INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

U.A: Arquitectura de Computadoras

“Práctica 6: ALU 4 BITS”

Grupo: 3CV11

Profesor: Vega García Nayeli

Sánchez Becerra Ernesto Daniel

Código Implementación ALU 1 bit

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

entity ALU1 is
    Port ( a,b,aInvert,bInvert,cin : in STD_LOGIC;
           op : in STD_LOGIC_VECTOR (1 downto 0);
           cout,res : out STD_LOGIC);
end ALU1;

architecture Behavioral of ALU1 is
signal auxa, auxb, auxAND, auxOR, aux_XOR, auxSUMA, auxcout: std_logic;
begin
    auxa <= a when aInvert='0' else not a;
    auxb <= b xor bInvert;

    auxAND <= auxa and auxb;
    auxOR <= auxa or auxb;
    aux_XOR <= auxa xor auxb;
    auxSUMA <= auxa xor auxb xor cin;
    auxcout <= (auxa and cin) or  (auxa and auxb) or (auxb and cin);

    res <= auxAND when op="00" else
           auxOR when op="01"  else
           aux_XOR when op="10" else
           auxSUMA;

    cout <= auxcout when op="11" else '0';
end Behavioral;
```

Código package ALU

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

package paqueteALU is
    component ALU1 is
        Port(a,b,aInvert,bInvert,cin: in STD_LOGIC;
             op: in STD_LOGIC_VECTOR(1 downto 0);
             cout,res: out STD_LOGIC);
    end component;
end package;
```

Código Implementación ALU 4 bits

```
library IEEE;
library work;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.std_logic_unsigned.ALL;
use work.paqueteALU.ALL;

entity ALU4b is
    Port ( a,b,aluop : in STD_LOGIC_VECTOR (3 downto 0);
           res : inout STD_LOGIC_VECTOR (3 downto 0);
           cout,bandera_c,bandera_z,bandera_n,bandera_ov : inout STD_LOGIC);
end ALU4b;

architecture Behavioral of ALU4b is
signal c: std_logic_vector(4 downto 0);
begin
    --aluop(3): aInvert, aluop(2):bInvert, aluop(1): op(1), aluop(0):op(0)
    c(0) <= aluop(2);
    ciclo: for i in 0 to 3 generate
        elemento1: ALU1
        Port map(
            a => a(i),
            b => b(i),
            cin => c(i),
            aInvert => aluop(3),
            bInvert => aluop(2),
            op =>aluop(1 downto 0),
            res => res(i),
            cout => c(i+1)
        );
    end generate;
    cout<=c(4);
    bandera_c<=c(4);
    bandera_z<= '1' when res=0 else '0';
    bandera_n<= res(3);
    bandera_ov<= c(4) xor c(3);

end Behavioral;
```

Código de simulación

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.numeric_std.ALL;
entity ALU4_TB is
--  Port ( );
end ALU4_TB;
architecture Behavioral of ALU4_TB is
    component ALU4b
        Port ( a,b,aluop : in STD_LOGIC_VECTOR (3 downto 0);
               res : inout STD_LOGIC_VECTOR (3 downto 0);
               cout,bandera_c,bandera_z,bandera_n,bandera_ov : inout STD_LOGIC);
    end component;
    signal a,b,aluop : STD_LOGIC_VECTOR (3 downto 0);
    signal res : STD_LOGIC_VECTOR (3 downto 0);
    signal cout,bandera_c,bandera_z,bandera_n,bandera_ov : STD_LOGIC;
begin
    practica4: ALU4b Port map(
    a => a,
    b => b,
    aluop => aluop,
    res => res,
    cout => cout,
    bandera_c => bandera_c,
    bandera_z => bandera_z,
    bandera_n => bandera_n,
    bandera_ov => bandera_ov);
stimulus: process
begin
    a<="0101";
    b<="1110";
    aluop<="0011";
    wait for 20ns;

    a<="0101";
    b<="1110";
    aluop<="0111";
    wait for 20ns;

    a<="0101";
    b<="1110";
    aluop<="0100";
    wait for 20ns;

    a<="0101";
    b<="1110";
    aluop<="1101";
    wait for 20ns;

