Minimization of switching functions

Note Title 5/21/2021

Karnaugh map (K-map) method

systematic method of simplifying the Boolean expressions

K-map is a chart or a graph, composed of an arrangement of adjacents cells, each representing a particular combination of variables in sum or product form.

* it is tedious for problems involving 5 or more variables

An n variable function → 2^n possible combinations of products in SOP form

2^n possible comb. of sums in POS form.

two variable K-map	$2^2 = 4$	cells / squares
three " "	$2^3 = 8$	cells / squares
four " "	$2^4 = 16$	cells / squares and so on.

SOP Sum (OR) of products (AND) form / minterm

POS product (AND) of sum (OR) form. / maxterm.

for simplicity minterms and maxterms are usually rep. as binary words in terms of 0s and 1s. instead of actual variables.

m, m_0, m_1, m_2, ... m_5 ... m_7.

M_1, M_0, M_1, M_2 ...

Two variable K-map

$2^2 = 4$ possible combination of the input variables A & B

$\bar{A}\bar{B}$, $\bar{A}B$, $A\bar{B}$ and AB (in the SOP form) is called a minterm

$\bar{A}\bar{B} = m_0$, $\bar{A}B = m_1$, $A\bar{B} = m_2$, $AB = m_3$

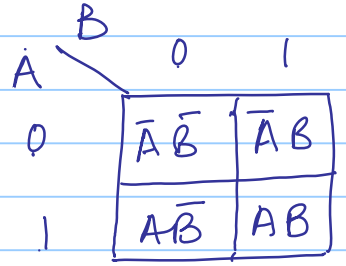
$f = \bar{A}\bar{B} + A\bar{B} + AB$

$F = m_0 + m_2 + m_3 = \sum m(0, 2, 3)$

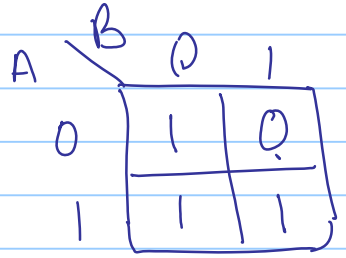
Truth table

Minterm	Inputs A	Inputs B	Output f
0	0	0	1
1	0	1	0
2	1	0	1
3	1	1	1

Mapping of SOP expressions



$\sum m(0, 2, 3)$



Ques Map the expression $f = \bar{A}B + A\bar{B}$
(01 → 10)

$f = m_1 + m_2$

$= \sum m(1, 2)$

A \ B	0	1
0	0	1
1	1	0

K-map of $\sum m(1, 2)$

Minimization of SOP expressions

$\bar{A}\bar{B}$ and $\bar{A}B$

The necessary (but not sufficient) cond^n for adjacency of minterms is that their decimal designations must differ by a power of 2.

m_0 and m_1 $\quad f_1 = m_0 + m_1 = \bar{A}\bar{B} + \bar{A}B = \bar{A}(\bar{B} + B) = \bar{A}$

m_0 and m_2 $\quad f_2 = m_0 + m_2 = \bar{A}\bar{B} + A\bar{B} = \bar{B}(\bar{A} + A) = \bar{B}$

m_1 and m_3 $\quad f_3 = m_1 + m_3 = \bar{A}B + AB = (\bar{A} + A)B = B$

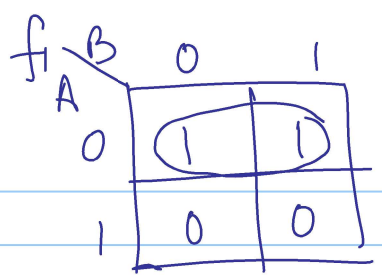
m_2 and m_3 $\quad f_4 =$ $\qquad\qquad = A$

m_0, m_1, m_2 and m_3, $\quad f_5 = m_0 + m_1 + m_2 + m_3$

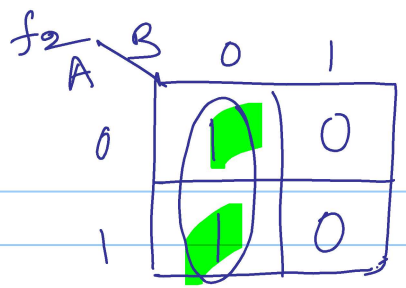
$= \bar{A}\bar{B} + \bar{A}B + A\bar{B} + AB$

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

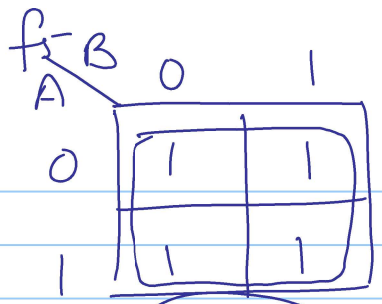
$= \bar{A} + A = 1.$



$f_1 = \bar{A}$



$f_2 = \bar{B}$

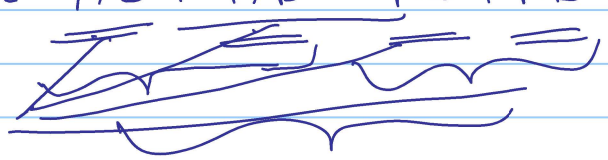


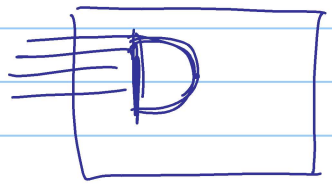
eliminate 2 - variables

Quad

$f_3 = 1$

$f_4 = \bar{A}\bar{B} + \bar{A}B + A\bar{B} + AB$





Mapping of POS exp.

* Each sum term in the standard POS expression is called a MAXTERM.

* A func^{n} in two variables (A, B) has possible 4 max.

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

↓ ↓ ↓ ↓

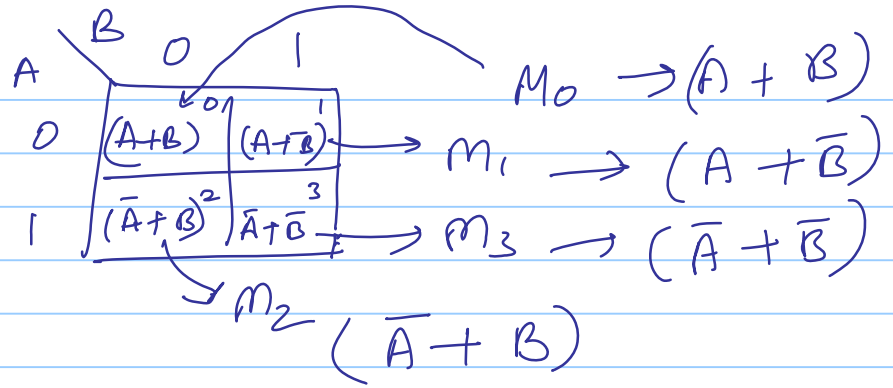
M_0 M_1 M_2 M_3

non complemented variable -0

complemented " -1

0's are placed in the squares corresponding to the max terms which are present in the expression

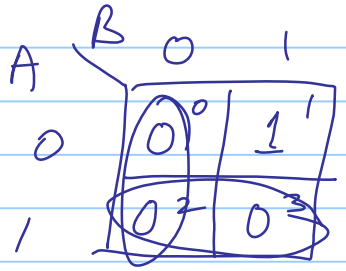
Two variable K-map



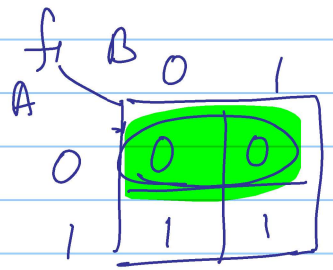
Ques Plot the expression $f = (A+B)(\bar{A}+B)(\bar{A}+\bar{B})$ on the K. map

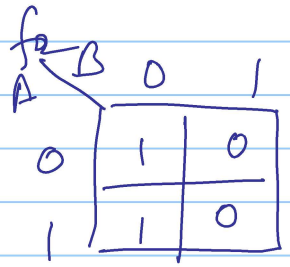
POS

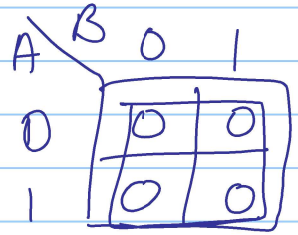
$f = \Pi\, M(0, 2, 3)$



Minimization of POS exp.







$f_1 = \Pi M(0, 1)$
$= A$

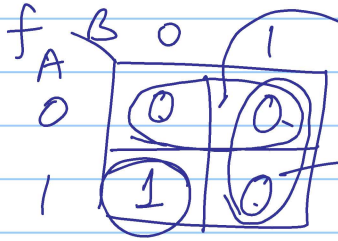
$f_2 = \Pi M(1, 3)$
$= \bar{B}$

$f_3 = 0 = \Pi M(0, 1, 2, 3)$

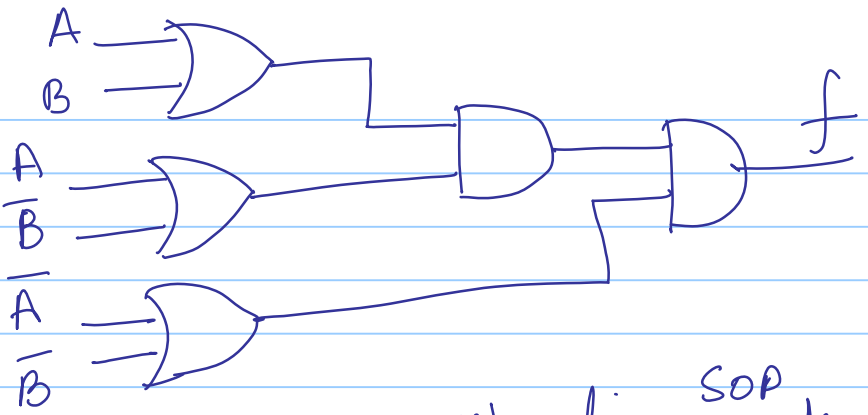
Ques Reduce the expression $f = (A+B)(A+\bar{B})(\bar{A}+\bar{B})$ using mapping

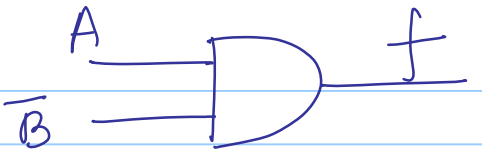
Sol.

$f = \Pi M(0, 1, 3)$



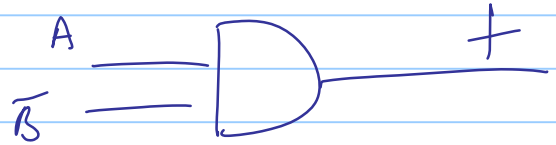
$f = (A)(\bar{B})$
$= A\bar{B}$





Reduced form.

corresponding SOP expression is $\sum_1 m_2$ $A\bar{B}$.

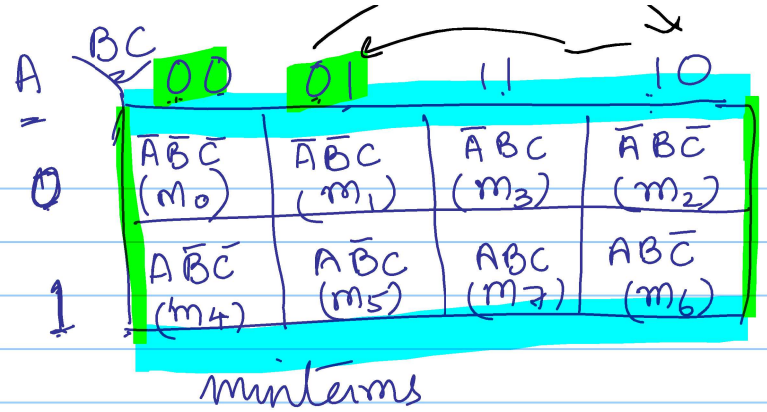


Three-variable K-map

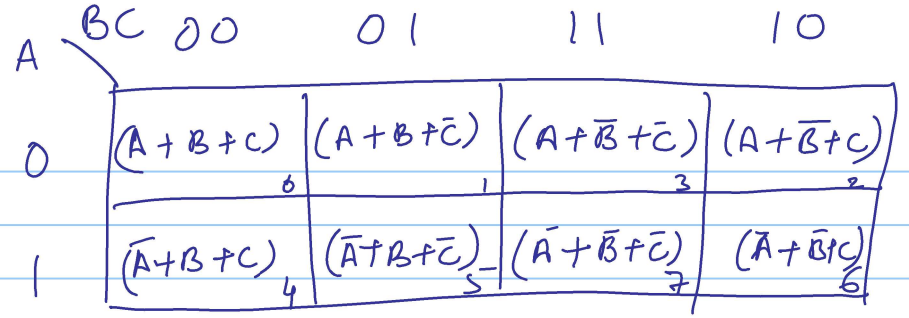
A function in 3-variables (A, B, C) SOP form.

	A	B	C		SOP	minterms.	Maxterms	POS
0	0	0	0	→	$\bar{A}\bar{B}\bar{C}$ →	m_0	M_0 →	$(A+B+C)$
1	0	0	1	→	$\bar{A}\bar{B}C$ →	m_1	M_1 →	$(A+B+\bar{C})$
2	0	1	0	→	$\bar{A}B\bar{C}$ →	m_2	M_2 →	$(A+\bar{B}+C)$
3	0	1	1	→	$\bar{A}BC$ →	m_3	M_3 →	$(A+\bar{B}+\bar{C})$
4	1	0	0	→	$A\bar{B}\bar{C}$ →	m_4	M_4 →	$(\bar{A}+B+C)$
5	1	0	1	→	$A\bar{B}C$ →	m_5	M_5 →	$(\bar{A}+B+\bar{C})$
6	1	1	0	→	$AB\bar{C}$ →	m_6	M_6 →	$(\bar{A}+\bar{B}+C)$
7	1	1	1	→	ABC —	m_7	M_7 →	$(\bar{A}+\bar{B}+\bar{C})$

A 3-variable K-map will have $2^3 = 8$ squares or cells.

