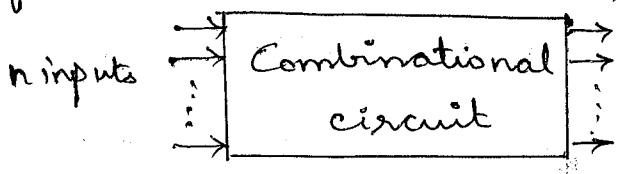
COMBINATIONAL CIRCUITS.

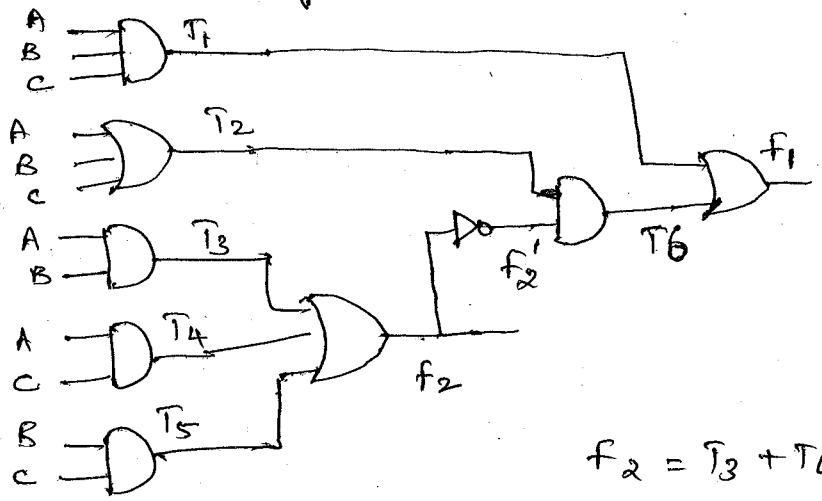
- Logic gates whose output at any time are determined from the present combination of input.
- performs an operation that can be specified logically by a set of boolean function
- No feedback.



ANALYSIS PROCEDURE

1. Label all gate output that are a function of input Variables with arbitary symbols. Determine the boolean function for each gate output.
2. Label the gates that are a function of input Variables and previously labelled gates with other arbitary symbols. find the boolean function for these gates.
3. Repeat the process until the outputs of the circuit are obtained.

1. Find boolean function and truth table for the circuit



$T_1 = ABC$

$T_2 = A + B + C$

$T_3' = AB$

$T_4 = AC$

$T_5 = BC$

$f_2 = T_3 + T_4 + T_5$

$= AB + AC + BC$ ——— (1)

$T_6 = T_2 f_2'$

$F_2' = (AB + AC + BC)'$

$= (AB)'(AC)'(BC)'$

$f_2' = (A'+B')(A'+C')(B'+C')$ —— (2)

$f_1 = T_1 + T_6$

$= ABC + T_2 F_2'$

$= ABC + [(A+B+C)(A'+B')(A'+C')(B'+C')]$

$= ABC + [(A+B+C)(A'+B'C')(B'+C')]$

$= ABC + [(A+B+C)(A'B'+A'C'+B'C'B'+B'C'C')]$

$= ABC + [AA'B' + AA'C' + AB'C' + A'B'B + A'C'B + BB'C' + A'B'C + A'C'C + B'C'C]$

$F_1 = ABC + AB'C' + A'BC' + A'B'C$ —— (3)

$F_2 = AB + AC + BC$ —— (4).

TRUTH TABLE.

Let 3 variable be A, B, C.

input			output	
A	B	C	$F_1 = ABC+AB'C'+A'BC'+A'B'C$	$F_2 = AB+AC+BC$
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

DESIGN PROCEDURE

1. From the specification of circuit, determine the required number of inputs and outputs and assign a symbol to each.
2. Obtain the truth table that defines the required relationship between input and output.
3. Obtain the simplified boolean function using K-map.
4. Draw logic diagram.

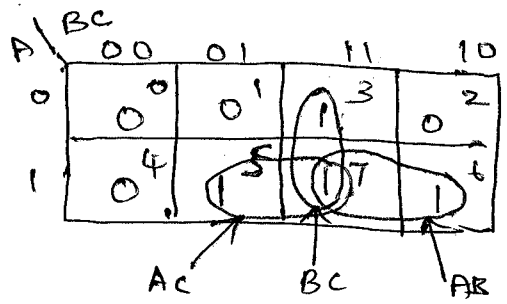
---

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

① Truth table

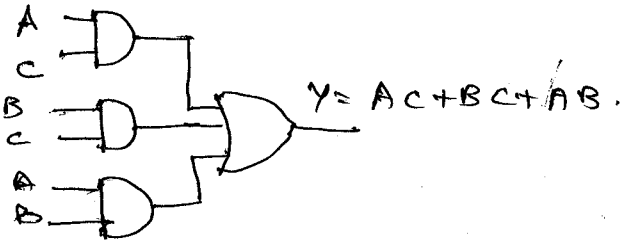
input			Y
A	B	C	
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

② K-map.



$Y = AC + BC + AB$

③ Logic diagram AND-OR



ADDER:

Addition consist of four possible elementary operations

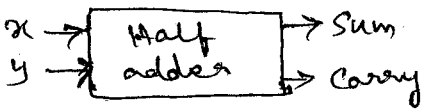
$0+0=0$
$0+1=1$
$1+0=1$ → The combinational circuit that performs the addition of two bit is called half adder.
$1+1=10_2$ → addition of three bit is a full adder.

HALF ADDER

- 2 bit addition, has 2 inputs x and y and two output sum and carry.

Augend → x

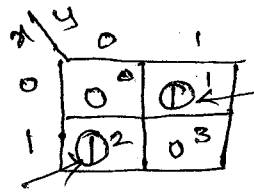
Addend → y



Truth table

Input		Output	
x	y	Carry	Sum
0	0	0	0
0	1	0	1
1	0	0	1
1	1	0	0

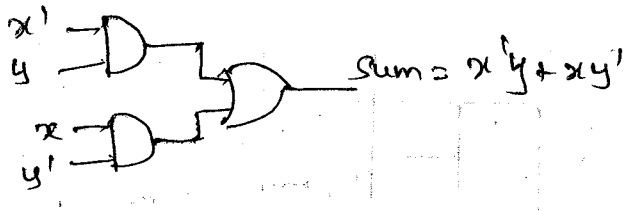
Sum k-map



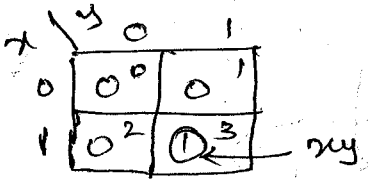
Sum = $x'y + xy'$ (SOP)

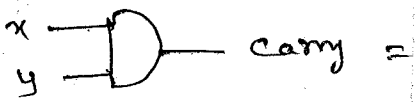
$= x \oplus y$ (Xor)

SOP implementation AND-OR

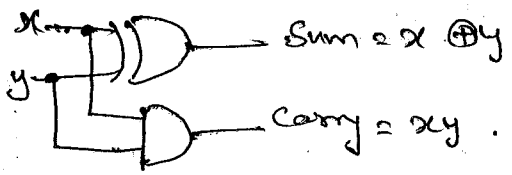


Carry k-map.





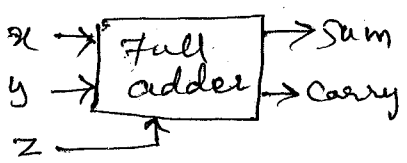
EX-OR implementation



* Carry output is one only when both inputs are one
* Sum represents the least significant bit.

FULL ADDER

* forms arithmetic sum of three bits.
* consist of 3 inputs and 2 outputs
* x, y input represent the two significant bits to be added and the third input z, represents the carry from the previous lower significant position
* 2 output sum and carry.



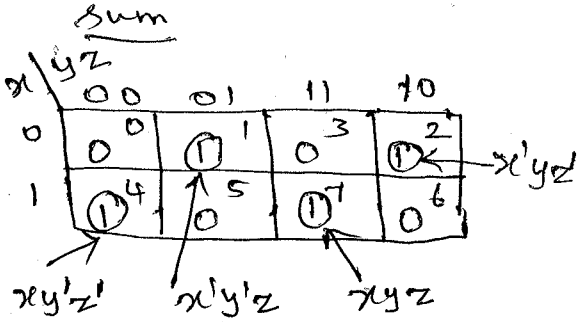
* When all input bits are '0', the output is '0'.
* When only one input is equal to '1' then the sum = 1.

* If 2 or 3 inputs are equal to 1, carry = 1.

Truth table

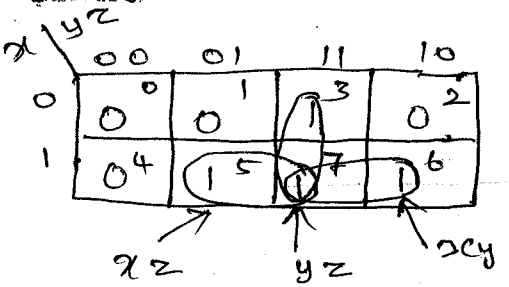
input			output	
x	y	z	Carry	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Sum



Sum = $xy'z' + x'y'z + xyz + x'yz'$ (SOP)

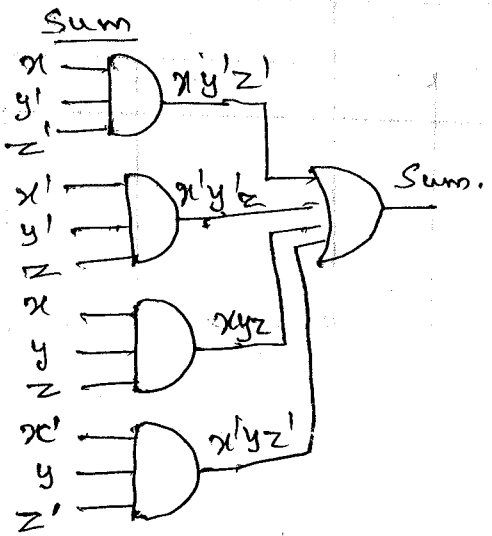
Carry



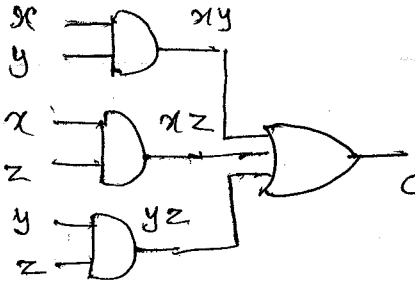
Carry = $xz + yz + xy$ (SOP)

SOP IMPLEMENTATION

Sum



Carry



EX-OR Implemtation → Sum

Sum = $x'y'z + x'yz' + xy'z' + xyz$

$= z(x'y' + xy) + z'(x'y + xy')$

$= z(x \oplus y)' + z'(x \oplus y)$

B A' B' A

Sum = $z \oplus (x \oplus y)$

EX-OR implementation

Carry = $xy + xz + yz$

$= xy + z(x+y)$

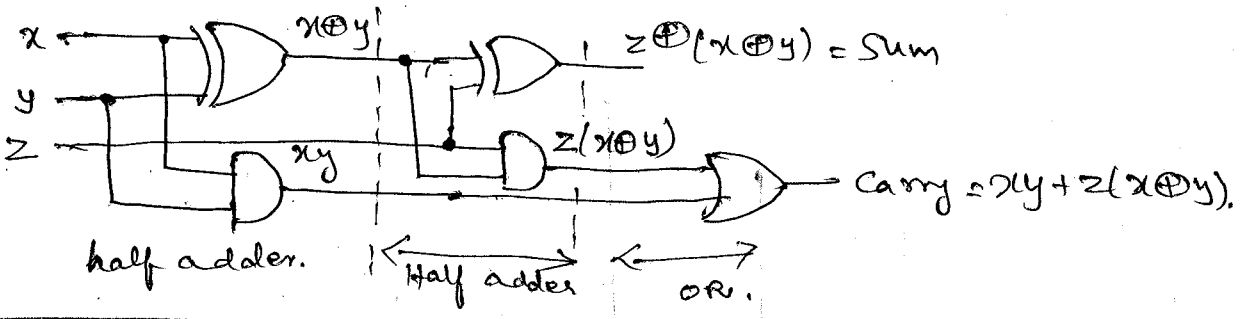
$= xy + z[x(y+y') + y(x+x')]$

$= xy + z[xy + xy' + xy + x'y]$

$= xy + xyz + z(x \oplus y)$

$= xy(1+z) + z(x \oplus y)$

Carry = $xy + z(x \oplus y)$



SUBTRACTOR

Minuent Subtrahend

0 − 0 = 0

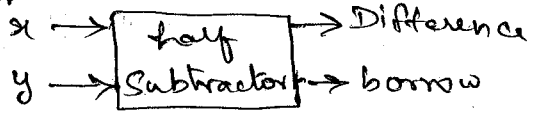
0 − 1 = 1 with borrow 1

1 − 0 = 1

1 − 1 = 0

HALF SUBTRACTOR.

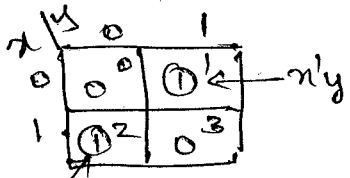
* 2 bit subtraction, has 2 input x, y and 2 output difference and borrow.



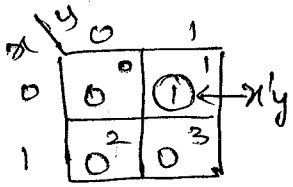
Truth table

input		output	
x	y	Difference	borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

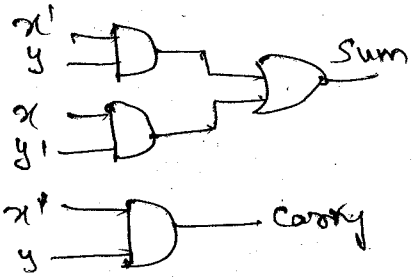
Difference



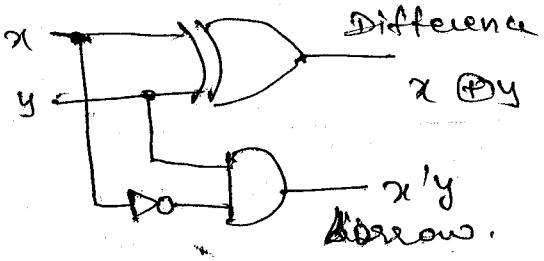
Borrow



SOP Implementation

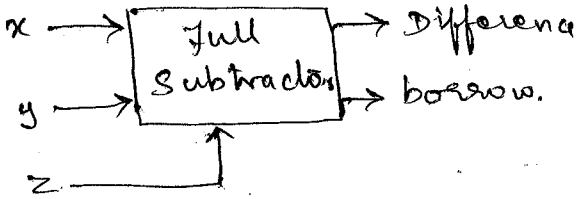


EX-OR implementation



FULL SUBTRACTOR

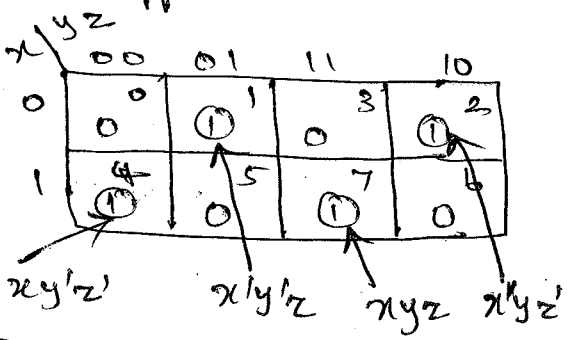
* It has 3 inputs and 2 outputs.
* x, y and z are inputs to be subtracted in which z represents borrow from the next stage.



Truth table

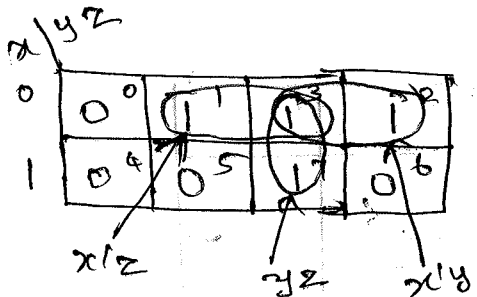
input			output	
x	y	z	Difference	borrow
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Difference



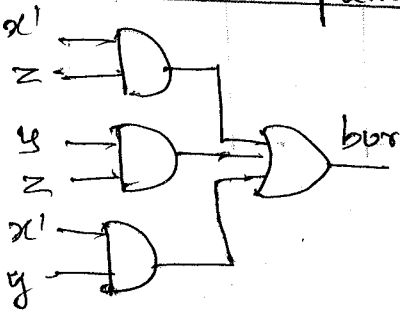
Difference = $xy'z' + x'y'z + xyz + x'yz'$

Borrow



Borrow = $x'z + yz + x'y$.

borrow - SOP implementation



Borrow - Ex-OR implementation

Borrow = $x'z + yz + x'y$

$= z(x'+y) + x'y$

$= z[x'(y+y') + y(x+x')] + x'y$

$= z(x'y + x'y' + xy + x'y) + x'y$

$= z(x'y + (x \oplus y)') + x'y$

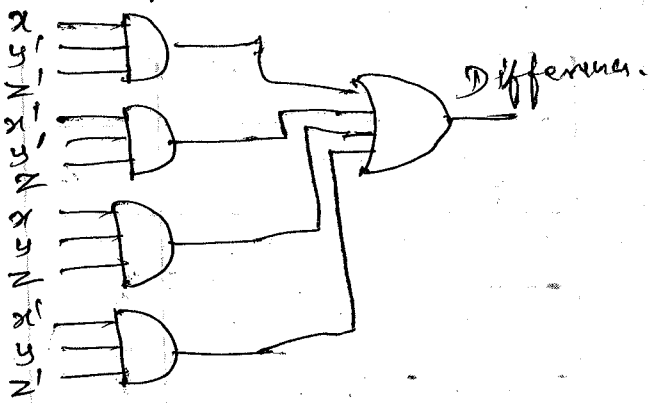
$= x'yz + z(x \oplus y)' + x'y$

$= x'y(1+z) + z(x \oplus y)'$

Borrow $= x'y + z(x \oplus y)'$

SOP implementation

Difference



EX-OR implementation

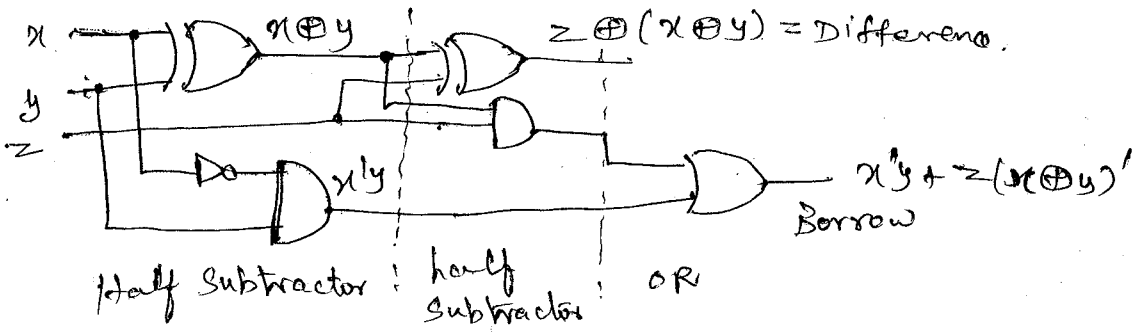
Difference = $xy'z' + x'y'z + xyz + x'yz'$

$= z'(xy' + x'y) + z(xy + x'y')$

$= z'(x \oplus y) + z(x \oplus y)'$

B' A B A'

$= z \oplus (x \oplus y)$



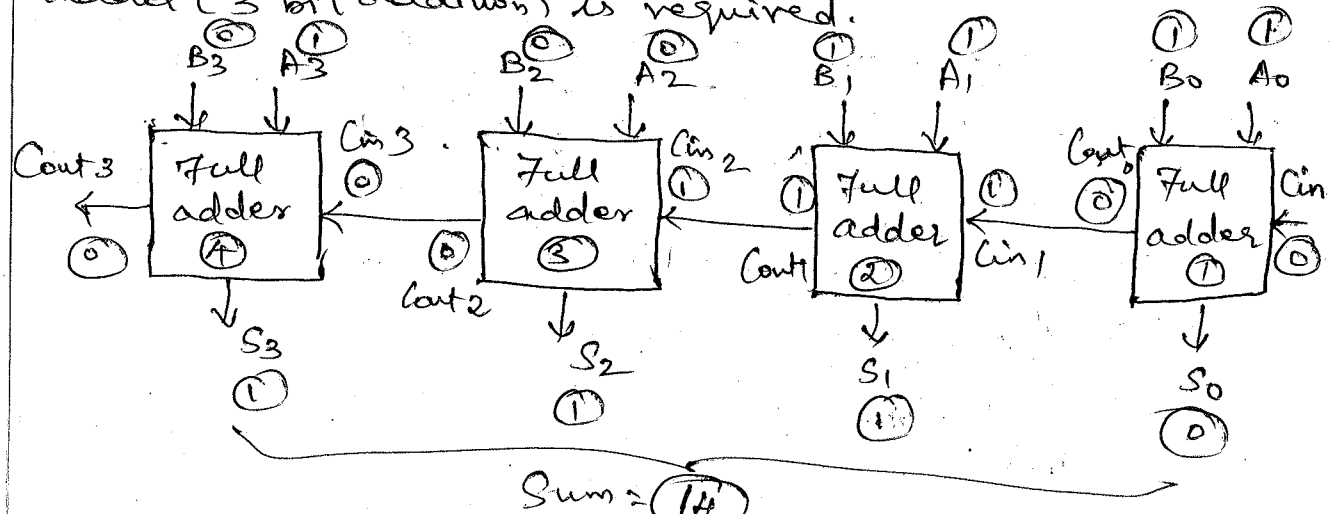
4 bit binary adder

- digital circuit that produces the arithmetic sum of two binary numbers.
- Constructed with full adder connected in cascade with output carry from each full adder connected as the input carry for the next full adder.

