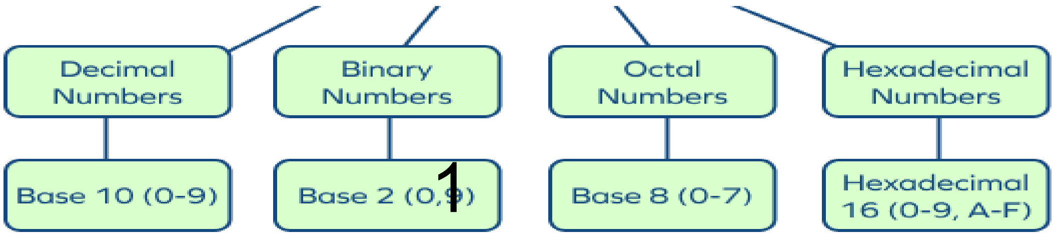
Lec._1

Number systems /أنظمة الأرقام



1. **The Binary Number System**: has the base 2 and uses only 2 symbols or digits (0, 1) to form other numbers. نظام الأرقام الثنائية
2. **The Octal Number System** has the base-8 and uses only 8 symbols or digits (0, 1, 2, 3, 4, 5, 6, 7) used to form other numbers.نظام الأرقام الثماني
3. **The Decimal Number System** : The most commonly used number, which has base 10 and uses only 10 symbols or digits (0, 1, 2, 3, 4, 5, 6, 7,8, 9) to form other numbers.نظام الأرقام العشري
4. **The Hexadecimal Number System**: has base 16 and uses only 16 symbols or digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F) to form other numbers. نظام الأرقام الست عشري

The main advantage of using the **Binary and octal** number system are that it uses fewer digits than the decimal and hexadecimal number system. So, it has fewer calculations and thereby less calculation errors.

الميزة الرئيسية لاستخدام نظام الأرقام الثنائي والثماني هي أنه يستخدم أرقامًا أقل من نظام الأرقام العشري والست عشري. لذا، فهي تحتوي على عدد أقل من الحسابات وبالتالي أخطاء حسابية أقل.

Table to compare number systems

10^3 10^2 10^1 10^0	8^3 8^2 8^1 8^0	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	16^2 16^1 16^0
Decimal	octal	Binary										hexadecimal
0000	0000	0	0	0	0	0	0	0	0	0	0	000
0001	0001	0	0	0	0	0	0	0	0	0	1	001
0002	0002	0	0	0	0	0	0	0	0	1	0	002
0003	0003	0	0	0	0	0	0	0	0	1	1	003
0004	0004	0	0	0	0	0	0	0	1	0	0	004
0005	0005	0	0	0	0	0	0	0	1	0	1	005
0006	0006	0	0	0	0	0	0	0	1	1	0	006
0007	0007	0	0	0	0	0	0	0	1	1	1	007
0008	0010	0	0	0	0	0	0	1	0	0	0	008
0009	0011	0	0	0	0	0	0	1	0	0	1	009
0010	0012	0	0	0	0	0	0	1	0	1	0	00A
0011	0013	0	0	0	0	0	0	1	0	1	1	00B
0012	0014	0	0	0	0	0	0	1	1	0	0	00C
0013	0015	0	0	0	0	0	0	1	1	0	1	00D
0014	0016	0	0	0	0	0	0	1	1	1	0	00E
0015	0017	0	0	0	0	0	0	1	1	1	1	00F
0016	0020	0	0	0	0	0	1	0	0	0	0	010
0017	0021	0	0	0	0	0	1	0	0	0	1	011
0018	0022	0	0	0	0	0	1	0	0	1	0	012
0019	0023	0	0	0	0	0	1	0	0	1	1	013
0020	0024	0	0	0	0	0	1	0	1	0	0	014
0021	0025	0	0	0	0	0	1	0	1	0	1	015
0022	0026	0	0	0	0	0	1	0	1	1	0	016
0023	0027	0	0	0	0	0	1	0	1	1	1	017
0024	0030	0	0	0	0	0	1	1	0	0	0	018
0025	0031	0	0	0	0	0	1	1	0	0	1	019
0026	0032	0	0	0	0	0	1	1	0	1	0	01A
0027	0033	0	0	0	0	0	1	1	0	1	1	01B
0028	0034	0	0	0	0	0	1	1	1	0	0	01C
0029	0035	0	0	0	0	0	1	1	1	0	1	01D
0030	0036	0	0	0	0	0	1	1	1	1	0	01E
0031	0037	0	0	0	0	0	1	1	1	1	1	01F
0032	0040	0	0	0	0	1	0	0	0	0	0	020
:	:	:	:	:	:	:	:	:	:	:	:	:
0266	0412	0	1	0	0	0	0	1	0	1	0	10A
0267	0413	0	1	0	0	0	0	1	0	1	1	10B
:	:	:	:	:	:	:	:	:	:	:	:	:
1022	1776	1	1	1	1	1	1	1	1	1	0	3FE
1023	1777	1	1	1	1	1	1	1	1	1	1	3FF

Frequently Asked Questions on Binary Number System ….

Q1) What is a binary number system?

A number system where a number is represented by using only two digits (0 and 1) with a base 2 is called a binary number system. For example, 1001_2 is a binary number.

Q2) What is a bit?

A bit is a single digit in the binary number. For example, 101 is three-bit binary numbers, where 1, 0 and 1 are the bits.

Q3) How to convert a decimal number into a binary number? Give an example.

To convert a decimal number into its equivalent binary number, we divide the decimal number by 2 each time, till we get 0 as a dividend. Let us take an example to convert 13_{10} into a binary number.

13	÷	2:	6	and	remainder	1
6	÷	2:	3	and	remainder	0
3	÷	2:	1	and	remainder	1
1	÷	2:	0	and	remainder	1

Now we take the bits from the last remainder to first remainder, i.e.(MSB to LSB). Hence,$13_{10} = 1101_2$

Q4) What is the use of binary numbers?

Binary numbers are commonly used in computer architecture. Since the computer understands only the language of two digits 0's and 1's, therefore the programming is done using a binary number system.

Q5) What is the value of 163 in binary?

The value of 163 in binary is 10100011.

Q6) How is 200 represented in binary?

200 is the decimal number. The binary form of 200 is 11001000_2.

Lec._2

Binary Arithmetic Operations

Like we perform the arithmetic operations in numerals, in the same way, we can perform addition, subtraction, multiplication and division operations on Binary numbers. Let us learn them one by one.

Binary Addition

Adding two binary numbers will give us a binary number itself. It is the simplest method. Addition of two single-digit binary number is given in the table below.

Binary Numbers		Addition
0	0	0
0	1	1
1	0	1
1	1	0; Carry →1

Let us take an example of two binary numbers and add them.

For example: Add 1101_2 and 1001_2.
Solution:

$$\begin{array}{r} 1101 \\ +1001 \\ \hline 10110 \\ \hline \end{array}$$

Binary Subtraction

Subtracting two binary numbers will give us a binary number itself. It is also a straightforward method. Subtraction of two single-digit binary number is given in the table below.

Binary Numbers		**Subtraction**
0	0	0
0	1	1; Borrow 1
1	0	1
1	1	0

Let us take an **example** of two binary numbers and subtract them**:** Subtract $1101_{2,}$ and 1010_2.

Solution: $1101_2 – 1010_2 = 0011_2$

Binary Multiplication

The multiplication process is the same for the binary numbers as it is for numerals. Let us understand it with example.

Example: Multiply 1101_2 and 1010_2.

```
       1101
     × 1010
  -----------
       0000
      1101
     0000
    1101
  -----------
   10000010
  -----------
```

Binary Division

The binary division is similar to the decimal number division method. We will learn with an example here.

Example: Divide 1010_2 by 10_2

```
10)1010(101
   10
  ---------
     010
      10
  ---------
       0
  ---------
```

Uses of Binary Number System

Binary numbers are commonly used in computer applications. All the coding and languages in computers such as C, C++, Java, etc. use binary digits 0 and 1 to write a program or encode any digital data. The computer understands only the coded language. Therefore these 2-digit number system is used to represent a set of data or information in discrete bits of information.

Problems and Solutions

Let us practice some of the problems for better understanding:

Question 1: **What is binary number 1.1 in decimal?**

Solution:

Step 1: 1 on the left-hand side is on the one's position, so it's 1.

Step 2: The one on the right-hand side is in halves, so it's

1 × ½

Step 3: so, 1.1 = 1.5 in decimal.

Question 2: Write 10.11_2 in Decimal?

Solution:

$10.11 = 1 \times (2)^1 + 0\ (2)^0 + 1\ (½)^1 + 1(½)^2$

= 2 + 0 + ½ + ½

= 2.75

So, 10.11 is 2.75 in Decimal.

Next >>> let us understand how the interconversions between these systems are done. القادم >>> دعونا نفهم كيف تتم التحويلات البينية بين هذه الأنظمة

Lec._3

Convert Between Number Systems / التحويل بين أنظمة الأرقام

For example: $(4)_{10}$ in binary is $(100)_2$.

Here, 4 is represented in the decimal number system, where we can represent the number using the digits from 0-9. However, in a binary number system, we use only two digits, such as 0 and 1.

Now, let's discuss how to convert 4 in binary number system.دعونا نناقش كيفية تحويل رقم 4 الى النظام الثنائي

The following steps help to convert 4 in binary divide the number 4 by 2. Use the integer quotient obtained in this step as the dividend for the next step.

Continue this step, until the quotient becomes 0.

Dividend	Remainder	Rank
4/2 = 2	0	2^0 Least Significant Bit (LSB)
2/2 = 1	0	2^1
1/2 = 0	1	2^2 Most Significant Bit (MSB)
$(100)_2$. = $(4)_{10}$ لذلك		

For example: (61)$_{10}$ How to convert in binary number system **(?)$_2$**.

Dividend	Remainder	Rank	To check Convert binary to decimal
61/2 = 30	1	2^0 (LSB)	1X1
30/2 = 15	0	2^1	0X2
15/2 = 7	1	2^2	1X4
7/2 = 3	1	2^3	1X8
3/2 = 1	1	2^4	1X16
1/2 = 0	1	2^5 (MSB)	1X32
(111101)$_2$. = (61)$_{10}$ لذلك			Sum all =61

Octal and hexadecimal Number System – Conversions, Examples

Octal Every digit has to be converted to a 3-bit binary number. Thus, we get the binary equivalent of the number.

hexadecimal Every digit has to be converted to a 4-bit binary number. Thus, we get the binary equivalent of the number.

Let's understand this with the help of an example.

Example: Convert (16)$_8$ **into a binary number.** Then to **hexadecimal**

Solution: (16)$_8$ is an octal number.

With the above conversion, we can write

$1_8=001_2$ and $6_8=110_2$

Thus, $(16)_8=(001110)_2$

So $(16)_8=(0E)_{16}$

Example: Convert $(16)_{16}$ **into a binary number.** Then to octal

Solution: $(16)_{16}$ is an hexadecimal number.

