INGENIERÍA DE COMPUTADORES III

Solución al examen de Septiembre 2012

PREGUNTA 1 (2 puntos)

1.a) (1 punto) Dibuje el diagrama conceptual correspondiente al fragmento de código *Fragmento 1*.

1.b) (1 punto) Dibuje el diagrama conceptual correspondiente al fragmento de código *Fragmento 2*.

```
---- Fragmento 1-------------
signal a, b, r : unsigned (7 downto 0);
signal x, y    : unsigned (3 downto 0);
...
r <=  a+b when x+y>0 else
      a-b when x>y and y>0 else
      a;
```

```
 ---- Fragmento 2-------------
signal s: std_logic_vector (1 downto 0);
signal a, b : std_logic;
...
process(s)
begin
 case s is
   when "00" =>
     a <= '1'; b <= '0';
   when "01" =>
     a <= '1'; b <= '1';
   when others =>
     a <= '0'; b <= '0';
 end case;
end process;
```

Solución a la Pregunta 1

A continuación se muestran los diagramas conceptuales.

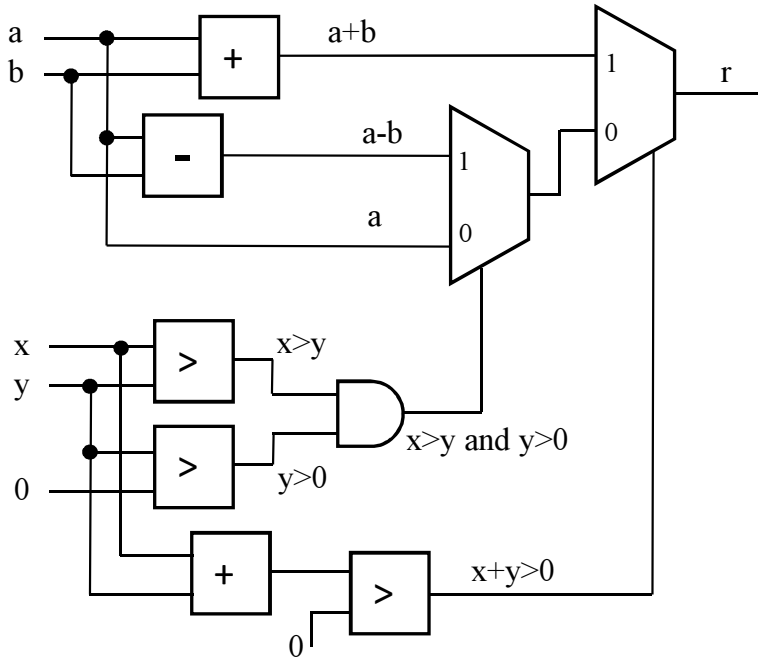


Figura 1.1: Diagrama conceptual correspondiente a Fragmento 1.

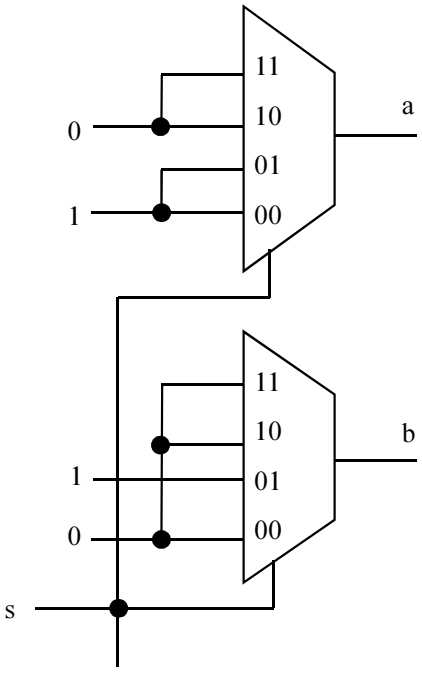


Figura 1.2: Diagrama conceptual correspondiente a Fragmento 2.

