Practica 4

Unidad aritmética/lógica de 4 bits

1. Implementar el circuito para la ALU diseñado en clase.

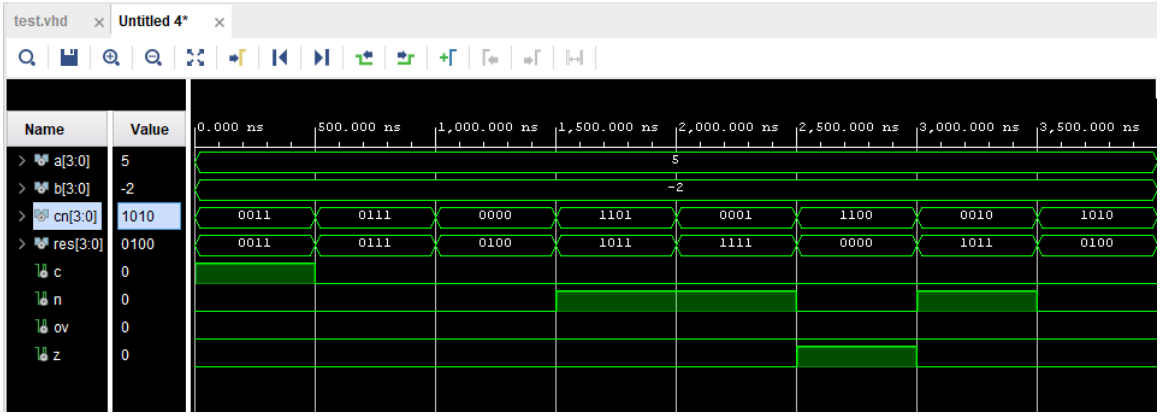
2. Generar la simulación con los siguientes estímulos y verificar que, tanto el vector de resultado como los bits de las banderas correspondan a los datos que se muestran en la tabla para TODOS los casos.

Estado de Banderas				Operación				
				1	0	0	0	Cn
				0	1	0	1	A = 5
OV	N	Z	C	1	1	1	0	B = -2
0	0	0	1	0	0	1	1	A+B
0	0	0	0	0	1	1	1	A-B
0	0	0	0	0	1	0	0	AND
0	1	0	0	1	0	1	1	NAND
0	1	0	0	1	1	1	1	OR
0	0	1	0	0	0	0	0	NOR
0	1	0	0	1	0	1	1	XOR
0	0	0	0	0	1	0	0	XNOR
				1	1	1	0	Cn
OV	N	Z	C	0	1	0	1	A = 5
				0	1	1	1	B = 7
1	1	0	0	1	1	0	0	A+B
				1	1	1	0	Cn
OV	N	Z	C	0	1	0	1	A
				0	1	0	1	B
0	0	1	1	0	0	0	0	A-B
0	1	0	0	1	0	1	0	NAND (NOT)

Simulaciones.

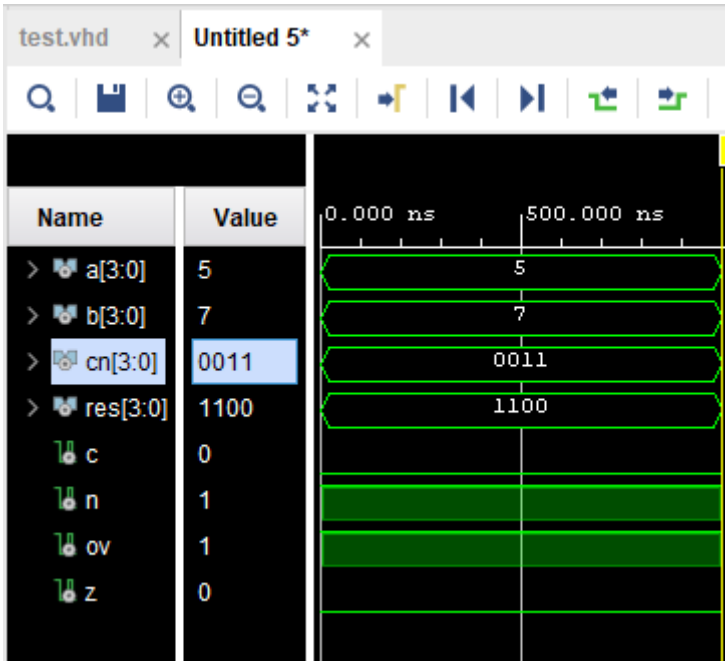
Primer bloque

Operandos A = 5 y B = -2



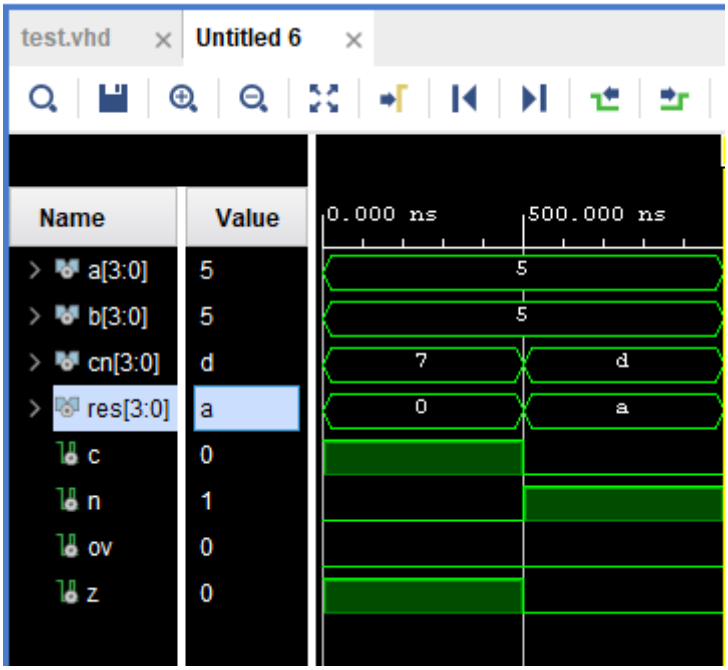
Bloque 2

Operandos A = 5 y B = 7



Bloque 3

Operandos A = 5 y B = 5



Códigos de implementación.

Archivo Suma1 (Sumador completo de un bit)

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY sum1 IS
    PORT (
        a, b, cin : IN STD_LOGIC;
        s, cout : OUT STD_LOGIC);
END sum1;

ARCHITECTURE Behavioral OF sum1 IS

BEGIN

    s <= a XOR b XOR cin;
    cout <= (a AND b) OR (b AND cin) OR (a AND cin);
END Behavioral;
```

Archivo alu1 (Unidad aritmética lógica de 1 bit)

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY alu1 IS
    PORT (
        a, b, cin, sela, selb : IN STD_LOGIC;
        op : IN STD_LOGIC_VECTOR (1 DOWNTO 0);
        res, cout : OUT STD_LOGIC);
END alu1;

ARCHITECTURE Behavioral OF alu1 IS

    COMPONENT sum1 IS
        PORT (
            a, b, cin : IN STD_LOGIC;
            s, cout : OUT STD_LOGIC);
    END COMPONENT;

    SIGNAL aux_a, aux_b, res_and, res_or, res_xor, res_suma : std_logic;
BEGIN


    aux_a <= a XOR sela;
    aux_b <= b XOR selb;
    res_and <= aux_a AND aux_b;
    res_or <= aux_a OR aux_b;
```

```
    res_xor <= aux_a XOR aux_b;

    sumador : sum1 PORT MAP(
        a => aux_a,
        b => aux_b,
        cin => cin,
        s => res_suma,
        cout => cout
    );

    res <= res_and WHEN op = "00" ELSE
        res_or WHEN op = "01" ELSE
        res_xor WHEN op = "10" ELSE
        res_suma;
END Behavioral;
```

Archivo alun (Unidad Aritmética Lógica de 4 bits)

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
ENTITY alun IS
    GENERIC (
        no : INTEGER := 4
    );
    PORT (
        a, b : IN STD_LOGIC_VECTOR (no - 1 DOWNTO 0);
        cn : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
        res : OUT STD_LOGIC_VECTOR (no - 1 DOWNTO 0);
        c : OUT STD_LOGIC;
        n : OUT STD_LOGIC;
        ov : OUT STD_LOGIC;
        z : OUT STD_LOGIC
    );
END alun;

ARCHITECTURE Behavioral OF alun IS

    COMPONENT alu1 IS
        PORT (
            a, b, cin, sela, selb : IN STD_LOGIC;
            op : IN STD_LOGIC_VECTOR (1 DOWNTO 0);
            res, cout : OUT STD_LOGIC);
    END COMPONENT;

    SIGNAL ca : std_logic_vector(no DOWNTO 0);
    SIGNAL re : std_logic_vector(no - 1 DOWNTO 0);
    SIGNAL q : std_logic;
```

```
BEGIN
    ca(0) <= cn(2);
    ciclo : FOR i IN 0 TO no - 1 GENERATE
        objeto1 : alu1 PORT MAP
        (
            a => a(i),
            b => b(i),
            cin => ca(i),
            sela => cn(3),
            selb => cn(2),
            op(1) => cn(1),
            op(0) => cn(0),
            res => re(i),
            cout => ca(i + 1)
        );
        --q<=re(i) or q;
    END GENERATE;
    c <= ca(no) WHEN (cn AND "0011") = "0011" ELSE
        '0';
    ov <= ca(no) XOR ca(no - 1) WHEN (cn AND "0011") = "0011" ELSE
        '0';
    n <= re(no - 1);
    --ov <= ca(no) xor ca(no-1);
    res <= re;
    --sq <= '0';
    z <= NOT(re(0) OR re(1) OR re(2) OR re(3));
```

Código de simulación

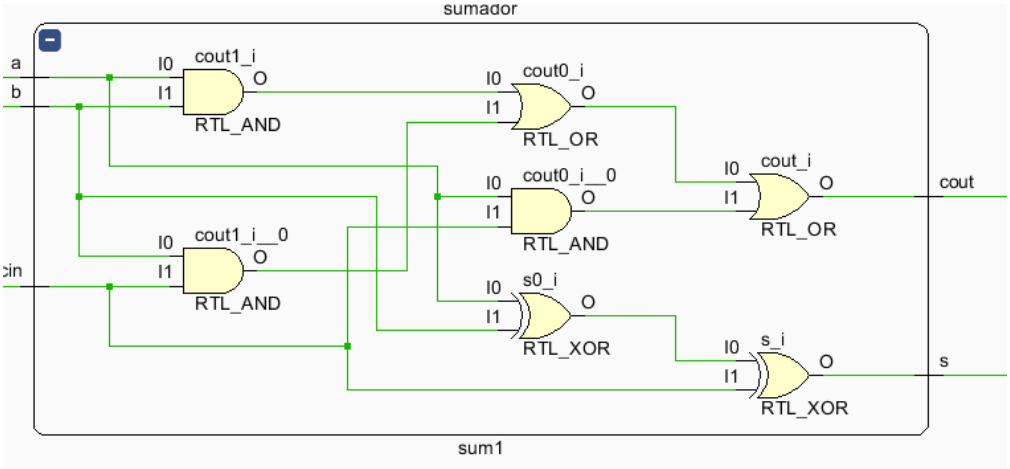
```
library ieee;
use ieee.std_logic_1164.all;

entity tb_alun is
end tb_alun;

architecture tb of tb_alun is

    component alun
        port (a   : in std_logic_vector (3 downto 0);
              b   : in std_logic_vector (3 downto 0);
              cn  : in std_logic_vector (3 downto 0);
              res : out std_logic_vector (3 downto 0);
              c   : out std_logic;
              n   : out std_logic;
              ov  : out std_logic;
              z   : out std_logic);
    end component;
```

```
    signal a   : std_logic_vector (3 downto 0);
    signal b   : std_logic_vector (3 downto 0);
    signal cn  : std_logic_vector (3 downto 0);
    signal res : std_logic_vector (3 downto 0);
    signal c   : std_logic;
    signal n   : std_logic;
    signal ov  : std_logic;
    signal z   : std_logic;

begin

    dut : alun
    port map (a   => a,
              b   => b,
              cn  => cn,
              res => res,
              c   => c,
              n   => n,
              ov  => ov,
              z   => z);

    stimuli : process
    begin
        -- Bloque 1
        a <= "0101";
        b <= "1110";
        cn <= "0011";
        wait for 500ns;
        cn <= "0111";
        wait for 500ns;
        cn <= "0000";
        wait for 500ns;
        cn <= "1101";
        wait for 500ns;
        cn <= "0001";
        wait for 500ns;
        cn <= "1100";
        wait for 500ns;
        cn <= "0010";
        wait for 500ns;
        cn <= "1010";
        wait for 500ns;

        -- Bloque 2
        a <= "0101";
        b <= "0111";
        cn <= "0011";
        wait for 500ns;
```

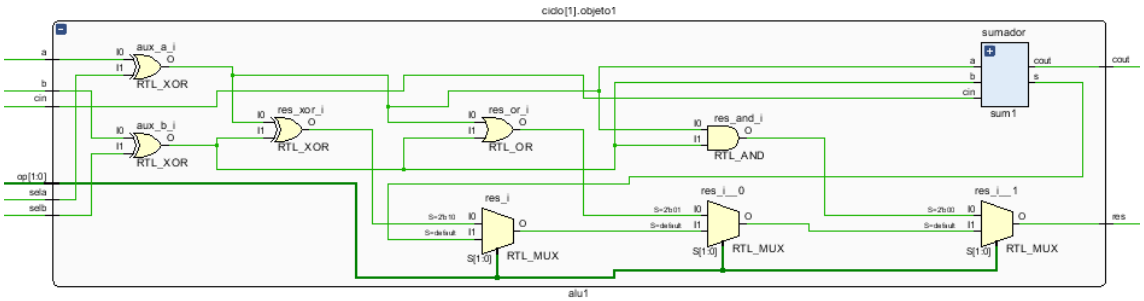
```
        -- Bloque 3
        a <= "0101";
        b <= "0101";
        cn <= "0111";
        wait for 500ns;
        cn <= "1101";
        wait;

        wait;
    end process;

end tb;
```

Diagramas RTL

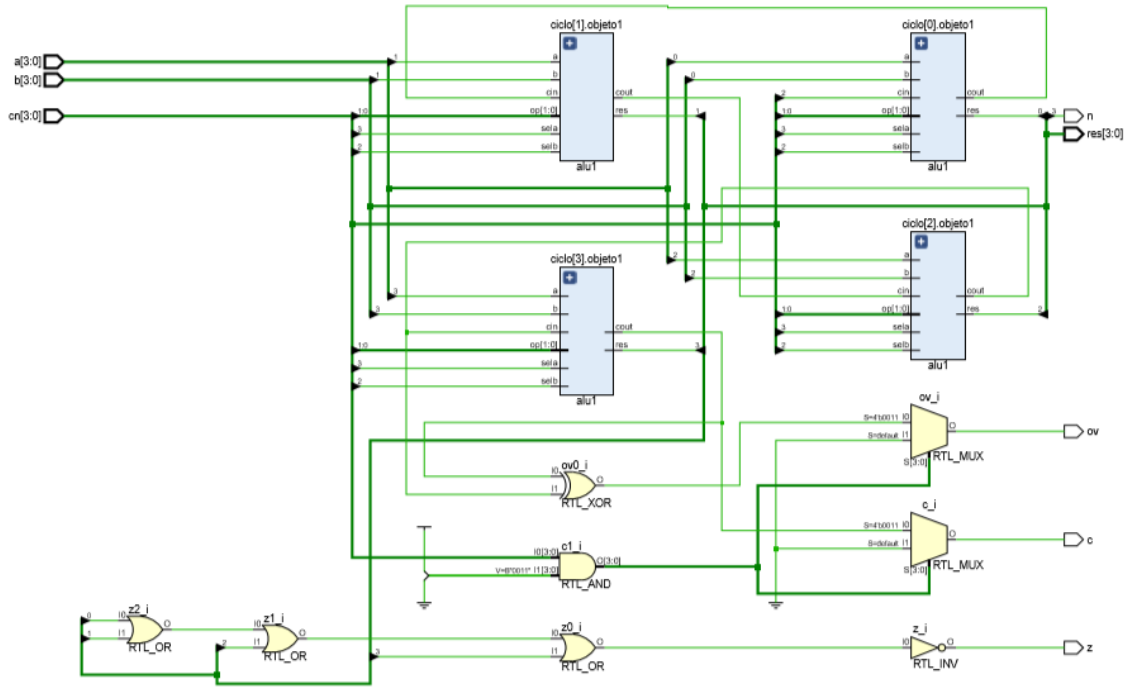
Suma1



Alu1



Alun


a[3:0]
b[3:0]
cn[3:0]
ciclo[1].objeto1
ciclo[0].objeto1
ciclo[2].objeto1
ciclo[3].objeto1
alu1
n
res[3:0]
ov_j
RTL_MUX
ov
ov0_j
RTL_XOR
c_j
RTL_MUX
c
c1_i
RTL_AND
z2_i
RTL_OR
z1_i
RTL_OR
z0_i
RTL_OR
z_i
RTL_INV
z