INGENIERÍA DE COMPUTADORES 3

Solución al examen de Septiembre 2018

PREGUNTA 1 (2 puntos)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `clk`, `x1`, `x2`, `x3`, `x4`, `x5` y `x6` entre los instantes 0 y 100 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono2 is
end entity crono2;
architecture crono2 of crono2 is
  signal x1, x2, x3, x4, x5, x6 : std_logic;
  signal clk : std_logic:='0';
begin
  x1 <= '0' , '1' after 10 ns, '0' after 20 ns, '1' after 25 ns;
  proc1: process (clk)
  begin
      if ( falling_edge(clk) ) then
        x4 <= x3;
        x3 <= x2;
        x2 <= x1;
      end if;
  end process;
  clk <= not clk after 10 ns;
  proc2: process (clk) is
    variable t1, t2, t3: std_logic;
  begin
    if (falling_edge(clk) ) then
       t1:= x1;
       t2:= t1;
       t3:= t2;
       x5 <= t1;
       x6 <= x5;
    end if;
  end process;
end architecture crono2;
```

Solución a la Pregunta 1

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

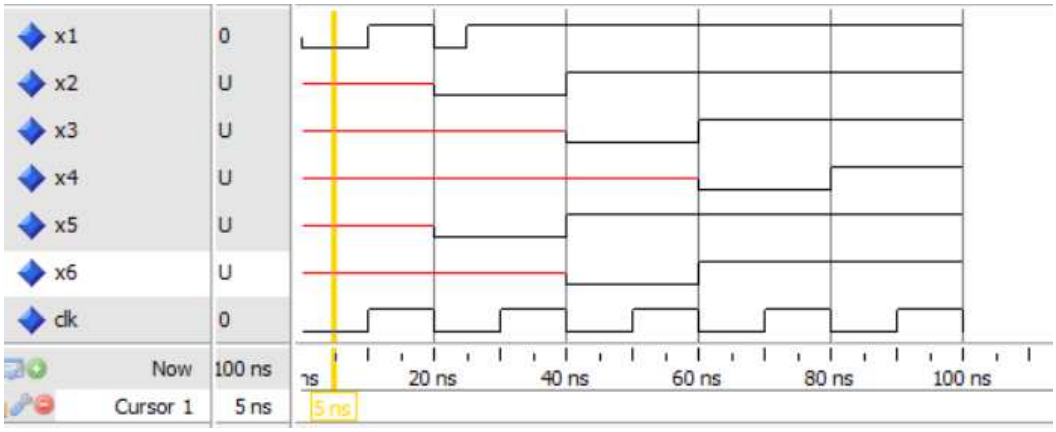


Figura 1.1: Cronograma de evolución de las señales.

PREGUNTA 2 (3 puntos)

Diseñe un contador binario síncrono de cuatro bits que opere en el flanco de subida de la señal de reloj. El contador tiene la siguiente **entity**:

```
entity contadorBinario is
   port( q     : out std_logic_vector(3 downto 0);
         d     : in std_logic_vector(3 downto 0);
         syn_clr, en, load : in std_logic;
         clk, reset : in std_logic);
end entity contadorBinario;
```

La señal de salida `q` indica el valor actual de la cuenta. La señal de entrada `d` se carga en paralelo cuando se habilita la carga. La señal de entrada `reset` es activa a nivel alto y realiza un reset asíncrono del circuito poniendo la cuenta a cero. El contador tiene unas señales de entrada que permiten, en el flanco de subida de la señal de reloj, poner la cuenta a cero (reset síncrono), cargar un valor específico en la cuenta (señal de entrada `d`), activar la cuenta o pausarla, tal como se indica en la siguiente tabla. Cuando la cuenta está activada, se pasa cíclicamente por los valores “0000”, “0001”, ..., “1110”, “1111”.

`syn_clr`	`load`	`en`	Operación
1	-	-	Pone el contador a cero (reset síncrono)
0	1	-	Carga en paralelo
0	0	1	Activa la cuenta
0	0	0	Pausa la cuenta

En el diseño únicamente pueden emplearse los dos siguientes paquetes de la librería IEEE:

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

Solución a la Pregunta 2

La **architecture** del contador binario se muestra en el Código 1.1.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity contadorBinario is
    port( q       : out std_logic_vector(3 downto 0);
          d       : in std_logic_vector(3 downto 0);
          syn_clr, en, load : in std_logic;
          clk, reset : in std_logic);
end entity contadorBinario;

architecture contadorBinario of contadorBinario is
   signal r_reg: unsigned(3 downto 0);
   signal r_next: unsigned(3 downto 0);
begin
   --registro
   process (clk, reset)
   begin
      if (reset = '1') then
         r_reg <= (others => '0');
      elsif (rising_edge(clk)) then
         r_reg <= r_next;
      end if;
   end process;
 -- logica siguiente estado
    r_next <= (others => '0') when syn_clr = '1' else
              unsigned(d) when load = '1' else
              r_reg + 1 when en='1' else
              r_reg;
 -- logica de salida
    q <= std_logic_vector(r_reg);
end contadorBinario;
```

Código VHDL 1.1: Architecture del contador binario.

PREGUNTA 3 (2 puntos)

Programe en VHDL un banco de pruebas para el circuito que ha diseñado al contestar a la Pregunta 2. La señal de reloj (`clk`) debe tener un periodo de 20 ns e inicialmente valer '0'. El primer flanco de subida de la señal de reloj se ha de producir en el instante 10 ns. El programa de test debe realizar consecutivamente las acciones siguientes:

1. *Reset*. La señal de reset ha de tener el valor '1' durante los primeros 15 ns.
2. *Mantener la cuenta activa durante tres periodos de la señal de reloj*.
3. *Cargar en la cuenta el valor "1001"*.
4. *Pausar la cuenta*.

El programa de test debe mostrar mensajes de error en el caso de que la señal de salida no tome el valor esperado. Dibuje el cronograma de las señales aplicadas al circuito y las salidas esperadas.

Solución a la Pregunta 3

En la Figura 1.2 se muestra el cronograma de evolución de las señales.

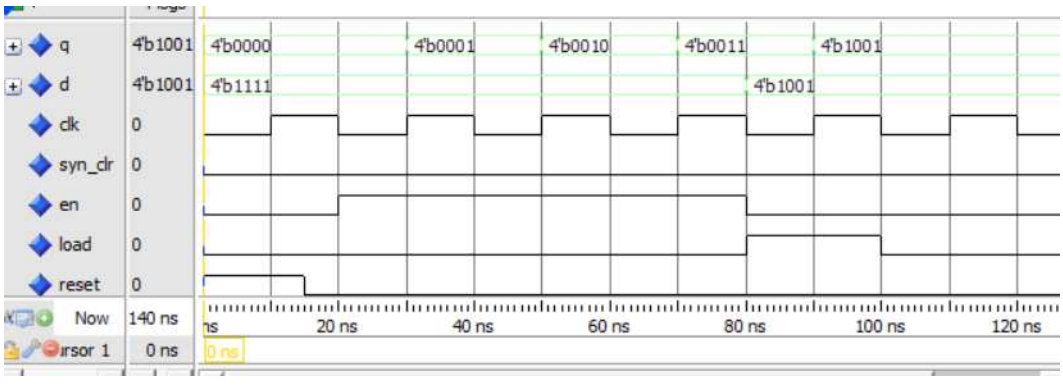


Figura 1.2: Cronograma de evolución de las señales correspondientes a la Pregunta 3.

El código VHDL del banco de pruebas se muestra en Código VHDL 1.2–1.3.

```
-------------------------------------
-- Banco de pruebas del contador binario
-- ascendente y descendente de 4 bits
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity bp_contador is
end entity bp_contador;
architecture bp_contador of bp_contador is
   constant PERIODO    : time          := 20 ns; -- Reloj
   signal     q, d           : std_logic_vector(3 downto 0);-- Salidas UUT
   signal     clk         : std_logic   := '0';      -- Entradas UUT
   signal     syn_clr, en, load, reset   : std_logic;
component contadorBinario is
    port( q       : out std_logic_vector(3 downto 0);
          d       : in std_logic_vector(3 downto 0);
          syn_clr, en, load : in std_logic;
          clk, reset : in std_logic);
 end component contadorBinario;
     -- Procedimiento para comprobar las salidas
    procedure comprueba_salidas
       ( esperado_q              :          std_logic_vector(3 downto 0);
         actual_q                 :          std_logic_vector(3 downto 0);
         error_count              : inout integer) is
    begin
        -- Comprueba q
        if ( esperado_q /= actual_q ) then
            report "ERROR: Estado esperado ("      &
                    std_logic'image(esperado_q(3))    &
                    std_logic'image(esperado_q(2))    &
                    std_logic'image(esperado_q(1))    &
                    std_logic'image(esperado_q(0))    &
                    "), estado actual ("             &
                    std_logic'image(actual_q(3))      &
                    std_logic'image(actual_q(2))      &
                    std_logic'image(actual_q(1))      &
                    std_logic'image(actual_q(0))      &
                    "), instante: "                   &
                    time'image(now);
            error_count := error_count + 1;
        end if;
     end procedure comprueba_salidas;
```

