EXPERIMENT-1

Objective: To design and simulate various gates using VHDL.

Resources Required:

Hardware Requirement: Computer

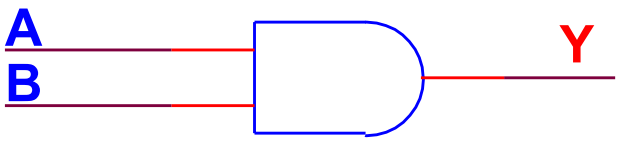
Software Requirement: XILINX 8.2 Software

Theory:

Logic gates are the essential building blocks of digital circuits. These basic logic gates are used in Embedded Systems, Microcontrollers, Microprocessors, etc.

a) **AND Gate:** A logic circuit whose output is logic '1' if and only if all of its inputs are logic '1'.

Logic diagram



Truth table

Inputs		Output
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

Boolean Expression

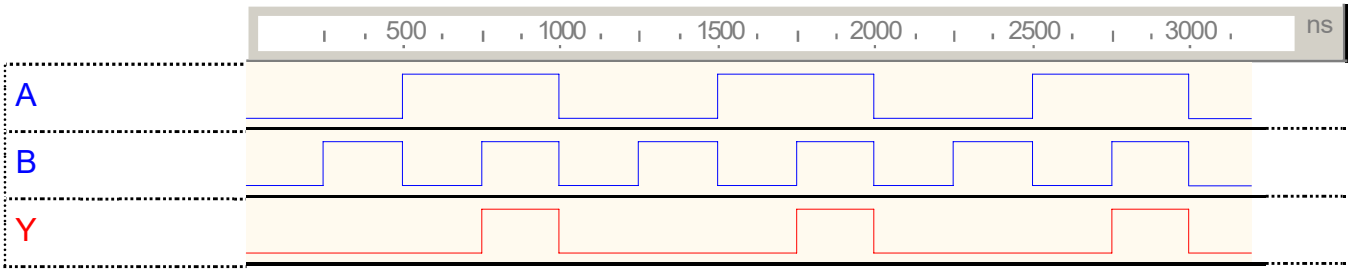
$$Y = A \text{ AND } B$$
$$= A.B$$

VHDL Code:

```
--The IEEE standard 1164 package, declares std_logic, etc.
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
      Entity Declarations
entity andgate is
Port (A : in std_logic;
      B : in std_logic;
      Y : out std_logic
);
end andgate;
```

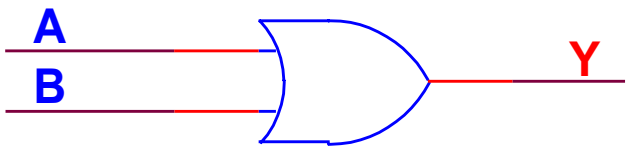
```
architecture AND GATE of andgate is
begin
Y<= A and B ;
end AND GATE;
```

Output:



b) **OR Gate:** A logic gate whose output is logic '0' if and only if all of its inputs are logic '0'.

