

Computer Organization & Architecture Laboratory  
PCCCS492

**EXPERIMENT NO.:**

**TITLE:** Basic digital logic based programming using VHDL

**OBJECTIVE:** To write the VHDL program for different logic gates.

TITLE: Basic digital logic based programming  
using VHDL

Computer Organization & Architecture Laboratory  
PCCCS-192

EXPERIMENT NO.:

TITLE: Basic digital logic based programming  
using VHDL

OBJECTIVE: To write the VHDL program for different logic gates.

I

OBJECTIVE: To write the VHDL program for different logic gates.

THEORY: The basic logic gates are the building blocks of more complex logic circuits. These logic gates perform the basic Boolean functions, such as AND, OR, NAND, NOR, NOT, Exclusive-OR, Exclusive-NOR. Each gate has one or two binary inputs, A and B, and one binary output Y. A gate can be extended to have multiple inputs if the binary operation it represents is commutative and associative. Of all the available gates, NAND and NOR gates are called Universal gates because these two gates can design any other gates such as AND, OR and NOT.

Decimal

10

14

24

16 8 4 2 1

1 1 0 0 0

Decimal

10

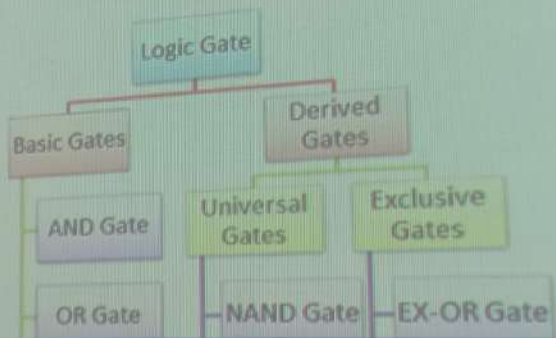
14

24

168421  
11000

