Digital Logic :

Lecture 1 :

Introduction to Digital Logic

Boolean Algebra, Logic gates

Digital Logic :

Next Topic :

GATE weightage, Reference Book

Website : https://www.goclasses.in/

GATE:

DL → Scoring, Easy

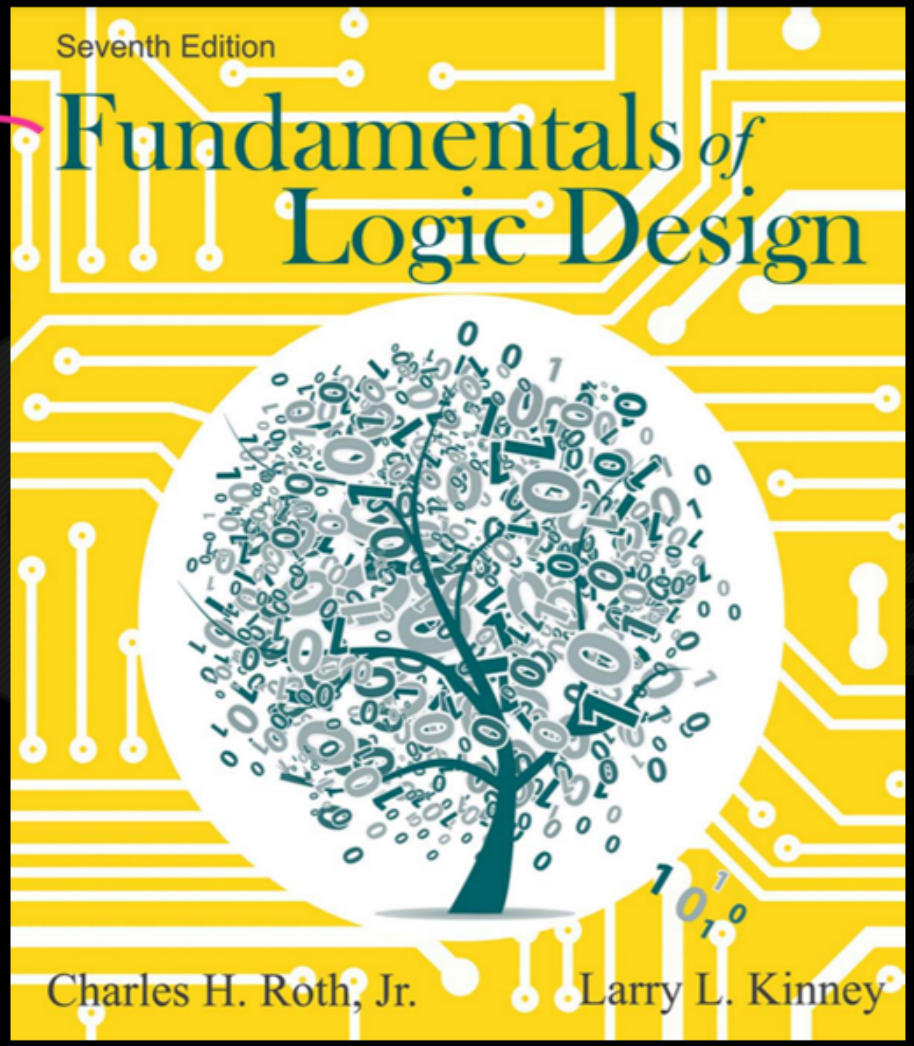
4-6 marks

1. Boolean Algebra, minimization
2. Number system
3. Combinational ckts
4. Sequential ckts

Lots of Questions

↓

Homework



Digital Logic :

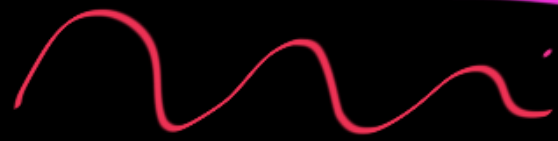
Next Topic :

Introduction of Digital Logic

Website : https://www.goclasses.in/

Two type of systems :

