Código asignatura	Nombre asignatura
71012018	**Ingeniería de Computadores III**
Fecha alta y origen	Convocatoria
04/11/2015	Junio 2014 (1ª Semana)
Página Asignatura	

INGENIERÍA DE COMPUTADORES 3

Solución al examen de Junio 2014, 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`, `x5` entre los instantes 0 y 60 ns.

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

architecture crono1 of crono1 is
   signal x1, x2, x3, x4 : std_logic;
   signal x5 : std_logic := '0';
begin
 process (x1, x5)
    variable temp  : std_logic;
  begin
     temp := x1;
     x2 <= x1;
     x3 <= x2;
     x4 <= temp;
 end process;
 x1 <= '0', '1' after 10 ns, '0' after 20 ns,
       '1' after 30 ns;
 x5 <= x1 after 15 ns;
end architecture crono1;
```

Solución a la Pregunta 1

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

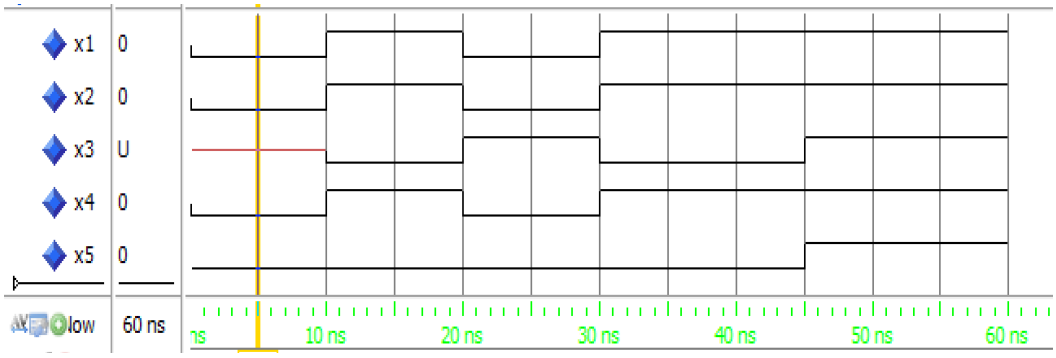


Figura 1.1: Cronograma de evolución de las señales.

PREGUNTA 2 (2.5 puntos)

Escriba en VHDL, de las dos formas que se detallan a continuación, la **architecture** que describe el comportamiento de un circuito combinacional desplazador a la izquierda. El circuito ha de desplazar la señal de entrada a la izquierda de 0 a 7 bits rellenado con '0'. En la descripción del circuito no se pueden emplear ni operadores ni funciones de desplazamiento lógico. La **entity** del circuito se muestra a continuación.

```
entity desplazador is
   port ( y   : out std_logic_vector(7 downto 0);
          a   : in  std_logic_vector(7 downto 0);
          ctrl  : in  std_logic_vector(2 downto 0) );
end entity desplazador;
```

La señal `a` es la señal a desplazar y el valor de la señal `ctrl` indica el número de bits a desplazar.

2.a) (1.25 puntos) Empleando una sentencia concurrente condicional (**when - else**). Dibuje el diagrama conceptual del circuito diseñado.

2.b) (1.25 puntos) Empleando una asignación concurrente de selección (**with - select**). Dibuje el diagrama conceptual del circuito diseñado.

Solución a la Pregunta 2

Las dos **architecture** que describen el comportamiento de un circuito combinacional desplazador se muestran en Código VHDL 1.1–1.2.

Los diagramas conceptuales a los que dan lugar estos dos diseños se muestran respectivamente en las Figura 1.2–1.3.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity desplazador is
      port ( y   : out std_logic_vector(7 downto 0);
             a   : in  std_logic_vector(7 downto 0);
             ctrl  : in  std_logic_vector(2 downto 0) );
end entity desplazador;

architecture desplazador of desplazador is
begin
 y <= a when (ctrl = "000")
      else a(6 downto 0)&'0' when (ctrl = "001")
      else a(5 downto 0)&"00" when (ctrl = "010")
      else a(4 downto 0)&"000" when (ctrl = "011")
      else a(3 downto 0)&"0000" when (ctrl = "100")
      else a(2 downto 0)&"00000" when (ctrl = "101")
      else a(1 downto 0)&"000000" when (ctrl = "110")
      else a(0)&"0000000";
 end architecture desplazador;
```