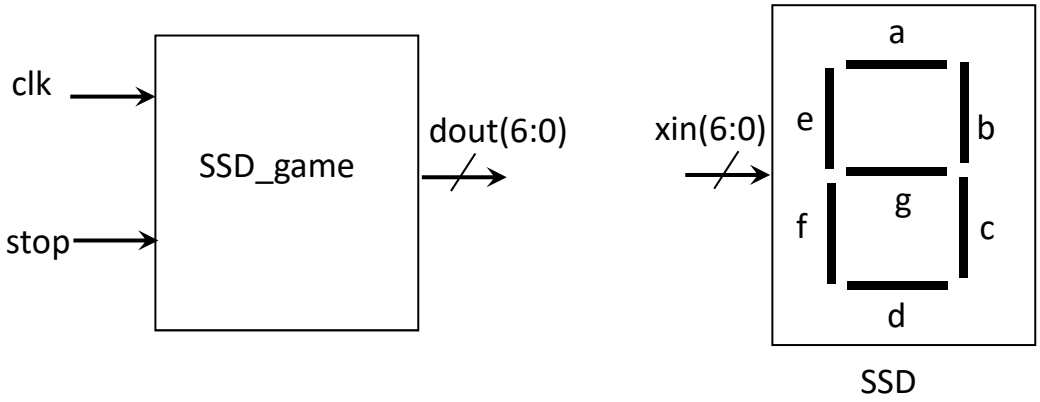
Código VHDL 1.2: Diseño del banco de pruebas del contador binario.

```
begin
   -- Instanciar y conectar UUT
   uut : component contadorBinario port map
                     (q, d, syn_clr, en, load, clk, reset);
   reset <=   '1',
              '0' after 15 ns;
   clk <= not Clk after (PERIODO/2);
   gen_vec_test : process is
      variable temp  : unsigned (3 downto 0);
      variable error_count : integer := 0; -- Núm. errores
   begin
      report "Comienza la simulación";
       -- Vectores de test y comprobación del resultado
       syn_clr <= '0';en <= '0';
       load <= '0';d <= "1111";
       wait for PERIODO;
       en <= '1';
      for i in 0 to 2 loop
        temp := TO_UNSIGNED(i,4);
        comprueba_salidas( std_logic_vector(temp), q, error_count);
        wait for PERIODO;
      end loop;
      load <= '1';d<= "1001";
      temp := "1001";
      en <= '0';
      wait for PERIODO;     -- 1
      comprueba_salidas( std_logic_vector(temp), Q, error_count);
      load <= '0';
      wait for PERIODO;
      -- Informe final
      if (error_count = 0) then
           report "Simulación finalizada sin errores";
      else
           report   "ERROR: Hay "                &
                     integer'image(error_count ) &
                     " errores.";
      end if;
      wait;        -- Final del bloque process
   end process gen_vec_test;
end architecture bp_contador;
```

Código VHDL 1.3: Continuación del banco de pruebas del contador.

PREGUNTA 4 (3 puntos)

Se quiere diseñar un circuito denominado SSD_game para poder conectarse con un display de siete segmentos (SSD). En la siguiente figura se muestra el circuito a diseñar y el SSD con el que se pretende conectar.



