DLD (DIGITAL LOGIC DESIGN)

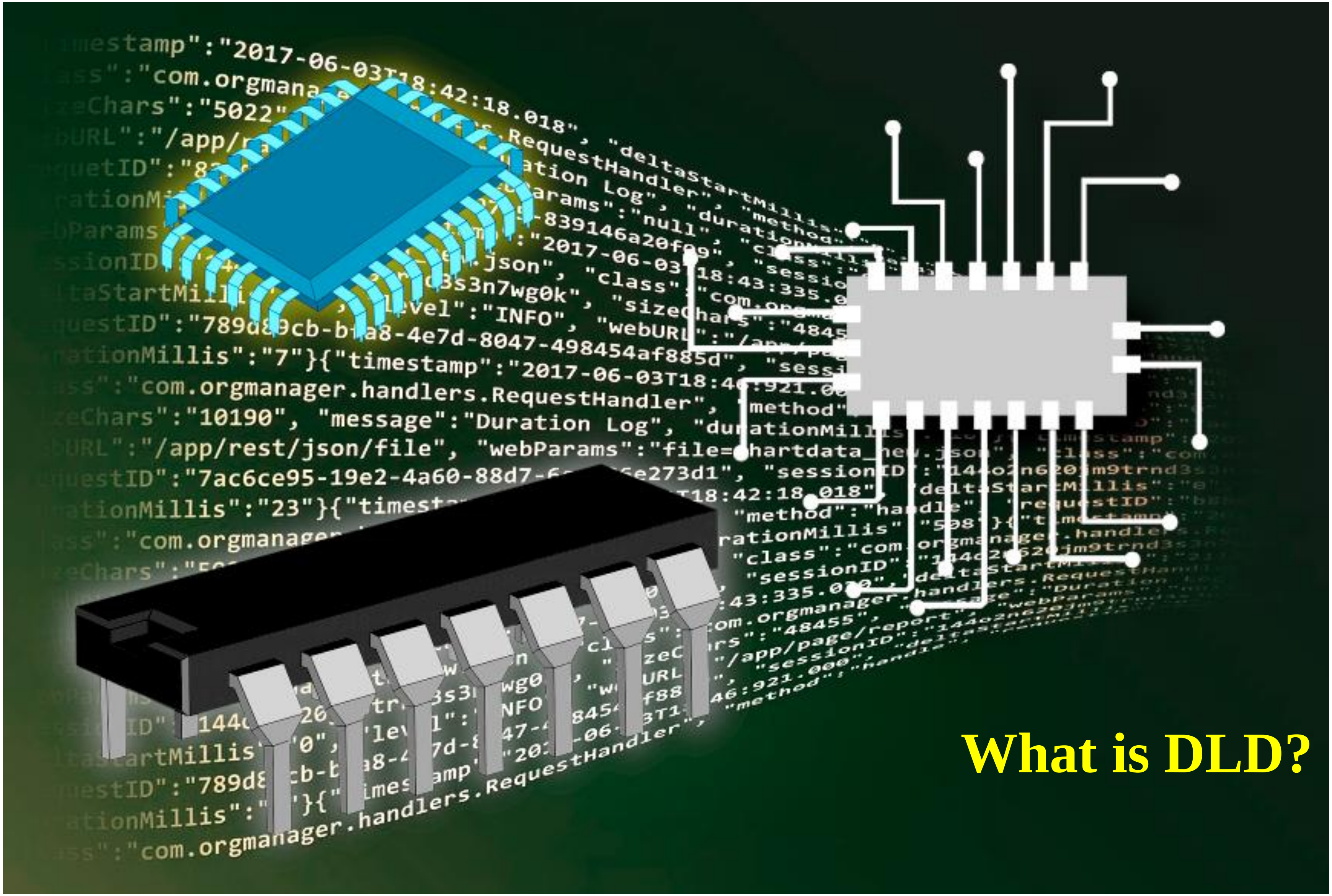
Dr. Kamal Kishor Jha

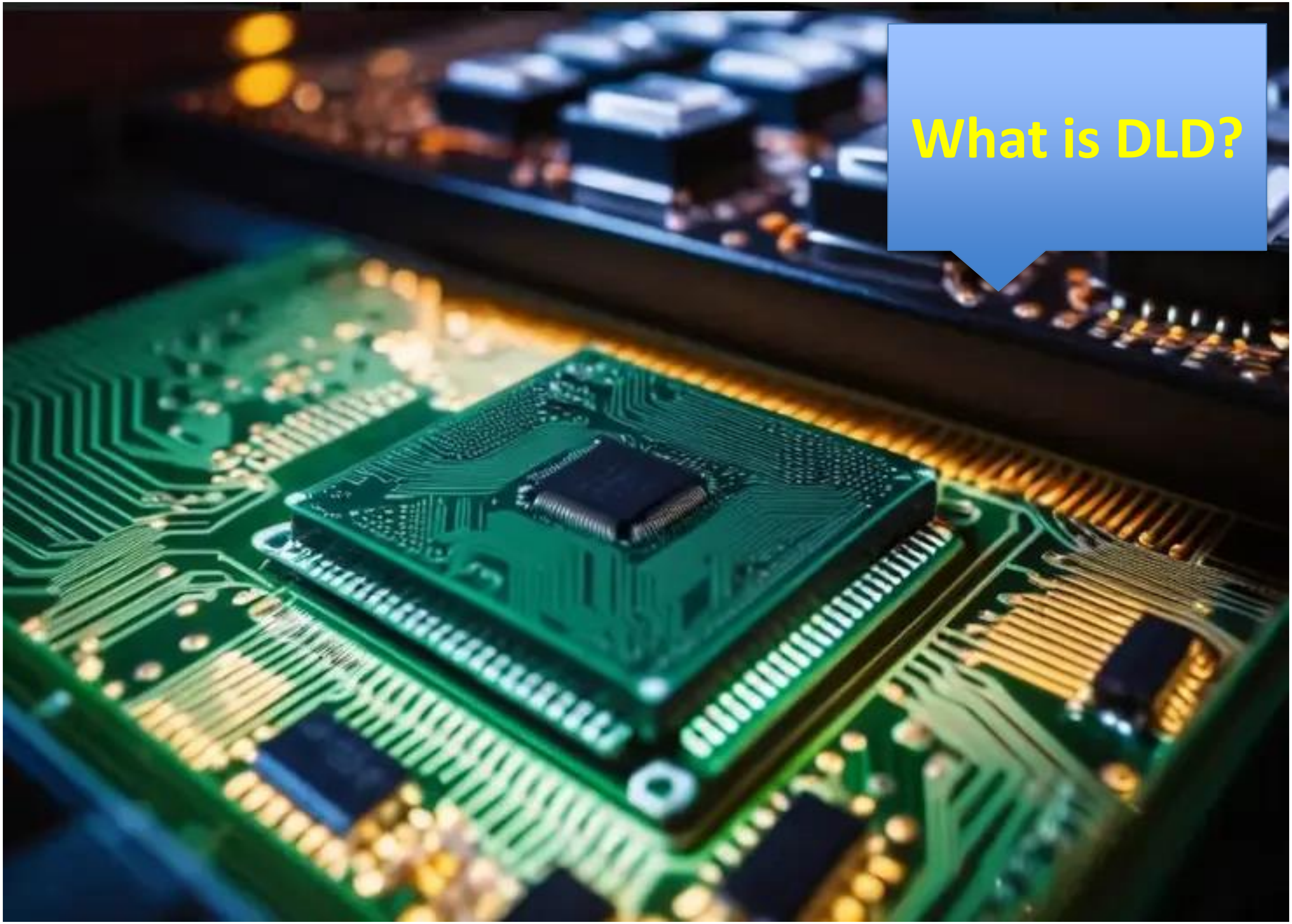
Assistant Professor

IIIT Vadodara

Textbooks

- “Digital Design” By M. Morris Mano.
- Complementary Material “Logic and Computer Design Fundamentals” by M. Morris Mano & Charles R Kime.
- “Digital Design”, F. Vahid, 2010 (2nd Edition).
- “Digital Design and Computer Architecture” , D.M. Harris and S.L. Harris, Morgan Kaufmann, 2013 (2nd Edition).


What is DLD?


What is DLD?

- Digital
  - Concerned with the interconnection among digital components and modules
    - » Best Digital System example is General Purpose Computer
- Logic Design
  - Deals with the basic concepts and tools used to design digital hardware consisting of logic circuits
