Código VHDL PRACTICA 03.
Componente ALU 1 BIT

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity ALU_1BIT is
   Port ( a, b, sel_a, sel_b, cin : in  STD_LOGIC;
          operacion : in  STD_LOGIC_VECTOR (1 downto 0);
          cout, resul : out  STD_LOGIC);
end ALU_1BIT;

architecture Behavioral of ALU_1BIT is

signal a_aux, b_aux, ab, aorb, axorb, amasb, cout_aux, resul_aux : STD_LOGIC;

begin

process(operacion, sel_a, sel_b, a, b, cin, a_aux, b_aux, ab, aorb, axorb, amasb)
begin

        a_aux <= a xor sel_a;
        b_aux <= b xor sel_b;

        ab <= a_aux and b_aux;
        aorb <= a_aux or b_aux;
        axorb <= a_aux xor b_aux;
        amasb <= (a_aux xor b_aux) xor cin;
        cout_aux <= (b_aux and cin) or (a_aux and cin) or (a_aux and b_aux);

        case operacion is

                when "00" => resul_aux <= ab;
                when "01" => resul_aux <= aorb;
                when "10" => resul_aux <= axorb;
                when others => resul_aux <= amasb;

        end case;

end process;

cout <= cout_aux;
resul <= resul_aux;

end Behavioral;
```

Código VHDL PRACTICA 03.
CONJUNTO

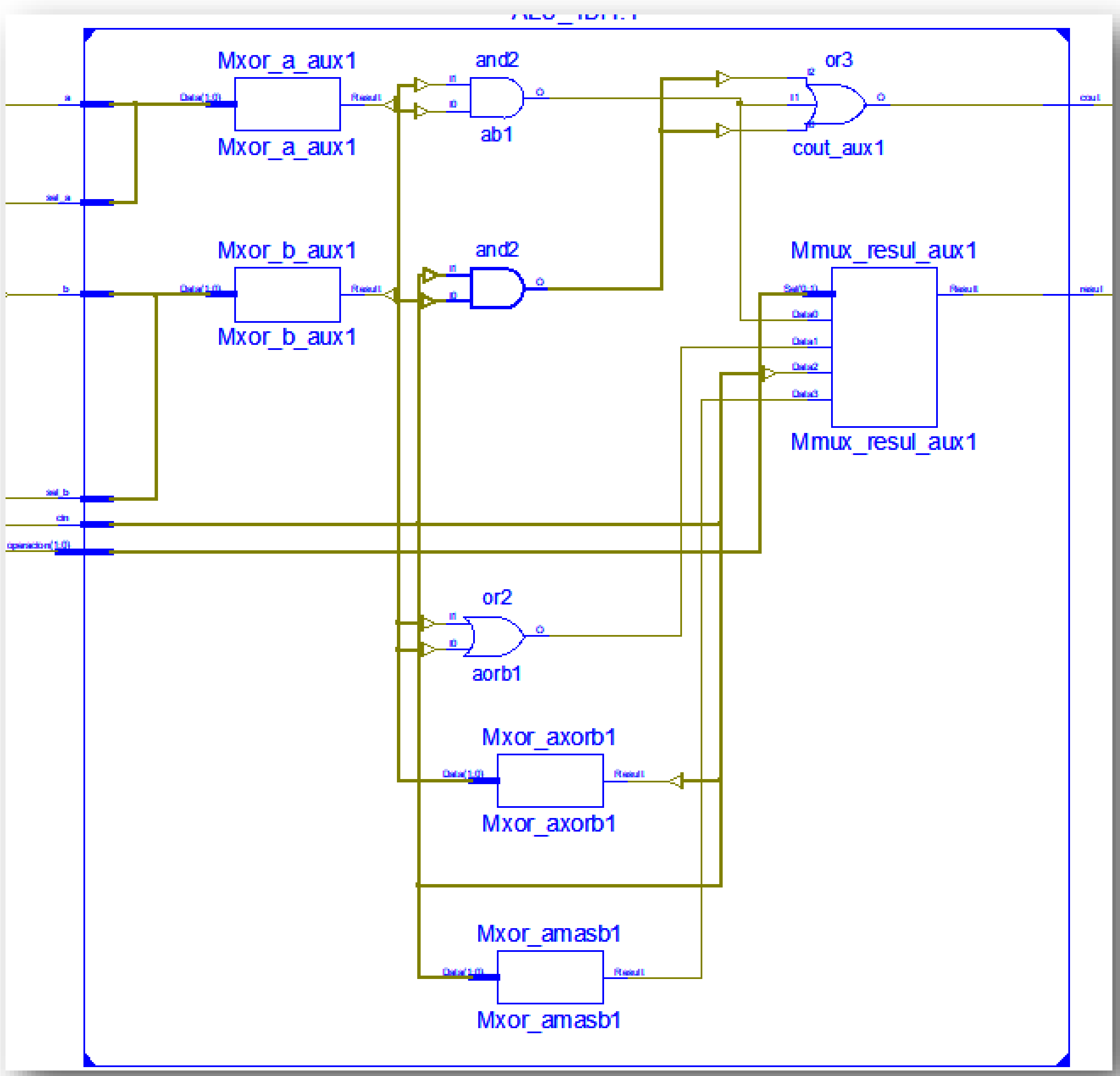
```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity PRACTICA03 is
    Port ( A,B : in  STD_LOGIC_VECTOR (3 downto 0);
           operacion : in  STD_LOGIC_VECTOR (3 downto 0);
           Z, Cout, Ovf, N : out  STD_LOGIC;
           R : out  STD_LOGIC_VECTOR (3 downto 0));
end PRACTICA03;

