Bilkent University
EE102-02 Lab 4 Report:
Arithmetic Logic Unit

Mehmet Emre Uncu - 22003884

Purpose:

This lab's goal was to create an arithmetic logic unit (ALU) using VHDL to do at least one bitwise operation and one shift operation. The mathematical functions are addition, subtraction, comparator, left logical shifter, one’s complement, AND, NOR, XNOR gates. The VHDL code is organized into modules. The code would be implemented on the BASYS3 at the end of the lab.

Methodology:

Initially, eight different functions were determined for the three-bit system. First, a half adder was made. Then a full adder was created using this half adder. Eight different functions were designed using these two components or logic gates. At the end of each design, the design was checked and visualized by creating the RLT schematic. The waveform was created with a testbench, and the design was verified. The constraint file is created, and the bitstream is generated. Finally, the design was simulated with switches and LEDs on BASYS3.

Design Specifications:

In the ALU, there are 7 inputs and 1 output. First, the desired function is selected with the selection(m) switches in the three-bit system. Then, x1, x2, x3 and y1, y2, y3 three-bit numbers are determined by switches. The four-bit output is projected via LEDs.

The VHDL code is organized into modules. The top-level module, "top.vhdl" outputs the outcomes of the specified operation and toggles between user options.

There are ten sub-modules under the top module:

- halfadder.vhdl
- fulladder.vhdl
- addition.vhdl
- substraction.vhdl
- comparator.vhdl
- left_logical_shifter.vhdl
- ones_complement.vhdl
- andgate.vhdl
- norgate.vhdl
- xnorgate.vhdl

Arithmetic Logic Unit (ALU) implemented using VHDL. The finalized RTL schematics can be seen in Figure A. RTL schematics of components used in combinational circuits can also be seen in Figure (1-10).

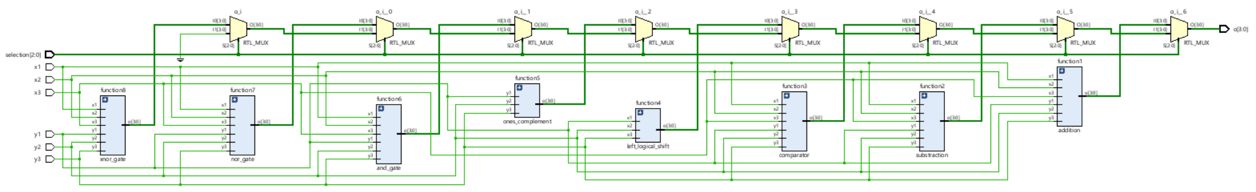
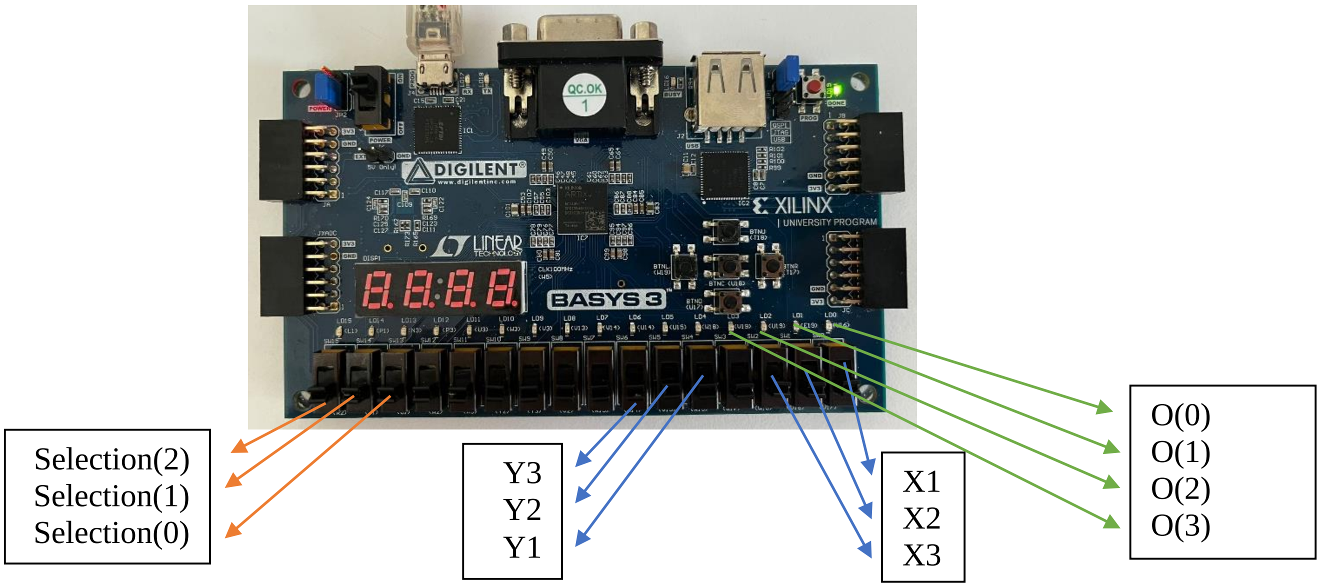


Figure i: RTL Schematic of ALU

BASYS3 FPGA was programmed via a constraints file for implementation according to the following assignments:

x1: V17 switch	selection(0): U1 switch	o(0): U16 LED
x2: V16 switch	selection(1): T1 switch	o(1): E19 LED
x3: W16 switch	selection(2): R2 switch	o(2): U19 LED
y1: W15 switch		o(3): V19 LED
y2: V15 switch		
y3: W14 switch		

Table 1: Input and Output Assignments



Results:

The testbench was written for simulation, and the waveform was created with three different input pairs (“111” and “111”, “111” and “000”, “101” and “010”). Verified that all functions work correctly when the results are checked, as seen in Figure B.

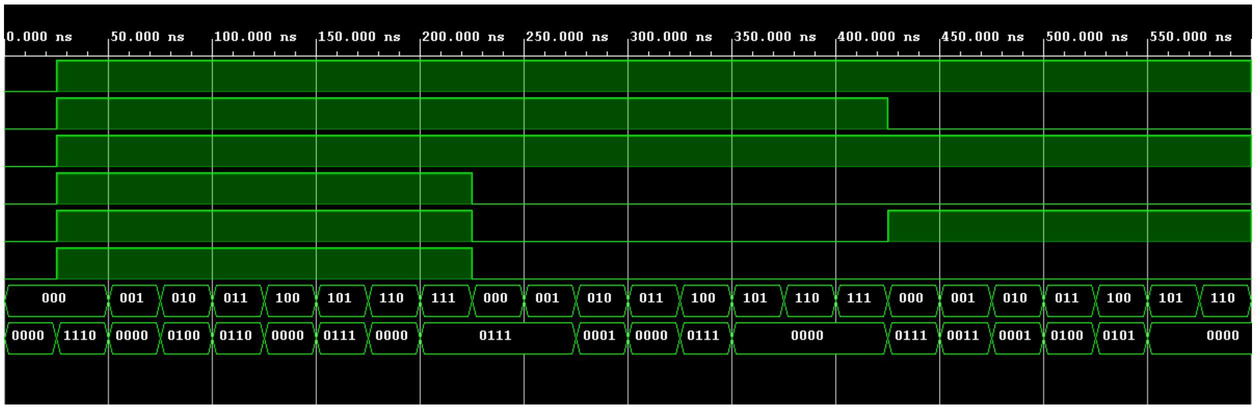


Figure ii: The Waveform of ALU

Comparing the expected output with the output on the FPGA after entering random inputs into each function to verify that the implementation is working correctly.

i. ADDITION

Selection: “000”

Input 1: “100”

Input 2: “111”

Expected: “1011”

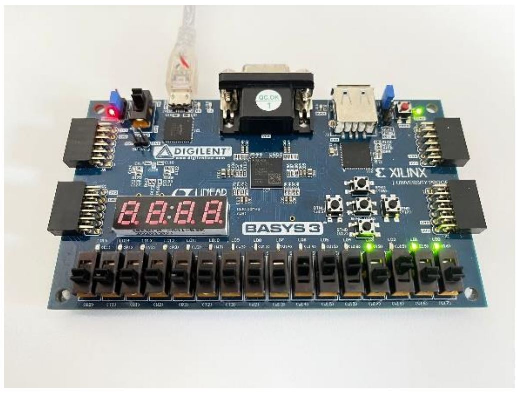


Figure i addition

ii. SUBTRACTION

Selection: “001”

Input 1: “010”

Input 2: “101”

Expected: “0101”

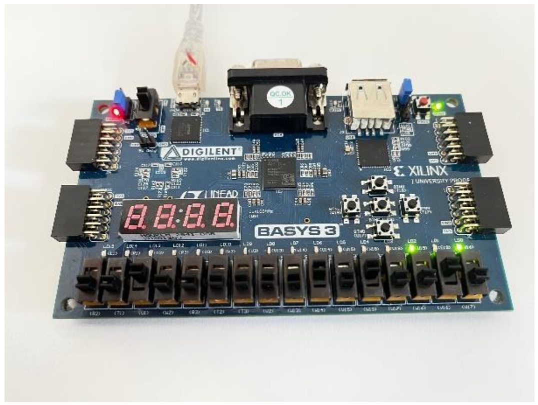


Figure ii subtraction

iii. COMPARATOR

Selection: “010”

Input 1: “111”

Input 2: “111”

Expected: “0100”

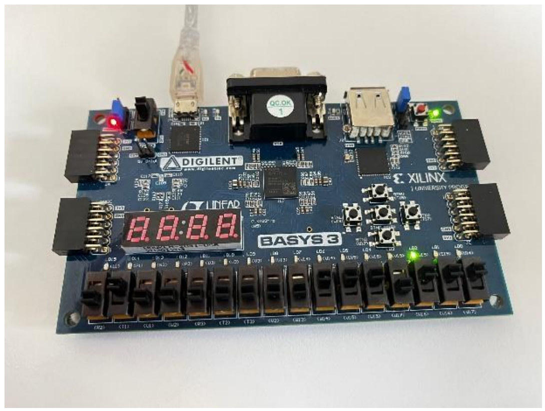


Figure iii comparator

iv. LEFT_LOGICAL_SHIFT

Selection: “011”

