Ministry of Higher Education and Scientific Research
العلمي البحث و العالي التعليم وزارة

BADJI MOKHTAR-ANNABA UNIVERSITY
UNIVERSITE BADJI MOKHTAR-ANNABA

جامعة باجي مختار-عنابة

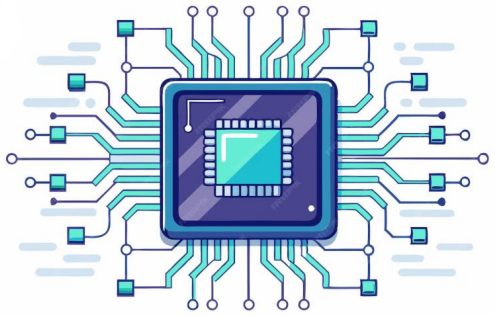
Faculty of technology
Electronics departement

Embedded Computing Systems course

Teaching method : Distance learning

Course 2

Chapter 1 : Reminders on combinatorial and sequential logic
Numbers system



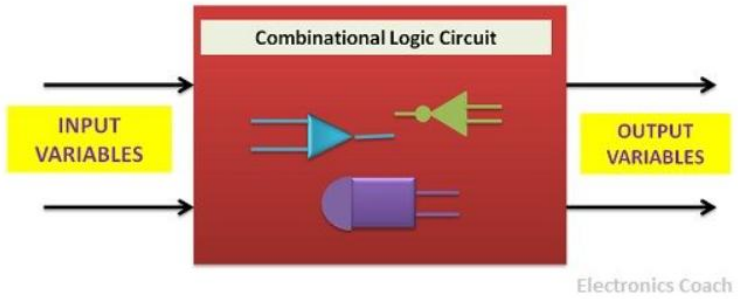
Teaching by
Dr. MERABTI Nardjes

EAD3/ DD3
Promotion : 2025/2026

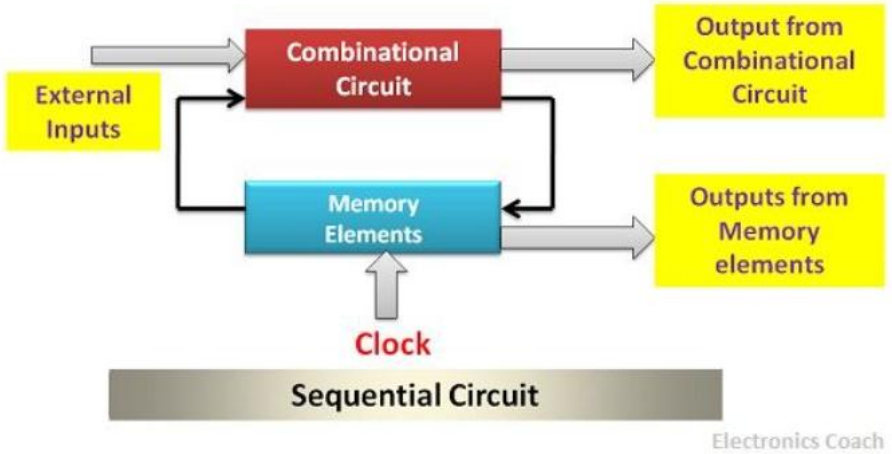
General Introduction

Embedded systems are based on digital electronics; They process information in binary form (0 and 1) through logic circuits that perform deterministic operations; Two main types of circuits :

- **Combinational circuits:** whose output depends only on the current inputs.



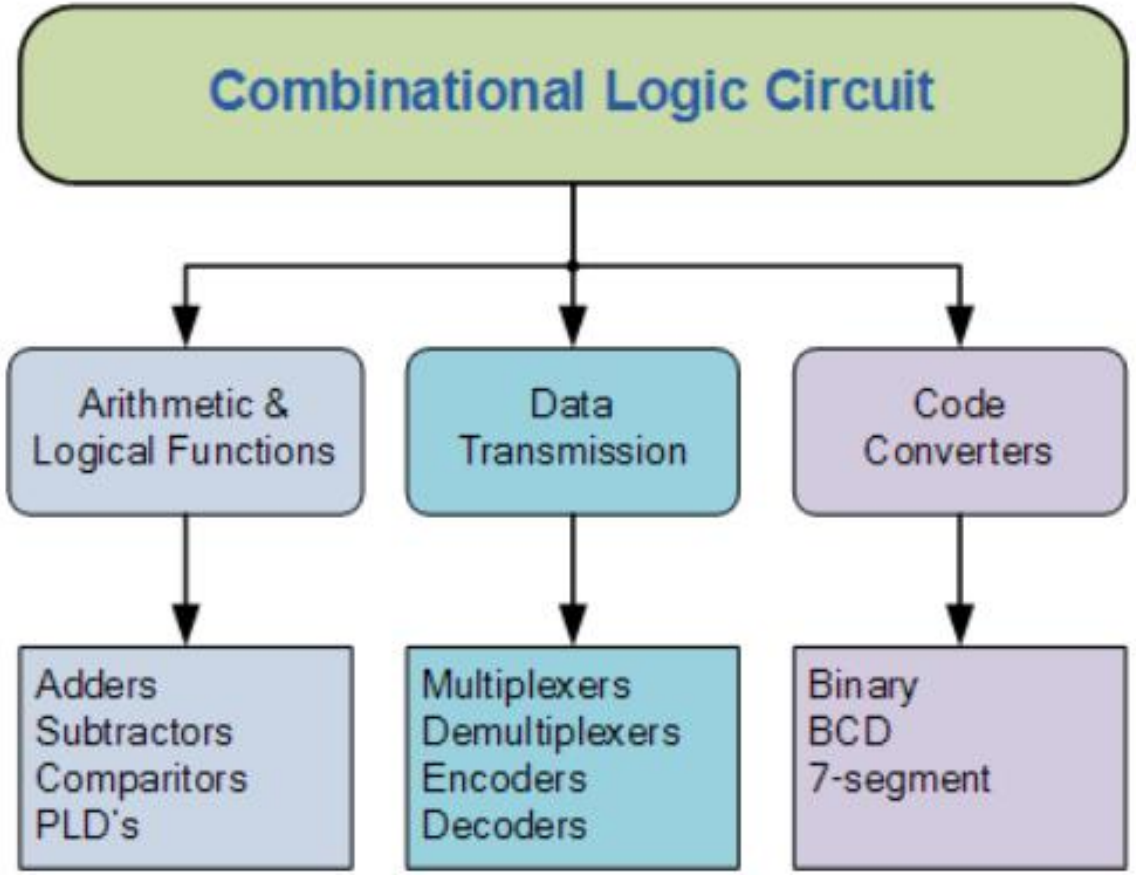
- **Sequential circuits:** whose output depends on both the current inputs and the past state (**memory**).



❖ **The table below provides a general overview of the differences between combinational and sequential circuits:**

Differences	Combinational Circuits	Sequential Circuits
Output	Output depends only on current input values	Output depends on current input and previous state
Feedback	No feedback path from output to input	Contains feedback path, creating a loop
Timing	No concept of time delay or clock signal	Relies on clock signals for synchronization
State	Stateless, as there is no stored information	Possesses state, storing information temporarily
Components	Consists of logic gates and combinational elements	Involves flip-flops, registers, and memory units
Functionality	Performs boolean operations and logic functions	Performs both combinational and sequential tasks
Design Complexity	Generally simpler to design and analyze	Typically more complex to design and troubleshoot
Application	Used in data processing, arithmetic operations, etc.	Applied in memory units, CPUs, and control units
Memory Requirement	Requires less memory	Requires additional memory for storing state
Examples	Adders, decoders, multiplexers	Counters, shift registers, finite state machines

Combinational logic circuits have “no memory”, “timing” or “feedback loops” thus, there operation is instantaneous.

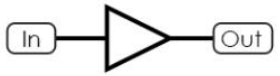


Digital Logic : A logic gate is a small digital circuit that performs a basic operation on binary inputs (0 or 1) and gives a binary output. Microprocessors and microcontrollers are made of millions of these gates.

Logic gates allow microprocessors and microcontrollers to:

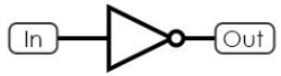
- ✓ Calculate, decide, store, and communicate
- ✓ Transform simple signals into complex actions
- ✓ Form the foundation of all digital intelligence

BUF (Buffer)



In	Out
0	0
1	1

NOT (Inverter)



In	Out
0	1
1	0

AND

In1	In2	Out
0	0	0
0	1	0
1	0	0
1	1	1

NAND (NOT AND)

In1	In2	Out
0	0	1
0	1	1
1	0	1
1	1	0

OR

In1	In2	Out
0	0	0
0	1	1
1	0	1
1	1	1

NOR (NOT OR)

In1	In2	Out
0	0	1
0	1	0
1	0	0
1	1	0

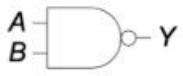
XOR (Exclusive Or)

In1	In2	Out
0	0	0
0	1	1
1	0	1
1	1	0

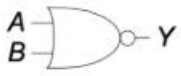
XNOR (NOT XOR)

In1	In2	Out
0	0	1
0	1	0
1	0	0
1	1	1

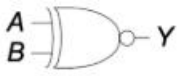
$Y = A \oplus B$



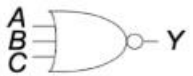
$Y = \overline{AB}$



$Y = \overline{A + B}$



$Y = \overline{A \oplus B}$



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

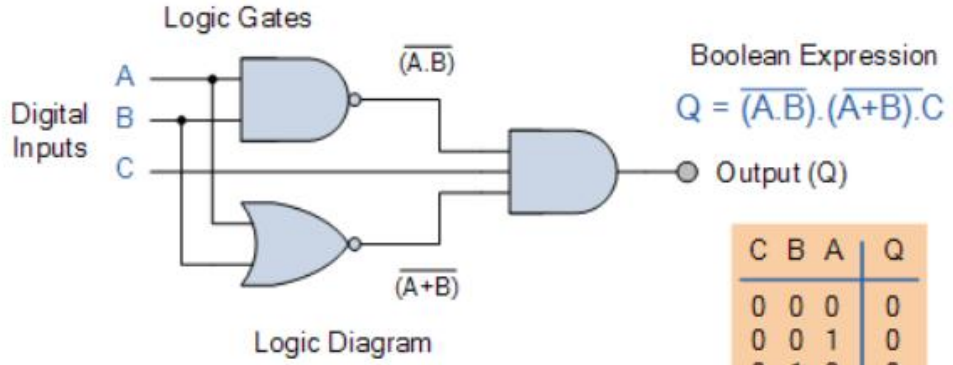
$Y = ABC$

The three main ways of specifying the function of a combinational logic circuit are:

1. **Boolean Algebra** – This forms the algebraic expression showing the operation of the logic circuit for each input variable either True or False that results in a logic “1” output.

2. Truth Table – A truth table defines the function of a logic gate by providing a concise list that shows all the output states in tabular form for each possible combination of input variable that the gate could encounter.

3. Logic Diagram – This is a graphical representation of a logic circuit that shows the wiring and connections of each individual logic gate, represented by a specific graphical symbol, that implements the logic circuit.

