Lab 1: Familiarizing with the syntax, data types, and operators of VHDL.

Lab 2: Design of Basic Logic Gates using VHDL.

Lab 3: Design of Half Adder and Full Adder using VHDL.

Lab 4: Design of Multiplexer and De multiplexer using VHDL.

Lab 5: Design 4-bit binary-to-gray and gray-to-binary code converters using VHDL.

Lab 6: Design 8-bit parity generator and checker circuits using VHDL.

Lab 7: Design Encoder and Decoder using VHDL.

Lab 8: Design 2's Complement Adder-Subtractor using VHDL.

Lab 9: Design of Registers using VHDL (SR flip-flop or JK flip-flop or D flip-flop or T flip-flop).

Lab 10: Design 4-bit ALU using VHDL.

Lab 11: Design of CPU using VHDL.

Lab 12: Simulation of 5 stage or 4 stage or 3 stage pipelining.

Lab 13: Simulation of Booth addition and subtraction of signed 2's complement data. (Implement using VHDL or C)

Lab 14: Simulation of Boot multiplication and division algorithm. (Implement using VHDL or C program)

EXPERIMENT No-2

Aim:

To Design Logic Gates using VHDL.

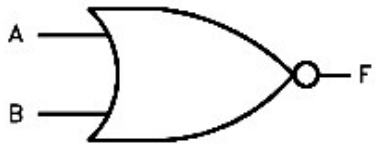
Description:

A logic gate performs a logical operation on one or more logic inputs and produces a single logic output. The logic normally performed is Boolean logic and found in digital circuit. These gates are the AND, OR, NOT, NAND, NOR, EXOR and EXNOR gates. The basic operations are described below with the aid of truth tables.

AND Gate

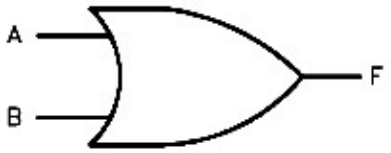
INPUT		OUTPUT
A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

NOR Gate



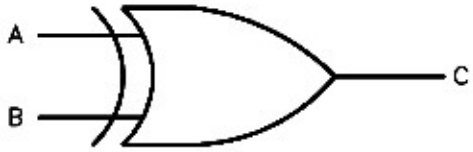
INPUT		OUTPUT
A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

OR Gate



INPUT		OUTPUT
A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

Exclusive OR Gate

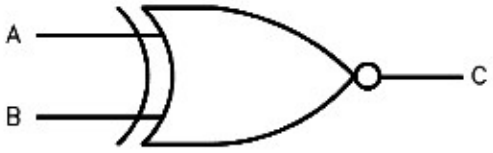


INPUT		OUTPUT
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

NAND Gate

INPUT		OUTPUT
A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

EXCLUSIVE NOR Gate



INPUT		OUTPUT
A	B	C
0	0	1
0	1	0
1	0	0
1	1	1

NOT Gate

INPUT	OUTPUT
A	F
0	1
1	0

VHDL Program:

library IEEE;

```
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
entity all_gates is
Port ( a : in STD_LOGIC;
b : in STD_LOGIC;
c : out STD_LOGIC;
c1 : out STD_LOGIC;
c2 : out STD_LOGIC;
c3 : out STD_LOGIC;
c4 : out STD_LOGIC;
c5 : out STD_LOGIC;
c6 : out STD_LOGIC );
end all_gates;
architecture Behavioral of all_gates is
begin
c <= a and b;
c1 <= a or b;
c2 <= a nand b;
c3 <= a nor b;
c4 <= a xor b;
c5 <= a xnor b;
c6 <= not b;
end Behavioral;
```

Waveforms / Output: