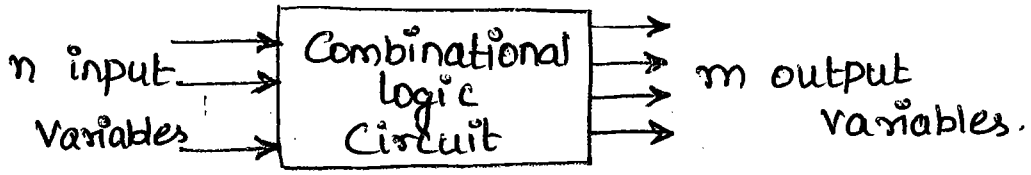
Unit 3 : ARITHMETIC AND STANDARD COMBINATIONAL MODULE AND NETWORKS (10 hrs)

1. Adders
2. Subtractors (1)
3. Binary parallel adders,
4. Parallel Subtractors (1)
5. Binary decoders and encoders (2)
   - Priority encoders. (1)
6. Multiplexers (1)
   - MUX as universal Combinational Modules (1)
7. Demultiplexers (1)
8. ALU Module and Comparator modules. (2)

Introduction

* When logic gates are connected together to produce a specified output for certain specified combinations of input variables, with no storage involved, the resulting circuit is called "Combinational logic circuit".

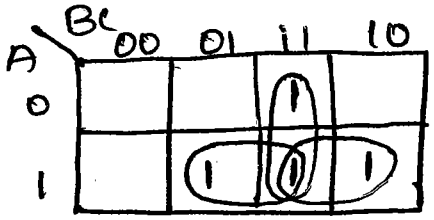


Design procedure:

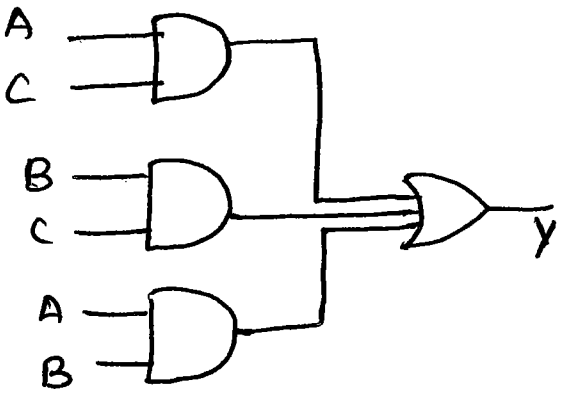
1. The problem definition.
2. The determination of number of available input variables and required output variables.
3. Assigning letter symbols to input and output variable
4. The derivation of truth table indicating the relationships between input and output variables.
5. Obtain simplified Boolean expressions for each output.
6. Obtain the logic diagram.

1. Design a Combination logic circuit with three input variables that will produce a logic 1 output when more than one input variables are logic 1

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



$Y = AB + BC + AC$



DESIGN OF ADDERS

Simple addition

$0 + 0 = 0$

$0 + 1 = 1$

$1 + 0 = 1$

$1 + 1 = (10)_2$

* If the sum value is of two digits

MSB = carry

LSB = Sum.

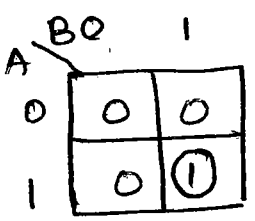
* The logic circuit performing this operation is called half adder. (two bits)
* The circuit performing addition of three bits (two significant bit and a previous carry) is called full adder.

Half Adder:

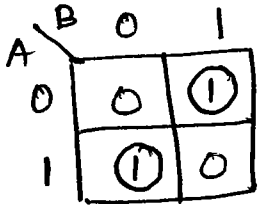
Truth table

A	B	carry	sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

K-map



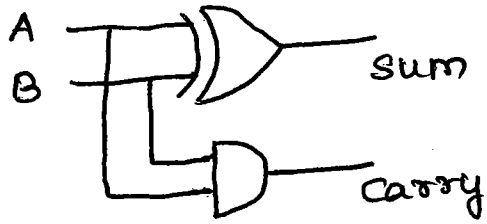
Carry = AB



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

$= A \oplus B$

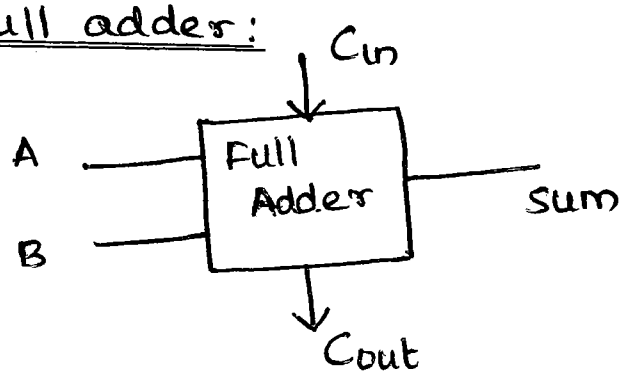
Logic circuit



disadvantage:

In multidigit addition we have to add two bits along with the carry of previous digit.

Full adder:

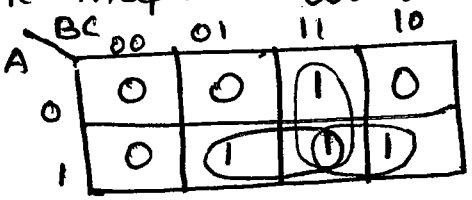


C_{in} → carry from the previous lower significant position.

A, B → Inputs

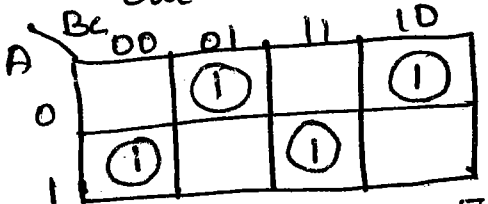
A	B	Cin	Cout	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

k-map for carry



$C_{out} = BC + AC + AB$

k-map for Sum



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

Simplication:

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

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

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

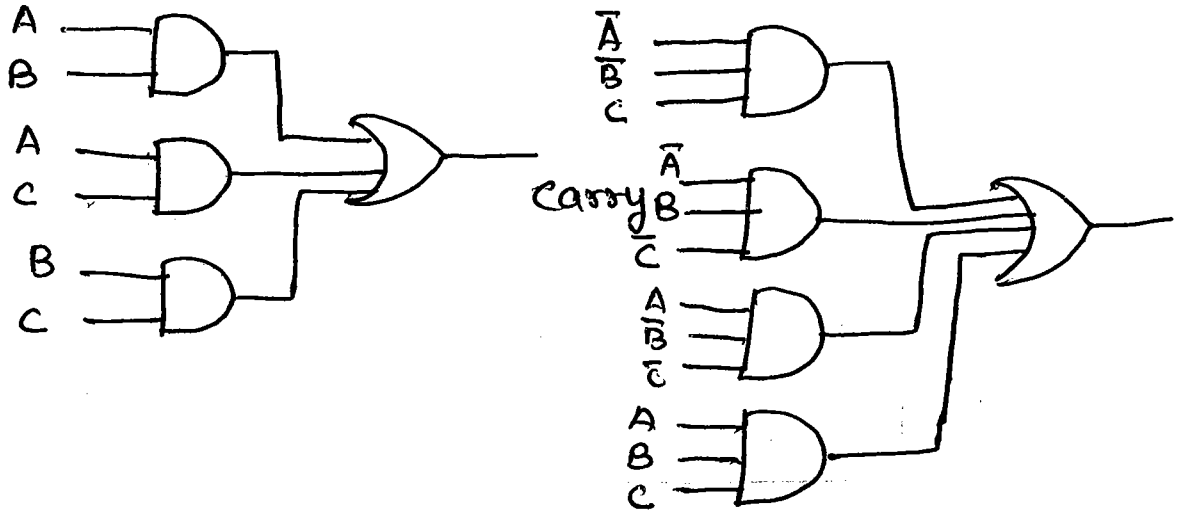
$= C(A \bar{\oplus} B) + \bar{C}(A \oplus B)$

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

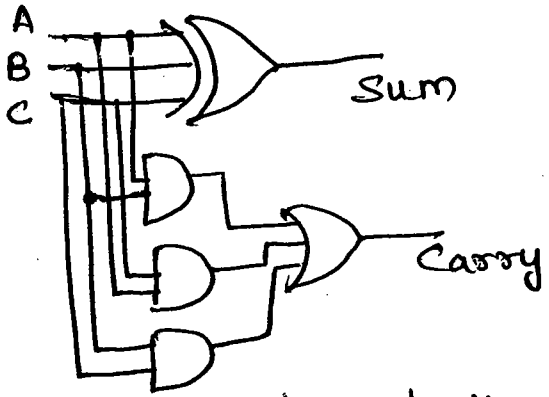
So

$= C \oplus (A \oplus B)$

Logic circuit for sum and carry.



Implementation of full adder.



Full adder using two half adders:

Carry of full adder

$C = AB + BC + AC$

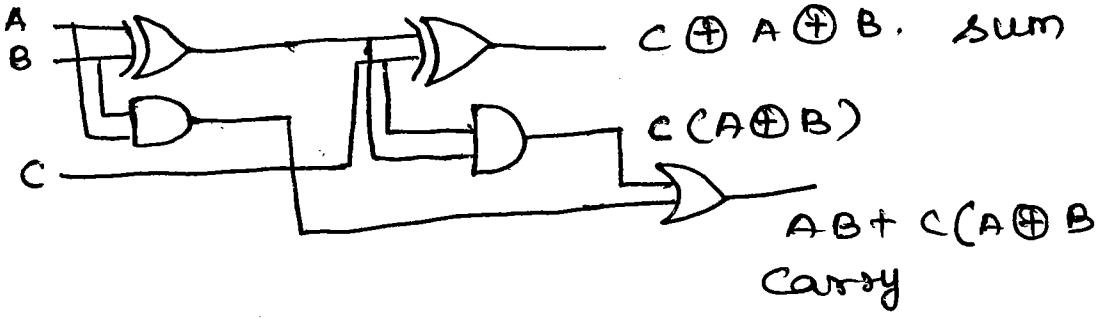
$= AB + AC(B+\bar{B}) + BC(A+\bar{A})$

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

$= AB(1 + C_{in} + C_{in}) + A\bar{B}C + \bar{A}BC.$

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

$= AB + C(A \oplus B)$



DESIGN OF SUBTRACTOR

Simple subtraction.

$0 - 0 = 0$

$0 - 1 = 1$ with 1 borrow.

$1 - 0 = 1$

$1 - 1 = 0$

1) Half subtractor

2) Full subtractor.

Half Subtractor:

Inputs		outputs	
A	B	Difference	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

Difference

A \ B	0	1
0	0	(1)
1	(1)	0

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

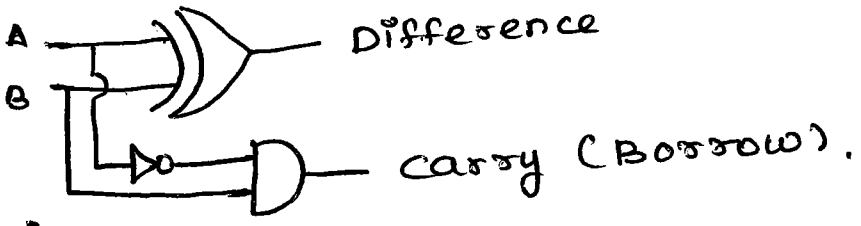
$= A \oplus B$

Borrow

A \ B	0	1
0	0	(1)
1	0	0

$= \bar{A}B$

Logic circuit



Limitations:

* In multidigit subtraction, we have to subtract two bits along with the borrow of the previous bit subtraction.

* Effectively such subtraction requires subtraction of three bits. This is not possible with half subtractor.

Full subtractor:

The circuit has three inputs A, B, B_{in} and two outputs D and B_{out}.

D - Difference

B_{out} - Borrow.

Inputs			outputs	
A	B	B_{in}	D	B_{out}
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

K-map simplification of D and Bout

for D

A \ BC	00	01	11	10
0	0	1	0	1
1	1	0	1	0

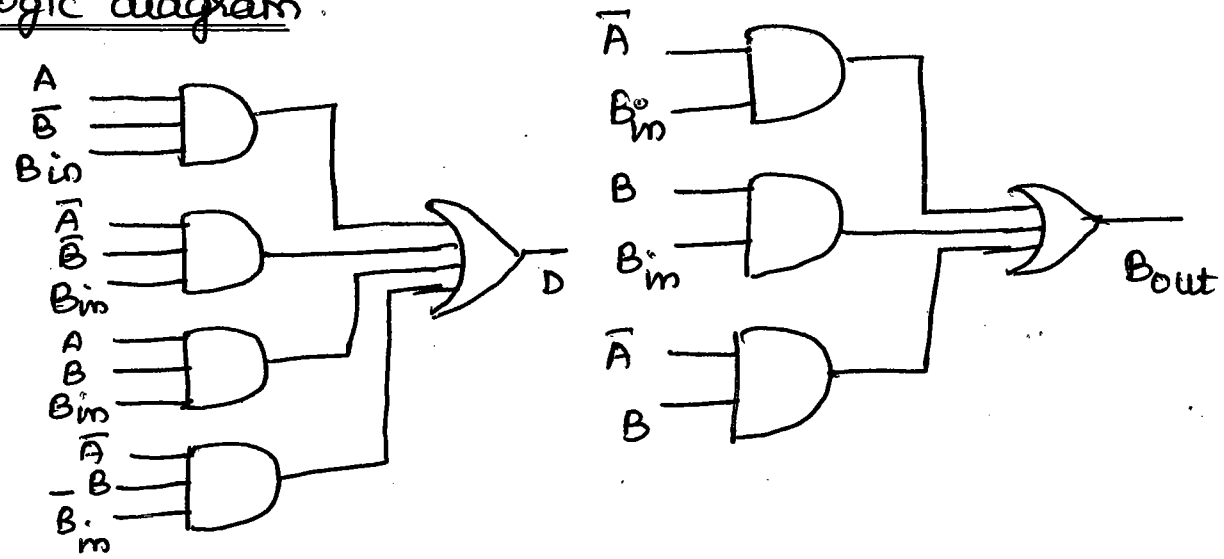
$D = A\bar{B}\bar{B_{in}} + \bar{A}\bar{B}B_{in} + ABB_{in} + \bar{A}B\bar{B_{in}}$

for Bout

A \ BB_{in}	00	01	11	10
0	0	1	1	1
1	0	0	1	0

$B_{out} = \bar{A}B_{in} + BB_{in} + \bar{A}B$

Logic diagram



Simplification of D.

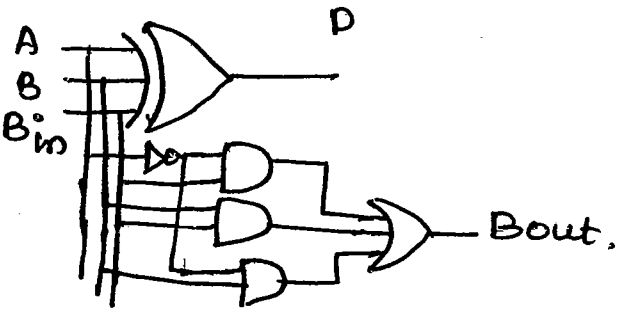
$= A\bar{B}\bar{B_{in}} + \bar{A}\bar{B}B_{in} + ABB_{in} + \bar{A}B\bar{B_{in}}$

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

$= B_{in}(A \odot B) + \bar{B_{in}}(A \oplus B)$

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

$= B_{in} \oplus A \oplus B.$



Full subtractor with two half subtractor.

$$B_{out} = \bar{A}B_{in} + \bar{A}B + BB_{in}$$

$$= \bar{A}B + \bar{A}B_{in}(B+\bar{B}) + BB_{in}(A+\bar{A})$$

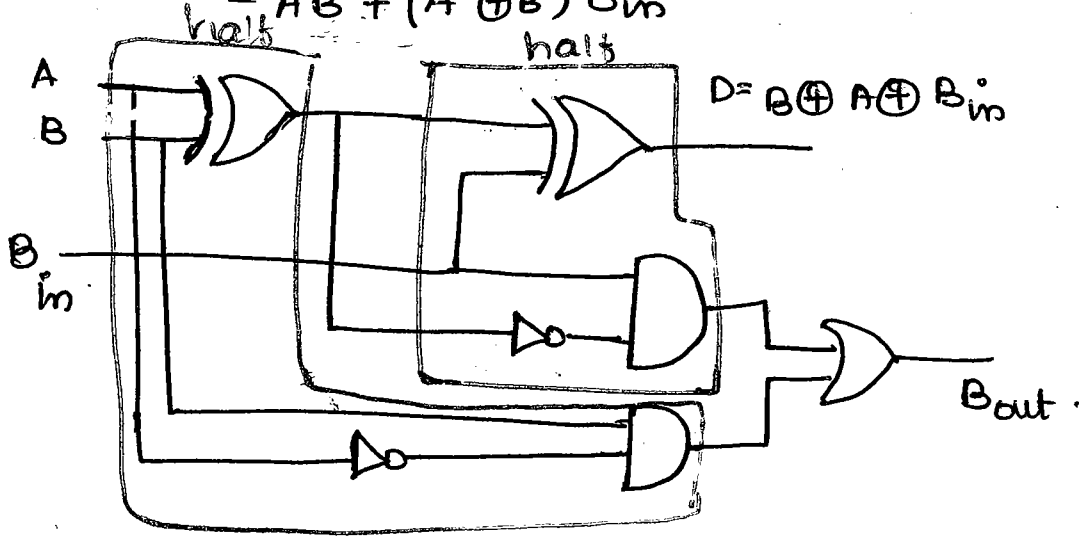
$$= \bar{A}B + \bar{A}B_{in}B + \bar{A}\bar{B}B_{in} + ABB_{in} + \bar{A}BB_{in}$$

$$= \bar{A}B + \bar{A}B_{in}B + \bar{A}\bar{B}B_{in} + ABB_{in}$$

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

$$= \bar{A}B + (A \odot B)B_{in}$$

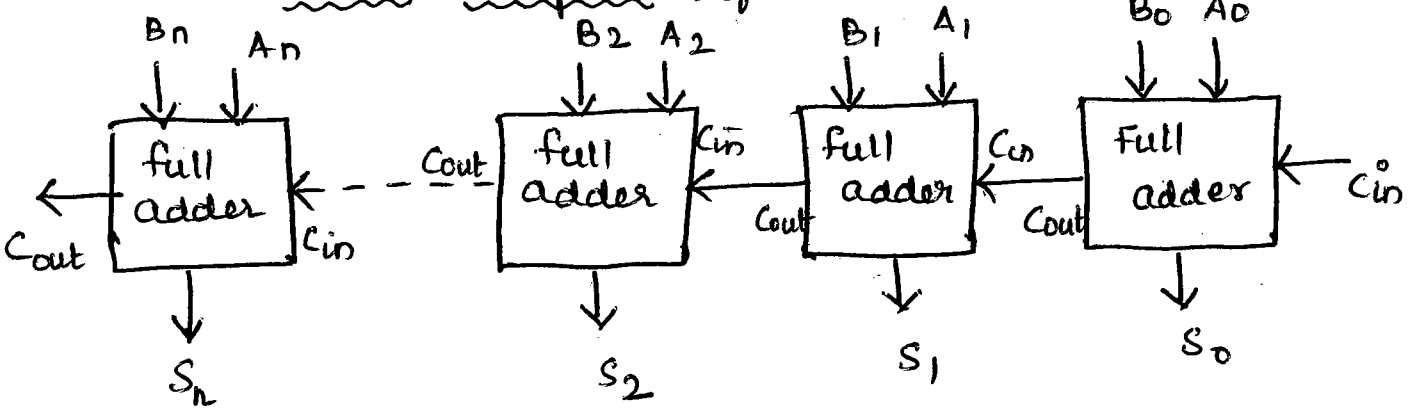
$$= \bar{A}B + (\overline{A \oplus B})B_{in}$$



Parallel Adder:

* Here single full adder is capable of adding two one bit numbers and an input carry.
* Parallel adder is constructed mainly to add binary numbers with more than one bit.
* The carry of the full adder is given as C_{in} of the next full order.

