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

INGENIERÍA DE COMPUTADORES 3

Solución al examen de Junio 2015, 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` entre los instantes 0 y 80 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 : std_logic;
begin
  x1 <= '1','0' after 15 ns,
        '1' after 20 ns, '0' after 35 ns,
        '1' after 40 ns;
  Proc1: process
     variable valor  : std_logic;
  begin
     for i in 0 to 3 loop
       x2 <= x1;
       wait for 10 ns;
       x3 <= x1 or x2;
     end loop;
       x4 <= x1 or x2;
     wait;
  end process;
  Proc2: process
     variable valor  : std_logic;
  begin
     x1 <= '0';
     wait for 5 ns;
     x1 <= '1';
     wait for 15 ns;
  end process;
end architecture cronoW;
```

Solución a la Pregunta 1

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

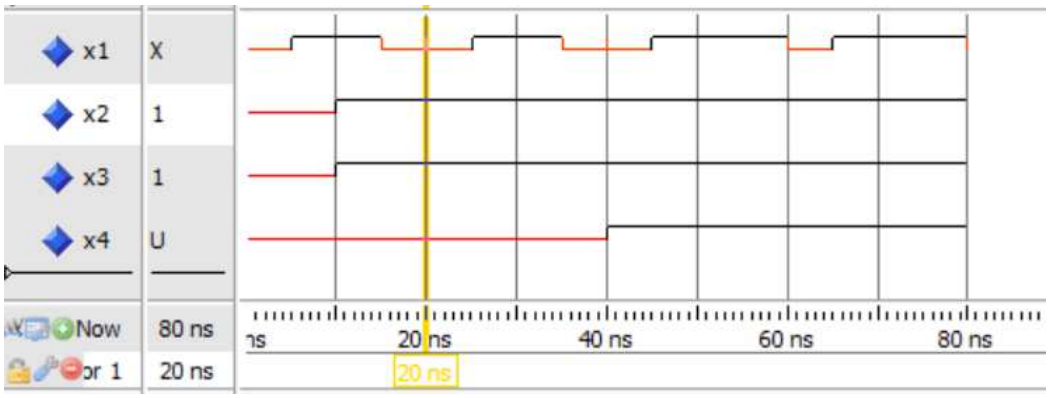

Figura 1.1: Cronograma de evolución de las señales.

PREGUNTA 2 (2.5 puntos)

Diseñe usando VHDL el circuito mostrado en la siguiente figura, que está compuesto por una unidad aritmético lógica (ALU) y un registro de desplazamiento. La ALU realiza operaciones sobre dos operandos de 8 bits, denominados `A` y `B`. La salida de la ALU es la entrada al registro del desplazamiento. El registro de desplazamiento opera en el flanco de subida de la señal de reloj `CLK`. La señal `S`, de 4 bits, determina la operación realizada por el circuito tal como se indica en las dos tablas de operaciones mostradas a continuación.

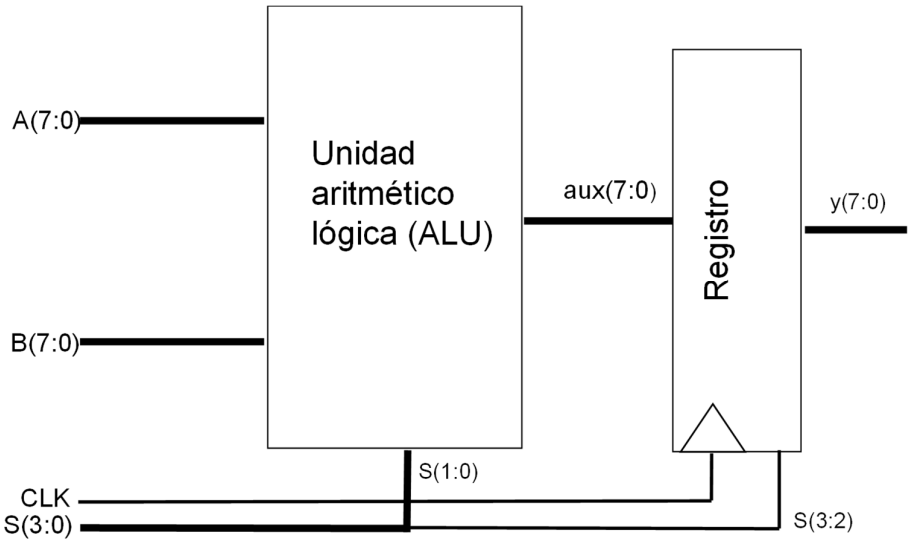

Tabla 1: Operaciones del registro.

S(3)S(2)	**Operación**
00	Carga de la señal `aux` desplazada un bit a la derecha, introduciendo un '0' en el bit más significativo
01	Carga de la señal `aux` desplazada un bit a la izquierda, introduciendo un '0' en el bit menos significativo
10	Carga de la señal `aux` rotada un bit a la derecha
11	Carga de la señal `aux` sin modificarla

Escriba en VHDL la **architecture** que describe el comportamiento del circuito empleando un bloque **process** que describa el comportamiento de la ALU y otro bloque **process** que describa el comportamiento del registro. 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 ALUReg is
   port( Y    : out std_logic_vector (7 downto 0);
         A, B : in  std_logic_vector (7 downto 0);
         CLK  : in  std_logic;
         S    : in  std_logic_vector (3 downto 0) );
end entity ALUReg;
```

Solución a la Pregunta 2

La **architecture** que describe el comportamiento del circuito se muestra en Código VHDL 1.1.

```
-----------------------------------------
-- ALU con registro
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity ALUReg is
    port( Y      : out std_logic_vector ( 7 downto 0);
          A, B   : in std_logic_vector (7 downto 0);
          CLK    : in std_logic;
          S      : in std_logic_vector ( 3 downto 0));
end entity ALUReg;

architecture ALUReg of ALUReg is
    signal aux : std_logic_vector(7 downto 0);
    begin
      --ALU
       alu: process (S, A, B)
       begin
          case S (1 downto 0) is
               when "00" =>
                  aux <= std_logic_vector(signed(A) + signed(B));
               when "01" =>
                  aux <= std_logic_vector (signed(A) - signed(B))
                         ;
               when "10" =>
                  aux <= A and B;
               when others =>
                  aux <= A or B;
          end case;
      end process alu;
      --Registro
      reg:process(clk)
      begin
          if rising_edge(clk) then
          case S (3 downto 2) is
               when "00" =>
                  Y <= '0' & aux(7 downto 1);
               when "01" =>
                  Y <= aux(6 downto 0)&'0';
               when "10" =>
                  Y <= aux(0)& aux(7 downto 1);
               when others =>
                  Y <= aux;
          end case;
          end if;
      end process reg;
end architecture ALUReg;
-----------------------------------------
```

Código VHDL 1.1: Diseño de la ALU con el registro.

PREGUNTA 3 (3.5 puntos)

Escriba la **architecture** que describe el comportamiento de un circuito contador módulo m programable, con reset asíncrono activo a nivel alto. Es decir, que cuenta desde 0 hasta el valor m-1, incrementando en cada ciclo de reloj su valor en 1. El circuito ha de tener la **entity** siguiente:

```
entity contadorProg is
   port( q : out std_logic_vector (3 downto 0);
         clk, reset : in std_logic ;
         m : in  std_logic_vector (3 downto 0) );
end entity contadorProg;
```

Cuando la señal `reset` tiene el valor '1', pone la señal `q` a "0000". La señal `q` se interpreta como un número binario sin signo. Si la señal `reset` tiene el valor '0', en cada flanco de subida de la señal de reloj la señal `q` incrementa su valor en uno, pasando cíclicamente del valor 0 al valor m-1. Por ejemplo, si la señal `m` es "0100" la señal `q` toma cíclicamente los valores "0000", "0001", "0010", "0011", "0000", ...

Solución a la Pregunta 3

El código VHDL del contador programable se muestra en Código VHDL 1.2.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity contadorProg is
    port( q: out std_logic_vector (3 downto 0);
          clk, reset : in std_logic ;
          m : in std_logic_vector (3 downto 0));
end entity contadorProg;
architecture contadorProg of contadorProg is
  signal r_reg: unsigned (3 downto 0);
  signal r_next: unsigned (3 downto 0);
begin
  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;
  r_next <= (others=>'0') when r_reg=(unsigned(m)-1) else
            r_reg + 1;
  q <= std_logic_vector(r_reg);
end contadorProg;
```

Código VHDL 1.2: Diseño del circuito contador.

PREGUNTA 4 (2 puntos)

Programe en VHDL un banco de pruebas para el circuito que ha diseñado al contestar a la Pregunta 3. La señal de reloj (`clk`) debe tener un periodo de 20 ns e inicialmente valer '0'. El programa de test debe realizar consecutivamente las acciones siguientes:

1. *Reset y m con valor 5*. La señal de reset ha de tener el valor '1' durante 20 ns. La señal `m` ha de tener el valor "0101".
2. *Empezar la cuenta*. Comprobar que el circuito pasa por los valores "0000", "0001", ..., "0100" y "0000".
3. *Reset y m con valor 15*. La señal de reset ha de tener el valor '1' durante 20 ns. La señal `m` ha de tener el valor "1111".
4. *Empezar la cuenta*. Comprobar que el circuito pasa por los valores "0000", "0001", ..., "1110" y "0000".

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 contador se muestra en Código VHDL 1.3–1.4.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity bp_contador2 is
    constant PERIODO     : time      := 20 ns; -- PERIODO clk

end entity bp_contador2;

architecture bp_contador2 of bp_contador2 is
    signal q   : std_logic_vector(3 downto 0); --Salida UUT
    signal m    : std_logic_vector(3 downto 0); --Entradas UUT
    signal clk : std_logic := '0';
    signal reset: std_logic;
    component contadorProg is
    port( q: out std_logic_vector (3 downto 0);
          clk, reset : in std_logic ;
          m : in std_logic_vector (3 downto 0));
    end component contadorProg;


begin
```

Código VHDL 1.3: Banco de pruebas del contador.

```
 UUT : component contadorProg port map
         (q, clk, reset, m);
 clk <= not clk after (PERIODO/2);

vec_test : process is
     variable temp  : std_logic_vector (3 downto 0);
     variable error_count : integer := 0;
   begin
     report "Comienza la simulación";
     m <= "0101";
     reset <= '1';
     wait for PERIODO/2;
     wait until falling_edge(clk);
     reset <= '0';
     for i in 0 to 4 loop
        temp := std_logic_vector(to_unsigned(i,4));
        if(temp /= q) then
            report "ERROR";
            error_count := error_count +1;
        end if;
        wait for PERIODO;
     end loop;
     temp := "0000";
     if(temp /= q) then
        report "ERROR";
        error_count := error_count +1;
     end if;
     m <= "1111";
     reset <= '1';
     wait for PERIODO;
     reset <= '0';
     for i in 0 to 14 loop
        temp := std_logic_vector(to_unsigned(i,4));
        if(temp /= q) then
            report "ERROR";
            error_count := error_count +1;
        end if;
        wait for PERIODO;
     end loop;
     temp := "0000";
     if(temp /= q) then
        report "ERROR";
        error_count := error_count +1;
     end if;

       -- 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_contador2;
```

Código VHDL 1.4: Continuación del banco de pruebas del contador.