Let us consider two 4 bit number added taken as A_i and B_i.

Subscript i	3	2	1	0	
input carry C_i	0	1	1	0	
Augend A_i	1	0	1	1	(11)
Addend B_i	0	0	1	1	(3) 3 bit full adder
Sum S_i	1	1	1	0	(14) Ans.
Output carry C_{i+1}	0	0	1	1	

* for adding two → 4 bit binary numbers we need a bit input carry ∴ for each bit addition one full adder (3 bit addition) is required.



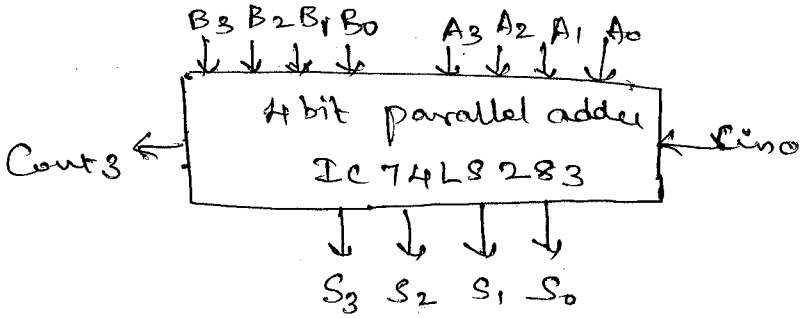
* Always the input carry bit of LSB C_{in0} is equal to zero ($C_{in0} = 0$) in full adder 1.

* The output of full adder 1 has sum and carry, this carry is connected as the input carry C_{in1} of full adder 2 and continues....

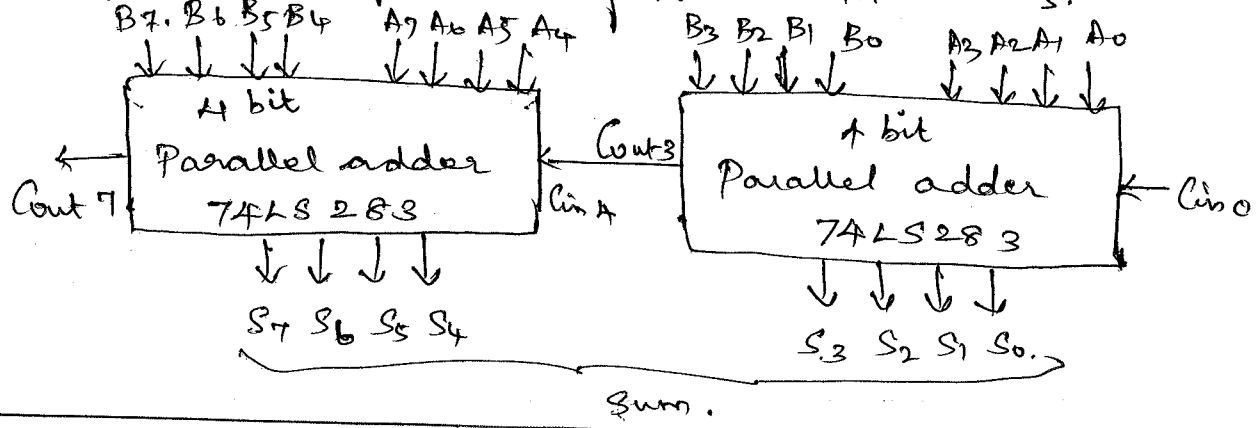
* each full adder gives the one sum bit.

Design a 4 bit parallel adder using full-adder.

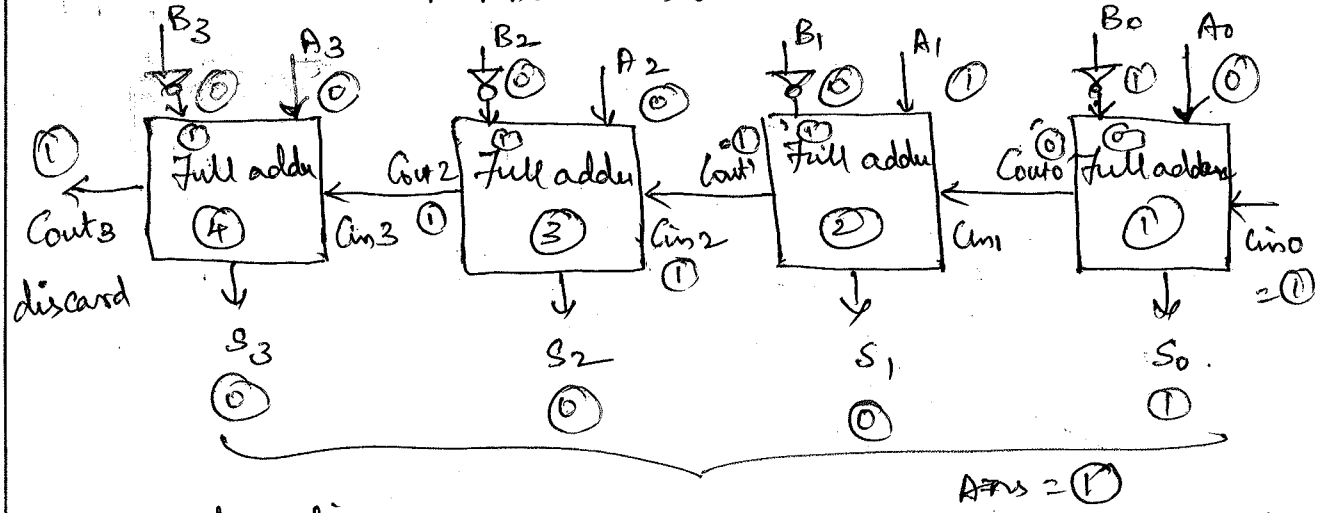
IC 74LS83 / 74LS283.



Design an 8 bit adder using two IC 74LS283s.



PARALLEL SUBTRACTOR.



* Subtraction can be done by means of complements.

* We know A−B means take 2's complement for B and add A.
* 2's complement means take 1's complement & 1 to LSB.
* for 1's complement use NOT gate and to add 1 to LSB is added to sum through input carry.

A B
2 − 1 = 1 → 0001
↓ ↓
0010 0001

take 1's complement for B = 1110
+1 = +1
1111

Add B to A

① $A_3\ A_2\ A_1\ A_0 = 2 = 0010$
$B_2\ B_2\ B_1\ B_0 = 1 = 0001$

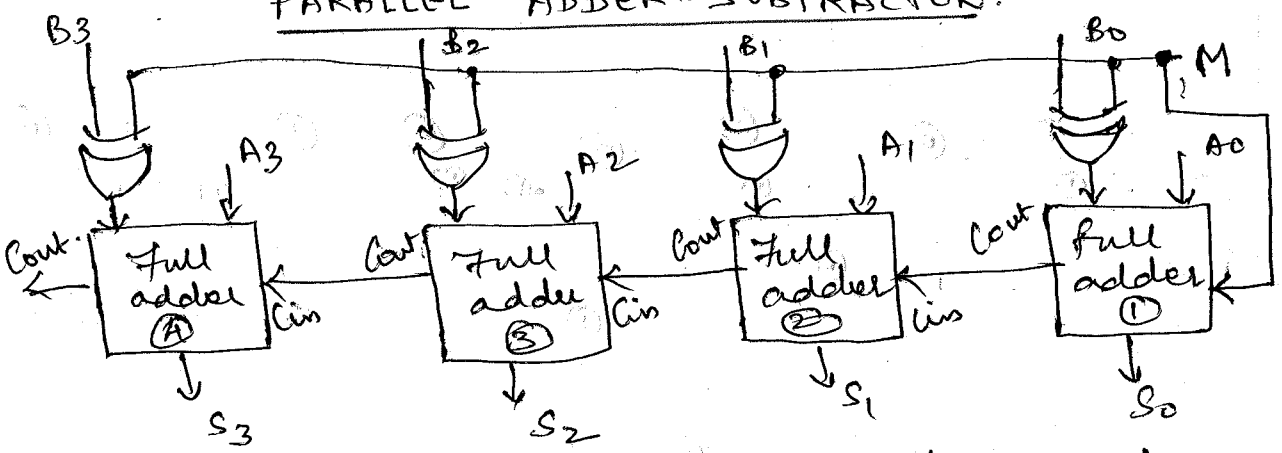
② Complement B (−1) = 1110

③ Add A and B and carry = 1

0010
1110
1 ← Cin = 1

discard carry ← 1] 0001
①

PARALLEL ADDER - SUBTRACTOR.

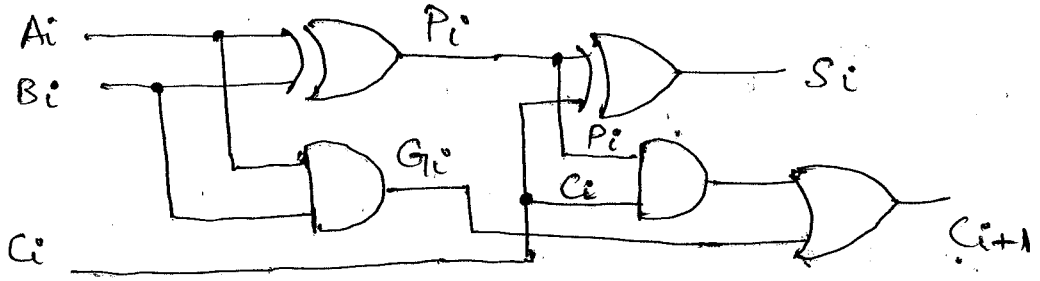


* When M=0, the circuit is an adder, we have B ⊕ 0 = B. The full adder receive the value of B, the input carry is '0' and the circuit performs A+B.

* When M=1, the circuit becomes a subtractor, we have $B \oplus 1 = \bar{B}$ and $C_{in0} = 1$. The input to B are all complemented and 1 is added through the input carry. The circuit performs the operation A + 2's complement of B (ie) A − B.

LOOK-AHEAD CARRY ADDER.

In the parallel adder the carry output of each full adder stage is connected to carry input of the next higher-order stage. The sum and carry outputs of any stage cannot be produced until the input carry occurs, this leads to time delay in addition process. This delay is known as carry propagation delay. If each full-adder is considered to have a propagation delay of 30ns, then S_3 will not reach its correct value until 90ns after LSB carry is generated. Total time required to perform addition is 90 + 30 = 120ns.



$$P_i = A_i \oplus B_i$$

$$G_i = A_i B_i$$

$$S_i = P_i \oplus C_i \quad \text{—— (1)}$$

$$C_{i+1} = G_i + P_i C_i \quad \text{—— (2)}$$

$i = 1$ in (2)

$$C_2 = G_1 + P_1 C_1 \quad \text{—— (3)}$$

$i = 2$ in (2)

$$C_3 = G_2 + P_2 C_2$$

Sub (3)

$$C_3 = G_2 + P_2 (G_1 + P_1 C_1)$$

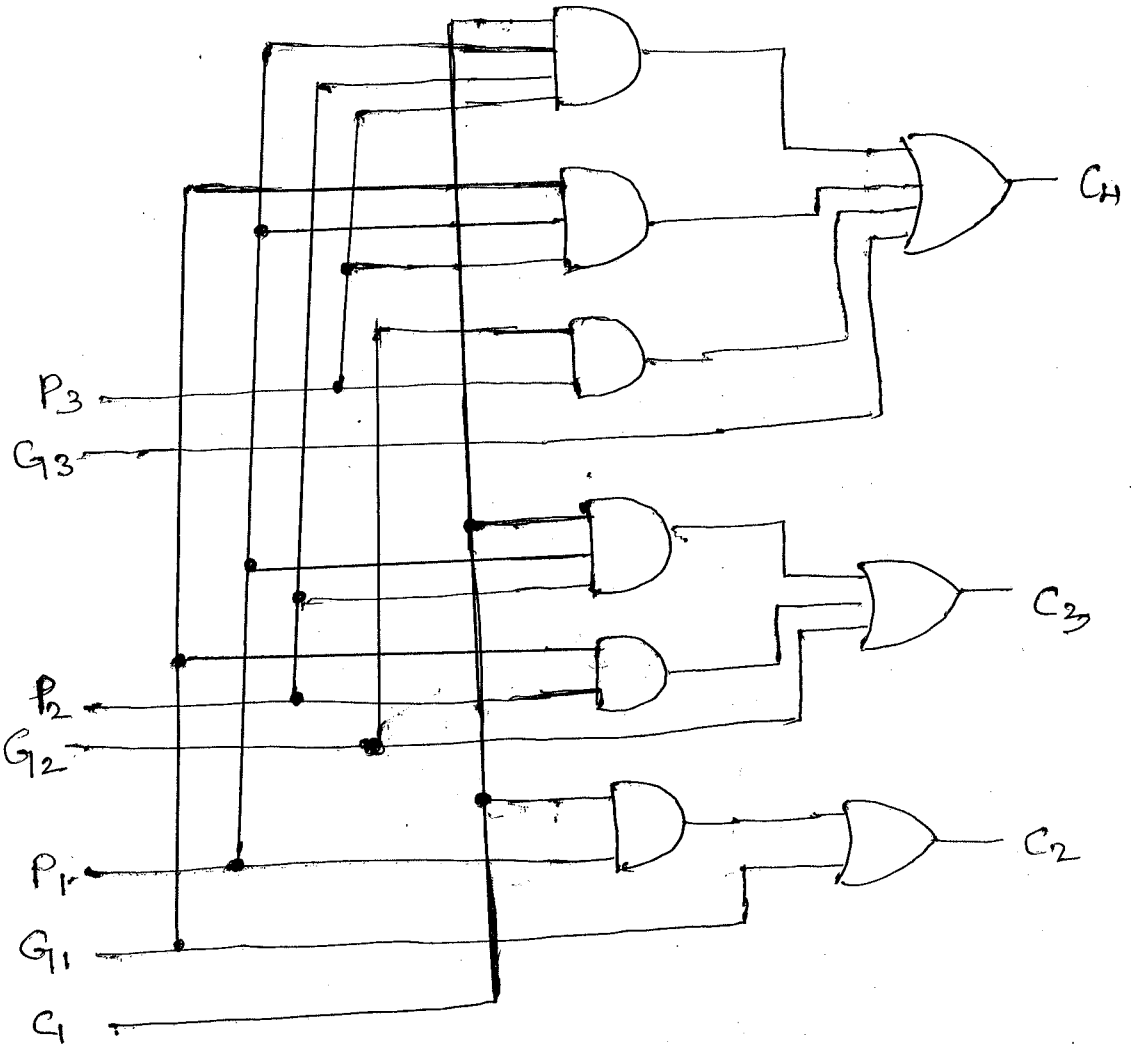
$$C_3 = G_2 + P_2 G_1 + P_2 P_1 C_1 \quad \text{—— (4)}$$

Sub $i=3$ in (2)

$$C_4 = G_3 + P_3 C_3$$

Sub (4) $$= G_3 + P_3 [G_2 + P_2 G_1 + P_2 P_1 C_1)$$

$$C_4 = G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_1 \quad \text{—— (5)}$$



4 bit parallel adder with look ahead carry generator.

	C_4	C_3	C_2	C_1
	1	0	1	0

(5) A → A_4 A_3 A_2 A_1

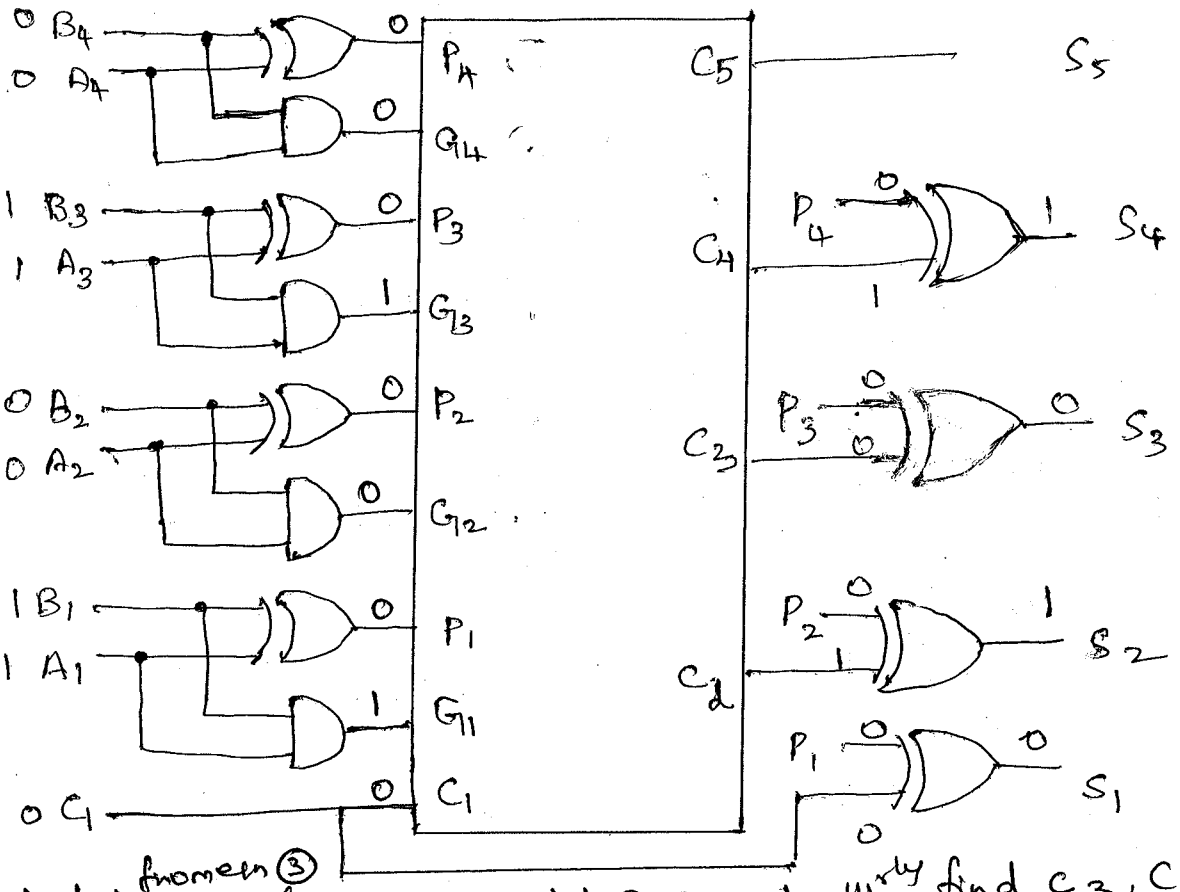
0	1	0	1

(5) B → B_4 B_3 B_2 B_1

0	1	0	1

Sum =

S_4	S_3	S_2	S_1
1	0	1	0



Note: from eqn (3) $C_2 = G_1 + P_1 C_1 = 1 + 0 \cdot 0 = 1$ Similarly find C_3, C_4.

* EX-OR gate generates P_i and AND gate generates G_i.
* Carries are generated using look-ahead carry generator and applied as input to the second EX-OR gate.
* The other input to EX-OR is P_i.
* The second EX-OR gate generates sum output
* Each output generated after a delay of two levels of gate. S_2 through S_4 has equal delay.

IC 74182 is a lookahead carry generator

The 74182 carry look ahead generator accepts upto four pairs of active low carry propagate $(\overline{P_0}, \overline{P_1}, \overline{P_2}, \overline{P_3})$ and carry generate high carries $(C_{n+x}, C_{n+y}, C_{n+z})$ across four groups of binary adders. The 74182 also has active low carry Propagate $(\overline{P})$ and carry generate $(\overline{G})$ outputs which may be used for further levels of look ahead.

