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)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `x1`, `x2`, `x3`, `x4`, `x5` y `x6` entre los instantes 0 y 100 ns.

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

entity crono is
end entity crono;

architecture crono of crono is
  signal x1, x2, x3, x4, x5, x6 : std_logic;
begin
  x1 <= '0', '1' after 10 ns,
        '0' after 15 ns,'1' after 25 ns,
        '0' after 30 ns;
  x2 <= transport x1 after 8 ns;
  Proc1: process(x1,x2)
  begin
     x3 <= x1 xor x2;
  end process;
   x4 <= x1 after 8 ns;
   x5 <= x1 xor x2;
   x6 <= x5;
end architecture crono;
```

Pregunta 2 (3 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`, y un operando de un bit denominado `cin`. La salida de la ALU es la entrada al registro de desplazamiento. El registro de desplazamiento opera en el flanco de subida de la señal de reloj `CLK`. La señal `S`, de 5 bits, determina la operación realizada por el circuito tal como se indica en las dos tablas de operaciones mostradas a continuación.

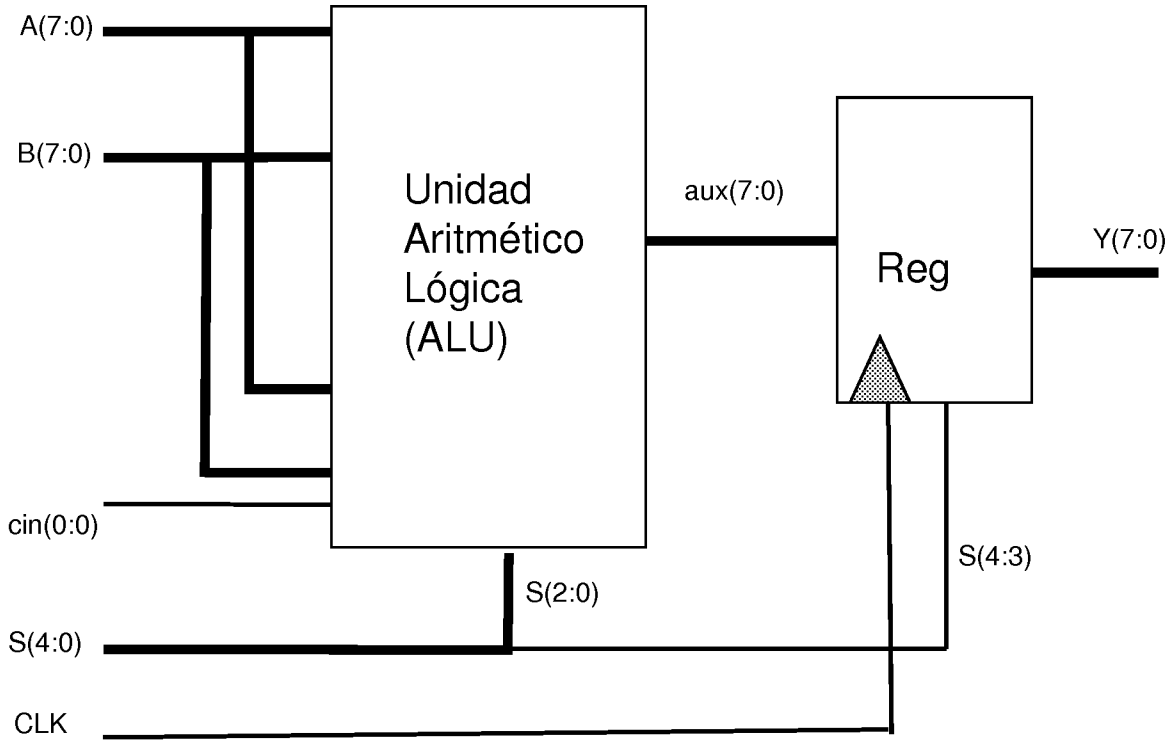


Tabla 1: Operaciones del registro.

`S(4)S(3)`	**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

Tabla 2: Operaciones de la ALU.

S(2)S(1)S(0)	**Operación**
000	A + B
001	A + B + cin
010	A + 1
011	B - 1
100	not B
101	A or B
110	A and B
111	A xor B

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);
         cin  : in std_logic_vector (0 downto 0);
         CLK : in std_logic;
         S  : in std_logic_vector ( 4 downto 0));
end entity ALUReg;
```

Pregunta 3 (3 puntos)

Realice el diseño usando VHDL de un contador síncrono ascendente módulo 6. Es decir, la salida del circuito (señal `y`) toma cíclicamente los valores "000", "001", "010", "011", "100" y "101". Si la cuenta está habilitada, se obtiene un nuevo valor de la salida en cada flanco de subida de la señal de reloj. El contador debe tener una entrada `c` activa a nivel alto que permita habilitar y deshabilitar la cuenta, la señal de reloj de entrada `clock` y una entrada `reset` síncrona activa a nivel alto, que pone la cuenta a '0'. Describa el comportamiento del circuito en términos de una máquina de Moore. Dibuje el diagrama de estados que describe el comportamiento del circuito.

Pregunta 4 (2 puntos)

Programe el banco de pruebas del circuito secuencial que ha diseñado en la Pregunta 3. 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.

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.

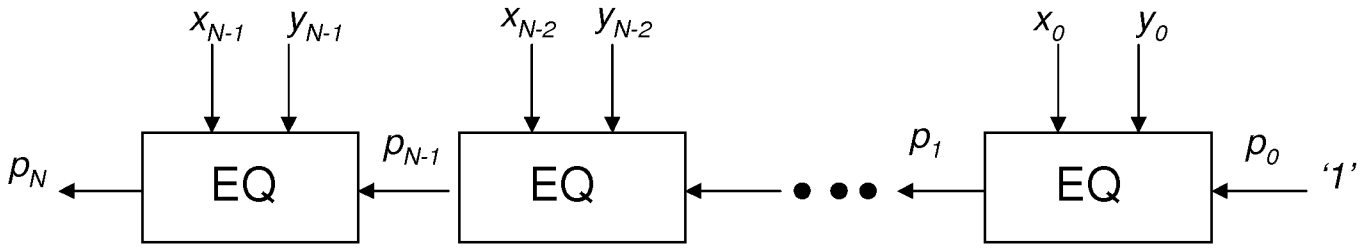
Pregunta 1 (2 puntos)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `x1`, `x2`, `x3`, `x4` y `x5` entre los instantes 0 y 50 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono1 is
end entity crono1;
architecture crono1 of crono1 is
  signal x1, x2, x3, x4, x5 : std_logic;
begin
  x1 <= '0', '1' after 10 ns,
        '0' after 15 ns,'1' after 25 ns,
        '0' after 30 ns;
  Proc1: process
  begin
    x2 <= '1';
    wait for 5 ns;
    x2 <= '0';
    wait for 10 ns;
    x2 <= '1';
    wait for 5 ns;
    x2 <= '0';
  end process;
   x3 <= x1 after 5 ns;
  Proc2: process
     variable v1  : std_logic;
  begin
    for i in 0 to 3 loop
       v1 := x1 xor x2;
       x4 <= v1;
       x5 <= x4;
       wait for 5 ns;
     end loop;
     wait;
  end process;
end architecture crono1;
```

Pregunta 2 (3 puntos)

Diseñe un circuito que compare si dos números de N bits tienen igual valor. Construya el circuito de manera iterativa, tal como se muestra en la siguiente figura.



Diseñe inicialmente un comparador de 1 bit, que tenga las tres entradas siguientes: los dos operandos de un bit (x_i e y_i) y el resultado de la comparación de la etapa anterior (p_i). p_i es un bit cuyo valor es '1' sólo si todas las parejas de bits comparadas hasta esa etapa son iguales. La salida del comparador de 1 bit, p_{i+1}, es el resultado de la comparación llevada a cabo en esa etapa, teniendo en cuenta el valor de las tres entradas. Finalmente, encadene N comparadores de 1 bit para implementar el comparador de N bits, usando para ello la sentencia **generate**. N ha de ser una constante de tipo **generic**.

Pregunta 3 (2 puntos)

Programe el banco de pruebas del circuito combinacional que ha diseñado en la Pregunta 2 para un valor de N igual a 6. 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.

Pregunta 4 (3 puntos)

Diseñe usando VHDL el siguiente circuito secuencial síncrono. El circuito opera en el flanco de subida de la señal de reloj (`clk`) y tiene como entradas la señal de reloj `clk`, una señal `reset` asíncrona activa a nivel alto y una señal de un bit `X`. El circuito tiene una señal de salida de un bit `Z`.

La señal de salida `Z` tiene valor '1' durante un ciclo de reloj cuando la secuencia de los tres últimos valores de la entrada sea "111", "110' ó "000", y tiene valor '0' en cualquier otro caso. La señal `reset` inicializa el circuito poniéndolo en un estado en que los últimos tres bits recibidos por la señal de entrada `X` son "001".

Diseñe el circuito como una máquina de Moore. Escriba en una tabla, cuya estructura se muestra a continuación, todos los estados de la máquina, indicando para cada estado cuál es el siguiente estado cuando la señal de entrada vale `X = '0'` y cuando la señal de entrada vale `X = '1'`. Especifique además el valor de la señal de salida `Z` correspondiente a cada estado.

Tabla de estados.

Estado	Estado siguiente si `X = '0'`	Estado siguiente si `X = '1'`	`Z`

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)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `clk`, `x1`, `x2`, `x3`, `x4`, `x5` y `x6` entre los instantes 0 y 100 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono2 is
end entity crono2;
architecture crono2 of crono2 is
  signal x1, x2, x3, x4, x5, x6 : std_logic;
  signal clk : std_logic:='0';
begin
  x1 <= '0' , '1' after 10 ns, '0' after 20 ns, '1' after 25 ns;
  proc1: process (clk)
  begin
      if ( falling_edge(clk) ) then
        x4 <= x3;
        x3 <= x2;
        x2 <= x1;
      end if;
  end process;
  clk <= not clk after 10 ns;
  proc2: process (clk) is
    variable t1, t2, t3: std_logic;
  begin
    if (falling_edge(clk) ) then
       t1:= x1;
       t2:= t1;
       t3:= t2;
       x5 <= t1;
       x6 <= x5;
    end if;
  end process;
end architecture crono2;
```

Pregunta 2 (3 puntos)

Diseñe un contador binario síncrono de cuatro bits que opere en el flanco de subida de la señal de reloj. El contador tiene la siguiente **entity**:

```
entity contadorBinario is
   port( q      : out std_logic_vector(3 downto 0);
         d      : in std_logic_vector(3 downto 0);
         syn_clr, en, load : in std_logic;
         clk, reset : in std_logic);
end entity contadorBinario;
```

La señal de salida `q` indica el valor actual de la cuenta. La señal de entrada `d` se carga en paralelo cuando se habilita la carga. La señal de entrada `reset` es activa a nivel alto y realiza un reset asíncrono del circuito poniendo la cuenta a cero. El contador tiene unas señales de entrada que permiten, en el flanco de subida de la señal de reloj, poner la cuenta a cero (reset síncrono), cargar un valor específico en la cuenta (señal de entrada `d`), activar la cuenta o pausarla, tal como se indica en la siguiente tabla. Cuando la cuenta está activada, se pasa cíclicamente por los valores "0000", "0001", ..., "1110", "1111".

`syn_clr`	`load`	`en`	Operación
1	-	-	Pone el contador a cero (reset síncrono)
0	1	-	Carga en paralelo
0	0	1	Activa la cuenta
0	0	0	Pausa la cuenta

En el diseño únicamente pueden emplearse los dos siguientes paquetes de la librería IEEE:

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

Pregunta 3 (2 puntos)

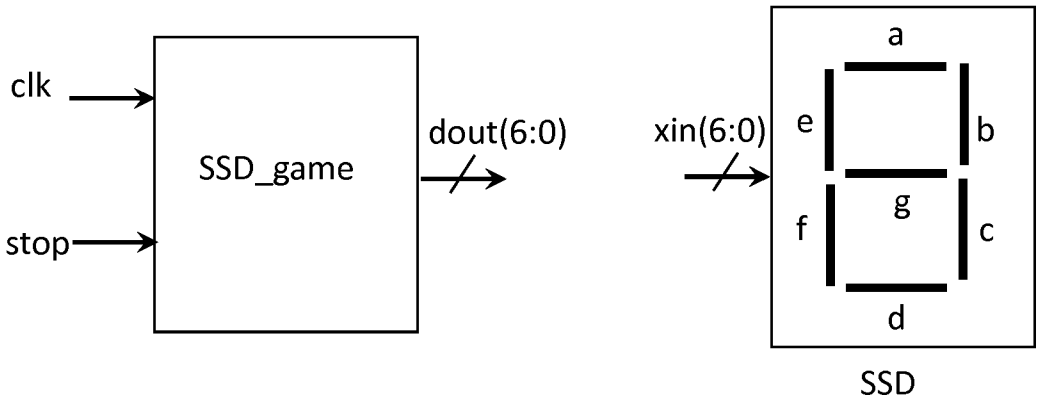
Programe en VHDL un banco de pruebas para el circuito que ha diseñado al contestar a la Pregunta 2. La señal de reloj (`clk`) debe tener un periodo de 20 ns e inicialmente valer '0'. El primer flanco de subida de la señal de reloj se ha de producir en el instante 10 ns. El programa de test debe realizar consecutivamente las acciones siguientes:

1. *Reset*. La señal de reset ha de tener el valor '1' durante los primeros 15 ns.
2. *Mantener la cuenta activa durante tres periodos de la señal de reloj.*
3. *Cargar en la cuenta el valor "1001".*
4. *Pausar la cuenta.*

El programa de test debe mostrar mensajes de error en el caso de que la señal de salida no tome el valor esperado. Dibuje el cronograma de las señales aplicadas al circuito y las salidas esperadas.

Pregunta 4 (3 puntos)

Se quiere diseñar un circuito denominado SSD_game para poder conectarse con un display de siete segmentos (SSD). En la siguiente figura se muestra el circuito a diseñar y el SSD con el que se pretende conectar.



El circuito SSD_game es un circuito síncrono que opera en el flanco de subida de la señal de reloj. El circuito contiene dos señales de entrada: la señal de reloj `clk` y la señal asíncrona `stop`. El circuito tiene una señal de salida de 7 bits llamada `dout` que se puede conectar con el circuito SSD. Consideramos que la señal de reloj que se va a conectar a la entrada del circuito tiene una frecuencia de 1 kHz (periodo de 1 ms).

El objetivo de los bits 0 a 6 de la señal `dout` es iluminar (valor '1') o apagar (valor '0') los segmentos a, b, c, d, e, f y g del circuito SSD, respectivamente. Por ejemplo, cuando `dout`(6) = '1' el segmento g está iluminado y cuando `dout`(6) = '0' el segmento g está apagado.

