

# Reporte de practica 4

González Pardo Adrian

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;
```

```

19 begin
20
21
22 auxa <= a xor sela;
23 auxb <= b xor selb;
24
25 and1 <= auxa and auxb;
26 or1 <= auxa or auxb;
27 xor1 <= auxa xor auxb;
28
29
30 sumador1: sumador
31     Port map(
32         a => auxa,
33         b => auxb,
34         cin => cin,
35         s => auxs,
36         cout => auxc
37     );
38
39     process(and1, or1, xor1, auxs, op)
40     begin
41         case op is
42             when "00" => res <= and1;
43             cout <= '0';
44             when "01" => res <= or1;
45             cout <= '0';
46             when "10" => res <= xor1;
47             cout <= '0';
48             when others => res <= auxs;
49             cout <= auxc;
50         end case;
51     end process;
52
53
54
55
56
57
58 end Behavioral;

```

### *Código fuente de ALU de 1 bit*

```

1 library IEEE;
2 use IEEE.STD_LOGIC_1164.ALL;
3
4 entity AluN is
5     generic(
6         m: integer:= 4);
7     Port ( a, b : in STD_LOGIC_VECTOR (m-1 downto 0);
8           aluop : in STD_LOGIC_VECTOR (3 downto 0);
9           res : out STD_LOGIC_VECTOR (m-1 downto 0);
10          n,z,ov,co : out STD_LOGIC
11        );

```

```

12 end AluN;
13
14 architecture Behavioral of AluN is
15 component alu1 is
16     Port ( a, b, sela, selb, cin : in STD_LOGIC;
17           op : in STD_LOGIC_VECTOR (1 downto 0);
18           res : out STD_LOGIC;
19           cout : out STD_LOGIC);
20 end component;
21 signal c: std_logic_vector(m downto 0);
22 signal aux: std_logic_vector(m-1 downto 0);
23
24 begin
25
26     c(0) <= aluop(2);
27     ciclo : for i in 0 to m-1 generate
28         bitA : alu1
29         Port map(
30             a => a(i),
31             b => b(i),
32             sela => aluop(3),
33             selb => aluop(2),
34             cin => c(i),
35             op => aluop(1 downto 0),
36             res => aux(i),
37             cout => c(i+1)
38         );
39     end generate;
40
41     process(aux,aluop,c)
42     variable auxT:std_logic;
43     begin
44         auxT:='0';
45         for i in 0 to m-1 loop
46             auxT:=auxT or aux(i);
47         end loop;
48         z <= not (auxT);
49         res <=aux;
50         co <= c(m);
51         n <= aux(m-1); --signo
52         --z <= '1' when aux = "0000" else '0'; --cero
53         ov <= c(m) xor c(m-1);--overflow
54     end process;
55
56
57 end Behavioral;

```

*Código fuente de la ALU de m bits generalizada*

## 2. Test-Bench VHDL Código

```

1 library IEEE;

```

```

2 use IEEE.STD_LOGIC_1164.ALL;
3 entity simAlumN is
4     generic(
5         m: integer:=4);
6     -- Port ( );
7 end simAlumN;
8
9 architecture Behavioral of simAlumN is
10     component AluN
11         Port ( a, b : in STD_LOGIC_VECTOR (m-1 downto 0);
12             aluop : in STD_LOGIC_VECTOR (3 downto 0);
13             res : out STD_LOGIC_VECTOR (m-1 downto 0);
14             n,z,ov,co : out STD_LOGIC
15             );
16     end component;
17     signal a:STD_LOGIC_VECTOR (m-1 downto 0):=x"0";
18     signal b:STD_LOGIC_VECTOR (m-1 downto 0):=x"0";
19     signal aluop:STD_LOGIC_VECTOR (3 downto 0):=x"0";
20     signal res :STD_LOGIC_VECTOR (m-1 downto 0):=x"0";
21     signal ov:STD_LOGIC:='0';
22     signal n:STD_LOGIC:='0';
23     signal z:STD_LOGIC:='0';
24     signal co:STD_LOGIC:='0';
25 begin
26     alu:AluN
27         Port map(
28             a=>a,
29             b=>b,
30             aluop=>aluop,
31             res=>res,
32             n=>n,
33             z=>z,
34             ov=>ov,
35             co=>co
36         );
37     p3:process
38     begin
39         --Suma
40         a<=x"5";
41         b<="1110";
42         aluop<="0011";
43         wait for 10 ns;
44         --Resta
45         aluop<="0111";
46         wait for 10 ns;
47         --AND
48         aluop<="0000";
49         wait for 10 ns;
50         --NAND
51         aluop<="1101";
52         wait for 10 ns;
53         --OR
54         aluop<="0001";

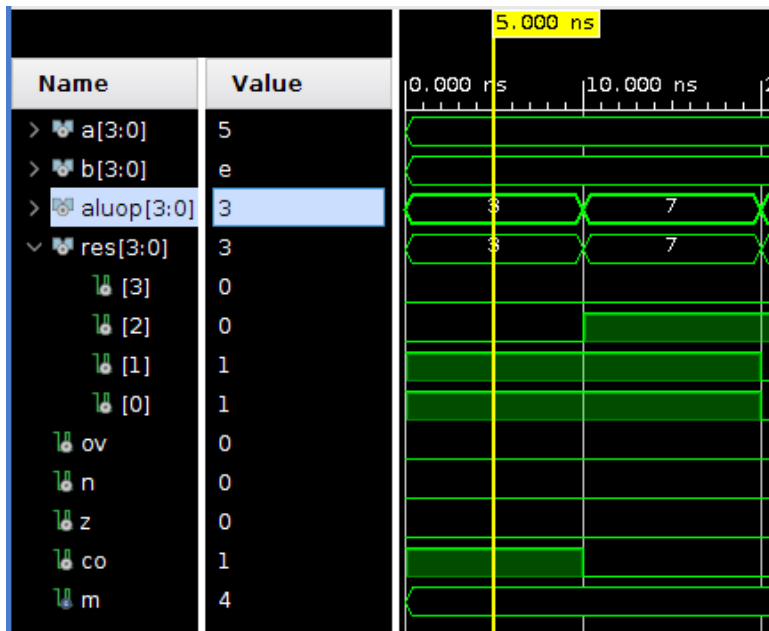
```

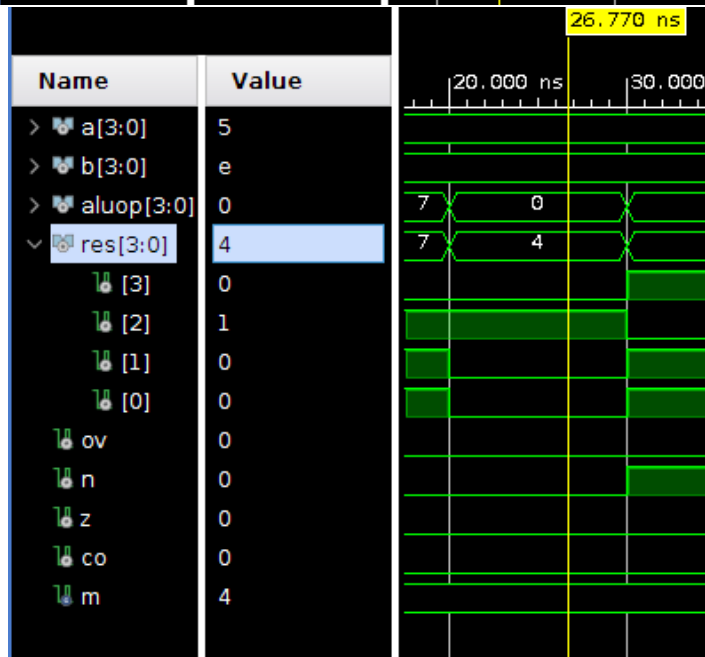
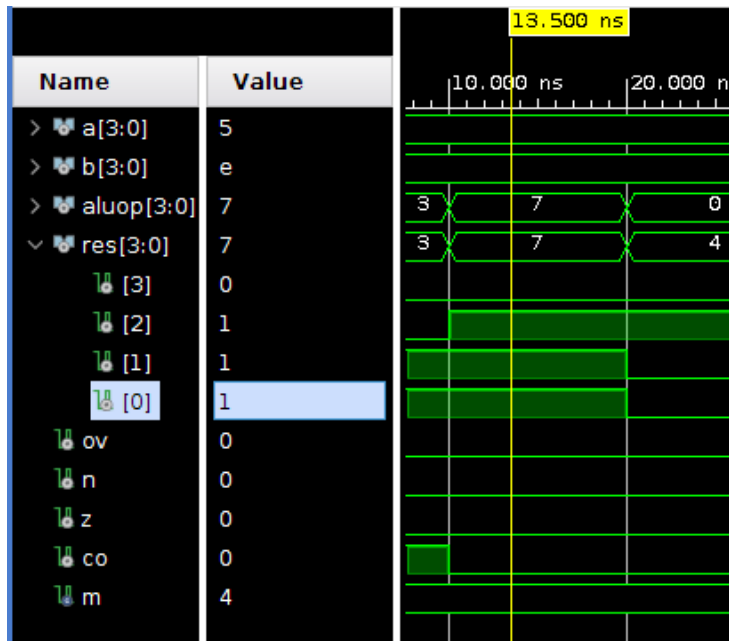
```

