Figura 1.1: Cronograma de evolución de las señales.

PREGUNTA 2 (2.5 puntos)

Escriba en VHDL la **architecture** que modela el comportamiento de los dos siguientes componentes:

a. (1 punto) Una latch D con entrada enable activa a nivel alto y cuya **entity** se muestra a continuación. Emplee en la descripción de la **architecture** una sentencia **if**.

```
entity D_latch is
   port( Q    : out std_logic;
         D : in std_logic;
         Enable   : in std_logic);
end entity D_latch;
```

b. (1.5 puntos) Un biestable (*flip-flop*) JK disparado por el flanco de subida del reloj (`clk`) y con reset asíncrono (`reset_n`) activado a nivel bajo. La **entity** se muestra a continuación.

```
entity biestableJK is
   port ( q, q_n              : out std_logic;
          clk, J, K, reset_n  : in std_logic );
end entity biestableJK;
```

Solución a la Pregunta 2

La **architecture** que describe la latch D y el biestable JK se muestran en Código VHDL 1.1 y Código VHDL 1.2, respectivamente.

```
library IEEE;
use IEEE.std_logic_1164.all;
architecture Behavior OF D_latch is
begin
     process(D, Enable)
     begin
          if Enable='1' then
               Q <= D;
          end if;
     end process;
end Behavior;
```

Código VHDL 1.1: Architecture de la latch D.

```
-------------------------------------------------
-- Biestable JK con reset asíncrono en nivel LOW
library IEEE;
use IEEE.std_logic_1164.all;
architecture biestableJK of biestableJK is
    signal q_interna : std_logic;
begin
    q   <= q_interna;
    q_n <= not q_interna;
    process (reset_n, clk) is
        variable JK : std_logic_vector(1 downto 0);
    begin
        if (reset_n = '0') then
            q_interna <= '0';
        elsif rising_edge(clk) then
            JK := J & K;
            case (JK) is
                when "01"   => q_interna <= '0';
                when "10"   => q_interna <= '1';
                when "11"   => q_interna <= not q_interna;
                when others => null;
            end case;
        end if;
    end process;
end architecture biestableJK;
-------------------------------------------------
```

Código VHDL 1.2: Architecture del biestable JK.

PREGUNTA 3 (3.5 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 '1' (`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

En la Figura 1.2 se muestra el diagrama de estados del circuito detector de secuencias.

El código VHDL del detector de secuencias se muestra en Código VHDL 1.3.

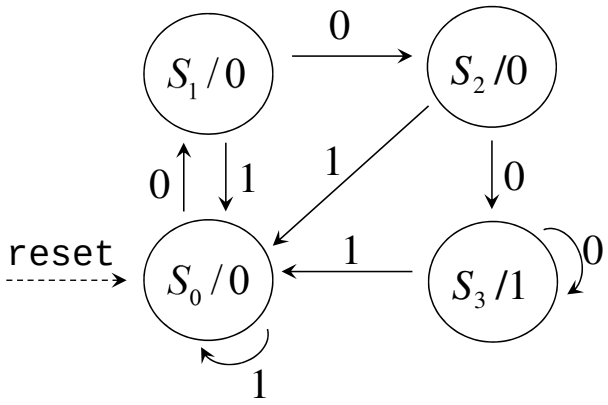

Figura 1.2: Diagrama de estados del circuito detector de secuencias.

```
-------------------------------------------------
---Detector de 3 o mas ceros 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;
 --Cálculo del próximo 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: S0
                     if X = '0' then
                          internal_state <= "01";
                     else
                          internal_state <= "00";
                     end if;
                when "01" =>    --Estado actual: S1
                     if X = '0' then
                          internal_state <= "10";
                     else
                          internal_state <= "00";
                     end if;
                when "10" =>    --Estado actual: S2
                     if X = '0' then
                          internal_state <= "11";
                     else
                          internal_state <= "00";
                     end if;
                 when "11" =>    -- Estado actual: S3
                      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 1.3: Diseño del circuito detector.

PREGUNTA 4 (2 puntos)

Programe el banco de pruebas del circuito secuencial que ha diseñado en 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 diseñado va comprobando que el detector de secuencia pase por los arcos de transición de estados en el orden mostrado en la Figura 1.3.

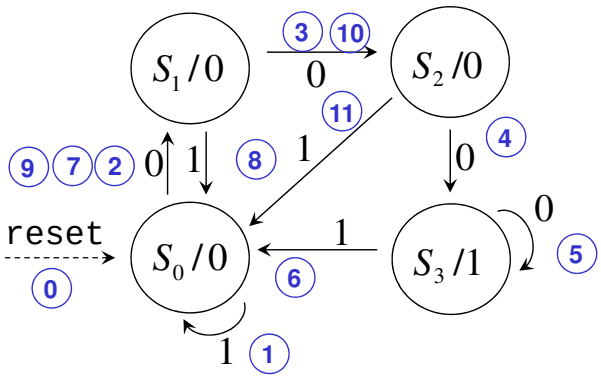

Figura 1.3: Transiciones de estado testeadas en el banco de pruebas del circuito detector de secuencias.

El código VHDL correspondiente al banco de pruebas del detector se muestra en Código VHDL 1.4–1.5.

```
  ------------------------------------------------------
-- 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;
begin
```

Código VHDL 1.4: Banco de pruebas del detector.

```
  -- 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 <= '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 1.5: Continuación del banco de pruebas del detector.