The logic equations provided at the outputs of 74182

$$C_{n+x} = G_0 + P_0 C_n$$

$$C_{n+y} = G_1 + P_1 G_0 + P_1 P_0 C_n$$

$$C_{n+z} = G_2 + P_2 G_1 + P_2 P_1 G_0$$

$$\overline{G} = \overline{G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0}$$

$$\overline{P} = \overline{P_3 P_2 P_1 P_0}$$

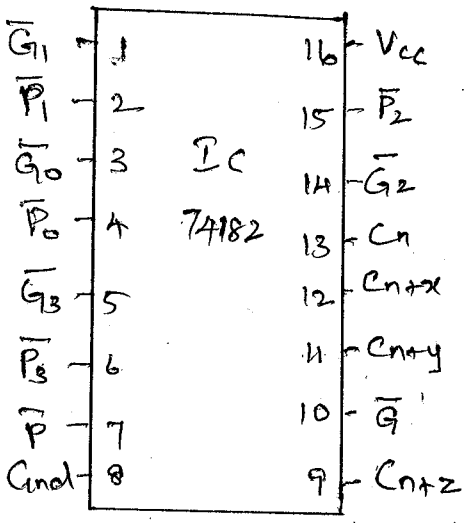
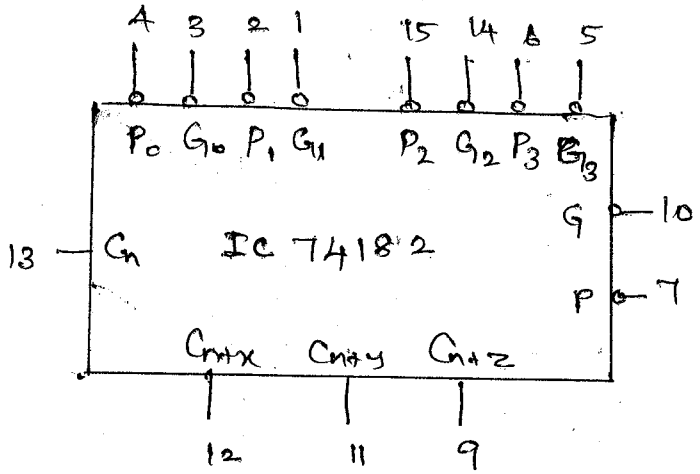


Fig 35



1. Show the construction of 4 bit adder using IC 74182

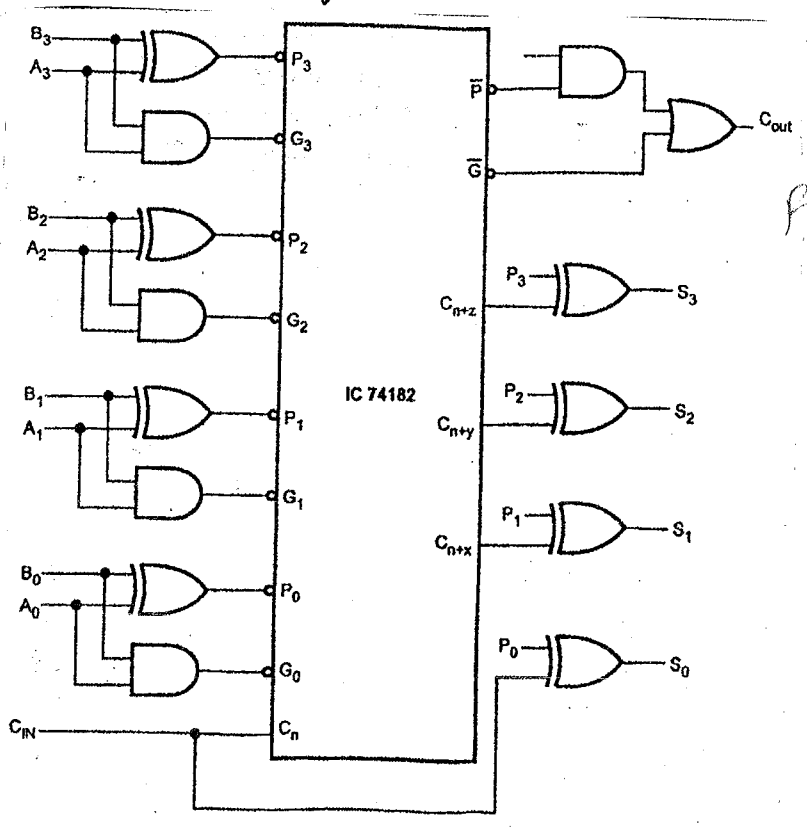
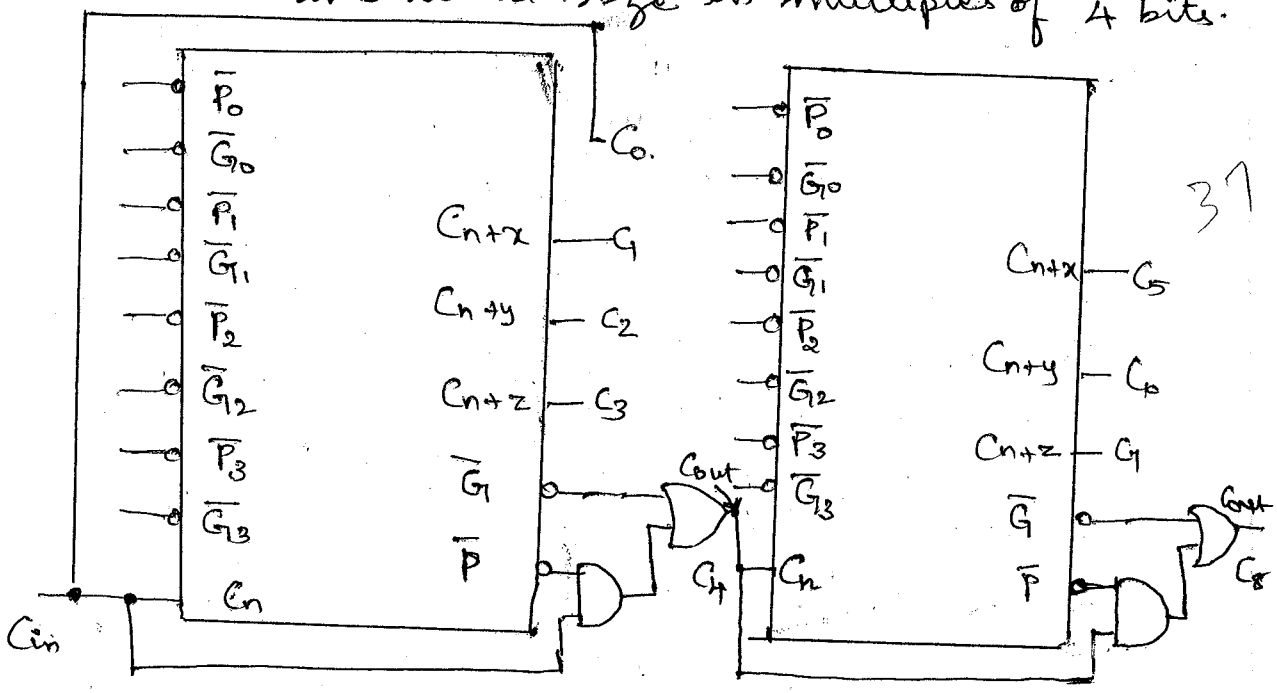


Fig 36

2. Construct the look ahead carry generator to accomodate higher word size.

Look ahead carry generators can be cascaded to increase the word size in multiples of 4 bits.



CIRCUITS FOR ARITHMETIC OPERATION.

BCD ADDER.

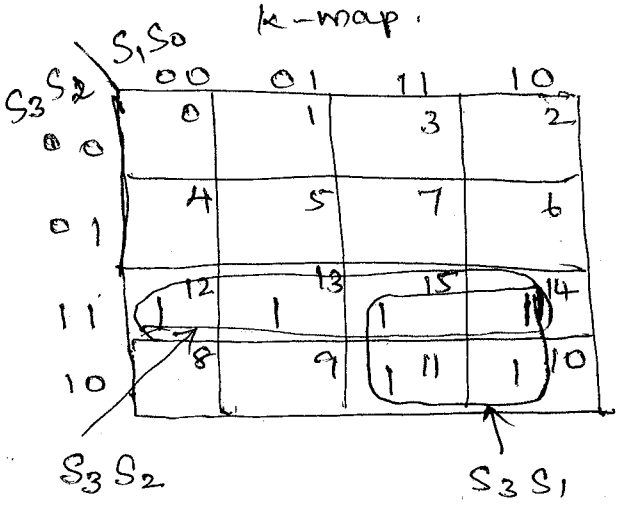
* 4 bit binary adder for initial addition
* Logic circuit to detect sum > 9.
* One more 4 bit adder to add 0110_2 in the sum if sum is >9 or carry = 1.

Input				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

> 9

Input				output	
S_3	S_2	S_1	S_0	Y	
1	0	0	0	0	valid. No change add 0000
1	0	0	1	0	
1	0	1	0	1	invalid add 0110
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:



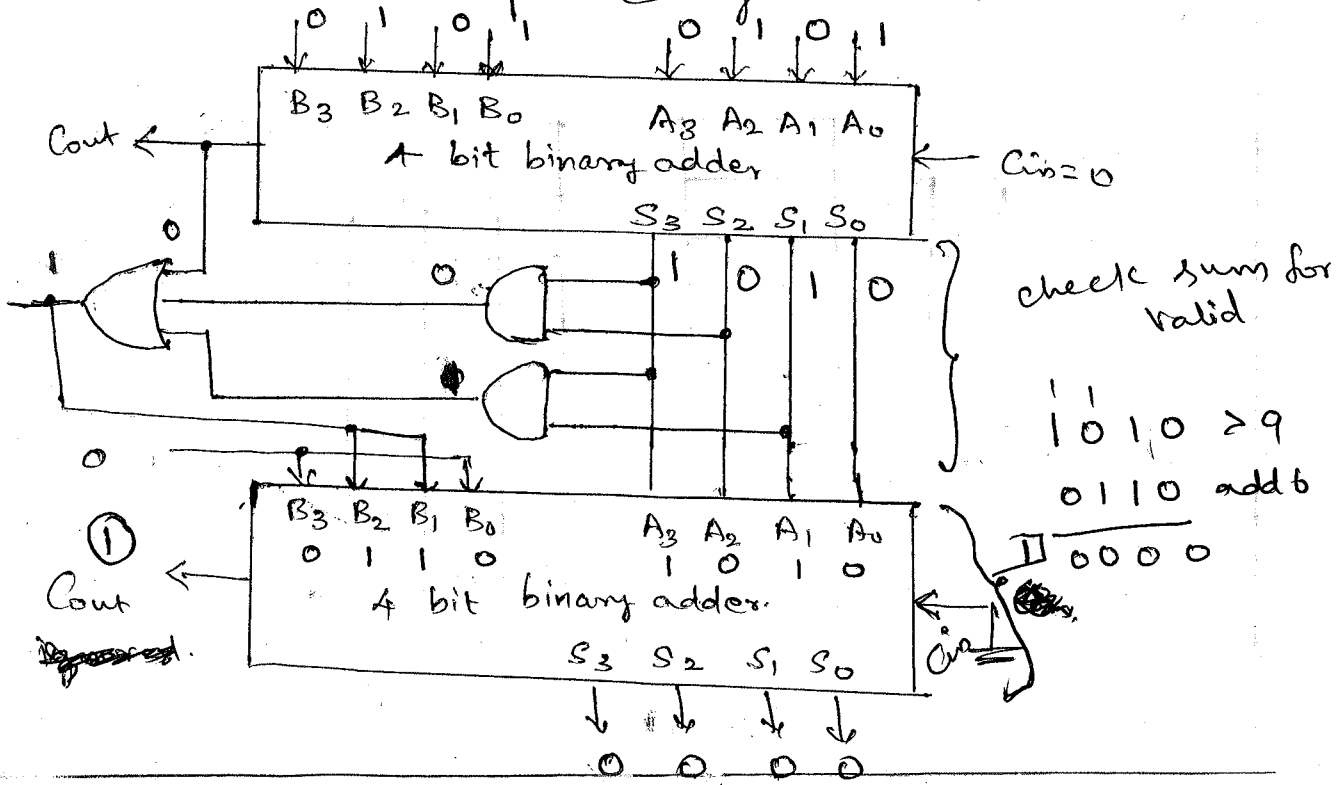
$$Y = S_3S_2 + S_3S_1$$

If Sum > 9 then Y = 1 and carry Cout = 1

→ for both the condition add $(0110)_2$

→ Eg: 5 = 0101
5 = 0101
1010 > 9 (two bcd numbers with)

→ The two BCD numbers together with input carry are first added in the top 4 bit binary adder to produce a binary sum. When the output carry is equal to zero (ie when sum ≤ 9 and Cout = 0) nothing (zero) is added to binary sum. When it is equal to one (ie when sum > 9 or Cout = 1) 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.



BINARY MULTIPLIER

2 x 2 (Half adder)

$0 \times 0 = 0$
$0 \times 1 = 0$
$1 \times 0 = 0$
$1 \times 1 = 1$

$A_1\ A_0$
$\times\ B_1\ B_0$

B_0A_1 B_0A_0

Carry of P_1 ↙ add B_1A_1 B_1A_0 ↘ add

P_3 P_2 P_1 P_0

$P_0 = B_0A_0$

$P_1 = B_0A_1 + B_1A_0$

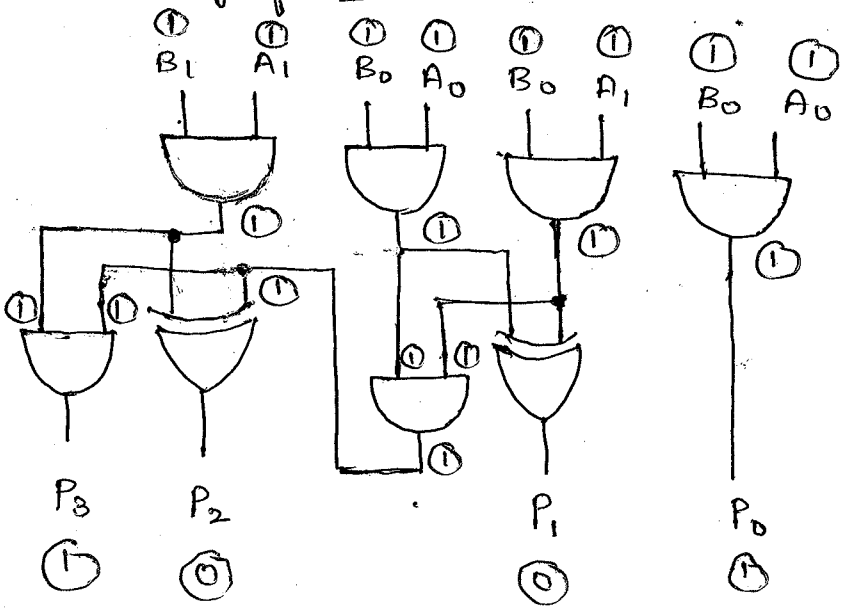
$P_2 = B_1A_1 + \text{Carry out of } P_1$

$P_3 = \text{Carry out of } P_2$.

$A_1\ A_0$
$1\ \ 1$
$B_1\ B_0$
$1\ \ 1$

① 1 1 Carry
1 1 X
1 0 0 1
$P_3\ P_2\ P_1\ P_0$

P_3 → Carry of P_2.



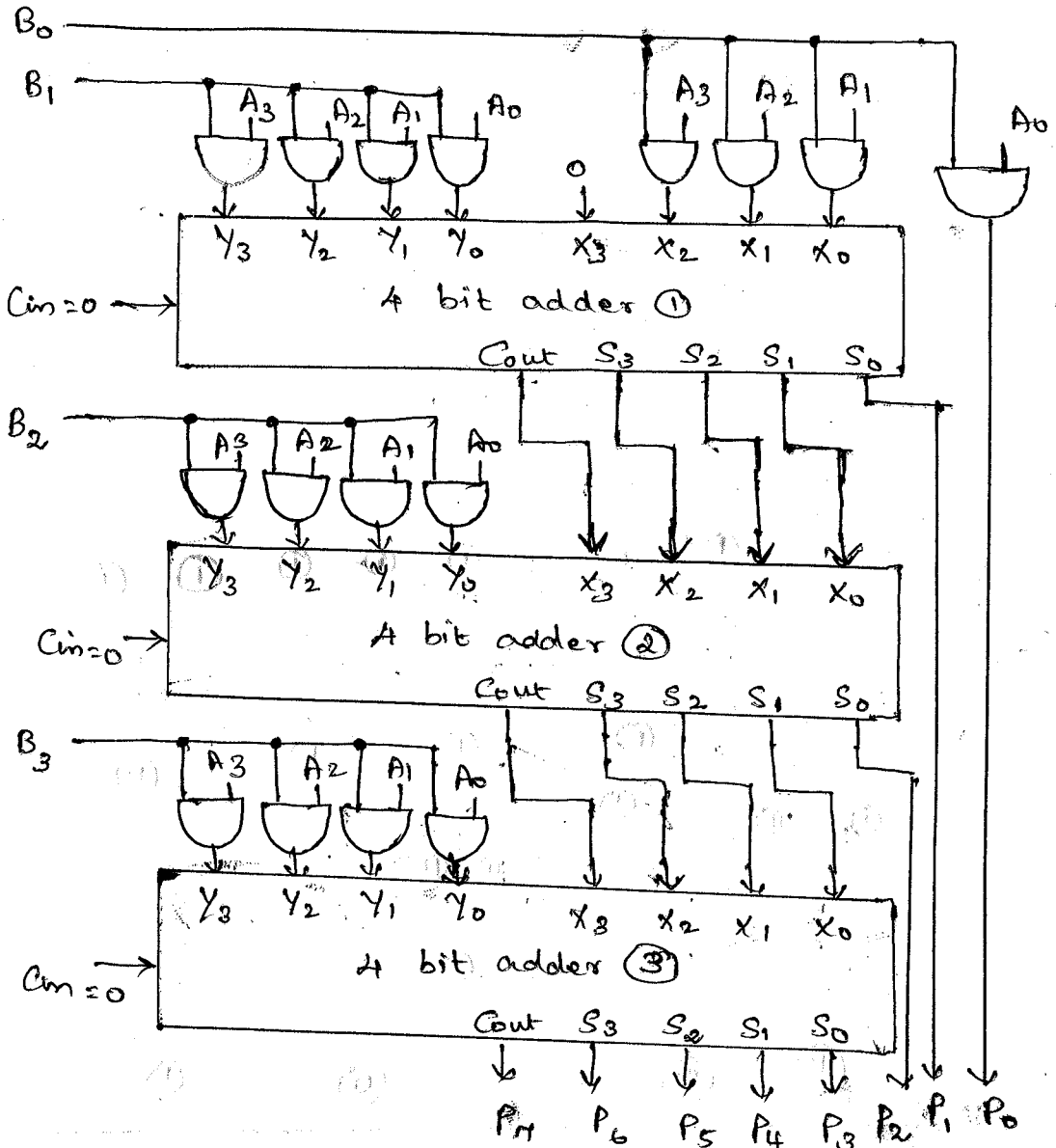
4 bit by 4 bit

$A_3\ A_2\ A_1\ A_0$
$B_3\ B_2\ B_1\ B_0$

B_0A_3 B_0A_2 B_0A_1 B_0A_0 ↙ 4 bit adder.
B_1A_3 B_1A_2 B_1A_1 B_1A_0 $\times$ ↙ 4 bit adder
B_2A_3 B_2A_2 B_2A_1 B_2A_0 $\times$ ↙ 4 bit adder
B_3A_3 B_3A_2 B_3A_1 B_3A_0 $\times$

P_7 P_6 P_5 P_4 P_3 P_2 P_1 P_0

P_7 → Carry of P_6.



$P_0 = A_0 B_0$

$P_1 = B_0 A_1 + B_1 A_0$

$P_2 = B_0 A_2 + B_1 A_1 + B_2 A_0 + P_1$ Carry.

$P_3 = B_0 A_3 + B_1 A_2 + B_2 A_1 + B_3 A_0 + P_2$ Carry.

$P_4 = B_1 A_3 + B_2 A_2 + B_3 A_1 + P_3$ Carry.

$P_5 = B_2 A_3 + B_3 A_2 + P_4$ carry.

$P_6 = B_3 A_3 + P_5$ Carry

$P_7 = P_6$ carry.

CODE CONVERSION

There is a wide variety of binary codes used in digital systems. Some of these codes are binary-coded-decimal (BCD), Excess-3, Gray code and so on. Each code has different advantages and applications. Hence many times it is required to convert one code to another.

1. BINARY to BCD CONVERTER

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

K-map

For B_0

DC \ BA	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	0	1	1	0
10	0	1	1	0

$B_0 = A$

For B_1

DC \ BA	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	1	1	0	0
10	0	0	0	0

$B_1 = DC\overline{B} + \overline{D}B$

For B_2

DC \ BA	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	1	1
10	0	0	0	0

$B_2 = \overline{D}C + CB$

For B_3

DC \ BA	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	0	0
10	1	1	0	0

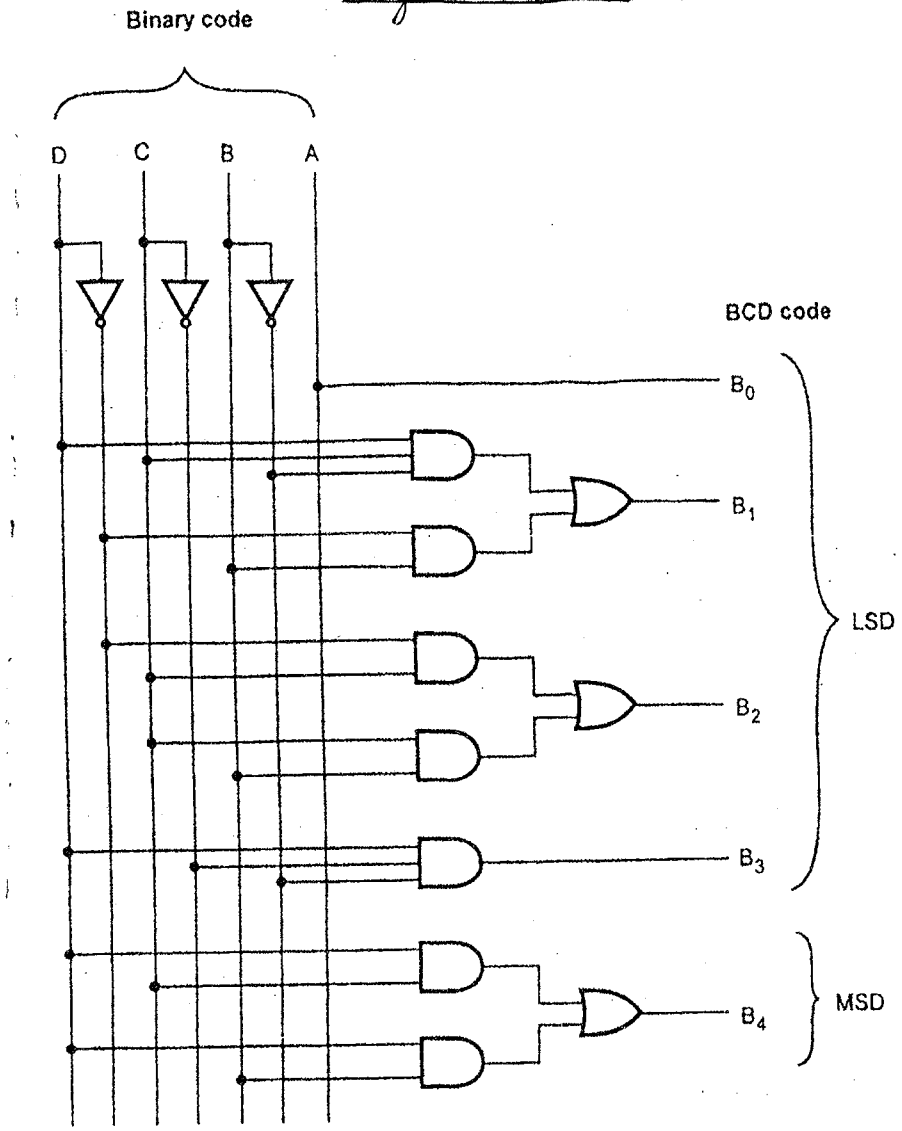
$B_3 = D\overline{C}\overline{B}$

For B_4

DC \ BA	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	1	1

$B_4 = DC + DB$

Logic diagram



Binary to BCD converter

BCD to BINARY CONVERTER

Truth table:

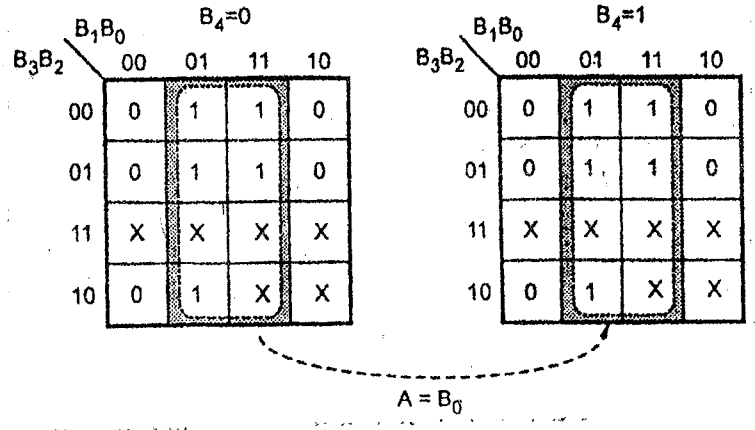
B_4	B_3	B_2	B_1	B_0	E	D	C	B	A
0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	1
0	0	0	1	0	0	0	0	1	0
0	0	0	1	1	0	0	0	1	1
0	0	1	0	0	0	0	1	0	0
0	0	1	0	1	0	0	1	0	1
0	0	1	1	0	0	0	1	1	0
0	0	1	1	1	0	0	1	1	1
0	1	0	0	0	0	1	0	0	0
0	1	0	0	1	0	1	0	0	1
1	0	0	0	0	0	1	0	1	0
1	0	0	0	1	0	1	0	1	1

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

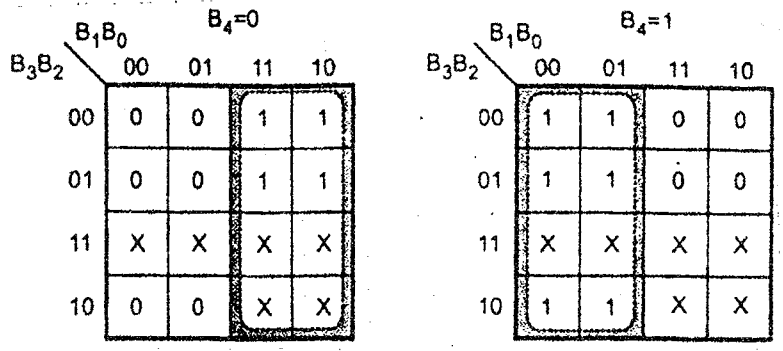
Truth table for BCD to binary converter

K-map simplification

For A



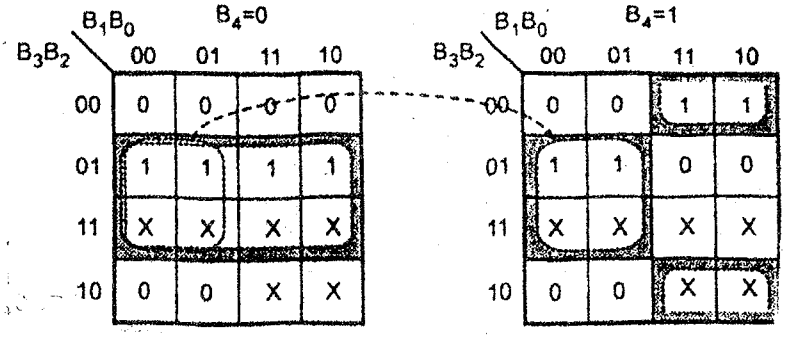
For B.



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

$= B_1 \oplus B_4$

For C



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

For D

$B_4=0$

B_3B_2 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	X	X	X	X
10	1	1	X	X

$B_4=1$

B_3B_2 \ B_1B_0	00	01	11	10
00	1	1	1	1
01	1	1	0	0
11	X	X	X	X
10	0	0	X	X

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

For E

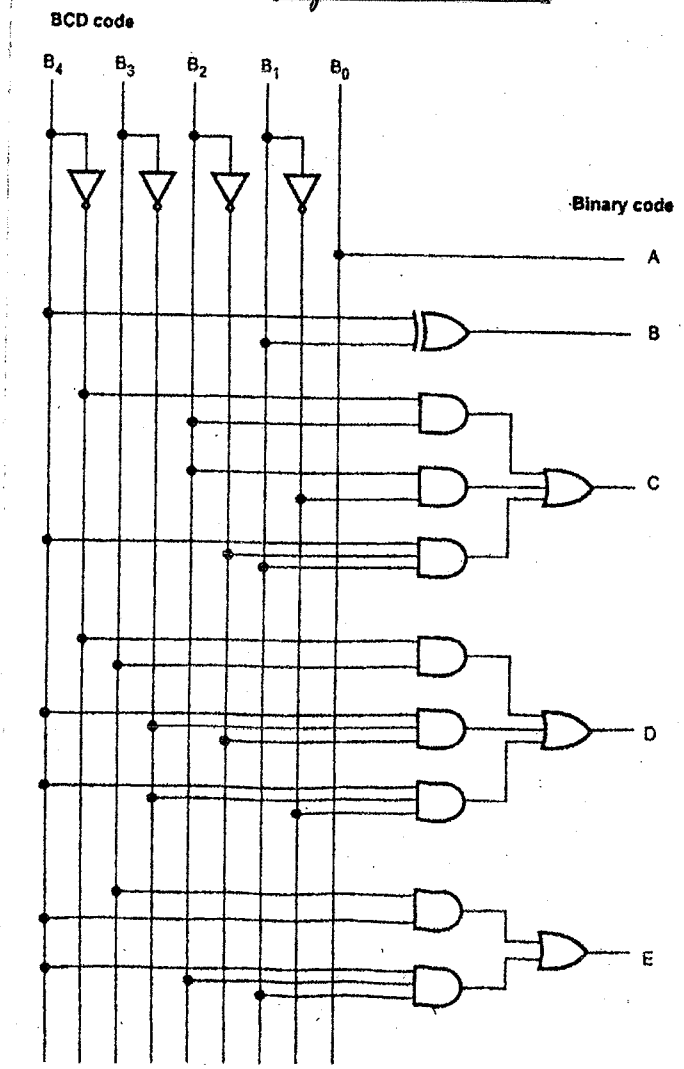
$B_4=0$

B_3B_2 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	X	X	X	X
10	0	0	X	X

$B_4=1$

B_3B_2 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	X	X	X	X
10	1	1	X	X

Logic diagram



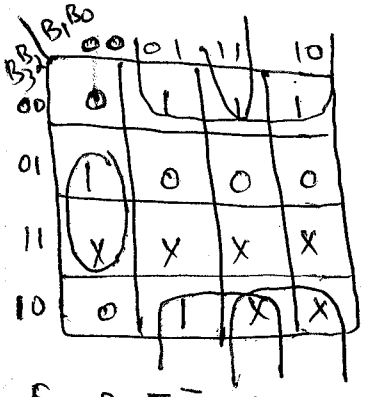
BCD to binary code converter

BCD to EXCESS-3.

* Excess-3 is modified form of BCD.
* Excess-3 code derived by adding 3 to BCD numbers.

Decimal	B_3	B_2	B_1	B_0	E_3	E_2	E_1	E_0
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

for E_2



$E_2 = B_2\bar{B}_1\bar{B}_0 + \bar{B}_2(B_0+B_1)$

K-map Simplification

For E_0

B_3B_2 \ B_1B_0	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	X	X	X	X
10	1	0	X	X

$E_0 = \bar{B}_0$

For E_1

B_3B_2 \ B_1B_0	00	01	11	10
00	1	0	1	0
01	1	0	1	0
11	X	X	X	X
10	1	0	X	X

$E_1 = \bar{B}_1\bar{B}_0 + B_1B_0$
$= B_1 \odot B_0$

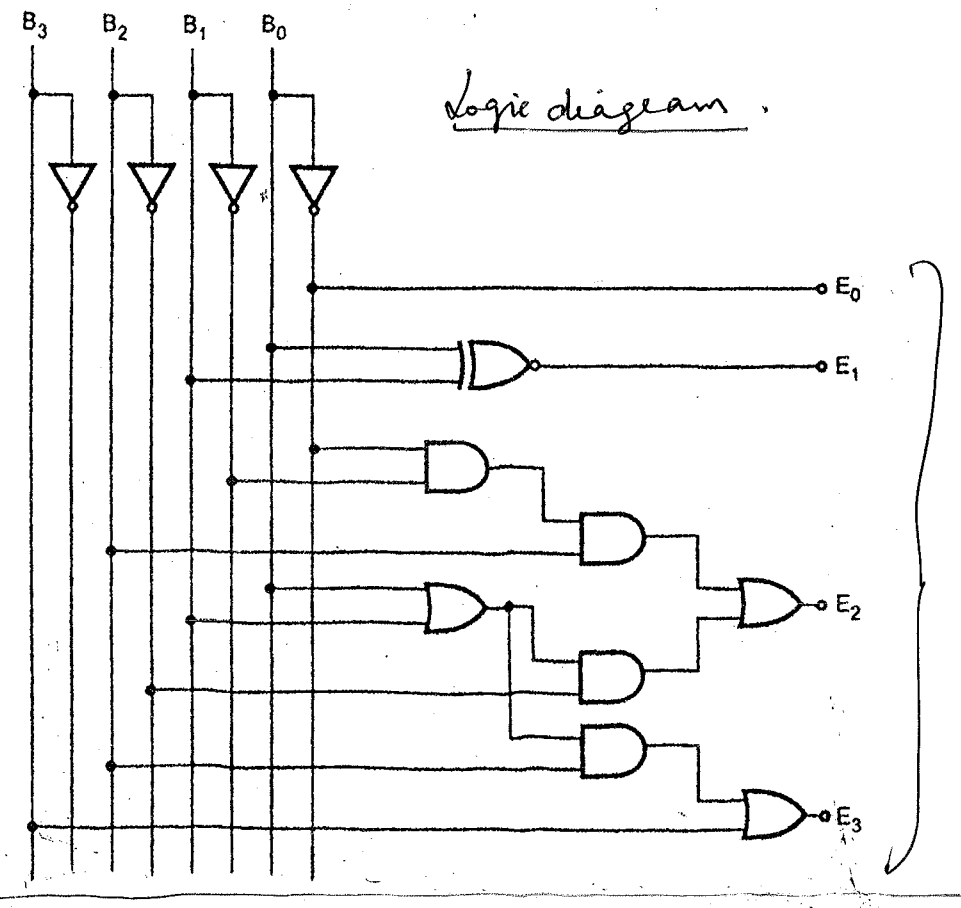
For E_3

B_3B_2 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

$\therefore E_3 = B_3 + B_2(B_0 + B_1)$

BCD code

Logic diagram.



EXCESS-3 to CD Code Converter

Truth table

E_3	E_2	E_1	E_0	B_3	B_2	B_1	B_0
0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	1	0
0	1	1	0	0	0	1	1
0	1	1	1	0	1	0	0
1	0	0	0	0	1	0	1
1	0	0	1	0	1	1	0
1	0	1	0	0	1	1	1
1	0	1	1	1	0	0	0
1	1	0	0	1	0	0	1

K-map simplification

For B_0

E_3E_2 \ E_1E_0	00	01	11	10
00	X	X	0	X
01	1	0	0	1
11	1	X	X	X
10	1	0	0	1

$B_0 = \bar{E}_0$

For B_1

E_3E_2 \ E_1E_0	00	01	11	10
00	X	X	0	X
01	0	1	0	1
11	0	X	X	X
10	0	1	0	1

$B_1 = \bar{E}_1E_0 + E_1\bar{E}_0$
$= E_1 \oplus E_0$

For B_2

E_3E_2 \ E_1E_0	00	01	11	10
00	X	X	0	X
01	0	0	1	0
11	0	X	X	X
10	1	1	0	1

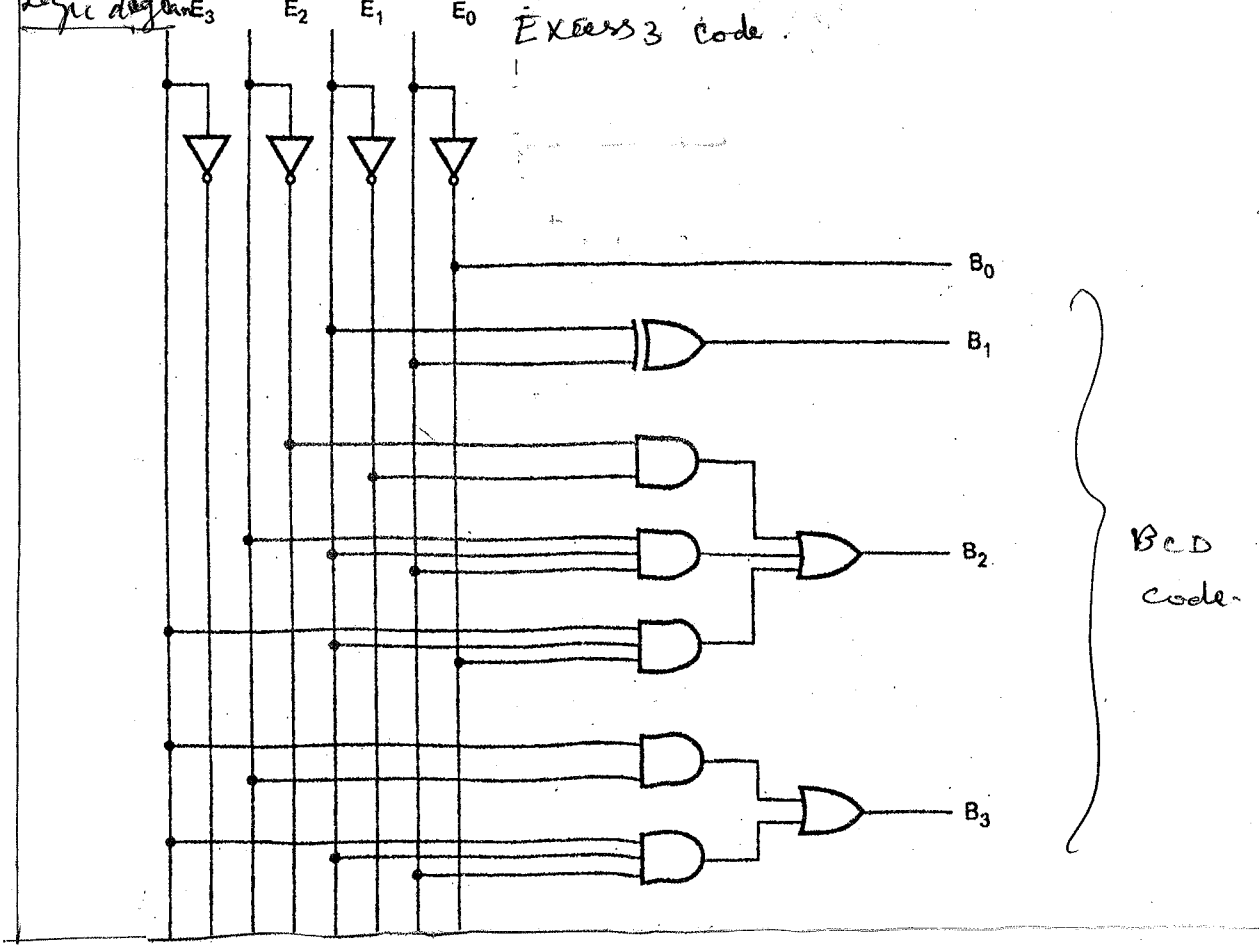
$B_2 = \bar{E}_2\bar{E}_1 + E_2E_1E_0 + E_3E_1\bar{E}_0$

For B_3

E_3E_2 \ E_1E_0	00	01	11	10
00	X	X	0	X
01	0	0	0	0
11	1	X	X	X
10	0	0	1	0

$B_3 = E_3E_2 + E_3E_1E_0$

Logic diagram



BINARY to GRAY CODE CONVERTER

× Gray code is often used in digital system has only one bit in numerical representation changes between successive numbers.

Truth table

Decimal	Binary code				Gray code			
	D	C	B	A	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

K-map Simplification

For G_0

DC \ BA	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

$G_0 = \bar{B}A + B\bar{A}$

$= B \oplus A$

For G_1

DC \ BA	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

$G_1 = C\bar{B} + \bar{C}B$

$= C \oplus B$

For G_2

DC \ BA	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

$G_2 = \bar{D}C + D\bar{C}$

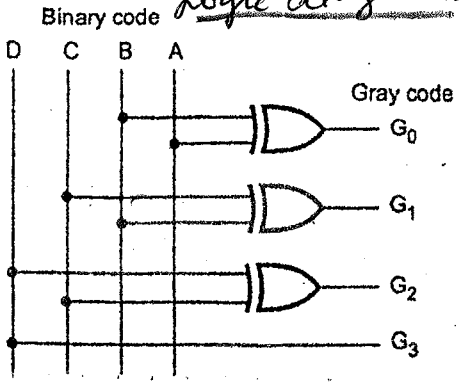
$= D \oplus C$

For G_3

DC \ BA	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$G_3 = D$

Logic diagram



GRAY CODE to BINARY CODE CONVERTER

Truth table

Gray code				Binary code			
G_3	G_2	G_1	G_0	D	C	B	A
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1

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

K-map Simplification

① For A

G_3G_2 \ G_1G_0	00	01	11	10
00	0	(1)	0	(1)
01	(1)	0	(1)	0
11	0	(1)	0	(1)
10	(1)	0	(1)	0

$A = \bar{G}_3\bar{G}_2\bar{G}_1G_0 + \bar{G}_3\bar{G}_2G_1\bar{G}_0 + \bar{G}_3G_2\bar{G}_1\bar{G}_0 + \bar{G}_3G_2G_1G_0 + G_3G_2\bar{G}_1G_0 + G_3G_2G_1\bar{G}_0 + G_3\bar{G}_2\bar{G}_1\bar{G}_0 + G_3\bar{G}_2G_1G_0$

$= \bar{G}_3\bar{G}_2(\bar{G}_1G_0 + G_1\bar{G}_0) + \bar{G}_3G_2(\bar{G}_1\bar{G}_0 + G_1G_0) + G_3G_2(\bar{G}_1G_0 + G_1\bar{G}_0) + G_3\bar{G}_2(\bar{G}_1\bar{G}_0 + G_1G_0)$

$= \bar{G}_3\bar{G}_2(G_1 \oplus G_0) + \bar{G}_3G_2(\overline{G_1 \oplus G_0}) + G_3G_2(G_1 \oplus G_0) + G_3\bar{G}_2(\overline{G_1 \oplus G_0})$.

$= \{(G_1 \oplus G_0)[\bar{G}_3\bar{G}_2 + G_3G_2)\} + \{(\overline{G_1 \oplus G_0})(\bar{G}_3G_2 + G_3\bar{G}_2)\}$

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

$A = (G_1 \oplus G_0) \oplus (G_3 \oplus G_2)$

② For B

G_3G_2 \ G_1G_0	00	01	11	10
00	0	0	(1)	(1)
01	(1)	(1)	0	0
11	0	0	(1)	(1)
10	(1)	(1)	0	0

$B = \bar{G}_3\bar{G}_2G_1 + \bar{G}_3G_2\bar{G}_1 + G_3G_2G_1 + G_3\bar{G}_2\bar{G}_1$.

$= G_1(\bar{G}_3\bar{G}_2 + G_3G_2) + \bar{G}_1(\bar{G}_3G_2 + G_3\bar{G}_2)$

$= G_1(\overline{G_3 \oplus G_2}) + \bar{G}_1(G_3 \oplus G_2)$

$B = G_1 \oplus (G_3 \oplus G_2)$

For C

G_3G_2 \ G_1G_0	00	01	11	10
00	0	0	0	0
01	(1)	(1)	(1)	(1)
11	0	0	0	0
10	(1)	(1)	(1)	(1)

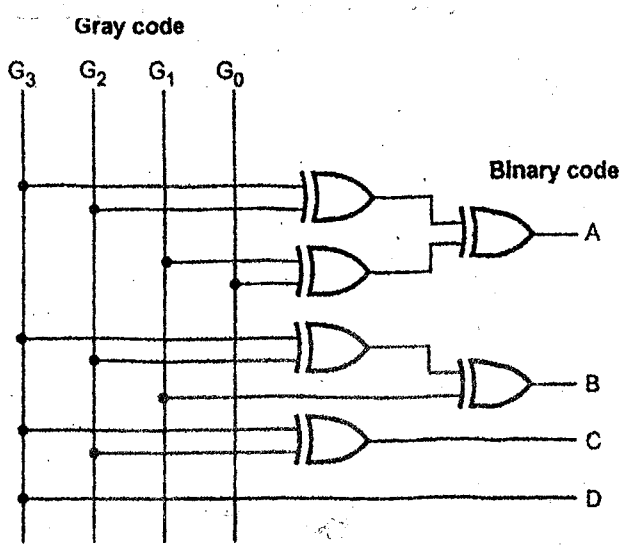
$C = \bar{G}_3G_2 + G_3\bar{G}_2$

$C = G_3 \oplus G_2$.

