Código asignatura	Nombre asignatura
71012018	**Ingeniería de Computadores III**
Fecha alta y origen	Convocatoria
04/11/2015	Junio 2012 (2ª Semana)
Página Asignatura	

INGENIERÍA DE COMPUTADORES III

Solución al examen de Junio 2012, Segunda 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 50 ns.

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

architecture cronoW of cronoW is
  signal x1, x2, x3, x4, x5 : std_logic;
begin
  x1 <= '1', '0' after 10 ns,
        '1' after 20 ns, '0' after 30 ns, '1' after 40 ns;
  x2 <= '0', '1' after 25 ns;
  Proc1: process
     variable valor  : std_logic;
  begin
    wait for 10 ns;
    x3 <= x1;
    for i in 0 to 3 loop
       valor := x1 or x2;
       x4 <= valor;
       x5 <= x4;
       wait for 10 ns;
     end loop;
     wait;
  end process;
end architecture cronoW;
```

Solución a la Pregunta 1

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

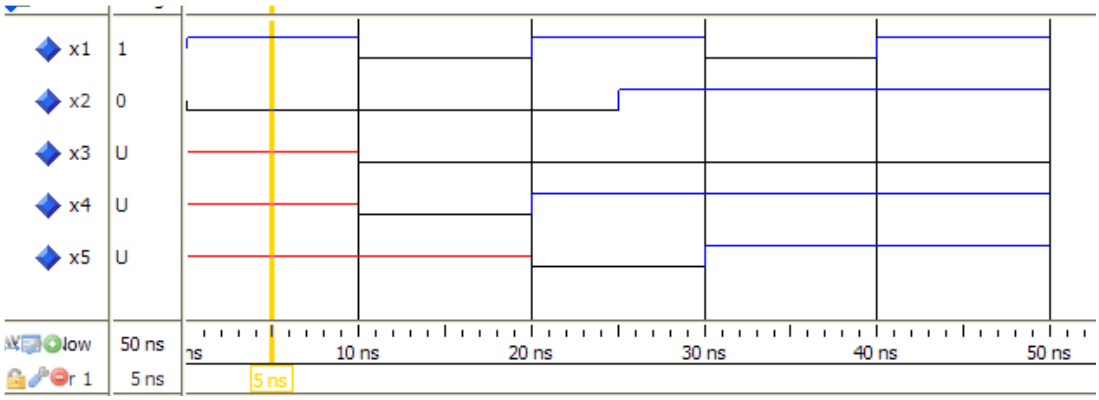


Figura 1.1: Cronograma.

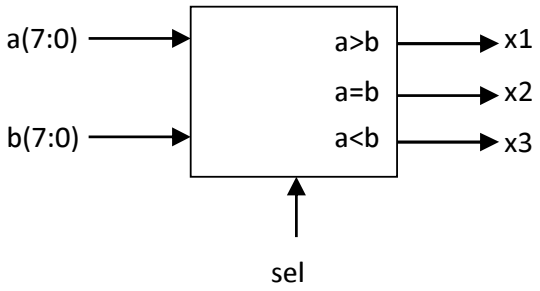
En este diseño existen dos asignaciones concurrentes a las señales `x1` y `x2`. Dado que ninguna de estas dos señales se ha inicializado, en $t = 0$ ambas señales valen 'U'. La señal `x1` actualiza su valor en los instantes δ, 10 ns, 20 ns, 30 ns y 40 ns a los valores '1', '0', '1', '0' y '1', respectivamente. La señal `x2` actualiza su valor en los instantes δ y 25 ns a los valores '0' y '1', respectivamente.

Existe también en este diseño un bloque **process** sin lista de sensibilidad y con sentencias **wait**. Inicialmente, este bloque detiene su ejecución hasta que transcurren 10 ns. Por ello, la sentencia secuencial de asignación `x3 <= x1` se ejecuta en el instante $t = 10$ ns y se planifica la asignación del nuevo valor a la señal `x3` en el instante $10 + \delta$ ns. Posteriormente, se ejecuta el bucle **for** cuatro veces, en los instantes 10 ns, 20 ns, 30 ns y 40 ns. Finalmente, se para la ejecución del bloque **process** al llegar a la última sentencia **wait**.

Obsérvese que en un bloque **process** la asignación a una variable tiene efecto inmediato, mientras que la asignación a una señal se produce tras pasar un tiempo δ. Debido a ello, la señal `x5` está retardada 10 ns respecto a la señal `x4`. Analicemos, por ejemplo, la ejecución del bucle **for** en $t = 10$ ns. Esta ejecución da lugar a que se asigne a la señal `x4` en $t = 10 + \delta$ ns el valor '0', que es el resultado de la or lógica entre los valores que tienen las señales `x1` y `x2` en $t = 10$ ns. Por tanto, la señal `x4` tiene el valor 'U' hasta que en $t = 10 + \delta$ ns toma el valor '0'. Esta ejecución del bucle causa también que se asigne en $t = 10 + \delta$ ns a la señal `x5` el valor que tiene la señal `x4` en $t = 10$ ns, es decir, 'U'.

PREGUNTA 2 (3 puntos)

Escriba en VHDL la **architecture** que describe el comportamiento del circuito combinacional mostrado en la siguiente figura.



Este circuito realiza la comparación de las señales de 8 bits `a` y `b`. Estas señales son interpretadas como números binarios con o sin signo dependiendo del valor de la señal de entrada `sel`. Si `sel='1'`, se realiza la comparación entre `a` y `b` interpretándolos como números binarios con signo. Por el contrario, si `sel='0'` la comparación entre `a` y `b` se hace interpretándolos como números binarios sin signo. El circuito tiene tres salidas, `x1`, `x2` y `x3` que valen '1' sólo en el caso de que $a > b$, $a = b$ y $a < b$, respectivamente. A continuación se muestra la **entity** del circuito incluyendo la declaración de las únicas librerías que se han de usar en el diseño.

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

entity comparador is port
   ( x3, x2, x1 : out std_logic;
     a, b       : in  std_logic_vector(7 downto 0);
     sel        : in std_logic );
end entity comparador;
```