With the above conversion, we can write

$1_{16}=0001_2$ and $6_{16}=0110_2$

Thus, $(16)_{16}=(00010110)_2$

So $(16)_{16}=(26)_8$

Convert each hex digit to 4 binary digits and then convert each 3 binary digits to octal digits

قم بتحويل كل رقم سداسي عشري إلى 4 أرقام ثنائية ثم قم بتحويل كل 3 أرقام ثنائية إلى أرقام ثماني

Conversion of Octal to Decimal Numbers

Converting octal to decimal is a simple process!

A number in the octal system is expanded with the base of eight, where each digit is multiplied with the power of 8, based on its position.

After the octal is converted to decimal, it has a base of 10.

Example: Convert $(321)_8$ **to decimal form.**

$(321)_8=(3\times8^2)+(2\times8^1)+(1\times8^0)$

$=(3\times64)+(2\times8)+(1\times1)$

$=192+16+1 =209_{10}$

Thus, $(321)_8=(209)_{10}$

Conversion of Decimal to Octal Number

In this conversion, the decimal number is divided by 8 each time a reminder is obtained from the previous digit. Let us understand this conversion with the help of an example.

Example: Convert 416_{10} **to octal.**

Divide 416 by the octal base number, 8.

Division by 8	Quotient	Remainder
416÷8	52	0
52÷8	6	4
6÷8	0	6

We stop when the quotient value becomes 0. By writing the remainders in reverse order, we get the equivalent octal number. Thus, the octal representation of 416_{10} is 640_8.

Conversion of Octal to Hexadecimal Numbers

The simplest way is to first convert the octal number to a decimal, and then the decimal to a hexadecimal number.

Let us understand octal to hexadecimal conversion with the help of an example.

Example: Convert $(70)_8$ **to hexadecimal.**

Step 1: Octal to Decimal

$(70)_8=(7\times8^1)+(0\times8^0)$

$(70)_8=(56)_{10}$

Step 2: Decimal to hexadecimal

Now, we need to convert $(56)_{10}$ to a hexadecimal number.

Divide the number 56 by 16 until the number in the quotient value becomes 0.

Write remainders in reverse order.

Therefore, $(70)_8=(38)_{16}$

Octal Multiplication Table

You can multiply octal numbers in two ways. One way is to convert octal to decimal: perform the decimal multiplication to get the product and convert the result back to octal. The second way is simply using the octal multiplication table. Example:

×	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	20	24	30	34
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

Facts about the Octal Number System

- In 1801, James Anderson criticized the French for basing the metric system on decimal arithmetic. He suggested the base 8 and he coined the term octal.
- The main advantage of using octal numbers is that it uses fewer digits than the decimal and hexadecimal number system. So, it has fewer computations and less computational errors.
- The octal number system is widely used in computer application sectors and digital numbering systems. The octal number is also used in the aviation sector in the form of a code.
- The octal system is similar to the hexadecimal system because they are both easily converted to binary, where octal is equal to three-digit binary and hexadecimal is equal to four-digit binary.

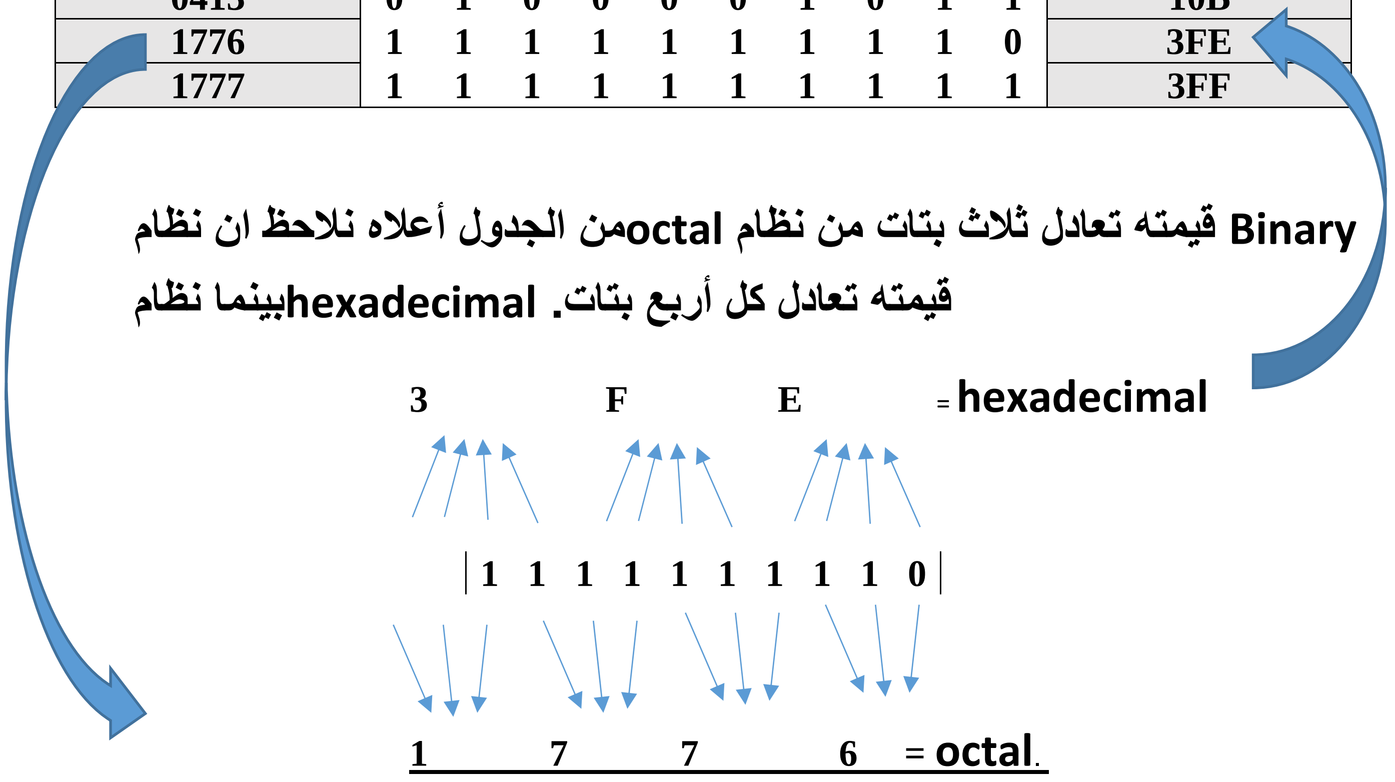
Lec._4

مقارنة بين النظام الثماني والسادس عشر

8^3 8^2 8^1 8^0	10^3 10^2 10^1 10^0	16^2 16^1 16^0
octal	Decimal	**hexadecimal**
0412	0266	10A
0413	0267	10B
1776	1022	3FE
1777	1023	3FF

8^3 8^2 8^1 8^0	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	16^2 16^1 16^0
octal	Binary										**hexadecimal**
0412	0	1	0	0	0	0	1	0	1	0	10A
0413	0	1	0	0	0	0	1	0	1	1	10B
1776	1	1	1	1	1	1	1	1	1	0	3FE
1777	1	1	1	1	1	1	1	1	1	1	3FF

من الجدول أعلاه نلاحظ ان نظام octal قيمته تعادل ثلاث بتات من نظام Binary بينما نظام hexadecimal قيمته تعادل كل أربع بتات.



التحقق:

1. نظام **octal** <= $1 \times 8^3 + 7 \times 8^2 + 7 \times 8^1 + 6 \times 8^0$

$= 6 \times 1 + 56 + 448 + 512$

$= 1022$ بالنظام **decimal.**

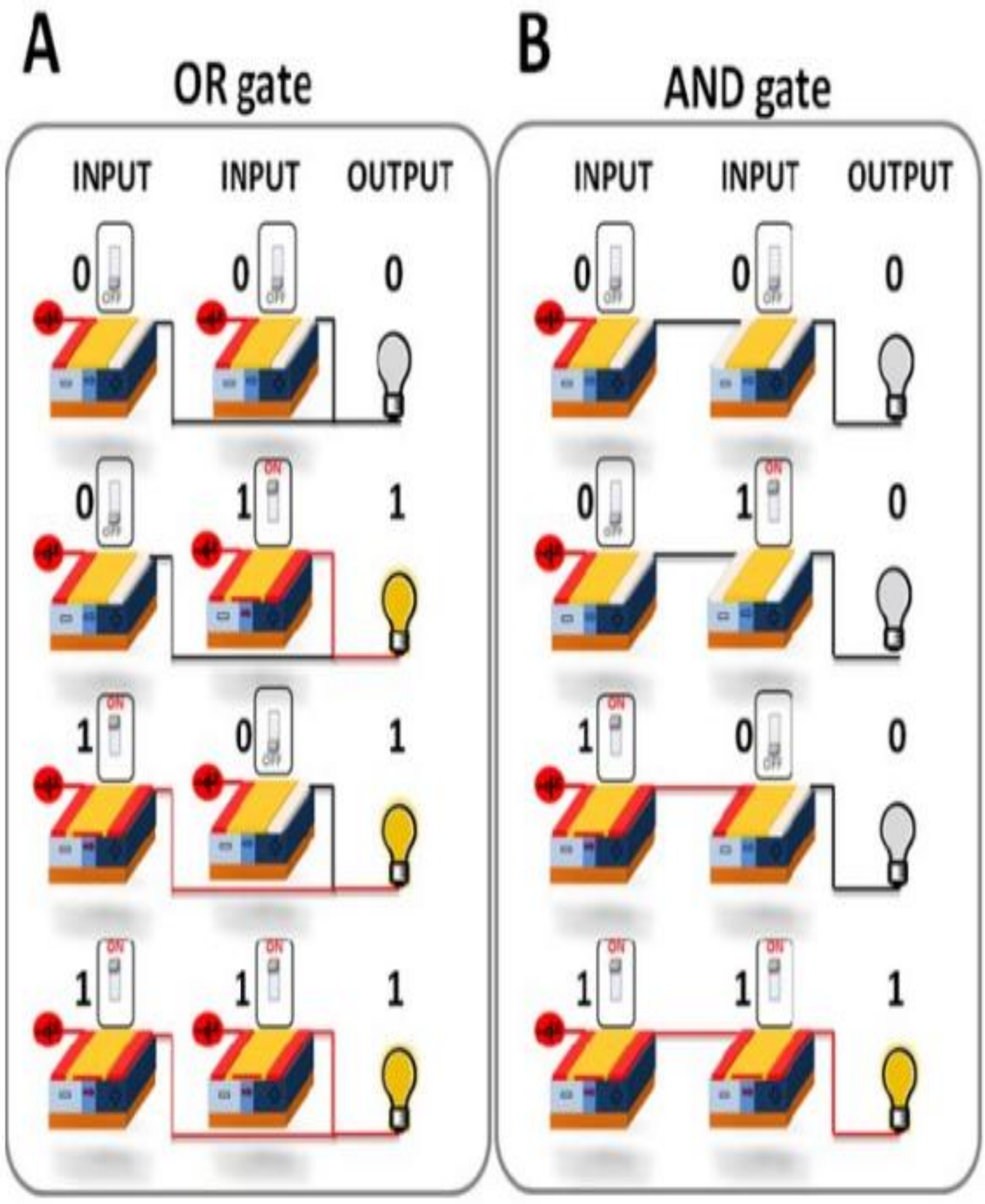
2. نظام **hexadecimal** = > $3 \times 16^2 + F \times 16^1 + E \times 16^0$

= $14 \times 1 + 15 \times 16 + 3 \times 256$

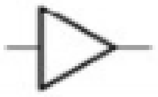
= 1022 بالنظام **decimal**.