① Analogue System : Continuous System



43.216

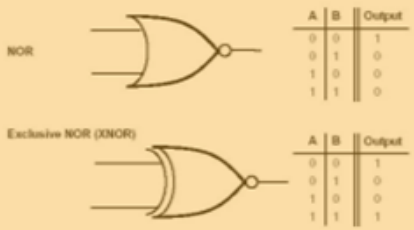
ex : Temp
Radio
Sound

② Digital System ; Discrete system

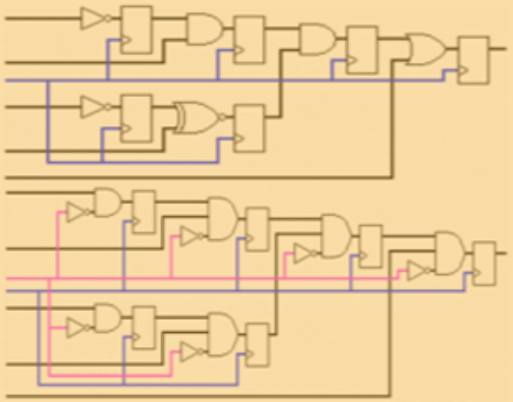
finite symbols we use

watch

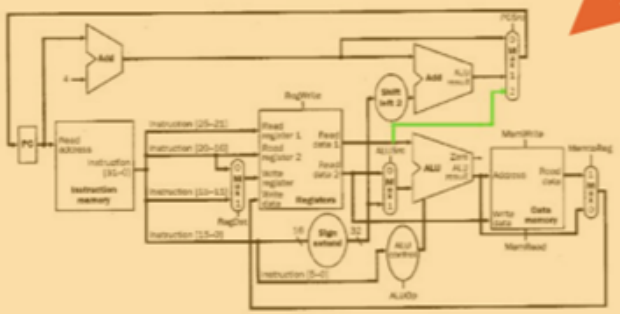
transistors



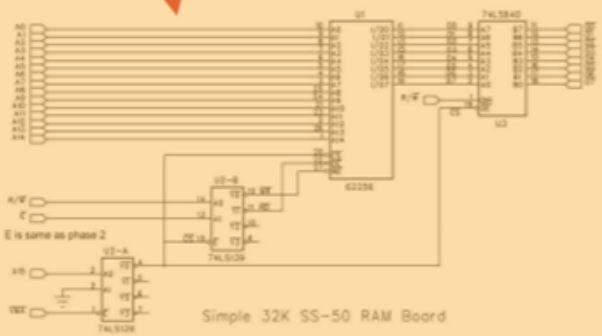
logic gates



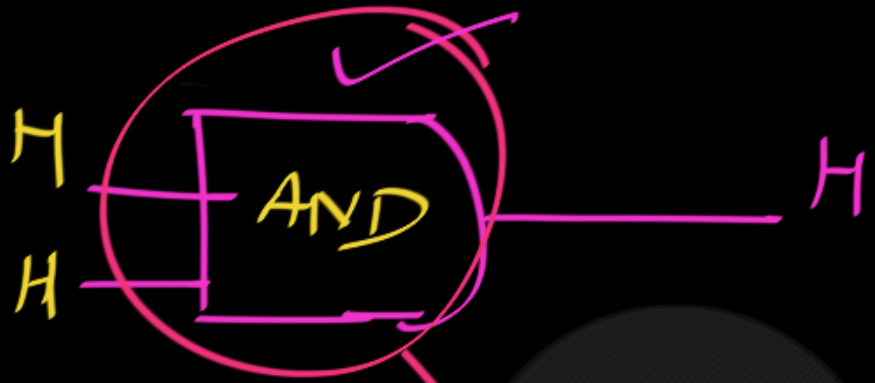
logic circuits

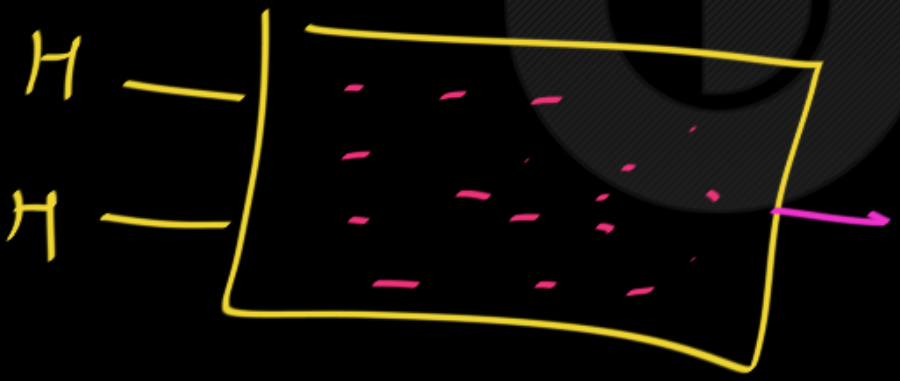


processor



memory





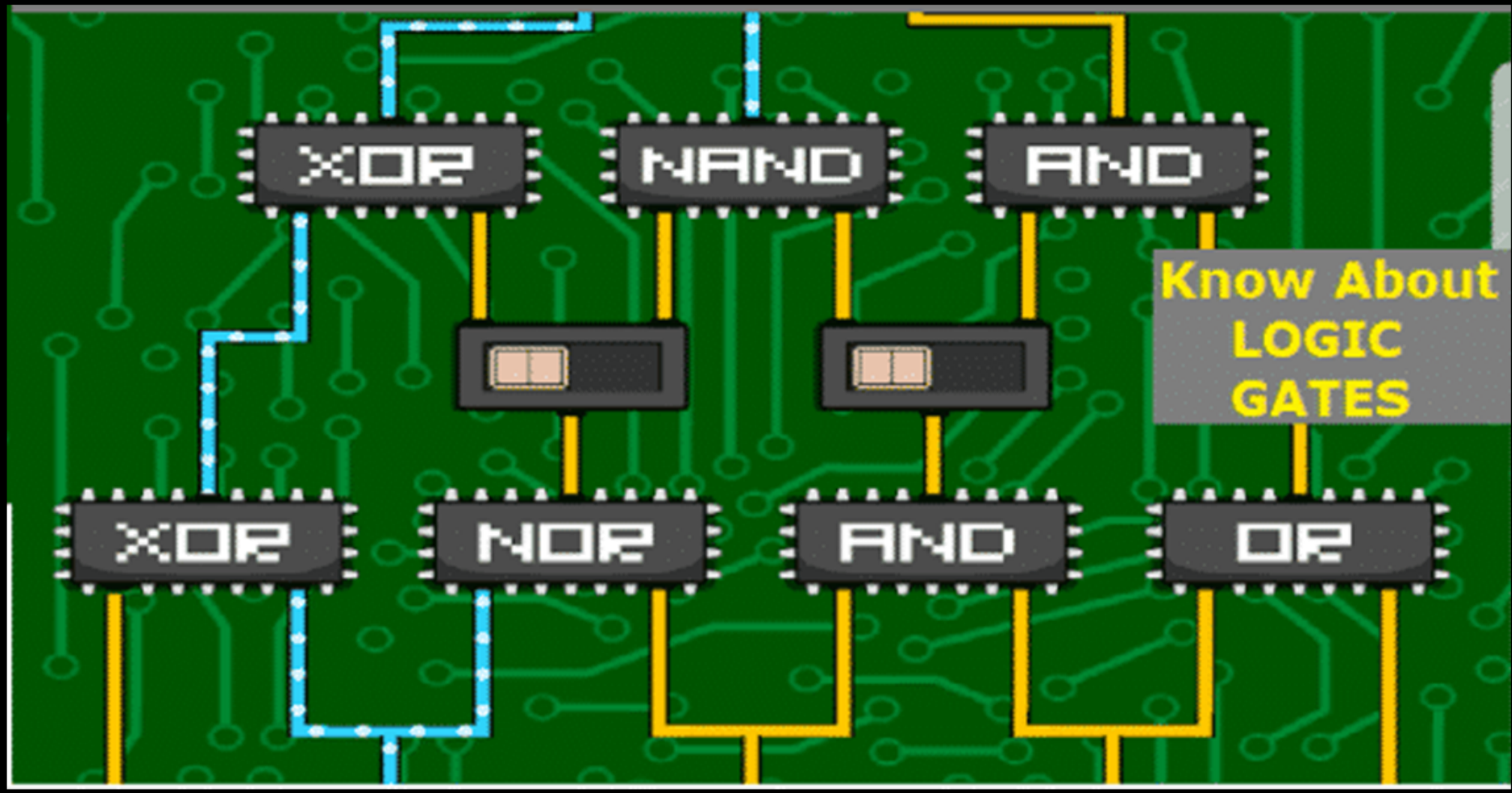
CS People

↓

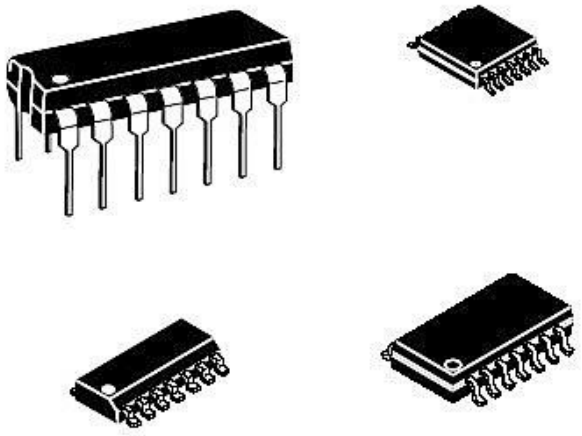
Semantics of Digital Circuits

↓

behaviour


XOR
NAND
AND
Know About
LOGIC
GATES
XOR
NOR
AND
OR

Logic gates actually look like weird bugs in real life!



However, the diagrams we use are easier to understand

What we will study ?

Semantics (behaviour) of Digital Circuits

Eg: AC

Digital Logic :

Next Topic :

Boolean Algebra

(Mathematical Model of Digital Circuits)

Website : https://www.goclasses.in/

Boolean Algebra → George Boole (1843)

↳ Two-valued system (Algebra)

↳ World of 1, 0

Inspired from

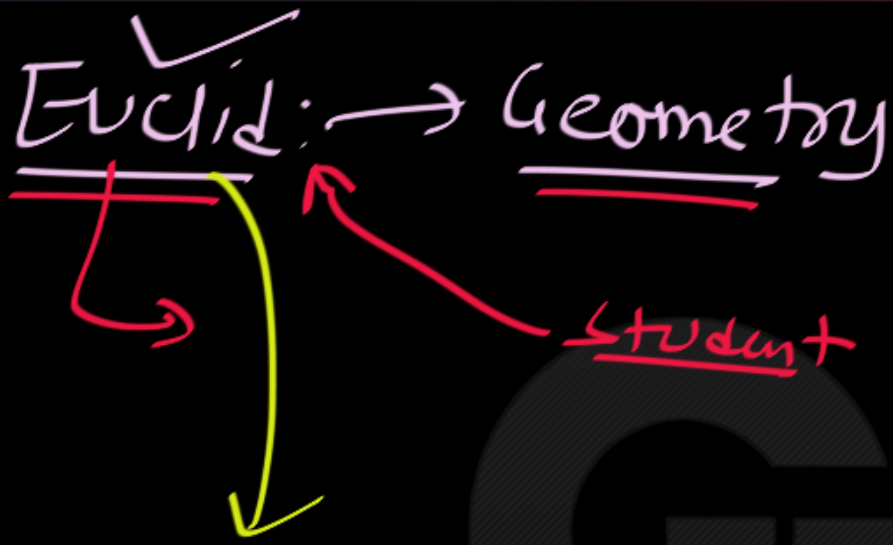
Prop. Logic → World of T, F

(Aristotal, Socrates)

Boolean Algebra → Concept
(1843)

After 100 years

Shehan → Used BA to model/Represent/Describe Digital systems.



Knowledge is worth acquiring for the sake of knowledge.

Boolean Algebra

In this unit you will study Boolean algebra, the basic mathematics needed for the logic design of digital systems.

In this unit you will study Boolean algebra, the basic mathematics needed for the logic design of digital systems.

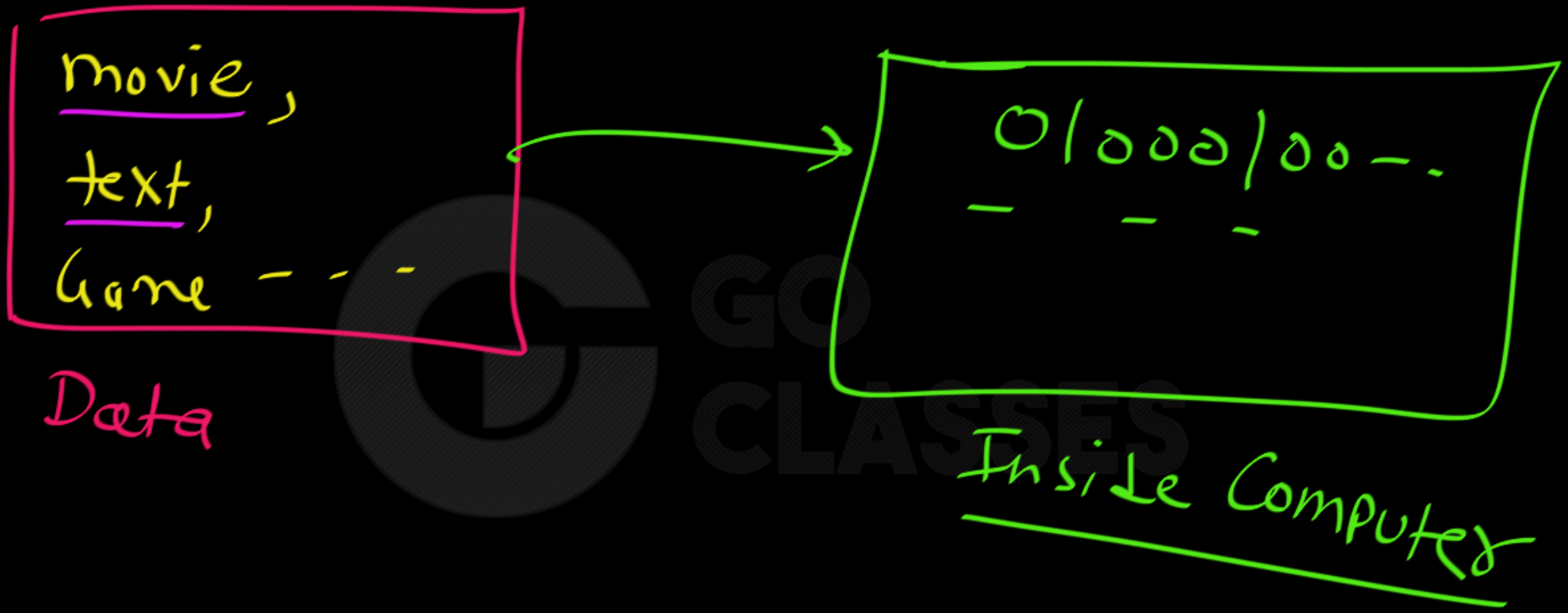
The basic mathematics needed for the study of logic design of digital systems is Boolean algebra.

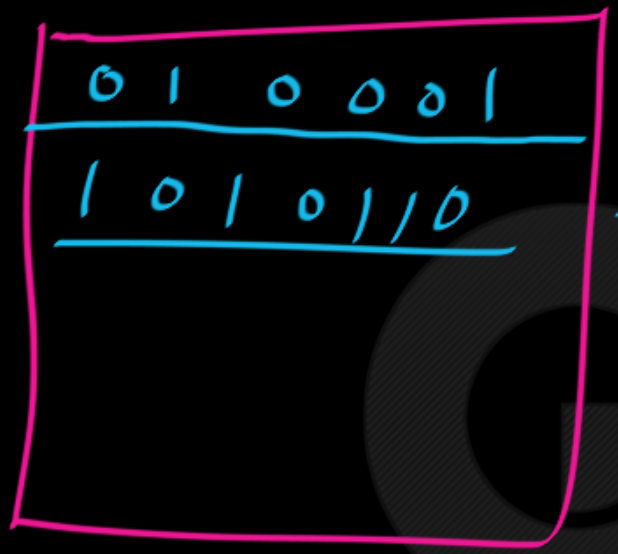
George Boole developed Boolean algebra in 1847 and used it to solve problems in mathematical logic. Boolean algebra has many other applications, including set theory and mathematical logic; however, we primarily consider its application to switching circuits.

Boolean Algebra:

2-Valued system (Algebra)

{0, 1}, OR, AND, Complement, ...

