A) CÓDIGO DE IMPLEMENTACIÓN:

CÓDIGO DE ALU DE 4 BITS:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity ALU4bitss is
 Port ( sela,selb,cin : in STD_LOGIC;
        a,b : in STD_LOGIC_VECTOR(3 downto 0);
        res : out STD_LOGIC_VECTOR(3 downto 0);
        cout,ov,z,N : out STD_LOGIC;
        op : in STD_LOGIC_VECTOR (1 downto 0));
end ALU4bitss;

architecture Behavioral of ALU4bitss is
component ALU is
    Port ( a,b,sela,selb,cin : in STD_LOGIC;
           res : out STD_LOGIC;
           cout: out STD_LOGIC;
           op : in STD_LOGIC_VECTOR (1 downto 0));
end component;
signal inversor : STD_LOGIC_VECTOR(3 DOWNTO 0);
signal cc : STD_LOGIC_VECTOR(4 downto 0);
signal aux : STD_LOGIC;
signal temp : STD_LOGIC_VECTOR(3 downto 0);
begin
cc(0) <= cin;
ciclo : for i in 0 to 3 generate
    inversor(i) <= b(i) xor cin;
    c : ALU port map(
        a => a(i),
        b => b(i),
        cin => cc(i),
        res => temp(i),
        cout => cc(i+1),
        sela => sela,
        selb => selb,
        op => op
    );
res <= temp;
end generate;
cout <= cc(4)AND op(0) AND op(1);
ov <= (cc(3) xor cc(4)) AND op(0) AND op(1);
N <= temp(3);
z <= not(temp(0) or temp (1) or temp (2) or temp(3));
end Behavioral;
```

CÓDIGO DE ALU:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity ALU is
    Port ( a,b,sela,selb,cin : in STD_LOGIC;
           res : out STD_LOGIC;
           cout : out STD_LOGIC;
           op : in STD_LOGIC_VECTOR (1 downto 0));
end ALU;
architecture Behavioral of ALU is
component Sumador1bit is
    Port ( a,b,cin : in STD_LOGIC;
           S, cout : out STD_LOGIC);
end component;
signal auxa, auxb, auxand, auxor, auxXor, suma : STD_LOGIC;
begin
    auxa <= a xor sela;
    auxb <= b when selb = '0' else (not b);
    auxand <= auxa and auxb;
    auxor <= auxa or auxb;
    auxXor <= auxa xor auxb;

    sumador : Sumador1bit
    Port map(
        a => auxa,
        b => auxb,
        cin => cin,
        s => suma,
        cout => cout
    );
    process (auxand, auxor, auxXor, suma, op)
    begin
        case op is
            when "00" => res <= auxand;
            when "01" => res <= auxor;
            when "10" => res <= auxXor;
            when others => res <= suma;
        end case;
    end process;

end Behavioral;
```

CÓDIGO DE SUMADOR:

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

entity Sumador1Bit is
    Port ( a,b,cin : in STD_LOGIC;
           S, cout : out STD_LOGIC);
end Sumador1Bit;

architecture Behavioral of Sumador1Bit is

begin
    S <= a xor b xor cin;
    cout <= (a and b) or (a and cin) or (b and cin);

end Behavioral;
```

B) CÓDIGO DE SIMULACIÓN:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity TB_ALU4bits is
--  Port ( );
end TB_ALU4bits;

architecture Behavioral of TB_ALU4bits is
component ALU4bitss is
 Port ( sela,selb,cin : in STD_LOGIC;
          a,b : in STD_LOGIC_VECTOR(3 downto 0);
          res : out STD_LOGIC_VECTOR(3 downto 0);
          cout,ov,z,N : out STD_LOGIC;
          op : in STD_LOGIC_VECTOR (1 downto 0));
end component;
SIGNAL sela,selb,cin : STD_LOGIC;
SIGNAL a,b : STD_LOGIC_VECTOR(3 downto 0);
SIGNAL res : STD_LOGIC_VECTOR(3 downto 0);
SIGNAL cout,ov,z,N : STD_LOGIC;
SIGNAL op : STD_LOGIC_VECTOR (1 downto 0);
begin
u1 : ALU4bitss
   Port Map(
   a => a,
   b => b,
   cin => cin,
   res => res,
   sela => sela,
   selb => selb,
   cout => cout,
   op => op,
   ov => ov,
   z => z,
   N => N
   );
   process
      begin
      a <= "0101";
      b <= "0101";
      sela <= '1';
      selb <= '1';
      cin <= '1';
      op <= "01";
      wait;
   end process;
end Behavioral;
```