Lec._5

Logic gates (AND, OR, NOT, NAND, NOR, XOR, XNOR)

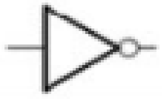


YES



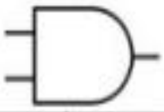
INPUT	OUTPUT
A	
0	0
1	1

NOT



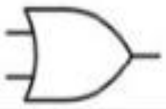
INPUT	OUTPUT
A	
0	1
1	0

AND



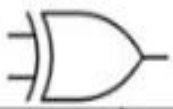
INPUT		OUTPUT
A	B	
0	0	0
1	0	0
0	1	0
1	1	1

OR



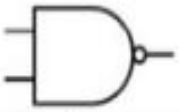
INPUT		OUTPUT
A	B	
0	0	0
1	0	1
0	1	1
1	1	1

XOR



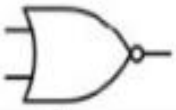
INPUT		OUTPUT
A	B	
0	0	0
1	0	1
0	1	1
1	1	0

NAND



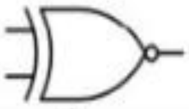
INPUT		OUTPUT
A	B	
0	0	1
1	0	1
0	1	1
1	1	0

NOR



INPUT		OUTPUT
A	B	
0	0	1
1	0	0
0	1	0
1	1	0

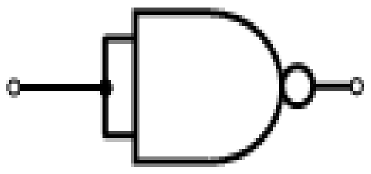
XNOR



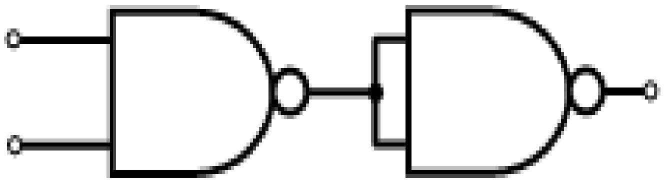
INPUT		OUTPUT
A	B	
0	0	1
1	0	0
0	1	0
1	1	1

Making other logic gates from NAND gates

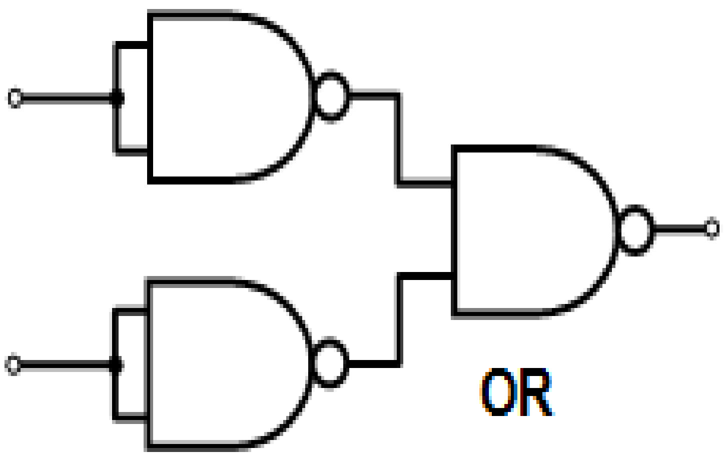
The circuits below show you how to make a NOT, OR, NOR and AND gate using NAND gates.



NOT



AND


OR

Lec._6

Logic simplification (Boolean theorem) & (Demorgan’s theorem):

Boolean theorem

Name	AND form	OR form
Identity law	$1A = A$	$0 + A = A$
Null law	$0A = 0$	$1 + A = 1$
Idempotent law	$AA = A$	$A + A = A$
Inverse law	$A\bar{A} = 0$	$A + \bar{A} = 1$
Commutative law	$AB = BA$	$A + B = B + A$
Associative law	$(AB)C = A(BC)$	$(A + B) + C = A + (B + C)$
Distributive law	$A + BC = (A + B)(A + C)$	$A(B + C) = AB + AC$
Absorption law	$A(A + B) = A$	$A + AB = A$
De Morgan's law	$\overline{AB} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A}\bar{B}$

Demorgan’s theorem

Name	AND form	OR form
De Morgan's law	$\overline{AB} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A}\bar{B}$

Boolean Algebra simplification is not that difficult to understand if you realise that the use of the symbols or signs of: “**+**” and “**.**” represent the operation of logical functions.

- Logical functions test whether a **condition** or state is either *TRUE* or *FALSE* but not both at the same time. So depending on the result of that test, a digital circuit can then decide to do one thing or another.

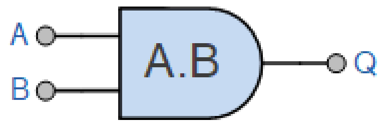
As we saw in the **Laws of Boolean Algebra** tutorial, that Boolean algebra is the mathematics of logic and that the application of various switching theory rules can be applied to simplify long or complex switching algebra notation, and which can also be applied to logic gates and basic digital circuits.

let’s first remind ourselves of a few basic symbols, meanings and laws relating to the three main functions of: **AND**, **OR**, and **NOT**.

The Logic AND Operation

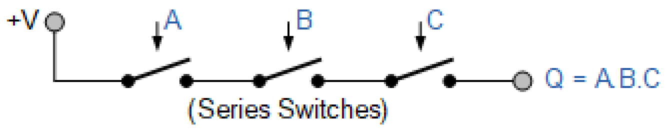
The Boolean expression of A and B is equivalent to A*B. The AND operator is commonly denoted by a single dot or full stop symbol, (**.**). This gives us the Boolean expression of: A.B, or simple AB.

2-input Logic AND Gate



Then it is clear that the logical AND function is used to compare two or more input conditions and returns TRUE only if all of the conditions occur together. The logical AND operation and Boolean expression of A.B.C can be shown in switching algebra as being:

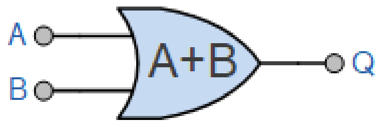
Series (AND) Switching Representation



The Logic OR Operation

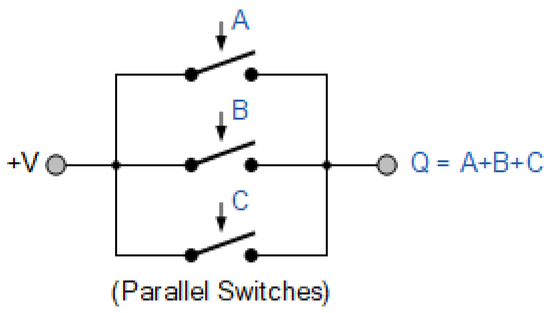
the Boolean expression of A or B is equivalent to A+B. The OR operator is commonly denoted by a plus sign, (+) giving us the Boolean expression of: A+B.

The 2-input Logic OR Gate



Then it is clear that the logical OR function is used to compare two or more input conditions and returns TRUE only if either one of the conditions occurs. The logical OR operation of addition represents a parallel connection with the order in which the switches are connected in parallel being unimportant as it can extend to any number of parallel-connected switches. So our Boolean expression of A+B+C can be shown in switching algebra as being:

Parallel (OR) Switching Representation

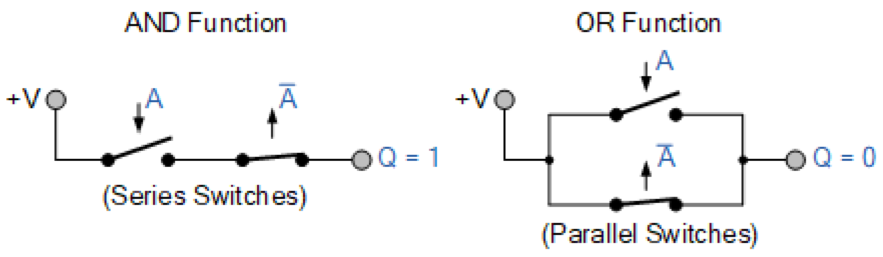


The Logic NOT Operation

The logical NOT operation is simply an inversion or complementation function of a Boolean value and is not considered as a separate variable. The *NOT function* is so called because its output state is "NOT" the same as its input state, (hence its name as an inverter).

This means that if switch A is open, A means that the switch is closed. In other words, the Boolean expression for a NOT function is the output is "0" if the input is "1" and the output is "1" if the input is "0".

NOT Representation



Lec._7

Number systems (excess-3, gray code, conversions, operations, complements)

BCD:

BCD or **Binary Coded Decimal** is a coding scheme used to represent decimal number (0 to 9) in the form of binary digits of a group of 4-bits. Binary coded decimal is the simplest form to convert decimal numbers into their equivalent binary format. Although, binary coded decimal or BCD is not the same as the normal binary representation.
In binary coded decimal (BCD) coding scheme, each decimal digit is represented as a group of 4-bit binary number. For a multi-digit decimal number, each digit of the decimal number is encoded separately in the BCD.
As we know, a 4-bit binary number can represent 16 decimal digits, but in binary coded decimal, BCD codes 1010, 1011, 1100, 1101, 1110, and 1111 equivalent to decimal 10, 11, 12, 13, 14, and 15 are considered illegal combinations.
كما نعلم، يمكن أن يمثل الرقم الثنائي المكون من 4 بتات 16 رقمًا عشريًا، ولكن في النظام العشري المشفر ثنائيًا، تكون رموز
BCD codes 1010, 1011, 1100, 1101, 1110, and 1111 مكافئة الى النظام الرقمي العشري 10, 11, 12,13,14 and 15 ((تعتبر مجموعات غير قانونية)).

The following is the truth table representing binary coded decimal (BCD) equivalent of decimal digits from 0 to 9 –

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

Example: Decimal to Binary Coded Decimal Conversion

The following example shows how a decimal number is converted to a BCD code −

Convert $(125)_{10}$ into its equivalent binary coded decimal (BCD) code.

Decimal number	1				2				5			
BCD Weights	8	4	2	1	8	4	2	1	8	4	2	1
BCD Code	0	0	0	1	0	0	1	0	0	1	0	1

Hence, the binary coded decimal for $(125)_{10}$ is $(0001\ 0010\ 0101)_2$.

Excess-3 code:

The excess-3 code (or XS3) is a non-weighted code used to express code used to express decimal numbers. It is a self-complementary binary coded decimal (BCD) code and numerical system which has biased representation. It is particularly significant for arithmetic operations as it overcomes shortcoming encountered while using 8421 BCD code to add two decimal digits whose sum exceeds 9.

Excess-3 arithmetic uses different algorithm than normal non-biased BCD or binary **positional number system**.

Representation of Excess-3 Code

Excess-3 codes are unweighted and can be obtained by adding 3 to each decimal digit then it can be represented by using 4 bit binary number for each digit. An Excess-3 equivalent of a given binary binary number is obtained using the following steps:

- Find the decimal equivalent of the given binary number.
- Add +3 to each digit of decimal number.
- Convert the newly obtained decimal number back to binary number to get required excess-3 equivalent.

You can add 0011 to each four-bit group in binary coded decimal number (BCD) to get desired excess-3 equivalent.