Solución a la Pregunta 2

Los Códigos VHDL 1.1 y 1.2 son dos posibles soluciones a esta pregunta. En la primera solución se ha empleado únicamente código concurrente y en la segunda solución un bloque **process**. Obsérvese que al inicio del bloque **process** se ha dado valor a todas las señales de salida del circuito. Otra alternativa sería realizar una asignación a cada una de las señales de salida en cada rama de las sentencias **if**.

```
-------------------------------------
-- Comparador de números de 8 bits
-- interpretados como binario con signo
-- o sin signo
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity comparador is port
( x3, x2, x1 : out std_logic;
         a, b : in    std_logic_vector(7 downto 0);
         sel  : in std_logic );
end entity comparador;

architecture comparador of comparador is
    signal x1_signed, x1_unsigned: std_logic;
    signal x2_signed, x2_unsigned: std_logic;
    signal x3_signed, x3_unsigned: std_logic;
begin
    x1_signed <= '1' when signed(a) > signed(b) else '0';
    x2_signed <= '1' when signed(a) = signed(b) else '0';
    x3_signed <= '1' when signed(a) < signed(b) else '0';
    x1_unsigned <= '1' when a > b else '0';
    x2_unsigned <= '1' when a = b else '0';
    x3_unsigned <= '1' when a < b else '0';
    x1 <= x1_signed when sel = '1' else x1_unsigned;
    x2 <= x2_signed when sel = '1' else x2_unsigned;
    x3 <= x3_signed when sel = '1' else x3_unsigned;
end architecture comparador;
-------------------------------------
```