    - » Circuits to perform arithmetic operations (+, -, x, ÷)

- Digital Signal : Decimal values are difficult to represent in electrical systems. It is easier to use two voltage values than ten.

- Digital Signals have two basic states:

  1 (logic "high", or H, or "on")

  0 (logic "low", or L, or "off")

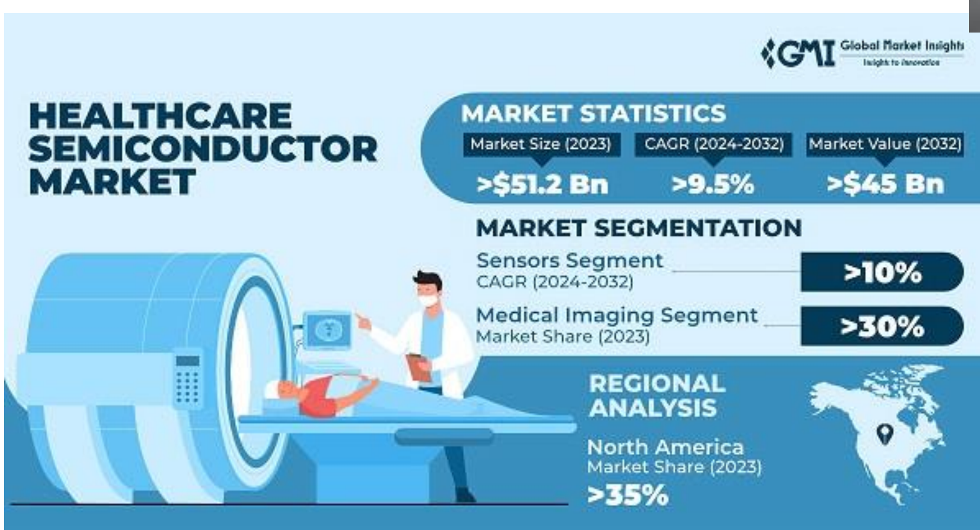
- Digital values are in a *binary* format. Binary means 2 states.

- A good example of binary is a light (only on or off)

Power switches have labels "1" for on and "0" for off.

Motivation

- Microelectronic technologies have revolutionized our world: cell phones, internet, rapid advances in medicine, etc.
- The Semiconductor industry grown tremendously.

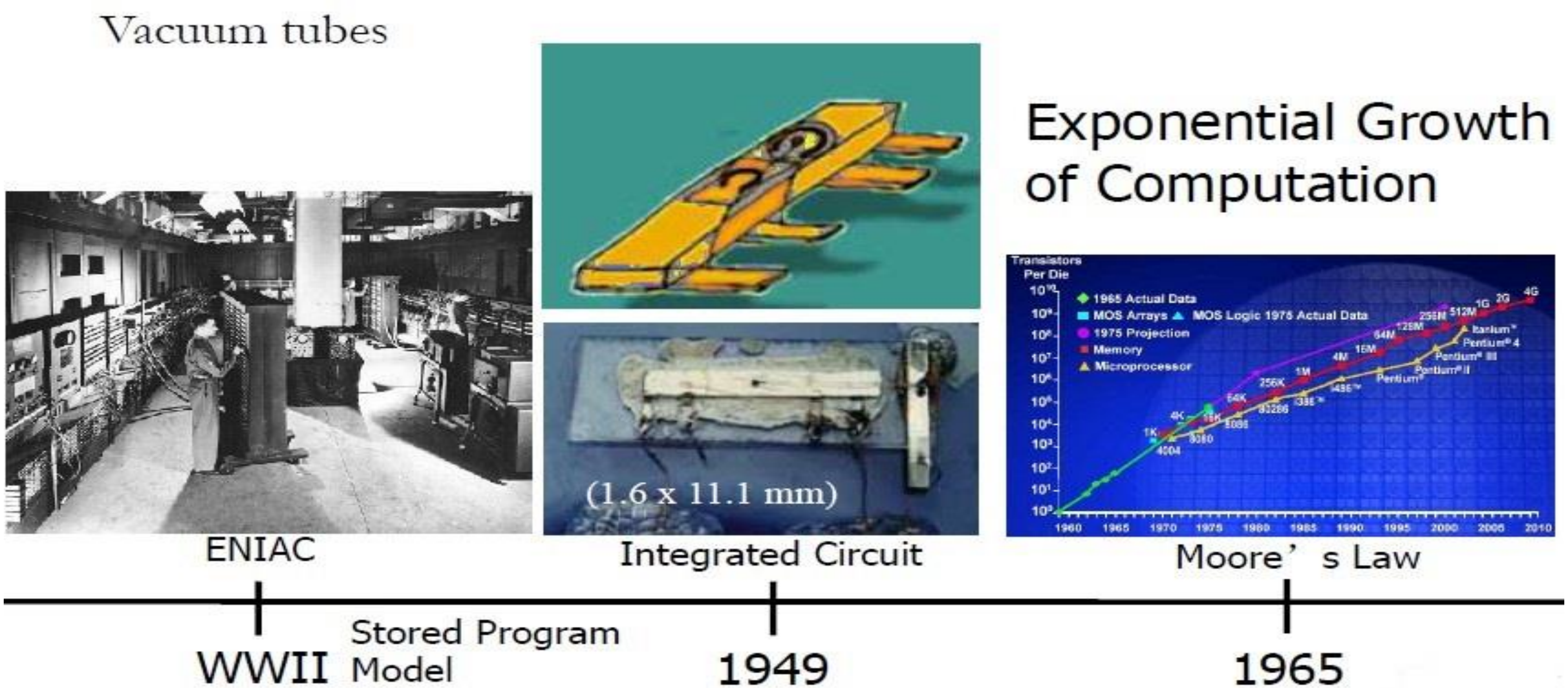


Motivation

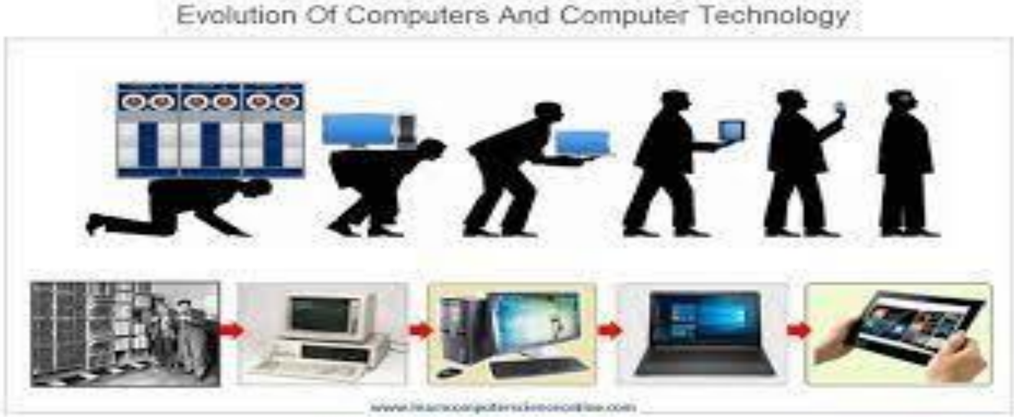
- The semiconductor industry is projected to reach $**726.73 billion by 2027.**

The Digital Revolution

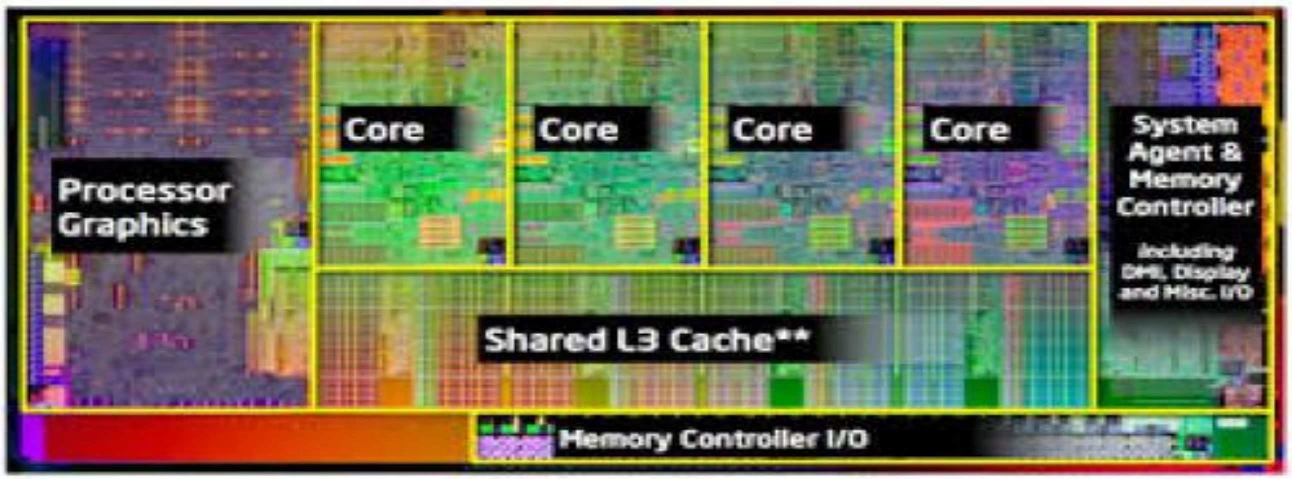
- IC (Integrated Circuit): Many digital operations on the same material.

