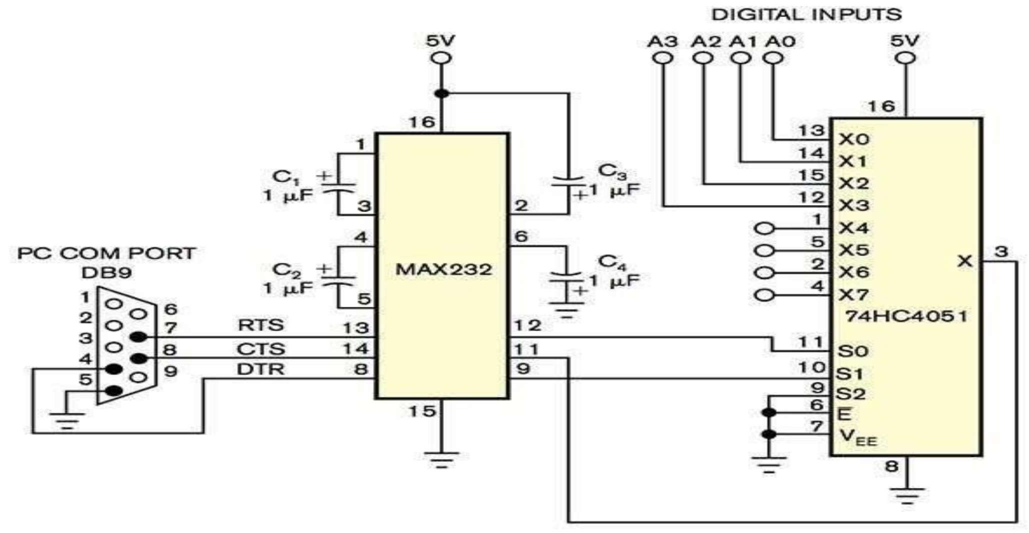
UNIVERSITY OF WOLVERHAMPTON
HERALD COLLEGE KATHMANDU

Lecture 2

4CS015: Digital Ele



Prepared by: Uttam Acharya

1. Coverage

1.1. Boolean Logic and Logic Gates

1.1.1. Truth Table

1.2. Boolean Algebra Laws

1.3. Combinational Circuit

1.1.A Boolean Logic

George Boole (1815-1864)

- "An Investigation into the Laws of Thought"
- Defined an algebra for solving logical problems
- Limited to dealing with facts True or False
- Now known as Boolean Algebra

Basic Logic Definitions

- In a Logic System a variable can have one of two possible states.
- Single capital letters are used to represent variables.
- The bits 1 and 0 are also used as constants.

TRUE	**ON**	**CLOSED**	**‘1’**	**Yes**	**5v**
FALSE	**OFF**	**OPEN**	**‘0’**	**No**	**1v**

UNIVERSITY OF WOLVERHAMPTON
HERALD COLLEGE KATHMANDU

1.1.B Logic States

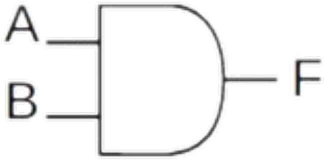
- If a switch is closed:
  - The light will be ON.
  - This can represent Logic TRUE.
- If a switch is open:
  - The light will be OFF.
  - This can represent Logic FALSE.
- The switch is a Logic variable.

1.1.B.a Boolean Basics: Operators

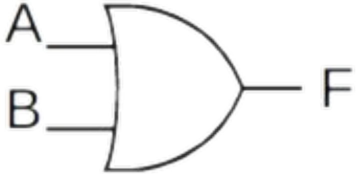
- Boole defined three basic operations that could be used with these Boolean variables.
  - AND
  - OR
  - NOT
- All logical expressions can be built from these three.

1.1.B.a Boolean Basics: Operators

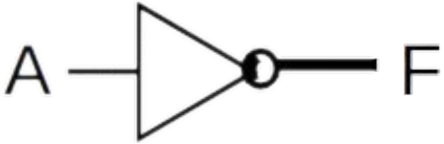
- Logical **AND**



- Logical **OR**

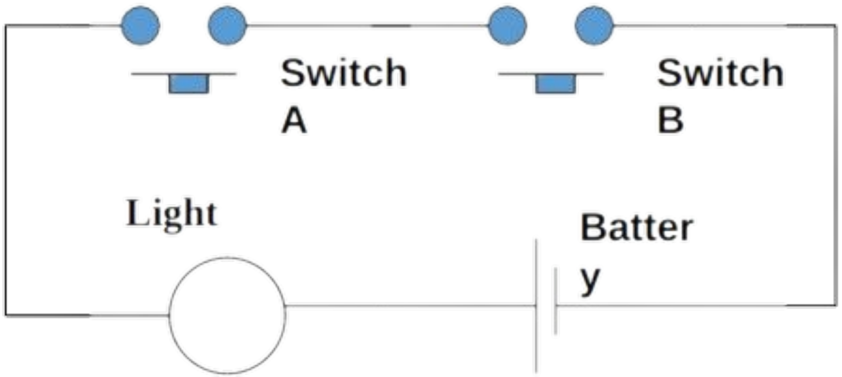


- Logical **NOT**



1.1.B.a.a Logical AND

- How can we switch the light on?



1.1 Boolean Logic & Logic Gates

1.1.A Boolean Logic

1.1.B Logic States

a.Boolean Operators

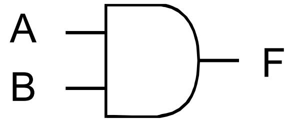
a.a Logical AND

1.1.B.a.a Logical AND

- Boolean Expression:

  F = A AND B or $F = A \cdot B$

- Gate Diagram:



- Truth Table:

Input A	Input B	Output F
0	0	0
0	1	0
1	0	0
1	1	1

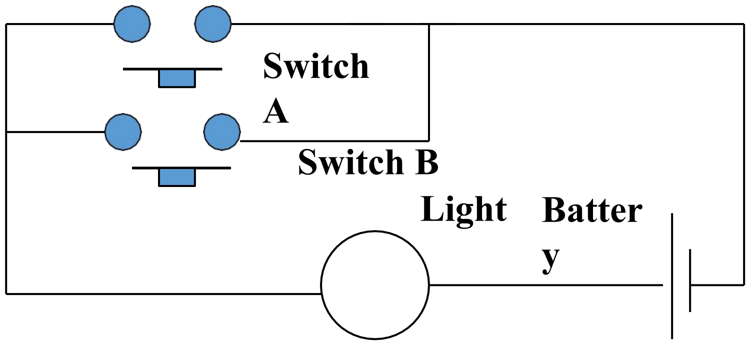
1.1.B.a.a AND Relationship

- Boolean representation . **(Period)**
- If F,A and B are Boolean variables.
- Then the expression $F = A \cdot B$ means
    - F is only true when A AND B are both true.
- As A is capable of being 1 or 0 and B is capable of being 1 or 0 there are 4 possible states. 00, 01, 10 or 11

Input A	Input B	Output F
0	0	0
0	1	0
1	0	0
1	1	1

1.1.B.a.b Logical OR

- How can we switch on the light?



1.1 Boolean Logic & Logic Gates

1.1.A Boolean Logic

1.1.B Logic States

a.Boolean Operators

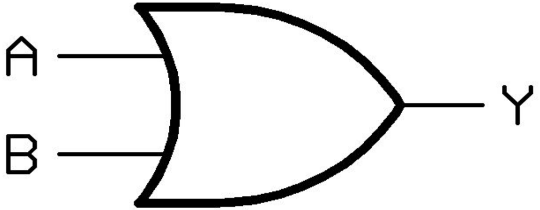
a.b Logical OR

1.1.B.a.b Logical OR

- Boolean Expression:

  F = A OR B alternatively **F =A+B**

- Gate Diagram:



- Truth Table:

Input A	Input B	Output F
0	0	0
0	1	1
1	0	1
1	1	1

1.1.B.a.b OR Relationship

- Boolean representation + **(Plus)**
- If F,A and B are Boolean variables.
- Then the expression F = A+B means
  - F is only true when A OR B, OR both, are true.
- As A is capable of being 1 or 0 and B is capable of being 1 or 0 there are 4 possible states. 00, 01, 10 or 11

Input A	Input B	Output F
0	0	0
0	1	1
1	0	1
1	1	1

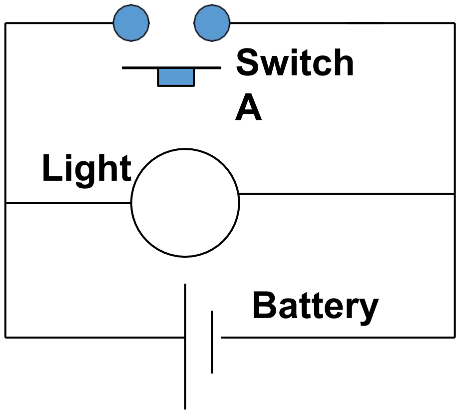
1.1 Boolean Logic & Logic Gates

1.1.A Boolean Logic

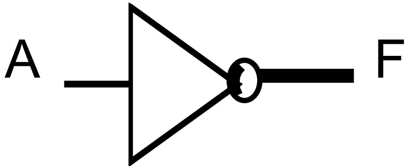
1.1.B Logic States

a.Boolean Operators

1.1.B.a.c Logical Not



- How can *we* switch the light off?
- Boolean Expression:
- **F = NOT A** or F=!A or F=$\bar{A}$
- Gate Diagram:



- Truth Table:

Input A	Output F
0	1
1	0

1.1.B.a.c Boolean Not

1.1 Boolean Logic & Logic Gates

1.1.A Boolean Logic

1.1.B Logic States

a.Boolean Operators

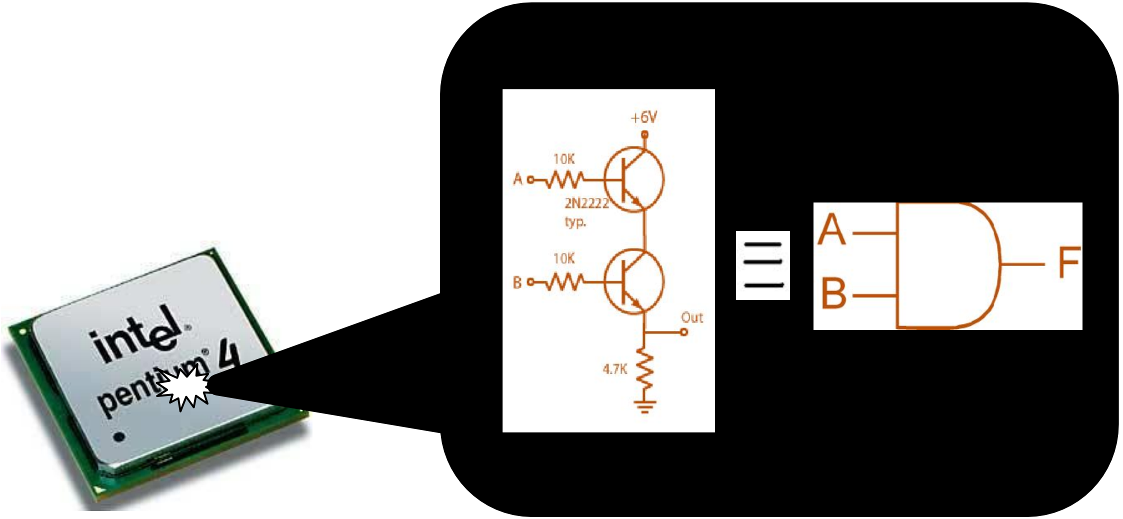
a.c Logical Not

- The NOT relationship reverses the value.
- NOT True is False etc…
- The Symbol(—) used is usually a bar above the variable or expression to be reversed
- E.g if A= true then $\bar{A}$ = false
- In some circumstances we use !
- E.g. if B = true !B = false (easier to type)

1.1 Boolean Logic & Logic Gates

- 1.1.A Boolean Logic
- 1.1.B Logic States

1.1.B Integrated Circuit and Logic Gates



1.1.B Other Logic Gates

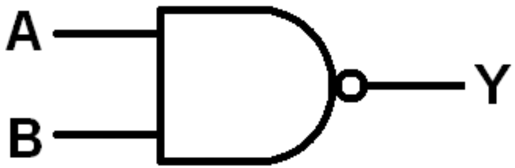
- To make life a little easier the basic logical functions are expanded to include:
  - **NAND**
    - This is an AND with a NOT output.
  - **NOR**
    - This is an OR with a NOT output.
  - **XOR**
    - This is the Exclusive OR function.
  - **XNOR**
    - This is Complement of XOR.

1.1.B NAND

- Boolean Expression:

  F = NOT(A AND B) or $F = \overline{A.B}$

- Gate Diagram:



- Truth Table:

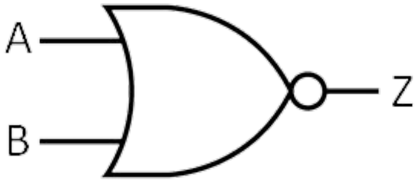
Input A	Input B	Output F
0	0	1
0	1	1
1	0	1
1	1	0

1.1.B NOR

- Boolean Expression:

  $$\mathbf{F = NOT(A\ OR\ B)\ or\ \overline{F} = \overline{A+B}}$$

- Gate Diagram:



- Truth Table:

Input A	Input B	Output F
0	0	1
0	1	0
1	0	0
1	1	0

1.1 Boolean Logic & Logic Gates

1.1.A Boolean Logic

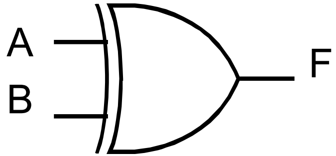
1.1.B Logic States

1.1.B XOR

- Boolean Expression:

  F = A XOR B or F = A⊕B

- Gate Diagram:



- Truth Table:

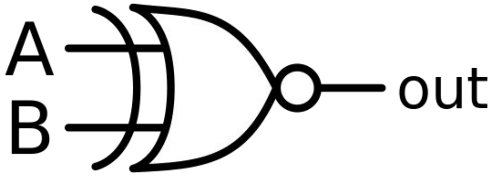
Input A	Input B	Output F
0	0	0
0	1	1
1	0	1
1	1	0

1.1.B XNOR

- Boolean Expression:

  F = A XNOR B or F = A ⊙ B

- Gate Diagram:



- Truth Table:

Input A	Input B	Output F
0	0	1
0	1	0
1	0	0
1	1	1

1.2 Boolean Algebra Laws

The operations +, . And ' consequently satisfy the basic laws 1, 2 and 3 of Boolean algebra. That is:

$A + B \equiv B + A$
$A \cdot B \equiv B \cdot A$

Commutative Laws

$(A + B) + C \equiv A + (B + C)$
$(A \cdot B) \cdot C \equiv A \cdot (B \cdot C)$

Associative Laws

$A \cdot (B + C) \equiv (A \cdot B) + (A \cdot C)$
$A + (B \cdot C) \equiv (A + B) \cdot (A + C)$

Distributive Laws

1.2 Boolean Algebra Laws

Identity Law

A + 0 = A

A . 1 = A

Negation Law

low = high

(0 = 1)

Idempotent Law

A + A = A

A . A = A

Double Negation Law

$\overline{\overline{A}} = A$

Complement Law

A . A = 0

A + A = 1

Domination Law

A + high = high

(A + 1 = 1)

A . low = low

A . 0 =0

Absorption Law

A + (A .B) = A

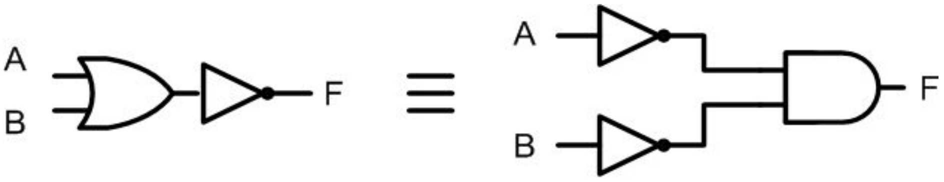
A . (A + B) = A

1.1 Boolean Logic & Logic Gates

1.2 Boolean Algebra Laws

1.2 DeMorgan’s Laws

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



$\overline{(A + B)}$ $\quad\quad$ $\overline{A} \bullet \overline{B}$

A	B	A+B	$\overline{(A+B)}$	$\overline{A}$	$\overline{B}$	$\overline{A}.\overline{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

1.1 Boolean Logic & Logic Gates

1.2 Boolean Algebra Laws

Question 1 : NAND and NOR gates are called universal gates, why?

1.2.a Precedence of operators

- As with normal mathematics when working out the value of a function it is very important to do it in the right order.
- NOT AND OR
- Parenthesis (brackets) override in the normal way.
- When a bar goes above more than 1 symbol it becomes a bracket that reverses.

1.2 Example

1.1 Boolean Logic & Logic Gates

1.2 Boolean Algebra Laws

a. Precedence of operators

Where

A=1
B=1
C=0
D=1
E=0
G=1

Evaluate these examples

Example 1. F= A.B.C

Example 2. F= C.A.B

Example 3. F= $A+\overline{B}+C$

Example 4. F= A.B+C

Example 5. F= A+B.C

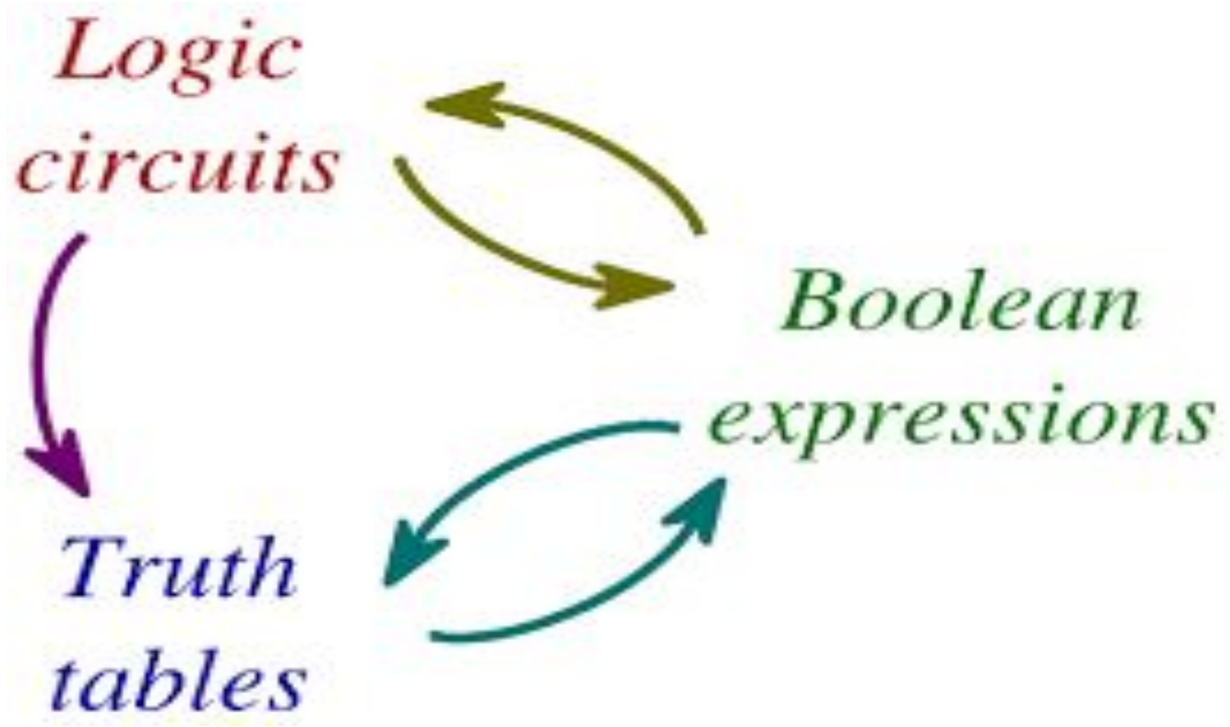
Example 6. F= $\overline{A+B}.C$

Example 7. F= A .B + (C +A) .D+E+G

1.1 Boolean Logic & Logic Gates

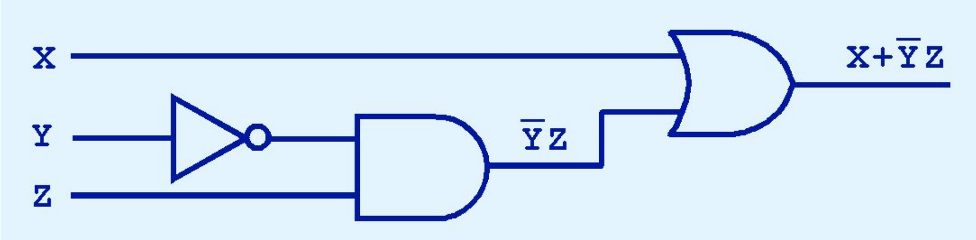
1.3 Circuit Design

1.3 Circuit Design



1.3 Digital Component

- The main thing to remember is that combinations of gates implement Boolean functions.
- The circuit below implements the Boolean function: $F(X,Y,Z) = X+\overline{Y}Z$



We simplify our Boolean expressions so that we can create simpler circuits.

1.3.a Digital Component

- We have designed a circuit that implements the Boolean function: $F(X,Y,Z) = X+\overline{Y}Z$
- This circuit is an example of a *combinational logic* circuit.
- Combinational logic circuits produce a specified output (almost) at the instant when input values are applied.
  - In a later section, we will explore circuits where this is not

    the case.

1.3.b Truth Table

1.1 Boolean Logic & Logic Gates

1.3 Circuit Design

1.3.a Digital Component

$F(x, y, z) = x\bar{z} + y$

x	y	z	$\bar{z}$	$x\bar{z}$	$x\bar{z}+y$
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	0	1

1.1 Boolean Logic & Logic Gates

1.3 Circuit Design

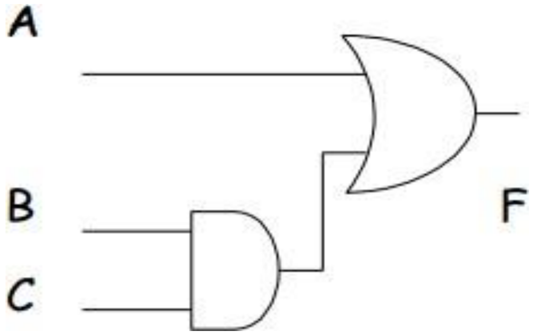
1.3.a Digital Component

1.3.c Simple Combinational Circuit

Example

$$
\begin{aligned}
F &= A.\overline{B} + A.B + B.C \\
&= A.(\overline{B} + B) + B.C \\
&= A.1 + B.C \\
&= A + B.C
\end{aligned}
$$

Circuit after Simplification



1.3.c Simple Combinational Circuit

1.1 Boolean Logic & Logic Gates

1.3 Circuit Design

1.3.a Digital Component

Example

$$F = \overline{A}.B.C + A.\overline{B}.C + A.B.\overline{C} + A.B.C$$

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

$$= (\overline{A}.B.C + A.B.C) + (A.\overline{B}.C + A.B.C) + (A.B.\overline{C} + A.B.C)$$

$$= (\overline{A} + A).\ B.C + (\overline{B} + B).\ C.A + (\overline{C} + C).\ A.B$$

$$= B.C + C.A + A.B$$

1.3.c Simple Combinational Circuit

Example

Simplify:

X = (A.B.C) +(A.B'.C) +(A'.B.C)

1.3.d Exercises

Simplify and construct the logic circuit:

1. **A’.B’ + (A.B)’**
2. **(A + B).(A + B) + A.(A + B’)**
3. **(A.B’.C’ + A’.B’.C+A.B.C+A’.B.C’)**

Summary

- We have looked at the basic logic gates:
  - Identifying OR, AND, NOT, NAND, NOR and XOR.
- We have seen that gates can be joined together to form Combinatorial Logic.

Thank you