These are following excess-3 codes for decimal digits −

Decimal Digit	BCD Code	Excess-3 Code
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

The codes 0000 and 1111 are not used for any digit.

Example-1 −Convert decimal number 23 to Excess-3 code.

So, according to excess-3 code we need to add 3 to both digit in the **decimal number** then convert into 4-bit binary number for result of each digit. Therefore,

= 23+33=56 =0101 0110 which is required excess-3 code for given decimal number 23.

Example-2 −Convert decimal number 15.46 into Excess-3 code.

According to excess-3 code we need to add 3 to both digit in the decimal number then convert into 4-bit binary number for result of each digit. Therefore,

= 15.46+33.33=48.79 =0100 1000.0111 1001 which is required excess-3 code for given decimal number 15.46.

Advantages of Excess-3 Codes

These are following advantages of Excess-3 codes,

- These are unweighted binary decimal codes.
- These are self-complementary codes.
- These use biased representation.
- The codes 0000 and 1111 are not used for any digit which is an advantage for memory organization as these codes can cause fault in transmission line.
- It has no limitation, and it considerably simplifies arithmetic operations.
- It is particularly significant for arithmetic operations as it overcomes shortcoming encountered while using 8421 BCD code to add two decimal digits whose sum exceeds 9.

Gray Codes:

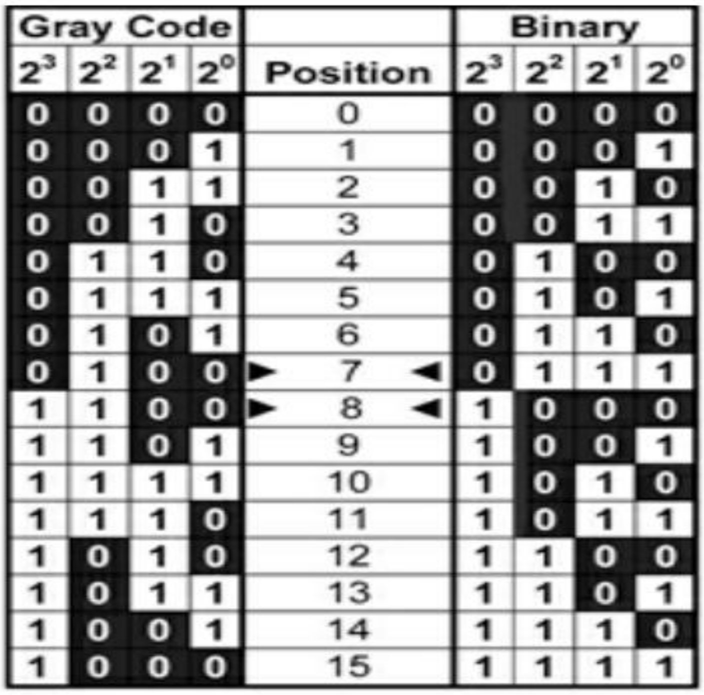
Gray code is a form of binary and the most popular absolute encoder output type. This lecture will explain Gray Code, discuss converting Gray Code to Binary, explain how to use software to convert to gray code, and converting Gray Code to Binary using logic (XOR).

Gray Code uses a different method of incrementing from one number to the next. With Gray Code, only one bit changes state from one position to another. This feature allows a system designer to perform some error checking (i.e., if more than one bit changes, the data must be incorrect).

In table below, explain the difference between Binary and Gray Code.

Gray Code is the most popular absolute encoder output type because its use prevents certain data errors that can occur with Binary during state changes.

For example, in a highly capacitive circuit (or sluggish system response), a Binary state change from 0011 to 0100 could cause the counter/PLC to see 0111. This sort of error is not possible with Gray Code, so the data is more reliable.

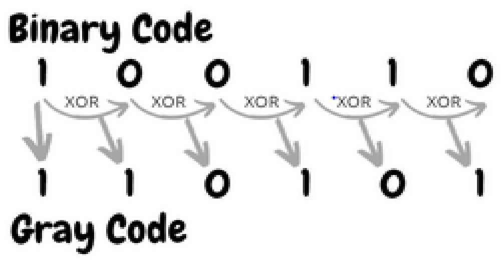


Gray Code					Binary			
2^3	2^2	2^1	2^0	Position	2^3	2^2	2^1	2^0
0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	1
0	0	1	1	2	0	0	1	0
0	0	1	0	3	0	0	1	1
0	1	1	0	4	0	1	0	0
0	1	1	1	5	0	1	0	1
0	1	0	1	6	0	1	1	0
0	1	0	0	▶ 7 ◀	0	1	1	1
1	1	0	0	▶ 8 ◀	1	0	0	0
1	1	0	1	9	1	0	0	1
1	1	1	1	10	1	0	1	0
1	1	1	0	11	1	0	1	1
1	0	1	0	12	1	1	0	0
1	0	1	1	13	1	1	0	1
1	0	0	1	14	1	1	1	0
1	0	0	0	15	1	1	1	1

Note that even from position 7 to 8, Gray Code only changes one bit state.

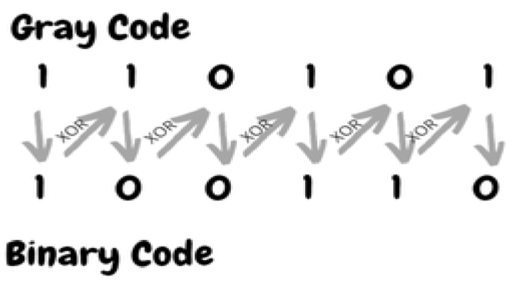
Binary to Gray conversion :

1. The Most Significant Bit (MSB) of the gray code is always equal to the MSB of the given binary code.
2. Other bits of the output gray code can be obtained by XORing binary code bit at that index and previous index.



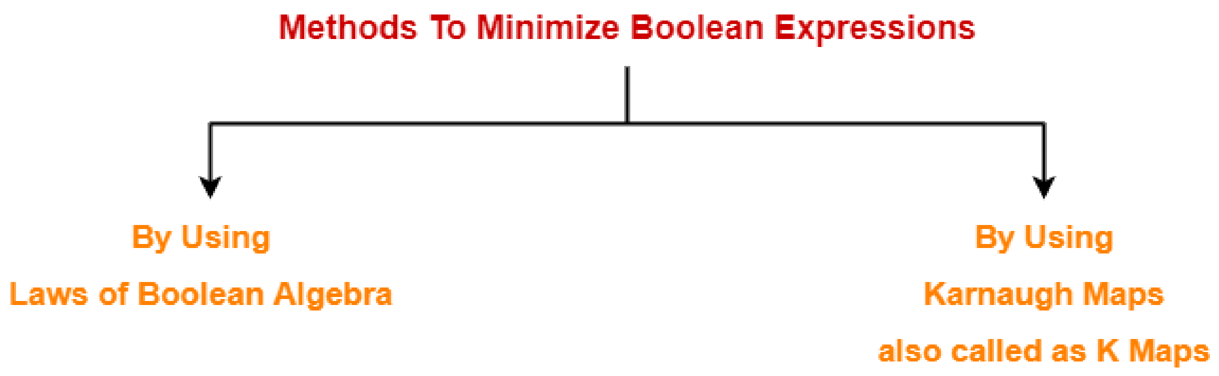
Gray to binary conversion :

1. The Most Significant Bit (MSB) of the binary code is always equal to the MSB of the given gray code.
2. Other bits of the output binary code can be obtained by checking the gray code bit at that index. If the current gray code bit is 0, then copy the previous binary code bit, else copy the invert of the previous binary code bit.



Lec._8

Karnaugh maps



1. By using laws of Boolean Algebra
2. By using Karnaugh Maps also called as K Maps

Karnaugh Map-

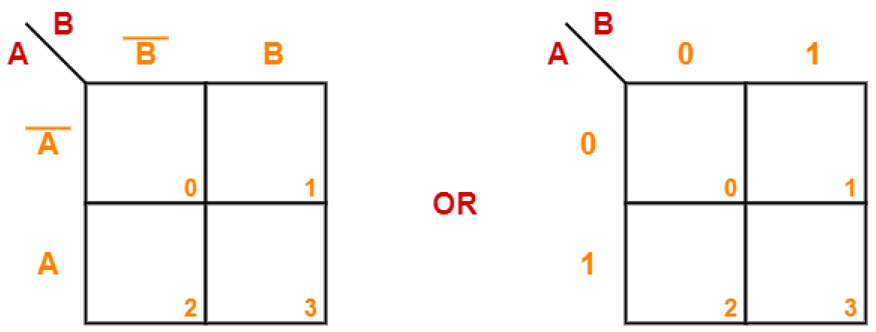
The Karnaugh Map also called as K Map is a graphical representation that provides a systematic method for simplifying the boolean expressions.

For a boolean expression consisting of n-variables, number of cells required in K Map = 2^n cells.

Two Variable K Map-

- Two variable K Map is drawn for a boolean expression consisting of two variables.
- The number of cells present in two variable K Map = 2^2 = 4 cells.
- So, for a boolean function consisting of two variables, we draw a 2 x 2 K Map.

Two variable K Map may be represented as-



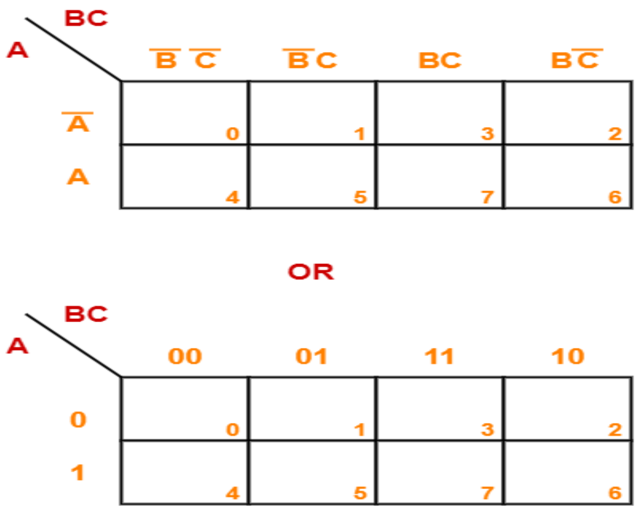
Two Variable K Map

Here, A and B are the two variables of the given boolean function.

Three Variable K Map-

- Three variable K Map is drawn for a boolean expression consisting of three variables.
- The number of cells present in three variable K Map = 2^3 = 8 cells.
- So, for a boolean function consisting of three variables, we draw a 2 x 4 K Map.

Three variable K Map may be represented as-



Three Variable K Map

Here, A, B and C are the three variables of the given boolean function.

Four Variable K Map-

- Four variable K Map is drawn for a boolean expression consisting of four variables.
- The number of cells present in four variable K Map = 2^4 = 16 cells.
- So, for a boolean function consisting of four variables, we draw a 4 x 4 K Map.

Four variable K Map may be represented as-

AB \ CD	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

Four Variable K Map

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	0	1	3	2
$\bar{A}B$	4	5	7	6
AB	12	13	15	14
$A\bar{B}$	8	9	11	10

OR

Here, A, B, C and D are the four variables of the given boolean function.

