Chapter 2

Subject Name: Digital Fundamentals

Subject code:-31307043

Learning Outcomes

- **At the end of this chapter, you must be able to**
- Understood Standard Boolean representation(SOP,POS)
- K-map reduction technique for the Boolean expression
- Develop logic circuits in standard SOP/POS form for the given logical expression.
- Build Design of Arithmetic circuits and code converter using K-map
- Build simple combinational circuits like Multiplexer, demultiplxer, decoder, encoder.
- Draw MUX/DEMUX tree for the given number of input and output lines.
- Develop the specified type of code converter.

Standard Representation

- Any logical expression can be expressed in the following two forms:

1. Sum of product (SOP) form
2. Product of sum(POS)Form

SOP Form

For Example, logical expression given is;

Sum

$$Y = A.B + B.C + A.C$$

Product

POS Form

For Example, logical expression given is;

Product

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

Sum

Standard or Canonical SOP & POS Forms

We can say that a logic expression is said to be in the standard (or canonical) SOP or POS form if each product term (for SOP) and sum term (for POS) consists of all the literals in their complemented or uncomplemented form.

Standard SOP

$$Y = \underset{\uparrow}{ABC} + \underset{\uparrow}{A\overline{B}\,\overline{C}} + \underset{\uparrow}{\overline{A}BC}$$

Each product term consists all the literals

Standard POS

$$Y = \underset{\uparrow}{(A+B+C)}.\underset{\uparrow}{(A+\overline{B}+\overline{C})}.\underset{\uparrow}{(\overline{A}+B+C)}$$

Each sum term consists all the literals

Example

Sr. No.	Expression	Type
1	$Y = AB + AB\overline{C} + \overline{A}BC$	Non Standard SOP
2	$Y = AB + A\overline{B} + \overline{A}\overline{B}$	Standard SOP
3	$Y = (\overline{A} + B).(A + \overline{B}).(\overline{A} + \overline{B})$	Standard POS
4	$Y = (\overline{A} + B).(A + \overline{B} + C)$	Non Standard POS

Convert given expression into its standard SOP form $Y = AB + A\overline{C} + BC$

$$Y = AB + A\overline{C} + BC$$

Missing literal is A

Missing literal is B

Missing literal is C

$$Y = AB.\underline{(C + \overline{C})} + A\overline{C}.\underline{(B + \overline{B})} + BC.\underline{(A + \overline{A})}$$

Term formed by ORing of missing literal & its complement

Convert given expression into its standard SOP form $Y = (A+B).(A+C).(B+\overline{C})$

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

Missing literal is A

Missing literal is B

Missing literal is C

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

Term formed by ANDing of missing literal & its complement

Concept of Minterm and Maxterm

Minterm: Each individual term in the standard SOP form is called as "Minterm".

- **Maxterm:** Each individual term in the standard POS form is called as "Maxterm".
- The concept of minterm and max term allows us to introduce a very convenient shorthand notation to express logic functions .

- Each minterm is represented by mi where i=0,1,2,3,…….,2^{n-1}
- Each maxterm is represented by Mi where i=0,1,2,3,…….,2^{n-1}
- If 'n' number of variables forms the function, then number of minterms or maxterms will be 2^n i.e. for 3 variables function f(A,B,C), the number of minterms or maxterms are $2^3=8$

Minterms & Maxterms for 2 variable/literal logic function

Variables		Minterms	Maxterms
A	B	mi	Mi
0	0	$\overline{A}\,\overline{B} = m_0$	$A + B = M_0$
0	1	$\overline{A}B = m_1$	$A + \overline{B} = M_1$
1	0	$A\overline{B} = m_2$	$\overline{A} + B = M_2$
1	1	$AB = m_3$	$\overline{A} + \overline{B} = M_3$

Minterms & Maxterms for 3 variable/literal logic function

Variables			Minterms	Maxterms
A	B	C	mi	Mi
0	0	0	$\overline{A}\,\overline{B}\,\overline{C} = m_0$	$A + B + C = M_0$
0	0	1	$\overline{A}\,\overline{B}C = m_1$	$A + B + \overline{C} = M_1$
0	1	0	$\overline{A}B\overline{C} = m_2$	$A + \overline{B} + C = M_2$
0	1	1	$\overline{A}BC = m_3$	$A + \overline{B} + \overline{C} = M_3$
1	0	0	$A\overline{B}\,\overline{C} = m_4$	$\overline{A} + B + C = M_4$
1	0	1	$A\overline{B}C = m_5$	$\overline{A} + B + \overline{C} = M_5$
1	1	0	$AB\overline{C} = m_6$	$\overline{A} + \overline{B} + C = M_6$
1	1	1	$ABC = m_7$	$\overline{A} + \overline{B} + \overline{C} = M_7$

Representation of Logical expression using minterm and maxterm

$$Y = \underbrace{ABC}_{m_7} + \underbrace{\overline{A}BC}_{m_3} + \underbrace{A\overline{B}\overline{C}}_{m_4} + \underbrace{A\overline{B}C}_{m_5}$$

$\longleftarrow$ **Logical Expression**

$\longleftarrow$ **Corresponding minterms**

$$Y = m_7 + m_3 + m_4 + m_5$$

$$Y = \Sigma m(3,4,5,7)$$

OR

$$Y = f(A,B,C) = \Sigma m(3,4,5,7)$$

where Σ denotes sum of products

$$Y = (\underbrace{A + \overline{B} + C}_{M_2}).(\underbrace{A + B + C}_{M_0}).(\underbrace{\overline{A} + \overline{B} + C}_{M_6})$$

$\longleftarrow$ **Logical Expression**

$\longleftarrow$ **Corresponding maxterms**

$$Y = M_2.M_0.M_6$$

$$Y = \Pi M(0,2,6)$$

OR

$$Y = f(A,B,C) = \Pi M(0,2,6)$$

where Π denotes product of sum

Karnaugh Map (K-map)

- In the algebraic method of simplification, we need to write lengthy equations, find the common terms, manipulate the expressions etc., so it is time consuming work.
- Thus “K-map” is another simplification technique to reduce the Boolean equation.
- It overcomes all the disadvantages of algebraic simplification techniques.
- The information contained in a truth table or available in the SOP or POS form is represented on K-map.

What are Karnaugh[1] maps?

- Karnaugh maps provide an alternative way of simplifying logic circuits.
- Instead of using Boolean algebra simplification techniques, you can transfer logic values from a Boolean statement or a truth table into a Karnaugh map.
- The arrangement of 0's and 1's within the map helps you to visualise the logic relationships between the variables and leads directly to a simplified Boolean statement.

- Karnaugh maps, or K-maps, are often used to simplify logic problems with 2, 3 or 4 variables.

$\overline{A}\,\overline{B}$

Cell = 2^n ,where n is a number of variables

For the case of 2 variables, we form a map consisting of 2^2=4 cells as shown in Figure

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

Maxterm

B \ A	0	1
0	00 0	10 2
1	01 1	11 3

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

Minterm

✓ 3 Variable K-map & its associated minterms

C \ AB	00	01	11	10
0	m_0	m_2	m_6	m_4
1	m_1	m_3	m_7	m_5

BC \ A	0	1
00	m_0	m_4
01	m_1	m_5
11	m_3	m_7
10	m_2	m_6

A \ BC	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

✓ 3 Variable K-map & its associated product terms

$C \backslash AB$	00	01	11	10
0	$\overline{A}\,\overline{B}\,\overline{C}$	$\overline{A}B\overline{C}$	$AB\overline{C}$	$A\overline{B}\,\overline{C}$
1	$\overline{A}\,\overline{B}C$	$\overline{A}BC$	ABC	$A\overline{B}C$

$A \backslash BC$	00	01	11	10
0	$\overline{A}\,\overline{B}\,\overline{C}$	$\overline{A}\,\overline{B}C$	$\overline{A}BC$	$\overline{A}B\overline{C}$
1	$A\overline{B}\,\overline{C}$	$A\overline{B}C$	ABC	$AB\overline{C}$

$BC \backslash A$	0	1
00	$\overline{A}\,\overline{B}\,\overline{C}$	$A\overline{B}\,\overline{C}$
01	$\overline{A}\,\overline{B}C$	$A\overline{B}C$
11	$\overline{A}BC$	ABC
10	$\overline{A}B\overline{C}$	$AB\overline{C}$

- 4 variables Karnaugh map

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

✓ 4 Variable K-map and its associated product terms

CD \ AB	00	01	11	10
00	$\overline{AB}\,\overline{CD}$	$\overline{ABCD}$	$AB\overline{CD}$	$A\overline{BC}\,\overline{D}$
01	$\bar{A}\bar{B}\bar{C}D$	$\bar{A}B\bar{C}D$	$AB\bar{C}D$	$A\overline{BC}D$
11	$\overline{AB}CD$	$\bar{A}BCD$	$ABCD$	$A\bar{B}CD$
10	$\bar{A}\bar{B}C\bar{D}$	$\bar{A}BC\bar{D}$	$ABC\bar{D}$	$A\bar{B}C\bar{D}$

AB \ CD	00	01	11	10
00	$\bar{A}\bar{B}\overline{CD}$	$\overline{ABC}D$	$\bar{A}\bar{B}CD$	$\overline{AB}C\bar{D}$
01	$\bar{A}B\bar{C}\bar{D}$	$\bar{A}B\bar{C}D$	$\bar{A}BCD$	$\bar{A}BC\bar{D}$
11	$AB\overline{CD}$	$AB\bar{C}D$	$ABCD$	$ABC\bar{D}$
10	$A\bar{B}\bar{C}\bar{D}$	$A\bar{B}\bar{C}D$	$A\bar{B}CD$	$A\bar{B}C\bar{D}$

Simplification of Karnaugh maps

- Once we plot the logic function or truth table on K-map, we have to use the grouping technique for simplifying the logic function.
- Grouping means the combining the terms in adjacent cells.
- The grouping of either 1's or 0's results in the simplification of Boolean expression.
- If we group the adjacent 1's then the result of simplification is SOP form
- If we group the adjacent 0's then the result of simplification is POS form
- While grouping, we should group most number of 1's.
- The grouping follows the binary rule i.e we can group 1,2,4,8,16,32,……..number of 1's. We cannot group 3,5,7,………number of 1's

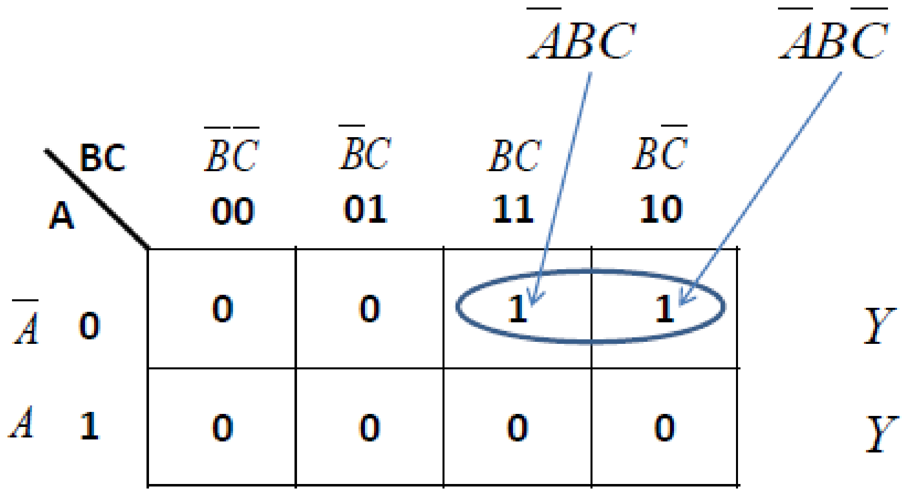
Pair: A group of two adjacent 1's is called as Pair

Quad: A group of four adjacent 1's is called as Quad

Octet: A group of eight adjacent 1's is called as Octet

Grouping of Two Adjacent 1's : Pair

✓ *A pair eliminates 1 variable*



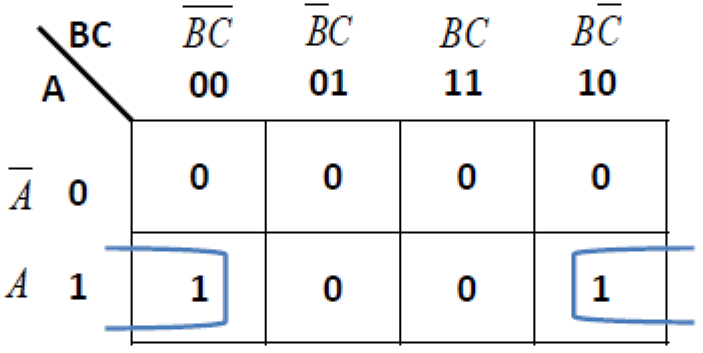
A \ BC	$\overline{B}\,\overline{C}$ 00	$\overline{B}C$ 01	BC 11	$B\overline{C}$ 10
$\overline{A}$ 0	0	0	1	1
A 1	0	0	0	0

$$Y = \overline{A}BC + \overline{A}B\overline{C}$$

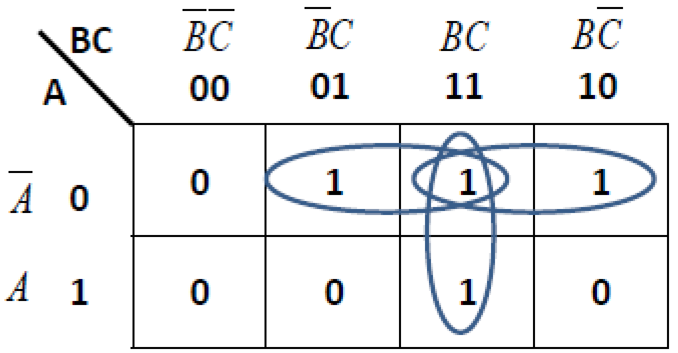
$$Y = \overline{A}B(C + \overline{C})$$

$$Y = \overline{A}B \qquad (\because C + \overline{C} = 1)$$

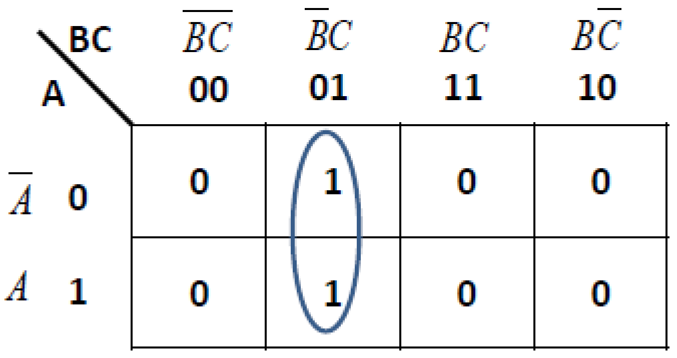
Grouping of Two Adjacent 1's : Pair



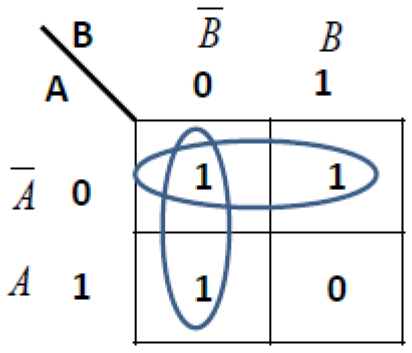
A \ BC	$\overline{B}\overline{C}$ 00	$\overline{B}C$ 01	BC 11	$B\overline{C}$ 10
$\overline{A}$ 0	0	0	0	0
A 1	1	0	0	1



A \ BC	$\overline{B}\overline{C}$ 00	$\overline{B}C$ 01	BC 11	$B\overline{C}$ 10
$\overline{A}$ 0	0	1	1	1
A 1	0	0	1	0



A \ BC	$\overline{B}\overline{C}$ 00	$\overline{B}C$ 01	BC 11	$B\overline{C}$ 10
$\overline{A}$ 0	0	1	0	0
A 1	0	1	0	0



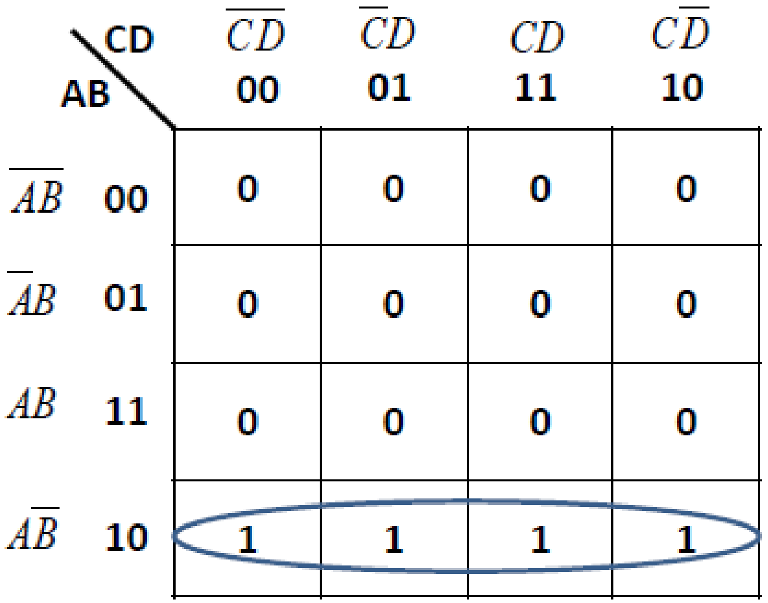
A \ B	$\overline{B}$ 0	B 1
$\overline{A}$ 0	1	1
A 1	1	0

Grouping of Two Adjacent 1's : Pair

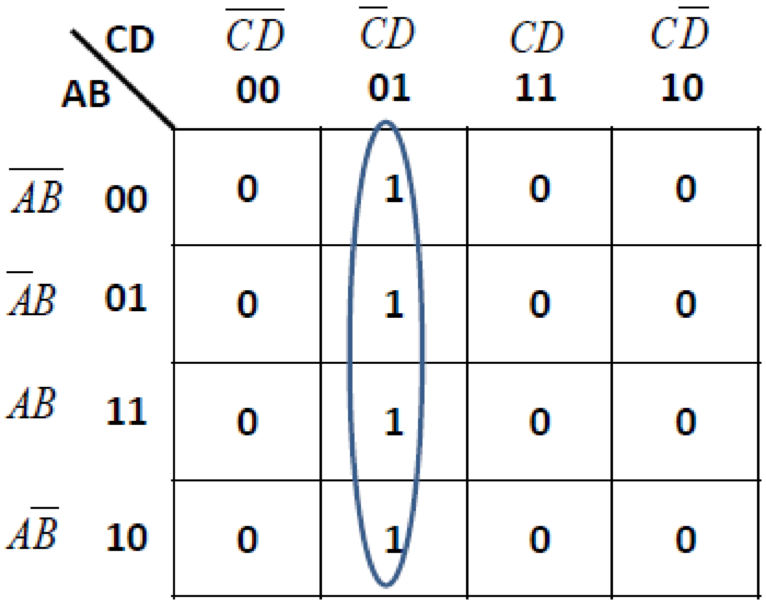
AB \ CD	$\overline{CD}$ 00	$\overline{C}D$ 01	CD 11	$C\overline{D}$ 10
$\overline{AB}$ 00	0	1	0	0
$\overline{A}B$ 01	0	0	0	0
AB 11	0	0	0	0
$A\overline{B}$ 10	0	1	0	0

Possible Grouping of Four Adjacent 1's : Quad

✓ *A Quad eliminates 2 variable*



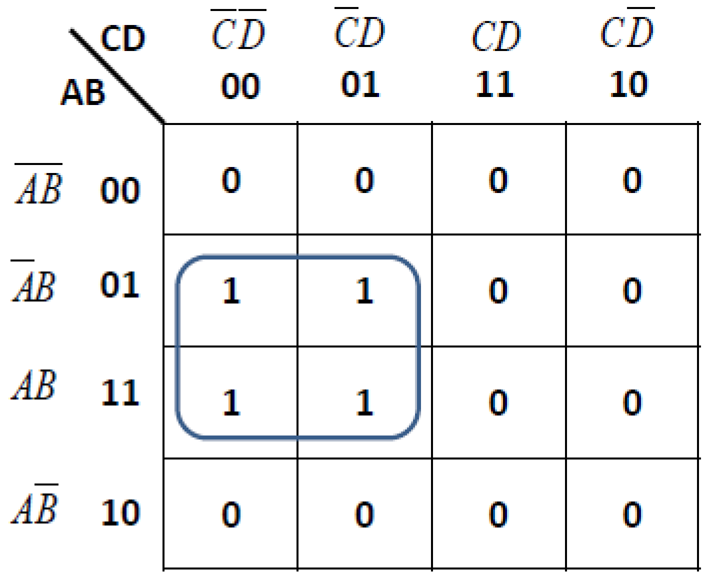
AB \ CD		$\bar{C}\bar{D}$ 00	$\bar{C}D$ 01	CD 11	$C\bar{D}$ 10
$\bar{A}\bar{B}$	00	0	0	0	0
$\bar{A}B$	01	0	0	0	0
AB	11	0	0	0	0
$A\bar{B}$	10	1	1	1	1



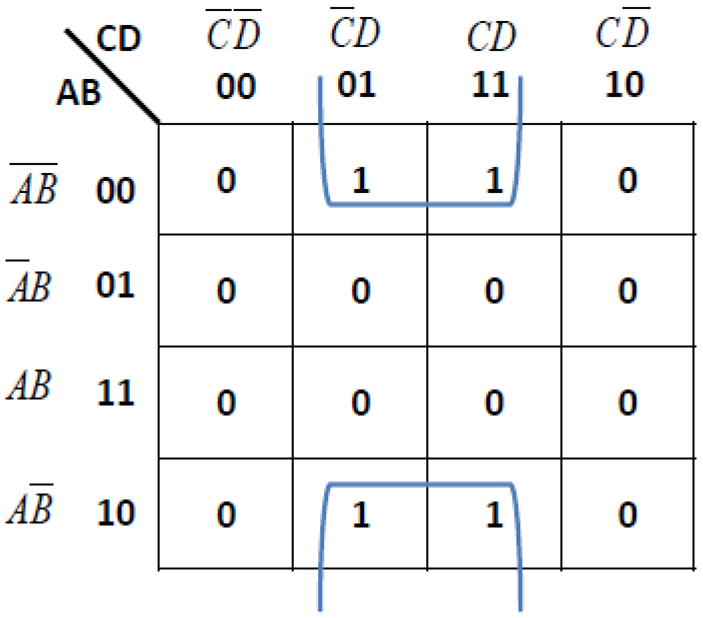
AB \ CD		$\bar{C}\bar{D}$ 00	$\bar{C}D$ 01	CD 11	$C\bar{D}$ 10
$\bar{A}\bar{B}$	00	0	1	0	0
$\bar{A}B$	01	0	1	0	0
AB	11	0	1	0	0
$A\bar{B}$	10	0	1	0	0

Possible Grouping of Four Adjacent 1’s : Quad

✓ *A Quad eliminates 2 variable*



AB \ CD	$\overline{C}\,\overline{D}$ 00	$\overline{C}D$ 01	CD 11	$C\overline{D}$ 10
$\overline{A}\,\overline{B}$ 00	0	0	0	0
$\overline{A}B$ 01	1	1	0	0
AB 11	1	1	0	0
$A\overline{B}$ 10	0	0	0	0



AB \ CD	$\overline{C}\,\overline{D}$ 00	$\overline{C}D$ 01	CD 11	$C\overline{D}$ 10
$\overline{A}\,\overline{B}$ 00	0	1	1	0
$\overline{A}B$ 01	0	0	0	0
AB 11	0	0	0	0
$A\overline{B}$ 10	0	1	1	0

Possible Grouping of Four Adjacent 1's : Quad

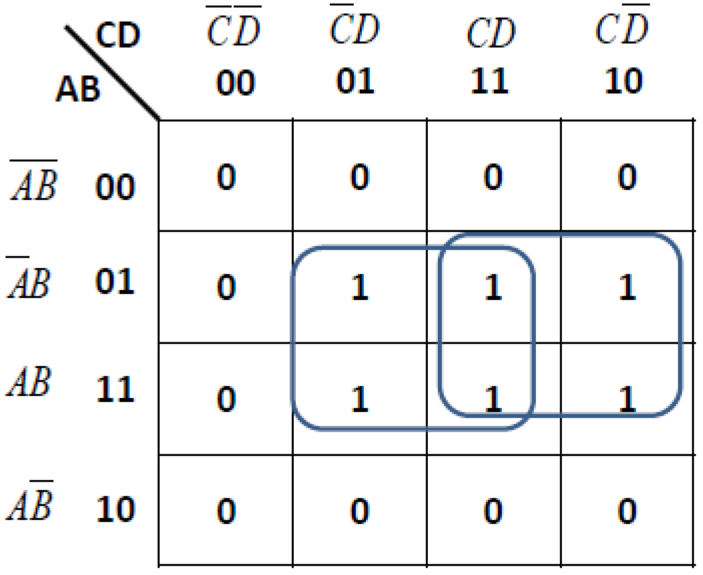
✓ *A Quad eliminates 2 variable*

AB \ CD	$\overline{C}\,\overline{D}$ 00	$\overline{C}D$ 01	CD 11	$C\overline{D}$ 10
$\overline{A}\,\overline{B}$ 00	1	0	0	1
$\overline{A}B$ 01	0	0	0	0
AB 11	0	0	0	0
$A\overline{B}$ 10	1	0	0	1

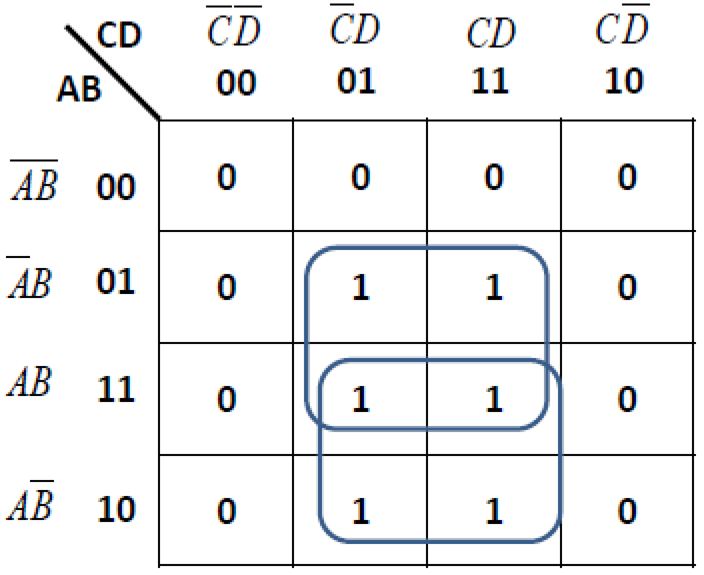
AB \ CD	$\overline{C}\,\overline{D}$ 00	$\overline{C}D$ 01	CD 11	$C\overline{D}$ 10
$\overline{A}\,\overline{B}$ 00	0	0	0	0
$\overline{A}B$ 01	1	0	0	1
AB 11	1	0	0	1
$A\overline{B}$ 10	0	0	0	0