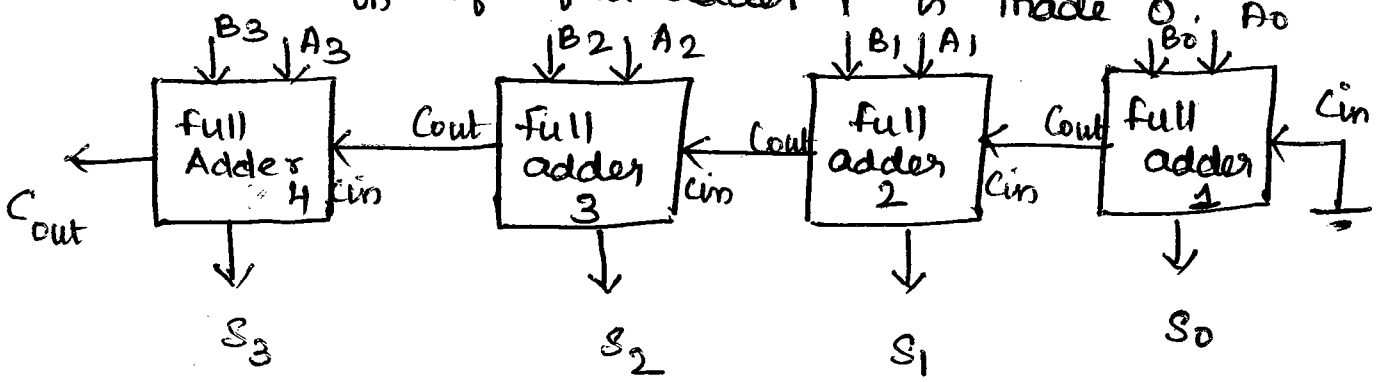
Block diagram of n-bit parallel adder.

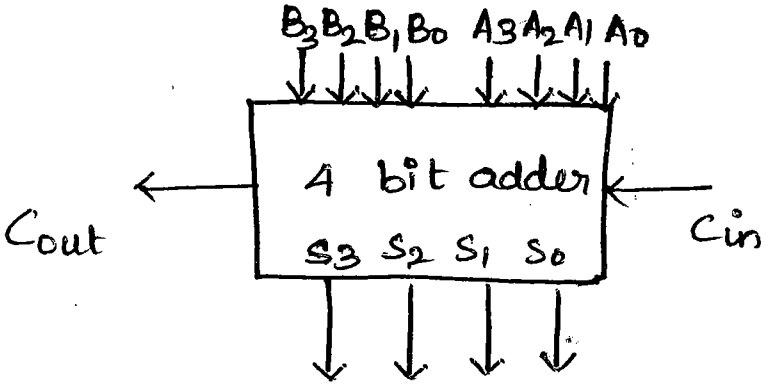


Exercise:

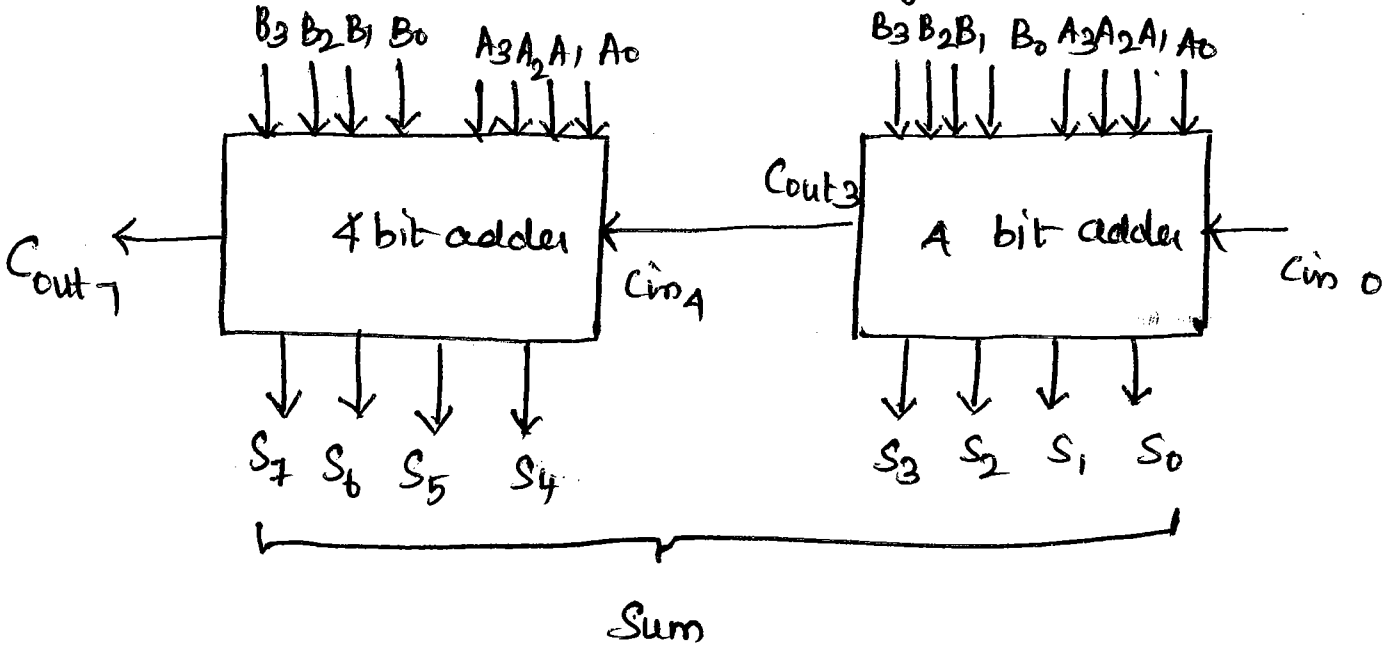
1. Design a 4 bit parallel adder using full adders.

* 4 Full adders.
* C_{in} of full adder 1 is made 0.





2. Design a 8 bit adder using two four bit adder.



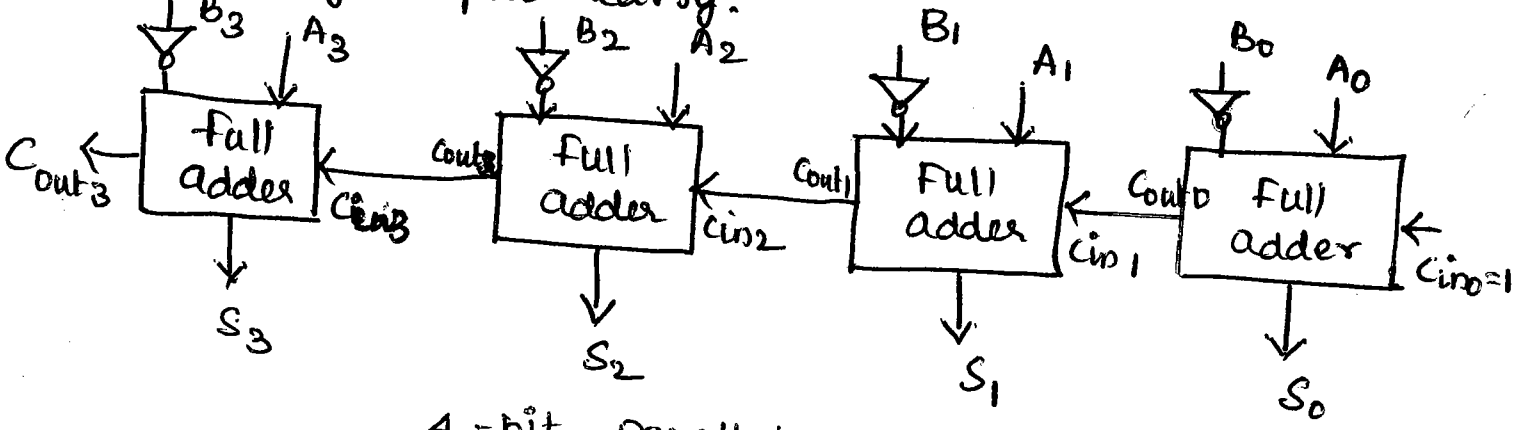
Parallel subtractor

Subtraction of binary numbers can be done most Conveniently by means of Complements.

Subtraction of A and B can be done by taking 2's complement of B and adding it with A.

2's Complement is done by taking 1's complement and adding 1 to 1's Complemented number

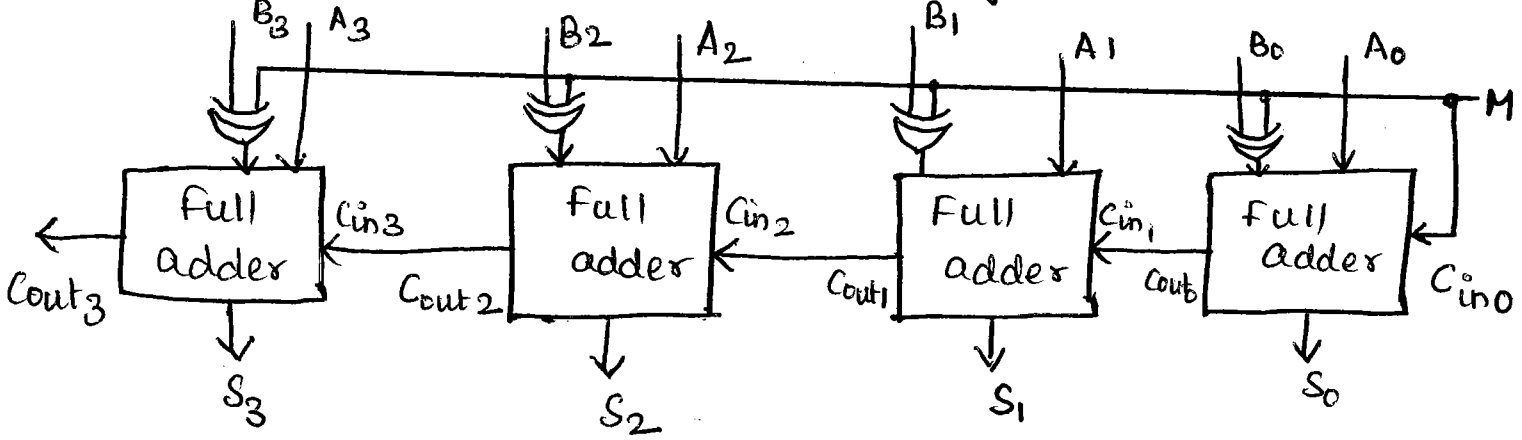
1's Complement is done by taking inversion of the number. Here 1 is added to the complemented number by input carry.



4-bit parallel subtractor.

Parallel Adder/ Subtractor:

* Addition and subtraction can be combined into one circuit with one Binary adder.
* This is done by including an ex-OR gate with each full adder.



If M=0 → circuit is an adder

M=1 → Circuit is an subtractor

* M Controls the operation of the Circuit.

EX-OR ⇒ one I/p → M

other I/p → B

If M = 0 $B \oplus 0 = B$; $C_{in_0} = 0$

If M = 1 $B \oplus 1 = \bar{B}$; $C_{in_0} = 1$

* B I/p is complemented and 1 is added through a input carry.

* A plus 2's complemented $\bar{B} = A - B$.

Note:

The parallel adder is a ripple carry adder in which the carry output of each full-adder stage is connected to carry input of the next order. Therefore, the sum and carry outputs of any stage cannot be produced until the input carry occurs, this leads to a time delay in the addition process. This delay is known as "carry propagation delay".

BCD ADDER:

The digital system handles the decimal number in the form of BCD. BCD adder adds two BCD digits and produce a sum of BCD.

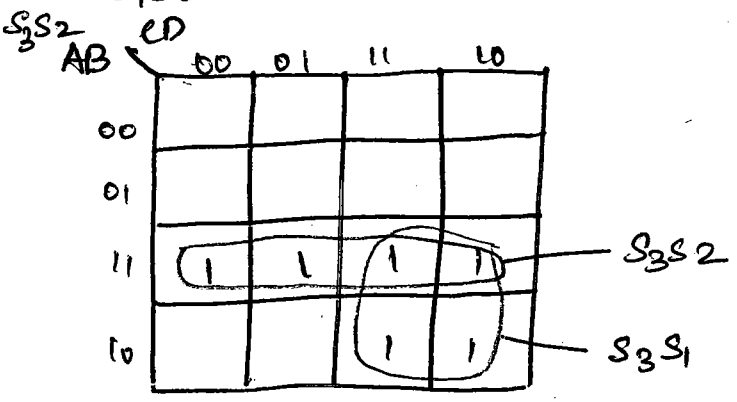
Implementation:

* 4 bit binary adder for initial addition
* Logic cicuit to detect sum greater than 9
* one more 4 bit greater than 9 or carry is 1

Truth Table.

Inputs				output
S_3	S_2	S_1	S_0	Y
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

K- map for Y



$$Y = S_3S_2 + S_3S_1$$

The logic circuit can be drawn to detect the sum greater than 9.

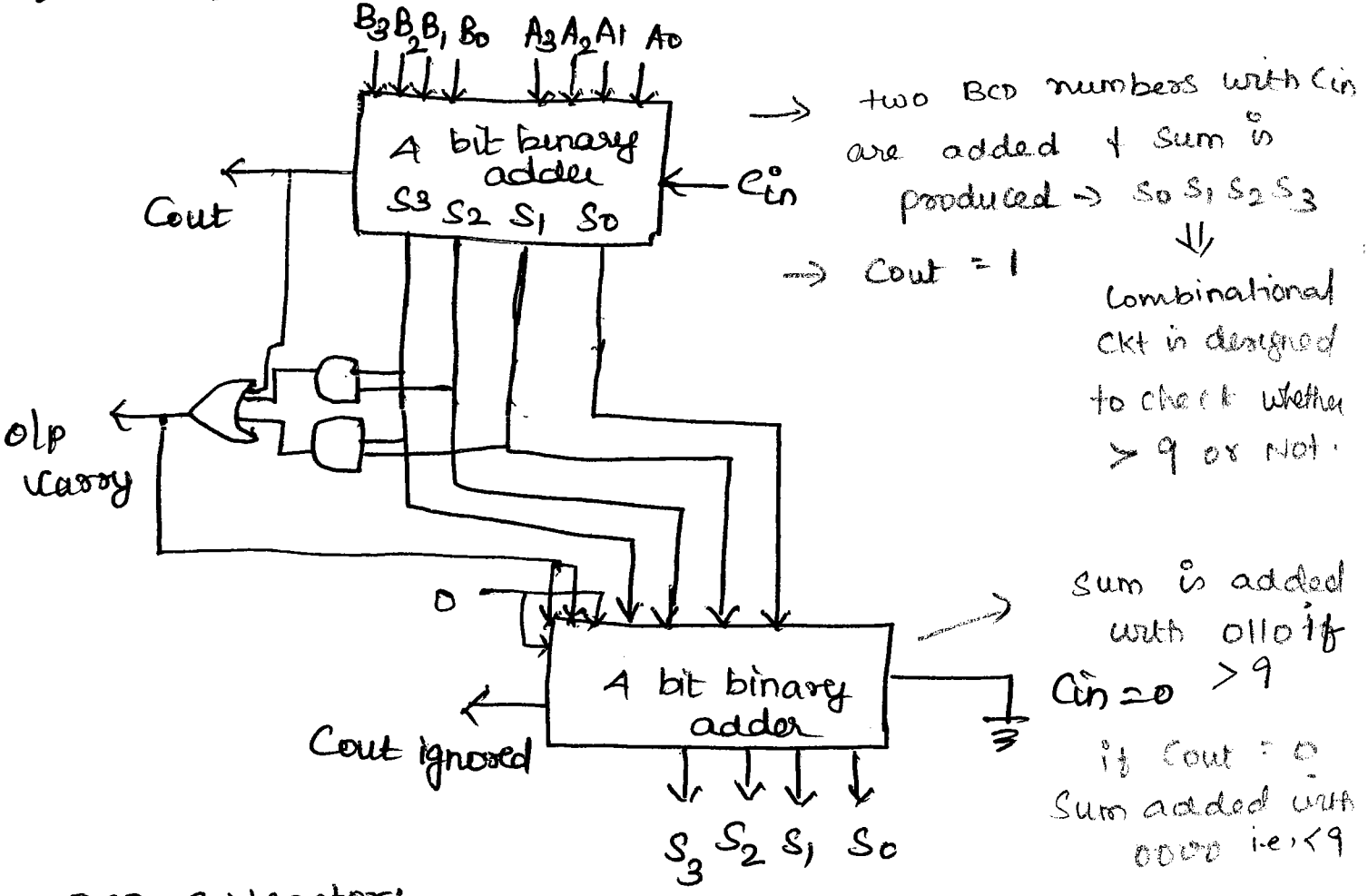
$Y=1$ indicates sum >9, we can put one more term Cout in the above expression to check whether carry is one.

If any one condition is satisfied we add 0110 in the sum

Steps: The two BCD numbers, together with C_{in} are first added in the top 4 bit binary adder to produce a binary sum.

* when the output carry is equal to zero, nothing is added to the binary sum.
* when the output carry is equal to one, binary 0110 is added to the binary sum through the bottom 4 bit binary adder
* The output carry generated from the bottom

binary adder can be ignored. Since it supplies information already available at the output carry terminal.



BCD Subtractor:

* Addition of signed BCD numbers can be performed by using 9's and 10's Complement.

Subtraction using 9's complement method

1. find the 9's complement of the -ve number
2. Add two numbers using BCD number.
3. If carry is generated add carry to the result. Otherwise find the 9's Complement.

