



20MCA13C COMPUTER ORGANIZATION AND ARCHITECTURE
UNIT I: : Binary Systems

FACULTY

Dr. K. ARTHI MCA, M.Phil., Ph.D.,
Assistant Professor,
Postgraduate Department of Computer Applications,
Government Arts College (Autonomous),
Coimbatore-641018.

20MCA13C COMPUTER ORGANIZATION AND ARCHITECTURE

UNIT I: Binary Systems: Digital Computers and Digital systems - Binary Numbers - Number Base Conversions - Octal and Hexadecimal number - Complements - Binary codes. **Boolean Algebra and Logic Gates:** Basic Definition - Axiomatic Definition of Boolean Algebra - Basic Theorems and Properties of Boolean Algebra - Boolean Functions - Canonical and Standard forms - Other Logic operations-Digital Logic Gates- Simplifications of Boolean Function.

UNIT II: Combinational Logic: Introduction - Design Procedure - Adders - Subtractors - Code Conversions - Multiplexer - Demultiplexer - Encoder - Decoder. **Sequential Logic:** Introduction - Flip-Flops - triggering Flip-flop - Excitation Tables - Registers - Shift registers - Ripple Counters - Synchronous counters - Timing Sequences.

UNIT III: Register Transfer Logic: Introduction -Arithmetic, Logic and Shift Micro-operations - Fixed Point Binary data - Arithmetic Shifts - Instruction Codes. **Micro Computer System Design:** Introduction - Instructions and Addressing modes - Stack, Subroutines and Interrupt - Input - Output interface- Direct Memory Access.

UNIT IV: CPU Organization: General Register Organization Types of Interrupts - RISC - Parallel Processing - Pipelining - Array Processors - Performance of a processor. **Input-Output Organization:** Peripheral Devices - Asynchronous Data Transfer (Strobe & Handshaking Method) - Modes of Transfer - Priority Interrupt - IOP.

UNIT V: Memory Organization: Types of Memory - Memory Hierarchy - Main Memory - Memory interface to CPU - Associative Memory - Cache Memory: Cache mapping schemes - Virtual Memory.

TEXT BOOKS:

1.Morris Mano M, “Digital Logic and Computer Design”, Pearson Education, 2016.

2.Morris Mano M, “Computer System Architecture”, Pearson Education, 2012.

REFERENCE BOOKS:

1.John Patrick Hayes, “Computer Architecture and Organization”, Tata McGraw Hill, 2007.

2.Albert Paul Malvino, Donald P. Leach, “Digital Principles and Applications”, Tata McGraw Hill, 2002



- **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**)

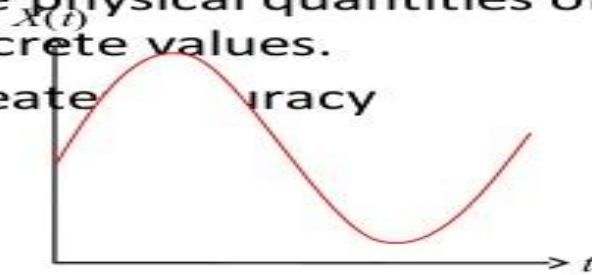
on



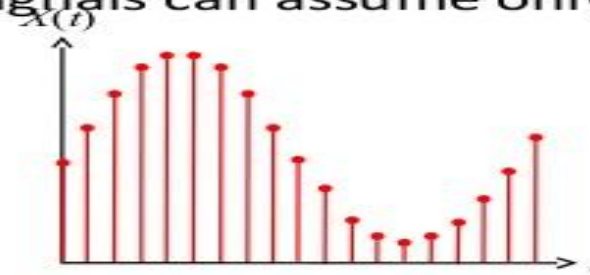
Power switches have labels “1” for on and “0” for off.

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.
 - Greater accuracy



Analog signal

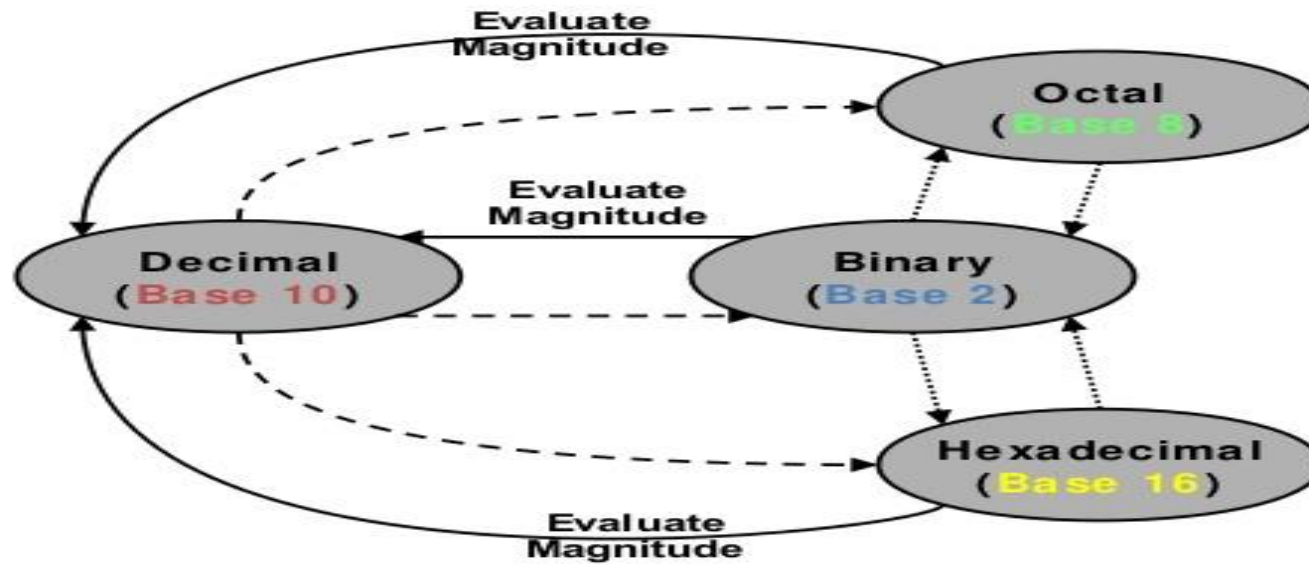


Digital signal

Digital Systems and Binary Numbers

- Digital age and information age
- Digital computers
 - General purposes
 - Many scientific, industrial and commercial applications
- Digital systems
 - Telephone switching exchanges
 - Digital camera
 - Electronic calculators, PDA's
 - Digital TV
- Discrete information-processing systems
 - Manipulate discrete elements of information
 - For example, {1, 2, 3, ...} and {A, B, C, ...}...

Number Base Conversions

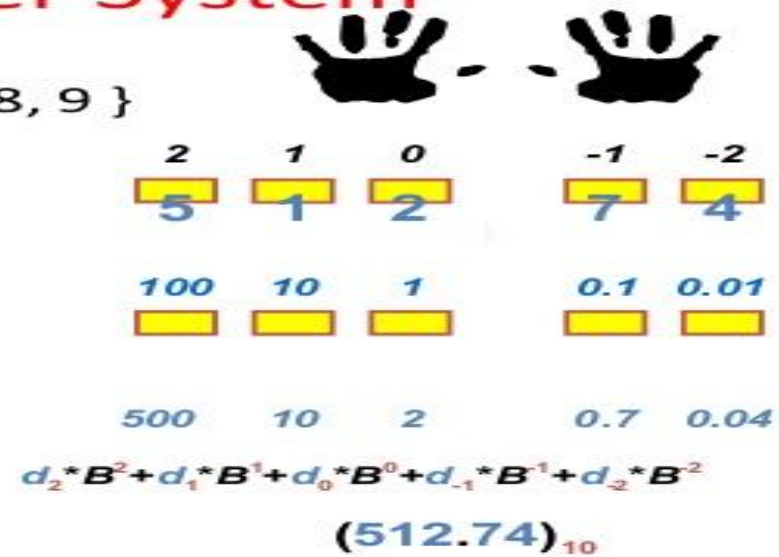


► TRUTH TABLE

Binary	Hex	Decimal
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

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



Binary Number System

- Base = 2
 - 2 digits { 0, 1 }, called *binary digits* 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

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

$$5 * 8^2 + 1 * 8^1 + 2 * 8^0 + 7 * 8^{-1} + 4 * 8^{-2}$$
$$= (330.9375)_{10}$$
$$(512.74)_8$$

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



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	$\mathbf{a_0 = 7}$
$21 / 8 =$	2	5	$\mathbf{a_1 = 5}$
$2 / 8 =$	0	2	$\mathbf{a_2 = 2}$

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

Example: $(0.3125)_{10}$

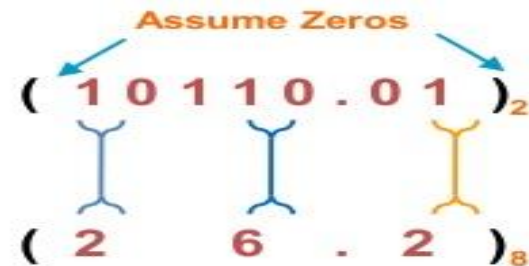
	Integer	Fraction	Coefficient
$0.3125 * 8 =$	2	5	$\mathbf{a_{-1} = 2}$
$0.5 * 8 =$	4	0	$\mathbf{a_{-2} = 4}$

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

Binary – Octal Conversion

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

Example:



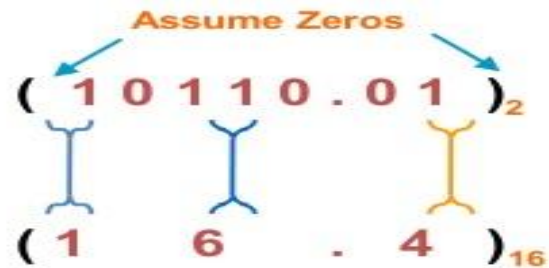
Works **both** ways (*Binary to Octal & Octal to Binary*)

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

Binary – Hexadecimal Conversion

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

Example:



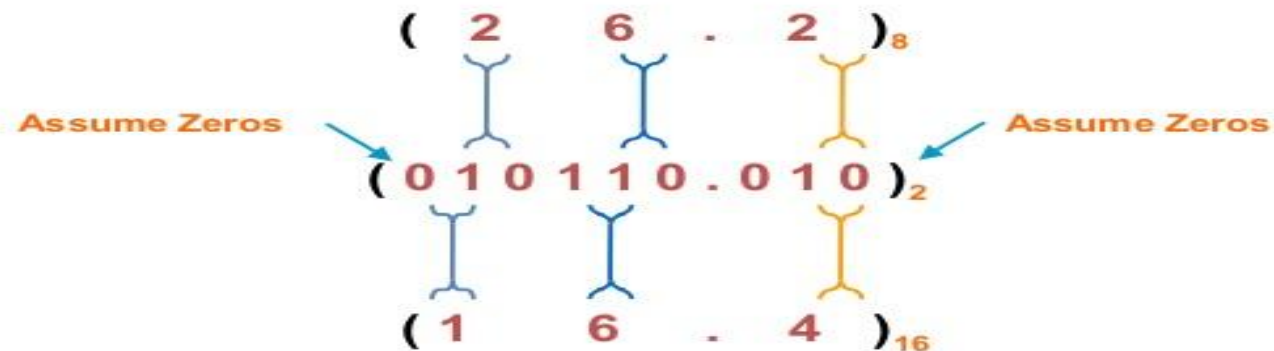
Works **both** ways (*Binary to Hex & Hex to Binary*)

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

Octal – Hexadecimal Conversion

- Convert to **Binary** as an intermediate step

Example:



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

1.5 Complements

- There are two types of complements for each base- r system: the radix complement and diminished radix complement.
- **Diminished Radix Complement - $(r-1)$'s Complement**
 - Given a number N in base r having n digits, the $(r-1)$'s complement of N is defined as:
$$(r^n - 1) - N$$
- **Example for 6-digit decimal numbers:**
 - 9's complement is $(r^n - 1) - N = (10^6 - 1) - N = 999999 - N$
 - 9's complement of 546700 is $999999 - 546700 = 453299$
- **Example for 7-digit binary numbers:**
 - 1's complement is $(r^n - 1) - N = (2^7 - 1) - N = 1111111 - N$
 - 1's complement of 1011000 is $1111111 - 1011000 = 0100111$
- **Observation:**
 - Subtraction from $(r^n - 1)$ will never require a borrow
 - Diminished radix complement can be computed digit-by-digit
 - For binary: $1 - 0 = 1$ and $1 - 1 = 0$

Complements

- **Radix Complement**

The r 's complement of an n -digit number N in base r is defined as $r^n - N$ for $N \neq 0$ and as 0 for $N = 0$. Comparing with the $(r - 1)$'s complement, we note that the r 's complement is obtained by adding 1 to the $(r - 1)$'s complement, since $r^n - N = [(r^n - 1) - N] + 1$.

- **Example: Base-10**

The 10's complement of 012398 is 987602

The 10's complement of 246700 is 753300

- **Example: Base-2**

The 2's complement of 1101100 is 0010100

The 2's complement of 0110111 is 1001001

Complements

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

Example $(10110000)_2$

$\Rightarrow (01001111)_2$

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:	1 0 1 1 0 0 0 0	1 0 1 1 0 0 0 0
1's Comp.:	0 1 0 0 1 1 1 1	
	+ 1	

	0 1 0 1 0 0 0 0	0 1 0 1 0 0 0 0



Complements

- Subtraction with Complements
 - The subtraction of two n -digit unsigned numbers $M - N$
 1. Add the minuend M to the r 's complement of the subtrahend N . Mathematically, $M + (r^n - N) = M - N + r^n$.
 2. If $M \geq N$, the sum will produce an end carry r^n , which can be discarded; what is left is the result $M - N$.
 3. If $M < N$, the sum does not produce an end carry and is equal to $r^n - (N - M)$, which is the r 's complement of $(N - M)$. To obtain the answer in a familiar form, take the r 's complement of the sum and place a negative sign in front.

Complements

- Example 1.5

– Using 10's complement, subtract $72532 - 3250$.

	$M =$	72532
10's complement of	$N =$	$+96750$
	Sum =	169282
	Discard end carry $10^5 =$	-100000
	Answer =	69282

- Example 1.6

– Using 10's complement, subtract $3250 - 72532$.

	$M =$	03250
10's complement of	$N =$	$+27468$
	Sum =	30718

→ There is no end carry.

→ Therefore, the answer is $-(10\text{'s complement of } 30718) = -69282$.

Complements

- Subtraction of unsigned numbers can also be done by means of the $(r - 1)$'s complement. Remember that the $(r - 1)$'s complement is one less than the r 's complement.
- Example 1.8
 - Repeat Example 1.7, but this time using 1's complement.

(a) $X - Y = 1010100 - 1000011$	
$X =$	1010100
1's complement of $Y =$	± 0111100
Sum =	10010000
End-around carry =	$\underline{+ \quad 1}$
Answer. $X - Y =$	0010001

