Digital Logic Design

Lecture No 09 : Other Codes, Logic Families

BEE-12CD, Fall 2021

Dated 29 Sept 2021

By Nasir Mahmood
nasir.mahmood@seecs.edu.pk
nasirm15@gmail.com

Gray Code

- It is sometimes convenient to use the Gray code to represent the digital data when it is converted from analog data
- The advantage of Gray code over straight binary number sequence is that only one bit in the code group changes when going from one number to the next

Gray Code

Decimal digit	Gray code
0	0000
1	0001
2	0011
3	0010
4	0110
5	0111
6	0101
7	0100

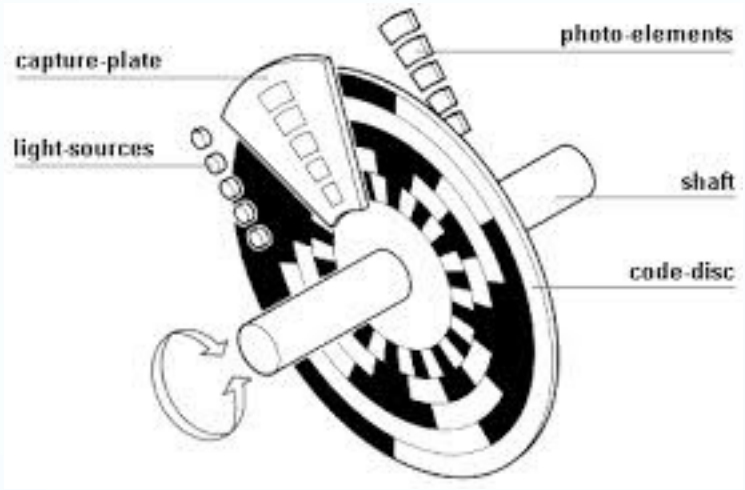
Decimal digit	Gray code
8	1100
9	1101
10	1111
11	1110
12	1010
13	1011
14	1001
15	1000

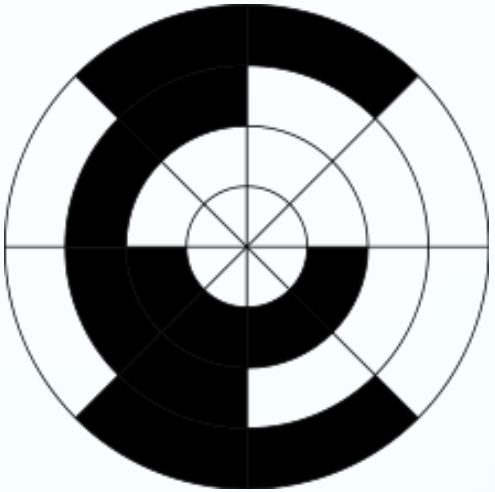
Gray Code Vs Binary Code

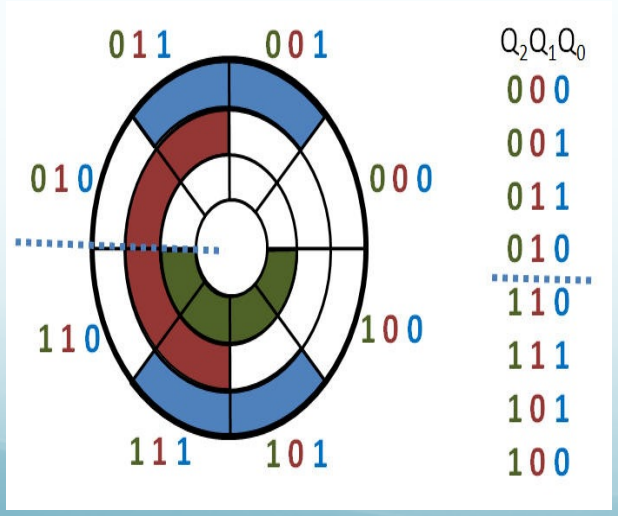
- Compare the number of bits changing when going from one number to the next:
  - In Gray code it is always 1 bit.

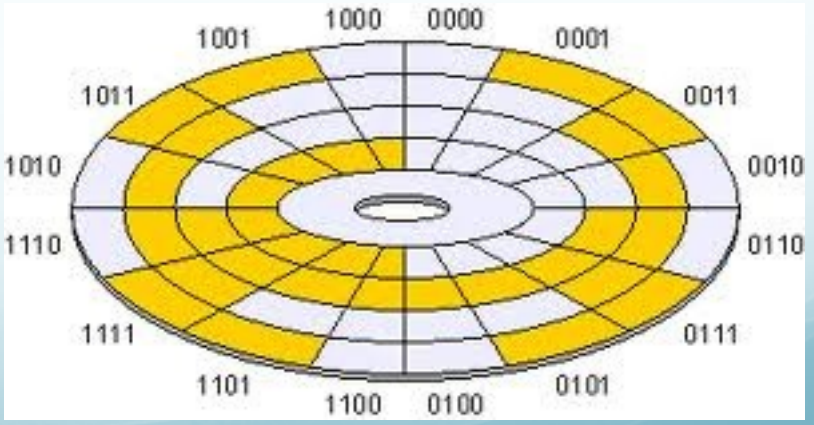
Binary Code	Bit Changes	Gray Code	Bit Changes
000		000	
	1		1
001		001	
	2		1
010		011	
	1		1
011		010	
	3		1
100		110	
	1		1
101		111	
	2		1
110		101	
	1		1
111		100	
	3		1
000		000	

Gray Code Application







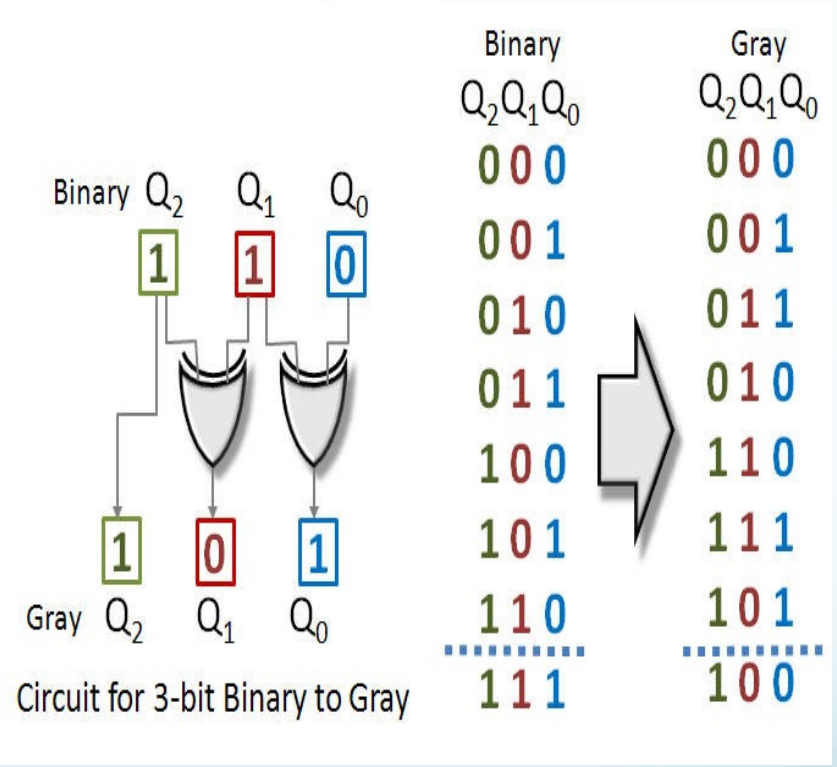


Conversion from Binary to Gray Code

- 1 0 1 1 0

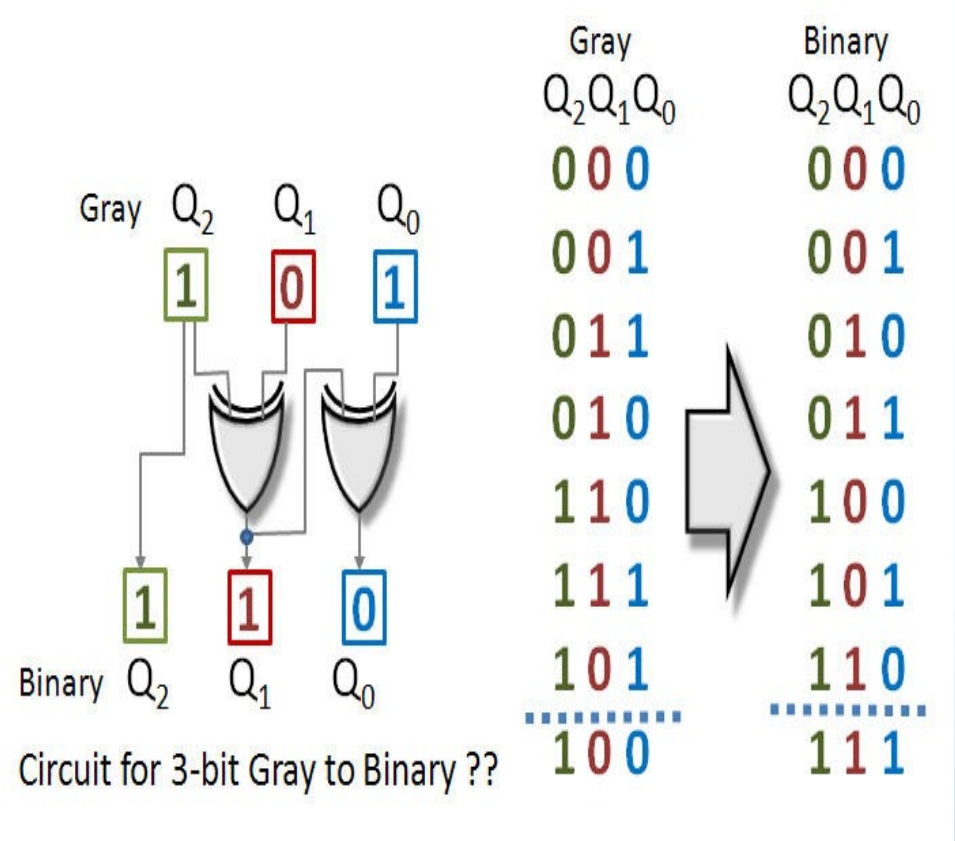
- First one copied
- Take ex-OR or add (Ignore Carry
  Left to right

- 1 1 1 0 1



Conversion from Gray to Binary Code

- 1 1 0 1 1
- First one copied
- Take ex-OR or add (Ignore Carry) diagonally from lower Left to upper right digit
- 1 0 0 1 0



ASCII Character Code

- The American Standard Code for Information Interchange (ASCII) uses seven bits to code 128 characters, representing the alphabets, decimal numbers, and various other symbols.
  - The following ASCII chart allows you to specify the characters in decimal representation by concatenating the column headings to the row headings.
    - For example, the character 5 is represented in binary as 0110101

ASCII Table

	$B_7B_6B_5$							
$B_4B_3B_2B_1$	000	001	010	011	100	101	110	111
0000	NULL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	"	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB		7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|
1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	_	o	DEL

ASCII Table (Contd ……)

Control Characters:

NULL	NULL	DLE	Data link escape
SOH	Start of heading	DC1	Device control 1
STX	Start of text	DC2	Device control 2
ETX	End of text	DC3	Device control 3
EOT	End of transmission	DC4	Device control 4
ENQ	Enquiry	NAK	Negative acknowledge
ACK	Acknowledge	SYN	Synchronous idle
BEL	Bell	ETB	End of transmission block
BS	Backspace	CAN	Cancel
HT	Horizontal tab	EM	End of medium
LF	Line feed	SUB	Substitute
VT	Vertical tab	ESC	Escape
FF	Form feed	FS	File separator
CR	Carriage return	GS	Group separator
SO	Shift out	RS	Record separator
SI	Shift in	US	Unit separator
SP	Space	DEL	Delete

Error-Detecting Code

- Error-Detecting uses an eighth bit (added to 7-bit ASCII character) to indicate parity.
  - A parity bit is an extra bit that is set to 0 or 1 as needed to make the total number of 1's either even or odd.
  - In an odd-parity code, the parity bit is specified so that the total number of ones is odd.
  - In an even-parity code, the parity bit is specified so that the total number of ones is even.
  - It detects one, three or any odd combination of errors but even combination of errors is undetected.


```
1 1 0 0 0 0 1 1
↑
Added even parity bit
```

```
0 1 0 0 0 0 1 1
↑
Added odd parity bit
```

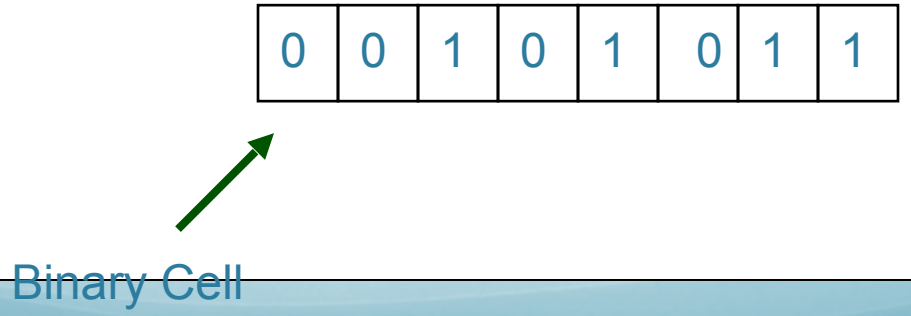
Parity Code Example

- Concatenate a parity bit to the ASCII code for the characters 0, X, and = to produce both odd-parity and even-parity codes.

Character	ASCII	Odd-Parity ASCII	Even-Parity ASCII
0	0110000	10110000	00110000
X	1011000	01011000	11011000
=	0111100	10111100	00111100

Binary Storage

- Binary storage represents the storage mechanisms for binary data stored in a computer.
  - A binary cell is a device that possesses two stable states and it can store a single state value (0 or 1). It stores single bit of data. examples: flip-flop circuits, ferrite cores, capacitor
  - A register is a group of binary cells. n cells allows the register to store n bits and thus 2^n possible states.
    - The type of information (BCD, ASCII, etc.) stored in a register has to be agreed upon by the users of the register.



Register Transfer

- A register transfer operation involves the transfer of binary information from one set of binary registers into other set of binary registers.
- The capture and storage of information requires:
  - An input register to store the key inputs from the keyboard
  - A processor register to store the data when processed by the CPU
  - A memory register in the memory unit to store the values

Register Transfer

- Data input at keyboard
- Shifted into place
- Stored in memory

  - NOTE: Data input in ASCII

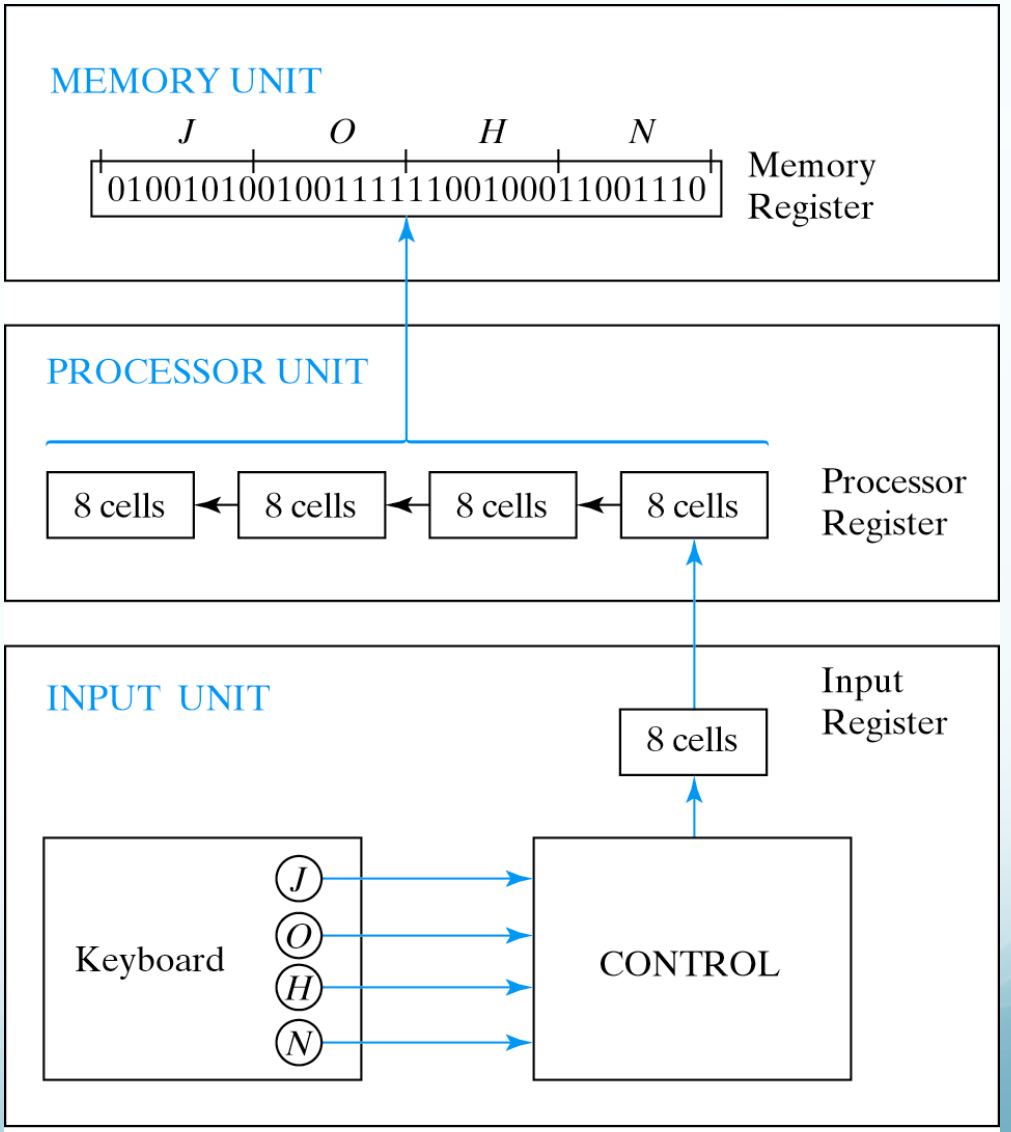


Fig. 1-1 Transfer of information with registers

Binary Information Processing

- The actual processing of binary information in a computer is completed by digital logic circuits which have been implemented to serve a specific purpose (i.e. addition).
  - The registers are accessed (read and write) when they are needed to complete an operation. For example we need two register sets to store two values to be added and a register set to store the result of the sum.
    - Furthermore, we need three registers in both the memory unit and in the processor.

Example Binary Information Processing

- We need processing
- We need storage
- We need communication

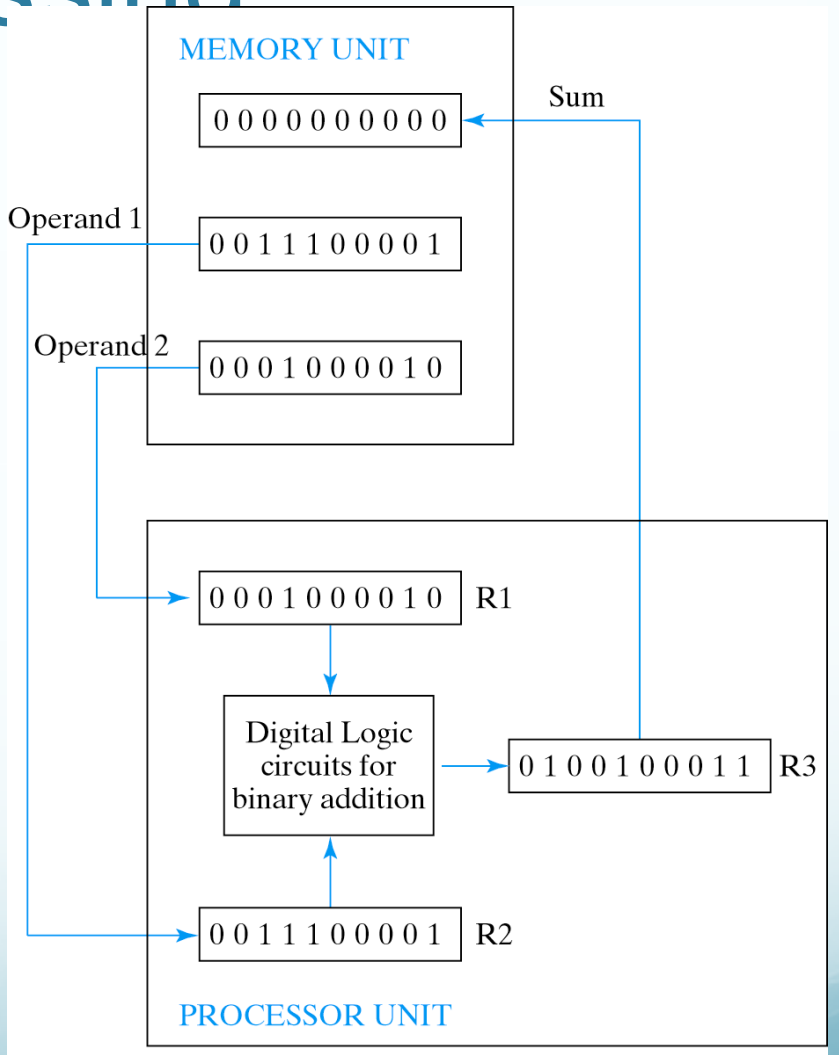


Fig. 1-2 Example of binary information processing

Binary Logic

- Binary logic consists of binary variables and logical operations.
  - The variables are designated by letters of the alphabet (A, B, C, x, y, z, etc.).
  - There are three basic logical operations:
    - AND
    - OR
    - NOT

Logical Operations

- AND is represented by a dot or the absence of an operator.
  - $x \cdot y = z$ or $xy = z$
  - Read as “x and y is equal to z”
  - Means that z=1 if and only if x=1 and y=1
- OR is represented by a plus sign.
  - $x+y = z$
  - Read as “x or y is equal to z”
  - Means that z=1 if x=1 or y=1 or both x=1 and y=1
- NOT is represented by a prime or an overbar.
  - $x'=z$ or $\overline{x} = z$
  - Read as not x is equal to z
  - Means that if x=1 then z=0 or if x=0 then z=1

Truth Tables

- Since each binary variable consists of value of 0 or 1, each combination of values for the variables involved in a binary operation has a specific result value.
- A truth table is a method of visualizing all possible combinations of the input values and the respective output values that occur due to the operation on the specified combination.

AND Truth Table

x	y	x·y
0	0	0
0	1	0
1	0	0
1	1	1

AND

A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

OR Truth Table

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

OR

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

NOT Truth Table

x	x'
0	1
1	0

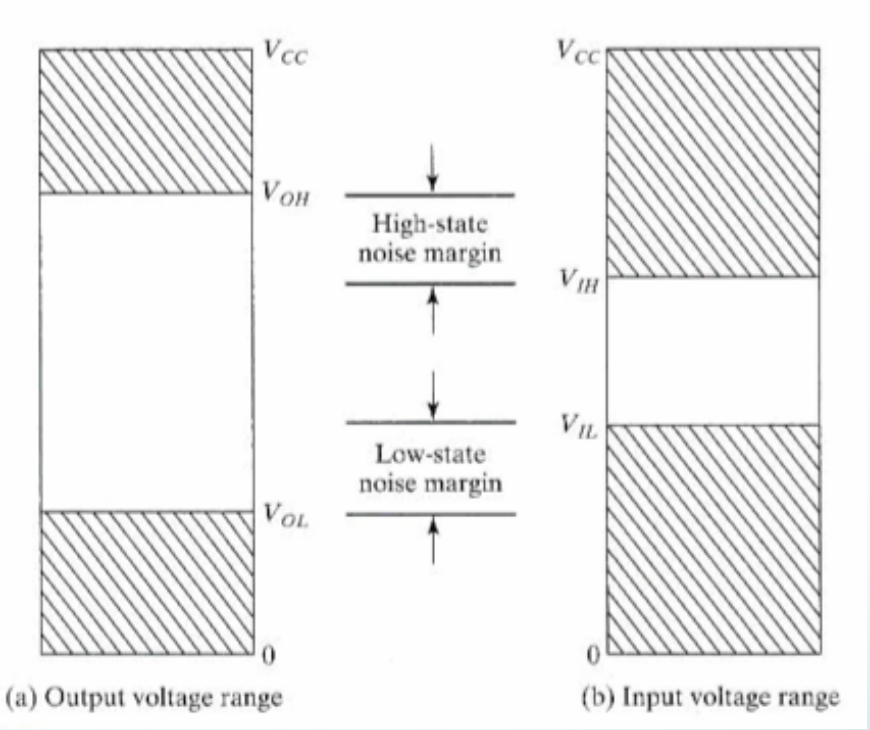
NOT

A	A'
0	1
1	0

Binary Signal

- Two separated voltage levels represents a binary variable equal to logic 1 or logic 0