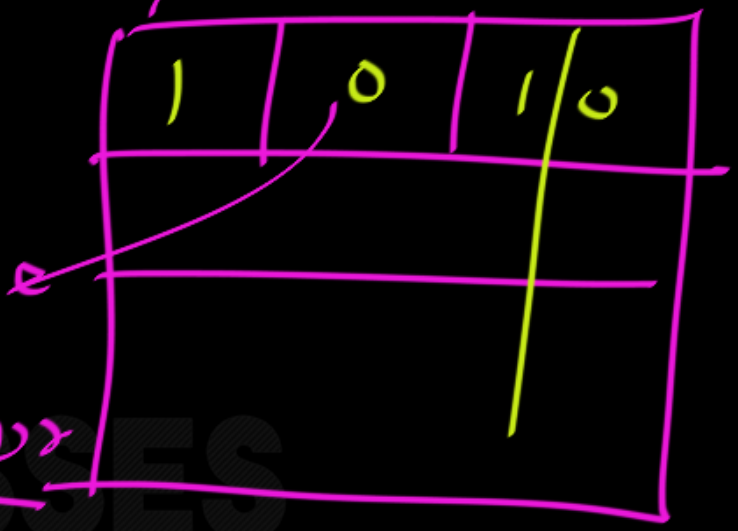

Movie,
text,
Game - - -
Data
01000100 - -
Inside Computer



Pen

Capacitor

Charge

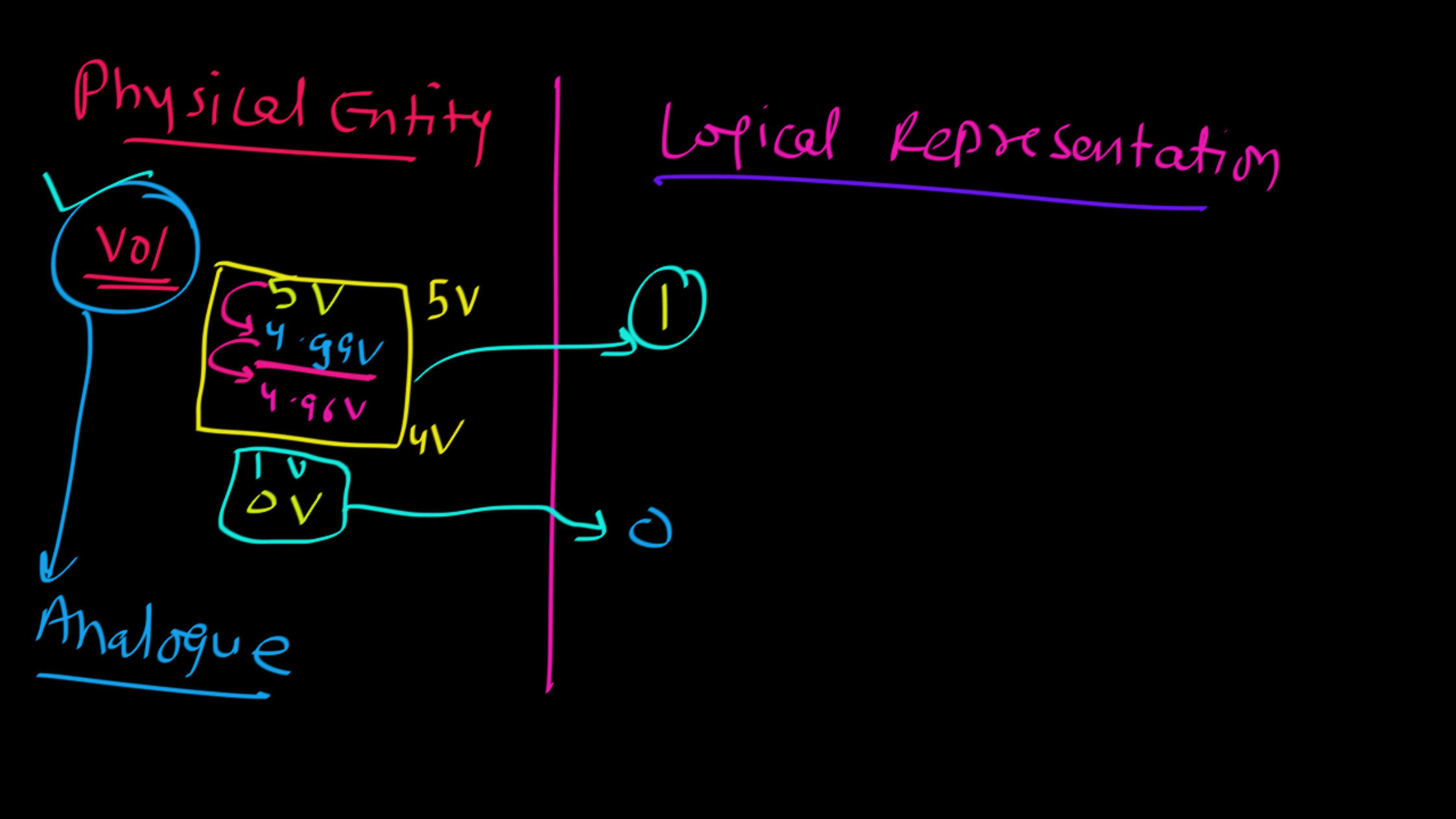


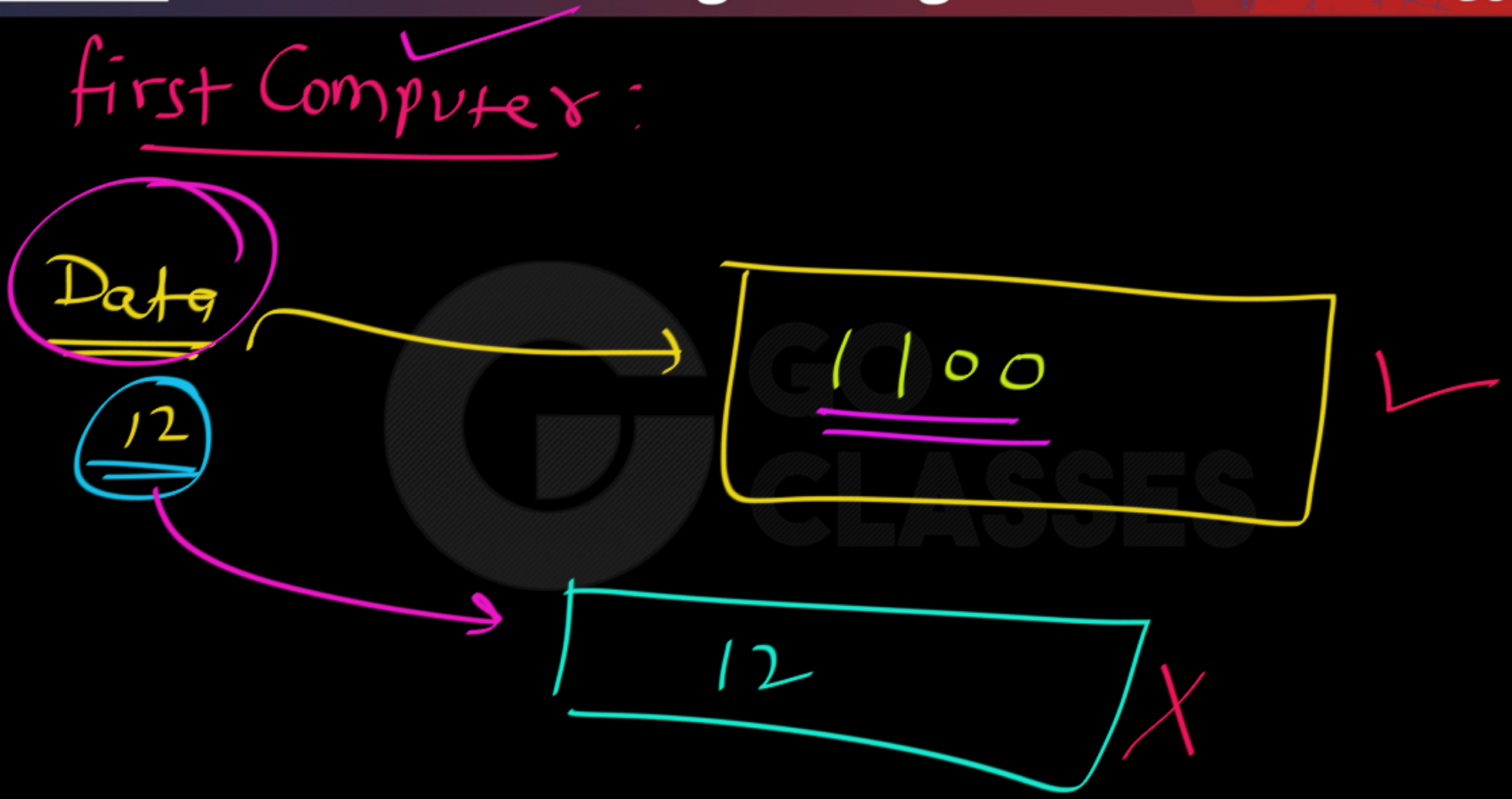
Capacitor

RA memory

RAM

RAM


Physical Entity
Logical Representation
Vol
5V
4.99V
4.96V
5V
4V
1V
0V
1
0
Analogue


first Computer :
Data
12
1100
L
12
X

Computer / Digital Ckt

lowest level

Transistors

Physical Device

On - off Device

Representation

1 0 — Human Interpretation

Actual things

A = 12

Compiler

A = 1100

Why Use Binary Numbers?

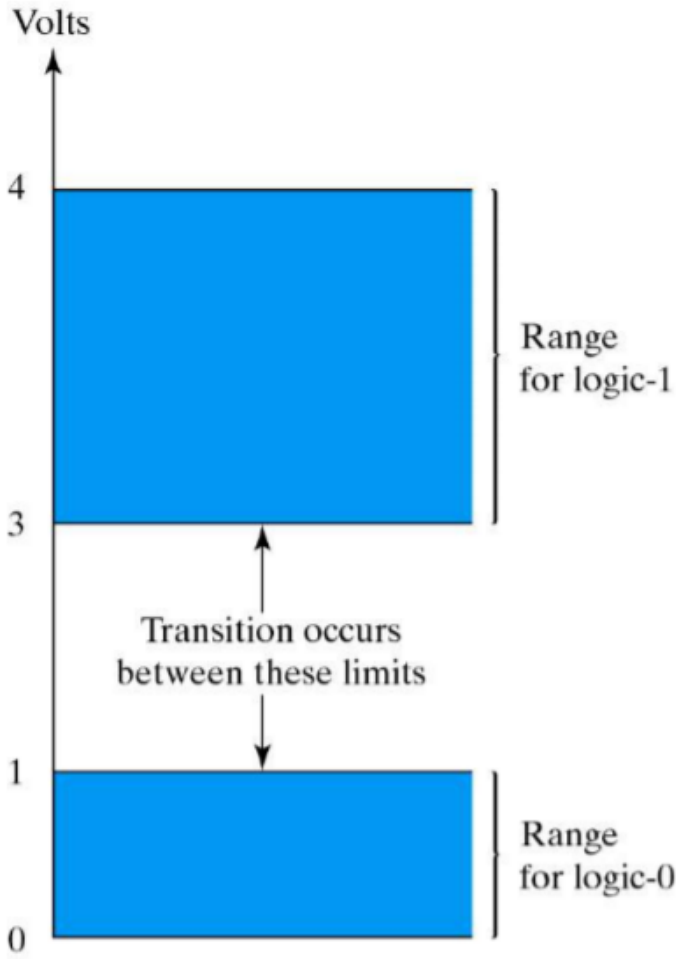
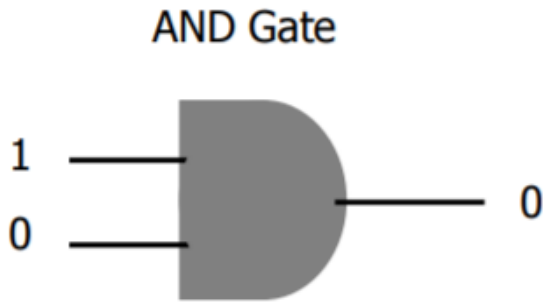


Fig. 1-3 Example of binary signals

- Easy to represent 0 and 1 using electrical values.
- Possible to tolerate noise.
- Easy to transmit data
- Easy to build binary circuits.