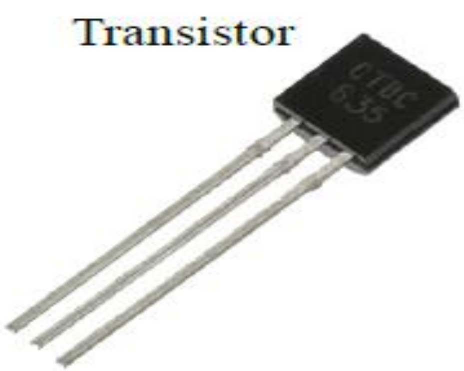


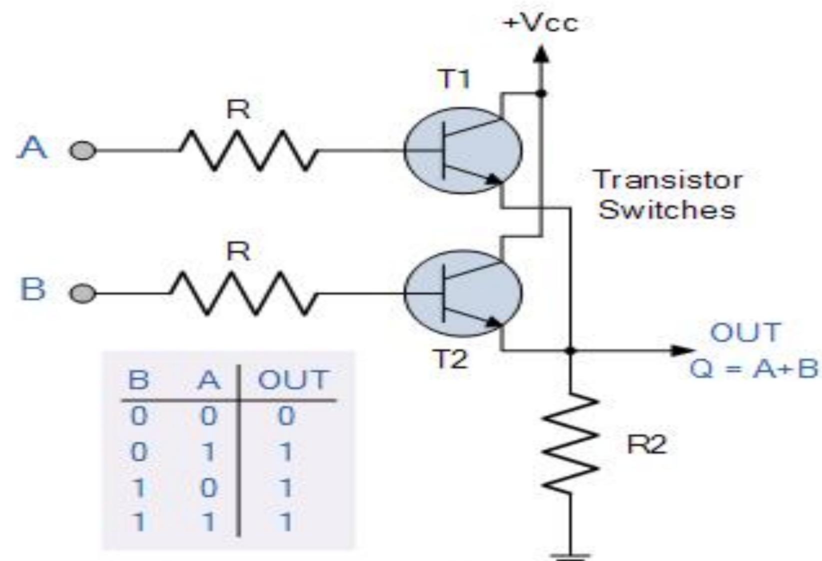
The Digital Revolution



Building complex circuits







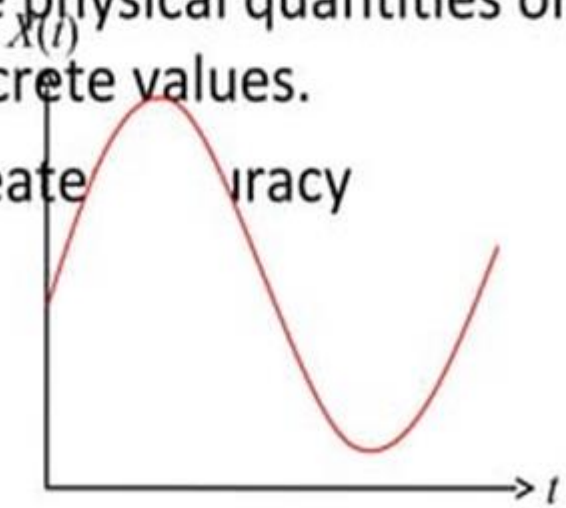
B	A	OUT
0	0	0
0	1	1
1	0	1
1	1	1

Scope

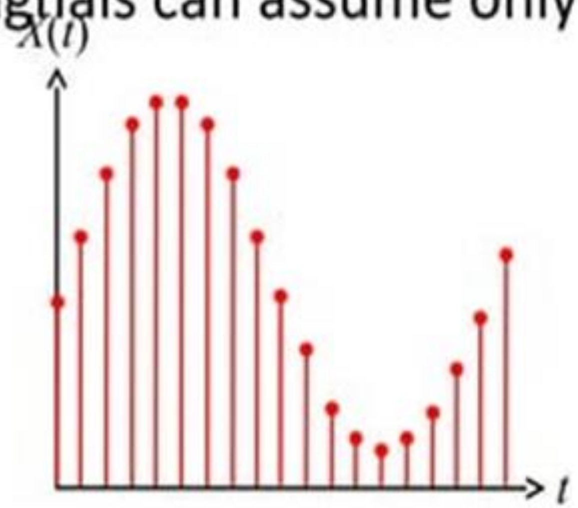
Subjects	**Building Blocks**	**Theory**
Combinational Logic	AND, OR, NOT, XOR	Boolean Algebra
Sequential Network	AND, OR, NOT, FF	Finite State Machine
Standard Modules	Operators, Interconnects, Memory	Arithmetics, Universal Logic
System Design	Data Paths, Control Paths	Methodologies

Analog and Digital Signal

- Analog system
  - The physical quantities or signals may vary continuously over a specified range.
- Digital system
  - The physical quantities or signals can assume only discrete values.
  - Greate[illegible]uracy



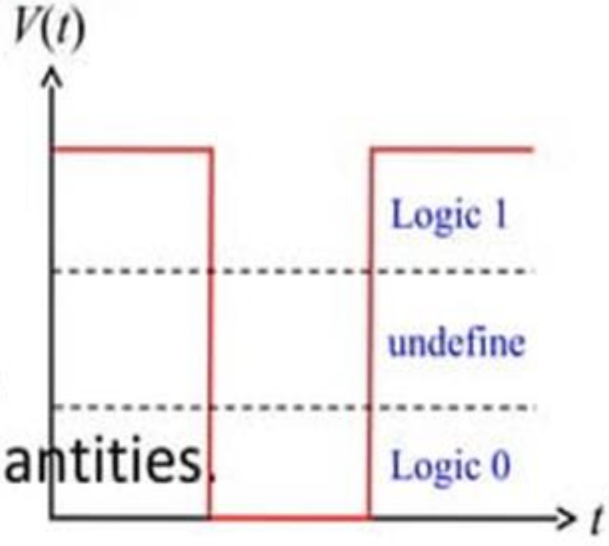
Analog signal



Digital signal

Binary Digital Signal

- An information variable represented by physical quantity.
- For digital systems, the variable takes on discrete values.
  - Two level, or binary values are the most prevalent values.
- Binary values are represented abstractly by:
  - Digits 0 and 1
  - Words (symbols) False (F) and True (T)
  - Words (symbols) Low (L) and High (H)
  - And words On and Off
- Binary values are represented by values or ranges of values of physical quantities.



Binary digital signal

Decimal Number System

- Base (also called radix) = 10
  - 10 digits { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }
- Digit Position
  - Integer & fraction
- Digit Weight
  - Weight = (*Base*) ^{Position}
- Magnitude
  - Sum of "*Digit* x *Weight*"
- Formal Notation

2	1	0	-1	-2
5	1	2	7	4
100	10	1	0.1	0.01
500	10	2	0.7	0.04

$$d_2 * B^2 + d_1 * B^1 + d_0 * B^0 + d_{-1} * B^{-1} + d_{-2} * B^{-2}$$

$$(512.74)_{10}$$

Octal Number System

- Base = 8
  - 8 digits { 0, 1, 2, 3, 4, 5, 6, 7 }
- Weights
  - Weight = (*Base*)^{Position}
- Magnitude
  - Sum of "*Digit* x *Weight*"
- Formal Notation