# Computer Organization & Architecture Laboratory

## PCCCS492



Decimal

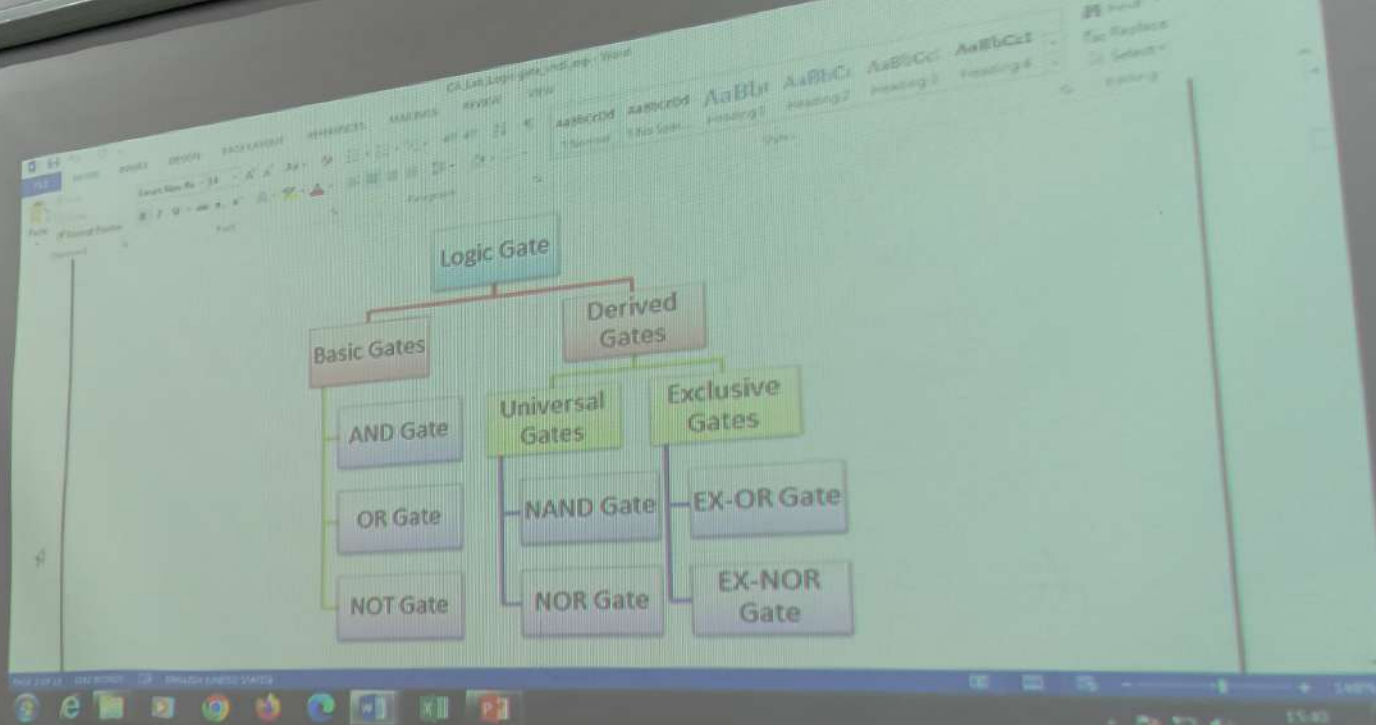
10

14

24

168421

11000





Decimal  
10  
14  
24  
16 8 4 2 1  
1 1 0 0 0

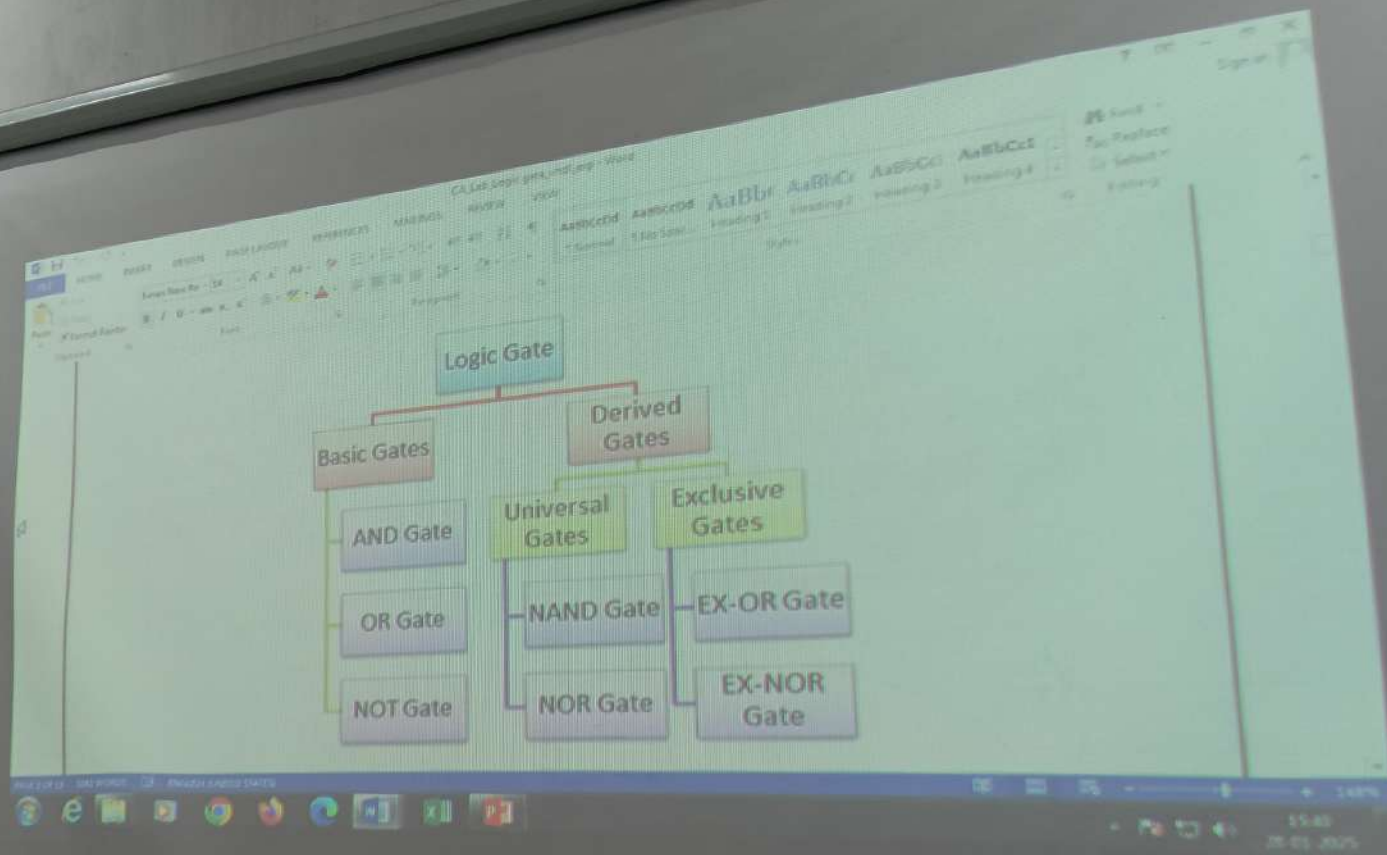


Fig. 1. Classification of Logic Gates

These basic logic gates are implemented as small-scale integrated circuits (SSICs) or as part of more complex medium scale (MSI) or very large-scale (VLSI) integrated circuits. Digital IC gates are classified not only by their logic operation, but also the specific logic-circuit family to which they belong. Each logic family has its own basic electronic circuit upon which more complex digital circuits and functions are developed. The following logic families are the most frequently used.

#### Digital Logic Functions:

Digital electronics is largely based on two-state or binary logic. The two logic states are designated "0" (low) and "1" (high) and defined electronically by fixed voltage levels. The low-state is usually the reference or ground potential in the circuit and the high-state is some positive voltage such as +5 V. A logic "gate" takes one or more logic-level inputs and produces a single logic-level output.

These basic logic gates are implemented as small-scale integrated circuits (SSICs) or as part of more complex medium scale (MSI) or very large-scale (VLSI) integrated circuits. Digital IC gates are classified not only by their logic operation, but also the specific logic-circuit family to which they belong. Each logic family has its own basic electronic circuit upon which more complex digital circuits and functions are developed. The following logic families are the most frequently used.

#### Digital Logic Functions:

Digital electronics is largely based on two-state or binary logic. The two logic states are designated "0" (low) and "1" (high) and defined electronically by fixed voltage levels. The low-state is usually the reference or ground potential in the circuit and the high-state is some positive voltage such as +5 V. A logic "gate" takes one or more logic-level inputs and produces a single logic-level output.



Normal

10

14

24

168421

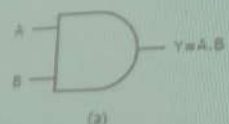
11000

CA\_Lab\_Logical\_symbols\_and\_gates\_Video

Logical Symbols and Truth Table:

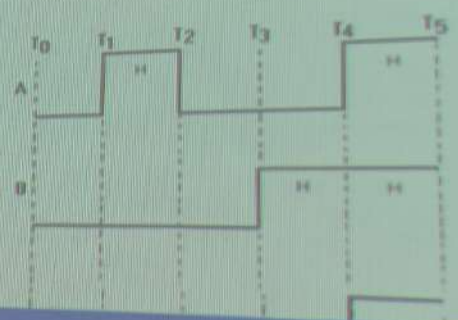
### AND GATE

The AND gate produces a HIGH output when all of the inputs are HIGH. When any inputs are LOW, the output is LOW.



(a)

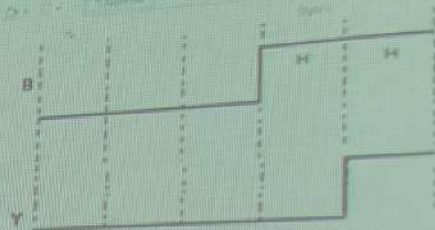
A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1



Find  
File  
Edit  
Format  
Tools  
Window  
Help

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

(b)



(c)

Fig. 2. (a) Symbol, (b) Truth Table, (c) Timing Diagram of AND gate

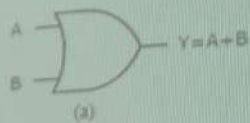
### OR GATE

The OR gate produces a HIGH output when any or all of the inputs is HIGH. When both inputs are LOW, the output

**OR GATE**  
The OR gate produces a HIGH output when any or all of the inputs is HIGH. When both inputs are LOW, the output is LOW.

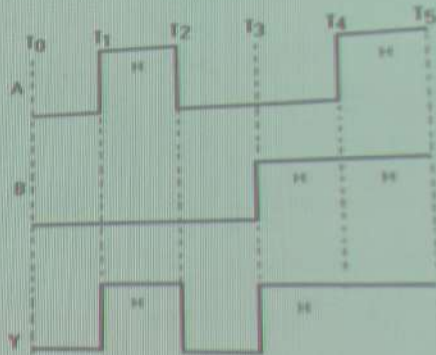
DATE \_\_\_\_\_

The OR gate produces a HIGH output when any or all of the inputs is HIGH. When both inputs are LOW, the output

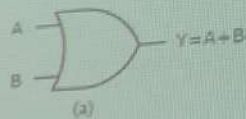


A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

(b)



(c)



A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

(b)

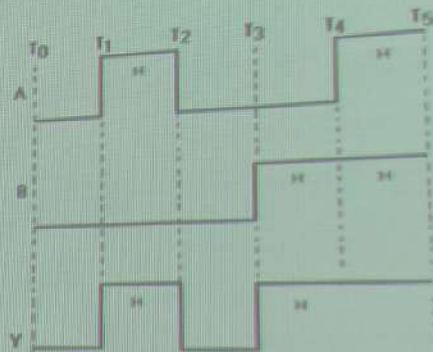


Fig. 3. (a) Symbol, (b) Truth Table, (c) Timing Diagram of OR gate



Fig. 3. (a) Symbol, (b) Truth Table, (c) Timing Diagram of OR gate

### NOT GATE

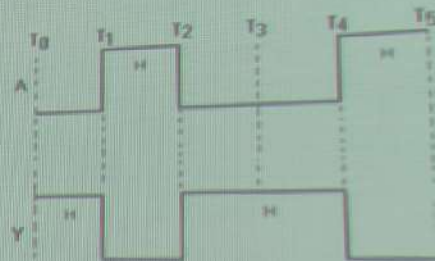
The NOT gate produces a HIGH output when input is LOW and a LOW output when input is HIGH.



(a)

A	Y
0	1
1	0

(b)



(c)

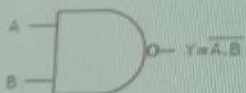


Computer Organization & Architecture Laboratory  
PCCCS492

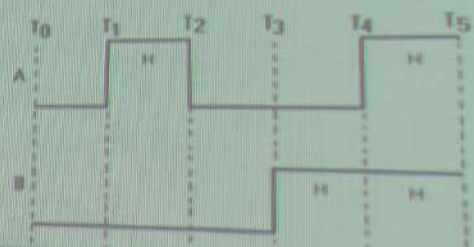
NAND GATE

The NAND gate produces a LOW output when all of the inputs are HIGH. When any inputs are

LOW, the output is HIGH.



(a)

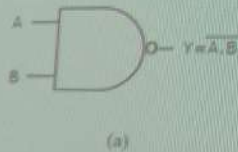


A	B	Y
---	---	---

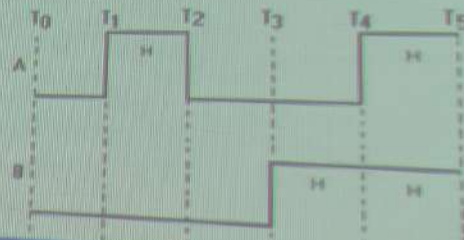
Computer Organization & Architecture Laboratory  
PCCCS492

### NAND GATE

The NAND gate produces a LOW output when all of the inputs are HIGH. When any inputs are LOW, the output is HIGH.



A	B	Y
---	---	---







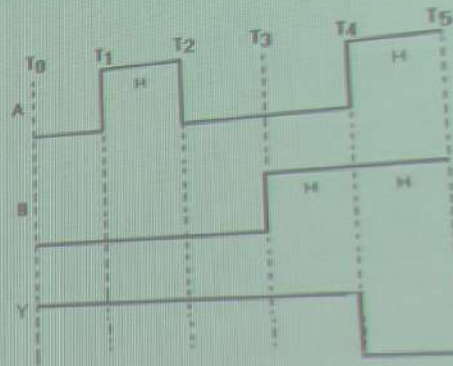
LOW, the output is HIGH.



(a)

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

(b)



(c)

Fig. 5. (a) Symbol, (b) Truth Table, (c) Timing Diagram of NAND gate

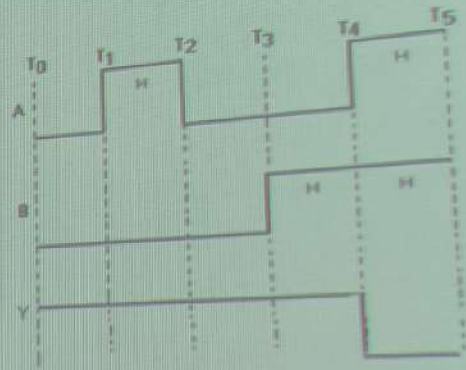
LOW, the output is HIGH.



(a)

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

(b)

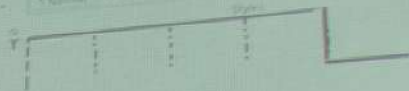


(c)

Fig. 5. (a) Symbol, (b) Truth Table, (c) Timing Diagram of NAND gate

Input	0	1
Output	1	0
Output	0	1

(b)



(c)

Fig. 5. (a) Symbol, (b) Truth Table, (c) Timing Diagram of NAND gate

### NOR GATE

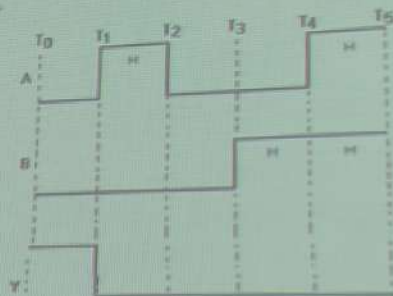
The NOR gate produces a HIGH output when all of the inputs are LOW. When any or all its inputs are HIGH, the output is LOW.



(a)

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

(b)



(c)

Fig. 6. (a) Symbol, (b) Truth Table, (c) Timing Diagram of NOR gate

### EXCLUSIVE OR GATE



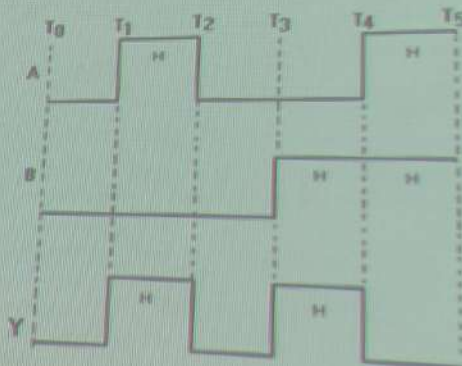
## EXCLUSIVE OR GATE

The exclusive OR gate is a modified OR gate that produces a HIGH output when only one of the inputs is HIGH. When both inputs are HIGH or when both inputs are LOW, the output is LOW.



(a)

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0





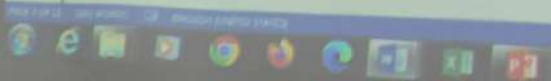
### PROBLEM STATEMENT:

1. Write down the VHDL program of two input AND gate with Data flow model.
2. Write down the VHDL program of two input OR gate with Data flow model.
3. Write down the VHDL program NOT with Data flow model.
4. Write down the VHDL program of three input NAND gate with Data flow model.
5. Write down the VHDL program of three input NOR gate with Data flow model.
6. Write down the VHDL program of three input XOR gate with Data flow model.
7. Write down the VHDL program of three input XNOR gate with Data flow model.

### PROBLEM STATEMENT:

1. Write down the VHDL program of two input AND gate with Data flow model.
2. Write down the VHDL program of two input OR gate with Data flow model.
3. Write down the VHDL program NOT with Data flow model.
4. Write down the VHDL program of three input NAND gate with Data flow model.
5. Write down the VHDL program of three input NOR gate with Data flow model.
6. Write down the VHDL program of three input XOR gate with Data flow model.
7. Write down the VHDL program of three input XNOR gate with Data flow model.

SOFTWARE USED:



### PROBLEM STATEMENT:

1. Write down the VHDL program of two input AND gate with Data flow model.
2. Write down the VHDL program of two input OR gate with Data flow model.
3. Write down the VHDL program NOT with Data flow model.
4. Write down the VHDL program of three input NAND gate with Data flow model.
5. Write down the VHDL program of three input NOR gate with Data flow model.
6. Write down the VHDL program of three input XOR gate with Data flow model.
7. Write down the VHDL program of three input XNOR gate with Data flow model.

SOFTWARE USED:



SOFTWARE USED:

PROGRAM:

1. Two input AND gate:

-- *THEM Code for AND gate*

-- *Header file declaration*

library IEEE;

use IEEE.all; logic [1:0] a,b;



## SOFTWARE USED:

### PROGRAM:

#### 1. Two input AND gate:

-- VHDL Code for AND gate

-- Header file declaration

```
library IEEE;
```

```
use IEEE std logic 1164.all;
```



-- VHDL Code for AND gate

-- Header file declaration

```
library IEEE;  
use IEEE.std_logic_1164.all;
```

-- Entity declaration

entity andGate is

port(A : in std\_logic; -- AND gate input  
B : in std\_logic; -- AND gate input  
Y : out std\_logic); -- AND gate output

end andGate;

-- Dataflow Modelling Style  
-- Architecture definition

architecture andLogic of andGate is

begin

architecture andLogic of andGate is

begin

$Y \leq A \text{ AND } B;$

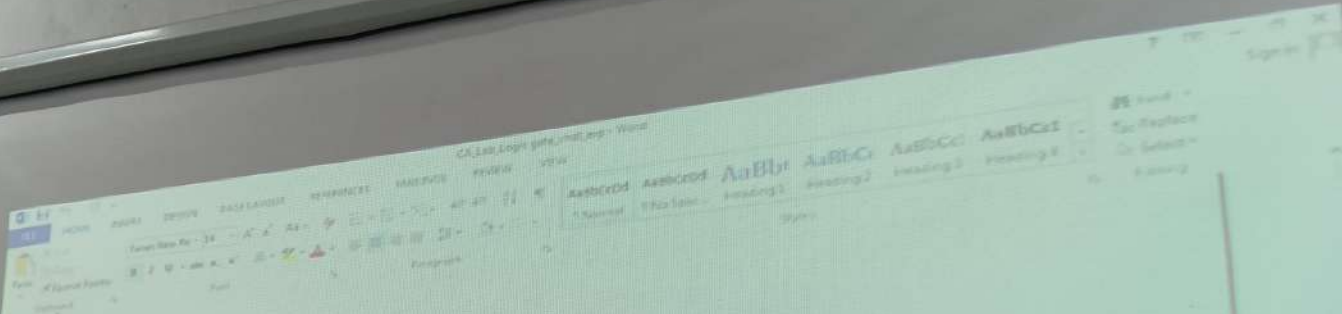
end andLogic;

## 2. Two input OR gate:

-- VHDL Code for OR gate

-- Header file declaration

library IEEE;



begin

Y <= A AND B;

end andLogic;

## 2. Two input OR gate:

-- VHDL Code for OR gate

-- Header file declaration

library IEEE;

use IEEE.std\_logic\_1164.all;

-- Entity declaration

Page 8 of 13 1042 Words READY TO PRINT



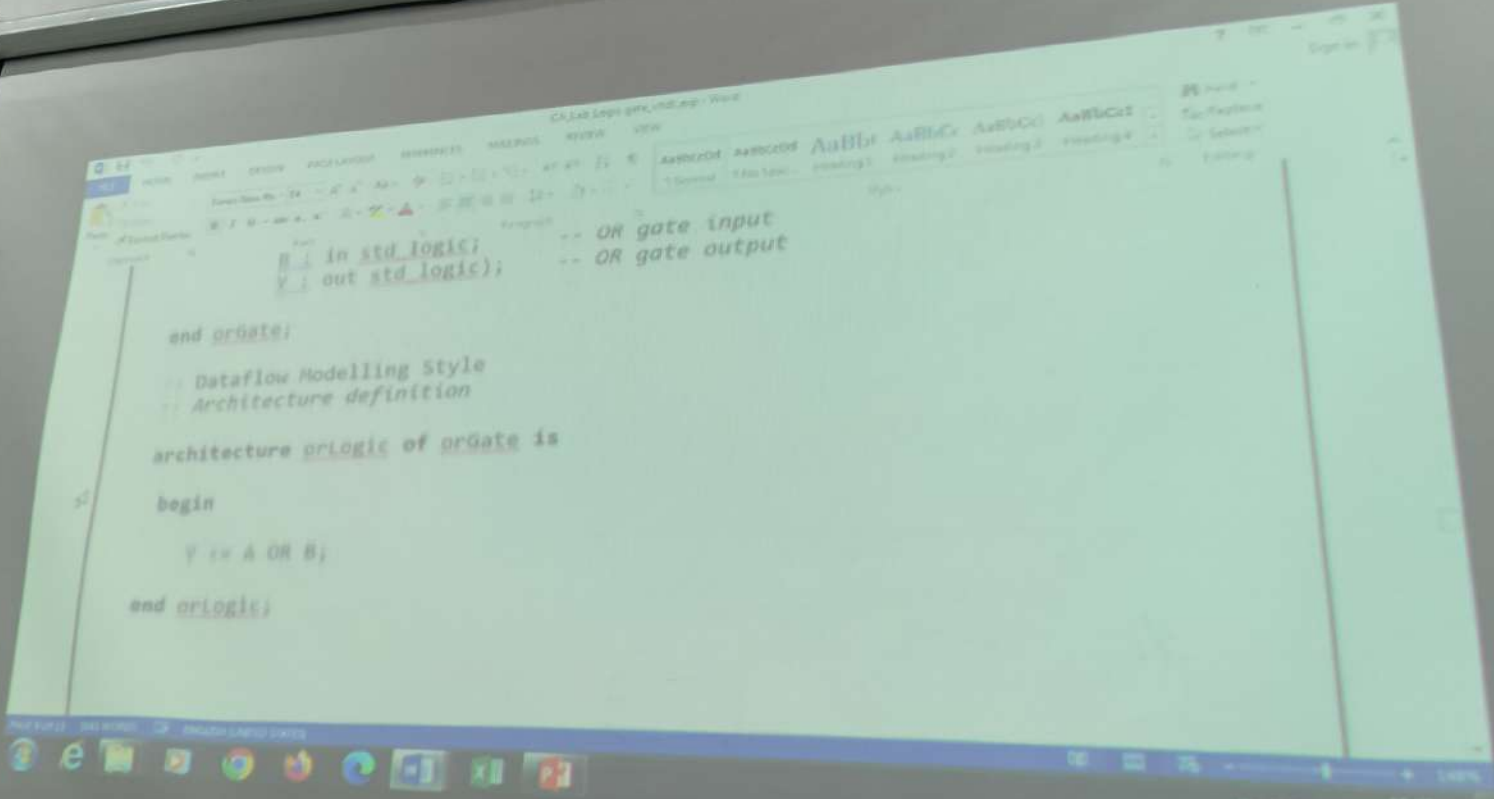
```
library IEEE;
use IEEE.std_logic_1164.all;

-- Entity declaration

entity orGate is
    port(A : in std_logic; -- OR gate input
```







```

    B : in std_logic;    -- OR gate input
    Y : out std_logic);  -- OR gate output

end orGate;

-- Dataflow Modelling Style
-- Architecture definition
architecture orLogic of orGate is
begin
    Y <= A OR B;

end orLogic;