All of the switching devices we will use are essentially two state devices (e.g., switches which are open or closed and transistors with high or low output voltages).

Boolean algebra, in which all of the variables assume only one of two values; this two-valued Boolean algebra is often called switching algebra.

Claude Shannon first applied Boolean algebra to the design of switching circuits in 1939.

How do we represent data in a computer?

At the lowest level, a computer is an electronic machine.

- works by controlling the flow of electrons

Easy to recognize two conditions:

1. presence of a voltage – we'll call this state "1"
2. absence of a voltage – we'll call this state "0"

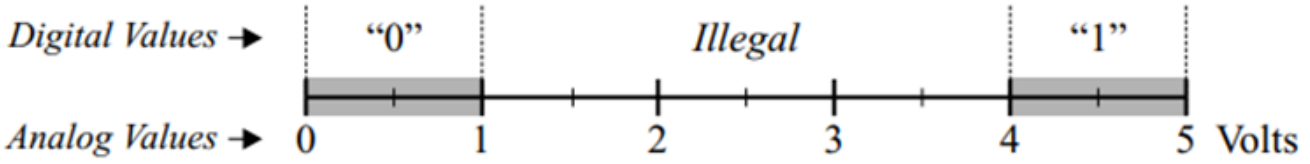
Computer is a binary digital system.

Digital system:

- finite number of symbols

Binary (base two) system:

- has two states: 0 and 1



Basic unit of information is the *binary digit*, or ***bit***.

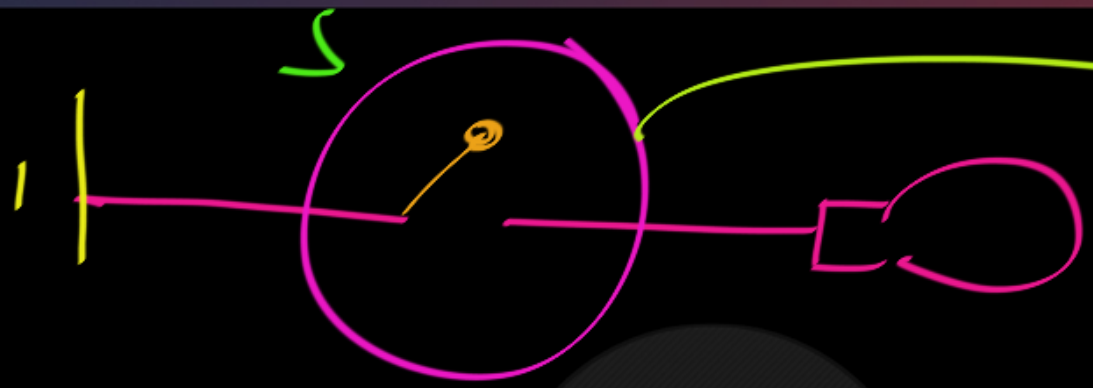
Values with more than two states require multiple bits.

- A collection of two bits has four possible states: 00, 01, 10, 11
- A collection of three bits has eight possible states:
- *A collection of **n** bits has **2^n** possible states.*

A Binary Switch

(a) Two states of a switch

$X = 0 \rightarrow$ switch open
$X = 1 \rightarrow$ switch closed



Switch

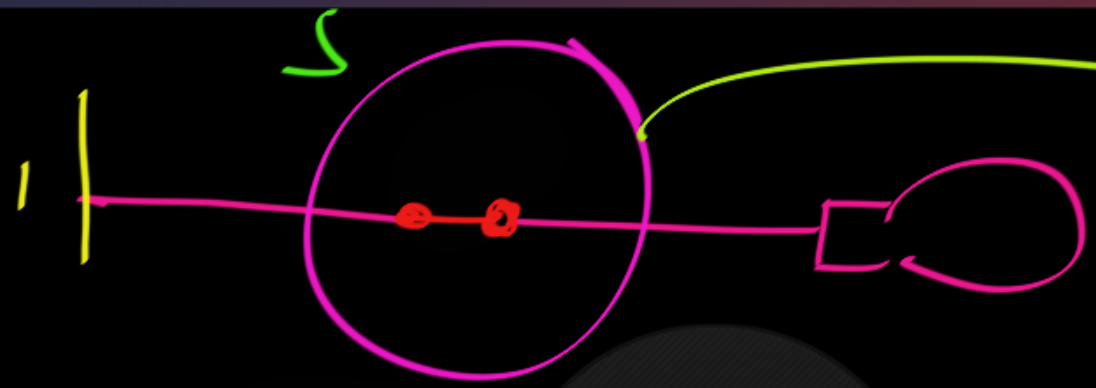
two states

open

closes

Open switch — Physical Entity

S = 0 — logically



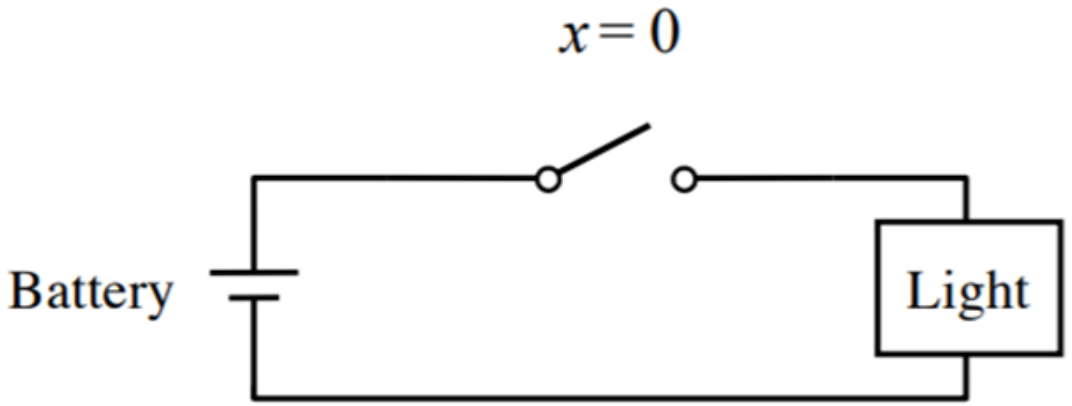
Switch — two states — open, closes

Closed switch — Physical Entity

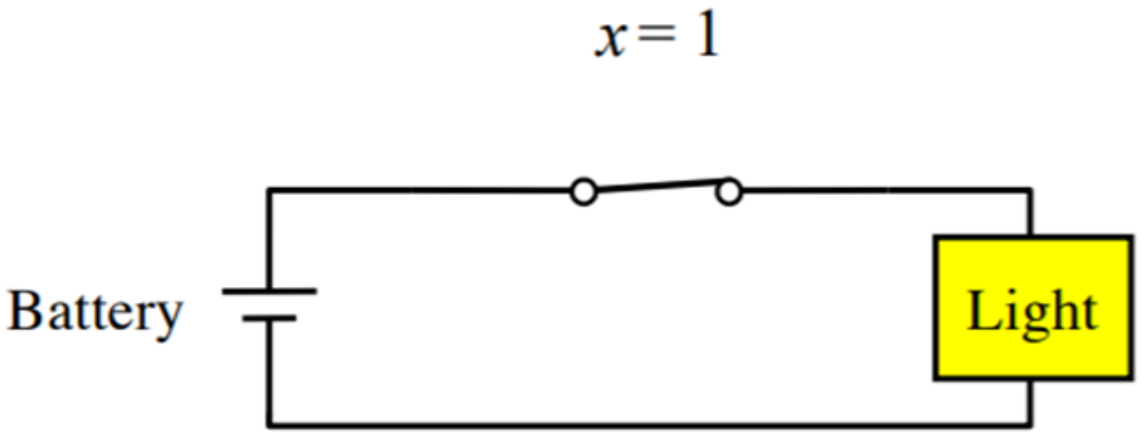
S = 1 — logically

S = 0, 1

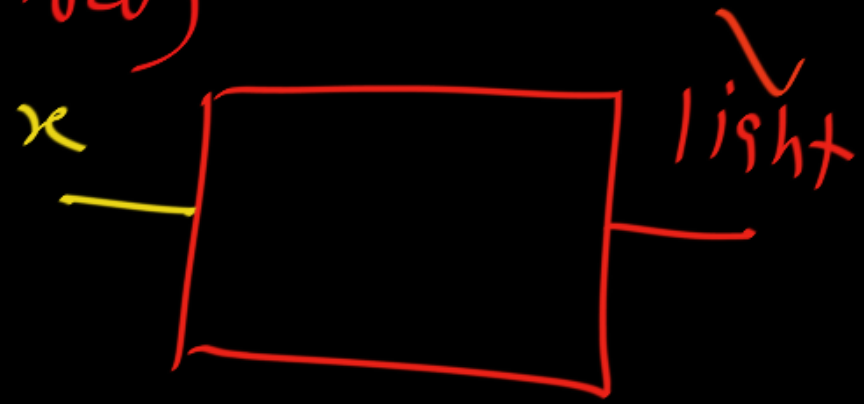
A Light Controlled by a Switch



A Light Controlled by a Switch

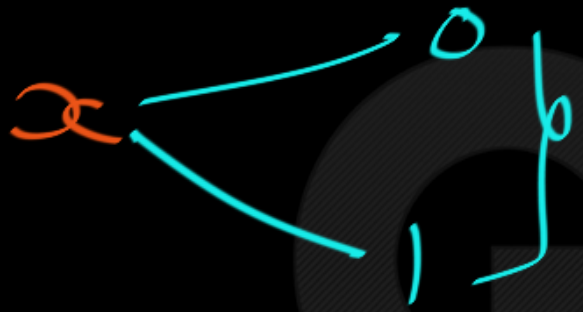


x	Light
0	0 (Not Glow)
1	1 (Glow)