-- N -> Signo, Cout-> C, Z -> Cero, Ovf -> Overflow
-- CarryOut y Overflow, valen 0 cuando son operaciones lógicas
architecture Behavioral of PRACTICA03 is

component ALU_1BIT is
    Port ( a, b, sel_a, sel_b, cin : in  STD_LOGIC;
           operacion : in  STD_LOGIC_VECTOR (1 downto 0);
           cout, resul : out  STD_LOGIC);
end component;

signal sel_aAux, sel_bAux : std_logic;
signal carries, R_aux : STD_LOGIC_VECTOR(3 downto 0);
signal oper_aux : STD_LOGIC_VECTOR(1 downto 0);

begin
        sel_aAux<=operacion(3);
        sel_bAux<=operacion(2);
        oper_aux<=operacion(1 downto 0);
        ALU01: ALU_1BIT port map(a=>A(0), b=>B(0), sel_a=>sel_aAux, sel_b=>sel_bAux, cin=>sel_bAux,
operacion=>oper_aux, cout=>carries(0), resul=>R_aux(0));
        ALU02: ALU_1BIT port map(a=>A(1), b=>B(1), sel_a=>sel_aAux, sel_b=>sel_bAux, cin=>carries(0),
operacion=>oper_aux, cout=>carries(1), resul=>R_aux(1));
        ALU03: ALU_1BIT port map(a=>A(2), b=>B(2), sel_a=>sel_aAux, sel_b=>sel_bAux, cin=>carries(1),
operacion=>oper_aux, cout=>carries(2), resul=>R_aux(2));
        ALU04: ALU_1BIT port map(a=>A(3), b=>B(3), sel_a=>sel_aAux, sel_b=>sel_bAux, cin=>carries(2),
operacion=>oper_aux, cout=>carries(3), resul=>R_aux(3));

        Z<= '1' when R_aux = "0000" else '0';


        R<=R_aux;

        N<=R_aux(3);

        Cout<= carries(3) when oper_aux = "11" else '0';
        Ovf<= (carries(3) xor carries(2)) when oper_aux = "11" else '0';