```

### 3. NOT gate:

-- VHDL Code for NOT gate

-- Header file declaration

library IEEE;

use IEEE std logic\_1164.all;

-- Entity declaration

entity notGate is

### 3. NOT gate:

-- VHDL Code for NOT gate

-- Header file declaration

```
library IEEE;  
use IEEE.std_logic_1164.all;
```

-- Entity declaration

entity notGate is

```
    port(A : in std_logic;    -- input  
          Y : out std_logic); -- output
```

CA, Lab, Logic gates, and Logic - Word

File Home Insert Layout References Mailings View


Font Paragraph Styles

Font Face Size Bold Italic Underline Color Text Color Background Color

Paragraph Left Center Right Justify Indent Bullets Numbering

Styles Normal Title 1 Section Header 2 Section Header 3 Section Header 4

end notGate;




University of Engineering and Management

Institute of Engineering & Management, New Town Campus

Department of Computer Science & Engineering

*Computer Organization & Architecture Laboratory*

PCCCS492



IEM

INSTITUTE OF ENGINEERING & MANAGEMENT

NEW TOWN CAMPUS

Taskbar: File Explorer, Microsoft Word, Microsoft Edge, Google Chrome, Firefox, VLC, PowerPoint, Word, Excel, PowerPoint

15:43

28-01-2025



- Dataflow Modelling Style
- *Architecture definition*

architecture notLogic of notGate is

begin

Y <= not(A);

end notLogic;

#### 4. Two input NAND gate:

```
library IEEE;  
use IEEE.std_logic_1164.all;  
entity nand_gate is  
  port(A: in std_logic;  
        B: in std_logic;  
        Y: out std_logic);  
end nand_gate;  
architecture nandLogic of nand_gate is
```

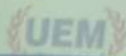
end nand\_gate;

architecture nandLogic of nand\_gate is

begin

Y <= not (A and B);

end nandLogic;



University of Engineering and Management



Page 28 of 32 | 100 WORDS | QR | English | English | 100%



13:43  
28-09-2025

### 5. Two input NOR gate:

```
library IEEE;  
use IEEE.std_logic_1164.all;  
entity nor_gate is  
  port(A: in std_logic;  
        B: in std_logic;  
        Y: out std_logic);  
end nor_gate;  
architecture norLogic of nor_gate is  
begin
```

architecture norLogic of nor\_gate is  
begin  
Y <= not(A OR B);  
end norLogic;

#### 6. Two input XOR gate:

library IEEE;  
use IEEE.std\_logic\_1164.all;  
entity xor\_gate is  
port(A: in std\_logic;



B: in std\_logic;

Y: out std\_logic);

end xor\_gate;

architecture xorLogic of xor\_gate is

begin

Y <= A xor B;

end xorLogic;

## 7. Two input XNOR gate:

library IEEE;

### 7. Two input XNOR gate:

```
library IEEE;  
use IEEE.std_logic_1164.all;  
  
entity xnor_gate is  
    port(A : in std_logic;  
         B : in std_logic;  
         Y : out std_logic);  
end xnor_gate;  
  
architecture xnorLogic of xnor_gate is  
begin
```



CONCLUSION:



17:44  
20-03-2023