Boolean Algebra – Introduction

So far you have learned about the computer numbering system (CNS)

In this section, we will explore

- Basic building blocks of logic systems
- AND, OR, NOT functions
- Relate logic diagrams and logic expressions
- Simplify logic circuits and verify it using truth tables

Boolean algebra – branch of algebra where the values of the variables are truth variables (true/false, 1/0...) instead of real numbers

Boolean Algebra – Background

George Boole (1815-1864), English mathematician, *An Investigation of the Laws of Thought*, (1854)

George Boole's work is based on the logic concepts of Aristotle (384-322 BC). Translated the logic concepts into mathematical equations.

Aristotle first wrote his concepts on deductive reasoning in *Prior Analytics*, (350 BC). This work was 1 part of a 6 part series on logic that was later complied in 40 BC, called *Organon*.

Claude Shannon (1916-2001), American mathematician and electrical engineer, who took the concepts of Boolean algebra and created circuits that could solve mathematical operations, *A Symbolic Analysis of Relay and Switching Circuits*, (1938).

Algebra, study of mathematical symbols and how to manipulate such symbols. It binds together all branches of mathematics. First documented in Babylon (~1000 BC). Other civilizations (Egypt, China, India, Greece, Americas...) used geometrical means to solve mathematical problems...

Logic Variables

Use Boolean algebra to manipulate logic variables

Logic variable is either True or False...it CANNOT be partly True or False

If the variable is NOT false, then it must be true

Boolean algebra is best suited to variables that have 2 states... T/F, Y/N, 1/0, On/Off…

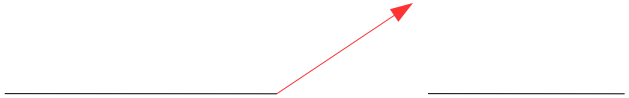
Terms that are synonyms

True	T	Yes	On	Closed	High	1	5 V
False	F	No	Off	Open	Low	0	0 V

Logic Variables

A switch can be used to describe the value of any 2 state variable (B) because it can only be either “off” or “on”...and not in between

B = true



B = false

Logic Operations

There are 3 basic logic operations

1. Conjunction, logical product, AND (•)
2. Disjunction, logical sum, OR (+)
3. Negation, NOT, ($\bar{}$)

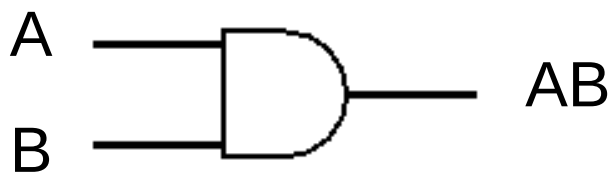
All functions within a computer can be performed by a combination of the 3 basic logic operations

Precedence order for logic operations follows as
1st NOT
2nd AND
3rd OR

Expressions inside ( ) will be evaluated first, overriding the precedence order

AND Function

For the AND function, the output is true only when ALL the inputs are true...otherwise the output is false



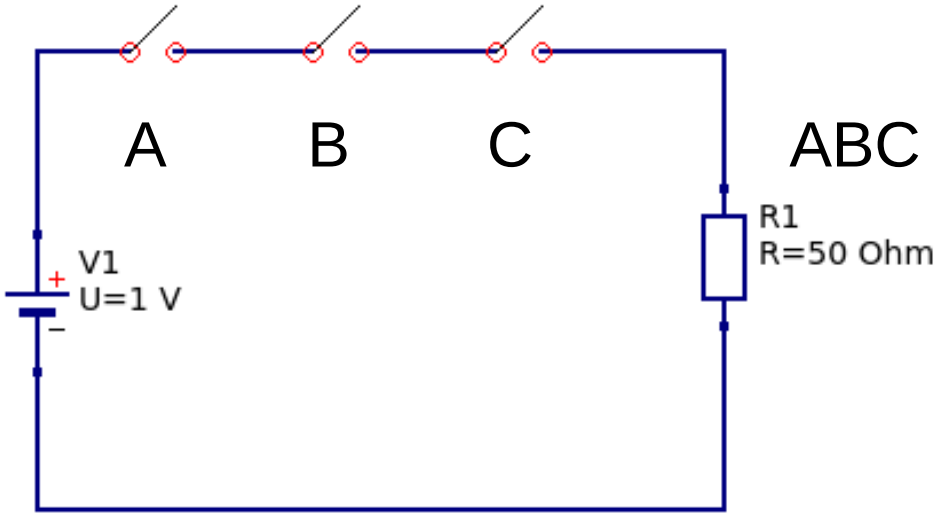
Condition	A	B	AB
1	F	F	F
2	F	T	F
3	T	F	F
4	T	T	T

