Código asignatura	Nombre asignatura
71012018	**Ingeniería de Computadores III**
Fecha alta y origen	Convocatoria
04/11/2015	Septiembre 2013 (Original)
Página Asignatura	

INGENIERÍA DE COMPUTADORES III

Solución al examen de Septiembre 2013

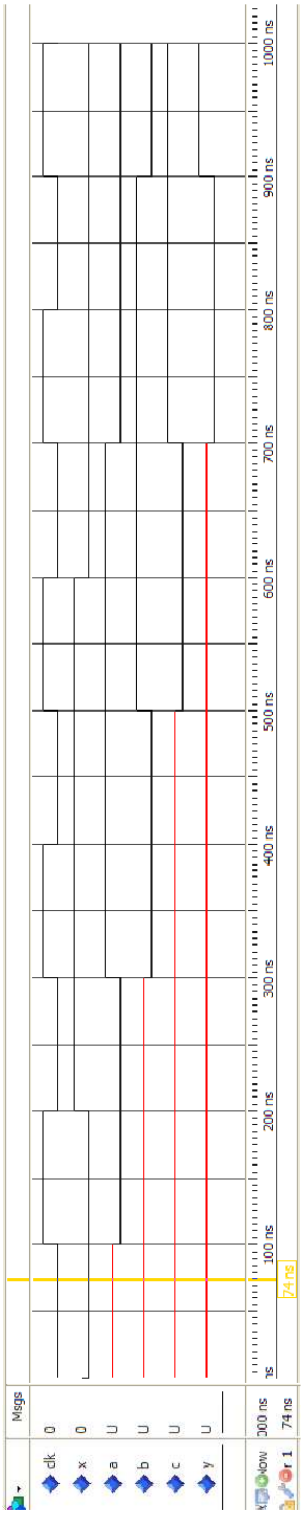
PREGUNTA 1 (2 puntos)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `clk`, `x`, `a`, `b`, `c` e `y` entre los instantes 0 y 1000 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono1 is
end entity crono1;
architecture crono1 of crono1 is
  signal a, b, c, x, y: std_logic;
  constant PERIODO : time       := 200 ns;
  signal clk : std_logic:='0';
begin
   process (clk)
   begin
      if ( rising_edge(clk) ) then
         a <= x; b <= a; c <= b; y <= c;
      end if;
   end process;
  clk <= not clk after (PERIODO/2);
  gen_vec_test : process is
  begin
    x<='0'; wait until falling_edge(clk);
    x<='1'; wait until falling_edge(clk);
    x<='1'; wait until falling_edge(clk);
    x<='0'; wait until falling_edge(clk);
    wait;
  end process gen_vec_test;
end architecture crono1;
```

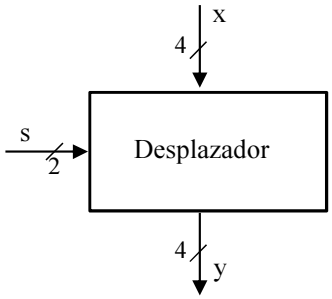
Solución a la Pregunta 1

En la siguiente figura se muestra el cronograma solución a la Pregunta 1.



PREGUNTA 2 (3 puntos)

A continuación, se muestra el símbolo lógico y la tabla de operaciones de un circuito combinacional desplazador.



s(1)	s(0)	y(3)	y(2)	y(1)	y(0)	Operación
0	0	x(2)	x(1)	x(0)	'0'	Desplaza a la izquierda rellenando con 0
0	1	'1'	x(3)	x(2)	x(1)	Desplaza a la derecha rellenando con 1
1	0	x(2)	x(1)	x(0)	x(3)	Rota a la izquierda
1	1	x(0)	x(3)	x(2)	x(1)	Rota a la derecha

2.a) (0.25 puntos) Escriba en VHDL la **entity** y la **architecture** que describe el comportamiento de una puerta NOT.

2.b) (0.25 puntos) Escriba en VHDL la **entity** y la **architecture** que describe el comportamiento de una puerta AND de 3 entradas.

2.c) (0.25 puntos) Escriba en VHDL la **entity** y la **architecture** que describe el comportamiento de una puerta OR de 2 entradas.

2.d) (0.25 puntos) Dibuje el diagrama a nivel de puertas lógicas del circuito combinacional desplazador. Emplee para ello únicamente puertas NOT, puertas AND de 3 entradas y puertas OR de 2 entradas.

2.e) (2 puntos) Escriba en VHDL la **entity** y la **architecture** que describe la estructura del circuito desplazador siguiendo el diagrama dibujado en el apartado anterior y empleando las puertas lógicas cuyo diseño ha realizado al contestar los tres primeros apartados. Las señales de entrada y salida del circuito desplazador han de tener los mismos nombres que los mostrados en la figura del símbolo lógico del circuito desplazador.

Solución a la Pregunta 2

El código correspondiente a las puertas lógicas NOT, AND y OR se muestra en Código VHDL 1.1–1.3.

