CCE 221

UNIQUE
WE GIVE PREMIUM QUALITY OVER QUANTITY
Write Your Idea

Name Reem

Roll/ID 1802071 Course Code

Sub/Course Title Faculty/Dept CSE

Institute

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

Reference Book

~~Diti~~ Digital logic an Computer Design

by Morris Mano

5th edition

Plan Title : Online Class-01

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date : 28/02/2021

Chapter - 01

Binary Systems

A decimal number - $7392 = 7\times10^3+3\times10^2+9\times10^1+2\times10^0$

* Decimal number system is said to be of base or radix 10, because it uses ten digits & the co-efficients are multiplied by power of 10.

  * radix 7, radix 5, radix 4

* Binary $(11010.11) = 1\times2^4+1\times2^3+0\times2^2+1\times2^1+0\times2^0+1\times2^{-1}+1\times2^{-2}$

  $= 26.75$ (Decimal)

* Base-5 number is

  $(4021.2)_5 = 4\times5^3+0\times5^2+2\times5^1+1\times5^0+2\times5^{-1}$

  $= 511.$

Coefficient values → 0, 1, 2, 3, 4

* Base-16 number :

$$(B65F)_{16} = 11 \times 16^3 + 6 \times 16^2 + 5 \times 16^1 + 15 \times 1$$

=

A = 10

B = 11

C = 12

D = 13

E = 14

F = 15

* No base conversions

Binary to decimal → $(1010.011)_2 = 2^3 + 2^1 + 2^{-1} + 2^{-3}$

$= (10.375)_{10}$

Octal to decimal → $(630.4)_8 = 6 \times 8^2 + 3 \times 8^1 + 0 \times 8^0 + 4 \times 8^{-1}$

$= (408.5)_{10}$

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Convert decimal 41 to binary

2	41	1
2	20	0
2	10	0
2	5	1
2	2	0
	1	

$\frac{41}{2}$ =	Integer quotent	remainder	co-effici
	20	$+ 1/2$	$a_0 = 1$
$\frac{20}{2}$ =	10	$+ 0$	$a_1 = 0$
$\frac{10}{2}$ =	5	$+ 0$	$a_2 = 0$
$\frac{5}{2}$ =	2	$+ 1/2$	$a_3 = 1$
$\frac{2}{2}$ =	1	$+ 0$	$a_4 = 0$
$\frac{1}{2}$ =	0	$+ 1/2$	$a_5 = 1$

$(41)_{10} = (101001)_2$

$= (a_5 a_4 a_3 a_2 a_1 a_0)_2$

* The conversions from decimal ~~to~~ integers to any base - r system is similar to the above example, except that division is done by r instead of 2.

Ex-1.2: Decimal 153 to Octal

8	153	
8	19	1
8	2	3
	0	2

↑

$= (231)_8$

Ex-1.3:

	Integer	Fraction	Co-efficient
$\cdot 6875 \times 2$	1	$\cdot 3750$	$a_{-1} = 1$
$\cdot 3750 \times 2$	0	$\cdot 7500$	$a_{-2} = 0$
$\cdot 7500 \times 2$	1	$\cdot 5000$	$a_{-3} = 1$
$\cdot 5000 \times 2$	1	$\cdot 0000$	$a_{-4} = 1$

$(\cdot 6875)_{10} = (\cdot a_{-1}\ a_{-2}\ a_{-3}\ a_{-4})_2$

$= (\cdot 1011)_2$

Plan Title : Online Class -02

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date : 15/03/2021

* Complement কেন use করা হয় ?

⇒ Complement is used in digital computers for simplifying the subtraction operation and logical manipulation.

* r's complement of N : $r^n - N$

* 10's complement of $(52520)_{10} = 10^5 - 52520$

N = positive number

r = integer part of n digit

$= 100000$

52520

47480

* 10's complement of $(0.3267)_{10}$ is $1 - .3267 = .6733$

* 10's complement of $(25.639)_{10}$ is $10^2 - 25.639$

$= 100$

25.639

74.361

* 2's complement of $(101100)_2 = (2^6)_{10} - 101100$

$= 1000000$

101100

010100

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

*2's complement of $(.0110)_2 = (1 - .0110)_2$

$= 1$

0.0110

$.1010$

{* (r-1)'s complement of $N = r^n - r^{-m} - N$

m = no. of fraction part

$10^{-m} = 10^{-0} = 1$

*9's complement of $(52520)_{10} = 10^5 - 10^{-m} - 52520$

$= 10^5 - 1 - 52520$

$= 99999$

52520

47479

*9's complement of $(0.3267)_{10} = 1 - 10^{-4} - 0.3267$

$= 1 - \frac{1}{1000} - .3267$

$= 1 - 0.0001 - .3267$

$= .9999 - .3267$

$= .6732$

No- integer part $10^{n} = 10^0 = 1$

$(25.639)_{10} = 10^2 - 10^{-3} - 25.639$

$= 99.999 - 25.639$

$= 74.360$

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

1's complement of $(101100)_2$ is $(2^6-1)-101100$

$$= \begin{array}{r} 111111 \\ 101100 \\ \hline 010011 \end{array}$$

1's complement of $(.0110)_2 = (1-2^{-4})_{10} - (0.0110)$

$$= \begin{array}{r} 0.1111 \\ 0.0110 \\ \hline 0.1001 \end{array}$$

* Subtraction with r's complement

Ex-1.5: Using 10's complement, subtract 72532 - 3250

M = 72532 , N = 03250

10's complement of N = +96750

Sum = 169282

Discard end carry 10^5 = -100000

69282

Answer = 69282

Ex-1.6: Using 10's complement, subtract 3250 - 72532

M = 03250

10's complement of N = +27468

Sum = 30718 [No end carry]

Answer = - (10's complement of 30718)

= - 100000

30718

- 69282

Ex-1.7:

M = 1010100

N = 1000100

M = 1010100

2's Complement of N = +0111100

10010000 [end carry 1]

Answer = 1000

2's complement of 10001100 is 01110011

+ 1

01110100

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Reflected Code / Gray Code

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

(Reflected code generation)

Plan Title : Chapter- 02 "Boolean Algebra and Logic Gates"

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Operator Precedence

Operator precedence for evaluating Boolean expression is ① Parenthesis, ② NOT, ③ AND, ④ OR

e.g. for $(x+y)'$

$= (1+0)'$ → expression inside parenthesis is evaluated first.

$= 1'$

$= 0$

Venn Diagram=

* Boolean Function

$F_1 = xyz'$, $F_1 = 1$ when only $x=1$, $y=1$ & $z=0$

$F_2 = x+y'z$, $F_2 = 1$ if $x=1$ or if $y=0$ while $z=1$

Similarly F_3 & F_4

* Implementation of Boolean function with gates=

* Complement of a function=

(Ex- 2.1, 2.2, 2.3) + অন্য প্রমান

* Canonical & Standard Forms =

(Ex - 2.4, 2.5)

* Universal Gate =

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

Chapter - 03

Simplification of Boolean Function

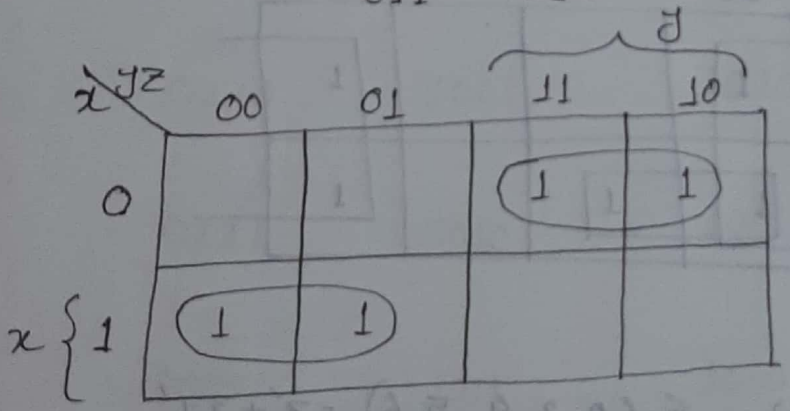
* Map Method : Proposed by Veitch & slightly modified by Karnaugh (2) also known as "Veitch diagram" Karnaugh map.

* Three variable map

Ex - 3.1 : Simplify the boolean function

$F = x'yz + x'yz' + xy'z' + xyz'$

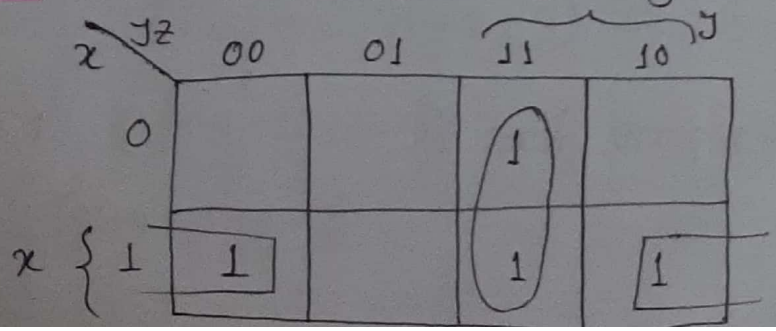
011 010 100 101



[যেই box এ বসবে সেখানে 1 দিতে হবে]

$F = xy' + x'y$ [uncommon part বাদ দিয়ে দিতে হবে]

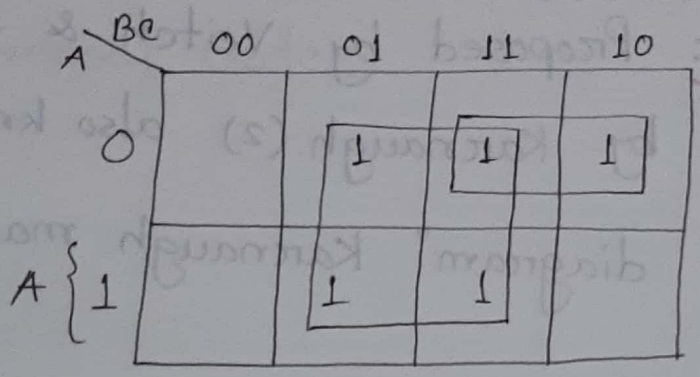
Ex - 3.2 : $F = x'yz + xy'z' + xyz + xyz'$



$F = xz' + yz$

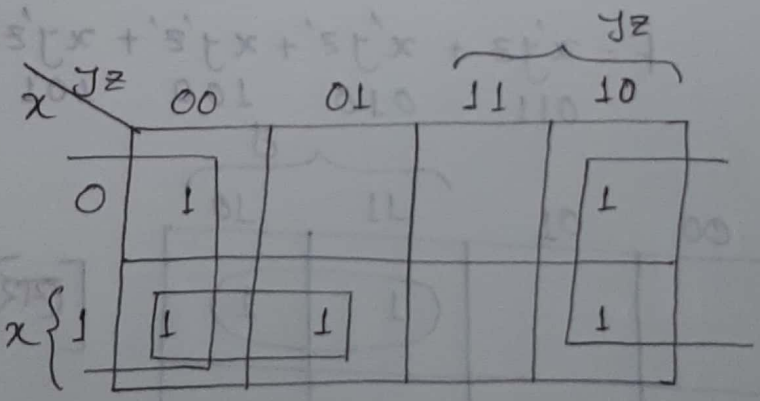
Ex - 3.3:

$$F = A'C + A'B + AB'C + BC$$



$$F = C + A'B$$

Ex - 3.4:



$$f(x, y, z) = \Sigma(0, 2, 4, 5, 6) = z' + xy'$$

pdf - 69 page 3.2 পর্যন্ত

Plan Title : Online Class -04

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date : 21 / 04 / 2021

* Binary Codes for the Decimal Digits (1st Chapter)

Decimal Digits	(BCD) 8421	Excess-3	84-2-1	2421	Biquinary 5043210
0	0000	0011	0000	0000	0100001
1	0001	0100	0111	0001	0100010
2	0010	0101	0110	0010	0100100
3	0011	0110	0101	0011	0101000
4	0100	0111	0100	0100	0110000
5	0101	1000	1011	1011	1000001
6	0110	1001	1010	1100	1000010
7	0111	1010	1001	1101	1000100
8	1000	1011	1000	1110	1001000
9	1001	1100	1111	1111	1010000

- Excess-3 : (Decimal digit + 3) এর Binary মান
- 84-2-1 : (8+4-2-1 = 9) self complementing Code
- 2421 : (2+4+2+1 = 9)
- 5043210 :

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date : / /

* Error detection Code (chapter-1)

যখন কোনো তথ্য পাঠানো হয় তখন ঐ তথ্যের আগে একটা parity যোগ করা হয়, reciever সেটা recieve করে দেখে সেটা ঠিক আছে কিনা।

* Parity bit

Odd Parity

Message	P
0000	1
0001	0
0010	0
0011	1
0100	0
0101	1
0111	0
1000	0
1001	1
1010	1
1011	0
1100	1
1101	0

* even সংখ্যক 1 থাকলে 1 else 0.

Even Parity

Message	P
0000	0
0001	1
0010	1
0011	0
0100	1
0101	0
0111	1
1000	1
1001	0
1010	0
1011	1
1100	0
1101	1

* Odd সংখ্যক 1 থাকলে 1 else 0.

Plan Title :

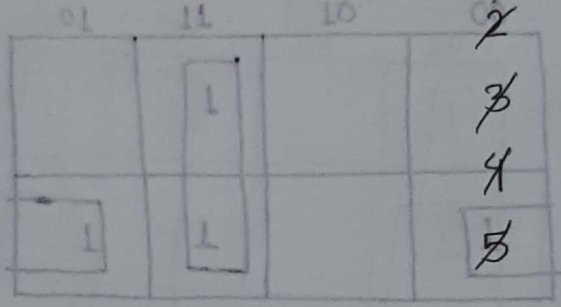
Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

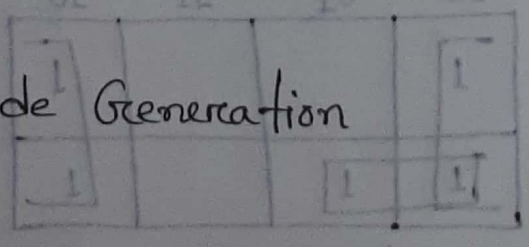
* Gray Code (Chapter-1)

Four bit Gray Code

Gray Code	Decimal Equivalent
0000	0
0001	1
0011	2
0010	3
0110	4
0111	5
0101	6
0100	7
1100	8
1101	9
1111	10
1110	11
1010	12
1011	13
1001	14
1000	15



** Gray Code Generation



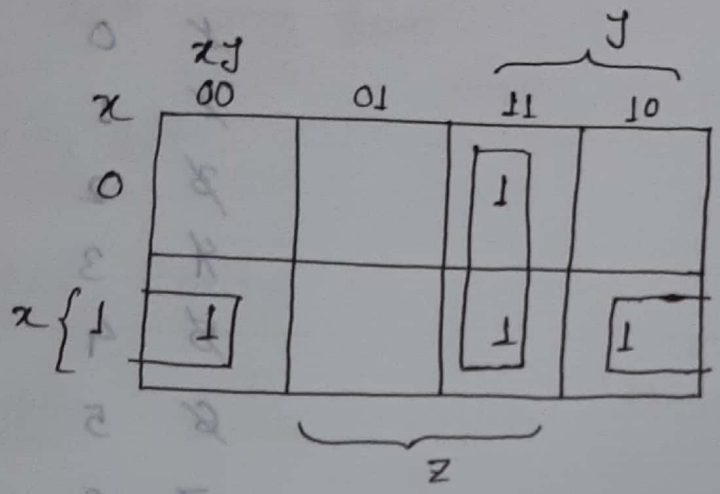
Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Example-3.2: Simplify the boolean function

$$F(x, y, z) = \Sigma(3, 4, 6, 7)$$

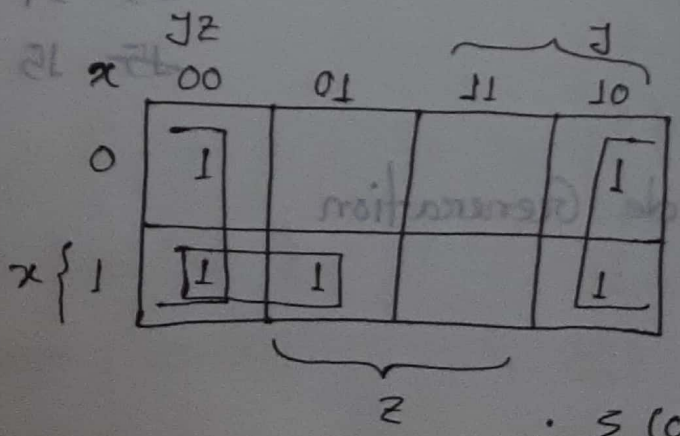


Coverage এর কমে 2 এর power এ নিতে হবে এবং তা adjacent হইতে হবে।

$$\Sigma(3, 4, 6, 7) = yz + xz'$$

* Example-3.3: Simplify the boolean function

$$F(x, y, z) = \Sigma(0, 2, 4, 5, 6)$$



$$\therefore \Sigma(0, 2, 4, 5, 6) = z' + xy'$$

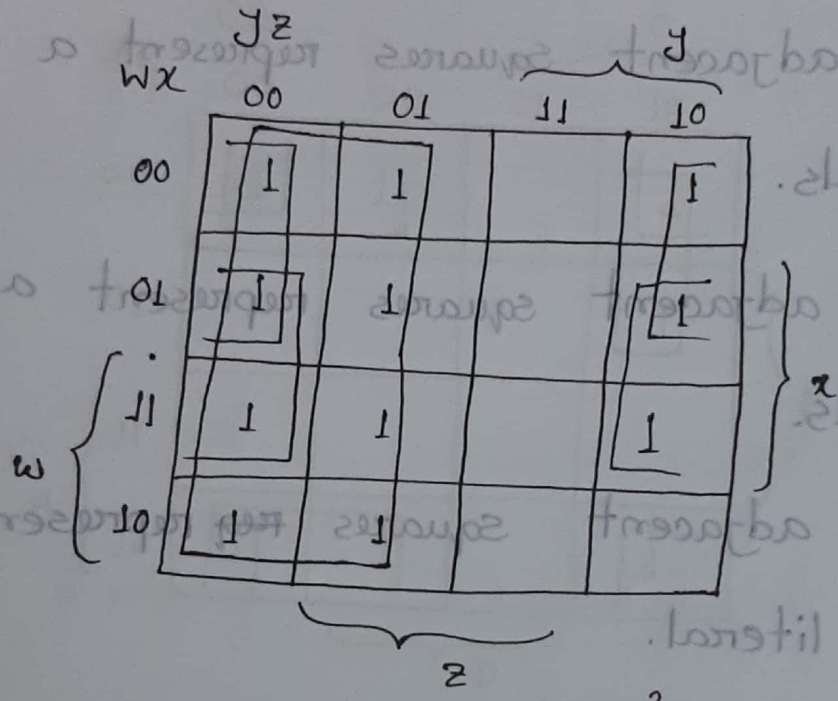
Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Example -3.5: Simplify the boolean function

$$f(w,x,y,z) = \Sigma(0,1,2,4,5,6,8,9,12,13,14)$$



* 2^3 এর Coverage নিতে হবে

$\therefore F(w,x,y,z) = \Sigma(0,1,2,4,5,6,8,9,12,13,14) = y' + wz' + xz'$

*Coverage এর সংখ্যা বাড়লে digit এর সংখ্যা কমে.

8 টা হইলে simplified digit 1 টা

4 টা হইলে " " 2 টা

2 টা হইলে " " 3 টা

1 টা হইলে " " 4 টা

* 16 টা হইলে common 1.

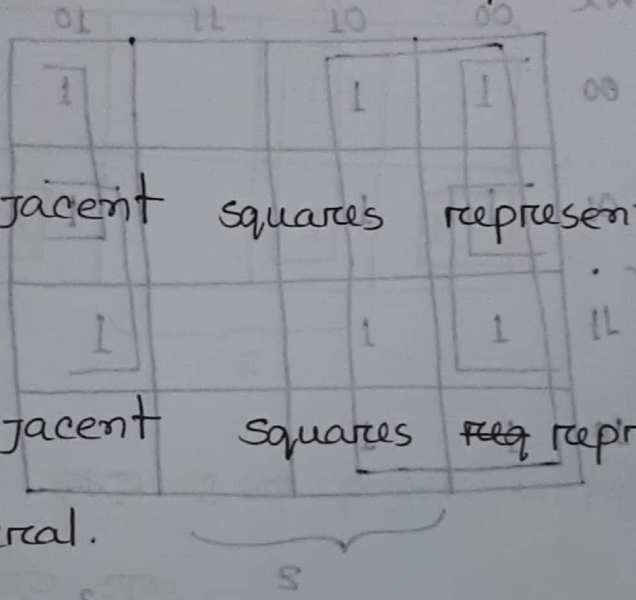
Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Four-variable map এর কাজ -

- One square represents one minterm, giving a term of four literals.
- Two adjacent squares represent a term of three literals.
- Four adjacent squares represent a term of two literals.
- Eight adjacent squares represent a term of one literal.
- Sixteen adjacent squares represent the function equal to 1.