Possible Grouping of Four Adjacent 1's : Quad

✓ *A Quad eliminates 2 variable*



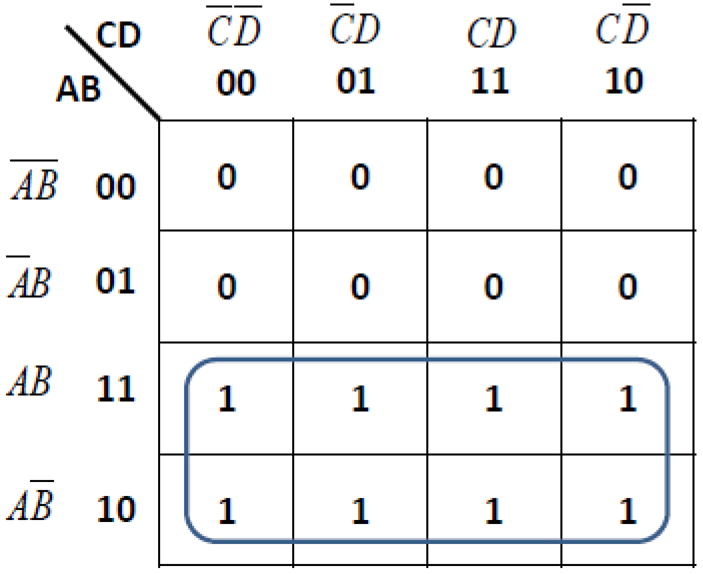
AB \ CD	$\overline{C}\,\overline{D}$ 00	$\overline{C}D$ 01	CD 11	$C\overline{D}$ 10
$\overline{A}\,\overline{B}$ 00	0	0	0	0
$\overline{A}B$ 01	0	1	1	1
AB 11	0	1	1	1
$A\overline{B}$ 10	0	0	0	0



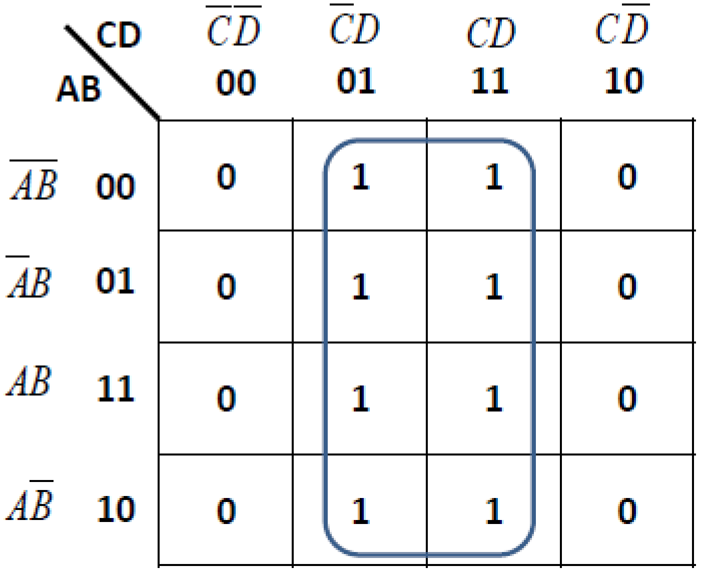
AB \ CD	$\overline{C}\,\overline{D}$ 00	$\overline{C}D$ 01	CD 11	$C\overline{D}$ 10
$\overline{A}\,\overline{B}$ 00	0	0	0	0
$\overline{A}B$ 01	0	1	1	0
AB 11	0	1	1	0
$A\overline{B}$ 10	0	1	1	0

Possible Grouping of Eight Adjacent 1's : Octet

✓ *A Octet eliminates 3 variable*

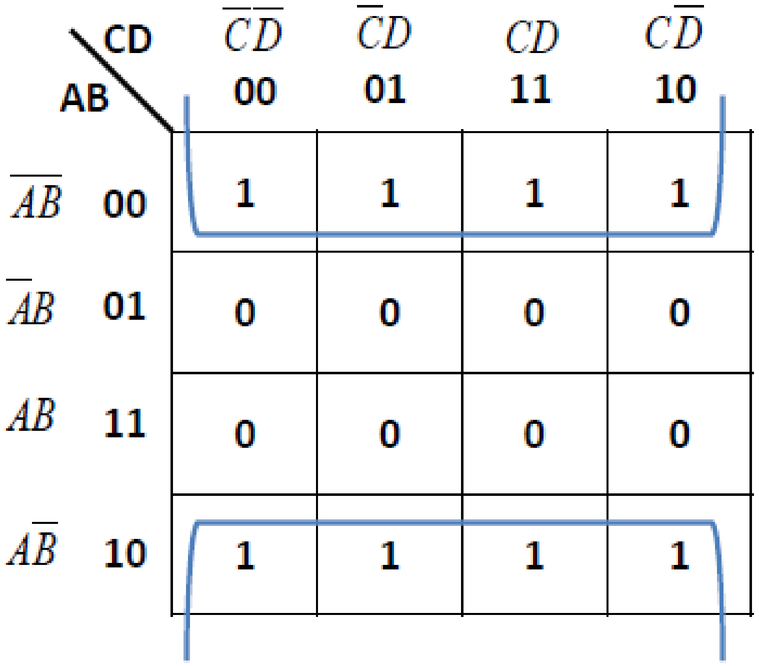


AB \ CD	$\overline{C}\,\overline{D}$ 00	$\overline{C}D$ 01	CD 11	$C\overline{D}$ 10
$\overline{A}\,\overline{B}$ 00	0	0	0	0
$\overline{A}B$ 01	0	0	0	0
AB 11	1	1	1	1
$A\overline{B}$ 10	1	1	1	1

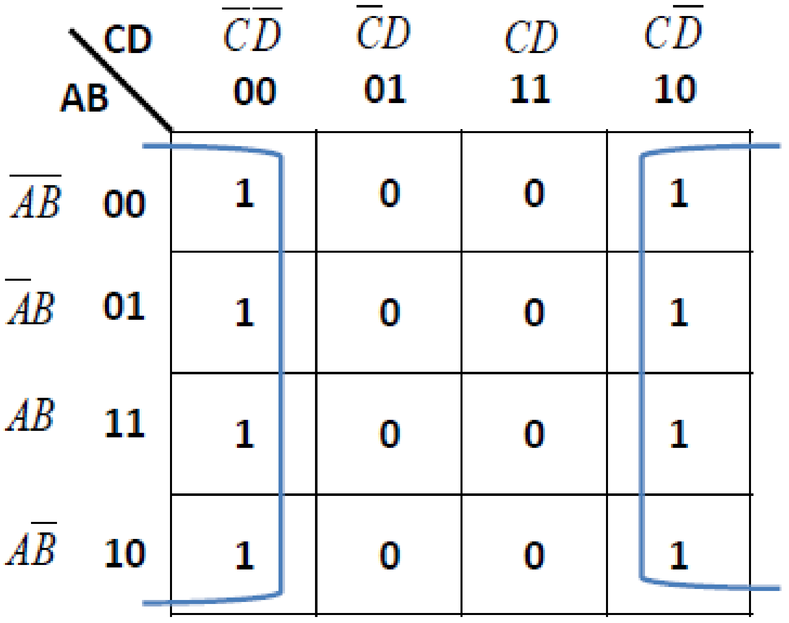


AB \ CD	$\overline{C}\,\overline{D}$ 00	$\overline{C}D$ 01	CD 11	$C\overline{D}$ 10
$\overline{A}\,\overline{B}$ 00	0	1	1	0
$\overline{A}B$ 01	0	1	1	0
AB 11	0	1	1	0
$A\overline{B}$ 10	0	1	1	0

✓ *A Octet eliminates 3 variable*



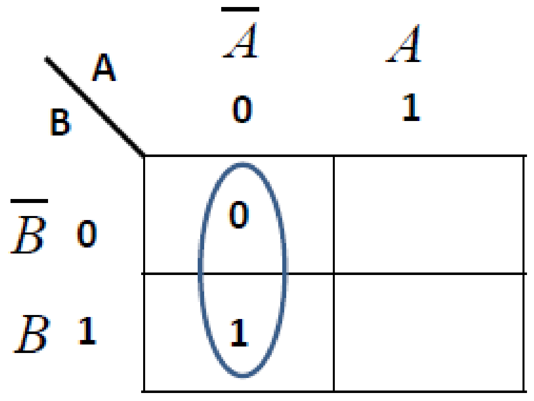
CD / AB	$\overline{C}\overline{D}$ 00	$\overline{C}D$ 01	CD 11	$C\overline{D}$ 10
$\overline{A}\overline{B}$ 00	1	1	1	1
$\overline{A}B$ 01	0	0	0	0
AB 11	0	0	0	0
$A\overline{B}$ 10	1	1	1	1



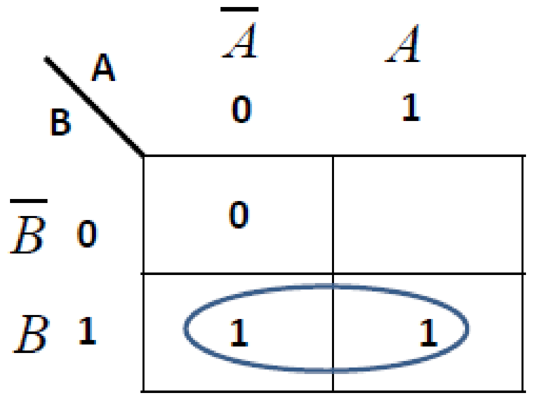
CD / AB	$\overline{C}\overline{D}$ 00	$\overline{C}D$ 01	CD 11	$C\overline{D}$ 10
$\overline{A}\overline{B}$ 00	1	0	0	1
$\overline{A}B$ 01	1	0	0	1
AB 11	1	0	0	1
$A\overline{B}$ 10	1	0	0	1

Rule for K MAP Simplification

1. ***Groups may not include any cell containing a zero.***

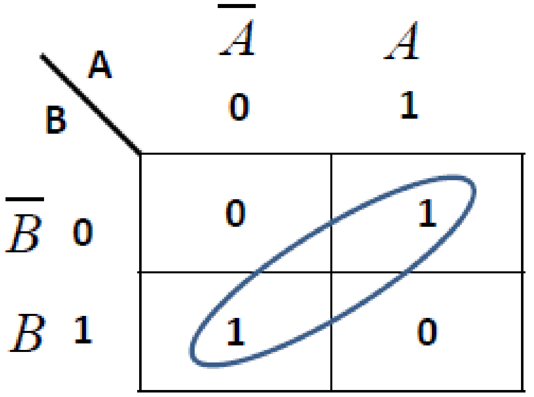


Not Accepted

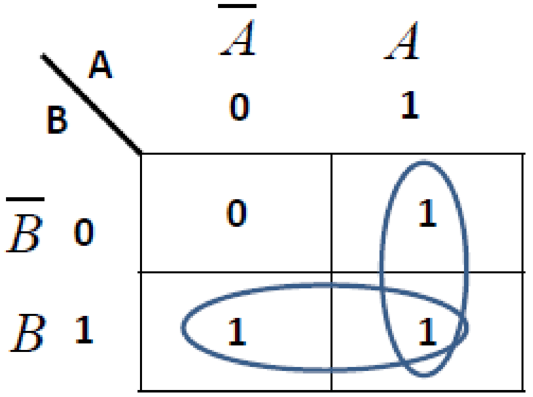


Accepted

2. Groups may be horizontal or vertical, but may not be diagonal

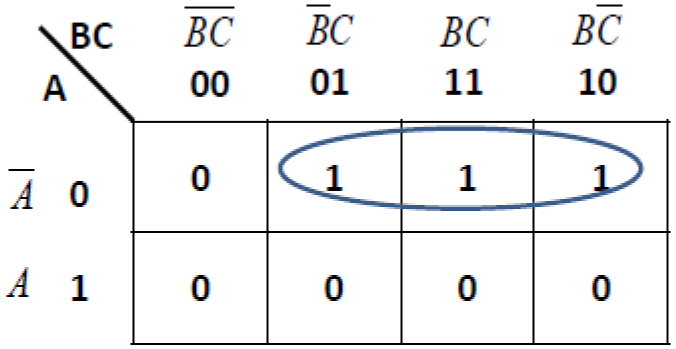


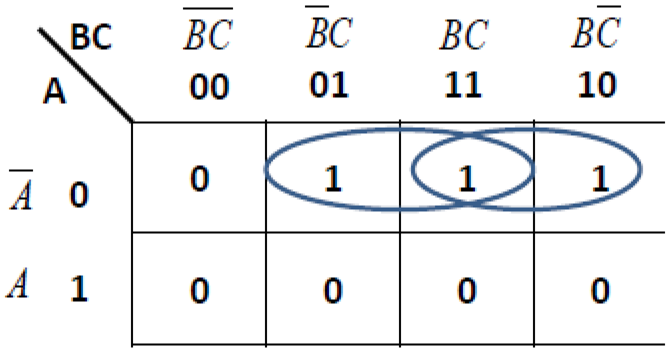
Not Accepted

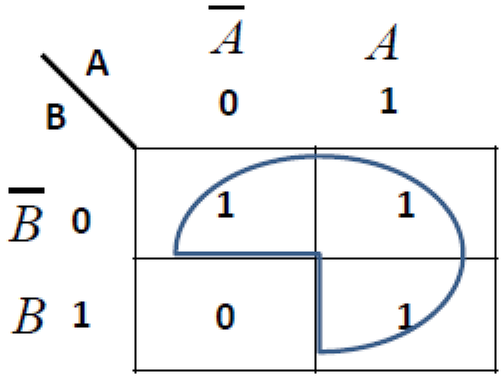


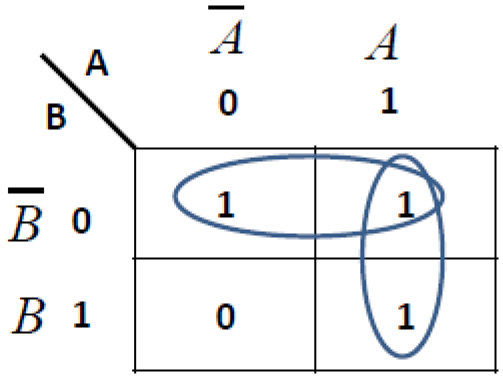
Accepted

3. Groups must contain 1,2,4,8 or in general 2^n cells





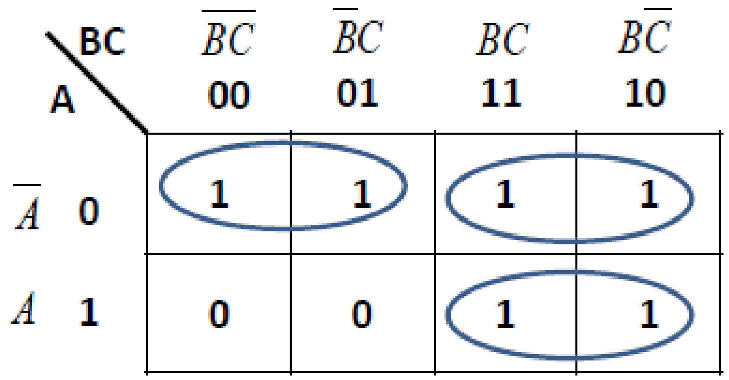




Not Accepted

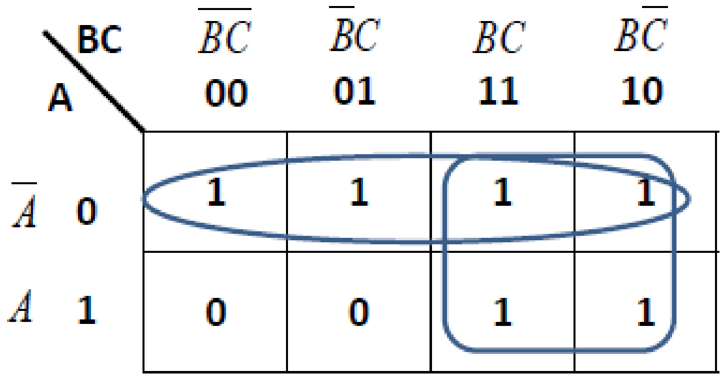
Accepted

4. Each group should be as large as possible



A \ BC	$\overline{B}\overline{C}$ 00	$\overline{B}C$ 01	BC 11	$B\overline{C}$ 10
$\overline{A}$ 0	1	1	1	1
A 1	0	0	1	1

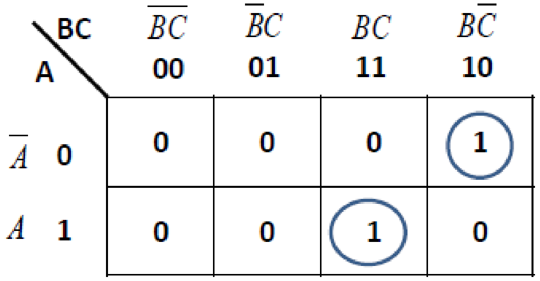
Not Accepted



A \ BC	$\overline{B}\overline{C}$ 00	$\overline{B}C$ 01	BC 11	$B\overline{C}$ 10
$\overline{A}$ 0	1	1	1	1
A 1	0	0	1	1

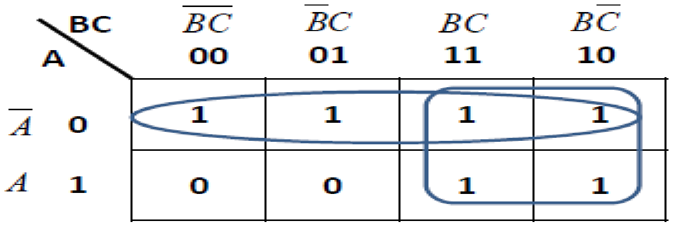
Accepted

5. Each cell containing a one must be in at least one group



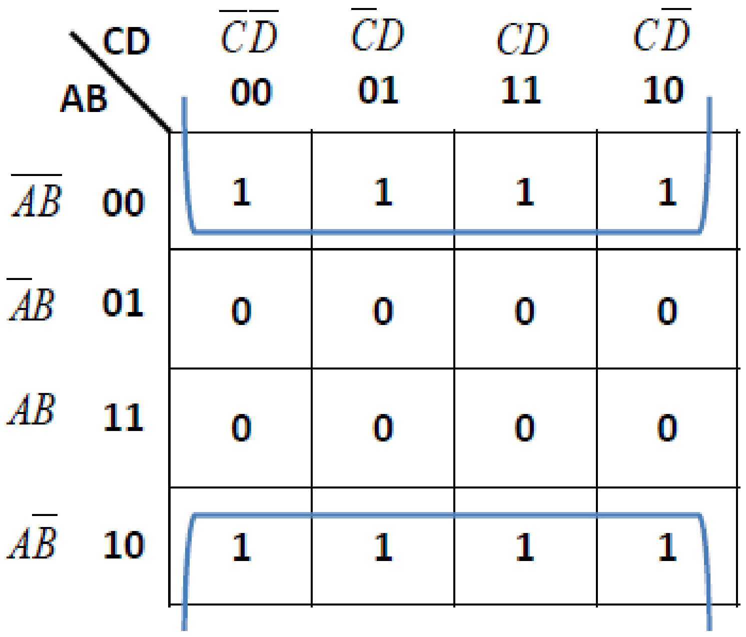
A \ BC	$\overline{BC}$ 00	$\overline{B}C$ 01	BC 11	$B\overline{C}$ 10
$\overline{A}$ 0	0	0	0	1
A 1	0	0	1	0

6. Groups may be overlap

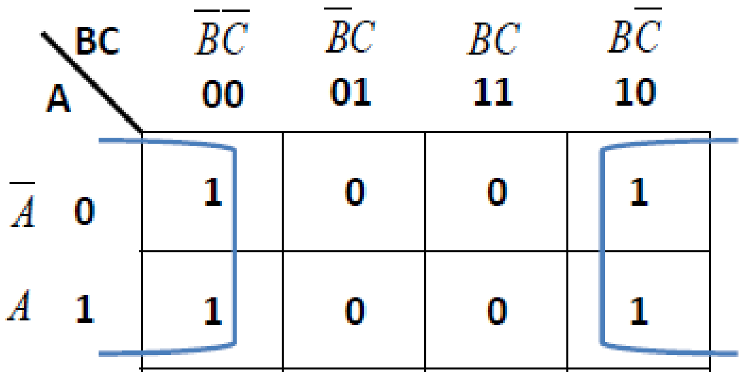


A \ BC	$\overline{BC}$ 00	$\overline{B}C$ 01	BC 11	$B\overline{C}$ 10
$\overline{A}$ 0	1	1	1	1
A 1	0	0	1	1

7. Groups may wrap around the table. The leftmost cell in a row may be grouped with rightmost cell and the top cell in a column may be grouped with bottom cell

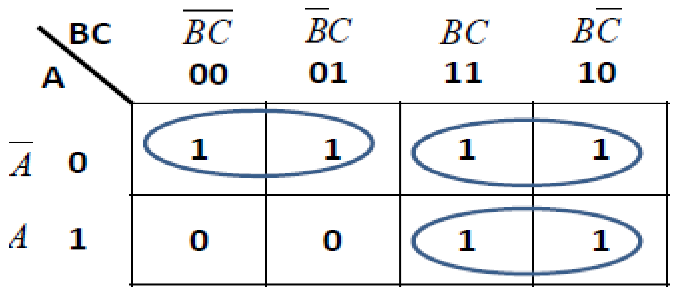


AB \ CD		$\overline{C}\,\overline{D}$ 00	$\overline{C}D$ 01	CD 11	$C\overline{D}$ 10
$\overline{A}\,\overline{B}$	00	1	1	1	1
$\overline{A}B$	01	0	0	0	0
AB	11	0	0	0	0
$A\overline{B}$	10	1	1	1	1

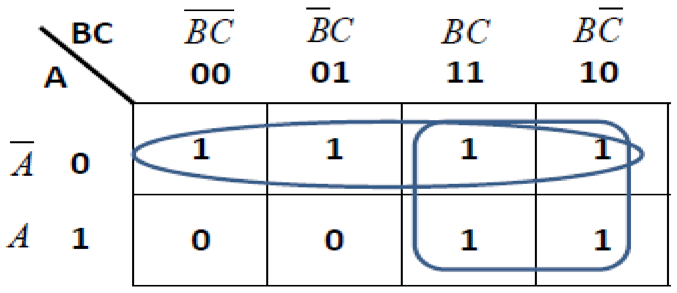


A \ BC		$\overline{B}\,\overline{C}$ 00	$\overline{B}C$ 01	BC 11	$B\overline{C}$ 10
$\overline{A}$	0	1	0	0	1
A	1	1	0	0	1

8. There should be as few groups as possible, as long as this does not contradict any of the previous rules.



Not Accepted



Accepted

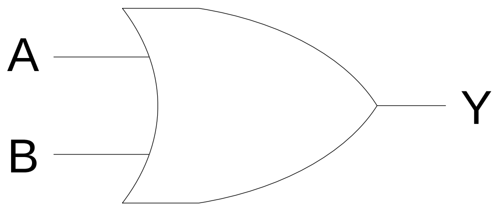
9. A pair eliminates one variable.

10. A Quad eliminates two variables.

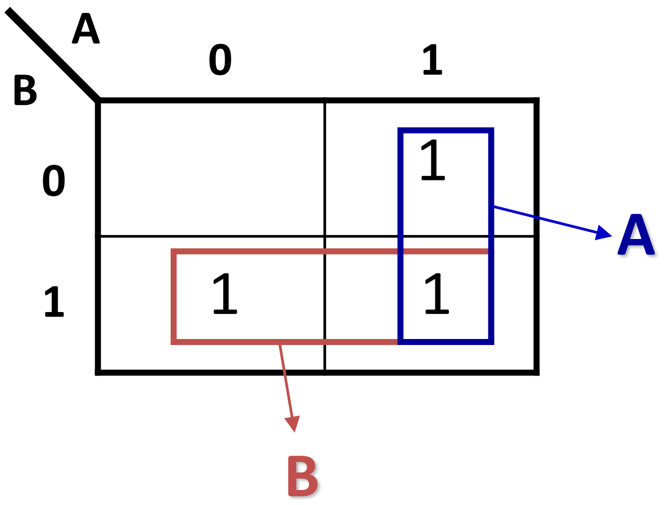
11. A octet eliminates three variables

Example 1

. The map for a 2-input OR gate looks like this:



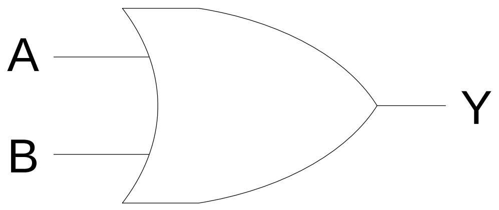
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1



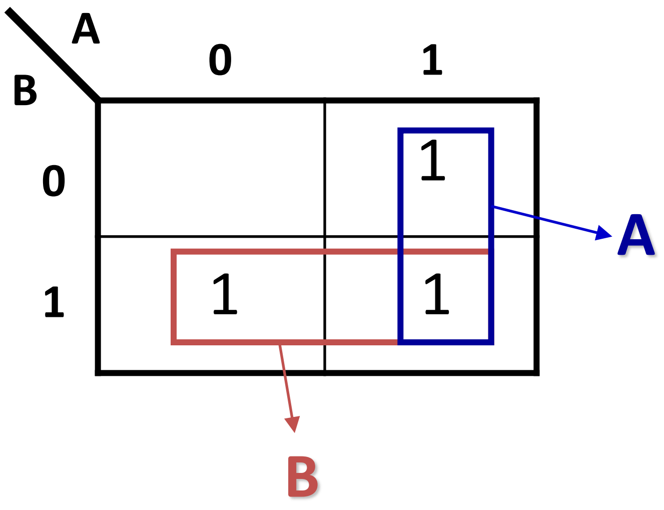
A+B

Example 1

. The map for a 2-input OR gate looks like this:



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

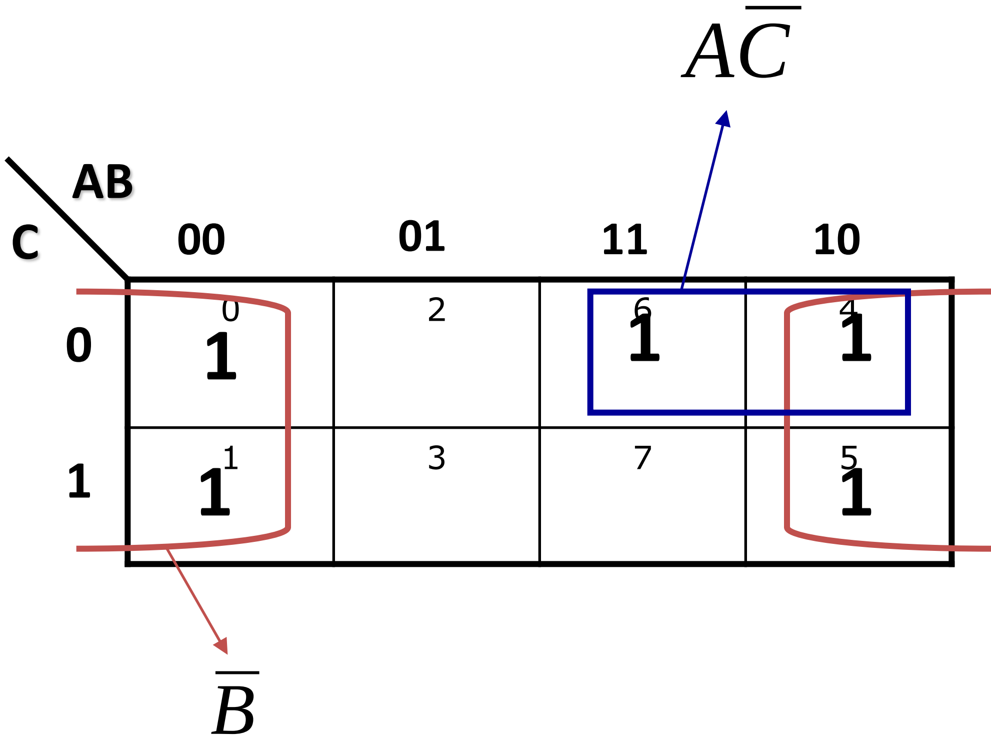


$\boxed{A+B}$

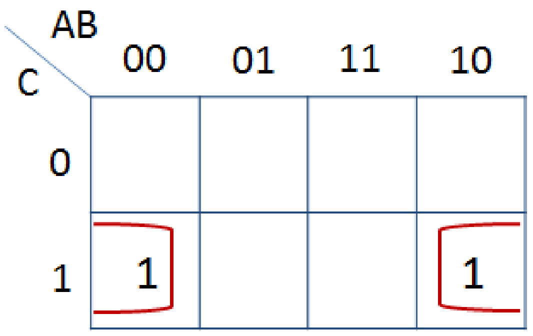
Example 2

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

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

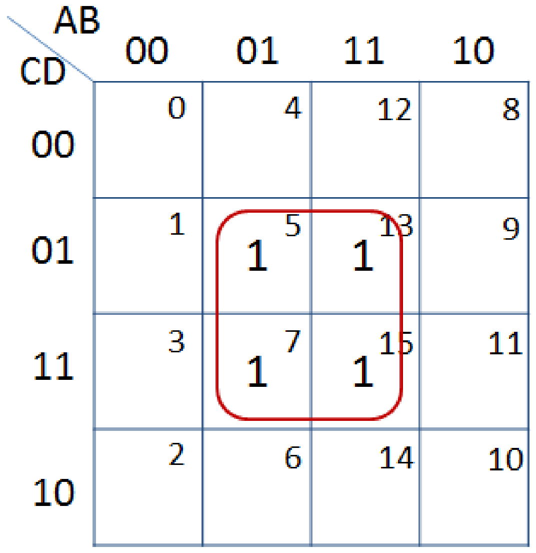


$f = AB'C + A'B'C$



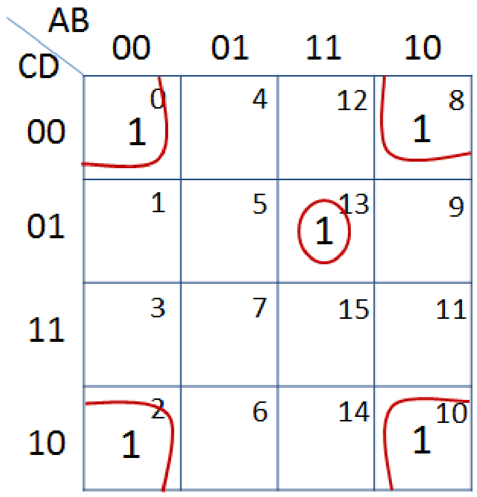
$f = B'C$

$f = ABC'D + ABCD + A'BC'D + A'BCD$



$f = BD$

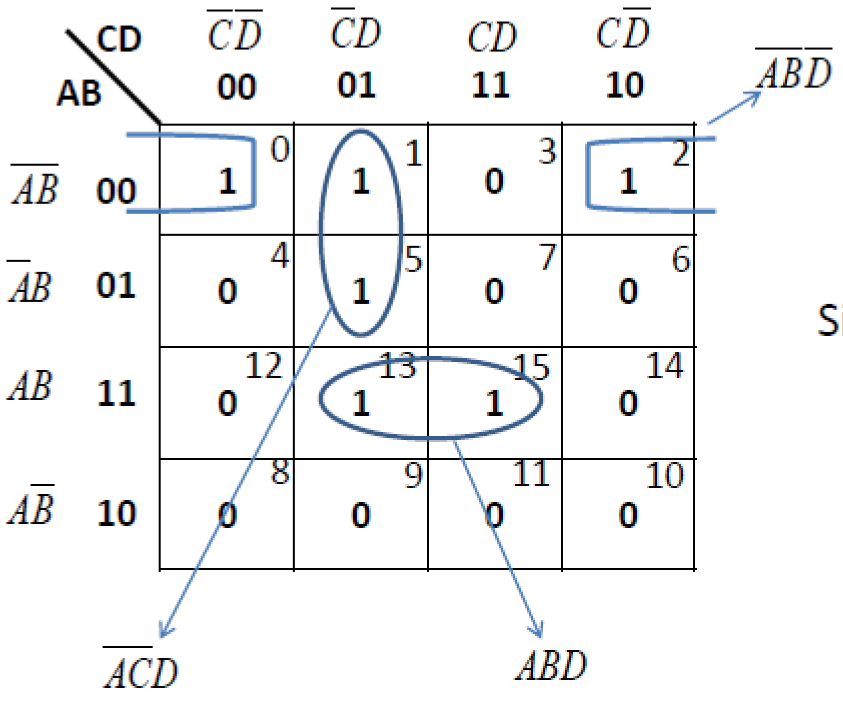
$f = ABC'D + AB'C'D' + AB'CD' + A'B'CD' + A'B'C'D'$



$f = ABC'D + B'D'$

Example 3

$$Y = \Sigma m(0,1,2,5,13,15)$$



CD
AB
$\overline{C}\overline{D}$ 00
$\overline{C}D$ 01
CD 11
$C\overline{D}$ 10
$\overline{A}\overline{B}$ 00
$\overline{A}B$ 01
AB 11
$A\overline{B}$ 10
$\overline{A}\overline{B}\overline{D}$
$\overline{A}\overline{C}D$
ABD

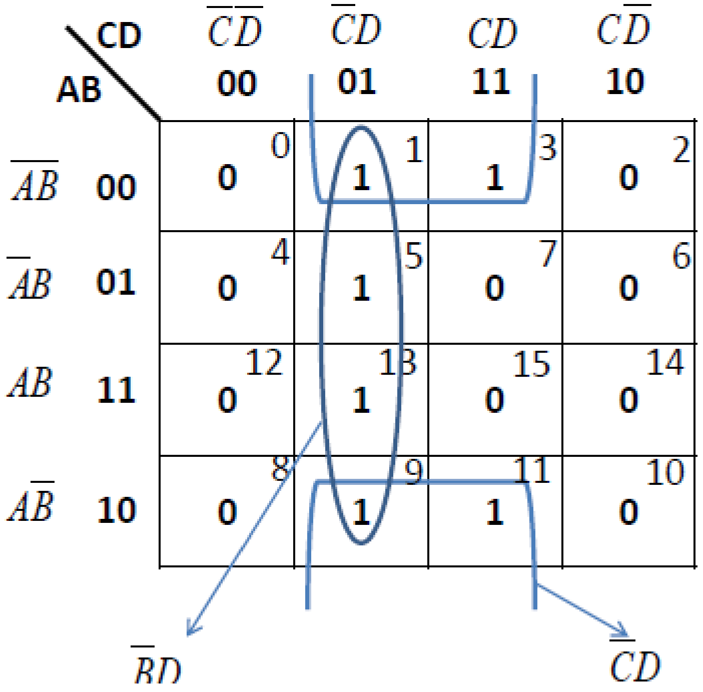
	00	01	11	10
00	1 (0)	1 (1)	0 (3)	1 (2)
01	0 (4)	1 (5)	0 (7)	0 (6)
11	0 (12)	1 (13)	1 (15)	0 (14)
10	0 (8)	0 (9)	0 (11)	0 (10)

Simplified Boolean expression

$$Y = \overline{A}\,\overline{B}\,\overline{D} + \overline{A}\,\overline{C}D + ABD$$

Example 4

$$f(A,B,C,D) = \Sigma m(1,3,5,9,11,13)$$



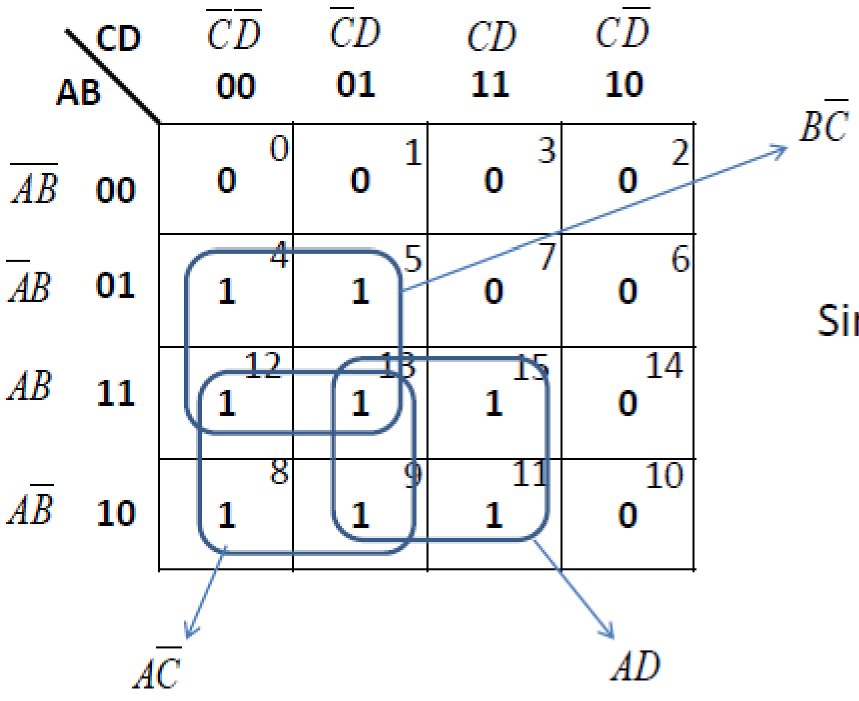
Simplified Boolean expression

$$f = \overline{B}D + \overline{C}D$$

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

Example 5

$$f(A,B,C,D) = \Sigma m(4,5,8,9,11,12,13,15)$$



AB \ CD	$\bar{C}\bar{D}$ 00	$\bar{C}D$ 01	CD 11	$C\bar{D}$ 10
$\bar{A}\bar{B}$ 00	0 (0)	0 (1)	0 (3)	0 (2)
$\bar{A}B$ 01	1 (4)	1 (5)	0 (7)	0 (6)
AB 11	1 (12)	1 (13)	1 (15)	0 (14)
$A\bar{B}$ 10	1 (8)	1 (9)	1 (11)	0 (10)

Simplified Boolean expression

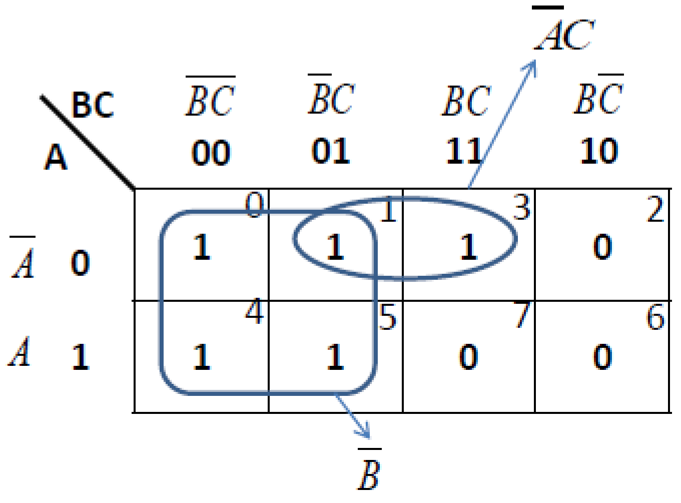
$$f = B\bar{C} + A\bar{C} + AD$$

Exercise

Solve the following expression with K-maps;

1. $f_1(A,B,C) = \Sigma m(0,1,3,4,5)$
2. $f_2(A,B,C) = \Sigma m(0,1,2,3,6,7)$

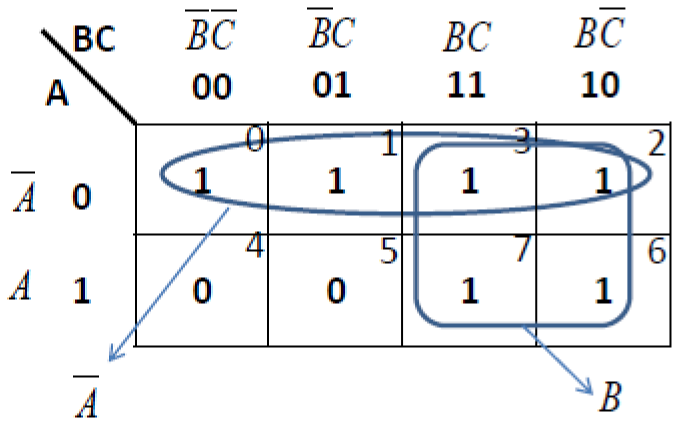
$$f_1(A,B,C)=\Sigma m(0,1,3,4,5)$$



Simplified Boolean expression

$$f_1=\overline{A}C+\overline{B}$$

$$f_2(A,B,C)=\Sigma m(0,1,2,3,6,7)$$

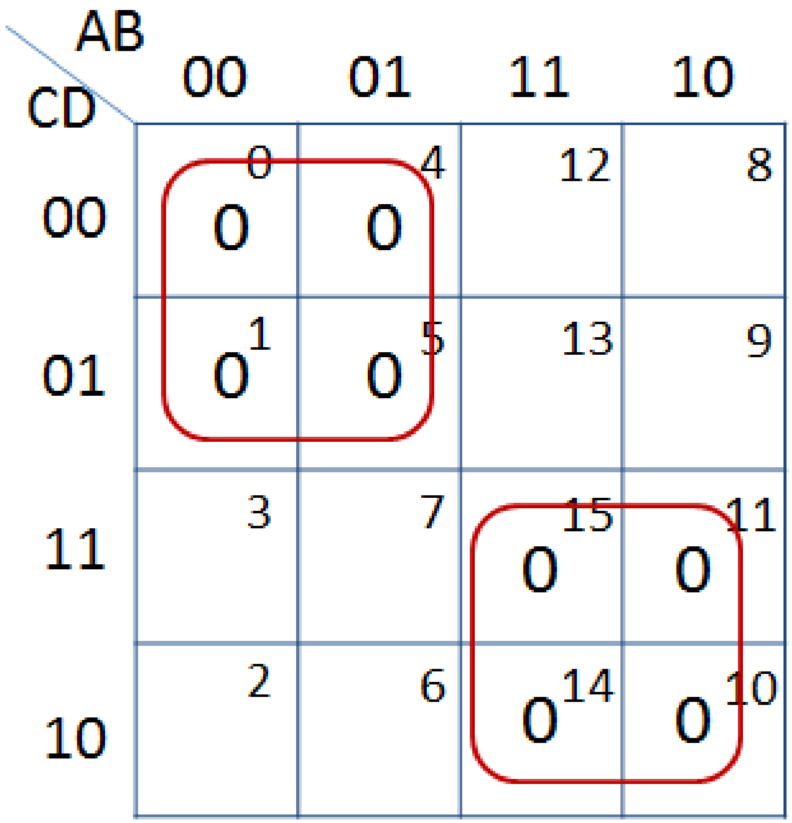


Simplified Boolean expression

$$f_2=\overline{A}+B$$

Example 6

$f(A,B,C,D) = \prod_M (0,1,4,5,10,11,14,15)$



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

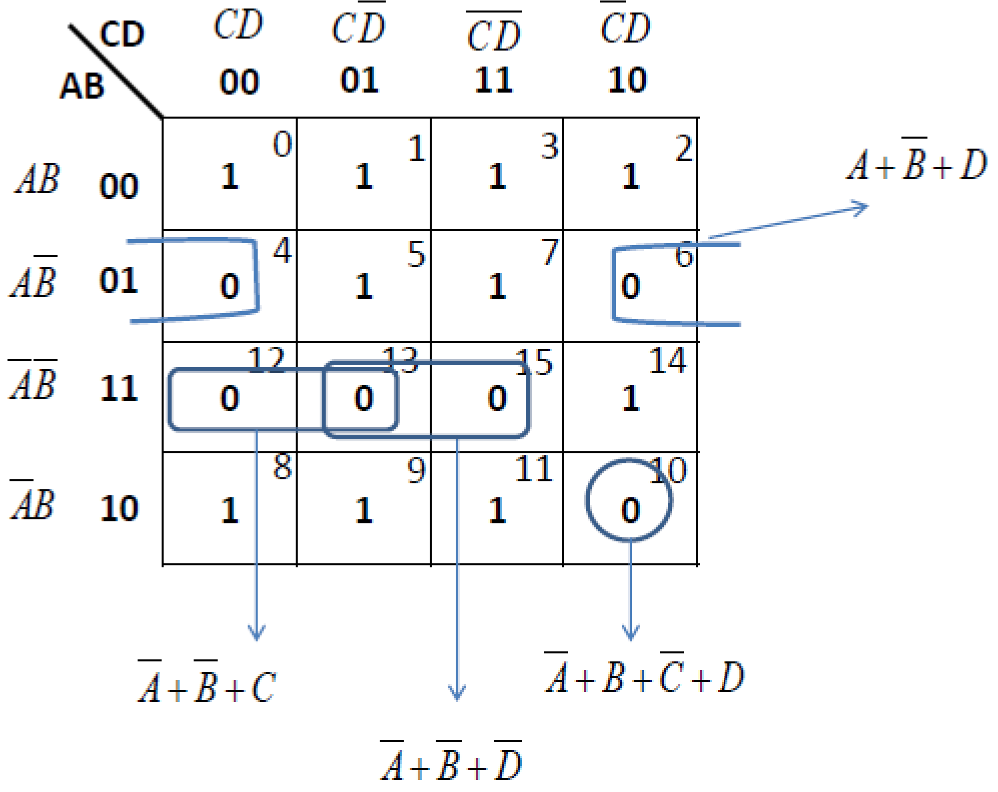
$$f = (A + C)(A' + C')$$

Exercise

Simplify ;

$$f(A,B,C,D) = \Pi M(4,6,10,12,13,15)$$

$$f(A,B,C,D) = \Pi M(4,6,10,12,13,15)$$



Simplified Boolean expression

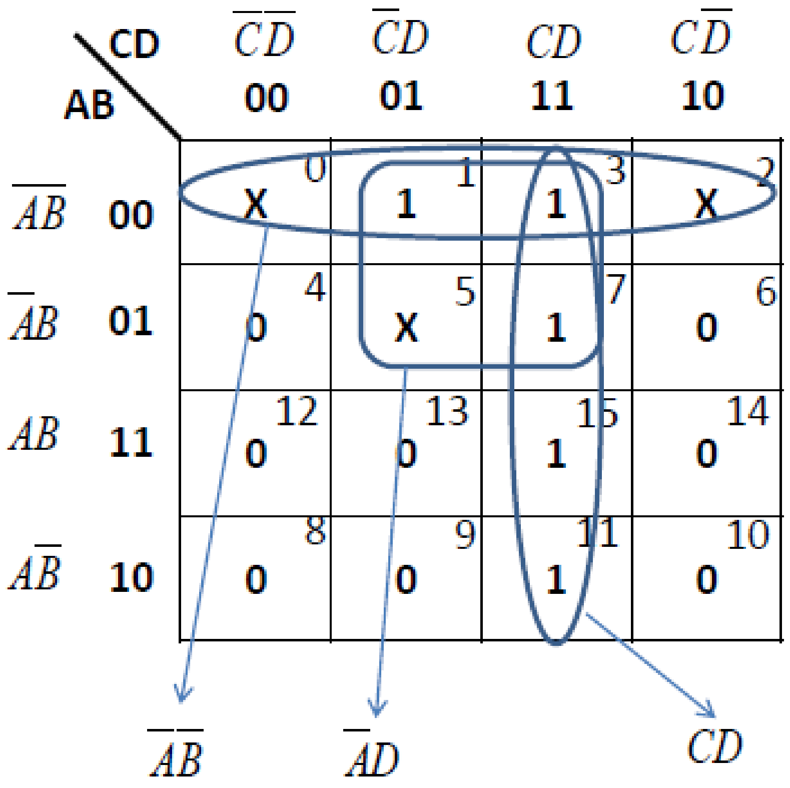
$$f = (\overline{A}+B+\overline{C}+D)(A+\overline{B}+D)(\overline{A}+\overline{B}+\overline{D})(\overline{A}+\overline{B}+C)$$

Don’t care conditions

- For SOP form we enter 1’s corresponding to the combinations of input variables which produce a high output and we enter 0’s in the remaining cells of the K-map.
- For POS form we enter 0’s corresponding to the combinations of input variables which produce a high output and we enter 1’s in the remaining cells of the K-map.
- But it is not always true that the cells not containing 1’s (in SOP) will contain 0’s, because some combinations of input variable do not occur.
- Also for some functions the outputs corresponding to certain combinations of input variables do not matter.
- In such situations we have a freedom to assume a 0 or 1 as output for each of these combinations.
- These conditions are known as the “Don’t Care Conditions” and in the K-map it is represented as ‘X’, in the corresponding cell.
- The don’t care conditions may be assumed to be 0 or 1 as per the need for simplification

K-map and don't care conditions - Example

$$f(A,B,C,D) = \Sigma m(1,3,7,11,15) + d(0,2,5)$$

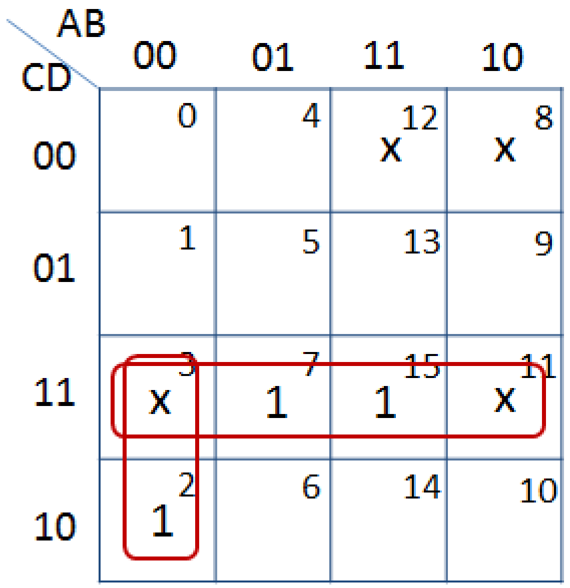


Simplified Boolean expression

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

example

$f(A,B,C,D) = \sum_m (2,7,15) + d(3,8,11,12)$



$f = CD + A'B'C$

Variable Entered Map(VEM)

- It can be used to plot an n- variable on n-1 variable map.
- Possible to reduce the map dimension by two or three in some cases.
- Advantage of using VEM occurs in design problems involving multiplexers.

Plotting VEM

- Map-entered variable for 3 variable function
- Consider the function

$$X = A'B'C' + ABC' + AB'C' + ABC$$

- Considering value of X to be function of map location and the variable C and plotting 2 variable map.

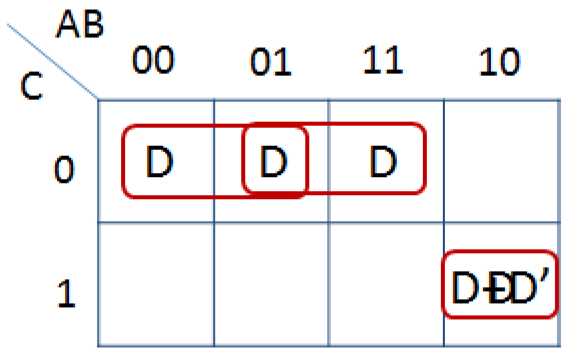
B \ A	0	1
0	X=1 if C′=1	X=1 if C′=1
1	X=0	X=1 if C′=1 or if C = 1

B \ A	0	1
0	C′	C′
1	0	C + C′

Reducing expression with VEM

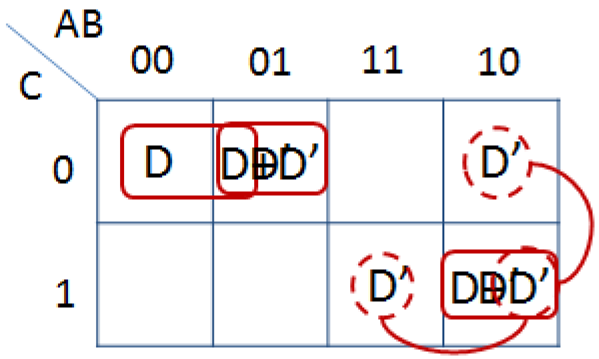
- Choose group of similar terms to cover all nonzero terms appearing in the map except "don't care" terms.
- Rest complete process is similar to k-map.

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



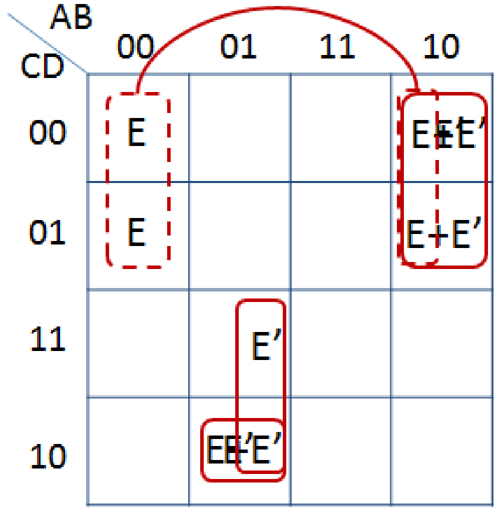
$$f = A'C'D + BC'D + AB'C$$

$f = A'B'C'D + A'BC'D' + A'BC'D + AB'C'D' + AB'CD' + AB'CD + ABCD'$



$f = A'C'D \;+ A'BC' + ACD' + AB'D' + AB'C$

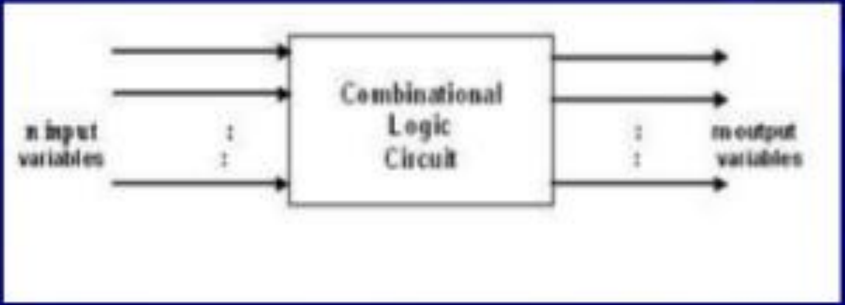
$f = A'B'C'D'E + A'B'C'DE + A'BCD'E' + A'BCD'E + AB'C'D'E' + AB'C'D'E + AB'C'D + A'BCDE'$



$f = B'C'E + AB'C' + A'BCD' + A'BCE'$

Combinational logic circuit

- Combinational circuits consists of logic gates whose outputs depends on the present inputs .They have no memory element .
- It consists of input variables , logic gates & output variables .



Multiplexer

- A multiplexer(MUX) is a device that allows digital information from several sources to be routed onto a single line for transmission over that line to a common destination.
- Consider an integer 'm', which is constrained by the following relation:

  $m = 2^n$, where m and n are both integers.

- A **m-to-1** Multiplexer has
  - m Inputs: $I_0, I_1, I_2, \ldots\ldots\ldots\ldots\ldots I_{(m-1)}$
  - One Output: Y
  - n Control inputs: $S_0, S_1, S_2, \ldots\ldots S_{(n-1)}$
  - One (or more) Enable input(s)

  Such that Y may be equal to one of the inputs, depending upon the control inputs.

N:1 multiplexer

- It is a logic circuit that accept several data i/p & allows only one of them at a time to get through to o/p.
- It means sharing..it occurs when several pheripheral devices share a single T.L. Or Bus to communicate with computer**.**
- **Types of multiplexer:**
- 1) 2:1 multiplexer, 2)4:1 multiplexer
- 3)8:1 multiplexer, 2)16:1 multiplexer
- It improve the reliability of the digital systems because it reduces it the no. of external wired connection.
- It act like a digital controlled single pole, multiple way switch

- Multiplexer is a circuit which has a number of inputs but only one output.
- Multiplexer is a circuit which transmits large number of information signals over a single line.
- Multiplexer is also known as “Data Selector” or MUX.

Necessity of Multiplexers

- In most of the electronic systems, the digital data is available on more than one lines. It is necessary to route this data over a single line.
- Under such circumstances we require a circuit which select one of the many inputs at a time.
- This circuit is nothing but a multiplexer. Which has many inputs, one output and some select lines.
- Multiplexer improves the reliability of the digital system because it reduces the number of external wired connections

Block diagram of multiplxer

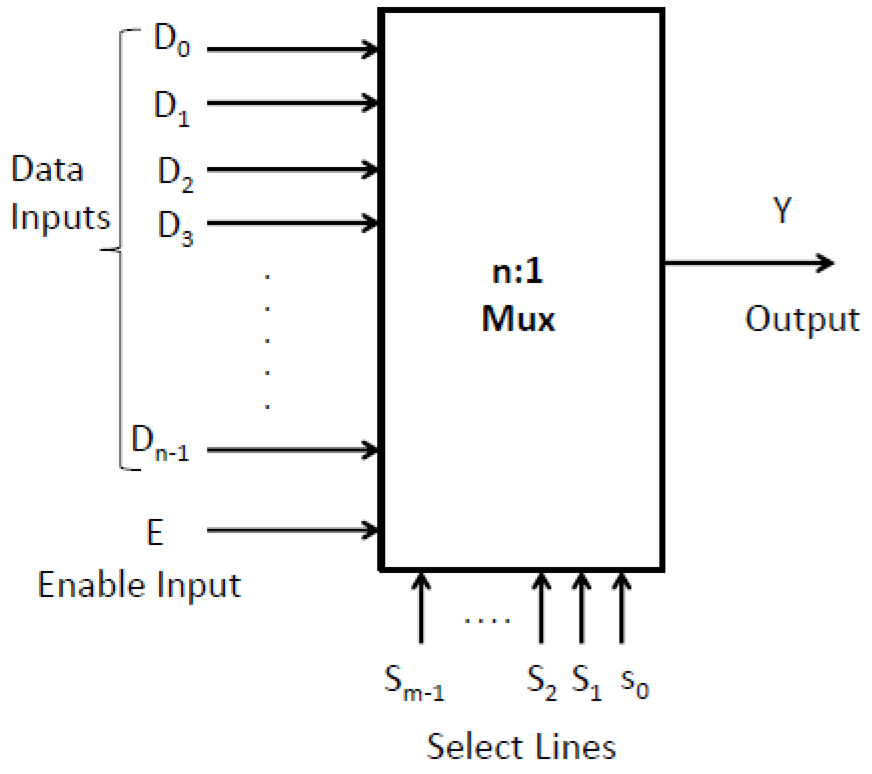


Fig. General Block Diagram

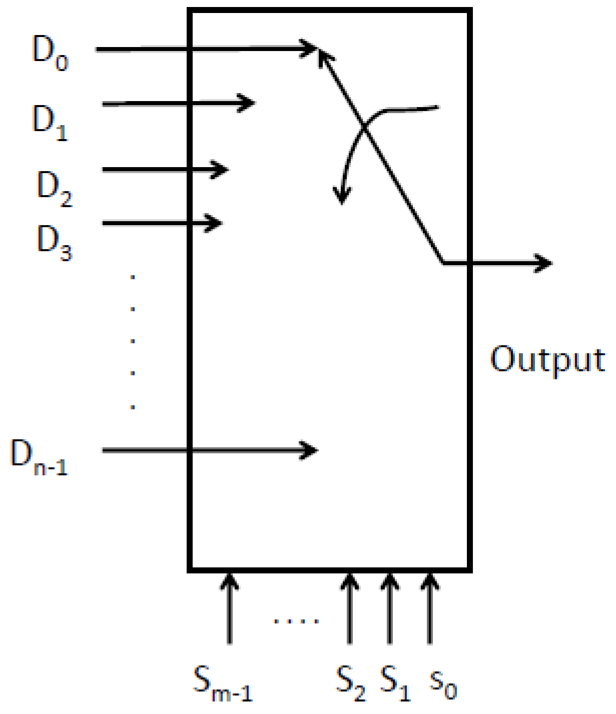
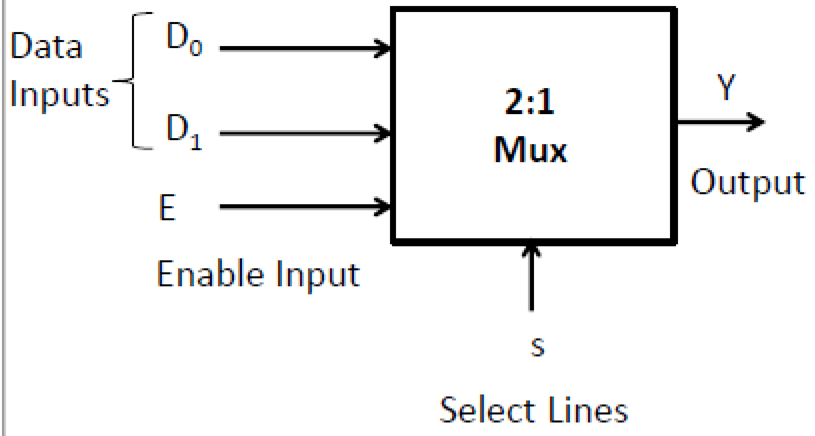


Fig. Equivalent Circuit

2:1 Multiplexer

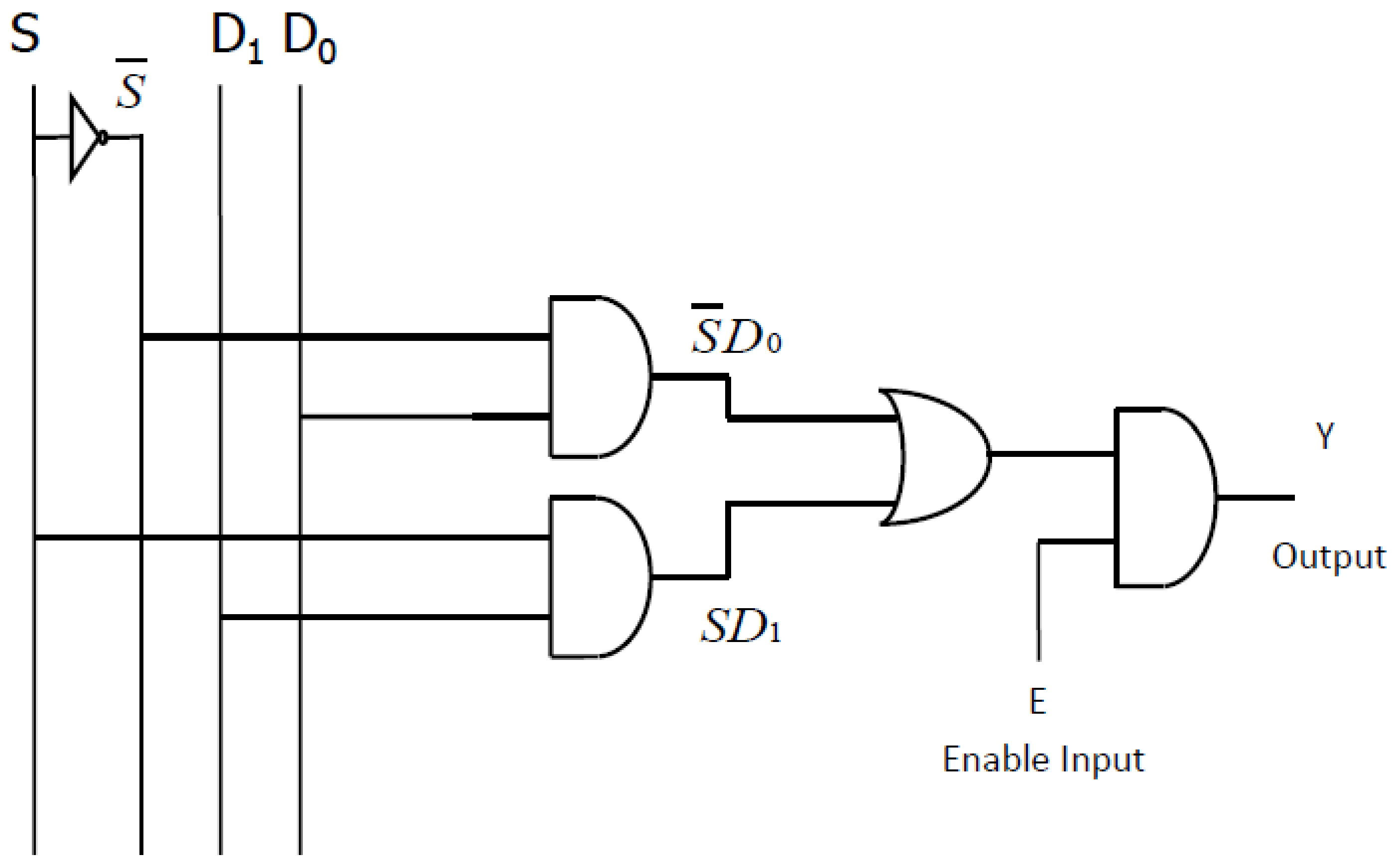


Block Diagram

Truth Table

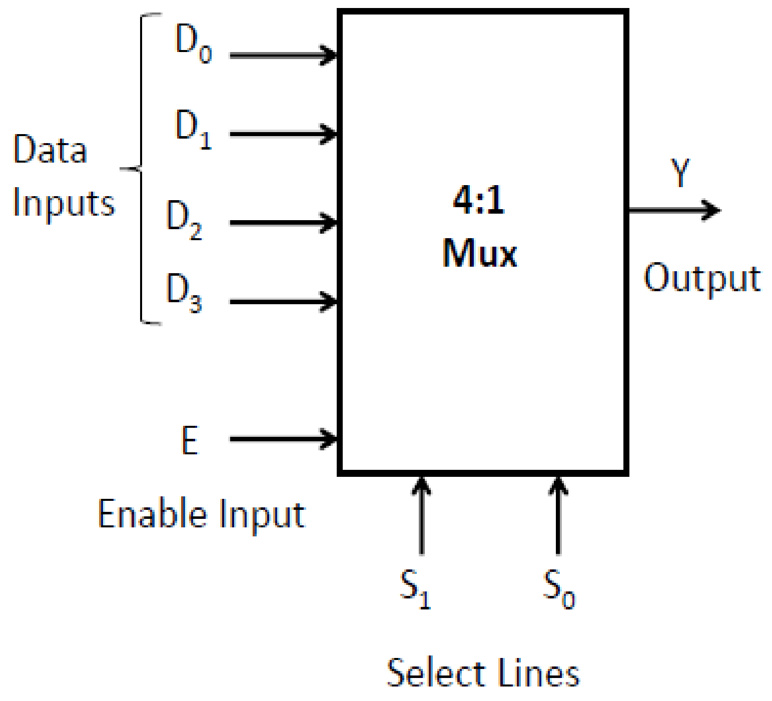
Enable i/p (E)	**Select i/p (S)**	**Output (Y)**
0	X	0
1	0	D_0
1	1	D_1

Logic diagram of 2:1 multiplexer



4:1 Multiplexer

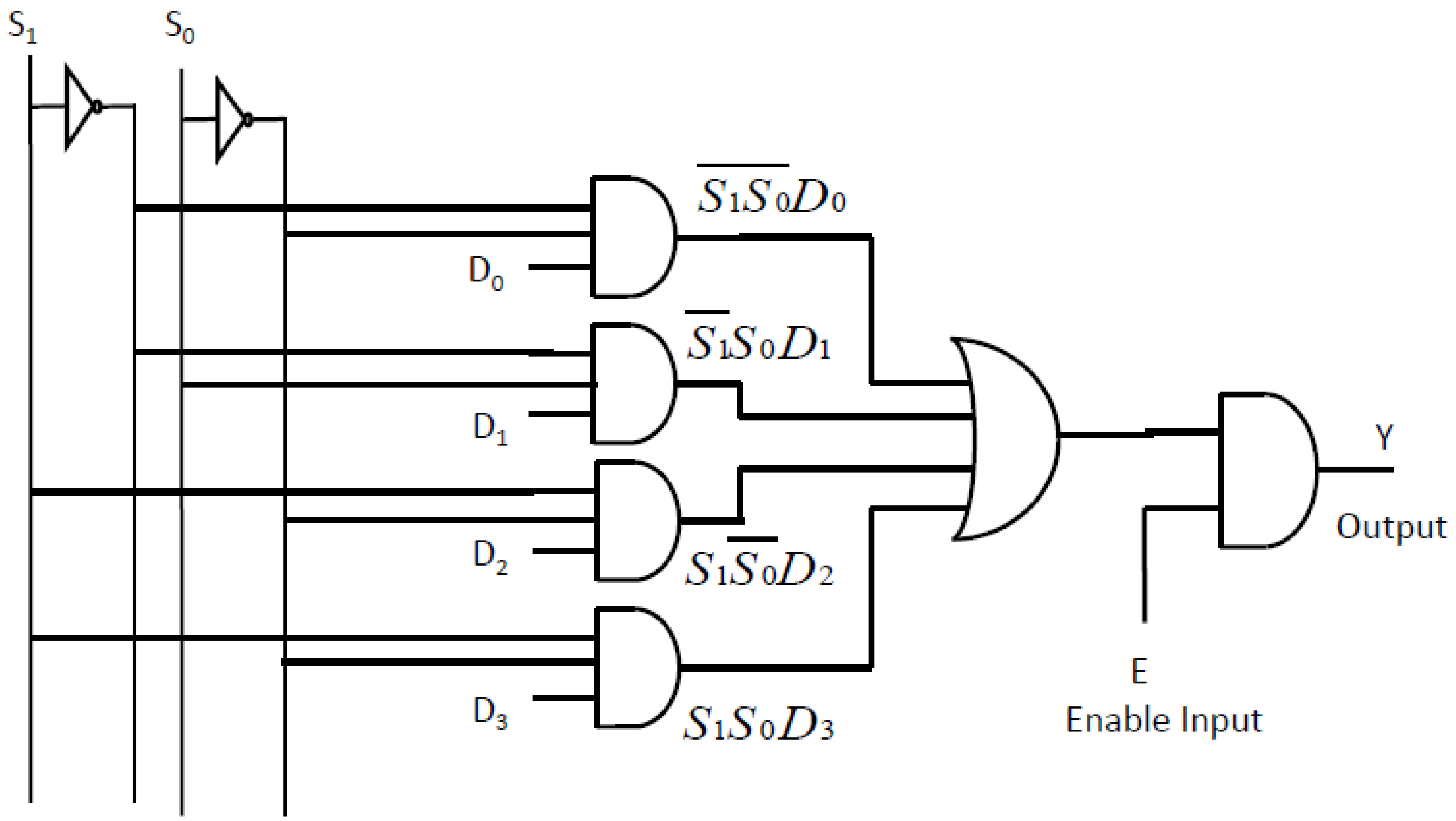
Block Diagram



Truth Table

Enable i/p	Select i/p		Output
E	**S_1**	**S_0**	**Y**
0	X	X	0
1	0	0	D_0
1	0	1	D_1
1	1	0	D_2
1	1	1	D_3

Realization of 4:1 Mux using gates



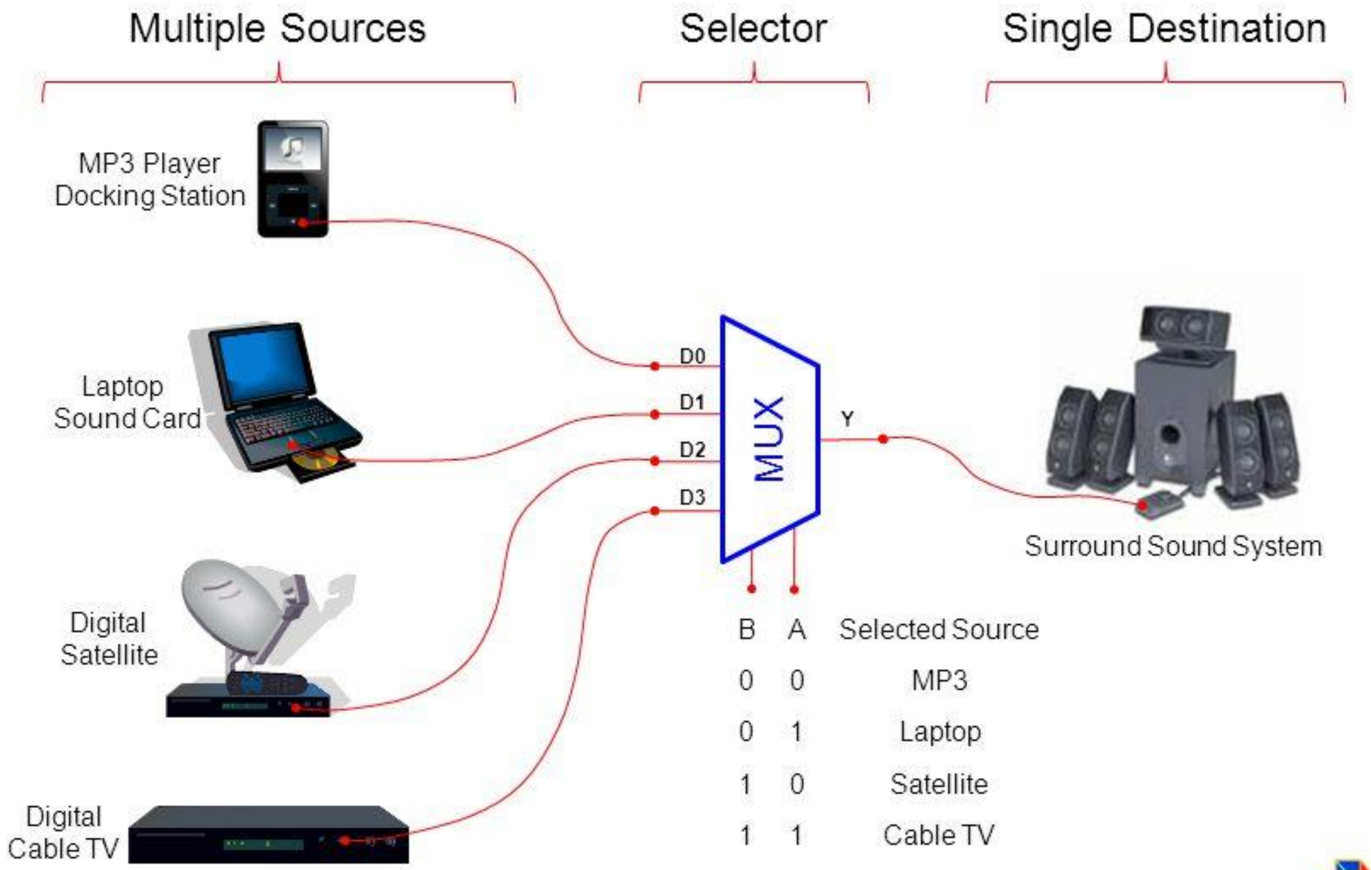
Advantage of multiplexer

- It reduces the number of wires.
- So it reduces the circuit complexity and cost.
- We can implement many combinational circuits using Mux.
- It simplifies the logic design.
- It does not need the k-map and simplification.

Application of multiplexer

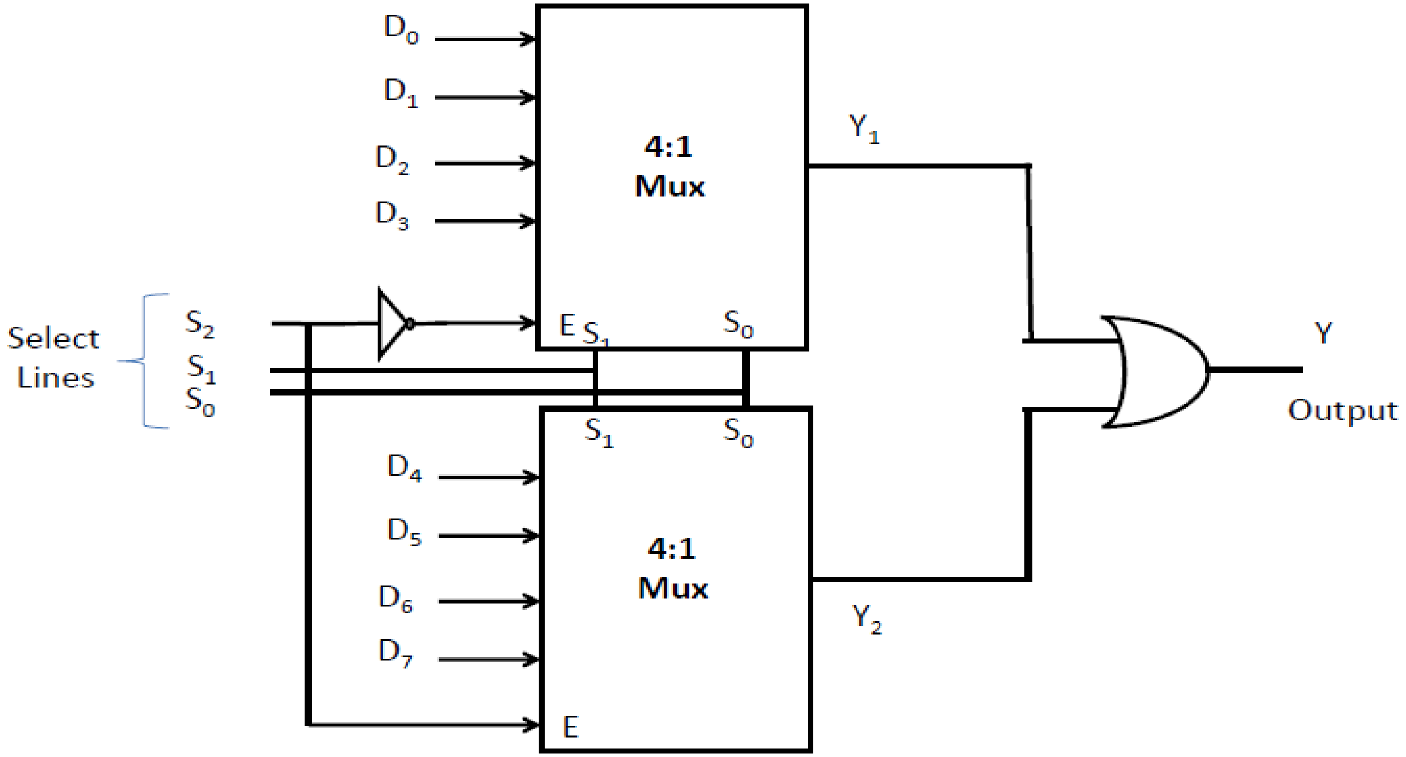
- Logic function generation
- Data selection
- Data routing
- Operation sequencing
- Parallel-to-serial conversion
- Waveform generation

Typical Application of a MUX



B	A	Selected Source
0	0	MP3
0	1	Laptop
1	0	Satellite
1	1	Cable TV

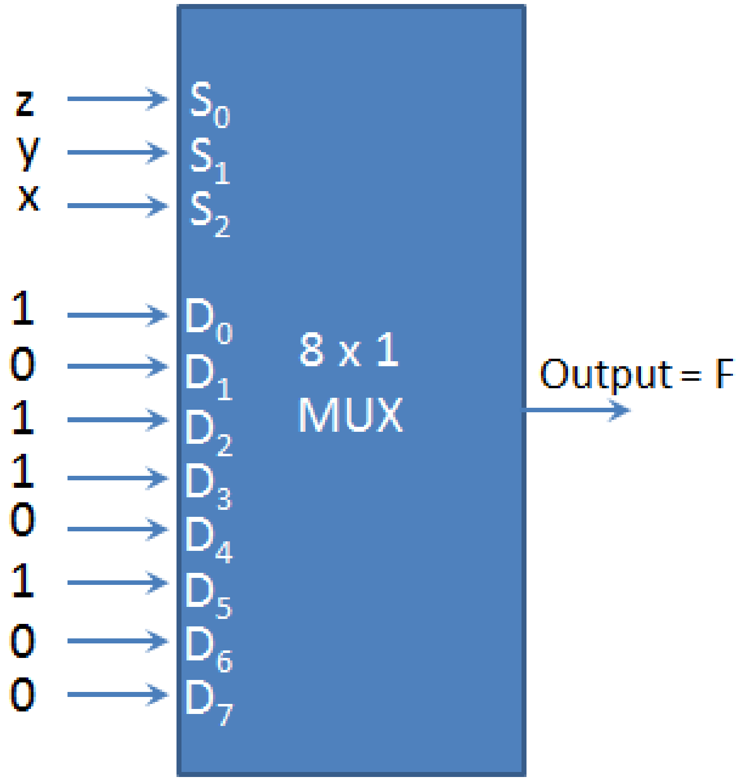
8:1 USING 4:1 MULTIPLXER



- Implement the following function using 8 to 1 MUX

$$F(x,y,z) = \Sigma\, m(0,2,3,5)$$

S_2	S_1	S_0	F
x	y	z	
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



Logic function generator

- Multiplexer with n-data select inputs can implement any function of n + 1 variables.
- The first n variables of the function as the select inputs and to use the least significant input variable and its complement to drive some of the data inputs.
- If the single variable is denoted by D, each data output of the multiplexer will be D, D', 1, or 0.
- Suppose, we wish to implement a 4-variable logic function using a multiplexer with three data select inputs.
- Let the input variables be A, B, C, and D; D is the LSB.

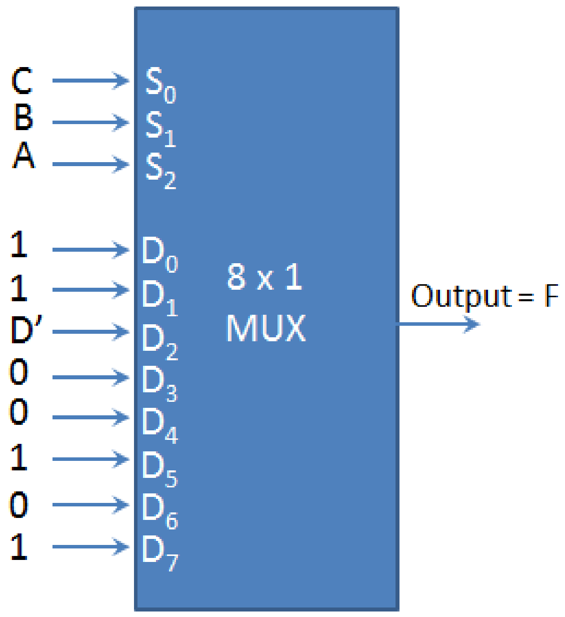
Logic function generator

- A truth table for the function F(A, B, C, D) is constructed with ABC has the same value twice once with D = 0 and again with D = 1.
- The following rules are used to determine the connections that should be made to the data inputs of the multiplexer.
  1. If F = 0 both times when the same combination of ABC occurs, connect logic 0 to the data input selected by that combination.
  2. If F = 1 both times when the same combination of ABC occurs, connect logic 1 to the data input selected by that combination.
  3. If F is different for the two occurrences of a combination of ABC, and if F = D in each case, connect D to the data input selected by that combination.

- 4. if F is different for the two occurrence of a combination of ABC ,and if F=D’ in each case, connect D’ to the data input selected by that combination.

Example: Use a multiplexer having 3 data select inputs to implement the logic for the Function given below. Also realize the same using a 16:1 MUX.F=∑m(0,1,2,3,4,10,11,14,15)

S1 A	S2 B	S3 C	D	F	
0	0	0	0	1	F=1
0	0	0	1	1	
0	0	1	0	1	F=1
0	0	1	1	1	
0	1	0	0	1	F=D’
0	1	0	1	0	
0	1	1	0	0	F=0
0	1	1	1	0	
1	0	0	0	0	F=0
1	0	0	1	0	
1	0	1	0	1	F=1
1	0	1	1	1	
1	1	0	0	0	F=0
1	1	0	1	0	
1	1	1	0	1	F=1
1	1	1	1	1	



Exercise

1.Use a 4 x 1 MUX to implement the logic function:

$F(A,B,C)=\sum m(0,2,3,5)$

2.Use a 4 x 1 MUX to implement the logic function:

$F(A,B,C)=\sum m(1,2,4,7)$

De-Multiplexer

- A demultiplexer(DEMUX) is a device that allows digital information from one source to be routed onto a multiple lines for transmission over different destinations.
- Consider an integer ‘m’, which is constrained by the following relation:

  $m = 2^n$, where m and n are both integers.
- A **1-to-m** Demultiplexer has
  - One Input: D
  - m Outputs: $O_0, O_1, O_2, \ldots\ldots\ldots\ldots\ldots O_{(m-1)}$
  - n Control inputs: $S_0, S_1, S_2, \ldots\ldots S_{(n-1)}$
  - One (or more) Enable input(s)

  Such that D may be transfer to one of the outputs, depending upon the control inputs.

- A de-multiplexer performs the reverse operation of a multiplexer i.e. it receives one input and distributes it over several outputs.
- At a time only one output line is selected by the select lines and the input is transmitted to the selected output line.
- It has only one input line, n number of output lines and m number of select lines.
- **Types of Demultiplexer :**
- 1) 1:2 demultiplexer, 2)1:4 demultiplexer
- 3)1:8 demultiplexer, 2)1:16 demultiplexer

Block Diagram of De-multiplexer

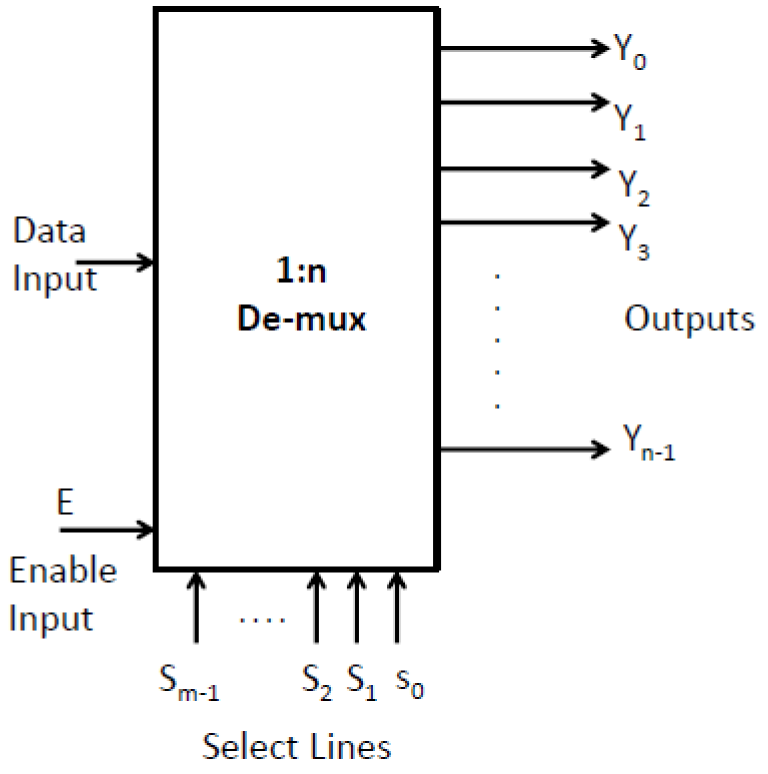


Fig. General Block Diagram

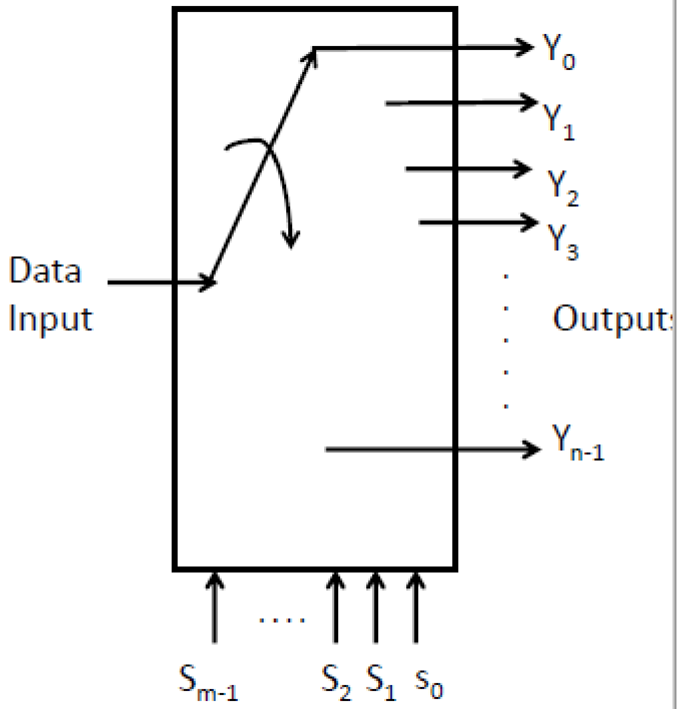
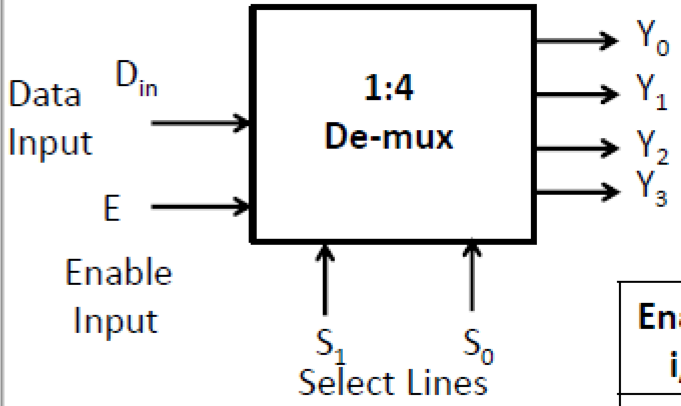


Fig. Equivalent Circuit

1:4 demultiplxer

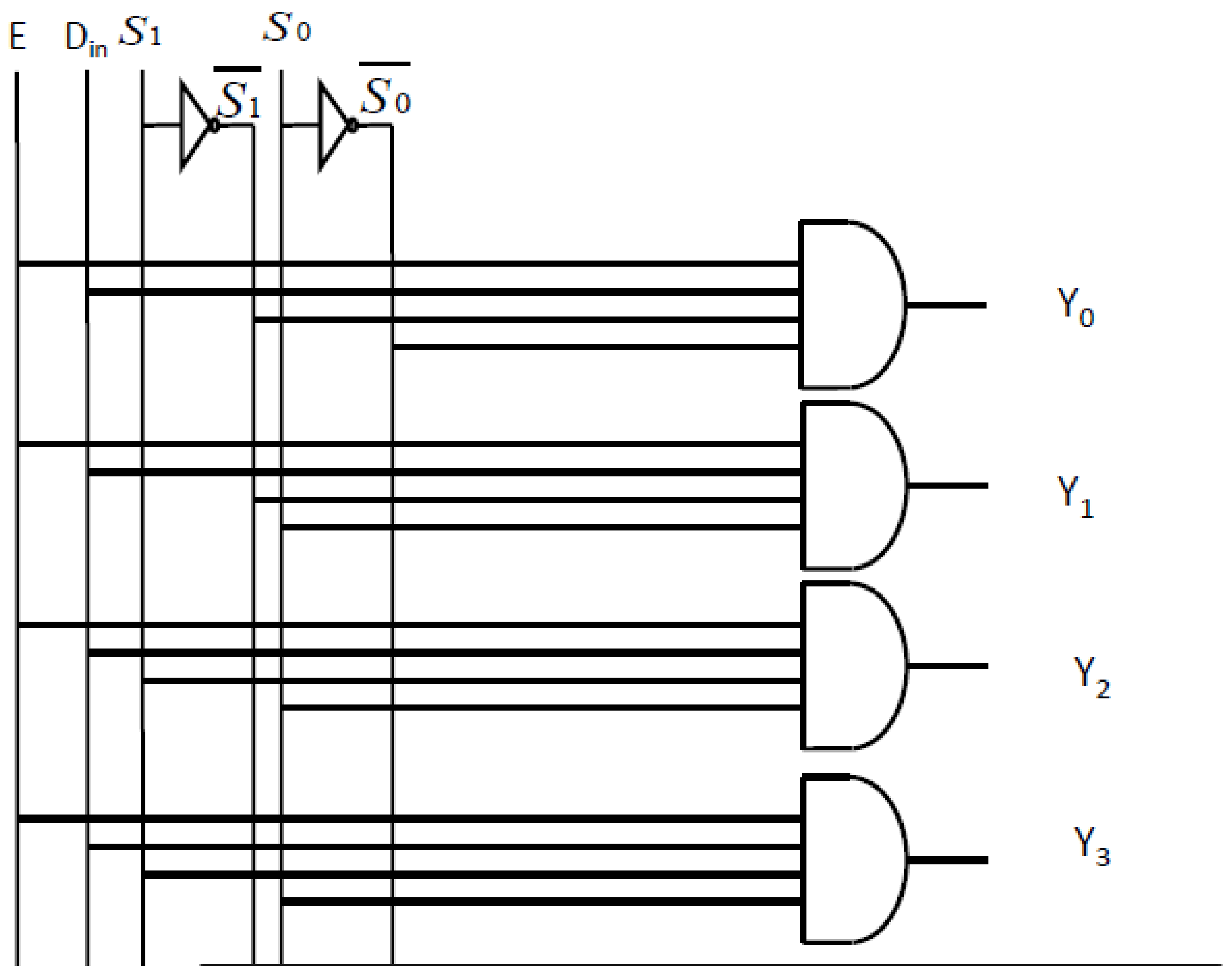


Block Diagram

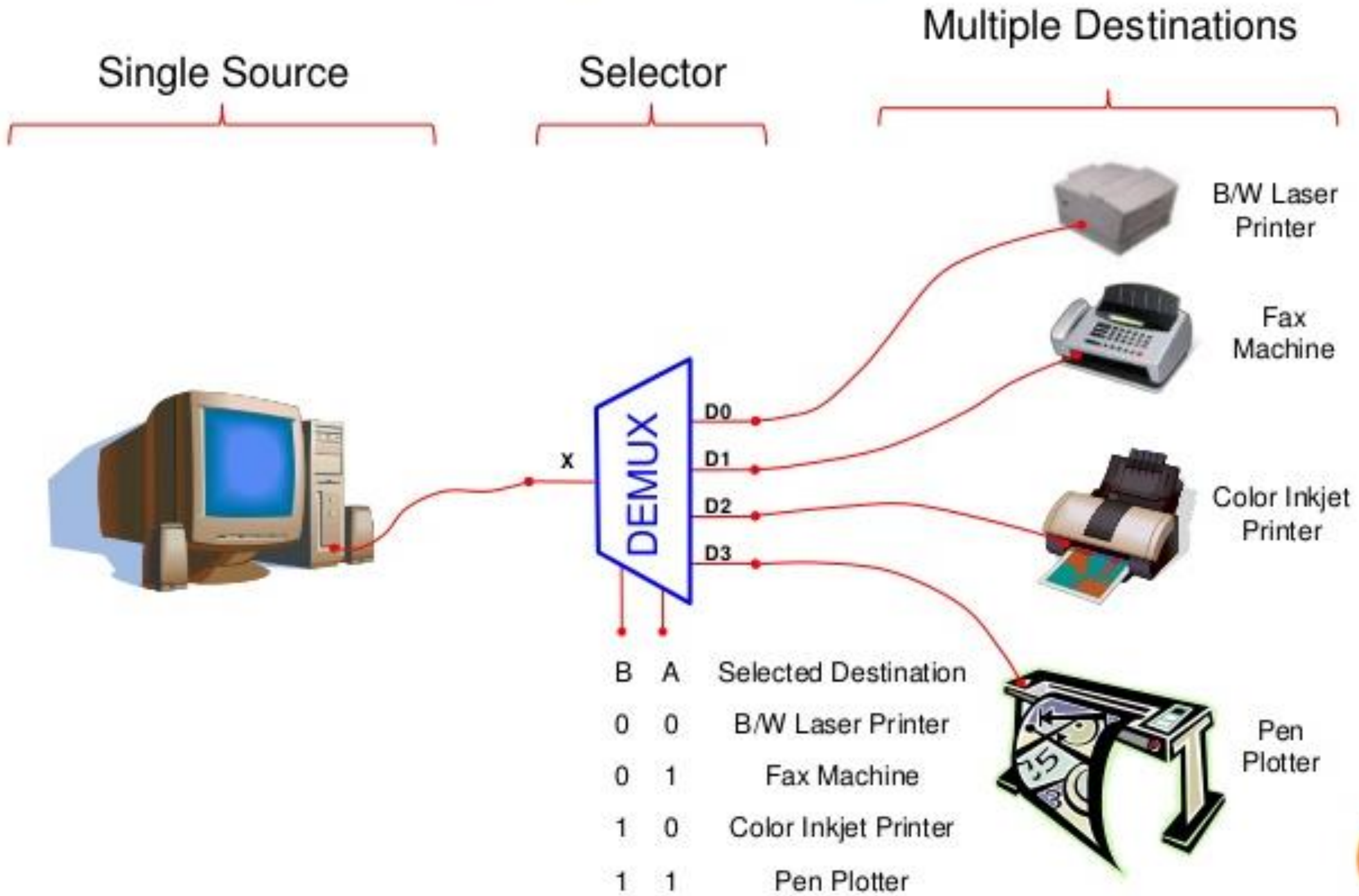
Truth Table

Enable i/p	Select i/p		Outputs			
E	**S_1**	**S_0**	**Y_0**	**Y_1**	**Y_2**	**Y_3**
0	X	X	0	0	0	0
1	0	0	D_{in}	0	0	0
1	0	1	0	D_{in}	0	0
1	1	0	0	0	D_{in}	0
1	1	1	0	0	0	D_{in}

1:4 De-mux using basic gates



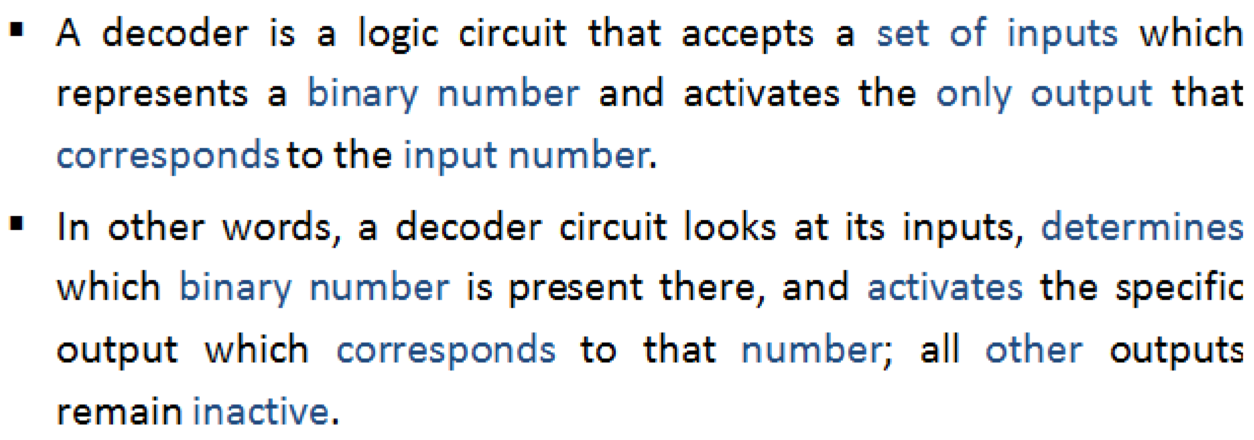
TYPICAL APPLICATION OF A DEMUX



B	A	Selected Destination
0	0	B/W Laser Printer
0	1	Fax Machine
1	0	Color Inkjet Printer
1	1	Pen Plotter

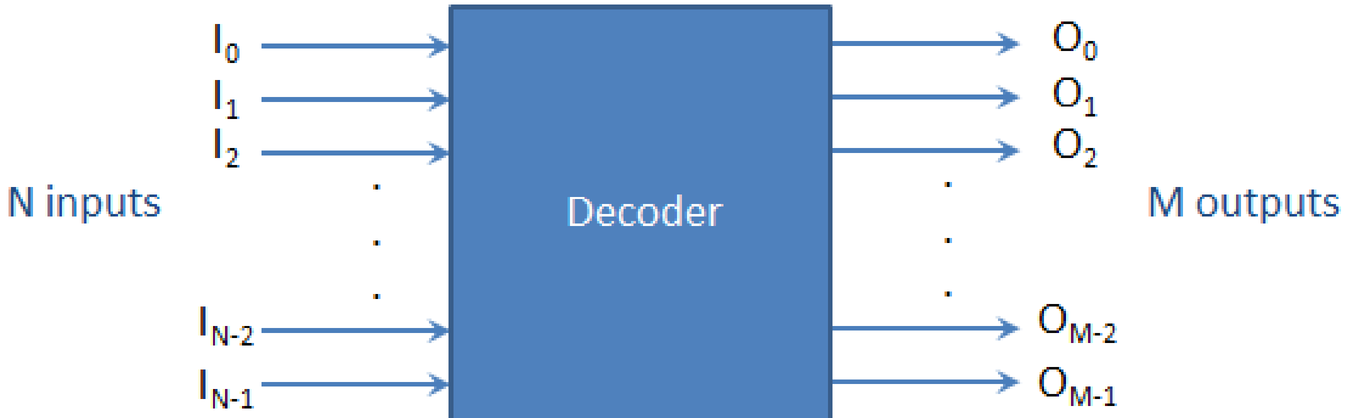
Decoder

- It is combinational circuit.
- It has n inputs and 2 ^{n} outputs.
- It is identical to demultiplxer without any data input.
- E-enable useful for cascading
- **Application:**
- Code converters, BCD to seven segments decoder.



- A decoder is a logic circuit that accepts a set of inputs which represents a binary number and activates the only output that corresponds to the input number.
- In other words, a decoder circuit looks at its inputs, determines which binary number is present there, and activates the specific output which corresponds to that number; all other outputs remain inactive.

- In its general form, a decoder has N input lines to handle N bits and M output lines such that only one output line is activated for each one of the possible combinations of inputs.



$M = 2^N$

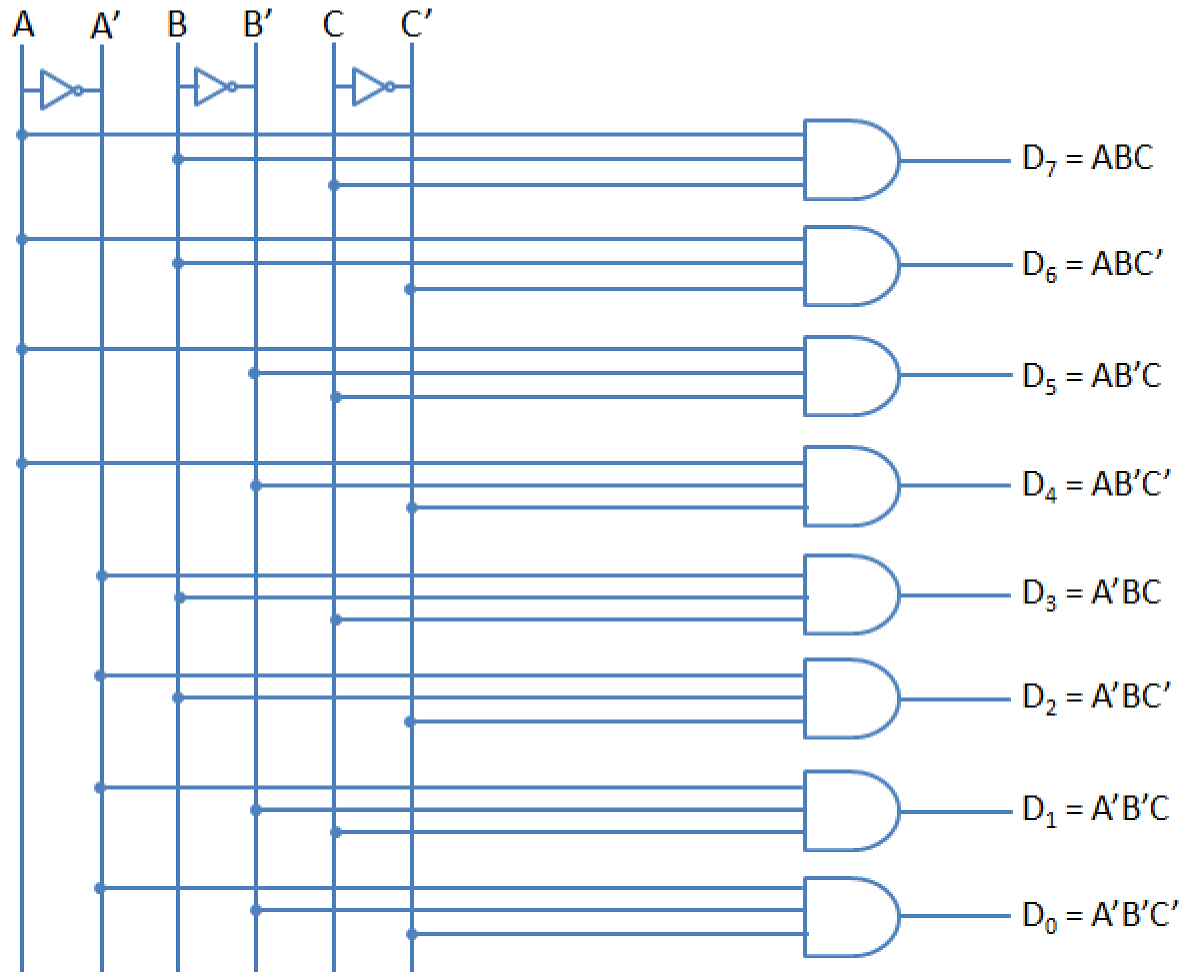
3 to 8 line Decoder

- It can be implemented using AND gates to achieve active – HIGH output.
- For active outputs ,NAND Gates Are Used.

Truth Table

Inputs			Outputs							
A	B	C	D_0 A'B'C'	D_1 A'B'C	D_2 A'BC'	D_3 A'BC	D_4 AB'C'	D_5 AB'C	D_6 ABC'	D_7 ABC
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

Logic diagram

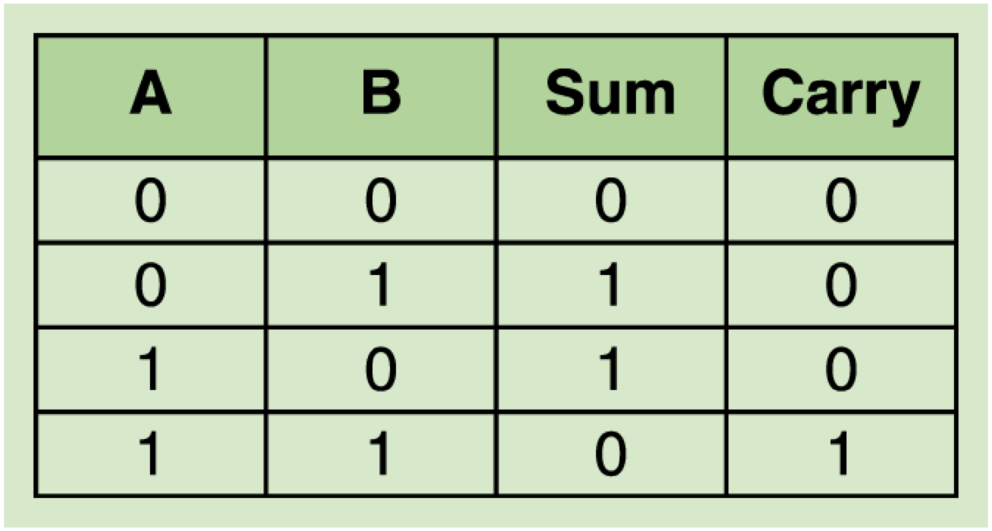


Adders

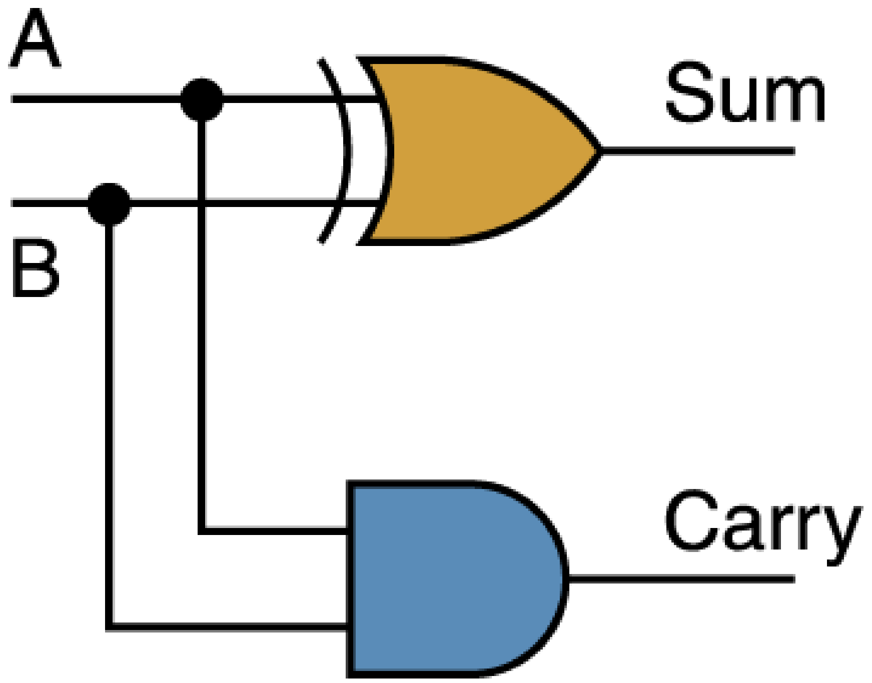
- At the digital logic level, addition is performed in binary
- Addition operations are carried out by special circuits called, appropriately, **adders**
- **It is a combinational logic circuit with two input and two output.**
- **Circuit has two output “carry” and “sum”.**

Half Adders

- The result of adding two binary digits could produce a *carry value*
- Recall that1+1 = 10 in base two
- A circuit that computes the sum of two bits and produces the correct carry bit is called a **half adder**



A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

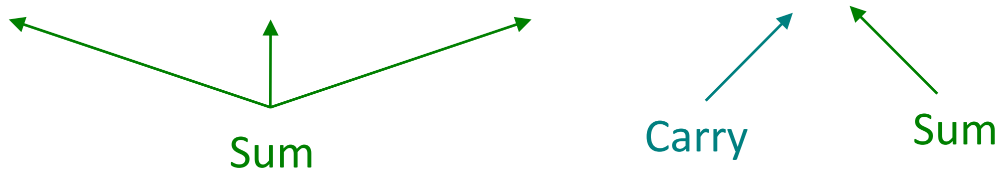


- Circuit diagram representing a half adder
- Two Boolean expressions:

$$\text{sum} = A \oplus B$$
$$\text{carry} = AB$$

Page 103

Limitation: In multi –digit addition we have to add two bits along with the carry of previous digit addition. Such addition requires addition of 3 bits . This is not possible in half adder.



$$\begin{array}{rrrr} 0 & \quad 0 & \quad 1 & \quad 1 \\ + \ \underline{0} & \quad + \ \underline{1} & \quad + \ \underline{0} & \quad + \ \underline{\quad 1} \\ 0 & \quad 1 & \quad 1 & \quad 10 \end{array}$$

The Full Adder

```
      0          0          0          0
      0          0          1          1
  +   0      +   1      + 0    +   1
      0          1          1        10

                                          Carry-in

      1          1          1          1
      0          0          1          1
  +   0      +   1      + 0    +   1
      1         10         10        11
```

Carry-out   Sum

Full Adder

- In a full adder, three bits can be added at a time. The third bit is a carry from a less significant column.

Inputs			Outputs	
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	0	1
1	1	0	0	1
1	1	1	1	1

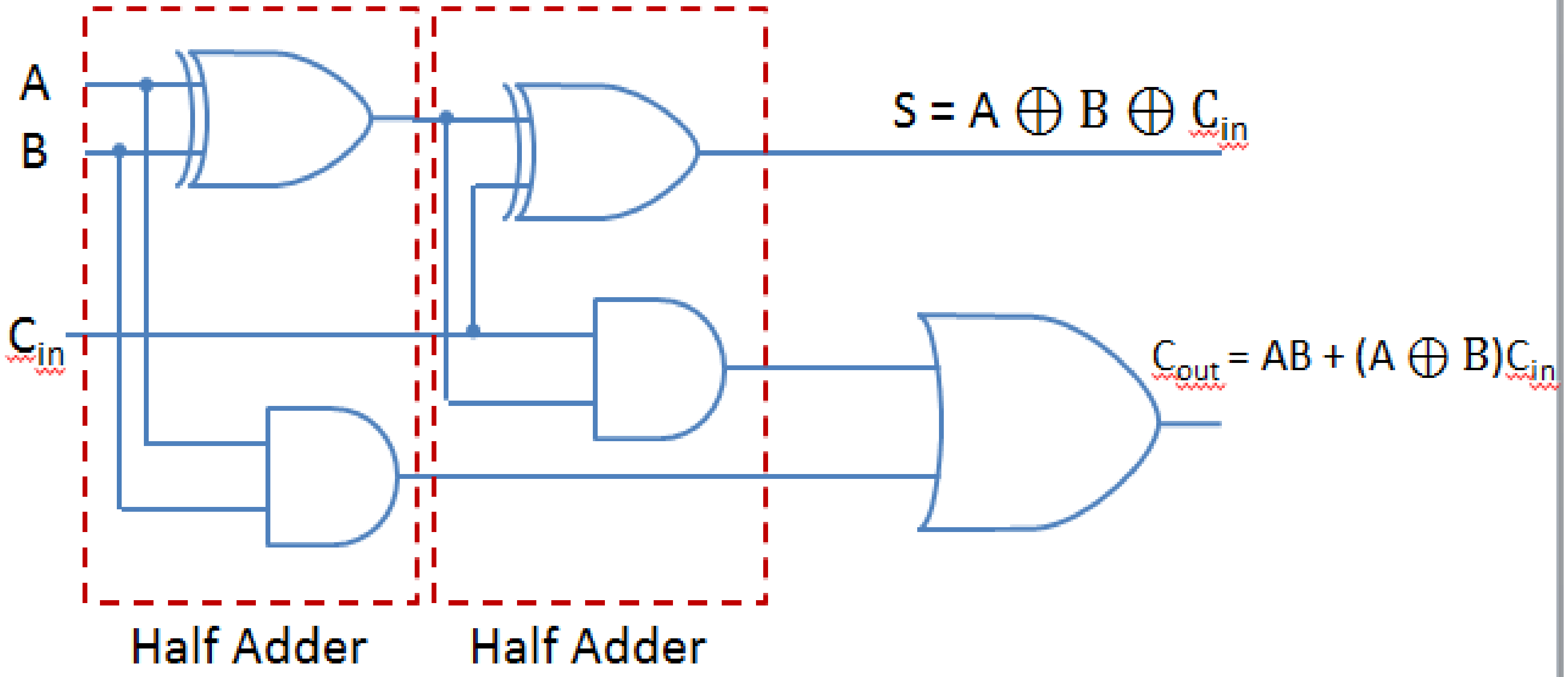
$$\begin{aligned} S &= A'B'C_{in} + A'BC_{in}' + AB'C_{in}' + ABC_{in} \\ &= (AB' + A'B)C_{in}' + (AB + A'B')C_{in} \\ &= (A \oplus B)C_{in}' + (A \oplus B)'C_{in} \\ &= A \oplus B \oplus C_{in} \end{aligned}$$

$$\begin{aligned} C_{out} &= A'BC_{in} + AB'C_{in} + ABC_{in}' + ABC_{in} \\ &= AB + (A \oplus B)C_{in} \end{aligned}$$

$S = A \oplus B \oplus C_{in}$

$C_{out} = AB + (A \oplus B)C_{in}$

Logic diagram for full adder

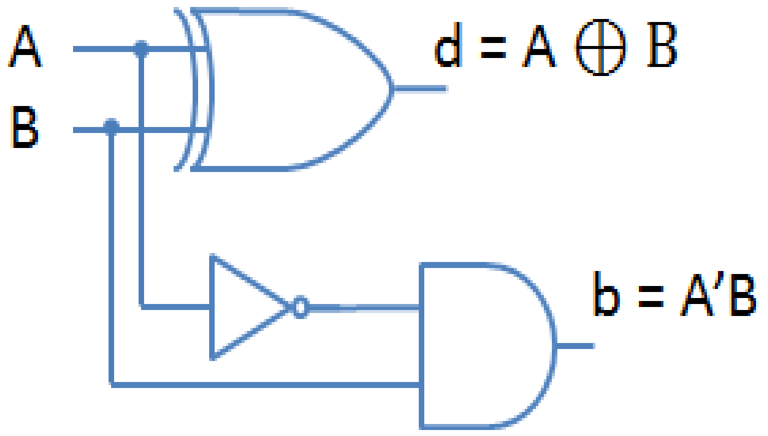


A
B
C_{in}
$S = A \oplus B \oplus C_{in}$
$C_{out} = AB + (A \oplus B)C_{in}$
Half Adder
Half Adder

Half SubtraCtor

- Subtracts one bit from the other and produces the difference.
- Other output is to specify if 1 is borrowed

Inputs		Outputs	
A	B	d	b
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



Full Subtractor

- Half subtractor can be used only for LSB subtraction.
- Borrow from LSBs affects the subtraction in the next higher column.
- Subtrahend bit is subtracted from minuend and consider borrow if generated from preceding column.
- Full subtractor is a combinational circuit with 3 inputs (A, B, b_i)
- Subtraction = $A - B - b_i$

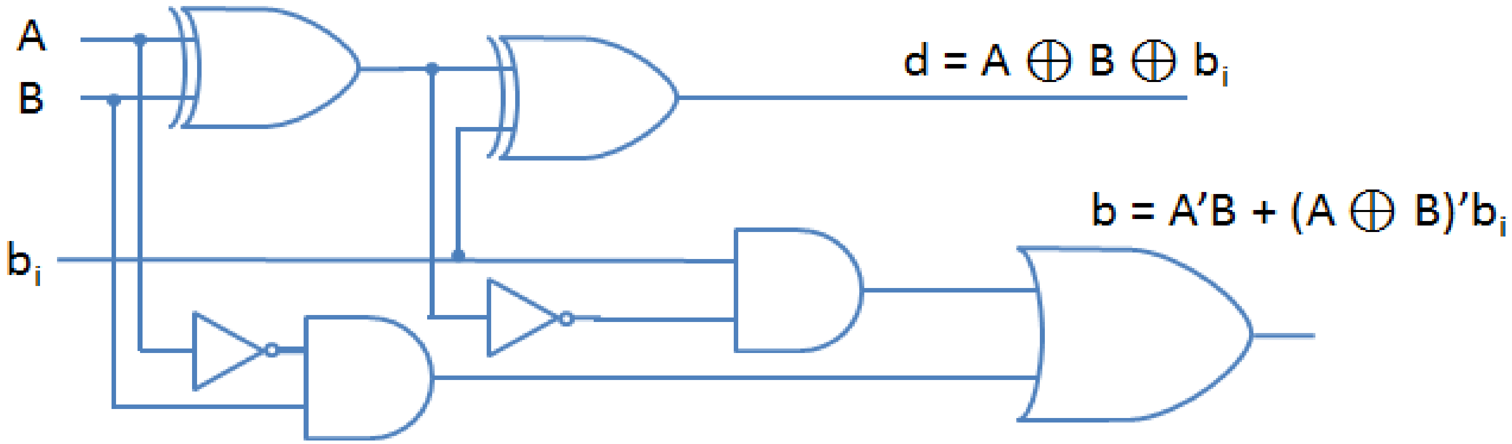
Inputs			Outputs	
A	B	b_i	d	b
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

$$\begin{aligned} d &= A'B'b_i + A'Bb_i' + AB'b_i' + ABb_i \\ &= (AB' + A'B)b_i' + (AB + A'B')b_i \\ &= (A \oplus B)b_i' + (A \oplus B)'b_i \\ &= A \oplus B \oplus b_i \end{aligned}$$

$$\begin{aligned} b &= A'B'b_i + A'Bb_i' + A'Bb_i + ABb_i \\ &= A'B(b_i + b_i') + (AB + A'B')b_i \\ &= A'B + (A \oplus B)'b_i \end{aligned}$$

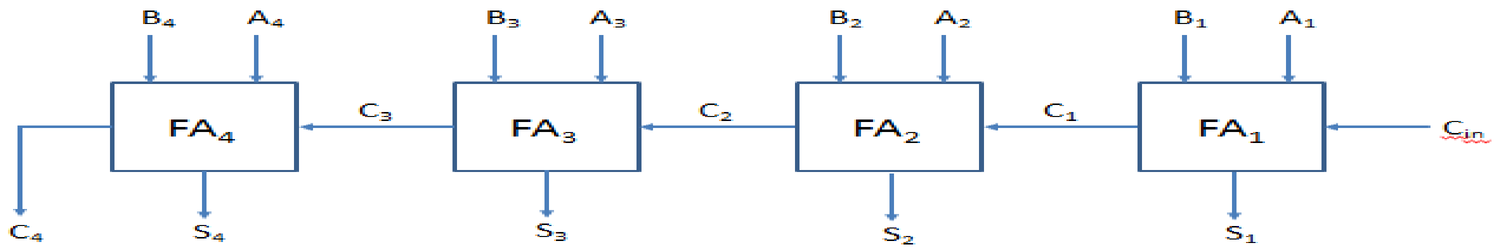
$d = A \oplus B \oplus b_i$

$b = A'B + (A \oplus B)'b_i$



Binary parallel adder

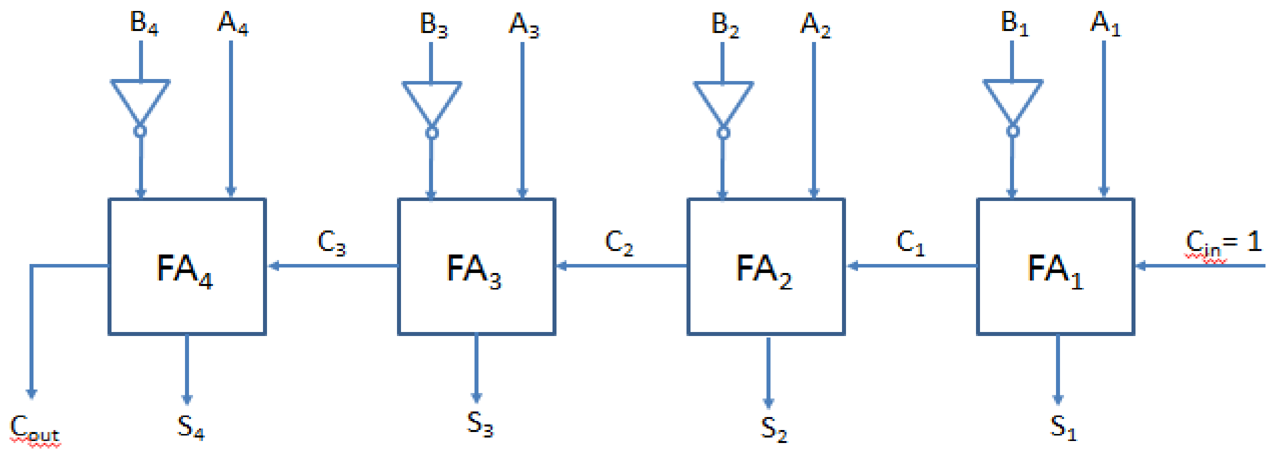
- The full adder is capable of adding two single digit binary numbers along with a carry input.
- But in practice we need to add binary numbers which are much longer than one bit.
- To add two n-bit binary numbers we need to use the n-bit parallel adder.
- It uses a number of full adders in cascade.
- The carry output of the previous full adder is connected to the carry input of the next full adder.
- **4 – Bit Parallel Adder using full adder**



Binary Parallel Subtractor

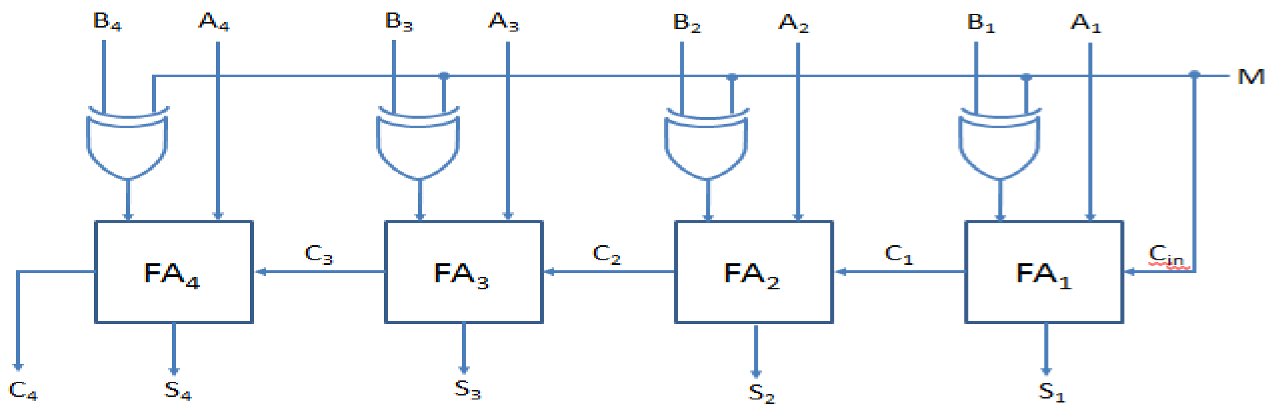
-Its carried out by 2's complement . 2's complement can be obtained by 1's complement and adding 1 to the least significant pair of bits.

-



Binary Adder Subtractor

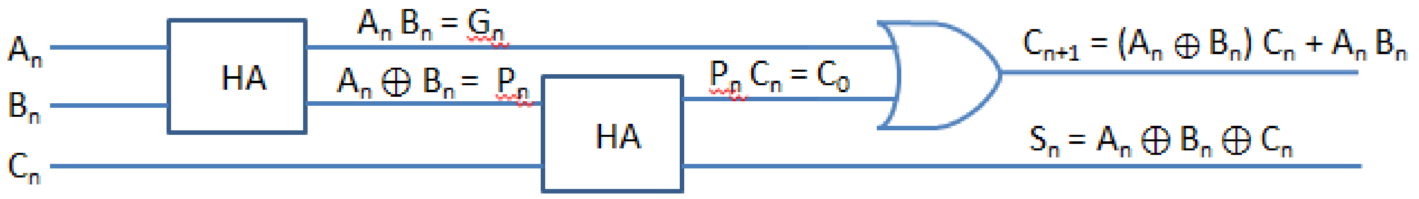
- Addition And Subtraction Operation Are Combined Into One Circuit With One Common Binary Adder .this is done by including an X-OR adder ,when M=1 circuit become subtractor, M=0 Its Adder.



- When M = 0, Circuit is an adder.
- When M = 1. Circuit becomes a subtractor.

Look ahead carry adder

- Speeds up the process by eliminating the ripple carry delay.
- The method of speeding up the addition process is based on: *carry generate* and *carry propagate* functions.
- Consider one full adder stage; say the nth stage of a parallel adder shown in below Figure.



A_n
B_n
C_n
HA
$A_n B_n = G_n$
$A_n \oplus B_n = P_n$
HA
$P_n C_n = C_0$
$C_{n+1} = (A_n \oplus B_n) C_n + A_n B_n$
$S_n = A_n \oplus B_n \oplus C_n$

- $S_n = P_n \oplus C_n$ where $P_n = A_n \oplus B_n$
- $C_{n+1} = G_n + P_n C_n$ where $G_n = A_n B_n$
- Possible to express the output carry of a higher significant stage in terms of applied input variables A, B and carry-in to the LSB adder.
- Based on these, the expression for the carry-outs of various full adders are as follows:

$C_2 = G_1 + P_1 C_1$

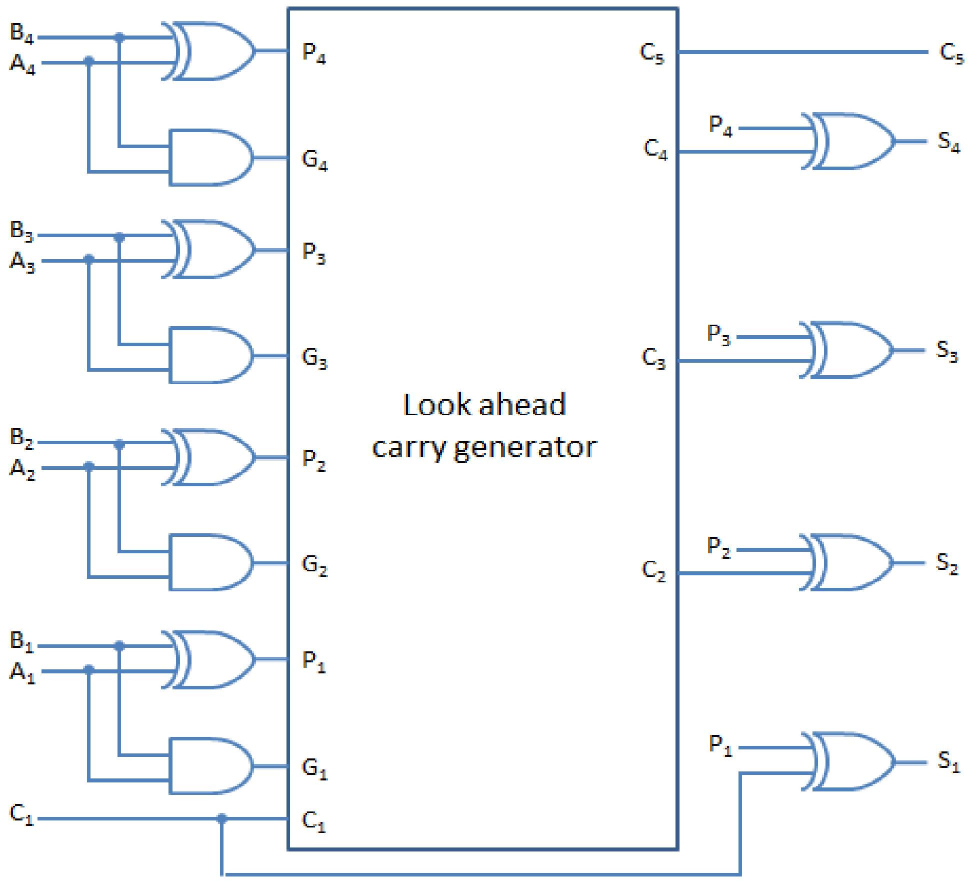
$C_3 = G_2 + P_2 C_2$

$= G_2 + P_2 (G_1 + P_1 C_1)$

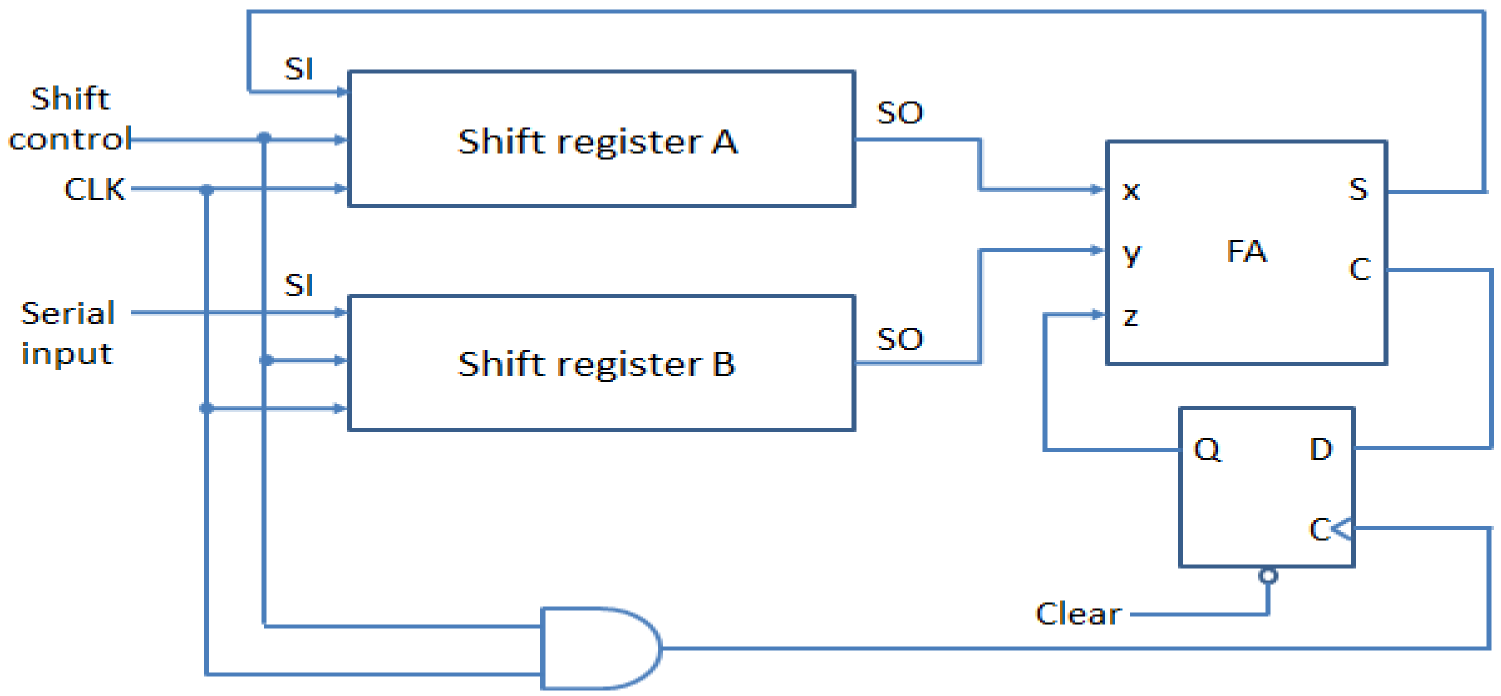
$= G_2 + P_2 G_1 + P_2 P_1 C_1$

$C_4 = G_3 + P_3 C_3 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_1$

$C_5 = G_4 + P_4 C_4 = G_4 + P_4 G_3 + P_4 P_3 G_2 + P_4 P_3 P_2 G_1 + P_4 P_3 P_2 P_1 C_1$

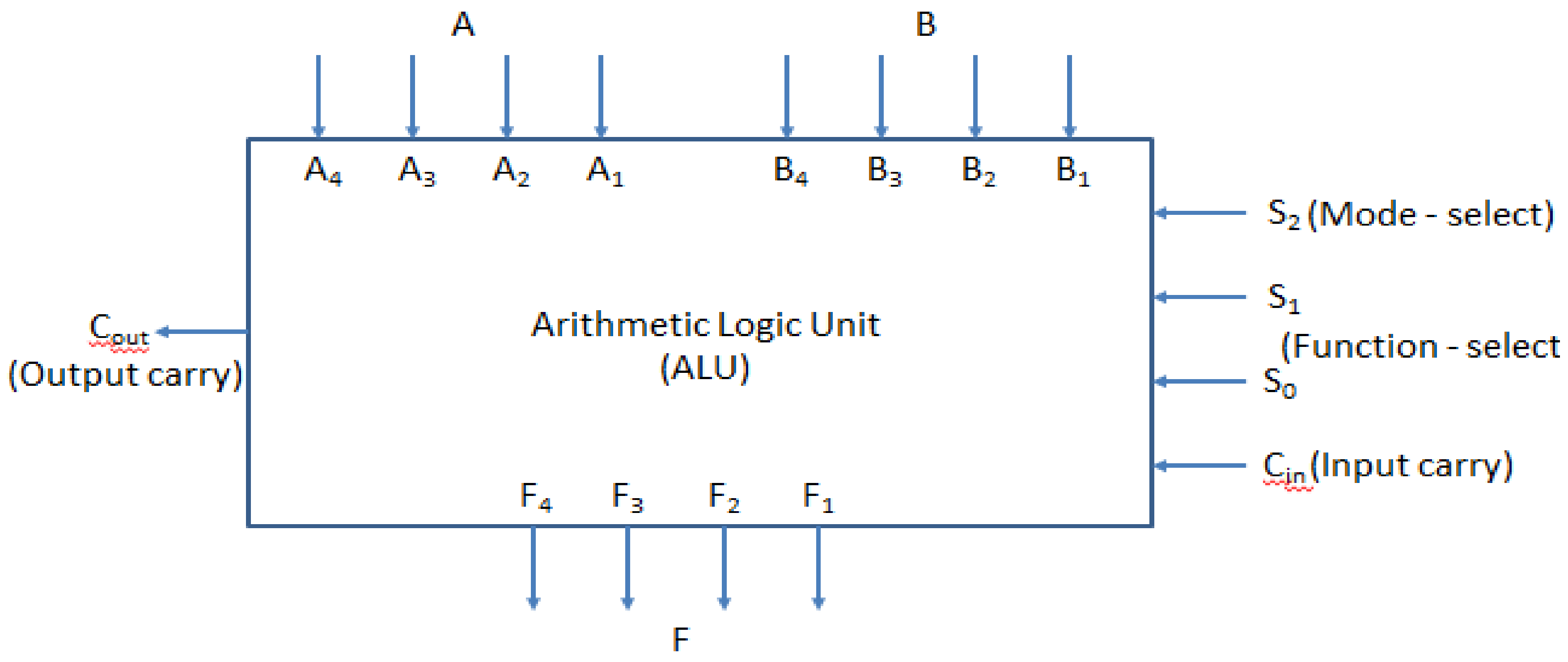

B4
A4
P4
G4
B3
A3
P3
G3
B2
A2
P2
G2
B1
A1
P1
G1
C1
Look ahead carry generator
C5
C4
C3
C2
S4
S3
S2
S1

Serial adder



Arithmetic logic unit

Block diagram OF ALU



Selection				Output	Function
S_2	S_1	S_0	C_{in}		
0	0	0	0	$F = A$	Transfer A
0	0	0	1	$F = A + 1$	Increment A
0	0	1	0	$F = A + B$	Addition
0	0	1	1	$F = A + B + 1$	Add with carry
0	1	0	0	$F = A - B - 1$	Subtract with borrow
0	1	0	1	$F = A - B$	Subtraction
0	1	1	0	$F = A - 1$	Decrement A
0	1	1	1	$F = A$	Transfer A
1	0	0	X	$F = A + B$	OR
1	0	1	X	$F = A \oplus B$	XOR
1	1	0	X	$F = A . B$	AND
1	1	1	X	$F = A'$	Complement A

Comparator

- A comparator is a logic circuit used to compare the magnitudes of two binary numbers.
- Comparator circuit provides 3 outputs
  1. $A = B$
  2. $A > B$
  3. $A < B$
- X-NOR gate is a basic comparator (the output is 1 if and only if the input bits coincide).
- 2 binary numbers are equal if and only if all their corresponding bits coincide.
- For e.g. $A_3A_2A_1A_0$ and $B_3B_2B_1B_0$ are equal if and only if $A_3=B_3$, $A_2=B_2$, $A_1=B_1$ and $A_0=B_0$

$$\text{Equality} = (A_3 \odot B_3)\ (A_2 \odot B_2)\ (A_1 \odot B_1)\ (A_0 \odot B_0)$$

1- Bit magnitude comparator

- Let the 1-bit numbers be $A = A_0$ and $B = B_0$
- If $A_0 = 1$ and $B_0 = 0$ then $A > B$

  $A > B : G = A_0B_0'$

- If $A_0 = 0$ and $B_0 = 1$ then $A < B$

  $A < B : L = A_0'B_0$

- If $A_0 = 1$ and $B_0 = 1$ (coincides) then $A = B$

  $A = B : E = A_0 \odot B_0$

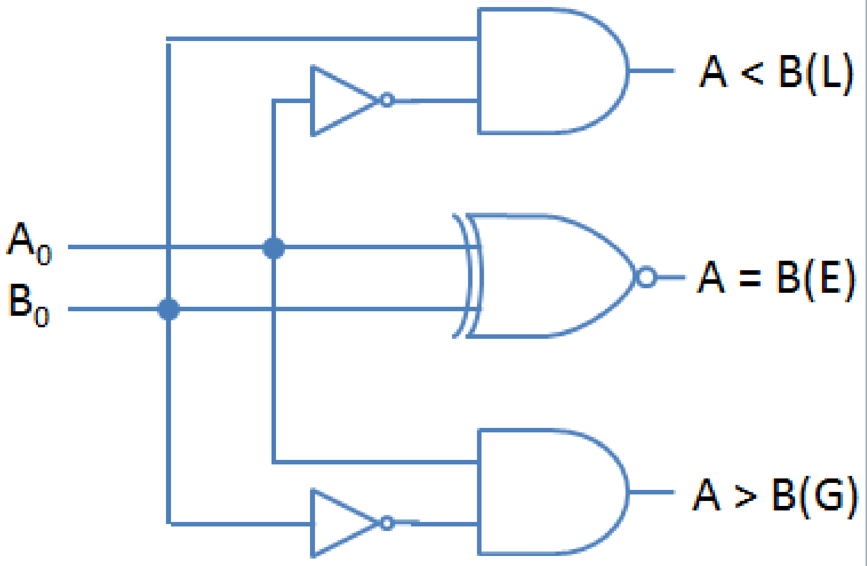
- Truth table and logic diagram are as follows

A_0	B_0	L	E	G
0	0			
0	1			
1	0			
1	1			

$A < B : L = A_0'B_0$

$A = B : E = A_0 \odot B_0$

$A > B : G = A_0B_0'$



2- Bit magnitude comparator

- Let the two 2-bit numbers be $A = A_1A_0$ and $B = B_1B_0$

1. If $A_1 = 1$ and $B_1 = 0$, then $A > B$ or
2. If A_1 and B_1 coincide and $A_0 = 1$ and $B_0 = 0$, then $A > B$.

$$A > B : G = A_1B_1' + (A_1 \odot B_1)\, A_0B_0'$$

1. If $A_1 = 0$ and $B_1 = 1$, then $A < B$ or
2. If A_1 and B_1 coincide and $A_0 = 0$ and $B_0 = 1$, then $A < B$.

$$A < B : L = A_1'B_1 + (A_1 \odot B_1)\, A_0'B_0$$

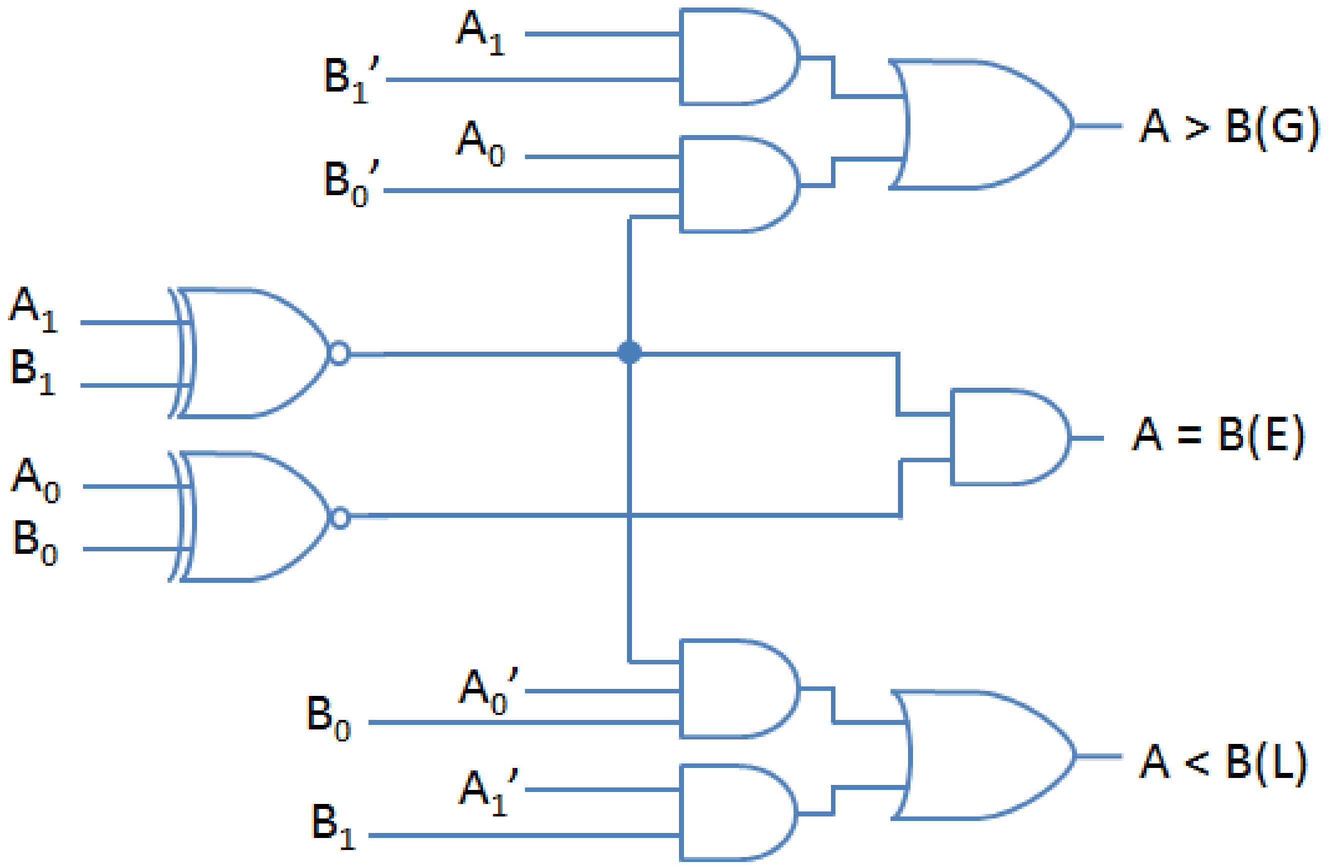
1. If A_1 and B_1 coincide and if A_0 and B_0 coincide then $A = B$.

$$A = B : E = (A_1 \odot B_1)(A_0 \odot B_0)$$

Table 1. Truth Table of 2-Bit Magnitude Comparator

INPUT				OUTPUT		
A1	**A0**	**B1**	**B0**	**A>B**	**A=B**	**A<B**
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

- Logic diagram for a 2-bit comparator is as follows

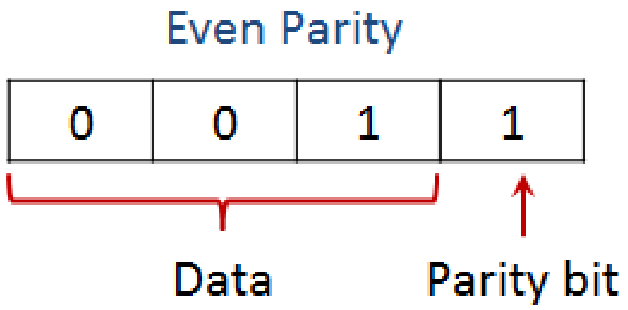


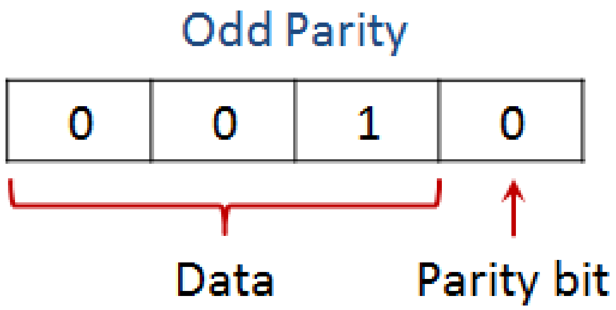
Exercise

- Design 4-bit magnitude comparator

Parity Generator

- Binary data, when transmitted and processed is susceptible to noise that can alter its 1s to 0s and 0s to 1s.
- To detect such errors, an additional bit called *parity bit* is added to the data bits and the word containing the data bits and the parity bit is transmitted.
- At the receiving end the number of 1s in the word received are counted and the error, if any, is detected.





3-bit parity generator even parity

- Let the 3-bit input be A, B and C.
- For even parity, a parity bit 1 is added such that the total number of 1s in the 3-bit input and parity bit together is even.

Truth Table

Inputs			Outputs
A	**B**	**C**	**parity bit (f)**
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

K - Map

C \ AB	00	01	11	10
0		$\boxed{1}$		$\boxed{1}$
1	$\boxed{1}$		$\boxed{1}$	

$$f = A'B'C + A'BC' + ABC + AB'C'$$

$$f = A'(B'C + BC') + A(BC + B'C')$$

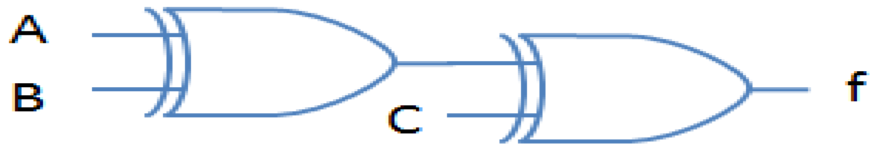
$$f = A'(B \oplus C) + A(B \oplus C)'$$

$$f = A \oplus B \oplus C$$

Logic diagram

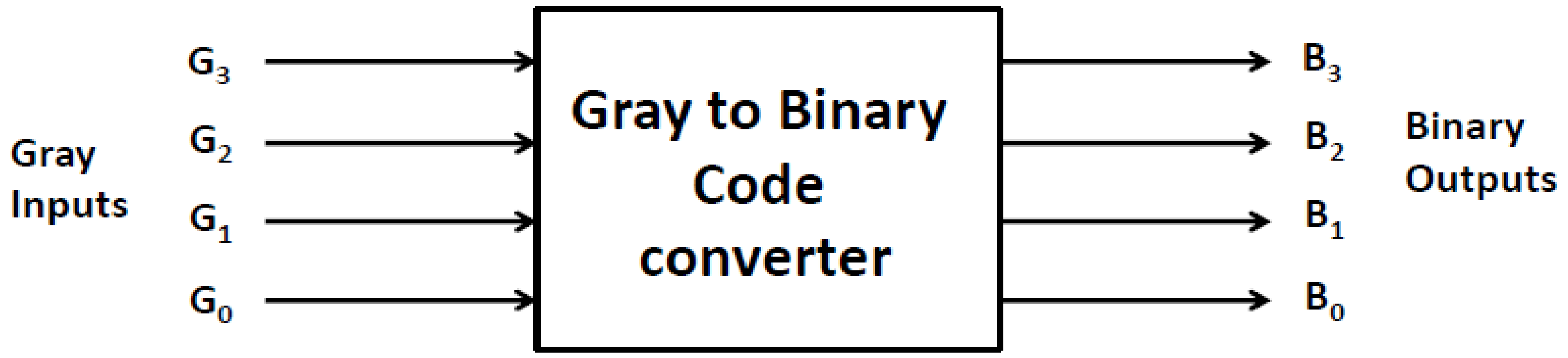
$f = A \oplus B \oplus C$

Logic Diagram



Design of gray to binary code converter

Block Diagram:



Truth Table :

Gray Inputs				Binary Outputs			
G_3	G_2	G_1	G_0	B_3	B_2	B_1	B_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

K-map for B_0:

$G_3G_2 \backslash G_1G_0$		G_1G_0 00	G_1G_0 01	G_1G_0 11	G_1G_0 10
$\overline{G_3}\overline{G_2}$	00	0 ^{0}	1 ^{1}	0 ^{3}	1 ^{2}
$\overline{G_3}G_2$	01	1 ^{4}	0 ^{5}	1 ^{7}	0 ^{6}
G_3G_2	11	0 ^{12}	1 ^{13}	0 ^{15}	1 ^{14}
$G_3\overline{G_2}$	10	1 ^{8}	0 ^{9}	1 ^{11}	0 ^{10}

$$B_0 = \overline{G_3}\,\overline{G_2}\,\overline{G_1}G_0 + \overline{G_3}\,\overline{G_2}G_1\overline{G_0} + \overline{G_3}G_2\overline{G_1}\,\overline{G_0} + \overline{G_3}G_2G_1G_0$$
$$+G_3\overline{G_2}\,\overline{G_1}G_0 + G_3\overline{G_2}G_1G_0 + G_3G_2\overline{G_1}G_0 + G_3G_2G_1\overline{G_0}$$
$$B_0 = G_3 \oplus G_2 \oplus G_1 \oplus G_0$$

K-map for B_1:

$G_3G_2 \backslash G_1G_0$		$\overline{G_1}\overline{G_0}$ 00	$\overline{G_1}G_0$ 01	G_1G_0 11	$G_1\overline{G_0}$ 10
$\overline{G_3}\overline{G_2}$	00	0 ^{0}	0 ^{1}	1 ^{3}	1 ^{2}
$\overline{G_3}G_2$	01	1 ^{4}	1 ^{5}	0 ^{7}	0 ^{6}
G_3G_2	11	0 ^{12}	0 ^{13}	1 ^{15}	1 ^{14}
$G_3\overline{G_2}$	10	1 ^{8}	1 ^{9}	0 ^{11}	0 ^{10}

$$B_1 = \overline{G_3}\,\overline{G_2}G_1 + \overline{G_3}G_2\overline{G_1} + G_3G_2G_1 + G_3\overline{G_2}\,\overline{G_1}$$

$$B_1 = G_3 \oplus G_2 \oplus G_1$$

K-map for B$_2$:

$G_3G_2 \backslash G_1G_0$	G_1G_0 00	G_1G_0 01	G_1G_0 11	G_1G_0 10
$\overline{G_3}\,\overline{G_2}$ 00	$0\ {}^{0}$	$0\ {}^{1}$	$0\ {}^{3}$	$0\ {}^{2}$
$\overline{G_3}G_2$ 01	$1\ {}^{4}$	$1\ {}^{5}$	$1\ {}^{7}$	$1\ {}^{6}$
G_3G_2 11	$0\ {}^{12}$	$0\ {}^{13}$	$0\ {}^{15}$	$0\ {}^{14}$
$G_3\overline{G_2}$ 10	$1\ {}^{8}$	$1\ {}^{9}$	$1\ {}^{11}$	$1\ {}^{10}$

$$B_2 = \overline{G_3}G_2 + G_3\overline{G_2}$$

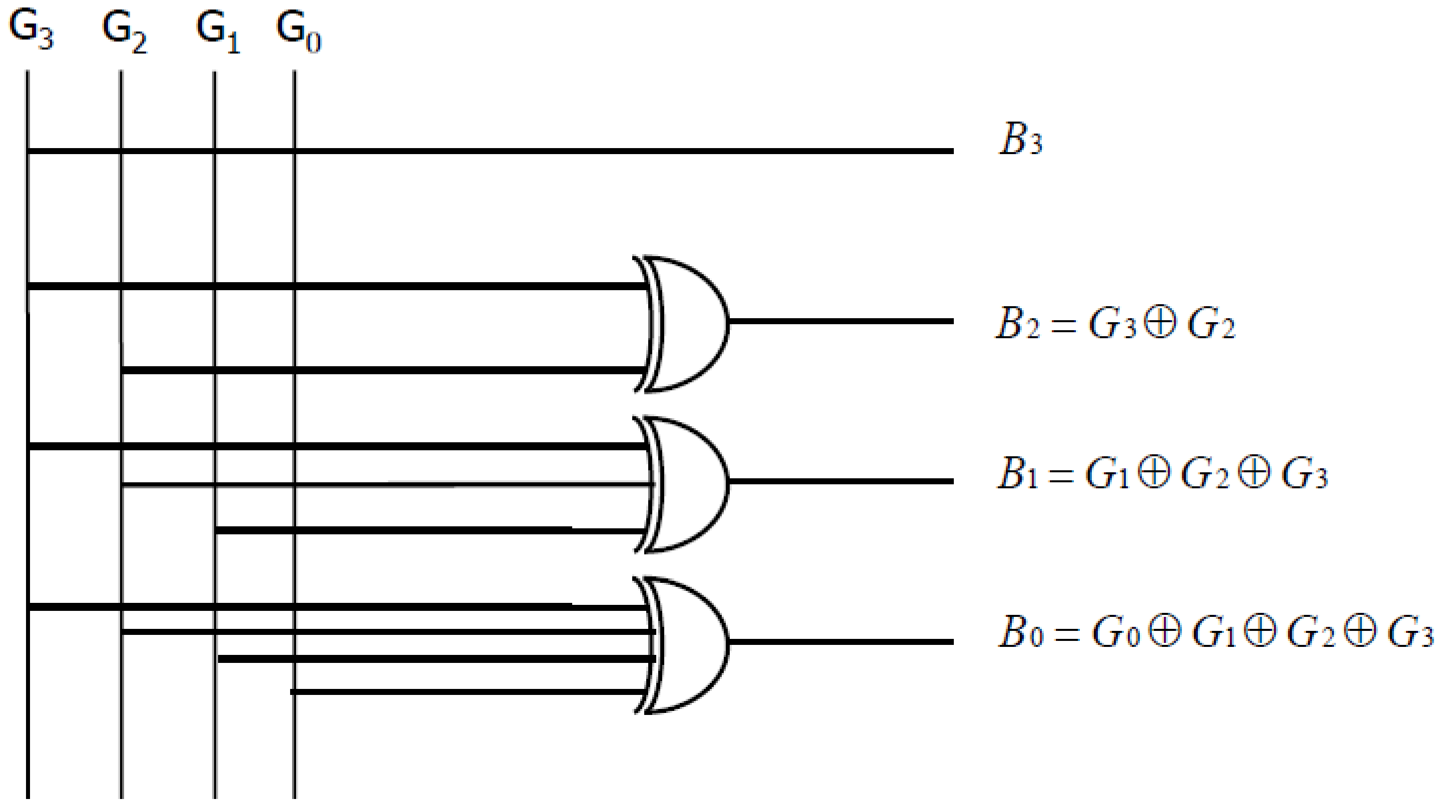
$$B_1 = G_3 \oplus G_2$$

K-map for B$_3$:

$G_3G_2 \backslash G_1G_0$	$\overline{G_1}\,\overline{G_0}$ 00	$\overline{G_1}G_0$ 01	G_1G_0 11	$G_1\overline{G_0}$ 10
$\overline{G_3}\,\overline{G_2}$ 00	$0\ {}^{0}$	$0\ {}^{1}$	$0\ {}^{3}$	$0\ {}^{2}$
$\overline{G_3}G_2$ 01	$0\ {}^{4}$	$0\ {}^{5}$	$0\ {}^{7}$	$0\ {}^{6}$
G_3G_2 11	$1\ {}^{12}$	$1\ {}^{13}$	$1\ {}^{15}$	$1\ {}^{14}$
$G_3\overline{G_2}$ 10	$1\ {}^{8}$	$1\ {}^{9}$	$1\ {}^{11}$	$1\ {}^{10}$

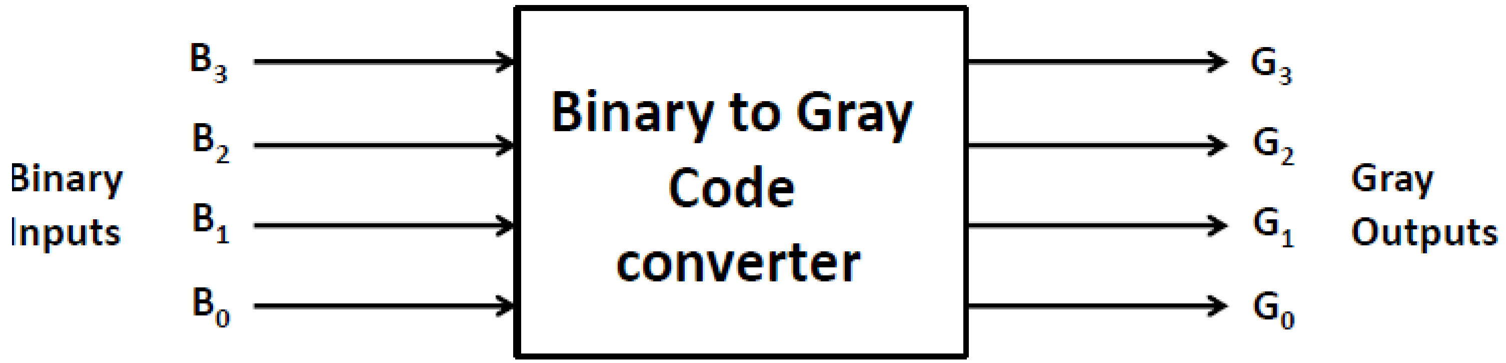
$$B_3 = G_3$$

Logic Diagram:


G_3
G_2
G_1
G_0
B_3
$B_2 = G_3 \oplus G_2$
$B_1 = G_1 \oplus G_2 \oplus G_3$
$B_0 = G_0 \oplus G_1 \oplus G_2 \oplus G_3$

Design od binary to Gray

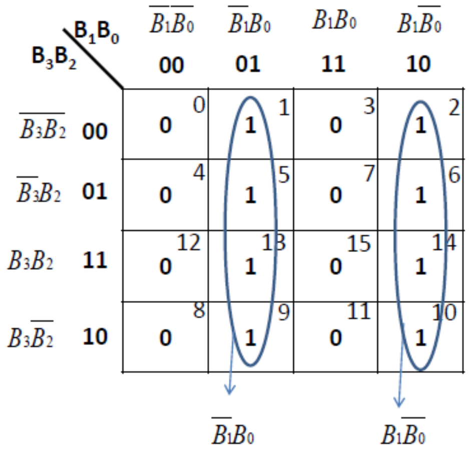
Block Diagram:



Truth Table :

Binary Inputs				Gray Outputs			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

K-map for G_0:

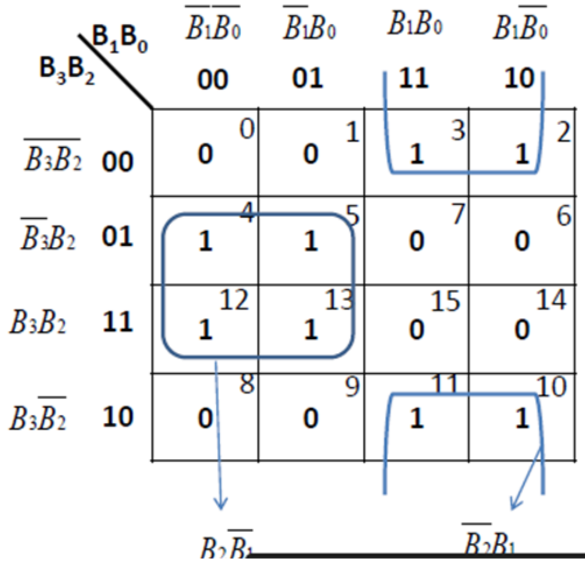


B_3B_2 \ B_1B_0	$\overline{B_1}\overline{B_0}$ 00	$\overline{B_1}B_0$ 01	B_1B_0 11	$B_1\overline{B_0}$ 10
$\overline{B_3B_2}$ 00	0^{0}	1^{1}	0^{3}	1^{2}
$\overline{B_3}B_2$ 01	0^{4}	1^{5}	0^{7}	1^{6}
B_3B_2 11	0^{12}	1^{13}	0^{15}	1^{14}
$B_3\overline{B_2}$ 10	0^{8}	1^{9}	0^{11}	1^{10}

$$G_0 = \overline{B_1}B_0 + B_1\overline{B_0}$$

$$\therefore G_0 = B_0 \oplus B_1$$

K-map for G_1:

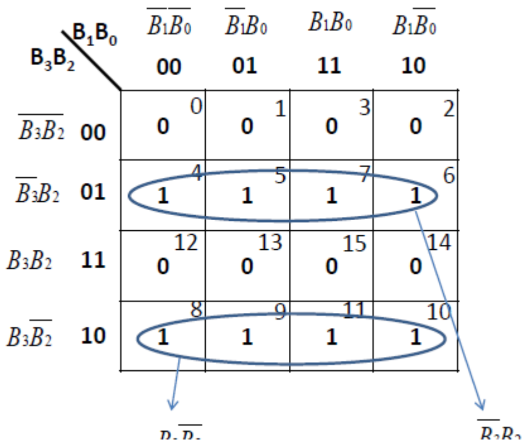


B_3B_2 \ B_1B_0	$\overline{B_1}\overline{B_0}$ 00	$\overline{B_1}B_0$ 01	B_1B_0 11	$B_1\overline{B_0}$ 10
$\overline{B_3B_2}$ 00	0^{0}	0^{1}	1^{3}	1^{2}
$\overline{B_3}B_2$ 01	1^{4}	1^{5}	0^{7}	0^{6}
B_3B_2 11	1^{12}	1^{13}	0^{15}	0^{14}
$B_3\overline{B_2}$ 10	0^{8}	0^{9}	1^{11}	1^{10}

$$G_1 = \overline{B_2}B_1 + B_2\overline{B}$$

$$\therefore G_1 = B_2 \oplus B_1$$

K-map for G_2:

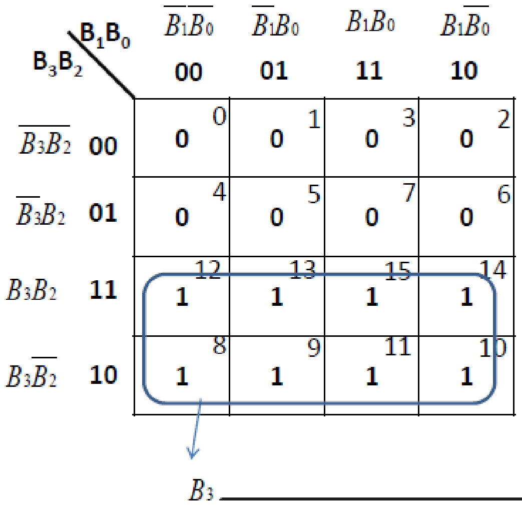


$B_3B_2 \backslash B_1B_0$		$\overline{B_1}\overline{B_0}$ 00	$\overline{B_1}B_0$ 01	B_1B_0 11	$B_1\overline{B_0}$ 10
$\overline{B_3}\overline{B_2}$	00	0^{0}	0^{1}	0^{3}	0^{2}
$\overline{B_3}B_2$	01	1^{4}	1^{5}	1^{7}	1^{6}
B_3B_2	11	0^{12}	0^{13}	0^{15}	0^{14}
$B_3\overline{B_2}$	10	1^{8}	1^{9}	1^{11}	1^{10}

$$G_2 = \overline{B_3}B_2 + B_3\overline{B_2}$$

$$\therefore G_2 = B_3 \oplus B_2$$

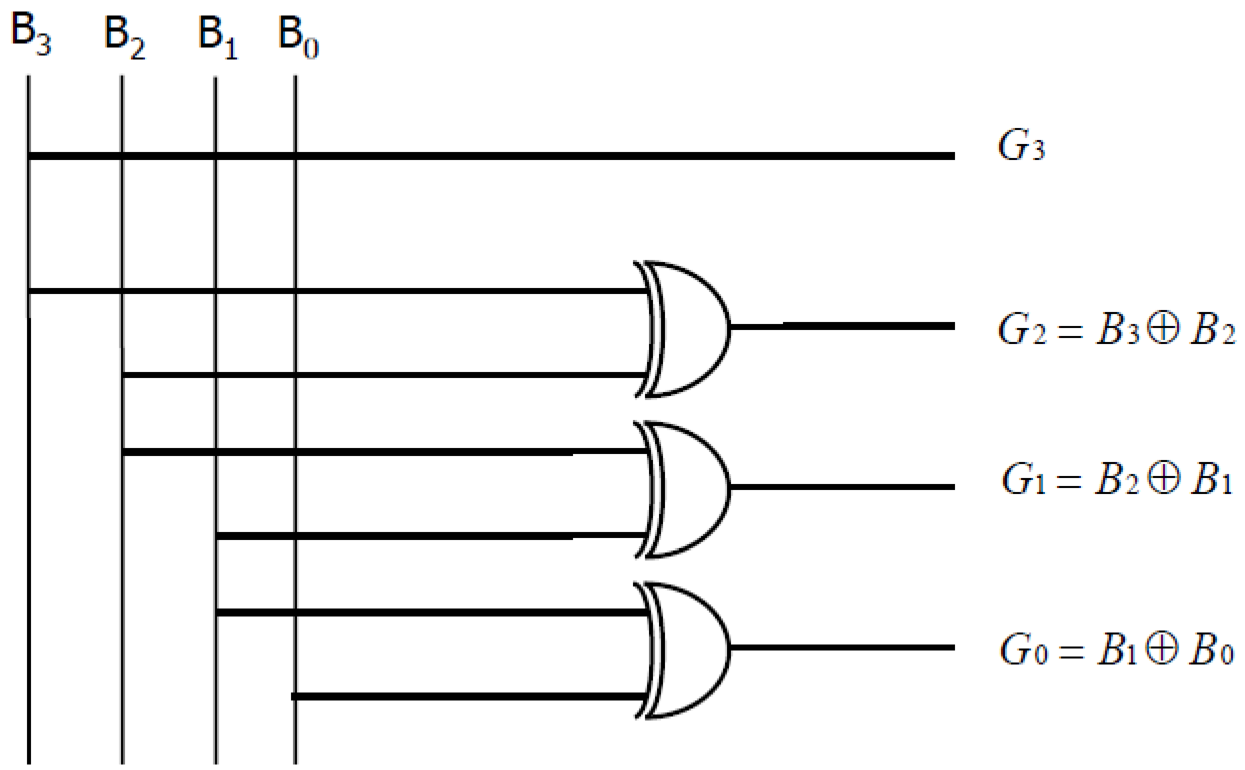
K-map for G_3:



$B_3B_2 \backslash B_1B_0$		$\overline{B_1}\overline{B_0}$ 00	$\overline{B_1}B_0$ 01	B_1B_0 11	$B_1\overline{B_0}$ 10
$\overline{B_3}\overline{B_2}$	00	0^{0}	0^{1}	0^{3}	0^{2}
$\overline{B_3}B_2$	01	0^{4}	0^{5}	0^{7}	0^{6}
B_3B_2	11	1^{12}	1^{13}	1^{15}	1^{14}
$B_3\overline{B_2}$	10	1^{8}	1^{9}	1^{11}	1^{10}

$$G_3 = B_3$$

Logic Diagram:



BCD to Excess -3 code converter

8421 BCD				XS - 3			
B_4	B_3	B_2	B_1	X_4	X_3	X_2	X_1
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

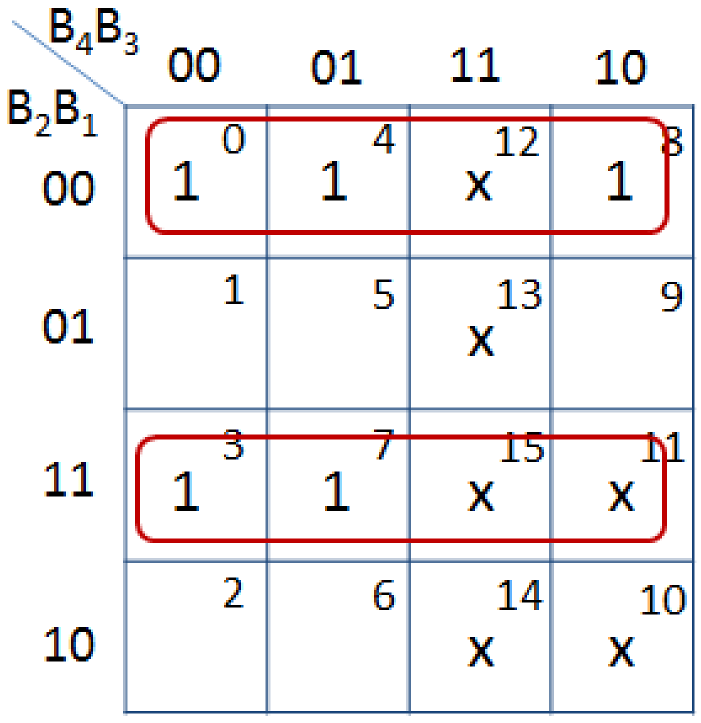
$$X_4 = \sum m(5,6,7,8,9) + d(10,11,12,13,14,15)$$

$$X_3 = \sum m(1,2,3,4,9) + d(10,11,12,13,14,15)$$

$$X_2 = \sum m(0,3,4,7,8) + d(10,11,12,13,14,15)$$

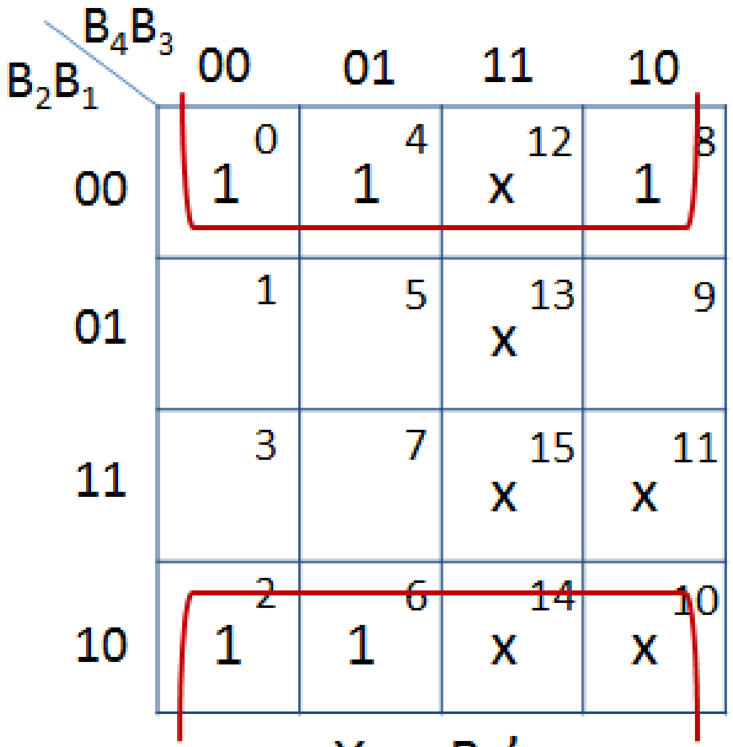
$$X_1 = \sum m(0,2,4,6,8) + d(10,11,12,13,14,15)$$

$X_2 = \sum m(0,3,4,7,8) \ + d(10,11,12,13,14,15)$ $X_1 = \sum m(0,2,4,6,8) \ + d(10,11,12,13,14,15)$



B_2B_1 \ B_4B_3	00	01	11	10
00	1 ^{0}	1 ^{4}	x ^{12}	1 ^{8}
01	^{1}	^{5}	x ^{13}	^{9}
11	1 ^{3}	1 ^{7}	x ^{15}	x ^{11}
10	^{2}	^{6}	x ^{14}	x ^{10}

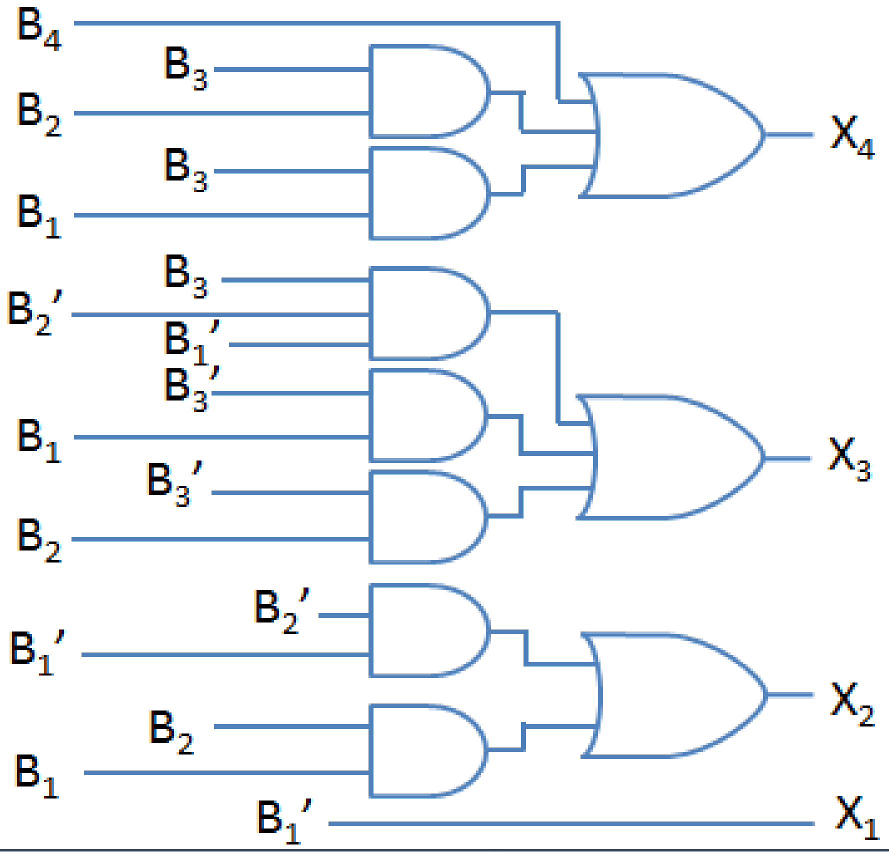
$X_2 = B_2'B_1' + B_2B_1$



B_2B_1 \ B_4B_3	00	01	11	10
00	1 ^{0}	1 ^{4}	x ^{12}	1 ^{8}
01	^{1}	^{5}	x ^{13}	^{9}
11	^{3}	^{7}	x ^{15}	x ^{11}
10	1 ^{2}	1 ^{6}	x ^{14}	x ^{10}

$X_1 = B_1'$

- Logic diagram for a BCD to Excess-3 is as follows

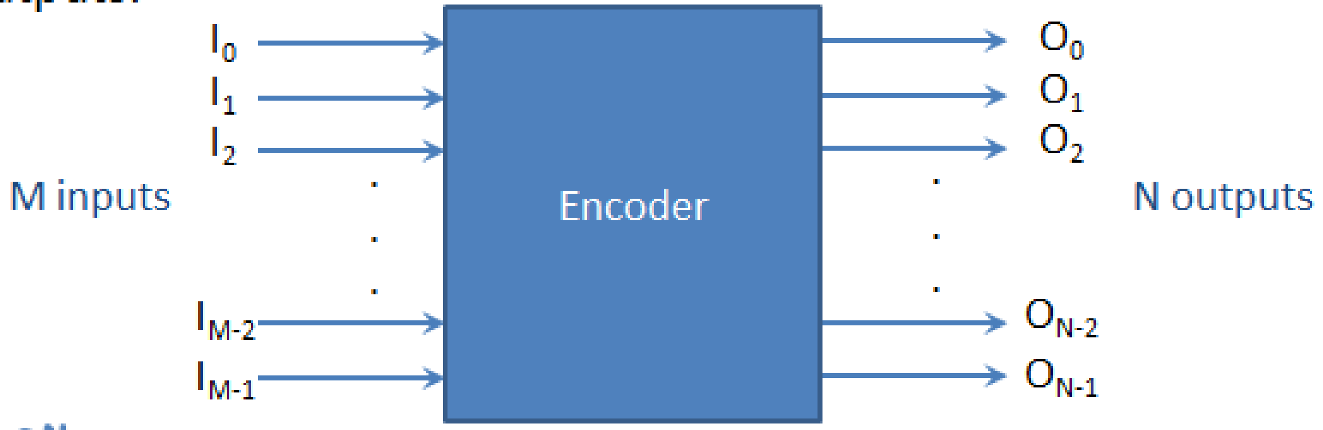


Exercise

- Binary to BCD Converter
- BCD To Gray Code Converter

Encoder

- Device to convert familiar numbers or symbols into coded format.
- It has a number of input lines, only one of which is activated at a given time, and produces an N-bit output code depending on which input is activated.
- Figure shows the block diagram of an encoder with M inputs and N outputs.



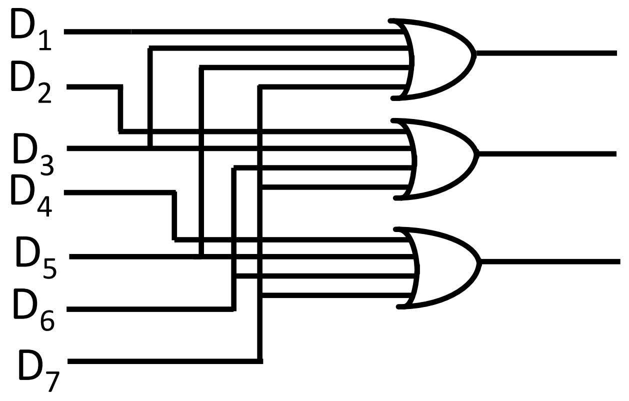
$M = 2^N$

- It perform inverse operation of the decoder.
- An encoder has “m” numbers of input lines and “n” no. of output lines.

Types of encoder:

- Priority encoder
- Decimal to BCD encoder
- Octal to binary encode
- Hexadecimal to binary encoders

Octal-to-Binary Encoder



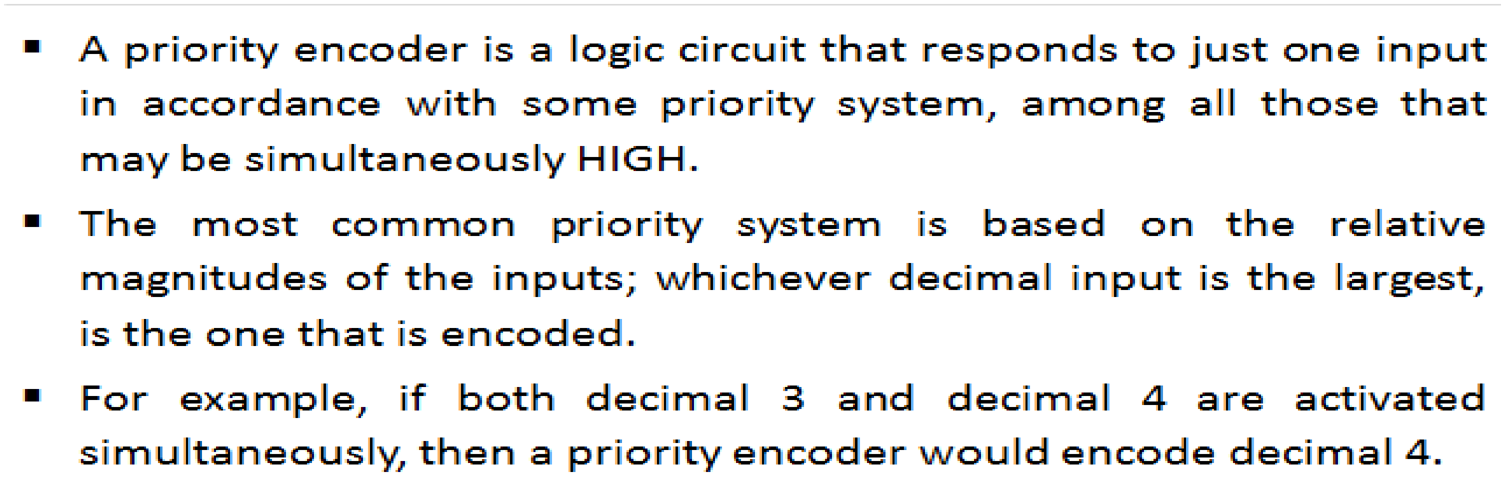
- $A_2 = D_4 + D_5 + D_6 + D_7$
- $A_1 = D_2 + D_3 + D_6 + D_7$
- $A_0 = D_1 + D_3 + D_5 + D_7$

Octal digits		Binary		
		A_2	A_1	A_0
D0	0	0	0	0
D1	1	0	0	1
D2	2	0	1	0
D3	3	0	1	1
D4	4	1	0	0
D5	5	1	0	1
D6	6	1	1	0
D7	7	1	1	1

Comparison between Encoder & Decoder

Sr. No.	Parameter	Encoder	Decoder
1	Input applied	Active input signal (original message signal)	Coded binary input
2	Output generated	Coded binary output	Active output signal (original message)
3	Input lines	2^n	n
4	Output lines	N	2^n
5	Operation	Simple	Complex
6	Applications	E-mail , video encoders etc.	Microprocessors, memory chips etc.

Priority encoder



- A priority encoder is a logic circuit that responds to just one input in accordance with some priority system, among all those that may be simultaneously HIGH.
- The most common priority system is based on the relative magnitudes of the inputs; whichever decimal input is the largest, is the one that is encoded.
- For example, if both decimal 3 and decimal 4 are activated simultaneously, then a priority encoder would encode decimal 4.

[illegible] at the same time, and the principal function of the encoder in those cases is to select the input with the highest priority.

4-Bit priority encoder

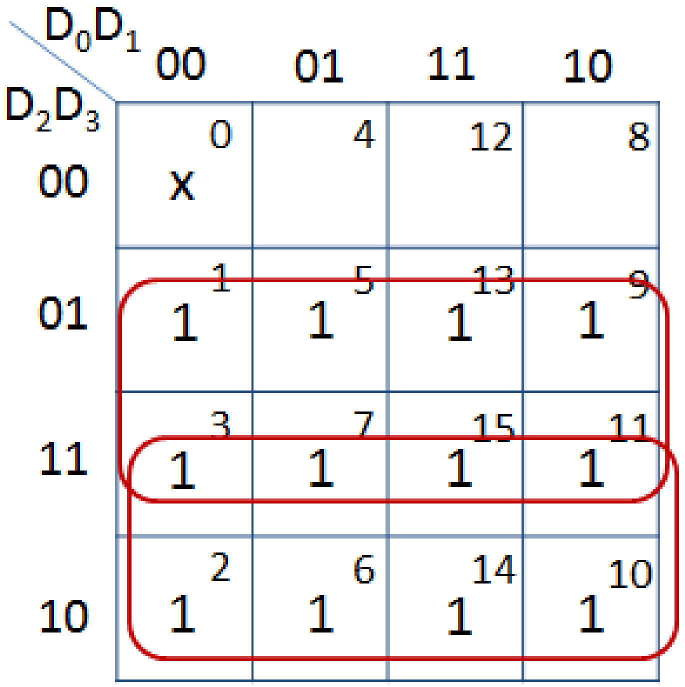
Truth Table

Inputs				Outputs		
D_0	D_1	D_2	D_3	A	B	V
0	0	0	0	x	x	0
1	0	0	0	0	0	1
x	1	0	0	0	1	1
x	x	1	0	1	0	1
x	x	x	1	1	1	1

$$A = \sum\nolimits_m (1, 2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 15) \qquad B = \sum\nolimits_m (1, 3, 4, 5, 7, 9, 11, 12, 13, 15)$$

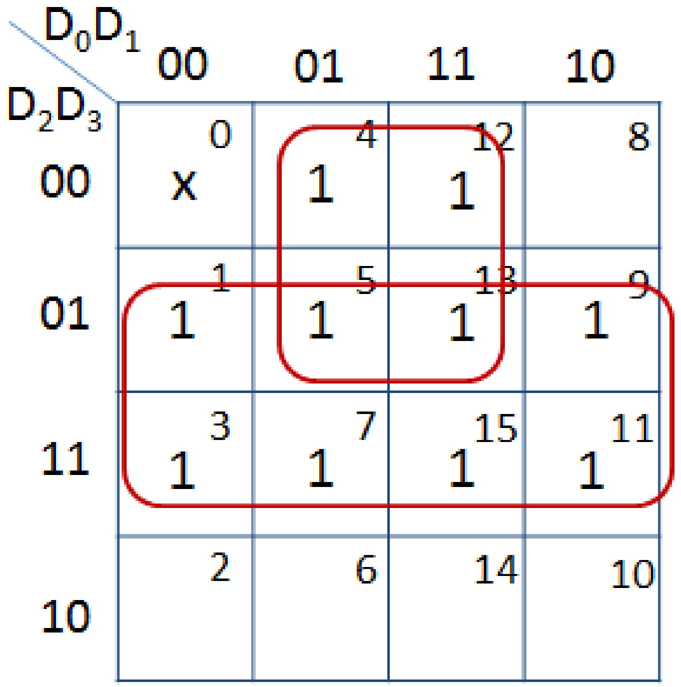
$$V = \sum\nolimits_m (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)$$

$A = \sum_m (1, 2, 3, 5, 6, 7, 9, 10, 11, 13, 14, 15)$ $\quad B = \sum_m (1, 3, 4, 5, 7, 9, 11, 12, 13, 15)$



$D_2D_3 \backslash D_0D_1$	00	01	11	10
00	0: X	4	12	8
01	1: 1	5: 1	13: 1	9: 1
11	3: 1	7: 1	15: 1	11: 1
10	2: 1	6: 1	14: 1	10: 1

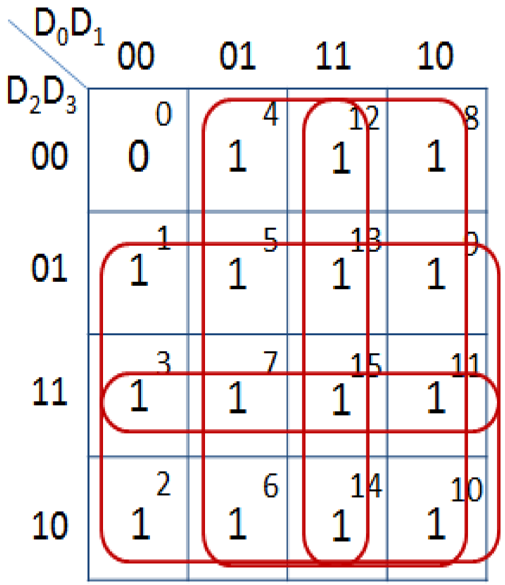
A = $D_3 + D_2$



$D_2D_3 \backslash D_0D_1$	00	01	11	10
00	0: X	4: 1	12: 1	8
01	1: 1	5: 1	13: 1	9: 1
11	3: 1	7: 1	15: 1	11: 1
10	2	6	14	10

B = $D_3 + D_2' D_1$

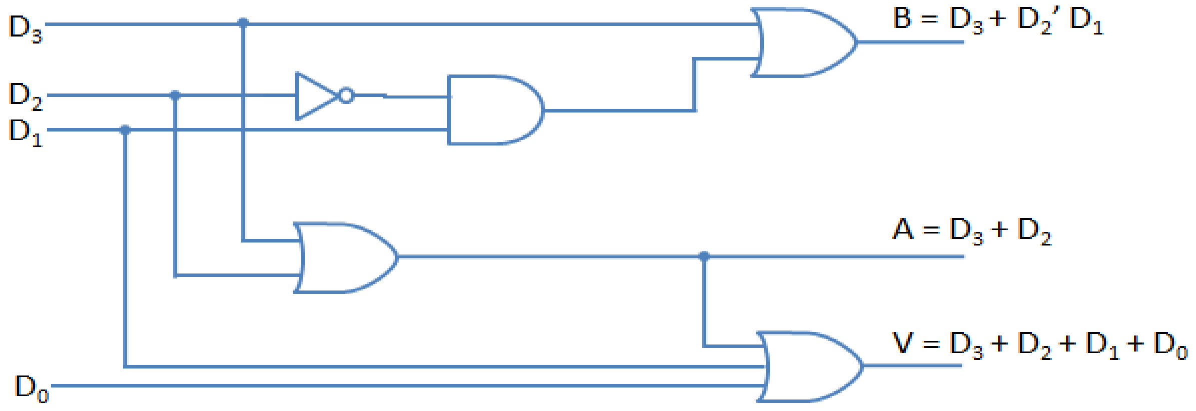
$$V = \sum_m (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15)$$



$D_2D_3 \backslash D_0D_1$	00	01	11	10
00	0 (0)	1 (4)	1 (12)	1 (8)
01	1 (1)	1 (5)	1 (13)	1 (9)
11	1 (3)	1 (7)	1 (15)	1 (11)
10	1 (2)	1 (6)	1 (14)	1 (10)

$$V = D_3 + D_2 + D_1 + D_0$$

Priority Encoder



Tabulation method(Quine Mccluskey method)

- Fundamental idea for tabulation method is combining theorem.

$$PA + PA' = P$$

- The above theorem is applied repeatedly to all adjacent pairs of terms which results in prime implicants.
- Consider the minimization of the expression

$$\begin{aligned}\Sigma_m(0,1,4,5) &= A'B'C' + A'B'C + AB'C' + AB'C \\ &= A'B'(C + C') + AB'(C + C') \\ &= A'B' + AB' \\ &= B'(A + A') \\ &= B'\end{aligned}$$

Procedure for minimization using tabulation method

1. List all the minterms.
2. Arrange all minterms in groups of the same number of 1s in their binary representation in column 1. Start with the least number of 1s group and continue with groups of increasing number of 1s.
3. Compare each term of the lowest index group with every term in the succeeding group. Whenever possible, combine the two terms being compared by means of the combining theorem. Two terms from adjacent groups are combinable, if their binary representations differ by just a single digit in the same position; the combined terms consist of the original fixed representation with the differing one replaced by a dash (-). Place a check mark (√) next to every term, which has been combined with at least one term and write the combined terms in column 2. Repeat this by comparing each term in a group of index i with every term in the group of index i + 1, until all possible applications of the combining theorem have been exhausted.

4. Compare the terms generated in step 3 in the same fashion; combine two terms which differ by only a single 1 and whose dashes are in the same position to generate a new term. Two terms with dashes in different positions cannot be combined. Write the new terms in column 3 and put a check mark next to each term which has been combined in column 2. Continue the process with terms in columns 3, 4 etc. until no further combinations are possible. The remaining *unchecked terms* constitute the *set of prime implicants* of the expression.
5. List all the prime implicants and draw the *prime implicant chart*. (The don't cares if any should not appear in the prime implicant chart).
6. Obtain the *essential prime implicants* and write the minimal expression.

Tabulation method of reduction

- Consider function $f = \sum_m (0,1,6,7,8,9,13,14,15)$

Step – 1: List all minterms

Minterms	Binary Designation
0	0 0 0 0
1	0 0 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
13	1 1 0 1
14	1 1 1 0
15	1 1 1 1

Step – 2: Arrange all minterms in groups of same number of 1s

Column 1		
Index	**Minterms**	**Binary Designation**
Index 0	0	0 0 0 0
Index 1	1	0 0 0 1
	8	1 0 0 0
Index 2	6	0 1 1 0
	9	1 0 0 1
Index 3	7	0 1 1 1
	13	1 1 0 1
	14	1 1 1 0
Index 4	15	1 1 1 1

Step – 3: Compare each term of the lowest index group with every term in the succeeding group till no change

Column 1		
Index	Minterms	Binary Designation
Index 0	0	0 0 0 0 √
Index 1	1 8	0 0 0 1 √ 1 0 0 0 √
Index 2	6 9	0 1 1 0 √ 1 0 0 1 √
Index 3	7 13 14	0 1 1 1 √ 1 1 0 1 √ 1 1 1 0 √
Index 4	15	1 1 1 1 √

Column 2	
Pairs	A B C D

Step – 4: Compare the terms generated in step 3 in the same fashion until no further combinations are possible

Pairs	A B C D	
0, 1	$\underline{0}$ 0 0 _	√
0, 8	_ 0 0 $\underline{0}$	√
1, 9	_ 0 0 $\underline{1}$	√
8, 9	$\underline{1}$ 0 0 _	√
6, 7	$\underline{0}$ 1 1 _	√
6, 14	_ 1 1 $\underline{0}$	√
9,13	1 _ 0 1	S
7, 15	_ 1 1 $\underline{1}$	√
13, 15	1 1 _ 1	R
14, 15	$\underline{1}$ 1 1 _	√

Quads	A B C D
	Q
---	---
	P

Exercise

- Reduce the function to simplest form using tabulation method

$$f = \sum_m (1,2,3,5,6,7,8,9,12,13,15)$$

$$BD + AC' + A'C + C'D$$

$$BD + AC' + A'C + A'D$$