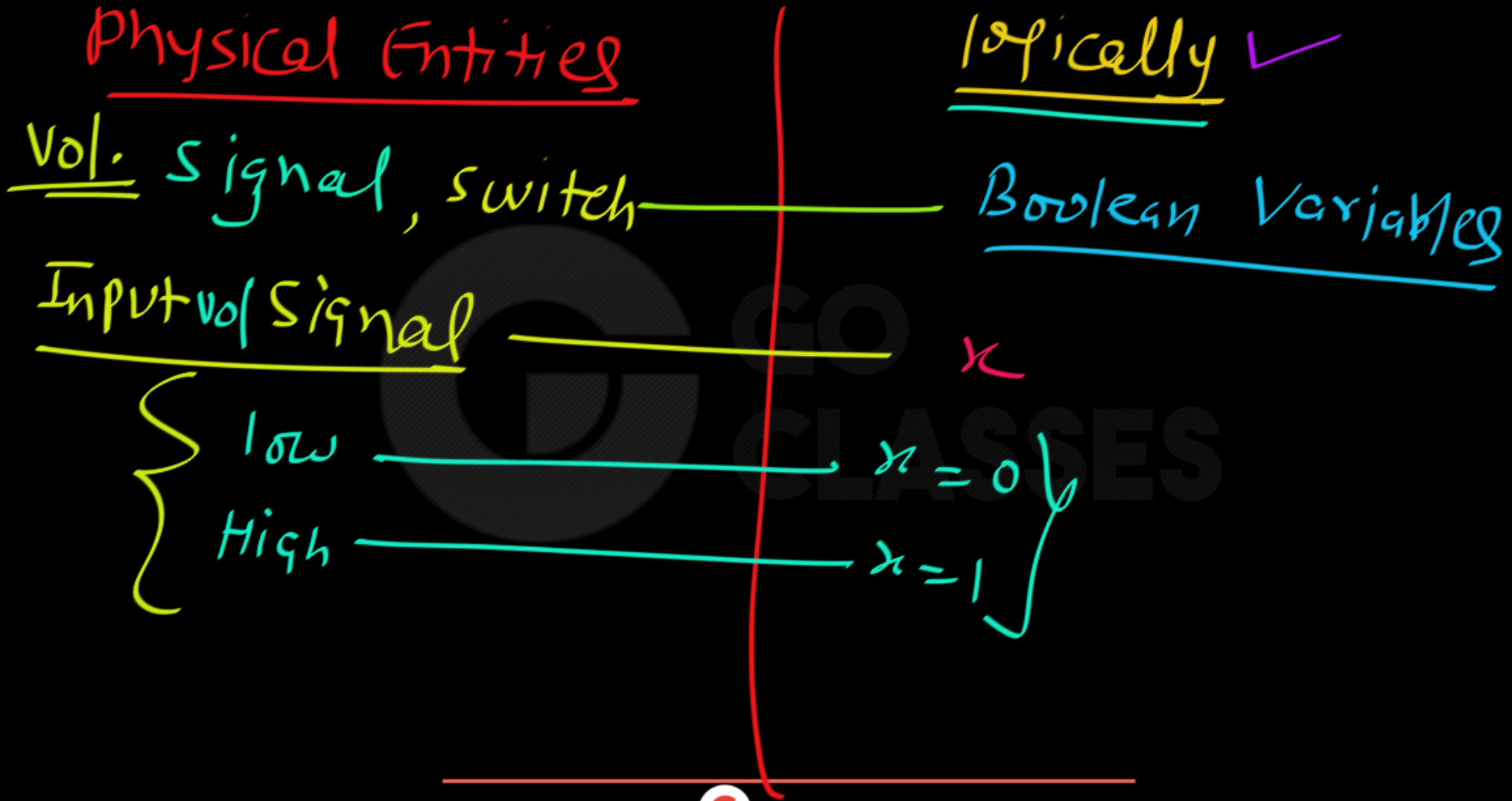


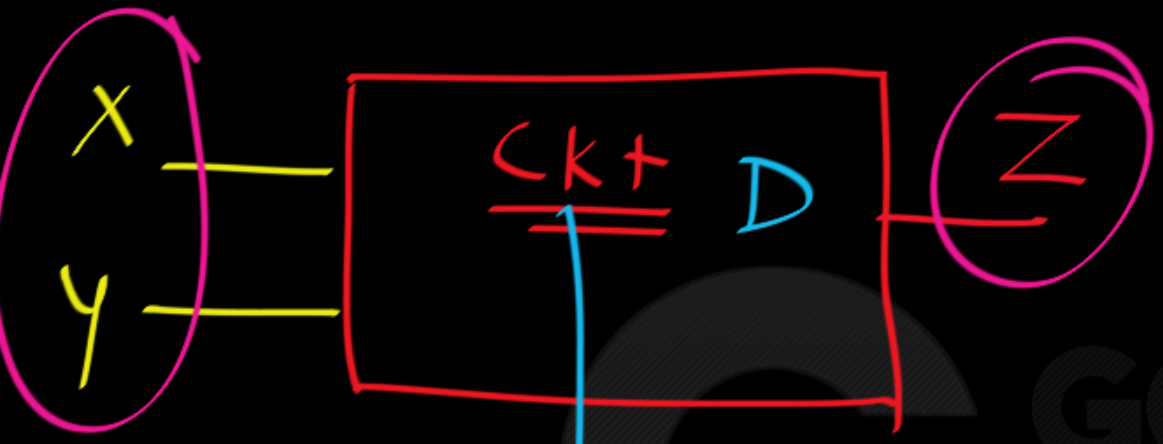
Boolean Variable

x, y, s, ---



We will use a Boolean variable, such as X or Y, to represent the input or output of a switching circuit. We will assume that each of these variables can take on only two different values. The symbols "0" and "1" are used to represent these two different values. Thus, if X is a Boolean (switching) variable, then either $X = 0$ or $X = 1$.


Physical Entities
Logically
Vol. signal, switch
Boolean Variables
Input Vol Signal
x
low
x = 0
High
x = 1



functionality

Semantics of Ckt D

X	Y	Z
H	H	H
L	H	L
H	L	L
L	H	L

Z = High iff both X, Y are high

X	Y	Z
1	1	1
0	1	0
1	0	0

1 Binary Vs Boolean

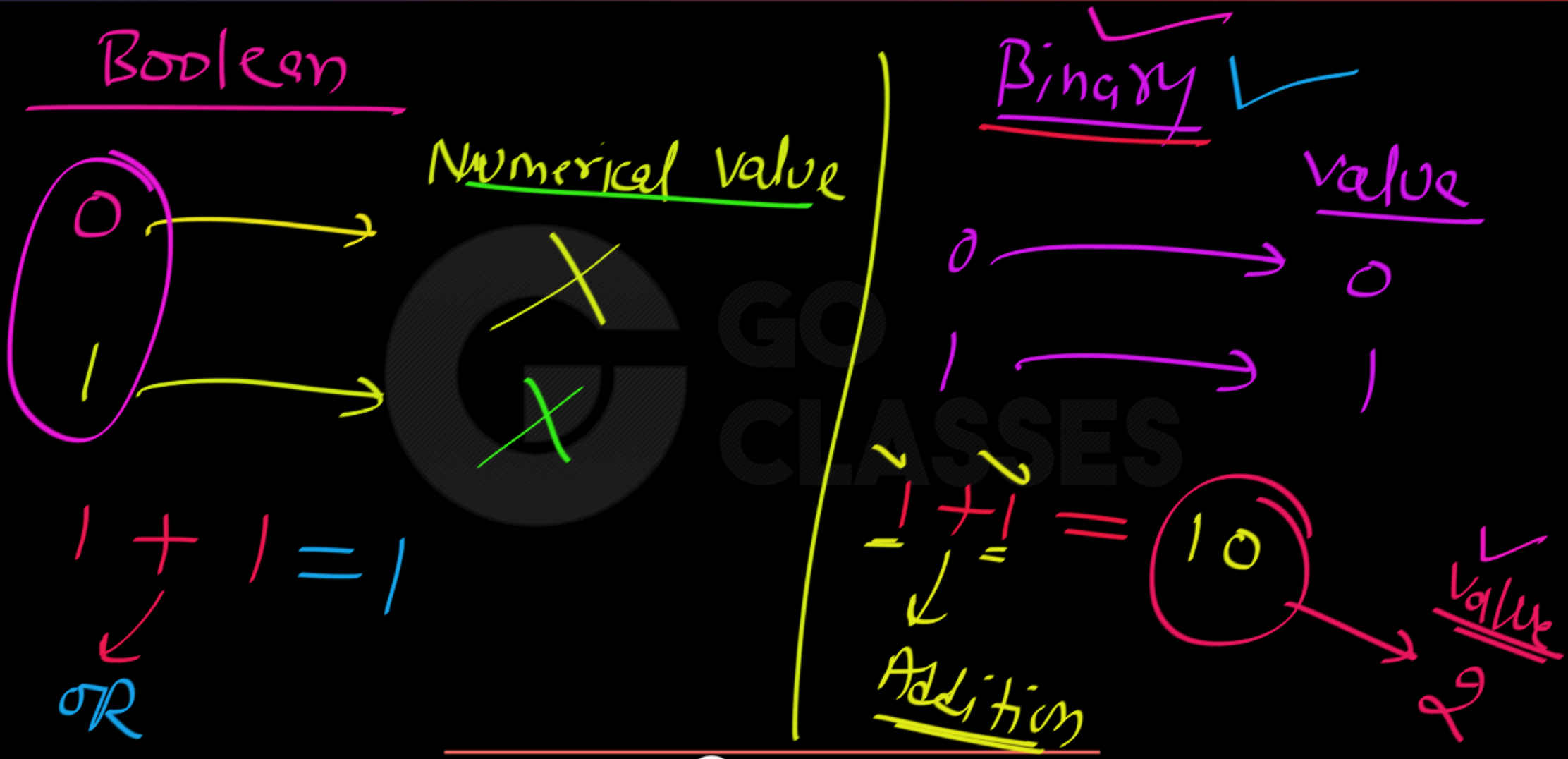
The symbols "0" and "1" used in Boolean algebra do not have a numeric value;
instead they represent two different states in a logic circuit and are the two values of a switching variable. In a logic gate circuit, 0 represents a range of low voltages, and 1 represents a range of high voltages. In a switch circuit, 0 represents an open switch, and 1 represents a closed circuit. In general, 0 and 1 can be used to represent the two states in any binary-valued system.