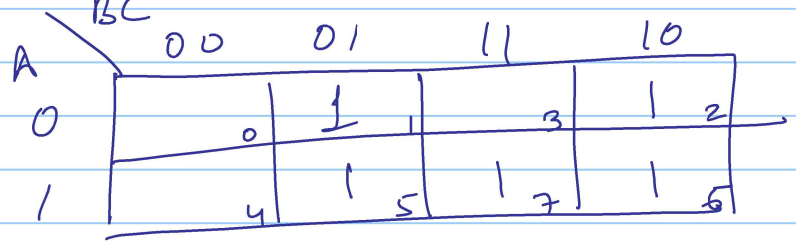


minterms



maxterms.

Ques $f = \underbrace{\bar{A}\bar{B}\bar{C}}_{m_1} + \underbrace{A\bar{B}C}_{m_5} + \underbrace{\bar{A}B\bar{C}}_{m_2} + \underbrace{AB\bar{C}}_{m_6} + \underbrace{ABC}_{m_7}$



Ques $f = \underset{M_0}{(A+B+C)}\,\underset{M_5}{(\bar{A}+B+\bar{C})}\,\underset{M_7}{(\bar{A}+\bar{B}+\bar{C})}\,\underset{M_3}{(A+\bar{B}+\bar{C})}\,\underset{M_6}{(\bar{A}+\bar{B}+C)}$

A \ BC	00	01	11	10
0	0 (0)	1 (1)	0 (3)	1 (2)
1	1 (4)	0 (5)	0 (7)	0 (6)

Minimization of SOP and POS expressions

Combining of these adjacent squares in a k-map containing 1s or (0s) for the purpose of simplification of a SOP (or POS) form is called looping

Pair - 2, Quad - 4, Octo - 8

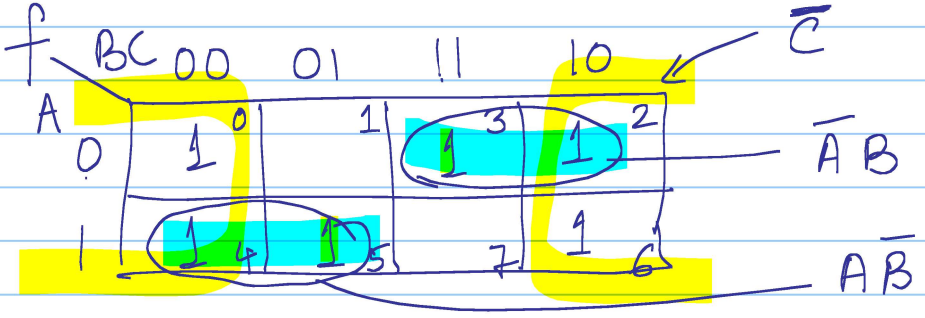
Ques $f = m_0 + m_1 + m_2 + m_3 = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC$

(A, B, C)

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

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

Ques $f = \sum m(0, 2, 3, 4, 5, 6)$ reduce and implement



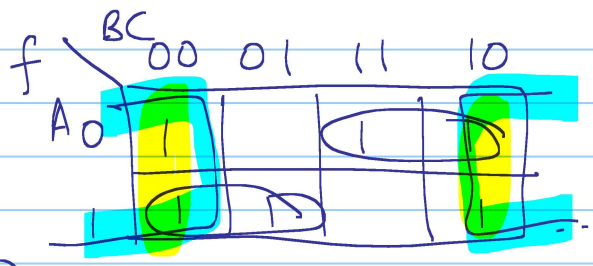
8, 4, 2

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

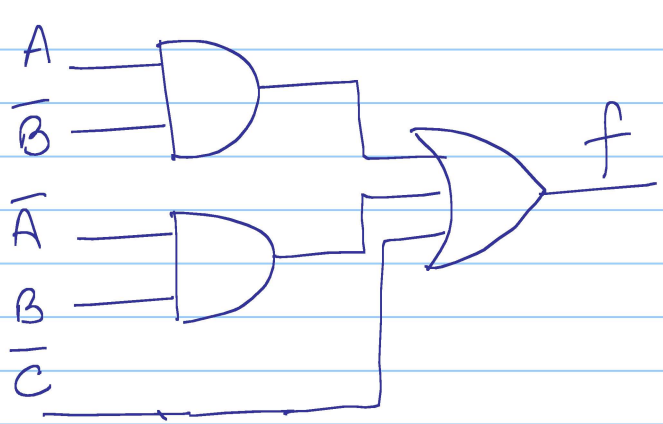
Minimal SOP expression

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

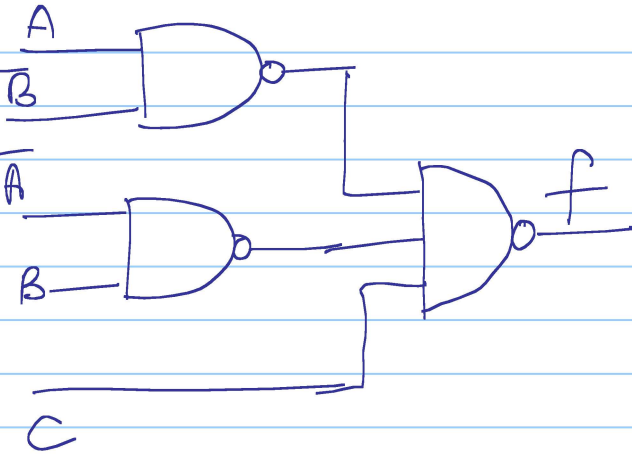
$$= \overline{\overline{A\bar{B}} \cdot \overline{\bar{A}B} \cdot C}$$



(a) K-map.



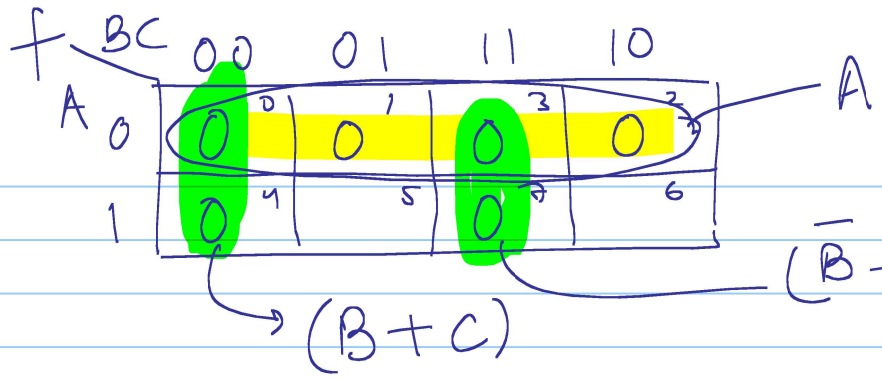
(b) AOI logic



(c) NAND logic

Ques: Reduce the exp. $f = \Pi M(0,1,2,3,4,7)$ using mapping and implement it in AOI logic and in NOR logic.

AOI → AND- OR- INVERT logic.

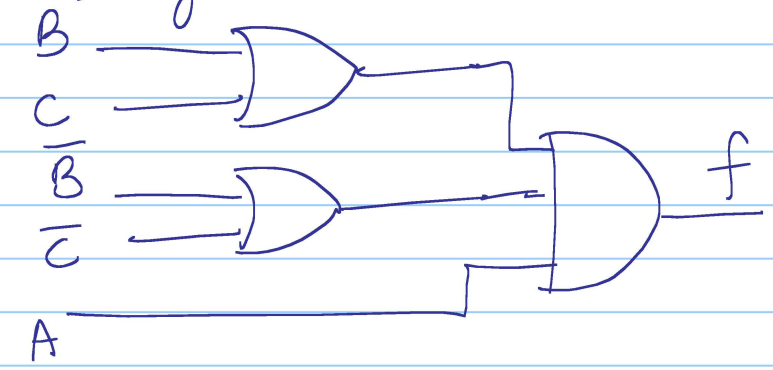


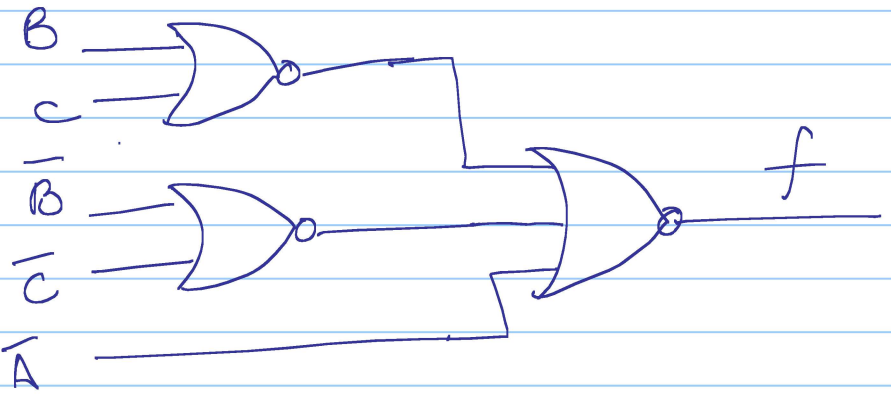
$$f = (B+C)(\bar{B}+\bar{C})A$$

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

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

AOI logic



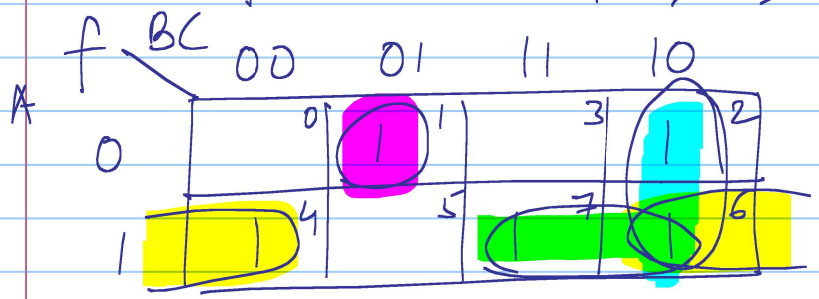


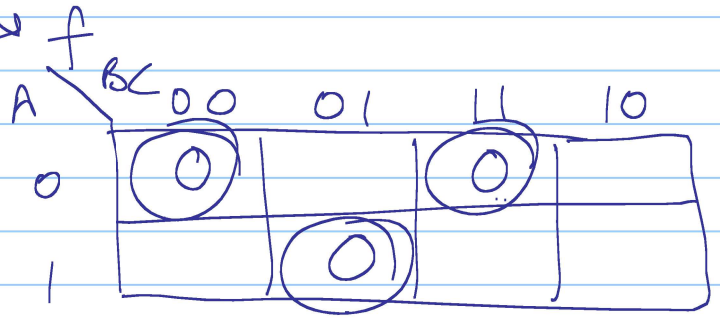
Ques $f = \sum m(1, 2, 4, 6, 7)$ and implement using universal gates.

→ SOP

NOR

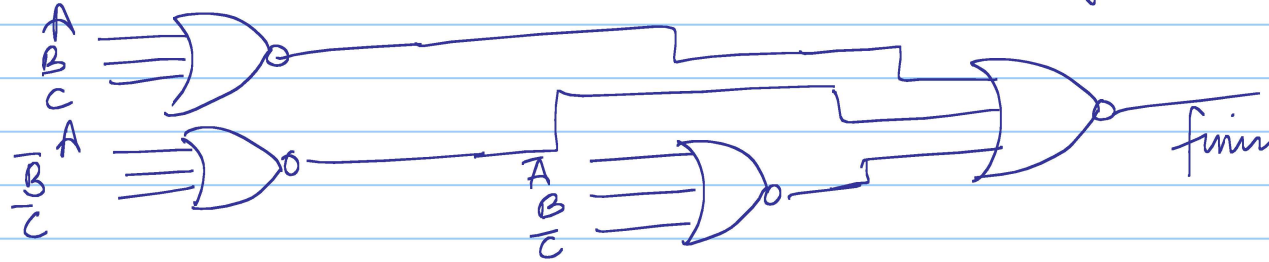
$f = \prod M(0, 3, 5)$ (POS) →





$f_{min} = A\bar{C} + AB + B\bar{C} + \bar{A}\bar{B}C$

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



logic diagram.

Four variable K-map

(A, B, C, D) can have $2^4 = 16$ possible combinations of i/p var.

SOP

$\bar{A}\bar{B}\bar{C}\bar{D} \rightarrow m_0$

$\bar{A}\bar{B}\bar{C}D \rightarrow m_1$

$\vdots$

$ABCD \rightarrow m_{15}$

POS

$A+B+C+D \rightarrow M_0$

$A+B+C+\bar{D} \rightarrow M_1$

$\vdots$

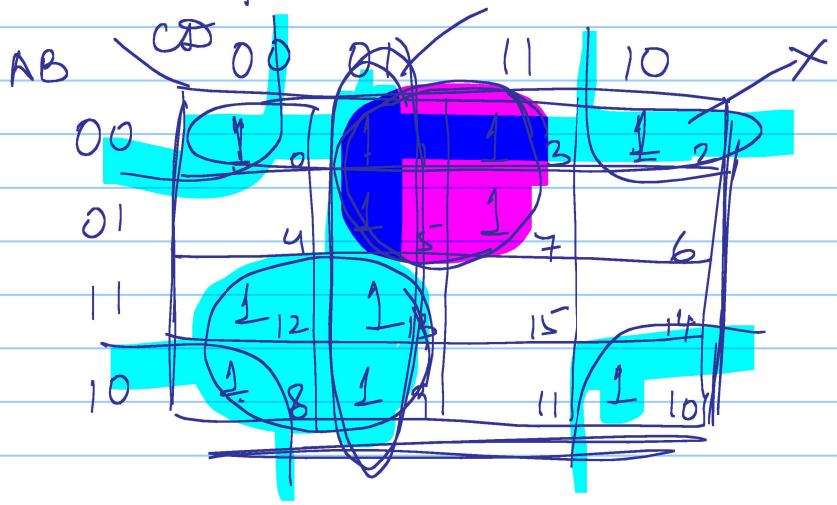
$\bar{A}+\bar{B}+\bar{C}+\bar{D} \rightarrow M_{15}$

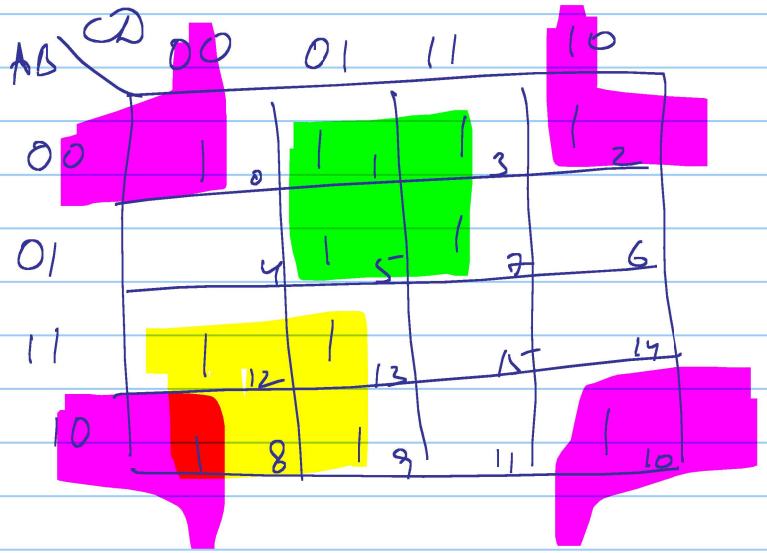
f / AB \ CD	00	01	11	10
00	$\bar{A}\bar{B}\bar{C}\bar{D}$ m_0 0	$\bar{A}\bar{B}\bar{C}D$ 1	$\bar{A}\bar{B}CD$ 3	$\bar{A}\bar{B}C\bar{D}$ 2
01	$\bar{A}B\bar{C}\bar{D}$ 4	$\bar{A}B\bar{C}D$ 5	$\bar{A}BCD$ 7	$\bar{A}BC\bar{D}$ 6
11	$AB\bar{C}\bar{D}$ 12	$AB\bar{C}D$ 13	$ABCD$ 15	$ABC\bar{D}$ 14
10	$A\bar{B}\bar{C}\bar{D}$ 8	$A\bar{B}\bar{C}D$ 9	$A\bar{B}CD$ 11	$A\bar{B}C\bar{D}$ 10

AB \ CD	00	01	11	10
00	$(A+B+C+D)_0$	$A+B+C+\bar{D}$ 1	$A+B+\bar{C}+\bar{D}_3$	$A+B+\bar{C}+D_2$
01	$A+\bar{B}+C+D_4$	$A+\bar{B}+C+\bar{D}_5$	$A+\bar{B}+\bar{C}+\bar{D}_7$	$A+\bar{B}+\bar{C}+D_6$
11	$\bar{A}+\bar{B}+C+D_{12}$	$\bar{A}+\bar{B}+C+\bar{D}_{13}$	$\bar{A}+\bar{B}+\bar{C}+\bar{D}_{15}$	$\bar{A}+\bar{B}+\bar{C}+D_{14}$
10	$\bar{A}+B+C+D_8$	$\bar{A}+B+C+\bar{D}_9$	$\bar{A}+B+\bar{C}+\bar{D}_{11}$	$\bar{A}+B+\bar{C}+D_{10}$

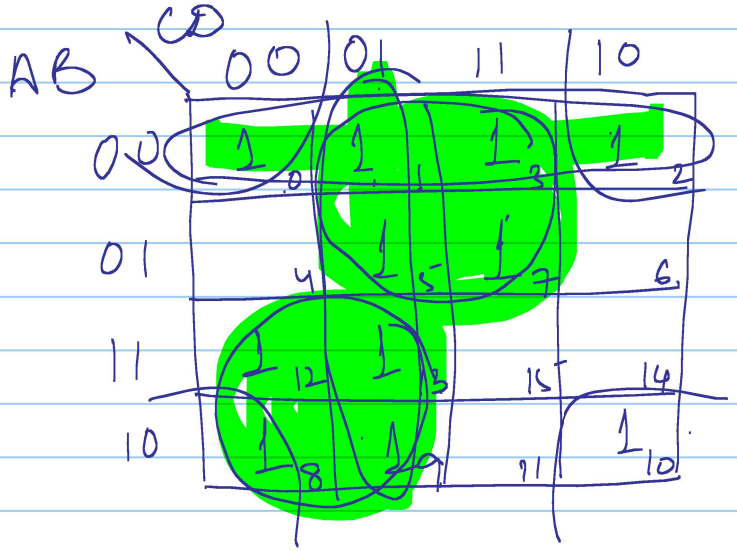
Ques: Reduce using mapping $f = \sum m(0, 1, 2, 3, 5, 7, 8, 9, 10, 12, 13)$ and implement the real minimal expression in universal logic.

Sol $f = \prod M(4, 6, 11, 14, 15)$

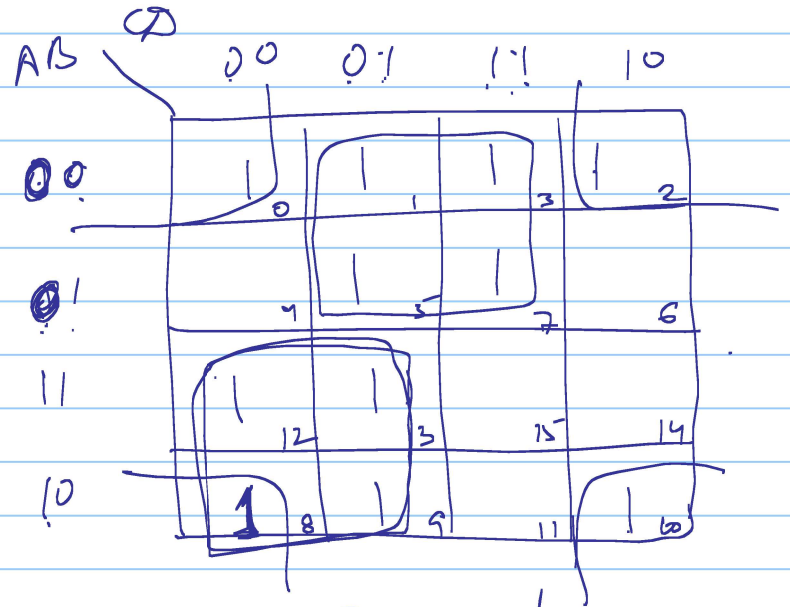




$$f = \sum m(0, 1, 2, 3, 5, 7, 8, 9, 10, 12, 13)$$

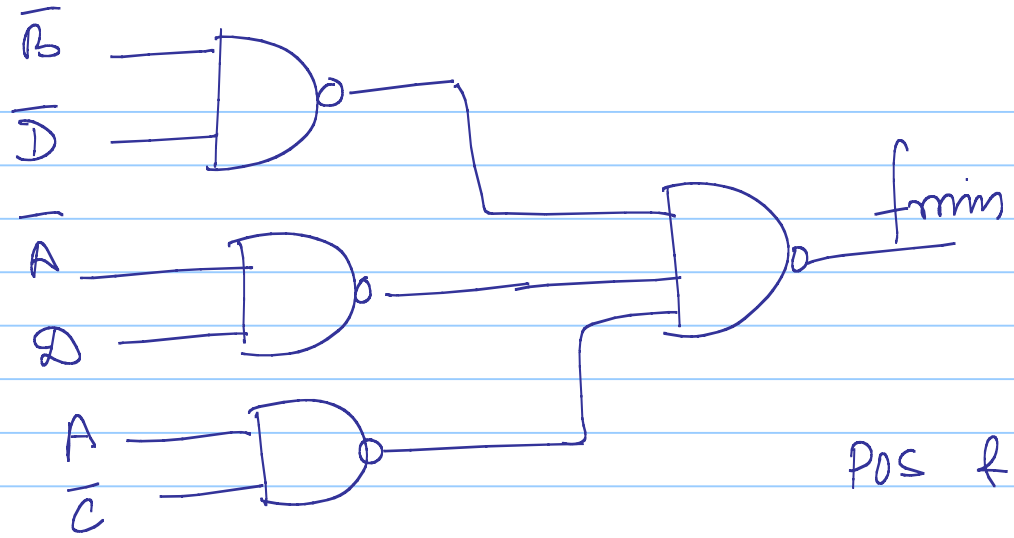


5-quads



3-quads.

$$f_{min} = \bar{B}\bar{D} + \bar{A}D + A\bar{C}$$



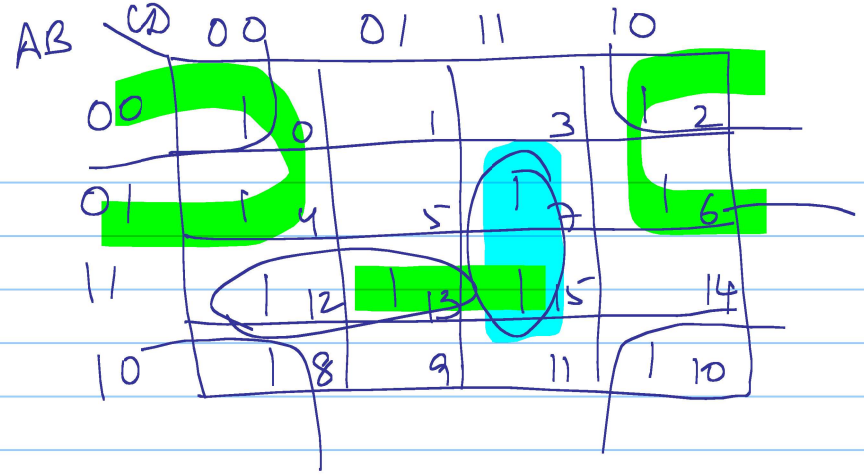
$$\overline{\overline{f_{min}}} = \overline{\overline{\bar{B}\bar{D} + \bar{A}D + A\bar{C}}}$$

$$f_{min} = \overline{\bar{B}\bar{D} \cdot \bar{A}D \cdot A\bar{C}}$$

POS & NOR implementation → H.W

Ques $f = \sum m(0, 2, 4, 6, 7, 8, 10, 12, 13, 15)$

POS form $f = \Pi M(1, 3, 5, 9, 11, 14)$

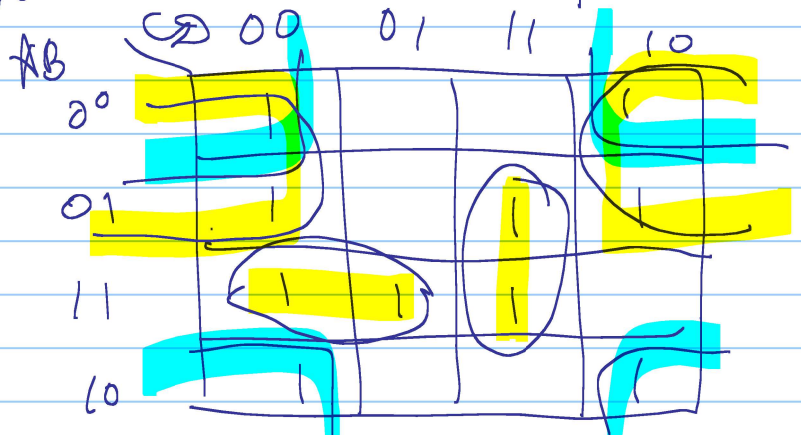


$$fun = \bar{C}\bar{D} + \bar{B}\bar{D} + ABD + BC\bar{D} + \bar{A}\bar{D}$$

$$fun = \bar{A}\bar{D} + \bar{C}\bar{D} + \bar{B}\bar{D} + ABD + BC\bar{D}$$

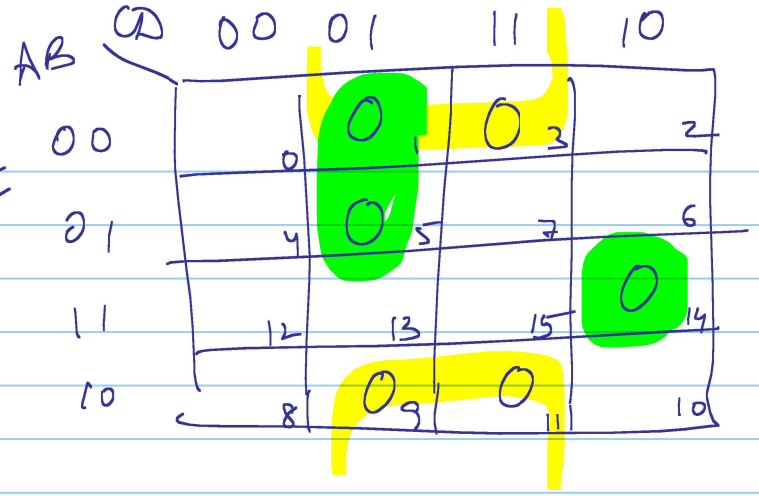
SOP minimal expression

SOP minimal fun =

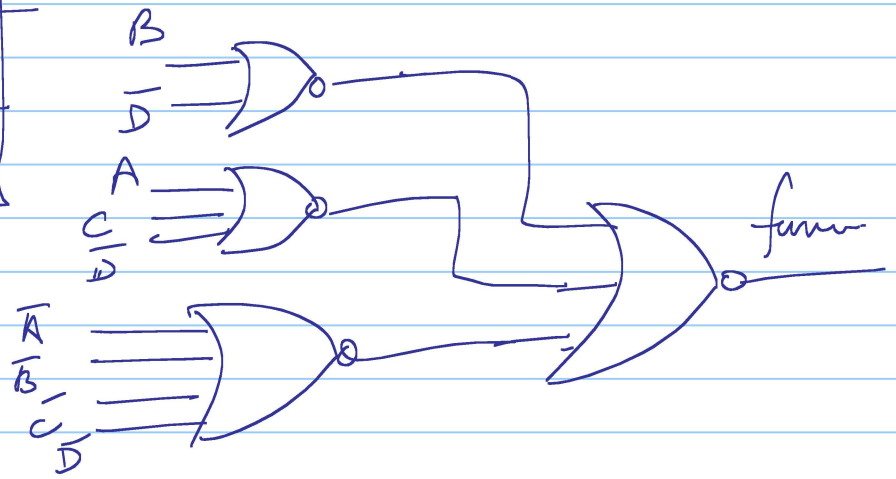


$$fun = \bar{A}\bar{D} + \bar{B}\bar{D} + AB\bar{C} + BC\bar{D}$$

POS



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



* Prime Implicants, Essential Prime implicants, Redundant Prime Implicants and Selective Prime implicants.

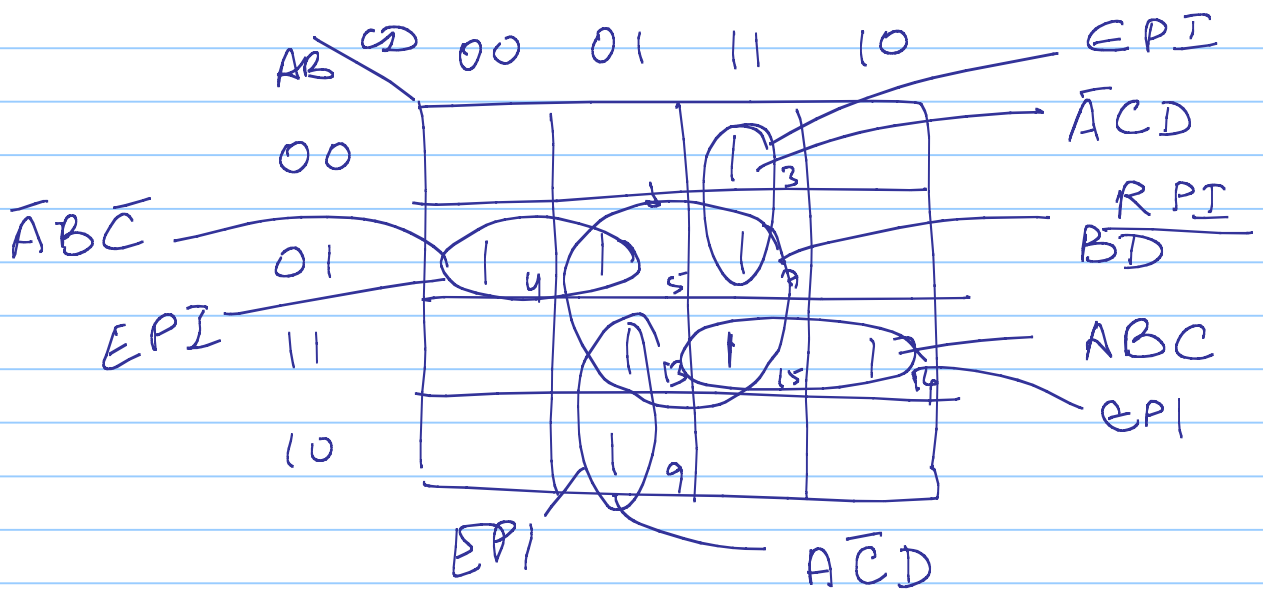
Each square or rectangle made up of bunch of adjacent minterms is called a SUBCUBE.

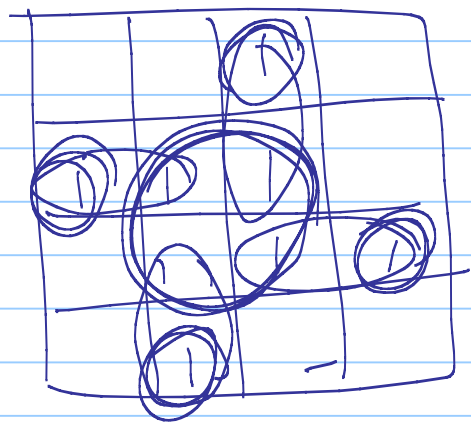
Each of these subcube is called a prime implicant (PI).

the prime implicant which contain atleast one 1 which cannot be covered by other prime implicant is called an ESSENTIAL PRIME IMPLICANT (EPI)

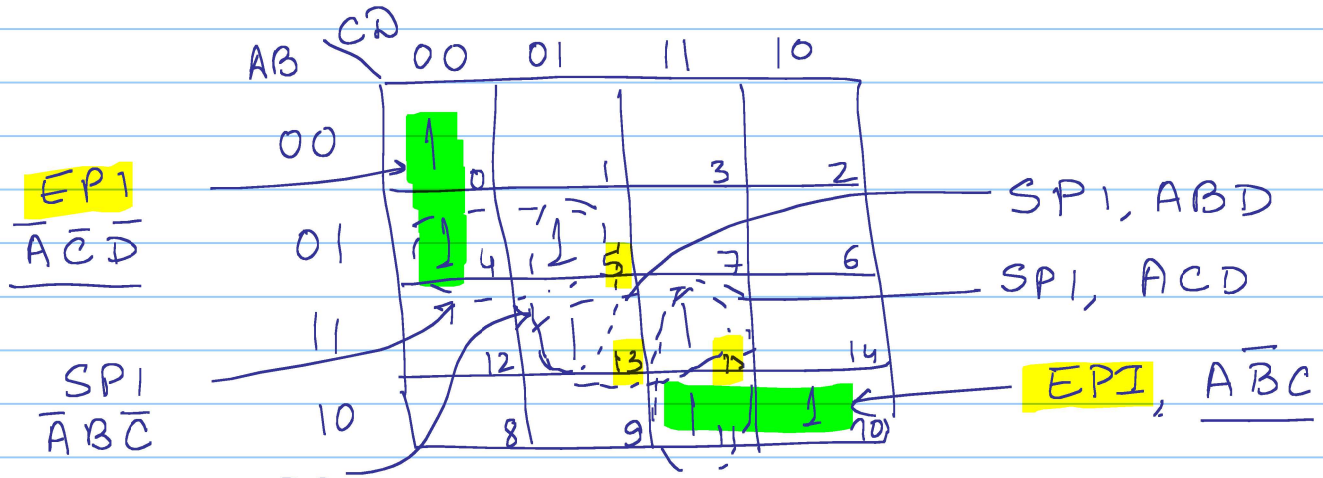
The prime implicant whose each 1 is covered atleas by one EPI is called or redundant prime implicant (RPI).

A prime implicant which is neither an essential prime implicant nor a redundant prime implicant is called a selective prime implicant (SPI).





Q $f(A,B,C,D) = \sum m(0,4,5,10,11,13,15)$



Essential and prime implicants

"Minimal SOP form of a func" need not be unique"

The minimal SOP form is obtained by including two EPIs and

selecting a set of SPIs to cover the remaining SPI, (5, 13, 15)

ⓐ (4, 5) and (13, 15) → $\bar{A}B\bar{C} + ABD$

ⓑ (5, 13) and (13, 15) → $B\bar{C}D + ABD$

ⓒ (5, 13) and (15, 11) → $B\bar{C}D + ACD$

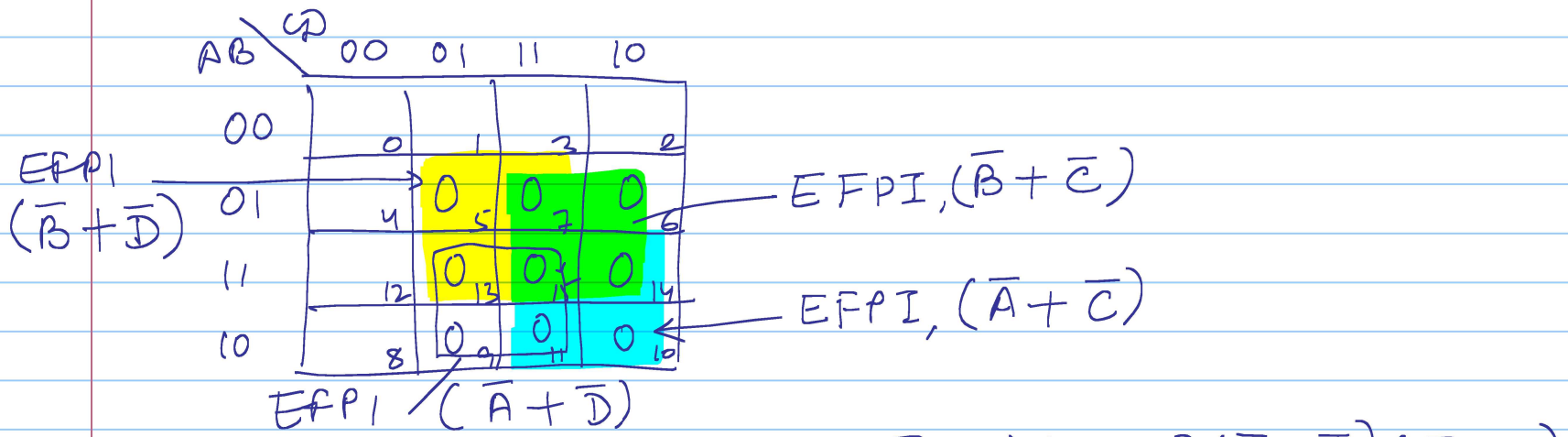
$F(A, B, C, D) = \bar{A}\bar{C}\bar{D} + A\bar{B}C + ⓐ + ⓑ + ⓒ$ } This func" has three different MSP form.

(4) False Prime Implicants, Essential False Prime Implicants, Redundant False prime Implicants and Selective False Prime Implicants

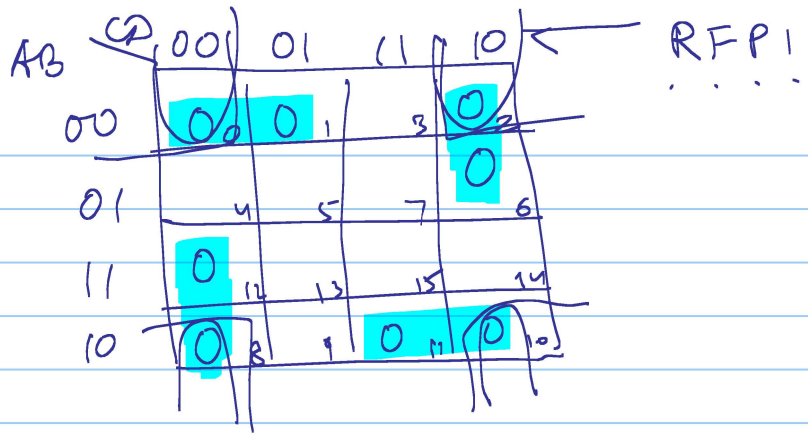
The maxterms are called as "false minterms".

The func^{n} $F(A,B,C,D) = \sum m(0,1,2,3,4,8,12)$

$= \Pi M(5,6,7,9,10,11,13,14,15)$



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



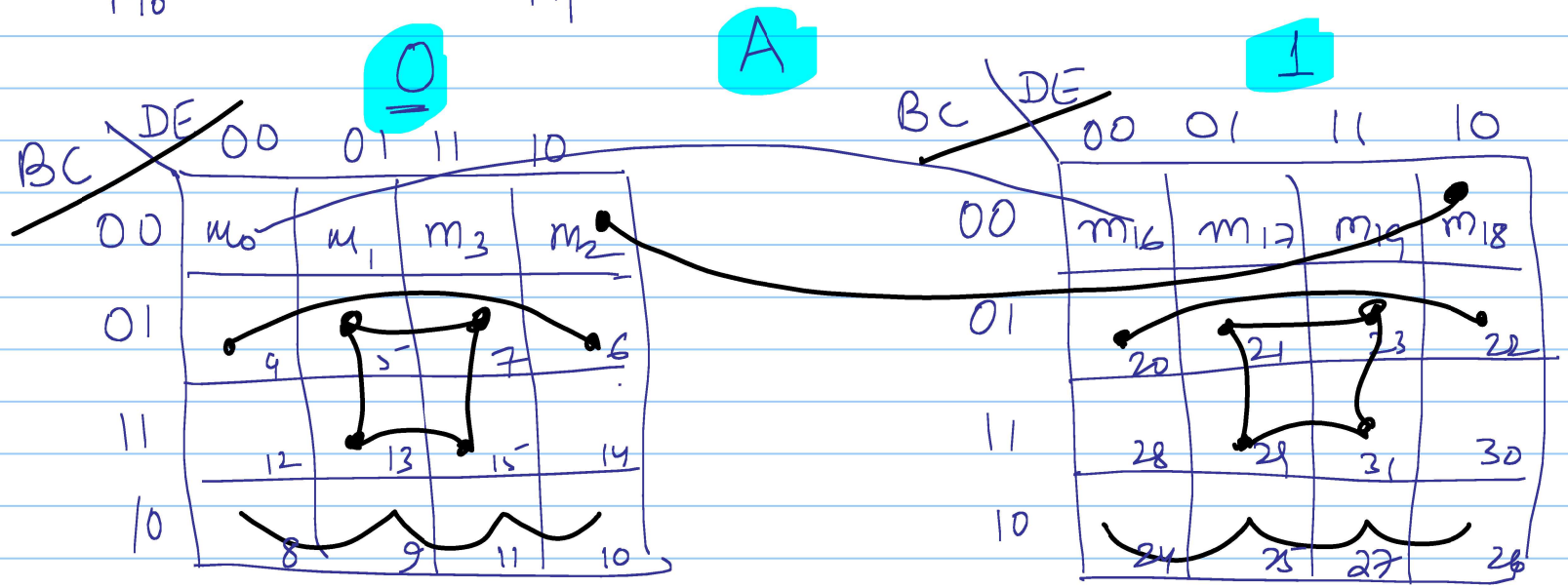
Five variable K-map $F(A,B,C,D,E) = 2^5 = \underline{32}$ possible comb.

$\bar{A}\bar{B}\bar{C}\bar{D}\bar{E}$, $\bar{A}\bar{B}\bar{C}\bar{D}E$, ... , $ABCDE$. SOP

$\downarrow$ m_0 $\quad$ $\downarrow$ m_1 $\quad$ $\downarrow$ m_{31}

$A+B+C+D+E$, $A+B+C+D+\bar{E}$, ..., $\bar{A}+\bar{B}+\bar{C}+\bar{D}+\bar{E}$ POS.

M_0 M_1 M_{31}



ABCDE

0

2-sq. 4-sq, 8-sq, 16 sq, 32-sq.

Don't Care Combinations

The value of o/p is unspecified because the i/p combination are invalid or because the precise value of the o/p is of no consequence.

A	B	C	o/p
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	X
1	1	1	X

not valid

X → don't care comb^n or optional comb^n.

0 1 → unspecified

Invalid combinations in excess-3 code

A	B	C	D	
0	0	0	0	don't care
0	0	0	1	
0	0	1	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

8421 code

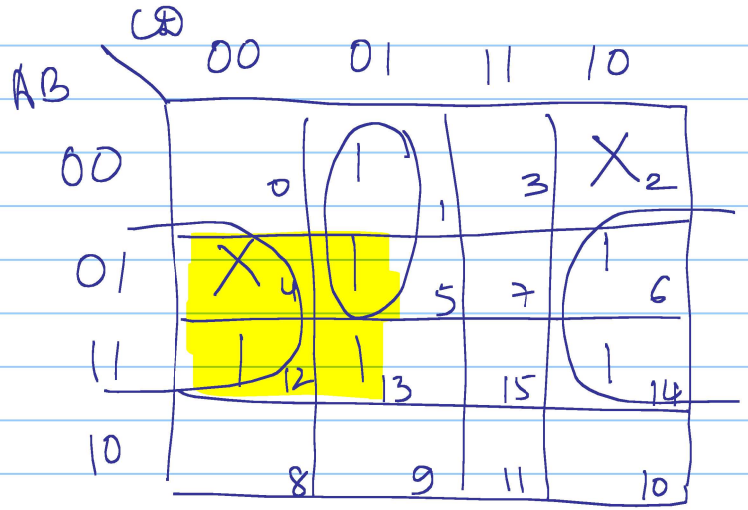
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

X, d or ϕ → don't care cond^{n}.

Ques Reduce the expression $f = \sum m(1, 5, 6, 12, 13, 14) + d(2, 4)$ and implement the real minimal expression in universal logic.

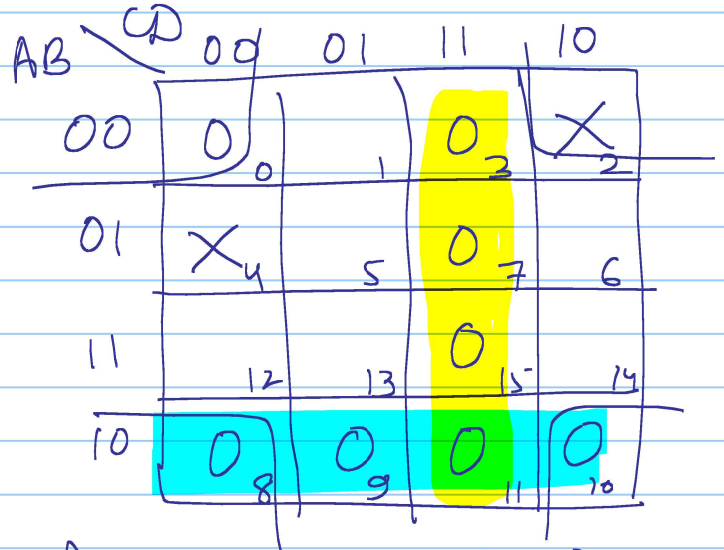
(b) The given expression written the POS form is

$$f = \Pi M(0, 3, 7, 8, 9, 10, 11, 15) \cdot \Pi d(2, 4)$$



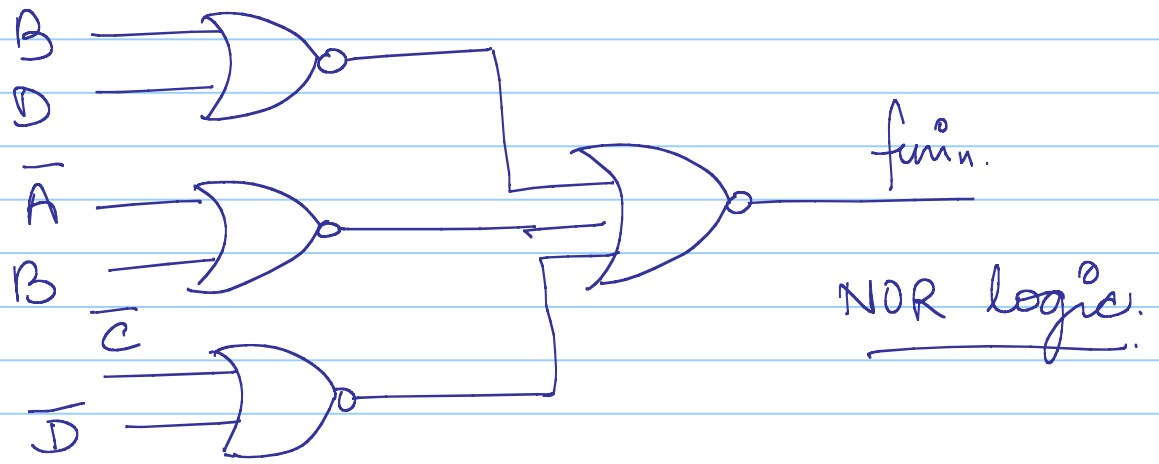
$$f_{min} = B\bar{C} + B\bar{D} + \bar{A}\bar{C}D$$

SOP K-map.



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

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



Ques Minimize the following expressions using k-map

a) $F(A, B, C, D) = \sum m(1, 4, 7, 10, 13) + \sum d(5, 14, 15)$

b) $F(A, B, C, D) = \sum m(4, 5, 7, 12, 14, 15) + \sum d(3, 8, 10)$

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

d) $F(A, B, C, D) = \sum m(0, 1, 2, 3, 4, 5) + d(10, 11, 12, 13, 14, 15$

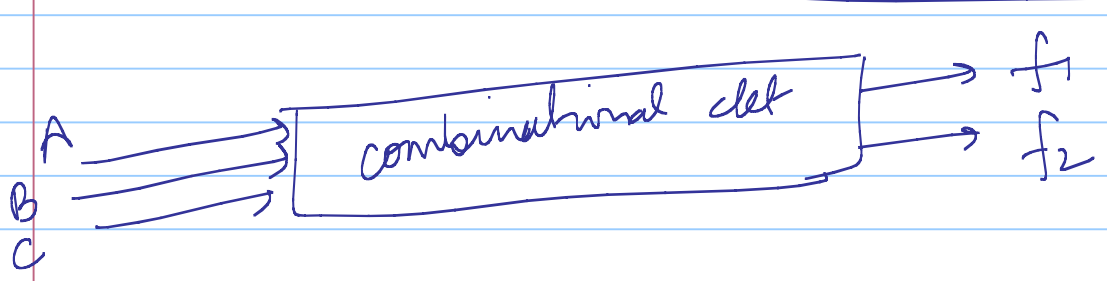
Ques: (a) Design a logic ckt using minimum no. of basic gates for $f = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BC\bar{D} + A\bar{B}\bar{C}D + AB\bar{C}\bar{D} + AB\bar{C}D + ABC\bar{D}$

(b) Reduce the following exp. $F = \bar{B}\bar{A} + \bar{A}B + A\bar{B}$

(c) Find the O/P of a 4-variable K-map when all the cells are filled with logic LOW.

(d) Indicate the essential false prime implicants in the K-map for part (c).

Minimization of multiple O/P circuits



$f_1(A,B,C) = \sum m(0,1,2,5,6,7)$

$f_2(A,B,C) = \sum m(2,4,5,6)$

Ques Minimize and implement the following multiple o/p func^n

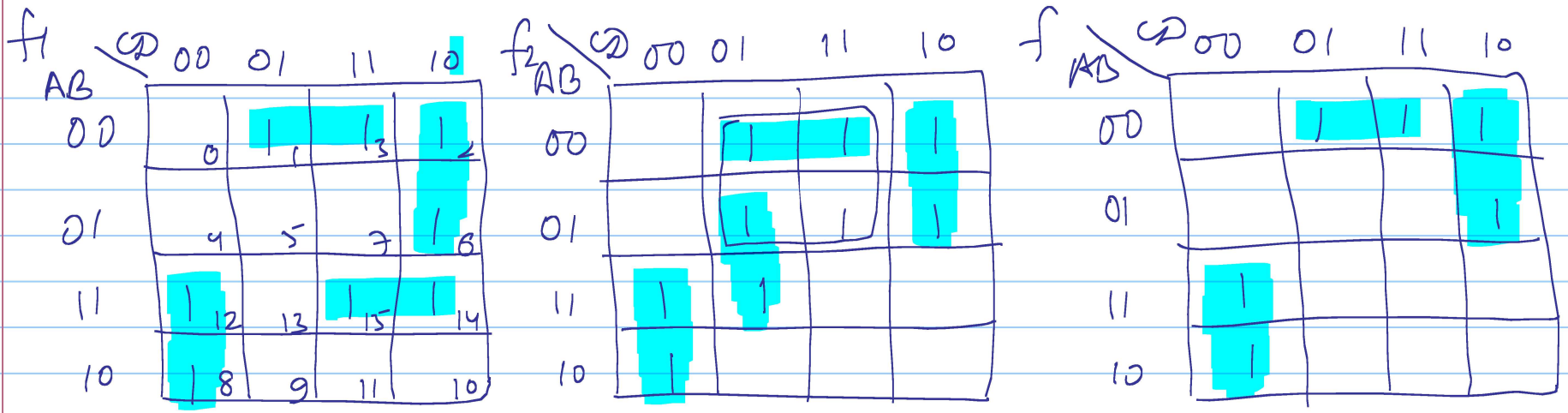
$$f_1 = \sum m(1, 2, 3, 6, 8, 12, 14, 15)$$

$$f_2 = \prod M(0, 4, 9, 10, 11, 14, 15)$$

$$f_2 = \sum m(1, 2, 3, 5, 6, 7, 8, 12, 13)$$

First form a function f with the minterms common to both the functions, i.e, $f = f_1 \cdot f_2$

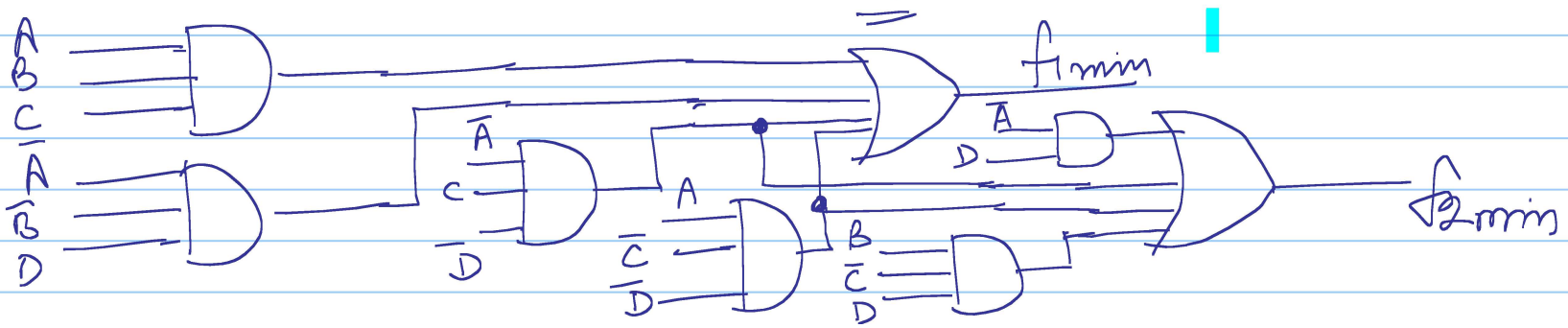
$$f = f_1 \cdot f_2 = \sum m(1, 2, 3, 6, 8, 12)$$



$f_1 = A\bar{C}\bar{D} + \bar{A}C\bar{D} + \bar{A}\bar{B}D + ABC$

$f_2 = A\bar{C}\bar{D} + \bar{A}C\bar{D} + B\bar{C}D + \bar{A}D$

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



Ques Minize the following multiple o/p func^n

$$f_1 = \sum m(0,2,6,10,11,12,13) + d(3,4,5,14,15)$$

$$f_2 = \sum m(1,2,6,7,8,13,14,15) + d(3,5,12)$$

Generating minterms

f_1	f_2	$f = f_1 \cdot f_2$
0	0	0
0	1	0
1	0	0
0	X	0
X	0	0
1	X	1
X	1	1
1	1	1
X	X	X

Note Title 5/28/2021

Limitations of Karnaugh (K)- map

- Tedious beyond 7 or more variables
- Manual technique and simplification process is heavily dependent on the human abilities.
- It cannot be programmed.

Quine and McCluskey method / Tabular method to simplify a Boolean expressions.

Implementation of Logic functions

(1) Two-level implementation

Each i/p has to pass through only two gates to reach the o/p this is called as two-level implementation.

Both SOP & POS form result in two level logic

AND & and OR gates or only NAND or only NOR

The implementation of Boolean exp. with only NAND gates requires the func' in SOP form

NOR ⟶ POS

$$F = AB + CD$$

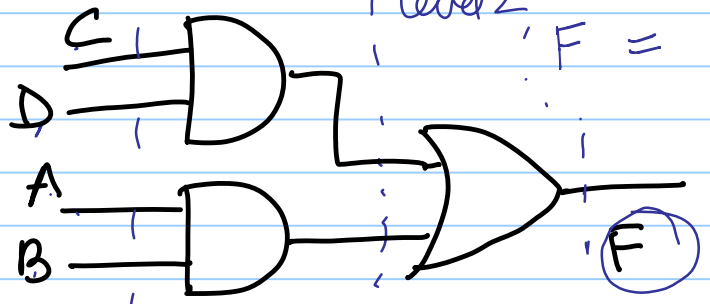
ⓐ AND-OR logic and ⓑ NAND-NAND logic

$F = AB + CD$

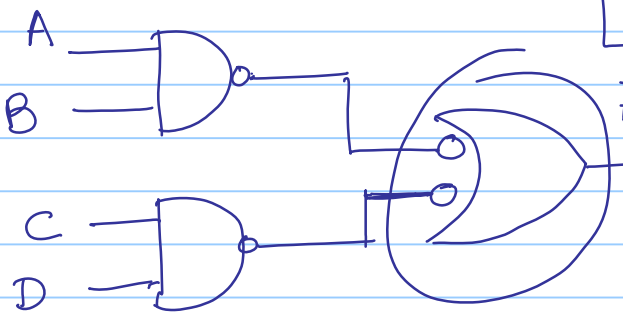
$\overline{\overline{F}} = \overline{AB + CD}$

$F = \overline{\overline{AB} \cdot \overline{CD}}$

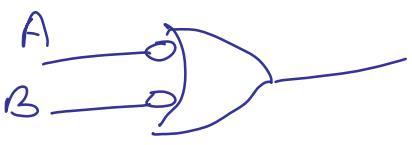
Level 1 | Level 2



AND-OR gates



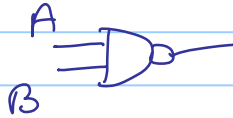
NAND gates

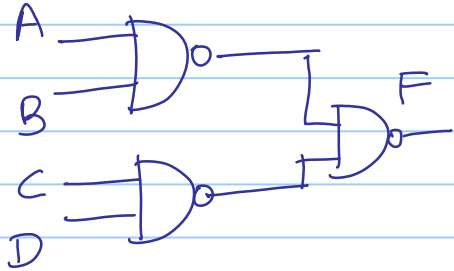


$f = \bar{A} + \bar{B}$

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

$f = \overline{A \cdot B}$





NAND gates

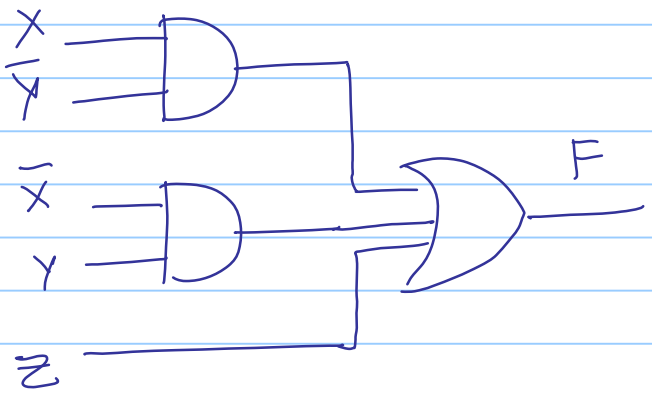
The implementation of the func^{4}

$F = X\bar{Y} + \bar{X}Y + Z$

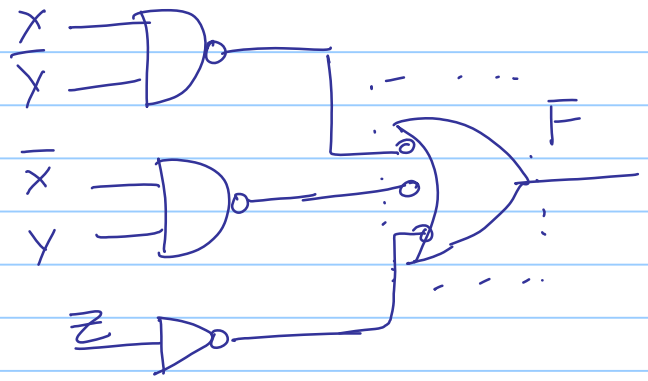
AND-OR logic and NAND-NAND logic

$$F = X\bar{Y} + \bar{X}Y + Z$$

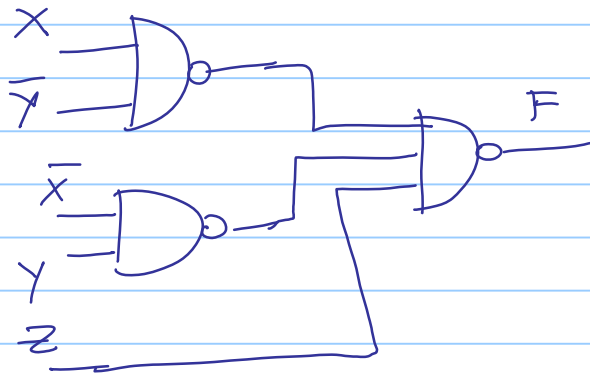
$$\bar{\bar{F}} = \overline{\overline{X\bar{Y} + \bar{X}Y + Z}} = \overline{\overline{X\bar{Y}} \cdot \overline{\bar{X}Y} \cdot \overline{Z}}$$



AND-OR gates



NAND gates

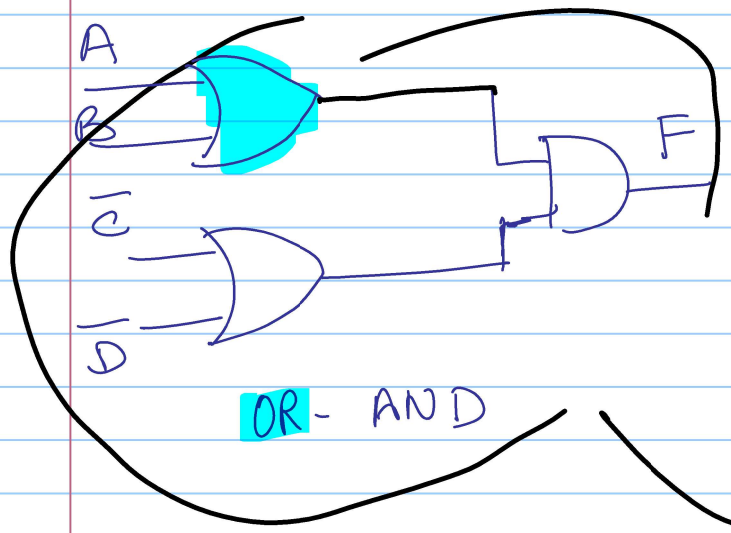


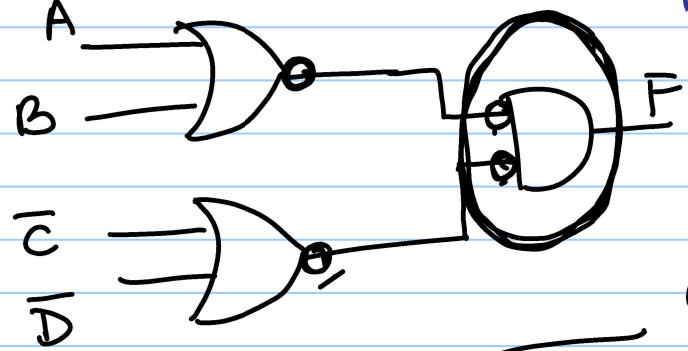
NAND gates

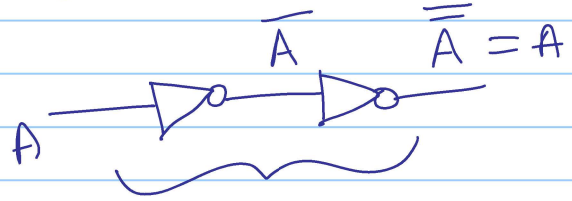
The implementation of Boolean exp with only NOR gates requires that the func^n be in POS form.

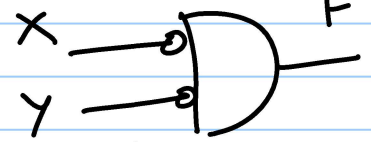
$$F = (A + B)\cdot(\bar{C} + \bar{D})$$

@ OR-AND and (b) NOR-NOR logic









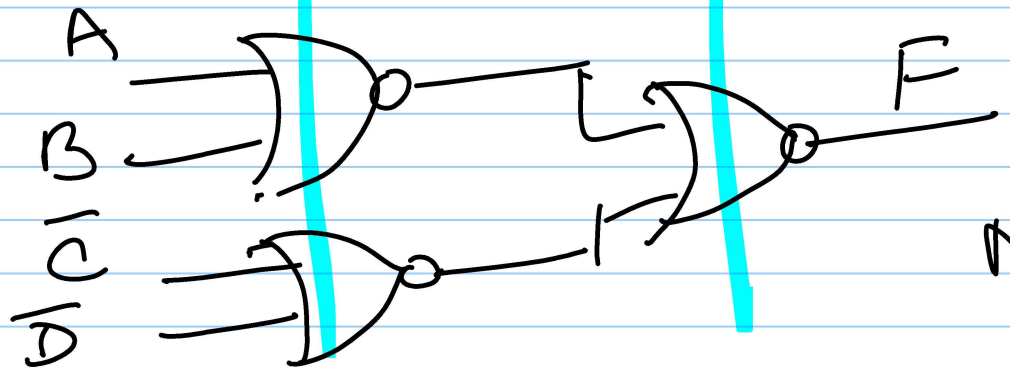
$\bar{F} = \bar{X}\,\bar{Y}$

$F = X + Y$

$$F = \overline{X} \cdot \overline{Y}$$

$$F = F = \overline{\overline{X} \cdot \overline{Y}} = \overline{\overline{X} + \overline{Y}}$$

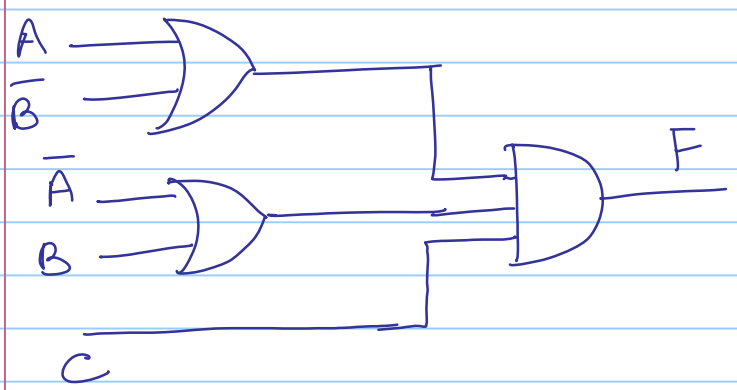
$$F = \overline{X + Y}$$

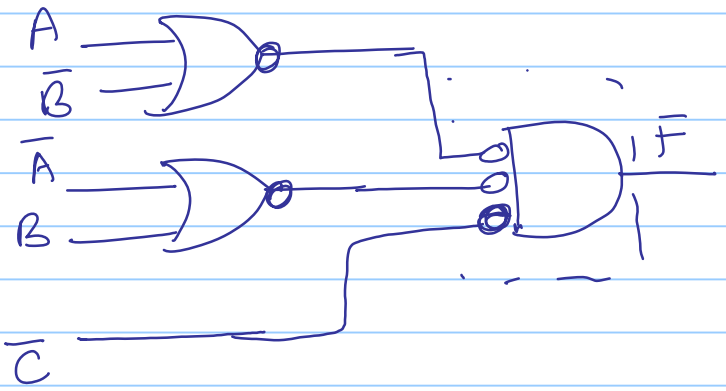


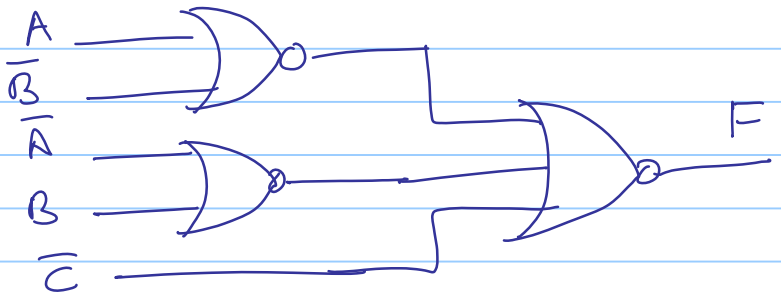
NOR - NOR implementation

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

(a) OR-AND logic (b) NOR logic







NOR - NOR logic.