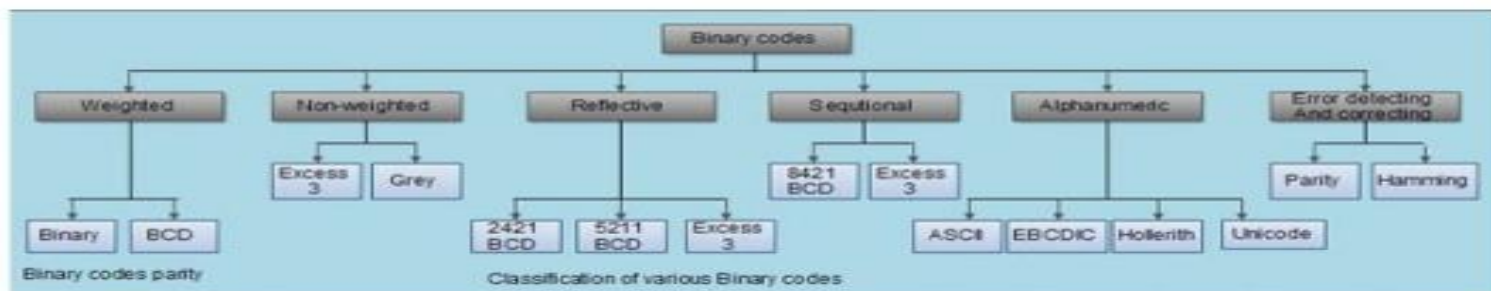
(b) $Y - X = 1000011 - 1010100$	
$Y =$	1000011
1's complement of $X =$	$\underline{+ 0101011}$
Sum =	1101110



There is no end carry, Therefore, the answer is $Y - X = -$ (1's complement of 1101110) = -0010001 .

1.7 Binary Codes

Digital data is represented, stored and transmitted as groups of binary digits also known as binary code.



- **Weighted codes:** In weighted codes, each digit is assigned a specific weight according to its position.
- **Non-weighted codes:** In non-weighted codes are not appositionally weighted.
- **Reflective codes:** A code is reflective when the code is self complementing. In other words, when the code for 9 is the complement the code for 0, 8 for 1, 7 for 2, 6 for 3 and 5 for 4.
- **Sequential codes:** In sequential codes, each succeeding 'code is one binary number greater than its preceding code.
- **Alphanumeric codes:** Codes used to represent numbers, alphabetic characters, symbols
- **Error defecting and correcting codes:** Codes which allow error defection and correction are called error detecting and' correcting codes.

- **BCD Code**

- A number with k decimal digits will require 4k bits in BCD.
- Decimal 396 is represented in BCD with 12bits as 0011 1001 0110, with each group of 4 bits representing one decimal digit.
- A decimal number in BCD is the same as its equivalent binary number only when the number is between 0 and 9.
- The binary combinations 1010 through 1111 are not used and have no meaning in BCD.

Table 1.4
Binary-Coded Decimal (BCD)

Decimal Symbol	BCD Digit
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Example: Consider decimal 185 and its corresponding value in BCD and binary:

$$(185)_{10} = (0001\ 1000\ 0101)_{BCD} = (10111001)_2$$

BCD addition

4	0100	4	0100	8	1000
+5	+0101	+8	+1000	+9	+1001
9	1001	12	1100	17	10001
			+0110		+0110
			10010		10111

Binary Codes

- Other Decimal Codes

Table 1.5
Four Different Binary Codes for the Decimal Digits

Decimal Digit	BCD 8421	2421	Excess-3	8, 4, -2, -1
0	0000	0000	0011	0000
1	0001	0001	0100	0111
2	0010	0010	0101	0110
3	0011	0011	0110	0101
4	0100	0100	0111	0100
5	0101	1011	1000	1011
6	0110	1100	1001	1010
7	0111	1101	1010	1001
8	1000	1110	1011	1000
9	1001	1111	1100	1111
Unused bit combinations	1010	0101	0000	0001
	1011	0110	0001	0010
	1100	0111	0010	0011
	1101	1000	1101	1100
	1110	1001	1110	1101
	1111	1010	1111	1110

Binary Codes

- **Gray Code**

- The advantage is that only bit in the code group changes in going from one number to the next.

- Error detection.
- Representation of analog data.
- Low power design.

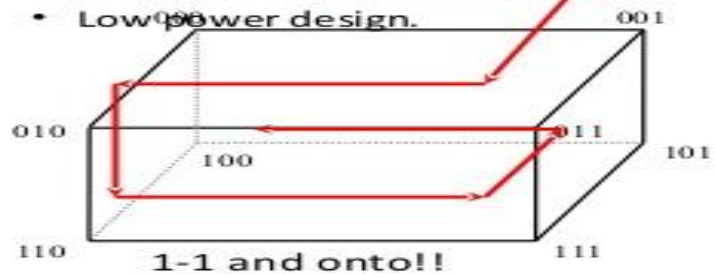


Table 1.6
Gray Code

Gray Code	Decimal Equivalent
0000	0
0001	1
0011	2
0010	3
0110	4
0111	5
0101	6
0100	7
1100	8
1101	9
1111	10
1110	11
1010	12
1011	13
1001	14
1000	15

- American Standard Code for Information Interchange (ASCII) Character Code

Table 1.7
American Standard Code for Information Interchange (ASCII)

$b_4b_3b_2b_1$	$b_7b_6b_5$							
	000	001	010	011	100	101	110	111
0000	NUL	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 Character Code**

Control characters

NUL	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

ASCII Character Codes and Properties

- American Standard Code for Information Interchange (Refer to Table 1.7)
- A popular code used to represent information sent as character-based data.
- It uses 7-bits to represent:
 - 94 Graphic printing characters.
 - 34 Non-printing characters.
- Some non-printing characters are used for text format (e.g. BS = Backspace, CR = carriage return).
- Other non-printing characters are used for record marking and flow control (e.g. STX and ETX start and end text areas).
- ASCII has some interesting properties:
 - Digits 0 to 9 span Hexadecimal values 30_{16} to 39_{16}
 - Upper case A-Z span 41_{16} to $5A_{16}$
 - Lower case a-z span 61_{16} to $7A_{16}$
 - Lower to upper case translation (and vice versa) occurs by flipping bit 6.

- **Error-Detecting Code**

- To detect errors in data communication and processing, an eighth bit is sometimes added to the ASCII character to indicate its parity.
- A **parity bit** is an extra bit included with a message to make the total number of 1's either even or odd.

- **Example:**

- Consider the following two characters and their even and odd parity:

	With even parity	With odd parity
ASCII A = 1000001	01000001	11000001
ASCII T = 1010100	11010100	01010100

• Error-Detecting Code

- **Redundancy** (e.g. extra information), in the form of extra bits, can be incorporated into binary code words to detect and correct errors.
- A simple form of redundancy is **parity**, an extra bit appended onto the code word to make the number of 1's odd or even. Parity can detect all single-bit errors and some multiple-bit errors.
- A code word has **even parity** if the number of 1's in the code word is even.
- A code word has **odd parity** if the number of 1's in the code word is odd.
- Example:

Message A: 10001001 **1** (even parity)

Message B: 10001001 **0** (odd parity)

Binary Logic

- **Definition of Binary Logic**

- Binary logic consists of binary variables and a set of logical operations.
- The variables are designated by letters of the alphabet, such as A, B, C, x, y, z , etc, with each variable having two and only two distinct possible values: 1 and 0,
- Three basic logical operations: AND, OR, and NOT.

1. **AND:** This operation is represented by a dot or by the absence of an operator. For example, $x \cdot y = z$ or $xy = z$ is read “ x AND y is equal to z ,” The logical operation AND is interpreted to mean that $z = 1$ if only $x = 1$ and $y = 1$; otherwise $z = 0$. (Remember that x, y , and z are binary variables and can be equal either to 1 or 0, and nothing else.)
2. **OR:** This operation is represented by a plus sign. For example, $x + y = z$ is read “ x OR y is equal to z ,” meaning that $z = 1$ if $x = 1$ or $y = 1$ or if both $x = 1$ and $y = 1$. If both $x = 0$ and $y = 0$, then $z = 0$.
3. **NOT:** This operation is represented by a prime (sometimes by an overbar). For example, $x' = z$ (or $\bar{x} = z$) is read “not x is equal to z ,” meaning that z is what x is not. In other words, if $x = 1$, then $z = 0$, but if $x = 0$, then $z = 1$. The NOT operation is also referred to as the complement operation, since it changes a 1 to 0 and a 0 to 1.

Binary Logic gates

- Truth Tables, Boolean Expressions, and Logic Gates

AND

x	y	z
0	0	0
0	1	0
1	0	0
1	1	1

$$z = x \cdot y = xy$$

OR

x	y	z
0	0	0
0	1	1
1	0	1
1	1	1

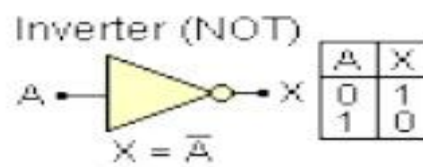
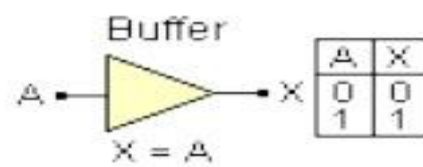
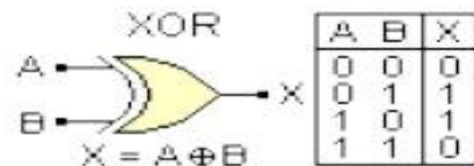
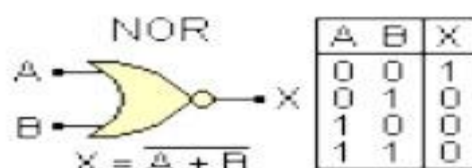
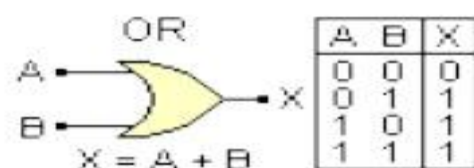
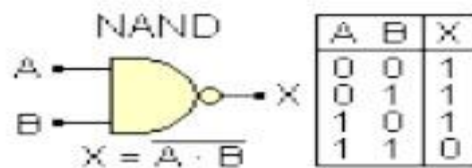
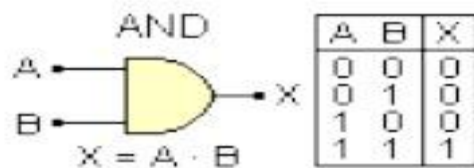
$$z = x + y$$