Logic diagram



Truth table

Inputs		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Boolean Expression

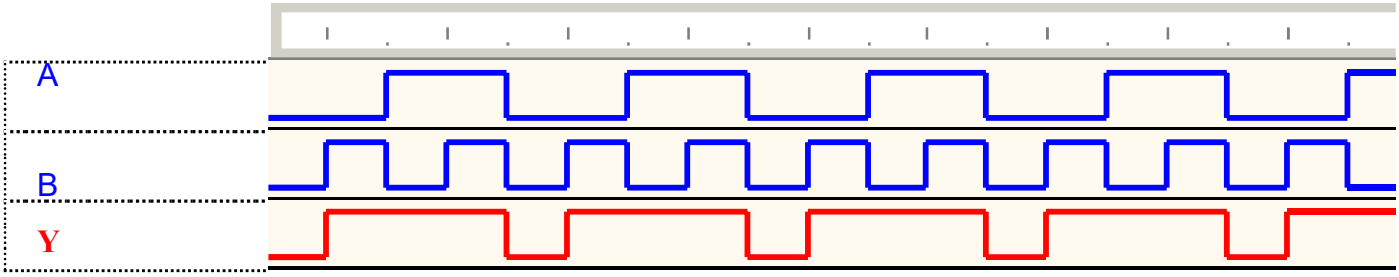
$$Y = A \text{ OR } B$$
$$= A + B$$

VHDL Code:

```
--The IEEE standard 1164 package, declares std_logic, etc.
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
      Entity Declarations
entity orgate is
Port (A : in std_logic;
      B : in std_logic;
      Y : out std_logic
```

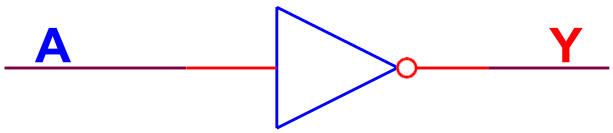
```
);
end orgate;

architecture OR GATE of orgate is
begin
Y<= A or B ;
end OR GATE;
```

Output:



c) **NOT Gate:** A logic gate whose output is complement of its input.

Logic diagram



Truth table

Inputs	Output
A	Y
0	1
1	0

Boolean Expression

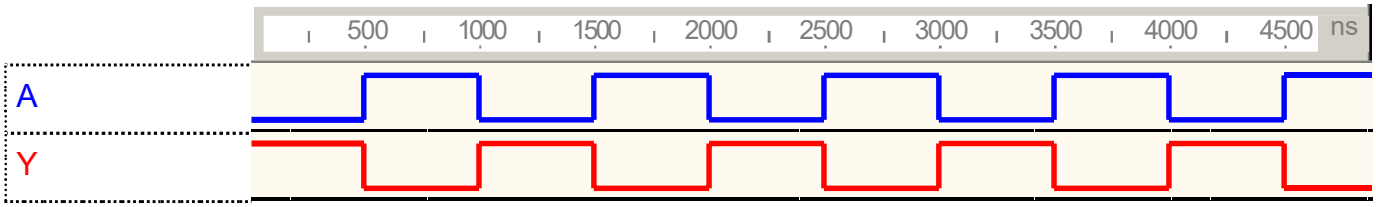
Y = NOT A

VHDL Code:

```
--The IEEE standard 1164 package, declares std_logic, etc.
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
      Entity Declarations
entity notgate is
Port (A : in std_logic;
       Y : out std_logic
);
```

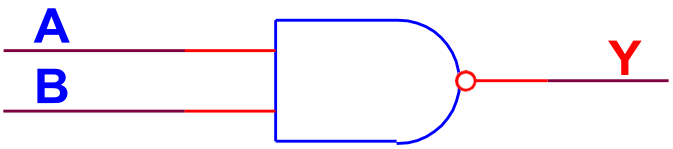
```
end notgate;

architecture NOT GATE of notgate is
begin
Y<= not A ;
end NOT GATE;
```

Output:



d) **NAND Gate:** A logic gate which gives logic '0' output if and only if all of its inputs are logic '1'