Input 2: “101”

Expected: “0010”

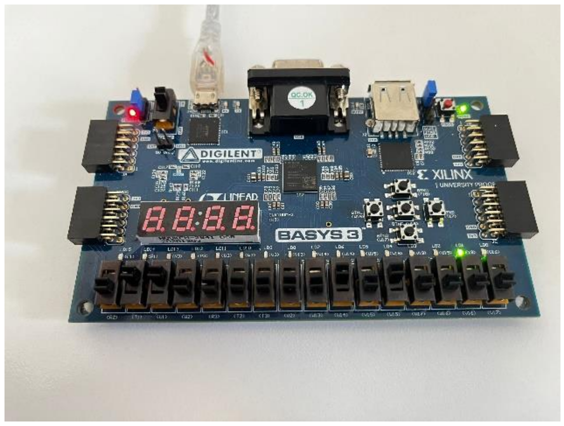


Figure iv left logic shift

v. ONES_COMPLEMENT

Selection: “100”

Input 2: “111”

Expected: “0000”

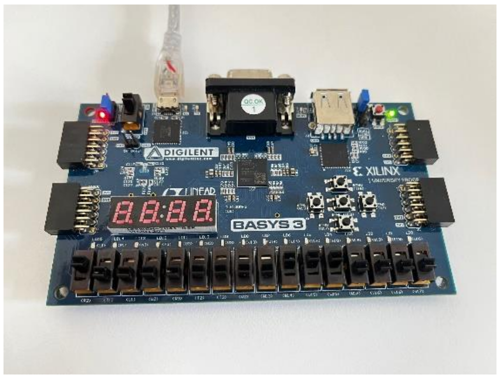


Figure v ones complement

vi. AND_GATE

Selection: “101”

Input 1: “111”

Input 2: “110”

Expected: “0110”

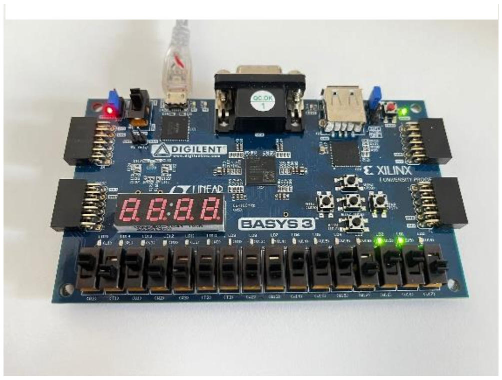


Figure vi and gate

vii. NOR_GATE

Selection: “110”

Input 1: “000”

Input 2: “000”

Expected: “0111”

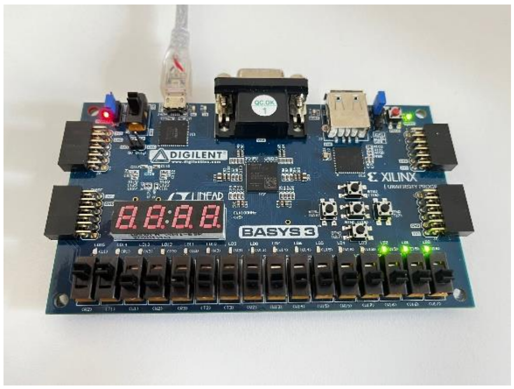


Figure vii nor gate

viii. XNOR_GATE

Selection: “111”

Input 1: “001”

Input 2: “011”

Expected: “0101”

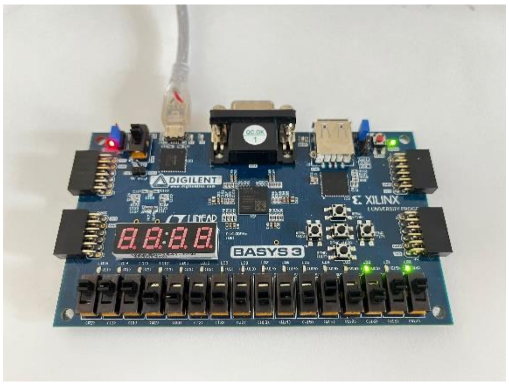


Figure viii xnor gate

Conclusion:

Using VHDL and Basys3, an Arithmetic Logic Unit was constructed in this lab. Three 3-bit variables are inputted into the design, and eight different functions are used to generate a 4-bit output. The VHDL code is organized into modules. The half adder made with logic gates was used to construct the full adder. So how to use a function as a module within another function was learned. Then eight different functions were designed using these two sub-modules or logic gates. The designs were checked with the help of RTL schematics and waveforms. Finally, the design was implemented on the FPGA. The outputs are compared with the simulation's waveform and the equation's expected results so the design is verified. No errors were encountered in this lab, except for human errors that occurred during circuit design. These errors were fixed by schematizing and simulating with the Vivado. Thanks to all this, it was learned to build ALU and use modules/sub-modules.