Siempre que la señal `stop` tenga el valor '0', el circuito genera unos valores de la señal `dout` que hace que se enciendan cíclicamente con cada flanco de subida de la señal de reloj los siguientes segmentos del SSD (mientras el resto de segmentos se mantienen apagados): segmento a, segmentos b y c, segmento d, segmentos d y e, segmento e, segmentos e y f.

Si la señal `stop` tiene el valor '1', el circuito pasa asíncronamente a un estado en que sólo esté encendido el segmento a del display. El circuito permanece en ese estado hasta que la señal `stop` tenga el valor '0'.

El sistema tiene que permanecer en los estados donde sólo hay un segmento del display iluminado durante un tiempo de 80 ms, y en los estados donde hay dos segmentos del display iluminados durante 30 ms.

Programe en VHDL el circuito SSD_game describiendo su comportamiento como una máquina de estados de tipo Moore. Dibuje el diagrama de estado correspondiente al diseño realizado.

UNED CALATAYUD

EXAMEN DE RESERVA NO DISPONIBLE

El contenido de este examen de reserva no está disponible, conforme al acuerdo del Consejo de Gobierno de la UNED de 11 de noviembre de 2015, en el que se acordó:

- No publicar los exámenes de reserva no utilizados en la valija virtual de centros nacionales.

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)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `x1`, `x2`, `x3`, `x4` y `x5` entre los instantes 0 y 100 ns.

```
entity crono1 is
end entity crono1;

architecture crono1 of crono1 is
  signal x1, x2, x3, x4, x5 : std_logic;
begin
  x1 <= '1', '0' after 10 ns,
        '1' after 25 ns,'0' after 30 ns,
        '1' after 40 ns;
  Proc1: process
  begin
    x2 <= '1';
    wait for 10 ns;
    x2 <= '0';
    wait for 15 ns;
    x2 <= '1';
    wait for 20 ns;
    x2 <= '0';
  end process;
   x3 <= (x1 xor x2) after 15 ns;
   x4 <= x1 xor x2;
   x5 <= transport (x1 xor x2) after 15 ns;
end architecture crono1;
```

Pregunta 2 (2 puntos)

Escriba en VHDL, de las formas que se detallan a continuación, la **architecture** que describe el comportamiento de un circuito combinacional codificador de 4 a 2 con prioridad. A continuación, se muestran la **entity** del circuito y su tabla de la verdad.

```
entity codificadorPrioridad4a2 is
   port (  codigo     : out std_logic_vector(1 downto 0);
           activo     : out std_logic;
           x          : in  std_logic_vector(3 downto 0) );
end entity codificadorPrioridad4a2;
```

x	codigo	activo
1 – – –	11	1
0 1 – –	10	1
0 0 1 –	01	1
0 0 0 1	00	1
0 0 0 0	00	0

En el código VHDL de la **architecture**, emplee para evaluar la señal `codigo`:

2.a) (0.5 puntos) Una asignación concurrente condicional (**when** - **else**).

2.b) (0.5 puntos) Una asignación concurrente de selección (**with** - **select**).

2.c) (0.5 puntos) Una sentencia **if**.

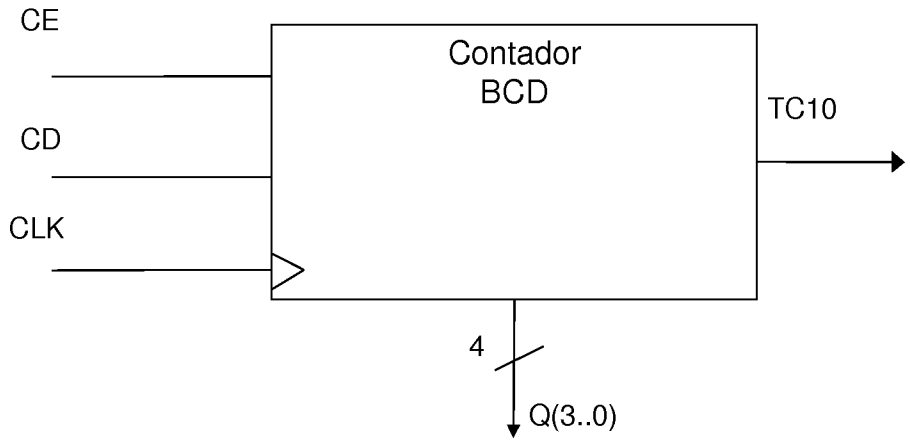
2.d) (0.5 puntos) Una sentencia **case**.

Pregunta 3 (2 puntos)

Programe el banco de pruebas del circuito combinacional que ha diseñado en la Pregunta 2. 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.

Pregunta 4 (4 puntos)

Diseñe usando VHDL un circuito contador síncrono BCD cuyo símbolo se muestra en la siguiente figura. El circuito tiene las siguientes señales de entrada: una señal de reloj (`CLK`), una señal de un bit (`CE`) y una señal de un bit (`CD`). Tiene las dos siguientes señales de salida: una señal de 4 bits (`Q`) y una señal de un bit (`TC10`).



El circuito tiene el comportamiento descrito a continuación.

La señal `CD` tiene prioridad sobre la señal `CE`. Mientras la señal de entrada `CD` tiene el valor '1', la señal de salida `Q` tiene el valor "0000".

Cuando la señal de entrada `CE` tiene el valor '1', la señal de salida `Q` toma cíclicamente en el flanco de subida de la señal de reloj los valores "0000", "0001", "0010", "0011", "0100", "0101", "0110", "0111", "1000" y "1001". Si la señal de entrada `CE` tiene el valor '0', se mantiene el valor de la señal de salida `Q`.

La señal de salida `TC10` tiene valor '1' mientras la señal `CE` tiene el valor '1' y la señal `Q` tiene el valor "1001".

Escriba en VHDL la **architecture** que describe el comportamiento del circuito en términos de una máquina de Moore. Dibuje el diagrama de estados correspondiente al circuito que ha diseñado.

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)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `y1`, `y2`, `y3` e `y4` entre los instantes 0 y 400 ns.

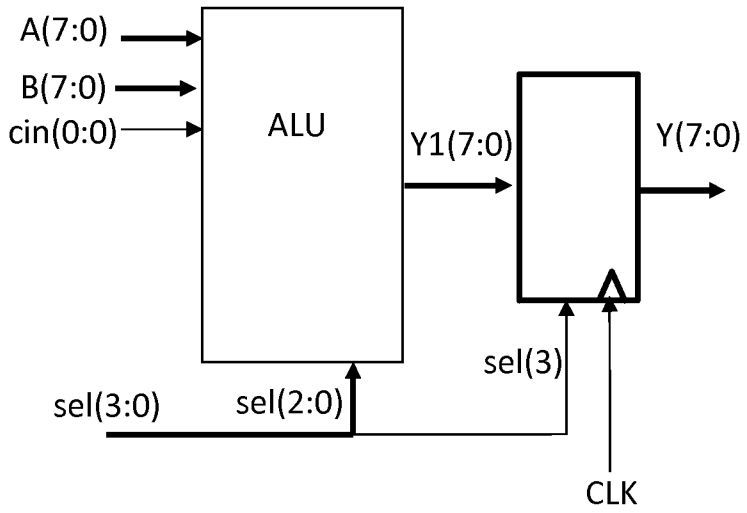
```
library IEEE;
use IEEE.std_logic_1164.all;

entity crono2 is
end entity crono2;

architecture crono2 of crono2 is
 signal y1, y2, y3, y4: std_logic;
begin
    y1 <= '0',
          '1' after 100 ns;
    y2 <= '1',
          '0' after 100 ns,
          '1' after 200 ns,
          '0' after 250 ns;
    y3 <= y2 after 100 ns;
    y4 <= transport y2 after 100 ns;
  Proc1: process
  begin
    y1 <= '0';
    wait for 50 ns;
    y1 <= '0';
  end process;
end architecture crono2;
```

Pregunta 2 (3 puntos)

A continuación, se muestra el circuito, la tabla de operaciones y la **entity** de una ALU seguida por un registro que opera en el flanco de subida de la señal de reloj.



Unidad	Sel	Operación
ALU	-000	A+B
	-001	A+B+cin
	-010	B-1
	-011	A+1
	-100	A and B
	-101	A or B
	-110	A xor B
	-111	not B
Registro	0---	Carga
	1---	Desplaza

```
entity ALUReg is
   port( Y    : out std_logic_vector ( 7 downto 0);
         A, B : in std_logic_vector (7 downto 0);
         sel  : in std_logic_vector ( 3 downto 0);
         cin  : in std_logic_vector(0 downto 0);
         CLK  : in std_logic);
end entity ALUReg;
```

La ALU realiza operaciones sobre dos operandos de 8 bits, denominados `A` y `B`. La operación que realiza la ALU se especifica con los tres bits menos significativos de la señal `sel`. La salida de la ALU (señal `Y1`) es la entrada de un registro. El registro realiza las dos siguientes operaciones que se especifican con `sel(3)`:

- Cuando la señal `sel(3)` tiene el valor '0' se carga en el registro la señal `Y1` en el flanco de subida de la señal de reloj.
- Cuando la señal `sel(3)` tiene el valor '1' se desplaza el contenido del registro 1 bit a la derecha y se introduce un cero por la izquierda.

Escriba en VHDL la **architecture** que describe el comportamiento del circuito, empleando

para ello una sentencia de asignación concurrente de selección (**with–select**) para describir el comportamiento de la ALU y un 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
```

Pregunta 3 (2 puntos)

Programe en VHDL un banco de pruebas para el circuito que ha diseñado al contestar a la Pregunta 2. La señal de reloj (`CLK`) debe tener un periodo de 100 ns e inicialmente valer '0'. El programa de test debe realizar las acciones siguientes:

1. *Dar valores a las señales* `A`, `B` *y* `cin` .
   `A` ha de valer "00000000". `B` ha de valer "10000000". `cin` ha de valer "0".
2. *Realizar la operación A or B y cargar el resultado en el registro.*
3. *Realizar tres operaciones de desplazamiento del registro.*
4. *Realizar la operación A xor B y cargar el resultado en el registro.*

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.

Pregunta 4 (3 puntos)

Diseñe un circuito secuencial síncrono capaz de detectar cuando le llega la secuencia de bits "0101" por su entrada serie de un bit. El circuito no detecta secuencias solapadas. Por ejemplo, en la secuencia "01010111" se detectaría una única vez la secuencia "0101". La **entity** del circuito se muestra a continuación.

```
entity detector is
   port( Y     : out std_logic;
         state : out std_logic_vector(2 downto 0);
         X     : in std_logic;
         reset : in std_logic;
         clk   : in std_logic);
end entity detector;
```

El circuito tiene una señal de reloj (`clk`), un entrada serie de un bit (`X`), una señal de reset asíncrona activa en '1' (`reset`), una señal que indica el estado en que se encuentra el circuito (`state`) y una señal de salida de un bit (`Y`).

La señal `Y` se pone a '1' cuando se detecta la secuencia "0101" sin solapamientos.

La señal `reset` pone el circuito en su estado inicial. Todos los cambios tienen lugar en el flanco de subida de la señal de reloj.

Escriba en VHDL la **architecture** que describe el comportamiento del circuito en términos de una máquina de Moore. Dibuje el diagrama de estados correspondiente al circuito que ha diseñado indicando la valor de la señal de salida `Y` en cada estado.

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.

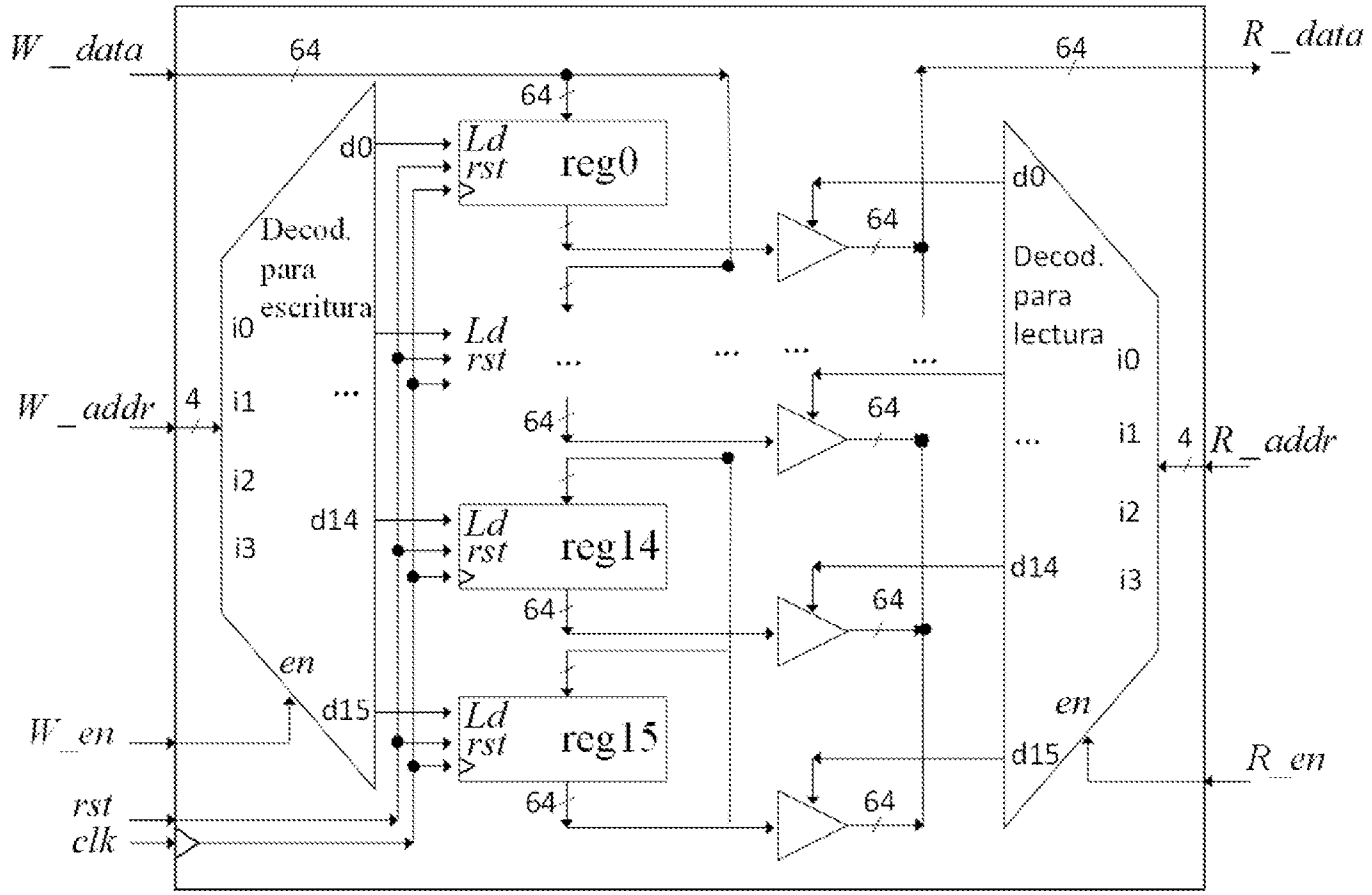
Pregunta 1 (2 puntos)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `x1`, `x2`, `x3`, `x4`, `x5` y `x6` entre los instantes 0 y 200 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono1 is
end entity crono1;
architecture crono1 of crono1 is
   signal x1, x2 : std_logic := '0';
   signal x3, x4, x5, x6 : std_logic;
begin
 process (x1)
    variable temp1, temp2, temp3: std_logic;
  begin
     if (rising_edge(x1)) then
     temp1 := x2;
     temp2 := temp1;
     temp3 := temp2;
     x3 <= temp3;
     x4 <= x2;
     x5 <= x4;
     x6 <= x5;
    end if;
  end process;
  x1 <= not x1 after 20 ns;
  x2 <= '0',  '1' after 30 ns, '0' after 90 ns;
end architecture crono1;
```

Pregunta 2 (3 puntos)

Se pretende diseñar un *register file* de tamaño 16*64 que opera en el flanco de subida de la señal de reloj. Es decir, que contiene 16 registros, cada uno de los cuales tiene 64 bits. Este circuito se puede diseñar empleando dos decodificadores, 16 registros de 64 bits y 16 buffer triestado de 64 bits, tal como se muestra en la siguiente figura.



El circuito tiene una señal de salida `R_data` de 64 bits y una señal de entrada `W_data` de 64 bits.

Tiene asímismo una señal síncrona de reset a nivel alto llamada `rst` y una señal de reloj `clk`. La señal de reset pone todos los bits de todos los registros a cero.

Las señales de entrada `W_en` y `R_en` se emplean para habilitar la escritura y la lectura, respectivamente.

Las señales de entrada `W_addr` y `R_addr` se emplean para direccionar uno de los 16 registros.

Cuando la señal `rst` tiene valor '0' y la señal `W_en` tiene valor '1', el registro cuya dirección sea `W_addr` se carga en el flanco de subida de la señal de reloj con la señal de entrada `W_data`.

Cuando la señal `R_en` tiene valor '1', la señal de salida `R_data` toma el valor de la salida del registro cuya dirección sea `R_addr`. Por el contrario, cuando la señal `R_en` tiene valor '0', la señal de salida `R_data` toma el valor de alta impedancia.

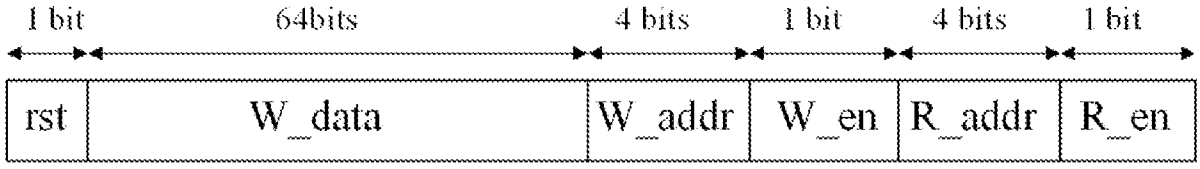
La **entity** del circuito se muestra a continuación.

```
entity RegFile16x64 is
   port ( R_data : out std_logic_vector(63 downto 0);
          W_data : in  std_logic_vector(63 downto 0);
          R_addr, W_addr: in std_logic_vector(3 downto 0);
          R_en, W_en: in std_logic;
          clk, rst: in std_logic );
end entity RegFile16x64;
```

Escriba en VHDL la **architecture** que describe el comportamiento del *register file* empleando para ello dos bloques **process**.

Pregunta 3 (2.5 puntos)

Diseñe un banco de pruebas para el *register file* que ha diseñado en la Pregunta 2 que permita realizar una inspección visual de las señales de entrada y de salida del circuito. El banco de pruebas debe generar una señal de reloj de periodo 10 ns y valor inicial '0'. El resto de valores de las señales de entrada los ha de leer de un fichero llamado *vectores.txt*. Cada una de las líneas de este fichero de texto consta de una palabra de 75 bits cuyo significado se muestra en la siguiente figura.



Pregunta 4 (2.5 puntos)

Diseñe un circuito secuencial síncrono capaz de detectar si el número de unos que le ha llegado por su entrada serie de un bit es par o impar. La **entity** del circuito se muestra a continuación.

```
entity detector is
   port( Y     : out std_logic;
         X     : in std_logic;
         reset : in std_logic;
         clk   : in std_logic);
end entity detector;
```

El circuito tiene una señal de reloj (`clk`), un entrada serie de un bit (`X`), una señal de reset asíncrona activa en '1' (`reset`) y una señal de salida de un bit (`Y`).

La señal `reset` pone el circuito en su estado inicial. Este estado inicial indica que no ha llegado ningún '1' por su entrada serie. Es decir, que el número de unos que ha llegado hasta el momento es un número par. Todos los cambios tienen lugar en el flanco de subida de la señal de reloj.

La señal `Y` se pone a '1' sólo si por la entrada `X` ha llegado un número impar de unos. Por el contrario, si el número de unos que ha llegado por la entrada `X` es par, la señal `Y` se pone a '0'.

Escriba en VHDL la **architecture** que describe el comportamiento del circuito en términos de una máquina de Moore. Dibuje el diagrama de estados correspondiente al circuito que ha diseñado.

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)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `clk`, `x`, `y1` e `y2` entre los instantes 0 y 1000 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono2 is
end entity crono2;
architecture crono2 of crono2 is
  signal x, y1, y2 : std_logic;
  signal clk : std_logic:='0';
begin
   process (clk)
    variable a, b, c: std_logic;
   begin
      if ( rising_edge(clk) ) then
         a := x;
         b := a;
         c := b;
         y1 <= c;
         y2 <= y1;
      end if;
   end process;
  clk <= not clk after 100 ns;
  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;
end architecture crono2;
```

Pregunta 2 (3 puntos)

Diseñe un circuito secuencial síncrono que permita controlar el funcionamiento de una máquina expendedora sencilla. El usuario de la máquina puede introducir únicamente monedas de 5 céntimos y 10 céntimos. Cuando se introducen exactamente 15 céntimos, la máquina expendedora saca un chicle. Este sistema de control debe emitir una señal de apertura (señal `abrir`) cuando ha detectado que se han introducido 15 céntimos (una moneda de 5 céntimos y una moneda de 10 céntimos, o tres monedas de 5 céntimos). El sistema de control recibe dos señales de entrada: `cinco` y `diez`. La señal `cinco` es una señal de entrada del circuito que tiene valor '1' durante un tiempo sólo si el usuario introduce una moneda de 5 céntimos. La señal `diez` es una señal de entrada del circuito que tiene valor '1' durante un tiempo sólo si el usuario introduce una moneda de 10 céntimos .

La **entity** del circuito se muestra a continuación.

```
entity maquina is
   port( abrir     : out std_logic;
         cinco     : in std_logic;
         diez : in std_logic;
         reset : in std_logic;
         clk   : in std_logic);
end entity maquina;
```

El circuito tiene una señal de reloj (`clk`), dos entradas de un bit (`cinco` y `diez`), una señal de reset asíncrona activa en '1' (`reset`) y una señal de salida de un bit (`abrir`).

La señal `reset` pone el circuito en su estado inicial. Este estado inicial indica que no ha recibido ninguna moneda.

La señal `abrir` se pone a '1' sólo si la máquina ha rebido exactamente 15 céntimos. Esta señal ha de permanece con el valor '1' sólo durante un periodo de la señal de reloj `clk`.

Escriba en VHDL la **architecture** que describe el comportamiento del circuito en términos de una máquina de Moore. Dibuje el diagrama de estados correspondiente al circuito que ha diseñado.

Pregunta 3 (3 puntos)

Diseñe usando VHDL un contador módulo 16 ascendente/ descendente con carga paralela y señal reset asíncrona activa a nivel alto. Realice el diseño describiendo el comportamiento del circuito. El circuito opera en el flanco de subida de la señal de reloj. Tiene la siguiente **entity**:

```
entity contador is
   port( Q:  out std_logic_vector( 3 downto 0);
         D: in std_logic_vector( 3 downto 0);
         Clk: in std_logic;
         Load: in std_logic;
         Reset: in std_logic;
         Count: in std_logic;
         Down: in std_logic );
end entity contador;
```

A continuación se describe el funcionamiento del circuito contador.

- *Reset asíncrono activo a nivel alto.*

  Mientras `Reset` vale '1', el contador tiene el valor "0000".

- *Carga síncrona.*

  Mientras `Reset` vale '0' y la señal `Load` tiene valor '1', se carga el valor de la señal de entrada `D` en el registro en el flanco de subida de la señal de reloj.

- *Cuenta síncrona ascendente.*

  Mientras `Reset` vale '0', la señal `Load` tiene valor '0', la señal `Count` que habilita la cuenta vale '1' y la señal `Down` vale '0', se realiza la cuenta síncrona ascendente. Es decir, el contador pasa cíclicamente en cada flanco de subida de la señal de reloj por la secuencia "0000", "0001", "0010", "0011", "0100", "0101", "0110", "0111", "1000", "1001", "1010", "1011", "1100", "1101", "1110" y "1111".

- *Cuenta síncrona descendente.*

  Mientras `Reset` vale '0', la señal `Load` tiene valor '0', la señal `Count` que habilita la cuenta vale '1' y la señal `Down` vale '1', se realiza la cuenta síncrona descendente. Es decir, el contador pasa cíclicamente en cada flanco de subida de la señal de reloj por la secuencia "1111", "1110", "1101","1100","1011", "1010", "1001", "1000", "0111","0110","0101","0100", "0011","0010","0001" y "0000".

- *Cuenta deshabilitada.*

  Mientras `Reset` vale '0', la señal `Load` tiene valor '0' y la señal `Count` que habilita la cuenta vale '0', el contador mantiene el valor de la señal de salida `Count`.

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. *Realizar durante tres periodos de la señal de reloj la cuenta ascendente.*
3. *Cargar en la señal de salida* `Count` *el valor "1111".*
4. *Realizar durante cuatro periodos de la señal de reloj la cuenta descendente.*
5. *Deshabilitar la cuenta.*

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)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `a`, `b`, `c`, `d` entre los instantes 0 y 100 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono1 is
end entity crono1;
architecture crono1 of crono1 is
   signal a, b, c, d : std_logic;
begin
    bloque1 : process
    begin
       b <= '0';
       wait for 5 ns;
       a <= '1';
       b <= '1';
       wait for 10 ns;
       a <= '0';
       b <= '0';
       wait for 5 ns;
       a <= '1';
       wait for 15 ns;
       b <= '1';
       wait for 5 ns;
       b<= '1';
    end process bloque1;
    bloque2 : process
    begin
       wait for 20 ns;
       a <= '1';
       wait for 20 ns;
       a <= '0';
    end process bloque2;
    c <= b after 10 ns;
    d <= transport b after 10 ns;
end architecture crono1;
```

Pregunta 2 (2.5 puntos)

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

a. (1 punto) Una latch D con entrada enable activa a nivel alto y cuya **entity** se muestra a continuación. Emplee en la descripción de la **architecture** una sentencia **if**.

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

b. (1.5 puntos) Un biestable (*flip-flop*) JK disparado por el flanco de subida del reloj (`clk`) y con reset asíncrono (`reset_n`) activado a nivel bajo. La **entity** se muestra a continuación.

```
entity biestableJK is
   port ( q, q_n              : out std_logic;
          clk, J, K, reset_n  : in std_logic );
end entity biestableJK;
```

Pregunta 3 (3.5 puntos)

Diseñe un circuito secuencial síncrono capaz de detectar cuando le llegan al menos tres ceros consecutivos por su entrada. La **entity** del circuito se muestra a continuación. El circuito tiene una señal de reloj (`clk`), un entrada serie de un bit (`X`), una señal de reset asíncrona activa en '1' (`reset`), una señal que indica el estado en que se encuentra el circuito (`state`) y una señal de salida de un bit (`Y`). La señal `Y` se pone a '1' si por la entrada `X` se han recibido tres o más ceros consecutivos. La señal `reset` pone el circuito en su estado inicial. Todos los cambios tienen lugar en el flanco de subida de la señal de reloj. Escriba en VHDL la **architecture** que describe el comportamiento del circuito en términos de una máquina de Moore. Dibuje el diagrama de estados correspondiente al circuito que ha diseñado.

```
entity detector is
   port( Y     : out std_logic;
         state : out std_logic_vector(1 downto 0);
         X     : in std_logic;
         reset : in std_logic;
         clk   : in std_logic);
end entity detector;
```

Pregunta 4 (2 puntos)

Programe el banco de pruebas del circuito secuencial que ha diseñado en la Pregunta 3. 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.

INGENIERÍA DE COMPUTADORES III

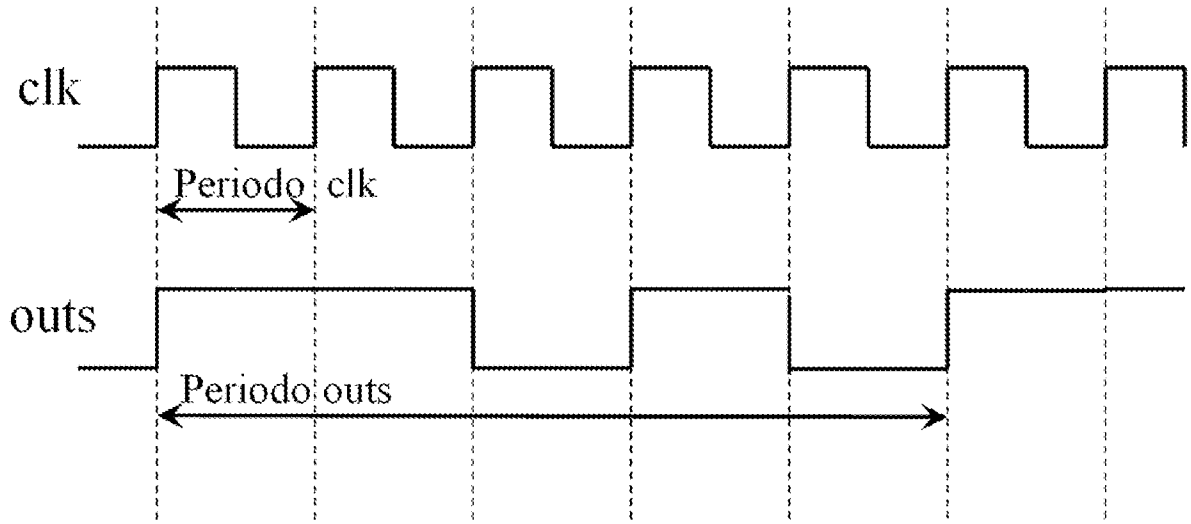
INSTRUCCIONES
Por favor, entregue esta primera hoja de enunciado junto con el examen.
Dispone de 2 horas para realizar el examen.
MATERIAL PERMITIDO: Ninguno.

Pregunta 1 (2 puntos)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `x1`, `x2`, `x3`, `x4` y `x5` 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, x5 : std_logic;
begin
  x1 <= '1', '0' after 5 ns,
        '1' after 15 ns,'0' after 20 ns,
        '1' after 40 ns;
  Proc1: process
  begin
    x2 <= '1';
    wait for 5 ns;
    x2 <= '0';
    wait for 15 ns;
    x2 <= '1';
    wait for 10 ns;
    x2 <= '0';
  end process;
   x3 <= x1 after 10 ns;
  Proc2: process
     variable valor  : std_logic;
  begin
    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;
```

Pregunta 2 (2.5 puntos) Diseñe un generador de señales que obtenga la forma de onda mostrada en la parte inferior de la figura siguiente (señal `outs`) a partir de una señal de reloj `clk`. Describa su comportamiento como una máquina de estado finito de Moore.



Pregunta 3 (3.5 puntos)

Diseñe usando VHDL un desplazador de nBits bits que opere en el flanco de subida de la señal de reloj, con carga de datos síncrona, reset síncrono activo a nivel alto y con una señal para seleccionar desplazamiento a la izquierda o a la derecha. El desplazador tiene la siguiente **entity**.

```
entity desplazador is
   generic (nBits: integer:=8);
   port (  dout : out std_logic_vector( nBits-1 downto 0);
           data: in std_logic_vector( nBits-1 downto 0);
           clk, load, izqda, reset  : in  std_logic );
end desplazador;
```

Las entradas al circuito son la señal de reloj (`clk`), la señal de carga (`load`), la señal de reset (`reset`), la señal que selecciona el tipo de desplazamiento (`izqda`) y la señal `data`. El circuito tiene una única señal de salida llamada `dout` y una constante *generic* que indica el número de bits de las señales de entrada y salida de datos.

La señal de carga tiene prioridad sobre la señal de reset, y la señal de reset tiene prioridad sobre la operación de desplazamiento. Cuando las señales `load` y `reset` valen '0', se realiza en el flanco de subida de la señal de reloj el desplazamiento de un bit a la izquierda o a la derecha dependiendo del valor de la señal `izqda`. Cuando la señal `izqda` vale '0', se realiza el desplazamiento de un bit a la derecha introduciendo un '0' a la izquierda y cuando vale '1' se realiza el desplazamiento de un bit a la izquierda introduciendo un '0' a la derecha.

Pregunta 4 (2 puntos)

Programe en VHDL un banco de pruebas para el circuito que ha diseñado al contestar a la Pregunta 3 para un valor de la constante `nBits` igual a 8. 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*. La señal `reset` ha de tener el valor '1' durante 15 ns.
2. *Cargar el valor "11111100"*.
3. *Realizar 4 desplazamientos a la izquierda*. Comprobar que el circuito pasa consecutivamente por los valores "11111000", "11110000", "11100000" y "11000000".
4. *Cargar el valor "10111100"*.
5. *Realizar 4 desplazamientos a la derecha*. Comprobar que el circuito pasa por los valores "01011110", "00101111", "00010111" y "00001011".

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.

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)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `x1`, `x2`, `x3` y `x4` entre los instantes 0 y 100 ns.

```
use IEEE.std_logic_1164.all;
entity crono1 is
end entity crono1;
architecture crono1 of crono1 is
  signal x1, x2, x3, x4 : std_logic;
begin
  proc0: process
  begin
     x1  <=  TRANSPORT '0' AFTER 10 NS;
     x1  <=  TRANSPORT '1' AFTER 20 NS;
     x1  <=  TRANSPORT '1' AFTER 15 NS;
     x1  <=  TRANSPORT '0' AFTER 30 NS;
     wait;
  end process;
  Proc1: process
  begin
    x2 <= '1';  wait for 10 ns;
    x2 <= '0';  wait for 20 ns;
    x2 <= '1';  wait for 5 ns;
    x2 <= '0';
  end process;
   x3 <= x1 xor x2 after 10 ns;
  Proc2: process
  begin
    for i in 0 to 3 loop
       x4 <= x2;  wait for 5 ns;
     end loop;
     wait;
  end process;
end architecture crono1;
```

Pregunta 2 (2.5 puntos)

Se pretende diseñar un circuito comparador cuya salida (`F`) vale '1' si el valor del número de cuatro bits de entrada (`x`) es mayor que el número decimal 9. Los cuatro bits de entrada se interpretan como un número binario sin signo. La **entity** del circuito se muestra a continuación.

```
entity comparaXmayor9 is port
 ( F : out std_logic;
   x : in  std_logic_vector(3 downto 0) );
end entity comparaXmayor9;
```

2.a) (1 punto) Escriba la tabla de verdad de la salida (`F`) en función de la entrada (`x`). Escriba la función lógica (`F`) en función de (`x`) obtenida a partir de dicha tabla de verdad. Escriba en VHDL la **architecture** del circuito comparador empleando únicamente sentencias de asignación concurrente y operadores lógicos.

2.b) (1.5 puntos) Programe el banco de pruebas del circuito combinacional que ha diseñado en la Pregunta 2.a. 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.

Pregunta 3 (2.5 puntos)

Escriba el código VHDL de la **architecture** que describe el comportamiento de un circuito contador binario de 3 bits cuya salida sigue cíclicamente la secuencia "001", "100", "110" y "111", con señal de reset asíncrona activa a nivel alto. La **entity** del circuito se muestra a continuación.

```
entity contador is
 port (  count          : out std_logic_vector (2 downto 0);
         clk, reset, c : in  std_logic );
end contador;
```

Las entradas al circuito son la señal de reloj (`clk`), la señal de reset asíncrono activo a nivel alto (`reset`) y la señal que habilita la cuenta (`c`). La salida del circuito contador es la señal de 3 bits `count`.

El funcionamiento del circuito debe ser el siguiente:

- *Reset asíncrono activo a nivel alto.*

  Mientras `reset` vale '1', el contador tiene el valor "001".

- *Cuenta síncrona.*

  Mientras `reset` vale '0' y la señal `c` que habilita la cuenta vale'1', el contador pasa cíclicamente en cada flanco de subida de la señal de reloj por la secuencia "001", "100", "110" y "111".

- *Cuenta deshabilitada.*

  Mientras `reset` vale '0' y la señal `c` que habilita la cuenta vale'0', el contador mantiene el valor de la señal de salida `count`.

Describa el comportamiento del circuito en términos de una máquina de Moore. El estado del contador ha de tener el mismo valor que la señal de salida `count`. En caso de que el circuito esté en un estado distinto a los estados "001", "100", "110" y "111", debe pasar en el flanco de subida de la señal de reloj al estado "001". Dibuje el diagrama de estados en el que ha basado su diseño.

Pregunta 4 (3 puntos)

Diseñe un circuito secuencial síncrono capaz de detectar cuando le llega la secuencia de bits "1101" por su entrada serie de un bit. La **entity** del circuito se muestra a continuación.

```
entity detector is
   port( Y     : out std_logic;
         state : out std_logic_vector(1 downto 0);
         X     : in std_logic;
         reset : in std_logic;
         clk   : in std_logic);
end entity detector;
```

El circuito tiene una señal de reloj (`clk`), un entrada serie de un bit (`X`), una señal de reset asíncrona activa en '1' (`reset`), una señal que indica el estado en que se encuentra el circuito (`state`) y una señal de salida de un bit (`Y`).

La señal `Y` se pone a '1' si por los últimos cuatro bits de su entrada `X` ha llegado la secuencia "1101". El circuito ha de ser capaz de detectar secuencias solapadas.

La señal `reset` pone el circuito en su estado inicial. Todos los cambios tienen lugar en el flanco de subida de la señal de reloj.

Escriba en VHDL la **architecture** que describe el comportamiento del circuito en términos de una máquina de Mealy. Dibuje el diagrama de estados correspondiente al circuito que ha diseñado, indicando en cada arco de transición los valores de la señal de entrada `X` y de la señal de salida `Y`.

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)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `clk`, `x`, `y1` e `y2` entre los instantes 0 y 1000 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono2 is
end entity crono2;
architecture crono2 of crono2 is
  signal x, y1, y2 : std_logic;
  signal clk : std_logic:='0';
begin
   process (clk)
    variable a, b, c: std_logic;
   begin
      if ( rising_edge(clk) ) then
         a := x;
         b := a;
         c := b;
         y1 <= c;
         y2 <= y1;
      end if;
   end process;
  clk <= not clk after 100 ns;
  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;
end architecture crono2;
```

Pregunta 2 (3 puntos)

2.a) (0.75 puntos) Escriba en VHDL la **architecture** de un multiplexor 4:1 empleando una sentencia concurrente condicional (**when - else**). La **entity** del multiplexor 4:1 se muestra a continuación.

```
entity mux4_1 is
  port (  y: out std_logic;
          s: in std_logic_vector(1 downto 0);
          x: in std_logic_vector(3 downto 0)    );
end entity mux4_1;
```

2.b) (0.25 puntos) Dibuje el diagrama circuital de un circuito desplazador de barril de 4 bits empleando para ello únicamente multiplexores 4:1 como el descrito en el apartado 2.a. Este circuito desplazador tiene una señal de entrada `s` de 2 bits para seleccionar la operación, una señal de entrada `x` de 4 bits y una señal de salida `y` de 4 bits. La tabla de operaciones del circuito se muestra a continuación.

s	Operación	y
”00”	Ninguna	x(3)x(2)x(1)x(0)
”01”	Rota 1 bit a la izquierda	x(2)x(1)x(0)x(3)
”10”	Rota 2 bits a la izquierda	x(1)x(0)x(3)x(2)
”11”	Rota 3 bits a la izquierda	x(0)x(3)x(2)x(1)

2.c) (2 puntos) Escriba en VHDL la **architecture** que describe la estructura del circuito desplazador siguiendo el diagrama dibujado en el apartado anterior y empleando el multiplexor cuyo diseño ha realizado al contestar el primer apartado. La **entity** del circuito se muestra a continuación.

```
entity despBarril 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 despBarril;
```

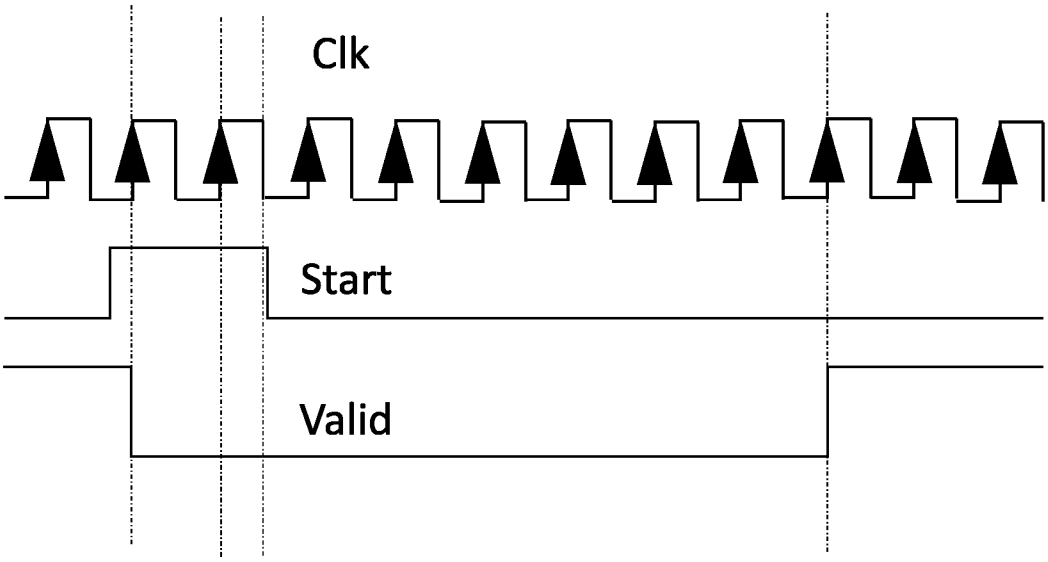
Pregunta 3 (3 puntos)

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 síncrono. 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.

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 pueden 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`.



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`.

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.

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)

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 crono is
end entity crono;
architecture crono of crono is
  signal x1, x2, x3, x4 : std_logic;
begin
  x2 <= '0', '1' after 25 ns,
        '0' after 35 ns,'1' after 50 ns;
  Proc1: process
  begin
    x1 <= '0';
    wait for 10 ns;
    x1 <= '1';
    wait for 15 ns;
    x1 <= '0';
  end process Proc1;
  Proc2: process
  begin
    x1 <= '0';
    wait for 15 ns;
    x1 <= '1';
    wait for 10 ns;
    x1 <= '0';
    wait;
  end process Proc2;
  Proc3: process (x2)
  begin
    x3 <= x2;
    x4 <= x3;
  end process Proc3;
end architecture crono;
```

Pregunta 2 (2.5 puntos)

Diseñe un circuito combinacional que tenga una señal de entrada de 8 bits llamada `data` y una señal de salida de 4 bits llamada `zeros`. La señal `zeros` se interpreta como un número binario sin signo. Esta señal ha de tener como valor el número de ceros existentes en `data` antes de encontrar el primer '1', empezando a contar por el extremo izquierdo (bit más significativo) de `data` . Por ejemplo, el valor de la señal de entrada `data` "00110110" da como resultado un valor de la señal `zeros` de "0010" (valor 2 en notación decimal).

El diseño ha de tener la siguiente **entity**:

```
entity numceros is
   port( zeros   : out std_logic_vector ( 3 downto 0);
         data  : in std_logic_vector(7 downto 0));
end entity numceros;
```

Pregunta 3 (3.5 puntos)

Diseñe usando VHDL un contador binario ascendente de 8 bits que opere en el flanco de subida de la señal de reloj, con carga de datos síncrona y reset asíncrono. El contador tiene la siguiente **entity**.

```
entity contador is
   port (  count : out std_logic_vector( 7 downto 0);
           data: in std_logic_vector( 7 downto 0);
           clk, reset, load  : in  std_logic );
end contador;
```

Las entradas al circuito son la señal de reloj (`clk`), la señal de reset (`reset`) asíncrona y activa a nivel bajo, la señal de carga (`load`) y la señal `data`. El circuito tiene una única señal de salida llamada `count`.

Cuando la señal `reset` vale '0', todos los bits de la señal `count` se ponen a 0. En caso contrario, en cada flanco de subida de la señal de reloj, si la señal `load` vale '0' se incrementa en '1' el valor de la señal de salida y si `load` vale '1' se asigna a la señal de salida el valor de la señal `data`.

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 '1'. El programa de test debe realizar consecutivamente las acciones siguientes:

1. *Reset*. La señal de reset ha de tener el valor '0' durante 15 ns.
2. *Empezar la cuenta*. Comprobar que el circuito pasa consecutivamente por los valores "00000000" a "00000011".
3. *Cargar el valor "11111100"*. La señal `load` ha de valer '1' . La señal `data` ha de tener el valor "11111100".
4. *Seguir la cuenta*. Comprobar que el circuito pasa por los valores "11111100", "11111101", "11111110", "11111111" y "00000000".

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.

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)

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;
```

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.

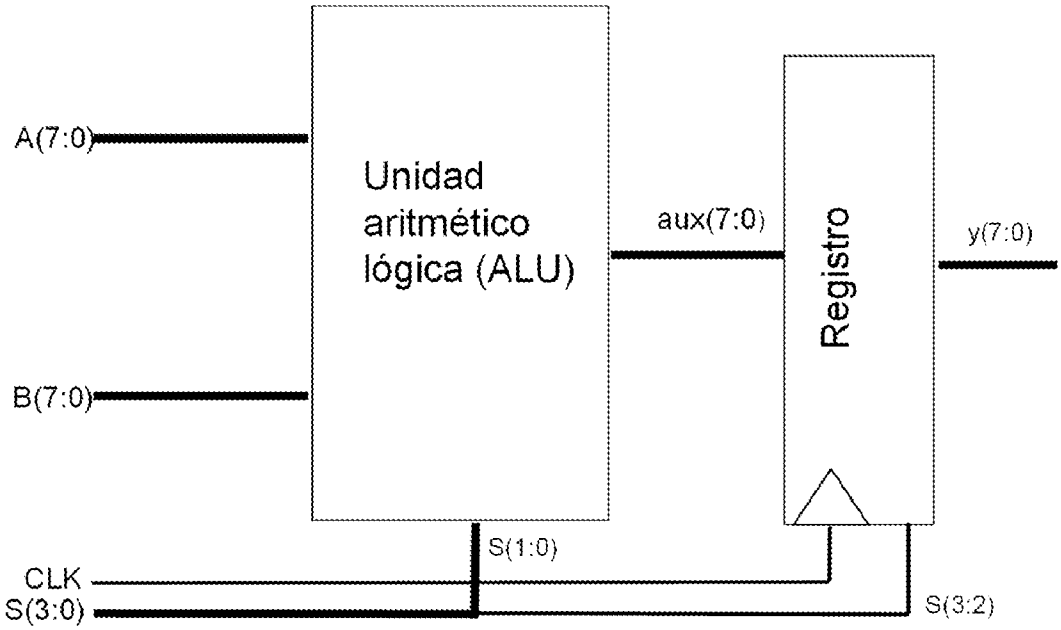


Tabla 1: Operaciones del registro.

`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

Tabla 2: Operaciones de la ALU.

S(1)S(0)	**Operación**
00	A + B
01	A - B
10	A and B
11	A or B

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;
```

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", ...

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.

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)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `z1`,`z2`, `z3` y `z4` entre los instantes 0 y 60 ns.

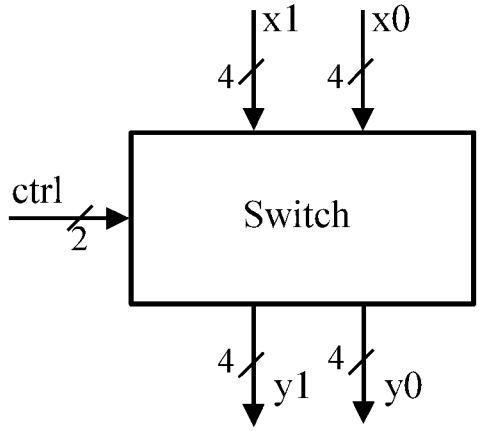
```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono1 is
end entity crono1;

architecture crono1 of crono1 is
  constant PER : time :=10 ns;
  signal z1: std_logic:='0';
  signal z2, z3, z4: std_logic;
begin
  process is
  begin
    z2<='0';     wait until falling_edge(z1);
    z2<='1';     wait until rising_edge(z1);
    z2<='0';     wait until falling_edge(z1);
    z2<='1';     wait until rising_edge(z1);
    z2<='0';     wait until rising_edge(z1);
    wait;
  end process;
  process (z1)
  begin
    if(rising_edge(z1)) then
      z3<=z2;
    end if;
    z4<=z3;
  end process;
  z1<=not z1 after (PER/2);
end architecture crono1;
```

Pregunta 2 (3 puntos)

Escriba en VHDL, de las cuatro formas que se detallan a continuación, la **architecture** que describe el comportamiento de un circuito combinacional cuya **entity**, símbolo lógico y tabla de operaciones se muestran a continuación.

```
entity switch is
 port (  y1, y0      : out std_logic_vector(3 downto 0);
         ctrl        : in  std_logic_vector(1 downto 0);
         x1, x0      : in std_logic_vector(3 downto 0) );
end entity switch;
```



ctrl(1)	ctrl(0)	y1	y0
0	0	x1	x0
0	1	x0	x1
1	0	x0	x0
1	1	x1	x1

2.a) (0.75 puntos) Empleando sentencias concurrentes condicionales (**when** - **else**).

2.b) (0.75 puntos) Empleando asignaciones concurrentes de selección (**with** - **select**).

2.c) (0.75 puntos) Empleando un bloque **process** con sentencias **if**.

2.d) (0.75 puntos) Empleando un bloque **process** con sentencias **case**.

Pregunta 3 (3 puntos)

Diseñe un circuito secuencial síncrono capaz de detectar cuando los últimos 8 bits recibidos por su entrada serie (`DataSIN`) son iguales a un cierta palabra patrón de 8 bits almacenada en el circuito.

La palabra patrón es cargada a través de la entrada paralelo (`DataPIN`). La carga se produce en el flanco de subida de la señal de reloj, cuando la señal `Load` valga '1'.

Asimismo, el circuito posee una salida de 8 bits llamada `Patron`, que en todo momento muestra la palabra patrón internamente almacenada en el circuito.

Por otra parte, los últimos 8 bits recibidos por la entrada serie `DataSIN` van almacenándose en un registro interno del circuito llamado `RegCONT`, de manera que el último bit recibido por la entrada serie es el bit menos significativo de la palabra almacenada en el registro. Por ejemplo, si el valor de la señal de entrada `DataSIN` en los últimos 8 flancos de subida consecutivos de la señal de reloj (`Clk`) es '1', '1', '1', '1', '1', '1', '0', '0', entonces el contenido del registro `RegCONT` será "11111100".

El contenido del registro `RegCONT` es reseteado al valor "00000000" mediante la señal de reset síncrono (`Reset`) activa a nivel alto.

El circuito debe comparar si la palabra patrón coincide con la palabra del registro `RegCONT`. La salida `Y` debe valer '1' mientras ambas palabras de 8 bits sean iguales y debe valer '0' en caso contrario.

La **entity** del circuito se muestra a continuación.

```
entity detector is
   port( Y     : out std_logic;
         Patron: out std_logic_vector(7 downto 0);
         Clk   : in std_logic;
         Reset : in std_logic;
         Load  : in std_logic;
         DataSIN: in std_logic;
         DataPIN: in std_logic_vector(7 downto 0) );
end entity detector;
```

Escriba en VHDL la **architecture** que describe 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 programa de test debe realizar consecutivamente las acciones siguientes:

1. *Reset*. La señal de reset ha de tener el valor '1' durante 15 ns.
2. *Cargar el patrón "11111100"*. Para ello, la señal `Load` ha de valer '1' y la señal `DataPIN` ha de tener el valor "11111100".
3. *Introducir los valores adecuados por la entrada serie para que se reconozca el patrón de la entrada*. El banco de pruebas debe comprobar que la señal de salida `Y` va tomando los valores adecuados. En caso contrario, debe mostrar un mensaje indicándolo.

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;
```

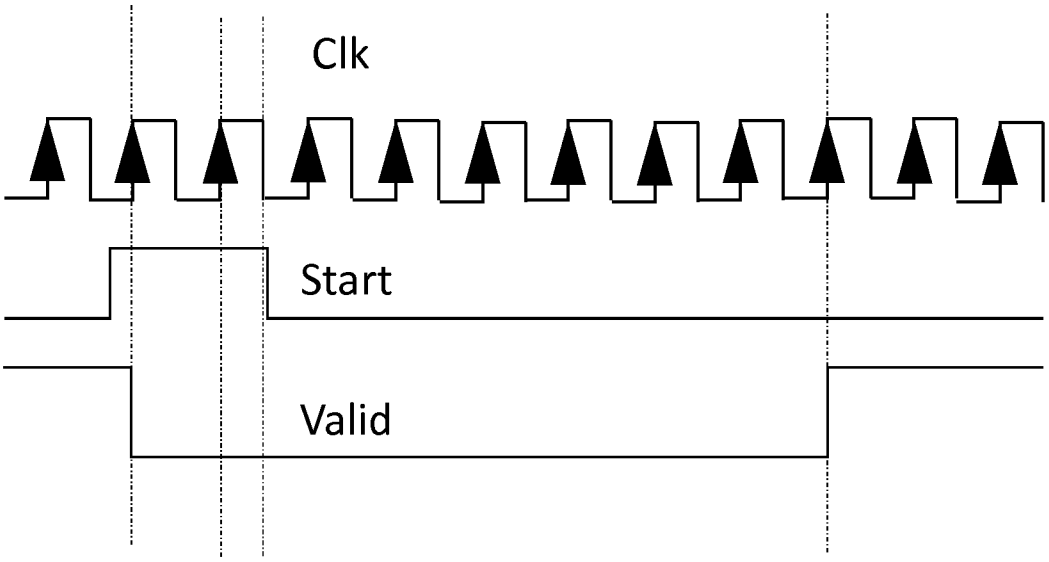
A continuación se describe el funcionamiento del circuito conversor.

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.

INGENIERÍA DE COMPUTADORES III

INSTRUCCIONES
Por favor, entregue todas las hojas de enunciado junto con el examen.
Dispone de 2 horas para realizar el examen.
MATERIAL PERMITIDO: Ninguno.

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 60 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono1 is
end entity crono1;

architecture crono1 of crono1 is
   signal x1, x2, x3, x4 : std_logic;
   signal x5 : std_logic := '0';
begin
 process (x1, x5)
    variable temp  : std_logic;
  begin
     temp := x1;
     x2 <= x1;
     x3 <= x2;
     x4 <= temp;
 end process;
 x1 <= '0', '1' after 10 ns, '0' after 20 ns,
       '1' after 30 ns;
 x5 <= x1 after 15 ns;
end architecture crono1;
```

Pregunta 2 (2.5 puntos)

Escriba en VHDL, de las dos formas que se detallan a continuación, la **architecture** que describe el comportamiento de un circuito combinacional desplazador a la izquierda. El circuito ha de desplazar la señal de entrada a la izquierda de 0 a 7 bits rellenado con '0'. En la descripción del circuito no se pueden emplear ni operadores ni funciones de desplazamiento lógico. La **entity** del circuito se muestra a continuación.

```
entity desplazador is
   port ( y   : out std_logic_vector(7 downto 0);
          a   : in  std_logic_vector(7 downto 0);
          ctrl  : in  std_logic_vector(2 downto 0) );
end entity desplazador;
```

La señal `a` es la señal a desplazar y el valor de la señal `ctrl` indica el número de bits a desplazar.

2.a) (1.25 puntos) Empleando una sentencia concurrente condicional (**when - else**). Dibuje el diagrama conceptual del circuito diseñado.

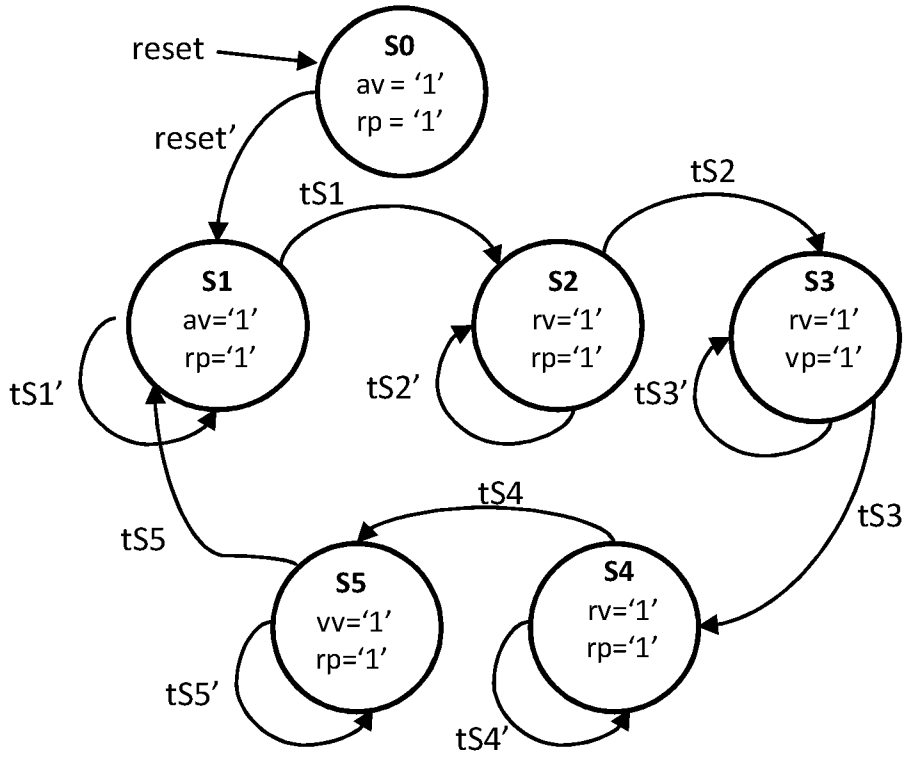
2.b) (1.25 puntos) Empleando una asignación concurrente de selección (**with - select**). Dibuje el diagrama conceptual del circuito diseñado.

Pregunta 3 (3.5 puntos) Diseñe un circuito secuencial síncrono para la regulación de un paso de peatones. El paso de peatones tiene dos semáforos: el semáforo de los vehículos y el semáforo de los peatones. El semáforo de los vehículos tiene tres lámparas: verde, amarillo y rojo. El semáforo de los peatones tiene dos lámparas: verde y rojo. La **entity** del circuito se muestra a continuación.

```
entity regulador is
   port (  rv, av, vv, rp, vp : out std_logic;
           clk, reset   : in  std_logic );
end regulador;
```

Las entradas al circuito son la señal de reloj (`clk`) de 60 Hz y la señal de reset (`reset`) asíncrona y activa a nivel alto. Las señales de salida `rv`, `av`, `vv`, `rp`, `vp` controlan, respectivamente, las lámparas roja, amarilla y verde del semáforo de vehículos y las lámparas roja y verde del semáforo de peatones. La lámpara está encendida cuando la señal que controla dicha lámpara está a '1' y está apagada cuando está a '0'.

El circuito funciona como una máquina de 6 estados cuyo diagrama de estados se muestra en la siguiente figura. Las transiciones entre estados se producen en el flanco de subida de la señal de reloj. En cada estado se indican las únicas señales de salida que valen '1' en dicho estado.



- Mientras la señal de `reset` esté a '1' el circuito permanece en el estado S0. En el estado S0 sólo están encendidas la lámpara amarilla del semáforo de vehículos y la roja del semáforo de peatones.
- En el estado S1 permanece un tiempo tS1 (5 s) y sólo están encendidas la lámpara amarilla del semáforo de vehículos y la roja del semáforo de peatones.
- En el estado S2 permanece un tiempo tS2 (5 s) y sólo están encendidas la lámpara roja del semáforo de vehículos y la roja del semáforo de peatones.
- En el estado S3 permanece un tiempo tS3 (45 s) y sólo están encendidas la lámpara roja del semáforo de vehículos y la verde del semáforo de peatones.
- En el estado S4 permanece un tiempo tS4 (10 s) y sólo están encendidas la lámpara roja del semáforo de vehículos y la roja del semáforo de peatones.
- En el estado S5 permanece un tiempo tS5 (45 s) y sólo están encendidas la lámpara verde del semáforo de vehículos y la roja del semáforo de peatones.

Escriba el código VHDL de la **architecture** que describe el comportamiento del circuito siguiendo el diagrama de estados mostrado anteriormente.

Pregunta 4 (2 puntos)

Programe en VHDL un banco de pruebas para el circuito que ha diseñado al contestar a la Pregunta 3. El banco de pruebas ha de resetear el circuito y comprobar que se producen correctamente las transiciones de estado hasta alcanzar el estado S5. 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.

INGENIERÍA DE COMPUTADORES III

INSTRUCCIONES
Por favor, entregue todas las hojas de enunciado junto con el examen.
Dispone de 2 horas para realizar el examen.
MATERIAL PERMITIDO: Ninguno.

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 80 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono2 is
end entity crono2;

architecture crono2 of crono2 is
   signal x1 : std_logic := '0';
   signal x2, x3, x4, x5 : std_logic;
begin
 process
  begin
    for i in 0 to 4 loop
      x3 <= x1;
      x4 <= x5;
      wait for 15 ns;
    end loop;
    wait;
 end process;
 x1 <= '0', '1' after 20 ns, '0' after 40 ns,
       '1' after 50 ns;
 x5 <= x1 after 15 ns;
 x2 <= x1;
end architecture crono2;
```

Pregunta 2 (2.5 puntos) Escriba en VHDL, de las dos formas que se detallan a continuación, la **architecture** que describe el comportamiento de un circuito codificador 4 a 2 con prioridad. La entrada 3 tiene mayor prioridad que la 2, ésta mayor que la 1 y finalmente la entrada 0 es la de menor prioridad. La **entity** del circuito se muestra a continuación.

```
entity codificador4a2 is
   port ( D  : out std_logic_vector(1 downto 0);
          Z  : out  std_logic;
          A  : in  std_logic_vector(3 downto 0) );
end entity codificador4a2;
```

El circuito tiene una señal de salida `Z` que ha de valer '0' sólo cuando todos los componentes de la señal A están a '0'.

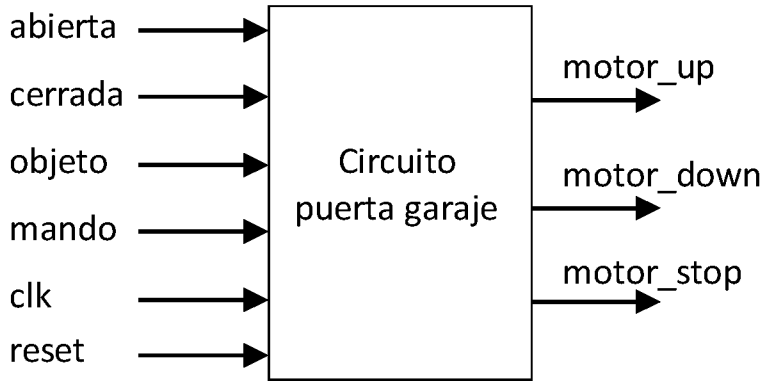
2.a) (1.25 puntos) Empleando un único bloque **process**.

2.b) (1.25 puntos) Describiendo la estructura del circuito. Siga para ello los siguientes pasos. Primero, dibuje el diagrama circuital del codificador con prioridad empleando únicamente las siguientes puertas lógicas: AND de dos entradas, OR de 2 entradas y NOT. Segundo, diseñe en VHDL dichas puertas lógicas. Finalmente, escriba la **architecture** del codificador siguiendo el diagrama circuital que ha dibujado previamente y usando las puertas lógicas que ha diseñado en el segundo paso.

Pregunta 3 (3.5 puntos)

Escriba la **architecture** que describe el comportamiento de un circuito para controlar una puerta de un garaje como una máquina de Moore síncrona, con transiciones en el flanco de subida de la señal de reloj.

La puerta del garaje tiene un motor para subir/bajar la puerta, un sensor para indicar que la puerta está completamente abierta, un sensor para indicar que la puerta está completamente cerrada, un mando remoto con un botón para accionar la puerta y una fotocélula para detectar si existe o no un objeto en la misma. El símbolo lógico del circuito y la **entity** del circuito se muestran a continuación.

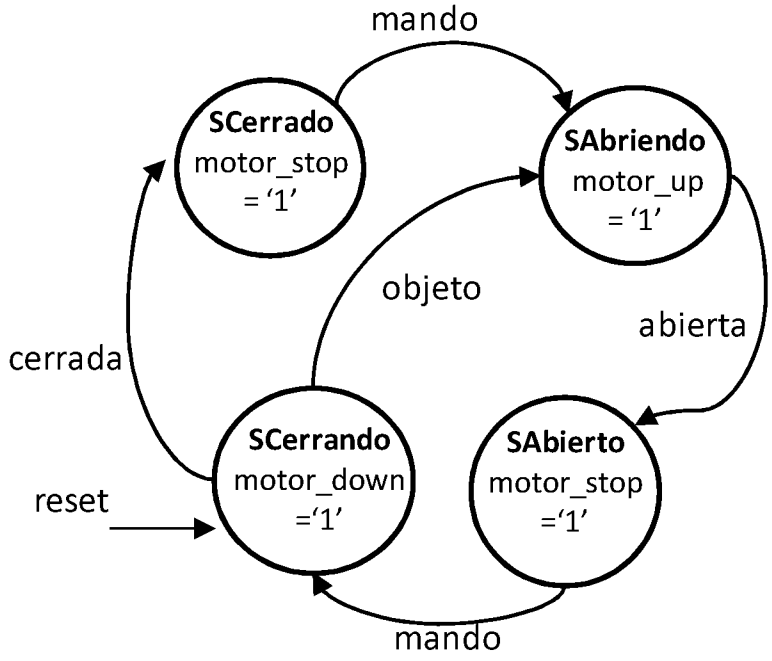



```
entity reguladorPuertaG is
  port (  motor_up, motor_down, motor_stop : out std_logic;
          abierta, cerrada, objeto, mando, clk, reset  : in  std_logic );
end reguladorPuertaG;
```

El significado de las señales de salida y entrada al circuito es el siguiente.

- Mientras la señal `motor_up` está a '1' el motor sube la puerta del garaje.
- Mientras la señal `motor_down` está a '1' el motor baja la puerta del garaje.
- El motor está parado mientras la señal `motor_stop` está a '1'.
- La señal `abierta` está a '1' mientras la puerta del garaje está completamente abierta.
- La señal `cerrada` está a '1' mientras la puerta del garaje está completamente cerrada.
- La señal `objeto` está a '1' mientras la fotocélula detecta un objeto.
- La señal `mando` está a '1' mientras está pulsado el botón del mando.
- El circuito tiene una señal de reset (`reset`) asíncrona activa a nivel alto.

Este circuito puede ser descrito con el diagrama de estados mostrado en la siguiente figura. Este diagrama tiene 4 estados: SCerrado, SAbriendo, SAbierto y SCerrando. Las transiciones entre estados se producen cuando la señal indicada en el arco de transición toma el valor '1'. En cada estado se indican las únicas señales de salida que tienen el valor '1' en dicho estado.



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 ser de 10 Hz e inicialmente valer '0'. El programa de test debe realizar consecutivamente las acciones siguientes:

1. *Reset*. Resetear el circuito. Todas las señales de entrada al circuito, excepto la señal de reset, han de tener el valor '0'.
2. *En el instante 30 s la fotocélula detecta un objeto* . Darle a la señal `objeto` el valor '1' durante 10 segundos (hasta el instante 40 s). Comprobar que el circuito está generando la señal de salida adecuada para subir la puerta del garaje.
3. *En el instante 90 s la puerta está completamente abierta*. Darle a la señal `abierta` el valor '1'. Comprobar que el circuito está generando la señal de salida adecuada para parar la puerta del garaje.
4. *En el instante 120 s se aprieta el botón del mando*. Darle a la señal `mando` el valor '1' durante 5 segundos (hasta el instante 125 s). Comprobar que el circuito está generando la señal de salida adecuada para bajar la puerta del garaje.
5. *En el instante 180 s la puerta está completamente cerrada*. Darle a la señal `cerrada` el valor '1'. Comprobar que el circuito está generando la señal de salida adecuada para parar la puerta del garaje.

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.

INGENIERÍA DE COMPUTADORES III

Apellidos: _____ Nombre: _____

DNI: _____ Centro Asociado en el que está MATRICULADO: _____

INSTRUCCIONES: Complete sus datos personales en la cabecera de esta hoja y en todas las demás hojas del examen.
Entregue esta hoja de enunciado junto con el examen.
Dispone de **2 horas** para realizar el examen.
MATERIAL PERMITIDO: Ninguno.

Pregunta 1 (2 puntos)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `x1`, `x2`, `x3`, `temp4`, `temp5`, `temp6` y `x4` entre los instantes 0 y 180 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono1 is
end entity crono1;

architecture crono1 of crono1 is
   signal x1, x2 : std_logic := '0';
   signal x3, x4 : std_logic;
   signal temp4, temp5, temp6: std_logic;
begin
 process (x1)
    variable temp1, temp2, temp3: std_logic;
  begin
     temp1 := x2;
     temp2 := temp1;
     temp3 := temp2;
     x3 <= temp3;
     temp4 <= x2;
     temp5 <= temp4;
     temp6 <= temp5;
     x4 <= temp6;
  end process;
  x1 <= not x1 after 20 ns;
  x2 <= '0',  '1' after 30 ns, '0' after 90 ns;
end architecture crono1;
```

Pregunta 2 (3 puntos) Diseñe un circuito digital combinacional que tenga como entrada una señal de 4 bits y como salida una señal de un bit que se ponga a '1' si la señal de entrada es "0010", "0011", "1010", "1011" ó "1111" y se ponga a '0' en cualquier otro caso. La **entity** del circuito digital se muestra a continuación.

```
entity circ2 is
port (   y: out std_logic;
          x: in std_logic_vector(3 downto 0)    );
end entity circ2;
```

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 3 entradas.

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

2.e) (2 puntos) Escriba en VHDL la **architecture** que describe la estructura del circuito combinacional siguiendo el diagrama dibujado en el apartado anterior y empleando las puertas lógicas cuyo diseño ha realizado al contestar los tres primeros apartados.

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.

Pregunta 4 (3 puntos)

Diseñe un circuito secuencial síncrono capaz de detectar cuando le llega la secuencia "110" por su entrada. La **entity** del circuito se muestra a continuación.

```
entity detector is
   port( Y     : out std_logic;
         X     : in std_logic;
         reset : in std_logic;
         clk   : in std_logic);
end entity detector;
```

El circuito tiene una señal de reloj (`clk`), una entrada serie de un bit (`X`), una señal de reset asíncrona activa en '0' (`reset`) y una señal de salida de un bit (`Y`). La señal `Y` se pone a '1' si los últimos tres bits que han llegado por la entrada (`X`) se corresponden con la secuencia "110". La señal `reset` pone el circuito en su estado inicial. Todos los cambios tienen lugar en el flanco de subida de la señal de reloj. Escriba en VHDL la **architecture** que describe el comportamiento del circuito en términos de una máquina de Moore. Dibuje el diagrama de estados correspondiente al circuito que ha diseñado.

INGENIERÍA DE COMPUTADORES III

Apellidos: _____ Nombre: _____

DNI: _____ Centro Asociado en el que está MATRICULADO: _____

INSTRUCCIONES: Complete sus datos personales en la cabecera de esta hoja y en todas las demás hojas del examen.
Entregue esta hoja de enunciado junto con el examen.
Dispone de **2 horas** para realizar el examen.
MATERIAL PERMITIDO: Ninguno.

Pregunta 1 (2 puntos)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `x1`, `x2`, `x3`, `x4`, `x5` y `x6` entre los instantes 0 y 100 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono2 is
end entity crono2;

architecture crono2 of crono2 is
   signal x1 : std_logic := '0';
   signal x2, x3, x4, x5, x6 : std_logic;
begin
 process
 begin
   x3 <= x1;
   wait for 10 ns;
   x4 <= x3;
   x5 <= x4;
   wait for 10 ns;
 end process;
 x1 <= '0', '1' after 20 ns, '0' after 40 ns,
         '1' after 50 ns;
 x6 <= x1 after 25 ns;
 x2 <= x1 after 15 ns;
end architecture crono2;
```

Pregunta 2 (3 puntos)

2.a) (0.75 puntos) Escriba en VHDL la **architecture** de un multiplexor 4:1 empleando una sentencia concurrente condicional (**when - else**). La **entity** del multiplexor 4:1 se muestra a continuación.

```
entity mux4_1 is
port (   y: out std_logic;
          s: in std_logic_vector(1 downto 0);
          x: in std_logic_vector(3 downto 0)    );
end entity mux4_1;
```

2.b) (0.25 puntos) Dibuje el diagrama circuital de un circuito desplazador de barril de 4 bits empleando para ello únicamente multiplexores 4:1 como el descrito en el apartado 2.a. Este circuito desplazador tiene una señal de entrada `s` de 2 bits para seleccionar la operación, una señal de entrada `x` de 4 bits y una señal de salida `y` de 4 bits. La tabla de operaciones del circuito se muestra a continuación.

s	Operación	y
”00”	Ninguna	x(3)x(2)x(1)x(0)
”01”	Rota 1 bit a la izquierda	x(2)x(1)x(0)x(3)
”10”	Rota 2 bits a la izquierda	x(1)x(0)x(3)x(2)
”11”	Rota 3 bits a la izquierda	x(0)x(3)x(2)x(1)

2.c) (2 puntos) Escriba en VHDL la **architecture** que describe la estructura del circuito desplazador siguiendo el diagrama dibujado en el apartado anterior y empleando el multiplexor cuyo diseño ha realizado al contestar el primer apartado. La **entity** del circuito se muestra a continuación.

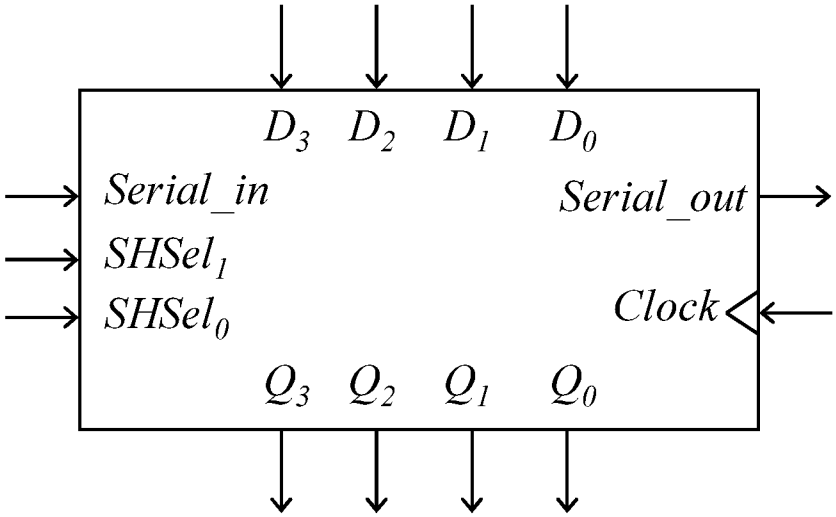
```
entity despBarril is
port (   y: out std_logic_vector(3 downto 0);
          x: in std_logic_vector(3 downto 0);
          s: in std_logic_vector(1 downto 0)    );
end entity despBarril;
```

Pregunta 3 (2 puntos)

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

Pregunta 4 (3 puntos)

Diseñe usando VHDL un registro de desplazamiento conversor serie-a-paralelo y paralelo-a-serie de 4 bits como el mostrado en la figura siguiente. El circuito tiene las siguientes entradas: señal de reloj, dos entradas de selección, una entrada serie y cuatro entradas para carga en paralelo. Las salidas del circuito son la salida serie y cuatro salidas en paralelo.



Para llevar a cabo la conversión serie-a-paralelo, en primer lugar deben cargarse los 4 bits en el registro a través de la entrada serie y a continuación realizarse la lectura en paralelo.

Por otro lado, para la conversión paralelo-a-serie, se realiza primero una escritura en paralelo y a continuación se realiza la lectura del contenido del registro bit a bit a través de la salida serie. Para ello, es necesario realizar tres operaciones de desplazamiento. En la tabla siguiente se muestran las operaciones del registro.

$SHSel_1$	$SHSel_0$	**Operación**
0	0	Mantiene valor
0	1	Carga en paralelo
1	0	Desplaza 1 bit a la derecha el contenido del registro y asigna a Q_3 el valor en *Serial_in*

El diseño del registro en VHDL debe realizarse describiendo el comportamiento del circuito, empleando para ello un único bloque **process**.

INGENIERÍA DE COMPUTADORES III

INSTRUCCIONES
Por favor, entregue todas las hojas de enunciado junto con el examen.
Dispone de 2 horas para realizar el examen.
MATERIAL PERMITIDO: Ninguno.

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 60 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity crono1 is
end entity crono1;

architecture crono1 of crono1 is
   signal x1, x2, x3, x4, x5 : std_logic;
begin
 process is
    variable temp  : unsigned (2 downto 0);
  begin
    for i in 0 to 3 loop
      temp := TO_UNSIGNED(i,3);
      x1 <= std_logic(temp(2));
      x2 <= std_logic(temp(1));
      x3 <= std_logic(temp(0));
      wait for 10 ns;
    end loop;
    wait;
  end process;
  x4 <= x3 after 5 ns;
  x5 <= x3 after 15 ns;
end architecture crono1;
```

Pregunta 2 (3 puntos)

Escriba en VHDL, de las cuatro formas que se detallan a continuación, la **architecture** que describe el comportamiento de un circuito combinacional multiplexor 8 a 1. La **entity** del circuito es:

```
entity mux is
   port ( y  : out std_logic;
          x  : in  std_logic_vector(7 downto 0);
          s  : in  std_logic_vector(2 downto 0) );
end entity mux;
```

2.a) (0.75 puntos) Empleando una sentencia concurrente condicional (**when - else**).

2.b) (0.75 puntos) Empleando una asignación concurrente de selección (**with - select**).

2.c) (0.75 puntos) Empleando un bloque **process** con una sentencia **if**.

2.d) (0.75 puntos) Empleando un bloque **process** con una sentencia **case**.

Pregunta 3 (3 puntos) Escriba el código VHDL de la **architecture** que describe el comportamiento de un circuito contador binario de 3 bits cuya salida sigue cíclicamente la secuencia "000", "011", "110", 101" y "111", con señal de reset asíncrona activa a nivel alto. La **entity** del circuito se muestra a continuación.

```
entity contador is
  port (  count      : out std_logic_vector (2 downto 0);
          clk, reset : in  std_logic );
end contador;
```

Las entradas al circuito son la señal de reloj (`clk`) y la señal de reset asíncrono activo a nivel alto (`reset`). La salida del circuito contador es la señal de 3 bits `count`.

El funcionamiento del circuito debe ser el siguiente:

- *Reset asíncrono activo a nivel alto*. Cuando `reset` pasa a valer '1', el contador se pone al valor "000".
- *Cuenta síncrona*. Mientras `reset` vale '0', el contador pasa en cada flanco de subida de la señal de reloj por la secuencia "000", "011", "110", "101" y "111".

El diseño debe realizarse describiendo el comportamiento del circuito en términos de una máquina de Moore.

Pregunta 4 (2 puntos)

Programe en VHDL un banco de pruebas para el contador que ha diseñado al contestar a la Pregunta 3. 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* . Incrementar el valor del contador hasta "111".
3. *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.

INGENIERÍA DE COMPUTADORES III

INSTRUCCIONES
Por favor, entregue todas las hojas de enunciado junto con el examen.
Dispone de 2 horas para realizar el examen.
MATERIAL PERMITIDO: Ninguno.

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 60 ns.

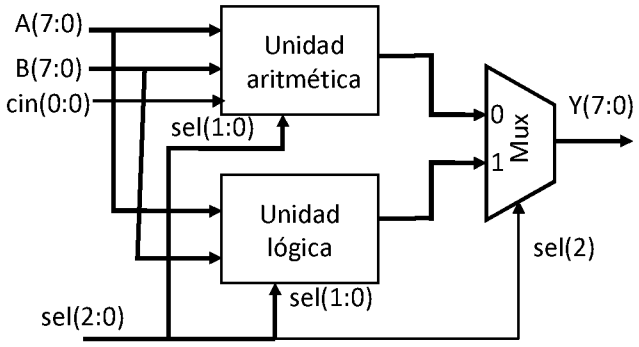
```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.numeric_std.all;
entity crono2 is
end entity crono2;

architecture crono2 of crono2 is
   signal x1  : unsigned (2 downto 0);
   signal x2, x3, x4, x5 : std_logic;
begin
 process is
  begin
    for i in 0 to 3 loop
      x1 <= to_unsigned(i,3);
      x2 <= std_logic(x1(2));
      x3 <= std_logic(x1(1));
      x4 <= std_logic(x1(0));
      wait for 10 ns;
    end loop;
    wait;
  end process;
  x5 <= x3 after 15 ns;
end architecture crono2;
```

Pregunta 2 (3 puntos)

A continuación, se muestra el circuito, la tabla de operaciones y la **entity** de una ALU. La ALU realiza operaciones sobre dos operandos de 8 bits, denominados `A` y `B`. La salida de la ALU se selecciona mediante el bit más significativo de la señal `sel`, mientras que la operación que realiza se especifica por los otros dos bits de esta señal. Escriba en VHDL la **architecture** que describe el comportamiento de la ALU, empleando para ello únicamente tres sentencias de asignación concurrente de selección. Asimismo, en el diseño únicamente pueden emplearse los dos siguientes paquetes de la librería IEEE:

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



Unidad	sel	Operación
Aritmética	000	A+1
	001	A+B
	010	A+B+cin
	011	B-1
Lógica	100	not A
	101	A and B
	110	A or B
	111	A xor B

```
entity ALU is
   port( Y    : out std_logic_vector ( 7 downto 0);
         A, B : in std_logic_vector (7 downto 0);
         sel  : in std_logic_vector ( 2 downto 0);
         cin  : in std_logic_vector(0 downto 0));
end entity ALU;
```

Pregunta 3 (2.5 puntos)

Diseñe un circuito secuencial síncrono capaz de detectar cuando le llegan al menos tres ceros consecutivos por su entrada. La **entity** del circuito se muestra a continuación. El circuito tiene una señal de reloj (`clk`), un entrada serie de un bit (`X`), una señal de reset asíncrona activa en '0' (`reset`), una señal que indica el estado en que se encuentra el circuito (`state`) y una señal de salida de un bit (`Y`). La señal `Y` se pone a '1' si por la entrada `X` se han recibido tres o más ceros consecutivos. La señal `reset` pone el circuito en su estado inicial. Todos los cambios tienen lugar en el flanco de subida de la señal de reloj. Escriba en VHDL la **architecture** que describe el comportamiento del circuito en términos de una máquina de

Moore. Dibuje el diagrama de estados correspondiente al circuito que ha diseñado.

```
entity detector is
   port( Y     : out std_logic;
         state : out std_logic_vector(1 downto 0);
         X     : in std_logic;
         reset : in std_logic;
         clk   : in std_logic);
end entity detector;
```

Pregunta 4 (2.5 puntos) Programe el banco de pruebas del circuito secuencial que ha diseñado. 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.

INGENIERÍA DE COMPUTADORES III

Apellidos: _____ Nombre: _____

DNI: _____ Centro Asociado en el que está MATRICULADO: _____

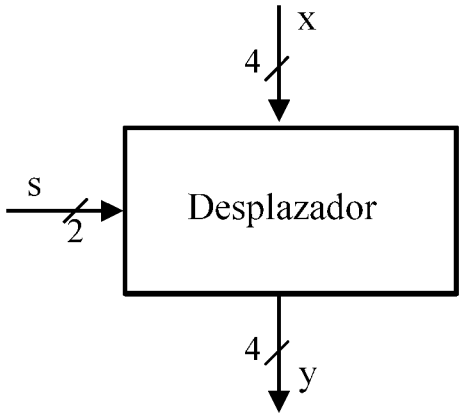
INSTRUCCIONES: Complete sus datos personales en la cabecera de esta hoja y en todas las demás hojas del examen.
Entregue esta hoja de enunciado junto con el examen.
Dispone de **2 horas** para realizar el examen.
MATERIAL PERMITIDO: Ninguno.

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;
```

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.

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.

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;
```

INGENIERÍA DE COMPUTADORES III

Apellidos: _____ Nombre: _____

DNI: _____ Centro Asociado en el que está MATRICULADO: _____

INSTRUCCIONES: Complete sus datos personales en la cabecera de esta hoja y en todas las demás hojas del examen.
Entregue esta hoja de enunciado junto con el examen.
Dispone de **2 horas** para realizar el examen.
MATERIAL PERMITIDO: Ninguno.

Pregunta 1 (2 puntos)

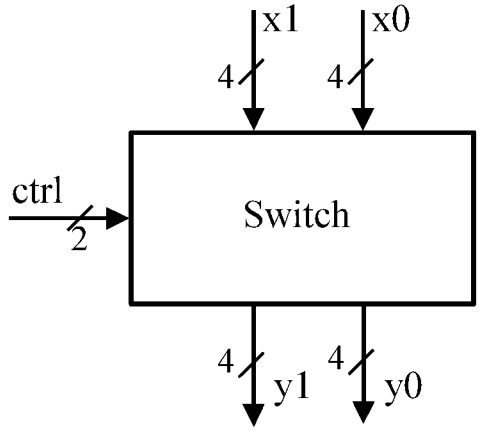
Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `clk`, `x`, `y1` e `y2` entre los instantes 0 y 1000 ns.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity crono2 is
end entity crono2;
architecture crono2 of crono2 is
  signal x, y1, y2 : std_logic;
  signal clk : std_logic:='0';
begin
   process (clk)
    variable a, b, c: std_logic;
   begin
      if ( rising_edge(clk) ) then
         a := x;
         b := a;
         c := b;
         y1 <= c;
         y2 <= y1;
      end if;
   end process;
  clk <= not clk after 100 ns;
  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;
end architecture crono2;
```

Pregunta 2 (3 puntos)

Escriba en VHDL, de las cuatro formas que se detallan a continuación, la **architecture** que describe el comportamiento de un circuito combinacional cuya **entity**, símbolo lógico y tabla de operaciones se muestran a continuación.

```
entity switch is
 port (  y1, y0      : out std_logic_vector(3 downto 0);
         ctrl        : in  std_logic_vector(1 downto 0);
         x1, x0      : in std_logic_vector(3 downto 0) );
end entity switch;
```



ctrl(1)	ctrl(0)	y1	y0
0	0	x1	x0
0	1	x0	x1
1	0	x0	x0
1	1	x1	x1

2.a) (0.75 puntos) Empleando sentencias concurrentes condicionales (**when** - **else**).

2.b) (0.75 puntos) Empleando asignaciones concurrentes de selección (**with** - **select**).

2.c) (0.75 puntos) Empleando un bloque **process** con sentencias **if**.

2.d) (0.75 puntos) Empleando un bloque **process** con sentencias **case**.

Pregunta 3 (2 puntos)

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

Pregunta 4 (3 puntos)

Diseñe un circuito secuencial síncrono capaz de detectar cuando le llega la secuencia "011" por su entrada. La **entity** del circuito se muestra a continuación. El circuito tiene una señal de reloj (`clk`), una entrada serie de un bit (`X`), una señal de reset asíncrona activa en '0' (`reset`) y una señal de salida de dos bits (`Y`). La señal `Y` puede valer "01", "10", "11" ó "00" dependiendo de si los últimos bits recibidos por la entrada serie de un bit (`X`) finalizan en "0", finalizan en "01", finalizan en "011" o no finalizan en ninguna de las anteriores terminaciones, respectivamente. La señal `reset` pone el circuito en su estado inicial. Todos los cambios tienen lugar en el flanco de subida de la señal de reloj. Escriba en VHDL la **architecture** que describe el comportamiento del circuito en términos de una máquina de Moore. Dibuje el diagrama de estados correspondiente al circuito que ha diseñado.

```
entity detector is
 port( Y     : out std_logic_vector(1 downto 0);
       X     : in std_logic;
       reset : in std_logic;
       clk   : in std_logic);
end entity detector;
```

INGENIERÍA DE COMPUTADORES III

INSTRUCCIONES
Por favor, entregue todas las hojas de enunciado junto con el examen.
Dispone de 2 horas para realizar el examen.
MATERIAL PERMITIDO: Ninguno.

Pregunta 1 (2 puntos)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `x1`, `x2`, `x3`, `s`, `y` entre los instantes 0 y 50 ns.

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

entity crono is
end entity crono;

architecture crono of crono is
  signal x1, x3, s, y : std_logic;
  signal x2  : std_logic := '0';
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;
  x3 <= '0',
        '1' after 10 ns,
        '0' after 30 ns;
  Proc1: process (x1, x2, x3)
  begin
    s <= x1;
    y <= s or x3;
    s <= X2;
  end process;
end architecture crono;
```

Pregunta 2 (3 puntos)

Se pretende diseñar un circuito comparador cuya salida (`F`) vale '1' si el valor del número de cuatro bits de entrada (`x`) es menor que el número decimal 9. Los cuatro bits de entrada se interpretan como un número binario sin signo. La **entity** del circuito se muestra a continuación.