NONDEGENERATE FORMS

i.e AND, OR, NAND and NOR

16 possible combinations when we assign one type of gate to one level and other type of gate to other level.

∴ Non degenerate forms are as follows

AND - OR	⟷	OR - AND
NAND - NAND	⟷	NOR - NOR
NOR - OR	⟷	NAND - AND
OR - NAND	⟷	AND - NOR.

← dual to each other

4-bit parallel subtractor

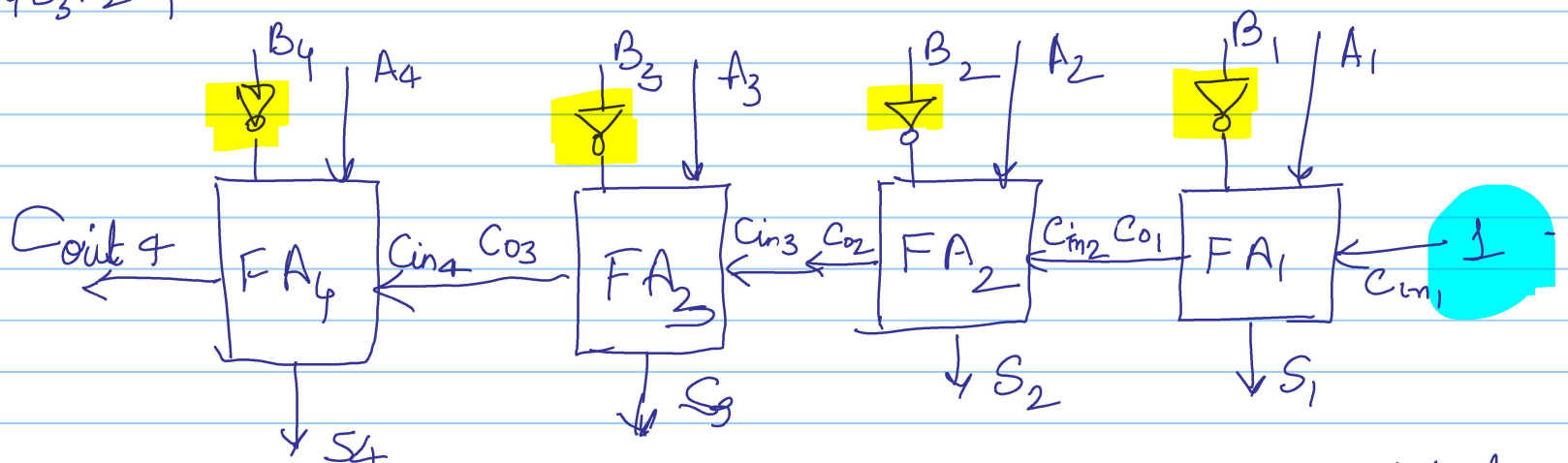
Note Title 6/8/2021

A − (B) = 2's complement of B and add with A.

↳ 1's complement of B and add 1 to LSB.

$A = A_4 A_3 A_2 A_1$

$B = B_4 B_3 B_2 B_1$



Logic diagram of a 4-bit parallel subtractor

Binary adder - subtractor

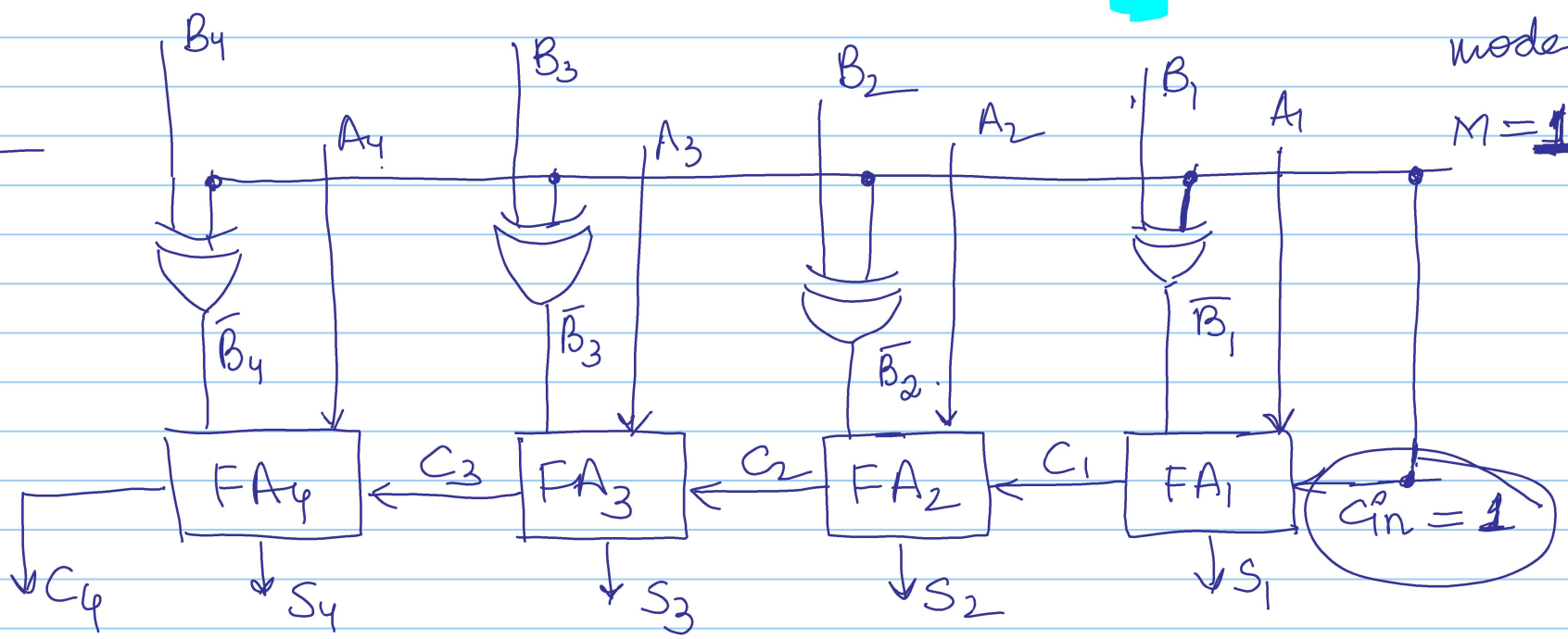
including X-OR gate with each full adder

A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

When A=0,
$B \oplus 0 = B$

M=1

mode i/p
M = 1



A - B = A + (2's complement of B)

A + (1's comp. of B + 1)

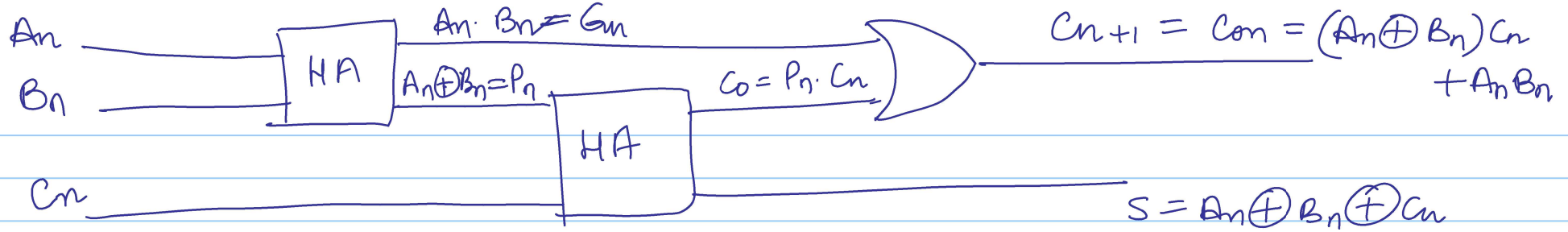
Logic diagram of a 4-bit binary adder-subtractor

Carry Look-ahead Adder (CLA)

1) Carry generate
2) Carry propagation func'.

$A = A_n \ldots A_4 A_3 A_2 A_1$

$B = B_n \ldots B_4 B_3 B_2 B_1$



A full adder (nth stage of a parallel adder)

A_n	B	C_n	C_o	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

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

A	B	C_{in}	Cout
→ 0	→ 1	0 ←	0 ←
→ 0	1	1 ←	1
1	0	0 ←	0 ←
→ 1	1	1 ←	1

$(A \oplus B) C_{in}$.

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

X-OR

$S_n = A_n \oplus B_n \oplus C_n = P_n \oplus C_n$ where $P_n = A_n \oplus B_n$

$Con = C_{n+1} = G_n + P_n \cdot C_n$ where $G_n = A_n \cdot B_n$

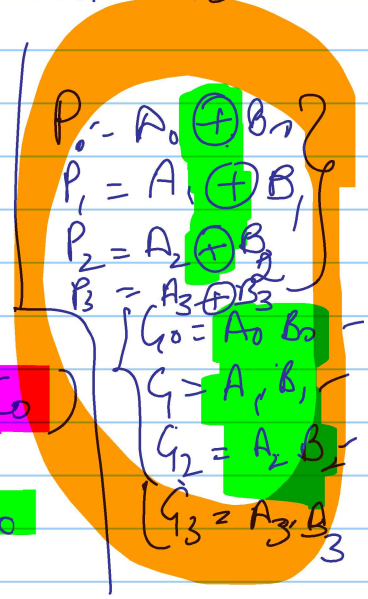
$C_1 = G_0 + P_0 \cdot C_0$ — (1)

$C_2 = G_1 + P_1 \cdot C_1 = G_1 + P_1 G_0 + P_1 P_0 C_0$

$C_3 = G_2 + P_2 C_2 = G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0$

$C_4 = G_3 + P_3 \cdot C_3 = G_3 + P_3 (G_2 + P_2 G_1 + P_2 P_1 G_0 + P_2 P_1 P_0 C_0)$

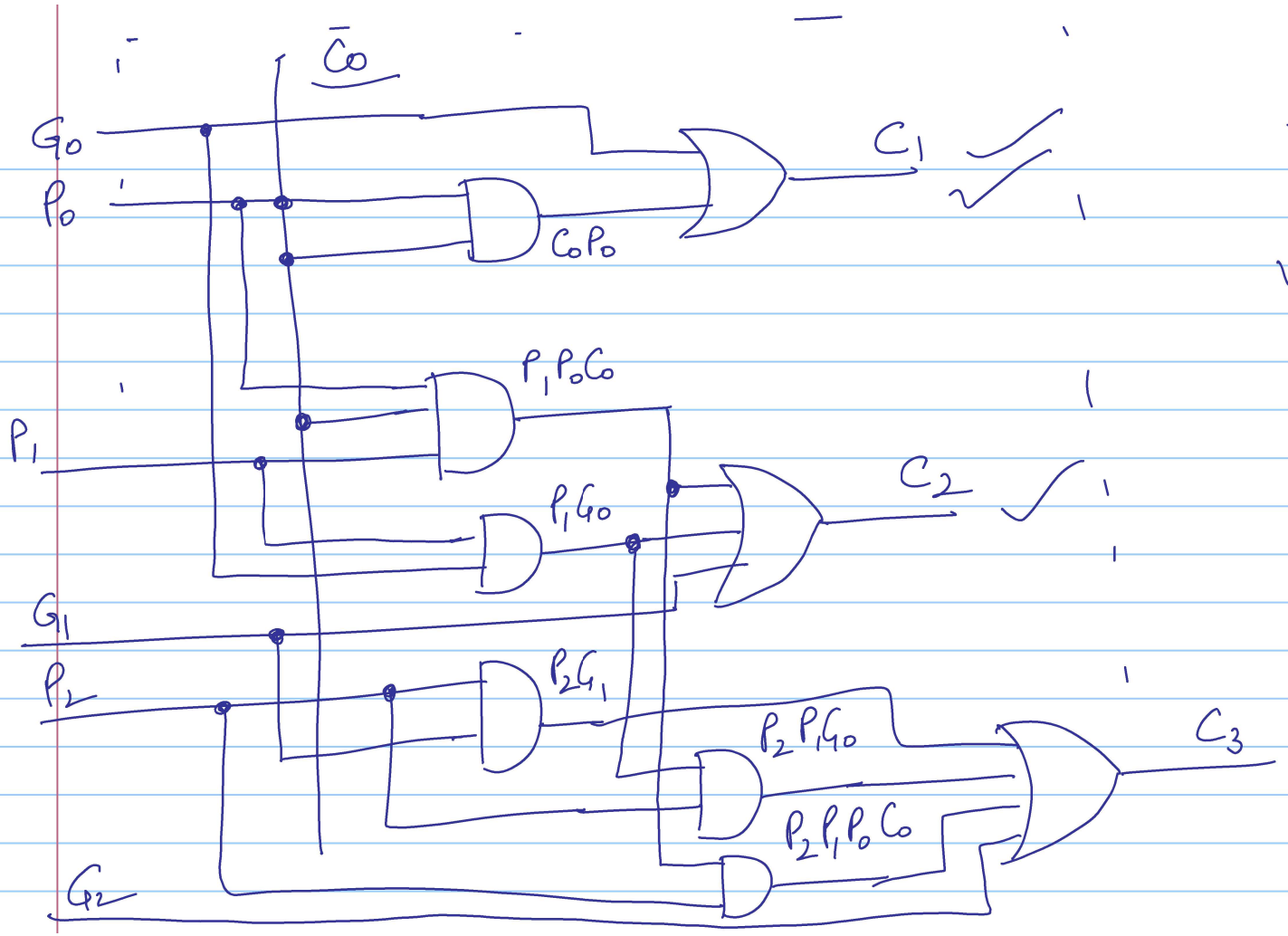
$= G_3 + P_3 G_2 + P_3 P_2 G_1 + P_3 P_2 P_1 G_0 + P_3 P_2 P_1 P_0 C_0$



for any n^{th} stage

$G_n = G_{n-1} + P_{n-1} G_{n-2} + P_{n-1} \cdot P_{n-2} G_{n-3} + \cdots + P_{n-1} \cdots P_0 C_0$