C) DIAGRAMAS RTL:

DIAGRAMA ALU DE 4 BITS:

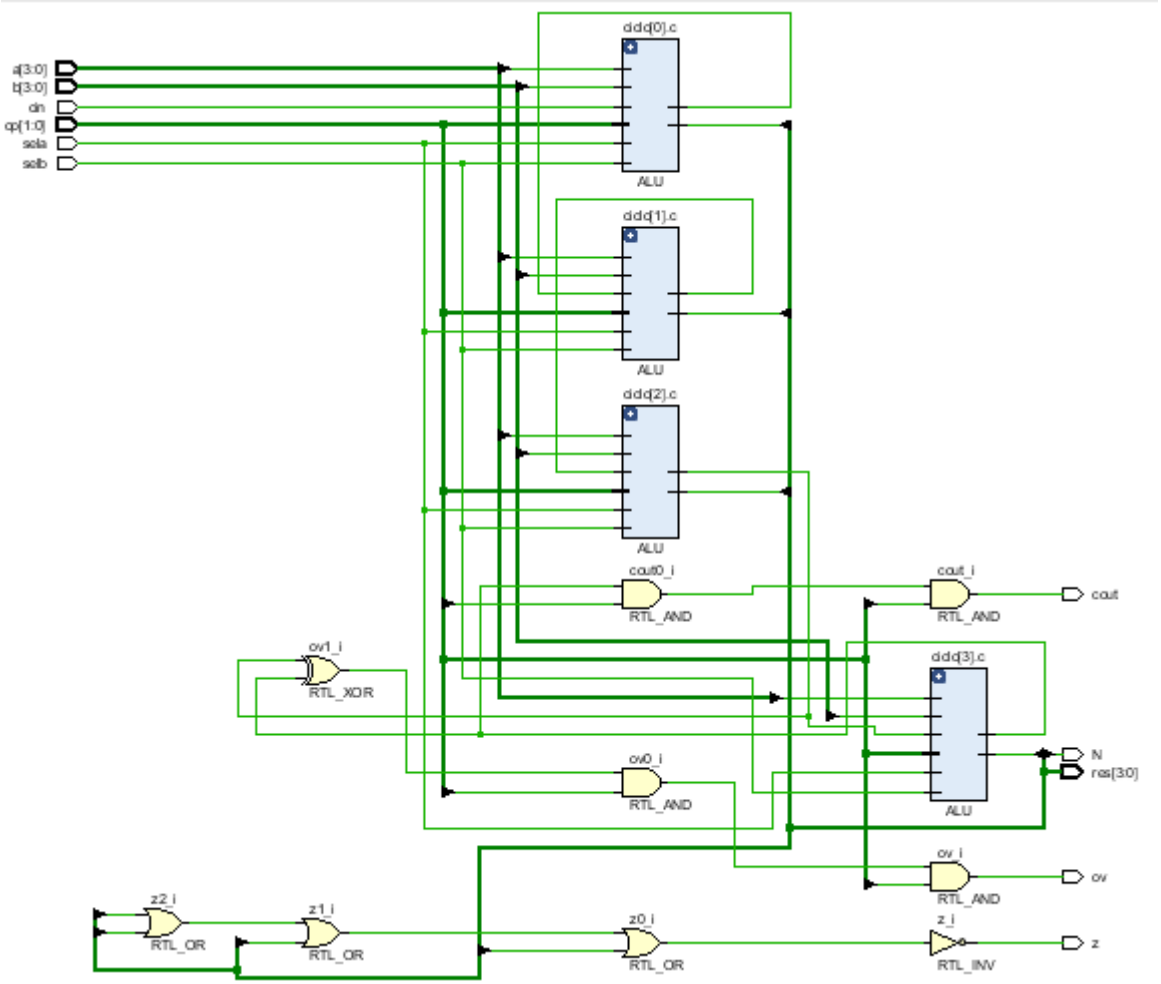


DIAGRAMA ALU 1 BIT:

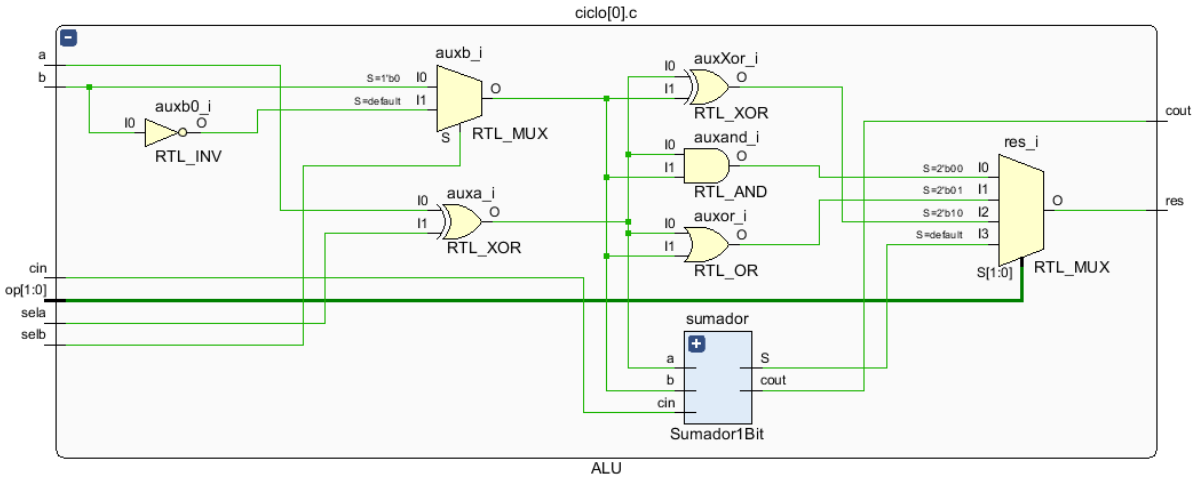
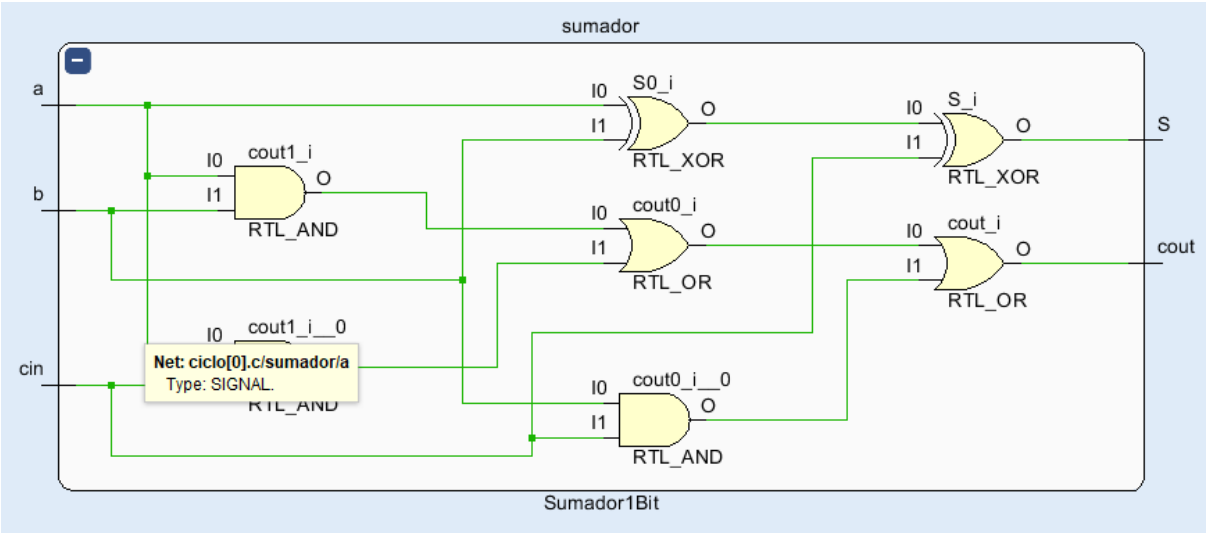


DIAGRAMA SUMADOR:

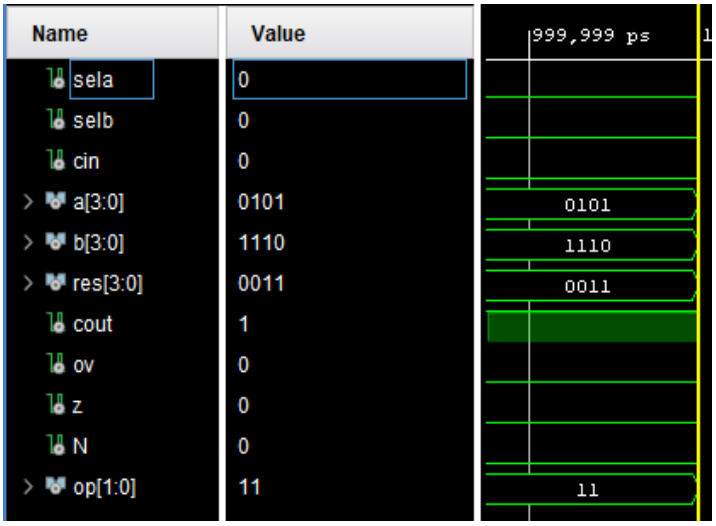


D) Simulaciones:

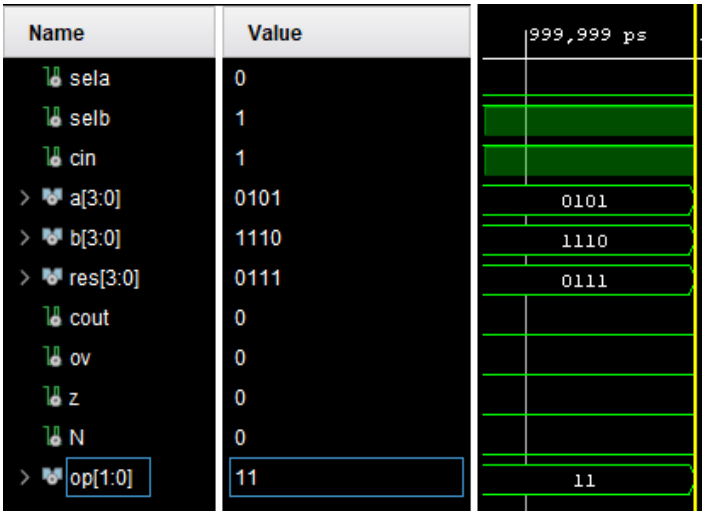
A=0101

B=1110

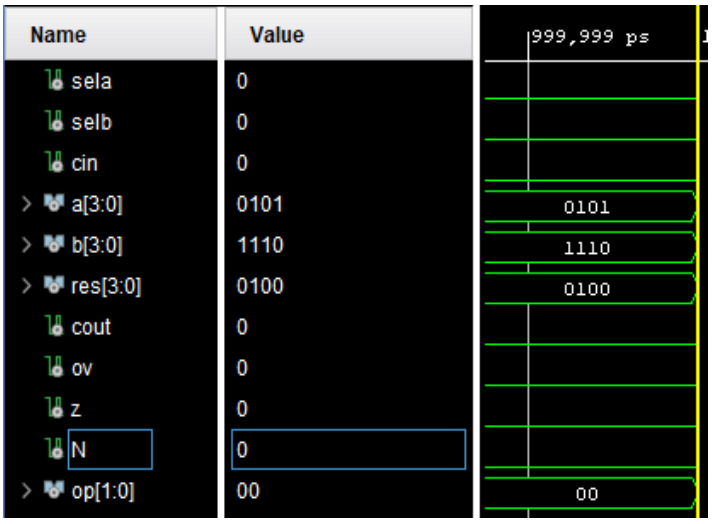
A+B:



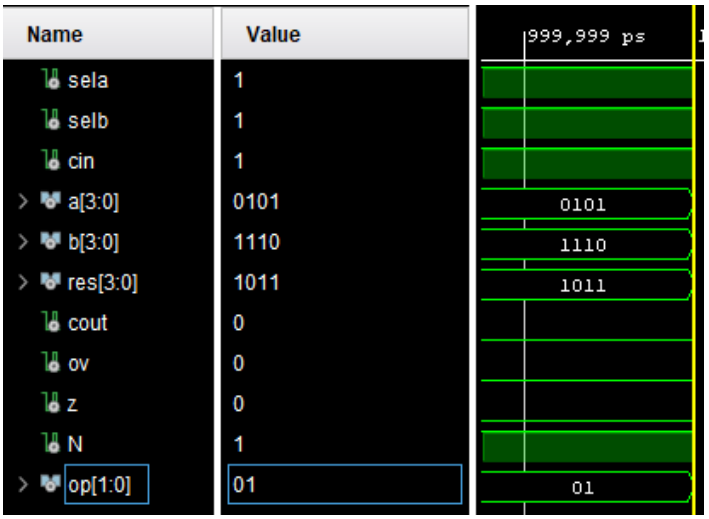
A-B:



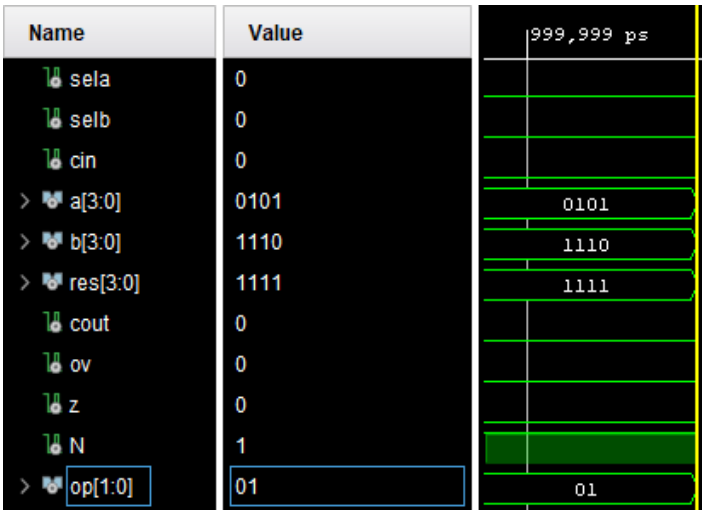
AND:



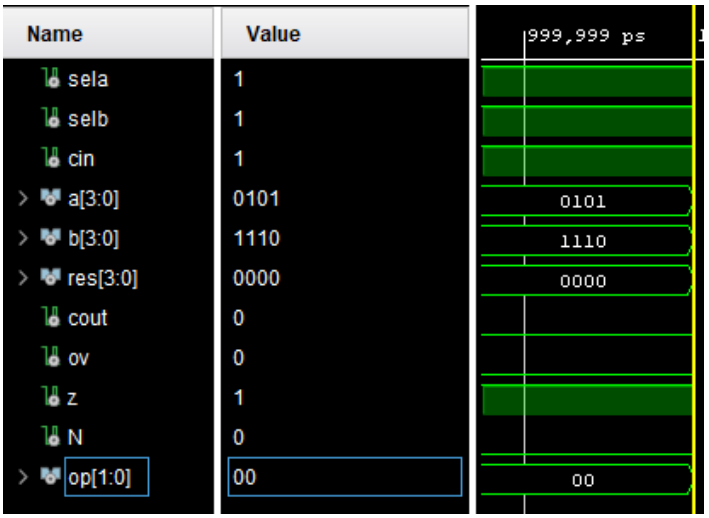
NAND:


Name
Value
999,999 ps
sela
1
selb
1
cin
1
a[3:0]
0101
0101
b[3:0]
1110
1110
res[3:0]
1011
1011
cout
0
ov
0
z
0
N
1
op[1:0]
01
01

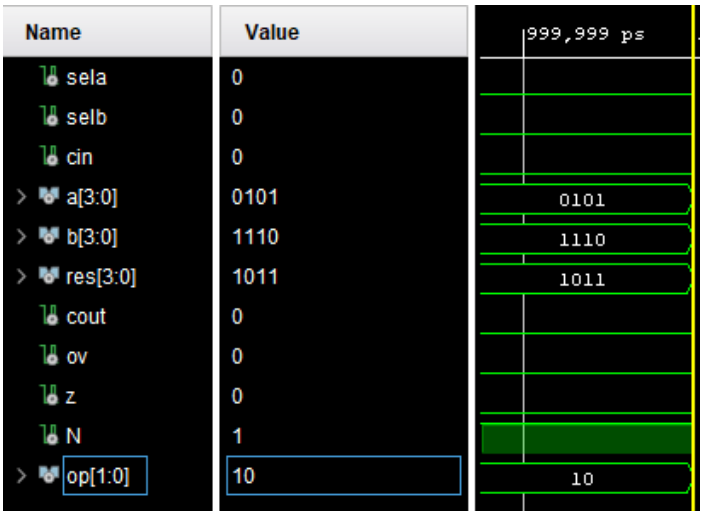
OR:


Name
Value
999,999 ps
sela
0
selb
0
cin
0
a[3:0]
0101
0101
b[3:0]
1110
1110
res[3:0]
1111
1111
cout
0
ov
0
z
0
N
1
op[1:0]
01
01

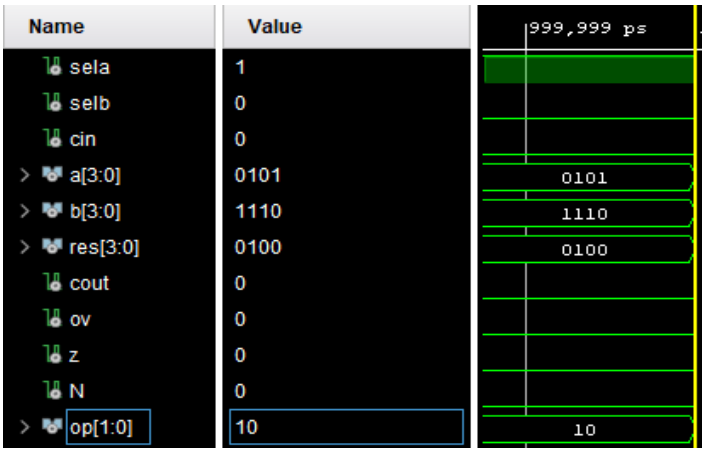
NOR:


Name
Value
999,999 ps
sela
1
selb
1
cin
1
a[3:0]
0101
0101
b[3:0]
1110
1110
res[3:0]
0000
0000
cout
0
ov
0
z
1
N
0
op[1:0]
00
00

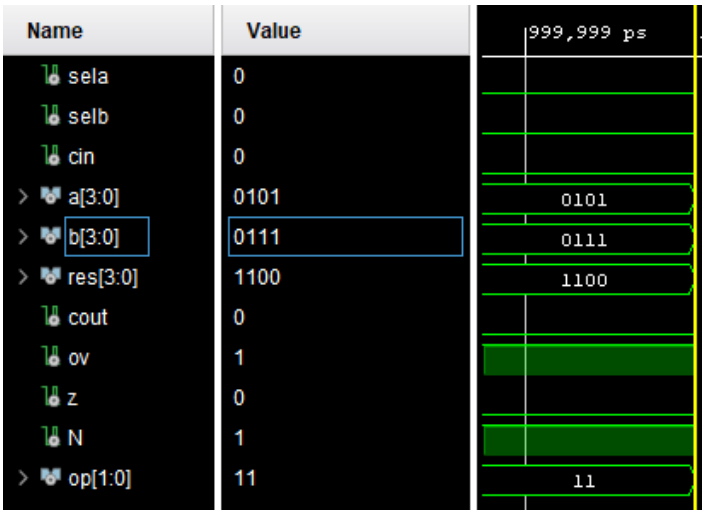
XOR:



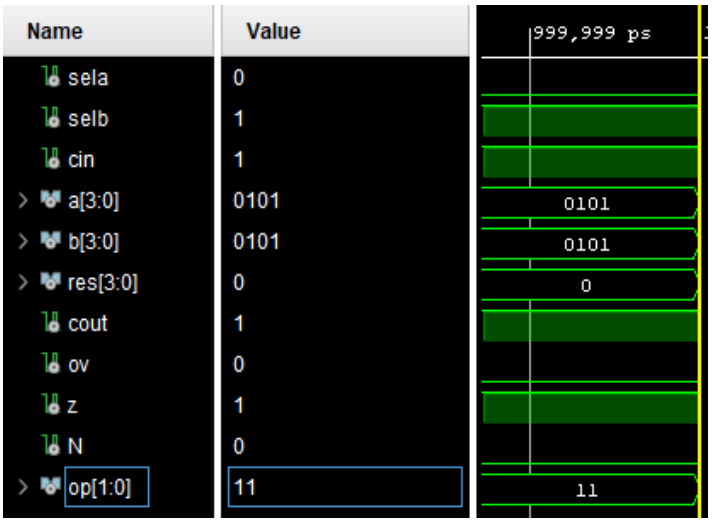
XNOR:



A=0101
B=0111
A+B:



A=0101
B=0101
A-B:



NAND: