Instituto Politécnico Nacional

Escuela Superior de Cómputo

Práctica 4 - ALU de N = 4 bits

Unidad de aprendizaje: Arquitectura de Computadoras

Grupo: 3CV1

Alumno(a):
Ramos Diaz Enrique

Profesor(a):
Vega García Nayeli

4 de marzo 2020

Índice

1. Código de implementación

1.1. Suma de 1 bit

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity sum1bit is
    Port ( a, b, cin : in STD_LOGIC;
           s, cout : out STD_LOGIC);
end sum1bit;

architecture Behavioral of sum1bit is
begin
    s <= a xor b xor cin;
    cout <= (a and cin) or (a and b) or (b and cin);
end Behavioral;
```

1.2. ALU de 1 bit

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ALU1Bit is
    Port ( a, b, sela, selb, cin : in STD_LOGIC;
           op : in STD_LOGIC_VECTOR (1 downto 0);
           s, cout : out STD_LOGIC);
end ALU1Bit;

architecture Behavioral of ALU1Bit is
    signal auxa, auxb, and1, or1, xor1, suma1 : STD_LOGIC;

    component sum1bit is
        Port ( a, b, cin : in STD_LOGIC;
               s, cout : out STD_LOGIC);
    end component;
begin
    auxa <= a xor sela;
    auxb <= b xor selb;

    and1 <= auxa and auxb;
    or1 <= auxa or auxb;
    xor1 <= auxa xor auxb;

    suma : sum1bit port map (
        a => auxa,
        b => auxb,
        cin => cin,
        s => suma1,
        cout => cout
    );

    process(op, and1, xor1, or1, suma1)
    begin
        case op is
            when "00" => s <= and1;
            when "01" => s <= or1;
            when "10" => s <= xor1;
            when others => s <= suma1;
            end case;
    end process;

end Behavioral;
```

1.3. ALU de N = 4 bits

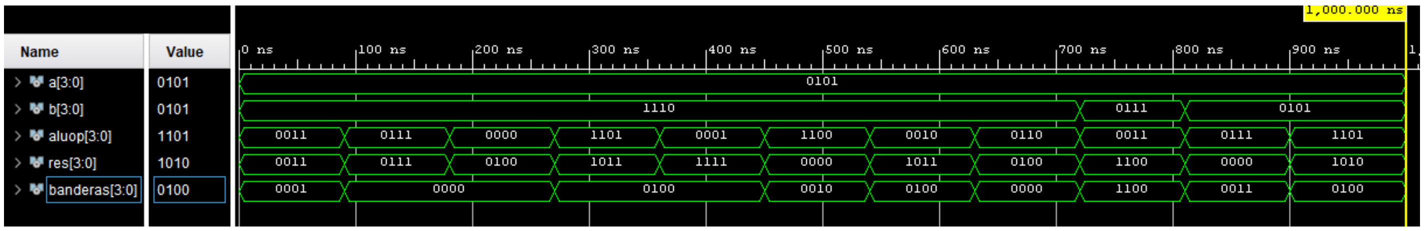
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ALUNBits is
    generic ( n : integer := 4);
    Port ( a, b : in STD_LOGIC_VECTOR (n-1 downto 0);
           aluop : in STD_LOGIC_VECTOR (3 downto 0);
           res : out STD_LOGIC_VECTOR (n-1 downto 0);
           banderas : out STD_LOGIC_VECTOR (3 downto 0));
end ALUNBits;