Plan Title :

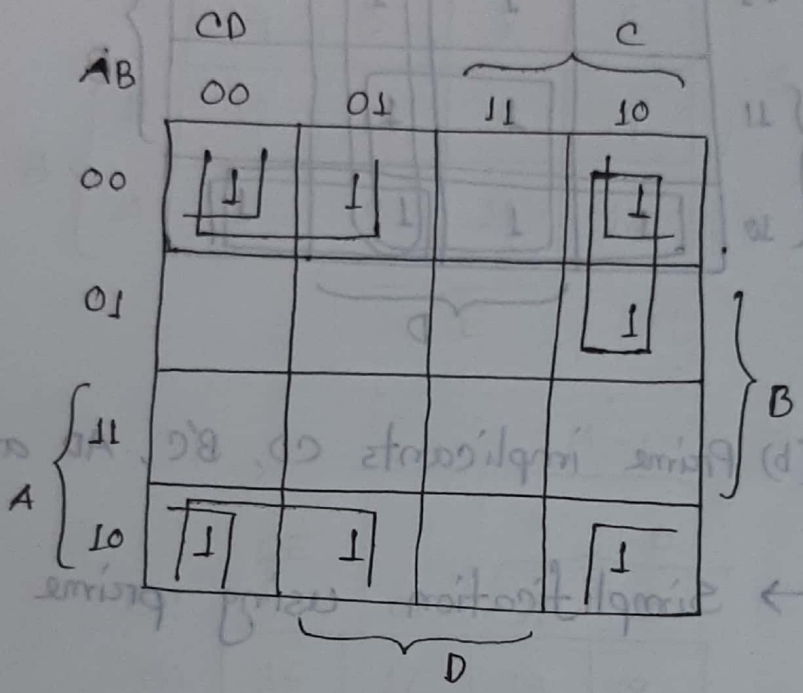
Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Example-3.6: Simplify the boolean function

$$F = A'B'C' + B'CD' + A'BCD' + AB'C'$$

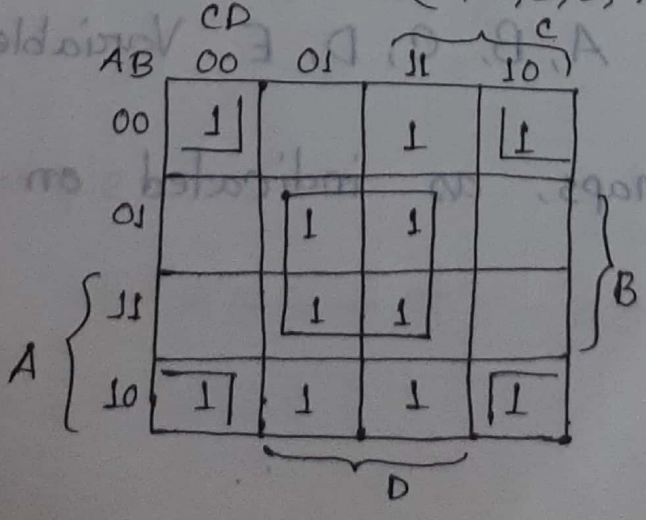
[4 digit ক নিতে হবে]



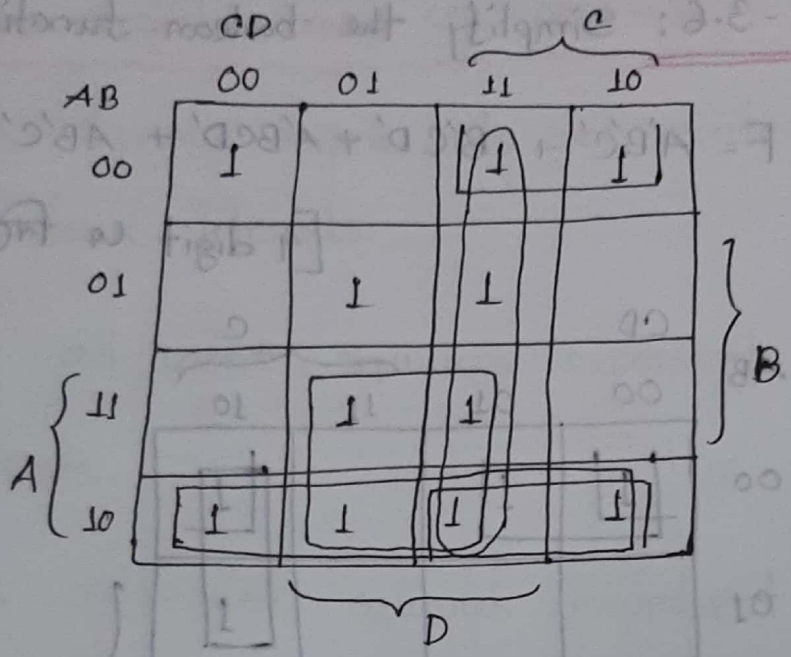
$$[A'B'C' + B'CD' + A'BCD' + AB'C' - B'D' + B'C' + A'CD']$$

* Considering the following boolean function:

$$F(A,B,C,D) = \Sigma(0,2,3,5,7,8,9,10,11,13,15)$$



(a) Essential Prime implicants BD and B'D

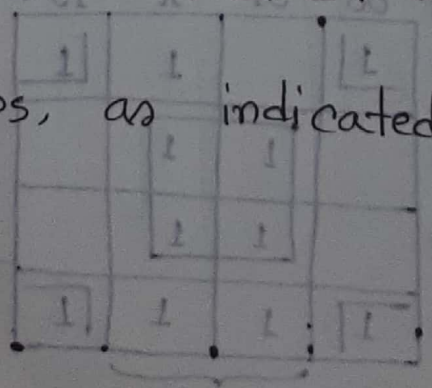


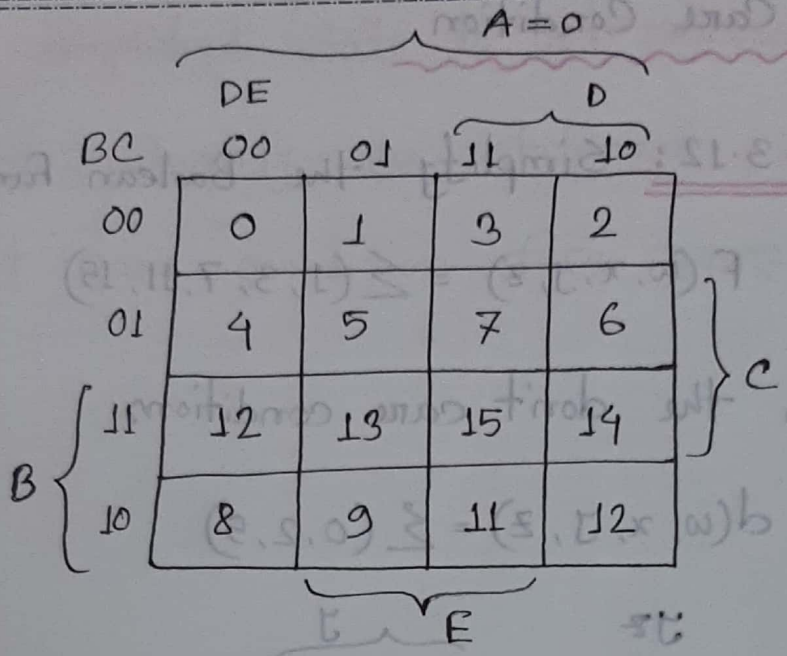
(b) Prime implicants CD, B'C, AD and AB'

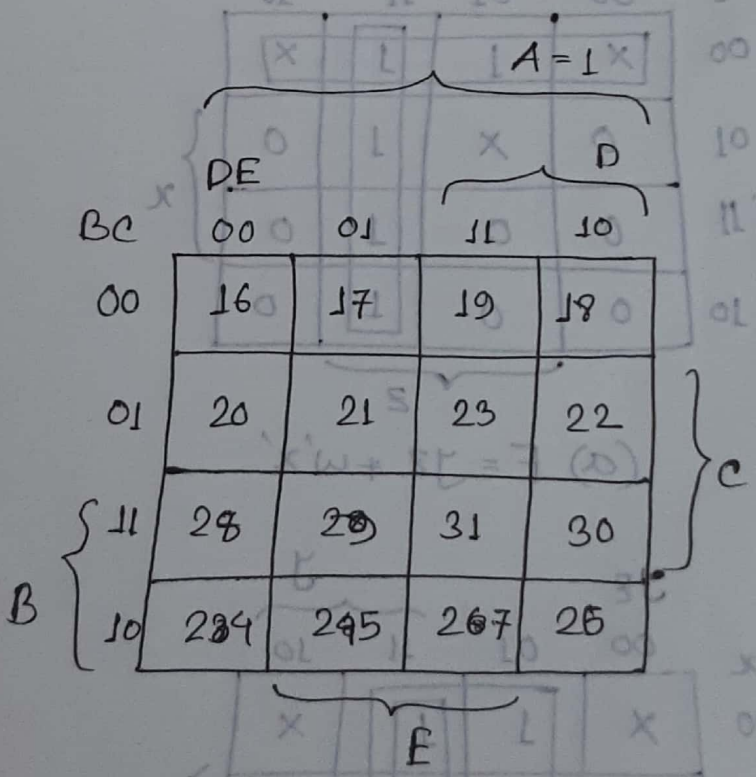
(a) + (b) → Simplification using prime implicants

* Five-Variable Map

The five variable map is shown in the following figure. It consists of two four-variable maps with variables A, B, C, D, E. Variable A distinguishes between two maps, as indicated on the top of the diagram.







* Example - 3.7

$F = A'B'E' + BD'E + ACE$ (Am)

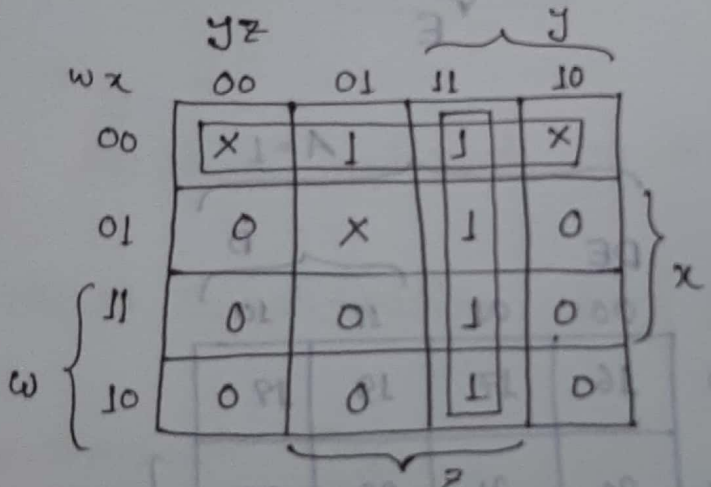
* Don't Care Condition

Example - 3.12: Simplify the Boolean function

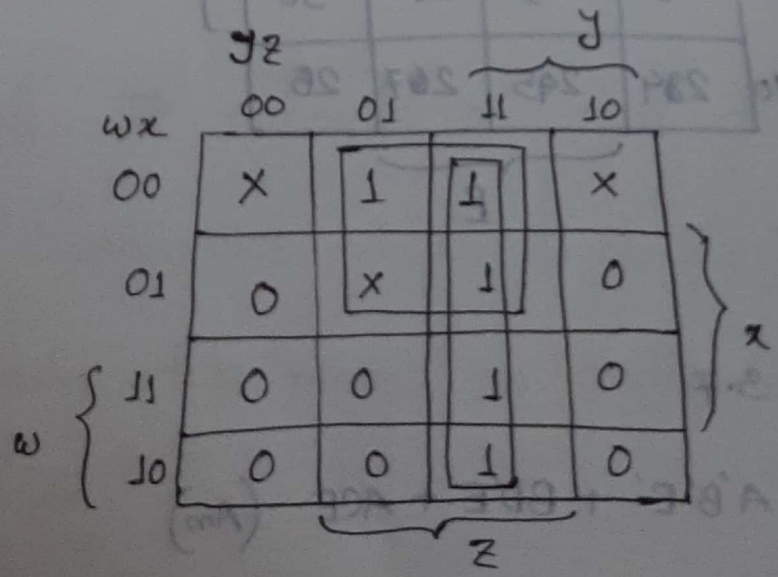
$$F(w,x,y,z) = \Sigma(1,3,7,11,15)$$

that has the don't care conditions

$$d(w,x,y,z) = \Sigma(0,2,5)$$



(a) $F = yz + w'x'$



(b) $F = yz + w'z$

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* The simplified state is not necessary to be always same. – Justify the statement.

⇒ Don't care condition দিয়ে বুঝাতে দিলে হবে।

Plan Title : Online Class-05

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date : 23./09./2021

* Veitch diagram or Karnaugh map=

* Tabulation Method / Mc-Clusky Method

Mc-Clusky Method (Tabular Form)

$f(a, b, c, d) = \Sigma (0, 5, 8, 9, 10, 11, 14, 15)$

Step-1

Group	Minterm	Variable
		A B C D
0	0 ✓	0 0 0 0
1	8 ✓	1 0 0 0
2	5	0 1 0 1
	9 ✓	1 0 0 1
	10 ✓	1 0 1 0
3	11 ✓	1 0 1 1
	14 ✓	1 1 1 0
4	15 ✓	1 1 1 1

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

Step = 2

Group	Matched Pair	Variable A B C D
0	0, 8	- 0 0 0
1	8, 9 ✓	1 0 0 -
	8, 10 ✓	1 0 - 0
2	9, 11 ✓	1 0 - 1
	10, 11 ✓	1 0 1 -
	10, 14 ✓	1 - 1 0
3	11, 15 ✓	1 - 1 1
	14, 15 ✓	1 1 1 -

Step = 3

Group	Matched Pair	Variable A B C D	
0	8, 9, 10, 11	1 0 - -	AB
	8, 10, 9, 11	1 0 - -	
1	10, 11, 14, 15	1 - 1 -	AC
	10, 14, 11, 15	1 - 1 -	
	0, 8	- 0 0 0	
		0 1 0 1	

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

Step = 4

P.I	0	5	8	9	10	11	14	15
8, 9, 10, 11			×	(×)	×	×		
10, 11, 14, 15					×	×	(×)	(×)
0, 8	(×)		×					
5		(×)						

$= AB' + AC + B'C'D' + A'BC'D$

* Example - 3.13, 3.14

** Tabular form, Karnaugh Map **

*****

- Complexity of K-Map increases with the increase in the number of variables.
- Tabulation method ensures to produce a simplified standard form (SoP = Sum of Product or PoS = Product of Sum) expression for a function.
- Suitable for machine computation
- First formulated by Quine and later improved by McClusky
- It consists of two parts
  - Determination of prime implicants
  - Selection of prime implicants - determination of essential prime implicants.

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

MAHBUB SIR

Chapter-04

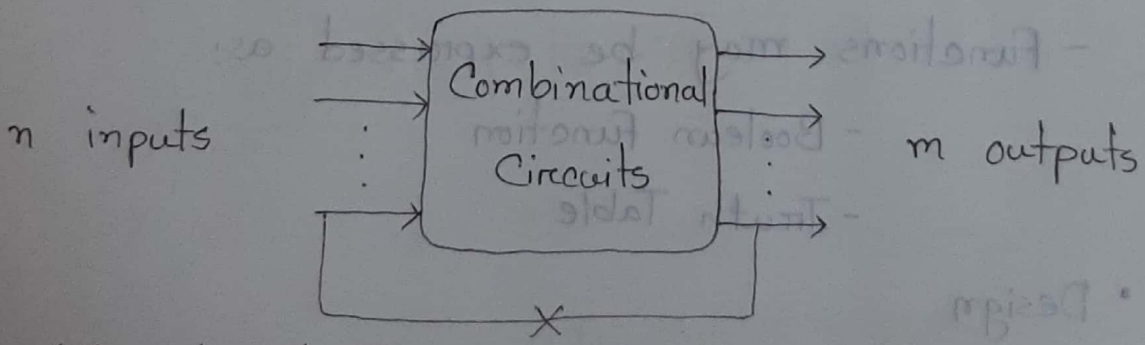
Combinational Logic

* Digital Logic Circuit two types

  I) Combinational logic

  II) Sequential logic

* Combinational Circuit: Output is function of input only. That means output is only dependent on input of current state. (i.e. no feedback)



When input changes, output may change (after a delay)

*কোনো feedback connection থাকবে না

* feedback connection আছে এরূপ state কে যে connection দিয়ে হচ্ছে সেইটার current state কে যখন input দেওয়া হবে সেই input এর সাথে ফিডব্যাক যোগ হয়।

* Sequential Circuits: If output of current state is dependent upon the input of present state and output of previous state, that is called sequential circuit.

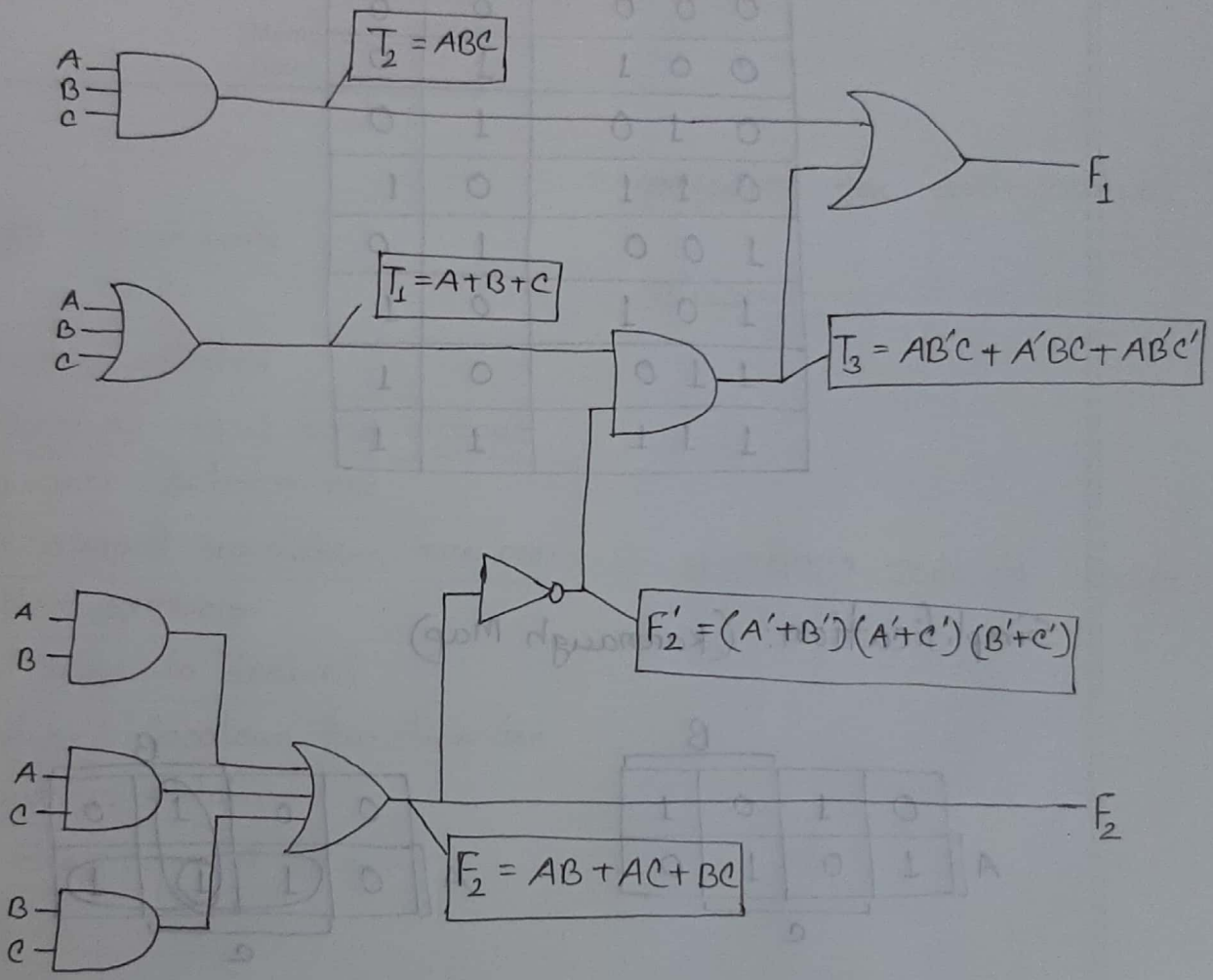
*** What is Combinational Circuit?
*** What is Sequential Circuit?
*** What are the difference between combinational and sequential circuits?

* Combinational Circuits

- Analysis
  - Given a circuit, find out its function
  - Functions may be expressed as:
    - Boolean function
    - Truth Table
- Design
  - Gived a desired function, determine its circuit
  - Function may be expressed as:
    - Boolean Function
    - Truth Table

* Analysis Procedure

- Boolean Expression Approach



$$F_1 = AB'C' + A'BC' + A'B'C + ABC$$

$$F_2 = AB + AC + BC$$

Plan Title :

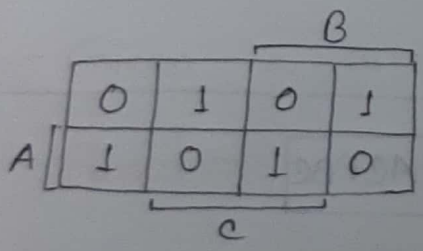
Sat / Sun / Mon / Tue / Wed / Thu / Fri

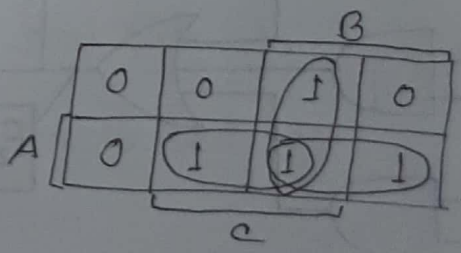
Date :...../...../.....

- Truth Table Approach

A	B	C	F_1	F_2
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

Simplification: (Karnaugh Map)





$F_1 = AB'C' + A'BC' + A'B'C + ABC$

$F_2 = AB + AC + BC$

* 137 Page – Analysis Procedure

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Design Procedure

• Given a problem statement:

of inputs and outputs

pression for each output

circuit

- a "BCD" code to "Exces

Mo Tu We Th Fr Sa Su

Memo No. _____

Date / /

Design Procedure

1. The Problem is stated
2. The number of input and output variables are determined
3. The input output variables are assigned letter symbols.
4. The truth table is derived
5. The simplified boolean function for each output is obtained.
6. The logic diagram is drawn.

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Design Procedure

- Given a problem statement:
  - Determine the number of inputs and outputs
  - Derive the truth table
  - Simplify the Boolean expression for each output
  - Produce the required circuit

Example:

Design a circuit to convert a "BCD" code to "Excess 3" code. (দেখানো হয় নাই)

* Binary Adder

- Half Adder
  - Adds 1-bit plus 1-bit
  - Produces Sum and Carry

$$\begin{array}{r} x \\ +\ y \\ \hline C\ S \end{array}$$

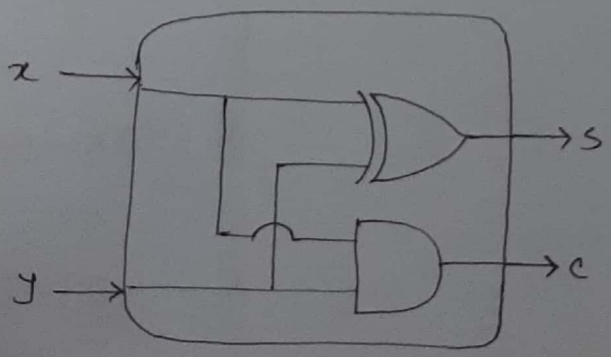
x y	C S
0 0	0 0
0 1	0 1
1 0	0 1
1 1	1 0

S = sum of minterm

$S = x'y + xy'$

$= x \oplus y$ [exclusive or]

$C = xy$



* যে কয়টি output variable হবে সেকয়টি Boolean function হবে।

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Simplify কেন করতে হয়?

⇒ Circuit এর size ছোট করা হলে Gate এর সংখ্যা কম লাগে, Gate এর সংখ্যা কম লাগলে Current এর পরিমান কম লাগে (বিদ্যুতের পরিমান), কম heat উৎপন্ন হবে, কম জায়গা লাগবে device design করার ক্ষেত্রে। Cost কম লাগবে

* Adders

Adder Circuit 2 types

1) Half-Adder (Addition of two bits)

11) Full-Adder (Addition of three bits)

* Half-Adder

x	y	c	s
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

$S = x'y + xy'$

$C = xy$

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

$C = xy$

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

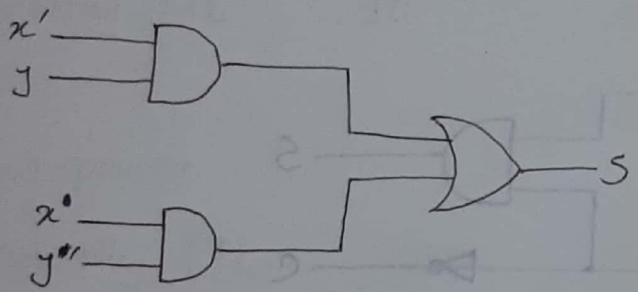
Date :...../...../.....

$S' = xy + x'y'$

$S' = C + x'y' \quad [C = xy]$

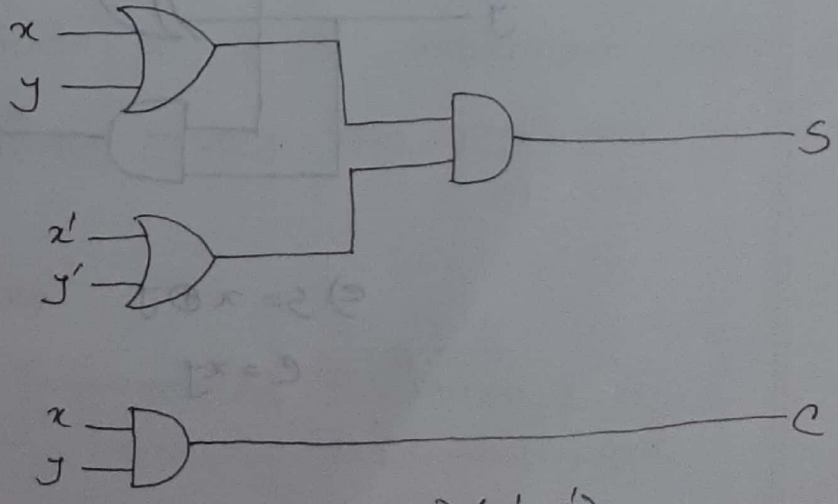
$S' = (C + x'y')'$

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



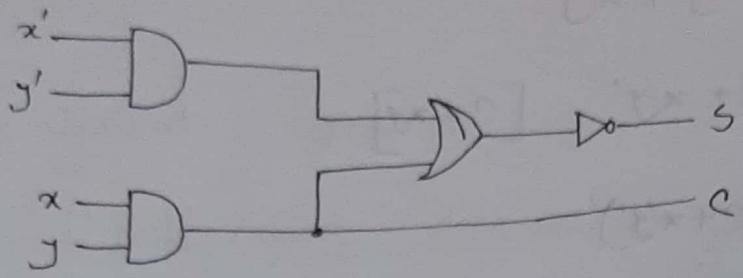
a) $S = x'y + xy'$

$C = xy$

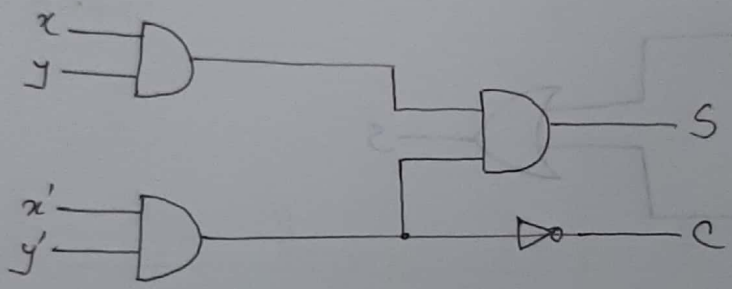


b) $S = (x+y)(x'+y')$

$C = xy$

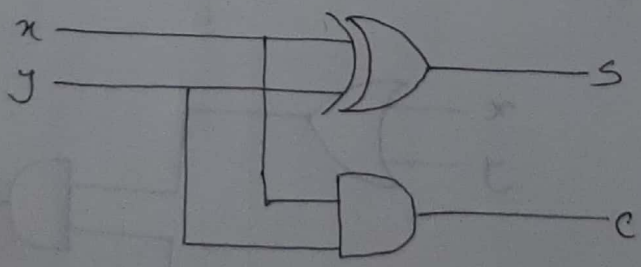


c) $S = (C + x'y')'$



d) $S = (x+y)(x'+y')$

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



e) $S = x \oplus y$

$C = xy$

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Table 2.8 Typical characteristics of IC logic families

IC Logic family	Fan-out	Power Dissipation (mW)	Propagation delay (ns)	Noise Margin (V)
Standard TTL	10	10	10	0.4
Schottky TTL	10	22	03	0.4
Low-power Schottky TTL	20	02	10	0.4
ECL	25	25	02	0.2
CMOS	50	0.1	25	03

Fan-out : একটা Gate এর output maximum কয়টা Gate এর input হিসেবে ব্যবহার করা যাবে সেটা হলো Fan-out

Propagation delay **

I^2L - Integrated Injection Logic

TTL - Transistor-Transistor Logic

ECL - Emitter Coupled Logic

CMOS - Complimentary Metal Oxide Semiconductor

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Full-Adder

3 bit binary if information যোগ করার জন্য একটা 0 যুক্ত দিতে হয়

$$\begin{array}{r} 0 \\ 1\ 1 \\ 1\ 1 \\ \hline 1\ 1\ 0 \end{array}$$

[2 bit , 1 bit carry]

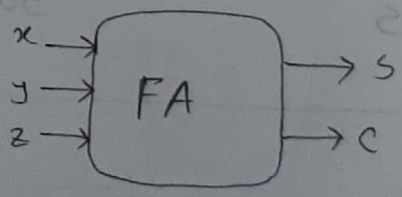
input 3 টি

output 2 টি (S, C)

- Adds 1-bit plus 1-bit plus 1-bit
- Produces sum and carry

Truth Table

x	y	z	C	S
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



$$\begin{array}{r} x \\ +\ y \\ +\ z \\ \hline C\ \ S \end{array}$$

Plan Title :

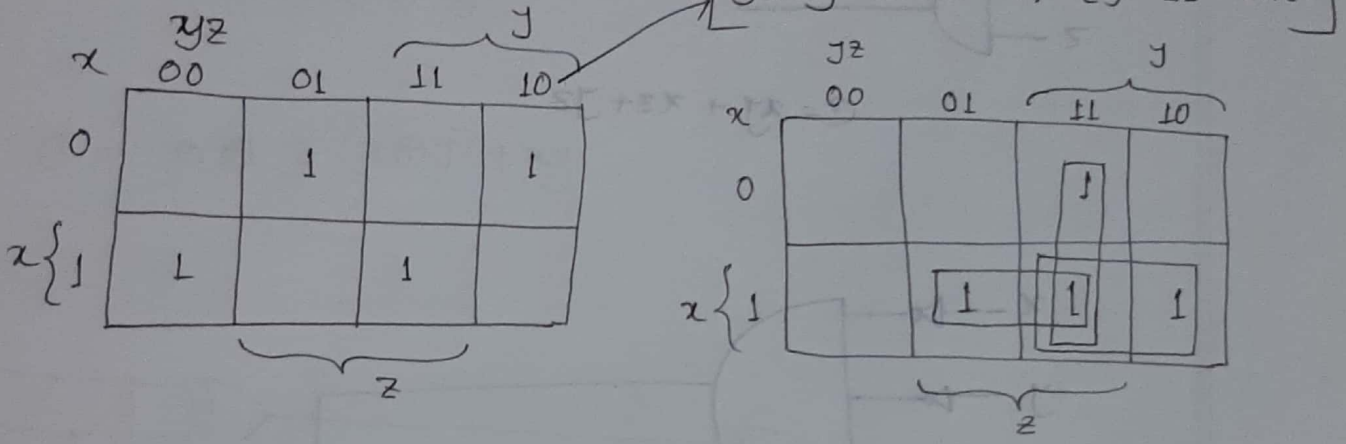
Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

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

$= x \oplus y \oplus z$

$C = xy + yz + xz$



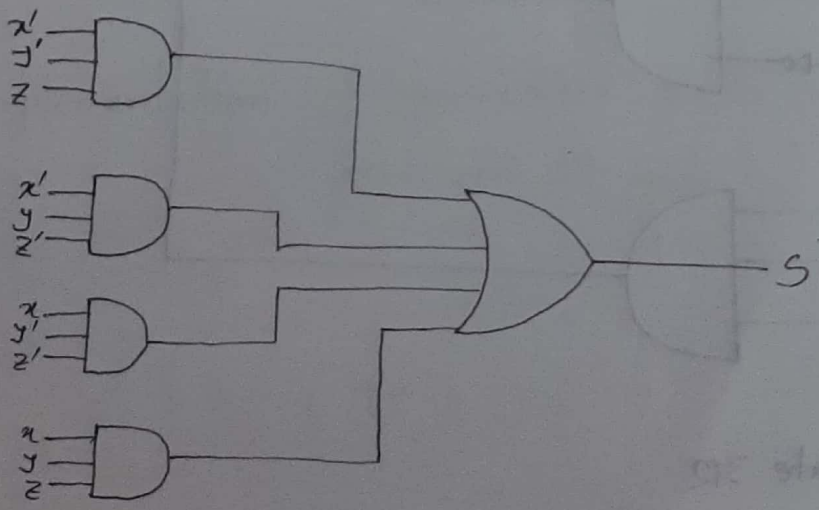
$S = x'y'z + x'yz' + xy'z' + xyz$

(Simplified করা যাবে না)

$C = xy + xz + yz$

(Simplified করা যাবে)

Fig: Karnaugh Maps for Full Adder

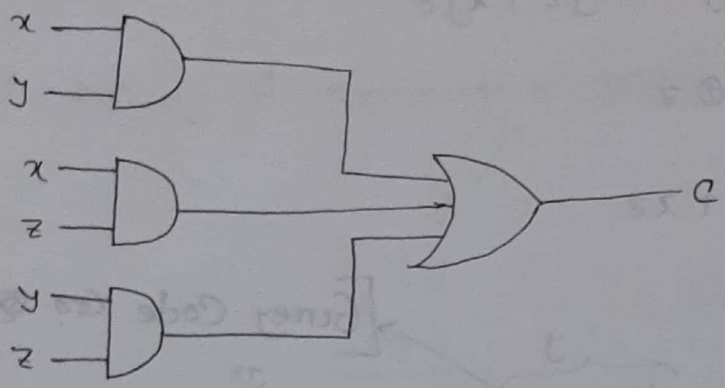


$S = x'y'z + x'yz' + xy'z' + xyz$

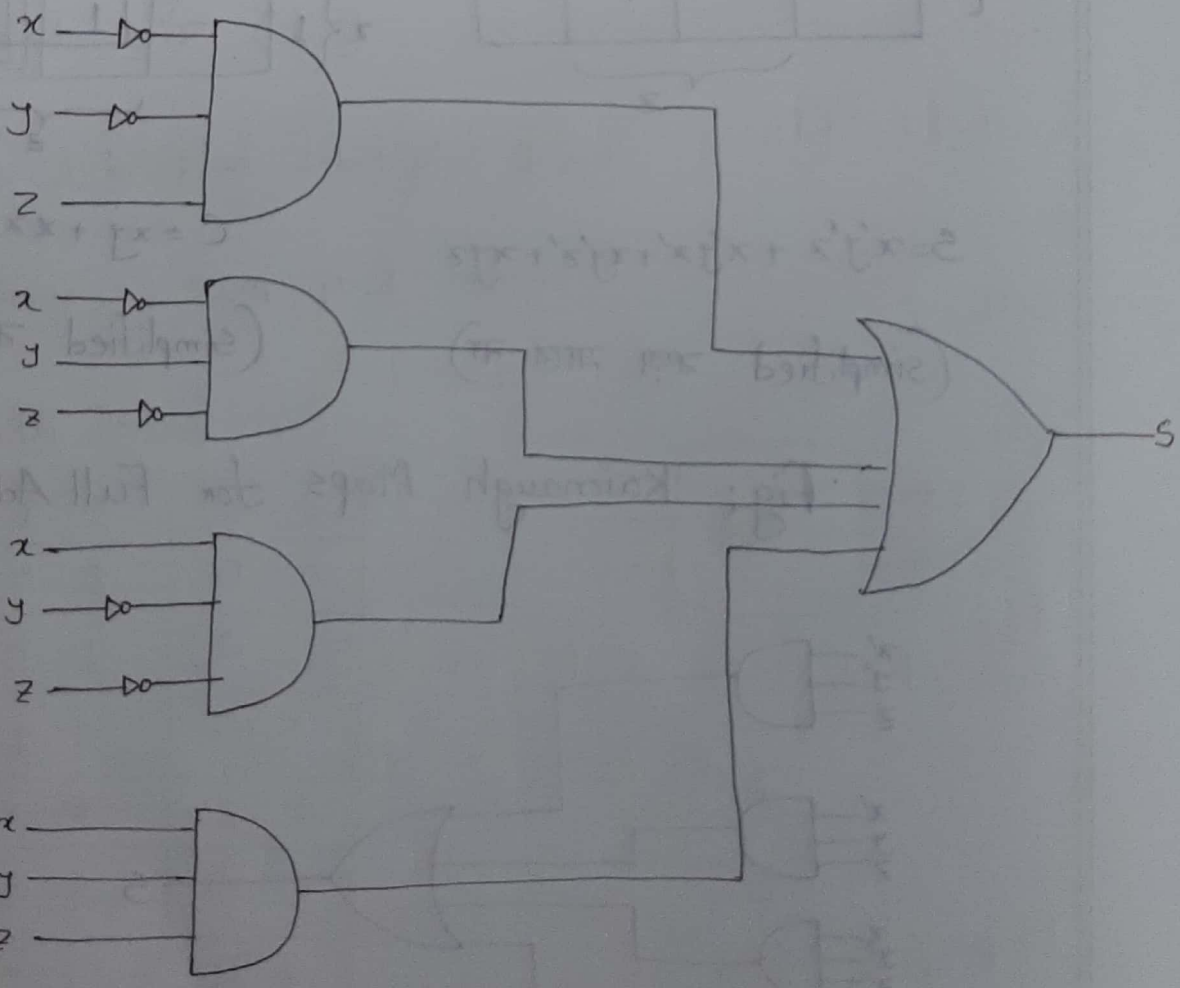
Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....



$C = xy + xz + yz$



**NOT Gate 3I2

Fig: Implementation of full-adder in sum of products

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

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

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

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

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

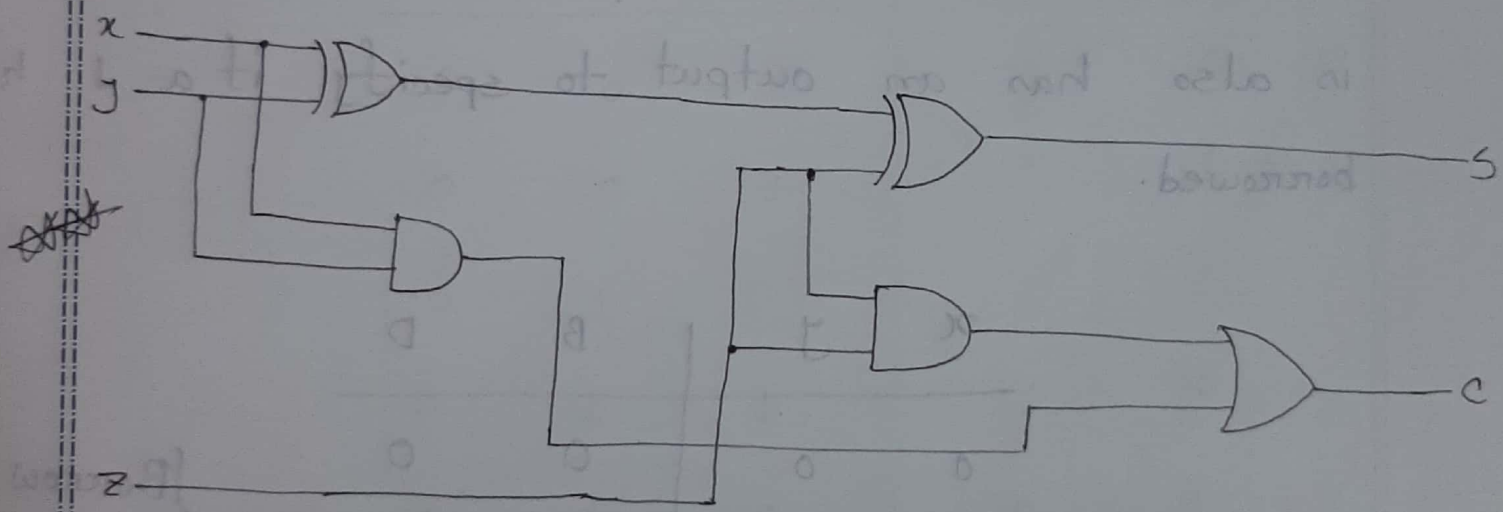


fig: Implementation of full-adder with two half-adders and an OR gate

* Subtractors

The subtraction of two binary numbers may be accomplished by taking the complement of the subtrahend and adding it to the minued.

* Half - Subtractor

A half subtractore is a combinational circuit that subtracts two bits and produces their difference. It is also has an output to specify if a 1 has been borrowed.

x	y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

{ Borrow
{ Difference

$$D = x'y + xy' = x \oplus y$$

$$B = x'y$$

*** Circuit design করতে হবে

Plan Title :

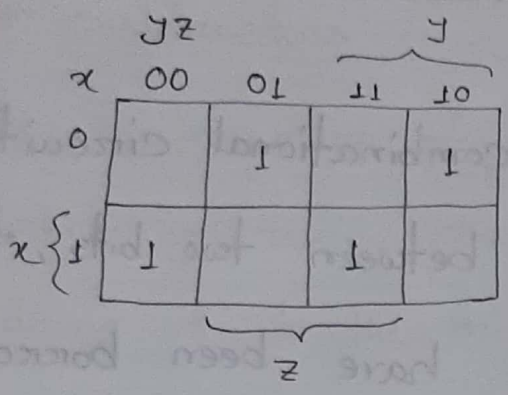
Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

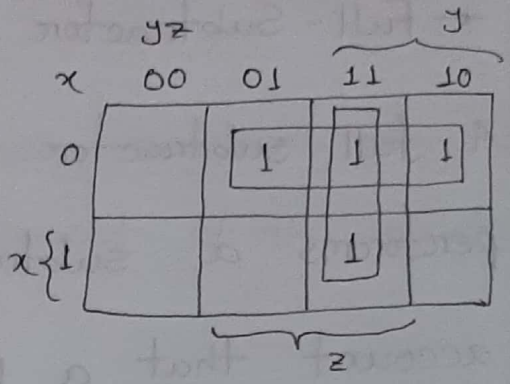
* Full - Subtractor

A full subtractor is a combinational circuit that performs a subtraction between two bits, taking into account that a 1 may have been borrowed by a lower significant stage.

x	y	z	B	D
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	1	0
1	0	0	0	1
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1



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



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

fig: Maps for Full-subtractor

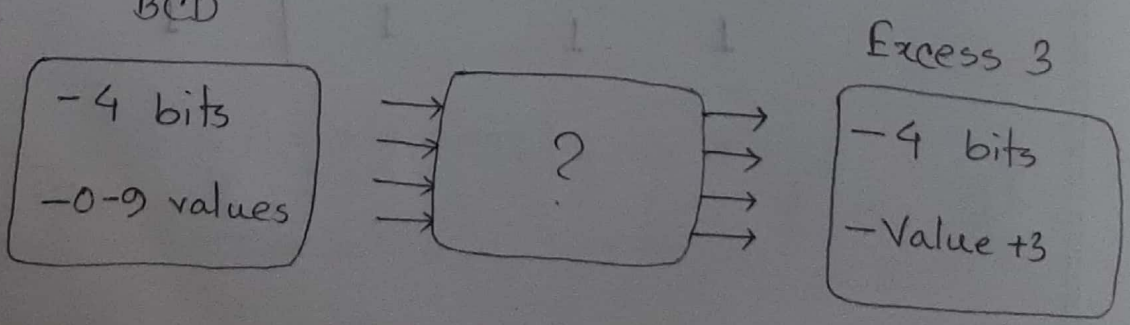
*** Circuit draw করতে হবে।

*** Boolean Algebra use করে further simplify

* Code Conversion

এক ধরনের circuit যেটা এক system এর code কে অন্য system এ convert করে। (Compatibility ও বলা হয়)

Design a circuit to convert a 'BCD' code to 'Excess 3'



Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* BCD - to - Excess 3 Converter

A	B	C	D	W	x	J	z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0
1	0	1	0	x	x	x	x
1	0	1	1	*	x	x	x
1	1	0	0	x	x	x	x
1	1	0	1	x	x	x	x
1	1	1	0	x	x	x	x
1	1	1	1	x	x	x	x

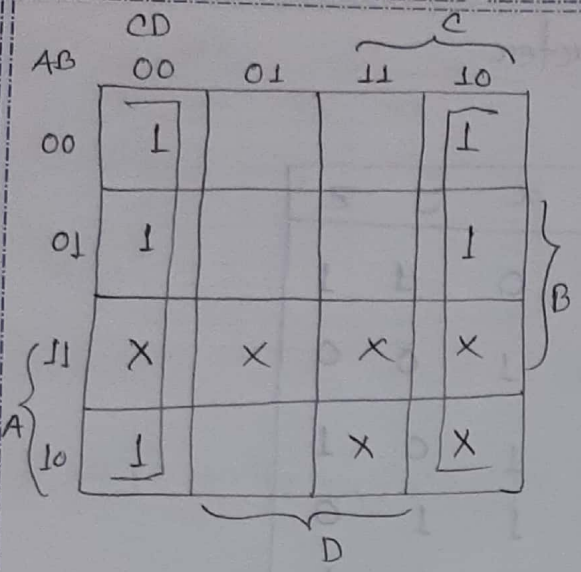
} Don't Care

*** Full equation গুলা লিখতে হবে।

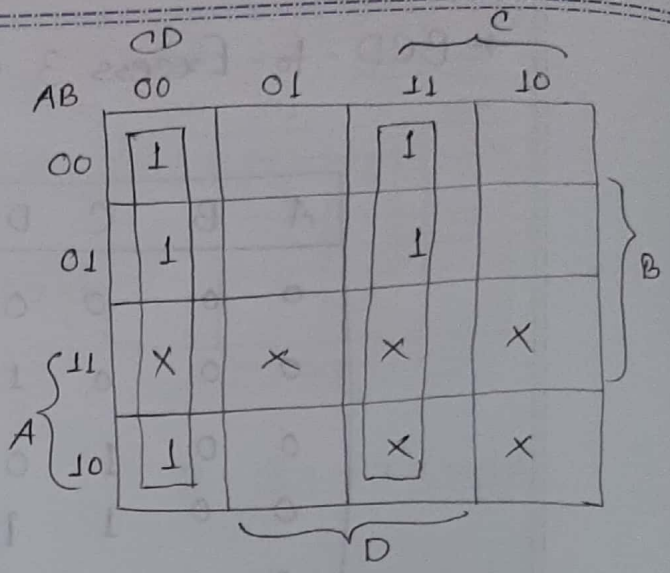
Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

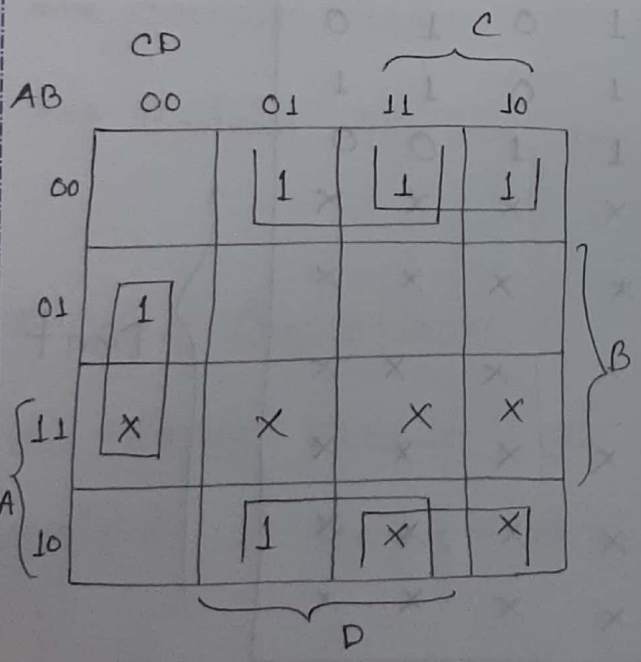
Date :...../...../.....



