Digital Codes

1- Binary Coded Decimal (BCD):

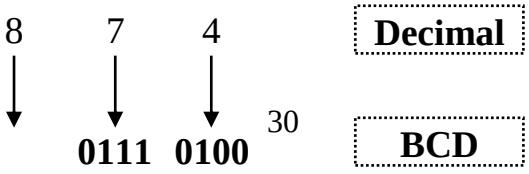
When numbers, letters, or words are represented by a special group of symbols, this is called encoding, and the group of symbols is called a code. Probably one of the familiar codes is the Morse code, where series of dots and dashes represent letters of the alphabet. We have seen that decimal numbers can be represented by an equivalent binary number. The group of 0s and 1s in the binary number can be thought of as a code representing the decimal number. When a decimal number is represented by its equivalent binary number, we call it (straight binary coding). We have seen that conversion between decimal and binary can become long and complicated for large numbers. For this reason, a means of encoding decimal numbers that combines some features of both the decimal and binary systems is used in certain situations.

The 8421 code is a type of binary coded decimal (BCD) code. Binary coded decimal means that each decimal digit, 0 though 9, is represented by a binary code of 4 bits. The designation 8421 indicates the binary weights of the four bits (2^3,2^2,2^1,2^0). The ease of conversion between 8421 code numbers and the familiar decimal numbers is the main advantage of this code. All you have to remember are the ten binary combinations that represent the ten decimal digits as shown in Table (1). The 8421 code is the predominant BCD code, and when referring to BCD, it always means the 8421 code unless otherwise stated.

Table (1)

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

To illustrate the BCD code, take a decimal number such as 874. Each digit is changed to its binary equivalent as follows:



It's also important to understand that a BCD number is not the same as a straight binary number. a straight binary code takes the complete decimal number and represents it in binary; the BCD code converts each decimal digit to binary individually . To illustrate, take the number 137 and compare its straight binary and BCD codes.

$137_{(10)} = 10001001_{(2)}$ (Binary)
$137_{(10)} = 000100110111$ (BCD)

The BCD code requires 12 bits while the straight binary code requires only 8 bits to represent 137. BCD is used in digital machines whenever decimal information is either applied as inputs or displayed as outputs. Digital voltmeter, frequency counters, and digital clocks, all use BCD because they display output information in decimal. BCD is not often used in modern high speed digital computers for the reason that the BCD code for a given decimal number requires more bits that the straight binary code and is therefore less efficient. This is important in digital computers because the number of places in memory where these bits can be stored is limited.

Example: convert each of the following decimal numbers to BCD:

(a) 35 (b) 98 (c) 170 (d) 2469

3	5	9	8	1	7	0	2	4	6	9
↓	↓	↓	↓	↓	↓	↓	↓	↓	↓	↓
0011	**0101**	**1001**	**1000**	**0001**	**0111**	**0000**	**0010**	**0100**	**0110**	**1001**

Example: convert each of the following BCD codes to decimal.

(a) 10000110 (b) 1001010001110000

10000110

8 6

1001010001110000

9 4 7 0

2- Gray Code:

The gray code is un-weighted and is not an arithmetic code; that is, there are no specific weights assigned to the bit positions. The important feature of the Gray code is that it exhibits only a single bit change from one code number to the next. Table (2) is a listing of the four bit gray code for decimal numbers 0 through 15. Notice the single bit change between successive gray code numbers. For instance, in going from decimal 3 to decimal 4, the gray code changes from 0010 to 0110, while the binary code changes from 0011 to 0100, a change of three bits. The only bit change is in the third bit from the right in the gray code; the other remain the same.

Table (2)

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

Binary-to-Gray Conversion:

Conversion between binary code and Gray code is sometimes useful. in the conversion process, the following rules apply:

- **The most significant bit (left-most) in the gray code is the same as the corresponding MSB in binary number.**

- **Going from left to right, add each adjacent pair of binary code bits to get the next gray code bit. Discard carry.**

Example: convert the binary number 10110 to Gray code.