architecture Behavioral of ALUNBits is
    signal c : STD_LOGIC_VECTOR(n downto 0);
    signal res_aux : STD_LOGIC_VECTOR (n-1 downto 0);
    component ALU1Bit is
        Port ( a, b, sela, selb, cin : in STD_LOGIC;
               op : in STD_LOGIC_VECTOR (1 downto 0);
               s, cout : out STD_LOGIC);
    end component;
begin
```

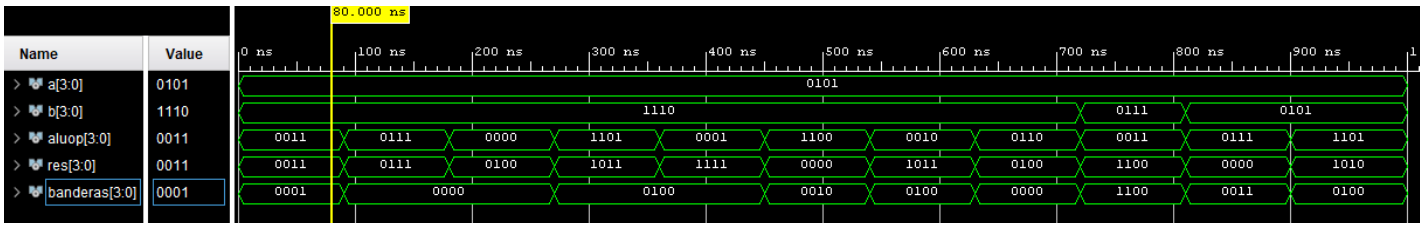
```
    -- sela, selb, op0, op1
    c(0) <= aluop(2);
    ciclo : for i in 0 to n-1 generate
        alu0 : ALU1Bit
            Port map (
                a => a(i),
                b => b(i),
                cin => c(i),
                sela => aluop(3),
                selb => aluop(2),
                s => res_aux(i),
                cout => c(i+1),
                op => aluop(1 downto 0)
            );
    end generate;
    res <= res_aux;

    process(res_aux, c, aluop)
        variable z: STD_LOGIC;
    begin
        case aluop(1 downto 0) is
            when "11" =>
                banderas(3) <= c(n) xor c(n-1); --OV
                banderas(0) <= c(n); -- C
            when others =>
                banderas(3) <= '0'; --OV
                banderas(0) <= '0'; --C
        end case;

        banderas(2) <= res_aux(n-1); -- N

        z := '0';
        zero: for j in n-1 downto 0 loop
            z := z or res_aux(j);
        end loop;

        banderas(1) <= not z; -- Z
    end process;
end Behavioral;
```

2. Código de simulación

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity test_bench is
end test_bench;

architecture Behavioral of test_bench is
    component ALUNBits is
        Port ( a, b : in STD_LOGIC_VECTOR (3 downto 0);
               aluop : in STD_LOGIC_VECTOR (3 downto 0);
               res : out STD_LOGIC_VECTOR (3 downto 0);
               banderas : out STD_LOGIC_VECTOR (3 downto 0));
    end component;

    signal a, b : STD_LOGIC_VECTOR (3 downto 0);
    signal aluop : STD_LOGIC_VECTOR (3 downto 0);
    signal res : STD_LOGIC_VECTOR (3 downto 0);
    signal banderas : STD_LOGIC_VECTOR (3 downto 0);
begin
    ALU: ALUNBits Port map (
        a => a,
        b => b,
        aluop => aluop,
        res => res,
        banderas => banderas
    );

    process begin
        a <= "0101"; -- 5
        b <= "1110"; -- -2
        aluop <= "0011"; -- A + B
        wait for 90 ns;

        aluop <= "0111"; -- A - B
        wait for 90 ns;

        aluop <= "0000"; -- A AND B
        wait for 90 ns;

        aluop <= "1101"; -- A NAND B
        wait for 90 ns;

        aluop <= "0001"; -- A OR B
        wait for 90 ns;

        aluop <= "1100"; -- A NOR B
        wait for 90 ns;

        aluop <= "0010"; -- A XOR B
        wait for 90 ns;

        aluop <= "0110"; -- A XNOR B
        wait for 90 ns;

        b <= "0111"; -- 7
        aluop <= "0011"; -- A + B
        wait for 90 ns;

        b <= "0101"; -- 5
        aluop <= "0111"; -- A - B
        wait for 90 ns;

        aluop <= "1101"; -- A NAND (NOT) B
        wait;
    end process;
end Behavioral;
```

3. Simulación

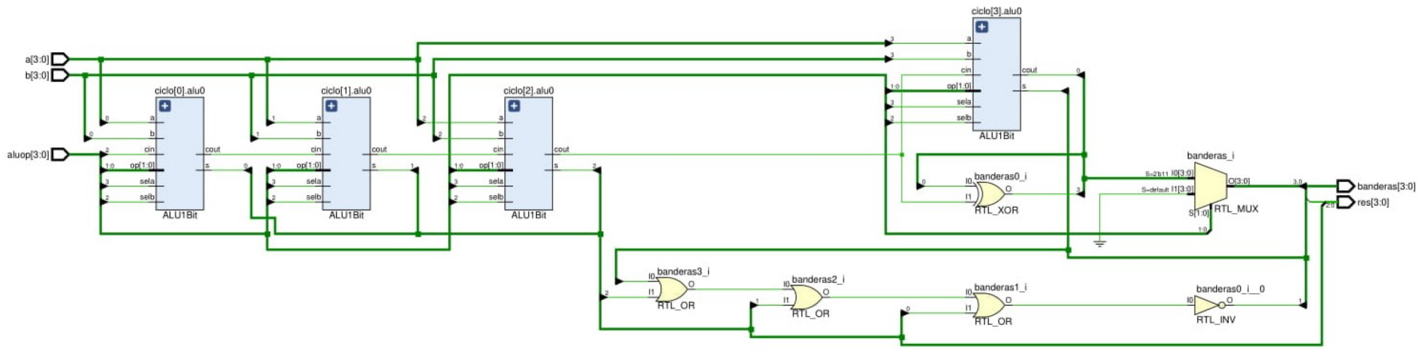




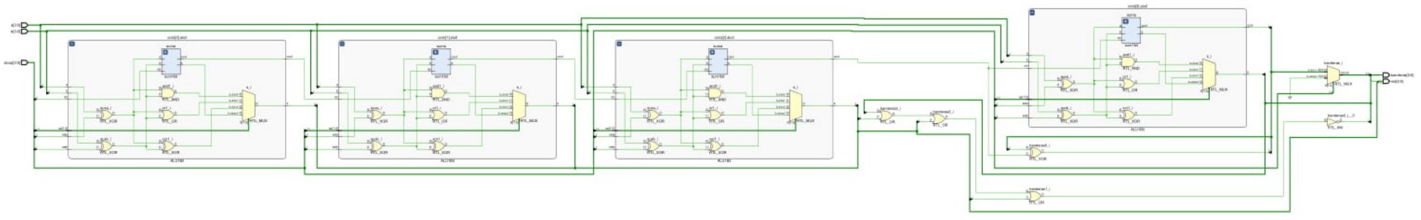
4. Diagramas RTL

4.1. Análisis RTL

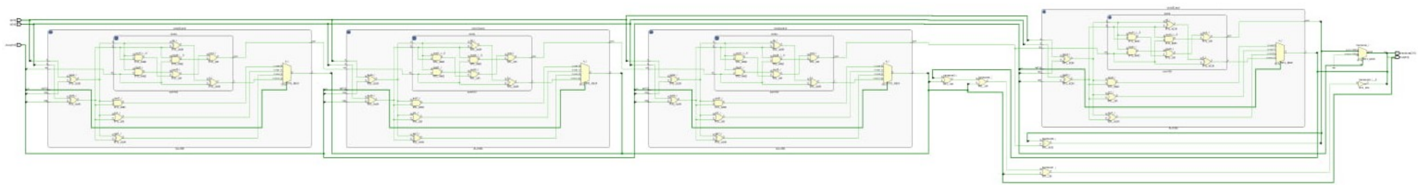
4.1.1. Diagrama comprimido



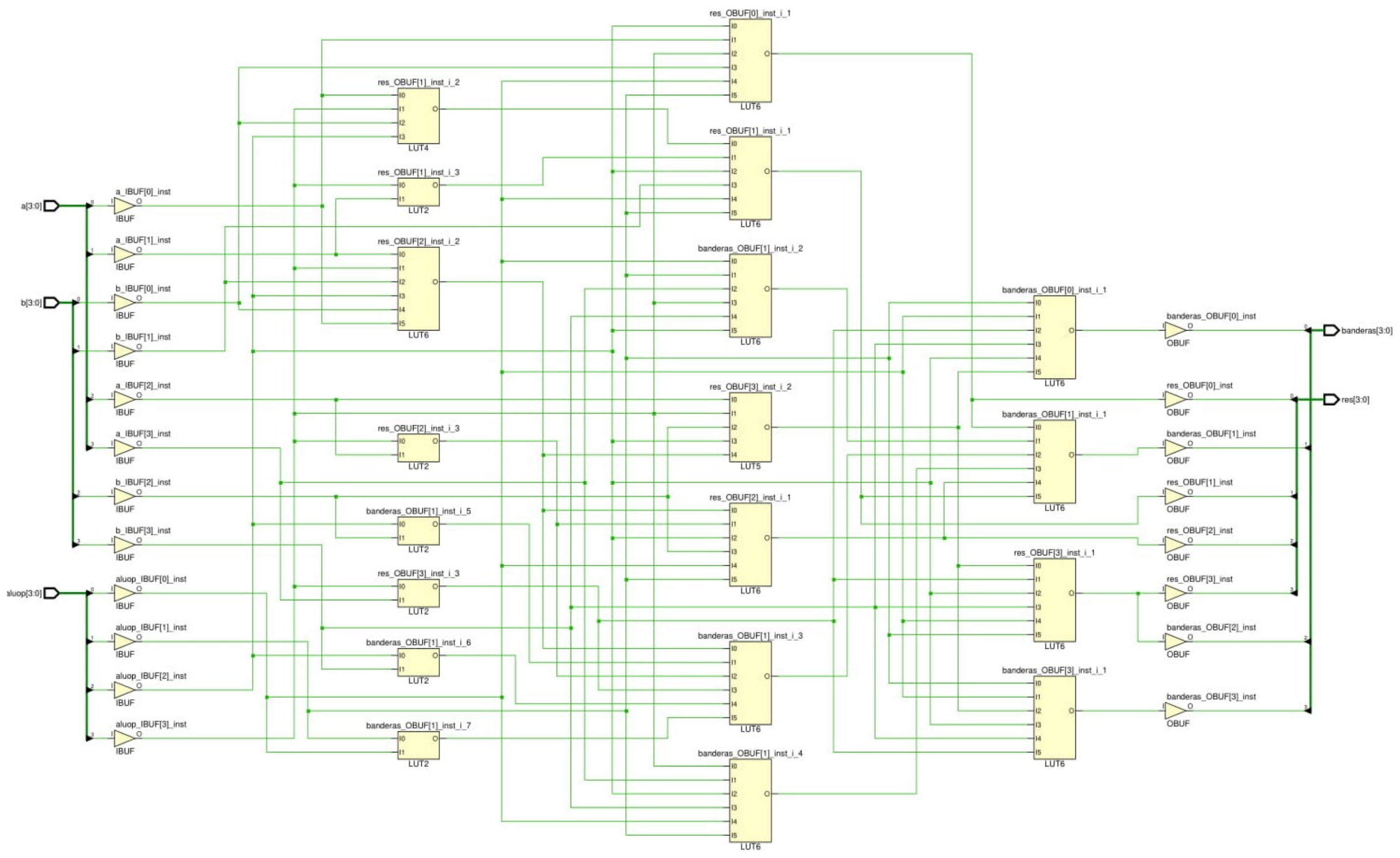
4.1.2. ALUs expandidas



4.1.3. ALUs y sumadores expandidos



4.2. Synthesis


a[3:0]
b[3:0]
aluop[3:0]
a_IBUF[0]_inst
a_IBUF[1]_inst
b_IBUF[0]_inst
b_IBUF[1]_inst
a_IBUF[2]_inst
a_IBUF[3]_inst
b_IBUF[2]_inst
b_IBUF[3]_inst
aluop_IBUF[0]_inst
aluop_IBUF[1]_inst
aluop_IBUF[2]_inst
aluop_IBUF[3]_inst
IBUF
res_OBUF[1]_inst_i_2
LUT4
res_OBUF[1]_inst_i_3
res_OBUF[2]_inst_i_2
res_OBUF[2]_inst_i_3
banderas_OBUF[1]_inst_i_5
res_OBUF[3]_inst_i_3
banderas_OBUF[1]_inst_i_6
banderas_OBUF[1]_inst_i_7
LUT2
res_OBUF[0]_inst_i_1
res_OBUF[1]_inst_i_1
banderas_OBUF[1]_inst_i_2
res_OBUF[3]_inst_i_2
LUT5
res_OBUF[2]_inst_i_1
banderas_OBUF[1]_inst_i_3
banderas_OBUF[1]_inst_i_4
banderas_OBUF[0]_inst_i_1
banderas_OBUF[1]_inst_i_1
res_OBUF[3]_inst_i_1
banderas_OBUF[3]_inst_i_1
LUT6
banderas_OBUF[0]_inst
res_OBUF[0]_inst
banderas_OBUF[1]_inst
res_OBUF[1]_inst
res_OBUF[2]_inst
res_OBUF[3]_inst
banderas_OBUF[2]_inst
banderas_OBUF[3]_inst
OBUF
banderas[3:0]
res[3:0]