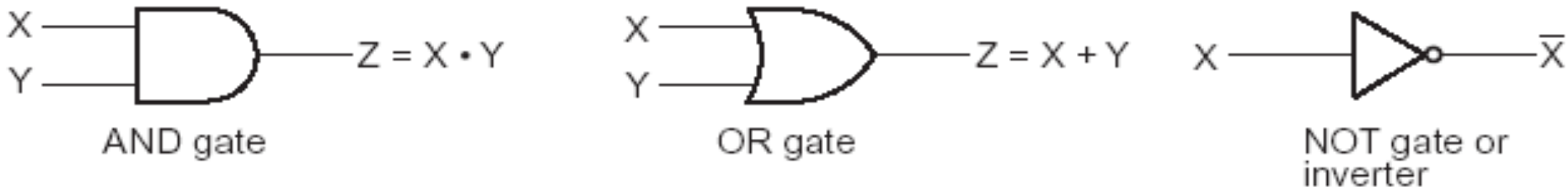
- **noise margin in a standard TTL NAND gate are *VOH*= 2.4 V, VoL = 0.4V, *VIH*= 2V, and *VIL*= 0.8V.**
- **The high-state noise margin is 2.4 - 2 = 0.4V, and the low-state noise margin is 0.8 - 0.4 = 0.4V**



Logic Gates

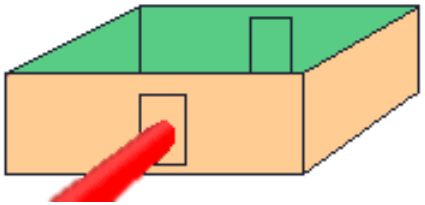
- Logic gates are electronic circuits that operate on one or more input signals to produce an output signal.
  - The state (high-low, on-off) of electricity on a line represents each of the two states for binary representation (1 or 0).

- Logic Gate Notation



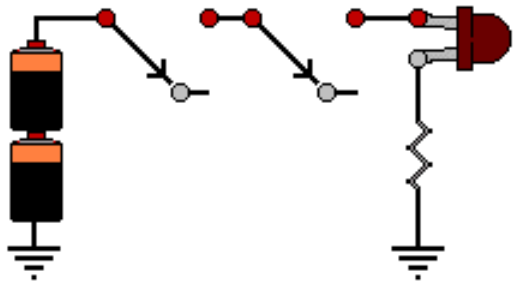
AND Logic Function

- Using door
  - Both doors are opened to pass the light



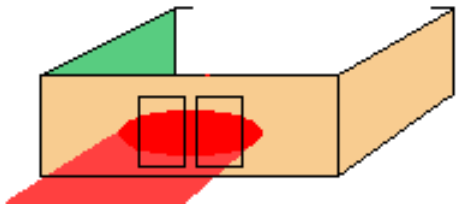
A	B	C
0	0	0
0	1	0
1	0	0
1	1	1

- **Using Switches**
  - **Switches are input and LED is output**
  - **Both switches closed (ON) to give output**



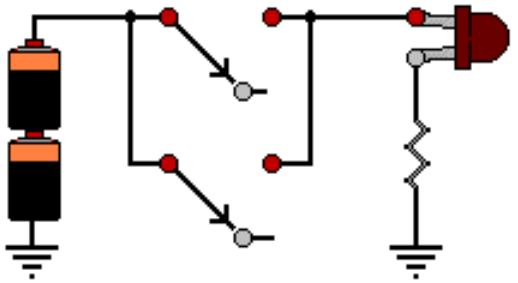
OR Logic Function

- Using door
  - Any one or both doors are opened to pass the light



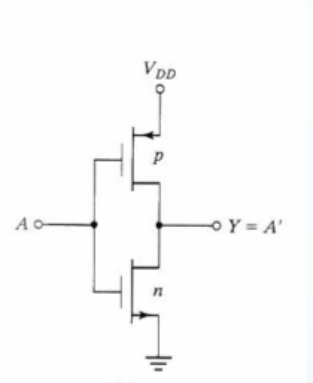
A	B	C
0	0	0
0	1	1
1	0	1
1	1	1

- **Using Switches**
  - **Switches are input and LED is output**
  - **Any one switch or both closed to give output**



IC Digital Logic Families

- **RTL** Resistor-transistor Logic
- **DTL** Diode Transistor Logic
- **TTL** Transistor-Transistor logic
- **ECL** Emitter-Coupled Logic
- **MOS** Metal-Oxide Semiconductor
- **CMOS** Complementary Metal-Oxide Semiconductor



Note : More information see Chapter 10

The End