Step 1: the left-most Gray code digit is the same as the left-most binary code bit.

```
10110      (Binary)
↓
1          (Gray)
```

Step 2: add the left-most binary code bit to the adjacent one:

```
1 + 0 110      (Binary)
    ↓
1   1          (Gray)
```

Step 3: add the next adjacent pair:

```
1  0 + 1 10      (Binary)
      ↓
1  1   1         (Gray)
```

Step 4: add the next adjacent pair and discard the carry:

```
1  0   1 + 1 0      (Binary)
         ↓
1  1   1   0        (Gray)
```

Step 5: add the last adjacent pair:

```
1  0  1   1 + 0      (Binary)
              ↓
1  1  1   0   1      (Gray)
```

Hence the Gray Code is 11101

Gray-to-Binary Conversion:

To convert from Gray code to binary, a similar method is used, but there are some differences. The following rules apply:

- **The most significant bit (left-most) in the binary code is the same as the corresponding bit in the Gray code.**
- **Add each binary code bit generated to the gray code bit in the next adjacent positions. Discard carry.**

Example: convert the Gray code number 11011 to binary.

Step 1: the left-most bits are the same.

```
11011          (Gray)
↓
1              (Binary)
```

Step 2: add the last binary code bit just generated to the gray code bit in the next position. Discard the carry.

```
1  1 011       (Gray)
  + ↓
1  0           (Binary)
```

Step 3: add the last binary code bit generated to the next Gray code bit.

```
1  1  0 11     (Gray)
     + ↓
1  0  0        (Binary)
```

Step 4: add the last binary code bit generated to the next Gray code bit.

```
1  1  0  1 1   (Gray)
        + ↓
1  0  0  1     (Binary)
```

Step 5: add the last binary code bit generated to the next Gray code bit. discard carry.

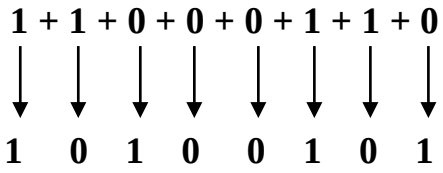
```
1  1  0  1  1  (Gray)
           + ↓
1  0  0  1  0  (Binary)
```

Hence the final binary number is 10010

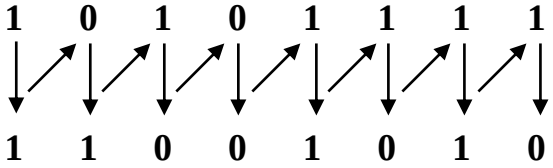
Example: (a) Convert the binary number 11000110 to Gray-code.

(b) Convert the Gray-code 10101111 to binary.

(a) Binary to Gray code:-

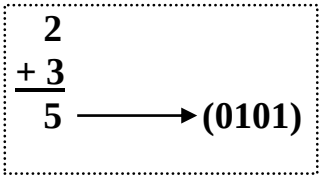


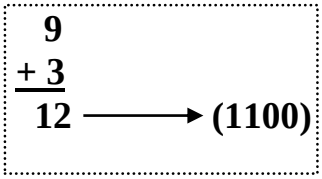
(b) Gray code to Binary:-



3- Excess-3 Code:

This is a digital code related to BCD that is derived by adding 3 to each decimal digit and then converting the result of that addition to 4-bit binary. This code is un-weighted. For instance, the excess-3 code for decimal 2 and 9 are:





The excess-3 code for each decimal digit is found by the same procedure. the entire code is shown in Table (3).

Table (3)

Decimal	BCD	Excess-3
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Example: convert each of the following decimal number to Excess-3 code:

(a) 13 (b) 430

First, add 3 to each digit in the decimal number, and then convert each resulting 4-bit sum to its equivalent binary code.

1 3
+ 3 + 3
4 6
↓ ↓
0100 0110 (Excess-3)

4 3 0
+ 3 + 3 +3
7 6 3
↓ ↓ ↓
0111 0110 0011 (Excess-3)

4- Alphanumeric Code:

In order to be very useful, a computer must be capable of handling non-numeric information. In other words, a computer must be able to recognize codes that represent numbers, letters, and special characters. These codes are classified as alphanumeric codes. The most common alphanumeric code, known as the American Standard Code for Information Interchange (ASCII), is used by most minicomputer and microcomputer manufacturers.

The ASCII is a seven-bit code in which the decimal digits are represented by the 8421 BCD code preceded by 011. The letters of the alphabet and other symbols and instructions are represented by other code combinations, shown in Table (4). For instance, the letter **A** is represented by 1000001 (41_{16}), the **comma** by 0101100 ($2C_{16}$) and the **ETX** (end of text) by 0000011 (03_{16}).

Table (4)

LSBs	**MSBs**							
	000 (0)	001 (1)	010 (2)	011 (3)	100 (4)	101 (5)	110 (6)	111 (7)
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	EXC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	|

1101	CR	GS	-	=	M	]	m	}
1110	SO	RS	.	>	N	↑	n	~
1111	SI	US	/	?	O	_	o	DEL

Example: determine the codes that are entered from the computer's keyboard when the following basic program statement is typed in. also express each in hexadecimal notation.

20 PRINT "A=";X

Character	*ASCII*	*Hexadecimal*
2	0110010	32
0	0110000	30
Space	0100000	20
P	1010000	50
R	1010010	52
I	1001001	49
N	1001110	4E
T	1010100	54
Space	0100000	20
"	0100010	22
A	1000001	41
=	0111101	3D
"	0100010	22
;	0111011	3B
X	1011000	58

Review Questions:

1. How many bits are required to represents the decimal numbers in the range from 0 to 999 using straight binary code? Using BCD code?
2. What is the binary weight of each 1 in the following BCD numbers?
   (a) 0010 (b) 1000 (c) 0001 (d) 0100
3. Convert the following binary numbers to Gray codes?
   (a) 1100 (b) 1010 (c) 11010
4. Convert the following Gray codes to binary?
   (a) 1000 (b) 1010 (c) 11101
5. Convert the following decimal numbers to Excess-3 code?
   (a) 3 (b) 87 (c) 349
6. Convert decimal 928 to Excess-3?

Boolean algebra:

Boolean algebra is the mathematics of digital systems. It is important that you understand is principles thoroughly because a basic knowledge of Boolean algebra is indispensable to the study and analysis of logic circuits. The Boolean expressions for the previously studied gates are as follows:

NOT Gate: if input is *A*, the output is $\overline{A}$

AND Gate: if inputs are *A* and *B*, the output is *A.B*

OR Gate : if inputs are *A* and *B*, the output is $A+B$

NAND Gate: if inputs are *A* and *B*, the output is $\overline{A.B}$

A	B	AB	$A+AB$
0	0	0	0

NOR Gate : if inputs are *A* and *B*, the output is $\overline{A+B}$

XOR Gate : if inputs are *A* and *B*, the output is $A\overline{B}+\overline{A}B=A\oplus B$

XNOR Gate: if inputs are *A* and *B*, the output is $\overline{A}\overline{B}+AB=A\odot B$

Laws of Boolean algebra:

1. *Commutative law:* $A+B=B+A$, $AB=BA$
2. *Associative law:* $A+(B+C)=(A+B)+C$, $A(BC)=(AB)C$
3. *Distributive law:* $A(B+C)=AB+AC$

Rules for Boolean algebra:

Table (1) lists 12 basic rules that are useful in manipulating and simplifying Boolean expressions.

Table (1)

1.	$A+0=A$	***7.***	$A.A=A$
2.	$A+1=1$	***8.***	$A.\overline{A}=0$
3.	$A.0=0$	***9.***	$\overline{\overline{A}}=A$
4.	$A.1=A$	***10.***	$A+AB=A$
5.	$A+A=A$	***11.***	$A+\overline{A}B=A+B$
6.	$A+\overline{A}=1$	***12.***	$(A+B)(A+C)=A+BC$

Example: Prove that $A+AB=A$.

$$A + AB = A(1 + B)$$
$$= A.1 = A$$

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

Example: Prove that $A + \overline{A}B = A + B$.

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

A	B	$\overline{A}B$	$A + \overline{A}B$	$A+B$
0	0	0	0	0
0	1	1	1	1
1	0	0	1	1
1	1	0	1	1

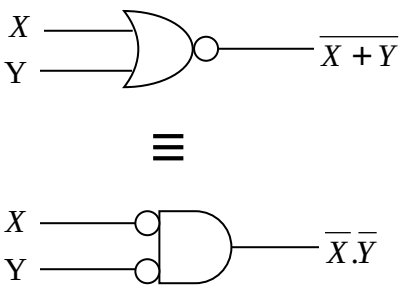
Equal

DeMorgan's Theorem:

Two of the most important theorems of Boolean algebra were contributed by a great mathematician named DeMorgan. DeMorgan's theorems are extremely useful in simplifying expressions in which a product of sum of variables is inverted. The two theorems are:

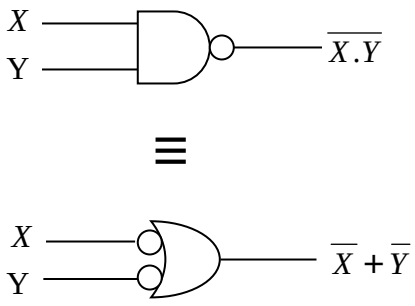
1- $\overline{X + Y} = \overline{X}.\overline{Y}$

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**



2- $\overline{X.Y} = \overline{X} + \overline{Y}$

X	Y	$\overline{X.Y}$	$\overline{X}+\overline{Y}$
0	0	**1**	**1**
0	1	**1**	**1**
1	0	**1**	**1**
1	1	**0**	**0**



Example: Apply DeMorgan's theorem to the expressions: $\overline{WXYZ}$ and $\overline{W+X+Y+Z}$.

$$\overline{WXYZ}=\overline{W}+\overline{X}+\overline{Y}+\overline{Z}$$

$$\overline{W+X+Y+Z}=\overline{W}\,\overline{X}\,\overline{Y}\,\overline{Z}$$

Example: Apply DeMorgan's theorem to each of the following expressions:

(a) $\overline{(A+B+C)D}$ **(b)** $\overline{A\overline{B}+\overline{C}D+EF}$

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

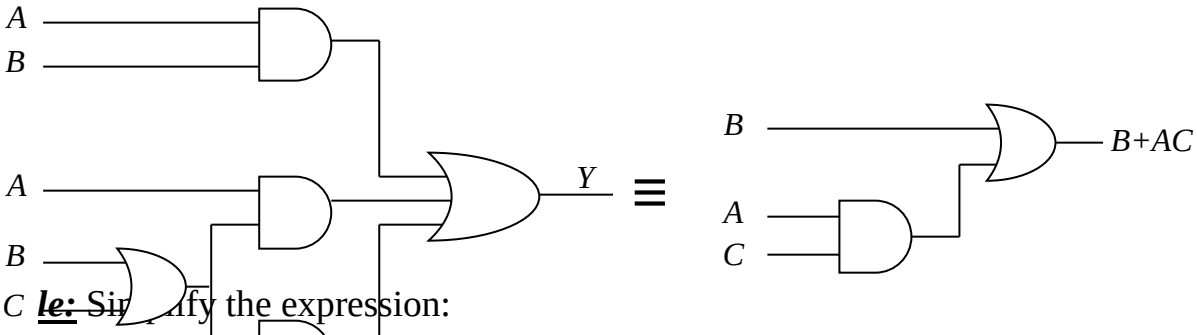
$$\overline{A\overline{B}+\overline{C}D+EF}=\overline{(A\overline{B})}\,\overline{(\overline{C}D)}\,\overline{(EF)}$$
$$=(\overline{A}+B)(C+\overline{D})(\overline{E}+\overline{F})$$

Simplification using Boolean algebra:

Many times in the application of Boolean algebra, we have to reduce a particular expression to its simplest form or change its form to a more convenient one to implement the expression most efficiently. The purpose of simplifying Boolean expression is to use the fewest gates possible to implement a given expression.

Example: Simplify the following expression: $Y=AB+A(B+C)+B(B+C)$.

$$AB+A(B+C)+B(B+C)=AB+AB+AC+BB+BC$$
$$=AB+AB+AC+B+BC$$
$$=AB+AC+B+BC$$
$$=AB+AC+B=B+AC$$



Example: Simplify the expression:

$$\overline{A}BC + B\,\overline{A}\overline{B}\overline{C} + \overline{A}\overline{B}\overline{C} + \overline{A}BC + ABC$$

$$BC(\overline{A} + A) + A\overline{B}\overline{C} + \overline{A}\overline{B}\overline{C} + A\overline{B}C$$

$$BC.1 + A\overline{B}(C + \overline{C}) + \overline{A}\overline{B}\overline{C}$$

$$BC + A\overline{B}.1 + \overline{A}\overline{B}\overline{C} = BC + \overline{B}(A + \overline{A}\overline{C}) = BC + \overline{B}(A + \overline{C}) = BC + A\overline{B} + \overline{B}\overline{C}$$

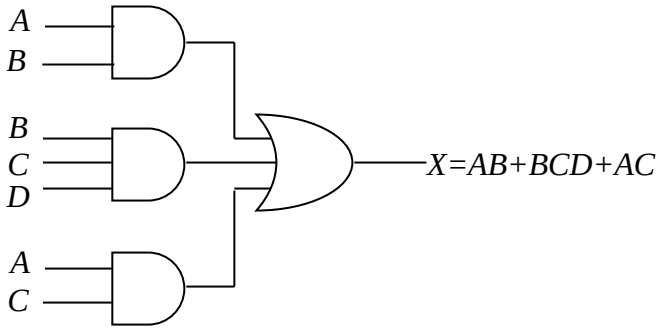
The Sum-of-Product (SOP) form:

When two or more product terms are summed by Boolean addition, the resulting expression is a sum of product (SOP). Some examples are:

$$AB + ABC$$

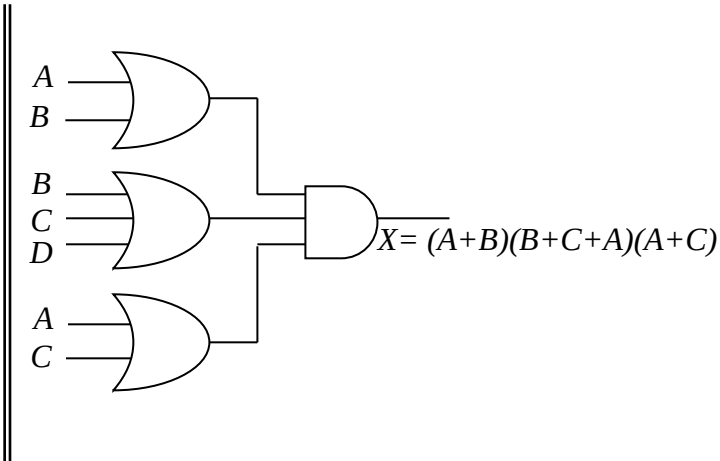
$$ABC + CDE + \overline{B}C\overline{D}$$

$$\overline{A}B + \overline{A}B\overline{C} + AC$$



The Product-of-Sum (POS) form:

When two or more sum terms are multiplied, the resulting expression is a product of sum (POS). Some examples are:



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

$(\overline{A}+\overline{B}+\overline{C})(C+\overline{D}+E)(\overline{B}+C+D)$

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

Example: Convert each of the following expression to **general** SOP form.

(a) $AB+B(CD+EF)$ (b) $\overline{(\overline{A+B})+C}$

(a): $AB+B(CD+EF)=AB+BCD+BEF$

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

Example: Convert the following expression into **standard** SOP form.

$A\overline{B}C+\overline{A}\overline{B}+AB\overline{C}D$

$*\ A\overline{B}C=A\overline{B}C(D+\overline{D})=A\overline{B}CD+A\overline{B}C\overline{D}$

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

$=\overline{A}\overline{B}CD+\overline{A}\overline{B}C\overline{D}+\overline{A}\overline{B}\overline{C}D+\overline{A}\overline{B}\overline{C}\overline{D}$

$hence: A\overline{B}C+\overline{A}\overline{B}+AB\overline{C}D=A\overline{B}CD+A\overline{B}C\overline{D}+\overline{A}\overline{B}CD+\overline{A}\overline{B}C\overline{D}+\overline{A}\overline{B}\overline{C}\overline{D}+AB\overline{C}D$

Example: Convert the following expression into **standard** POS form.

$(A+\overline{B}+C)(\overline{B}+C+\overline{D})(A+\overline{B}+\overline{C}+D)$

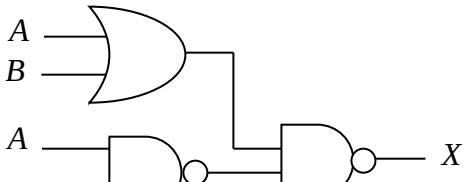
$*(A+\overline{B}+C)=A+\overline{B}+C+D\overline{D}=(A+\overline{B}+C+D)(A+\overline{B}+C+\overline{D})$

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

$hence: (A+\overline{B}+C)(\overline{B}+C+\overline{D})(A+\overline{B}+\overline{C}+D)=(A+\overline{B}+C+D)(A+\overline{B}+C+\overline{D})$

$(A+\overline{B}+C+\overline{D})(\overline{A}+\overline{B}+C+\overline{D})(A+\overline{B}+\overline{C}+D)$

Example: Determine if the circuit gives XOR, XNOR or neither in the output.

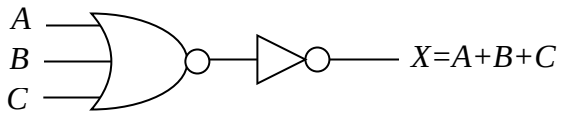


$$X = \overline{(A+B).(\overline{A.B})}$$
$$= (\overline{A+B}) + (\overline{\overline{A.B}})$$
$$= \overline{A}\overline{B} + AB \equiv \ XNOR\ \ Gate$$

Example: Determine the Boolean expression for a three-input NOR gate followed by an inverter.

The expression at the NOR output is $(\overline{A+B+C})$, which is then fed through an inverter to produce:

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



Example: Simplify the expression $Y = A\overline{B}D + A\overline{B}\overline{D}$.

$$Y = A\overline{B}(D+\overline{D}) = A\overline{B}.1 = A\overline{B}$$

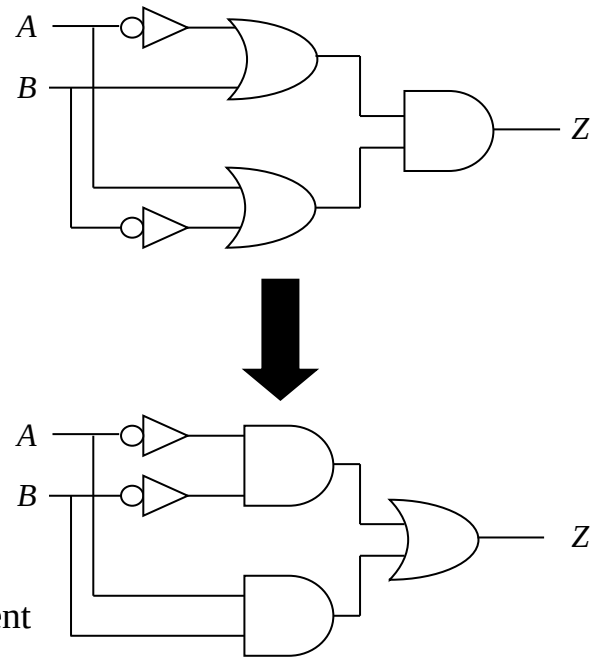
Example: Simplify the expression $X = ACD + \overline{A}BCD$.

$$X = CD(A+\overline{A}B) = CD(A+B) = ACD + BCD$$

Example: Simplify the logic circuit shown in the Figure below.

The expression for output Z is:

$$Z = (\overline{A}+B)(A+\overline{B})$$
$$= \overline{A}A + \overline{A}\overline{B} + BA + B\overline{B}$$
$$= \overline{A}\overline{B} + AB$$



If we compare the resulting circuit with the original one, we see that both circuits contain the same number of gates and connections. In this case, the simplification process produced an equivalent, but not simpler circuit. Also one could notice that the resulting output is equivalent

to exclusive NOR gate.

Example: Develop truth table for the expression $\overline{A}\overline{B}C + A\overline{B}\overline{C} + ABC$.

There are three variables in the domain, so there are eight possible binary values of the variables as listed in the left columns of Table (2).

Table (2)

A	*B*	*C*	*X*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Review Questions:

1. Apply DeMorgan's theorem to the following expressions:

(a) $\overline{ABC} + (\overline{\overline{D} + E})$ (b) $\overline{A + B + C} + \overline{\overline{D}E}$

(c) $(\overline{\overline{ABC}})(\overline{\overline{EFG}}) + (\overline{\overline{HIJ}})(\overline{\overline{KLM}})$

(d) $(\overline{A + \overline{\overline{BC}} + CD}) + \overline{\overline{BC}}$

2. Simplify the following expressions:

(a) $A + AB + A\overline{B}C$ $\quad Ans: A$

(b) $(\overline{A} + B)C + ABC$ $\quad Ans: C(\overline{A} + B)$

(c) $A\overline{B}C(BD + CDE) + A\overline{C}$ $\quad Ans: A(\overline{C} + \overline{B}DE)$

(d) $(\overline{A} + B)(A + B)$ $\quad Ans: B$

(e) $A\overline{C} + AB\overline{C}$ $\quad Ans: A\overline{C}$

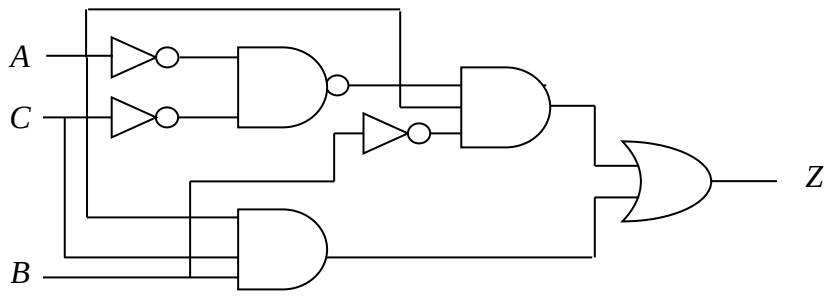
(f) $\overline{A}\overline{B}C\overline{D} + \overline{A}\overline{B}\overline{C}\overline{D}$ $\quad Ans: \overline{A}\overline{B}\overline{D}$

(g) $(\overline{A} + B)(A + B + D)\overline{D}$ $\quad Ans: B\overline{D}$

(h) $\overline{A}C(\overline{\overline{A}BD}) + \overline{A}B\overline{C}\overline{D} + A\overline{B}C$ $\quad Ans: \overline{B}C + \overline{A}\overline{D}(C + B)$

(i) $ABC + AB\overline{C} + A\overline{B}C$ $\quad Ans: A(B + C)$

3. Simplify the logic circuit shown in figure below: (ANS: $A(\overline{B} + C)$)



4. Draw the logic circuit represented by each expression.

(a) $AB + \overline{AB} + \overline{A}BC$ $\quad$ (b) $A + B[C + D(B + \overline{C})]$

5. Convert the following expressions to sum-of-product form:

(a) $(A + B)(C + \overline{B})$ $\quad$ (b) $(A + \overline{B}C)C$

6. Develop a truth table for each of the SOP expressions:

(a) $\overline{A}B + AB\overline{C} + \overline{AC} + A\overline{B}C$ $\quad$ (b) $\overline{X} + Y\overline{Z} + WZ + X\overline{Y}Z$

7. Prove that $(A+B)(A+C)=A+BC$.

The Karnaugh Map (K-Map):

The Karnaugh map provides a systematic method for simplifying Boolean expression and, if properly used, will produce the simplest SOP or POS expression possible. The K-map, like a truth table, is a means for showing the relationship between logic inputs and the desired output. The K-map is an array of **Cells** in which each cell represents a binary value of the input variables. The cells are arranged in a way so that simplification of a given expression is simply a matter of properly grouping the cells. The number of cells in a Karnaugh map is equal to the total number of possible input variable combinations as is the number of rows in a truth table. For three variables, the number of cells is 2^3=8. For four variables, the number of cells is 2^4=16.

Three examples of K-maps for two, three, and four variables, together with the corresponding truth tables are shown in Figrue below:

A	B	X	
0	0	1	$\overline{A}\,\overline{B}$
0	1	0	
1	0	0	
1	1	1	AB

$$X = \overline{A}\,\overline{B} + AB$$

	$\overline{B}$	B
$\overline{A}$	1	0
A	0	1

A	B	C	X	
0	0	0	1	$\overline{A}\,\overline{B}\,\overline{C}$
				$\overline{A}\,\overline{B}C$
				$\overline{A}B\overline{C}$