Reporte de practica 4

Pulido Bejarano Raymundo

Marzo 2020

1. Código VHDL

```
1
2  library IEEE;
3  use IEEE.STD_LOGIC_1164.ALL;
4
5
6  entity sumador is
7      Port ( a,b,cin : in STD_LOGIC;
8             s : out STD_LOGIC;
9             cout : out STD_LOGIC);
10 end sumador;
11
12 architecture Behavioral of sumador is
13 begin
14     s<=a xor b xor cin;
15     cout<= (a and cin)or(b and cin)or(a and b);
16
17 end Behavioral;
```

Código fuente de sumador de 1 bit

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity alu1 is
5      Port ( a, b, sela, selb, cin : in STD_LOGIC;
6             op : in STD_LOGIC_VECTOR (1 downto 0);
7             res : out STD_LOGIC;
8             cout : out STD_LOGIC);
9  end alu1;
10
11 architecture Behavioral of alu1 is
12
13 component sumador is
14     Port ( a,b,cin : in STD_LOGIC;
15            s : out STD_LOGIC;
16            cout : out STD_LOGIC);
17 end component;
18 signal auxa, auxb, auxs, and1, or1, xor1 ,auxc: std_logic;
```

```
begin


auxa <= a xor sela;
auxb <= b xor selb;

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


sumador1: sumador
    Port map(
    a => auxa,
    b => auxb,
    cin => cin,
    s => auxs,
    cout => auxc
    );

     process(and1, or1, xor1, auxs, op)
     begin
        case op is
            when "00" =>res <= and1;
            cout<='0';
            when "01" =>res <= or1;
            cout<='0';
            when "10" =>res <= xor1;
            cout<='0';
            when others =>res <= auxs;
            cout<=auxc;
        end case;
     end process;







end Behavioral;
```

Código fuente de ALU de 1 bit

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

entity AluN is
generic(
    m: integer:= 4);
    Port ( a, b : in STD_LOGIC_VECTOR (m-1 downto 0);
           aluop : in STD_LOGIC_VECTOR (3 downto 0);
           res : out STD_LOGIC_VECTOR (m-1 downto 0);
           n,z,ov,co : out STD_LOGIC
           );
```

```
end AluN;

architecture Behavioral of AluN is
component alu1 is
    Port ( a, b, sela, selb, cin : in STD_LOGIC;
           op : in STD_LOGIC_VECTOR (1 downto 0);
           res : out STD_LOGIC;
           cout : out STD_LOGIC);
end component;
signal c: std_logic_vector(m downto 0);
signal aux: std_logic_vector(m-1 downto 0);

begin

    c(0) <= aluop(2);
    ciclo : for i in 0 to m-1 generate
        bitA : alu1
        Port  map(
            a => a(i),
            b => b(i),
            sela => aluop(3),
            selb => aluop(2),
            cin => c(i),
            op  => aluop(1 downto 0),
            res => aux(i),
            cout => c(i+1)
        );
    end generate;

    process(aux,aluop,c)
    variable auxT:std_logic;
    begin
        auxT:='0';
        for i in 0 to m-1 loop
            auxT:=auxT or aux(i);
        end loop;
        z <= not (auxT);
        res <=aux;
        co <= c(m);
        n <= aux(m-1); --signo
        --z <= '1' when aux = "0000" else '0'; --cero
        ov <= c(m) xor c(m-1);--overflow
    end process;


end Behavioral;
```