Karnaugh Map Simplification Rules-

To minimize the given boolean function,

- We draw a K Map according to the number of variables it contains.
- We fill the K Map with 0's and 1's according to its function.
- Then, we minimize the function in accordance with the following rules.

Rule-01:

- We can either group 0's with 0's or 1's with 1's but we can not group 0's and 1's together.
- X representing don't care can be grouped with 0's as well as 1's.

NOTE

There is no need of separately grouping X's i.e. they can be ignored if all 0's and 1's are already grouped.

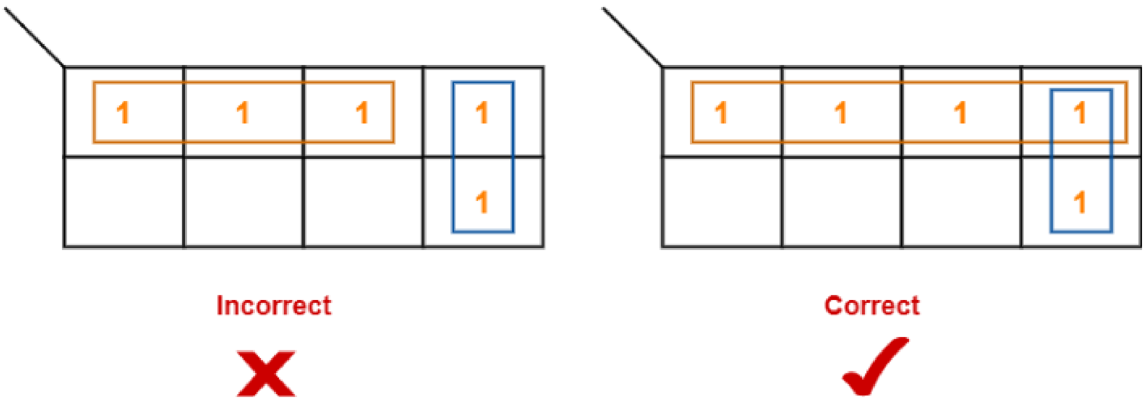
Rule-02:

- Groups may overlap each other.

Rule-03:

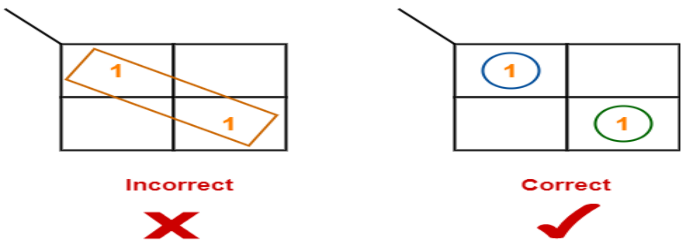
- We can only create a group whose number of cells can be represented in the power of 2.
- In other words, a group can only contain 2^n i.e. 1, 2, 4, 8, 16 and so on number of cells.

Example-

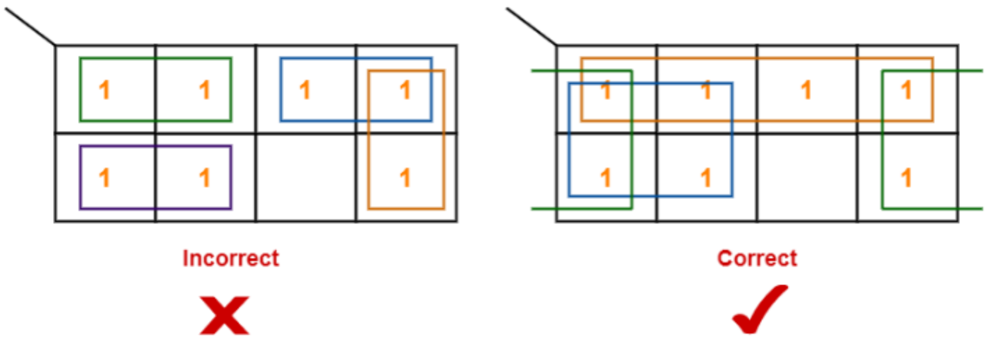


Rule-04:

- Groups can be only either horizontal or vertical.
- We can not create groups of diagonal or any other shape.



Example-



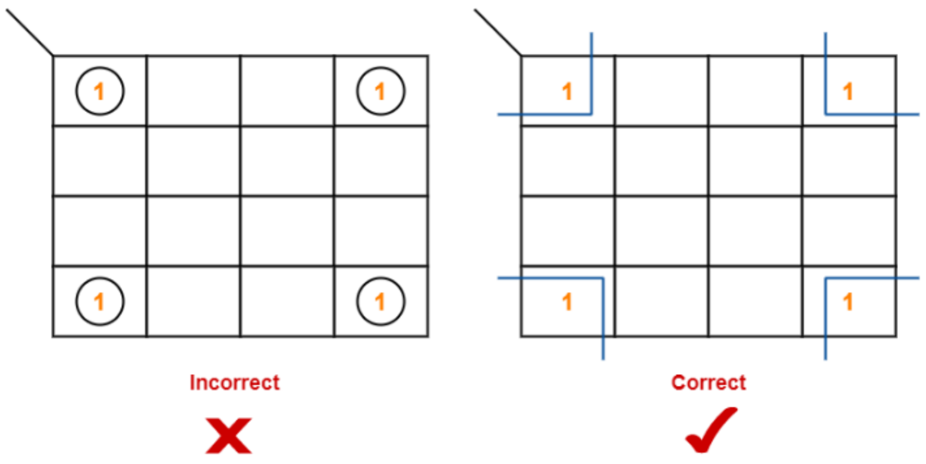
Rule-05:

- Each group should be as large as possible.

Rule-06:

- Opposite grouping and corner grouping are allowed.
- The example of opposite grouping is shown illustrated in Rule-05.
- The example of corner grouping is shown below.

Example-



Rule-07:

- There should be as few groups as possible.

PROBLEMS BASED ON KARNAUGH MAP-

Problem-01:

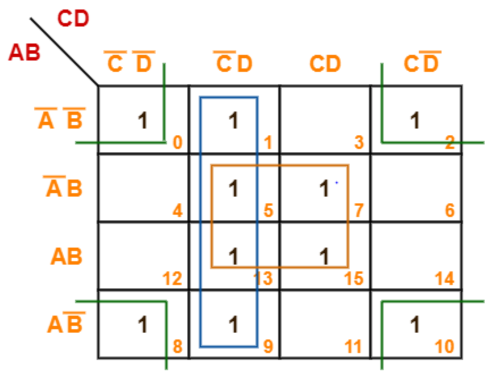
Minimize the following boolean function-

$$F(A, B, C, D) = \Sigma m(0, 1, 2, 5, 7, 8, 9, 10, 13, 15)$$

Solution-

- Since the given boolean expression has 4 variables, so we draw a 4 x 4 K Map.
- We fill the cells of K Map in accordance with the given boolean function.
- Then, we form the groups in accordance with the above rules.

Then, we have-



Now,

F(A, B, C, D)

= (A’B + AB)(C’D + CD) + (A’B’ + A’B + AB + AB’)C’D + (A’B’ + AB’)(C’D’ + CD’)

= BD + C’D + B’D’

Thus, minimized boolean expression is-

F(A, B, C, D) = BD + C’D + B’D’

Problem-02:

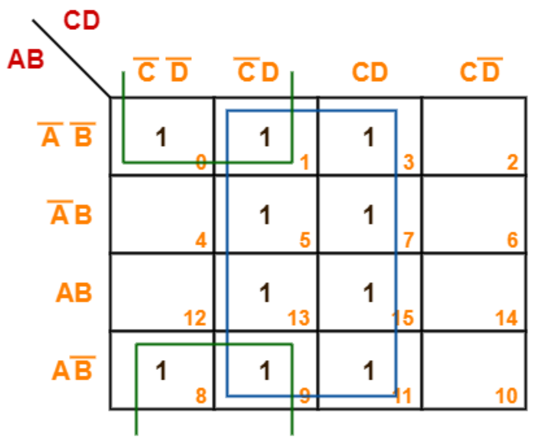
Minimize the following boolean function-

F(A, B, C, D) = Σm(0, 1, 3, 5, 7, 8, 9, 11, 13, 15)

Solution-

- Since the given boolean expression has 4 variables, so we draw a 4 x 4 K Map.
- We fill the cells of K Map in accordance with the given boolean function.
- Then, we form the groups in accordance with the above rules.

Then, we have-



Now,

F(A, B, C, D)

= (A’B’ + A’B + AB + AB’)(C’D + CD) + (A’B’ + AB’)(C’D’ + C’D)

= D + B’C’

Thus, minimized boolean expression is-

F(A, B, C, D) = B’C’ + D

Problem-03:

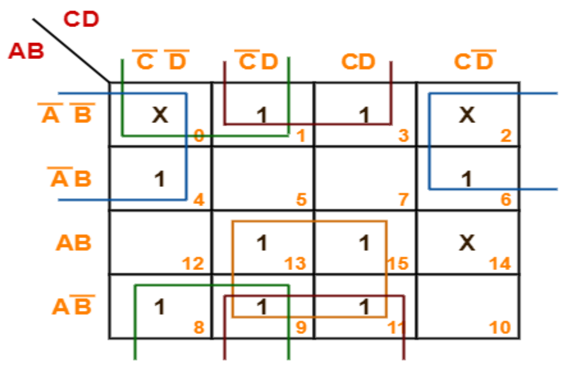
Minimize the following boolean function-

F(A, B, C, D) = Σm(1, 3, 4, 6, 8, 9, 11, 13, 15) + Σd(0, 2, 14)

Solution-

- Since the given boolean expression has 4 variables, so we draw a 4 x 4 K Map.
- We fill the cells of K Map in accordance with the given boolean function.
- Then, we form the groups in accordance with the above rules.

Then, we have-



Now,

F(A, B, C, D)

= (AB + AB’)(C’D + CD) + (A’B’ + AB’)(C’D + CD) + (A’B’ + AB’)(C’D’ + C’D) + (A’B’ + A’B)(C’D’ + CD’)

= AD + B’D + B’C’ + A’D’

Thus, minimized boolean expression is-

F(A, B, C, D) = AD + B’D + B’C’ + A’D’

Problem-04:

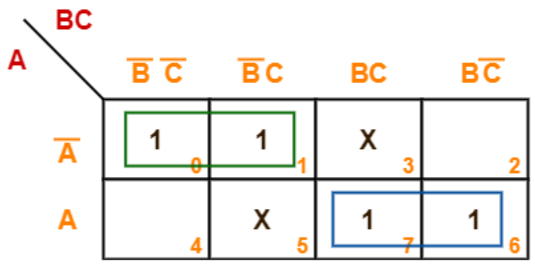
Minimize the following boolean function-

F(A, B, C) = Σm(0, 1, 6, 7) + Σd(3, 5)

Solution-

- Since the given boolean expression has 3 variables, so we draw a 2 x 4 K Map.
- We fill the cells of K Map in accordance with the given boolean function.
- Then, we form the groups in accordance with the above rules.

Then, we have-



Now,

F(A, B, C)

= A'(B’C’ + B’C) + A(BC + BC’)

= A’B’ + AB

Thus, minimized boolean expression is-

F(A, B, C) = AB + A’B’

NOTE-

- It may be noted that there is no need of considering the quad group.
- This is because even if we consider that group, we will have to consider the other two duets.
- So, there is no use of considering that quad group.