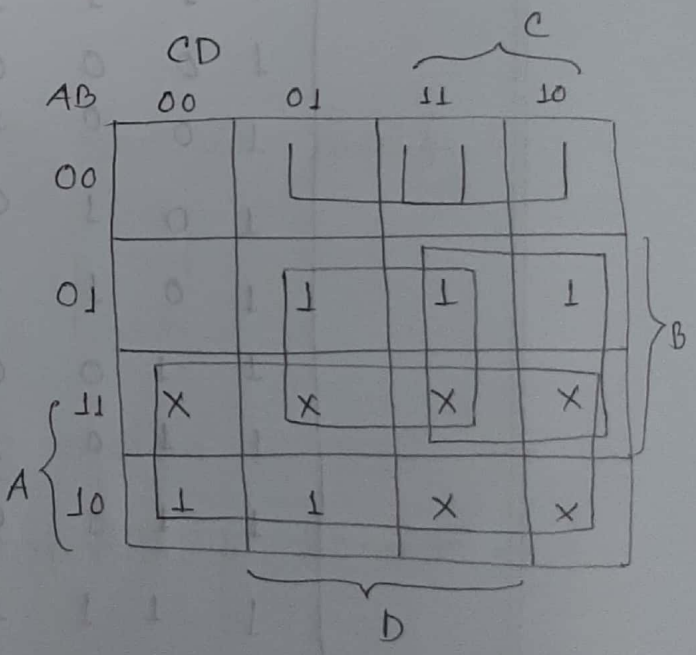
$z = D'$



$y = CD + C'D'$



$x = B'C + B'D + BC'D'$



$w = A + BC + BD$

Fig: MAPs for BCD to Excess-3 converter

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

*** Don't care গুলো কে 1 ধরতে হবে তবে; যত বেশি সংখ্যক Don't care নেয়া যায়।

*** 1 গুলো কে cover করতে গিয়ে যদি কোনো Don't care আশে পড়ে যায় তাহলে সেগুলো কে 1 consider করতে হবে।

$Z = D'$

$J = CD + C'D' = CD + (C+D)$

$x = B'C + B'D + BC'D'$

$= B'(C+D) + BC'D'$

$= B'(C+D) + B(C+D)'$

$\omega = A + BC + BD$

$= A + B(C+D)$

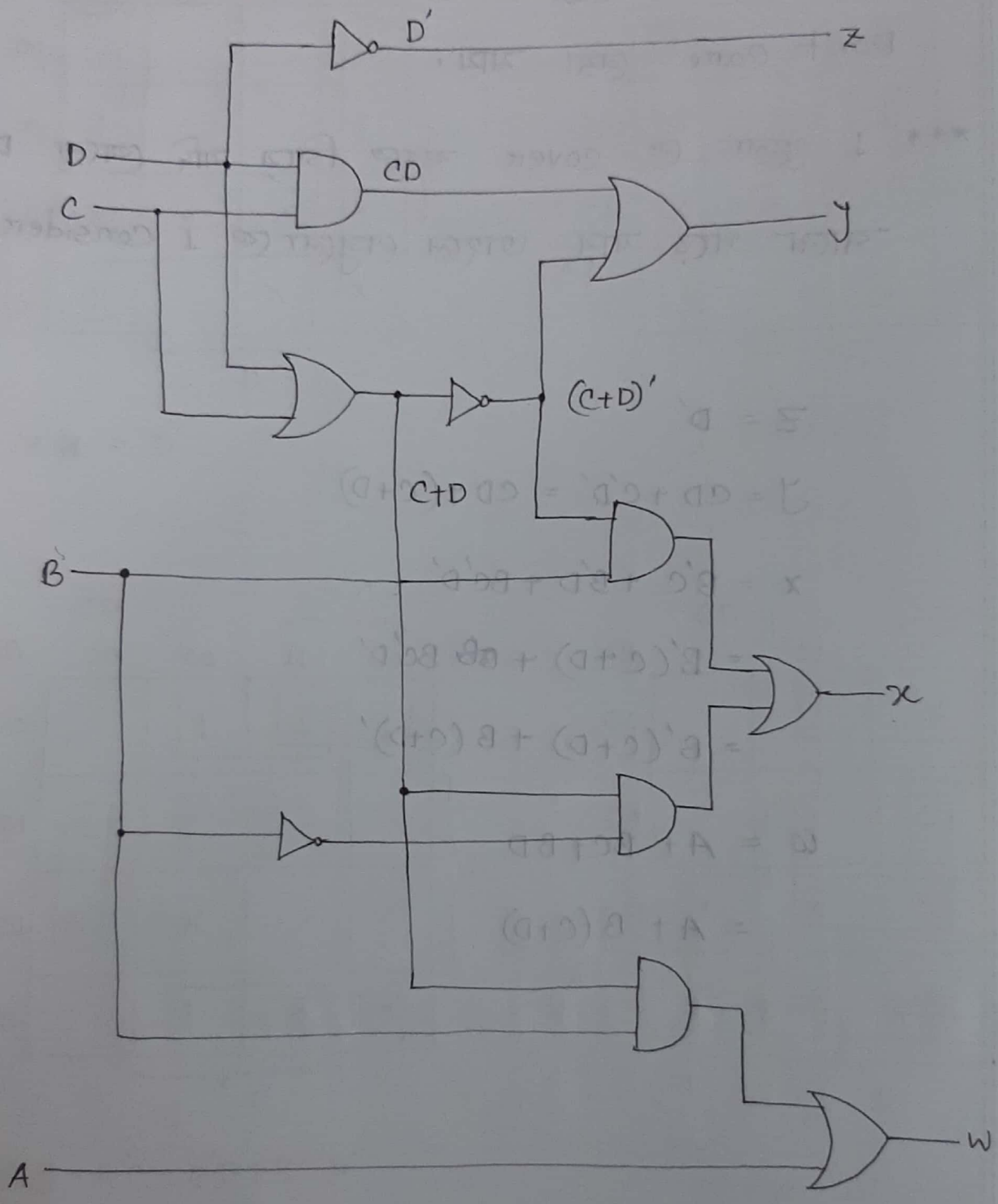


Fig: Logic diagram for BCD to Excess-3 code converter

Plan Title : Online Class – 04

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date : 30 / 05 / 2021

* Multilevel NAND Circuits

AND এবং OR এর compliment হলো NAND এবং NOR

* Universal Gate

NAND and NOR gates are universal gate. এই দুইটা gate এর যেকোনো combination দিয়ে যেকোনো basic gate এর functionalities অর্জন করা যায়.

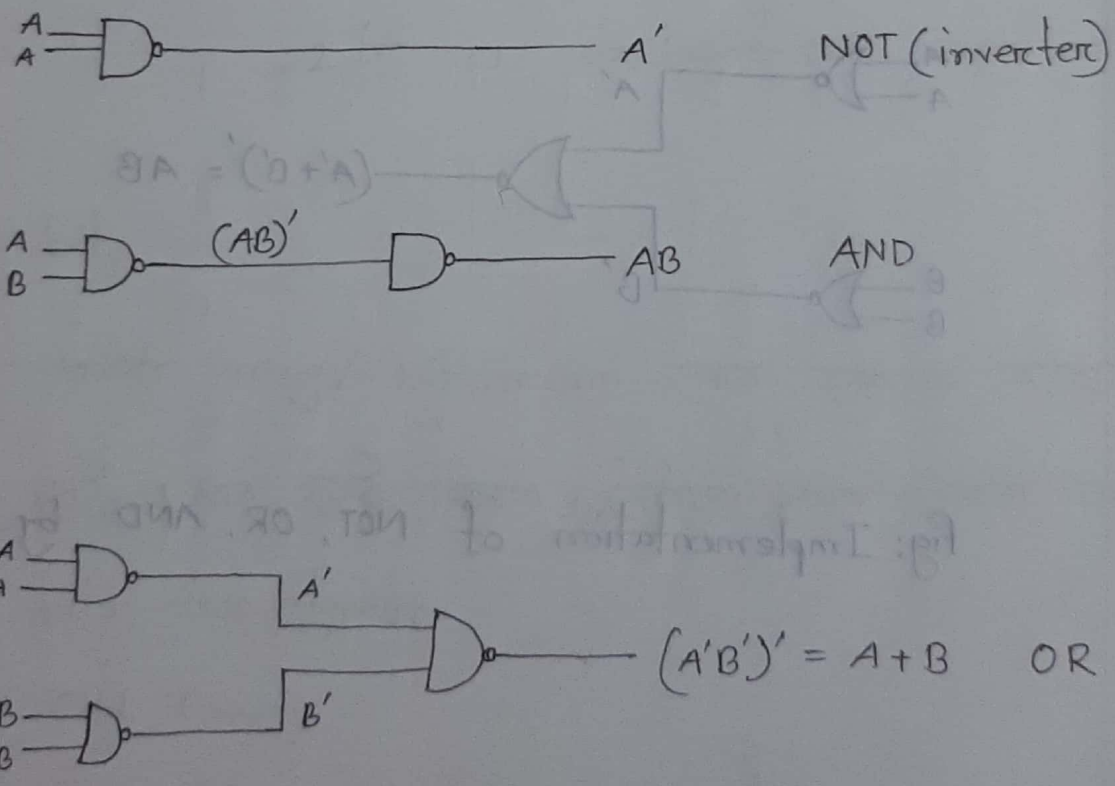


Fig: Implementation of NOT, AND, OR by NAND gates.

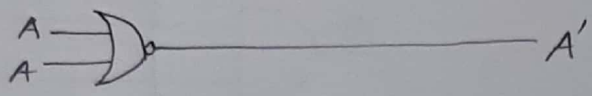
Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

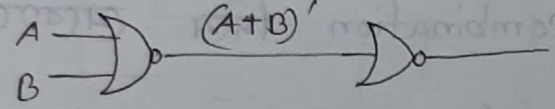
Date :...../...../.....

*** खुई साथ NAND gate दिए Half-adder circuit draw.

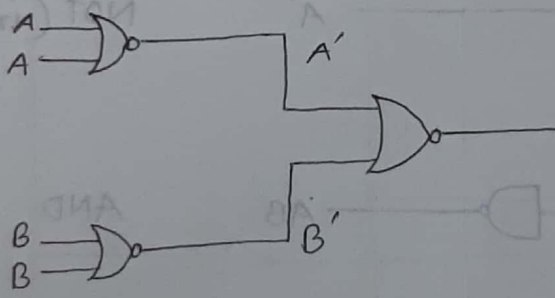
* Multilevel NOR circuits



NOT (inverter)



OR



AND

fig: Implementation of NOT, OR, AND by NOR gates

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Exclusive-OR (⊕) and Equivalence function (⊙)

x	y	F
0	0	0
0	1	1
1	0	1
1	1	0

$x \oplus y = x'y + xy'$

$x \odot y = xy + x'y'$

= 2 টা 1 exclusive-OR represent করে

= 2 টা 0 exclusive-NOR represent করে

Parity bit:

কোনো একটা msg/ bitstream থেকে সেখানে কতগুলো 1 আছে সেই 1 এর যেই হিসাব সেটাকে parity বলা হয়।

Parity bits two types:

1. Odd Parity
2. Even Parity

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

Odd Parity Generation :

Three-bit message			Parity bit generated
x	y	z	P
0	0	0	1
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	0

যদি message এ 1 এর সংখ্যা যে Odd হয় তাহলে parity 0 আর যদি 1 এর সংখ্যা even হয় তাহলে Parity 1.

$$P = x \oplus y \odot z$$

*** Boolean equation solve করতে হবে তারপর সেটা draw করতে হবে.

*** four bit Parity generator.

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

Tasks.

Memo No. _____

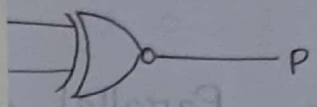
Date / /

1) Odd Parity check truth table থেকে boolean function বের করে $C = x \odot y \odot z \odot P$ prove করতে হবে।

2) Universal gate দিয়ে অন্য gate implementation.

3) Half adder, Full adder implementation using NAND & NOR gates.

4) Half subtractor, Full subtractor implementation using NAND & NOR gates.



ity generator

ity checker

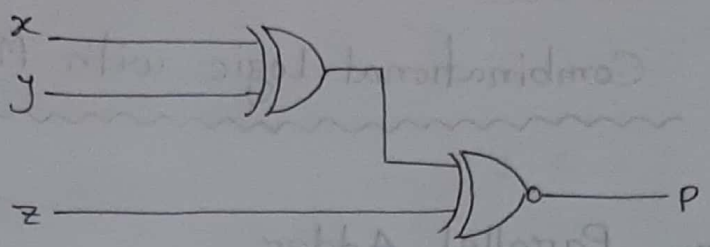


fig: 3-bit odd parity generatore

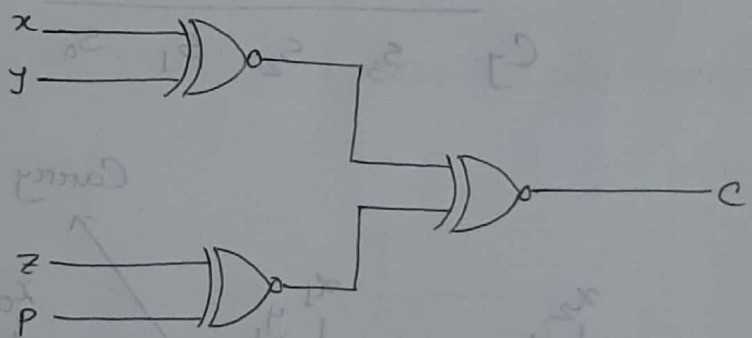


fig: 4-bit odd parity checker

$P = x \odot y \odot z$

$C = x \odot y \odot z \odot P$

Chapter - 05

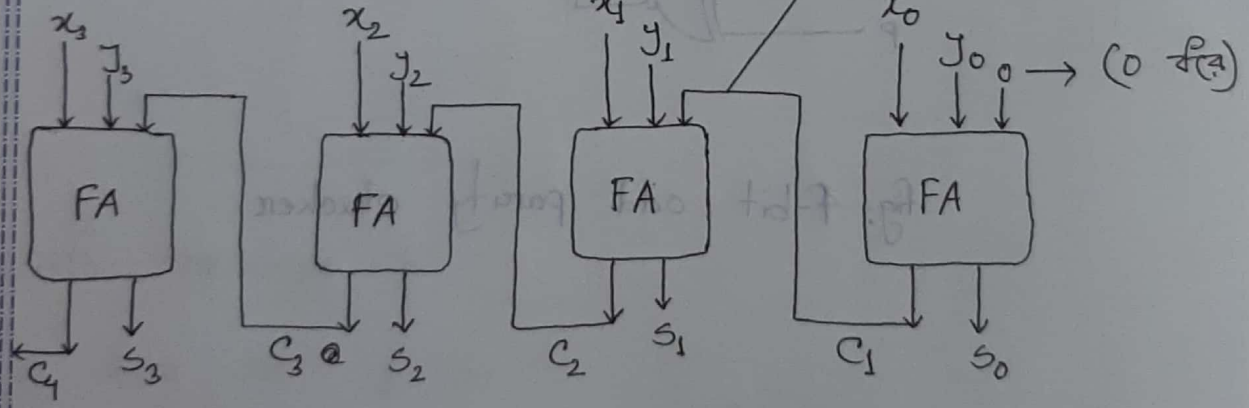
Combinational Logic with MSI & LSI

* Binary Parallel Adder

$$\begin{array}{cccccc} & C_3 & C_2 & C_1 & & \\ + & X_3 & X_2 & X_1 & X_0 & \\ + & Y_3 & Y_2 & Y_1 & Y_0 & \\ \hline C_4 & S_3 & S_2 & S_1 & S_0 & \end{array}$$

$$\begin{array}{r} 1010 \\ 1100 \\ \hline 10110 \end{array}$$

Carry Propagate Addition



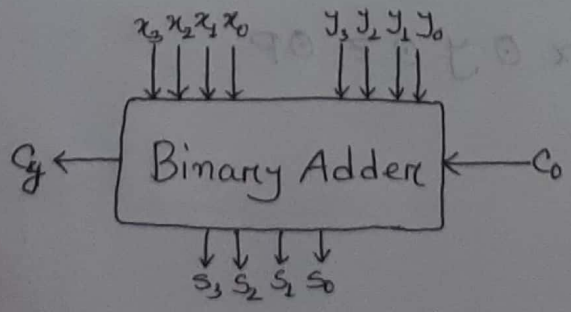


fig: 4-bit binary full adder block-diagram

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

Subscripti	4 3 2 1	
Input carry	0 1 1 0	C_i
साज्य ⟶ Augend	1 0 1 1	A_i
साजक ⟶ Addend	0 0 1 1	B_i
Sum	1 1 1 0	S_i
Output carry	0 0 1 1	C_{i+1}

** Designing a BCD-to-Excess-3 code converter using ~~those~~ 4-bit full adder.

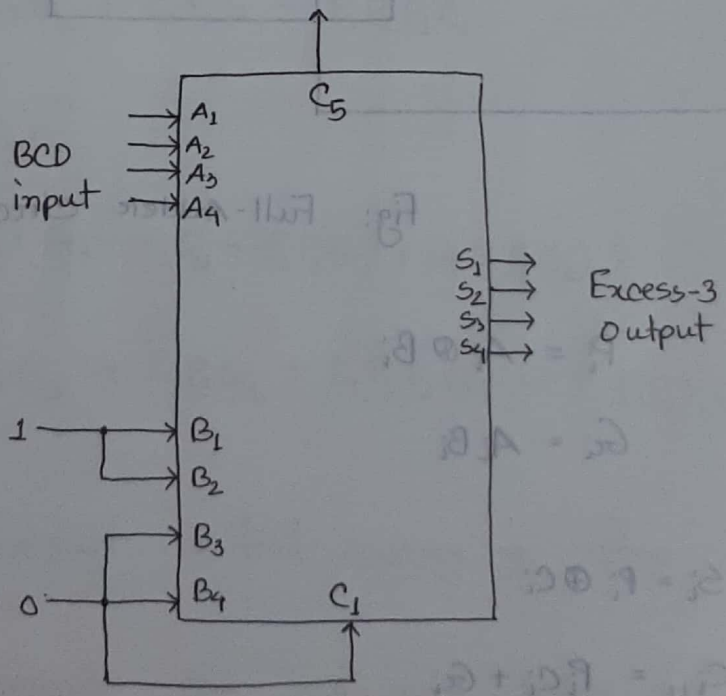


Fig: Block diagram

* Example 5.1

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* 4-bit Full adder slow কাজ করে।

→ Dependency কমানো গেলে circuit slow থাকবে না। সেক্ষেত্রে carry গুলো আলাদা generate করে একসাথে input দিলে parallelly একইসাথে সবগুলো generate হয়ে যাবে। তাহলে একটা circuit এর উপর আরেকটা circuit এর dependency কমে যাবে।

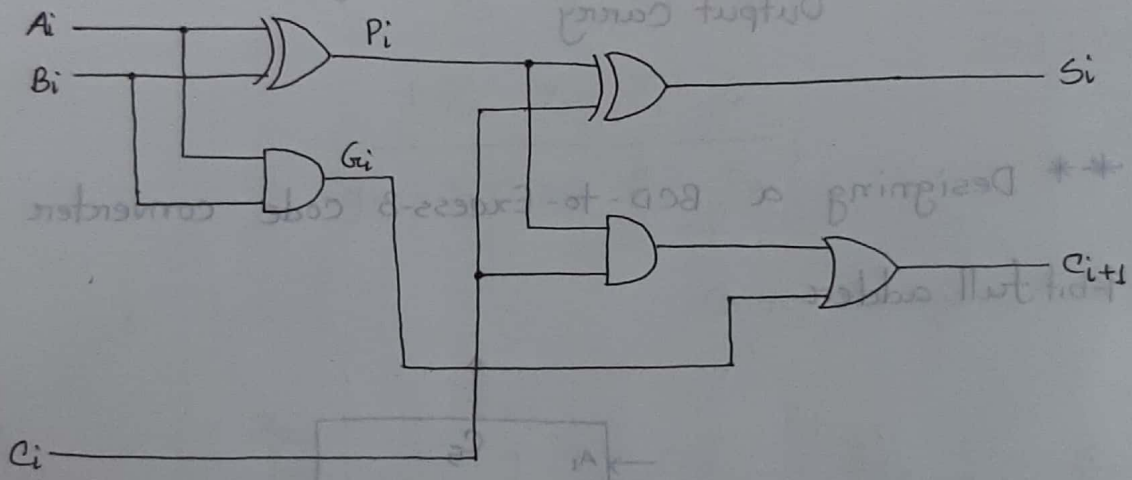


Fig: Full-Adder circuit

$$P_i = A_i \oplus B_i$$

$$G_i = A_i B_i$$

$$S_i = P_i \oplus C_i$$

$$C_{i+1} = P_i C_i + G_i$$

(i এর মান 0 থেকে শুরু করলে C_1 থেকে শুরু হবে)×

~~$C_1 = G_1 + P_1 C_1$~~

~~$C_2 = G_2 + P_2 C_2$~~

$C_2 = G_1 + P_1 C_1$

$C_3 = G_2 + P_2 C_2$

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

$= G_2 + P_2 G_1 + P_2 P_1 C_1$

$C_4 = G_3 + P_3 C_3$

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

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

$C_5 = G_4 + P_4 C_4$

$= G_4 + P_4 (G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 C_1)$

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

* এগুলো circuit initial carry-র উপর depend করে।

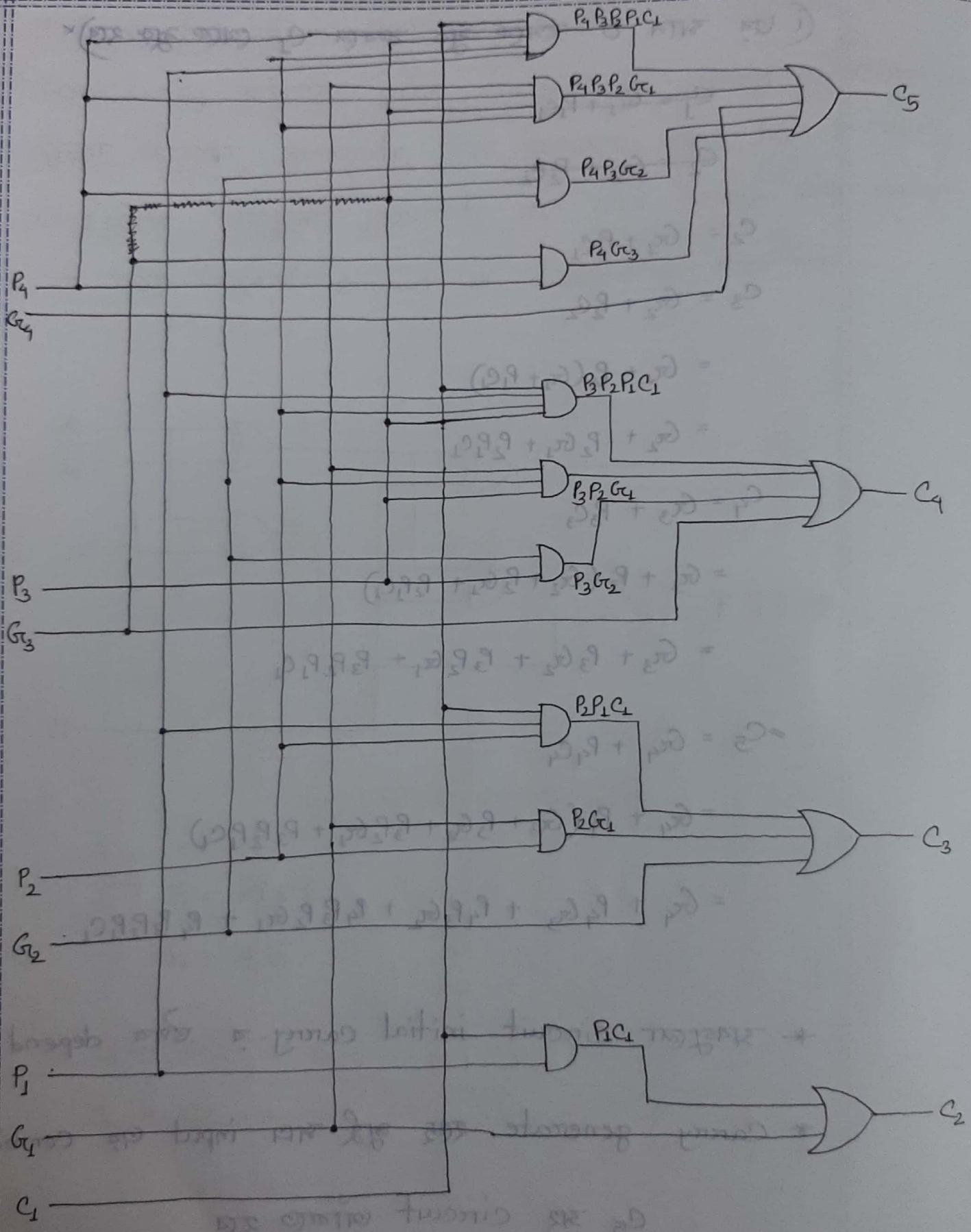
* Carry generate হচ্ছে শুধু আগ input এর combination এ।

C_5 এর circuit আঁকতে হবে

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....