Logic diagram



Truth table

Inputs		Output
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

Boolean Expression

$$Y = A \text{ NAND } B$$

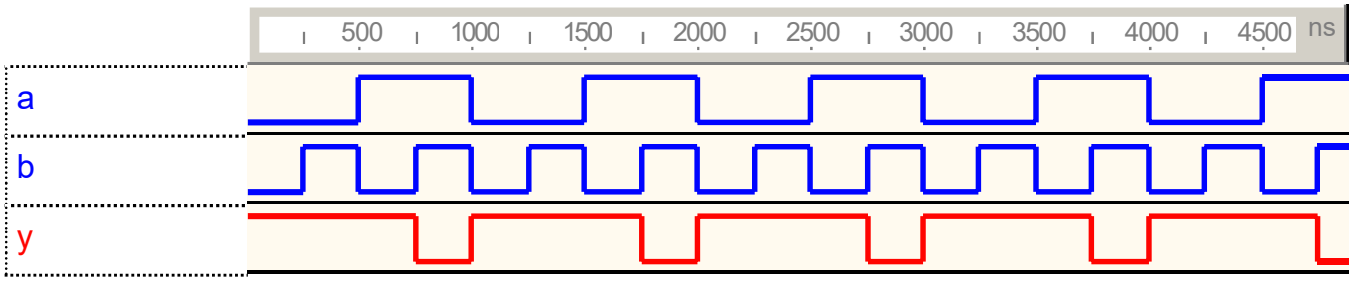
$$= \overline{A.B}$$

VHDL Code:

```
--The IEEE standard 1164 package, declares std_logic, etc.
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
      Entity Declarations
entity nandgate is
Port (A : in std_logic;
      B : in std_logic;
```

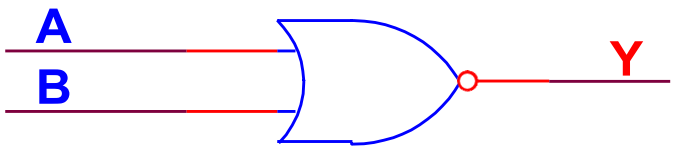
```
      Y : out std_logic
);
end nandgate;

architecture NAND GATE of nandgate is
begin
Y<= A nand B ;
end NAND GATE;
```

Output:



e) **NOR Gate:** A logic gate whose output logic is '1' if and only if all of its inputs are logic '0'

Logic diagram



Truth table

Inputs		Output
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

Boolean Expression

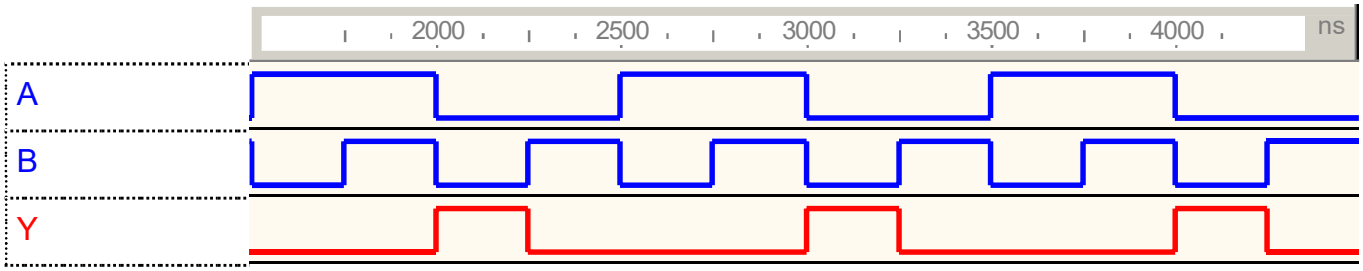
$$Y = A \text{ NOR } B$$
$$= \overline{A + B}$$

VHDL Code:

```
--The IEEE standard 1164 package, declares std_logic, etc.
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
```

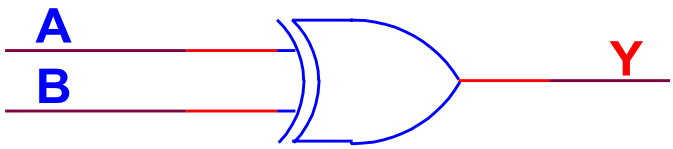
```
     Entity Declarations
entity norgate is
Port (A : in std_logic;
      B : in std_logic;
      Y : out std_logic
);
end norgate;
architecture NOR GATE of norgate is
begin
Y<= A nor B ;
end NOR GATE;
```

Output:



f) **XOR Gate:** A logic gate whose output is logic '0' when all the inputs are equal and logic '1' when they are unequal.

