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 `s1`, `s2`, `s3` y `s4` entre los instantes 0 y 90 ns. En el cronograma se debe indicar para cada una de estas señales el instante de tiempo en que la señal cambia de valor así como su nuevo valor.

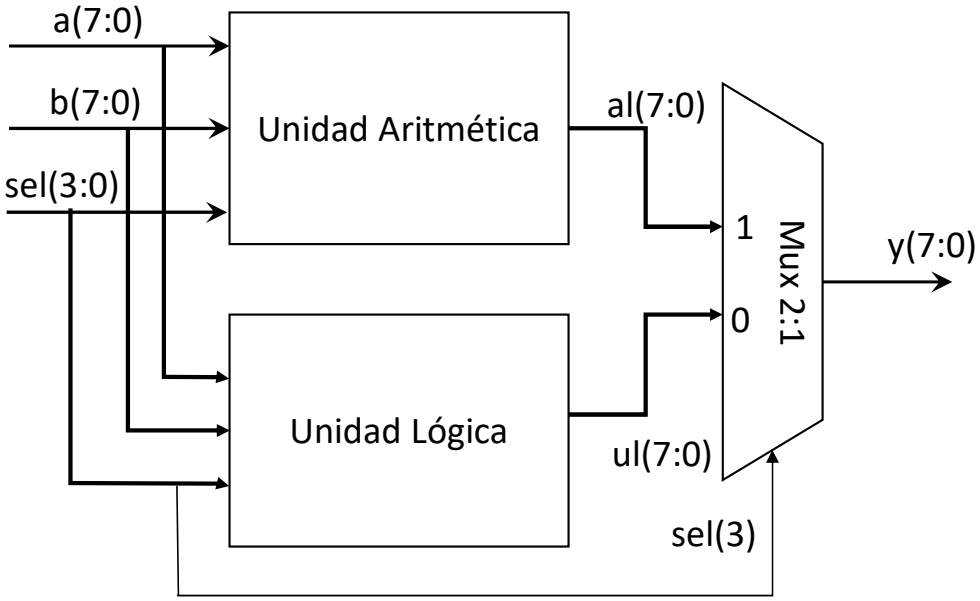
```
library IEEE;
use IEEE.std_logic_1164.all;

entity crono2 is
end entity crono2;

architecture crono2 of crono2 is
   signal s1: std_logic := '0';
   signal s2, s3, s4 : std_logic;
begin
   s1 <= '0',
         '1' after 80 ns;
   s1 <= '0',
         '1' after 40 ns;
   s2 <= '1',
         '0' after 10 ns,
         '1' after 20 ns,
         '0' after 25 ns;
   s3 <= s2 after 10 ns;
   s4 <= transport s2 after 10 ns;
end architecture crono2;
```

Pregunta 2 (3 puntos)

Programe en VHDL la ALU cuyo circuito se muestra en la figura siguiente y que realiza las operaciones descritas en la tabla mostrada a continuación.



Unidad	`sel`	**Operación**	**Descripción**
Unidad Lógica	0 0 0 0	not a	Complemento de a
	0 0 0 1	not b	Complemento de b
	0 0 1 0	a or b	OR lógico
	0 0 1 1	a and b	AND lógico
	0 1 0 0	a nor b	NOR lógico
	0 1 0 1	a nand b	NAND lógico
	0 1 1 0	a xnor b	XNOR lógico
	0 1 1 1	a xor b	XOR lógico
Unidad Aritmética	1 0 0 0	a	Deja pasar a
	1 0 0 1	$-a$	Opuesto de a
	1 0 1 0	$a+1$	Incrementa a
	1 0 1 1	b	Deja pasar b
	1 1 0 0	$-b$	Opuesto de b
	1 1 0 1	$b+1$	Incrementa b
	1 1 1 0	$a-b$	Resta de a y b
	1 1 1 1	$a+b$	Suma 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 es especificada por los otros tres bits de esta señal. Al realizar las operaciones aritméticas, las señales `a` y `b` se interpretan como números binarios con signo (representados en complemento a 2).

Se debe proporcionar un código VHDL del diseño de la ALU que describa la unidad aritmética empleando un bloque **process**, la unidad lógica empleando una sentencia **when–else** y el multiplexor empleando una sentencia **with–select**.

Se debe emplear el siguiente paquete:

```
package ALU_CONSTANTS is
   constant WIDTH     :  integer := 8;
                         -- Núm. bits de los operandos
   constant SEL_BITS  :  integer := 4;
                         -- Núm. bits selección de operación
end package ALU_CONSTANTS;
```

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

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

La ALU ha de tener la siguiente **entity**:

```
entity ALU is
 port(    y : out std_logic_vector (WIDTH-1 downto 0);
       a, b : in std_logic_vector (WIDTH-1 downto 0);
        sel : in std_logic_vector (SEL_BITS-1 downto 0) );
end entity ALU;
```

Pregunta 3 (2 puntos)

Programe el banco de pruebas del circuito combinacional que ha diseñado en la Pregunta 2. El banco de pruebas va a permitir visualizar el valor de las señales del circuito para todos los valores posibles de la señal de entrada `sel` y los siguientes valores de los operandos `a` y `b`.

- `a = "00000000", b = "00000000"`
- `a = "11111111", b = "00000000"`
- `a = "00000000", b = "11111111"`

El banco de pruebas debe permitir comprobar mediante inspección visual que los valores obtenidos de la UUT coinciden con los esperados. Al final del test, debe mostrarse un mensaje indicando que el test ha finalizado.

Pregunta 4 (3 puntos)

Programe en VHDL un circuito contador progresivo decimal de 2 dígitos. El contador realiza la cuenta de 00 a 99 de forma cíclica, pasando de un número al consecutivo en el flanco de subida de la señal de reloj siempre que la señal de cuenta esté habilitada y la señal de reset esté a nivel bajo. Como el contador es cíclico, se considera que el número que sigue al 99 es el 00. Cada dígito decimal se representa por un número binario de 4 bits. Por ejemplo, el número decimal 2 se representa por el número binario `"0010"`.

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

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

La **entity** del circuito contador se muestra a continuación:

```
entity counter is port
 ( digit2, digit1 : out std_logic_vector(3 downto 0);
   en             : in  std_logic;
   clk, reset     : in  std_logic );
end entity counter;
```

Las señales de entrada del circuito son las siguientes:

- La señal de reloj `clk`.
- Una señal `reset` asíncrona activa a nivel alto. Cuando esta señal está activa, la cuenta se pone a cero.
- La señal `en` permite habilitar y deshabilitar la cuenta. Si el valor de la señal `en` es '1', la cuenta está habilitada. En caso contrario, se para la cuenta, manteniéndose el valor de las señales de salida `digit1` y `digit2`. Mientras la cuenta está habilitada, el circuito pasa en cada flanco de subida de la señal de reloj de un número al siguiente. Por ejemplo, se pasa del valor decimal 00 a 01, 02, 03, etc.

Las señales de salida del circuito son las siguientes:

- La señal `digit1` muestra el valor binario del dígito decimal menos significativo. Por ejemplo, si el valor en decimal del dígito menos significativo es 9, la señal `digit1` toma el valor `"1001"`.
- La señal `digit2` muestra el valor binario del dígito decimal más significativo. Por ejemplo, si el valor en decimal del dígito más significativo es 2, la señal `digit2` toma el valor `"0010"`.