```
library IEEE;
use IEEE.std_logic_1164.all;

entity not1 is
    port ( y       : out std_logic;
           x       : in std_logic );
end entity not1;

architecture not1 of not1 is
begin
     y <= not x;
end architecture not1;
```

Código VHDL 1.1: Puerta lógica NOT.

```
library IEEE;
use IEEE.std_logic_1164.all;

entity and3 is
    port ( y       : out std_logic;
           x0, x1, x2 : in std_logic );
end entity and3;

architecture and3 of and3 is
begin
     y <= x2 and x1 and x0;
end architecture and3;
```

Código VHDL 1.2: Puerta lógica AND de 3 entradas.

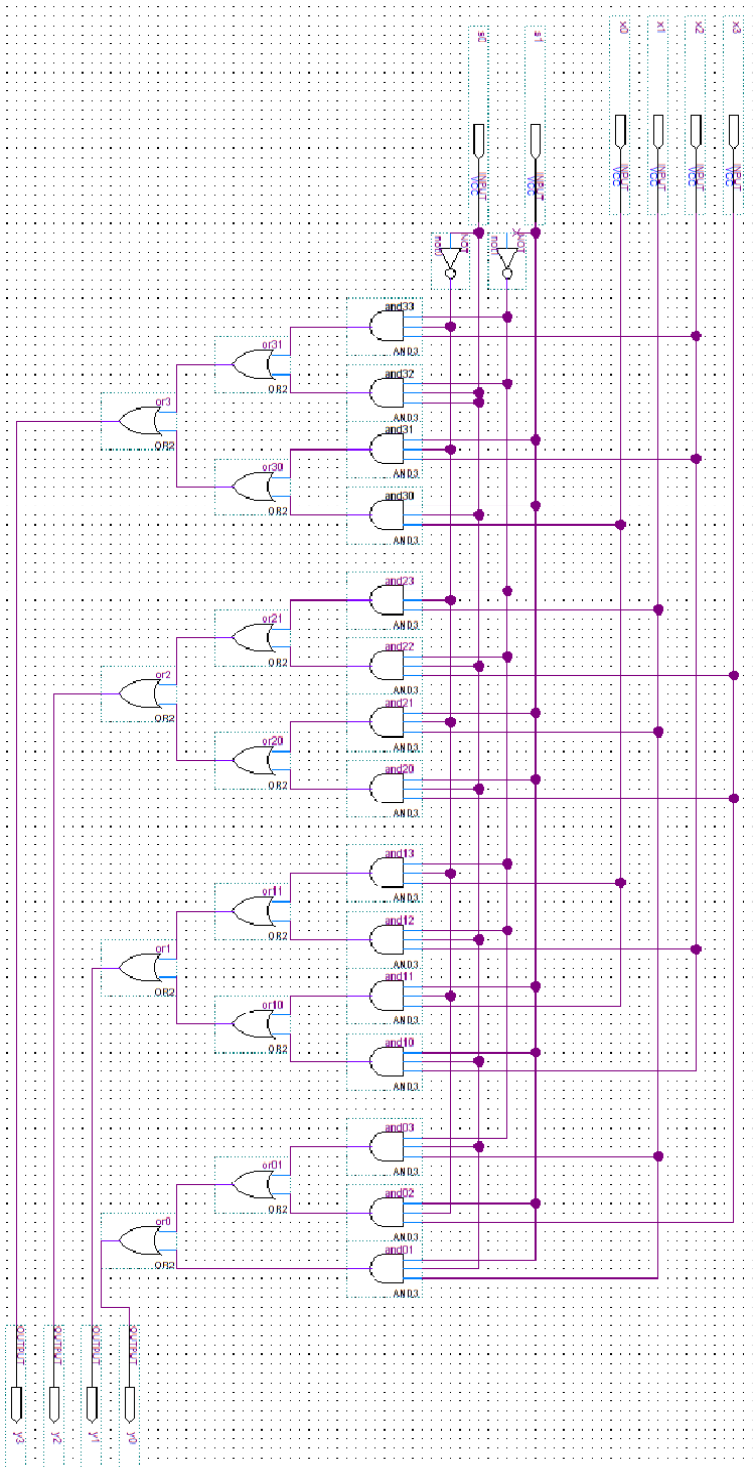
```
library IEEE;
use IEEE.std_logic_1164.all;

entity or2 is
    port ( y       : out std_logic;
           x1, x0 : in std_logic );
end entity or2;

architecture or2 of or2 is
begin
     y <= x1 or x0;
end architecture or2;
```

Código VHDL 1.3: Puerta lógica OR de 2 entradas.

A continuación se muestra la figura del diagrama circuital del circuito desplazador. El código VHDL de dicho circuito se muestra en Código VHDL 1.4–1.5.


s0
s1
x0
x1
x2
x3
and33
and32
and31
and30
and23
and22
and21
and20
and13
and12
and11
and10
and03
and02
and01
or31
or30
or3
or21
or20
or2
or11
or10
or1
or01
or0
AND3
OR2