T	On	Closed	1
F	Off	Open	0

- AB is true if and only if A and B are both true at the same time
- If either A or B is false then AB will be false
- Truth table is useful in showing all the possible conditions that may exist

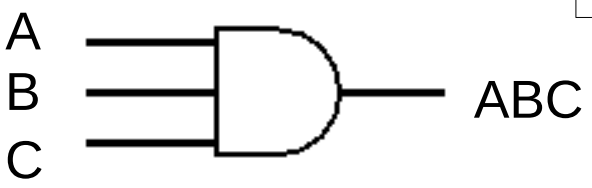
AND Function

In the circuit diagram, if any of the switches, A, B, C is open (false) then the bulb will not light



T	On	Closed	1
F	Off	Open	0

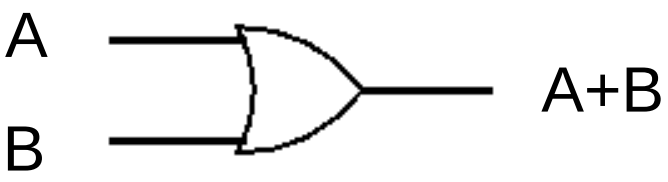
Condition	A	B	C	ABC
1	0	0	0	0
2	0	0	1	0
3	0	1	0	0
4	0	1	1	0
5	1	0	0	0
6	1	0	1	0
7	1	1	0	0
8	1	1	1	1



Bulb will only light, if all 3 switches are closed (true) at the same time

OR Function

For the OR function, output is true when any one of the inputs is true…otherwise output is false



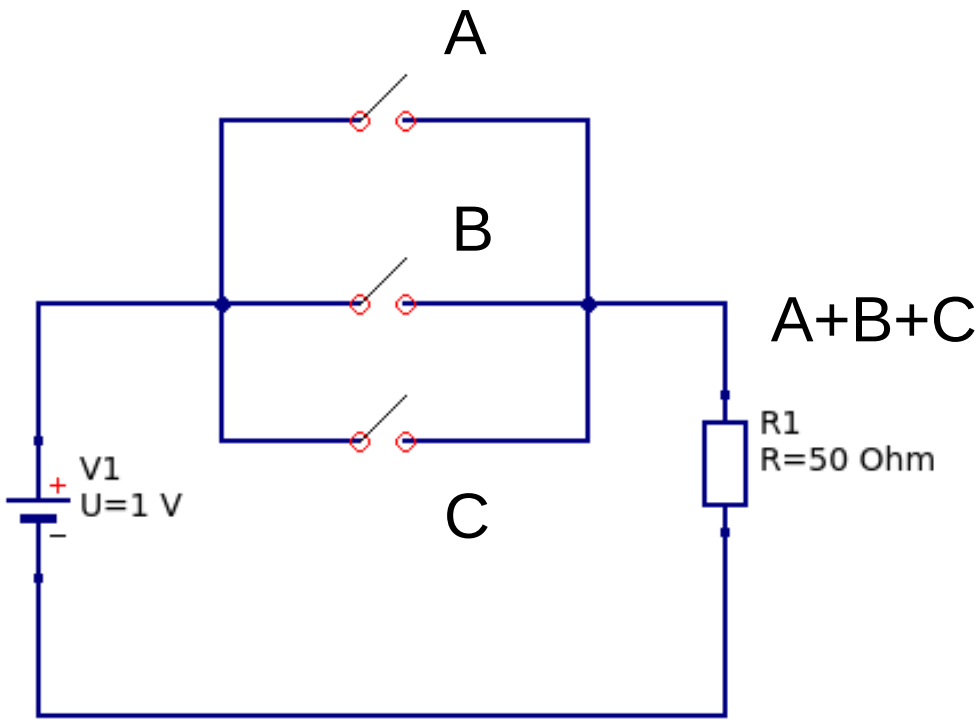
Condition	A	B	A+B
1	F	F	F
2	F	T	T
3	T	F	T
4	T	T	T