Boolean : 0 , 1

0	L open No
1	H Closes Yes

↳ Symbols to Represent two States of a Digital ckt

Binary → used to Represent a Numerical value.

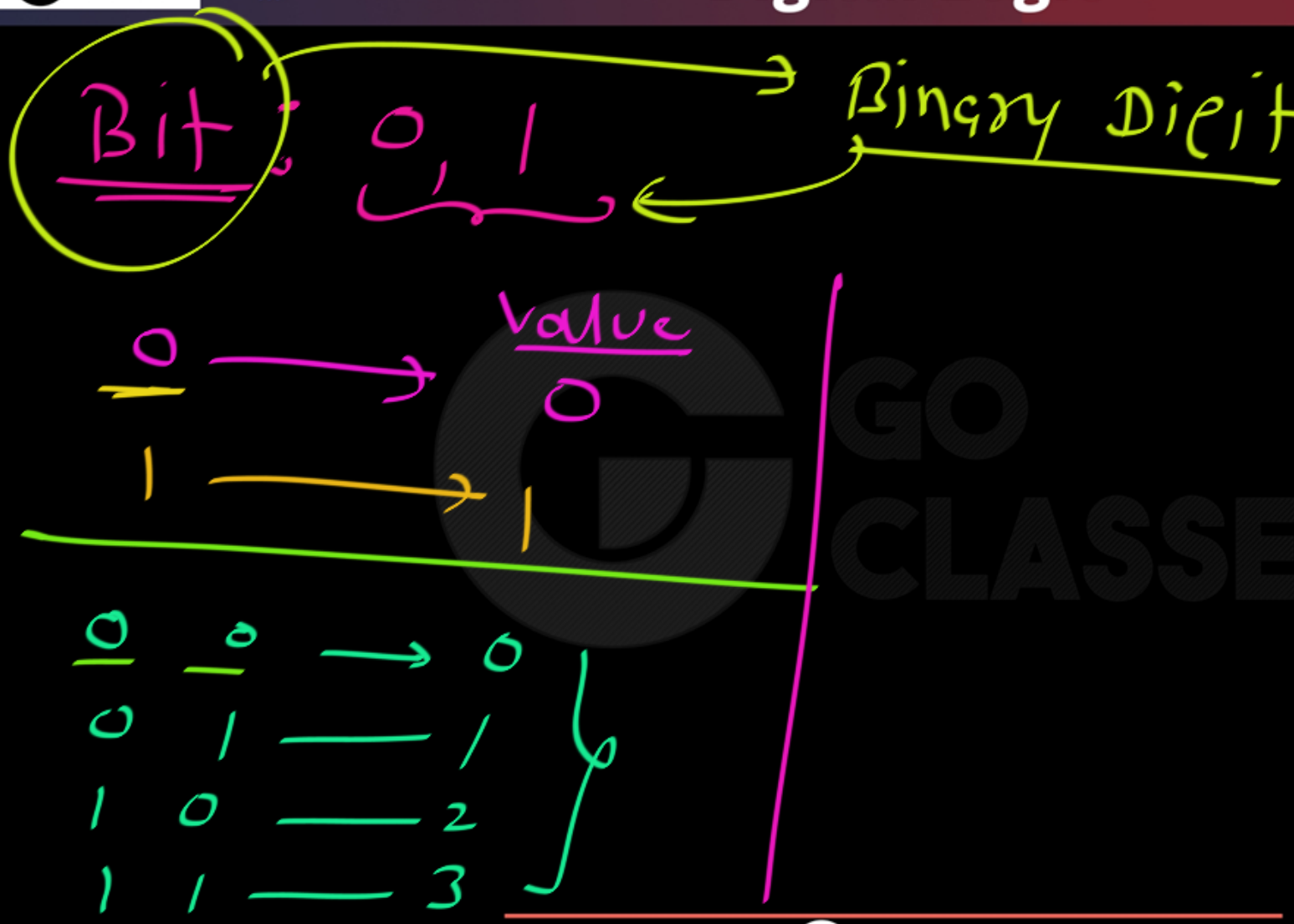

Boolean
0
1
Numerical value
1 + 1 = 1
OR
Binary
Value
0
0
1
1
1 + 1 = 10
Addition
Value
2

Value

Decimal system — 3

Binary system — 11

We will study in Number system.


Bit : 0, 1
Binary Digit
Value
0 → 0
1 → 1
0 0 → 0
0 1 → 1
1 0 → 2
1 1 → 3

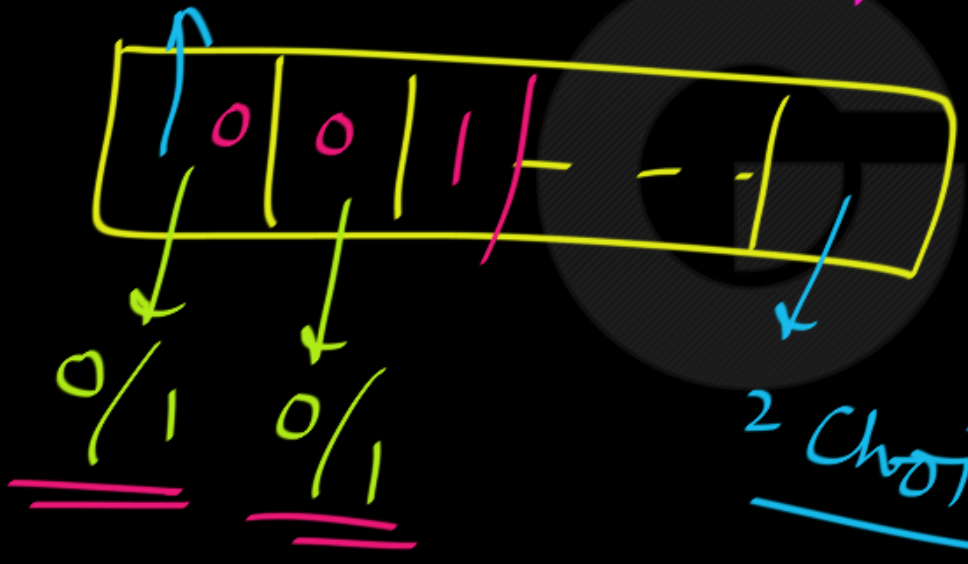
Bit Permutations

Decimal

1 bit	2 bits	3 bits	4 bits	
0 — 0	0 — 00	000 — 0	0000	1000
1 — 1	1 — 01	001 — 1	0001	1001
	2 — 10	010 — 2	0010	1010
	3 — 11	011 — 3	0011	1011
		100 — 4	0100	1100
		101 — 5	0101	1101
		110 — 6	0110	1110
		111 — 7	0111	1111

Each additional bit doubles the number of possible permutations

n bits → #Values that can be Represented = 2^n



Boolean Algebra : World of 0, 1

↳ 2 values Algebra ← {0, 1}

Every variable is a Boolean Variable.

x → 0 or 1

y → 0 or 1

0, 1

Boolean Algebra (2-valued Algebra)

$\{0, 1, x, y, z, \ldots, +, \cdot, \bar{x}, \oplus, \downarrow, \uparrow\}$

+ : OR
· : AND
$\bar{x}$: Complement

Boolean Algebra ≡ propositional logic

Boolean variable x, y	Propositional variable p, q
1	T
0	F
$\bar{x}$	$\bar{p}$
pq	$p \wedge q$
$p+q$	$p \vee q$

Basic Operations

The basic operations of Boolean (switching) algebra are called AND, OR, and complement (or not or inverse).

Complement / Inverse (Negation) :

$\overline{0} = 1$; $\overline{1} = 0$

Symbol: $\overline{x}$, x', $\sim x$, $\neg x$

x	$\overline{x}$
0	1
1	0

Truth Table (behaviour) of Complement operation.

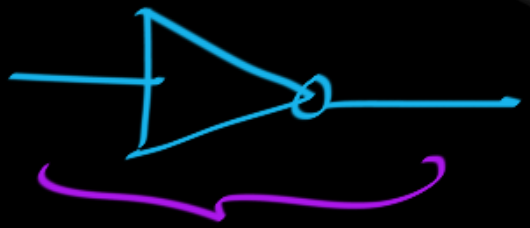
x	$y = \bar{x}$
H	L
L	H

Digital circuit

NOT GATE

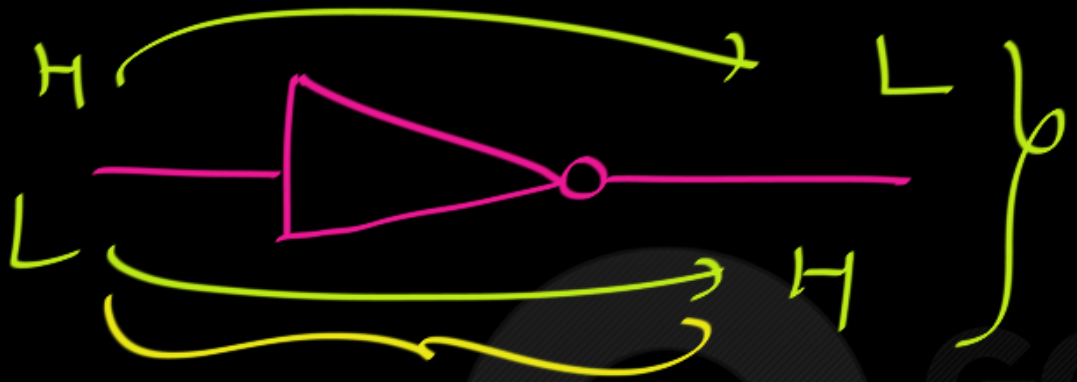
H → L

symbol

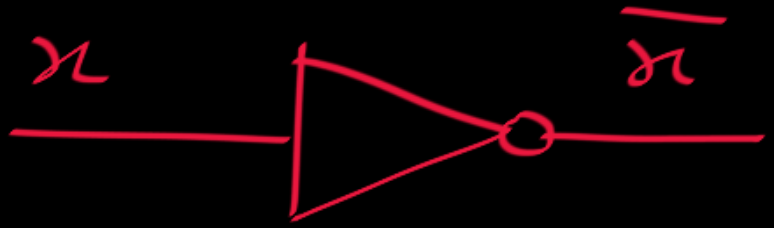


NOT GATE

L → H



NOT GATE / Inverter

$\overline{x} \equiv$ 

AND op^n:

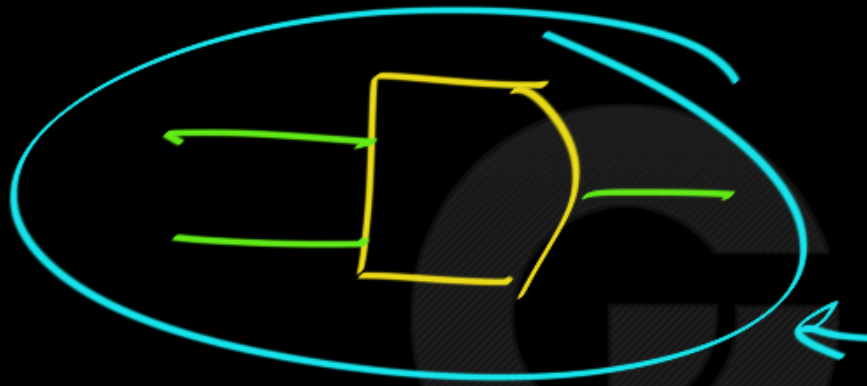
$x \cdot y \equiv xy$

$xy = 1$ iff both x, y are 1.