NOT

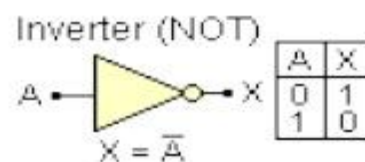
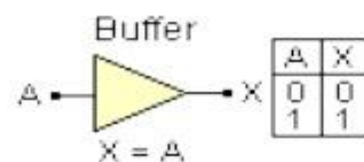
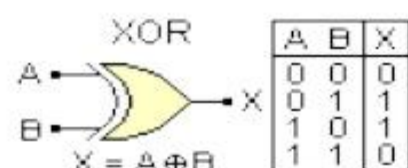
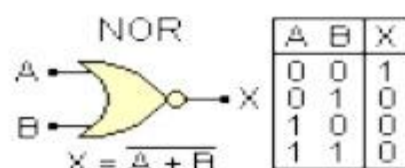
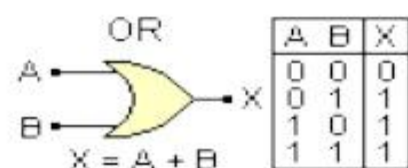
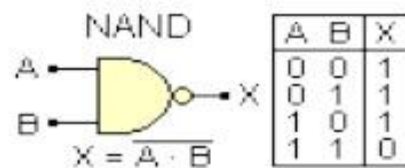
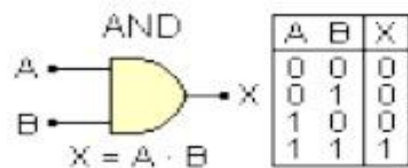
x	z
0	1
1	0

$$z = \bar{x} = x'$$





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

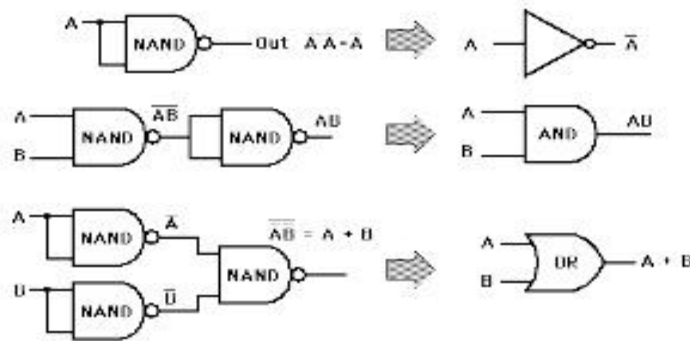


Logic Function	Boolean Notation
AND	$A \cdot B$
OR	$A + B$
NOT	\overline{A}
NAND	$\overline{A \cdot B}$
NOR	$\overline{A + B}$
EX-OR	$(A \cdot \overline{B}) + (\overline{A} \cdot B)$ or $A \oplus B$
EX-NOR	$\overline{(A \cdot \overline{B}) + (\overline{A} \cdot B)}$ or $\overline{A \oplus B}$