T	On	Closed	1
F	Off	Open	0

If either input is true then the output will be true
If both inputs is false then the output will be false

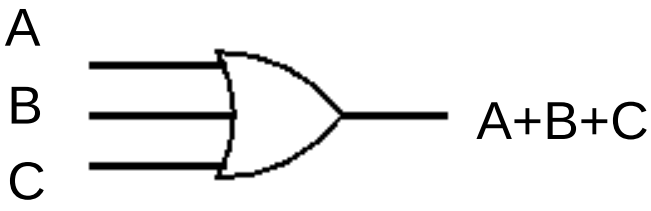
OR Function

In the circuit diagram, if any of the switches A, B, C is closed (true) then the bulb will light



Condition	A	B	C	A+B+C
1	0	0	0	0
2	0	0	1	1
3	0	1	0	1
4	0	1	1	1
5	1	0	0	1
6	1	0	1	1
7	1	1	0	1
8	1	1	1	1

T	On	Closed	1
F	Off	Open	0



Bulb will not light, if all 3 switches are open (false) at the same time

NOT Function

For the NOT function, the output is the complement of the input

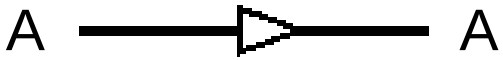
A	Ā
1	0
0	1

Logic circuit producing a NOT function is called an inverter

Inverter converts the state to its complement

Buffer Gate

For the buffer gate, the output is the exact same as the input



A	A
1	1
0	0

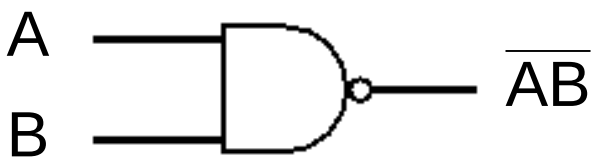
The buffer gate can be thought of as 2 NOT gates in series.

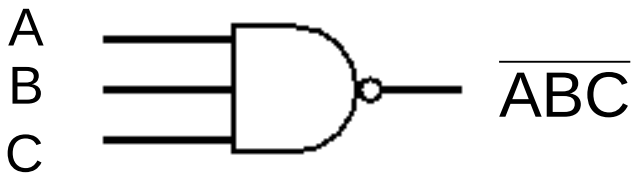
Some uses for a buffer gate

- Signal repeater...if digital signal is to be sent longer distance... possible distortion, noise...better to use repeater along to keep good signal strength

NAND Function

For the NAND function, the output is false only when ALL the inputs are true





Condition	A	B	$\overline{AB}$
1	0	0	1
2	0	1	1
3	1	0	1
4	1	1	0

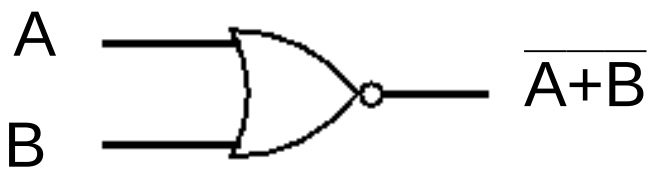
output of NAND is complement to that of an AND