Appendix:

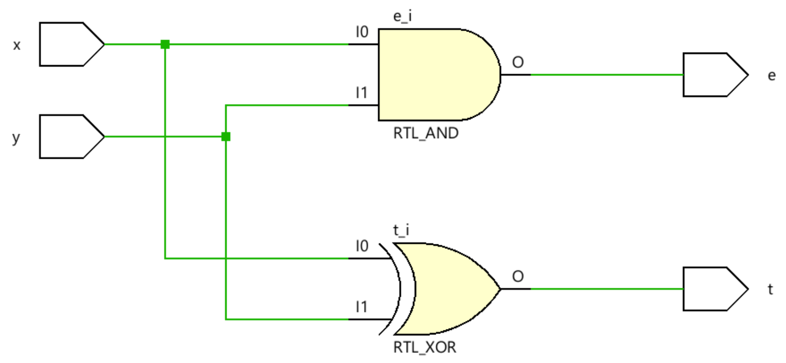


Figure iii: Half Adder Schematic

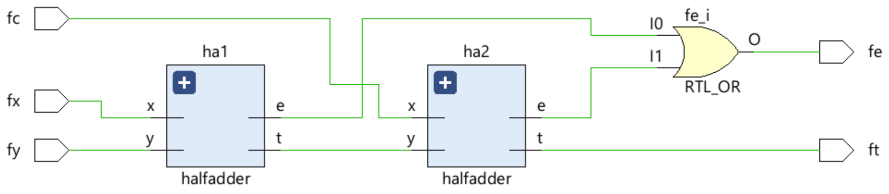


Figure iv: Full Adder Schematic

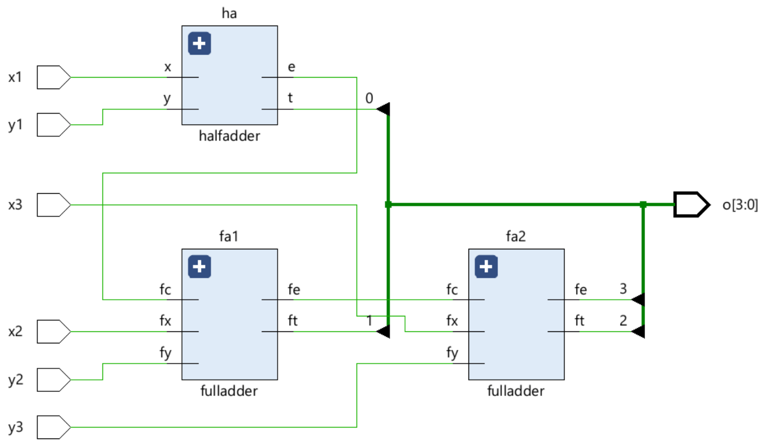


Figure v: Addition Schematic

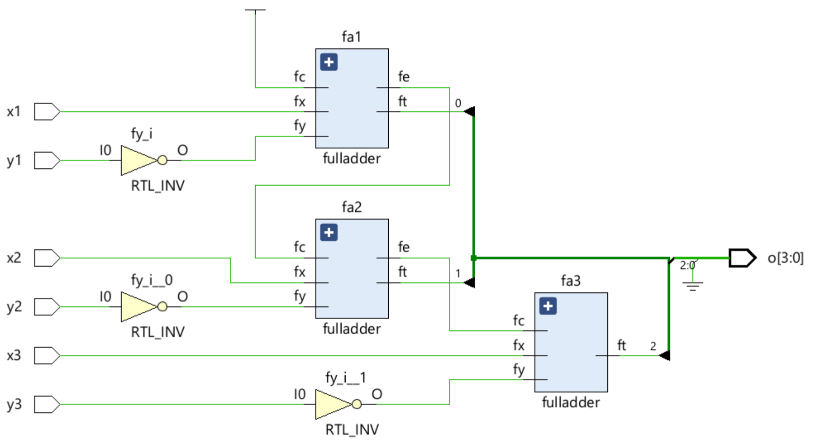


Figure vi: Subtraction Schematic

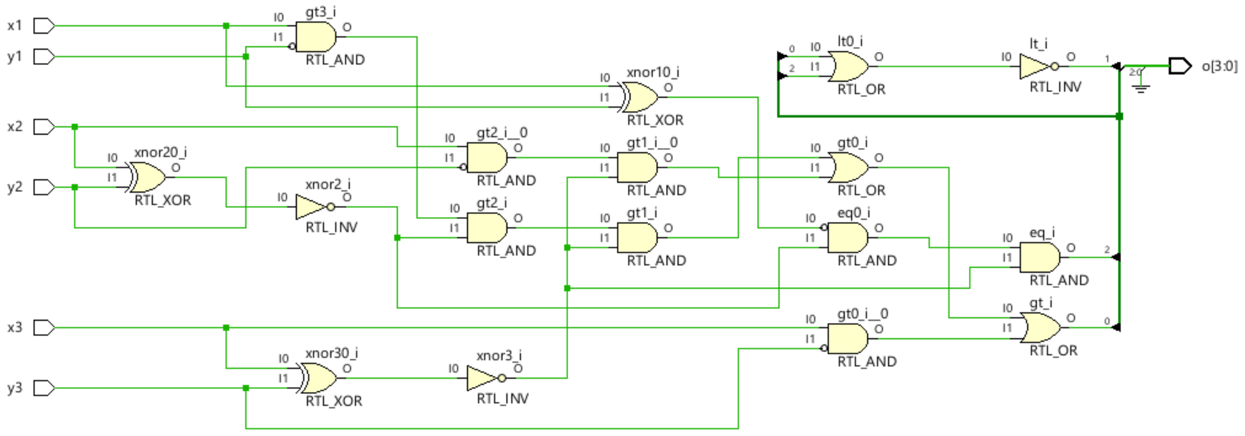


Figure vii: Comparator Schematic

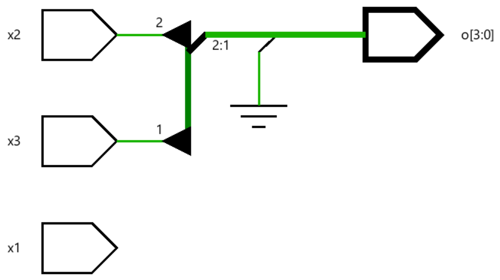