64	8	1	1/8	1/64
5	1	2	7	4
2	1	0	-1	-2

$5 * 8^{2} + 1 * 8^{1} + 2 * 8^{0} + 7 * 8^{-1} + 4 * 8^{-2}$

$=(330.9375)_{10}$

$(512.74)_{8}$

Binary Number System

- Base = 2
  - 2 digits { 0, 1 }, called *b*inary dig*its* or "*bits*"
- Weights
  - Weight = $(Base)^{Position}$
- Magnitude
  - Sum of "*Bit* x *Weight*"
- Formal Notation
- Groups of bits

4 bits = *Nibble*

8 bits = *Byte*

4	*2*	*1*	*1/2*	*1/4*
1	0	1	0	1
2	*1*	*0*	*-1*	*-2*

$$1 * 2^2 + 0 * 2^1 + 1 * 2^0 + 0 * 2^{-1} + 1 * 2^{-2}$$

$$= (5.25)_{10}$$

$$(101.01)_2$$

1 0 1 1

1 1 0 0 0 1 0 1

Hexadecimal Number System

- Base = 16
  - 16 digits { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F }
- Weights
  - Weight = $(Base)^{Position}$
- Magnitude
  - Sum of "*Digit* x *Weight*"
- Formal Notation

256	16	1		1/16	1/256
1	E	5	.	7	A
2	1	0		-1	-2

$$1 * 16^{2} + 14 * 16^{1} + 5 * 16^{0} + 7 * 16^{-1} + 10 * 16^{-2}$$

$$=(485.4765625)_{10}$$

$$(1E5.7A)_{16}$$

The Power of 2

n	2^n	n	2^n	
0	$2^0=1$	8	$2^8=256$	
1	$2^1=2$	9	$2^9=512$	
2	$2^2=4$	10	$2^{10}=1024$	Kilo
3	$2^3=8$	11	$2^{11}=2048$	
4	$2^4=16$	12	$2^{12}=4096$	
5	$2^5=32$	20	$2^{20}=1M$	Mega
6	$2^6=64$	30	$2^{30}=1G$	Giga
7	$2^7=128$	40	$2^{40}=1T$	Tera

Addition

- Decimal Addition

$$
\begin{array}{rrrl}
1 & 1 & & \leftarrow \textbf{Carry} \\
 & 5 & 5 & \\
+ & 5 & 5 & \\
\hline
1 & 1 & 0 &
\end{array}
$$

= ***Ten*** ≥ ***Base***

➔ **Subtract a Base**

Binary Addition

- Column Addition

	1	1	1	1	1	1		
		1	1	1	1	0	1	= 61
+			1	0	1	1	1	= 23
	1	0	1	0	1	0	0	= 84

$\geq (2)_{10}$

Binary Subtraction

- Borrow a "Base" when needed

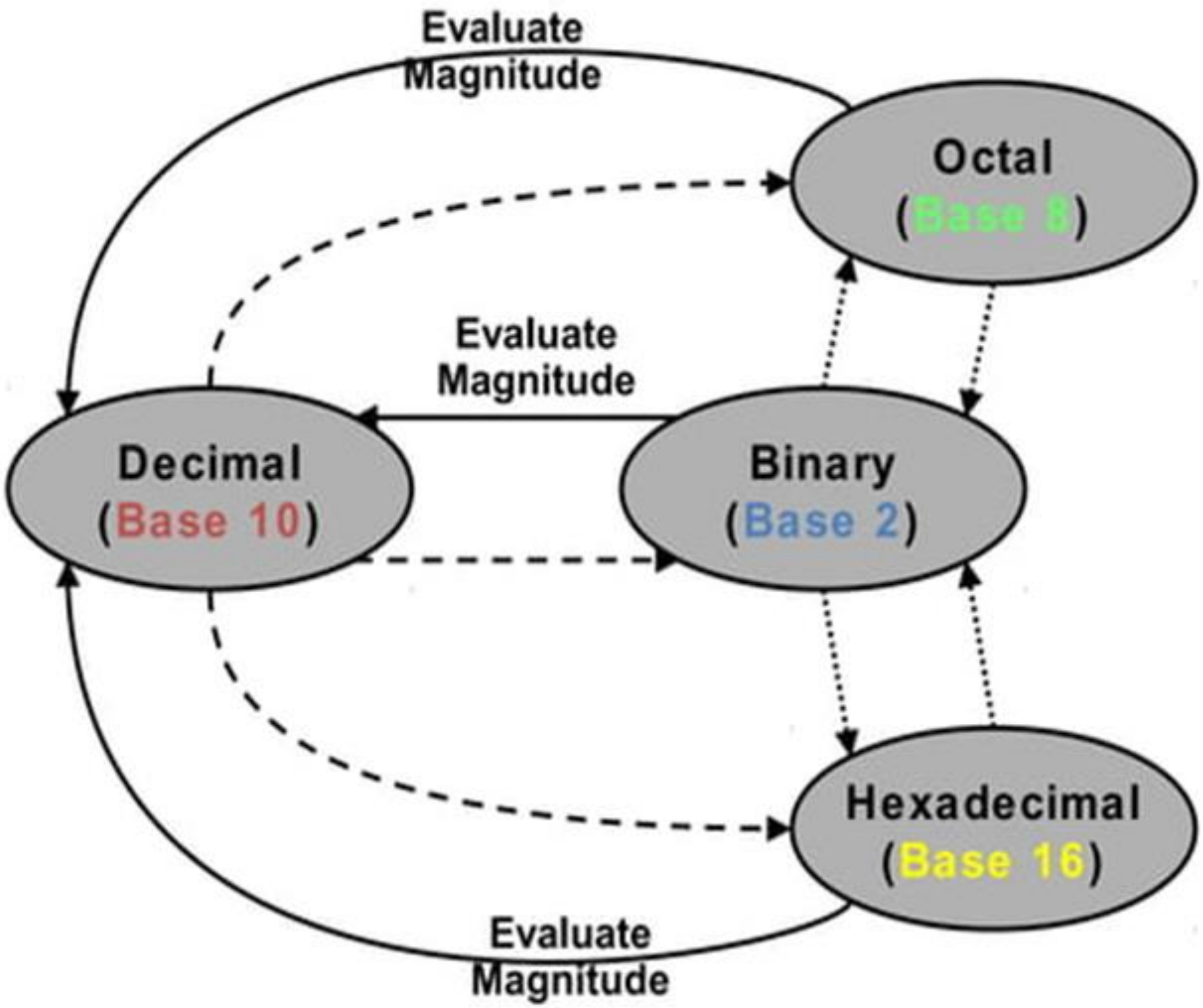
		1			2			$= (10)_2$
	0	$\cancel{2}$	2	0	0	2		
	$\cancel{1}$	0	0	$\cancel{1}$	$\cancel{1}$	0	1	= 77
−			1	0	1	1	1	= 23
	0	**1**	**1**	**0**	**1**	**1**	**0**	= 54