Código VHDL 1.1: Descripción del comportamiento del desplazador empleando una sentencia concurrente condicional (**when** - **else**).

```
library IEEE;
use IEEE.std_logic_1164.all;
entity desplazador2 is
      port ( y   : out std_logic_vector(7 downto 0);
             a   : in  std_logic_vector(7 downto 0);
             ctrl  : in  std_logic_vector(2 downto 0) );
end entity desplazador2;

architecture desplazador2 of desplazador2 is
begin
 with ctrl select
      y <= a when "000",
           a(6 downto 0)&'0' when "001",
           a(5 downto 0)&"00" when "010",
           a(4 downto 0)&"000" when "011",
           a(3 downto 0)&"0000" when "100",
           a(2 downto 0)&"00000" when "101",
           a(1 downto 0)&"000000" when "110",
           a(0)&"0000000" when others;
 end architecture desplazador2;
```

Código VHDL 1.2: Descripción del comportamiento del desplazador empleando una sentencia concurrente de selección (**with** - **select**).

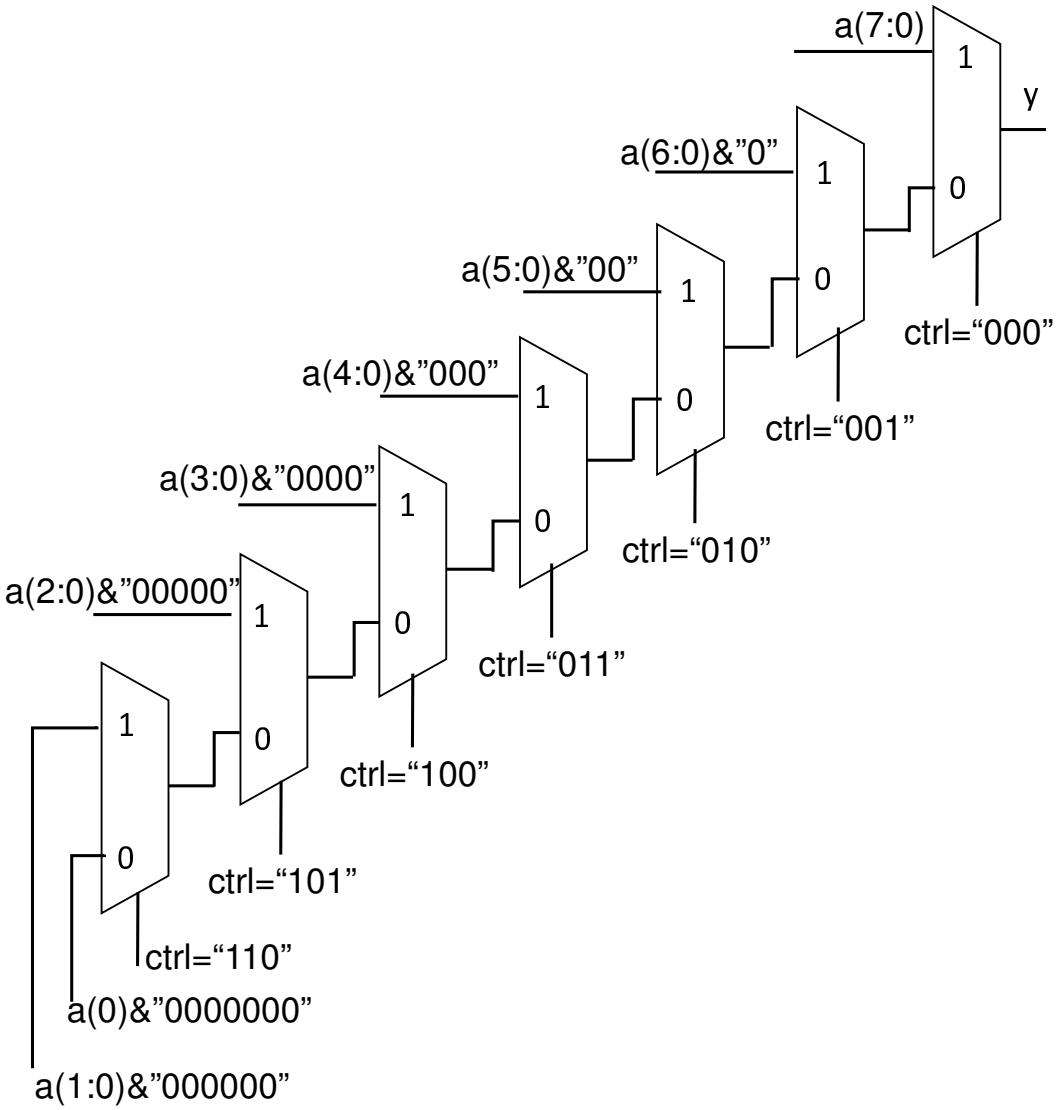


Figura 1.2: Diagrama conceptual del circuito implementado usando una sentencia concurrente condicional.

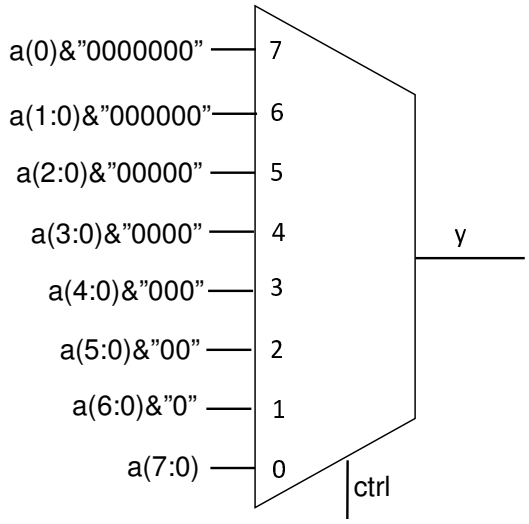


Figura 1.3: Diagrama conceptual del circuito implementado usando una sentencia concurrente de selección.

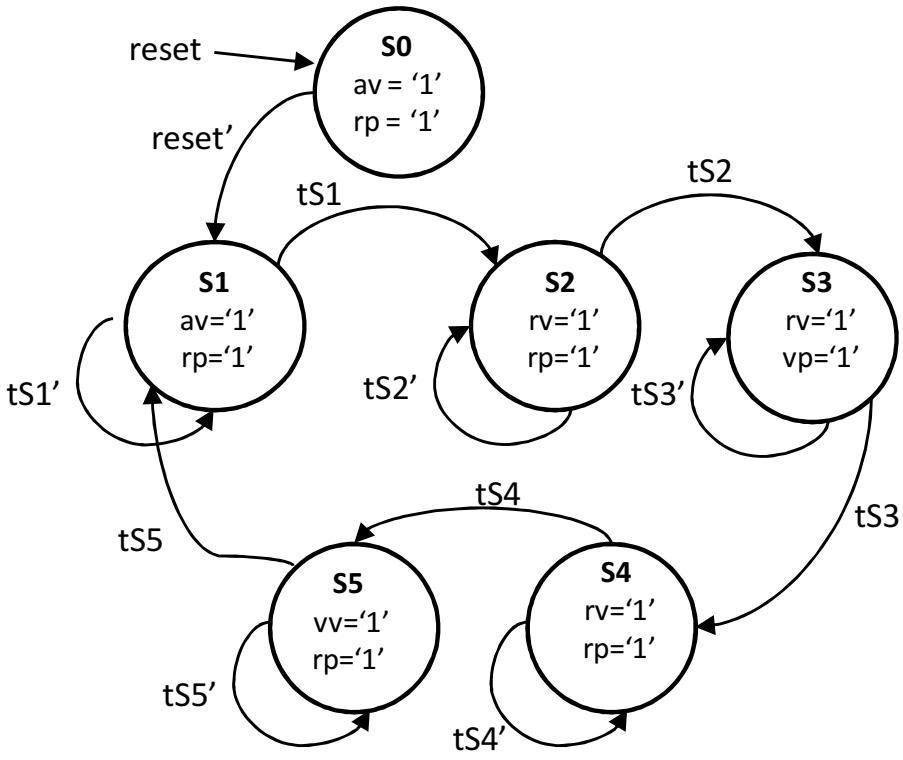
PREGUNTA 3 (3.5 puntos)

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

```
entity regulador is
   port (  rv, av, vv, rp, vp : out std_logic;
           clk, reset   : in  std_logic );
end regulador;
```

Las entradas al circuito son la señal de reloj (`clk`) de 60 Hz y la señal de reset (`reset`) asíncrona y activa a nivel alto. Las señales de salida `rv`, `av`, `vv`, `rp`, `vp` controlan, respectivamente, las lámparas roja, amarilla y verde del semáforo de vehículos y las lámparas roja y verde del semáforo de peatones. La lámpara está encendida cuando la señal que controla dicha lámpara está a '1' y está apagada cuando está a '0'.

El circuito funciona como una máquina de 6 estados cuyo diagrama de estados se muestra en la siguiente figura. Las transiciones entre estados se producen en el flanco de subida de la señal de reloj. En cada estado se indican las únicas señales de salida que valen '1' en dicho estado.



- Mientras la señal de `reset` esté a '1' el circuito permanece en el estado S0. En el estado S0 sólo están encendidas la lámpara amarilla del semáforo de vehículos y la roja del semáforo de peatones.
- En el estado S1 permanece un tiempo tS1 (5 s) y sólo están encendidas la lámpara amarilla del semáforo de vehículos y la roja del semáforo de peatones.
- En el estado S2 permanece un tiempo tS2 (5 s) y sólo están encendidas la lámpara roja del semáforo de vehículos y la roja del semáforo de peatones.
- En el estado S3 permanece un tiempo tS3 (45 s) y sólo están encendidas la lámpara roja del semáforo de vehículos y la verde del semáforo de peatones.
- En el estado S4 permanece un tiempo tS4 (10 s) y sólo están encendidas la lámpara roja del semáforo de vehículos y la roja del semáforo de peatones.
- En el estado S5 permanece un tiempo tS5 (45 s) y sólo están encendidas la lámpara verde del semáforo de vehículos y la roja del semáforo de peatones.

Escriba el código VHDL de la **architecture** que describe el comportamiento del circuito siguiendo el diagrama de estados mostrado anteriormente.

Solución a la Pregunta 3

El código VHDL del circuito regulador se muestra en Código VHDL 1.3–1.5.

```
Library IEEE;
use IEEE.std_logic_1164.all;
entity regulador is
   port (  rv, av, vv, rp, vp : out std_logic;
           clk, reset   : in  std_logic );
end regulador;
architecture regulador of regulador is
    constant PERIODO : time       := 16667 us;   --60 Hz
    constant timeMAX : integer := 2700; --45s*60ciclos/s
    constant tS1  : integer := 300; --5s*60ciclos/s
    constant tS2  : integer := 300;  --5s*60ciclos/s
    constant tS3  : integer := 2700;  --45s*60ciclos/s
    constant tS4  : integer := 600; --10*60ciclos/s
    constant tS5  : integer := 2700; --45*60ciclos/s
    type type_state is (S0, S1, S2, S3, S4, S5);
    signal  internal_state: type_state;
begin
```

Código VHDL 1.3: Diseño del circuito regulador.

```
proximo_estado: process (clk, reset)
   variable count: integer range 0 to timeMAX;
begin
   if (reset = '1') then
        internal_state <= S0;
        count := 0;
   elsif (rising_edge(clk)) then
        case internal_state is
             when S0 =>
               internal_state <= S1;
               count := 0;
             when S1 =>
               if (count = tS1) then
                  internal_state <= S2;
                  count := 0;
              end if;
             when S2 =>
               if (count = tS2) then
                    internal_state <= S3;
                    count := 0;
               end if;
             when S3 =>
               if (count = tS3) then
                    internal_state <= S4;
                    count := 0;
               end if;
             when S4 =>
                 if (count = tS4) then
                    internal_state <= S5;
                    count := 0;
               end if;
             when S5 =>
                 if (count = tS5) then
                    internal_state <= S1;
                    count := 0;
               end if;
       end case;
          count := count + 1;
   end if;
end process proximo_estado;
```