El circuito SSD_game es un circuito síncrono que opera en el flanco de subida de la señal de reloj. El circuito contiene dos señales de entrada: la señal de reloj `clk` y la señal asíncrona `stop`. El circuito tiene una señal de salida de 7 bits llamada `dout` que se puede conectar con el circuito SSD. Consideramos que la señal de reloj que se va a conectar a la entrada del circuito tiene una frecuencia de 1 kHz (periodo de 1 ms).

El objetivo de los bits 0 a 6 de la señal `dout` es iluminar (valor '1') o apagar (valor '0') los segmentos a, b, c, d, e, f y g del circuito SSD, respectivamente. Por ejemplo, cuando `dout`(6) = '1' el segmento g está iluminado y cuando `dout`(6) = '0' el segmento g está apagado.

Siempre que la señal `stop` tenga el valor '0', el circuito genera unos valores de la señal `dout` que hace que se enciendan cíclicamente con cada flanco de subida de la señal de reloj los siguientes segmentos del SSD (mientras el resto de segmentos se mantienen apagados): segmento a, segmentos b y c, segmento d, segmentos d y e, segmento e, segmentos e y f.

Si la señal `stop` tiene el valor '1', el circuito pasa asíncronamente a un estado en que sólo esté encendido el segmento a del display. El circuito permanece en ese estado hasta que la señal `stop` tenga el valor '0'.

El sistema tiene que permanecer en los estados donde sólo hay un segmento del display iluminado durante un tiempo de 80 ms, y en los estados donde hay dos segmentos del display iluminados durante 30 ms.

Programe en VHDL el circuito SSD_game describiendo su comportamiento como una máquina de estados de tipo Moore. Dibuje el diagrama de estado correspondiente al diseño realizado.

Solución a la Pregunta 4

En la Figura 1.3 se muestra el diagrama de estados del circuito SSD_game.

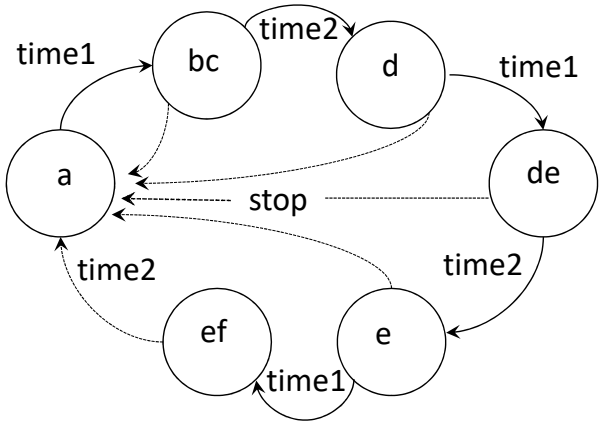


Figura 1.3: Diagrama de estados del circuito SSD_game.

El código VHDL del circuito SSD_game se muestra en Código VHDL 1.4.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity SSD_game is
   port(
        dout: out std_logic_vector(6 downto 0);
        clk, stop: in std_logic);
end entity SSD_game;
architecture SSD_game of SSD_game is
    constant time1: integer := 4; -- Valor 80 ms
    constant time2: integer := 2; -- Valor 30 ms
    type states is (a, bc,  d, de, e, ef);
    signal present_state, next_state : states;
    signal count: integer range 0 to 5;
    signal flip: bit;
begin
    process (clk, stop)
    begin
      if (stop = '1') then
          present_state <= a;
      elsif (rising_edge(clk)) then
          if ((flip = '1' and count = time1) or
                    (flip = '0' and count = time2)) then
             count <= 0;
             present_state <= next_state;
          else
             count <= count + 1;
          end if;
       end if;
    end process;
    process (present_state) is
    begin
      case present_state is
       when a =>
            dout <= "1000000";
            flip <= '1';
            next_state <= bc;
       when bc =>
            dout <= "0110000";
            flip <= '0';
            next_state <= d;
       when d =>
            dout <= "0001000";
            flip <= '1';
            next_state <= de;
       when de =>
            dout <= "0001100";
            flip <= '0';
            next_state <= e;
       when e =>
            dout <= "0000100";
            flip <= '1';
            next_state <= ef;
        when ef =>
            dout <= "0000110";
            flip <= '0';
            next_state <= a;
       end case;
     end process;
end architecture SSD_game;
```

Código VHDL 1.4: Diseño del circuito SSD_game.