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

Definition of AND op^n

Digital Circuit: → AND GATE



Symbol

OR operation: $a + b$ → OR (plus)

a	b	a + b
0	0	0
0	1	1
1	0	1
1	1	1

$a + b = 0$ iff both a, b are 0

Digital ckt: → OR GATE

→ logical symbol

Boolean Logic

- Binary digits (or bits) have two values: {1,0}
- All logical functions can be implemented in terms of three logical operations:

x	$\bar{x}$
0	1
1	0

x	y	x·y
0	0	0
0	1	0
1	0	0
1	1	1

x	y	x + y
0	0	0
0	1	1
1	0	1
1	1	1

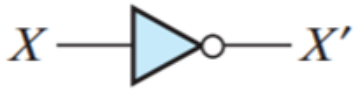
The complement of 0 is 1, and the complement of 1 is 0. Symbolically, we write

$$0' = 1 \quad \text{and} \quad 1' = 0$$

If X is a switching variable,

$$X' = 1 \text{ if } X = 0 \quad \text{and} \quad X' = 0 \text{ if } X = 1$$

An alternate name for complementation is inversion, and the electronic circuit which forms the inverse of X is referred to as an inverter. Symbolically, we represent an inverter by



where the circle at the output indicates inversion. A low voltage at the inverter input produces a high voltage at the output and vice versa.

In a general switch circuit, the value 0 is assigned to the connection between two terminals in the circuit if there is no connection (open circuit) between the terminals, and a 1 is assigned if there is a connection (closed circuit) between the terminals. If the switch circuit only contains two switches, the switch contacts must be connected in series or in parallel. When switch contacts A and B are connected in series, there is an open circuit between the terminals if either A or B or both are open (0), and there is a closed circuit between the terminals only if both A and B are closed (1).

$C = 0 \rightarrow$ open circuit between terminals 1 and 2
$C = 1 \rightarrow$ closed circuit between terminals 1 and 2

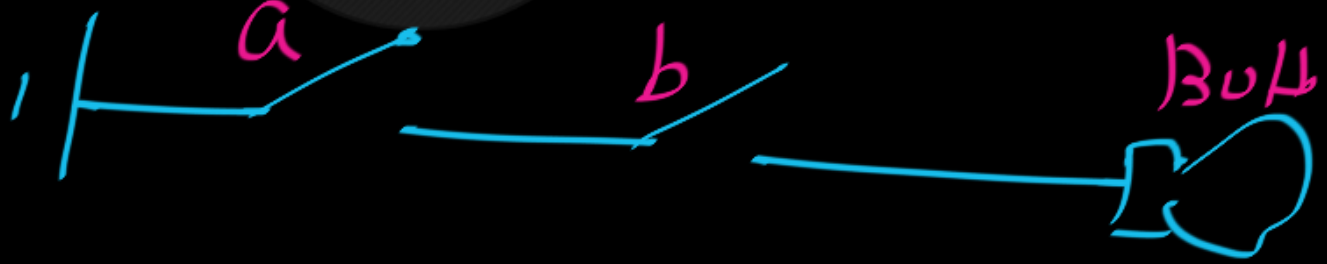
This is summarized in the following truth table:

A B	$C = A \cdot B$
0 0	0
0 1	0
1 0	0
1 1	1

The operation defined by the table is called AND and it is written algebraically as $C = A \cdot B$. The "$\cdot$" symbol is frequently omitted in a Boolean expression, and we will usually write AB instead of $A \cdot B$. The AND operation is also referred to as logical (or Boolean) multiplication.

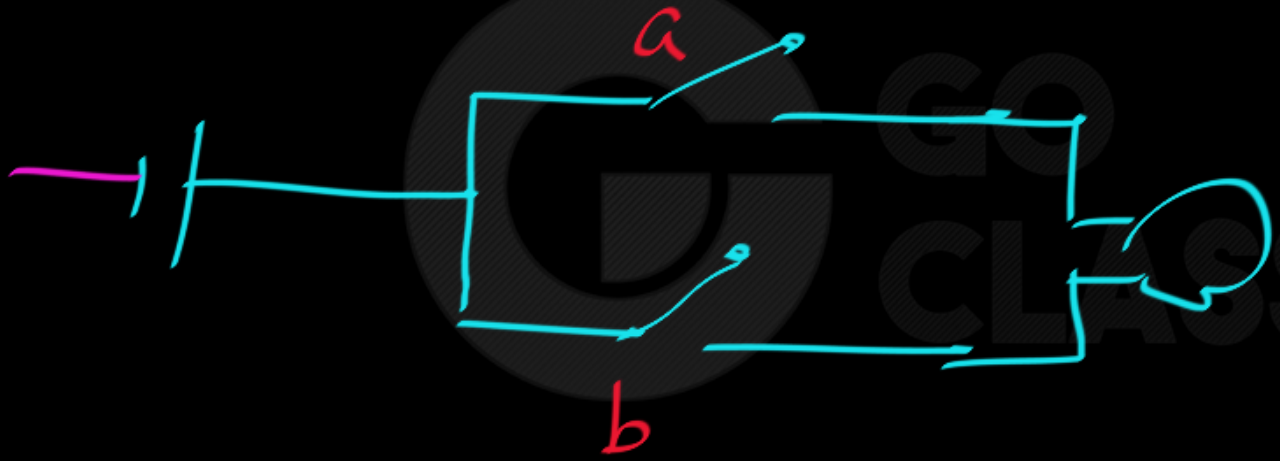
AND functionality using switches :

o/p = High Vol iff both inputs are High Vol.



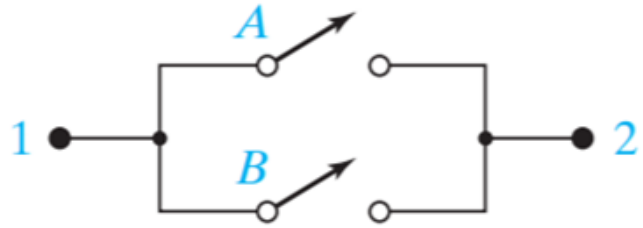
a	b	Bulb
1	1	1
0	1	0
1	0	0
0	0	0

OR function using switches :



a	b	Bulb
0	0	0
0	1	1
1	0	1
1	1	1

When switches A and B are connected in parallel, there is a closed circuit between the terminals if either A or B is closed (1), and there is an open circuit between the terminals only if both A and B are open (0).

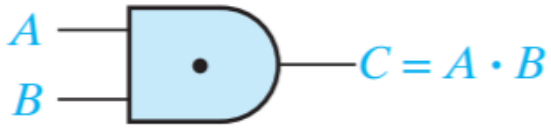


This is summarized in the following truth table:

A B	$C = A + B$
0 0	0
0 1	1
1 0	1
1 1	1

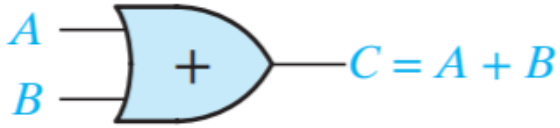
The operation defined by the table is called OR and it is written algebraically as $C = A + B$. This type of OR operation is sometimes referred to as inclusive OR as opposed to exclusive OR, which is defined later. The OR operation is also referred to as logical (or Boolean) addition.

A logic gate which performs the AND operation is represented by



The gate output is $C = 1$ if and only if the gate inputs $A = 1$ and $B = 1$.

A logic gate which performs the OR operation is represented by



The gate output is $C = 1$ if and only if the gate inputs $A = 1$ or $B = 1$ (or both).

Boolean Algebra :

Next Topic :

Boolean Expressions and Truth Table

Boolean variable: $x, y, z, \ldots$ $x \begin{cases} 0 \\ 1 \end{cases}$

Boolean Expression:

$0, 1, x, y, z, x+y, x \cdot y, \bar{x}, (x+y)z, x+(yz), \ldots$

Boolean Expression:

1. 0, 1
2. Boolean variable x, y, z, ...
3. If E_1, E_2 are boolean expressions:

$\overline{E_1}$, $\overline{E_2}$, $E_1 + E_2$, $E_1 \cdot E_2$, (E_1), ...

Boolean Expression:

1. 0, 1
2. Boolean Variable $x, y, z, \ldots$
3. If E_1, E_2 are boolean Expressions:

$\overline{E_1}$, $\overline{E_2}$, $E_1 + E_2$, $E_1 \cdot E_2$, (E_1) ...