end Behavioral;
```

Código VHDL PRACTICA 03.
TEST BENCH— PRACTICA CONJUNTA

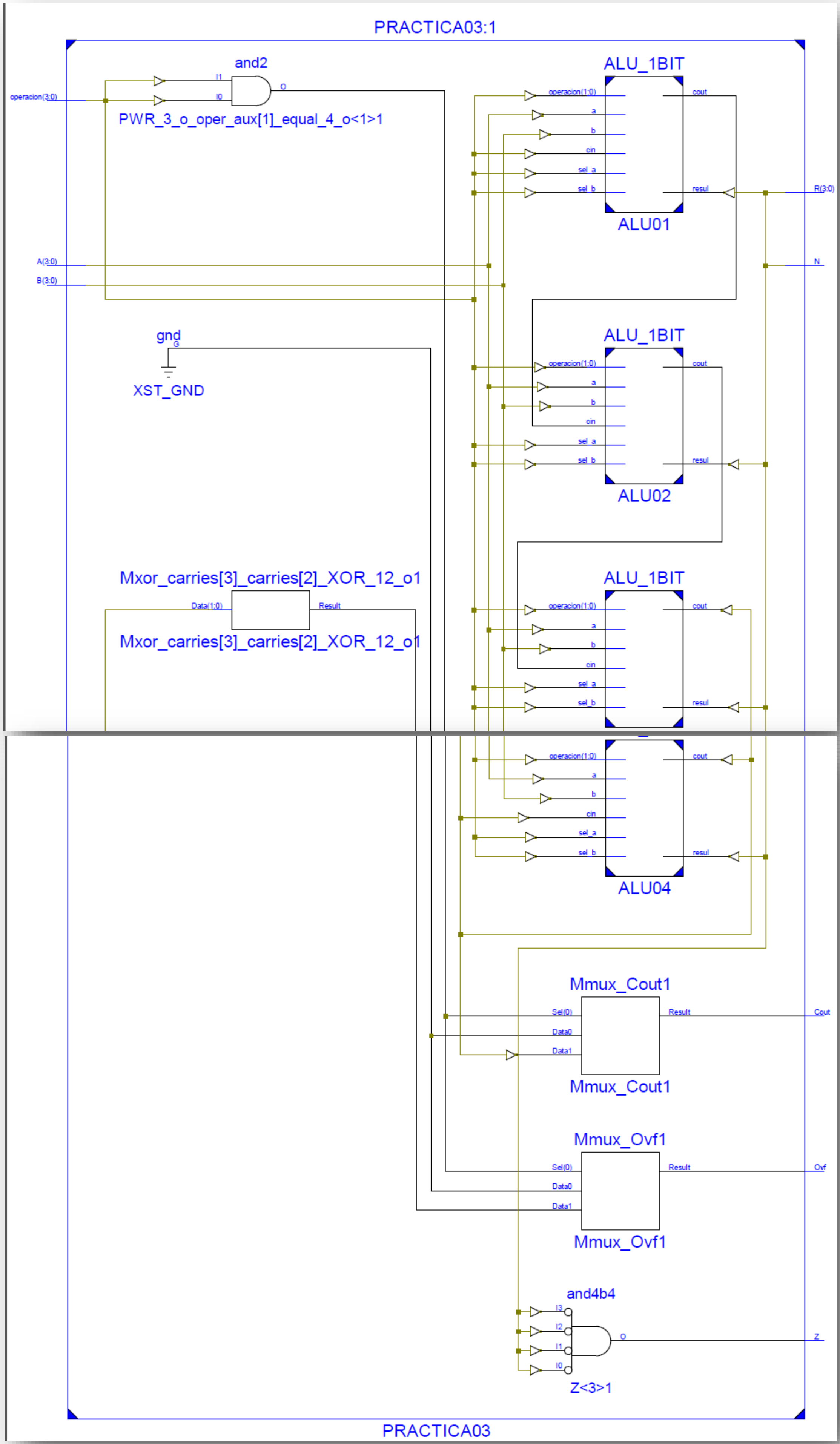
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY PRACTICA03_TB IS
END PRACTICA03_TB;

ARCHITECTURE behavior OF PRACTICA03_TB IS

    COMPONENT PRACTICA03
    PORT(
         A : IN  std_logic_vector(3 downto 0);
         B : IN  std_logic_vector(3 downto 0);
         operacion : IN  std_logic_vector(3 downto 0);
         Z : OUT  std_logic;
         Cout : OUT  std_logic;
         Ovf : OUT  std_logic;
         N : OUT  std_logic;
         R : OUT  std_logic_vector(3 downto 0)
        );
    END COMPONENT;

   --Inputs
   signal A : std_logic_vector(3 downto 0) := (others => '0');
   signal B : std_logic_vector(3 downto 0) := (others => '0');
   signal operacion : std_logic_vector(3 downto 0) := (others
=> '0');

         --Outputs
   signal Z : std_logic;
   signal Cout : std_logic;
   signal Ovf : std_logic;
   signal N : std_logic;
   signal R : std_logic_vector(3 downto 0);

BEGIN
         -- Instantiate the Unit Under Test (UUT)
   uut: PRACTICA03 PORT MAP (
          A => A,
          B => B,
          operacion => operacion,
          Z => Z,
          Cout => Cout,
          Ovf => Ovf,
          N => N,
          R => R
        );
   -- Stimulus process
   stim_proc: process
   begin
                  A<="0000";
                  B<="0000";      wait for 100 ns;

                  wait for 100 ns;
                     A<="0101";
                     B<="1110";
                     -- operacion = selA&selB&oper
                     operacion<="0011"; -- SUMA

                     wait for 100 ns;
                     operacion<="0111"; -- RESTA

                     wait for 100 ns;
                     operacion<="0000"; -- AND

                     wait for 100 ns;
                     operacion<="1101"; -- NAND

                     wait for 100 ns;
                     operacion<="0001"; -- OR

                     wait for 100 ns;
                     operacion<="1100"; -- NOR

                     wait for 100 ns;
                     operacion<="0010"; -- XOR

                     wait for 100 ns;
                     operacion<="1010"; --XNOR

                     wait for 100 ns;
                     A<="0101";
                     B<="0111";
                     operacion<="0011"; -- SUMA

                     wait for 100 ns;
                     A<="0101";
                     B<="0101";
                     operacion<="0111"; -- RESTA

                     wait for 100 ns;
                     operacion<="1101"; --NAND(NOT)

      -- insert stimulus here


      wait;
   end process;

END;
```

Código VHDL PRACTICA 03.

DIAGRAMA RTL -> COMPONENTE ALU



Código VHDL PRACTICA 03.
DIAGRAMA RTL. FINAL


PRACTICA03:1
and2
PWR_3_o_oper_aux[1]_equal_4_o<1>1
operacion(3:0)
A(3:0)
B(3:0)
gnd
XST_GND
Mxor_carries[3]_carries[2]_XOR_12_o1
Data(1:0)
Result
ALU_1BIT
ALU01
ALU02
ALU04
operacion(1:0)
a
b
cin
sel_a
sel_b
cout
resul
R(3:0)
N
Mmux_Cout1
Sel(0)
Data0
Data1
Cout
Mmux_Ovf1
Ovf
and4b4
Z<3>1
Z
PRACTICA03