```
entity comparaXmenor9 is port
        ( F : out std_logic;
          x : in  std_logic_vector(3 downto 0) );
end entity comparaXmenor9;
```

Escriba en VHDL la **architecture** del circuito comparador de las dos formas siguientes:

2.a) (1 punto) Empleando únicamente asignaciones concurrentes y operadores lógicos.

2.b) (2 puntos) Describiendo su estructura. Dibuje el diagrama del circuito al nivel de puertas lógicas. Escriba en VHDL la **entity** y **architecture** de todos los tipos de puertas lógicas que contiene el diseño del comparador. Finalmente, escriba en VHDL la **architecture** del comparador usando las puertas lógicas que ha definido previamente.

Pregunta 3 (2 puntos)

Programe en VHDL el banco de pruebas del comparador que ha diseñado al resolver la Pregunta 2. El banco de pruebas debe comprobar el funcionamiento del circuito de forma exhaustiva. El banco de pruebas debe generar un conjunto de vectores de test, comprobar 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.

Pregunta 4 (3 puntos)

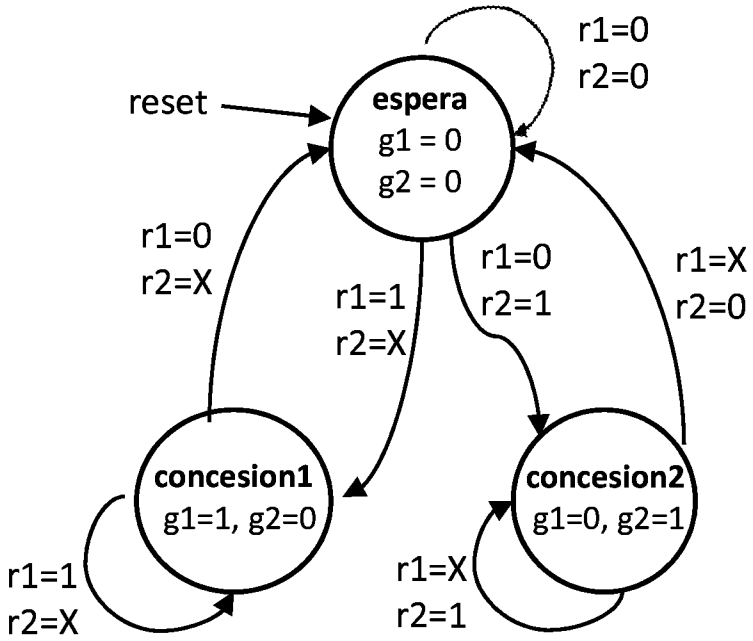
El circuito denominado *árbitro* permite controlar el acceso de varios dispositivos, denominados *clientes*, a un determinado recurso compartido. Este recurso compartido puede ser una memoria, un bus o cualquier componente de uso común. El objetivo del circuito árbitro es permitir en un momento dado el acceso de un único cliente al recurso. Para controlar el acceso de los clientes al recurso por medio del árbitro se usa un sistema de peticiones (*request*) y concesiones (*grant*).

Se quiere diseñar un árbitro que tenga dos clientes, que llamamos cliente1 y cliente2. La comunicación entre el cliente1 y el árbitro se realiza mediante las señales de petición (`r1`) y de concesión (`g1`). Análogamente, la comunicación entre el cliente2 y el árbitro se realiza mediante las señales `r2` y `g2`.

El cliente activa su señal de petición cuando quiere acceder al recurso compartido y el árbitro da acceso al cliente activando su señal de concesión correspondiente. Tras completar la tarea, el cliente libera el recurso y desactiva su señal de petición. Cuando ambos clientes realizan una petición simultánea, cliente1 tiene siempre la prioridad.

El comportamiento del árbitro puede describirse mediante una máquina de estados finitos con tres estados: espera, concesion1 y concesion2. El estado espera indica que el recurso está disponible y el árbitro está esperando peticiones. Los estados concesion1 y concesion2 indican que el recurso se ha dado al cliente1 y al cliente2, respectivamente. El árbitro ha de tener también una señal de reset asíncrona activa a nivel alto.

El diagrama de estados de esta máquina se muestra a continuación. Se ha empleado el símbolo X para indicar que el valor de la señal puede ser 0 ó 1.



Escriba en VHDL la **architecture** que describe el comportamiento de este árbitro de dos clientes como una máquina de estados de Moore síncrona, donde las transiciones tienen lugar en el flanco de subida de la señal de reloj.

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

```
entity arbitro is port
        ( g1, g2 : out std_logic;
          r1, r2, clk, reset : in  std_logic );
end entity arbitro;
```

INGENIERÍA DE COMPUTADORES III

INSTRUCCIONES
Por favor, entregue todas las hojas de enunciado junto con el examen.
Dispone de 2 horas para realizar el examen.
MATERIAL PERMITIDO: Ninguno.

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.

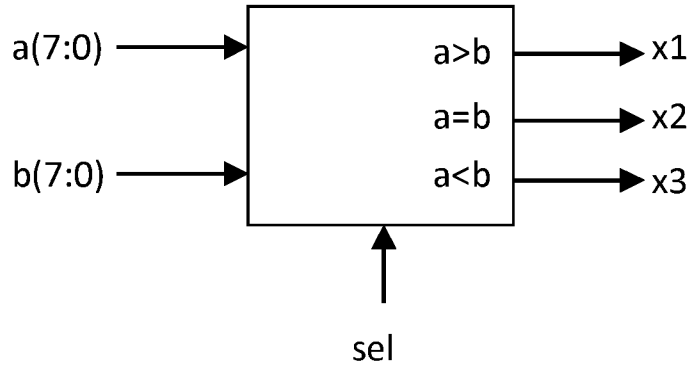
```
library IEEE;
use IEEE.std_logic_1164.all;

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;
```

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;
```

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:

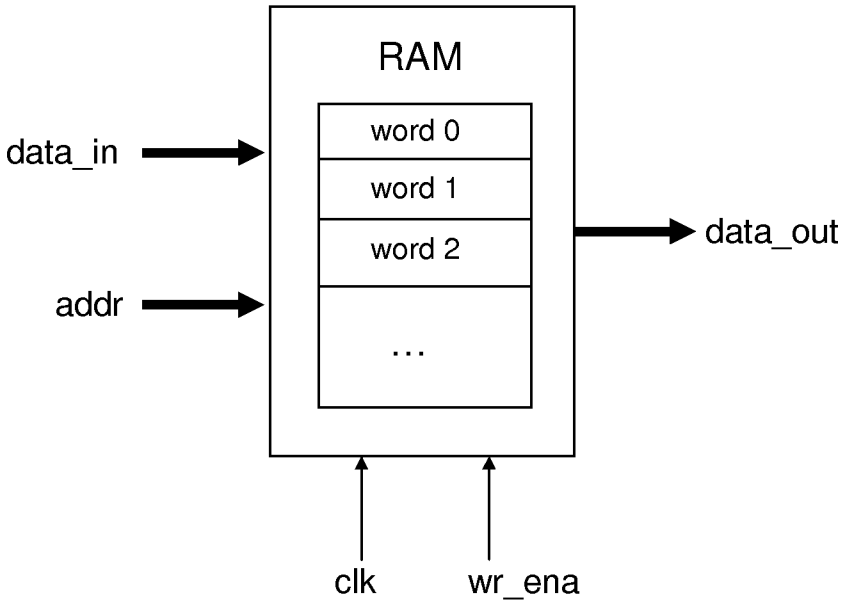
- `a="11111111",b="00000000",sel='1'`
- `a="11111111",b="00000000",sel='0'`
- `a="00001110",b="00001111",sel='1'`

- 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.

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**.

INGENIERÍA DE COMPUTADORES III

INSTRUCCIONES
Por favor, entregue todas las hojas 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+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;
```

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
```

Pregunta 3 (2 puntos)

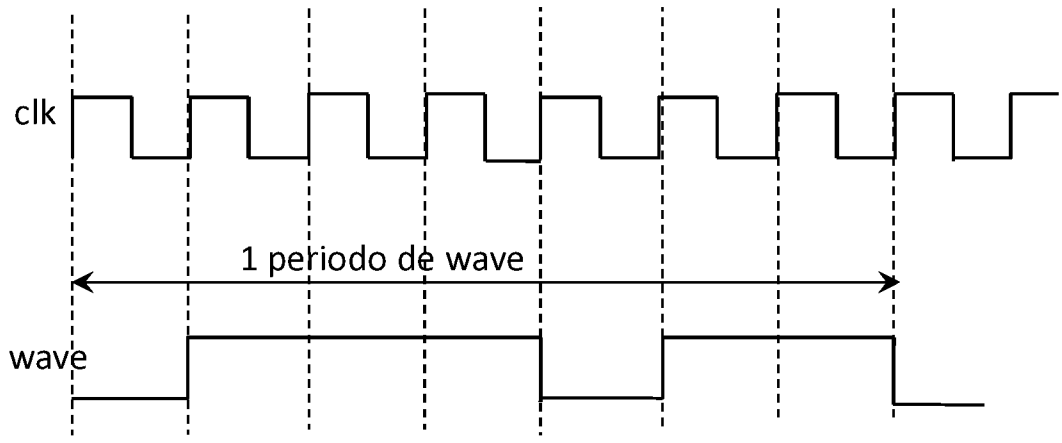
Programe en VHDL un banco de pruebas para el contador binario que ha diseñado al contestar a la Pregunta 2. La señal de reloj (`clk`) debe tener un periodo de 10 ns e inicialmente valer '0'. El programa de test debe realizar las acciones siguientes:

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.

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;
```

INGENIERÍA DE COMPUTADORES III

INSTRUCCIONES
Por favor, entregue todas las hojas de enunciado junto con el examen.
Dispone de 2 horas para realizar el examen.
MATERIAL PERMITIDO: Ninguno.

Pregunta 1 (2 puntos)

Tomando como base el siguiente código VHDL, dibuje el cronograma de evolución de las señales `x1`, `x2`, `x3`, `x4`, `x5`, `x6` entre los instantes 0 y 50 ns.

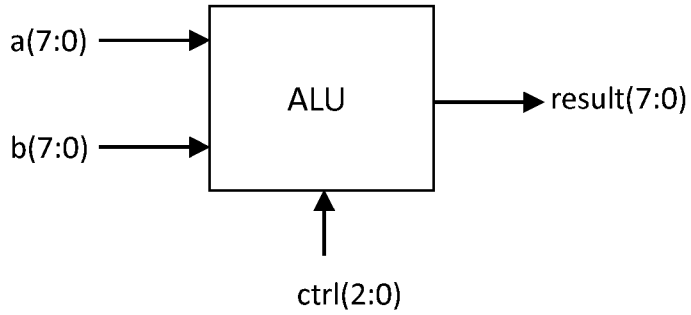
```
library IEEE;
use IEEE.std_logic_1164.all;

entity crono is
end entity crono;

architecture crono of crono is
  signal x1, x3, x4, x5, x6: std_logic;
  signal x2  : std_logic := '0';
begin
  x1 <= '1',
        '0' after 10 ns,
        '1' after 20 ns,
        '0' after 30 ns,
        '1' after 40 ns;
  x2 <= '0',
        '1' after 35 ns;
  Proc1: process (x1, x2, x3)
    variable valor  : std_logic;
  begin
    x4 <= x1 or x2;
    valor := x1 or x2;
    x5 <= x4;
    x6 <= valor;
    x3 <= x6;
  end process;
end architecture crono;
```

Pregunta 2 (3 puntos)

En la siguiente figura se muestran las señales de entrada y salida de una Unidad Aritmético Lógica (ALU).



La **entity** de dicho circuito es la siguiente:

```
entity alu is
   port( result : out std_logic_vector (7 downto 0);
         ctrl   : in  std_logic_vector (2 downto 0);
         a , b  : in  std_logic_vector (7 downto 0) );
end entity alu;
```

Las señales `a` y `b`, de 8 bits, son los operandos. La señal `ctrl`, de 3 bits, determina la operación realizada por el circuito, según se indica en la tabla siguiente. Se ha empleado el símbolo - para indicar que el valor puede ser 0 ó 1. El circuito realiza las operaciones aritméticas suma y resta interpretando los operandos como números binarios con signo.

ctrl	Operación
"0--"	a + 1
"100"	a + b
"101"	a - b
"110"	a and b
"111"	a or b

Diseñe en VHDL la **architecture** que describe el comportamiento del circuito, usando únicamente asignaciones concurrentes simples y asignaciones concurrentes condicionales (**when - else**), y empleando únicamente los dos siguientes paquetes de la librería IEEE:

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

Puede emplear en el diseño tantas señales auxiliares como considere necesario.

Pregunta 3 (2.5 puntos)

A continuación se muestra la **entity** de un circuito contador binario ascendente y descendente de 3 bits.

```
entity contador is
  port ( cuenta : out std_logic_vector(2 downto 0);
         clk    : in  std_logic;
         reset  : in  std_logic;
         up     : in  std_logic;
         down   : in  std_logic  );
end contador;
```

El circuito tiene una señal de 3 bits (`cuenta`) de salida y cuatro señales de entrada: la señal de reloj (`clk`), la señal de reset asíncrono activo a nivel alto (`reset`), la señal de cuenta ascendente (`up`) y la señal de cuenta descendente (`down`).

El funcionamiento del circuito debe ser el siguiente:

- *Reset asíncrono activo a nivel alto*. Cuando la señal `reset` pasa a valer 1, el contador se pone al valor "000".
- *Cuenta ascendente*. La señal `up` tiene prioridad sobre la señal `down`. Mientras `reset` vale '0' y `up` vale '1', sea cual sea el valor de la señal `down`, en cada flanco de subida de la señal de reloj el contador se incrementa en uno, hasta llegar al valor "111". La cuenta ascendente se detiene en "111", dado que el contador no tiene rebosamiento.
- *Cuenta descendente*. Mientras `reset` vale '0', `down` vale '1' y `up` vale '0', en cada flanco de subida de la señal de reloj el contador se decrementa en uno hasta llegar al valor "000". La cuenta descendente se detiene en "000".

Diseñe en VHDL la **architecture** que describe el comportamiento del circuito, empleando para ello un único bloque **process** y utilizando únicamente los dos siguientes paquetes de la librería IEEE:

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

Pregunta 4 (2.5 puntos)

Programe en VHDL un banco de pruebas para el contador binario que ha diseñado en la Pregunta 3.

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. Resetear el contador. La señal (`reset`) debe valer '1' únicamente entre los instantes 2 ns y 7 ns.
2. Incrementar el valor del contador hasta "111".
3. Mantener el valor "111"durante un ciclo de reloj.
4. Decrementar la cuenta hasta alcanzar el valor "000".

Si el valor de la señal de salida de la UUT no coincide con lo esperado, el programa de test debe mostrar el correspondiente mensaje.

Además, dibuje el cronograma de evolución que han de seguir las señales de entrada y salida de la UUT.