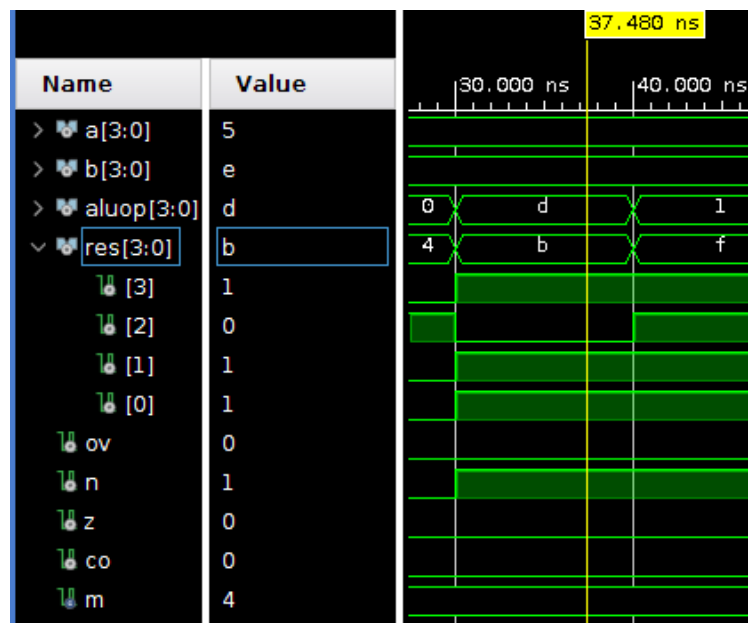
55     wait for 10 ns;
56     --NOR
57     aluop<="1100";
58     wait for 10 ns;
59     --XOR
60     aluop<="0010";
61     wait for 10 ns;
62     --XNOR
63     aluop<="1010";
64     wait for 10 ns;
65     --B=7 Suma
66     b<=x"7";
67     aluop<="0011";
68     wait for 10 ns;
69     --Resta
70     b<=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**

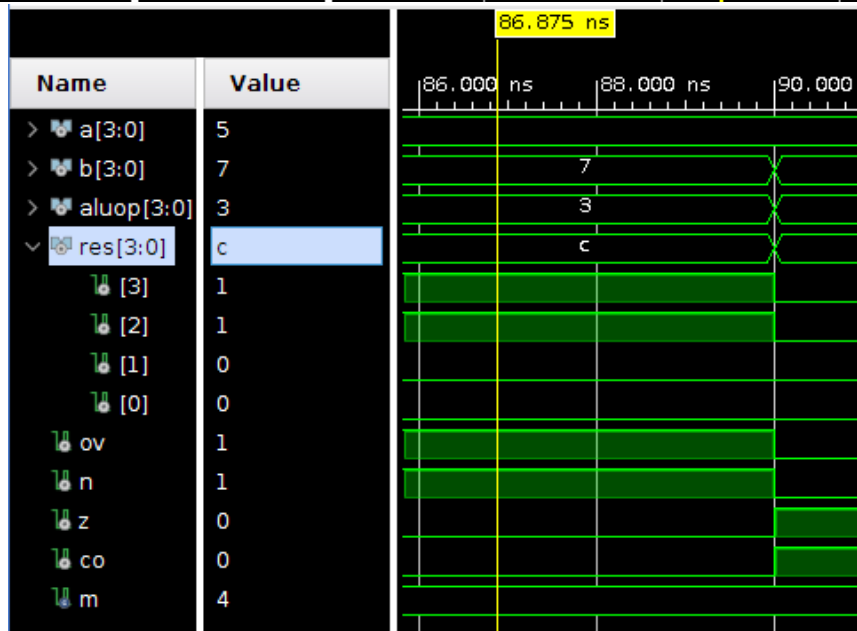
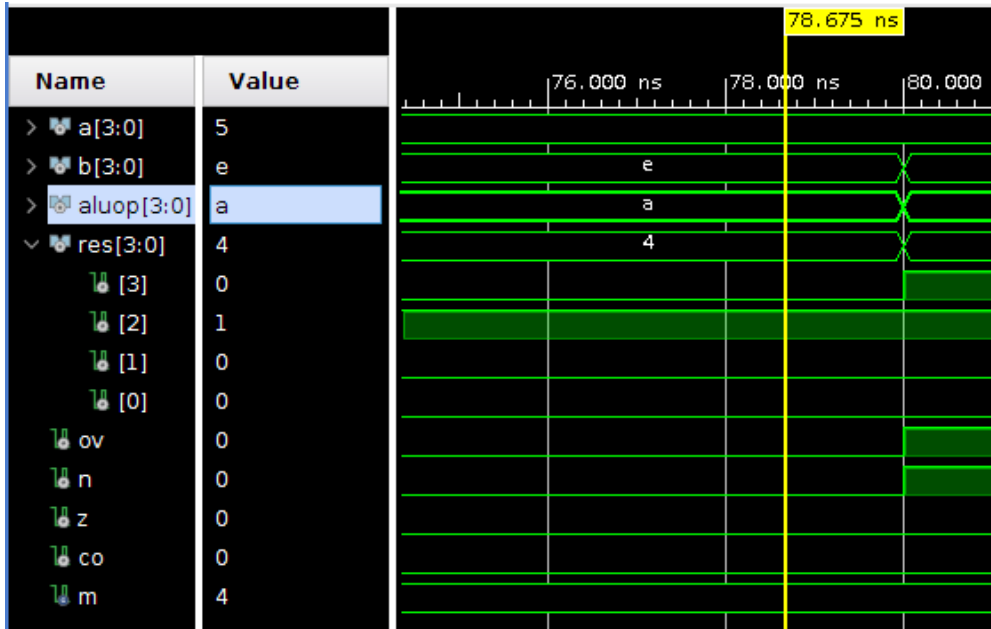