```
-------------------------------------------------------
---Desplazador
library IEEE;
use IEEE.std_logic_1164.all;

entity desplazador is
   port(y : out std_logic_vector(3 downto 0);
        x : in std_logic_vector(3 downto 0);
        s : in std_logic_vector(1 downto 0));
end entity desplazador;

architecture desplazador of desplazador is
   signal s1n, s0n : std_logic;
   signal sand33, sand32, sand31, sand30 : std_logic;
   signal sand23, sand22, sand21, sand20 : std_logic;
   signal sand13, sand12, sand11, sand10 : std_logic;
   signal sand03, sand02, sand01 : std_logic;
   signal sor31, sor30 : std_logic;
   signal sor21, sor20 : std_logic;
   signal sor11, sor10 : std_logic;
   signal sor01 : std_logic;
   signal sor3, sor2, sor1, sor0 : std_logic;

   component and3 is port
   ( y          : out std_logic;
     x0, x1, x2 : in  std_logic );
   end component and3;

   component or2 is port
   ( y       : out std_logic;
     x0, x1 : in  std_logic );
   end component or2;

   component not1 is port
   ( y : out std_logic;
     x : in  std_logic );
   end component not1;
begin
   not_1: not1 port map (s1n, s(1));
   not_0: not1 port map (s0n, s(0));

   and33: and3 port map (sand33, s1n, s0n, x(2));
   and32: and3 port map (sand32, s1n, s(0), s(0));
   and31: and3 port map (sand31, s(1), s0n, x(2));
   and30: and3 port map (sand30, s(1), s(0), x(0));

   and23: and3 port map (sand23, s1n, s0n, x(1));
   and22: and3 port map (sand22, s1n, s(0), x(3));
   and21: and3 port map (sand21, s(1), s0n, x(1));
   and20: and3 port map (sand20, s(1), s(0), x(3));

   and13: and3 port map (sand13, s1n, s0n, x(0));
   and12: and3 port map (sand12, s1n, s(0), x(2));
   and11: and3 port map (sand11, s(1), s0n, x(0));
   and10: and3 port map (sand10, s(1), s(0), x(2));

   and03: and3 port map (sand03, s1n, s(0), x(1));
   and02: and3 port map (sand02, s(1), s0n, x(3));
   and01: and3 port map (sand01, s(1), s(0), x(1));
-------------------------------------------
```

Código VHDL 1.4: Circuito desplazador.

```
-------------------------------------------

   or31: or2 port map (sor31, sand33, sand32);
   or30: or2 port map (sor30, sand31, sand30);

   or21: or2 port map (sor21, sand23, sand22);
   or20: or2 port map (sor20, sand21, sand20);

   or11: or2 port map (sor11, sand13, sand12);
   or10: or2 port map (sor10, sand11, sand10);

   or01: or2 port map (sor01, sand03, sand02);

   or3: or2 port map (y(3), sor31, sor30);
   or_2: or2 port map (y(2), sor21, sor20);
   or1: or2 port map (y(1), sor11, sor10);
   or0: or2 port map (y(0), sor01, sand01);

end architecture desplazador;
-------------------------------------------
```

Código VHDL 1.5: Continuación del código del circuito desplazador.

PREGUNTA 3 (2 puntos)

Programe en VHDL el banco de pruebas del circuito combinacional que ha diseñado al contestar a la Pregunta 2.e). 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 3

El código VHDL del banco de pruebas del circuito se muestra en Código VHDL 1.6–1.7.

```
-------------------------------------------------
-- Banco de pruebas del desplazador
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity bp_desplazador is
    constant DELAY      : time      := 10 ns; -- Retardo usado en el test
end entity bp_desplazador;

architecture bp_desplazador of bp_desplazador is
    signal y   : std_logic_vector(3 downto 0); --Salida UUT
    signal s         : std_logic_vector(1 downto 0); --Entradas UUT
    signal x  : std_logic_vector(3 downto 0);

    component desplazador is
        port ( y   : out    std_logic_vector(3 downto 0);
                  x          : in std_logic_vector(3 downto 0);
                  s  : in std_logic_vector(1 downto 0));
    end component desplazador;