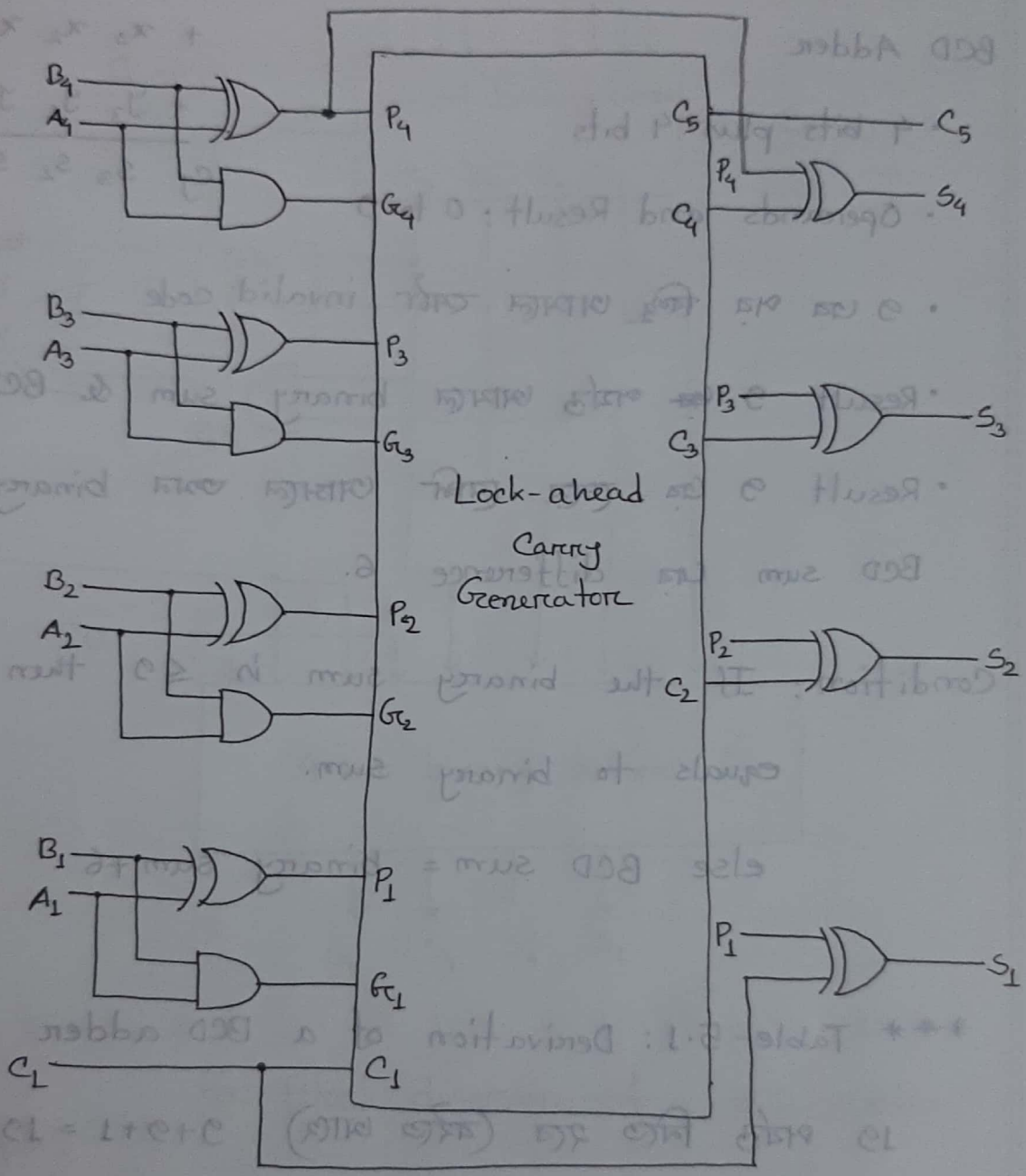


fig: 4-bit full-adders with look ahead carry

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Decimal Adder

BCD Adder

$$\begin{array}{r} + \; x_3\; x_2\; x_1\; x_0 \\ + \; y_3\; y_2\; y_1\; y_0 \\ \hline C_y\; S_3\; S_2\; S_1\; S_0 \end{array}$$

- 4 bits plus 4 bits
- Operands and Result : 0 to 9
- 9 এর পর কিছু আসলে সেটা invalid code
- Result 9 পর্যন্ত আসলে binary sum & BCD sum same
- Result 9 এর চেয়ে বেশি আসলে তখন binary sum আর BCD sum এর difference 6.

Condition : If the binary sum is ≤ 9 then BCD sum equals to binary sum.

else BCD sum = binary sum + 6

*** Table-5.1 : Derivation of a BCD adder

19 পর্যন্ত নিতে হবে (বহনে আছে) 9+9+1 = 19 [যদি previous Carry থাকে]

K এর মান যখন 1 হবে তখন 6 যোগ হবে।

$K + Z_8Z_4 + Z_8Z_2$ এর মান যখন 1 আসবে তখন 6 যোগ হবে।

$\therefore C = K + Z_8Z_4 + Z_8Z_2$ *Figure-5.6

otherwise BCD sum = binary sum

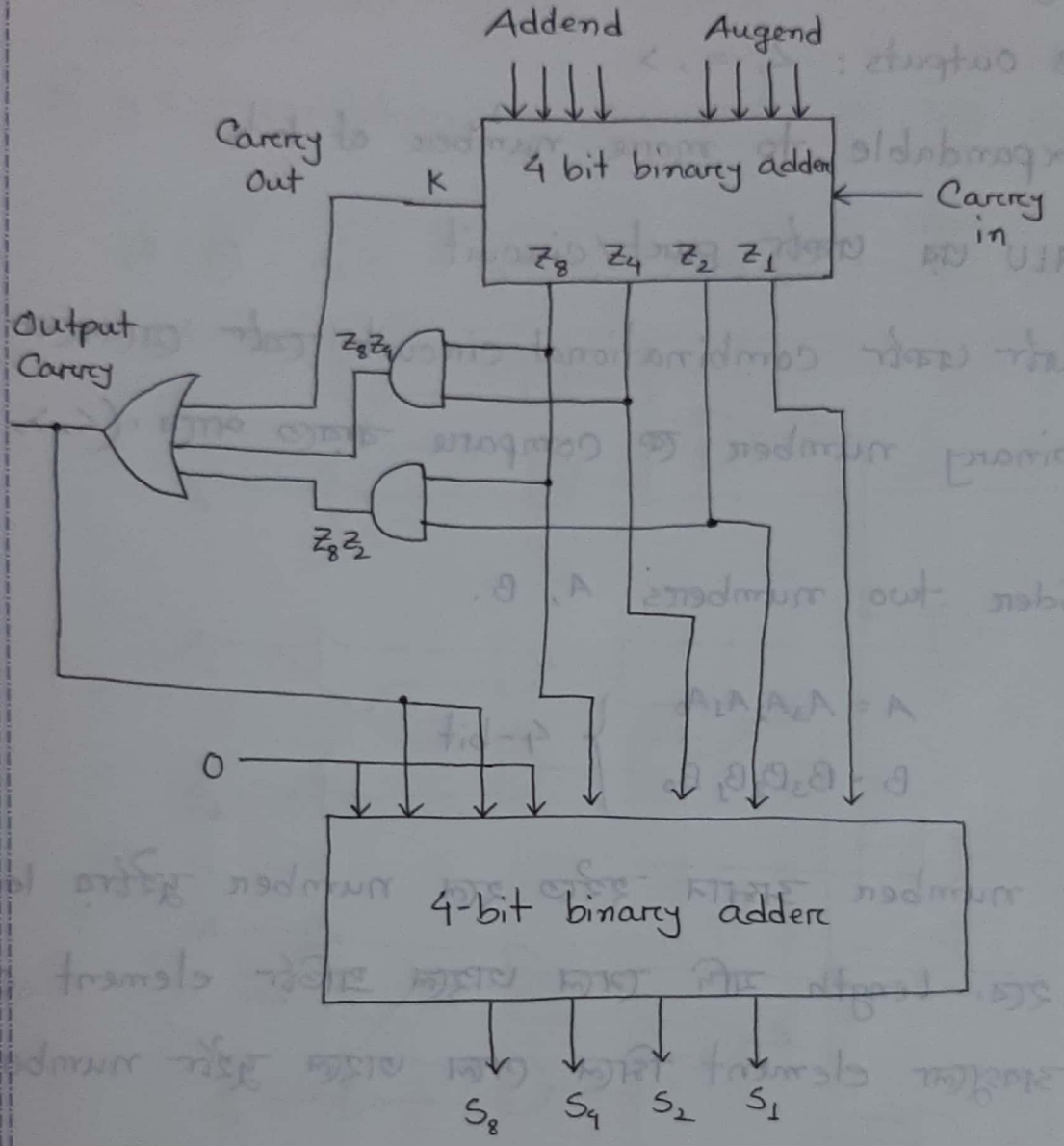


fig: Block diagram of a BCD adder

Problems 5.1 - 5.12

Example:

$$\begin{array}{r} 0111 \\ 1001 \\ \hline 10000 \end{array}$$

carry in = 0

binary sum = 0

binary carry = 1

9 + 7 = 16

1 6

* Magnitude Comparator

- 3 outputs : $<, =, >$
- Expandable to more number of bits
- ALU এর একটা part/circuit
- এটা একটা combinational circuit যেটা যেকোনো length এর binary number কে compare করতে পারে। $(<, >, =, \leq, \geq, !=)$

Consider two numbers A, B.

$$A = A_3A_2A_1A_0$$
$$B = B_3B_2B_1B_0$$
} 4-bit

দুইটা number সমান হইতে হলে number দুইটার length check করতে হবে। Length যদি মেলে তাহলে প্রতিটা element check করতে হবে। সবগুলো element মিলে গেলে তাহলে দুইটা number সমান হবে।

if $A_3 = B_3$ and $A_2 = B_2$ and $A_1 = B_1$ and $A_0 = B_0$; The two numbers are equal.

equivalence function,

$$x_i = A_iB_i + A_i'B_i' \quad ; \quad i = 0, 1, 2, 3 \ldots$$

where $x_i = 1$ only if the pair of bits in position i are equal, i.e. if both are 1's or both are 0's.

$$(A=B) = x_3x_2x_1x_0$$

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

$$(A>B) = A_3B_3' + x_3A_2B_2' + x_3x_2A_1B_1' + x_3x_2x_1A_0B_0'$$

এই ৪ টা condition এর যেকোনোটা যদি অন্য হয় তাহলে A বড়।

$$(A<B) = A_3'B_3 + x_3A_2'B_2 + x_3x_2A_1'B_1' + x_3x_2x_1A_0'B_0'$$

A<B,

A>B,

A=B

দিয়ে

A≤B

A≥B

A!=B

বের করতে হবে।

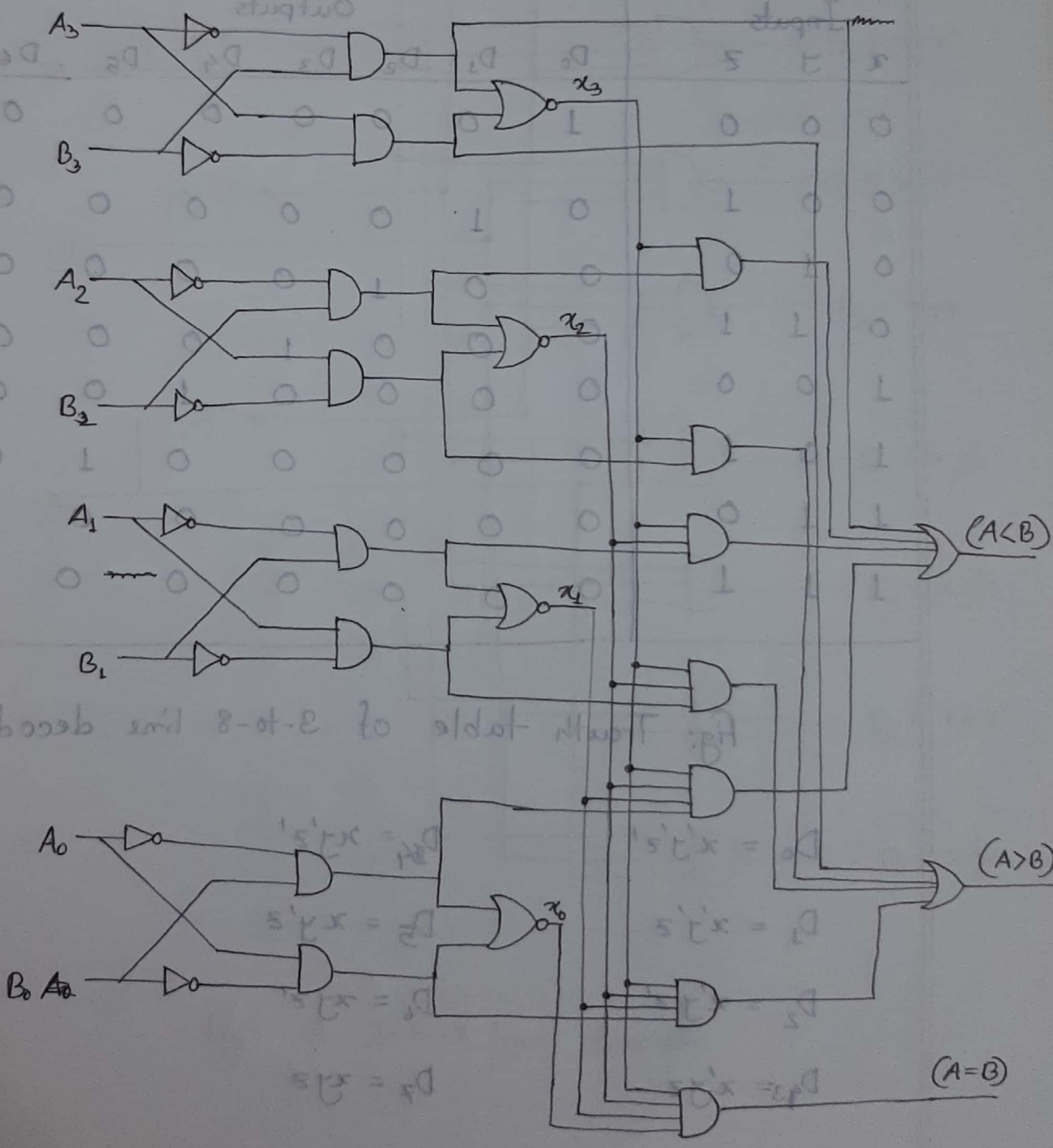


Fig: 4-bit magnitude comparator

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Decoders

A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2^n unique output lines. (Extract 'information' from the code)

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

Fig: Truth table of 3-to-8 line decoder

$D_0 = x'y'z'$

$D_1 = x'y'z$

$D_2 = x'yz'$

$D_3 = x'yz$

$D_4 = xy'z'$

$D_5 = xy'z$

$D_6 = xyz'$

$D_7 = xyz$

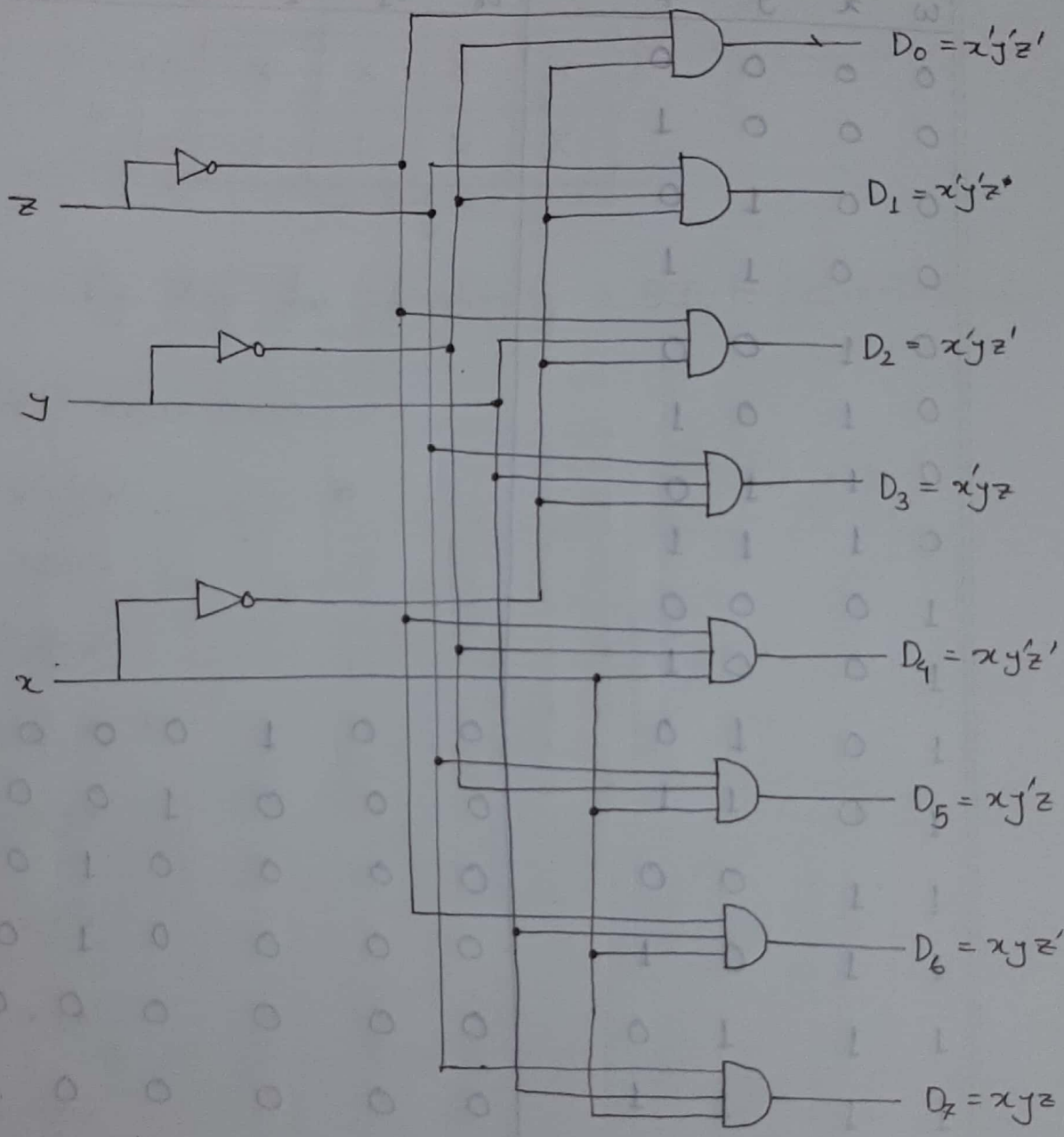


Fig: A 3-to-8 line decoder

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Example - 5.2 : Design a BCD-to-decimal decoder
*5-10 circuit

Inputs				Outputs									
w	x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	D_8	D_9
0	0	0	0										
0	0	0	1										
0	0	1	0										
0	0	1	1										
0	1	0	0										
0	1	0	1										
0	1	1	0										
0	1	1	1										
1	0	0	0										
1	0	0	1										
1	0	1	0	0	0	1	0	0	0	0	0	1	0
1	0	1	1	0	0	0	1	0	0	0	0	1	0
1	1	0	0	0	0	0	0	1	0	0	0	1	0
1	1	0	1	0	0	0	0	1	0	0	0	0	1
1	1	1	0	0	0	0	0	0	0	1	0	1	0
1	1	1	1	0	0	0	0	0	1	0	0	0	1

