INGENIERÍA DE COMPUTADORES 3

Solución al examen de Junio 2019, Primera Semana

PREGUNTA 1 (2 puntos)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `x1`, `x2`, `x3`, `x4`, `temp4` y `temp5` entre los instantes 0 y 100 ns.

```
use IEEE.std_logic_1164.all;
entity crono1 is
end entity crono1;

architecture crono1 of crono1 is
   signal x1, x2 : std_logic;
   signal x3, x4 : std_logic;
   signal temp4, temp5: std_logic;
begin
 process (x2)
    variable temp1, temp2: std_logic;
  begin
     temp1 := x2;
     temp2 := temp1;
     x3 <= temp2;
     temp4 <= x2;
     temp5 <= temp4;
     x4 <= temp5;
  end process;
  x1 <=  x3 after 20 ns;
  x2 <= '0',  '1' after 10 ns, '0' after 20 ns,
        '1' after 30 ns, '0' after 40 ns, '1' after 65 ns;
end architecture crono1;
```

Solución a la Pregunta 1

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

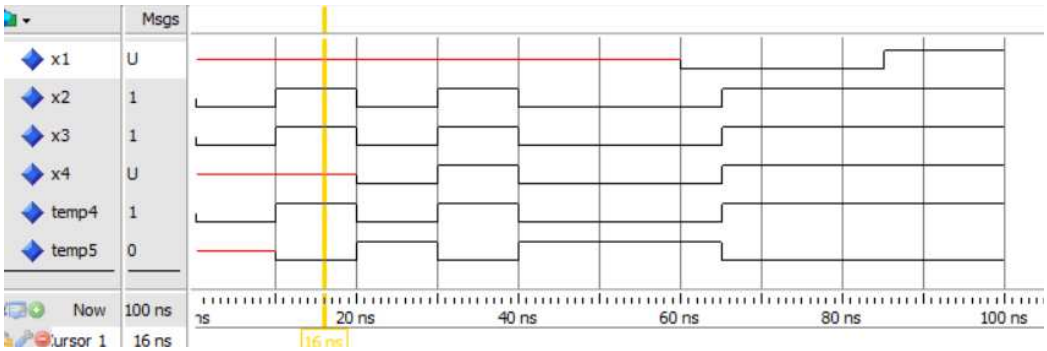


Figura 1.1: Cronograma de evolución de las señales.

PREGUNTA 2 (3 puntos)

Diseñe usando VHDL una calculadora aritmética descrita como un circuito secuencial síncrono que opera en el flanco de subida de la señal de reloj. El circuito tiene un registro interno de 8 bits y una señal de salida de 8 bits llamada `result`. La señal `result` tiene el mismo valor que el contenido del registro interno del circuito.

El circuito tiene las señales de entrada siguientes: una señal de entrada de datos de 8 bits llamada `dIn`, una señal de reloj `clk` y tres señales de control síncronas activas a nivel alto.

Las señales de control del circuito son las siguientes:

- `clear`: pone a '0' todos los bits del registro interno. Esta es la señal más prioritaria. Es decir, si está activa pone a '0' todos los bits del registro interno en el flanco de subida de la señal de reloj con independencia del valor del resto de señales de control.
- `load`: habilita la carga del valor de la señal de entrada `dIn` en el registro interno interpretados como binarios con signo. Esta es la segunda señal más prioritaria.
- `add`: habilita la carga en el registro interno del valor resultante de sumar el valor de la señal de entrada de datos `dIn` y el valor del registro interno. Esta es la señal menos prioritaria. Es decir, no habilita la carga del valor resultante de la suma si alguna de las otras señales de control tiene el valor '1'.

Escriba en VHDL la **architecture** que describe el comportamiento del circuito empleando únicamente un bloque **process** y sentencias concurrentes. Asimismo, en el diseño únicamente pueden emplearse los dos siguientes paquetes de la librería IEEE:

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

El circuito ha de tener la **entity** siguiente:

```
entity Calculadora is
   port( result: out std_logic_vector ( 7 downto 0);
         clk: in std_logic;
         clear, load, add  : in std_logic;
         dIn  : in std_logic_vector ( 7 downto 0));
end entity Calculadora;
```