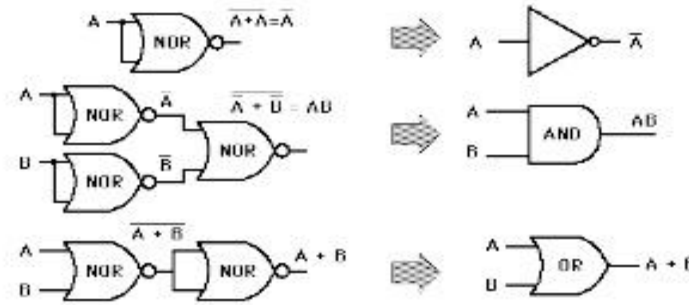
Universal Gate

- **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



Binary Logic

- Logic gates
 - Example of binary signals

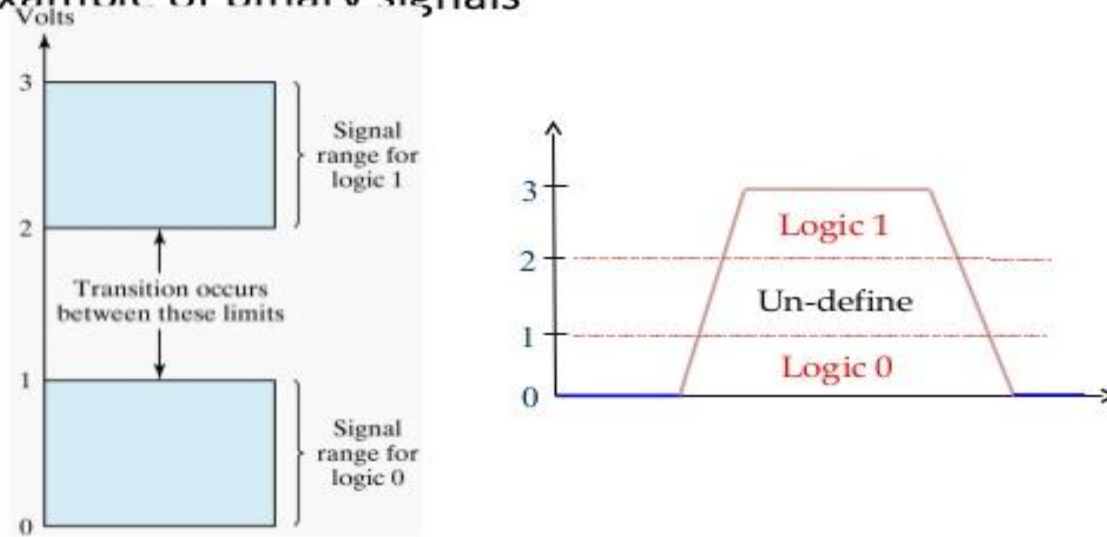


Figure 1.3 Example of binary signals

Binary Logic

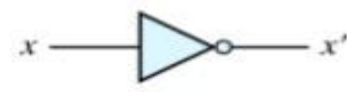
- Logic gates



(a) Two-input AND gate



(b) Two-input OR gate



(c) NOT gate or inverter

Fig. 1.4 Symbols for digital logic circuits

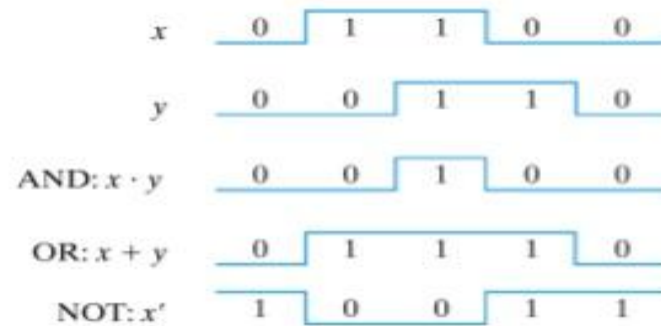
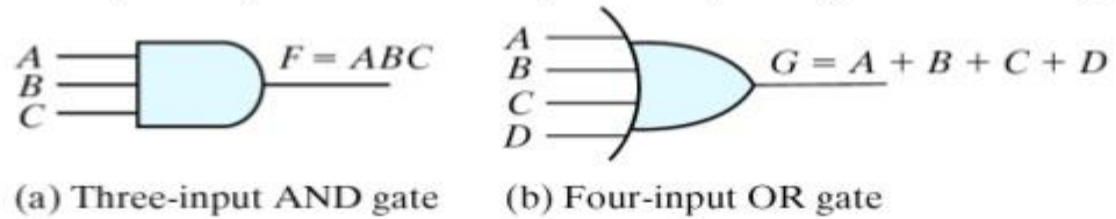


Fig. 1.5 Input-Output signals for gates

Binary Logic

- Logic gates
 - Graphic Symbols and Input-Output Signals for Logic



(a) Three-input AND gate (b) Four-input OR gate

Fig. 1.6 Gates with multiple inputs

Boolean Algebra

Boolean Algebra : George Boole(English mathematician), 1854

- Invented by George Boole in 1854
- An algebraic structure defined by a set $B = \{0, 1\}$, together with two binary operators (+ and \cdot) and a unary operator ($\bar{\quad}$)

"An Investigation of the Laws of Thought, on Which Are Founded the Mathematical Theories of Logic and Probabilities"

Boolean Algebra

{(1,0), Var, (NOT, AND, OR), Thms}

- Mathematical tool to expression and analyze **digital (logic) circuits**
- Claude Shannon, the first to apply Boole's work, 1938
 - "A Symbolic Analysis of Relay and Switching Circuits" at MIT
- This chapter covers Boolean algebra, Boolean expression and its evaluation and simplification, and VHDL program

Basic Functions and Basic Functions

Boolean functions : NOT, AND, OR,
 exclusive OR(XOR) : odd function exclusive NOR(XNOR) : even function(equivalence)

Boolean functions for (a) AND, (b) OR, (c) XOR, and (d) NOT

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

(a)

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

(b)

x	y	$x \oplus y$
0	0	0
0	1	1
1	0	1
1	1	0

(c)

x	x'
0	1
1	0

(d)

Basic functions

- AND $Z = X \cdot Y$ or $Z = XY$
 $Z = 1$ if and only if $X = 1$ and $Y = 1$, otherwise $Z = 0$
- OR $Z = X + Y$
 $Z = 1$ if $X = 1$ or if $Y = 1$, or both $X = 1$ and $Y = 1$. $Z = 0$ if and only if $X = 0$ and $Y = 0$
- NOT $Z = X'$ or
 $Z = 1$ if $X = 0$, $Z = 0$ if $X = 1$

Boolean functions for (a) NAND, (b) NOR, and (c) XNOR

x	y	NAND
0	0	1
0	1	1
1	0	1
1	1	0

(a)

x	y	NOR
0	0	1
0	1	0
1	0	0
1	1	0

(b)

x	y	XNOR
0	0	1
0	1	0
1	0	0
1	1	1

(c)

All possible binary boolean functions

x	y	0	\wedge	xy'	x	$x'y$	y	\oplus	\vee	NOR	XNOR	y'	$x + y'$	x'	$x' + y$	NAND	1
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Boolean Operations and Expressions

- Boolean Addition

- Logical OR operation

Ex 4-1) Determine the values of A, B, C, and D that make the sum term $A+B'+C+D'$



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

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

- Boolean Multiplication

- Logical AND operation

Ex 4-2) Determine the values of A, B, C, and D that make the product term $AB'CD'$



1

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

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

Basic Identities of Boolean Algebra

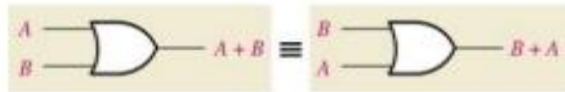
Basic Identities of Boolean Algebra

1.	$X+0 = X$	2.	$X \cdot 1 = X$	
3.	$X+1 = 1$	4.	$X \cdot 0 = 0$	<i>The relationship between a single variable X, its complement X', and the binary constants 0 and 1</i>
5.	$X+X = X$	6.	$X \cdot X = X$	
7.	$X+\bar{X} = 1$	8.	$X \cdot \bar{X} = 0$	
9.	$\overline{\bar{X}} = X$			
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.	$\overline{X+Y} = \bar{X} \cdot \bar{Y}$	17.	$\overline{X \cdot Y} = \bar{X} + \bar{Y}$	DeMorgan's

