INGENIERÍA DE COMPUTADORES 3

Solución al examen de Junio 2023, Segunda Semana

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.

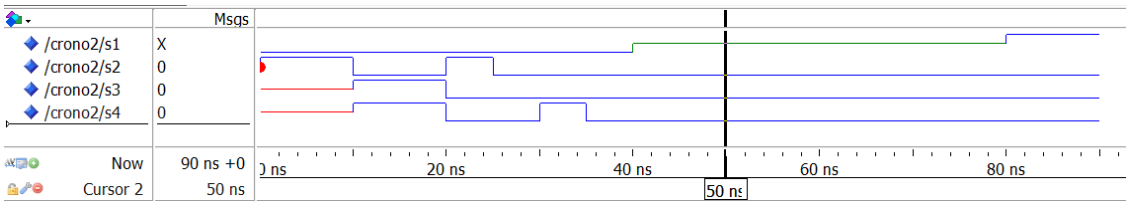
```
library IEEE;
use IEEE.std_logic_1164.all;

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;
```

Solución a la Pregunta 1

En la siguiente figura se muestra el cronograma de evolución de las señales.

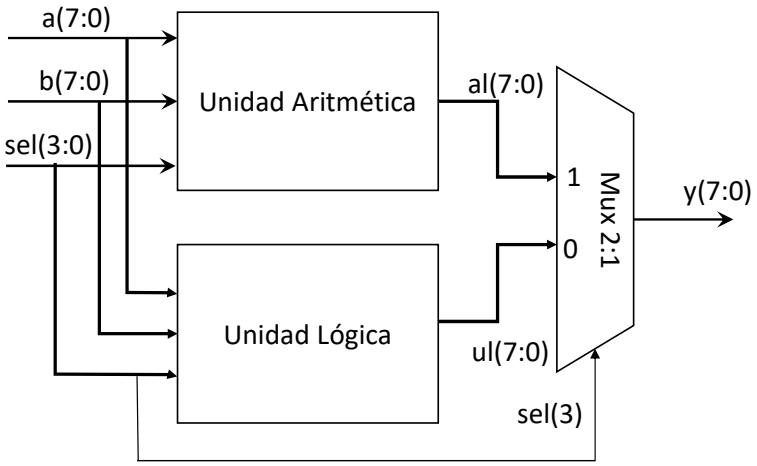


En la siguiente figura se muestra el valor de las señales teniendo en cuenta el retardo delta.

ps delta	/crono2/s1 /crono2/s2 /crono2/s3 /crono2/s4
0 +0	0 U U U
0 +1	0 1 U U
10000 +0	0 0 1 1
20000 +0	0 1 0 0
25000 +0	0 0 0 0
30000 +0	0 0 0 1
35000 +0	0 0 0 0
40000 +0	X 0 0 0
80000 +0	1 0 0 0

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;
```

Solución a la Pregunta 2

El Código VHDL 1.1 es el código de la ALU.

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

architecture ALU_concurrente of ALU is
   signal al, ul: std_logic_vector(WIDTH-1 downto 0);
begin
   --Unidad Aritmética
   arith:process(a,b,sel) is
   begin
      case sel(SEL_BITS-2 downto 0 ) is
         when "000" => al <= a;
         when "001" => al <= std_logic_vector( - signed(a) );
         when "010" => al <= std_logic_vector( signed(a) + 1 );
         when "011" => al <= b;
         when "100" => al <= std_logic_vector( - signed(b) );
         when "101" => al <= std_logic_vector( signed(b) + 1 );
         when "110" =>
               al <= std_logic_vector( signed(a) - signed(b) );
         when others =>
               al <= std_logic_vector( signed(a) + signed(b) );
      end case;
   end process;

   --Unidad Lógica
   ul <= not a when (sel(SEL_BITS-2 downto 0) = "000") else
         not b when (sel(SEL_BITS-2 downto 0) = "001") else
         a or b when (sel(SEL_BITS-2 downto 0) = "010") else
         a and b when (sel(SEL_BITS-2 downto 0) = "011") else
         a nor b when (sel(SEL_BITS-2 downto 0) = "100") else
         a nand b when (sel(SEL_BITS-2 downto 0) = "101") else
         a xnor b when (sel(SEL_BITS-2 downto 0) = "110") else
         a xor b;

   with sel(SEL_BITS-1) select   --Multiplexor
      y <= al when '1',
           ul when others;

end architecture ALU_concurrente;
```

Código VHDL 1.1: Architecture de la 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.

Solución a la Pregunta 3

El código VHDL del banco de pruebas se muestra en Código VHDL 1.2.

```
1  -- Banco de pruebas para la ALU
2  library IEEE;
3  use IEEE.std_logic_1164.all;
4  use IEEE.numeric_std.all;
5
6  entity bp_ALU is
7   constant DELAY        : time    :=  10 ns;
8  end entity bp_ALU;
9
10 architecture bp_ALU of bp_ALU is
11    signal a,b,y : std_logic_vector (WIDTH-1 downto 0);
12    signal sel   : std_logic_vector (SEL_BITS-1 downto 0);
13    type v_test is
14     array (0 to 2) of std_logic_vector(WIDTH-1  downto 0);
15    component ALU is
16       port (  y    : out std_logic_vector(WIDTH-1 downto 0);
17               a, b : in std_logic_vector (WIDTH-1 downto 0);
18               sel  : in std_logic_vector (SEL_BITS-1 downto 0) );
19    end component ALU;
20
21 begin
```

```
22    UUT : component ALU port map (y, a, b, sel);
23    main : process is --Generacion vectores test
24       variable a_test : v_test;
25       variable b_test : v_test;
26       variable ia, ib : integer;
27    begin
28       -- Vectores de test:
29       a_test(0) := (others=>'0');
30       a_test(1) := (others=>'1');
31       a_test(2) := (others=>'0');
32       b_test(0) := (others=>'0');
33       b_test(1) := (others=>'0');
34       b_test(2) := (others=>'1');
35       for i in 0 to 2 loop
36         a <= a_test(i);
37         b <= b_test(i);
38         ia :=TO_INTEGER(SIGNED(a_test(i)));
39         ib :=TO_INTEGER(SIGNED(b_test(i)));
40         for k in 0 to 2**SEL_BITS - 1 loop
41            sel  <= std_logic_vector(TO_SIGNED(k,SEL_BITS));
42            wait for DELAY;
43          end loop;
44       end loop;
45       wait for DELAY;
46
47       report "Finaliza la simulación";
48       wait;  -- Termina la simulación
49    end process main;
50 end architecture bp_ALU;
```

Código VHDL 1.2: Banco de pruebas de la ALU.

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"`.

Solución a la Pregunta 4

El Código VHDL 1.3 es el diseño del circuito contador.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;
architecture counter of counter is
begin
  process(clk,reset)
    variable temp1: integer range 0 to 10;
    variable temp2: integer range 0 to 10;
  begin
    if (reset='1') then
       temp1:= 0;
       temp2:=0;
    elsif (rising_edge(clk)) then
      if (en = '1') then
       temp1 := temp1+1;
       if (temp1=10) then
          temp1:=0;
          temp2:= temp2+1;
          if(temp2=10) then
            temp2:=0;
          end if;
       end if;
      end if;
     end if;
     digit1 <=  std_logic_vector(to_unsigned(temp1, digit1'length));
     digit2 <=  std_logic_vector(to_unsigned(temp2, digit2'length));
  end process;
end counter;
```

Código VHDL 1.3: Circuito contador.