Solución a la Pregunta 2

El código VHDL del circuito descrito en la Pregunta 2 se muestra en Código 1.1.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
architecture calculadora of Calculadora is
   signal dReg: std_logic_vector(7 downto 0);
begin
   process (clk) begin
    if rising_edge(clk) then
        if clear = '1' then
          dReg <= x"00";
        elsif load = '1' then
          dReg <= dIn;
        elsif add = '1' then
          dReg <= std_logic_vector(signed(dReg) + signed(dIn));
        end if;
     end if;
   end process;
   result <= dReg;
end calculadora;
```

Código VHDL 1.1: Architecture del circuito descrito en la Pregunta 2.

PREGUNTA 3 (3 puntos)

Diseñe un circuito secuencial síncrono para la regulación del semáforo de los vehículos en un paso de peatones. El semáforo de los vehículos tiene tres lámparas: verde, ámbar y rojo. La **entity** del circuito se muestra a continuación.

```
entity regulador is
   port (  rojo, verde, ambar : out std_logic;
           clk, reset, p   : in  std_logic );
end regulador;
```

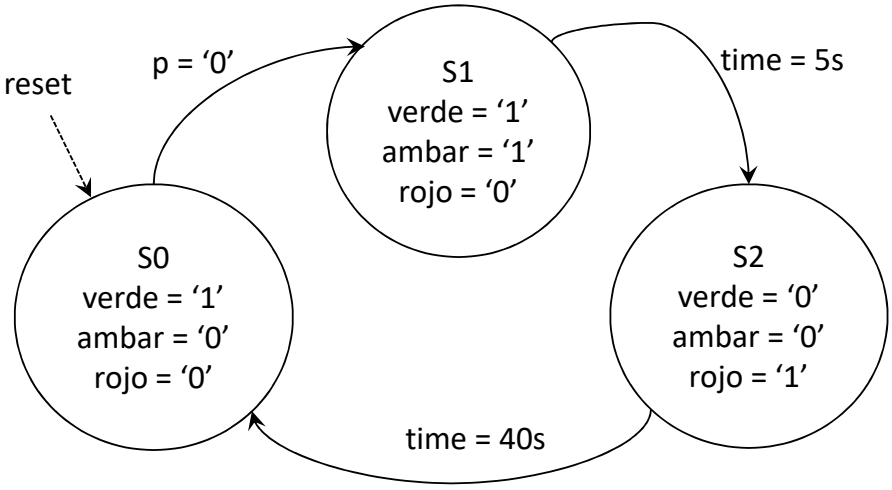
Las entradas al circuito son la señal de reloj (`clk`) de 1 Hz, la señal de reset asíncrona (`reset`) activa a nivel bajo y la señal síncrona activa a nivel bajo `p`. El valor de la señal `p` sólo se tiene en cuenta si únicamente está encendida la luz verde del semáforo de vehículos. Las señales de salida `rojo`, `verde` y `ambar` controlan, respectivamente, las lámparas roja, verde y ámbar del semáforo de vehículos. La lámpara está encendida cuando la señal que controla dicha lámpara está a '1' y está apagada cuando está a '0'. El circuito describe el siguiente comportamiento:

- La señal de reset enciende la luz verde del semáforo y apaga el resto de luces.
- La luz verde del semáforo de vehículos está encendida indefinidamente, siempre que la señal `p` tenga el valor '1'. Si la luz verde del semáforo está encendida y la señal `p` toma el valor '0', se realiza en el siguiente flanco de subida de la señal de reloj la siguiente acción: se enciende la luz ámbar sin apagar la luz verde.
- Las luces ámbar y verde deben permanecer encendidas 5 segundos. Transcurridos los 5 segundos se han de apagar ambas y encender la luz roja.
- La luz roja ha de permanecer encendida 40 segundos. Cuando se apaga la luz roja, se vuelve a encender la luz verde.

El circuito se ha de diseñar como una máquina de Moore que opera en el flanco de subida de la señal de reloj. Dibuje el diagrama de estados del circuito. Escriba el código VHDL de la **architecture** que describe el comportamiento del circuito siguiendo el anterior diagrama de estados.

Solución a la Pregunta 3

El diagrama de estados correspondiente al circuito regulador se muestra en la siguiente figura.



El código VHDL correspondiente al circuito regulador se muestra en Código VHDL 1.2.