PREGUNTA 2 (3 puntos)

Escriba el código VHDL de la **architecture** que describe el comportamiento de un circuito contador binario ascendente de 4 bits, con señal de reset asíncrona activa a nivel bajo y con carga paralelo síncrona. La **entity** del circuito se muestra a continuación.

```
entity contador is
  port ( count            : out std_logic_vector (3 downto 0);
         clk, reset, load : in  std_logic;
         data             : in  std_logic_vector (3 downto 0) );
end contador;
```

Las entradas al circuito son la señal de reloj (`clk`), la señal de reset asíncrono activo a nivel bajo (`reset`) y las señales de carga (`load`, `data`). La salida del circuito contador es la señal de 4 bits `count`.

El funcionamiento del circuito debe ser el siguiente:

- *Reset asíncrono activo a nivel bajo*. Cuando `reset` pasa a valer '0', el contador se pone al valor "0000".
- *Cuenta síncrona ascendente*. Mientras `reset` vale '1' y `load` vale '0', el contador se incrementa en uno en cada flanco de subida de la señal de reloj.
- *Carga paralelo síncrona*. Mientras `reset` vale '1' y `load` vale '1', en cada flanco de subida de la señal de reloj se carga en el contador el valor de la señal de entrada de 4 bits `data`.

El diseño debe realizarse describiendo el comportamiento del circuito. Para ello, deben emplearse tantas señales auxiliares y asignaciones concurrentes simples como sean necesarias, así como un único bloque **process**. Asimismo, en el diseño únicamente pueden emplearse los dos siguientes paquetes de la librería IEEE:

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

Solución a la Pregunta 2

La **architecture** describiendo el comportamiento del circuito contador se muestra en Código VHDL 1.1.

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

entity contador is
   port (   count              : out std_logic_vector (3  downto 0);
            clk, reset, load   : in    std_logic;
            data               : in    std_logic_vector (3  downto 0) );
end contador;