Fig: Truth table for 5-10 circuit

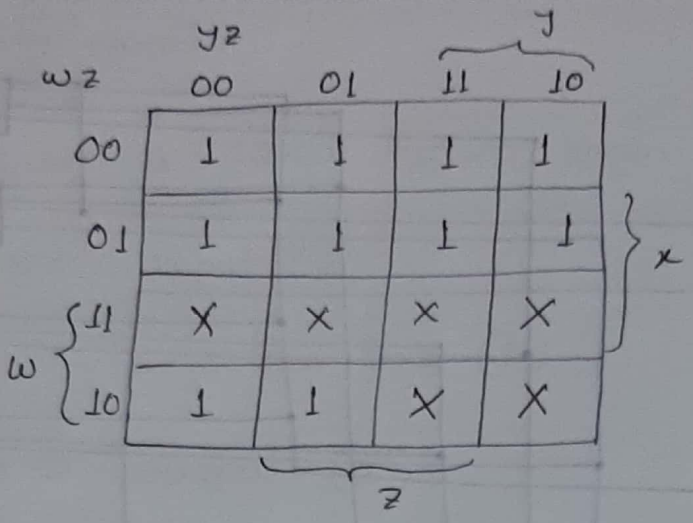


Fig: Map for simplifying a BCD-to-decimal decoder

Simplified equations -

$D_0 = w'x'y'z'$

$D_1 = w'x'y'z$

$D_2 = x'yz'$

$D_3 = x'yz$

$D_4 = xy'z'$

$D_5 = xy'z$

$D_6 = xyz'$

$D_7 = xyz$

$D_8 = wz'$

$D_9 = wz$

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

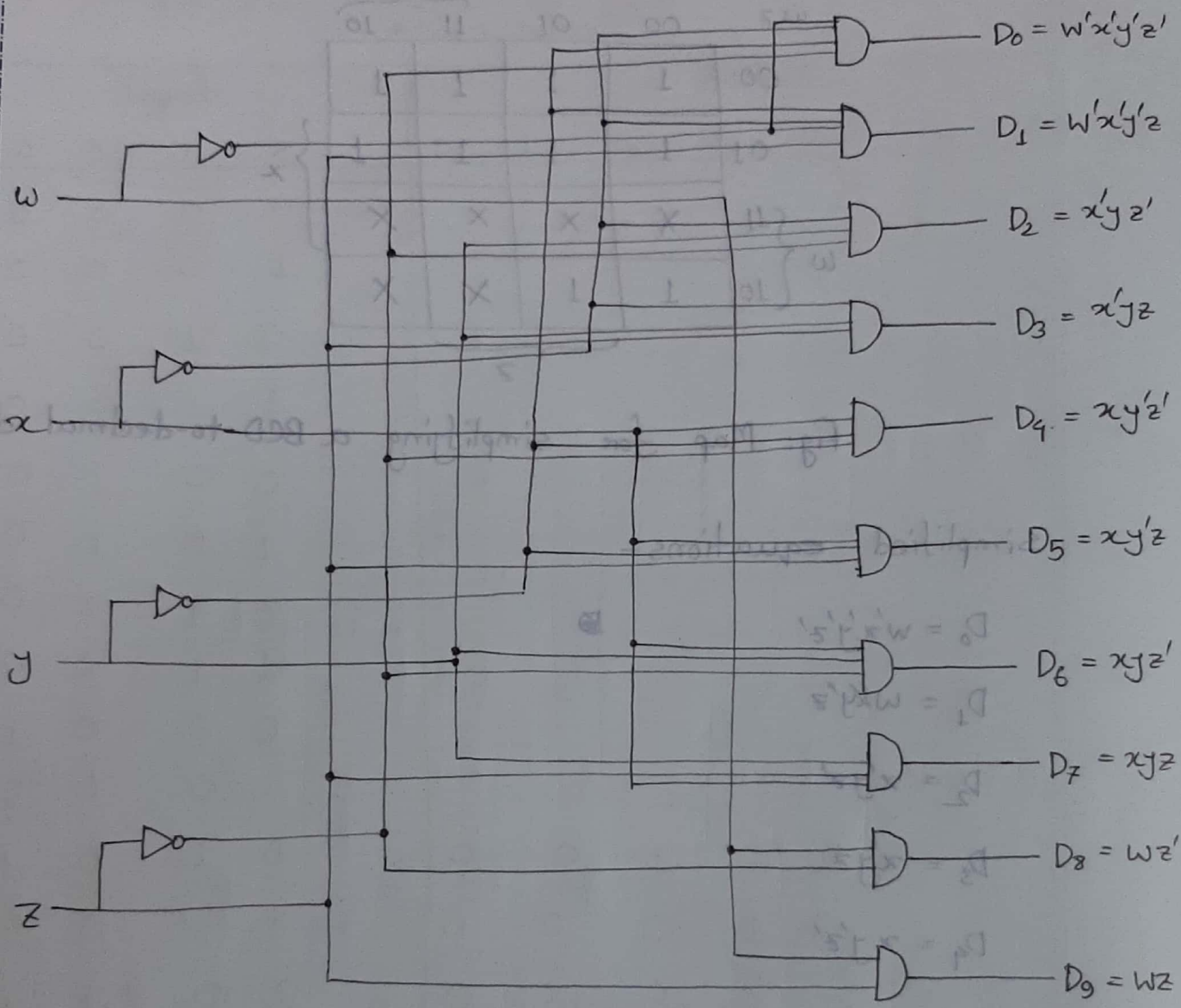


Figure: BCD-to-decimal decoder

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Demultiplexer

Demultiplexer এ তিনটি জিনিস থাকে।

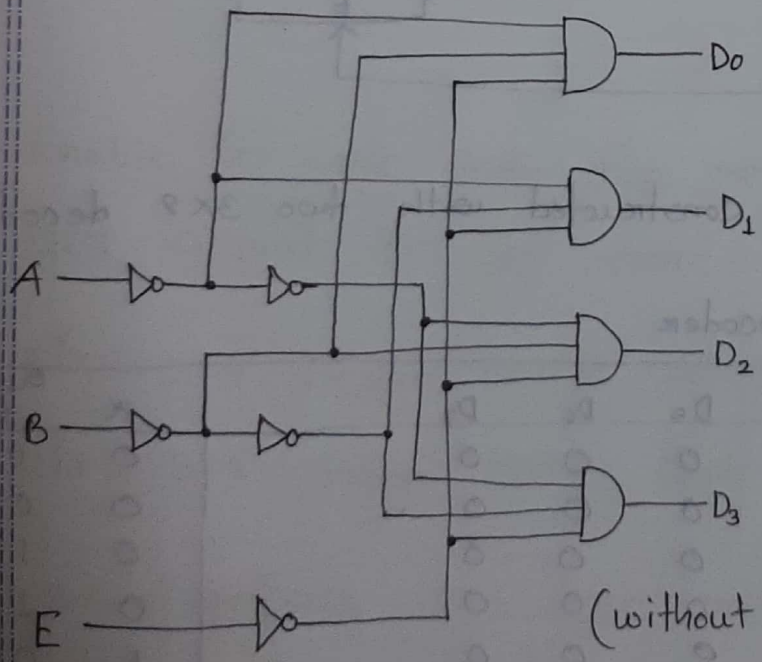
I) Enable input (E)

II) 2 টি selection line (N)

III) 4 টি output (2^N)

Decoder এর আগে একটি enable যোগ করে দিলে সেটাকে বলা হয় demultiplexer.

Enable এর মূল কাজ হচ্ছে control করা। এটির মান high করে রাখলে circuit কোনো কাজ করবে না।



(without inverter)

(a) Logic Diagram

E	A	B	D_0	D_1	D_2	D_3
1	x	x	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
0	1	1	0	0	0	1

(b) Truth Table

Fig: A 2-to-4 line decoder with enable (E) input

Plan Title : Online Class - 07

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date : 06/06/2021

* Encoders

An encoder is a digital function that produces a reverse operation from that of a decoder.

An encoder has 2^n (or less) input lines and n output lines.

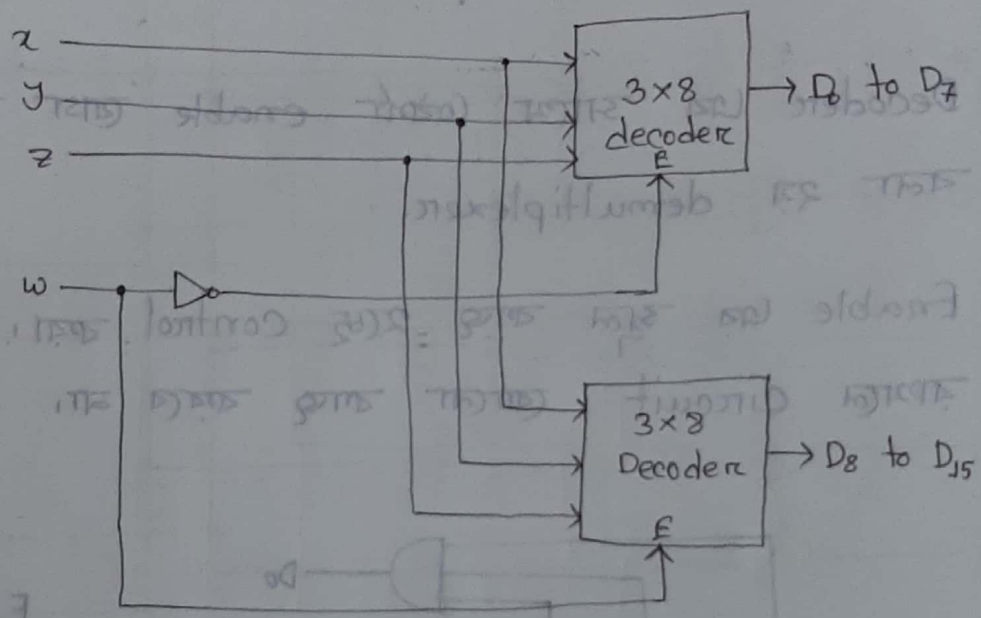


Fig: A 4×16 decoder constructed with two 3×8 decoders.

* Octal-to-binary encoder

Inputs								Outputs		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
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

Fig : Truth Table

Plan Title :

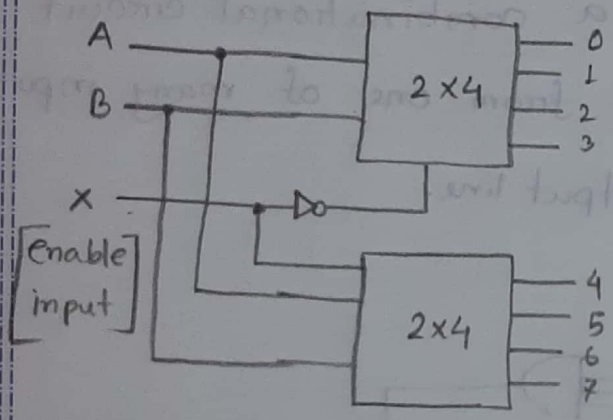
Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

*** Hexadecimal to Binary

*** Decimal to BCD

* 2×4 থেকে 3×8 Design



X	A	B
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Enable টা AND Gate দিয়ে করা হয়। AND Gate দিয়ে করলে পুরো circuit টা off থাকবে।

2টা 2×4 Design করলে একটা 3×8 হয়।

2টা 3×8 Design করলে একটা 4×16 হয়।

*** শুধুমাত্র 2×4 use করে 4×16 Design করতে হবে

Plan Title :

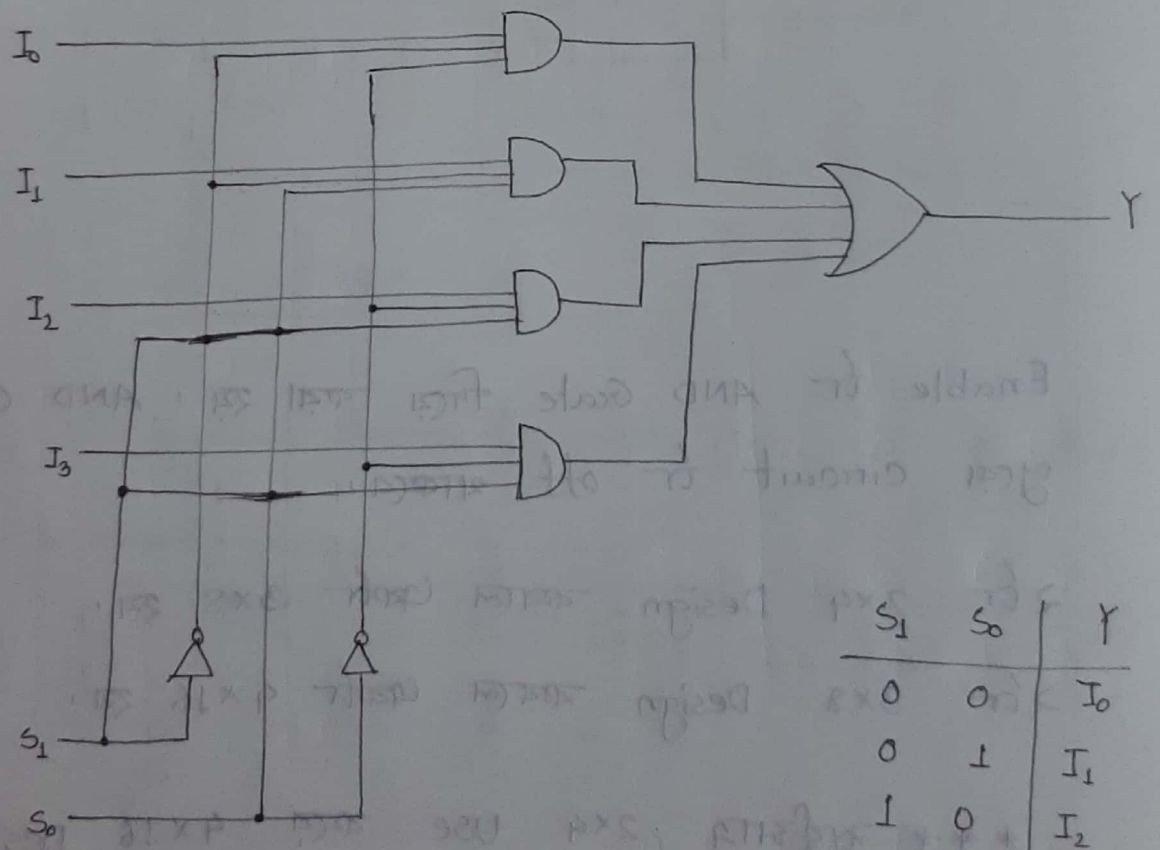
Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Multiplexer

- 2^n input
- n selection line
- 1 output

A digital multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line.



Block Diagram

S_1	S_0	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

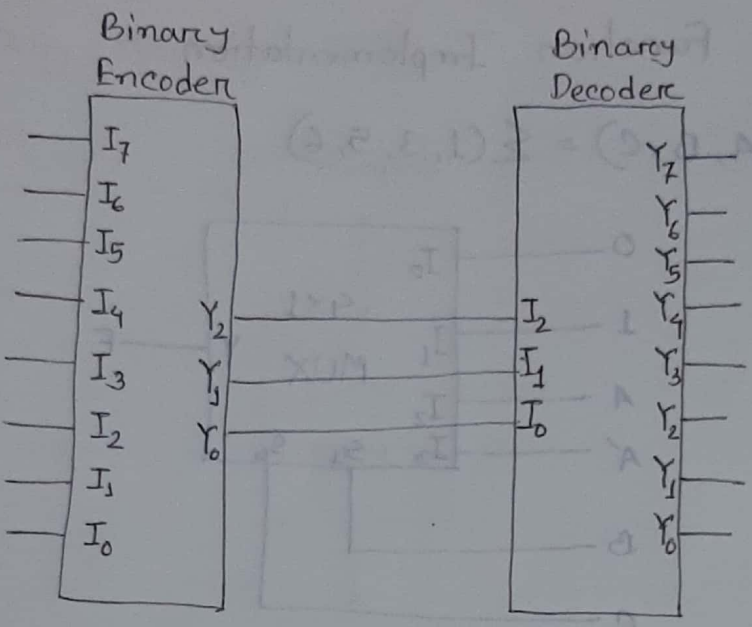
Function Table

Plan Title : Online Class – 09

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date : 07./06./2021

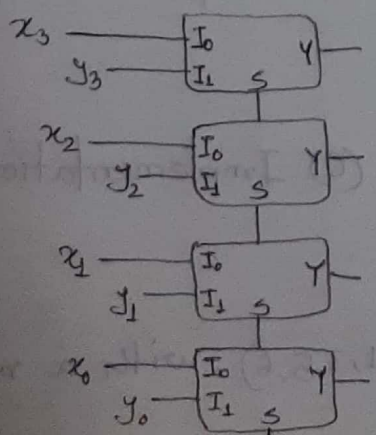
** Decoder / Encoder Pairs



* Multiplexer

A mutiplexer is a digital or combinational circuit that takes input from 2^n line and have two selection line and process it to make a single line output.

* Quad 2-to-1 Multiplexer



[slide থেকে]

Plan Title :

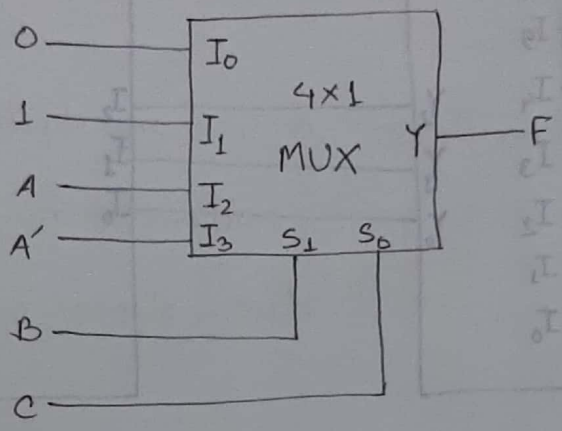
Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Quadruple 2-to-1 line Mutiplexer =

* Boolean Function Implementation

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



(a) Multiplexer Implementation

Minterm	A	B	C	F
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

(b) Truth Table

	I_0	I_1	I_2	I_3
A'	0	①	2	③
A	4	⑤	⑥	7
	0	1	A	A'

(c) Implementation Table

fig: Implementing $F(A, B, C) = \Sigma(1,3,5,6)$, with a multiplexer

** Example 5-4 =

* Read-Only Memory (ROM)

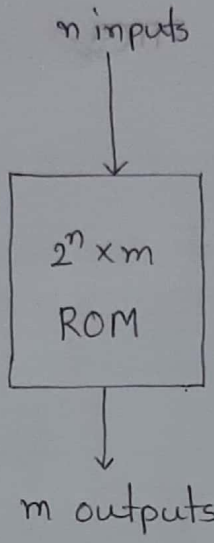


fig: ROM block diagram

** Logic construction of a 32 x 4 ROM

* Combinational Logic Implement

ROM এর সাহায্যে করা যায়.

A_1	A_0	F_1	F_2
0	0	0	1
0	1	1	0
1	0	1	1
1	1	1	0

figure-5.23

$F_1 = A_0'A_1 + A_0A_1' + A_0A_1$

$\Rightarrow F_2 = A_1'A_0' + A_1A_0'$

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Programmable Logic Array (PLA)

Self-study

Question আসতে পারে

Chapter-06

Sequential Logic

* Sequential Circuits

- Consists of a combinational circuit to which storage elements are connected to form a feedback path.

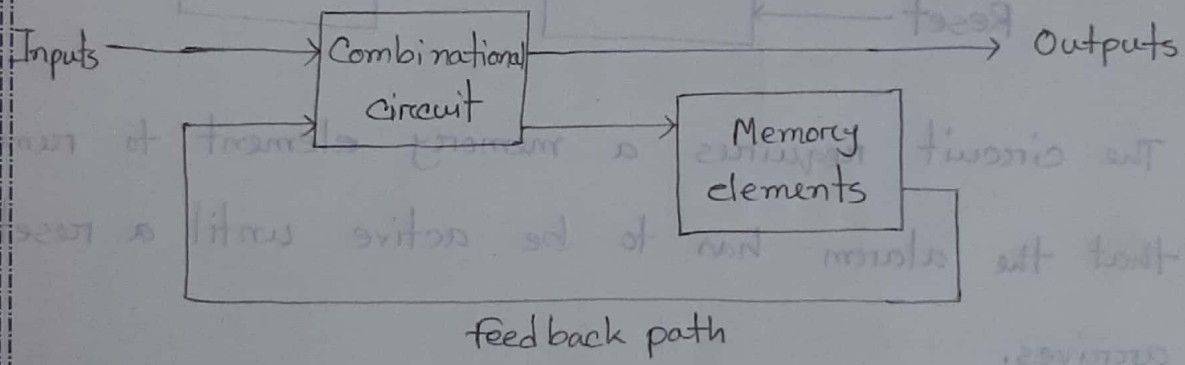


Fig: Block diagram of S.C.

- State: the state of the memory devices now, also called current state.
- Next states and outputs are functions of inputs and present states of storage elements.

Previous state
Current state
Next state.

previous state এর output, current state এর input নির্ণয় করবে next state এর output এর উপর.

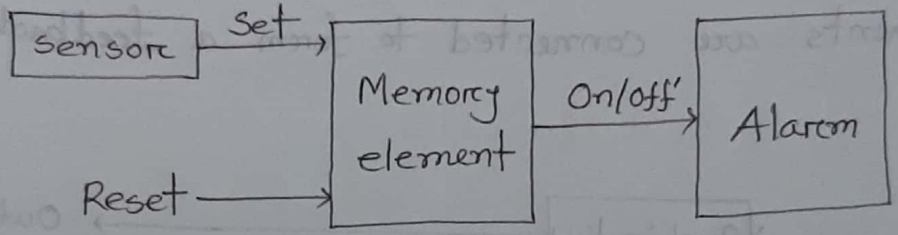
Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Alarm Control System

- Suppose we wish to construct an alarm circuit such that the output remains active (on) even after the sensor output that triggered the alarm goes off.



- The circuit requires a memory element to remember that the alarm has to be active until a reset signal arrives.

* Types of Sequential Circuits

- Asynchronous sequential circuit
- Synchronous sequential circuit

* Asynchronous sequential circuit

- Depends upon the input signals at any ~~east~~ instant of time and their change orders
- May have better performance but hard to design.

* Synchronous sequential circuit

- Defined from the knowledge of its signals at discrete instant of time.
- Much easier to design (preferred design style)
- Synchronized by a periodic train of clock pulse.

[যেখানে operation এর গুরুত্ব আছে কিন্তু data processing এর গুরুত্ব নাই। (Syn. & Asyn.)]

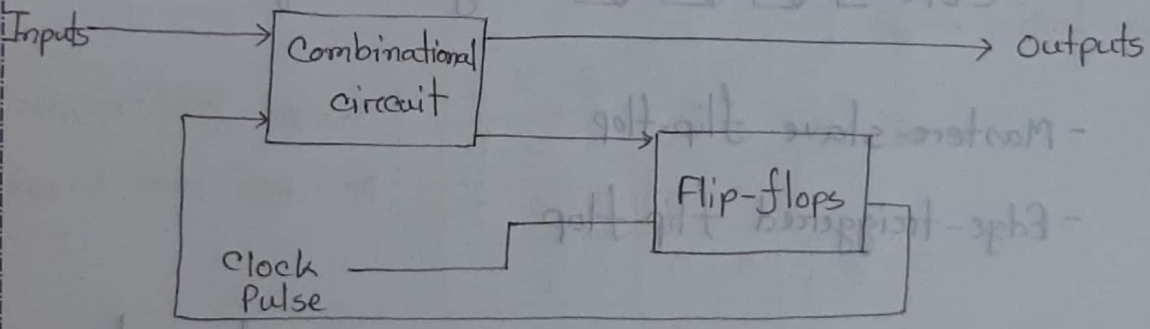


fig: Block diagram

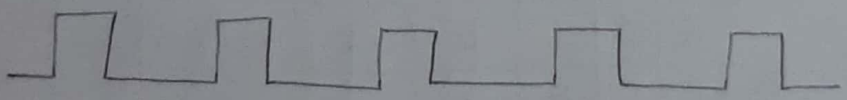


fig: Timing diagram of clock pulses

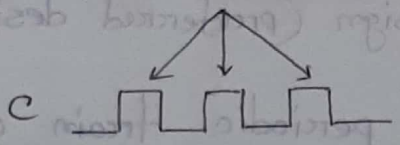
Figure: Synchronous Clocked Sequential Circuit

Flip-Flops: Flip-flop is an elementary memory element which have the ability to store 1 bit of binary information.

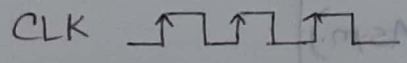
** Register তৈরী করতে flip-flop use করা।

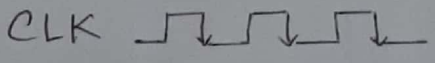
* Memory Elements

- Latch:- a level-sensitive memory element
  - SR latches
  - D latches



- Flip-Flop:- an edge-triggered memory element





  - Mastere-slave flip-flop
  - Edge-triggered flip-flop

- RAM and ROM a mass memory element.