Código fuente de la ALU de m bits generalizada

2. Test-Bench VHDL Código

```
library IEEE;
```

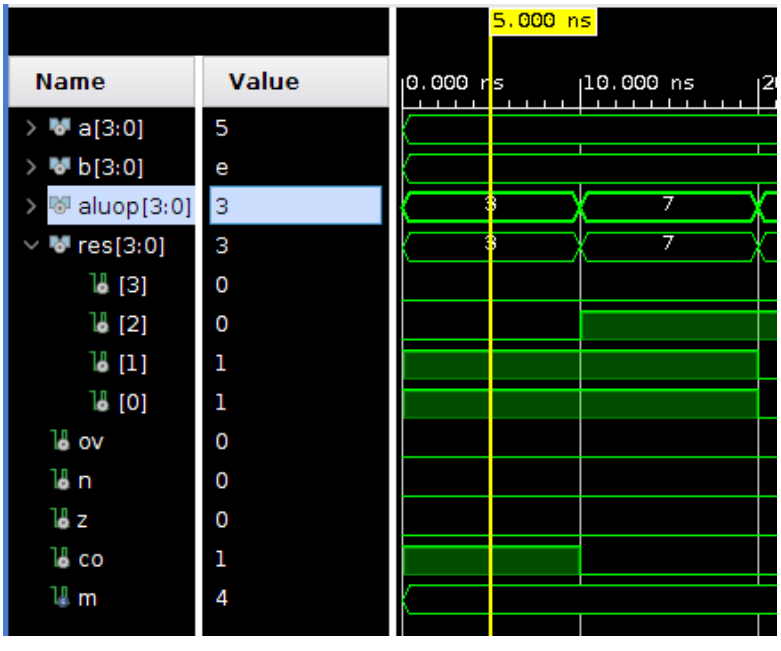
```
use IEEE.STD_LOGIC_1164.ALL;
entity simAlumN is
    generic(
        m: integer:=4);
--  Port ( );
end simAlumN;

architecture Behavioral of simAlumN is
    component AluN
        Port ( a, b : in STD_LOGIC_VECTOR (m-1 downto 0);
               aluop : in STD_LOGIC_VECTOR (3 downto 0);
               res : out STD_LOGIC_VECTOR (m-1 downto 0);
               n,z,ov,co : out STD_LOGIC
              );
    end component;
    signal a:STD_LOGIC_VECTOR (m-1 downto 0):=x"0";
    signal b:STD_LOGIC_VECTOR (m-1 downto 0):=x"0";
    signal aluop:STD_LOGIC_VECTOR (3 downto 0):=x"0";
    signal res :STD_LOGIC_VECTOR (m-1 downto 0):=x"0";
    signal ov:STD_LOGIC:='0';
    signal n:STD_LOGIC:='0';
    signal z:STD_LOGIC:='0';
    signal co:STD_LOGIC:='0';
begin
    alu:AluN
        Port map(
            a=>a,
            b=>b,
            aluop=>aluop,
            res=>res,
            n=>n,
            z=>z,
            ov=>ov,
            co=>co
        );
    p3:process
    begin
    --Suma
    a<=x"5";
    b<="1110";
    aluop<="0011";
    wait for 10 ns;
    --Resta
    aluop<="0111";
    wait for 10 ns;
    --AND
    aluop<="0000";
    wait for 10 ns;
    --NAND
    aluop<="1101";
    wait for 10 ns;
    --OR
    aluop<="0001";
```

```
55      wait for 10 ns;
56      --NOR
57      aluop<="1100";
58      wait for 10 ns;
59      --XOR
60      aluop<="0010";
61      wait for 10 ns;
62      --XNOR
63      aluop<="1010";
64      wait for 10 ns;
65      --B=7 Suma
66      b<=x"7";
67      aluop<="0011";
68      wait for 10 ns;
69      --Resta
70      b<=x"5";
71      aluop<="0111";
72      wait for 10 ns;
73      --Nand
74      aluop<="1101";
75      wait for 10 ns;
76      wait;
77      end process;
78  end Behavioral;
```