Problem-05:

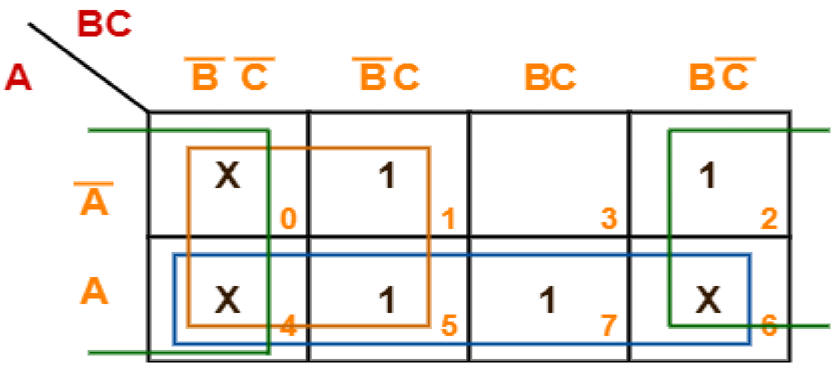
Minimize the following boolean function-

F(A, B, C) = Σm(1, 2, 5, 7) + Σd(0, 4, 6)

Solution-

- Since the given boolean expression has 3 variables, so we draw a 2 x 4 K Map.
- We fill the cells of K Map in accordance with the given boolean function.
- Then, we form the groups in accordance with the above rules.

Then, we have-



Now,

F(A, B, C)

= (A + A’)(B’C’ + B’C) + A(B’C’ + B’C + BC + BC’) + (A + A’)(B’C’ + BC’)

= B’ + A + C’

Thus, minimized boolean expression is-

F(A, B, C) = A + B’ + C’

Problem-06:

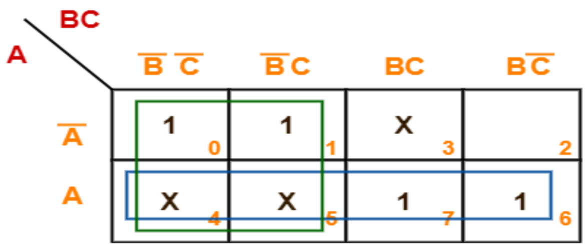
Minimize the following boolean function-

F(A, B, C) = Σm(0, 1, 6, 7) + Σd(3, 4, 5)

Solution-

- Since the given boolean expression has 3 variables, so we draw a 2 x 4 K Map.
- We fill the cells of K Map in accordance with the given boolean function.
- Then, we form the groups in accordance with the above rules.

Then, we have-



Now,

F(A, B, C)

= (A + A’)(B’C’ + B’C) + A(B’C’ + B’C + BC + BC’)

= B’ + A

Thus, minimized boolean expression is-

F(A, B, C) = A + B’

Problem-07:

Minimize the following boolean function-

F(A, B, C, D) = Σm(0, 2, 8, 10, 14) + Σd(5, 15)

Solution-

- Since the given boolean expression has 4 variables, so we draw a 4 x 4 K Map.
- We fill the cells of K Map in accordance with the given boolean function.
- Then, we form the groups in accordance with the above rules.

Then, we have-

Now,

F(A, B, C, D)

= (AB + AB’)CD’ + (A’B’ + AB’)(C’D’ + CD’)

= ACD’ + B’D’

Thus, minimized boolean expression is-

F(A, B, C, D) = ACD’ + B’D’

Problem-08:

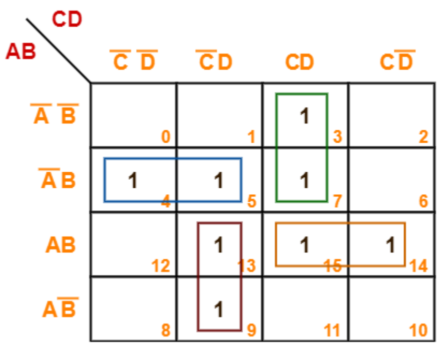
Minimize the following boolean function-

F(A, B, C, D) = Σm(3, 4, 5, 7, 9, 13, 14, 15)

Solution-

- Since the given boolean expression has 4 variables, so we draw a 4 x 4 K Map.
- We fill the cells of K Map in accordance with the given boolean function.
- Then, we form the groups in accordance with the above rules.

Then, we have-



Now,

F(A, B, C, D)

= A’B(C’D’ + C’D) + (A’B’ + A’B)(CD) + (AB + AB’)(C’D) + AB(CD + CD’)

= A’BC’ + A’CD + AC’D + ABC

Thus, minimized boolean expression is-

F(A, B, C, D) = A’BC’ + A’CD + AC’D + ABC

It is important to note that we are not considering the quad group because we have to consider the duets anyhow.

Problem-09:

Consider the following boolean function-

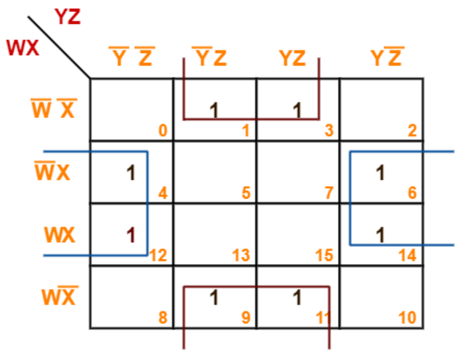
F(W, X, Y, Z) = Σm(1, 3, 4, 6, 9, 11, 12, 14)

This function is independent ________ number of variables. Fill in the blank.

Solution-

- Since the given boolean expression has 4 variables, so we draw a 4 x 4 K Map.
- We fill the cells of K Map in accordance with the given boolean function.
- Then, we form the groups in accordance with the above rules.

Then, we have-



Now,

F(W, X, Y, Z)

= (W'X + WX)(Y'Z' + YZ') + (W'X' + WX')(Y'Z + YZ)

= XZ' + X'Z

= X ⊕ Z

Thus, minimized boolean expression is-

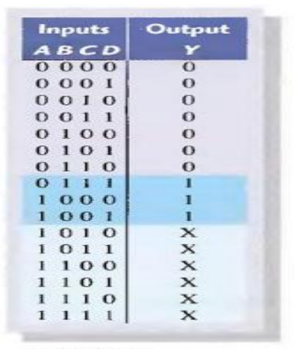
F(W, X, Y, Z) = X ⊕ Z

Clearly, the given boolean function depends on only two variables X and Z.

Hence, it is independent of other two variables W and Y.

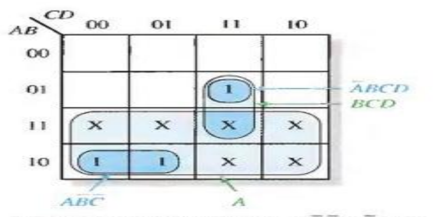
"Don't Care" Conditions

Sometimes a situation arises in which some input variable combinations are not allowed. For example, recall that in the BCD code there are six invalid combinations: 1010, 1011, 1100, 1101, 1110, and 1111. Since these unallowed states will never occur in an application involving the BCD code, they can be treated as "don't care" terms with respect to their effect on the output. That is, for these "don't care" terms either a 1 or a 0 may be assigned to the output: it really does not matter since they will never occur. The "don't care" terms can be used to advantage on the Karnaugh map. Fig.(4-9) shows that for each "don't care" term, an X is placed in the cell. When grouping the 1 s, the Xs can be treated as 1s to make a larger grouping or as 0s if they cannot be used to advantage. The larger a group, the simpler the resulting term will be.



Inputs	Output
A B C D	Y
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	1
1 0 0 0	1
1 0 0 1	1
1 0 1 0	X
1 0 1 1	X
1 1 0 0	X
1 1 0 1	X
1 1 1 0	X
1 1 1 1	X

(a) Truth table



(b) Without "don't cares" $Y = A\bar{B}\bar{C} + \bar{A}BCD$
With "don't cares" $Y = A + BCD$

The truth table in above: (a) describes a logic function that has a 1 output only when the BCD code for 7,8, or 9 is present on the inputs. If the "don't cares" are used as 1s, the resulting expression for the function is A + BCD, as indicated in part (b). If the "don't cares" are not used as 1s, the resulting expression is ABC + ABCD: so you can see the advantage of using "don't care" terms to get the simplest expression.

Lec._9

K-map forms:

In many digital circuits and practical problems, we need to find expressions with minimum variables. We can minimize Boolean expressions of 3, 4 variables very easily using K-map without using any Boolean algebra theorems.
K-map can take two forms:

1. Sum of product (SOP)
2. Product of Sum (POS)

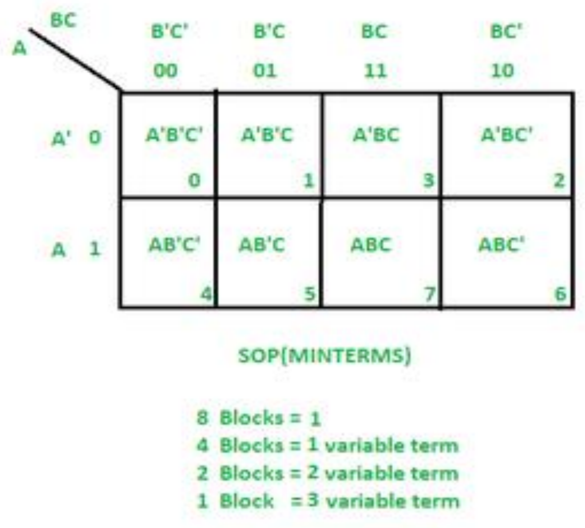
According to the need of problem. K-map is a table-like representation, but it gives more information than the TABLE. We fill a grid of the K-map with 0's and 1's then solve it by making groups.

Steps to Solve Expression using K-map

1. Select the K-map according to the number of variables.
2. Identify minterms or maxterms as given in the problem.
3. For SOP put 1's in blocks of K-map respective to the minterms (0's elsewhere).
4. For POS put 0's in blocks of K-map respective to the max terms (1's elsewhere).
5. Make rectangular groups containing total terms in power of two like 2,4,8 ..(except 1) and try to cover as many elements as you can in one group.
6. From the groups made in step 5 find the product terms and sum them up for SOP form.