* Latches

- binary storage element
- Can store a 0 or 1
- The most basic memory
- Easy to build
- Built with gates (NORs, NANDs, NOT)

Plan Title :

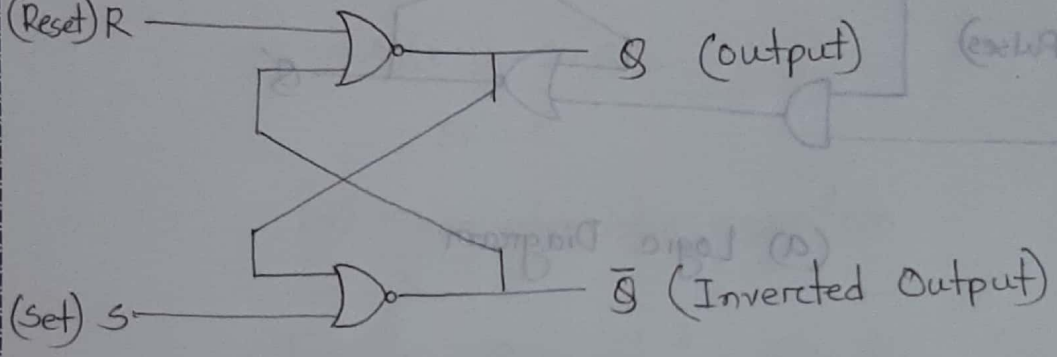
Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

** 8-bit এর register 8 টা flip-flop দিয়ে তৈরী হয়.

** একটা 1 kb memory তে 8000-9000 টা flipflop থাকে.

- SR Latch



Q_0 → initial value

[কোনো কিছু ~~Reset~~ Reset করা হলে Q=0]

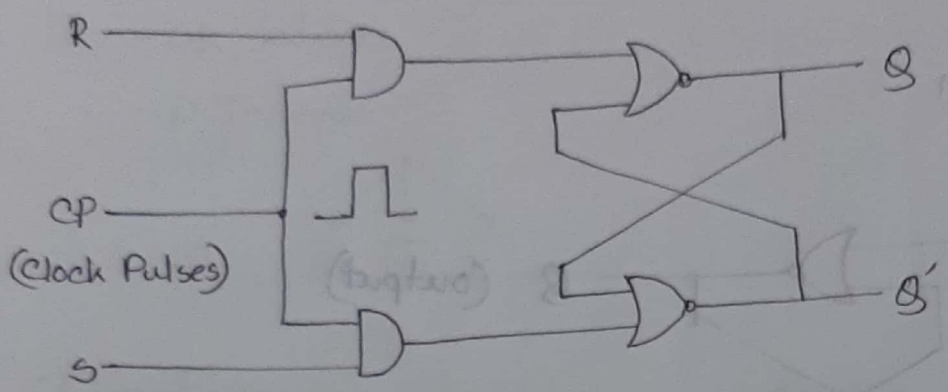
S	R	Q_0	Q	Q'	
0	0	0	0	1	$Q = Q_0$
0	0	1	1	0	
0	1	0	0	1	$Q = 0$
0	1	1	0	1	
1	0	0	1	0	$Q = 1$
1	0	1	1	0	
1	1	0	0	0	$Q = Q'$
1	1	1	0	0	$Q = Q'$

RS Flip Flop
SR Latch

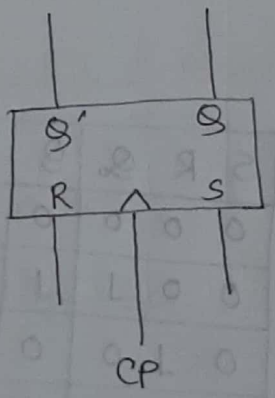
Flip-flop / Latch এর main target হলো memory তৈরী করা।
1 bit information store করা.

Latch আর flip-flop - internal operation এ পার্থক্য আছে কিন্তু বাস্তবিক output এ কোনো পার্থক্য নাই.

* Clocked RS Flip-Flop



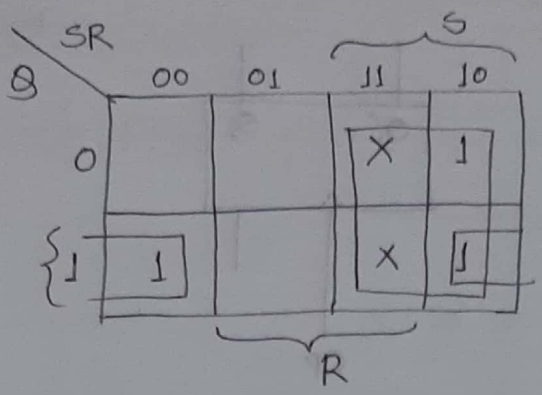
(a) Logic Diagram



(b) Geraphic Symbol

a) Characteristic table

Q	S	R	Q(t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	Indeterminate
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	Indeterminate



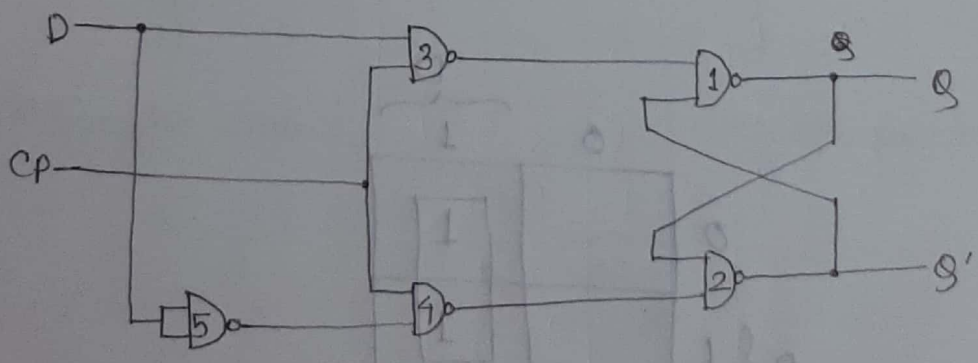
$Q(t+1) = S + R'Q$

$SR = 0$

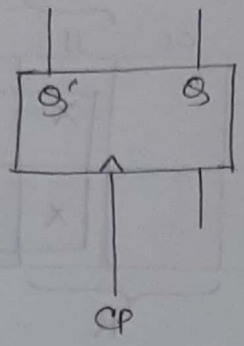
(d) Characteristic equation

Figure: Clocked RS flip-flop ((a), (b), (c), (d))

* D flip-flop



(a) Logic diagram with NAND gates

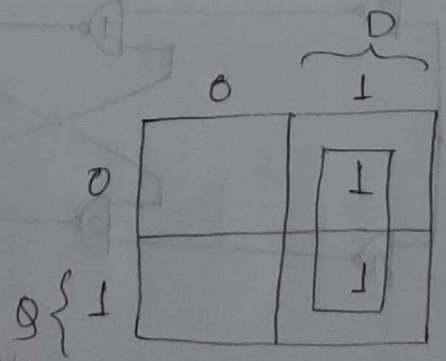


(b) Graphic Symbol

Q	D	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

[Sir এর খাতা বলতে হবে]

(c) Characteristic table



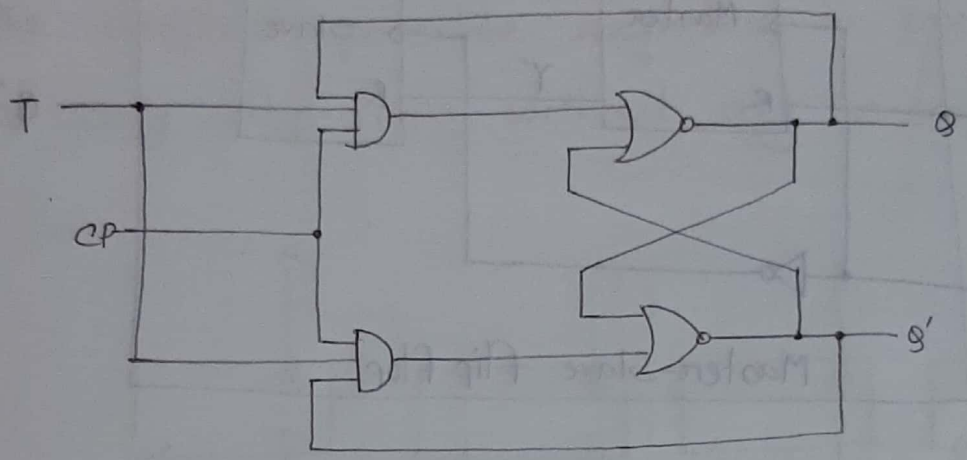
$Q(t+1) = D$

(d) Characteristic equation

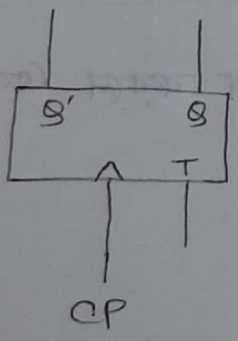
Figure : Clocked D flip-flop

* JK Flip-Flop=

* Triggering of Flip-Flop=



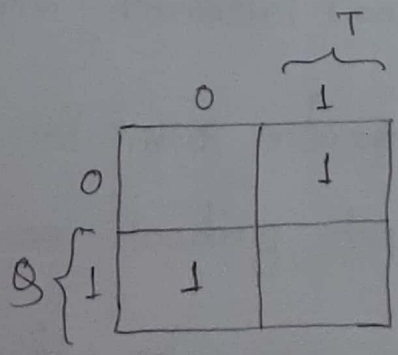
(a) Logic Diagram



(b) Graphic Circuit

Q	T	Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0

(c) Characteristic Table



$Q(t-1) = TQ' + T'Q$

(d) Characteristic equation

Figure: Clocked T flip-flop

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Master-slave flip-flop

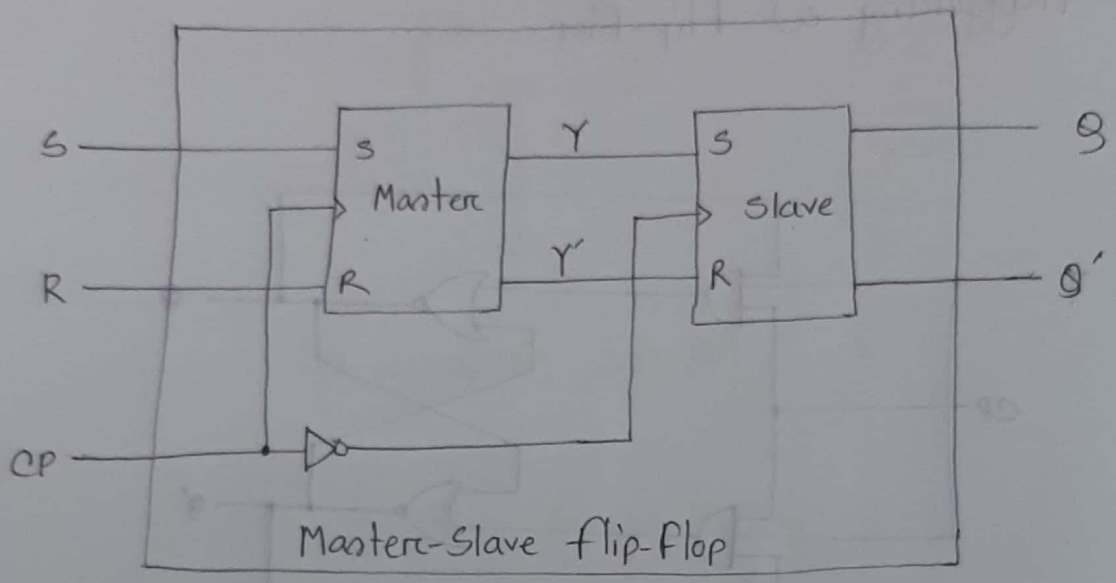


figure: Logic Diagram

* 4 ta master-slave flip-flop jora korle ekta shift register

Chapter - 07

Registers, Counters and the Memory Unit

- The simplest possible register is one that consists of only flip-flops without any external gates.

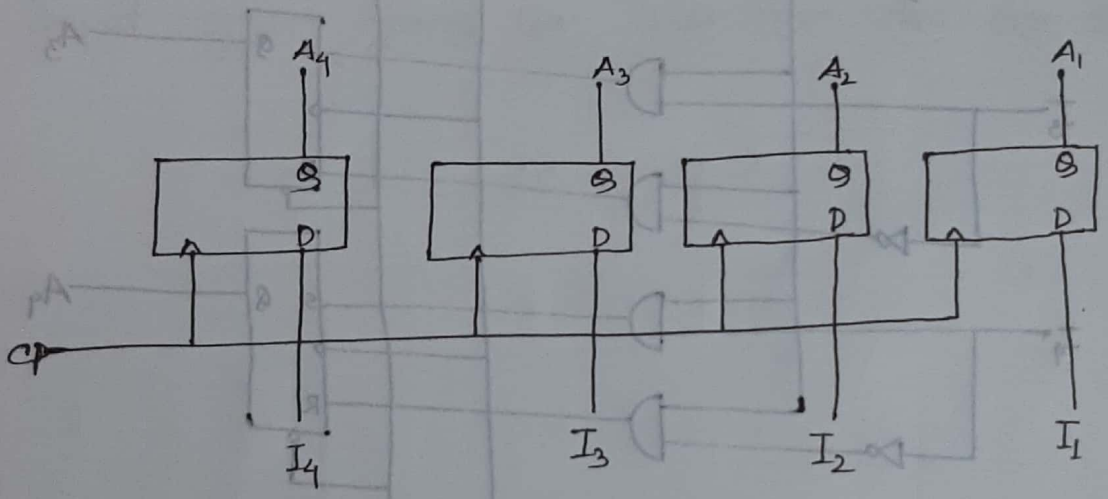


fig: 4-bit register

* Register with Parallel Load

- The transfer of new information into a register is refferred to as loading the register.
- If all the bits of the register are loaded simultaneously with a sig single clock pulse, we say that the loading is done in parallel.

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

(Load এর মান Normally 1 থাকে)

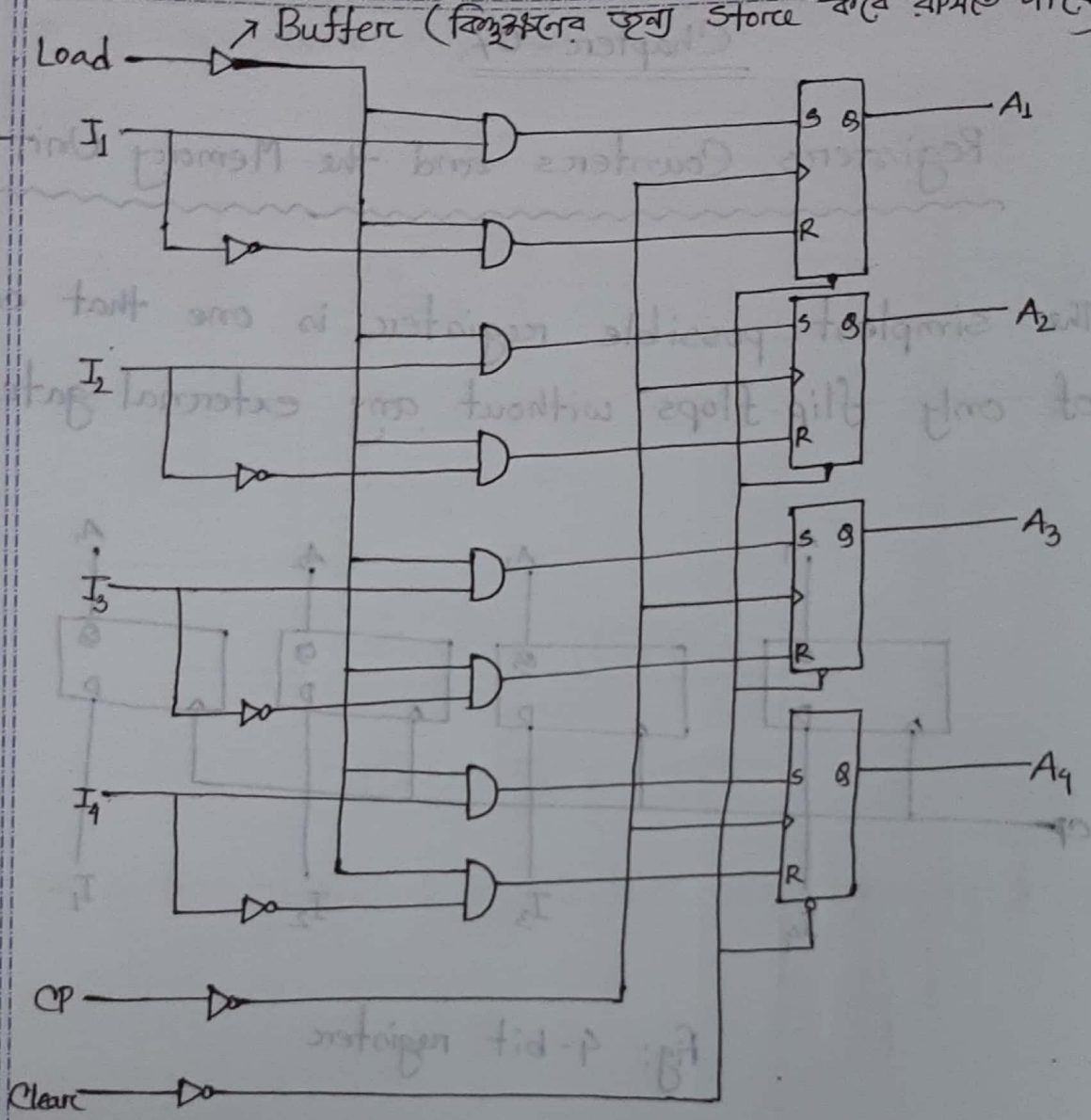


Fig: 4-bit register with parallel load

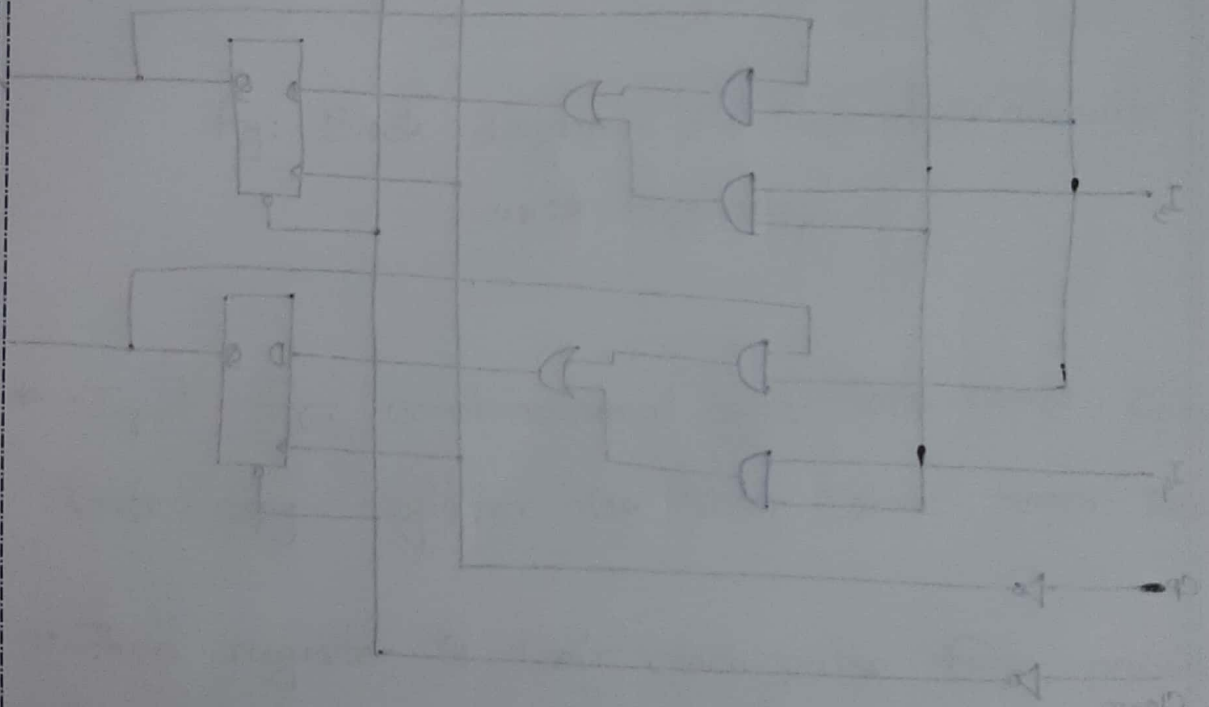
- Clear - flipflop কে reset করে ফেলে। Clear press করলে memory তে যা আছে সব clear হয়ে যাবে।
- এখানে Inverted CP use করা হয়েছে। অর্থাৎ CP low হলে বা ০ হলে, register এর content change হয় না। CP=1 হলে input information register-এ load হয়ে যাবে।

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

- Load apply করলে AND gate গুলো active হয়।
- Load এর মান আবশ্যিক 1 থাকে।
- Input যেটা দেয়া হবে সেটা অরাসরি যাবে S-এ এবং Inverted হয়ে যাবে R-এ।
- Load এবং CP যখন একই আসে, তখন এক pulse এর মধ্যে Input গুলো ঢুকবে এবং একসাথে বের হয়ে যাবে।



Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

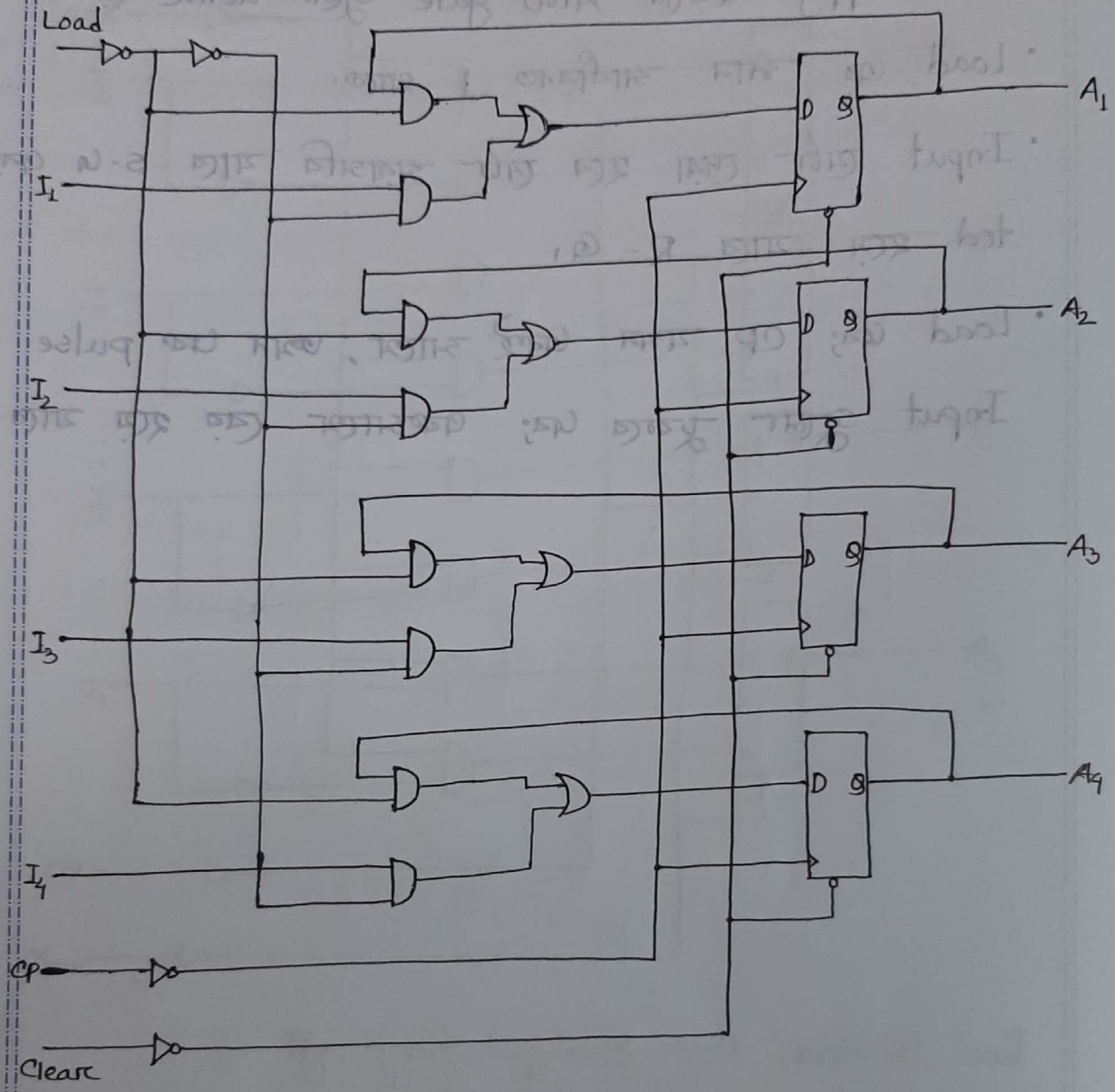


Fig: Register with parallel load using D flip-flops

- Load একবার inverted হয়ে আবার inverted হচ্ছে, অর্থাৎ original load input এর সাথে ANDing হচ্ছে + previous load এর output এর সাথে ANDing হচ্ছে; inverted load এর সাথে।

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Sequential logic Implementation

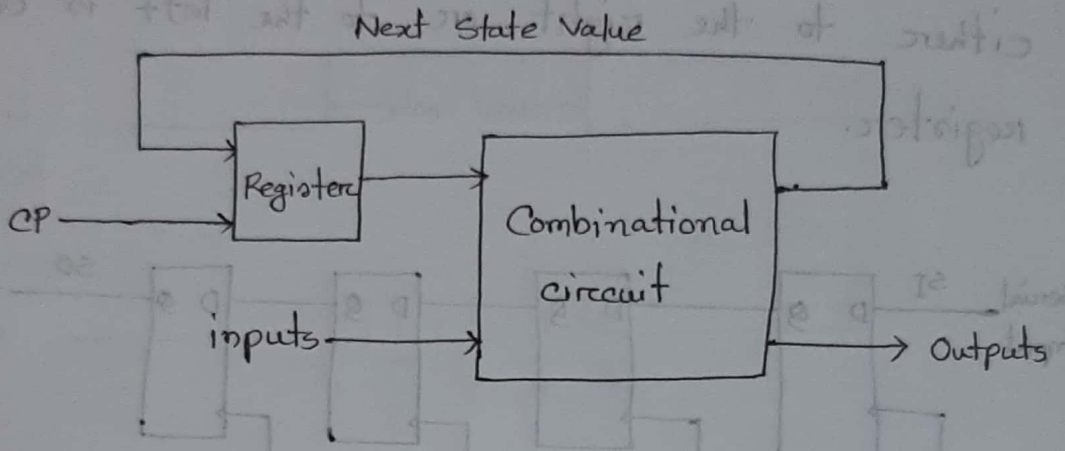


Fig: Block diagram of Sequential circuit with register

* • Input দিলে combinational logic কাজ করবে। Output এর একটা copy register এর মধ্যে save করবে। Then

• Then register এ একটা clock-pulse দিলে, previous state এর output আর current state এর input এর সাথে মিলে হয়ে output দিবে।

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Shift Register

- A register capable of shifting in binary information either to the right or to the left is called a shift register.