    a<="0101";
    b<="1110";
    aluop<="0001";
    wait for 20ns;

    a<="0101";
    b<="1110";
    aluop<="1100";
    wait for 20ns;

    a<="0101";
    b<="1110";
    aluop<="0010";
    wait for 20ns;

    a<="0101";
    b<="1110";
    aluop<="0100";
    wait for 20ns;

    a<="0101";
    b<="0111";
    aluop<="0011";
    wait for 20ns;

    a<="0101";
    b<="0101";
    aluop<="0111";
    wait for 20ns;

    a<="0101";
    b<="0101";
    aluop<="1101";
    wait for 20ns;
    wait;
 end process;
end Behavioral;
```

Diagrama RTL ALU 1 bit

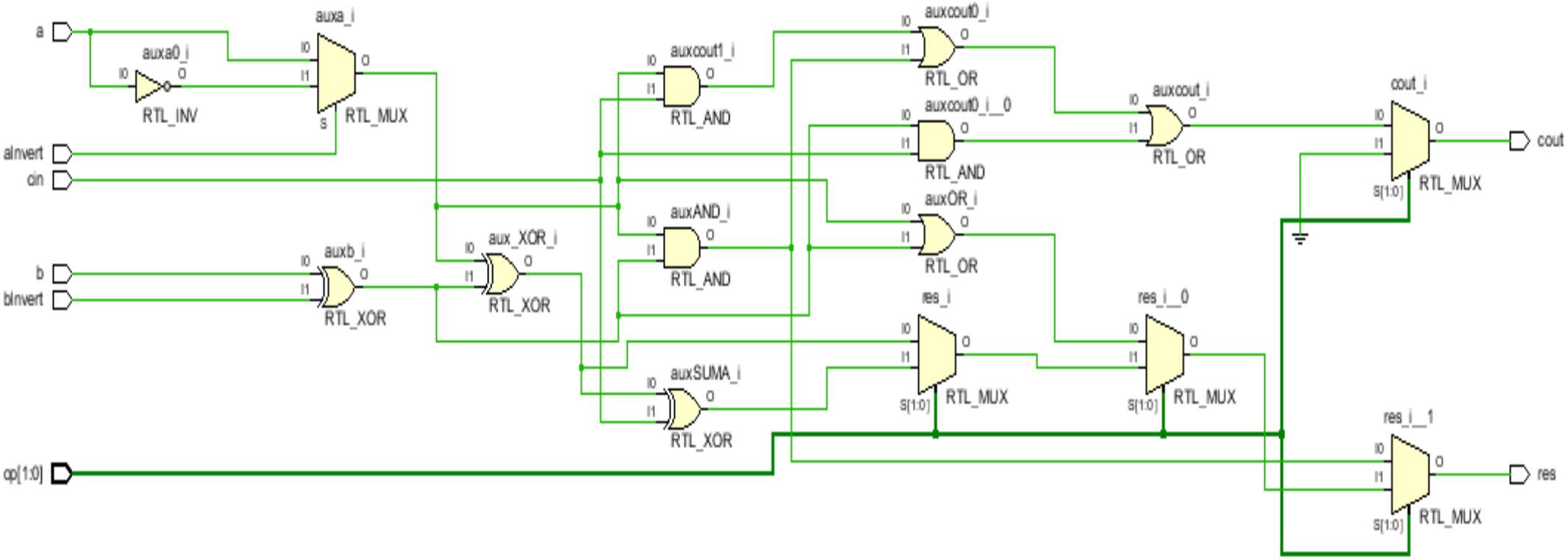
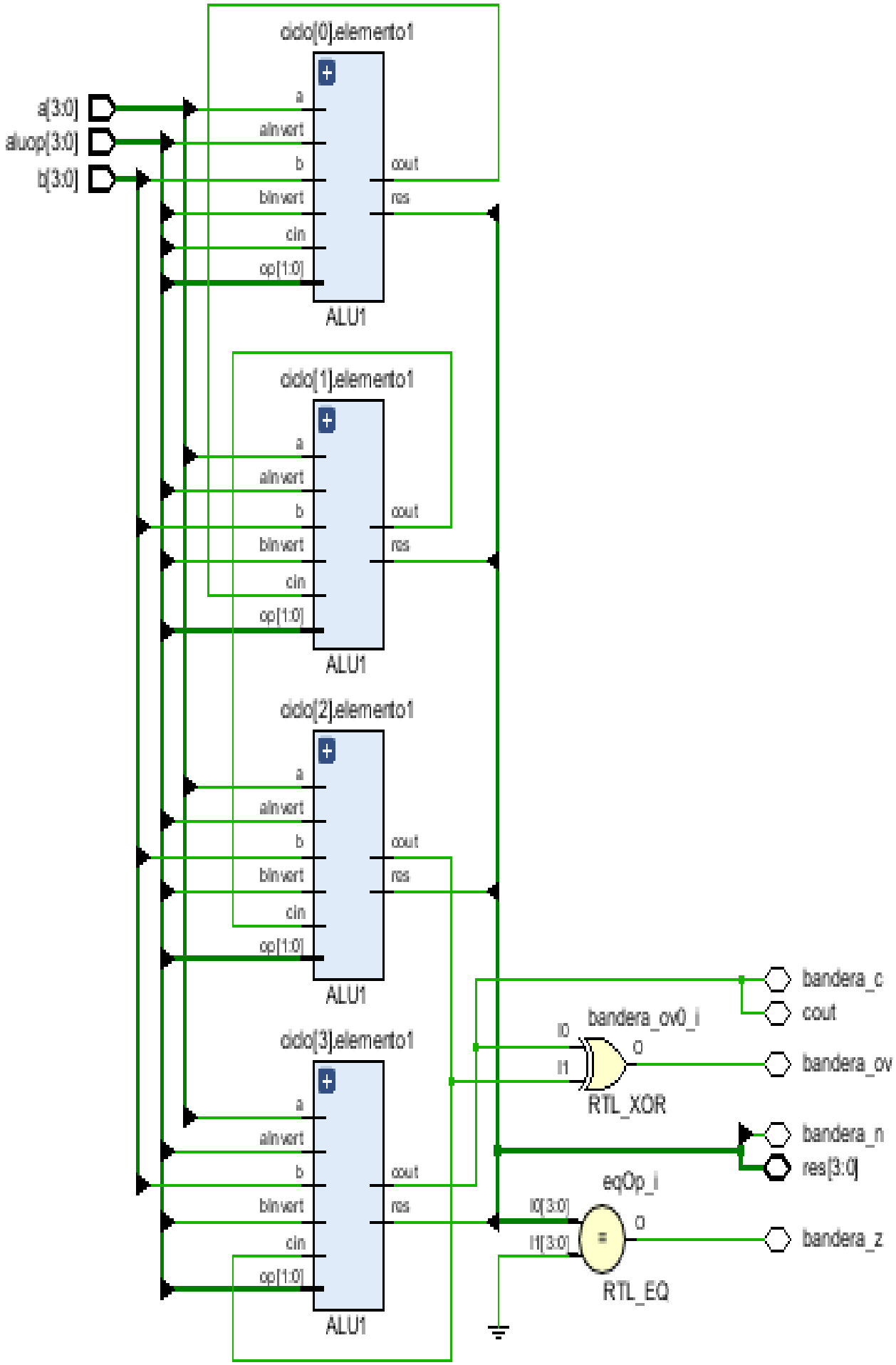


Diagrama RTL ALU 4 bits



Forma de onda

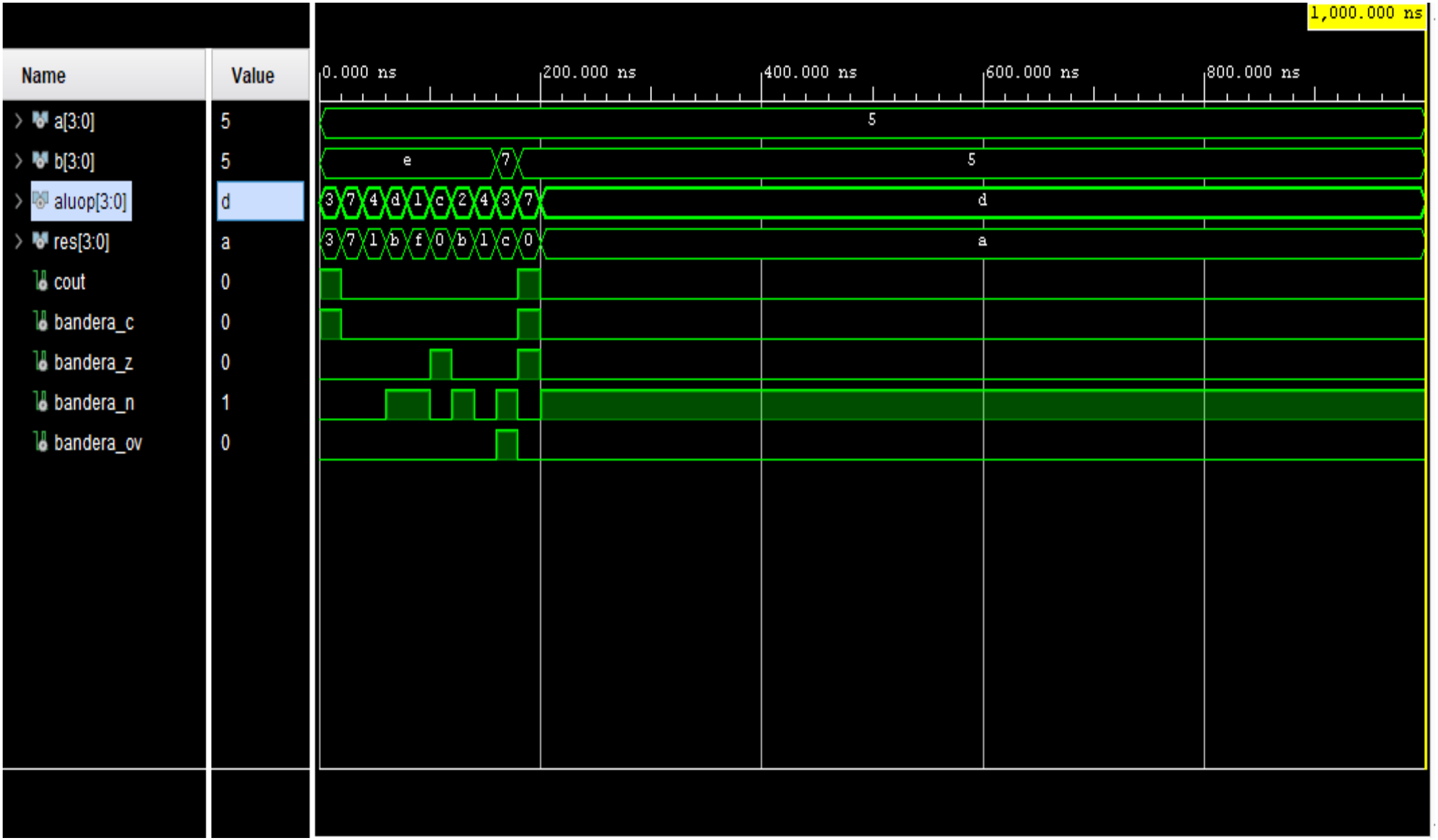


Tabla de resultados:

Estado de banderas				Operación				
				1	0	0	0	Cn
				0	1	0	1	A = 5
Bandera_ov	Bandera_n	Bandera_z	Bandera_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	1	0	1	NAND
0	1	0	0	0	0	0	1	OR
0	0	1	0	1	1	0	0	NOR
0	1	0	0	0	0	1	0	XOR
0	0	0	0	0	1	1	0	XNOR
-								
-				1	1	1	0	Cn
-				0	1	0	1	A = 5
-				0	1	1	1	B = 7
1	1	0	0	0	0	1	1	A + B
-								
-				1	1	1	0	Cn
-				0	1	0	1	A = 5
-				0	1	0	1	B = 5
0	0	1	1	0	1	1	1	A – B
0	1	0	0	1	1	0	1	NAND (NOT)