architecture comp  of contador  is
    signal count_i : unsigned (3  downto 0);
begin
   process (clk, reset)
   begin
      if (reset = '0') then
          count_i <= (others => '0');
      elsif (rising_edge(clk)) then
          if load = '1'  then
            count_i <= unsigned(data);
          else
            count_i <= count_i + 1;
          end if;
      end if;
   end process;
   count <= std_logic_vector(count_i);
end comp ;
```

Código VHDL 1.1: Descripción del comportamiento del contador binario ascendente de 4 bits.

PREGUNTA 3 (2 puntos)

Programe en VHDL un banco de pruebas para el contador binario que ha diseñado al contestar a la Pregunta 2. La señal de reloj (`clk`) debe tener un periodo de 10 ns e inicialmente valer ’0’. El programa de test debe realizar las acciones siguientes:

1. *Reset*. Resetear el contador.
2. *Cuenta síncrona ascendente*. Incrementar el valor del contador hasta “0011".
3. *Carga paralelo síncrona*. Cargar el valor “1101".
4. *Cuenta síncrona ascendente*. Incrementar el valor del contador hasta “1111".
5. *Reset*. Resetear el contador.

El correcto funcionamiento del circuito debe comprobarse mediante inspección visual. No es necesario que el banco de prueba compruebe que las salidas de la UUT son las esperadas. Dibuje el cronograma de evolución que han de seguir las señales de entrada y salida de la UUT.

Solución a la Pregunta 3

En la Figura 1.3 se muestra el cronograma que han de seguir las entradas y salidas de la UUT. El código VHDL del banco de pruebas del circuito se muestra en Código VHDL 1.2.

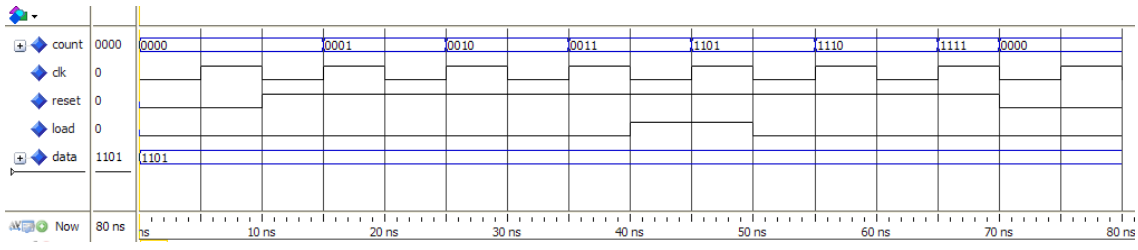


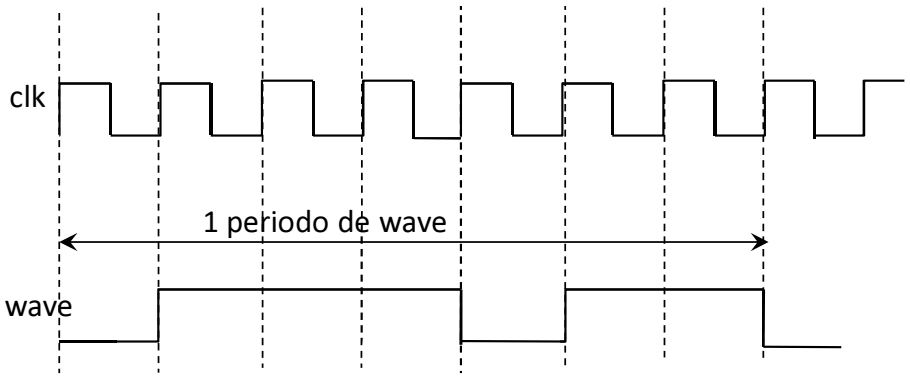
Figura 1.3: Cronograma que han de seguir las señales de entrada y salidas de la UUT.

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

entity bp_contador is
   constant PERIODO : time := 10 ns; -- Periodo reloj
end entity bp_contador;

architecture bp_contador of bp_contador is
   signal count : std_logic_vector (3 downto 0);   -- Conectar salidas UUT
   signal clk: std_logic := '0';-- Conectar entrada UUT
   signal reset, load : std_logic;   -- Conectar entradas UUT
   signal data : std_logic_vector (3 downto 0);   -- Conectar entrada UUT

   component contador is
      port ( count              : out std_logic_vector (3 downto 0);
             clk, reset, load : in   std_logic;
             data               : in   std_logic_vector (3 downto 0) );
   end component contador;

begin
   -- Instanciar y conectar UUT
   uut : component contador port map
         ( count => count,
           clk => clk, reset => reset, load => load,
           data => data );
   clk <= not clk after PERIODO/2;
   reset <= '0',
            '1' after ( PERIODO ),
            '0' after ( PERIODO*7);
   load <= '0',
           '1' after (4*PERIODO),
           '0' after (5*PERIODO);
   data <= "1101";
end architecture bp_contador;
-----------------------------------
```

Código VHDL 1.2: Banco de pruebas del contador.

PREGUNTA 4 (3 puntos)

Diseñe un generador de señales que obtiene la forma de onda mostrada a continuación (señal `wave`) a partir de una señal de reloj (`clk`). Describa su comportamiento como una máquina de estado finito de Moore cuyas transiciones se producen en el flanco de subida de la señal de reloj.



La **entity** de este circuito se muestra a continuación.

```
entity generador is
  port( wave : out std_logic;
        clk  : in  std_logic );
end entity generador;
```

Solución a la Pregunta 4

El código VHDL correspondiente al generador de señales se muestra en Código VHDL 1.3.

```
----------------------------------------------------
---Generador de ondas implementado como
---máquina de estados
library IEEE;
use IEEE.std_logic_1164.all;

entity generador is
   port(wave : out std_logic;
        clk  : in std_logic);
end entity generador;

architecture fsm of generador is
 signal state : std_logic_vector(2 downto 0);
begin
 proximo_estado: process (clk)
 begin
   if (rising_edge(clk)) then
      case state is
         when "000" => state <= "001";wave <='0';
         when "001" => state <= "010";wave <='1';
         when "010" => state <= "011";wave <='1';
         when "011" => state <= "100";wave <='1';
         when "100" => state <= "101";wave <='0';
         when "101" => state <= "110";wave <='1';
         when "110" => state <= "000";wave <='1';
         when others => state <= "000";wave <='0';
      end case;
   end if;
end process proximo_estado;
end architecture fsm;
-----------------------------------------
```

Código VHDL 1.3: Diseño del generador de señales.