For D

G_3G_2 \ G_1G_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	(1)	(1)	(1)	(1)
10	(1)	(1)	(1)	(1)

$D = G_3$



BCD to GRAY CODE CONVERTER.

BCD code				Gray code			
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

For G_0

B_3B_2 \ B_1B_0	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	X	X	X	X
10	0	1	X	X

$G_0 = \bar{B}_1B_0 + B_1\bar{B}_0$
$= B_1 \oplus B_0$

For G_1

B_3B_2 \ B_1B_0	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	x	x	x	x
10	0	0	x	x

$G_1 = B_2\bar{B}_1 + \bar{B}_2B_1$
$= B_2 \oplus B_1$

For G_2

B_3B_2 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	X	X	X	X
10	1	1	X	X

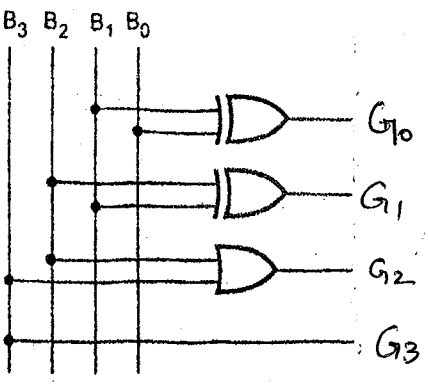
$G_2 = B_2 + B_3$

For G_3

B_3B_2 \ B_1B_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	X	X	X	X
10	1	1	X	X

$G_3 = B_3$

BCD code



GRAY TO BCD CODE CONVERTER.

Gray code				BCD code			
G_3	G_2	G_1	G_0	D	C	B	A
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	1	0	0	1	0
0	0	1	0	0	0	1	1
0	1	1	0	0	1	0	0
0	1	1	1	0	1	0	1
0	1	0	1	0	1	1	0
0	1	0	0	0	1	1	1
1	1	0	0	1	0	0	0
1	1	0	1	1	0	0	1
1	1	1	1	x	x	x	x
1	1	1	0	x	x	x	x
1	0	1	0	x	x	x	x
1	0	1	1	x	x	x	x
1	0	0	1	x	x	x	x
1	0	0	0	x	x	x	x

For A

G_3G_2 \ G_1G_0	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	X	X	X	X
10	1	0	X	X

$$A = \bar{G}_3\bar{G}_2\bar{G}_1G_0 + \bar{G}_2G_1\bar{G}_0 + G_2\bar{G}_1\bar{G}_0 + G_2G_1G_0 + G_3\bar{G}_0.$$

$$= G_2(\bar{G}_1\bar{G}_0 + G_1G_0) + \overline{G_0}(\bar{G}_2G_1 + G_3) + G_3\bar{G}_2\bar{G}_1G_0.$$

$$= G_2(\overline{G_1 \oplus G_0}) + \bar{G}_0(\bar{G}_2G_1 + G_3) + G_3\bar{G}_2\bar{G}_1G_0.$$

For B

G_3G_2 \ G_1G_0	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	X	X	X	X
10	1	1	X	X

$$B = G_2\bar{G}_1 + G_3 + \bar{G}_2G_1$$

For C

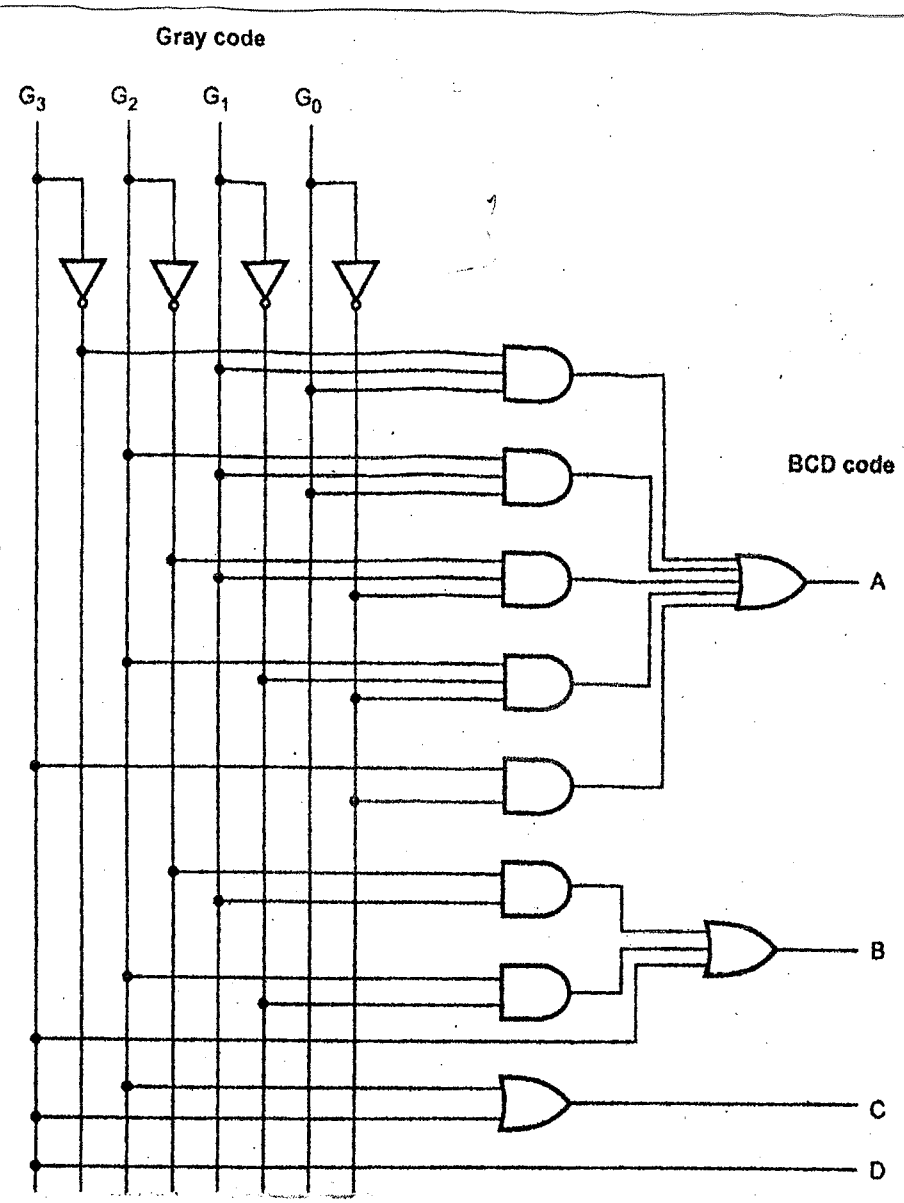
G_3G_2 \ G_1G_0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	X	X	X	X
10	1	1	X	X

$C = G_2 + G_3$

For D

G_3G_2 \ G_1G_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	X	X	X	X
10	1	1	X	X

$D = G_3$



4 bit Gray code to 2 digit BCD.

Gray code				BCD code				
G_3	G_2	G_1	G_0	E	D	C	B	A
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1
0	0	1	1	0	0	0	1	0
0	0	1	0	0	0	0	1	1
0	1	1	0	0	0	1	0	0
0	1	1	1	0	0	1	0	1
0	1	0	1	0	0	1	1	0
0	1	0	0	0	0	1	1	1
1	1	0	0	0	1	0	0	0
1	1	0	1	0	1	0	0	1
1	1	1	1	1	0	0	0	0
1	1	1	0	1	0	0	0	1
1	0	1	0	1	0	0	1	0
1	0	1	1	1	0	0	1	1
1	0	0	1	1	0	1	0	0
1	0	0	0	1	0	1	0	1

For A

G_3G_2 \ G_1G_0	00	01	11	10
00	0	(1)	0	(1)
01	(1)	0	(1)	0
11	0	(1)	0	(1)
10	(1)	0	(1)	0

$A = \bar{G}_3\bar{G}_2\bar{G}_1G_0 + \bar{G}_3\bar{G}_2G_1\bar{G}_0 + \bar{G}_3G_2\bar{G}_1\bar{G}_0$
$\bar{G}_3G_2G_1G_0 + G_3G_2\bar{G}_1G_0 + G_3G_2G_1\bar{G}_0$
$+ \cancel{G_3\bar{G}_2\bar{G}_1G_0} + \cancel{G_3G_2G_1\bar{G}_0} + G_3\bar{G}_2\bar{G}_1\bar{G}_0$
$+ G_3\bar{G}_2G_1G_0$.

$= \bar{G}_3\bar{G}_2(\bar{G}_1G_0 + G_1\bar{G}_0) + \bar{G}_3G_2(\bar{G}_1\bar{G}_0 + G_1G_0)$
$+ G_3G_2(\bar{G}_1G_0 + G_1\bar{G}_0) + G_3\bar{G}_2(\bar{G}_1\bar{G}_0 + G_1G_0)$

$A = \bar{G}_3\bar{G}_2(G_1 \oplus G_0) + \bar{G}_3G_2(\overline{G_1 \oplus G_0}) + G_3G_2(G_1 \oplus G_0) +$
$G_3\bar{G}_2(\overline{G_1 \oplus G_0})$

$= \{(G_1 \oplus G_0)(\bar{G}_3\bar{G}_2 + G_3G_2)\} + \{(\overline{G_1 \oplus G_0})(\bar{G}_3G_2 + G_3\bar{G}_2)$

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

$A = (G_1 \oplus G_0) \oplus (G_3 \oplus G_2)$.

For B

G_3G_2 \ G_1G_0	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	0	0	0	0
10	0	0	1	1

$B = \bar{G}_2G_1 + \bar{G}_3G_2\bar{G}_1$

For C

G_3G_2 \ G_1G_0	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	0	0

$C = \bar{G}_3G_2 + G_3\bar{G}_2\bar{G}_1$

For D

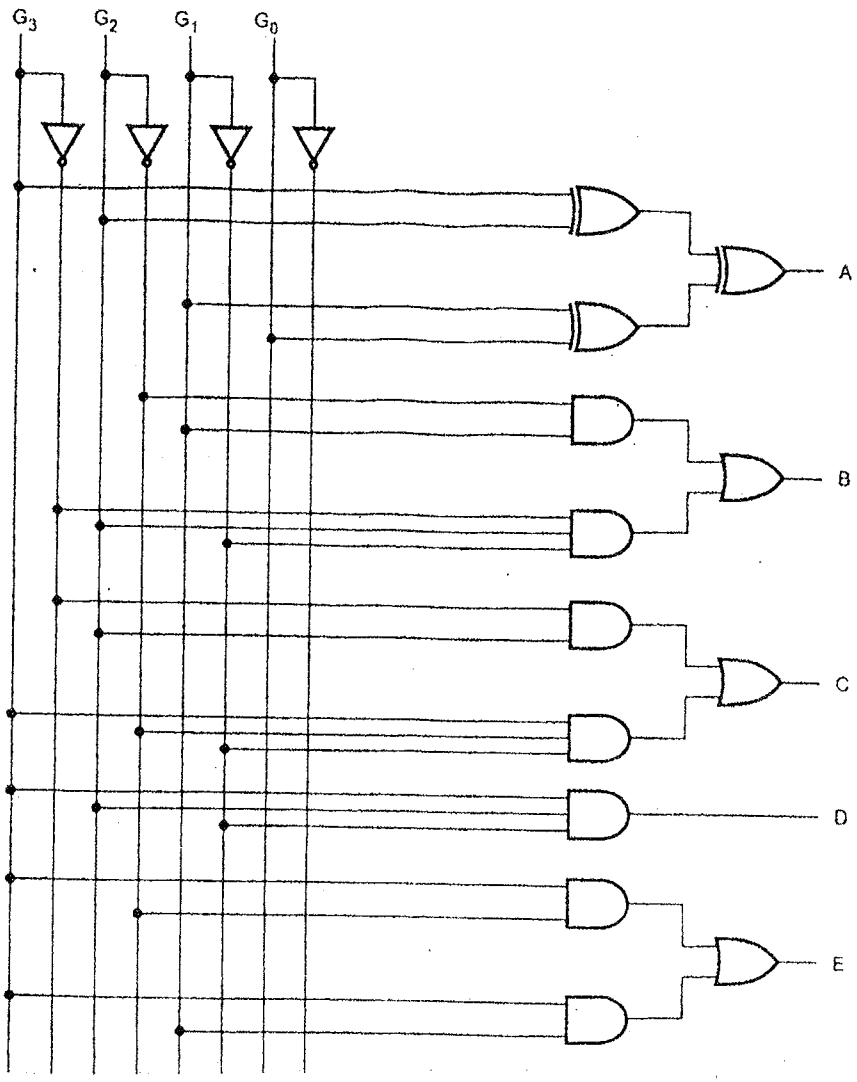
G_3G_2 \ G_1G_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	0	0
10	0	0	0	0

$D = G_3G_2\bar{G}_1$

For E

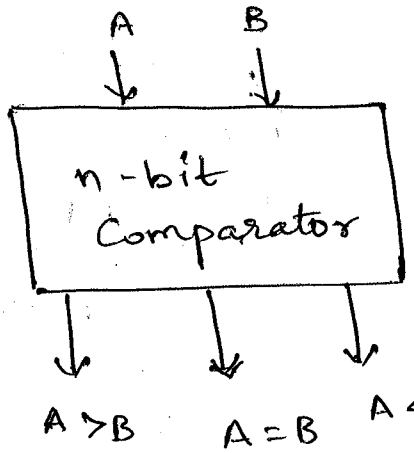
G_3G_2 \ G_1G_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	0	1	1
10	1	1	1	1

$E = G_3\bar{G}_2 + G_3G_1$


G_3
G_2
G_1
G_0
A
B
C
D
E

Binary Comparator / Magnitude Comparator

The circuit that compare relative magnitude of two binary numbers is called Magnitude comparator circuit.



Block diagram of Magnitude Comparator.

1 bit Magnitude comparator

It Compare one bit binary numbers.

inputs A & B.

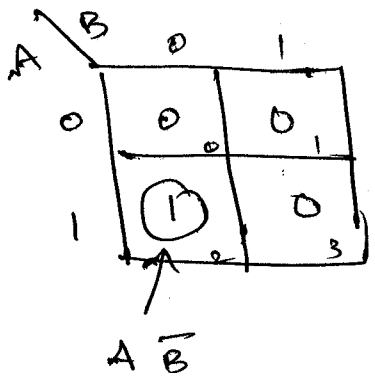
A = 0 , 1 B = 0 , 1

Truth table

input		output Y.		
A	B	$Y_{A>B}$	$Y_{A=B}$	$Y_{A<B}$
0	0	0	1	0
	1	0	0	1
1	0	1	0	0
	1	0	1	0

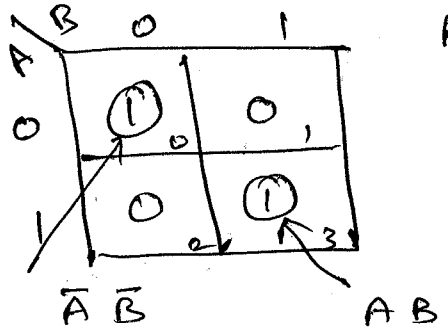
K-map:

$Y_{(A>B)}$



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

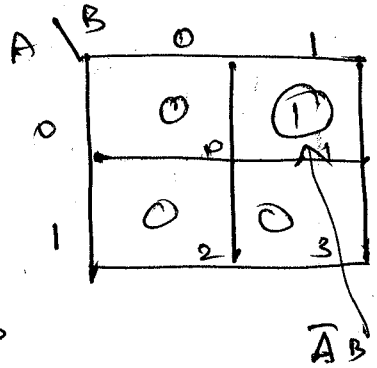
$Y_{(A=B)}$



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

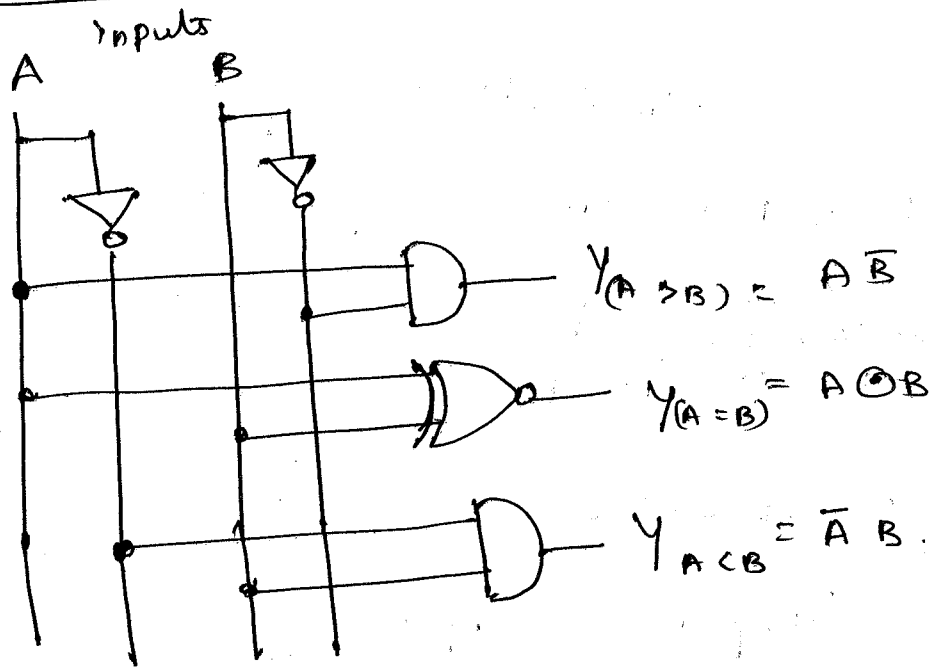
$= A \odot B.$

$Y_{(A<B)}$



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

Logic diagram:



2 bit Magnitude comparator:

It compares 2 bit numbers. Let the inputs be A and B. Where,

$A = A_1 A_0$

$B = B_1 B_0$

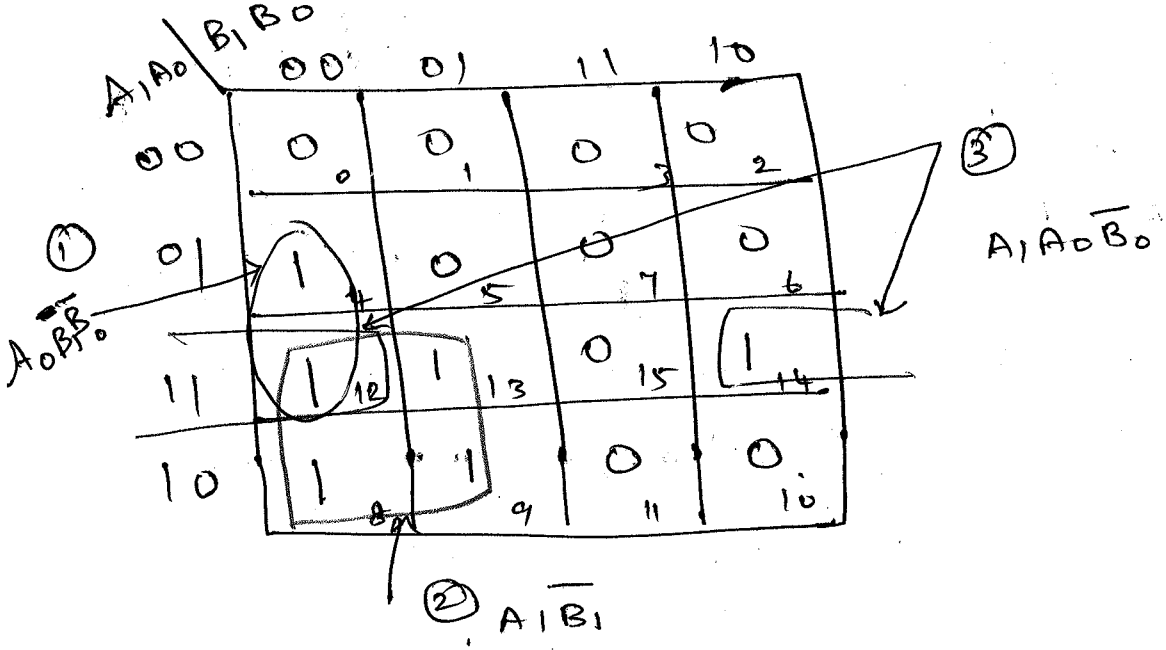
} each 2 bit binary number.

Truth table

inputs				output		
A		B		Y		
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	1	0	0	1
		1	0	0	0	1
		1	1	0	0	1
0	1	0	0	1	0	0
		0	1	0	1	0
		1	0	0	0	1
		1	1	0	0	1
1	0	0	0	1	0	0
		0	1	1	0	0
		1	0	0	1	0
		1	1	0	0	1
1	1	0	0	1	0	0
		0	1	1	0	0
		1	0	1	0	0
		1	1	0	1	0

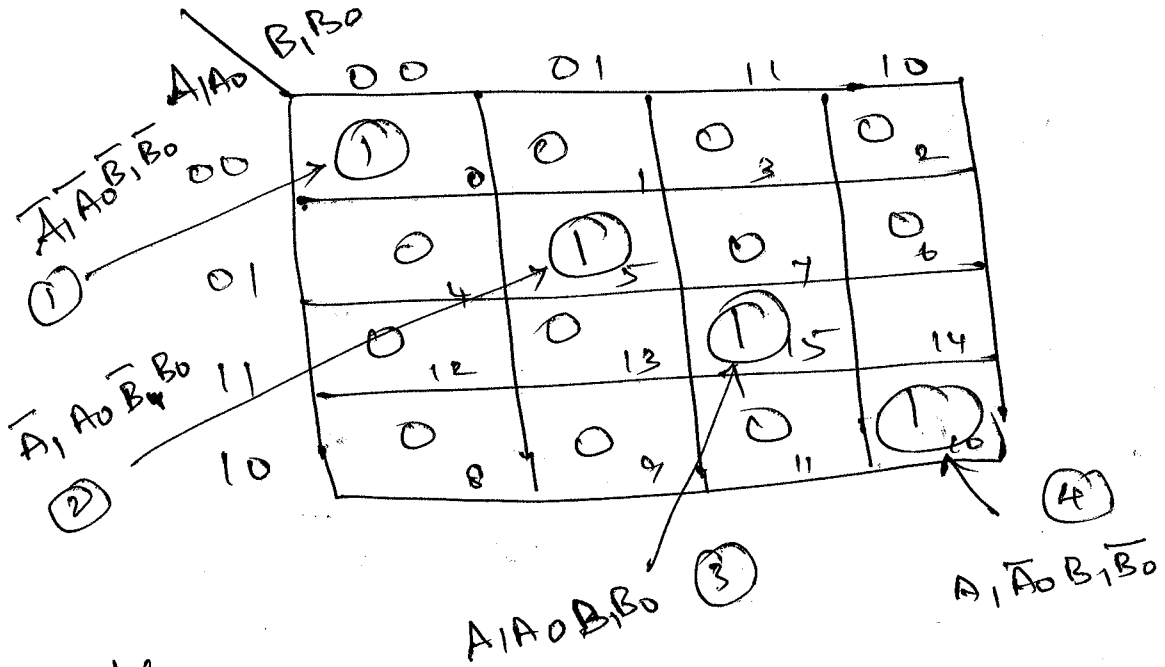
K-map

$Y_{A>B}$



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

$Y_{A=B}$



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

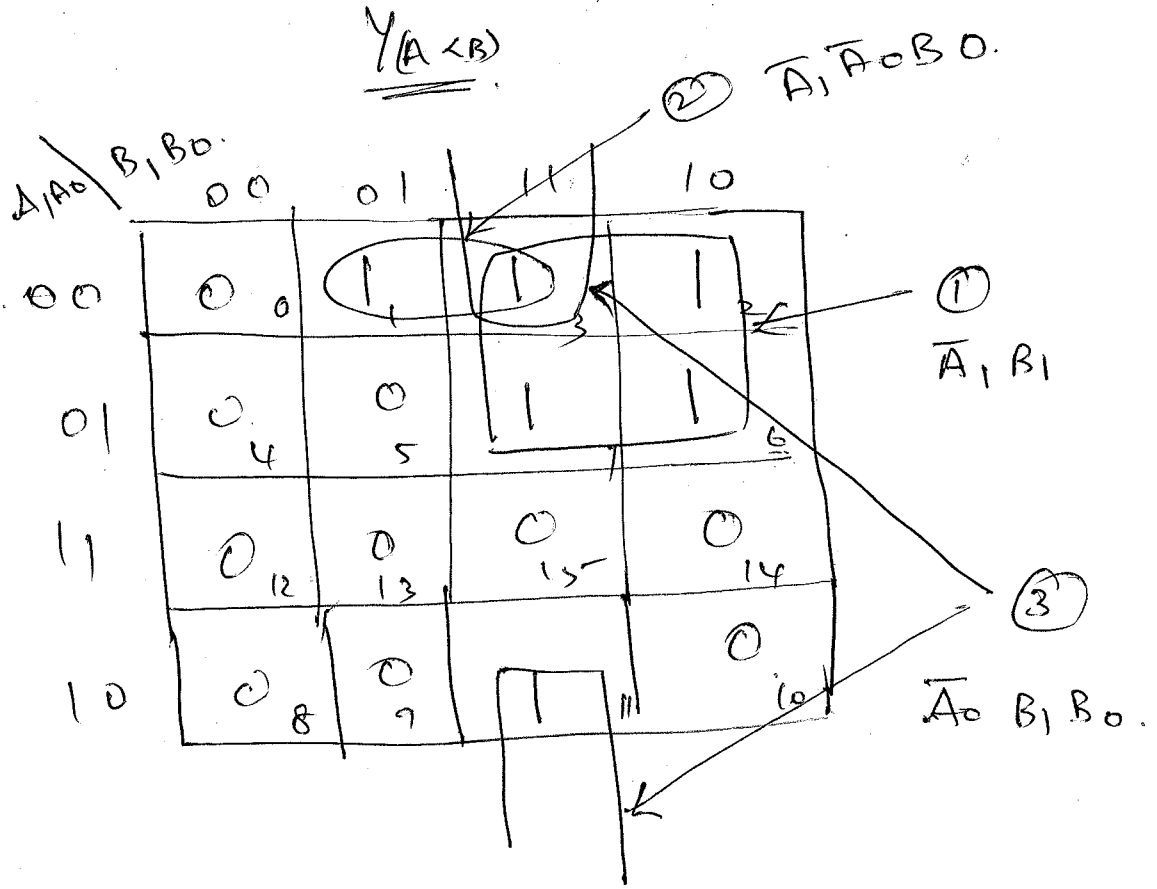
$$= \bar{A}_1\bar{B}_1(\bar{A}_0\bar{B}_0 + A_0B_0) + A_1B_1(A_0B_0 + \bar{A}_0\bar{B}_0)$$

$$= \bar{A}_1 \bar{B}_1 (A_0 \odot B_0) + A_1 B_1 (A_0 \odot B_0)$$

$$= (A_0 \odot B_0)(\bar{A}_1 \bar{B}_1 + A_1 B_1)$$

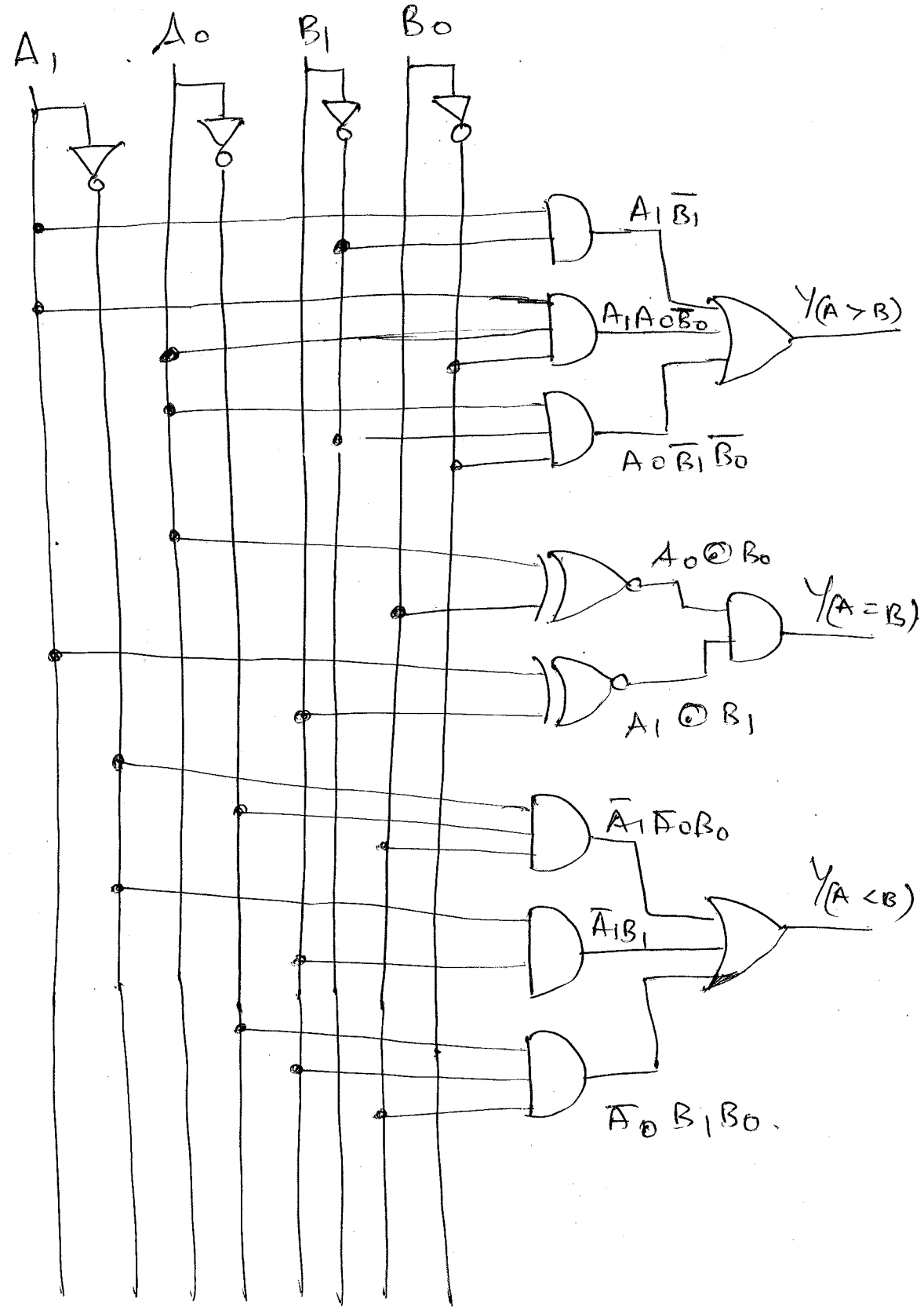
$$Y_{A=B} = (A_0 \odot B_0)(A_1 \odot B_1)$$

$Y_{(A<B)}$



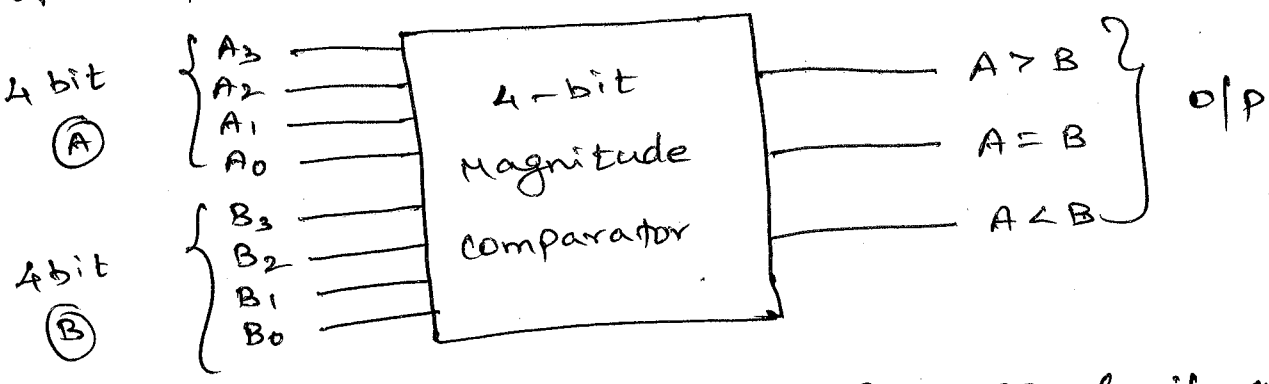
$$Y_{A<B} = \bar{A}_1 \bar{A}_0 B_0 + \bar{A}_1 B_1 + \bar{A}_0 B_1 B_0$$

Logic diagram:



4 - bit Magnitude comparator

A magnitude comparator is a combinational ckt. that compare two numbers A & B. outcome of comparison are A > B, A = B or A < B.



1) A = B : The two numbers are equal if all pairs of significant digits are equal: $A_3 = B_3$, $A_2 = B_2$, $A_1 = B_1$, $A_0 = B_0$. The output should be either '1' or '0' and the equality of each pair of bits can be expressed logically with an exclusive NOR function as $x_i = A_i B_i + A_i' B_i'$ where $i = 0, 1, 2, 3$

$$A = B = x_3 \cdot x_2 \cdot x_1 \cdot x_0$$

$x_3 = \overline{A_3 \oplus B_3}$; $x_2 = \overline{A_2 \oplus B_2}$; $x_1 = \overline{A_1 \oplus B_1}$; $x_0 = \overline{A_0 \oplus B_0}$

equal means both (A_i, B_i) should be '1' or '0' then ANS = 1
∴ XNOR is used.

$A = B \Rightarrow x_3 x_2 x_1 x_0$

= 1

AND gate

2) A > B

$A_3 > B_3 \Rightarrow A_3 B_3'$

$A_2 > B_2 \Rightarrow x_3 A_2 B_2'$

$A_1 > B_1 \Rightarrow x_3 x_2 A_1 B_1'$

$A_0 > B_0 \Rightarrow x_3 x_2 x_1 A_0 B_0'$

If any one is high then A > B
∴ OR operation,

Starting from MSB, check the magnitude of pairs. If 2 digit of pair are equal, we compare the next lower significant pair of digits. The comparison continues until pair of unequal digits is reached. A > B → A = 1, B = 0
A < B → A = 0, B = 1

$$A > B = A_3 B_3' + x_3 A_2 B_2' + x_3 x_2 A_1 B_1' + x_3 x_2 x_1 A_0 B_0'$$

3) $A < B$

$A_3 < B_3 \rightarrow A_3' B_3$

$A_2 < B_2 \rightarrow A_2' B_2$

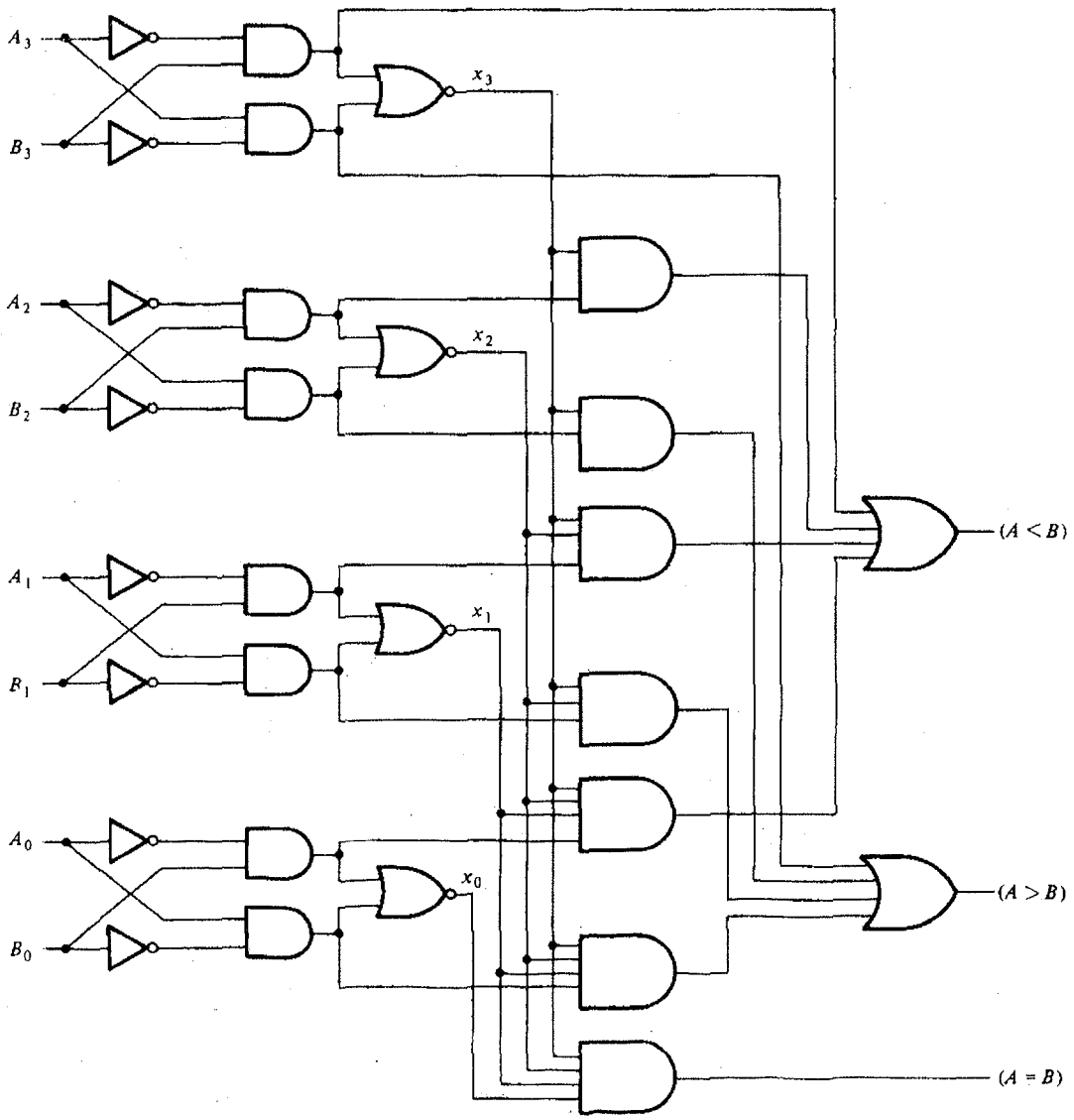
$A_1 < B_1 \rightarrow A_1' B_1$

$A_0 < B_0 \rightarrow A_0' B_0$

If any one is high, then $A < B$

∴ OR operation

$$A < B = A_3' B_3 + x_3 A_2' B_2 + x_3 x_2 A_1' B_1 + x_3 x_2 x_1 A_0' B_0.$$

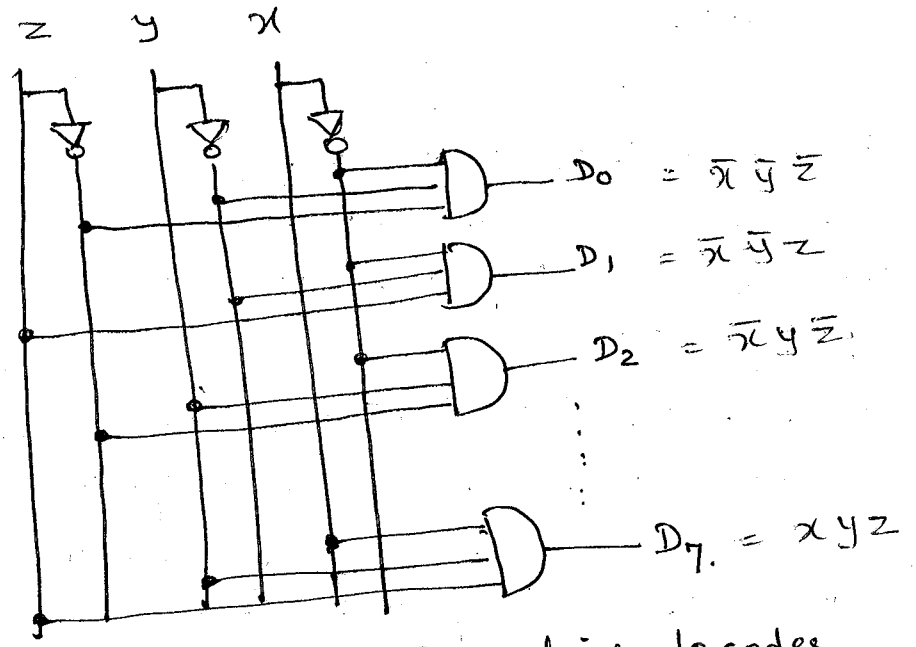


DECODER.

* A binary code of 'n' bits is capable of representing upto 2^n distinct elements of coded information.
* Decoder ⇒ a combinational circuit that converts binary information from 'n' input lines to maximum of 2^n unique output lines.
* If n-bit coded information has unused combination the decoder may have fewer than 2^n outputs.
* n to m line decoder where $m \leq 2^n$ generate 2^n or fewer minterms of 'n' input variables.
* decoder is also called as code converter such as BCD to seven-segment decoder.
* 3 inputs are decoded into 8 outputs, each representing one of the minterms of 3 input variables.
* for each possible input combination, there are seven outputs that are equal to '0' and only one that is equal to '1'.

3^{n} - 8^{m} line decoder.

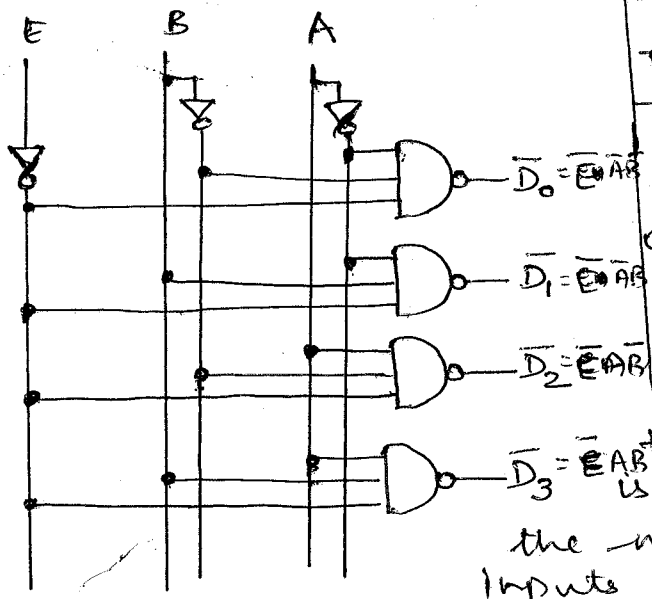
inputs	outputs							
x y z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0 0 0	1	0	0	0	0	0	0	0
0 0 1	0	1	0	0	0	0	0	0
0 1 0	0	0	1	0	0	0	0	0
0 1 1	0	0	0	1	0	0	0	0
1 0 0	0	0	0	0	1	0	0	0
1 0 1	0	0	0	0	0	1	0	0
1 1 0	0	0	0	0	0	0	1	0
1 1 1	0	0	0	0	0	0	0	1



3 : 8 line decoder.

CONSTRUCT DECODER WITH NAND.