   -- Procedure que calcula la salida (expected_s) y lo compara con el
   -- valor de salida que se pasa como argumento (actual_s)
   -- Si ambos valores no coinciden, se muestra un mensaje y se
   -- incrementa el contador de errores (error_count)
   procedure check_desp
       ( s             : in integer;
         x          : in std_logic_vector(3 downto 0);
         actual_y     : in std_logic_vector (3       downto 0);
         error_count: inout integer ) is
     variable expected_y : std_logic_vector(3 downto 0);
   begin
     case s is
          when 0        => expected_y := x(2)&x(1)&x(0)&'0';
          when 1        => expected_y := '1'&x(3)&x(2)&x(1);
          when 2        => expected_y := x(2)&x(1)&x(0)&x(3);
          when others => expected_y := x(0)&x(3)&x(2)&x(1);
     end case;
      assert( expected_y = actual_y)
        report "ERROR. Entrada: "  & integer'image(to_integer(unsigned(x))) &
                ", Operacion: "  & integer'image(s) &
                ", resultado esperado: " &
                integer'image(to_integer(unsigned(expected_y))) &
                ", resultado actual: " &
                integer'image(to_integer(unsigned(actual_y))) &
                " en el instante "&
                time'image(now);

     if (expected_y /= actual_y) then
       error_count := error_count + 1;
     end if;

   end procedure check_desp;
   -- Fin de la definición del procedure

-------------------------------------------------
```

Código VHDL 1.6: Banco de pruebas del circuito desplazador.

```
-------------------------------------------------
begin
  UUT : component desplazador port map
        (y, x, s);

vec_test : process is
    variable temp   : std_logic_vector (5 downto 0);
    variable error_count : integer := 0;
  begin
    report "Comienza la simulación";
    -- Generar todos los posibles valores de entrada
    for i in 0 to 2**6-1 loop
      temp := std_logic_vector(to_unsigned(i,6));
      s    <= temp(5 downto 4);
      x    <= temp(3 downto 0);
      wait for DELAY;

      check_desp(to_integer(unsigned(s)),x,y ,error_count);

    end loop;

     -- Informe mostrando el resultado del test
    report  "Finaliza la simulación:  " &  integer'image(error_count)  &  "
errores";
    wait;  -- Termina la simulación
  end process vec_test;
end architecture bp_desplazador;
-------------------------------------------------
```

Código VHDL 1.7: Continuación del banco de pruebas del circuito desplazador.

PREGUNTA 4 (3 puntos)

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

4.a) (1.5 puntos) Un latch D con entrada enable activa a nivel alto y una señal de reset asíncrona activa a nivel bajo, cuya **entity** se muestra a continuación. El latch tiene tres señales de entrada: `D`, `Enable` y `Reset`. El latch tiene una única señal de salida (`Q`), cuyo valor es el del estado del circuito. El latch tiene una señal de reset asíncrona activa a nivel bajo (`Reset`). La señal `Enable` habilita o deshabilita la carga del latch. Si la señal `Enable` vale '0' la carga del latch desde la entrada `D` está deshabilitada. Por el contrario, si la señal `Enable` vale '1' se asigna el valor de la entrada `D` a la salida `Q`.

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

4.b) (1.5 puntos) Un flip-flop D, disparado por el flanco de subida del reloj. Además de la entrada de reloj (`clk`), el flip-flop tiene otras tres señales de entrada: `D`, `Enable` y `Reset`. El flip-flop tiene una única señal de salida (`Q`), cuyo valor es el del estado del circuito. El flip-flop tiene una señal de reset síncrona activa a nivel alto (`Reset`). La señal `Enable` habilita o deshabilita la carga del flip-flop. Si la señal `Enable` vale '0' la carga del flip-flop desde la entrada `D` está deshabilitada. Por el contrario, si la señal `Enable` vale '1' se asigna el valor de la entrada `D` a la salida `Q` en el flanco de subida de la señal de reloj.

La **entity** del flip-flop D se muestra a continuación.

```
entity D_biestable is
port (                      Q: out std_logic;
        D, clk, Reset, Enable: in std_logic);
end entity D_biestable;
```

Solución a la Pregunta 4

La **architecture** de la latch D y el flip-flop D se muestran respectivamente en Código VHDL 1.8–1.9.

```
library IEEE;
use IEEE.std_logic_1164.all;
architecture behav of D_latch is
begin
   process (Enable, D, Reset) begin
      if (reset = '0') then
         Q <= '0';
      elsif (enable = '1') then
         Q <= D;
      end if;
   end process;
end behav;
```

Código VHDL 1.8: Latch D.

```
library IEEE;
use IEEE.std_logic_1164.all;
architecture behav of D_biestable is
begin
   process (clk) begin
      if (rising_edge(clk)) then
         if (Reset = '1') then
            Q <= '0';
         elsif (Enable = '1') then
            Q <= D;
         end if;
      end if;
   end process;
end behav;
```

Código VHDL 1.9: Flip-flop D.