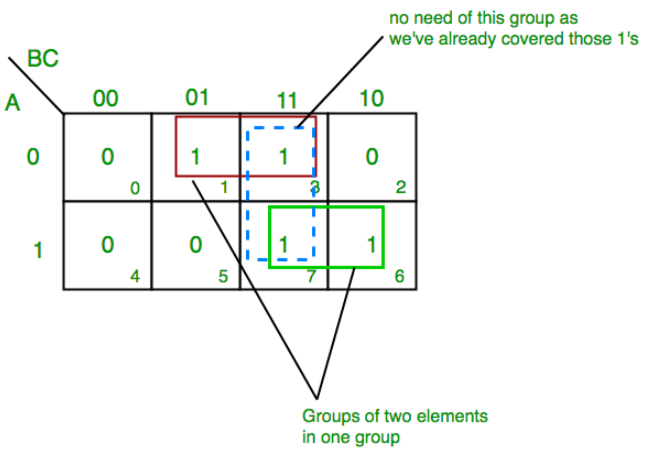
SOP FORM

1. K-map of 3 variables



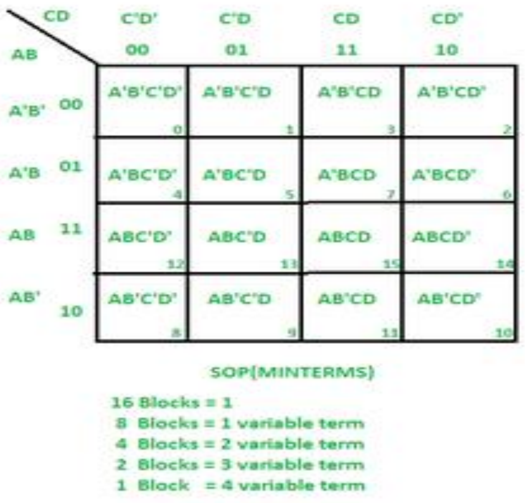
K-map SOP form for 3 variables

`Z=?A,B,C(1,3,6,7)`


no need of this group as we've already covered those 1's
BC
A
00
01
11
10
0
1
Groups of two elements in one group

From **red** group we get product term— `A’C`
From **green** group we get product term— `AB`
Summing these product terms we get- **Final expression (A’C+AB)**

2. K-map for 4 variables

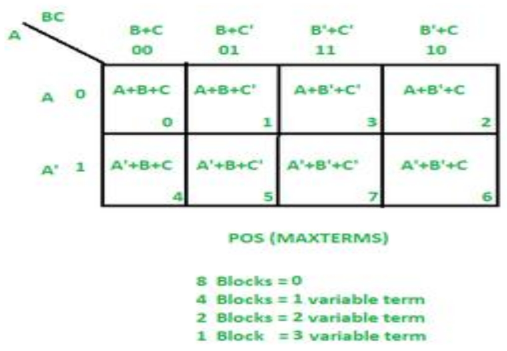


K-map 4 variable SOP form

`F(P,Q,R,S)=?(0,2,5,7,8,10,13,15)`
From **red** group we get product term— `QS`
From **green** group we get product term— `Q’S’`
Summing these product terms we get- **Final expression (QS+Q’S’)**.

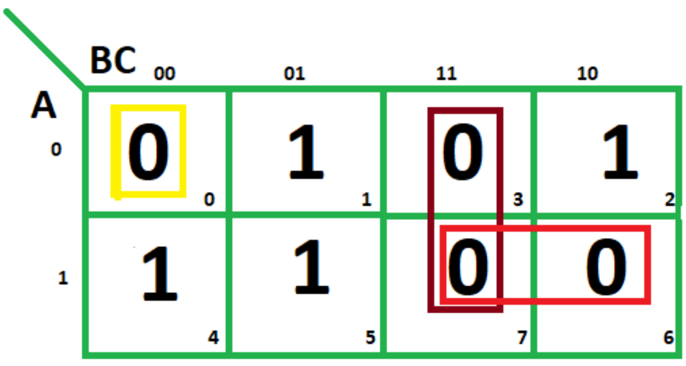
POS FORM

1. K-map of 3 variables



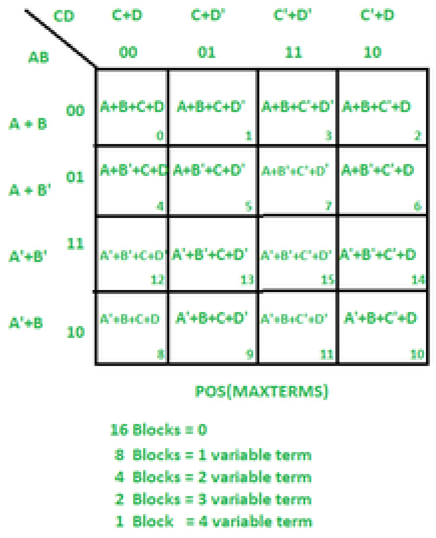
K-map 3 variable POS form

```
F(A,B,C)=?(0,3,6,7)
```



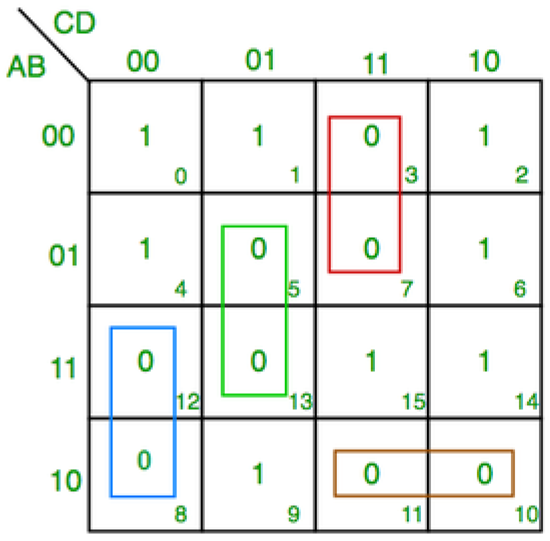
From **red** group we find terms
`A      B`
Taking complement of these two
`A'      B'`
Now **sum** up them
`(A' + B')`
From **brown** group we find terms
`B    C`
Taking complement of these two terms
`B’   C’`
Now sum up them
`(B’+C’)`
From **yellow** group we find terms
`A' B' C’`
Taking complement of these two
`A B C`
Now **sum** up them
`(A + B + C)`
We will take product of these three terms : **Final expression –**
`(A' + B’) (B’ + C’) (A + B + C)`

2. K-map of 4 variables



K-map 4 variable POS form

F(A,B,C,D)=?(3,5,7,8,10,11,12,13)



From **green** group we find terms
C’ D B
Taking their complement and summing them
(C+D’+B’)
From **red** group we find terms
C D A’
Taking their complement and summing them
(C’+D’+A)
From **blue** group we find terms
A C’ D’
Taking their complement and summing them
(A’+C+D)

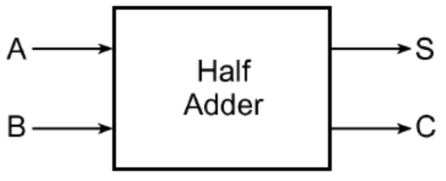
From **brown** group we find terms
A B’ C
Taking their complement and summing them
(A’+B+C’)
Finally we express these as product – **(C+D’+B’).(C’+D’+A).(A’+C+D).(A’+B+C’)**

<u>PITFALL</u>– *Always remember ***POS ? (SOP)’***
*The correct form is (**POS of F)=(SOP of F’)’**

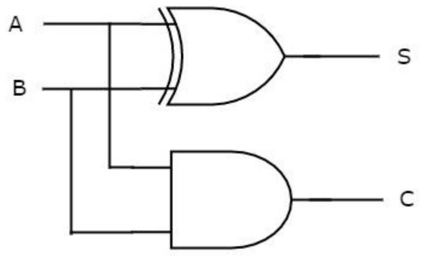
Lec._10

Arithmetic operations (half adder, full adder circuit)

Half Adder and Full Adder

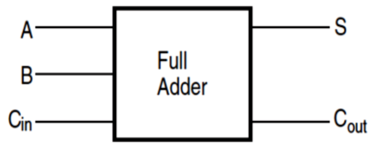


(a)

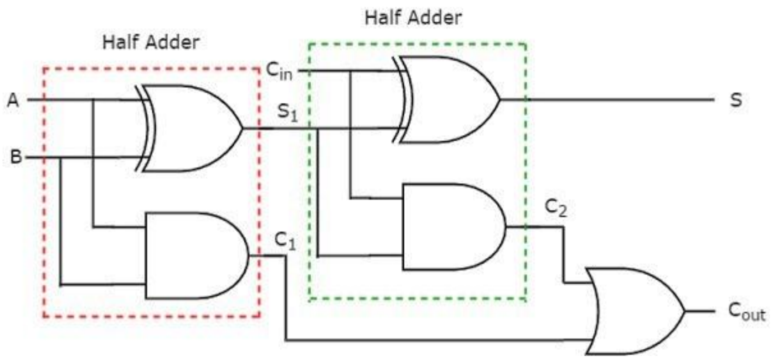


(b)

)a الدائرة المنطقية التي تمثل (Half Adder) b مخطط الكتلة ((Block diagram) لدائرة Half Adder.



(b)



(a)

.Full Adder لدائرة (Block diagram) مخطط الكتلة (b) Full Adder الدائرة المنطقية التي تمثل (a)

جدولHalf Adder

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

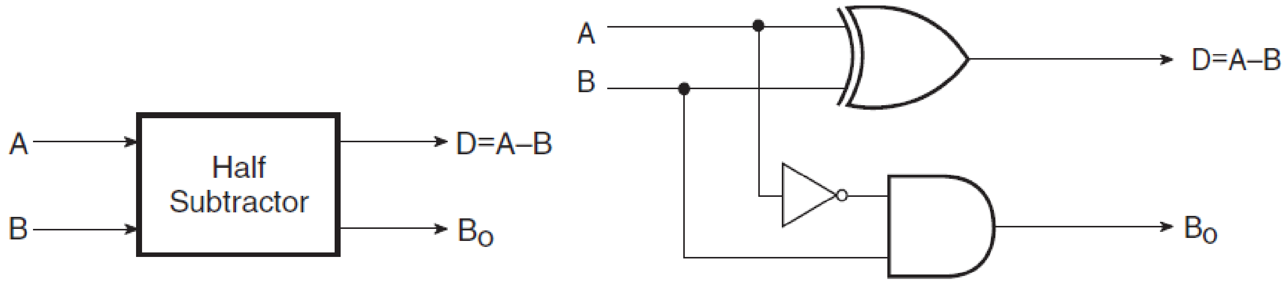
جدولFull adder

A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	0	1
1	1	1	1	1

Lec._11

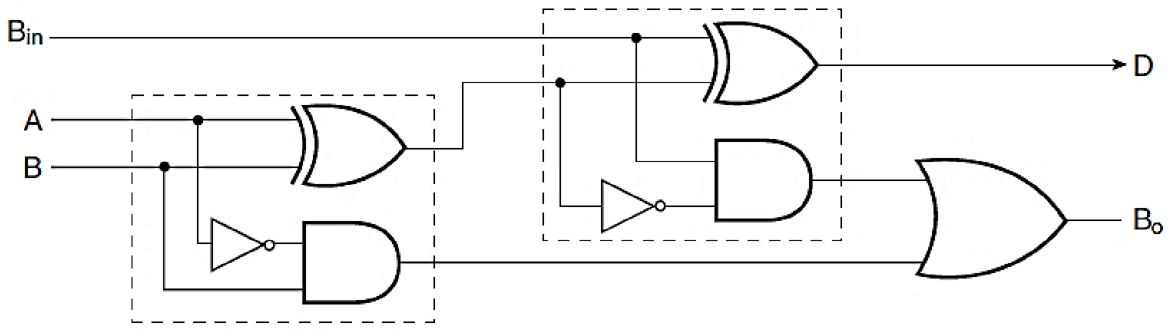
Arithmetic operations (Half subtractor, Full subtractor circuit)

Half Subtractor and Full Subtractor

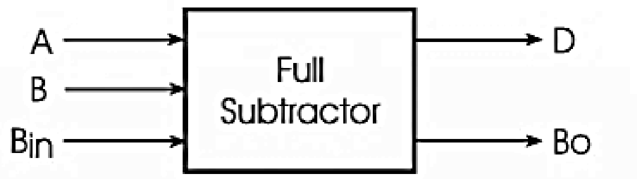


(a) (b)

الشكل 1)a الدائرة المنطقية التي تمثل (Half Subtractor) b مخطط الكتلة ((Block diagram) لدائرة Half Subtractor.



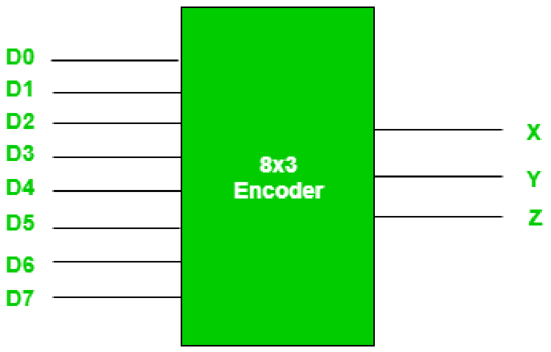
(a)

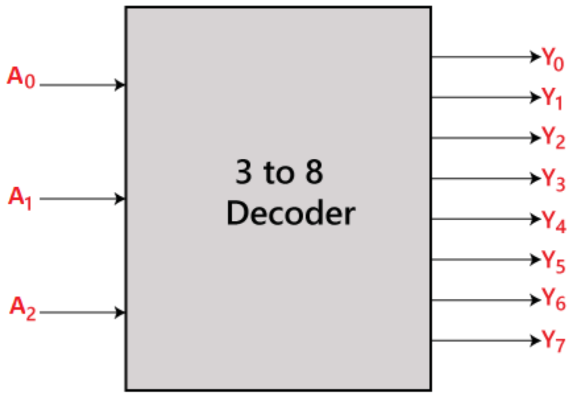


(b)

Lec._12

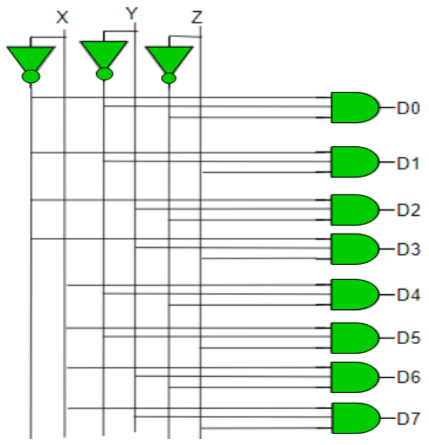
(decoder, encoder)





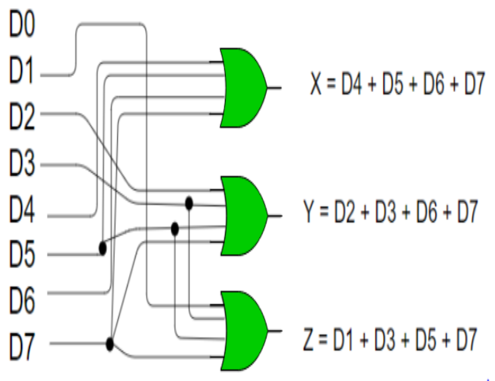
Truth Table Decoder:

X	Y	Z	D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1



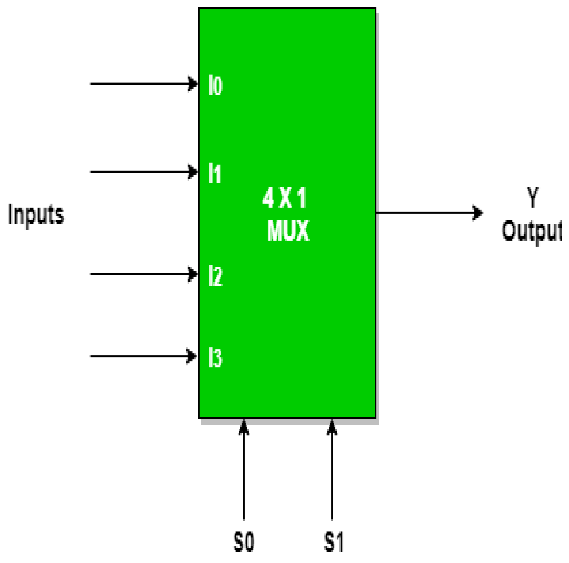
Truth Table Encoder:

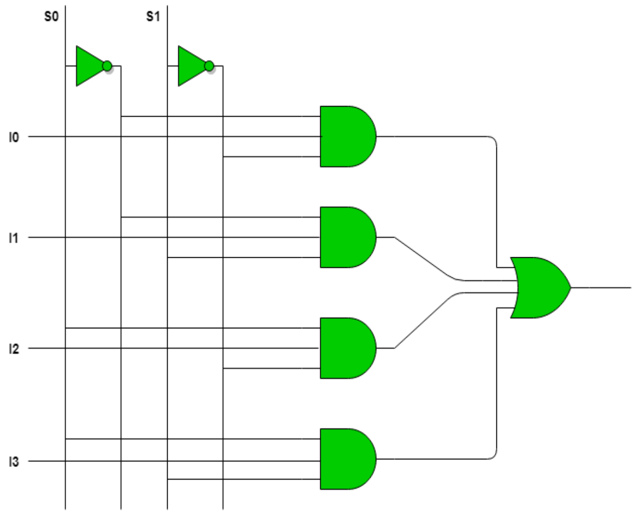
D7	D6	D5	D4	D3	D2	D1	D0	X	Y	Z
0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	1
0	0	0	0	0	1	0	0	0	1	0
0	0	0	0	1	0	0	0	0	1	1
0	0	0	1	0	0	0	0	1	0	0
0	0	1	0	0	0	0	0	1	0	1
0	1	0	0	0	0	0	0	1	1	0
1	0	0	0	0	0	0	0	1	1	1



Lec._1 3

Multiplexer

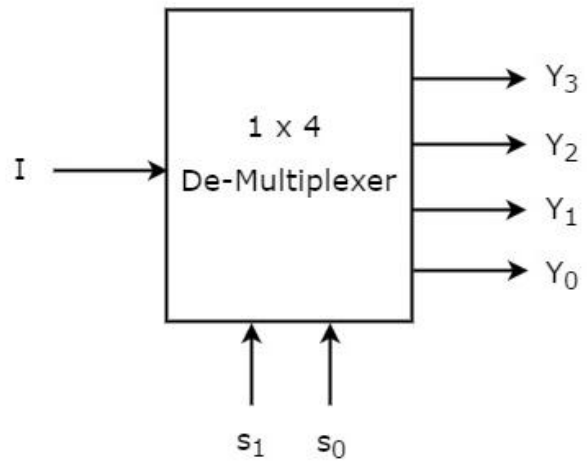


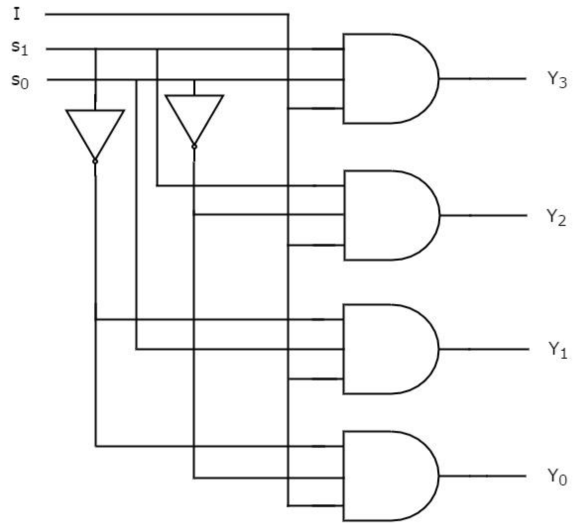


Inputs		Outputs
S_1	S_0	Y
0	0	I0
0	1	I1
1	0	I2
1	1	I3

Lec._1 4

Demultiplexer

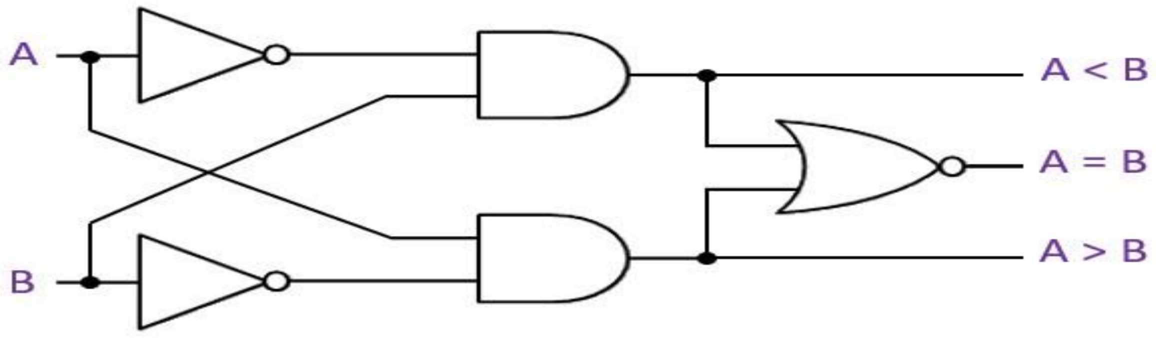


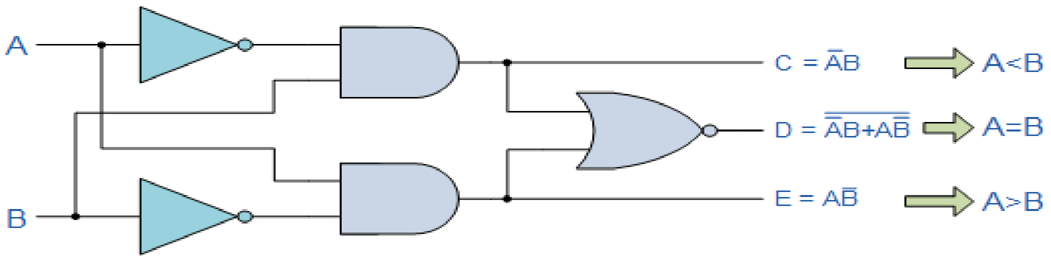


Inputs		Outputs			
S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	I
0	1	0	0	I	0
1	0	0	I	0	0
1	1	I	0	0	0

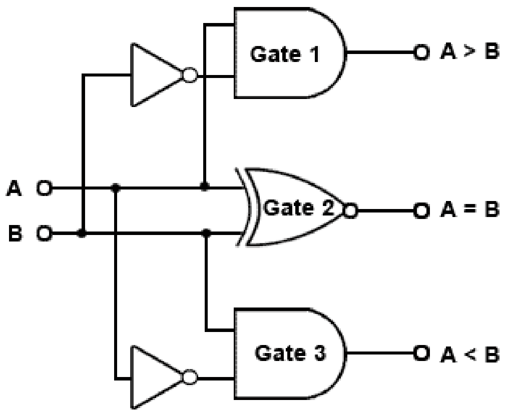
Lec._15

Comparators





OR



Inputs		Outputs		
B	A	A > B	A = B	A < B
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0