Figure viii: Left Logical Shift Schematic

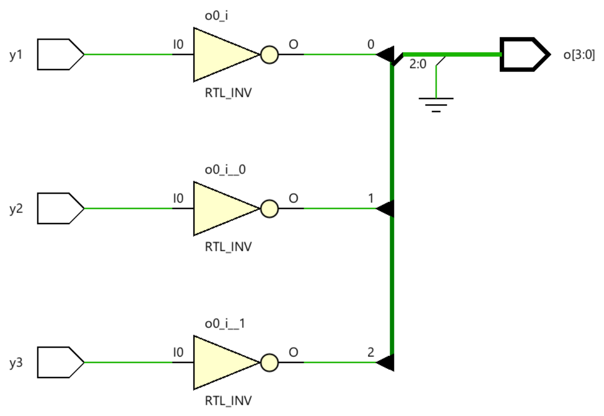


Figure ix: One's Complement Schematic

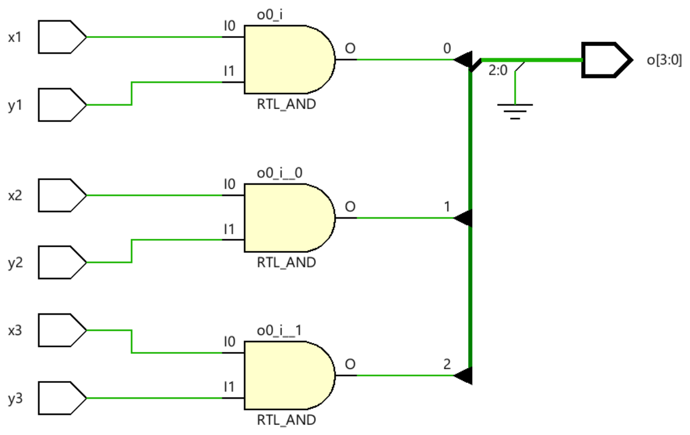


Figure x: AND Gate Schematic

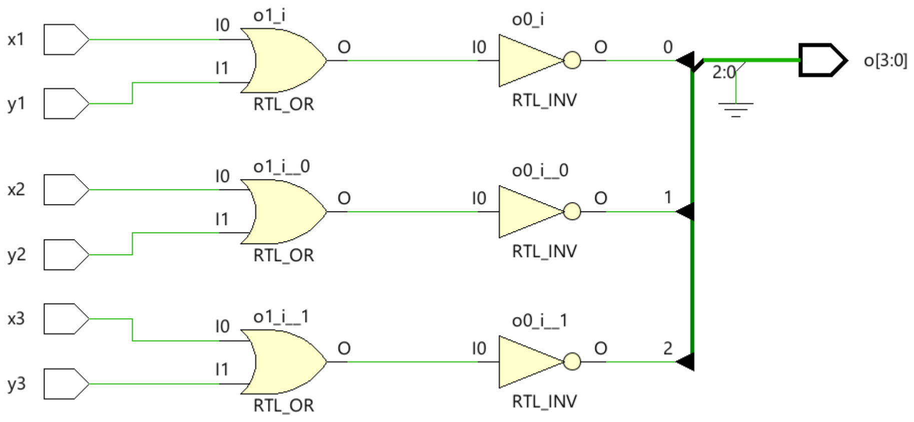


Figure xi: NOR Gate Schematic

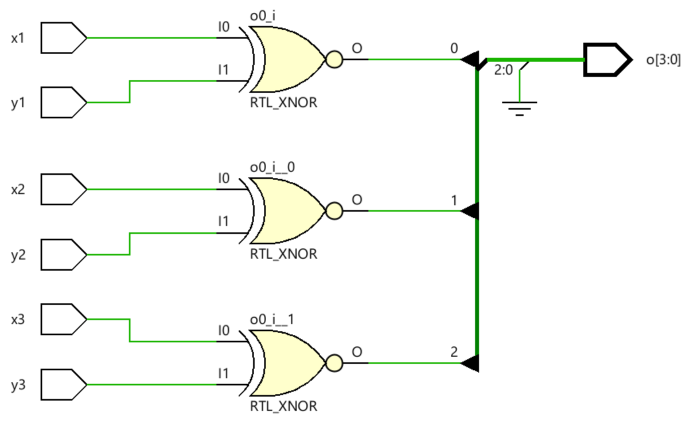


Figure xii: XNOR Gate Schematic

TOP

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;