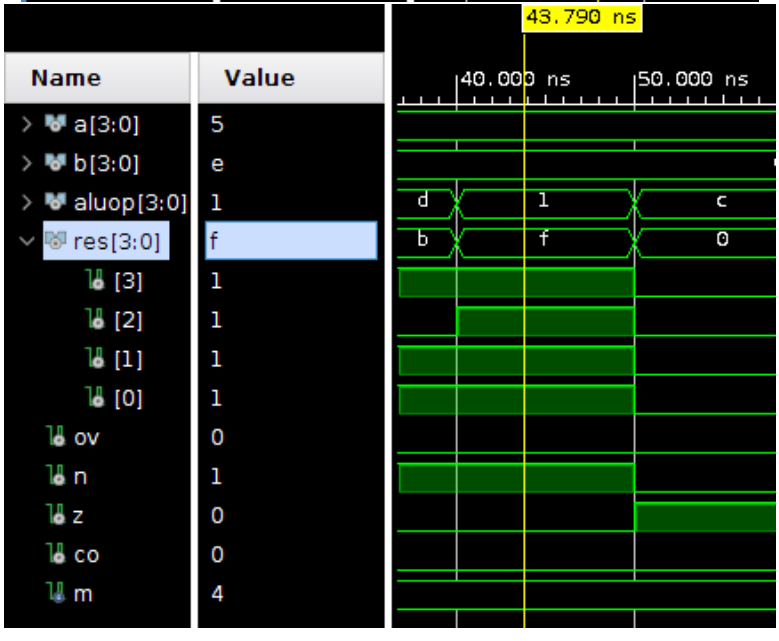
Código de simulación

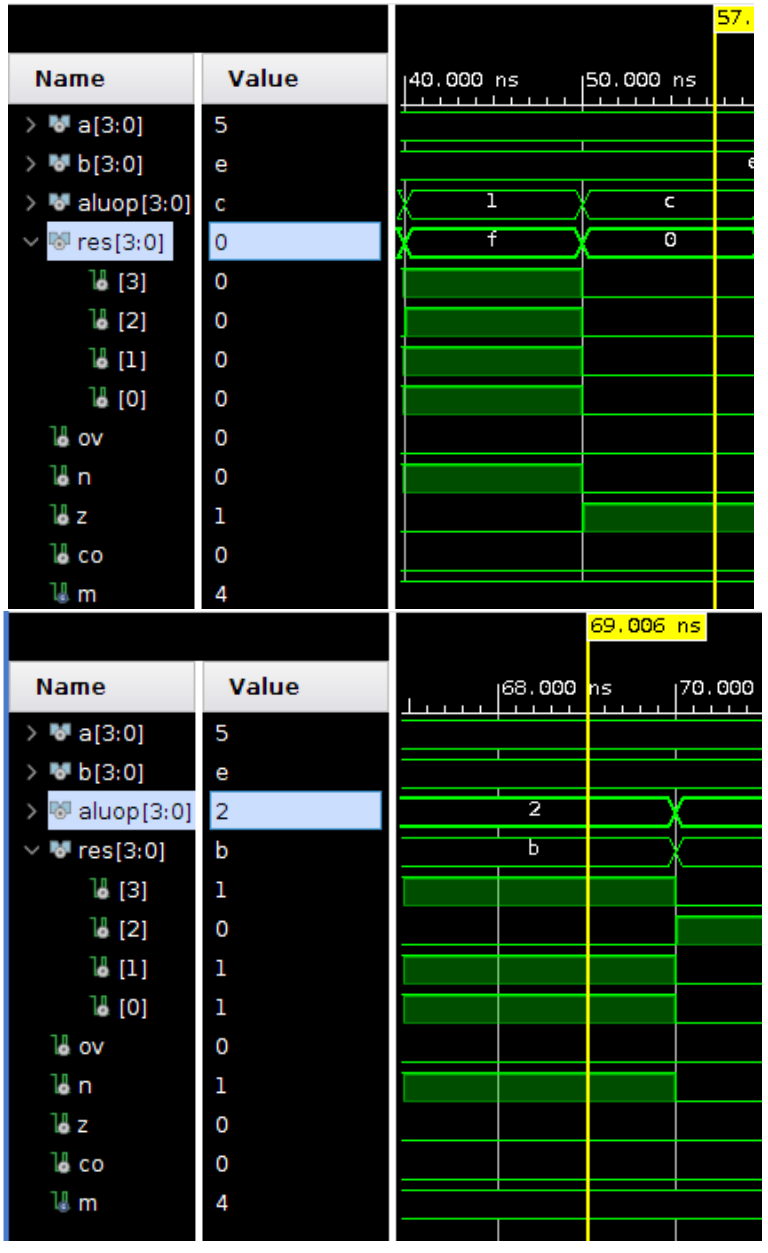
Anexo de fotos de la simulación a impulsos

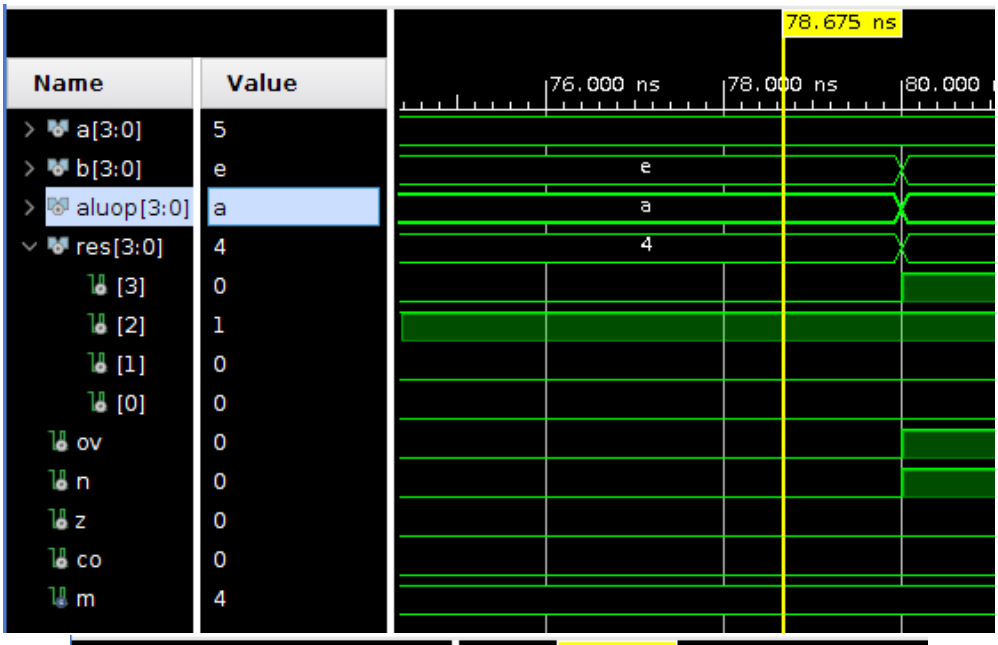


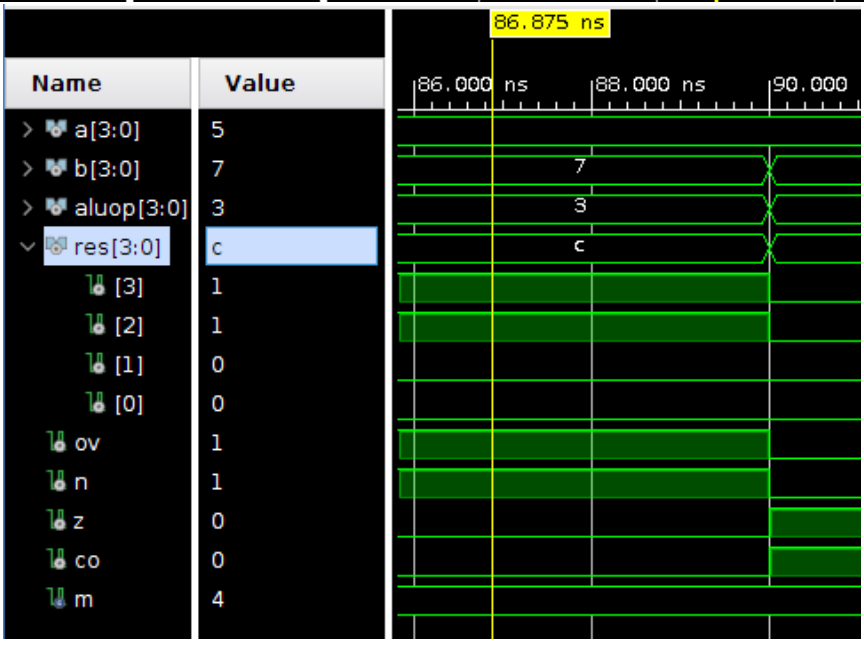

Name
Value
a[3:0]
5
b[3:0]
e
aluop[3:0]
7
res[3:0]
7
[3]
0
[2]
1
[1]
1
[0]
1
ov
0
n
0
z
0
co
0
m
4
13.500 ns
10.000 ns
20.000 n
3
7
0
3
7
4
Name
Value
a[3:0]
5
b[3:0]
e
aluop[3:0]
0
res[3:0]
4
[3]
0
[2]
1
[1]
0
[0]
0
ov
0
n
0
z
0
co
0
m
4
26.770 ns
20.000 ns
30.000
7
0
7
4