Addition of 10 and -2

10 ⇒ 1010
-2 ⇒ 0010
9's of -2 ⇒ 1101
1010
10111
ignore carry.
1
1000 Result.

Subtraction using 10's Complement.

1. Find the 10's complement of the negative number.
2. Add using BCD addition.
3. If carry is not generated find the 10's Comp. of the result.

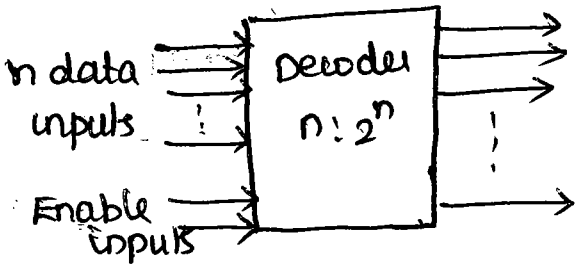
DECODERS:

* A decoder is a multiple-input, multiple output logic circuits which converts coded input into coded outputs, where the input and output codes are different.
* The input code generally has fewer bits than the output codes.
* One to one mapping from input code words to output code word.

Input = n ; Output = 2^n.

"A circuit which is capable of converting a code is called decoder"

General structure of the Decoder.



Binary decoder:

* A decoder which has an n-bit binary input code and a one activated output out of 2^n output code is called binary decoder.
* used to activate exactly one of 2^n outputs based on n-bit input value.

Example: 2 : 4 Decoder.

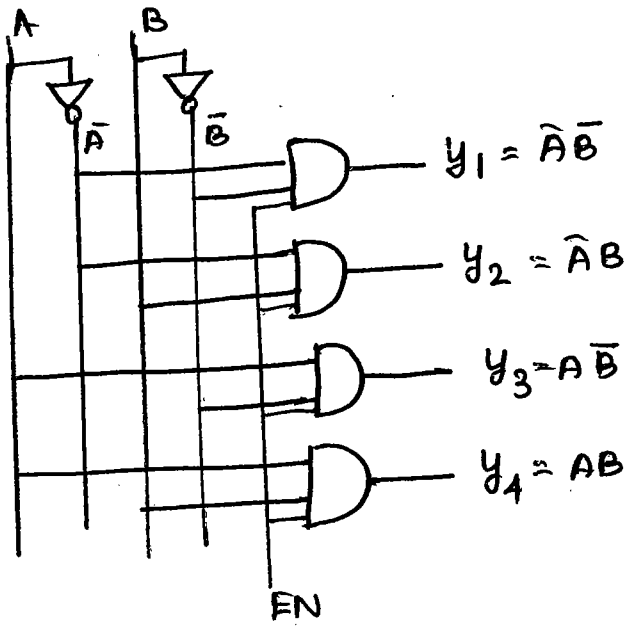
No. of inputs = 2 (A, B)

No. of Outputs = 2^2 = 4 (Y_1, Y_2 Y_3 Y_4)

EN - Enable to all gates.

Truth table

EN	A	B	y_1	y_2	y_3	y_4
0	X	X	0	0	0	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	0	1



2:4 decoder.

Example 2.

Draw the circuit for 3:8 decoder and explain

No. of inputs = 3

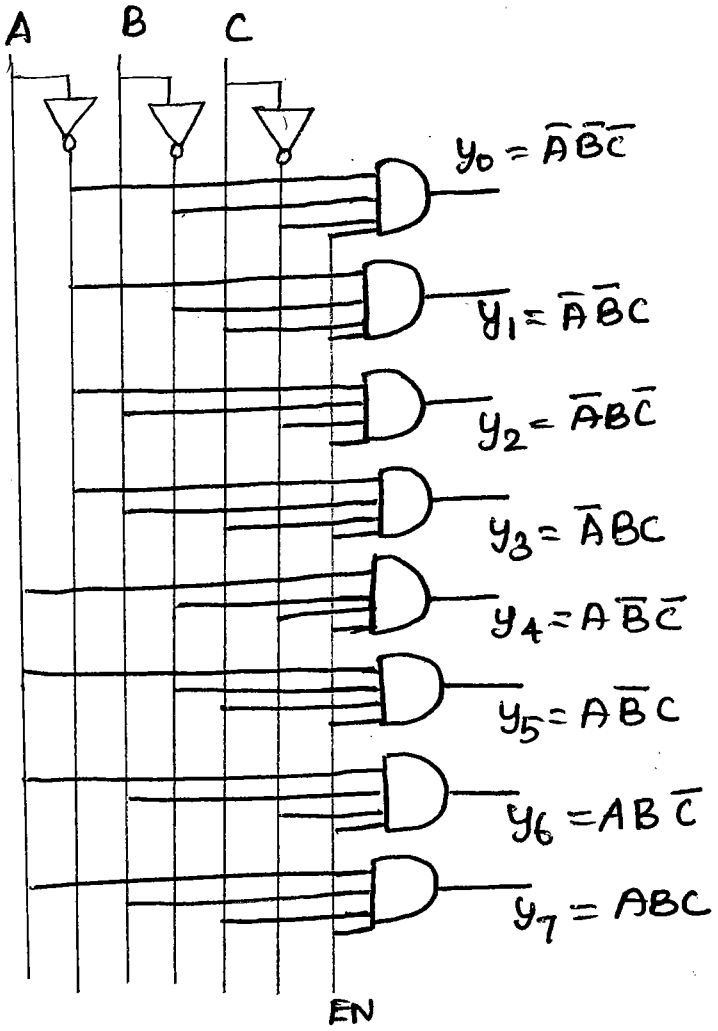
No. of outputs = $2^3 = 8$

EN - To all the gates.

Truth table.

EN	A	B	C	y_0	y_1	y_2	y_3	y_4	y_5	y_6	y_7
0	X	X	X	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0	0	0
1	0	1	1	0	0	0	1	0	0	0	0
1	01	0	0	0	0	0	0	1	0	0	0
1	1	0	1	0	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	0	0	1	0
1	1	1	1	0	0	0	0	0	0	0	1

EN - Enable bits



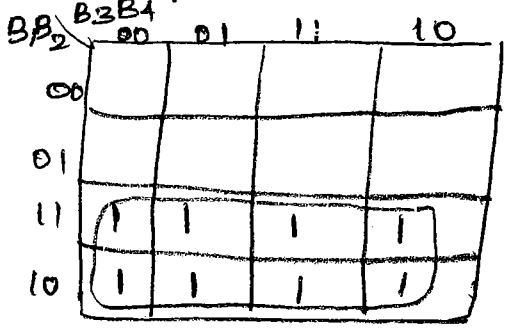
Decoders are also called as code Converter.

1. Binary - Gray Decoder
2. BCD - decimal Decoder.
3. BCD - 8even segment Decoder.

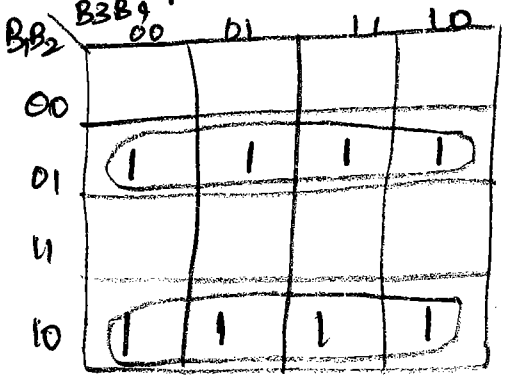
Binary - Gray Decoder.

Decimal	Binary $B_1B_2B_3B_4$	Gray $G_1G_2G_3G_4$
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7.	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001

K-map for G_1



K-map for G_2

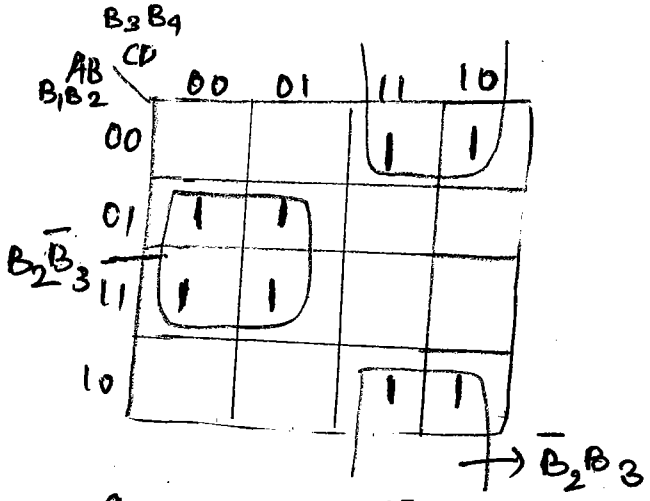


Binary
0 0 0 1
1
0 0 0 1

$G_1 = B_1$

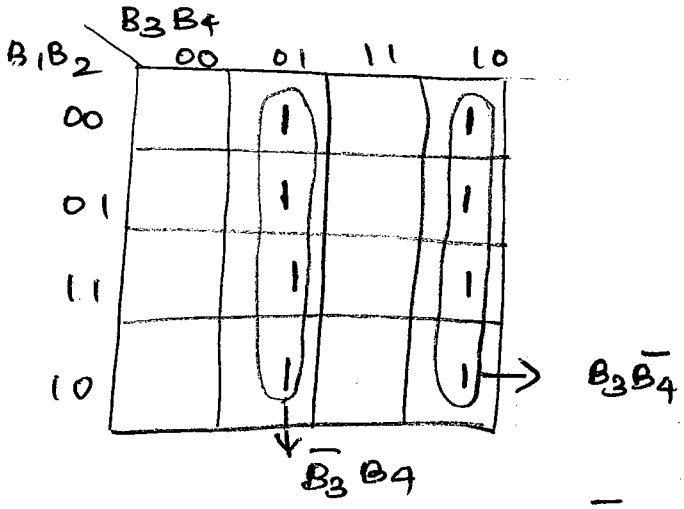
eg
Binary
0 0 1 0
1
0 0 1 1
gray

$G_2 = \bar{B}_1B_2 + B_1\bar{B}_2$
$= B_1 \oplus B_2$



$g_3 = B_2\bar{B}_3 + \bar{B}_2 B_3$

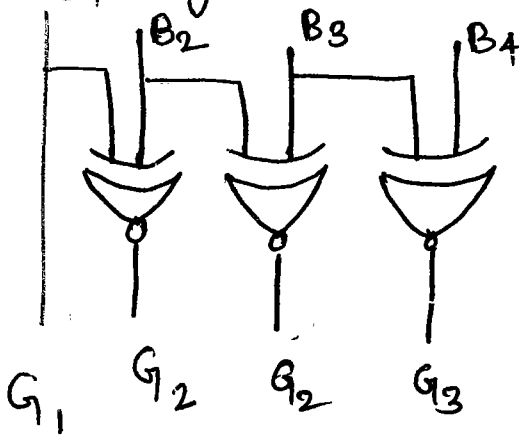
$= B_2 \oplus B_3$



$g_4 = \bar{B}_3 B_4 + B_3 \bar{B}_4$

$= B_3 \oplus B_4.$

B1 Logic circuit.



Binary to BCD Converter:

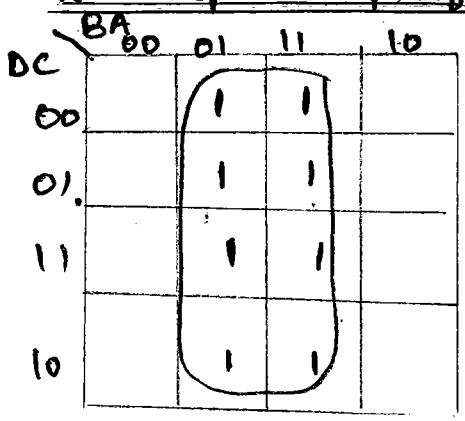
Binary code				BCD code					Binary code				BCD code.				
D	C	B	A	B_4	B_3	B_2	B_1	B_0	D	C	B	A	B_4	B_3	B_2	B_1	B_0
0	0	0	0		0	0	0	0	1	1	0	0	1	0	0	1	0
0	0	0	1		0	0	0	1	1	1	0	1	1	0	0	1	1
0	0	1	0		0	0	1	0	1	1	1	0	1	0	1	0	0
0	0	1	1		0	0	1	1	1	1	1	1	1	0	1	0	1
0	1	0	0		0	1	0	0									
0	1	0	1		0	1	0	1									
0	1	1	0		0	1	1	0									
0	1	1	1		0	1	1	1									
1	0	0	0		1	0	0	0									
1	0	0	1		1	0	0	1									
1	0	1	0	1	0	0	0	0									
1	0	1	1	1	0	0	0	1									

add 0110 after 1001

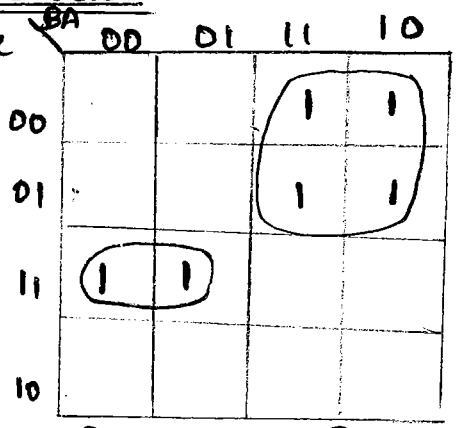
BCD valid till '9'

> 9 add 0110

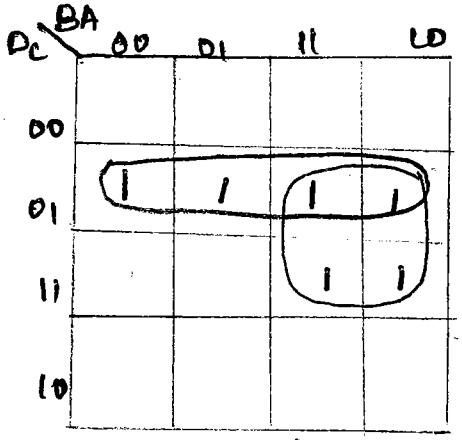
K- Map Simplification.



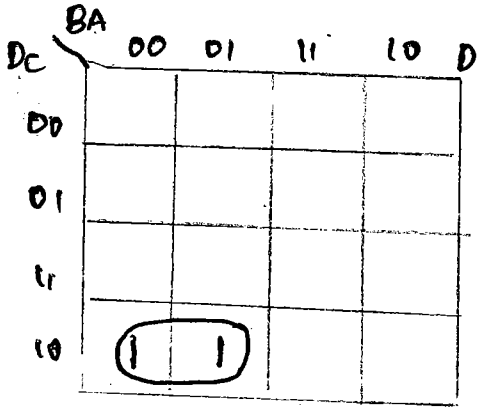
for $B_0 = A$



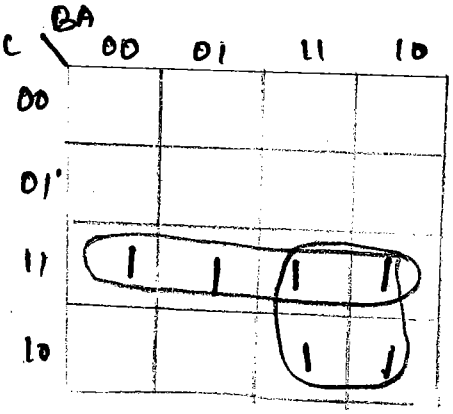
for $B_1 = DC\bar{B} + \bar{D}B$



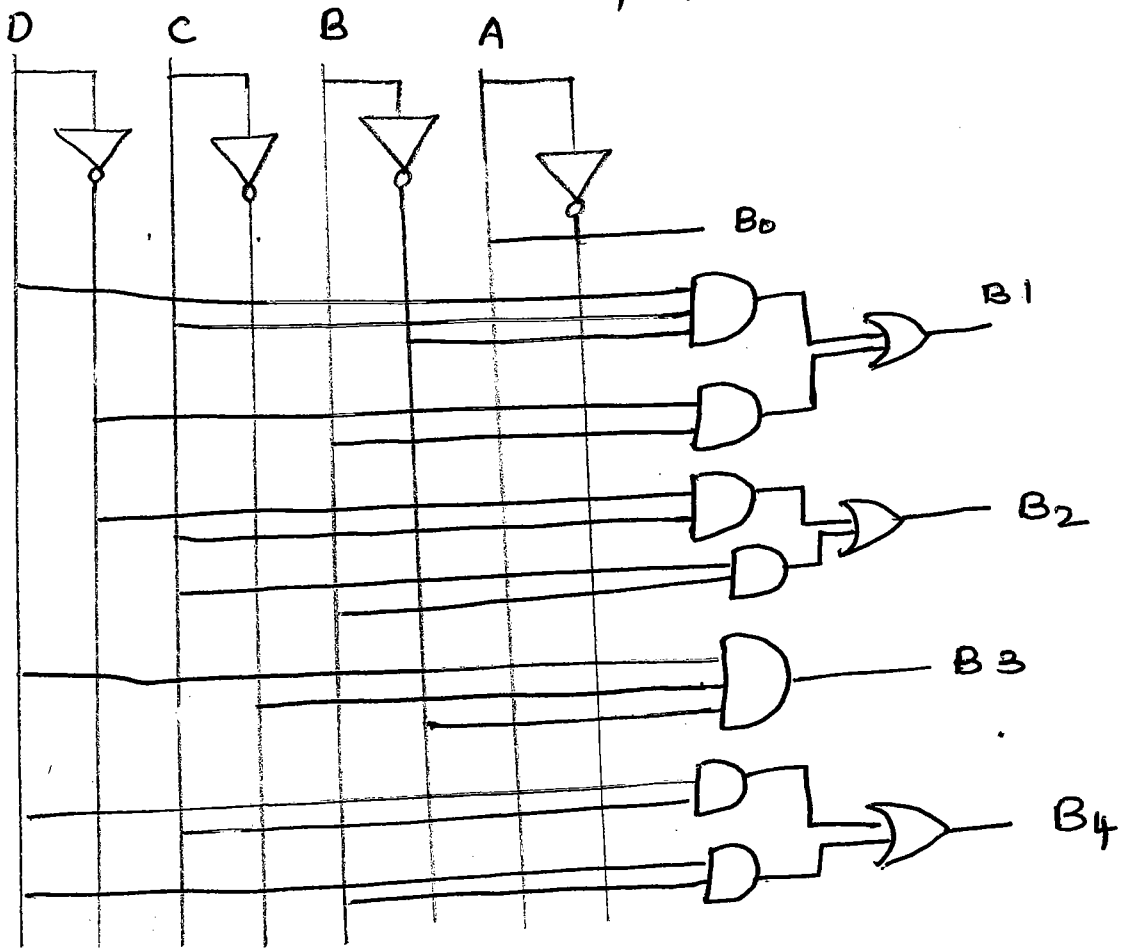
for $B_2 = \bar{D}C + CB$



for $B_3 = D\bar{C}B$



for $B_4 = DC + DB$

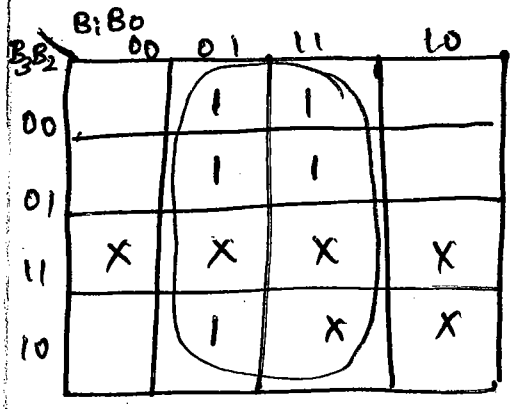


BCD to Binary Converter:

	B_3	B_2	B_1	B_0	F	D	C	B	A
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	0	0	1	0
0	0	0	1	1	0	0	0	1	1
0	0	1	0	0	0	0	1	0	0
0	0	1	0	1	0	0	1	0	1
0	0	1	1	0	0	0	1	1	0
0	0	1	1	1	0	0	1	1	1
0	1	0	0	0	0	1	0	0	0
0	1	0	0	1	0	1	0	0	1
1	0	0	0	0	0	1	0	1	0
1	0	0	0	1	0	1	0	1	1
1	0	0	1	0	0	1	1	0	0
1	0	0	1	1	0	1	1	0	1
1	0	1	0	0	0	1	1	1	0
1	0	1	0	1	0	1	1	1	1
1	0	1	1	0	1	0	0	0	0
1	0	1	1	1	1	0	0	0	1
1	1	0	0	0	1	0	0	1	0
1	1	0	0	1	1	0	0	1	1

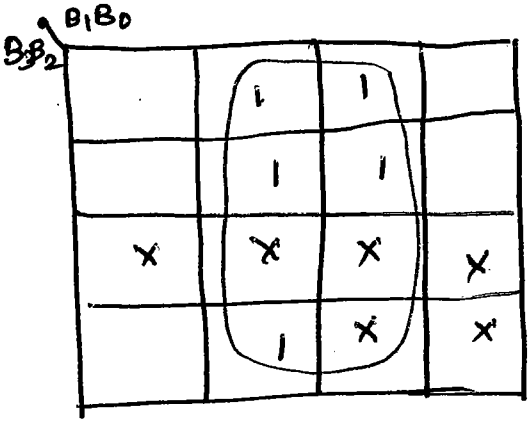
K-Map Simplification.

for A. $B_4 = 0$

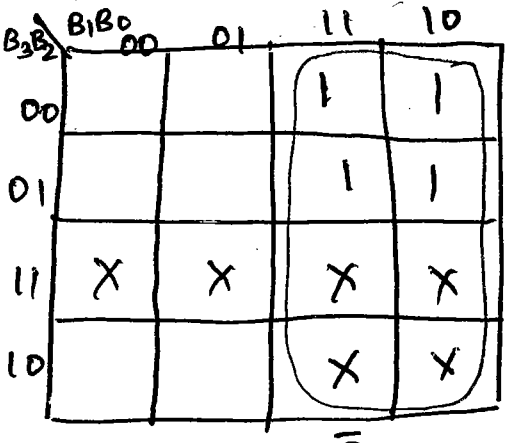


$A = B_0$

$B_4 = 1$

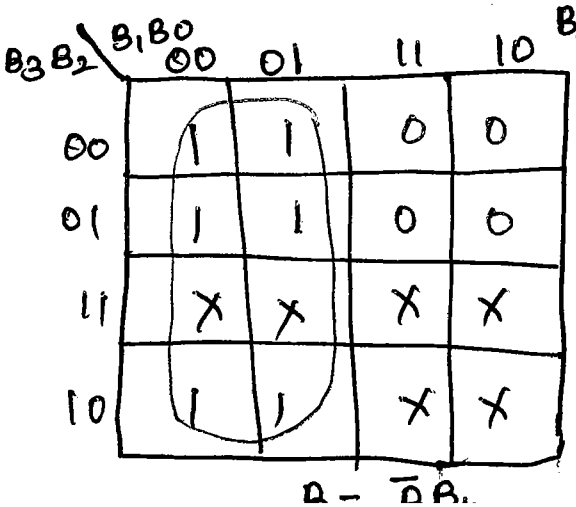


For B $B_4 = 0$



$B = B_1 \bar{B}_4$

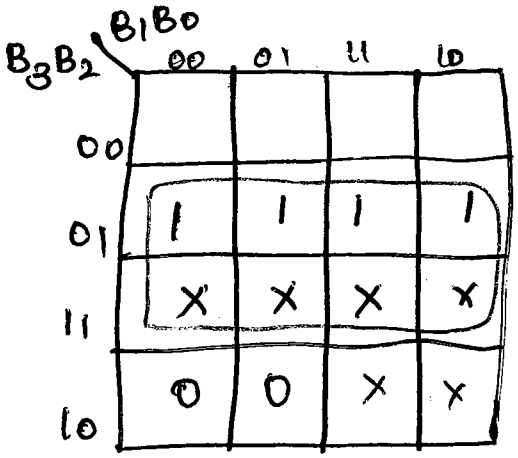
$B_4 = 1$



$B = \bar{B}_1 B_4$

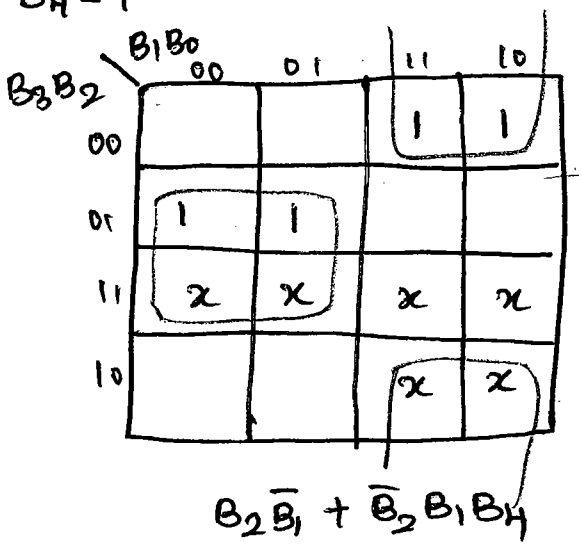
$$B = B_1 \bar{B}_4 + \bar{B}_1 B_4.$$

for C $B_4=0$



$C = \bar{B}_4 B_2$

$B_4 = 1$

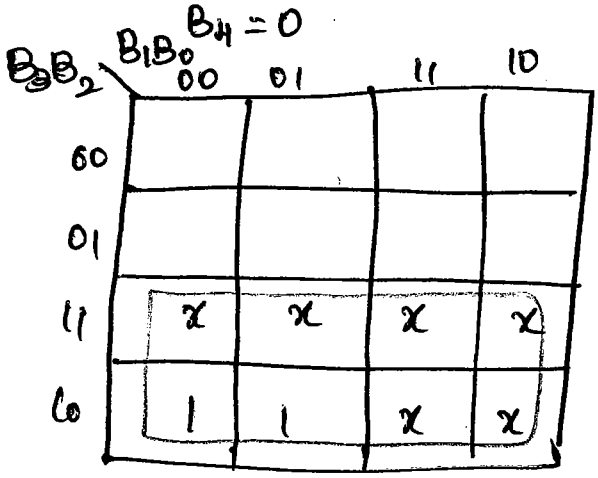


$B_2\bar{B}_1 + \bar{B}_2 B_1 B_4$

$C = \bar{B}_4 B_2 + B_2\bar{B}_1 + \bar{B}_2 B_1 B_4$

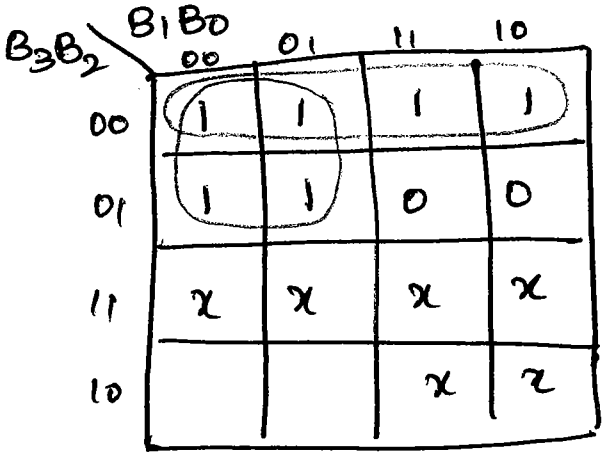
for D

$B_4 = 0$



$D = \bar{B}_4 B_3$

$B_4 = 1$

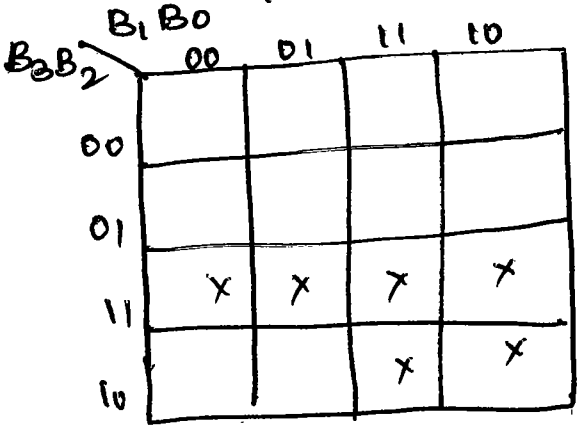


$B_4\bar{B}_3\bar{B}_1 + B_4\bar{B}_3\bar{B}_2$

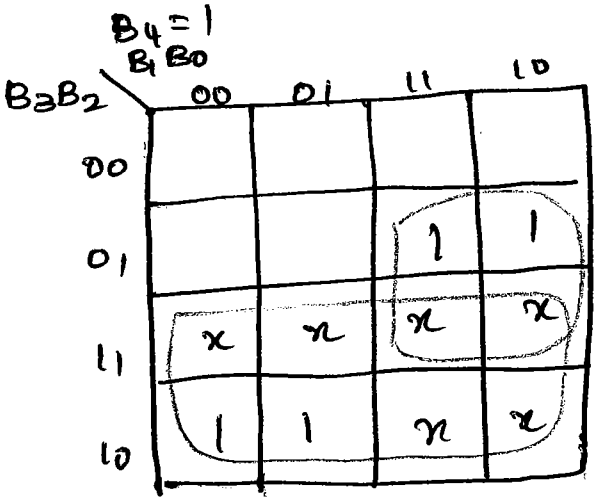
$D = \bar{B}_4 B_3 + B_4\bar{B}_3\bar{B}_1 + B_4\bar{B}_3\bar{B}_2$

for E

$B_4 = 0$

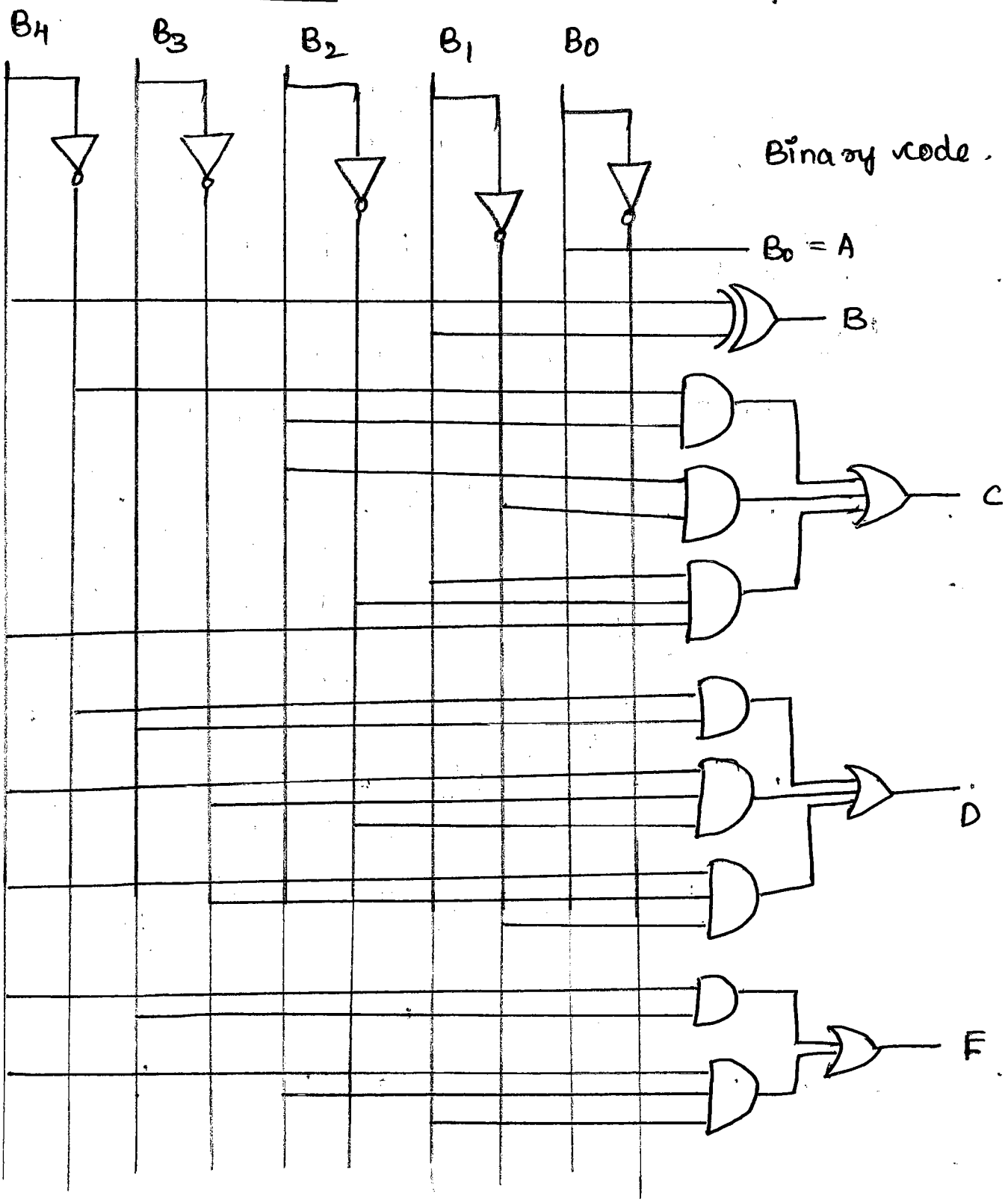


$B_4 = 1$



$E = B_3 B_4 + B_4 B_2 B_1$

Logic Circuit.



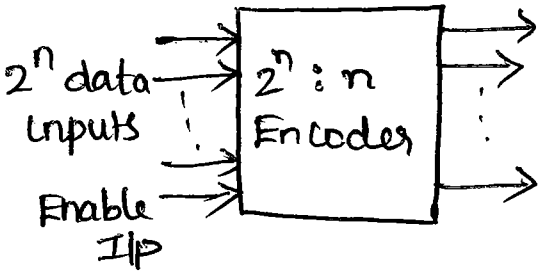
BCD - Binary Code Converter

Encoders:

It performs the invert operation of decoder.

It has 2^n Input lines and n output lines.

The output lines generate the binary code corresponding to input value.



n = no. of inputs
$n-1$ = select lines
2^n-1 = Data lines.

* "An encoder is a device whose inputs are decimal digits and/or alphabetic characters and whose outputs are the coded representation of those inputs".
* Encoding is the process of converting familiar numbers or symbols into a coded format.
* An encoder 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

Example

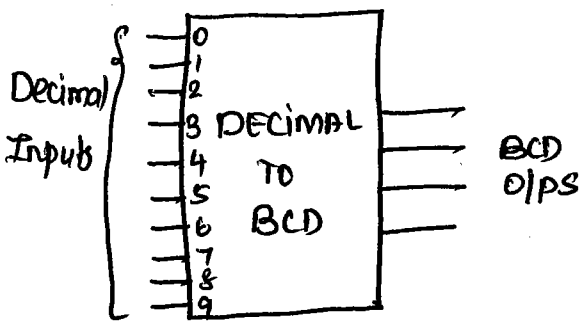
if there are M inputs and N outputs.

Normally the inputs are active low except one.

Decimal to BCD Encoder:

* It has 10 Input lines and 4 output lines.
* The BCD code is listed in the truth table and from this we can determine the relationship between each BCD bit and the decimal digit.
* There is no explicit input for a decimal 0.
* The BCD output is 0000 when the decimal inputs 1-9 are all 0.

Logic symbol.



Truth Table.

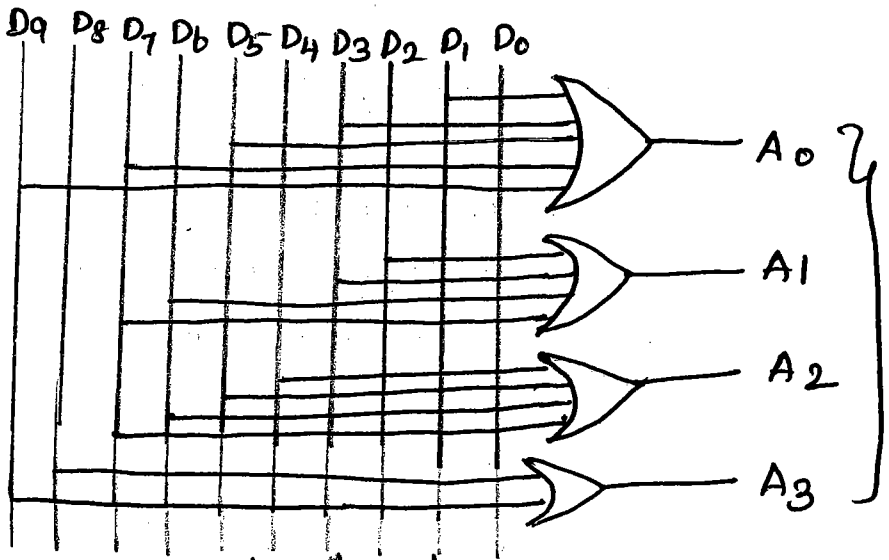
Decimal inputs		BCD O/P. A3	A2	A1	A0
D_0	0	0	0	0	0
D_1	1	0	0	0	1
D_2	2	0	0	1	0
D_3	3	0	0	1	1
D_4	4	0	1	0	0
D_5	5	0	1	0	1
D_6	6	0	1	1	0
D_7	7	0	1	1	1
D_8	8	1	0	0	0
D_9	9	1	0	0	1

$A_0 = D_1 + D_3 + D_5 + D_7 + D_9$.

$A_1 = D_2 + D_3 + D_6 + D_7$

$A_2 = D_4 + D_5 + D_6 + D_7$

$A_3 = D_8 + D_9$.



Logic diagram.

Octal to Binary Encoder.

* 8 input lines, 3 output lines and produces a 3-bit output code corresponding to the activated input.

* Only one input has a value of 1 at any given time, otherwise the circuit is meaningless.

Truth Table.

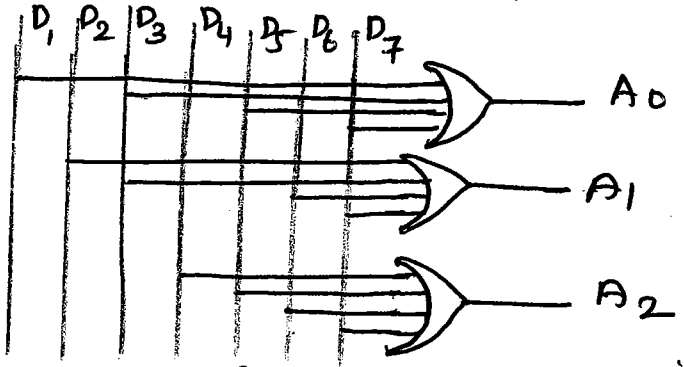
Octal digits		Binary A_2	A_1	A_0
D_0	0	0	0	0
D_1	1	0	0	1
D_2	2	0	1	0
D_3	3	0	1	1
D_4	4	1	0	0
D_5	5	1	0	1
D_6	6	1	1	0
D_7	7	1	1	1

from truth table

$A_0 = D_1 + D_3 + D_5 + D_7$

$A_1 = D_2 + D_3 + D_6 + D_7$

$A_2 = D_4 + D_5 + D_6 + D_7$



Here D_0 is not available in any expression so D_0 → Don't care.

PRIORITY ENCODER:

* The encoder discussed so far will operate correctly provided that one and only one decimal input is high at any given time.

* If two or more decimal inputs may become high at the same time.

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 priority will be based on the magnitude of the input, whichever is the largest is the one that is encoded.

Example 4-bit priority Encoder.

Inputs				output		
D_0	D_1	D_2	D_3	A	B	V
0	0	0	0	X	X	0
01	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

from the Truth table

$A = D_3 + D_2$

B =

$V = D_0 + D_1 + D_2 + D_3$

* Higher the subscript number, higher the priority of the input.

* So, D_3 has the highest priority

D_0 has the lowest priority.

* When D_3 is high regardless of other inputs, output is 11.

* D_2 has the next priority; $D_3 = 0$, $D_2 = 1$ output is 10

* The output for D_1 is generated only if higher priority inputs are zero

V => Valid output indicators.

A, B => two outputs.

* If all inputs are 0, V = 0, A, B not used.

V -> indicates, one or more of the inputs are equal to 1

K-map for A

D_0D_1 \ D_2D_3	00	01	11	10
00	x	1	1	1
01	0	1	1	1
11	0	1	1	1
10	0	1	1	1

K-map for B

D_0D_1 \ D_2D_3	00	01	11	10
00	x	1	1	0
01	1	1	1	0
11	1	1	1	0
10	0	1	1	0

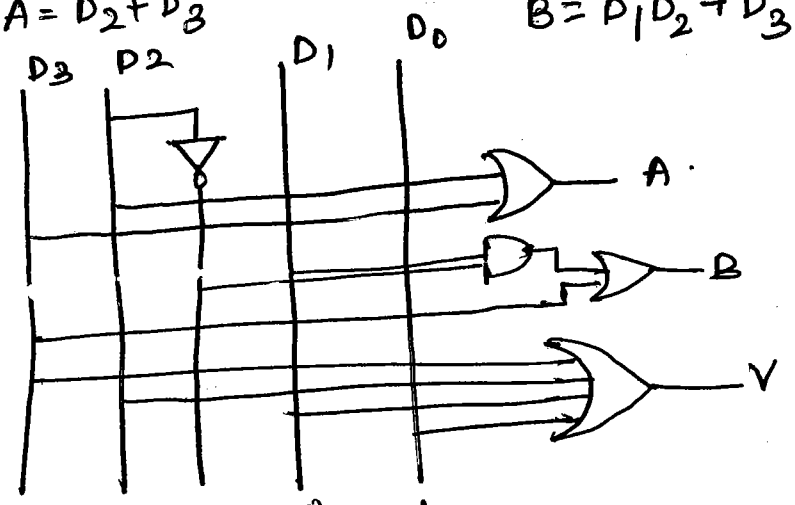
K-map for V

D_0D_1 \ D_2D_3	00	01	11	10
00	0	1	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	1	1

$A = D_2 + D_3$

$B = D_1\bar{D}_2 + D_3$

$V = D_1 + D_2 + D_3 + D_0$



Logic diagram.

MULTIPLEXERS: (Data selector)

↓ sharing.

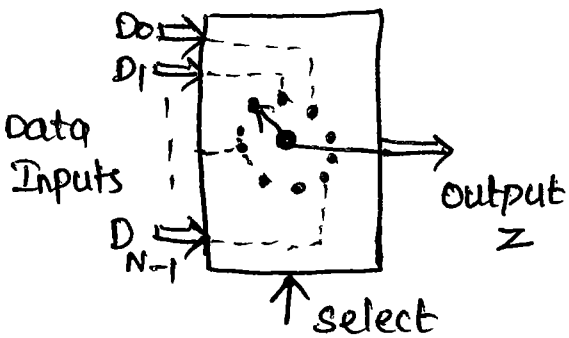
There are two types of multiplexing 1. Time multiplexing. 2. frequency multiplexing.

Time multiplexing: At any given time, one and only one device is using the line. since each device is given specific time intervals to use the line.

frequency multiplexing: several devices share a common line by transmitting at different frequencies.

* Multiplexer or data selector is a logic circuit that accepts several data inputs and allows only one of them at a time to get through to the output.
* The routing of the desired data input to the output is controlled by SELECT inputs.

Functional diagram of Multiplexer.



* inputs and outputs are drawn as large arrows to indicate that they may constitute one or more signal lines.

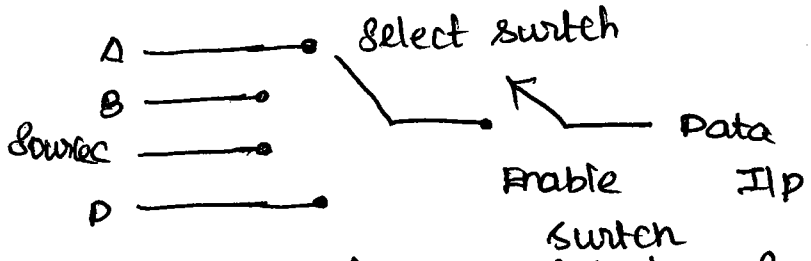
2^n - input lines

n - select lines whose bit combinations determine which input is selected.

Operation:

It acts like a digitally controlled multi-position switch. The digital code applied to the select inputs determines which data inputs will be switched to the outputs.

* "Multiplexer selects 1 out of N input data sources and transmits the selected data to a single output channel. called Multiplexing".



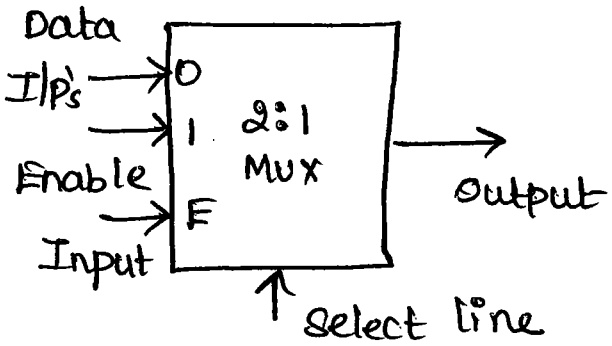
Analog Selector Switch

Example 1. 2 : 1 Multiplexer.

Inputs = $2^1 = 2$

Output = 1

Select lines = 1



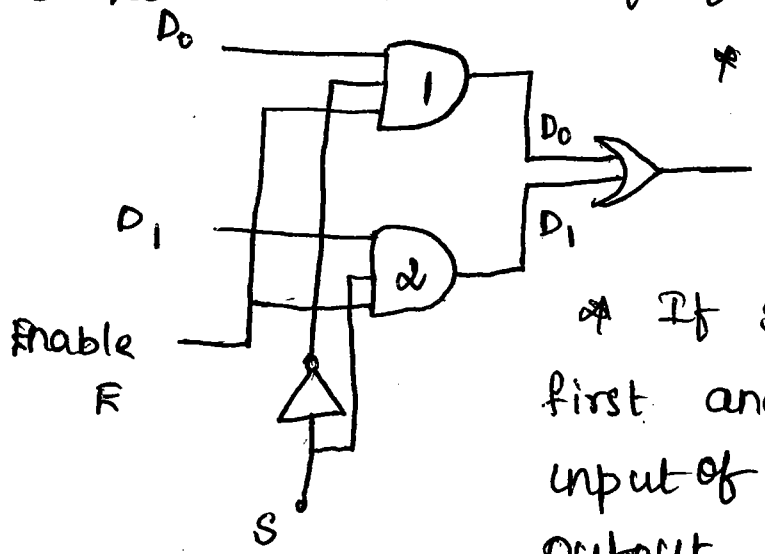
Truth Table.

E	S	Y
1	0	D_0
1	1	D_1
0	X	0

Working:

(i) when E = 0; O/p Y = 0 irrespective of I/p the circuit works only when E = 1.

* when S = 0, the inverted S, that is 1 gets applied as second input to first AND gate. Since S is applied directly as input to second AND gate; its output goes zero irrespective of first input.



* O/p is $Y = D_0$ since I/p of first AND gate is 1.

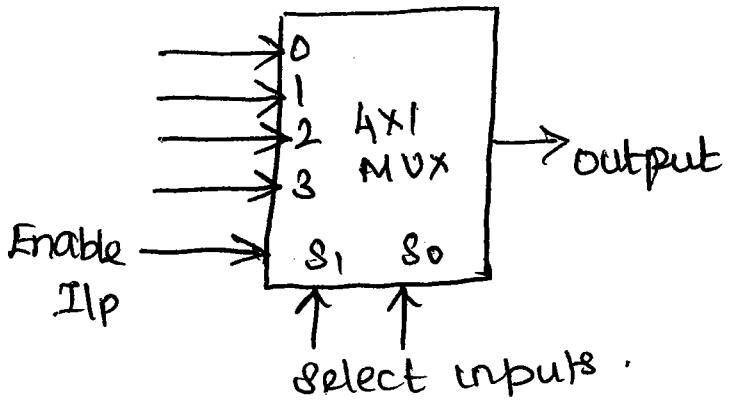
* If S = 1, second input of first and gate is 0 and second input of second gate is 1 so output is $Y = D_1$

Expression of the logic circuit is $y = D_0\bar{S}F + D_1SF$.

Truth table for 2:1 Mux.

F	S	D_1	D_0	y	
1	0	X	0	0	
1	0	X	1	1	$F\bar{S}D_0$
1	1	0	X	0	
1	1	1	X	1	FSD_1
0	X	X	X	0	

4:1 Multiplexer:



Function table.

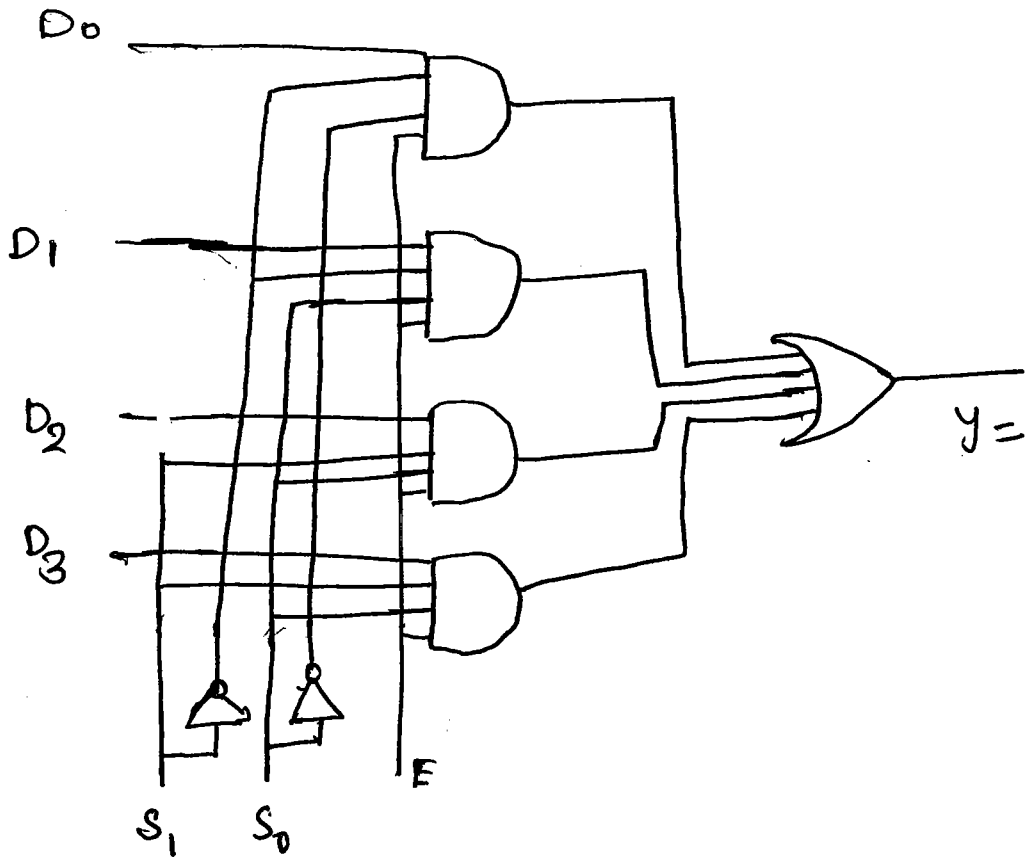
F	S_1	S_0	Y
1	0	0	D_0
1	0	1	D_1
1	1	0	D_2
1	1	1	D_3
0	X	X	0

When $S_1S_0 = 00$ AND gate associated with data input D_0 has two of its I/p equal to 1 and third I/p is D_0

The other three AND gates have atleast one I/p = 0 which makes their o/p = 0.

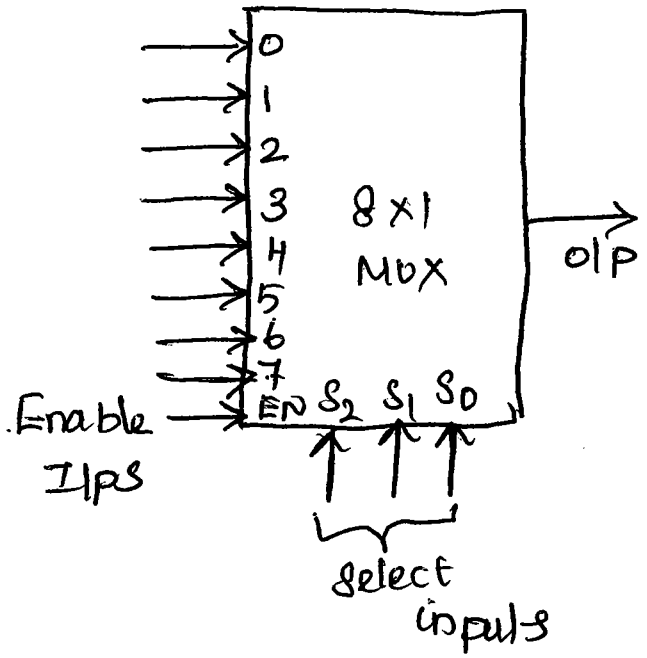
The OR gate o/p = D_0, D_0 is routed to the output y when $S_1S_0 = 00$.

Logic diagram.



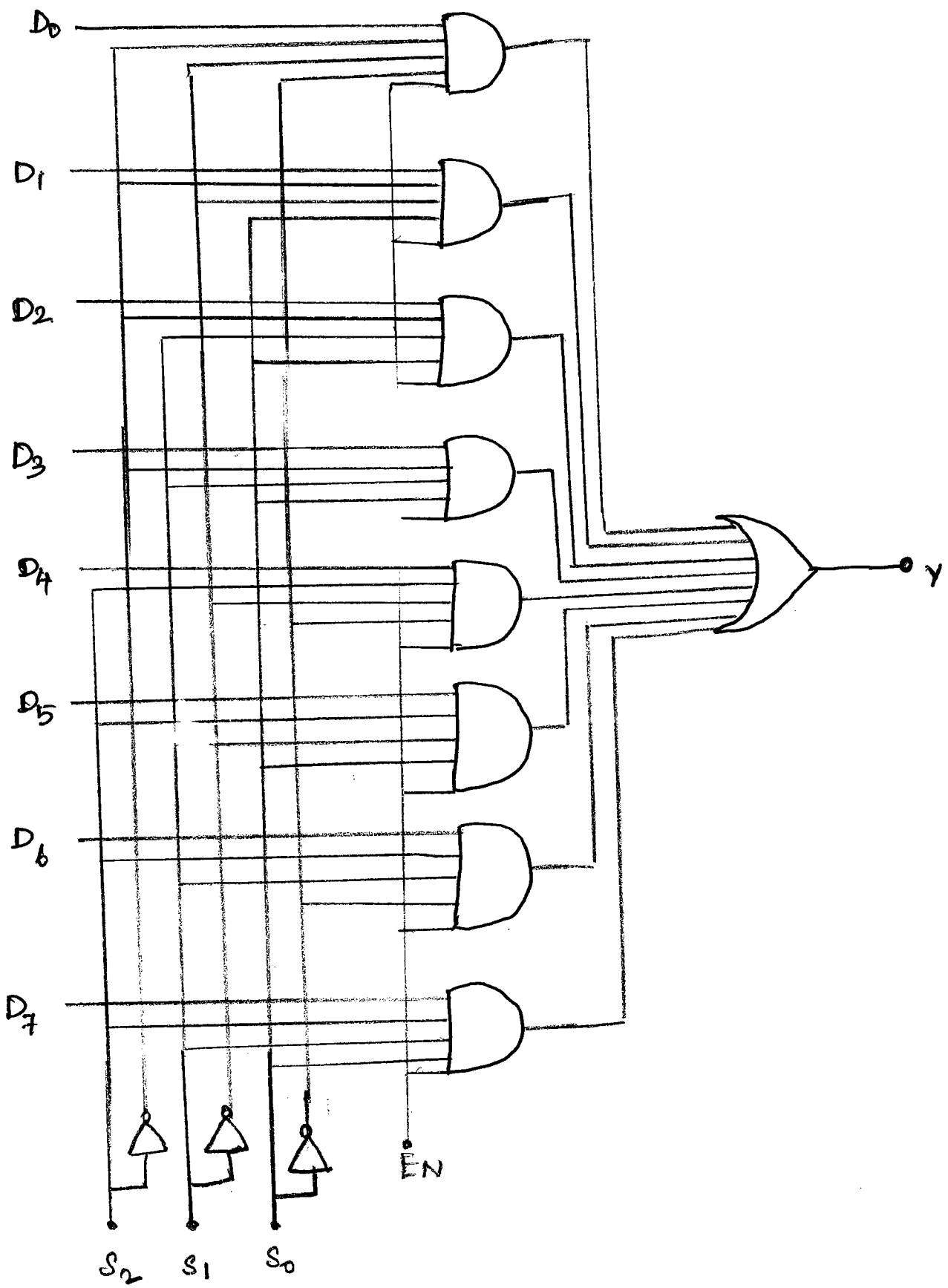
$$Y = E\bar{S}_1\bar{S}_0D_0 + E\bar{S}_1S_0D_1 + E S_1\bar{S}_0D_2 + E S_1S_0D_3.$$

8 : 1 Multiplexer.



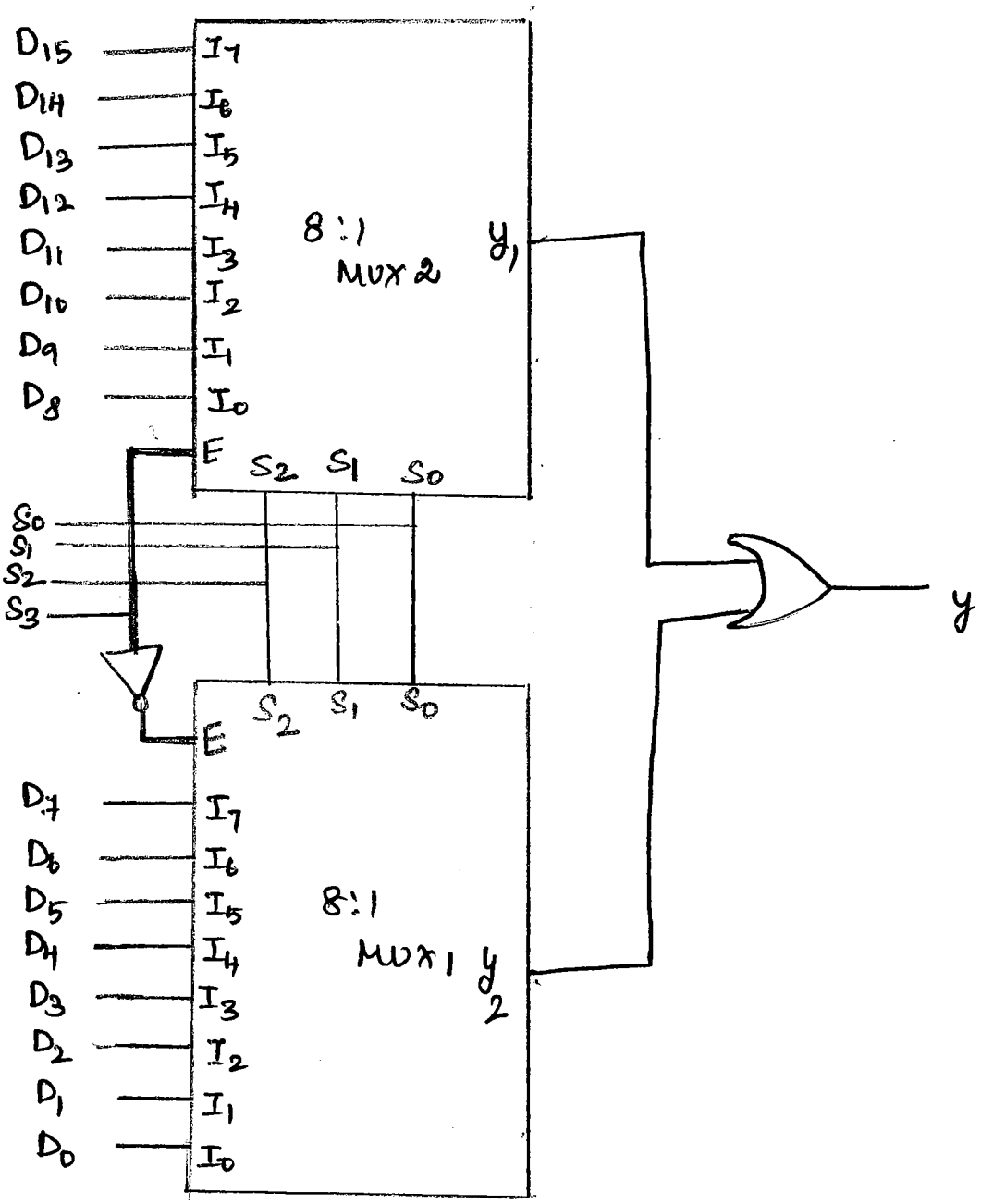
S_2	S_1	S_0	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

Logic diagram of 8 : 1 MUX.



Example:

1. Design 16:1 MUX using 8:1 MUX.

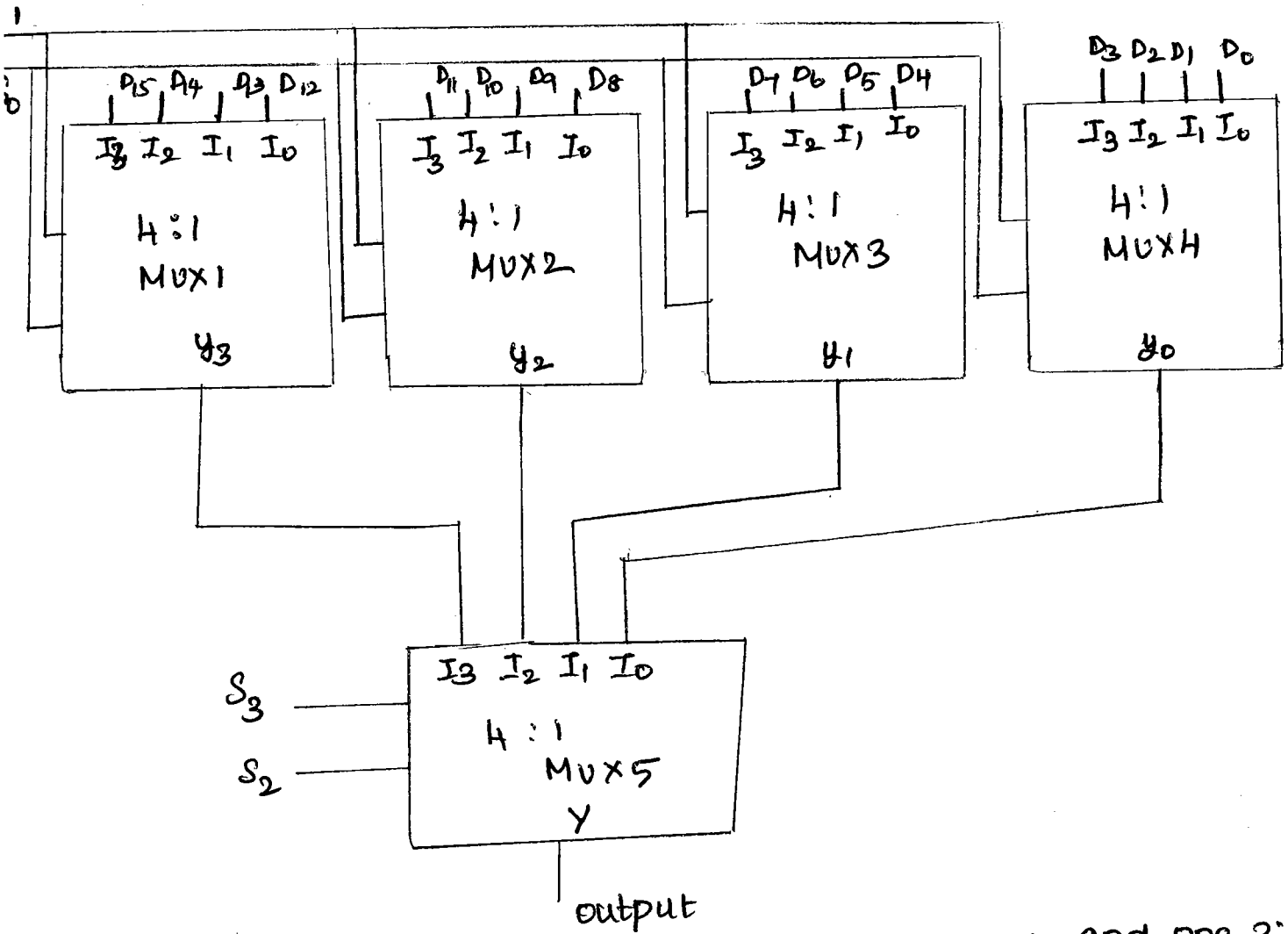


* S_1, S_2, S_0 are the select lines of two multiplexes connected in parallel.

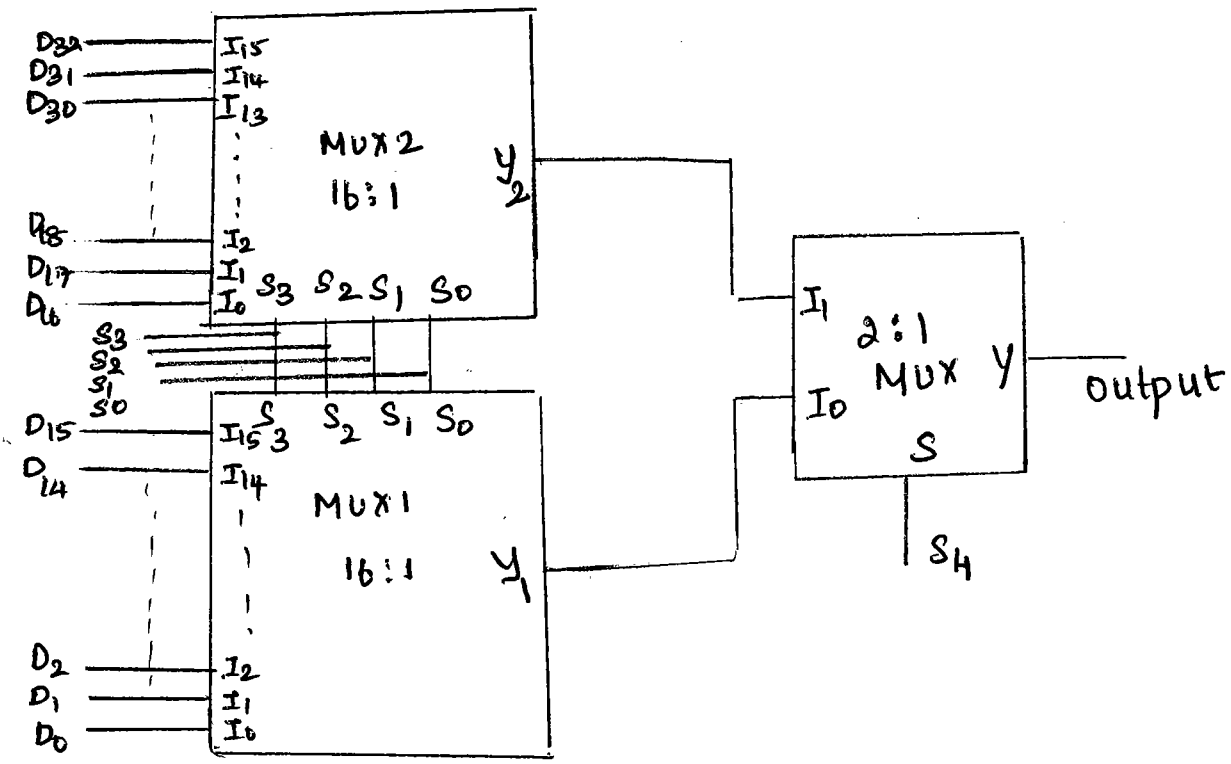
* Connect most significant select line S_3 such that when $S_3 = 0$ MUX1 is Enabled and $S_3 = 1$, MUX2 is Enabled.

* O/P of both mux are added and final O/P is Y.

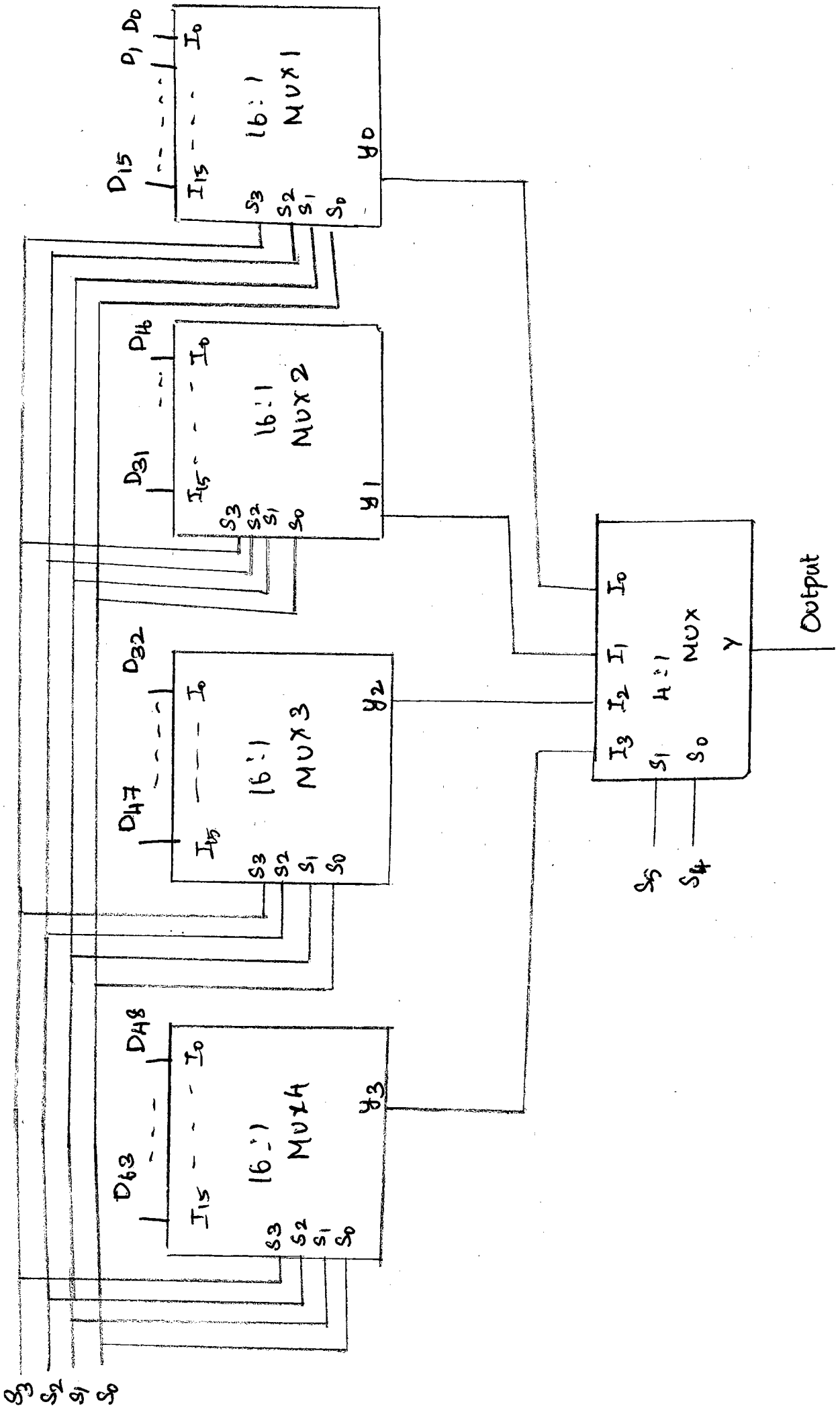
2. Design 16:1 MUX using 4:1 MUX.



3. Draw 32:1 multiplexer using two 16:1 mux and one 2:1 mux.



4. Draw 64:1 Multiplexer tree using 16:1 MUX



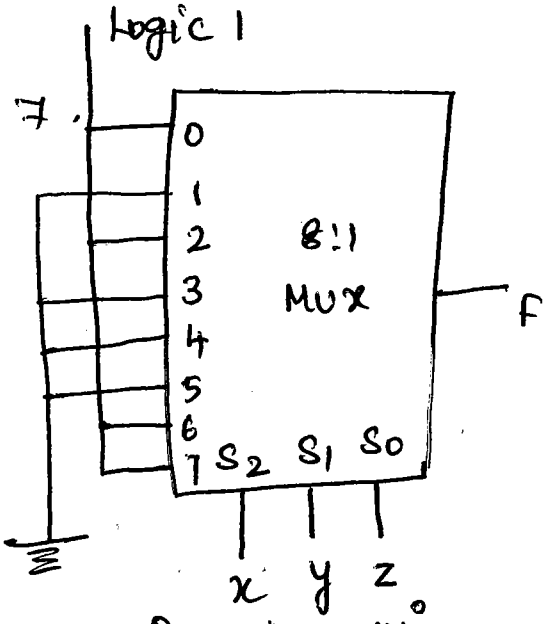
5. Implement the given function using multiplexer

$f(x,y,z) = \Sigma(0,2,6,7)$

Logic 1 for 0, 2, 6, 7.

$2^3 = 8:1$ MUX

Logic 0 for 1, 3, 5, 4



6. Implementation of following Boolean function using 4:1 mux.

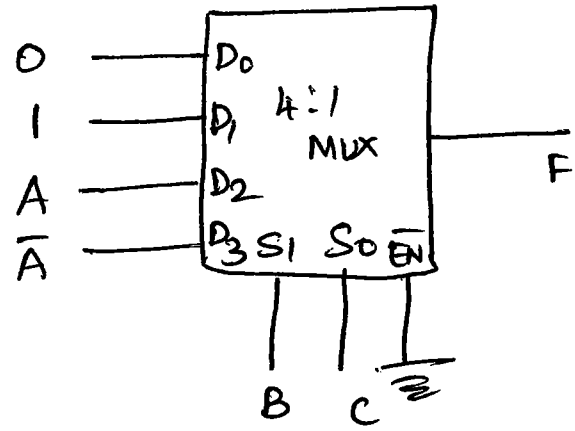
$f(A,B,C) = \Sigma m(1,3,5,6)$

Logic 1 for 1, 3, 5 6

Logic 0 for 0, 2, 4, 7

	D_0	D_1	D_2	D_3
$\bar{A}$	0	(1)	2	(3)
A	4	(5)	(6)	7
	0	1	A	$\bar{A}$

Implementation table



* If minterms in a column are not circled 0 is applied.
* If both are circled 1 is applied
* If row 2 is encircled A is applied.
* If row 1 is encircled $\bar{A}$ is applied.

Note: Implementation table is drawn by the Condition that

row 1 → with minterms where A is Complemented form.

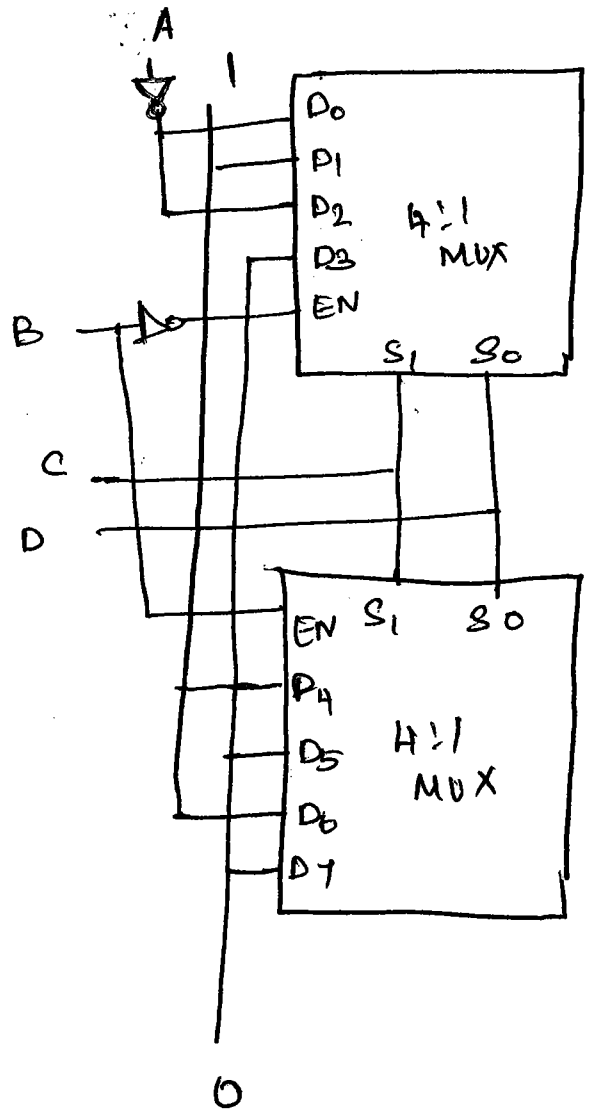
row 2 → with minterms where A is in uncomplemented form.

7. Implement the following Boolean function using 4:1 MUX

$f(A,B,C,D) = \Sigma m(0,1,2,4,6,9,12,14)$.

Implementation table.

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	(0)	(1)	(2)	3	(4)	5	(6)	7
A	8	(9)	10	11	(12)	13	(14)	15
	$\bar{A}$	1	$\bar{A}$	0	1	0	1	0



8. Implement the following function using 8:1 mux.

$F(A,B,C,D) = \bar{A}B\bar{D} + ACD + \bar{B}CD + \bar{A}\bar{C}D.$

Solution:

To standard SOP form

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

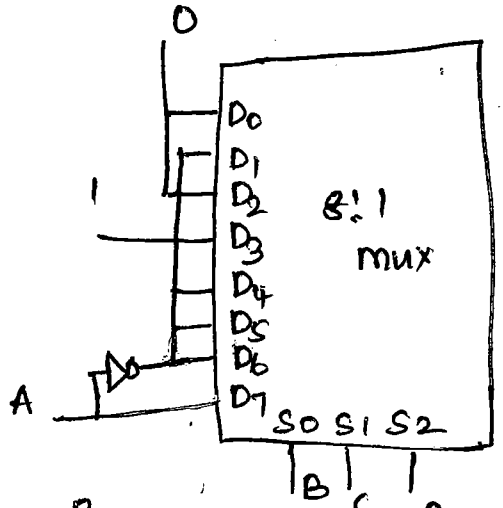
$= \bar{A}B\bar{D}C + \bar{A}B\bar{C}\bar{D} + ABCD + A\bar{B}CD + A\bar{B}CD + \bar{A}\bar{B}CD + \bar{A}B\bar{C}D + AB\bar{C}D + \bar{A}\bar{B}\bar{C}D.$

Truth Table:

AB CD		
0000	0	
0001	(1)	$\bar{A}\bar{B}\bar{C}D$
0010	2	
0011	(3)	$\bar{A}\bar{B}CD$
0100	(4)	$\bar{A}B\bar{C}\bar{D}$
0101	(5)	$\bar{A}B\bar{C}D$
0110	(6)	$\bar{A}BC\bar{D}$
0111	7	
1000	8	
1001	9	
1010	10	
1011	(11)	$A\bar{B}CD$
1100	12	
1101	13	
1110	14	
1111	(15)	$ABCD$

Implementation table

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	0	(1)	2	(3)	(4)	(5)	(6)	7
A	8	9	10	(11)	12	13	14	(15)
	0	$\bar{A}$	0	1	$\bar{A}$	$\bar{A}$	$\bar{A}$	A



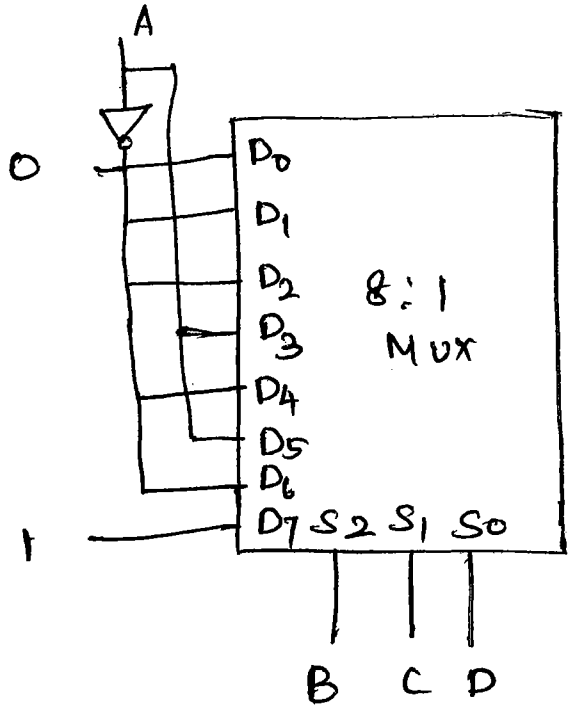
9. Implement the following Boolean function with 8:1 mux

$F(A,B,C,D) = \Pi M(0,3,5,8,9,10,12,14).$

Implementation Table

	0	$\bar{A}$	$\bar{A}$	A	$\bar{A}$	A	$\bar{A}$	1
	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	0	(1)	(2)	3	(4)	5	(6)	(7)
A	8	9	10	(11)	12	(13)	14	(15)

Note: Here the terms whichever not in the function is encircled because of POS function.



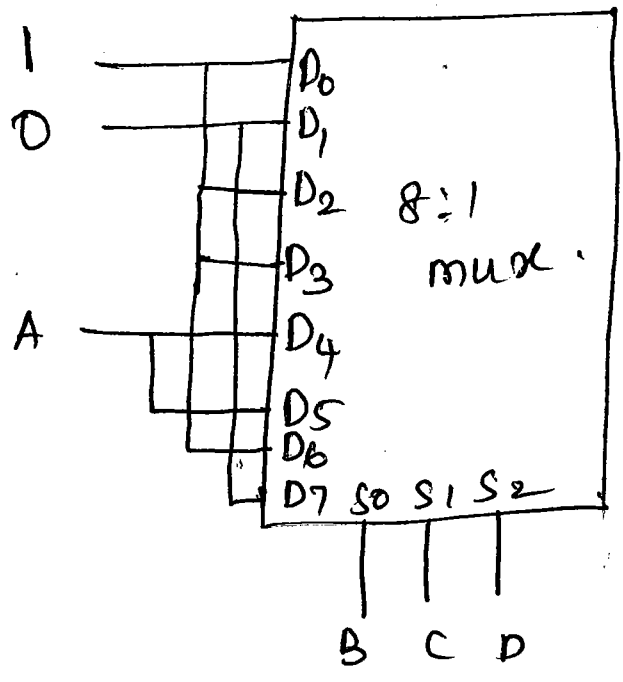
10) Implement with 8:1 mux

$f(A, B, C, D) = \Sigma m(0, 2, 6, 10, 11, 12, 13) + d(3, 8, 14)$.

don't care = 1

Implementation table

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
$\bar{A}$	(0)	1	(2)	(3)	4	5	(6)	7
A	(8)	9	(10)	(11)	(12)	(13)	(14)	15
	1	0	1	1	A	A	1	0



11. Implement full adder using two 4:1 mux.

Variables A, B, C

select inputs = 2 $\rightarrow n-1 = 3-1$

data inputs $2^n - 1 = 2^{(3-1)} = 4$

Sum = $\Sigma m(1, 2, 4, 7)$

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

A	B	C	Sum	Carry
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

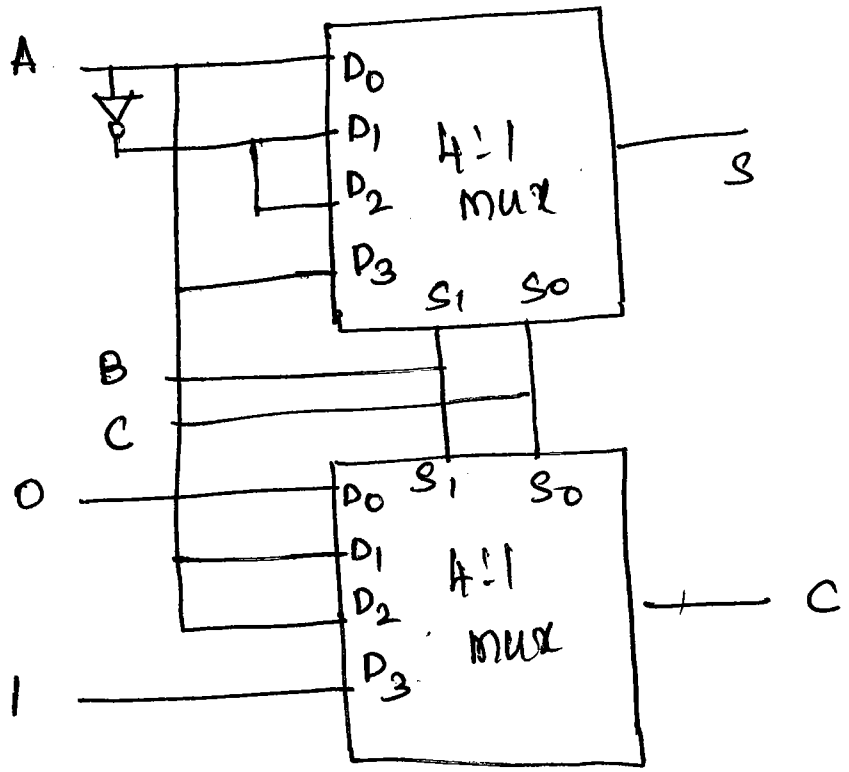
Implementation table

Sum

	D_0	D_1	D_2	D_3
$\bar{A}$	0	(1)	(2)	3
A	(4)	5	6	(7)
	A	$\bar{A}$	$\bar{A}$	A.

Carry

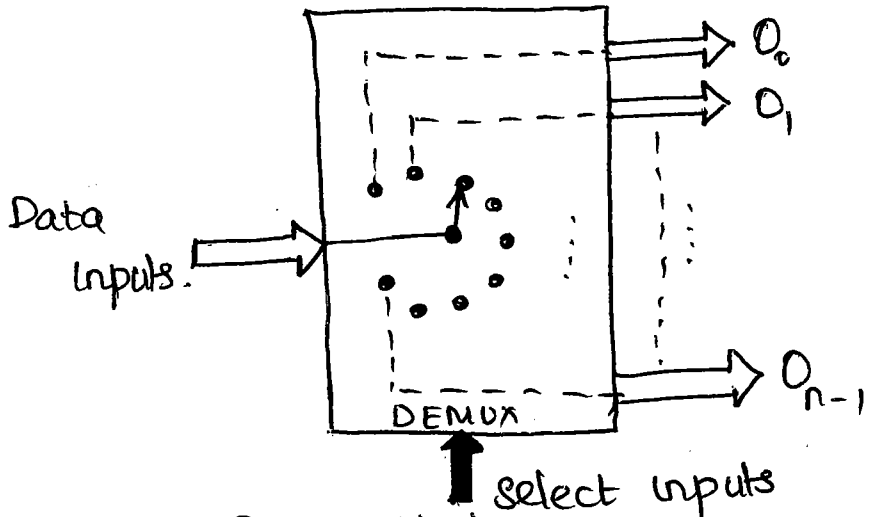
	D_0	D_1	D_2	D_3
$\bar{A}$	0	1	2	(3)
A	4	(5)	(6)	(7)
	0	A	A	1



DEMULTIPLEXER [DATA DISTRIBUTORS]

* Performs the reverse operation; it takes a single input and distributes it over several outputs.
* Demultiplexer is called as data distributors, since it transmits the same data to different destinations.
* The select input code determines the output line to which the input data will be transmitted.

functional diagram of Demultiplexer.

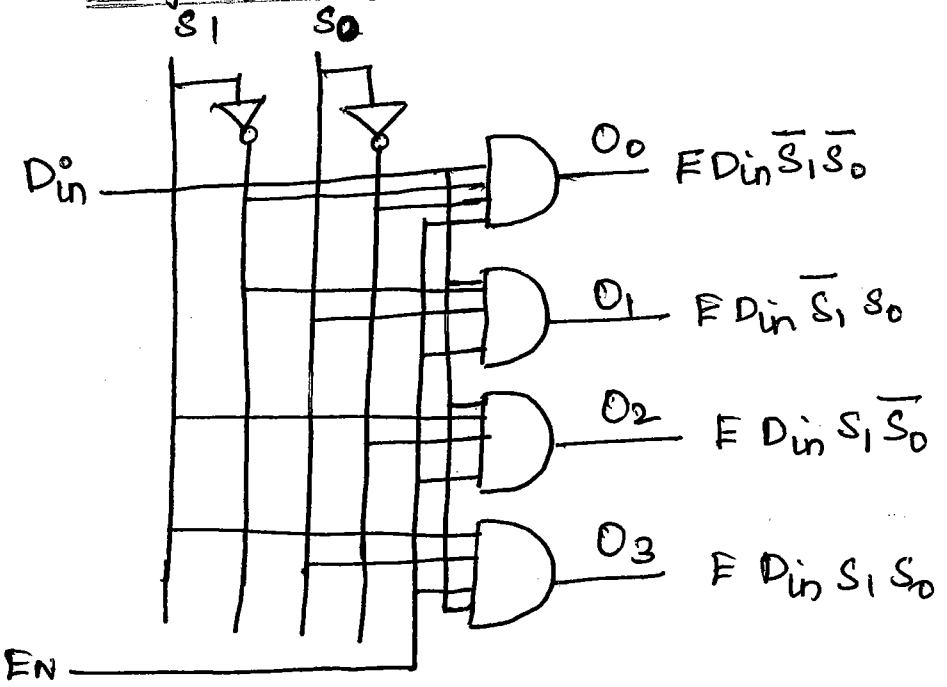


Types of De multiplexer:

(1) 1 : 4 DEMUX.

* Single input variables Din has a path to all four outputs, but the input information is directed to only one of the output lines depending on the select inputs.
* Enable input should be high to enable demux.
* The two select lines S_0 and S_1 enable only one gate at a time, the data appearing on the input line will pass through the selected gate to the associated output line.

Logic diagram.



functional table of 1:4 demultiplexer

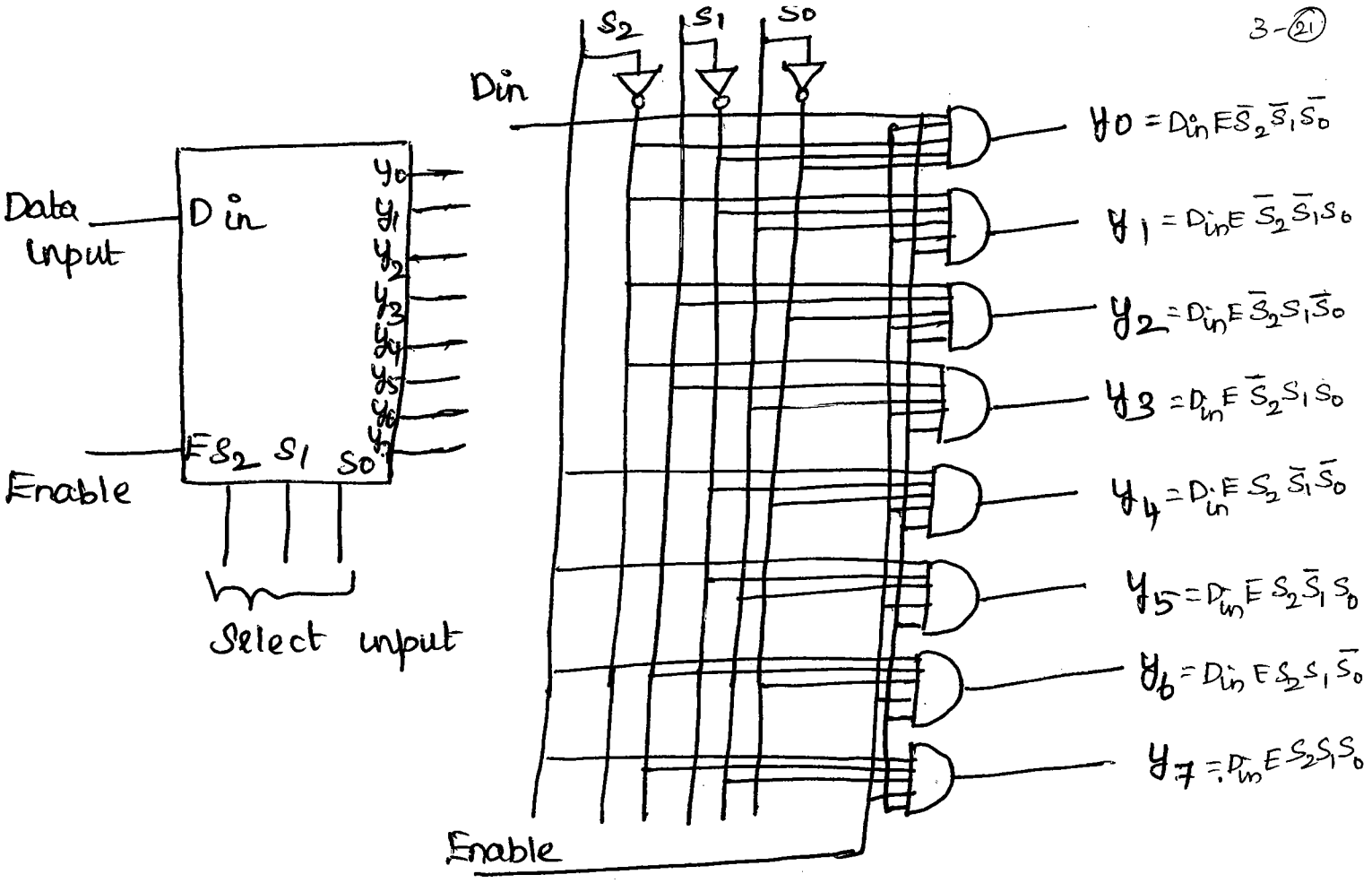
Select Code		output			
S_1	S_0	O_3	O_2	O_1	O_0
0	0	D_{in}	0	0	0
0	1	0	0	D_{in}	0
1	0	0	D_{in}	0	0
1	1	D_{in}	0	0	0

1:8 Demultiplexer:

D_{in} is given to all the AND gates

Only one of these gates will be enabled by the select input lines

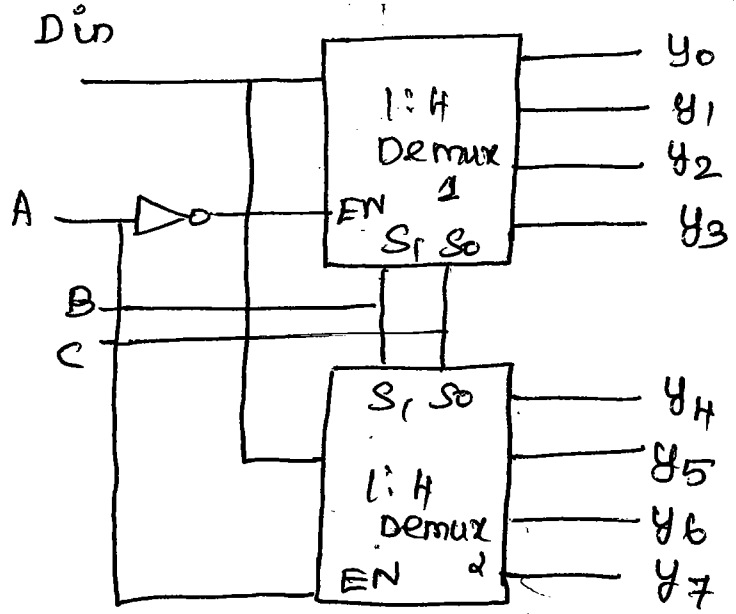
No. of select lines is three, S_0, S_1 and S_2.



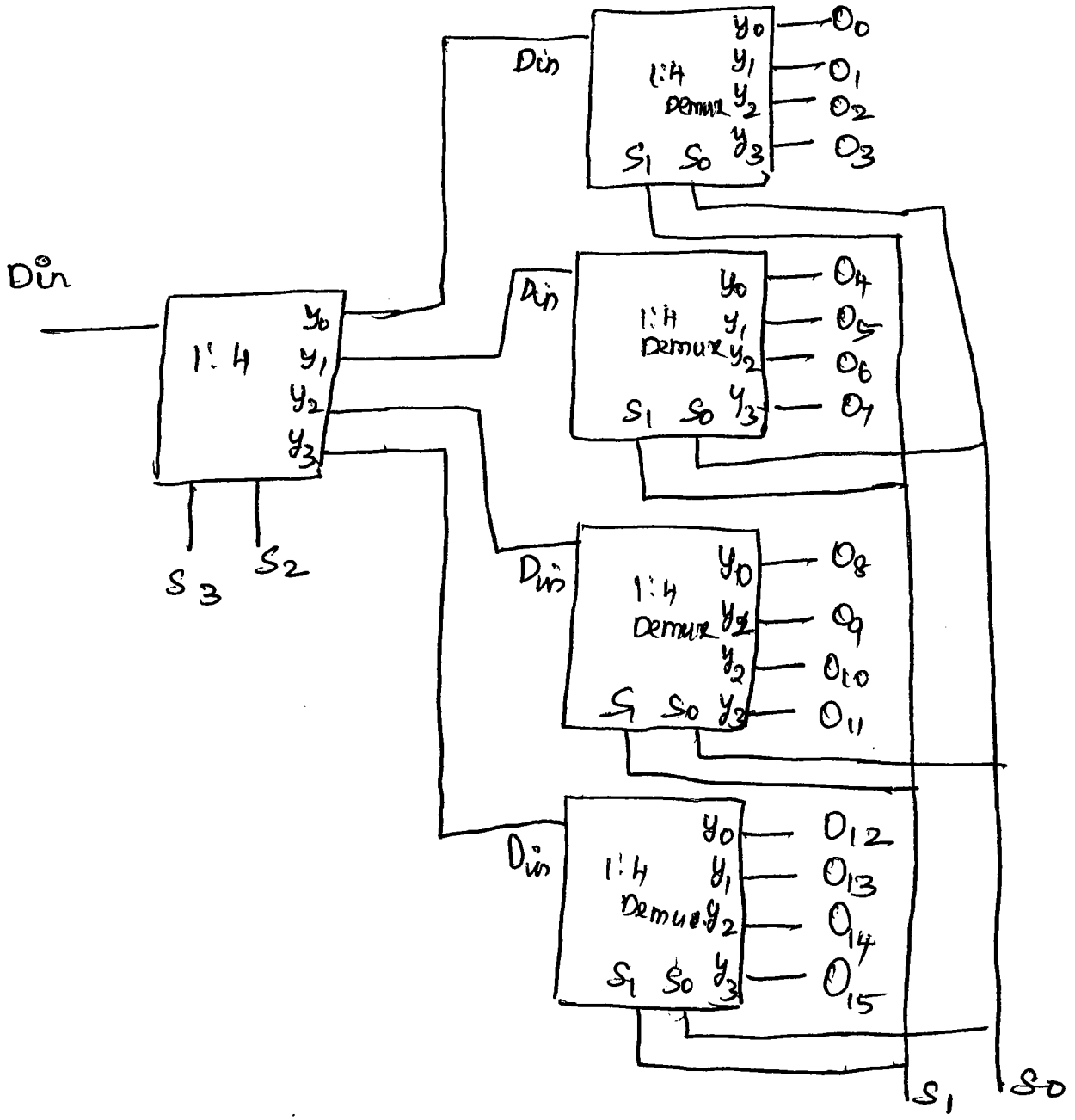
Truth Table:

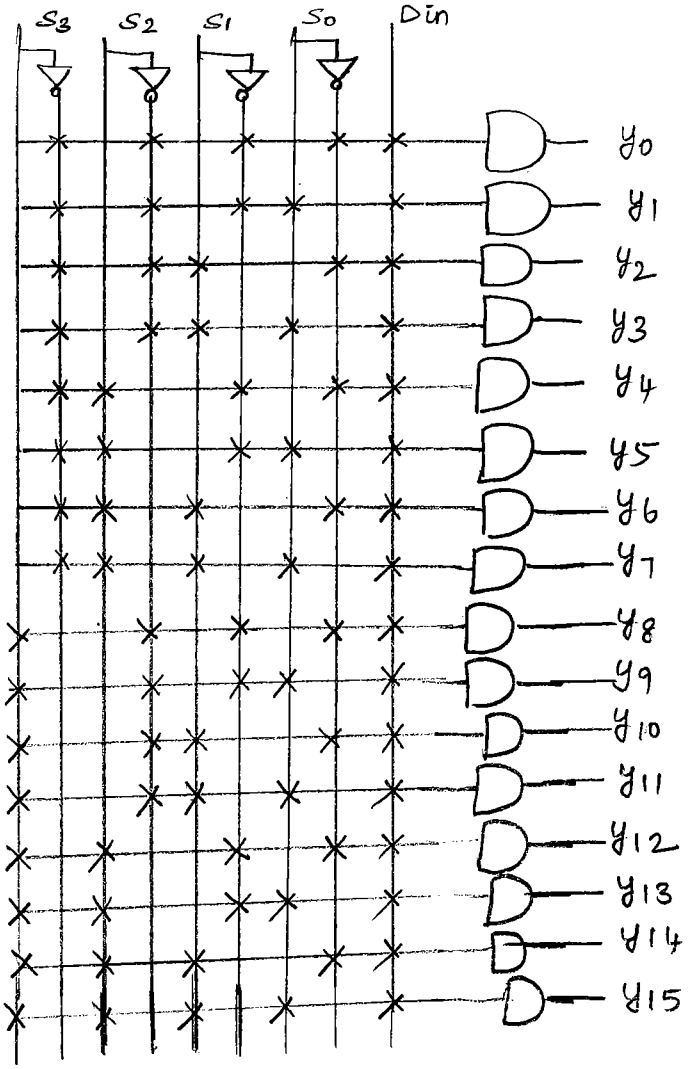
Select Code			outputs							
S_2	S_1	S_0	O_7	O_6	O_5	O_4	O_3	O_2	O_1	O_0
0	0	0	0	0	0	0	0	0	0	D_{in}
0	0	1	0	0	0	0	0	0	D_{in}	0
0	1	0	0	0	0	0	0	D_{in}	0	0
0	1	1	0	0	0	0	D_{in}	0	0	0
1	0	0	0	0	0	D_{in}	0	0	0	0
1	0	1	0	0	D_{in}	0	0	0	0	0
1	1	0	0	D_{in}	0	0	0	0	0	0
1	1	1	D_{in}	0	0	0	0	0	0	0

1. Design 1:8 demux using two 1:4 demux.



2. Implement 1:16 demux using 1:4 Demux.





Logic diagram of 1:16 Demux.

3. Implementation of full subtractor using Demux.

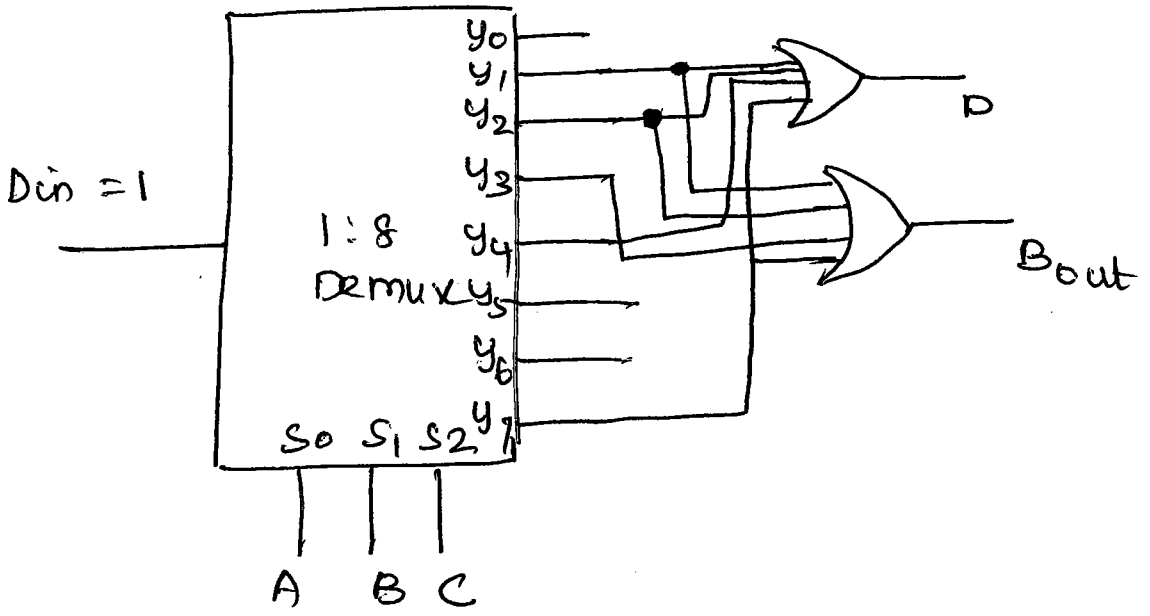
Truth table

A	B	C	Sub	Bout
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

Diff = Sub = $f(A,B,C) = \Sigma m(1,2,4,7)$

Bout = $f(A,B,C) = \Sigma m(1,2,3,7)$.

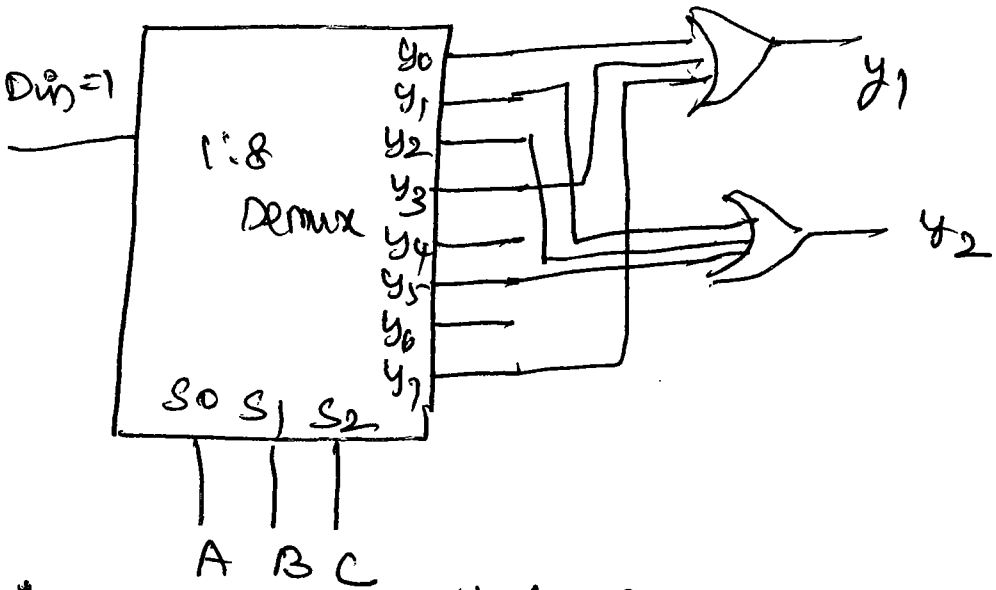
* with $D_{in} = 1$, demux gives the minterms at the o/p by ORing required minterms



Implement the following function using Demux.

$f_1(A,B,C) = \Sigma m(0,3,7)$

$f_2(A,B,C) = \Sigma m(1,2,5)$.



Applications of De-multiplexer.

1. It can be used as decoder
2. It can be used as data distributor
3. It is used in time division multiplexing at the receiving end as a data seperator.
4. It can be used to implement Boolean expressions.

Magnitude Comparator:

Comparator is a special Combinational Circuit designed primarily to compare the relative magnitude of two binary numbers.

n bit binary numbers as input A and B.

The outputs are $A>B$, $A<B$, $A=1$.

Depending upon the relative magnitude of the two number, one of the outputs will be high.

Design a two bit Comparator using gates.

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

$A>B$

A_1A_0 \ B_1B_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

$A>B = A_0\bar{B}_1\bar{B}_0 + \bar{B}_1A_1 + A_1A_0\bar{B}_0$

$A=B$

A_1A_0 \ B_1B_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

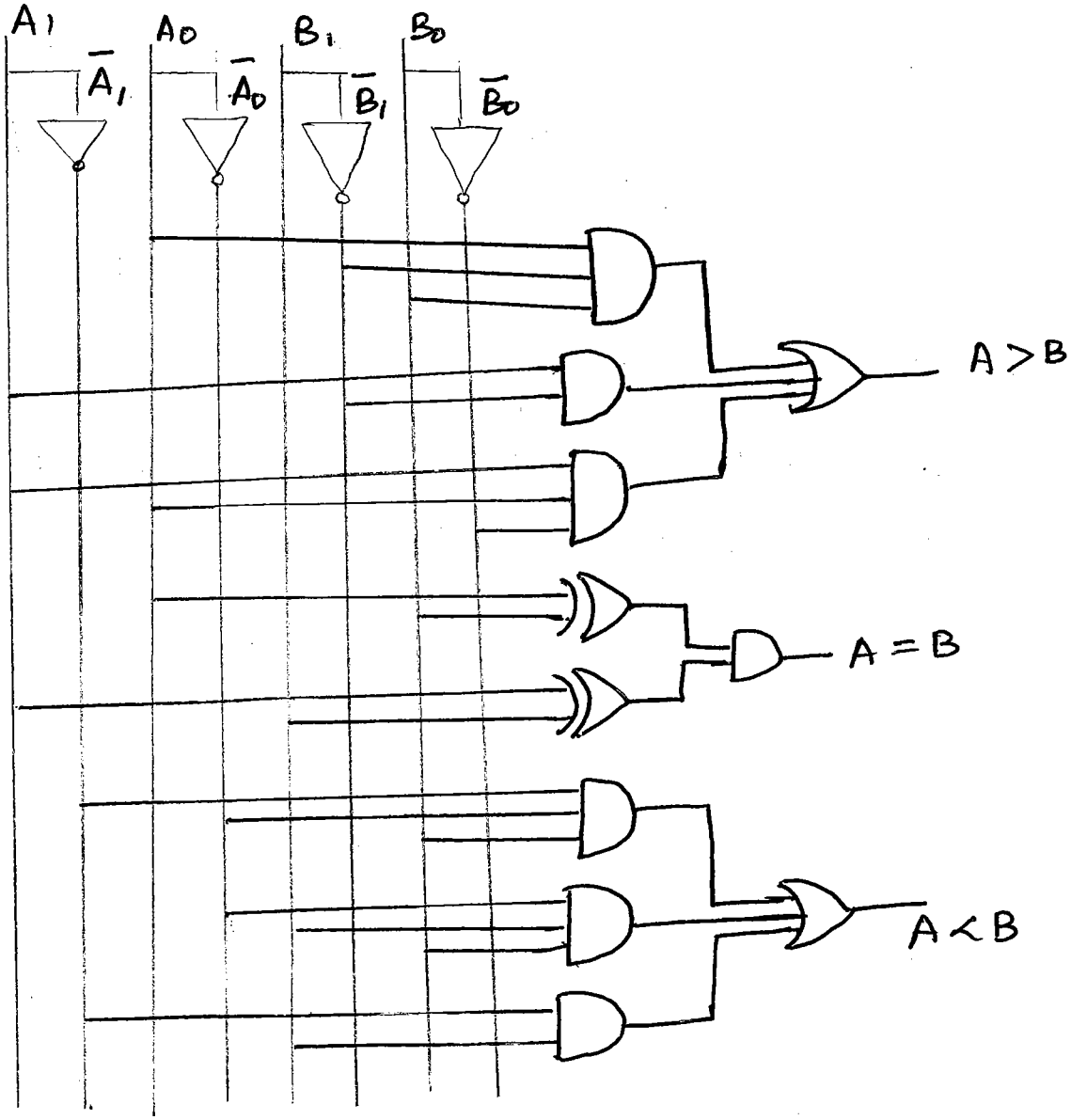
$= \bar{A}_1\bar{A}_0\bar{B}_1\bar{B}_0 + \bar{A}_1A_0\bar{B}_1B_0 + A_1A_0B_1B_0 + A_1\bar{A}_0B_1\bar{B}_0$

$= (A_0 \odot B_0)(A_1 \odot B_1)$

$A<B$

A_1A_0 \ B_1B_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

$\bar{A}_1\bar{A}_0B_0 + \bar{A}_0B_1B_0 + \bar{A}_1B_1$


A1
A0
B1
B0
$\bar{A}_1$
$\bar{A}_0$
$\bar{B}_1$
$\bar{B}_0$
A > B
A = B
A < B