Laws of Boolean Algebra

- **Commutative Law:** the order of literals does not matter

$$A + B = B + A$$



$$A B = B A$$



- **Associative Law:** the grouping of literals does not matter

$$A + (B + C) = (A + B) + C (=A+B+C)$$

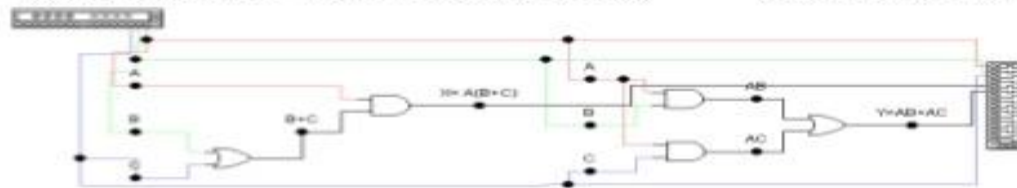


$$A(BC) = (AB)C (=ABC)$$



- **Distributive Law :** $A(B + C) = AB + AC$

$$(A+B)(C+D) = AC + AD + BC + BD$$



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**

- $F'(A,A', \cdot, +, 1,0) = F(A',A, +, \cdot, 0,1)$

- $(A \cdot B)' = A' + B'$ and $(A + B)' = A' \cdot B'$

- DeMorgan's theorem will help to simplify digital circuits using NORs and NANDs his theorem states

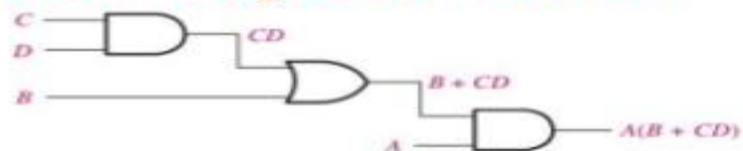


Inputs		Output	
X	Y	XY	X + Y
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

Inputs		Output	
X	Y	X + Y	XY
0	0	1	1
0	1	0	0
1	0	0	0
1	1	0	0

Boolean Analysis of Logic Circuits

- Constructing a Truth Table for a Logic Circuit



- Convert the expression into the min-terms containing all the input literals
- Get the numbers from the min-terms
- Putting '1's in the rows corresponding to the min-terms and '0's in the remains

$$\begin{aligned} \text{Ex) } A(B+CD) &= AB(C+C')(D+D') + A(B+B')CD = ABC(D+D') + ABC'(D+D') + ABCD + AB'CD \\ &= ABCD + ABCD' + ABC'D + ABC'D' + ABCD + AB'CD = ABCD + ABCD' + ABC'D + ABC'D' \\ &+ AB'CD = m_{11} + m_{12} + m_{13} + m_{14} + m_{15} = \Sigma(11, 12, 13, 14, 15) \end{aligned}$$

$$A(B+CD) = m_{11} + m_{12} + m_{13} + m_{14} + m_{15} = \Sigma(11, 12, 13, 14, 15)$$

Input				Output
A	B	C	D	A(B+CD)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Standard Forms of Boolean Expressions

- ❑ The Sum-of-Products(SOP) Form Ex) $AB+ABC$, $ABC+CDE+B'CD'$
- ❑ The Product-of-Sums(POS) Form Ex) $(A+B)(A+B+C)$, $(A+B+C)(C+D+E)(B'+C+D')$
- ❑ Principle of Duality : $SOP \Leftrightarrow POS$
- ❑ Domain of a Boolean Expression : The set of variables contained in the expression
Ex) $A'B+AB'C$: the domain is $\{A, B, C\}$

✓ Standard SOP Form (Canonical SOP Form)

- For all the missing variables, apply $(x+x')=1$ to the AND terms of the expression
- List all the min-terms in forms of the complete set of variables in ascending order

Ex : Convert the following expression into standard SOP form: $AB'C+A'B'+ABC'D$

Sol) domain= $\{A,B,C,D\}$, $AB'C(D'+D)+A'B'(C'+C)(D'+D)+ABC'D$

$=AB'CD'+AB'CD+A'B'C'D'+A'B'C'D+A'B'CD'+A'B'CD+ABC'D$

$=1010+1011+0000+0001+0010+0011+1101 = 0+1+2+3+10+11+13 =$

$\Sigma(0,1,2,3,10,11,13)$

Standard POS Form (Canonical POS Form)

- For all the missing variables, apply $(x'x)=0$ to the OR terms of the expression
- List all the max-terms in forms of the complete set of variables in ascending order

Ex : Convert the following expression into standard POS form: $(A+B'+C)(B'+C+D')(A+B'+C'+D)$

$$\begin{aligned} \text{Sol) domain} &= \{A, B, C, D\}, & (A+B'+C)(B'+C+D')(A+B'+C'+D) \\ &= (A+B'+C+D'D)(A'A+B'+C+D')(A+B'+C'+D) &= (A+B'+C+D') \\ & (A+B'+C+D)(A'+B'+C+D')(A+B'+C+D')(A+B'+C'+D) &= (0100)(0101) \\ & (0110)(1101) &= \Pi(4,5,6,13) \end{aligned}$$



THANK YOU

This content is taken from the text books and reference books prescribed in the syllabus.

This content is taken from the text books and reference books prescribed in the syllabus.