Binary Multiplication

- Bit by bit

				1	0	1	1	1
x					1	0	1	0
				0	0	0	0	0
			1	0	1	1	1	
		0	0	0	0	0		
	1	0	1	1	1			
	1	1	1	0	0	1	1	0

Number Base Conversions



Decimal (*Integer*) to Binary Conversion

- Divide the number by the 'Base' (=2)
- Take the remainder (either 0 or 1) as a coefficient
- Take the quotient and repeat the division

Example: $(13)_{10}$

	Quotient	Remainder	Coefficient
$13 / 2 =$	6	1	$a_0 = 1$
$6 / 2 =$	3	0	$a_1 = 0$
$3 / 2 =$	1	1	$a_2 = 1$
$1 / 2 =$	0	1	$a_3 = 1$

Answer: $(13)_{10} = (a_3\, a_2\, a_1\, a_0)_2 = (1101)_2$

MSB LSB

Decimal (*Fraction*) to Binary Conversion

- Multiply the number by the 'Base' (=2)
- Take the integer (either 0 or 1) as a coefficient
- Take the resultant fraction and repeat the division

Example: $(0.625)_{10}$

		Integer		Fraction	Coefficient
0.625	* 2 =	1	.	25	$a_{-1} = 1$
0.25	* 2 =	0	.	5	$a_{-2} = 0$
0.5	* 2 =	1	.	0	$a_{-3} = 1$

Answer: $(0.625)_{10} = (0.a_{-1}\, a_{-2}\, a_{-3})_2 = (0.101)_2$

MSB LSB

Decimal to Octal Conversion

Example: $(175)_{10}$

	Quotient	Remainder	Coefficient
175 / 8 =	21	7	$a_0 = 7$
21 / 8 =	2	5	$a_1 = 5$
2 / 8 =	0	2	$a_2 = 2$

Answer: $(175)_{10} = (a_2\ a_1\ a_0)_8 = (257)_8$

Example: $(0.3125)_{10}$

	Integer		Fraction	Coefficient
0.3125 * 8 =	2	.	5	$a_{-1} = 2$
0.5 * 8 =	4	.	0	$a_{-2} = 4$

Answer: $(0.3125)_{10} = (0.a_{-1}\ a_{-2}\ a_{-3})_8 = (0.24)_8$

Binary – Octal Conversion

- $8 = 2^3$
- Each group of 3 bits represents an octal digit

Octal	Binary
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Example:

Assume Zeros

$$(10110.01)_2$$

$$(2\ \ 6\ .\ 2)_8$$

Works both ways (*Binary* to *Octal* & *Octal* to *Binary*)

Binary – Hexadecimal Conversion

- $16 = 2^4$
- Each group of 4 bits represents a hexadecimal digit

Assume Zeros

Example: $(10110.01)_2$

$(1\quad 6\quad .\quad 4)_{16}$

Hex	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111

Works both ways (*Binary* to *Hex* & *Hex* to *Binary*)

Octal – Hexadecimal Conversion

- Convert to Binary as an intermediate step

Example:

$$(2 \quad 6 \;.\; 2)_8$$

Assume Zeros

Assume Zeros

$$(0\,1\,0\,1\,1\,0\,.\,0\,1\,0)_2$$

$$(1 \quad 6 \;.\; 4)_{16}$$

Works both ways (*Octal* to *Hex* & *Hex* to *Octal*)

Complements

- 1’s Complement (*Diminished Radix* Complement)
  - All ‘0’s become ‘1’s
  - All ‘1’s become ‘0’s

Example $(10110000)_2$

⇨ $(01001111)_2$

If you add a number and its 1’s complement …

$$
\begin{array}{r}
10110000 \\
+\ 01001111 \\
\hline
11111111
\end{array}
$$

Complements

- 2's Complement (*Radix* Complement)
  - Take 1's complement then add 1

OR

  - Toggle all bits to the left of the first '1' from the right

Example:

Number:	10110000	10110000
1's Comp.:	01001111	
	+ 1	
	01010000	01010000

Boolean Algebra and Switching Functions

Two-input AND (·)

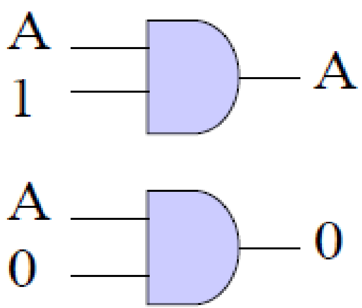
AND	A	B	Y
	0	0	0
	0	1	0
	1	0	0
	1	1	1

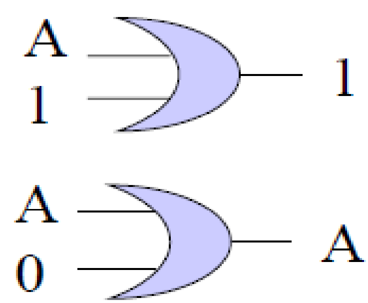
Two-input OR (+)

OR	A	B	Y
	0	0	0
	0	1	1
	1	0	1
	1	1	1

One-input NOT (Complement, ’)