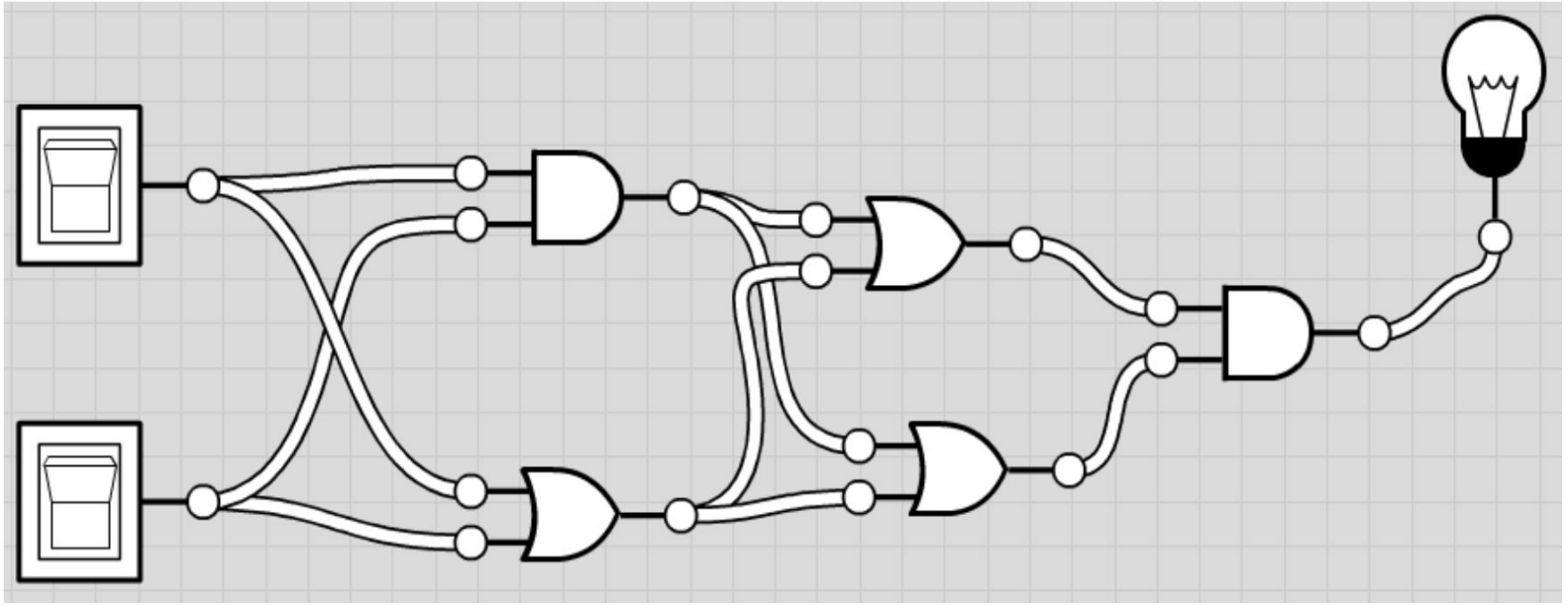


Typical Truth Table

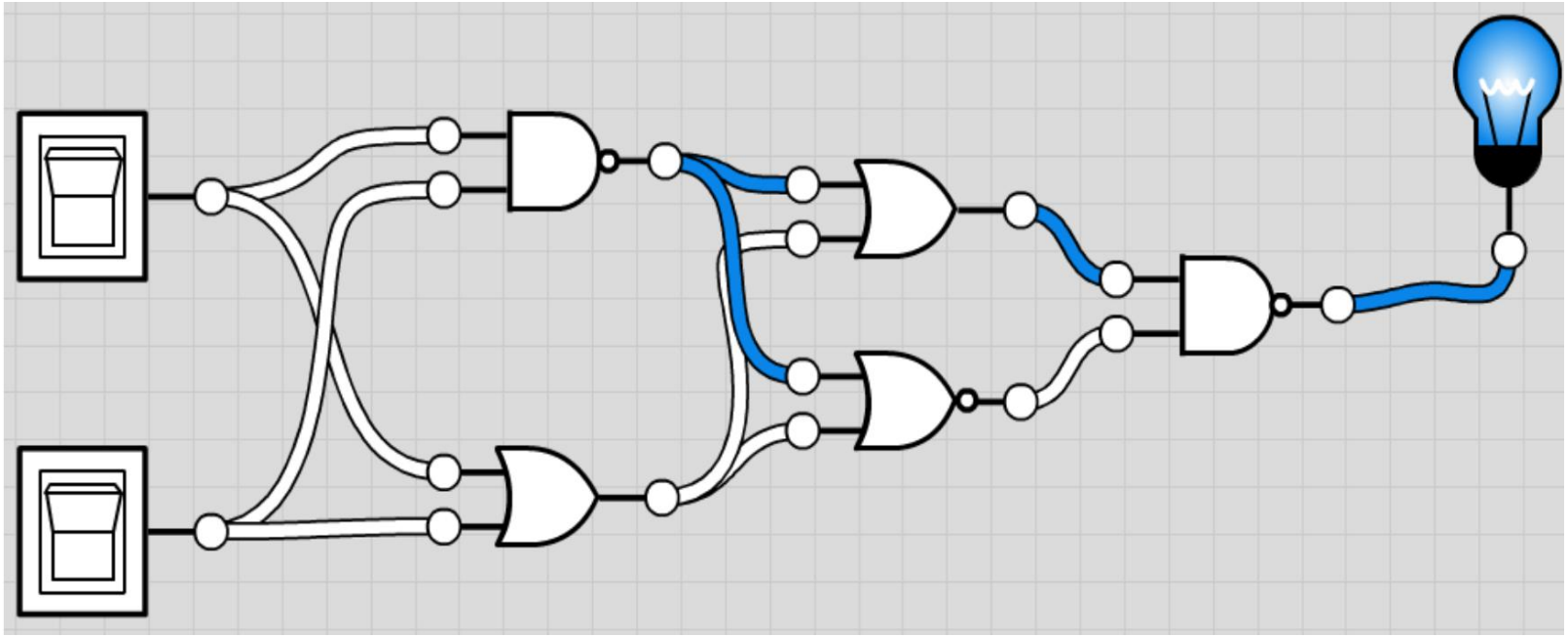
C	B	A	Q
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

- Example:

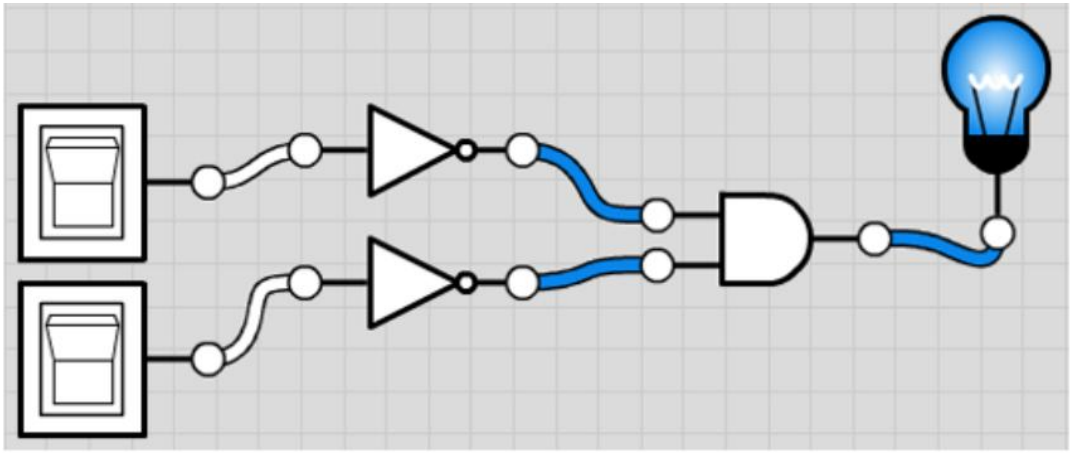
C=((A.B)+(A+B)).((A.B)+(A+B)



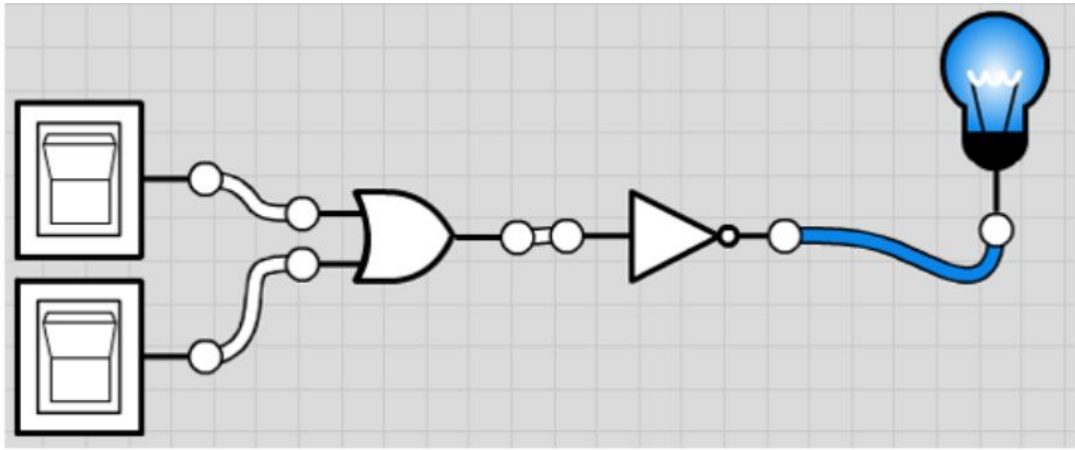
$C=\overline{(\overline{(A.B)}+(A+B)).\overline{(\overline{(A.B)}+(A+B))}}$



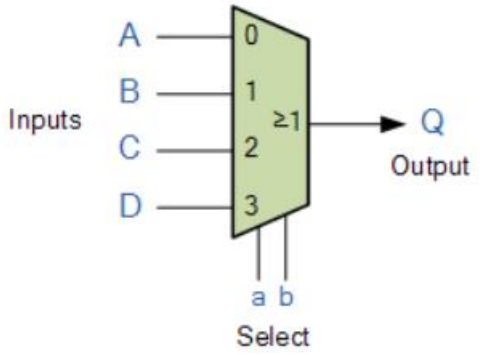
* **The Morgan's Law states** that: $\overline{A}.\overline{B} = \overline{A+B}$



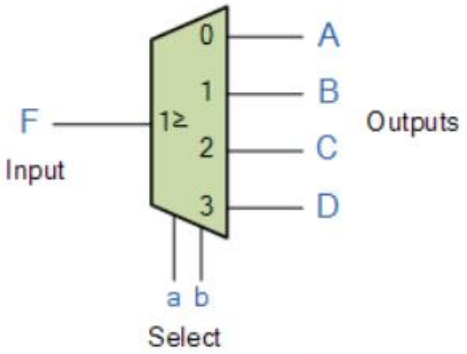
$C = \bar{A} \cdot \bar{B}$



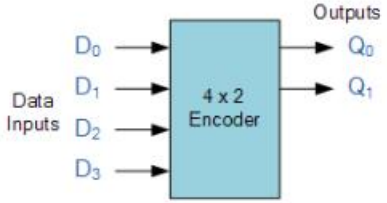
$C = \overline{A+B}$



The multiplexer (MUX) is a combinational logic circuit designed to switch one of several input lines to a single common output line

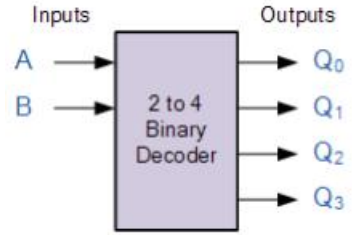


The demultiplexer is a combinational logic circuit designed to switch one common input line to one of several seperate output line



Inputs				Outputs	
D_3	D_2	D_1	D_0	Q_1	Q_0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1
0	0	0	0	x	x

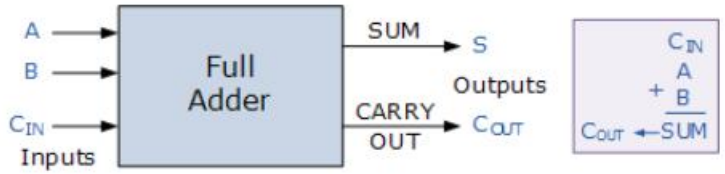
Priority Encoders(**Binary Encoder)** take all of their data inputs one at a time and converts them into an equivalent binary code at its output



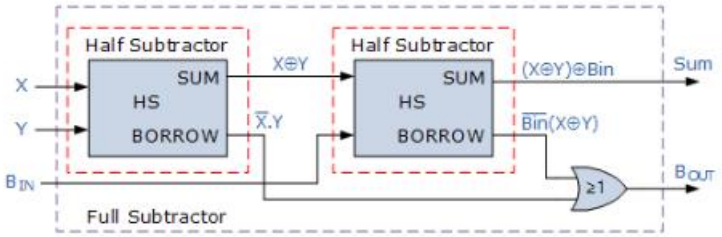
Truth Table

A	B	Q_0	Q_1	Q_2	Q_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

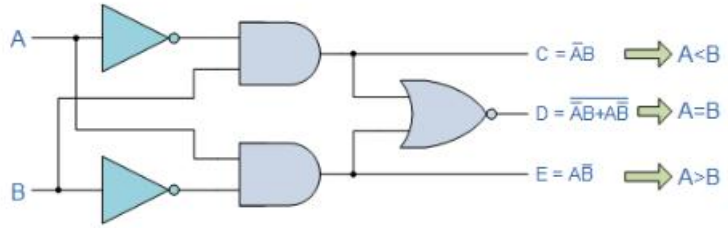
Binary Decoder is another combinational logic circuit constructed from individual logic gates and is the exact opposite to that of an Encoder



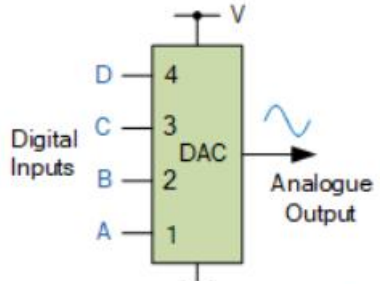
A basic Binary Adder circuit can be made from standard AND and Ex-OR gates allowing us to "add" together two single bit binary numbers, A and B.



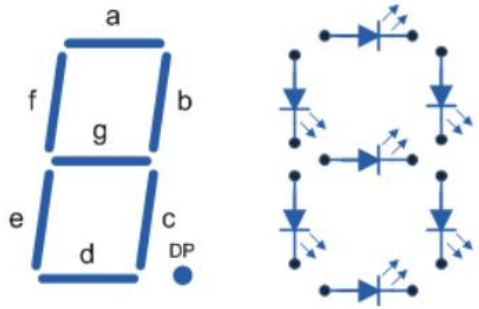
The Binary Subtractor is another type of combinational arithmetic circuit that produces an output which is the subtraction of two binary numbers



The Digital Comparator is another very useful combinational logic circuit used to compare the value of two binary digits



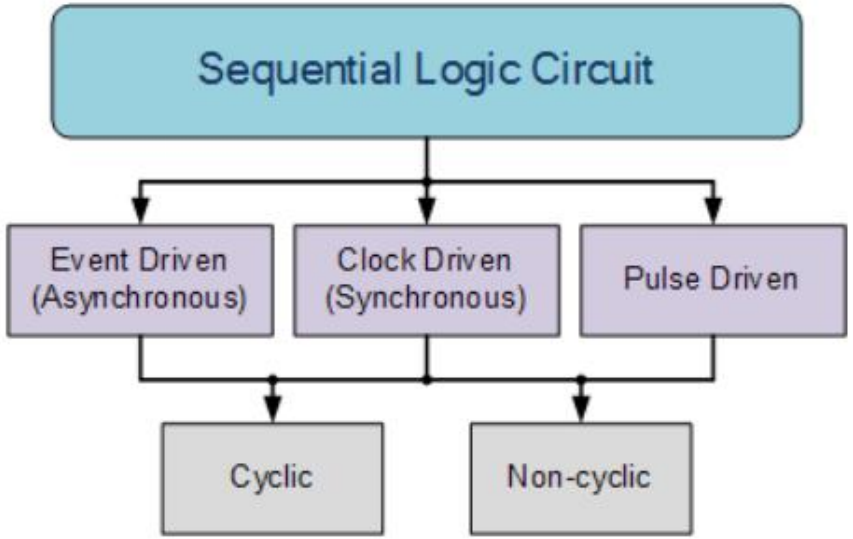
Binary DAC digital-to-analogue converters are a type of data converter which converts a digital binary number into an equivalent analogue output signal proportional to the value of the digital number

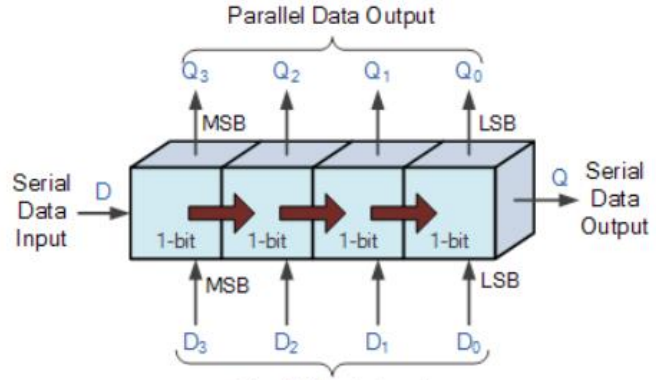


A Display Decoder is a combinational circuit which decodes and n-bit input value into a number of output lines to drive a display

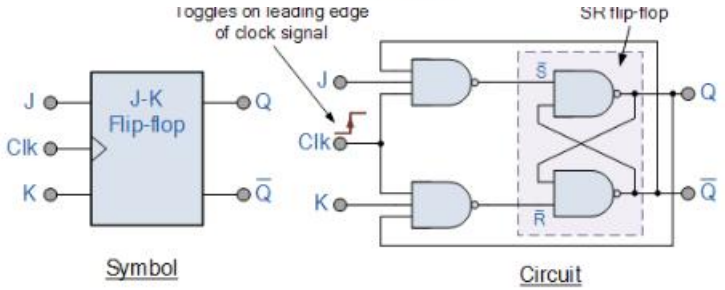
❖ **Sequential logic circuits** can be divided into the following three main categories:

1. Event Driven – asynchronous circuits that change state immediately when enabled.
2. Clock Driven – synchronous circuits that are synchronised to a specific clock signal.
3. Pulse Driven – which is a combination of the two that responds to triggering pulses.

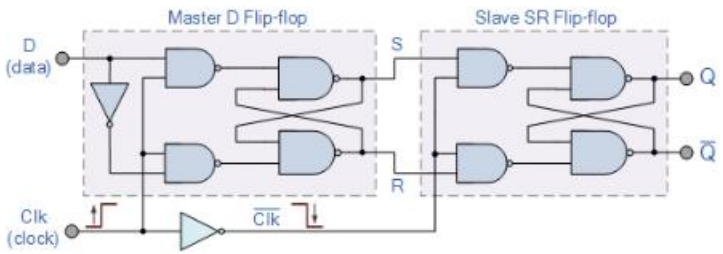




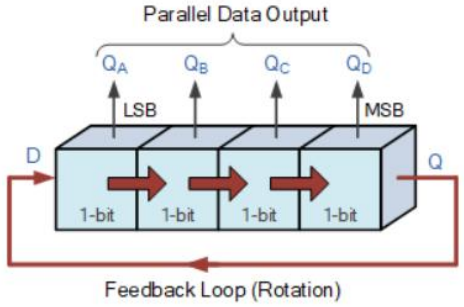
The Shift Register is another type of sequential logic circuit that can be used for the storage or the transfer of binary data

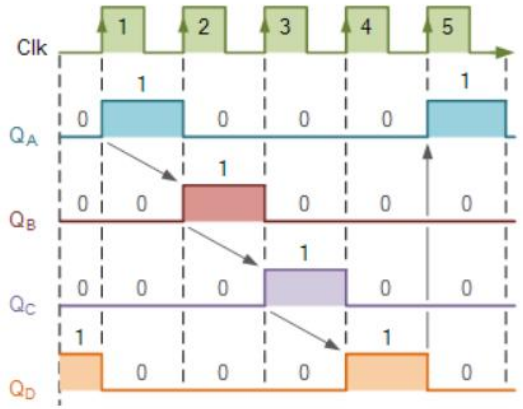


The JK Flip-flop is similar to the SR Flip-flop but there is no change in state when the J and K inputs are both LOW

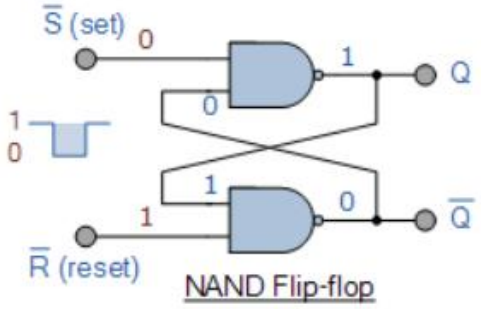


The D-type flip-flop is a modified Set-Reset flip-flop with the addition of an inverter to prevent the S and R inputs from being at the same logic level

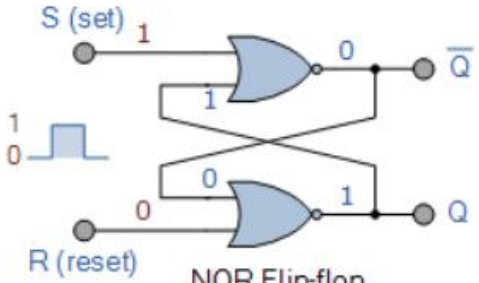




Consists of a number of counters connected together with the output fed back to the input



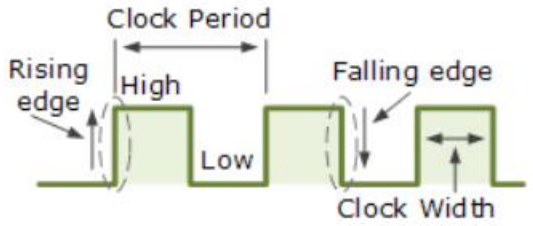
NAND Flip-flop



NOR Flip-flop

- **Basic NAND and NOR SR Flip-flops:** asynchronous sequential logic circuits. has two inputs S (set) and R (reset) and two outputs Q and Q’ with one of these outputs being the complement of the other.

Flip-Flop Type	Main Inputs	Function / Role	Simplified Truth Table	Key Feature
SR (Set–Reset)	S (Set), R (Reset)	Sets the output to 1 (Set) or resets it to 0 (Reset)	S=0, R=0 → No change S=1, R=0 → Q=1 S=0, R=1 → Q=0 S=1, R=1 → **Invalid / forbidden**	Oldest flip-flop; simple but has an invalid state
D (Data or Delay)	D	Copies the input D to the output Q on each clock edge	D=0 → Q=0 D=1 → Q=1	Most used in registers and memory (no invalid state)
JK	J, K	Combines SR and removes invalid state	J=0, K=0 → No change J=1, K=0 → Q=1 J=0, K=1 → Q=0 J=1, K=1 → **Toggle (invert Q)**	Most flexible flip-flop, used in counters
T (Toggle)	T	Toggles the output when T=1	T=0 → No change T=1 → Q toggles (0→1 or 1→0)	Very simple, used in binary counters



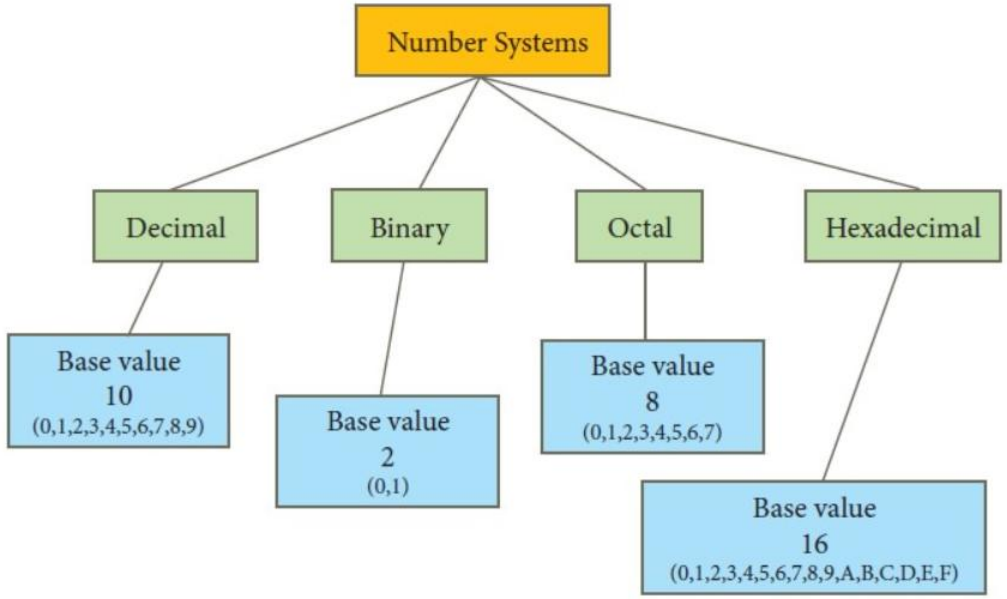
Multivibrators are sequential logic circuits that operate continuously between two distinct states of HIGH and LOW

Individual **Sequential Logic** circuits can be used to build more complex circuits such as Multivibrators, Counters, Shift Registers, Latches and Memories. they require the addition of some form of clock pulse or timing signal to cause them to change their state. Clock pulses are generally continuous square or rectangular shaped waveform

Sequential logic circuits which use a clock signal for synchronization are dependent upon the frequency and therefore the clock pulse width to activate their switching action. Sequential circuits can also change their switching state using either the rising edge, falling edge, or both edges of the clock signal

- **Active HIGH** if the state change occurs from a “LOW” to a “HIGH” on the clock’s pulse rising edge
- **Active LOW** if the state change occurs from a “HIGH” to a “LOW” on the clock’s pulses falling edge.
- **Clock Period** this is the time between successive transitions in the same direction: between two rising or two falling edges.
- **Clock Frequency :** the clock frequency is the reciprocal of the clock period, frequency = 1/clock period. ($f = 1/T$)

Types of Numbers system



Why numbers system are matter in Microprocessors & Microcontrollers

- **Binary:** Everything inside the processor is binary ; it's the language of the machine, Data is stored and manipulated in binary
- **Hexadecimal:** Used to simplify long binary numbers , Memory and registers) addresses are often shown in hexadecimal
- **Octal:** Useful for grouping binary bits (especially in older systems or low-level programming).
- **Decimal:** Used for input/output and user-facing data.

❖ Decimal (Base 10)

This is the number system we use every day: 0 to 9

- **Example:**

25, 100, 2025

Microprocessors convert decimal numbers into binary to process them

❖ Binary (Base 2)

Used inside the microprocessor

Only two digits: 0 and 1

Everything: numbers, instructions, data is stored and processed in binary

- **Example:**

1010 (binary) = 10 (decimal)

❖ Octal (Base 8)

Octal uses 8 digits: 0 to 7

It’s a shortcut for binary, just like hexadecimal

Each octal digit represents 3 binary bits

- **Example:**

Binary: 11001000

Group into 3 bits: 011 001 000

Octal: 310

Decimal: 200

❖ Hexadecimal (Base 16)

Used to simplify binary

Digits: 0–9 and A–F (A=10, B=11, ..., F=15)

Easier to read and write than long binary strings

- **Example:**

FF (hex) = 255 (decimal) = 11111111 (binary)

If you want to store the number 200:

Decimal: 200, **Binary:** 11001000, **Hexadecimal:** C8

❖ BCD = Binary Coded Decimal.

It means each decimal digit (0–9) is written in binary using 4 bits.

BCD is mainly used in microprocessors/microcontrollers for display purposes (digital watches, calculators, etc.), because it is easier to handle each digit directly.

- **Example:** Decimal 59

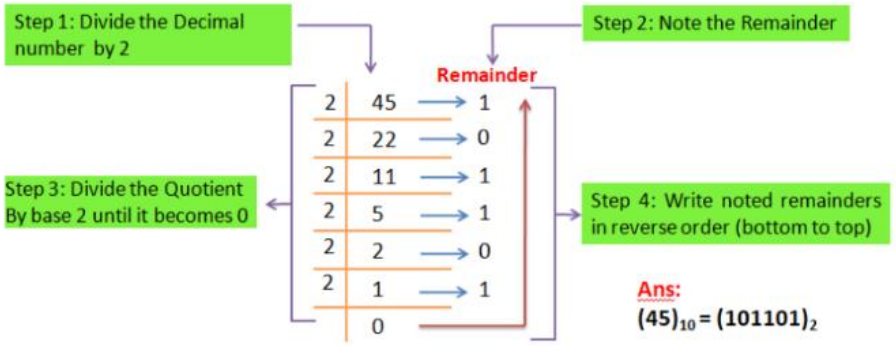
"5" → 0101 in binary

"9" → 1001 in binary

So in BCD: 59 = 0101 1001

Conversion between number system Decimal/ Binary

Example 1: Convert $(45)_{10}$ to Binary Number.



Decimal to Binary Conversion

Example 1: Convert $(10011)_2$ to Decimal Number

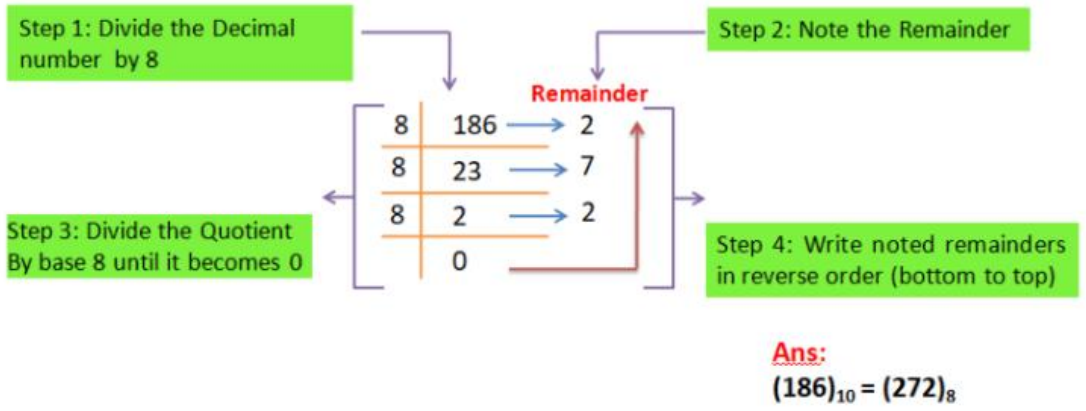
$(10011)_2$

Binary No	1	0	0	1	1	
Position No	4	3	2	1	0	
Positional Value	2^4	2^3	2^2	2^1	2^0	
Decimal Number	$1x2^4$ +	$0x2^3$ +	$0x2^2$ +	$1x2^1$ +	$1x2^0$	$=(23)_{10}$

Binary to Decimal Conversion

Conversion between number system Decimal/ Octal

Example : Convert $(186)_{10}$ to Octal Number.



Ans:

$(186)_{10} = (272)_8$

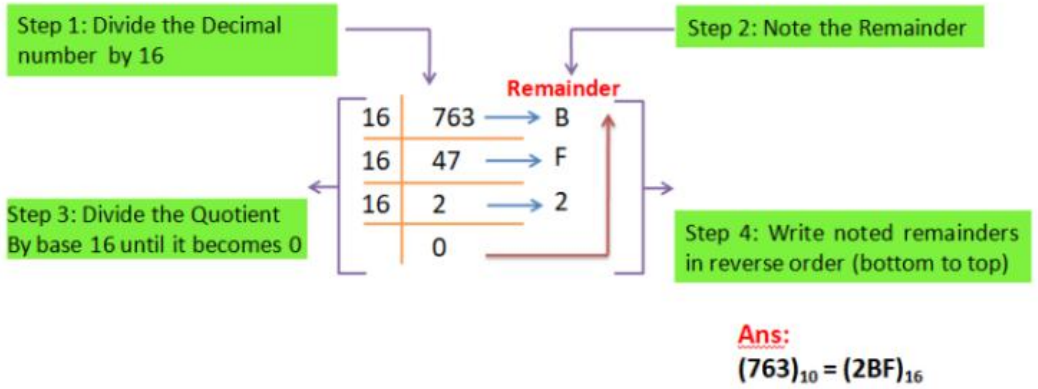
Example : Convert $(336)_8$ to Decimal Number

	$(336)_8$		
Octal No	3	3	6
Position No	2	1	0
Positional Value	8^2	8^1	8^0
Decimal Number	$3x8^2 + 3x8^1 + 6x8^0$		$=(222)_{10}$

Octal to decimal conversion

Conversion between number system Decimal/ Hexadecimal

Example : Convert $(763)_{10}$ to Hexadecimal Number.



Ans:
$(763)_{10} = (2BF)_{16}$

Decimal to Hexadecimal Conversion

Example : Convert $(7AC)_{16}$ to Decimal Number

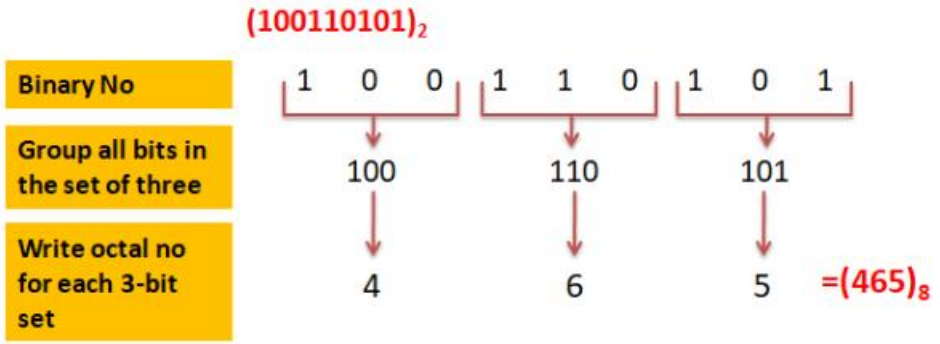
$(7AC)_{16}$

Hexadecimal No	7	A	C
Position No	2	1	0
Positional Value	16^2	16^1	16^0
Decimal Number	$7x16^2+$	$10x16^1$ +	$12x16^0$ $=(1964)_{10}$

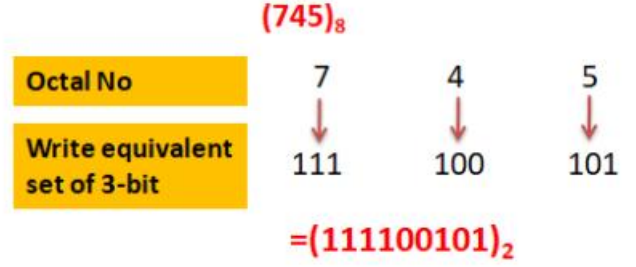
Hexadecimal to decimal conversion

Conversion between number system Binary/ Octal

Example : Convert $(100110101)_2$ in Octal Number



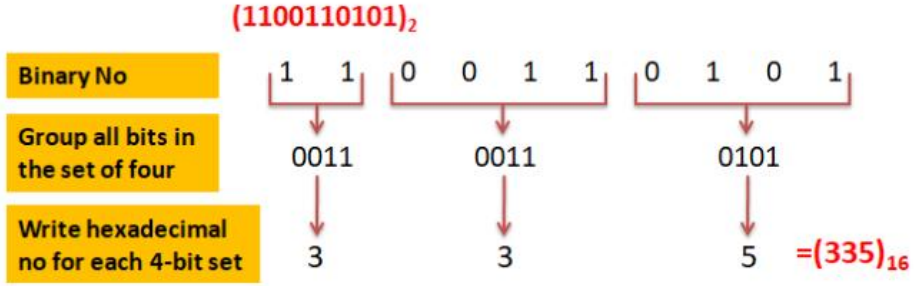
Example : convert $(745)_8$ into binary number



Octal to binary conversion

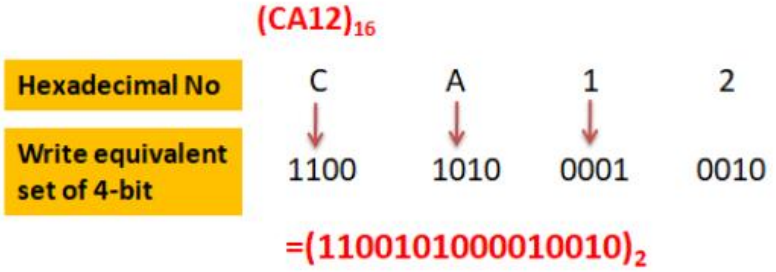
Conversion between number system Binary/Hexadecimal

Example : Convert $(1100110101)_2$ in hexadecimal Number



Binary to hexadecimal conversion

Example : convert $(CA12)_{16}$ into binary number



Hexadecimal to binary conversion

Conversion of Decimal Number with fractional part to Binary /Octal number / Hexadecimal number

Example : convert $(0.625)_{10}$ to binary number.

$(0.625)_{10}$

$0.625 \times 2 = 1.250$

$0.250 \times 2 = 0.500$

$0.500 \times 2 = 1.000$

$=(0.101)_2$

Example : convert $(0.175)_{10}$ to Octal number.

$(0.175)_{10}$

$0.175 \times 8 = 1.400$

$0.400 \times 8 = 3.200$

$0.200 \times 8 = 1.600$

$0.600 \times 8 = 4.800$

$0.800 \times 8 = 6.400$

$=(0.13146)_8$

Example : convert $(0.175)_{10}$ to Hexadecimal number.

$(0.175)_{10}$

$0.175 \times 16 = 02.800$

$0.800 \times 16 = 12.800$

$=(2C)_{16}$

Conversion of Binary Number with fractional part to Octal number

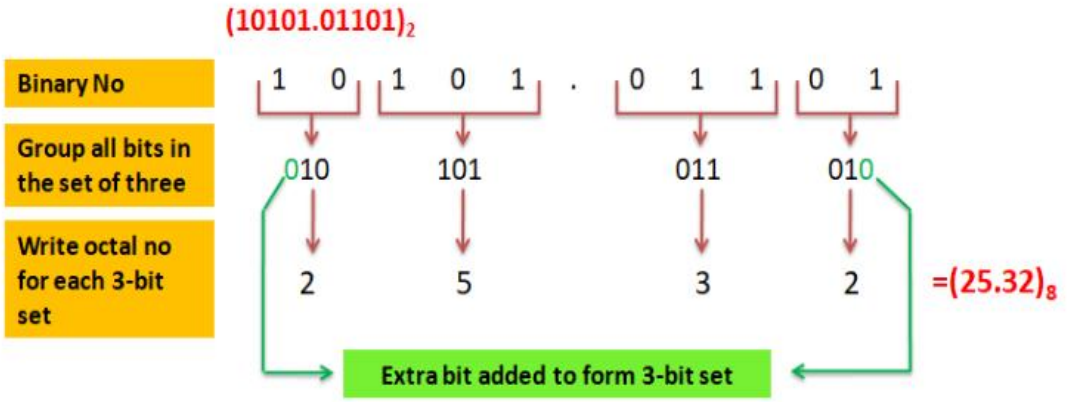
Example : Convert $(10011.11)_2$ to Decimal Number

$(10011.11)_2$

Binary No	1	0	0	1	1	.	1	1
Positional No	4	3	2	1	0	.	-1	-2
Positional Value	2^4	2^3	2^2	2^1	2^0	.	2^{-1}	2^{-2}
Decimal Number	$1x2^4$	+ $0x2^3$	+ $0x2^2$	+ $1x2^1$	+ $1x2^0$	.	$1x2^{-1}$	+ $1x2^{-2}$
	16	+ 0	+ 0	+ 2	+ 1	.	0.5	+ 0.25 $=(19.75)_{10}$

Binary with fractional part to decimal Conversion

Example : Convert $(10101.01101)_2$ in Octal Number



Example : Convert $(10101.01101)_2$ in Hexadecimal Number

Binary No 1 0 1 0 1 . 0 1 1 0 1

0 0 0 1 0 1 0 1 . 0 1 1 0 1 0 0 0

1 5 6 8

$= (15.68)_{16}$

Octal to Hexadecimal and Hexadecimal to Octal

To convert from Octal to Hexadecimal or Hexadecimal to Octal, first, convert them to binary and from binary to respective form.

Octal ------> Binary ------> Hexadecimal

Hexadecimal ------> Binary ------> Octal

Example: $(2\,C\,A)_{16} = (0\,0\,1\,0\,1\,1\,0\,0\,1\,0\,1\,0)_2 = (1\,3\,2\,1)_8$

$$(2\,C\,A)_{16} = \quad 2 \qquad C \qquad A$$

$$\mathbf{00}1\mathbf{0} \quad 11\mathbf{00} \quad \mathbf{1}010$$

$$1 \quad 3 \quad 2 \quad 1 \quad = (1\,3\,2\,1)_8$$

❖Number System Table

Decimal	Binary	Octal	Hexadecimal
0	0000	00	0
1	0001	01	1
2	0010	02	2
3	0011	03	3
4	0100	04	4
5	0101	05	5
6	0110	06	6
7	0111	07	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Decimal	Binay (BCD) 8 4 2 1
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1

BCD = "Binary Coded Decimal" → each decimal digit (0–9) is stored separately in 4 bits.
BCD keeps each digit separate for easier display on digital devices (like calculators or 7-segment displays).
BCD is different from pure binary:

Example:

Decimal 59 → BCD = 0101 1001

Decimal 59 → Pure binary = 111011

Binary addition

An important rule to be kept in mind is that just like real numbers, binary arithmetic calculations begin from the right side. For addition, we have four simple rules to remember:

0 + 0 = 0 ,

0 + 1 = 1 ,

1 + 0 = 1 , and

1 + 1 = 0 (with a carry to the adjacent left bit)

❖ **Example:**

	1 1 1 1 1	
$(85)_{10}$		01010101
$+(181)_{10}$	+	10110101
$(266)_{10}$		100001010

Binary Addition Example

Binary Subtraction

The binary subtraction has two new terms involved – the difference and the borrow. We have four main rules to remember for the binary Subtraction:

0 – 0 = 0 ,

0 – 1 = 1 , Borrow 1 from the next higher bit (since 0 < 1)

1 – 0 = 1

1 – 1 = 0

Example:

0011010	$(26)_{10}$
-0001100	$(12)_{10}$
0001110	$(14)_{10}$

Binary Multiplication

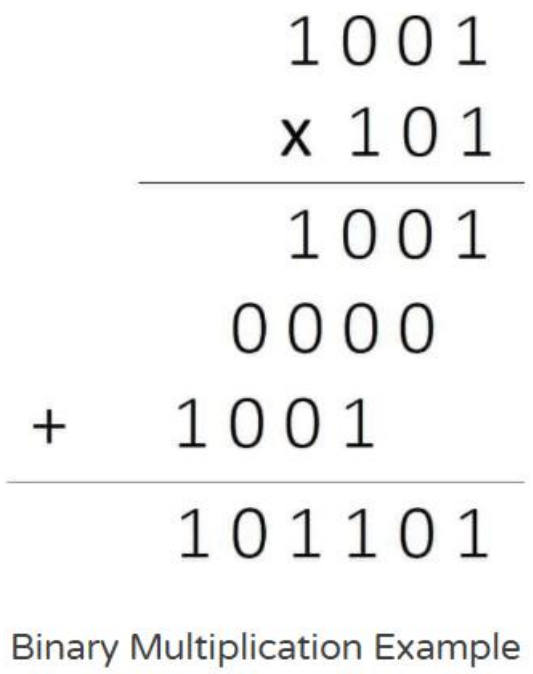
The binary multiplication is the easiest one when compared to the other operations! It is pretty similar to decimal multiplication ; any number multiplied with a 0 gives 0 as the product. Let us consider the four rules under this operation :

0 x 0 = 0 ,

0 x 1 = 0 ,

1 x 0 = 0, and

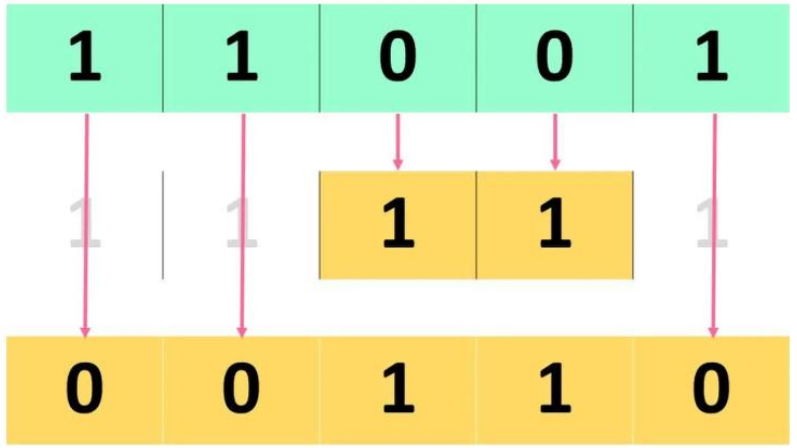
1 x 1 = 1



Binary Multiplication Example

1’s complement

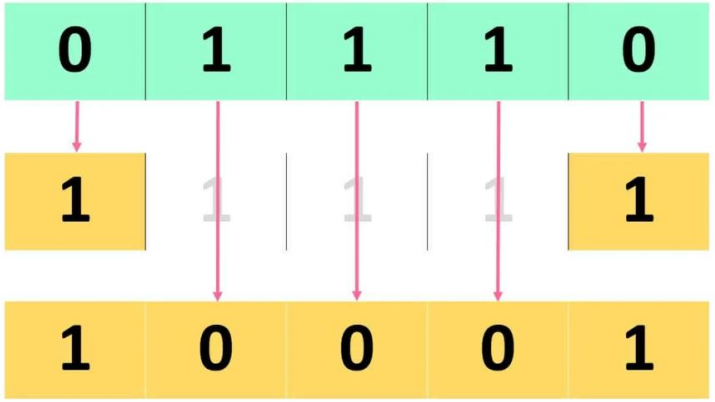
- Step 1: In the number, toggle all the pre-existing 0s present to 1s.
- Step 2: Now toggle all the pre-existing 1s present to 0s.



2’s complement (Complement to base 2)

- ✓ **Step 1:** In the number, toggle all the pre-existing 0s present to 1s.
- ✓ **Step 2:** Now toggle all the pre-existing 1s present to 0s.
- ✓ **Step 3:** Add binary 1 to the result of 1’s complement.

Example: Let us find the 2’s complement of **01110**

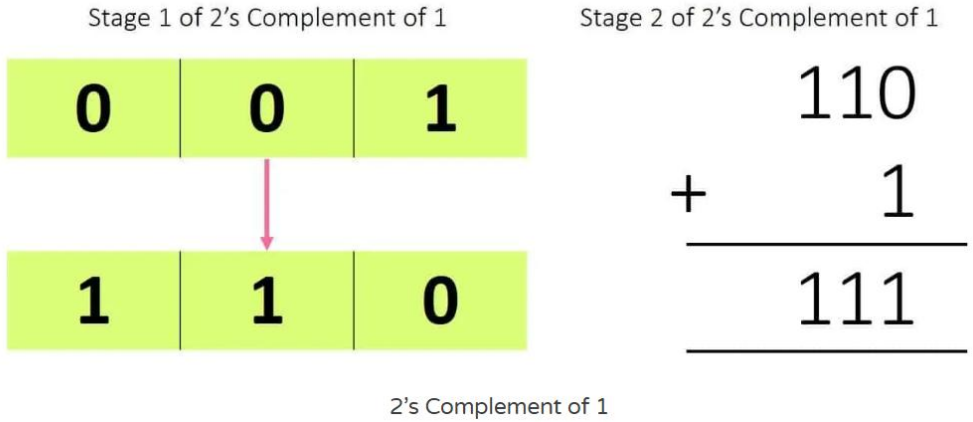


$$\begin{array}{r} {\scriptstyle 1} \\ 10001 \\ +\quad\quad 1 \\ \hline 10010 \end{array}$$

Bit Overflow

A bit overflow happens when it exceeds the capacity of a system. This usually occurs in the 2's complement operations. If we perform 2's complement on 2 N-bit numbers and add them, the answer sometimes exceeds the N-bit capacity, thus having an overflowing bit.

Example: Let us see an example of this. Let us add the 2's complement of 7 $(111)_2$ and 1 $(001)_2$. The system capacity is of 3 –bits.



The result of the 2's complement of both 7 and 1 have 3-bits each. Let us now see the addition of the 2's Complement of 1 and 7.

The sum is a 4-bit number, thus presenting a case of overflow.

This might create a problem in the functioning of circuitry and programs.