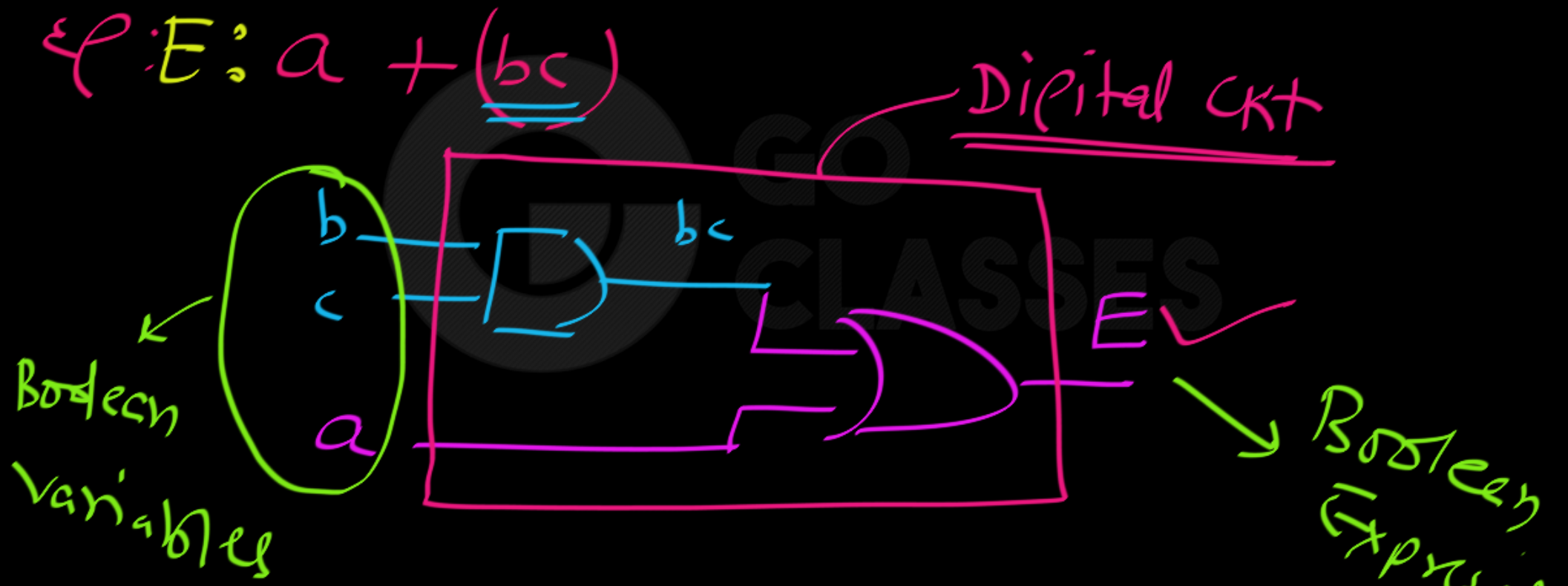
$\overline{x}$, $x + y$, xy, $xy + z$

Boolean expressions are formed by application of the basic operations to one or more variables or constants. The simplest expressions consist of a single constant or variable, such as 0, X, or Y'. More complicated expressions are formed by combining two or more other expressions using AND or OR, or by complementing another expression. Examples of expressions are

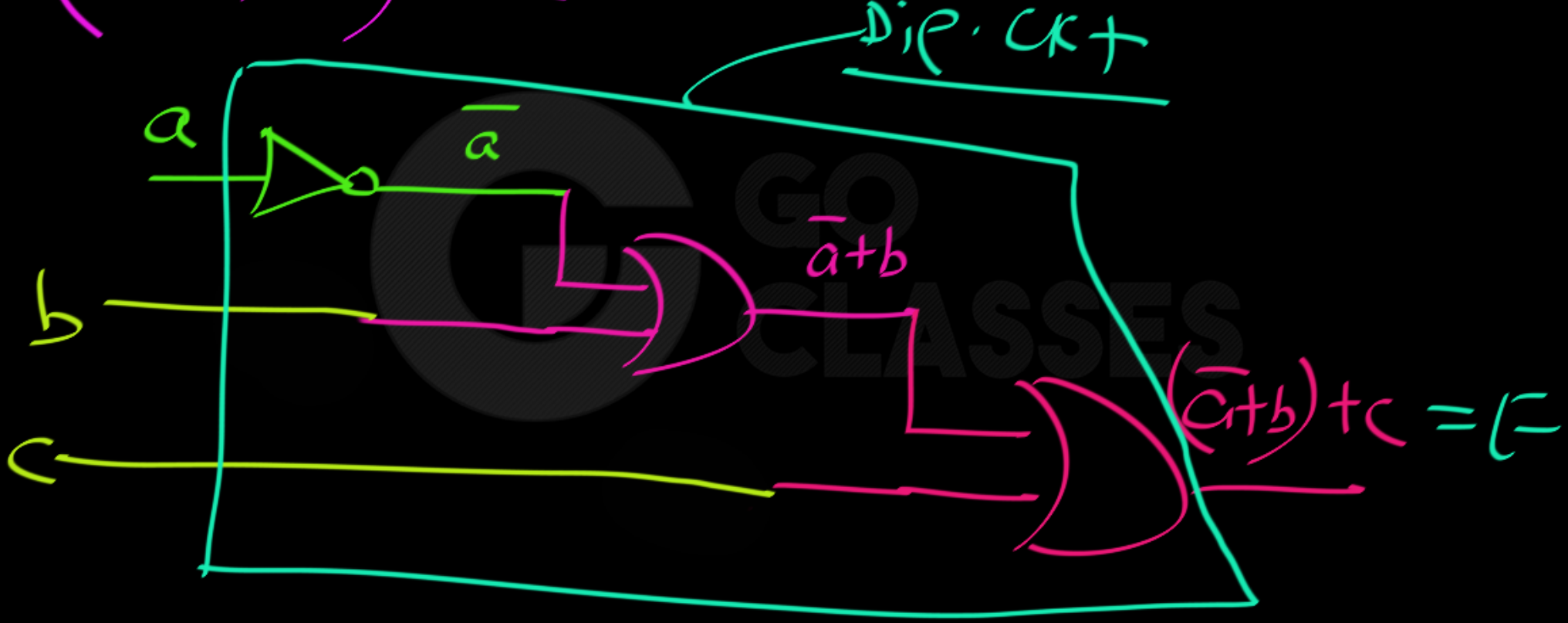
$$AB' + C \tag{2-1}$$
$$[A(C + D)]' + BE \tag{2-2}$$

Parentheses are added as needed to specify the order in which the operations are performed. When parentheses are omitted, complementation is performed first followed by AND and then OR. Thus in Expression (2-1), B' is formed first, then AB', and finally $AB' + C$.

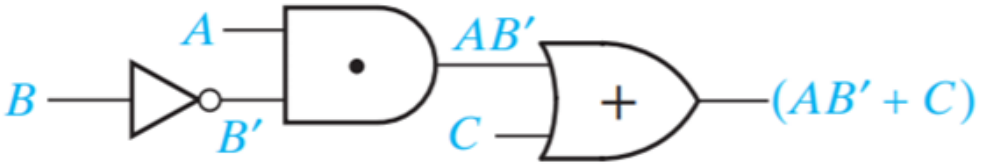
Each expression corresponds directly to a circuit of logic gates.



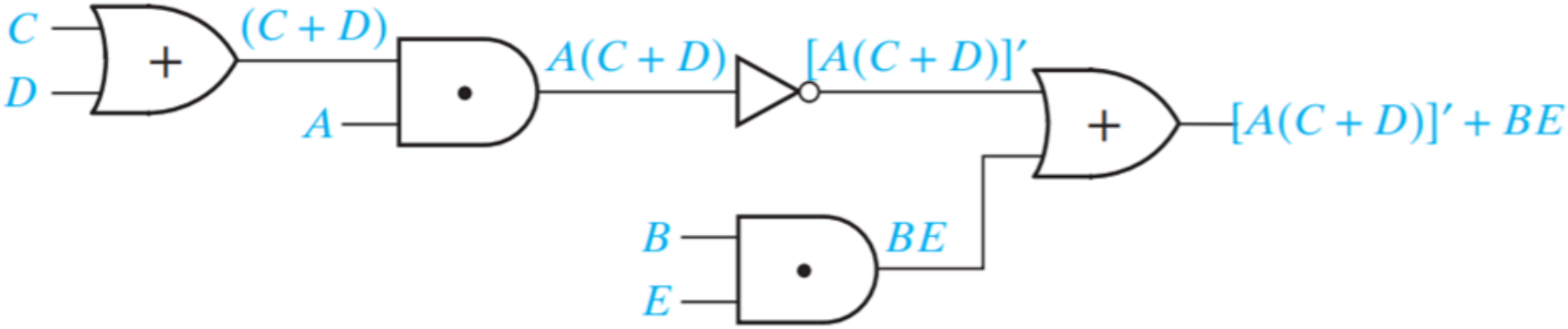
$E = (\overline{a} + b) + c$



Each expression corresponds directly to a circuit of logic gates. Figure 2-1 gives the circuits for Expressions (2-1) and (2-2).



(a)



(b)

An expression is evaluated by substituting a value of 0 or 1 for each variable. If $A = B = C = 1$ and $D = E = 0$, the value of Expression (2-2) is

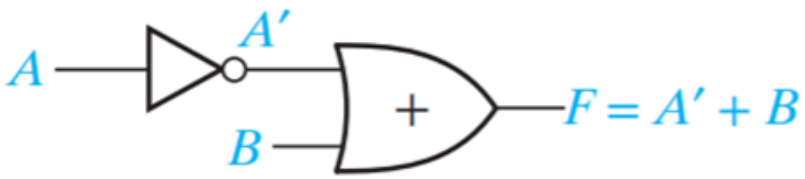
$$[A(C + D)]' + BE = [1(1 + 0)]' + 1 \cdot 0 = [1(1)]' + 0 = 0 + 0 = 0$$

Each appearance of a variable or its complement in an expression will be referred to as a *literal*. Thus, the following expression, which has three variables, has 10 literals:

$$ab'c + a'b + a'bc' + b'c'$$

Truth Table

A truth table (also called a table of combinations) specifies the values of a Boolean expression for every possible combination of values of the variables in the expression.



(a)

(b)

A	B	A′	F = A′ + B
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	1

$E: a + \bar{b}$

a	b	$a + \bar{b} = E$
0	0	1
0	1	0
1	0	1
1	1	1

$0 + 1 = 1$

$0 + \bar{1} = 0 + 0 = 0$

$1 + \bar{0} = 1 + 1 = 1$

$1 + \bar{1} = 1 + 0 = 1$

$E : (\bar{a}\,b) + c$

$(\bar{0}\,0) + 1 = 1$

$(\bar{0} \cdot 1) + 0 =$

$1 \cdot 1 + 0$

$= 1 + 0 = 1$

a	b	c	$(\bar{a}\,b) + c = E$
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Precedence of operators:

NOT > AND > OR

$a + b.c$

$(a+b)c$ ✗

$a + (bc)$

$$(a + b)c =$$

$$a + bc = a + (bc)$$

Boolean Logic 2

- Precedence rules just like decimal system
- Implied precedence: NOT > AND > OR
- Use parentheses as necessary

$$AB + C = (AB) + C$$

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

Boolean Logic: Example

D	X	A	$L=D\overline{X}+A$
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Boolean Logic: Example

D	X	A	$\overline{X}$	$D\overline{X}$	$L=D\overline{X} + A$
0	0	0	1	0	0
0	0	1	1	0	1
0	1	0	0	0	0
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	0	0	0
1	1	1	0	0	1

Boolean Logic: Example 2

X	Y	$XY + \overline{X}\overline{Y}$
0	0	
0	1	
1	0	
1	1	

Boolean Logic: Example 2

X	Y	$\overline{X}$	$\overline{Y}$	XY	$\overline{XY}$	$XY + \overline{XY}$
0	0	1	1	0	1	1
0	1	1	0	0	0	0
1	0	0	1	0	0	0
1	1	0	0	1	0	1