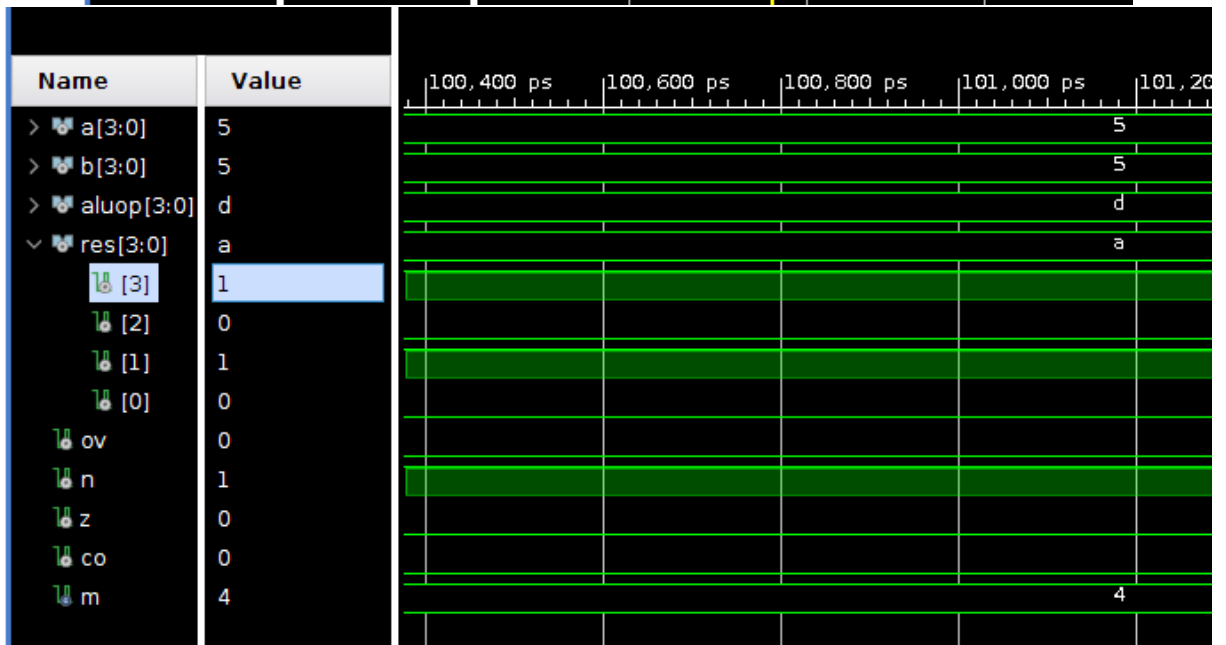
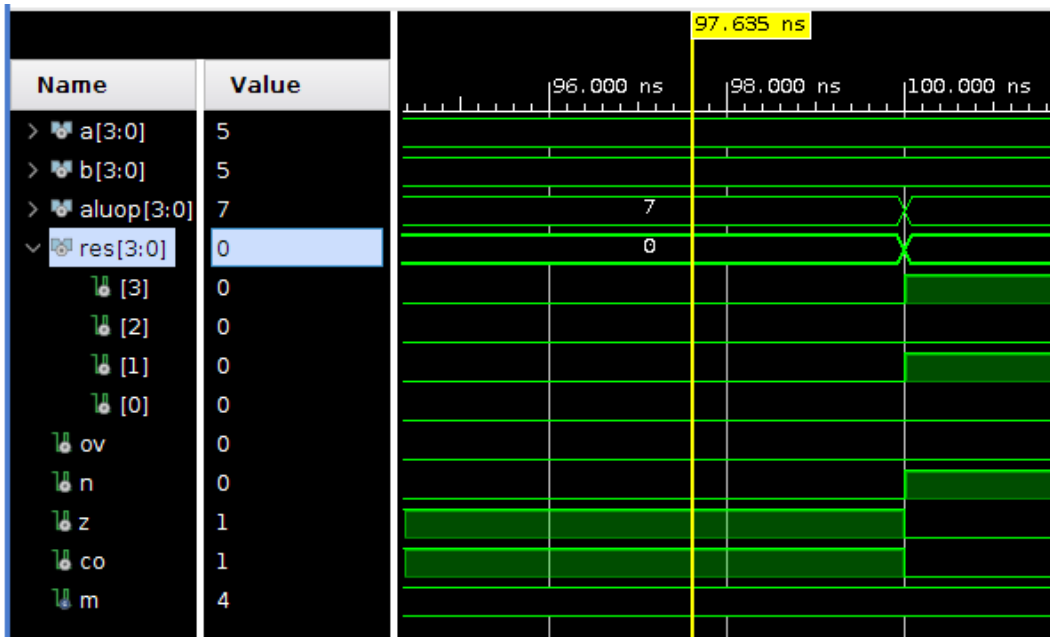










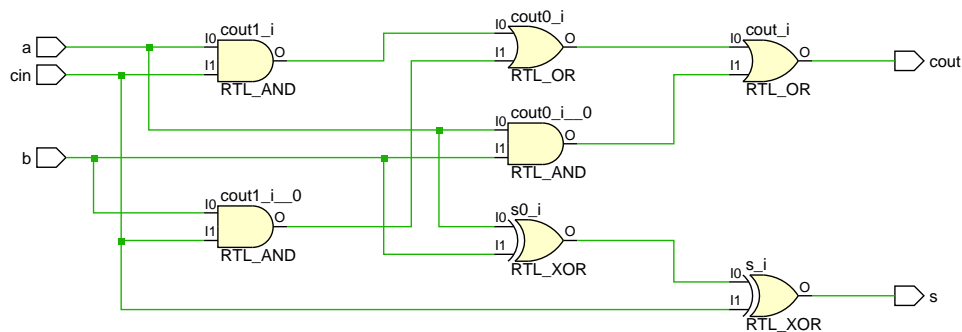


3. Tabla de resultados

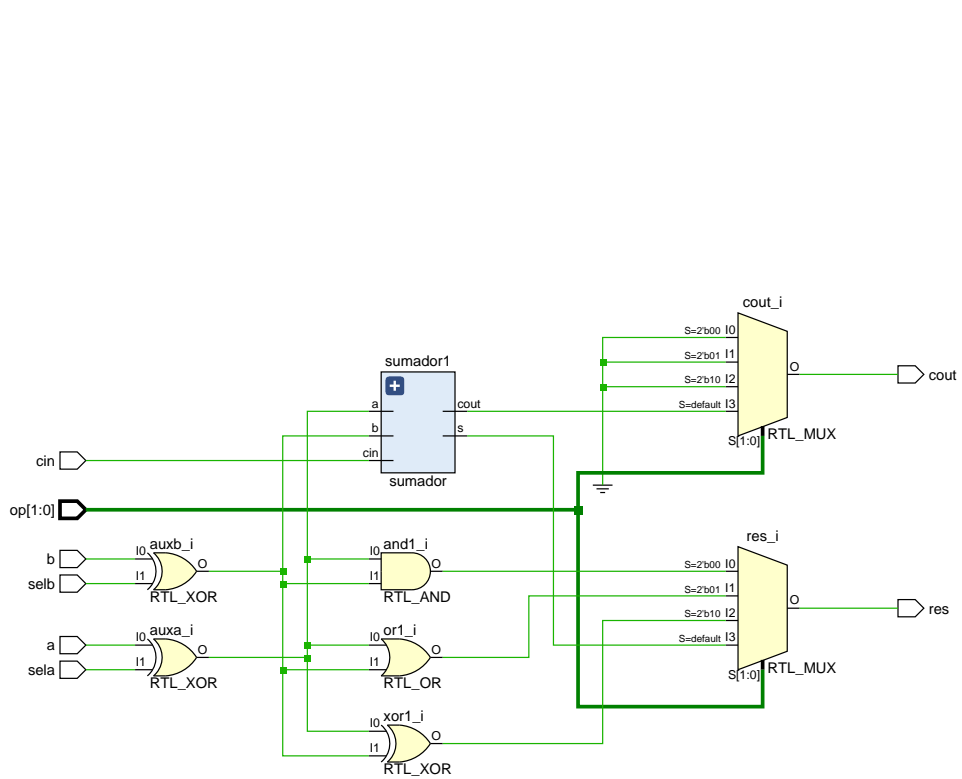
Estado de Banderas				Operación				
				1	0	0	0	Cn
				0	1	0	1	A = 5
OV	N	Z	C	1	1	1	0	B = -2
0	0	0	1	0	0	1	1	A+B
0	0	0	0	0	1	1	1	A-B
0	0	0	0	0	1	0	0	AND
0	1	0	0	1	0	1	1	NAND
0	1	0	0	1	1	1	1	OR
0	0	1	0	0	0	0	0	NOR
0	1	0	0	1	0	1	1	XOR
0	0	0	0	0	1	0	0	XNOR
				1	1	1	0	Cn
OV	N	Z	C	0	1	0	1	A = 5
				0	1	1	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*

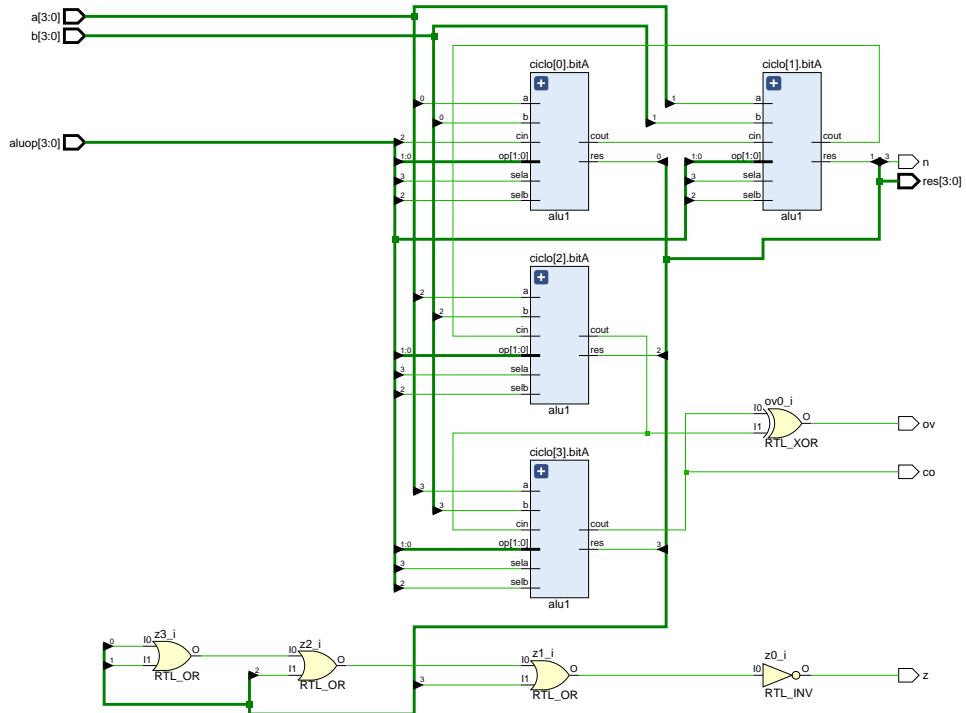


Diagrama RTL del archivo VHDL de la ALU de  $m$  bits