Logic diagram



Truth table

Inputs		Output
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Boolean Expression

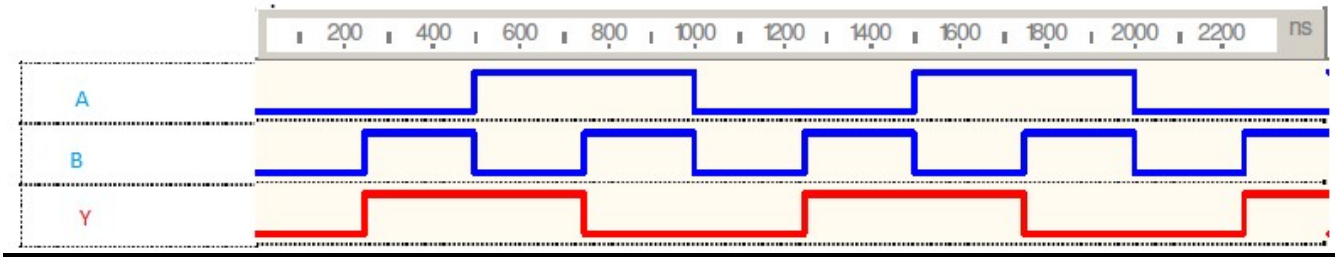
$$Y = A \text{ XOR } B$$
$$= \bar{A}B + \bar{B}A$$

VHDL Code:

```
--The IEEE standard 1164 package, declares std_logic, etc.
```

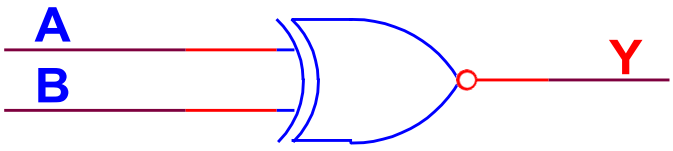
```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
      Entity Declarations
entity xorgate is
Port (A : in std_logic;
      B : in std_logic;
      Y : out std_logic
);
end xorgate;
architecture XOR GATE of xorgate is
begin
Y<= A xor B ;
end XOR GATE;
```

Output:



g) **XNOR Gate:** A logic gate that produces logic '1' only when the two inputs are equal.

Logic diagram



Truth table

Inputs		Output
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Boolean Expression

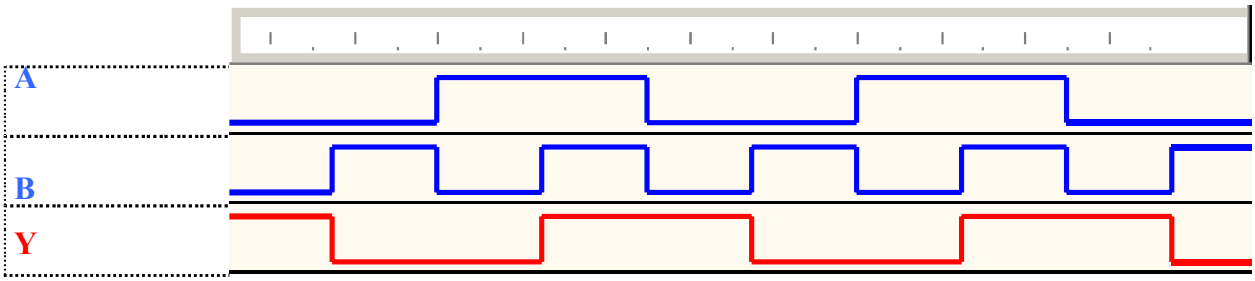
$$Y = A \text{ XNOR } B$$

$$= \bar{A}\bar{B} + AB$$

VHDL Code:

```
--The IEEE standard 1164 package, declares std_logic, etc.
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
      --Entity Declarations
entity xorgate is
Port (A : in std_logic;
      B : in std_logic;
      Y : out std_logic
);
end xnorgate;

architecture XNOR GATE of xnorgate is
begin
Y<= A xnor B ;
end XNOR GATE;
```

Output:



Results: VHDL codes of all logic gates are simulated & synthesized