entity top is

   Port ( x1, x2, x3, y1, y2, y3 : in STD_LOGIC;

        selection : in STD_LOGIC_VECTOR (2 downto 0);

        o : out STD_LOGIC_VECTOR (3 downto 0));

end top;


architecture Behavioral of top is


component addition

   Port ( x1, x2, x3, y1, y2,y3 : in STD_LOGIC;

        o : out STD_LOGIC_VECTOR(3 downto 0));

end component;


component substraction

   Port ( x1, x2, x3, y1, y2,y3 : in STD_LOGIC;

        o : out STD_LOGIC_VECTOR(3 downto 0));

end component;


component comparator

   Port ( x1, x2, x3, y1, y2,y3 : in STD_LOGIC;

        o : out STD_LOGIC_VECTOR(3 downto 0));
```

```
end component;

component left_logical_shift
   Port ( x1, x2, x3: in STD_LOGIC;
          o : out STD_LOGIC_VECTOR(3 downto 0));
end component;

component ones_complement
   Port ( y1, y2, y3: in STD_LOGIC;
          o : out STD_LOGIC_VECTOR(3 downto 0));
end component;

component and_gate
   Port ( x1, x2, x3, y1, y2,y3 : in STD_LOGIC;
          o : out STD_LOGIC_VECTOR(3 downto 0));
end component;

component nor_gate
   Port ( x1, x2, x3, y1, y2,y3 : in STD_LOGIC;
          o : out STD_LOGIC_VECTOR(3 downto 0));
end component;

component xnor_gate
   Port ( x1, x2, x3, y1, y2,y3 : in STD_LOGIC;
          o : out STD_LOGIC_VECTOR(3 downto 0));
end component;

signal inp_a1, inp_a2, inp_a3, inp_b1, inp_b2, inp_b3: std_logic;
signal o1, o2, o3, o4, o5, o6, o7, o8: std_logic_vector(3 downto 0);
begin
inp_a1 <= x1;
inp_a2 <= x2;
inp_a3 <= x3;
inp_b1 <= y1;
inp_b2 <= y2;
```

```
inp_b3 <= y3;


function1: addition
   port map(x1 => inp_a1, x2 => inp_a2, x3 => inp_a3,
            y1 => inp_b1, y2 => inp_b2, y3 => inp_b3,
            o => o1);
function2: substraction
   port map(x1 => inp_a1, x2 => inp_a2, x3 => inp_a3,
            y1 => inp_b1, y2 => inp_b2, y3 => inp_b3,
            o => o2);
function3: comparator
   port map(x1 => inp_a1, x2 => inp_a2, x3 => inp_a3,
            y1 => inp_b1, y2 => inp_b2, y3 => inp_b3,
            o => o3);
function4: left_logical_shift
   port map(x1 => inp_b1, x2 => inp_b2, x3 => inp_b3,
            o => o4);
function5: ones_complement
   port map(y1 => inp_b1, y2 => inp_b2, y3 => inp_b3,
            o => o5);
function6: and_gate
   port map(x1 => inp_a1, x2 => inp_a2, x3 => inp_a3,
            y1 => inp_b1, y2 => inp_b2, y3 => inp_b3,
            o => o6);
function7: nor_gate
   port map(x1 => inp_a1, x2 => inp_a2, x3 => inp_a3,
            y1 => inp_b1, y2 => inp_b2, y3 => inp_b3,
            o => o7);
function8: xnor_gate
   port map(x1 => inp_a1, x2 => inp_a2, x3 => inp_a3,
            y1 => inp_b1, y2 => inp_b2, y3 => inp_b3,
            o => o8);


process(selection, o1, o2, o3, o4, o5, o6, o7, o8)
begin
```

```
    if selection = "000" then
        o <= o1;
    elsif selection = "001" then
        o <= o2;
    elsif selection = "010" then
        o <= o3;
    elsif selection = "011" then
        o <= o4;
    elsif selection = "100" then
        o <= o5;
    elsif selection = "101" then
        o <= o6;
    elsif selection = "110" then
        o <= o7;
    elsif selection = "111" then
        o <= o8;
    else
        o <= "0000";
    end if;
end process;
end Behavioral;
```

HALFADDER

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

entity halfadder is
    Port ( x,y : in STD_LOGIC;
           e,t : out STD_LOGIC);
end halfadder;

architecture Behavioral of halfadder is

begin
e <=  x and y;
t <=  x xor y;
```

```
end Behavioral;
```

FULLADDER

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

entity fulladder is
   port ( fx,fy,fc : in STD_LOGIC;
          fe,ft : out STD_LOGIC);
end fulladder;

architecture Behavioral of fulladder is
signal ht1,he1,he2: std_logic;
component halfadder
   port ( x,y : in STD_LOGIC;
          e,t : out STD_LOGIC);
end component;
begin
ha1: halfadder
   port map(fx,fy,he1,ht1);
ha2: halfadder
   port map(fc,ht1,he2,ft);
fe <= he1 or he2;

end Behavioral;
```

ADDITION

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

entity addition is
   Port ( x1,x2,x3,y1,y2,y3 : in STD_LOGIC;
          o : out STD_LOGIC_VECTOR(3 downto 0));
end addition;

architecture Behavioral of addition is
```

```
signal he,fa1e :std_logic;

component halfadder

   port ( x,y : in STD_LOGIC;

          e,t : out STD_LOGIC);

end component;


component fulladder

   port ( fx,fy,fc : in STD_LOGIC;

          fe,ft : out STD_LOGIC);

end component;


begin

ha: halfadder

   port map(x1,y1,he,o(0));

fa1: fulladder

   port map(x2,y2,he,fa1e,o(1));

fa2: fulladder

   port map(x3,y3,fa1e,o(3),o(2));

end Behavioral;
```

SUBSTRACTION

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;


entity substraction is

   Port ( x1,x2,x3,y1,y2,y3 : in STD_LOGIC;

          o : out STD_LOGIC_VECTOR(3 downto 0));

end substraction;


architecture Behavioral of substraction is

signal fa1e,fa2e,ignore :std_logic;

component fulladder

   port ( fx,fy,fc : in STD_LOGIC;

          fe,ft : out STD_LOGIC);

end component;
```

```
begin
fa1: fulladder
   port map(x1,not y1,'1',fa1e,o(0));
fa2: fulladder
   port map(x2,not y2,fa1e,fa2e,o(1));
fa3: fulladder
   port map(x3,not y3,fa2e,ignore,o(2));
o(3) <= '0';
end Behavioral;
```

COMPARATOR

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

entity comparator is
   Port ( x1,x2,x3,y1,y2,y3 : in STD_LOGIC;
          o : out STD_LOGIC_VECTOR(3 downto 0));
end comparator;

architecture Behavioral of comparator is
signal xnor1, xnor2, xnor3, gt, eq, lt: std_logic;
begin
xnor1 <= not (x1 xor y1);
xnor2 <= not (x2 xor y2);
xnor3 <= not (x3 xor y3);
gt <= (x1 and ( not y1) and xnor2 and xnor3) or (x2 and (not y2) and xnor3) or (x3 and (not y3));
eq <= xnor1 and xnor2 and xnor3;
lt <= not (gt or eq);
o(0) <= gt;
o(1) <= lt;
o(2) <= eq;
o(3) <= '0';
end Behavioral;
```

LEFT_LOGICAL_SHIFT

```
library IEEE;
```

```
use IEEE.STD_LOGIC_1164.ALL;


entity left_logical_shift is
   Port ( x1,x2,x3 : in STD_LOGIC;
          o : out STD_LOGIC_VECTOR(3 downto 0));
end left_logical_shift;


architecture Behavioral of left_logical_shift is


begin
o(0) <= '0';
o(1) <= x3;
o(2) <= x2;
o(3) <= '0';


end Behavioral;
```

ONES_COMPLEMENT

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


entity ones_complement is
   Port ( y1,y2,y3 : in STD_LOGIC;
          o : out STD_LOGIC_VECTOR(3 downto 0));
end ones_complement;


architecture Behavioral of ones_complement is


begin
o(0) <= not y1;
o(1) <= not y2;
o(2) <= not y3;
o(3) <= '0';
end Behavioral;
```

AND_GATE

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

entity and_gate is
   Port ( x1,x2,x3,y1,y2,y3 : in STD_LOGIC;
          o : out STD_LOGIC_VECTOR(3 downto 0));
end and_gate;

architecture Behavioral of and_gate is

begin
o(0) <= x1 and y1;
o(1) <= x2 and y2;
o(2) <= x3 and y3;
o(3) <= '0';
end Behavioral;
```

NOR_GATE

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

entity nor_gate is
   Port ( x1,x2,x3,y1,y2,y3 : in STD_LOGIC;
          o : out STD_LOGIC_VECTOR(3 downto 0));
end nor_gate;

architecture Behavioral of nor_gate is

begin
o(0) <= x1 nor y1;
o(1) <= x2 nor y2;
o(2) <= x3 nor y3;
o(3) <= '0';
end Behavioral;
```

XNOR_GATE

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;


entity xnor_gate is

    Port ( x1,x2,x3,y1,y2,y3 : in STD_LOGIC;

           o : out STD_LOGIC_VECTOR(3 downto 0));

end xnor_gate;


architecture Behavioral of xnor_gate is


begin

o(0) <= x1 xnor y1;

o(1) <= x2 xnor y2;

o(2) <= x3 xnor y3;

o(3) <= '0';

end Behavioral;
```