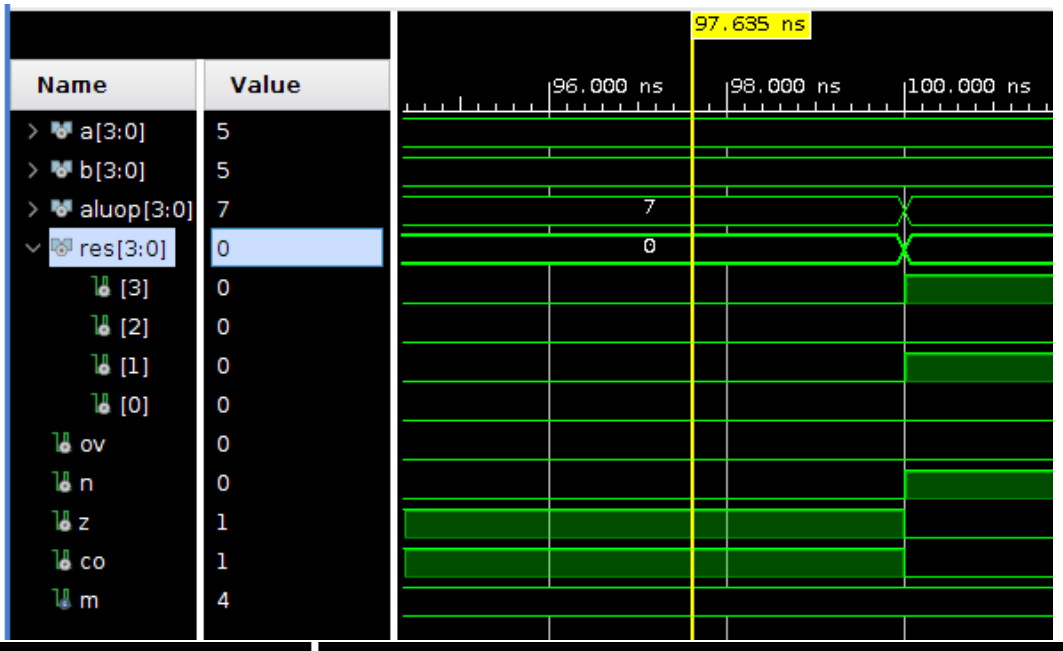

Name
Value
a[3:0]
5
b[3:0]
e
aluop[3:0]
d
res[3:0]
b
[3]
1
[2]
0
[1]
1
[0]
1
ov
0
n
1
z
0
co
0
m
4
37.480 ns
30.000 ns
40.000 ns
0
d
1
4
b
f