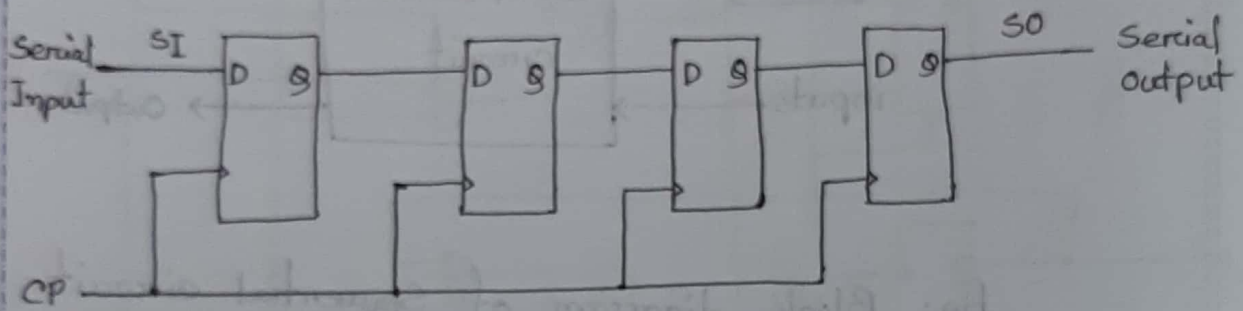
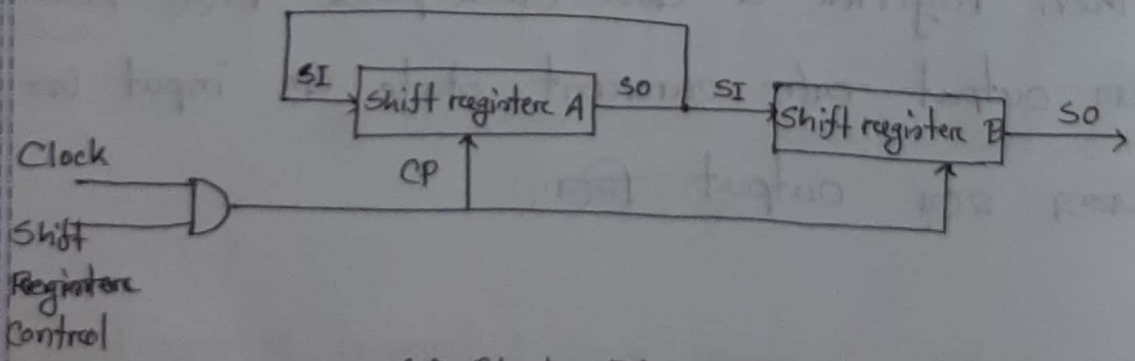


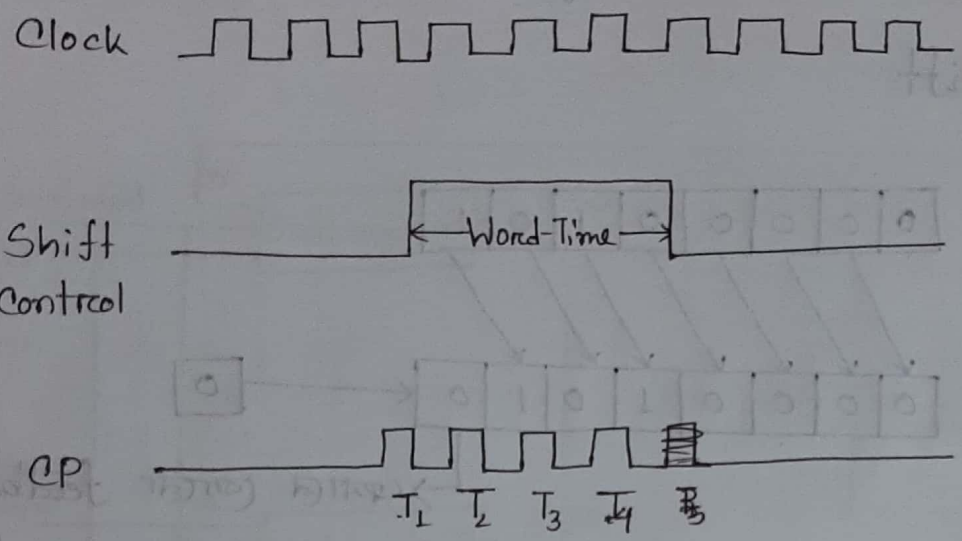
fig: Shift Register

* Serial Transfer

- Serial Transfer ৫ দুইটা ~~Block diagram~~ Shift register থাকে।



(a) Block Diagram



(b) Timing Diagram

Fig: Serial Transfer from Register A to Register B

Timing Pulse	Shift Register A	Shift Register B	Serial Output of B
Initial Value	1 0 1 1	0 0 1 0	0
After T_1	1 1 0 1	1 0 0 1	1
After T_2	1 1 1 0	1 1 0 0	0
After T_3	0 1 1 1	0 1 1 0	0
After T_4	1 0 1 1	1 0 1 1	1

Fig: Serial Transfer Example

• shift variable এর কাজ হলো serial transfer

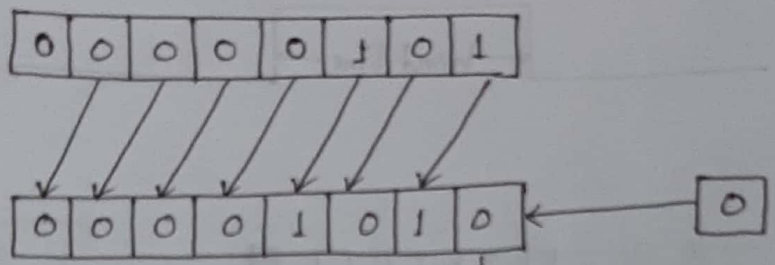
Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Bidirectional Shift Register with Parallel load

• Left shift

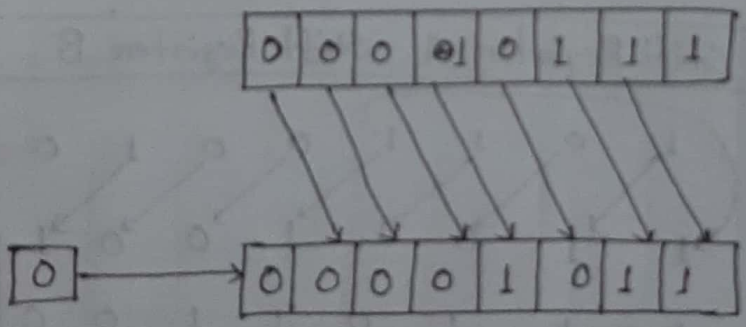


↳ কোনো কোনো feedback না থাকলে 0 হবে।

Rotate করলে বাড়লে পরে আবার কমলেও পারে।

- Left shift করলে value বাড়বে.

• Right shift



- Right shift করলে value কমে

*** 4-bit bidirectional shift register with parallel load

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Serial Addition

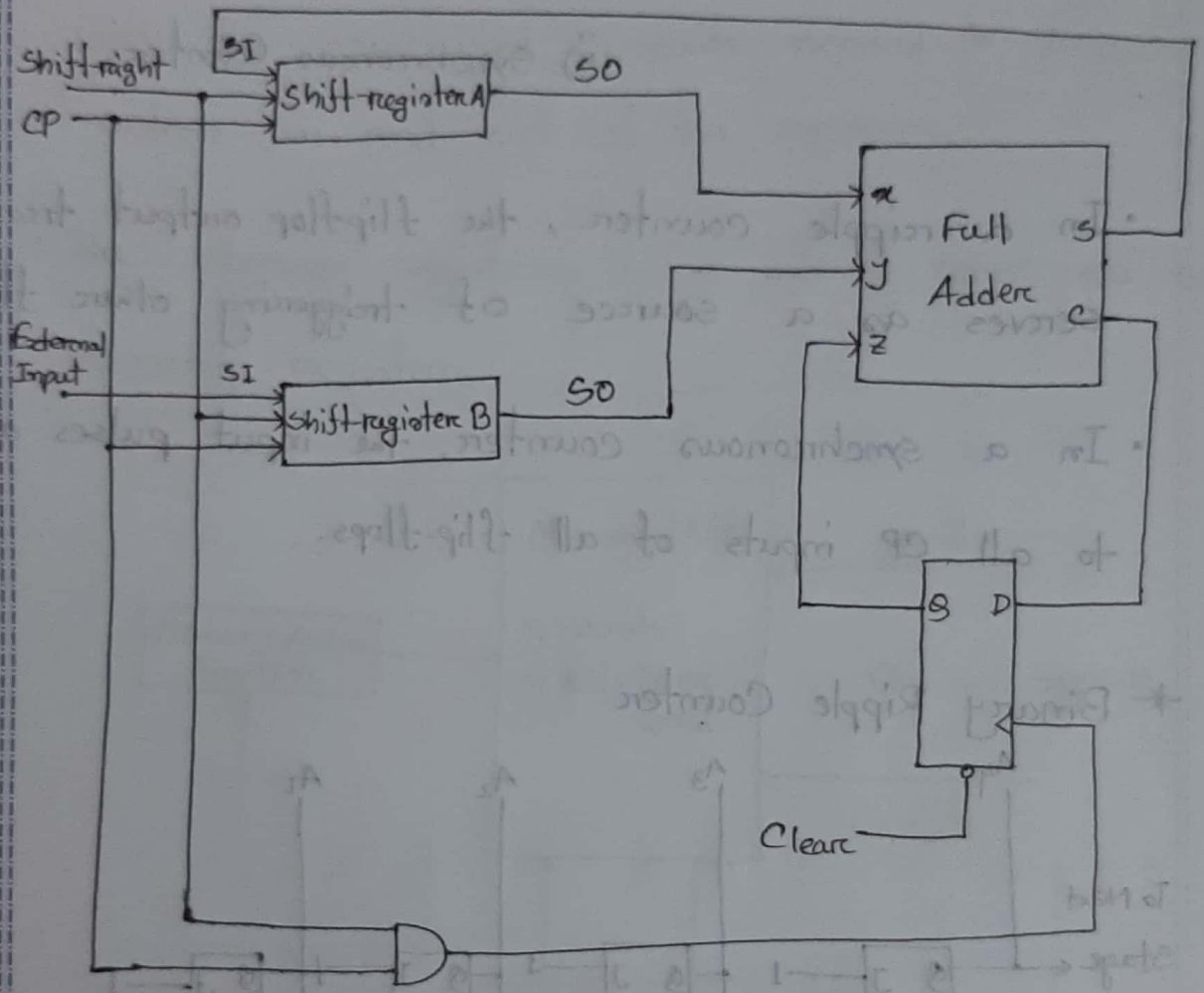


Fig: Serial Adder

• Serial adder use করলে অনেক circuit এর length কম যায়

*** Example - 7.3

Table - 7.3

* Second form of a serial adder

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

* Ripple Counters

MSI Counters → i) Ripple Counters

ii) Synchronous Counters.

- In a ripple counter, the flipflop output transition serves as a source of triggering other flip-flops.
- In a synchronous counter, the input pulses are applied to all CP inputs of all flip-flops.

* Binary Ripple Counter

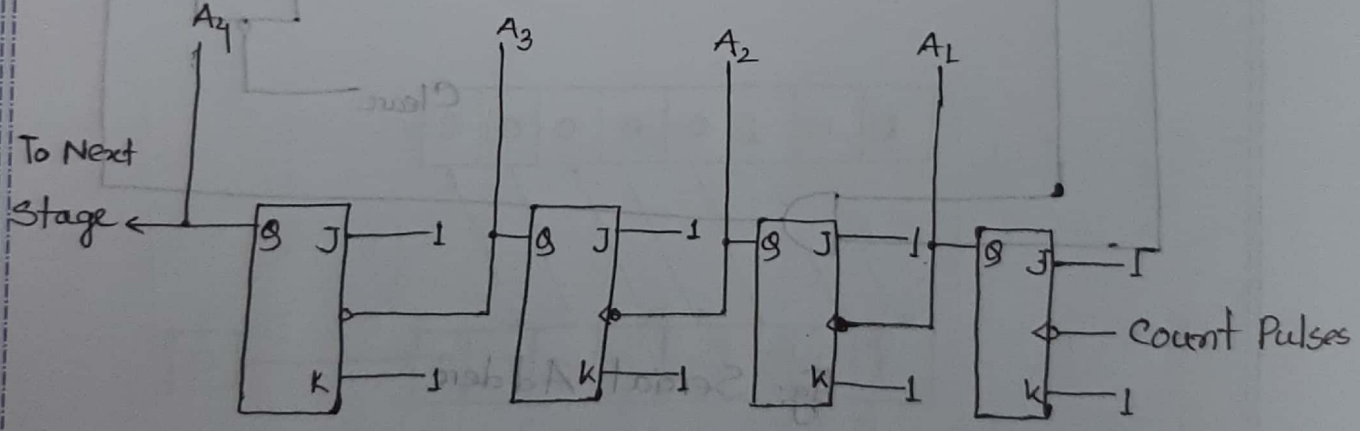


fig: 4-bit binary ripple counter

* Table - 7.4

*** BCD Ripple Counter

** Johnson Counter

* The Memory Unit

- A memory unit is a collection of storage registers together with the associated circuits needed to transfer information in and out of the registers.

- The storage registers in a memory unit are called memory registers.

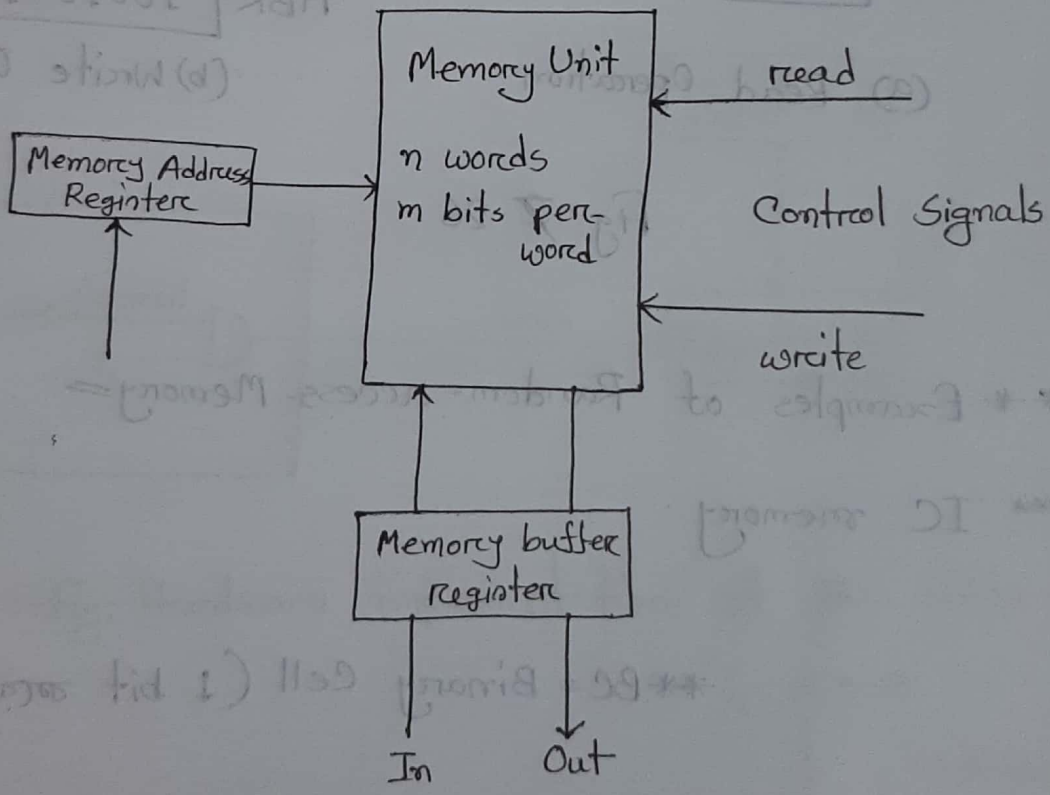


Fig: Block diagram of memory unit showing communication with environment

*** Information transfer during read and write operations (Fig-7.26)

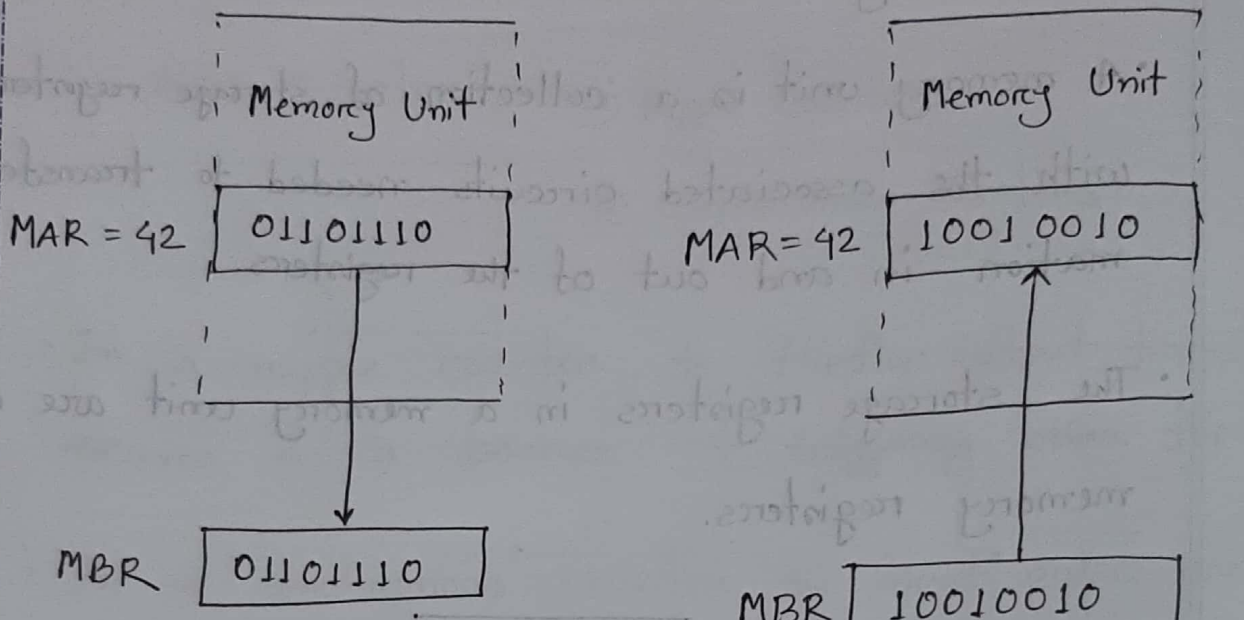


Fig - 7.26

** Examples of Random-access Memory=

** IC memory

**BC = Binary Cell (1 bit করে store করতে পারে)

Figure - 7.30 : IC memory 3 bit draw না করে 4/8 bit draw করতে হবে.

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

Chapter - 08

Register- Transfer Logic (RTL)

* 8.1

- কোনো Register থেকে Register-এ transfer বুঝানোর জন্য একটা special logic ব্যবহার করা হয়, সেটাকে RTL বলা হয়.

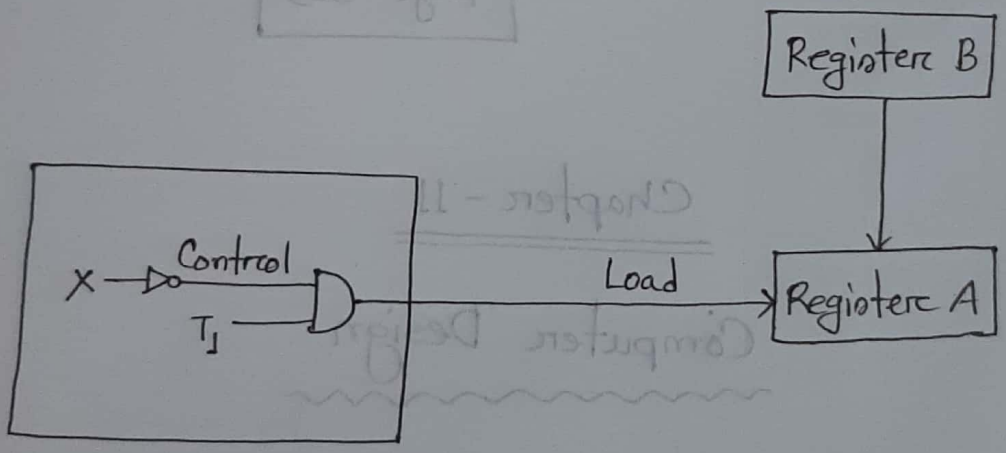


Fig: Hardware implementation of the statement:

$X'T_1 : A \leftarrow B$

↳ Source [ডান দিকে]

↳ Destination [বাম দিকে]

Plan Title :

Sat / Sun / Mon / Tue / Wed / Thu / Fri

Date :...../...../.....

Chapter - 09

Processor Logic Design

** Figure - 9.1: Processor registers and ALU connected through common busts

Page - 319

Chapter - 11

Computer Design

** Figure - 11.7: Detailed Block diagram for Computer

Page - 428