INGENIERÍA DE COMPUTADORES III

INSTRUCCIONES
Por favor, entregue esta primera hoja de enunciado junto con el examen.
Dispone de 2 horas para realizar el examen.
MATERIAL PERMITIDO: Ninguno.

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 when x+y>1 else
      a-b when x>y and y>0 else
      b;
```

```
 ---- Fragmento 2-------------
signal s: std_logic_vector (1 downto 0);
signal a, b, x : std_logic;
...
with s select
   x <= (a xor b) when "00",
        (a and b) when "01"|"10",
         '0' when others;
```

Pregunta 2 (3 puntos)

Diseñe en VHDL los circuitos combinacionales indicados a continuación.

2.a) (1 punto) Un circuito con tres entradas y una salida, tal que la salida valga '1' si dos o más de las entradas valen '1'. Realice el diseño empleando un bloque **process** con una o varias sentencias **if**. La **entity** del circuito se muestra a continuación.

```
entity majority is
   port ( Y       : out std_logic;
          A, B, C : in std_logic);
end entity majority;
```

2.b) (1 punto) Un circuito con siete entradas y una salida de tres bits, tal que los tres bits de salida indiquen la entrada con mayor prioridad que está a '1'. La **entity** del circuito se muestra a continuación. La entrada `Y7` es la de mayor prioridad y la `Y1` la de menor. Si ninguna de las entradas está a '1', la salida deberá ser "000". Realice el diseño empleando una asignación concurrente de selección.

```
entity priority is
   port ( DOUT                  : out std_logic_vector(2 downto 0);
          Y1,Y2,Y3,Y4,Y5,Y6,Y7 : in std_logic);
end entity priority;
```

2.c) (1 punto) Diseñe el circuito anterior empleando un bloque **process** con una o varias sentencias **if**.

Pregunta 3 (3 puntos)

Diseñe usando VHDL un circuito secuencial conversor serie-a-paralelo de 8 bits que opera en el flanco de subida de la señal de reloj y tiene reset asíncrono. El circuito tiene la siguiente **entity**:

```
entity conversor is
   port( DataOut:  out std_logic_vector( 7 downto 0);
         Valid: out std_logic;
         Clk: in std_logic;
         Reset: in std_logic;
         Start: in std_logic;
         DataIn: in std_logic );
end entity conversor;
```

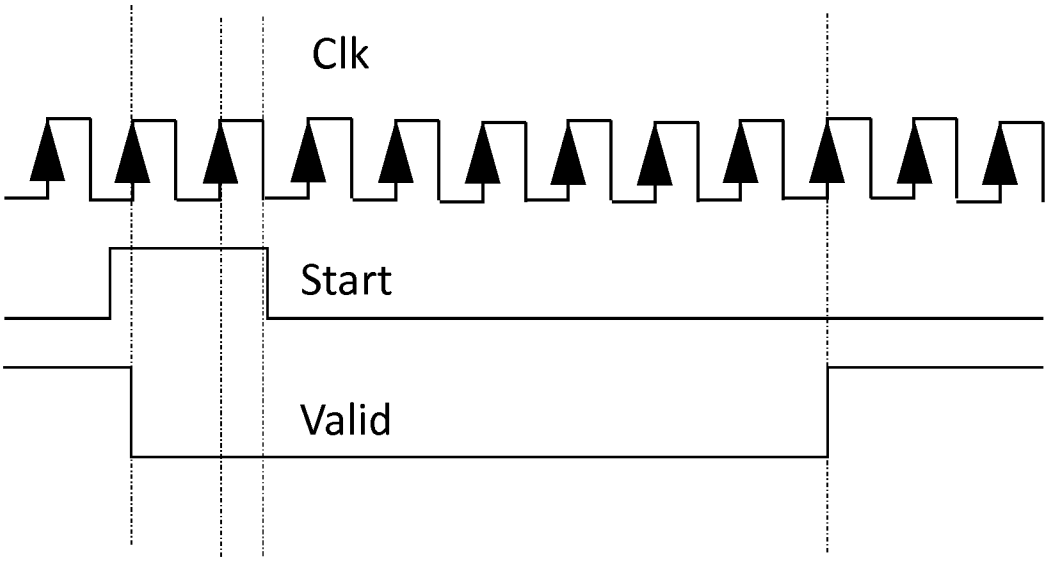
A continuación se describe el funcionamiento del circuito conversor.

Los bits recibidos por la entrada serie `DataIn` se almacenan en un registro interno del circuito llamado `Content` tal como se describe a continuación.

En el flanco de subida de la señal de reloj si la señal `Valid` vale '0' y la señal `Reset` vale '0', se introduce el valor de la señal de entrada serie `DataIn` en la posición más significativa del registro `Content` desplazando el contenido del registro a la derecha. El contenido del registro `Content` es reseteado al valor "00000000" en el flanco de subida de la señal de reloj cuando la señal `Reset` vale '1'. Si la señal `Reset` vale '0' y la señal `Valid` vale '1' se mantiene el valor del registro `Content`.

La señal de salida `Valid` se pone a'0' cuando la señal de entrada `Start` vale '1' en el flanco de subida de la señal de reloj. Si a partir de ese momento se producen 7 flancos de subida de la señal de reloj en los que la señal `Start` valga '0', la señal `Valid` pasa a tomar el valor '1'

en el séptimo flanco de subida de la señal de reloj. El valor de la señal `Valid` se mantiene a '1' siempre que la señal `Start` valga '0' en los flancos de subida de la señal de reloj. En el siguiente cronograma puede verse los valores que toma la señal de salida `Valid` en función de los valores que tienen la señal de reloj `Clk` y la señal de entrada `Start`.



El contenido del registro `Content` se carga en la salida paralelo `DataOut` siempre que la señal `Valid` tenga el valor '1' y la señal `Reset` valga '0'. Si la señal `Valid` tiene el valor '0' y la señal `Reset` vale '0', la señal `DataOut` mantiene su valor. La salida `DataOut` se resetea al valor "00000000" cuando la señal `Reset` vale '1'.

El diseño del registro en VHDL debe realizarse describiendo el comportamiento del circuito.

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 primer flanco de subida de la señal de reloj se ha de producir en el instante 10 ns. El programa de test debe realizar consecutivamente las acciones siguientes:

1. *Reset*. La señal de reset ha de tener el valor '1' durante los primeros 15 ns.
2. *Cargar en la señal de salida paralelo* `DataOut` *el valor "11111100"*. La señal `Start` ha de tener el valor '1' únicamente entre los instantes 40 ns y 60 ns, debiendo tener el valor '0' el resto del tiempo. La señal de entrada `DataIn` ha de valer inicialmente '0' y cambiar al valor '1' en el instante 80 ns. En consecuencia, la señal `DataIn` toma en ciclos de reloj consecutivos los valores '0', '0', '1', '1', '1', '1', '1' y '1'. Comprobar que las señales de salida `DataOut` y `Valid` toman los valores correctos. En caso contrario, el banco de pruebas debe mostrar un mensaje indicándolo.