Código VHDL 1.1: Descripción del comportamiento del comparador de número de 8 bit con o sin signo empleando sentencias when - else.

```
-------------------------------------
-- Comparador de números de 8 bits
-- interpretados como binario con signo
-- o sin signo
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity comparador is port
( x3, x2, x1 : out std_logic;
          a, b : in    std_logic_vector(7 downto 0);
          sel  : in std_logic );
end entity comparador;

architecture comparador of comparador is
begin
    process (a, b, sel)
    begin
       x1 <= '0';
       x2 <= '0';
       x3 <= '0';
       if (sel = '0') then    -- Sin signo
          if (a > b) then x1 <= '1';
          elsif (a = b) then x2 <= '1';
          else x3 <= '1';
          end if;
       else    -- Con signo
          if (signed(a) > signed(b)) then x1 <= '1';
          elsif (signed(a) = signed(b)) then x2 <= '1';
          else x3 <= '1';
          end if;
       end if;
    end process;
 end architecture comparador;
-------------------------------------
```

Código VHDL 1.2: Descripción del comportamiento del comparador de número de 8 bit con o sin signo empleando un bloque process.

PREGUNTA 3 (2.5 puntos)

Programe en VHDL un banco de pruebas del comparador que ha diseñado al resolver la Pregunta 2. El banco de pruebas debe comprobar el funcionamiento del circuito para los vectores de test siguientes:

- `a="11111111",b="00000000",sel='1'`
- `a="11111111",b="00000000",sel='0'`
- `a="00001110",b="00001111",sel='1'`
- `a="00001110",b="00001111",sel='0'`
- `a="01111111",b="01111110",sel='1'`
- `a="01111111",b="01111110",sel='0'`
- `a="00011110",b="00011110",sel='1'`
- `a="00011110",b="00011110",sel='0'`

El banco de pruebas debe comprobar para cada vector de test si la salida de la UUT es correcta, mostrar un mensaje cada vez que la salida de la UUT no sea correcta y mostrar un mensaje al finalizar el test en el que se indique el número total de salidas incorrectas.

Solución a la Pregunta 3

El código VHDL que describe el banco de pruebas del circuito comparador se muestra en Código VHDL 1.3–1.4.

```
-------------------------------------
-- Banco de pruebas comparador signed/unsigned
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;

entity bp_comparador is
end entity bp_comparador;

architecture bp_comparador of bp_comparador is
   signal x3, x2, x1 : std_logic;   -- Conectar salidas UUT
   signal a, b : std_logic_vector(7 downto 0);   -- Conectar entradas UUT
   signal sel : std_logic;   -- Conectar entradas UUT

   component comparador is port
      ( x3, x2, x1 : out std_logic;
           a, b : in   std_logic_vector(7 downto 0);
           sel   : in std_logic);
   end component comparador;

begin
   -- Instanciar y conectar UUT
   uut : component comparador port map
          ( x3, x2, x1, a, b, sel );

   gen_vec_test : process
      variable num_errores   : integer := 0;              -- Numero de errores
   begin
        a <= "11111111";
        b <= "00000000";
        sel <= '1';
        wait for 10 ns;
-- Comprueba resultado
        if (x1='1' or x2='1' or x3='0') then
        report("Error en el primer vector de test");
        num_errores := num_errores + 1;
        end if;
        sel <= '0';
        wait for 10 ns;
-- Comprueba resultado
        if (x1='0' or x2='1' or x3='1') then
        report("Error en el segundo vector de test");
        num_errores := num_errores + 1;
        end if;
        a <= "00001110";
        b <= "00001111";
        sel <= '1';
```

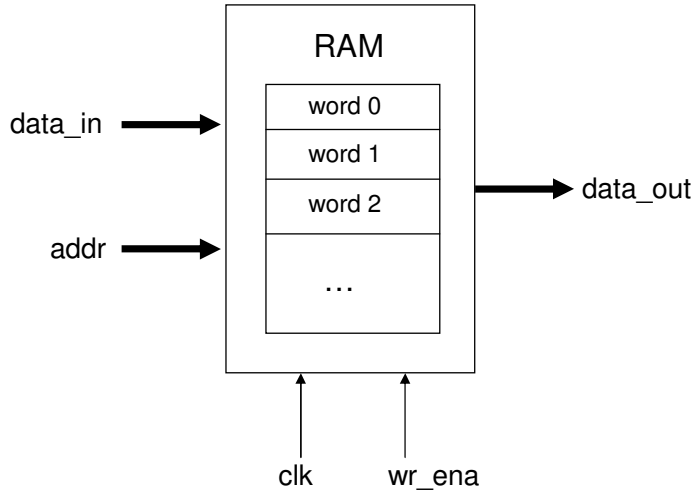
Código VHDL 1.3: Banco de pruebas del comparador diseñado en la Pregunta 2 (parte inicial).

```
        wait for 10 ns;
-- Comprueba resultado
        if (x1='1' or x2='1' or x3='0') then
        report("Error en el tercer vector de test");
        num_errores := num_errores + 1;
        end if;
        sel <= '0';
        wait for 10 ns;
-- Comprueba resultado
        if (x1='1' or x2='1' or x3='0') then
        report("Error en el cuarto vector de test");
        num_errores := num_errores + 1;
        end if;
        a <= "01111111";
        b <= "01111110";
        sel <= '1';
        wait for 10 ns;
-- Comprueba resultado
        if (x1='0' or x2='1' or x3='1') then
        report("Error en el quinto vector de test");
        num_errores := num_errores + 1;
        end if;
        sel <= '0';
        wait for 10 ns;
-- Comprueba resultado
        if (x1='0' or x2='1' or x3='1') then
        report("Error en el sexto vector de test");
        num_errores := num_errores + 1;
        end if;
        a <= "00011110";
        b <= "00011110";
        sel <= '1';
        wait for 10 ns;
-- Comprueba resultado
        if (x1='1' or x2='0' or x3='1') then
        report("Error en el septimo vector de test");
        num_errores := num_errores + 1;
        end if;
        sel <= '0';
        wait for 10 ns;
-- Comprueba resultado
        if (x1='1' or x2='0' or x3='1') then
        report("Error en el octavo vector de test");
        num_errores := num_errores + 1;
        end if;
      report "Test completo. Hay "      &
              integer'image(num_errores) &
              " errores.";
      wait; --Final simulación
   end process gen_vec_test;
end architecture bp_comparador;
-------------------------------------
```

Código VHDL 1.4: Banco de pruebas del comparador diseñado en la Pregunta 2 (parte final).

PREGUNTA 4 (2.5 puntos)

Diseñe en VHDL una memoria de acceso aleatorio que contenga 16 palabras de 8 bits (RAM 16×8). Tal como se muestra en la siguiente figura, la RAM tiene una bus de entrada de datos (`data_in`), un bus de salida de datos (`data_out`), un bus de direcciones (`addr`), una entrada de reloj (`clk`) y una entrada para habilitar la escritura (`wr_ena`).



La señal `data_in` se almacena en la posición indicada por `addr` cuando la señal `wr_ena` está a '1' y la señal `clk` está en el flanco de subida.

Por otro lado, el bus de salida de datos debe mostrar continuamente los datos seleccionados por `addr`.

Realice el diseño en VHDL de modo que el número de bits de cada palabra (`bits`) y el número de palabras de la memoria (`words`) sean constantes de tipo **generic**.

Solución a la Pregunta 4

El diseño de la memoria acceso aleatorio (RAM) se muestra el Código VHDL 1.5.

```
-----------------------------------------------
-- Memoria de acceso aleatorio (RAM) 16x8
-- Fichero: RAM.vhd
library IEEE;
use IEEE.std_logic_1164.all;

entity ram is
      generic( bits   : integer := 8; --Num. bits por palabra
               words  : integer := 16);--Num. palabras en mem.
      port( data_out  : out std_logic_vector(bits-1 downto 0);
            wr_ena    : in std_logic;
            clk       : in std_logic;
            addr      : in integer range 0 to words-1;
            data_in   : in std_logic_vector(bits-1 downto 0));
end entity ram;

architecture ram of ram is
      type vector_array is array (0 to words-1) of
           std_logic_vector(bits-1 downto 0);

      signal memory: vector_array;
begin
      process(clk, wr_ena)
      begin
           if(wr_ena = '1') then
                if rising_edge(clk) then
                     memory(addr) <= data_in;
                end if;
           end if;
      end process;
      data_out <= memory(addr);
end architecture ram;
-----------------------------------------------
```

Código VHDL 1.5: Diseño de la RAM.