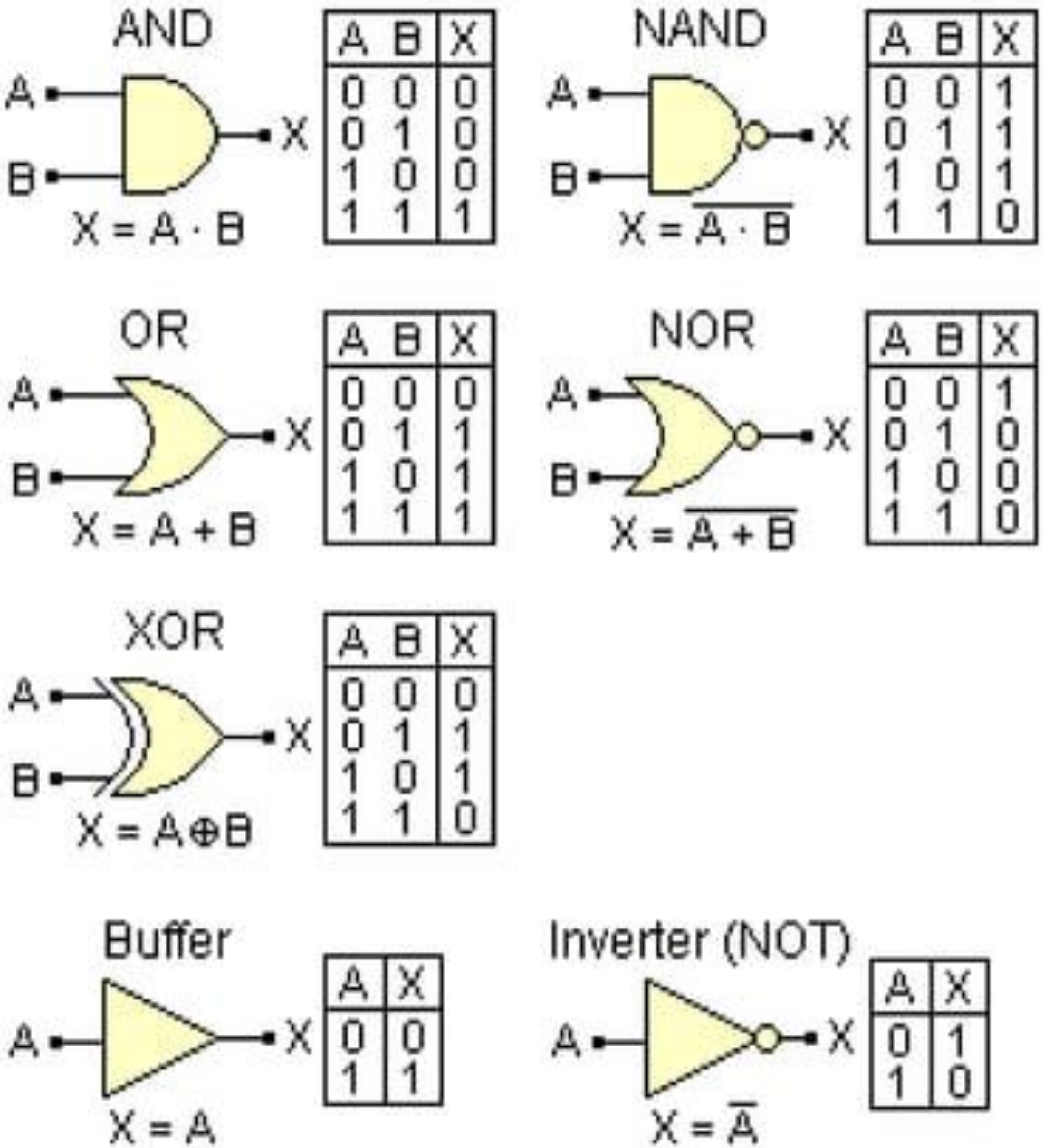
NOT	A	Y
	0	1
	1	0





For an AND gate,
0 at input blocks the other inputs and dominates the output
1 at input passes signal A

For an OR gate,
1 at input blocks the other inputs and dominates the output
0 at input passes signal A

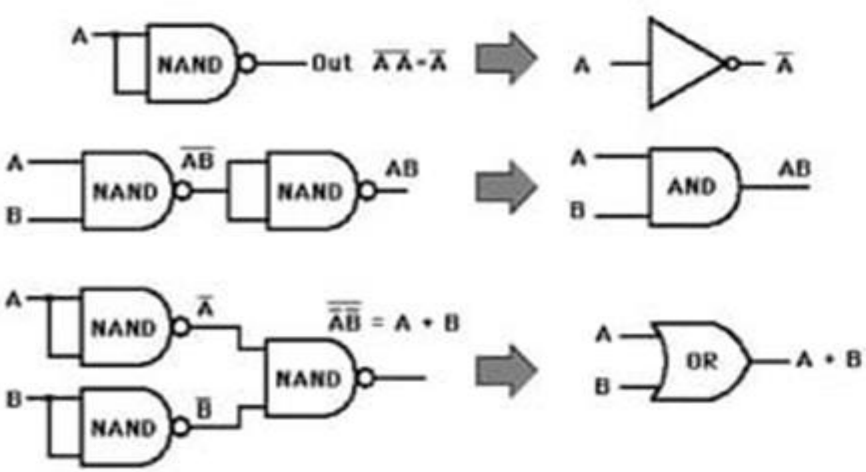


Logic Function	Boolean Notation
AND	A.B
OR	A+B
NOT	$\overline{A}$
NAND	$\overline{A.B}$
NOR	$\overline{A+B}$
EX-OR	$(A.\overline{B}) + (\overline{A}.B)$ or $A \oplus B$
EX-NOR	$(\overline{A}.\overline{B})$ + or $\overline{A \oplus B}$

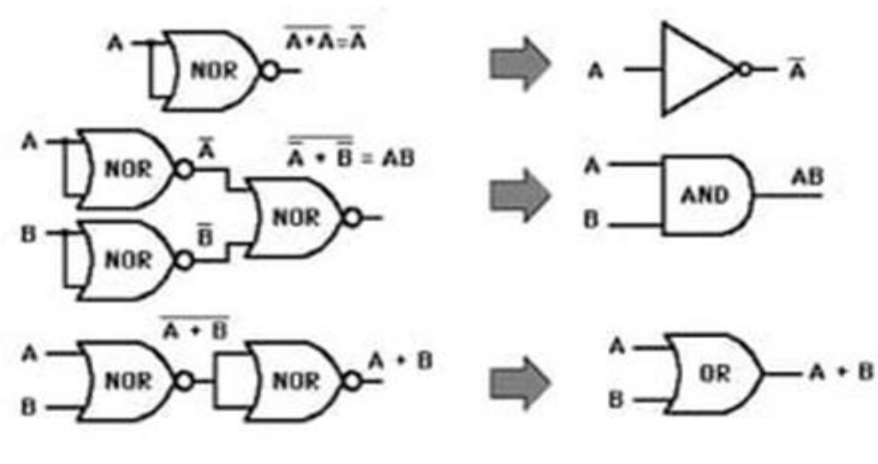
Universal Gate

- **NAND and NOR** Gates are called ***Universal Gates*** because AND, OR and NOT gates can be implemented &created by using these gates.

