BOOLEAN ALGEBRA AND LOGIC GATES

Boolean Algebra

The digital signals are discrete in nature and can assume one of the two values 0 or 1. A number system based on these two digits is known as binary number system.

In the middle of 19th century, an English mathematician George Boole developed rules for manipulations of binary variables, known as Boolean algebra. This is the basis of all digital systems.

Binary variables can be represented by a letter symbol such as A, B, X, Y..... The variable can have only one of the two possible values at any time viz. 0 or 1.

BASIC THEOREMS AND PROPERTIES OF BOOLEAN ALGEBRA

DUALITY

One part may be obtained from the other if the binary operators (AND and OR) and the identity elements (0 and 1) are interchanged. This important property of Boolean algebra is called the duality principle and states that every algebraic expression deducible from the postulates of Boolean Algebra remains valid if the operators and identity elements are interchanged.

If the dual of an algebraic expression is desired, we simply interchange OR and AND operators and replace 1's by 0's and 0's by 1's.

	$A+0=A$	$A.1=A$
	$A+\bar{A}=1$	$A.\bar{A}=0$
	$A+A=A$	$A.A=A$
	$A+1=1$	$A.0=0$
	$(\bar{\bar{A}})=A$	
Commutative	$A+B=B+A$	$AB=BA$
Associative	$A+(B+C)=(A+B)+C$	$A(BC)=(AB)C$
Distributive	$A(B+C)=AB+AC$	$A+BC=(A+B)(A+C)$
Absorption	$A+AB=A$	$A(A+B)=A$
	$A+\bar{A}B=A+B$	$A(\bar{A}+B)=AB$
	$AB+A\bar{B}=A$	$(A+B).(A+\bar{B})=A$
	$AB+\bar{A}C=(A+C)(\bar{A}+B)$	$(A+B)(\bar{A}+C)=AC+\bar{A}B$
	$AB+\bar{A}C+BC=AB+\bar{A}C$	$(A+B)(\bar{A}+C)(B+C)=(A+B)(\bar{A}+C)$

MORGAN'S THEOREM

$\overline{AB} = \bar{A} + \bar{B}$

$\overline{ABC\ldots} = \bar{A} + \bar{B} + \bar{C} \ldots$

$\overline{A+B} = \bar{A} \cdot \bar{B}$

$\overline{A+B+C+\ldots} = \bar{A} \cdot \bar{B} \cdot \bar{C} \cdot \ldots$

1) complement of sum of variables equals the product of complement of individual variables.

$\overline{A+B} = \bar{A} \cdot \bar{B}$

2) Complement of a product equals the sum of the complements.

$\overline{A \cdot B} = \bar{A} + \bar{B}$

$AB + \bar{A}C + BC = (A + \bar{A})$

$= AB + ABC + \bar{A}C + \bar{A}BC$

$= AB(1+C) + \bar{A}C(1+B)$

$= AB + \bar{A}C$

~~$A + \bar{A}B$~~ $= A + \bar{A}B + A\bar{A} + A\bar{A}$

$= A \cdot A + \bar{A}B + A\bar{A} + A\bar{A}$

$= A(A + \bar{A}) + B(\bar{A} + \bar{A}B)$

$= A + \bar{A}B$

$AB + \bar{A}C = A\bar{A} + AB + \bar{A}C$

$= A(\bar{A} + B) + \bar{A}C$

$= AB + ABC + \bar{A}C + \bar{A}BC$

$= AB + \bar{A}C + BC(A + \bar{A})$

$= AB + \bar{A}C + BC + A\bar{A}$

$= B(A+C) + \bar{A}(A+C)$

$= (\bar{A} + B)(A+C)$ Ans

$AB + \bar{A}D + \bar{A}\bar{B} + A\bar{D} + B\bar{C} = \overline{AB} \cdot \overline{\bar{A}D} \cdot \overline{\bar{A}\bar{B}} \cdot \overline{A\bar{D}} \cdot \overline{B\bar{C}}$

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

$= (0 + A\bar{B}D + 0 + 0 + 0 + \bar{A}B\bar{D} + 0 + 0 + 0 + 0)(\bar{B} + C)$

$= A\bar{B}D + A\bar{B}CD + \bar{A}BCD + \bar{A}B\bar{D} + C(A\bar{B}D + \bar{A}B\bar{D} + \bar{A}BCD)$

$= (A\bar{A} + A\bar{B} + \bar{A}\bar{D} + B\bar{D})(A\bar{A} + \bar{A}B + AD + BD)(\bar{B} + C)$

$= A\bar{B}\bar{A}B + A\bar{B}BD + A\bar{B}BD + \bar{A}B\bar{D} + AA\bar{D} + A\bar{D}BD + \bar{A}\bar{B}\bar{D} + \bar{A}BBD + AD\bar{A}D + B\bar{D}$

~~Code~~ Conversion

Binary to Gray

1) Identify inputs and outputs and assign a symbol
   Inputs (binary) - $B_3\ B_2\ B_1\ B_0$
   Outputs (gray) - $G_3\ G_2\ G_1\ G_0$

2) ~~Decimal~~ Draw truth table

3)

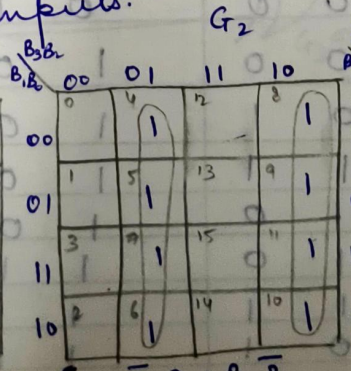
Decimal	I/Ps				O/Ps			
	B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

3) Reduce the logical expressions for each output. output in terms of inputs.

G_3

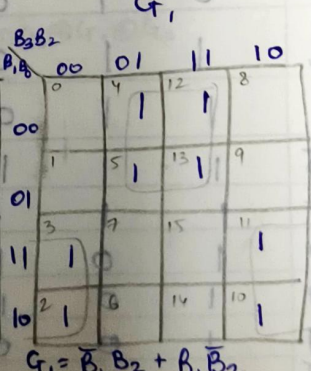
$G_3 = B_3$

G_2



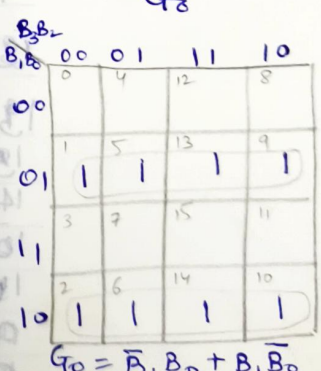
$G_2 = \bar{B}_3 B_2 + B_3 \bar{B}_2$
$= B_3 \oplus B_2$

G_1

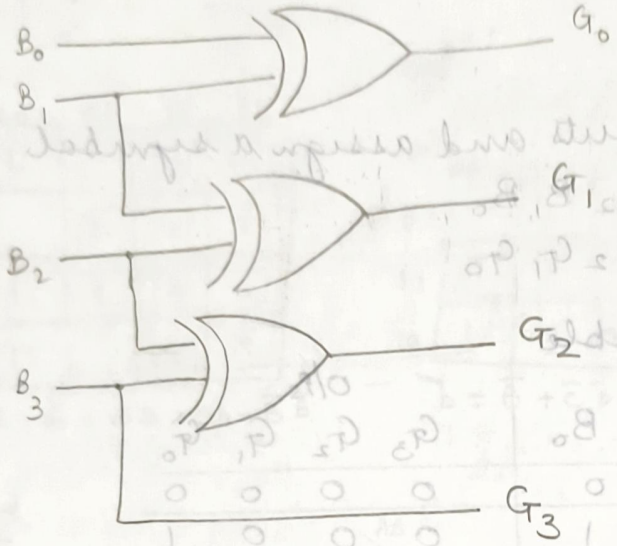


$G_1 = \bar{B}_1 B_2 + B_1 \bar{B}_2$
$= B_1 \oplus B_2$

G_0



$G_0 = \bar{B}_1 B_0 + B_1 \bar{B}_0$
$= B_1 \oplus B_0$



Gray to binary

1) Inputs ~~binary~~ Gray $G_3\ G_2\ G_1\ G_0$

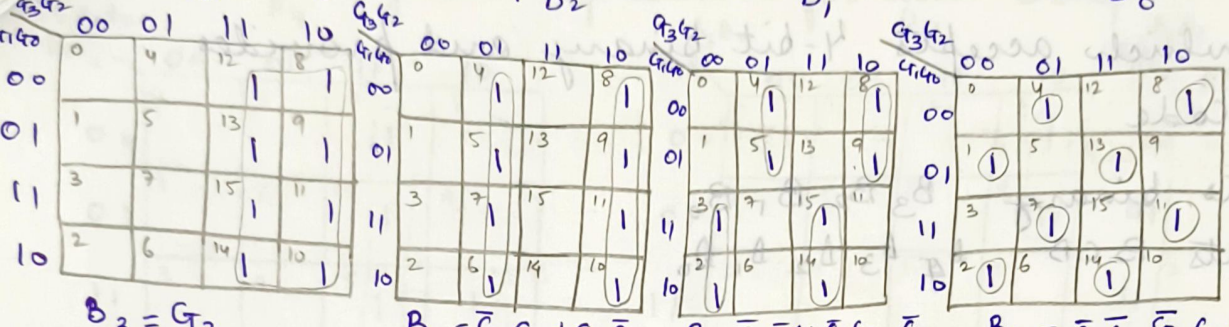
Outputs Binary $B_3\ B_2\ B_1\ B_0$

2) Truth table

Decimal	Gray				Binary			
	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
1	0	0	0	1	0	0	0	1
3	0	0	1	1	0	0	1	0
2	0	0	1	0	0	0	1	1
6	0	1	1	0	0	1	0	0
7	0	1	1	1	0	1	0	1
5	0	1	0	1	0	1	1	0
4	0	1	0	0	0	1	1	1
12	1	1	0	0	1	0	0	0
13	1	1	0	1	1	0	0	1
15	1	1	1	1	1	0	1	0
14	1	1	1	0	1	0	1	1
10	1	0	1	0	1	1	0	0
11	1	0	1	1	1	1	0	1
9	1	0	0	1	1	1	1	0
8	1	0	0	0	1	1	1	1

Reduce logical expression for each output.

B_3 B_2 B_1 B_0



$B_3 = G_3$

$B_2 = \bar{G}_3 G_2 + G_3 \bar{G}_2$

$= G_3 \oplus G_2$

$B_1 = \bar{G}_3 \bar{G}_2 G_1 + \bar{G}_3 G_2 \bar{G}_1 + G_3 G_2 G_1 + G_3 \bar{G}_2 \bar{G}_1$

$= \bar{G}_3 (\bar{G}_2 G_1 + G_2 \bar{G}_1) + G_3 (G_2 G_1 + \bar{G}_2 \bar{G}_1)$

$= G_1 \oplus G_2 \oplus G_3$

$B_0 = \bar{G}_3 \bar{G}_2 \bar{G}_1 G_0 + \bar{G}_3 \bar{G}_2 G_1 \bar{G}_0 + \bar{G}_3 G_2 \bar{G}_1 \bar{G}_0 + \bar{G}_3 G_2 G_1 G_0 + G_3 G_2 \bar{G}_1 G_0 + G_3 G_2 G_1 \bar{G}_0 + G_3 \bar{G}_2 \bar{G}_1 \bar{G}_0 + G_3 \bar{G}_2 G_1 G_0$

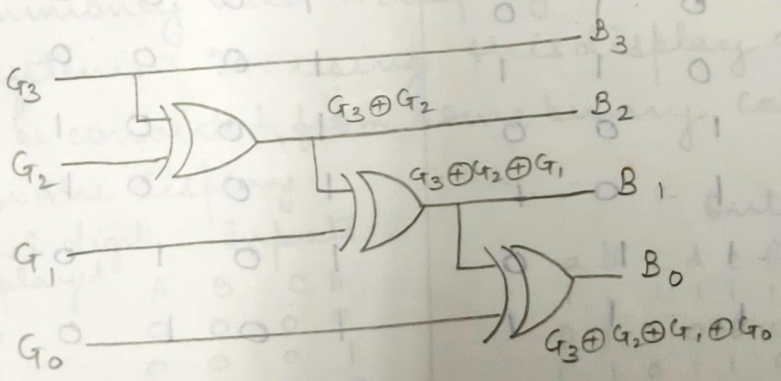
$= \bar{G}_3 \bar{G}_2 (\bar{G}_1 G_0 + G_1 \bar{G}_0) + \bar{G}_3 G_2 (\bar{G}_1 \bar{G}_0 + G_1 G_0) + G_3 G_2 (\bar{G}_1 G_0 + G_1 \bar{G}_0) + G_3 \bar{G}_2 (\bar{G}_1 \bar{G}_0 + G_1 G_0)$

$= \bar{G}_3 \bar{G}_2 (G_1 \oplus G_0) + \bar{G}_3 G_2 (\overline{G_1 \oplus G_0}) + G_3 G_2 (G_1 \oplus G_0) + G_3 \bar{G}_2 (\overline{G_1 \oplus G_0})$

$= (G_1 \oplus G_0)(\bar{G}_3 \bar{G}_2 + G_3 G_2) + (\overline{G_1 \oplus G_0})(\bar{G}_3 G_2 + G_3 \bar{G}_2)$

$= (G_1 \oplus G_0)(\overline{G_3 \oplus G_2}) + (\overline{G_1 \oplus G_0})(G_3 \oplus G_2)$

$= G_0 \oplus G_1 \oplus G_2 \oplus G_3$



COMBINATIONAL LOGIC DESIGN

A combinational circuit consists of logic gates whose outputs at any time are determined from only the present combination of inputs. A combinational circuit performs an operation that can be specified logically by a set of Boolean functions.

A combinational circuit consists of input variables, logic gates, and output variables. Combinational logic gates react to the values of the signals at their inputs and produce the value of the output signal, transforming binary information from the given input data to a required output data.

Design procedure

The design procedure involves the following steps:-

1) From the specifications of the circuit, determine the required number of inputs and assign a symbol to each.
2) Derive the truth table that defines the required relationship between inputs and outputs.
3) obtain the simplified Boolean functions for each output as a function of the input variables.
4) Draw the logic diagram and verify the correctness of the design (manually or by simulation).

The output binary functions are simplified by any available method, such as algebraic manipulation, K-map method or Tabular (Quine McCluskey) method.

Design Examples

Basic Adders

Adders are important in systems where numerical data are processed.

Half adder

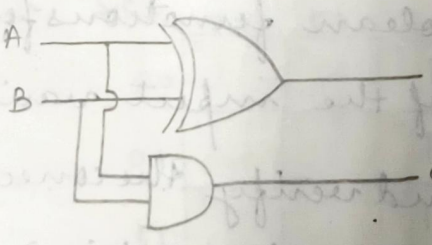
A logic circuit for the addition of two one-bit numbers is refered to as a half adder.

A and B are the two inputs. S(SUM) and C(CARRY) are the two outputs.

Inputs		Outputs	
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$S = \bar{A}B + A\bar{B} = A \oplus B$

$C = AB$



Full adder

A half adder has only two inputs and there is no provision to add a carry coming from the lower order bits when multibit addition is performed.

For this purpose, a third bit input terminal is added and this circuit is used to add A_n, B_n and C_{n-1}, where A_n and B_n are n^{th} order bits of numbers A and B respectively and C_{n-1} is the carry generated from the addition of $(n-1)^{th}$ order bits. This circuit is refered to as full adder.

Inputs			outputs	
A_n	B_n	C_{n-1}	S_n	C_n
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

K-map

S_n

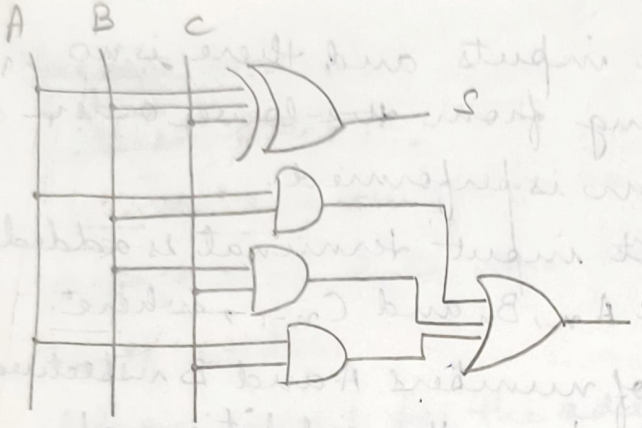
C_{n-1} \ A_nB_n	00	01	11	10
0	0	1	0	1
1	1	0	1	0

$S_n = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

C_n

C_{n-1} \ A_nB_n	00	01	11	10
0	0	0	1	0
1	0	1	1	1

$C_n = AB + AC + BC$



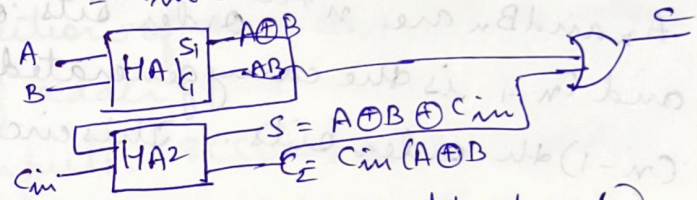
Full adder using

Carry of FA = $\Sigma m(3,5,6,7)$

$= \bar{A}BC_{in} + A\bar{B}C_{in} + ABC_{in} + ABC_{in}$

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

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



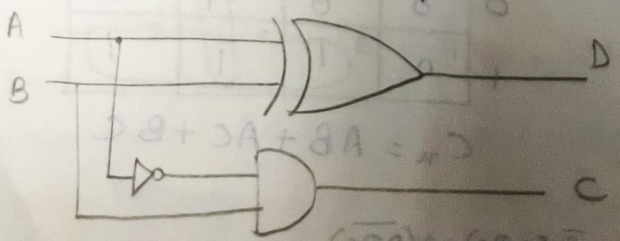
Half Subtractor

A logic circuit for the subtraction of BC subtrahend) from A (minuend) where A and B are 1-bit numbers is referred to as a half subtractor.

Inputs		Outputs	
A	B	D	C (Borrow)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$D = \bar{A}B + A\bar{B} = A \oplus B$

$C = \bar{A}B$

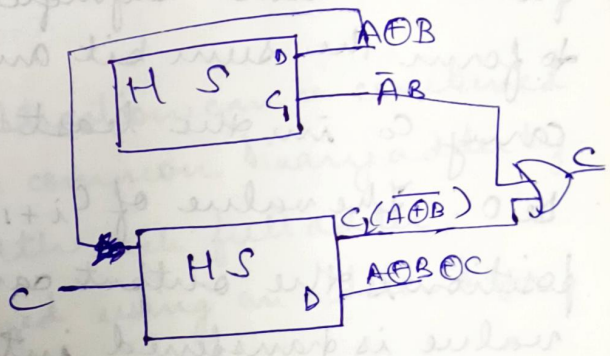


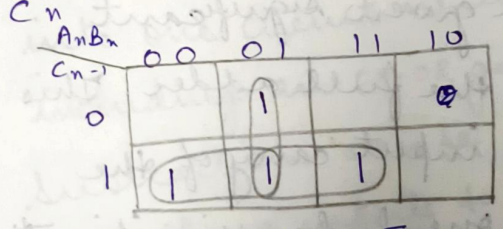
ull Subtractor

Just like a full adder, we require a full subtractor circuit for performing multibit subtraction wherein a borrow from a previous bit position may also be there.

Inputs			Outputs	
A_n	B_n	C_{n-1}	D_n	C_n
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

$D_n = A \oplus B \oplus C$





$C = \bar{A}B + BC + \bar{A}C$

$= \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + ABC$

$= C(\bar{A}\bar{B} + AB) + \bar{A}B(\bar{C} + C)$

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

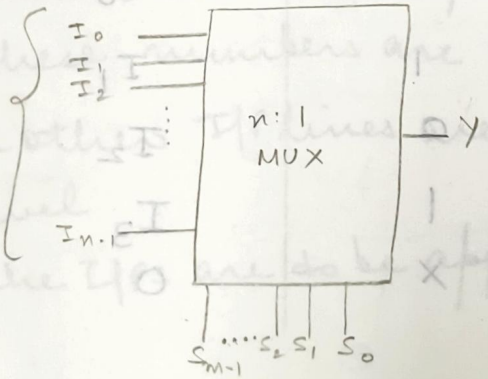
method.

Multiplexers and Demultiplexers

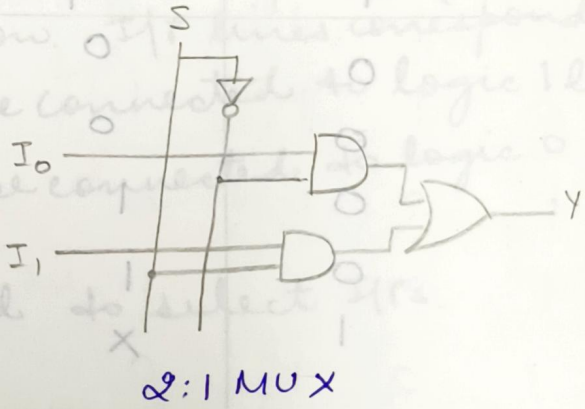
Multiplexer (Data Selector)

Multiplexer (or data selector) circuit is a combinational circuit that gates one of several inputs to a single output. The selection of a particular I/P line is controlled by a set of select lines. Normally, there are n input lines and m selection lines whose bit combinations determine which I/P is selected. $2^m = n$

Normally, a strobe (or enable) input (G) is incorporated which helps in cascading and it is generally active-low, which means it performs its intended operation when it is LOW.



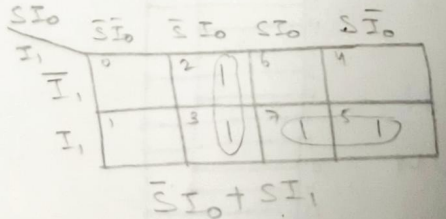
Block Diagram of MUX



2:1 MUX

S	Y
0	I_0
1	I_1

S	I_0	I_1	Y
0	0	0	0
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	1

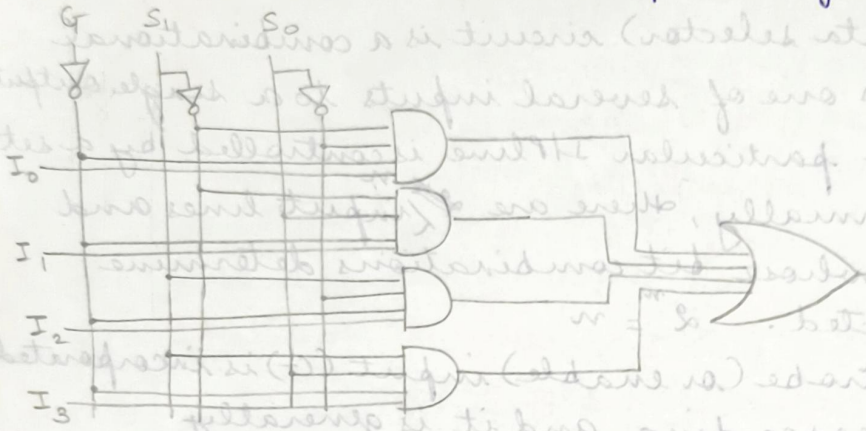


$\bar{S}I_0 + SI_1$

$Y = \bar{S}I_0 + SI_1$

$\Rightarrow$ I_0 selected when $S=0$, I_1 selected when $S_0=1$

A 4:1 MUX with strobe input is given as follows:



Enable I/P	Select I/Ps		Output
G	S_1	S_0	Y
0	0	0	I_0
0	0	1	I_1
0	1	0	I_2
0	1	1	I_3
1	X	X	0

$$Y = (\overline{S}_1\,\overline{S}_0\,I_0 + \overline{S}_1\,S_0\,I_1 + S_1\,\overline{S}_0\,I_2 + S_1\,S_0\,I_3)\cdot\overline{G}$$

COMBINATIONAL LOGIC DESIGN USING MULTIPLEXERS

Use of multiplexers offers the following advantages

1) Simplification of logic expression is not required
2) It minimises the IC package count.
3) Logic design is simplified.

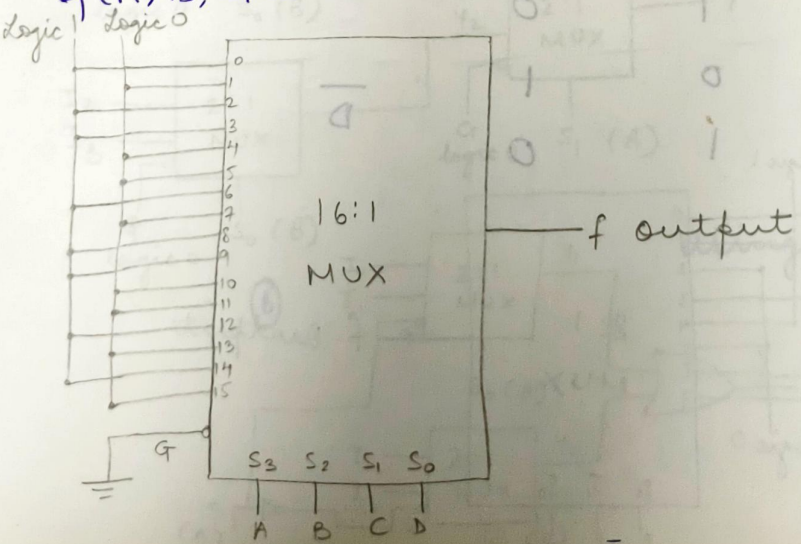
For using the MUX as a logic element, either the truth table or one of the canonical forms of logic expression must be available.

The design procedure is as follows:-

1) Identify the decimal number corresponding to each minterm in the expression. I/P lines corresponding to these numbers are to be connected to logic 1 level.
2) All other I/P lines are to be connected to logic 0 level.
3) The I/Ps are to be applied to select I/Ps.

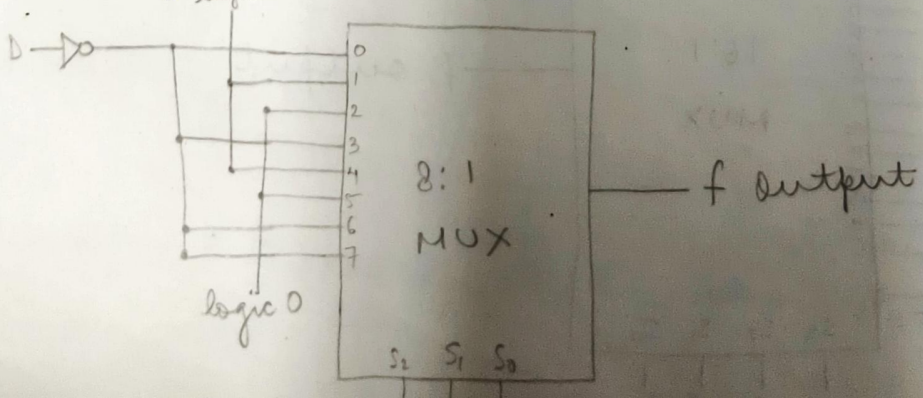
1. Implement the expression using a multiplexer.

$$f(A, B, C, D) = \sum m(0, 2, 3, 6, 8, 9, 12, 14)$$



2. Implement using 8:1 MUX

Inputs				Output	
	A	B	C	D	Y
0	0	0	0	0	1
1	0	0	0	1	0
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	$\overline{D}$
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	$\overline{D}$



A four variable truth table or logic expression can be realized by using 8:1 MUX. For this partition the truth table as shown. Here, the I/Ps A, B and C are connected to S_2, S_1 and S_0 respectively. Observe the relationship between I/p D and O/p Y for each group of two rows. There are four possible values of Y and these are 0, 1, D and $\bar{D}$.

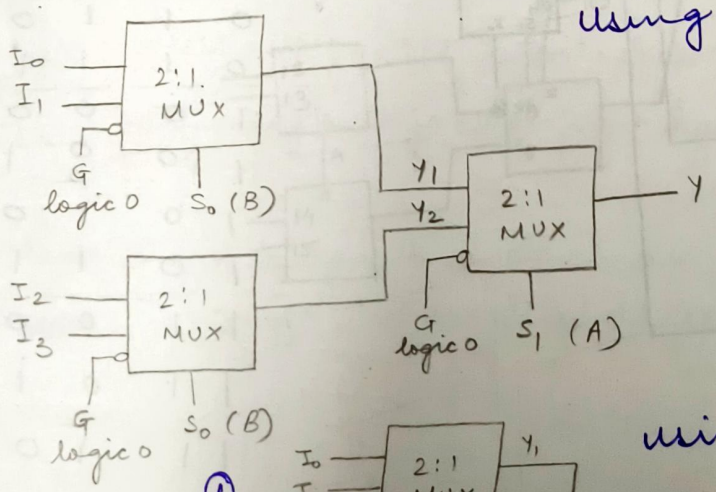
MUX trees

Since 16:1 are the largest available ICs, therefore to meet the larger I/P needs there should be a provision for expansion. This is achieved with the help of enable / strobe I/Ps and MUX stacks or trees are designed.

1. Design 4:1 MUX using 2:1 MUX

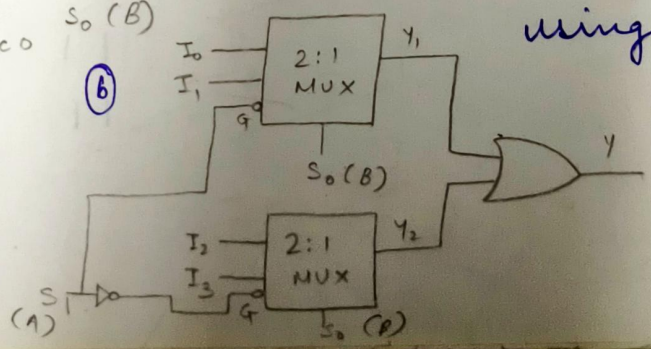
using 2:1 MUX only

(a)

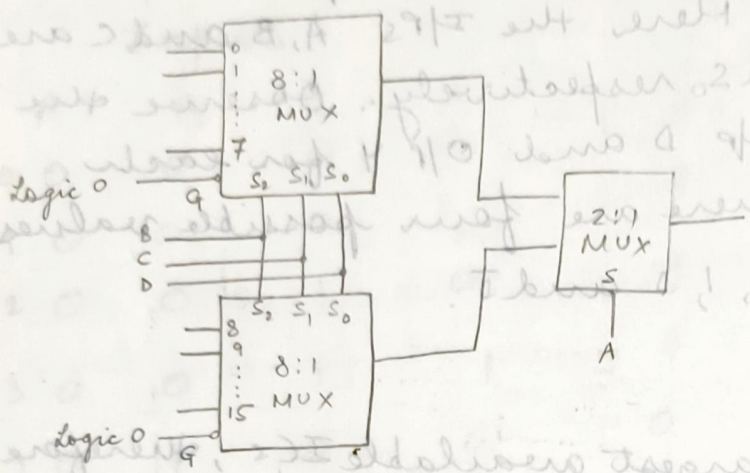


using 2:1 MUX and OR gate

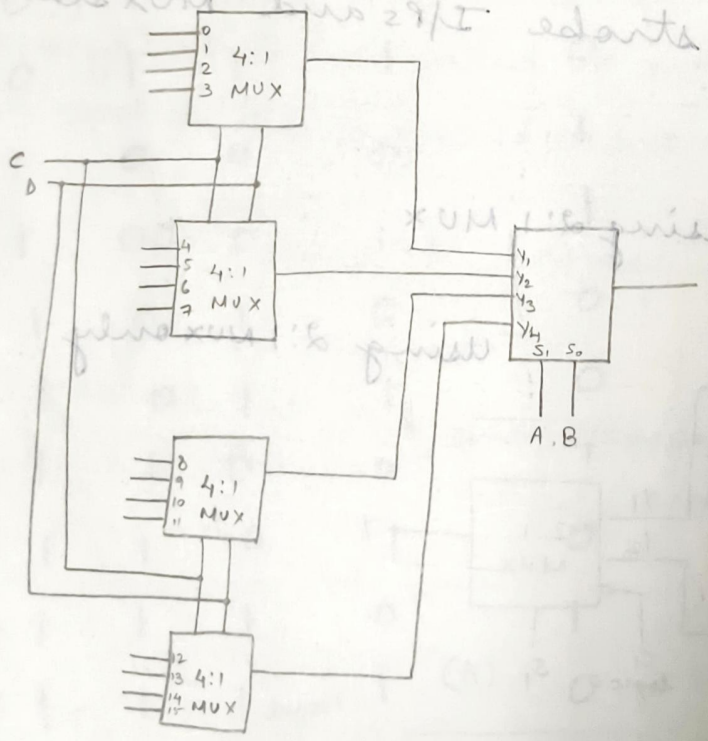
(b)



2. Design ~~32~~ 16:1 MUX using 8:1 MUX

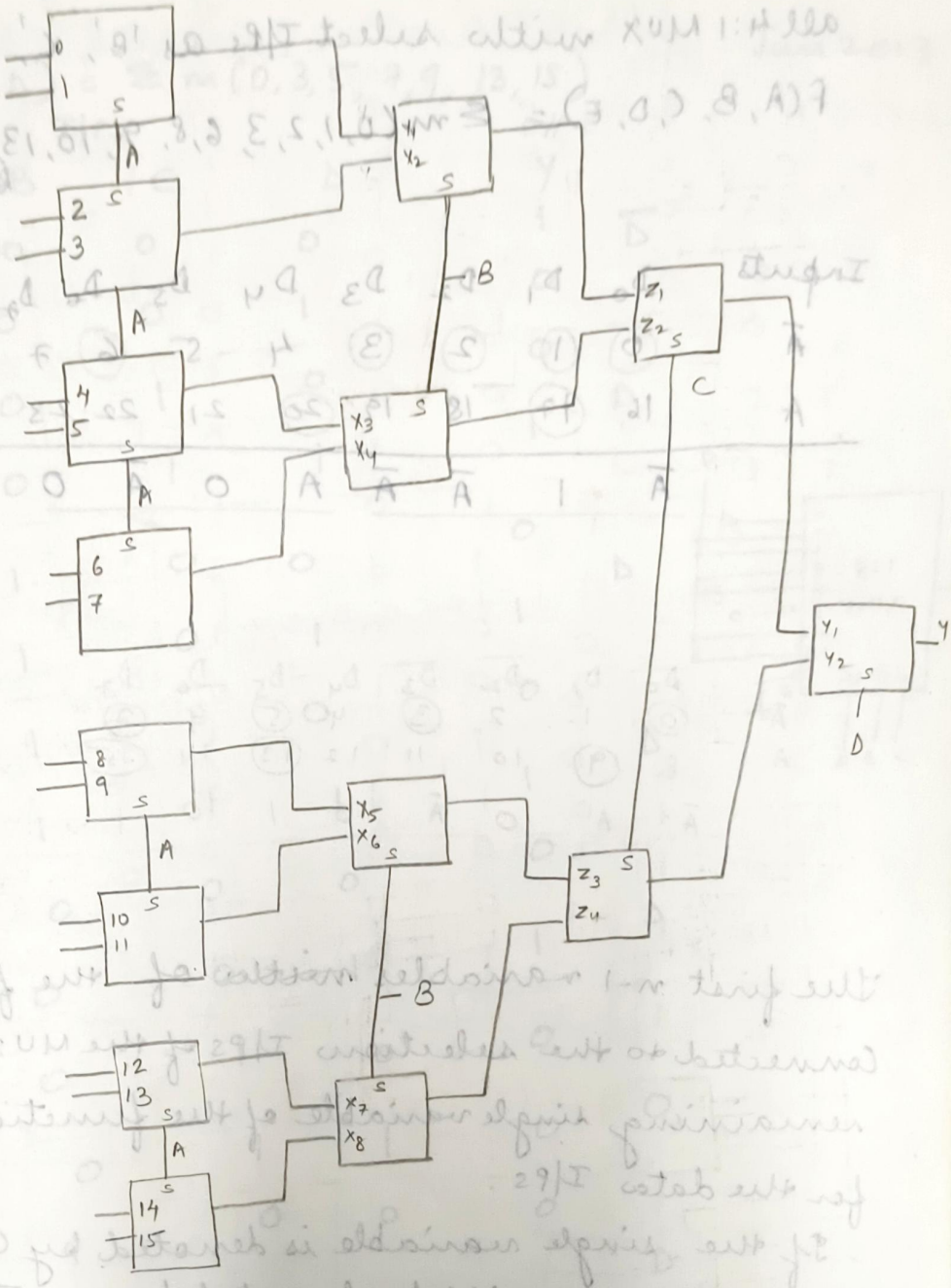


3. Design ~~8~~ 16:1 MUX using 4:1 MUX



A	B	C	D
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

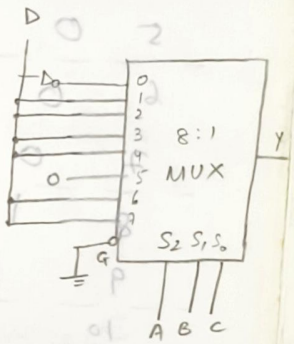
4. Design 16:1 MUX using 2:1 MUX



6. Implement the following using only one 8:1 MUX and few gates.

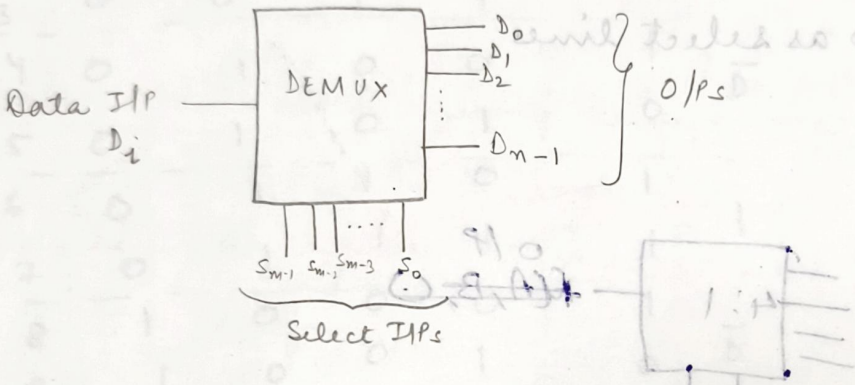
$f(A, B, C, D) = \Sigma m(0, 3, 5, 7, 9, 13, 15)$ Jun 2013

	I/Ps A	B	C	D	O/P Y	
0	0	0	0	0	1	$\overline{D}$
1	0	0	0	1	0	
2	0	0	1	0	0	D
3	0	0	1	1	1	
4	0	1	0	0	0	D
5	0	1	0	1	1	
6	0	1	1	0	0	D
7	0	1	1	1	1	
8	1	0	0	0	0	D
9	1	0	0	1	1	
10	1	0	1	0	0	0
11	1	0	1	1	0	
12	1	1	0	0	0	D
13	1	1	0	1	1	
14	1	1	1	0	0	D
15	1	1	1	1	1	



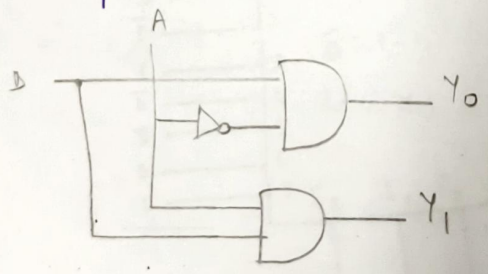
Demultiplexers (Data Distributor)

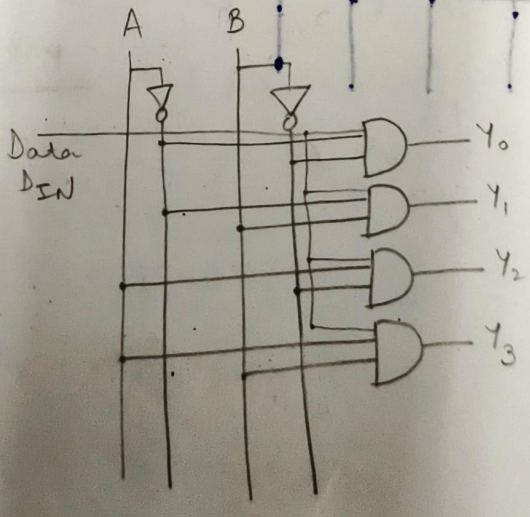
The demultiplexer performs the reverse operation of a MUX. It accepts a single I/P and distributes it over several O/Ps. The select I/P code determines to which O/P the data will be transmitted.



The number of O/P lines is n and the number of select lines is m, where $n = 2^m$. The data I/P D_i will appear on the O/P line selected by the select I/P.

For example, if the decimal equivalent of the select I/P is 4, then the data will appear on D_4 output line.



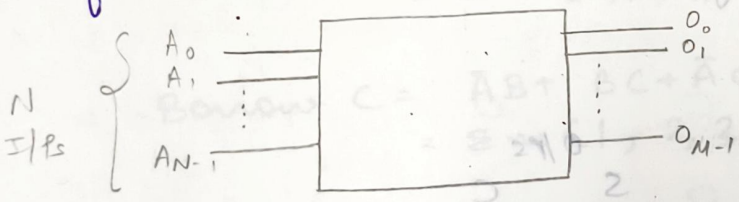


Demultiplexer trees

Since 4-line-to-16 line decoders are the largest available circuits in ICs, to meet the larger I/P need there should be a provision for expansion. This is made possible by using enable input terminal.

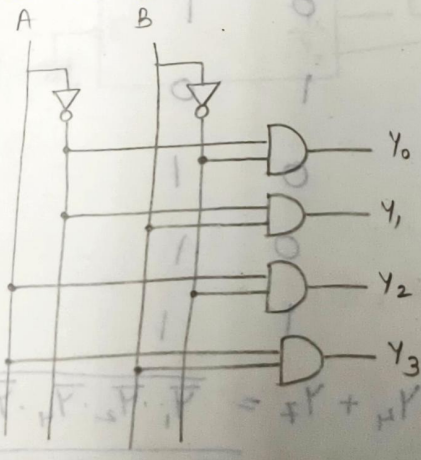
Decoder

A decoder is a digital logic circuit that converts N-bit binary input code into M output lines. Here each output line will be activated for only one of the possible combination of inputs.

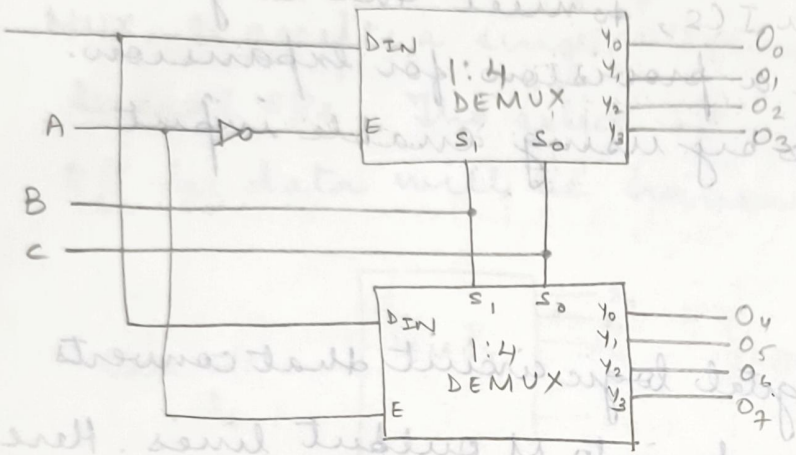


Comparison between Demux and Decoder

Decoder doesn't have D_{IN} (data I/P) line, whereas Demux has a data I/P line.



1. Cascade Demux, build 1:8 Demux using 1:4 Demux.



2. Design full adder using 3:8 decoder with active low O/Ps and NAND gates Dec 2013

Full adder

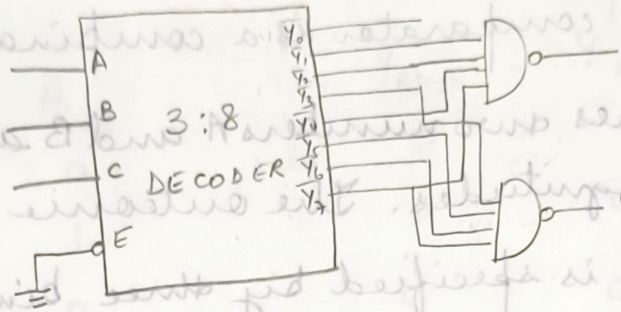
Truth table

	I/Ps			O/Ps	
	A	B	C	S	C
0	0	0	0	0	0
1	0	0	1	1	0
2	0	1	0	1	0
3	0	1	1	0	1
4	1	0	0	1	0
5	1	0	1	0	1
6	1	1	0	0	1
7	1	1	1	1	1

$$S = Y_1 + Y_2 + Y_4 + Y_7 = \overline{\overline{Y_1} \cdot \overline{Y_2} \cdot \overline{Y_4} \cdot \overline{Y_7}}$$

$$C = Y_3 + Y_5 + Y_6 + Y_7 = \overline{\overline{Y_3} \cdot \overline{Y_5} \cdot \overline{Y_6} \cdot \overline{Y_7}}$$

outputs are active low



3. Design full subtractor using 3:8 decoder and some gates.

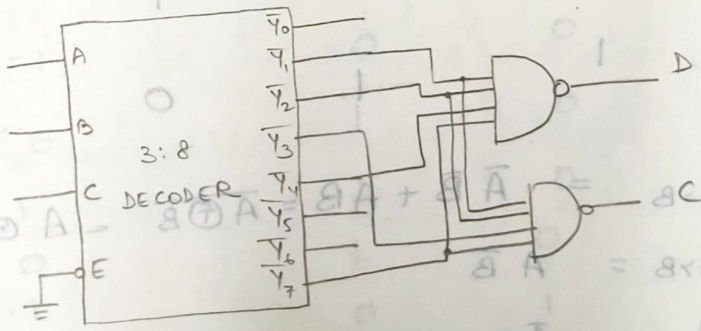
Full Subtractor

Difference $D = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

$= \Sigma m(1,2,4,7) = \overline{\bar{Y}_1 \cdot \bar{Y}_2 \cdot \bar{Y}_4 \cdot \bar{Y}_7}$

Borrow $C = \bar{A}B + BC + \bar{A}C$

$= \Sigma m(1,2,3,7) = \overline{\bar{Y}_1 \cdot \bar{Y}_2 \cdot \bar{Y}_3 \cdot \bar{Y}_7}$



+ $H_1 A_0 B_1 B_0$

MAGNITUDE COMPARATOR

A magnitude comparator is a combinational circuit that compares two numbers A and B and determines their relative magnitudes. The outcome of the comparison is specified by three binary variables that indicate whether $A > B$, $A = B$ or $A < B$.

Single bit Magnitude comparator

Inputs		Outputs		
A	B	$Y_{A=B}$	$Y_{A>B}$	$Y_{A<B}$
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

$$Y_{A=B} = \bar{A}\bar{B} + AB = \overline{A \oplus B} = A \odot B$$

$$Y_{A>B} = A\bar{B}$$

$$Y_{A<B} = \bar{A}B$$

$$Y_{A>B} = A_1\bar{B}_1 + x_1 A_0\bar{B}_0$$
$$= A_1\bar{B}_1 + (A_1B_1 + \bar{A}_1\bar{B}_1)A_0\bar{B}_0$$

$$Y_{A<B} = \bar{A}_1 B_1 + \bar{A}_1\bar{A}_0\bar{B}_1 B_0 + A_1\bar{A}_0 B_1 B_0$$
$$= \bar{A}_1 B_1 + (\bar{A}_1\bar{B}_1 + A_1B_1)\bar{A}_0 B_0$$

2-bit magnitude comparator

Inputs				Outputs		
A_1	A_0	B_1	B_0	$Y_{A>B}$	$Y_{A=B}$	$Y_{A<B}$
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

$Y_{A>B}$

B_1B_0 \ A_1A_0	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	1	0

$$Y_{A>B} = A_1\bar{B}_1 + A_0\bar{B}_1\bar{B}_0 + A_1A_0\bar{B}_0$$

$Y_{A=B}$

B_1B_0 \ A_1A_0	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

$$Y_{A=B} = \bar{A}_1\bar{A}_0\bar{B}_1\bar{B}_0 + \bar{A}_1A_0\bar{B}_1B_0 + A_1A_0B_1B_0 + A_1\bar{A}_0B_1\bar{B}_0$$

$Y_{A<B}$

B_1B_0 \ A_1A_0	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

$$Y_{A<B} = \bar{A}_1B_1 + \bar{A}_1\bar{A}_0B_0 + \bar{A}_0B_1B_0$$

COMBINATIONAL AND SEQUENTIAL CIRCUITS

In combinational circuits, the outputs at any instant of time depend upon the inputs present at that instant of time, which means there is no memory in these circuits.

In sequential circuits, the outputs at any instant of time depend upon present inputs as well as past inputs / outputs, which means that there are elements used to store information. These elements are known as memory.

The design requirements of combinational circuits may be specified in one of the following ways :-

1. A set of statements,
2. Boolean expression, and
3. Truth table

The following methods can be used to simplify the Boolean functions :-

1. Algebraic method
2. Karnaugh-map technique
3. Quine-McCluskey method

Standard Representations for Logic Functions

Logic functions are expressed in terms of logical variables. The values assumed by the logic functions as well as the logic variables are in binary form. Any arbitrary function can be expressed in the following forms :-

1. Sum-of-Products form (SOP), and
2. Product-of-Sums form (POS).

Logic function can be written in other forms as well, however the above two forms are conveniently suited in arriving at the standard methods for designing the circuits.

q:- $Y = (A + BC)(B + \bar{C}A)$

SOP form - $Y = (A + BC)(B + \bar{C}A)$

$= AB + A\bar{C}A + BCB + BC\bar{C}A$ - Distributive

$= AB + A\bar{C} + BC + 0$ - $\because A.A = A,\ C\bar{C} = 0$

$= AB + A\bar{C} + BC$

$\therefore$ Standard SOP form $= AB + A\bar{C} + BC$

POS form - $Y = (A + BC)(B + \bar{C}A)$

$= (A + B)(A + C)(B + \bar{C})(B + A)$ - Distributive

$= (A + B)(A + C)(B + \bar{C})$ - $B + A = A + B$

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

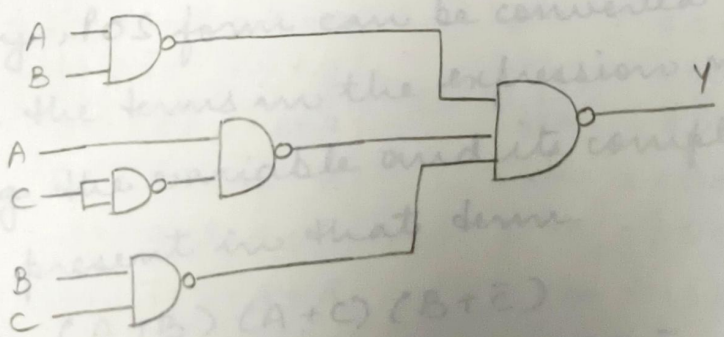
$\therefore$ Standard POS form $= (A + B)(A + C)(B + \bar{C})$

NAND realisation of SOP -

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

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

$Y = \overline{\overline{AB} \cdot \overline{A\bar{C}} \cdot \overline{BC}}$



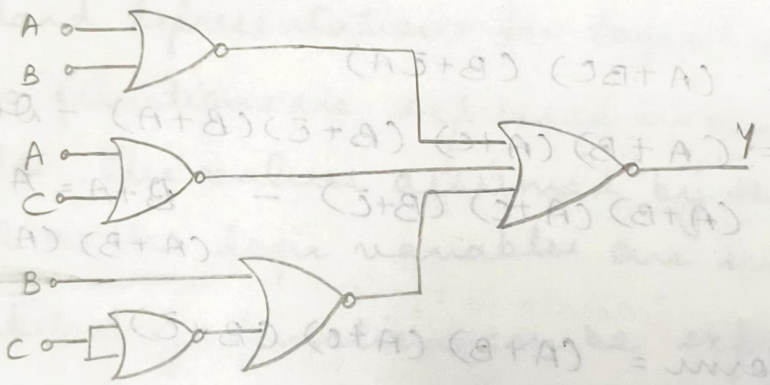
NOR realization of POS form -

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

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

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

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



All the individual terms do not involve all the three literals. If each of the terms in SOP and POS forms contains all the literals then these are known as canonical SOP and POS, respectively. Each individual term in canonical SOP form is called as minterm and in canonical POS form as maxterm.

SOP form can be converted to canonical SOP by ANDing the terms in the expression with terms formed by ORing the variable and its complement which are not present in that term.

eg:- $Y = AB + A\bar{C} + BC$

$= AB(C+\bar{C}) + A\bar{C}(B+\bar{B}) + BC(A+\bar{A})$ — $\because C+\bar{C} = 1$

$= ABC + AB\bar{C} + AB\bar{C} + A\bar{B}\bar{C} + ABC + \bar{A}BC$

$= ABC + AB\bar{C} + A\bar{B}\bar{C} + \bar{A}BC$ — $\because ABC + ABC = ABC$, $AB\bar{C} + AB\bar{C} = AB\bar{C}$

Similarly, POS form can be converted to canonical POS by ORing the terms in the expression with terms formed by ANDing the variable and its complement which are not present in that term.

eg:- $Y = (A+B)(A+C)(B+\bar{C})$

$= (A+B+C\bar{C})(A+B\bar{B}+C)(A\bar{A}+B+\bar{C})$ — $\because C\bar{C} = 0$

$= (A+B+C)(A+B+\bar{C})(A+B+C)(A+\bar{B}+C)(A+B+\bar{C})(\bar{A}+B+\bar{C})$ - Distributive

$= (A+B+C)(A+B+\bar{C})(A+\bar{B}+C)(\bar{A}+B+\bar{C})$

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

In general, for an n-variable logical function there are 2^n minterms and an equal number of maxterms.

Each minterm is expressed by m_i where the subscript i is the decimal equivalent of the natural binary number corresponding to the minterm with normal (uncomplemented) variables taken as 1's and the complemented variables taken as 0.

Similar to minterms, each maxterm is represented by M_i, where the subscript i is the decimal equivalent of the natural binary number corresponding to the maxterm with uncomplemented variables taken as 0's and the complemented variables taken as 1's.

VARIABLE A	B	C	D	MINTERM	MAXTERM
0	0	0	0	$\bar{A}\bar{B}\bar{C}\bar{D} = m_0$	$A+B+C+D = M_0$
0	0	0	1	$\bar{A}\bar{B}\bar{C}D = m_1$	$A+B+C+\bar{D} = M_1$
0	0	1	0	$\bar{A}\bar{B}C\bar{D} = m_2$	$A+B+\bar{C}+D = M_2$
0	0	1	1	$\bar{A}\bar{B}CD = m_3$	$A+B+\bar{C}+\bar{D} = M_3$
0	1	0	0	$\bar{A}B\bar{C}\bar{D} = m_4$	$A+\bar{B}+C+D = M_4$
0	1	0	1	$\bar{A}B\bar{C}D = m_5$	$A+\bar{B}+C+\bar{D} = M_5$
0	1	1	0	$\bar{A}BC\bar{D} = m_6$	$A+\bar{B}+\bar{C}+D = M_6$
0	1	1	1	$\bar{A}BCD = m_7$	$A+\bar{B}+\bar{C}+\bar{D} = M_7$
1	0	0	0	$A\bar{B}\bar{C}\bar{D} = m_8$	$\bar{A}+B+C+D = M_8$
1	0	0	1	$A\bar{B}\bar{C}D = m_9$	$\bar{A}+B+C+\bar{D} = M_9$
1	0	1	0	$A\bar{B}C\bar{D} = m_{10}$	$\bar{A}+B+\bar{C}+D = M_{10}$
1	0	1	1	$A\bar{B}CD = m_{11}$	$\bar{A}+B+\bar{C}+\bar{D} = M_{11}$
1	1	0	0	$AB\bar{C}\bar{D} = m_{12}$	$\bar{A}+\bar{B}+C+D = M_{12}$
1	1	0	1	$AB\bar{C}D = m_{13}$	$\bar{A}+\bar{B}+C+\bar{D} = M_{13}$
1	1	1	0	$ABC\bar{D} = m_{14}$	$\bar{A}+\bar{B}+\bar{C}+B = M_{14}$
1	1	1	1	$ABCD = m_{15}$	$\bar{A}+\bar{B}+\bar{C}+\bar{D} = M_{15}$

eg:- $Y = ABC + A\bar{B}C + A\bar{B}\bar{C} + \bar{A}BC$

$= \bar{A}BC + A\bar{B}\bar{C} + AB\bar{C} + ABC$

$= m_3 + m_4 + m_6 + m_7$

$= \Sigma m(3, 4, 5, 6)$

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

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

$= M_0 \cdot M_1 \cdot M_2 \cdot M_5$

$= \Pi M(0, 1, 2, 5)$

The above two equations represent the same logical function, therefore we notice that there is a complementary type of relationship between a function expressed in terms of minterms and in terms of maxterms.

Hence, if a logical function is specified in terms of minterms / maxterms, its maxterms / minterms representation can be determined by using this complementary property.

eg:- For a four-variable case if

$Y = \Sigma m(0, 3, 6, 7, 10, 12, 15)$

then $Y = \Pi M(1, 2, 4, 5, 8, 9, 11, 13, 14)$