TEST_TOP

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;


entity test_top is

--  Port ( );

end test_top;


architecture Behavioral of test_top is

component top

    Port ( x1, x2, x3, y1, y2, y3 : in STD_LOGIC;

           selection : in STD_LOGIC_VECTOR (2 downto 0);

           o : out STD_LOGIC_VECTOR (3 downto 0));

end component;

    signal x1, x2, x3, y1, y2, y3 : std_logic;

    signal selection: std_logic_vector(2 downto 0);

    signal o: std_logic_vector(3 downto 0);

begin

func: top
```

```
  port map(x1,x2,x3,y1,y2,y3, selection,o);
test_top: process
begin
    selection <= "000";
    x1 <= '0';
    x2 <= '0';
    x3 <= '0';
    y1 <= '0';
    y2 <= '0';
    y3 <= '0';
  wait for 25ns;
    selection <= "000";
    x1 <= '1';
    x2 <= '1';
    x3 <= '1';
    y1 <= '1';
    y2 <= '1';
    y3 <= '1';
  wait for 25ns;
    selection <= "001";
    x1 <= '1';
    x2 <= '1';
    x3 <= '1';
    y1 <= '1';
    y2 <= '1';
    y3 <= '1';
  wait for 25ns;
    selection <= "010";
    x1 <= '1';
    x2 <= '1';
    x3 <= '1';
    y1 <= '1';
    y2 <= '1';
    y3 <= '1';
  wait for 25ns;
```

```
    selection <= "011";
    x1 <= '1';
    x2 <= '1';
    x3 <= '1';
    y1 <= '1';
    y2 <= '1';
    y3 <= '1';
wait for 25ns;
    selection <= "100";
    x1 <= '1';
    x2 <= '1';
    x3 <= '1';
    y1 <= '1';
    y2 <= '1';
    y3 <= '1';
wait for 25ns;
    selection <= "101";
    x1 <= '1';
    x2 <= '1';
    x3 <= '1';
    y1 <= '1';
    y2 <= '1';
    y3 <= '1';
wait for 25ns;
    selection <= "110";
    x1 <= '1';
    x2 <= '1';
    x3 <= '1';
    y1 <= '1';
    y2 <= '1';
    y3 <= '1';
wait for 25ns;
    selection <= "111";
    x1 <= '1';
    x2 <= '1';
    x3 <= '1';

    y1 <= '1';

    y2 <= '1';

    y3 <= '1';

wait for 25ns;

    selection <= "000";

    x1 <= '1';

    x2 <= '1';

    x3 <= '1';

    y1 <= '0';

    y2 <= '0';

    y3 <= '0';

wait for 25ns;

    selection <= "001";

    x1 <= '1';

    x2 <= '1';

    x3 <= '1';

    y1 <= '0';

    y2 <= '0';

    y3 <= '0';

wait for 25ns;

    selection <= "010";

    x1 <= '1';

    x2 <= '1';

    x3 <= '1';

    y1 <= '0';

    y2 <= '0';

    y3 <= '0';

wait for 25ns;

    selection <= "011";

    x1 <= '1';

    x2 <= '1';

    x3 <= '1';

    y1 <= '0';

    y2 <= '0';
    y3 <= '0';

wait for 25ns;

    selection <= "100";

    x1 <= '1';

    x2 <= '1';

    x3 <= '1';

    y1 <= '0';

    y2 <= '0';

    y3 <= '0';

wait for 25ns;

    selection <= "101";

    x1 <= '1';

    x2 <= '1';

    x3 <= '1';

    y1 <= '0';

    y2 <= '0';

    y3 <= '0';

wait for 25ns;

    selection <= "110";

    x1 <= '1';

    x2 <= '1';

    x3 <= '1';

    y1 <= '0';

    y2 <= '0';

    y3 <= '0';

wait for 25ns;

    selection <= "111";

    x1 <= '1';

    x2 <= '1';

    x3 <= '1';

    y1 <= '0';

    y2 <= '0';

    y3 <= '0';

wait for 25ns;

    selection <= "000";
    x1 <= '1';

    x2 <= '0';

    x3 <= '1';

    y1 <= '0';

    y2 <= '1';

    y3 <= '0';

wait for 25ns;

    selection <= "001";

    x1 <= '1';

    x2 <= '0';

    x3 <= '1';

    y1 <= '0';

    y2 <= '1';

    y3 <= '0';

wait for 25ns;

    selection <= "010";

    x1 <= '1';

    x2 <= '0';

    x3 <= '1';

    y1 <= '0';

    y2 <= '1';

    y3 <= '0';

wait for 25ns;

    selection <= "011";

    x1 <= '1';

    x2 <= '0';

    x3 <= '1';

    y1 <= '0';

    y2 <= '1';

    y3 <= '0';

wait for 25ns;

    selection <= "100";

    x1 <= '1';

    x2 <= '0';

    x3 <= '1';
```

```
        y1 <= '0';
        y2 <= '1';
        y3 <= '0';
    wait for 25ns;
        selection <= "101";
        x1 <= '1';
        x2 <= '0';
        x3 <= '1';
        y1 <= '0';
        y2 <= '1';
        y3 <= '0';
    wait for 25ns;
        selection <= "110";
        x1 <= '1';
        x2 <= '0';
        x3 <= '1';
        y1 <= '0';
        y2 <= '1';
        y3 <= '0';
    wait for 25ns;
        selection <= "111";
        x1 <= '1';
        x2 <= '0';
        x3 <= '1';
        y1 <= '0';
        y2 <= '1';
        y3 <= '0';
end process;


end Behavioral;
```