Código VHDL 1.4: Continuación del diseño del circuito regulador.

```
salidas: process(internal_state)
begin
   rv <= '0'; vv <= '0'; av <= '0'; rp <= '0'; vp <= '0';
   case internal_state is
        when S0 =>
             av <= '1';  rp <= '1';
        when S1 =>
             av <= '1';  rp <= '1';
        when S2 =>
             rv <= '1';  rp <= '1';
        when S3 =>
             rv <= '1';  vp <= '1';
        when S4 =>
             rv <= '1';  rp <= '1';
        when S5 =>
             vv <= '1';  rp <= '1';
    end case;
end process salidas;

end architecture regulador;
```

Código VHDL 1.5: Final del diseño del circuito regulador.

PREGUNTA 4 (2 puntos)

Programe en VHDL un banco de pruebas para el circuito que ha diseñado al contestar a la Pregunta 3. El banco de pruebas ha de resetear el circuito y comprobar que se producen correctamente las transiciones de estado hasta alcanzar el estado S5. El banco de pruebas debe comprobar que los valores de las señales de salida 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 regulador se muestra en Código VHDL 1.6–1.7.

```
-------------------------------------
-- Banco de pruebas del regulador de semáforos
library IEEE;
use IEEE.std_logic_1164.all;

entity bp_regulador is
end entity bp_regulador;

architecture bp_regulador of bp_regulador is
  signal    z              : std_logic_vector(4 downto 0);    -- Salidas UUT
  signal    clk            : std_logic  := '0';     -- Entradas UUT
  signal    reset          : std_logic := '1';
  constant PERIODO : time       := 16667 us;  --60 Hz
  constant timeMAX : integer := 2700; --45s*60ciclos/s
  constant tS1  : integer := 300; --5s*60ciclos/s
  constant tS2  : integer := 300;  --5s*60ciclos/s
  constant tS3  : integer := 2700;  --45s*60ciclos/s
  constant tS4  : integer := 600; --10*60ciclos/s
  constant tS5  : integer := 2700; --45*60ciclos/s
  type type_state is (S0, S1, S2, S3, S4, S5);

  component regulador is
   port (  rv, av, vv, rp, vp : out std_logic;
           clk, reset   : in  std_logic );
  end component regulador;

   -- Procedimiento para comprobar las salidas
   procedure comprueba_salidas
      ( esperado_z             :       std_logic_vector(4 downto 0);
        actual_z               :       std_logic_vector(4 downto 0);
        error_count            : inout integer) is
   begin
        -- Comprueba q
        if ( esperado_z /= actual_z ) then
           report "ERROR: Estado esperado ("     &
                  std_logic'image(esperado_z(4))    &
                  std_logic'image(esperado_z(3))    &
                  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(4))    &
                  std_logic'image(actual_z(3))    &
                  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 1.6: Banco de pruebas del regulador.

```
  -- Instanciar y conectar UUT
  uut : component regulador port map
                  (z(4), z(3), z(2), z(1), z(0),  clk, reset);

  reset <=   '1',
             '0' after (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
      wait for PERIODO/10;    -- 2
      comprueba_salidas("01010", z, error_count);
      wait for PERIODO;    -- 2
      comprueba_salidas("01010", z, error_count);
      wait for (tS1)*PERIODO;    -- 3
      comprueba_salidas("10010", z, error_count);
      wait for tS2*PERIODO;    -- 4
      comprueba_salidas("10001", z, error_count);
      wait for tS3*PERIODO;    -- 5
      comprueba_salidas("10010", z, error_count);
      wait for tS4*PERIODO;    -- 6
      comprueba_salidas("00110", z, error_count);
      wait for tS5*PERIODO;    -- 7
      comprueba_salidas("01010", 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;        -- Final del bloque process, pero como
                 -- la architecture tiene sentencias de
                 -- asignación concurrente, esta sentencia
                 -- wait no finaliza la simulación
  end process gen_vec_test;
end architecture bp_regulador;
```

Código VHDL 1.7: Continuación del banco de pruebas del regulador.