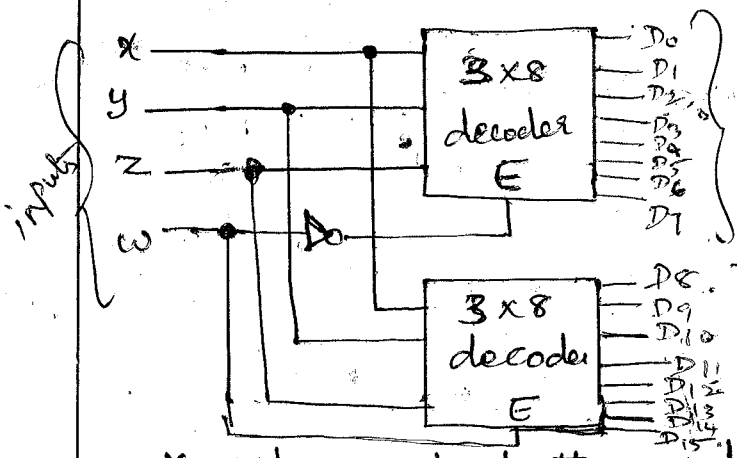
* AND with Inverter output
* Decoder include one or more enable input to control the circuit operation
* Decoder is enabled when E = 0
* only one one output can be equal to 0 at any given time all other outputs are equal to 1.



Input			output			
E	A	B	$\overline{D}_0$	$\overline{D}_1$	$\overline{D}_2$	$\overline{D}_3$
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

* The outputs whose value is equal to 0 represents the minterm selected by inputs A and B. (active low)
* Circuit disabled when E = 1, regardless of the values of other two inputs.

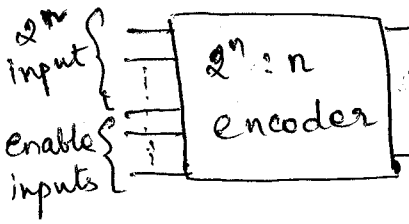
4 x 16 Decoder constructed with two 3 x 8 decoder.



* When w=0, top decoder is enabled and the other disabled.
* The bottom decoder output are all 0's and top 8 output generate minterms 0000 to 0111.

* When w=1, bottom decoder o/p generate minterms 1000 to 1111.

ENCODER.

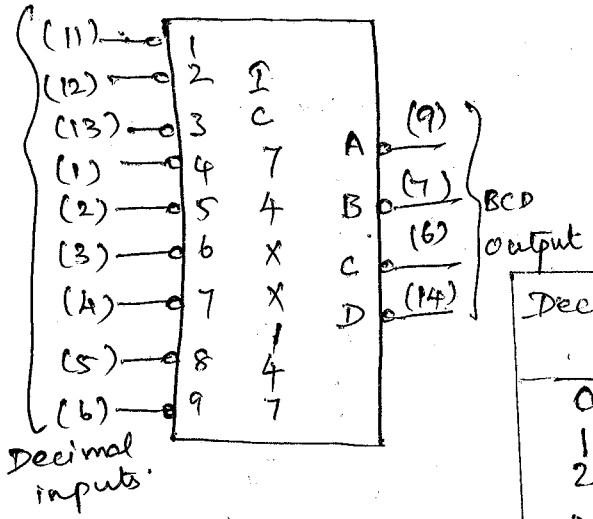


* 2^n input lines and n output lines.
* output lines generate binary code corresponding to input value.

DECIMAL TO BCD ENCODER.

Decimal → 10 input lines.

BCD → 4 output lines.



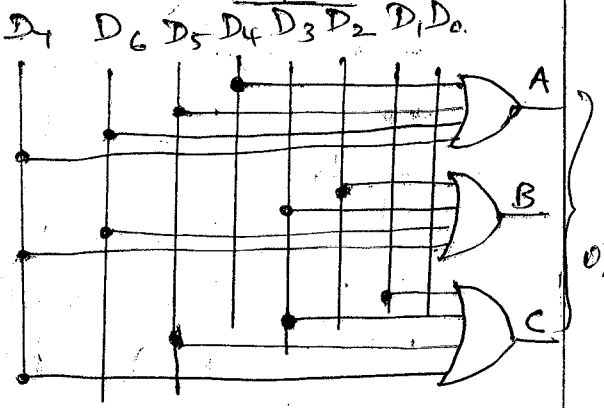
* active low → input and output
* No input line for decimal zero.
* When this condition occurs all input lines are zero.

Decimal	Inputs									Output			
	1	2	3	4	5	6	7	8	9	D	C	B	A
0	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	1	1	1	1	1	1	1	1	1	1	1	0
2	X	0	1	1	1	1	1	1	1	1	1	0	1
3	X	X	0	1	1	1	1	1	1	1	1	0	0
4	X	X	X	0	1	1	1	1	1	1	0	1	1
5	X	X	X	X	0	1	1	1	1	1	0	1	0
6	X	X	X	X	X	0	1	1	1	1	0	0	1
7	X	X	X	X	X	X	0	1	1	1	0	0	0
8	X	X	X	X	X	X	X	0	1	0	1	1	1
9	X	X	X	X	X	X	X	X	0	0	1	1	0

Octal to binary encoder.

* eight inputs, three outputs
* Only one input has the value 1 at any given time

Input.								output		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	A	B	C
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
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	1	0
0	0	0	0	0	0	0	1	1	1	1



* When all inputs are zero, the outputs are zero.
* Zero output can be generated when $D_0 = 1$.

PRIORITY ENCODER.

* Priority function
* If 2 or more inputs are equal to 1 at the same time, the input having highest priority.

Inputs.				output		
D_0	D_1	D_2	D_3	Y_1	Y_0	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

* V → Valid output indicator
* D_3 has the highest priority and D_0 has lowest priority.
* When D_3 input is high, regardless of other inputs output is 11.
* D_2 has next priority.
* $D_3 = 0$, $D_2 = 1$ regardless of lower priority input, output is 10.

K-map:

For Y_1

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

$Y_1 = D_2 + D_3$

For Y_0

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

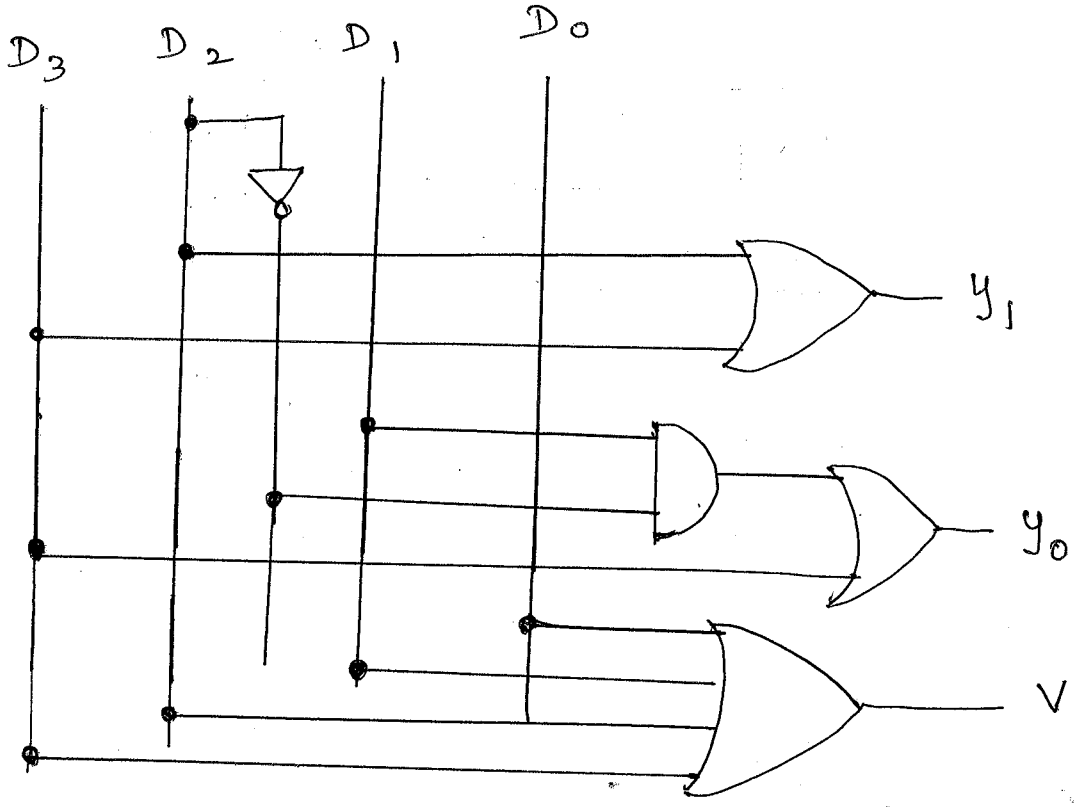
$Y_0 = D_1\bar{D}_2 + D_3$

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

$V = D_0 + D_1 + D_2 + D_3$

Logic diagram



Parity generator & checker

Parity generator

3 bit message			odd Parity bit	even parity bit
A	B	C		
0	0	0	1	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0		1
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

Even parity generator.

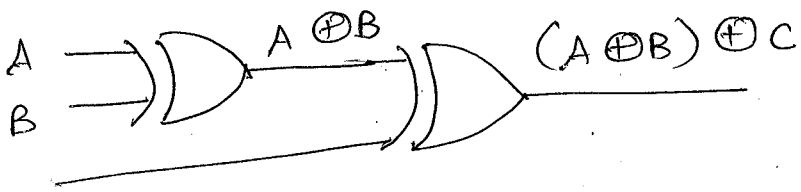
A \ BC	00	01	11	10
0	0	1	0	1
1	1	0	1	0

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

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

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

$= A \oplus B \oplus C$



Even parity checker

* If zero accepts

Decimal	4 bit received A B C D	parity error check
0	0 0 0 0	0
1	0 0 0 1	1
2	0 0 1 0	1
3	0 0 1 1	0
4	0 1 0 0	1
5	0 1 0 1	0
6	0 1 1 0	0
7	0 1 1 1	1
8	1 0 0 0	1
9	1 0 0 1	0
10	1 0 1 0	0
11	1 0 1 1	1
12	1 1 0 0	0
13	1 1 0 1	1
14	1 1 1 0	1
15	1 1 1 1	0

AB \ CD	00	01	11	10
00	0	1	0	1
01	1	0	1	0
11	0	1	0	1
10	1	0	1	0

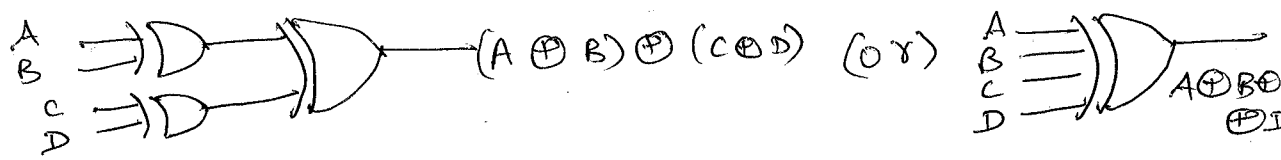
$= \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}\bar{D} + \bar{A}BCD + AB\bar{C}D + ABC\bar{D} + A\bar{B}\bar{C}\bar{D} + A\bar{B}CD$

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

$= \bar{A}\bar{B}[C \oplus D] + \bar{A}B[\overline{C \oplus D}] + AB[C \oplus D] + A\bar{B}[\overline{C \oplus D}]$

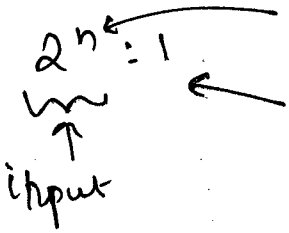
$= [C \oplus D][\bar{A}\bar{B} + AB] + [\overline{C \oplus D}][\bar{A}B + A\bar{B}]$

$= [C \oplus D][\overline{A \oplus B}] + [\overline{C \oplus D}][A \oplus B] = [C \oplus D] \oplus [A \oplus B]$

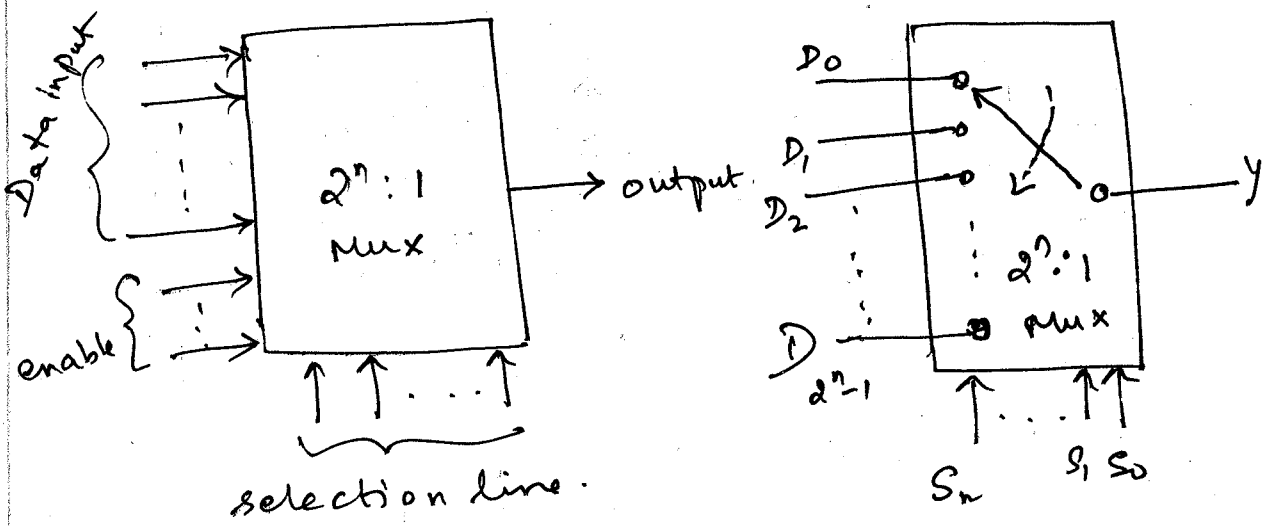


Multiplexer

It selects binary information from many input lines and directs it to a single output line.



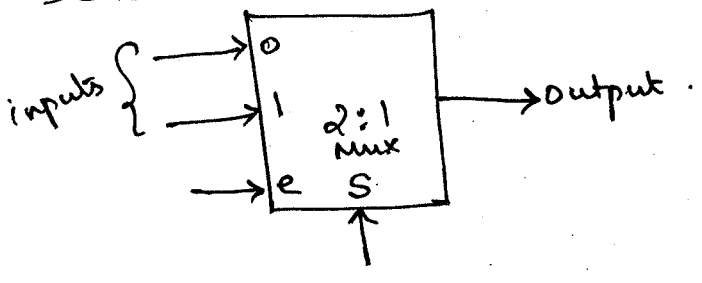
The selection of particular input lines is controlled by a set of selection lines. enable or strobe is used for cascading.



It is otherwise called data selector and is used in D/A converter, data acquisition system.

2 to 1 line Multiplexer

It has 2 inputs and one output. Since binary input is two, ~~enable~~ need is only one. Selection line.

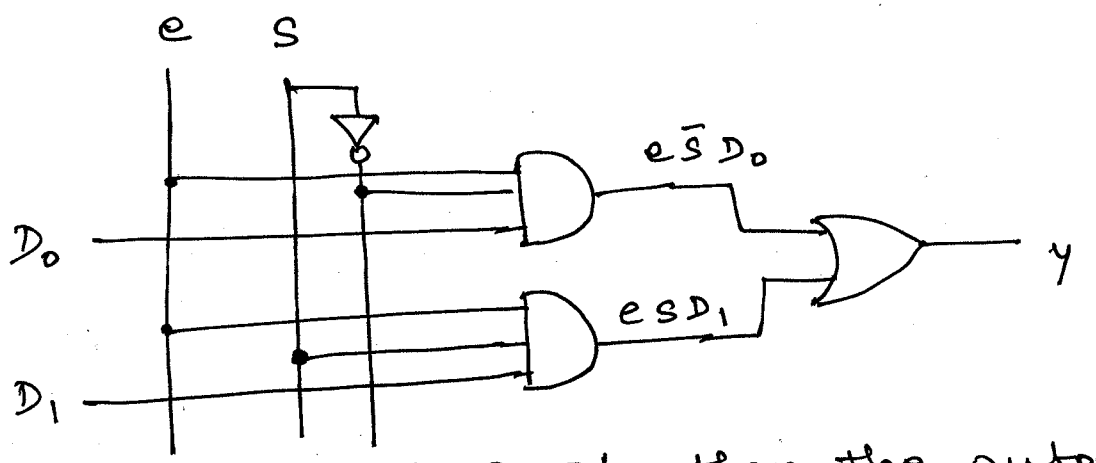


Truth table

e	S	Y	
1	0	D_0	$e\bar{S}D_0$
	1	D_1	eSD_1
0	x	0	

$Y = e\bar{S}D_0 + eSD_1$

logic diagram.

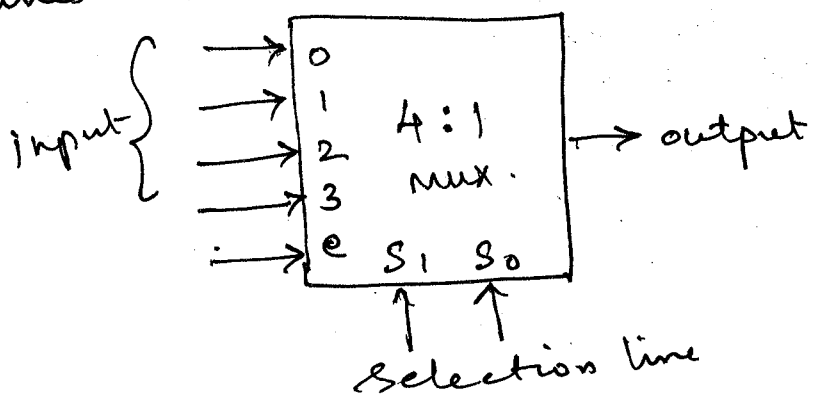


Case i) $S=0$, $e=1$, $D_0=1$, then the output is D_0.
otherwise output is '0'.

Case ii) $S=1$, $e=1$, $D_1=1$, then output is D_1,
otherwise output is '0'.

4 to 1 line multiplexer.

It has ~~four~~ inputs and one output. since 4 inputs (2^2) it needs two selection lines.

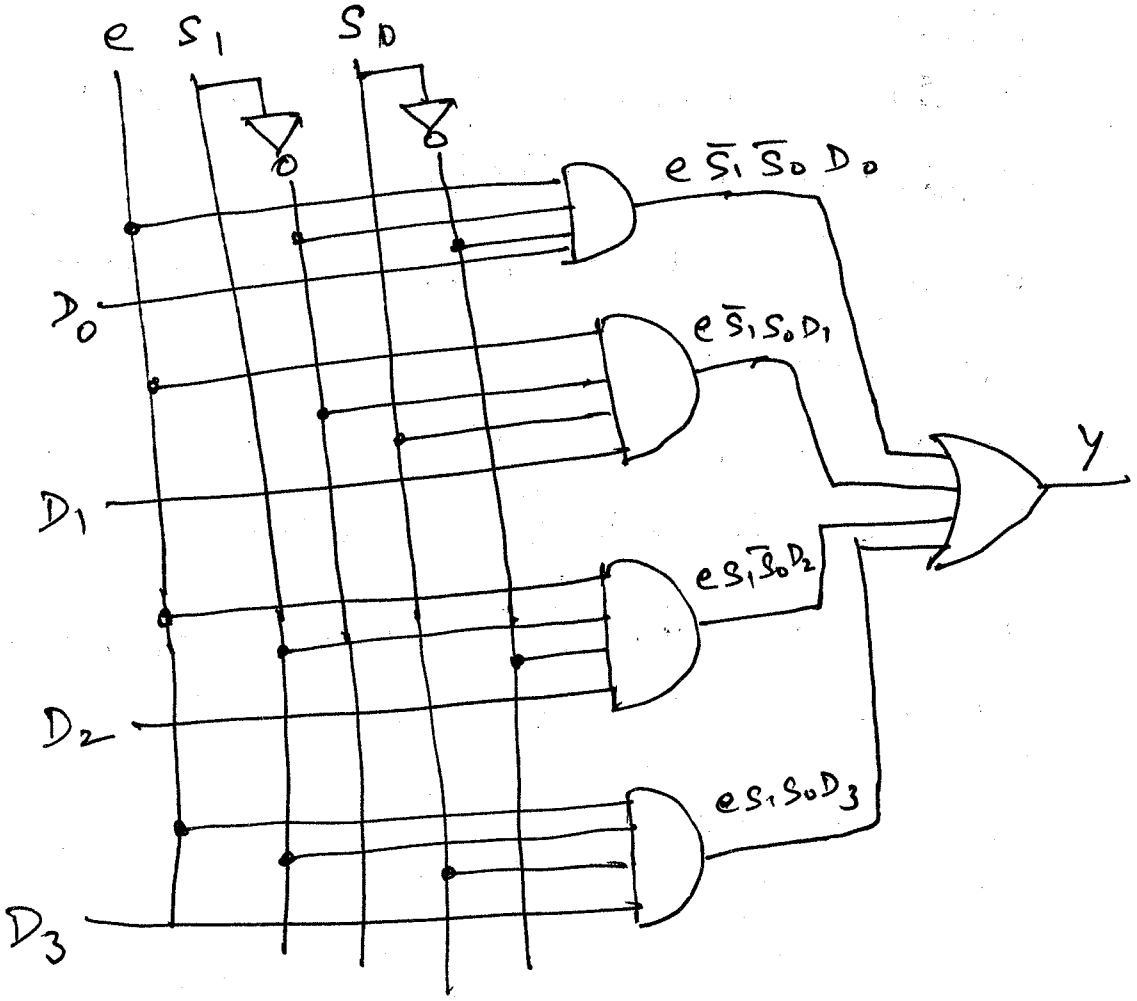


Truth table

e	S_1	S_0	Y	
1	0	0	D_0	$e\bar{S}_1\bar{S}_0D_0$
1	0	1	D_1	$e\bar{S}_1S_0D_1$
1	1	0	D_2	$eS_1\bar{S}_0D_2$
1	1	1	D_3	$eS_1S_0D_3$.
0	X	X	0	

$$Y = e\bar{S}_1\bar{S}_0D_0 + e\bar{S}_1S_0D_1 + eS_1\bar{S}_0D_2 + eS_1S_0D_3.$$

Logic diagram



If enable $e = 1$

Case i) $S_1 = 0$, $S_0 = 0$, $D_0 = 1$ then output is D_0 otherwise output is '0'

Case ii) $S_1 = 0$, $S_0 = 1$, $D_1 = 1$ then output is D_1, otherwise output is '0'

Case iii) $S_1 = 1$, $S_0 = 0$, $D_2 = 1$, then output is D_2, otherwise output is '0'.