NAND Gate Implementations



NOR Gate Implementations



Boolean Operations and Expressions

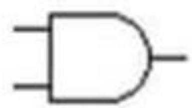
- Boolean Addition
  - Logical OR operation

Ex 4-1) Determine the values of A, B, C, ar... ...ke the sum term $A+B'+C+D'$

Sol) all literals must be '0' for the sum term to be '0'

$A+B'+C+D'=0+1'+0+1'=0 \rightarrow A=0, B=1, C=0,$ and $D=1$

- Boolean Multiplication
  - Logical AND operation



Ex 4-2) Determine the values of A, B, C, a 1

Sol) all literals must be '1' for the product term to be '1'

$AB'CD'=10'10'=1 \rightarrow A=1, B=0, C=1,$ and $D=0$

Basic Identities of Boolean Algebra

Basic Identities of Boolean Algebra

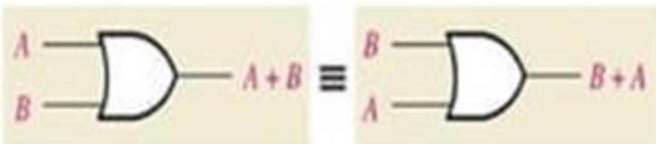
1.	$X+0=X$	2.	$X \cdot 1 = X$	
3.	$X+1=1$	4.	$X \cdot 0 = 0$	
5.	$X+X=X$	6.	$X \cdot X = X$	
7.	$X+\overline{X}=1$	8.	$X \cdot \overline{X} = 0$	
9.	$\overline{\overline{X}}=X$			
10.	$X+Y=Y+X$	11.	$XY=YX$	Commutative
12.	$X+(Y+Z)=(X+Y)+Z$	13.	$X(YZ)=(XY)Z$	Associative
14.	$X(Y+Z)=XY+XZ$	15.	$X+YZ=(X+Y)(X+Z)$	Distributive
16.	$\overline{X+Y}=\overline{X}\cdot\overline{Y}$	17.	$\overline{X \cdot Y}=\overline{X}+\overline{Y}$	DeMorgan's

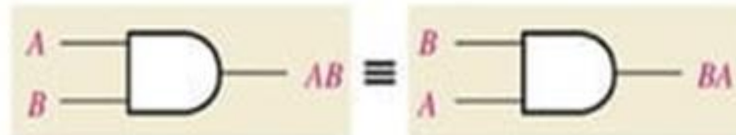
The relationship between a single variable X, its complement X', and the binary constants 0 and 1

Laws of Boolean Algebra

- Commutative Law: the order of literals does not matter

$A + B = B + A$ $\qquad$ $A\,B = B\,A$

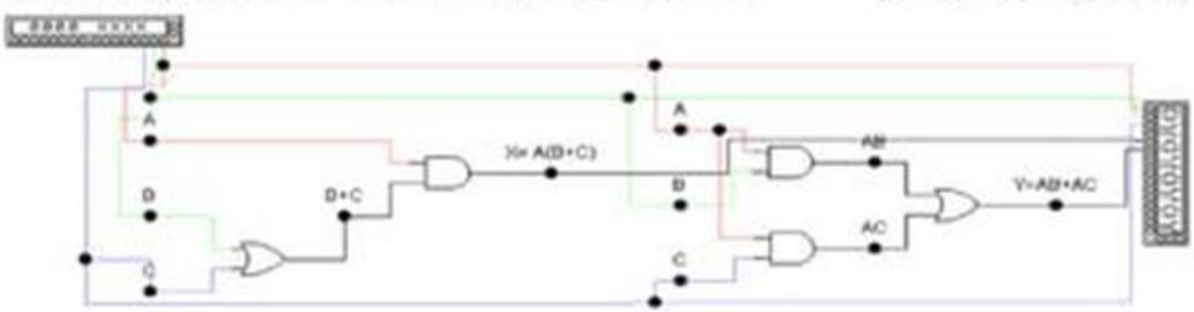




- Associative Law: the grouping of literals does not matter

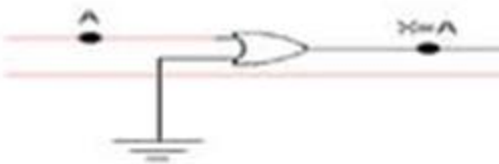
$A + (B + C) = (A + B) + C$ (=A+B+C) $\qquad$ $A(BC) = (AB)C$ (=ABC)

- Distributive Law : $A(B + C) = AB + AC$ $\qquad$ $(A+B)(C+D) = AC + AD + BC + BD$



Rules of Boolean Algebra

- ✓ $A+0=A$ In math if you add 0 you have changed nothing in Boolean Algebra ORing with 0 changes nothing



- ✓ $A \cdot 0=0$ In math if 0 is multiplied with you get 0. If you AND anything with 0 you get 0
- ✓ $A \cdot 1 = A$ ANDing anything with 1 will yield the anything
- ✓ $A+A = A$ ORing with itself will give the same result
- ✓ $A+A'=1$ Either A or A′ must be 1 so A + A′ =1
- ✓ $A \cdot A = A$ ANDing with itself will give the same result
- ✓ $A \cdot A' = 0$ In digital Logic 1′ =0 and 0′ =1, so AA′=0 since one of the inputs must be 0.
- ✓ $A = (A')'$ If you not something twice you are back to the beginning

✓ $A + A'B = A + B$

If A is 1 the output is 1   If A is 0 the output is B

✓ $A + AB = A$

✓ $(A + B)(A + C) = A + BC$

- DeMorgan's Theorem
  - $F'(A, A', \cdot, +, 1, 0) = F(A', A, +, \cdot, 0, 1)$
  - $(A \bullet B)' = A' + B'$ and $(A + B)' = A' \bullet B'$
  - DeMorgan's theorem will help to simplify digital circuits using NORs and NANDs his theorem states

NAND   Negative-OR

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

NOR   Negative-AND

Inputs		Output	
X	Y	$\overline{X+Y}$	$\overline{X}\,\overline{Y}$
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

Thank You