```
Library IEEE;
use IEEE.std_logic_1164.all;
entity regulador is
    port (   rojo, verde, ambar : out std_logic;
             clk, reset, p     : in    std_logic );
end regulador;
architecture regulador of regulador is
     signal   internal_state: std_logic_vector(1 downto 0);
     constant timeVA   : integer := 5;
     constant timeR   : integer := 40;
begin
proximo_estado: process (clk, reset)
    variable count: integer range 0 to timeR;
begin
    if (reset = '0') then
        internal_state <= "00";--VERDE
        count := 0;
    elsif (rising_edge(clk)) then

        case internal_state is
            when "00" => --VERDE
              if (p='0') then
                 internal_state <= "01";
                 count := 0;
              end if;
            when "01" =>   --VERDE AMBAR
              if (count = timeVA) then
                  internal_state <= "10";
                  count := 0;
             end if;
            when "10" =>   -- ROJO
              if (count = timeR) then
                   internal_state <= "00";
                   count := 0;
              end if;
              when others =>
                   internal_state <= "00";
       end case;
          count := count + 1;
    end if;
end process proximo_estado;
salidas: process(internal_state)
begin
     case internal_state is
          when "00" =>
               verde <= '1'; rojo <= '0'; ambar <= '0';
          when "01" =>
               verde <= '1'; ambar <= '1'; rojo <= '0';
          when "10" =>
               verde <= '0'; ambar <= '0'; rojo <= '1';
          when others =>
               verde <='1'; rojo<='0'; ambar<='0';
     end case;
end process salidas;
end architecture regulador;
```

Código VHDL 1.2: Circuito regulador de semáforos.

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 código VHDL del banco de pruebas se muestra en Código VHDL 1.3–1.4.

```
-------------------------------------
-- Banco de pruebas del regulador de semaforos
library IEEE;
use IEEE.std_logic_1164.all;
entity bp_ctrl_semaforo is
end entity bp_ctrl_semaforo;
architecture bp_ctrl_semaforo of bp_ctrl_semaforo is
   signal    z              : std_logic_vector(2 downto 0);    -- Salidas UUT
   signal    clk            : std_logic  := '0';     -- Entradas UUT
   signal    reset, p         : std_logic;
   constant PERIODO : time       := 1 sec;
   constant timeVA  : integer := 5;
   constant timeR   : integer := 40;
   component regulador is
       port (   rojo, verde, ambar : out std_logic;
                clk, reset, p      : in   std_logic );
   end component regulador;
    -- Procedimiento para comprobar las salidas
    procedure comprueba_salidas
        ( esperado_z            :        std_logic_vector(2 downto 0);
          actual_z               :        std_logic_vector(2 downto 0);
          error_count            : inout integer) is
    begin
         if ( esperado_z /= actual_z ) then
             report "ERROR: Estado esperado ("      &
                      std_logic'image(esperado_z(2))    &
                     std_logic'image(esperado_z(1))    &
                     std_logic'image(esperado_z(0))    &
                     "), estado actual ("             &
                     std_logic'image(actual_z(2))    &
                     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
```

Código VHDL 1.3: Diseño del banco de pruebas del regulador de semáforos (1/2).

```
  -- Instanciar y conectar UUT
  uut : component regulador port map
                      ( z(2), z(1), z(0),  clk, reset, p);
  reset <=   '0',
             '1' after (PERIODO);
  clk <= not clk after (PERIODO/2);
  p <= '1', '0' after (PERIODO), '1' after (2*PERIODO);
  gen_vec_test : process is
    variable error_count : integer := 0; -- Núm. errores
  begin
    report "Comienza la simulación";
     -- Vectores de test y comprobación del resultado
      wait for PERIODO;     -- 1
      comprueba_salidas("010", z, error_count);
      wait for PERIODO; -- Accion tras pulsar P
      comprueba_salidas("011", z, error_count);
      wait for timeVA*PERIODO;     -- 3
      comprueba_salidas("100", z, error_count);
      wait for timeR*PERIODO;     -- 4
      comprueba_salidas("010", z, error_count);
    -- Informe final
    if (error_count = 0) then
        report "Simulación finalizada sin errores";
    else
        report   "ERROR: Hay "                &
                  integer'image(error_count) &
                  " errores.";
    end if;
    wait;
  end process gen_vec_test;
end architecture bp_ctrl_semaforo;
```

Código VHDL 1.4: Continuación del banco de pruebas del regulador de semáforos (2/2).