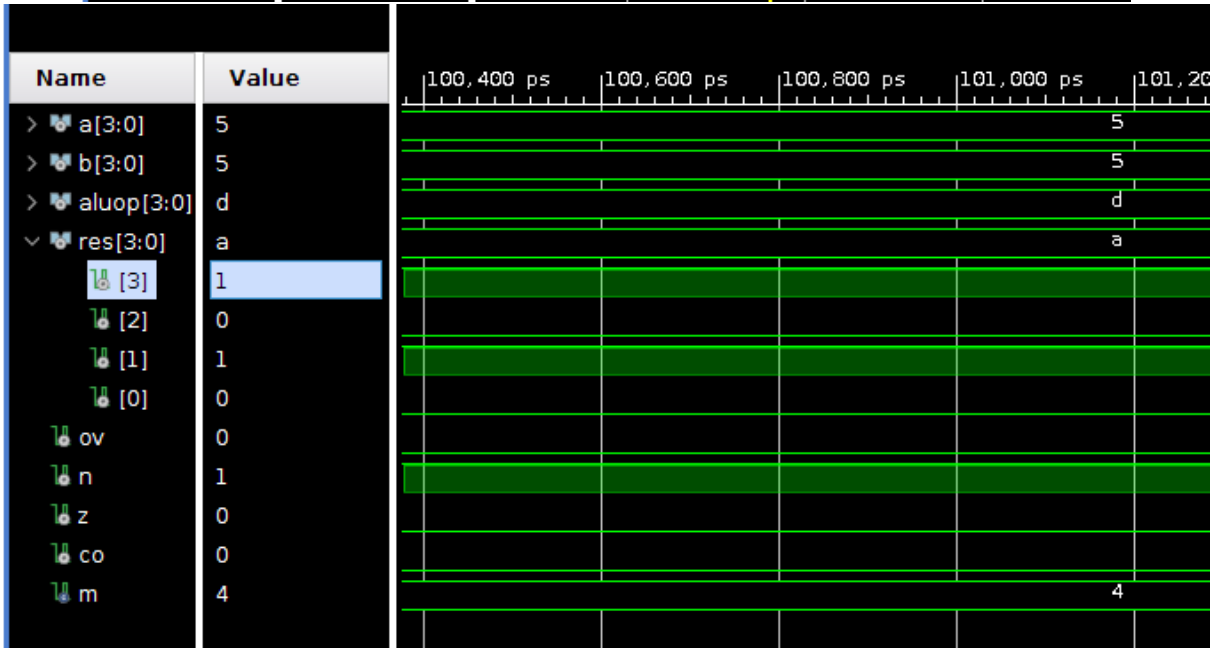

Name
Value
a[3:0]
5
b[3:0]
e
aluop[3:0]
1
res[3:0]
f
[3]
1
[2]
1
[1]
1
[0]
1
ov
0
n
1
z
0
co
0
m
4
43.790 ns
40.000 ns
50.000 ns
d
1
c
b
f
0


Name
Value
a[3:0]
5
b[3:0]
e
aluop[3:0]
c
res[3:0]
0
[3]
0
[2]
0
[1]
0
[0]
0
ov
0
n
0
z
1
co
0
m
4
40.000 ns
50.000 ns
57.
1
c
f
0
Name
Value
a[3:0]
5
b[3:0]
e
aluop[3:0]
2
res[3:0]
b
[3]
1
[2]
0
[1]
1
[0]
1
ov
0
n
1
z
0
co
0
m
4
69.006 ns
68.000 ns
70.000
2
b


Name
Value
a[3:0]
5
b[3:0]
e
aluop[3:0]
a
res[3:0]
4
[3]
0
[2]
1
[1]
0
[0]
0
ov
0
n
0
z
0
co
0
m
4
78.675 ns
76.000 ns
78.000 ns
80.000
e
a
4


Name
Value
a[3:0]
5
b[3:0]
7
aluop[3:0]
3
res[3:0]
c
[3]
1
[2]
1
[1]
0
[0]
0
ov
1
n
1
z
0
co
0
m
4
86.875 ns
86.000 ns
88.000 ns
90.000
7
3
c


Name
Value
a[3:0] 5
b[3:0] 5
aluop[3:0] 7
res[3:0] 0
[3] 0
[2] 0
[1] 0
[0] 0
ov 0
n 0
z 1
co 1
m 4
97.635 ns
96.000 ns
98.000 ns
100.000 ns
7
0


Name
Value
a[3:0] 5
b[3:0] 5
aluop[3:0] d
res[3:0] a
[3] 1
[2] 0
[1] 1
[0] 0
ov 0
n 1
z 0
co 0
m 4
100,400 ps
100,600 ps
100,800 ps
101,000 ps
5
5
d
a
4

3. Tabla de resultados

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)

Figura: Tabla que concuerda con los resultados obtenidos en la simulación

4. Diagrama RTL

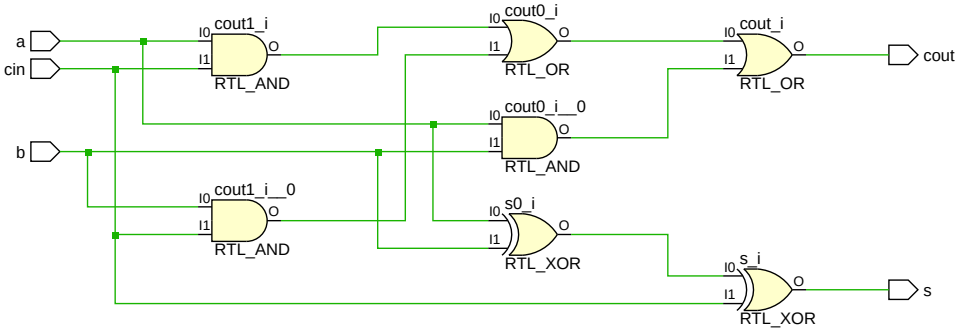


Diagrama RTL del archivo VHDL del sumador de 1 bit

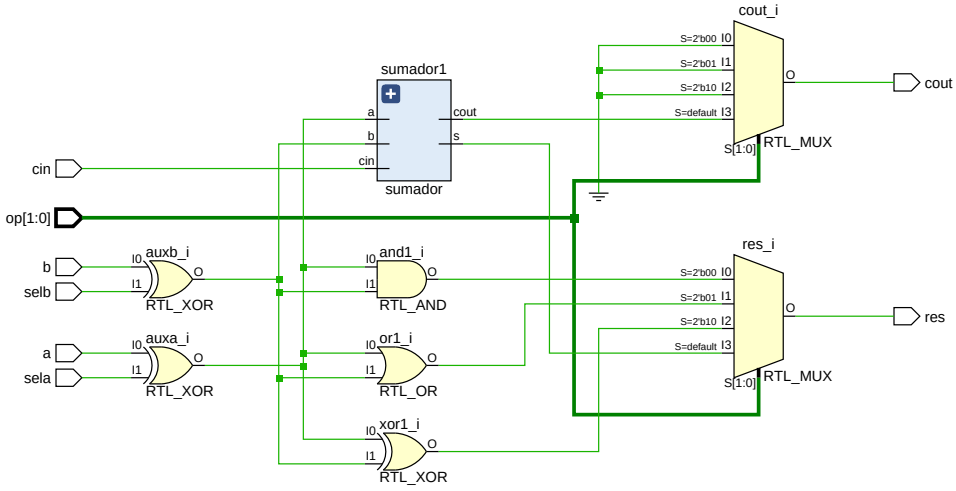


Diagrama RTL del archivo VHDL de la ALU de 1 bit

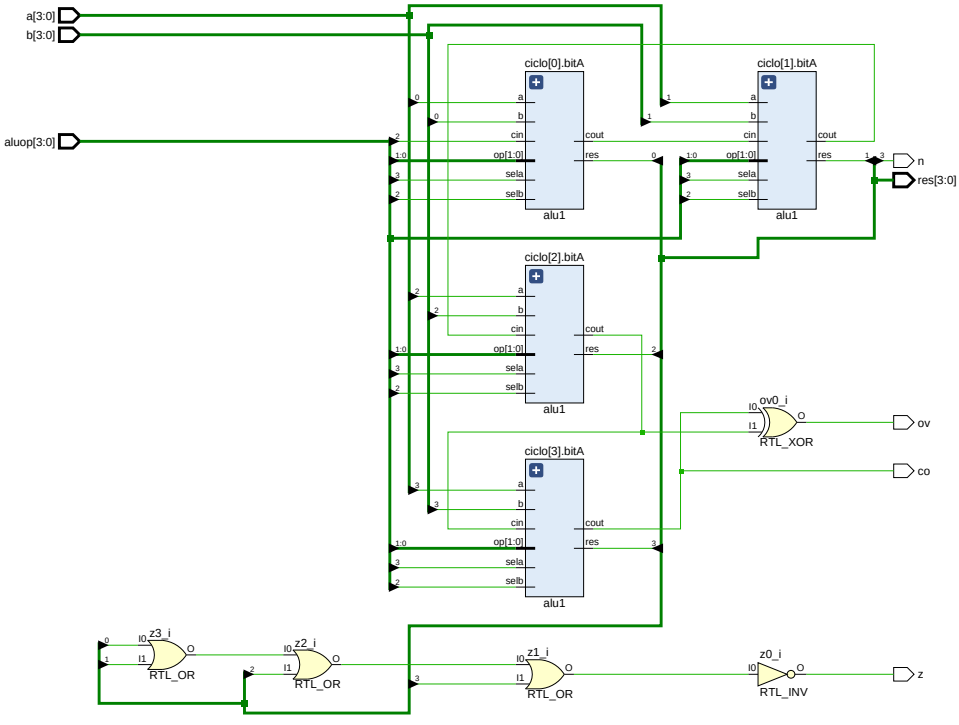


Diagrama RTL del archivo VHDL de la ALU de m bits