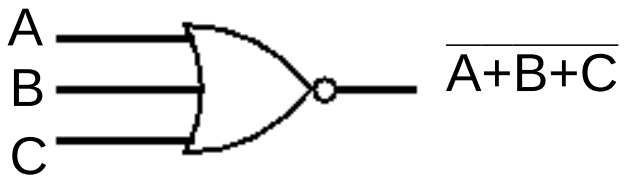
Condition	A	B	C	$\overline{ABC}$
1	0	0	0	1
2	0	0	1	1
3	0	1	0	1
4	0	1	1	1
5	1	0	0	1
6	1	0	1	1
7	1	1	0	1
8	1	1	1	0

T	On	Closed	1
F	Off	Open	0

NOR Function

For the NOR function, the output is true only when ALL inputs are false





Condition	A	B	$\overline{A+B}$
1	0	0	1
2	0	1	0
3	1	0	0
4	1	1	0

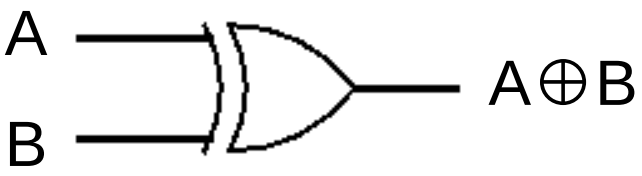
output of NOR is complement to that of an OR

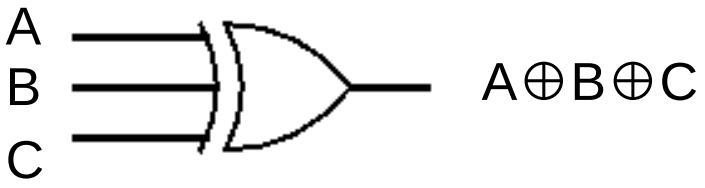
Condition	A	B	C	$\overline{A+B+C}$
1	0	0	0	1
2	0	0	1	0
3	0	1	0	0
4	0	1	1	0
5	1	0	0	0
6	1	0	1	0
7	1	1	0	0
8	1	1	1	0

T	On	Closed	1
F	Off	Open	0

XOR Function

For the XOR function, if one input is different than the other...then the output will be true





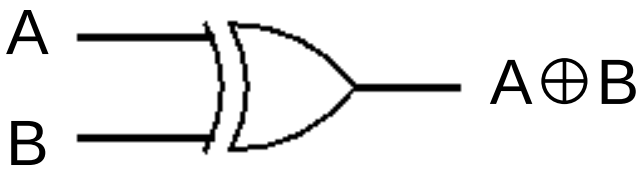
Condition	A	B	A⊕B
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	0

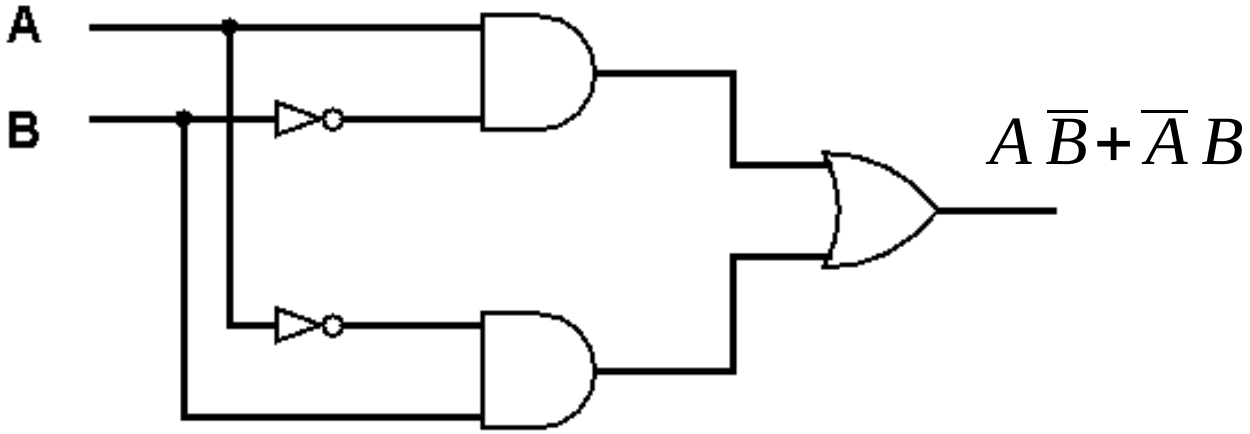
If both inputs are true or false then the output will be false

Condition	A	B	C	A⊕B⊕C
1	0	0	0	0
2	0	0	1	1
3	0	1	0	1
4	0	1	1	0
5	1	0	0	1
6	1	0	1	0
7	1	1	0	0
8	1	1	1	1

T	On	Closed	1
F	Off	Open	0

XOR Function



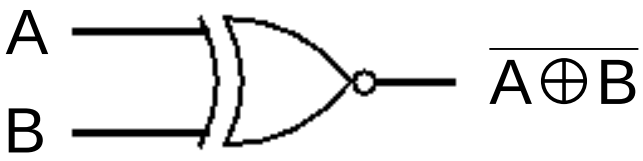


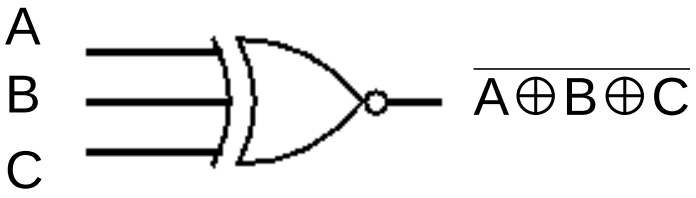
If the XOR function is used in any of the logic diagrams, then the reduction can be made using its equivalent form. The equivalent of XOR using AND, OR and NOT terms is

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

XNOR Function

For the XNOR function, if one input is different than the other...then the output will be false





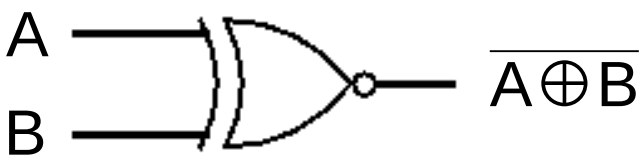
Condition	A	B	$\overline{A \oplus B}$
1	0	0	1
2	0	1	0
3	1	0	0
4	1	1	1

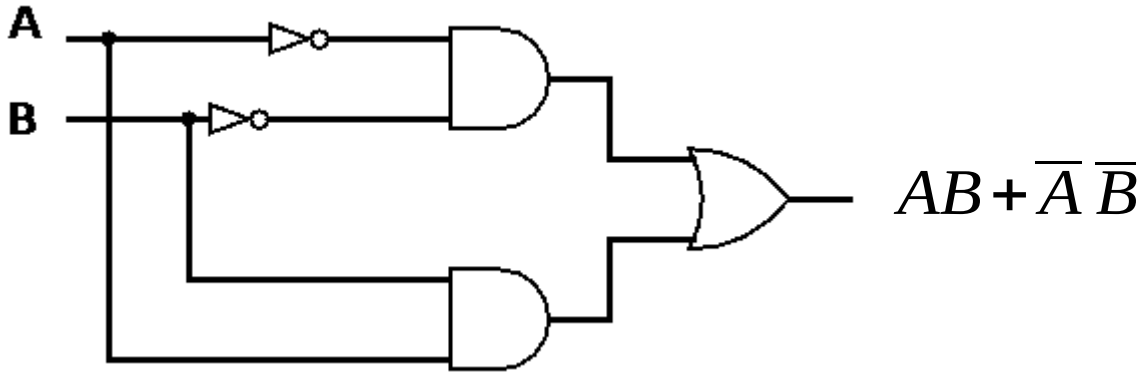
If both inputs are true or false then the output will be true

Condition	A	B	C	$\overline{A \oplus B \oplus C}$
1	0	0	0	1
2	0	0	1	0
3	0	1	0	0
4	0	1	1	1
5	1	0	0	0
6	1	0	1	1
7	1	1	0	1
8	1	1	1	0

T	On	Closed	1
F	Off	Open	0

XNOR Function

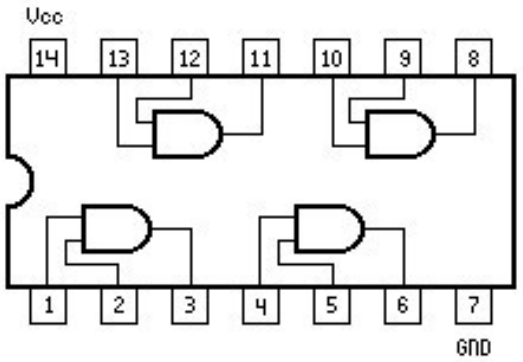




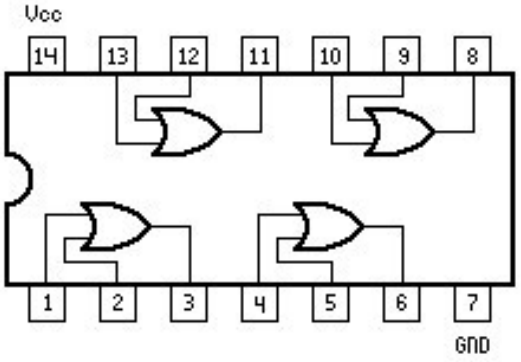
The XNOR function is the inverse of the XOR. If XNOR is used in any of the logic diagrams, then the reduction can be made using its equivalent form. The equivalent of XNOR using AND, OR and NOT terms is

$$\overline{A}\,\overline{B} + AB = (\overline{A} + B)(A + \overline{B}) = \overline{A \oplus B}$$

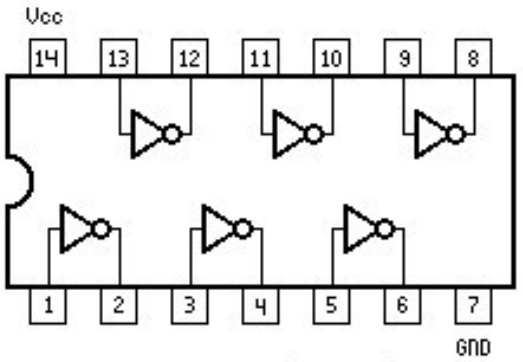
Logic Gates – PCB – DIP



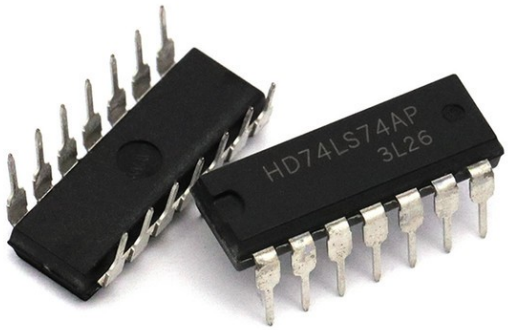
7408 (AND)



7432 (OR)



7404 (NOT)



Logic gates in dual inline packages (DIP)

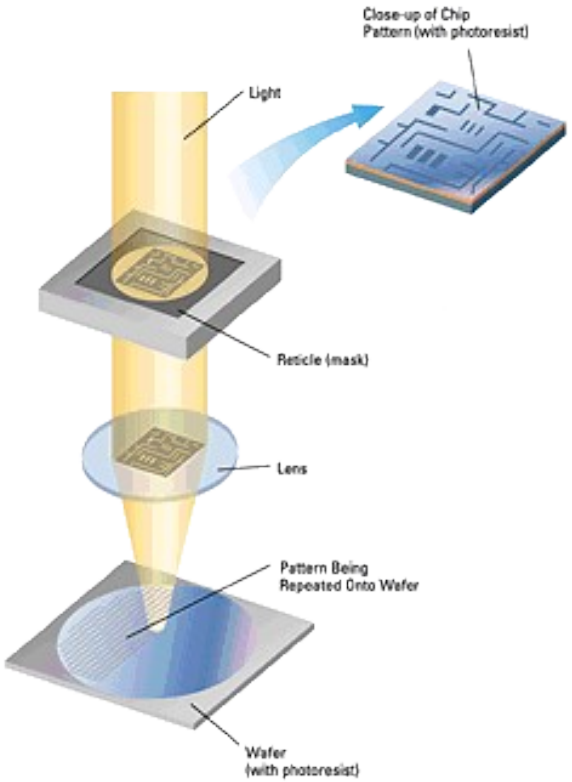
Printed circuit boards (PCB)

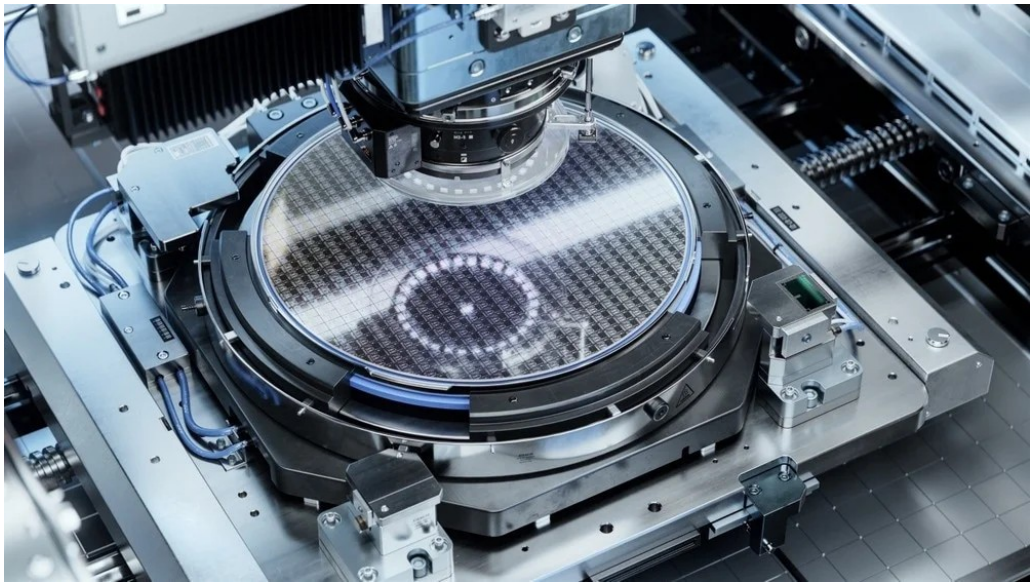
V_{cc} (Voltage Common Collector) is usually +5V DC with respect to GND

https://manual.eg.poly.edu/index.php/Digital_Logic
https://en.wikipedia.org/wiki/Printed_circuit_board
https://sparkfruit.ph/product/logic-gates-ic-hd74ls74ap/

Integrated Circuits – IC

Integrated circuits (IC) are produced using a multi-step process called photolithography where electronic circuits are created as layers onto a silicon wafer





https://www.azonano.com/article.aspx?ArticleID=6376
https://www.quora.com/What-are-the-limitations-of-optical-lithography

Review

Review question set 7 and 8