Case iv) $S_1 = 1$, $S_1 = 1$, $D_3 = 1$, then output is D_3, otherwise output is '0'.

Implementation of Multiplexer.

steps:

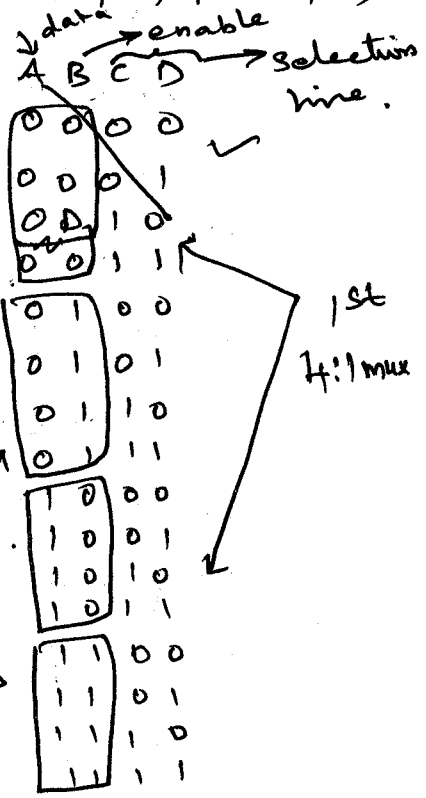
1. If 2 minterm in column are not circled '0' is applied.
2. If 2 minterm in column are circled the value is '1'.
3. Minterm in 1st row is circled, represent the Complemented variable.
4. Minterm in 2nd row is circled and minterm represent as variable.

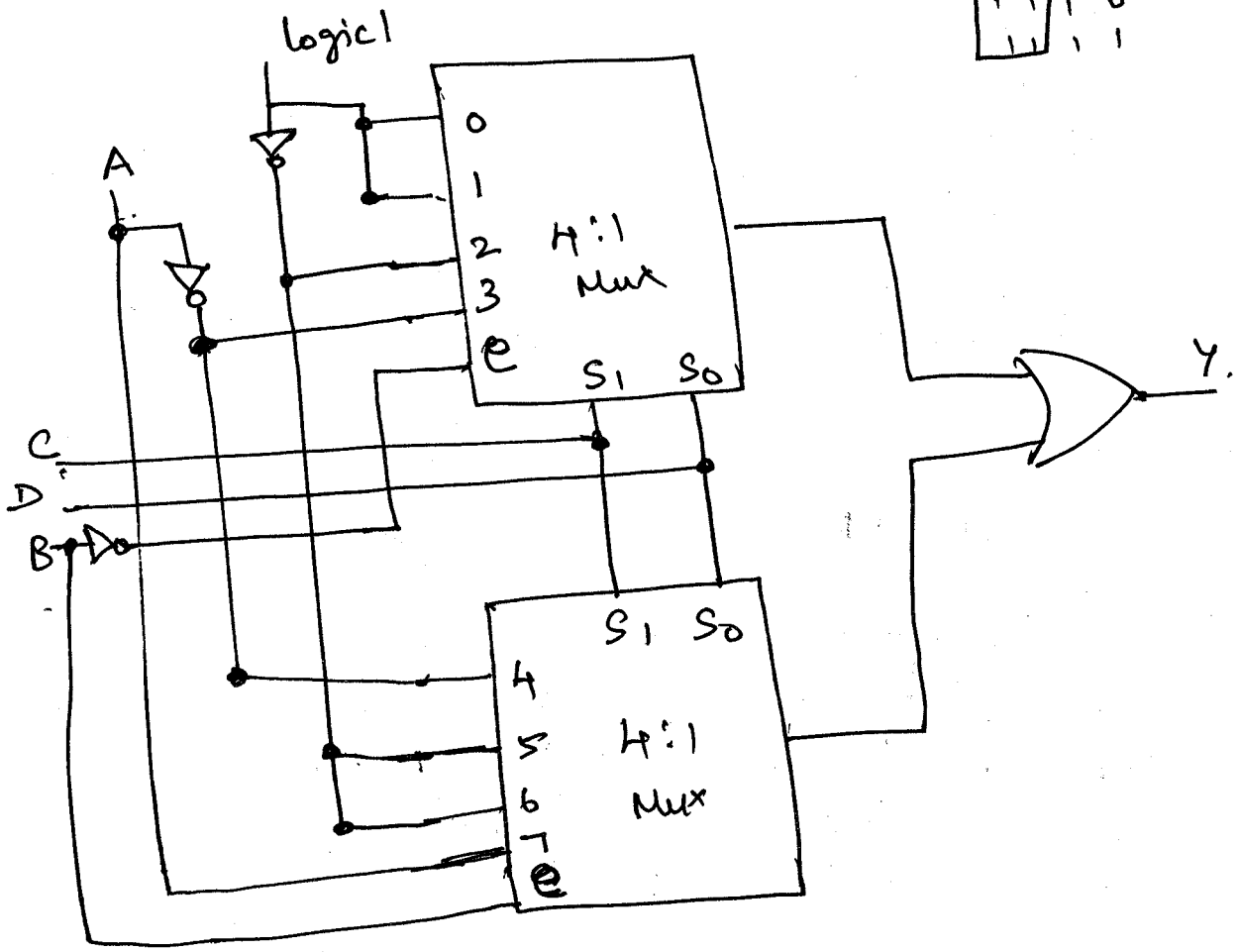
Implement the given function using 4:1 Mux. $f(A, B, C, D) = \Sigma m(0, 1, 3, 4, 8, 9, 15)$

	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	1	0	$\bar{A}$	$\bar{A}$	0	0	A

4:1 Mux

4:1 Mux. and 4:1 mux.

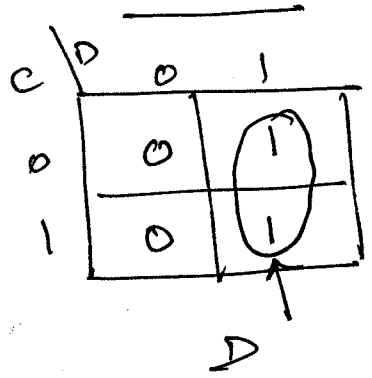




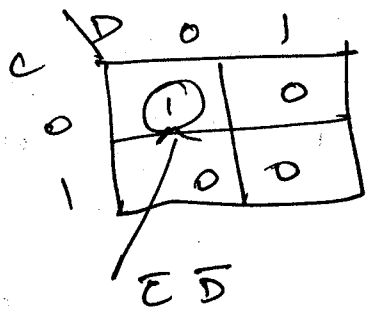
Implement the following boolean function with a 4 to 1 mux and external gates. Connect Inputs A and B to selection line. The input requirement for 4 data lines will be function of variable C and D. These values are obtained by expressing F as function of C & D for each of 4 cases when AB = 00, 01, 10, 11. These functions may have to be implemented with external gates.

$$F(A,B,C,D) = \sum m(1,3,4,11,12,13,15)$$

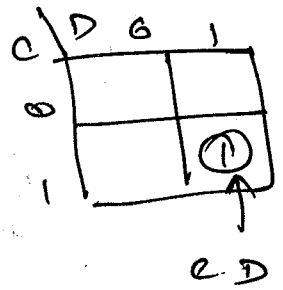
AB = 00



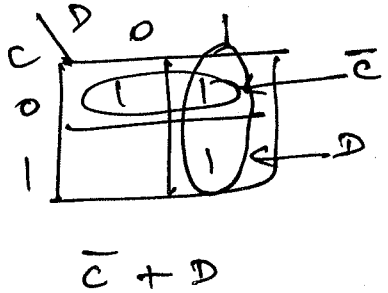
AB = 01

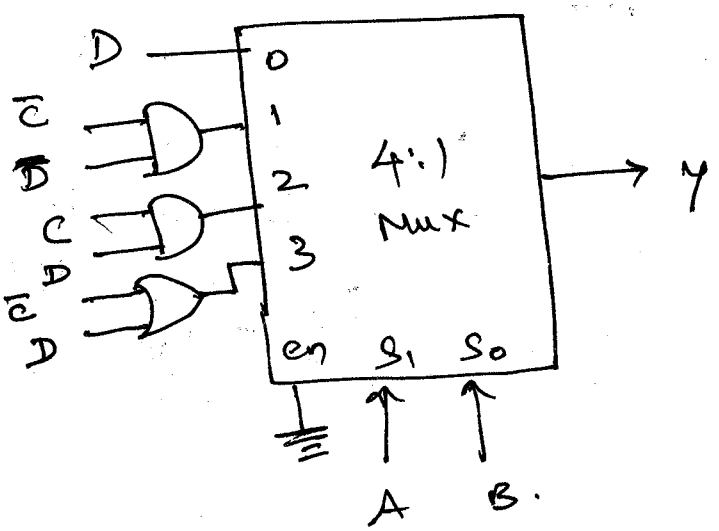


AB = 10



AB = 11





Implement using Multiplexer (IC74150)

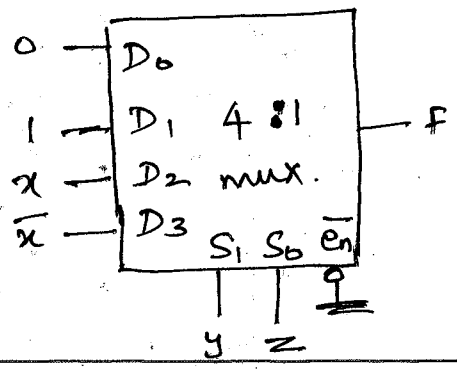
1. $f(x, y, z) = \Sigma(1, 3, 5, 6)$

Data → x | S: y z | f

	x	y	z	f
$\bar{x}$ 0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	0
x 4	1	0	0	0
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

	00	01	10	11
	D_0	D_1	D_2	D_3
$\bar{x}$	0	(1)	2	(3)
x	4	(5)	(6)	7
	0	1	x	$\bar{x}$

* 2 minterm in column are not circled '0' is applied.
* 2 minterm are circled Value = 1
* Minterm in 2nd row is circled and minterm is the Variable.
* Minterm in 1st row is circled, represent the complemented Variable.



2. $f(A, B, C, D) = \Sigma m(0, 1, 3, 4, 8, 9, 15)$

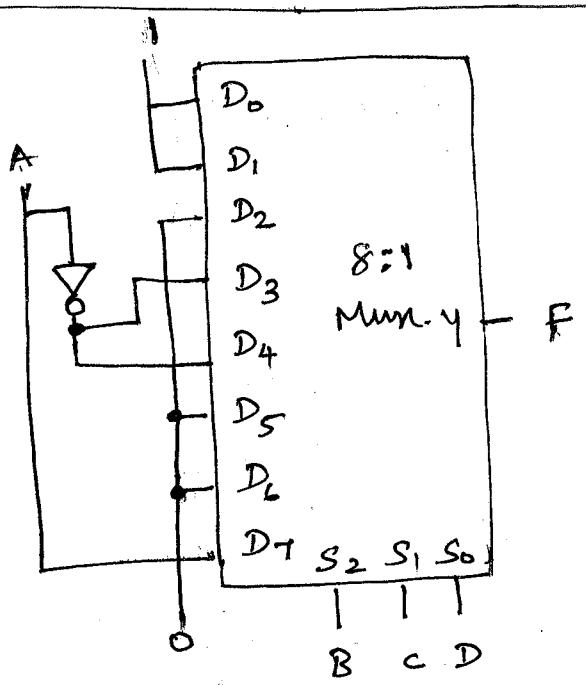
Selection line → A B C D

Data →

	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
	000	001	010	011	100	101	110	111
$\bar{A}$	(0)	(1)	2	(3)	(4)	5	6	7
A	(8)	(9)	10	11	12	13	14	(15)
	1	1	0	$\bar{A}$	$\bar{A}$	0	0	A

$\bar{A}$: 0 0 0 0 0 0 0 0

A: 1 0 0 0 1 1 1 1 0 0 1 1 0 1 0 1



DEMULTIPLEXER.

* receives information on a single line and transmits this information on one of 2^n possible Output lines.

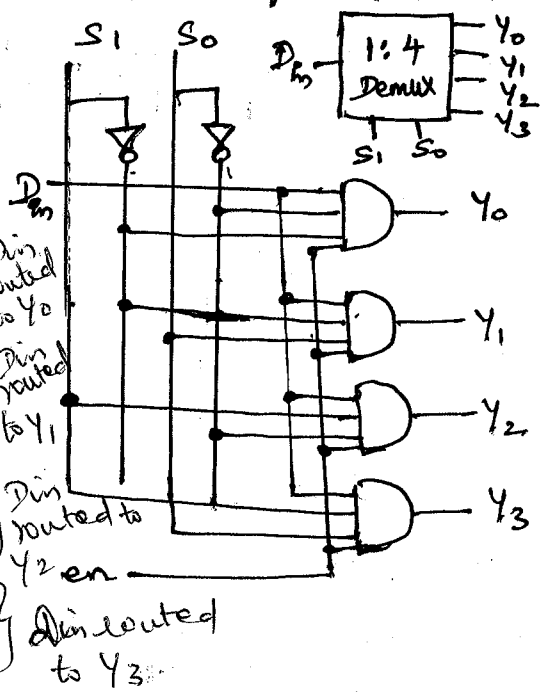
en	S_1	S_0	Din	Y_0	Y_1	Y_2	Y_3	
0	x	x	x	0	0	0	0	
1	0	0	0	0	0	0	0	
1	0	0	1	(1)	0	0	0	Din routed to Y_0
1	0	1	0	0	0	0	0	
1	0	1	1	0	(1)	0	0	Din routed to Y_1
1	1	0	0	0	0	0	0	
1	1	0	1	0	0	(1)	0	Din routed to Y_2
1	1	1	0	0	0	0	0	
1	1	1	1	0	0	0	(1)	Din routed to Y_3

en $\bar{S}_1\bar{S}_0$ Din

en $\bar{S}_1 S_0$ Din

en $S_1\bar{S}_0$ Din

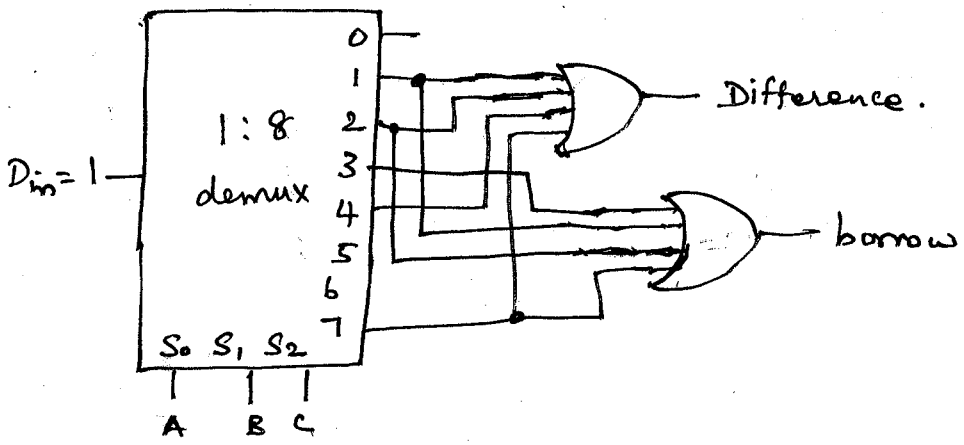
en $S_1 S_0$ Din



Implement full subtractor using demultiplexer. (IC 74154)

A	B	Bin	D	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

$D = \Sigma m(1, 2, 4, 7)$, $B = \Sigma m(1, 2, 3, 7)$.

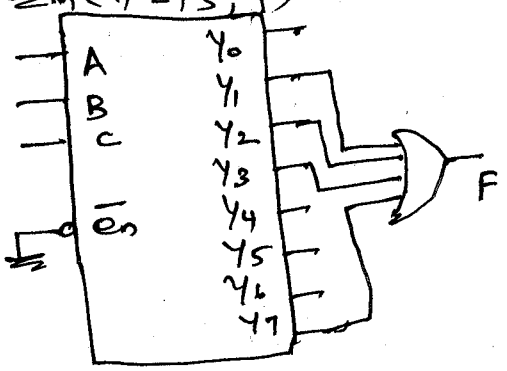


Realization of multiple output function using binary decoder.

For active high output.

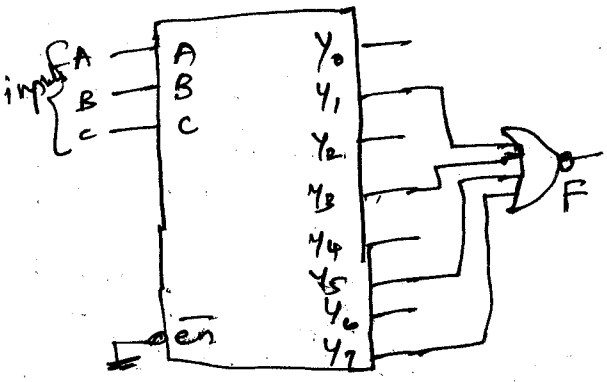
SOP function Implementation

$f = \Sigma m(1, 2, 3, 7)$



POS function implementation

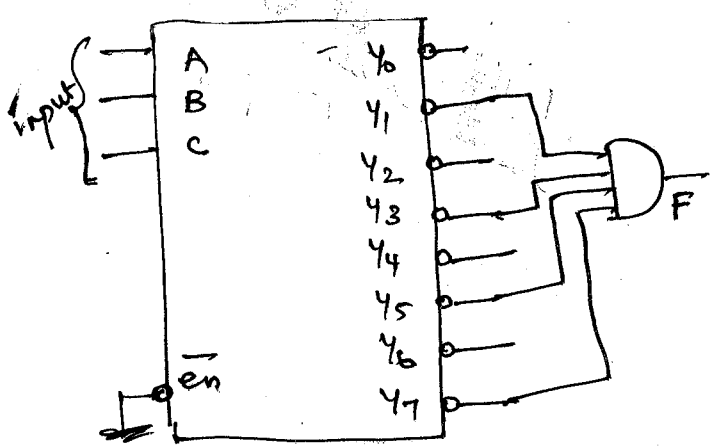
$f = \Pi m(1, 3, 5, 7)$



For Active low output

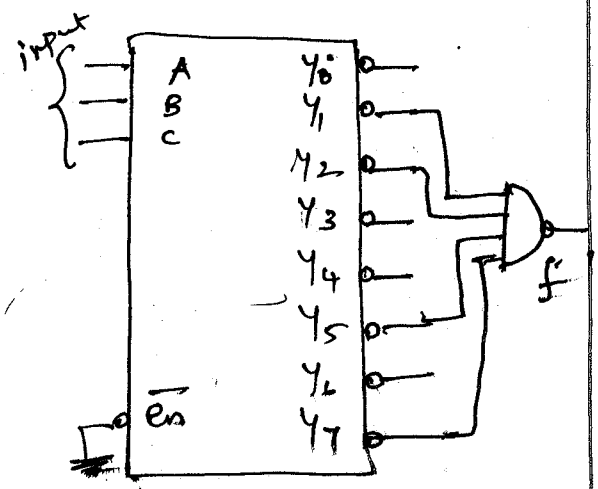
POS function implementation

$f = \Pi m(1, 3, 5, 7)$



SOP function Implementation

$f = \Sigma m(1, 2, 5, 7)$



Design full adder use 3:8 decoder.

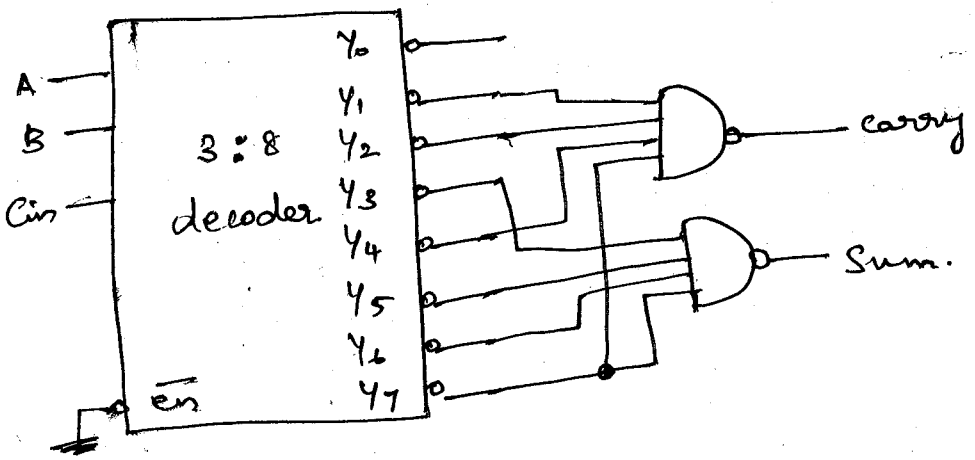
Input A B Cin	Carry	Sum
0 0 0	0	0
0 0 1	0	1
0 1 0	0	1
0 1 1	1	0
1 0 0	0	1
1 0 1	1	0
1 1 0	1	0
1 1 1	1	1

$sum = x'y'z + x'yz' + xy'z' + xyz$

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

$Carry = xy + xz + yz.$

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



Multiplexer Vs Decoder.

* Multiplexer has several data-input lines and a single output line. The selection of a particular input line is controlled by set of selection lines.
* The data of selected input line is routed to output line.
* Decoder has 'n' input lines and 2^n output lines. each o/p line represent one minterm
* Decoder activates one of the output line depending on the input Combination