7483



$C_1 = G_0 + P_0C_0$

$C_2 = G_1 + P_1G_0 + P_1P_0C_0$

$C_3 = G_2 + P_2G_1 + P_2P_1G_0 + P_2P_1P_0C_0$

$C_4 = G_3 + P_3G_2 + P_3P_2G_1 + P_3P_2P_1G_0 + P_3P_2P_1P_0C_0$

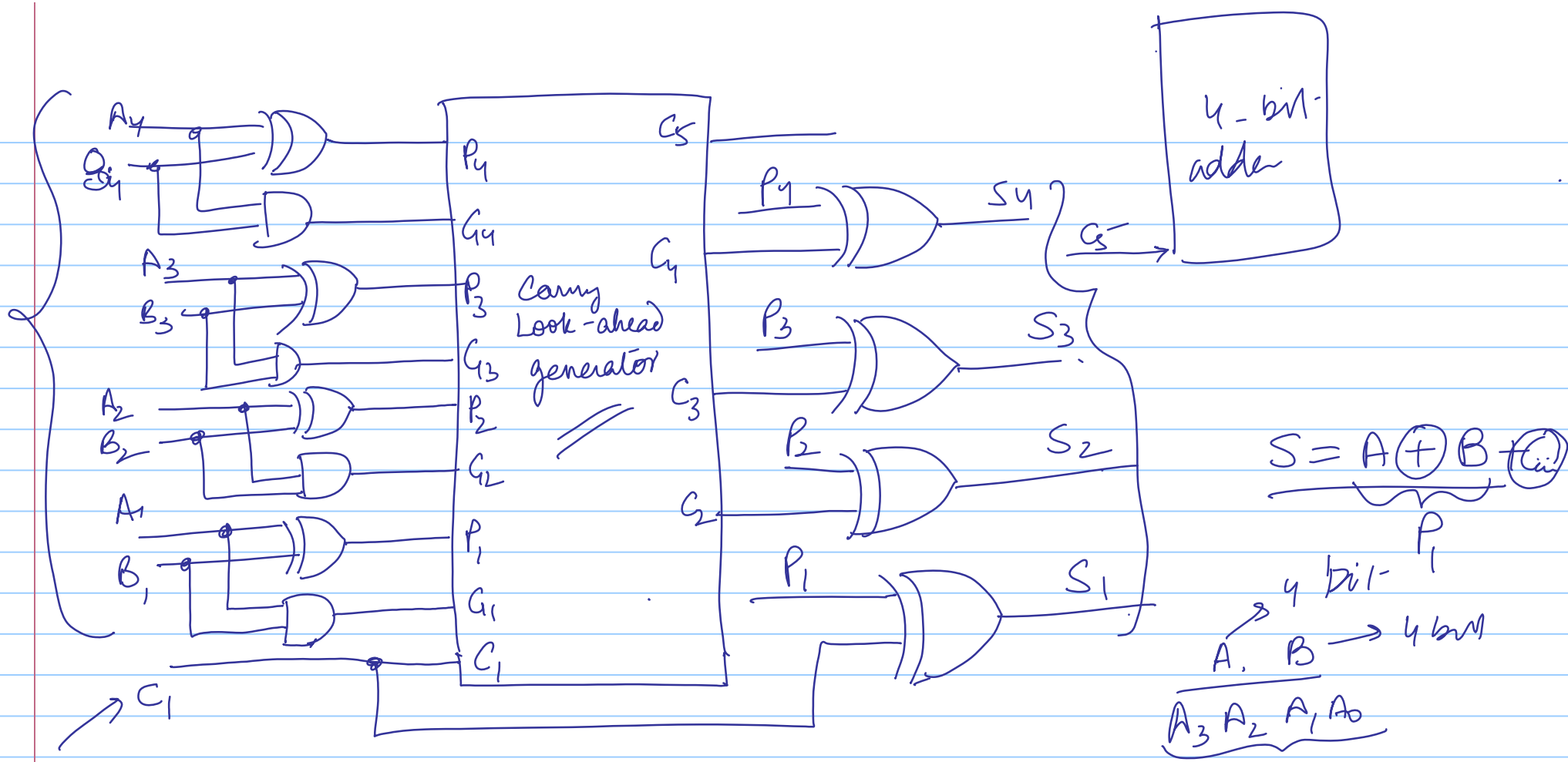
I/Ps: P_1, P_2, P_3, P_4

G_1, G_2, G_3, G_4

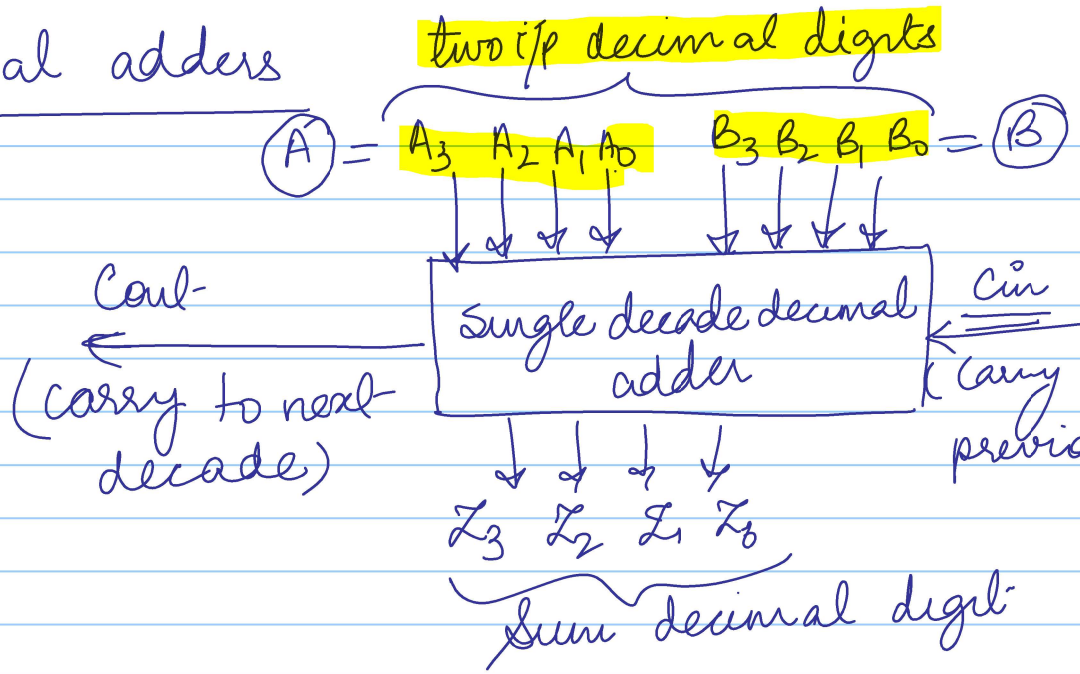
4-bit fast adder

7483

C_4


A4
B4
A3
B3
A2
B2
A1
B1
C1
P4
G4
P3
G3
P2
G2
P1
G1
C1
Carry Look-ahead generator
C5
C4
C3
C2
P4
P3
P2
P1
S4
S3
S2
S1
C5
4-bit adder
S = A ⊕ B ⊕ Ci
P
4 bit
A, B
4 bit
A3 A2 A1 A0

Decimal adders



Binary

$A_3 A_2 A_1 A_0 = A$ (0-9)

$B_3 B_2 B_1 B_0 = B$ (0-9)

BCD

8421

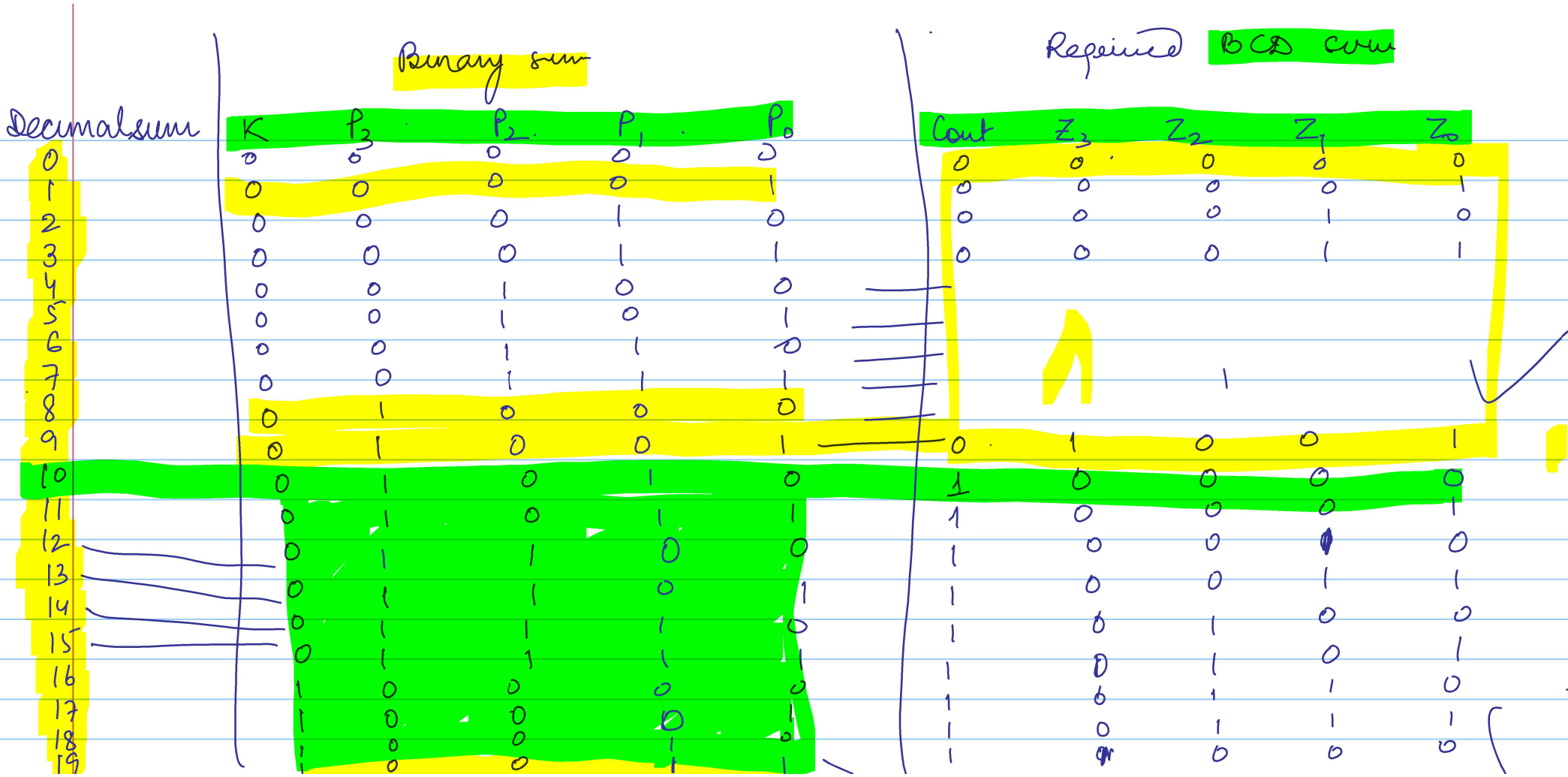
19

8 4 2 1

P Q R S

8421 (BCD) code / Natural BCD code

each decimal digit 0 – 9 — (4 bit-binary no.) coded by


Binary sum
Required BCD sum
Decimal sum
K
P3
P2
P1
P0
Cout
Z3
Z2
Z1
Z0

$K + P_3P_2 + P_3P_1 = K + P_3(P_2 + P_1)$ — 1 1 0 0 1

0110 should be added to the sum, to produce proper BCD result. This will also produce a carry to be added to the next decimal position.

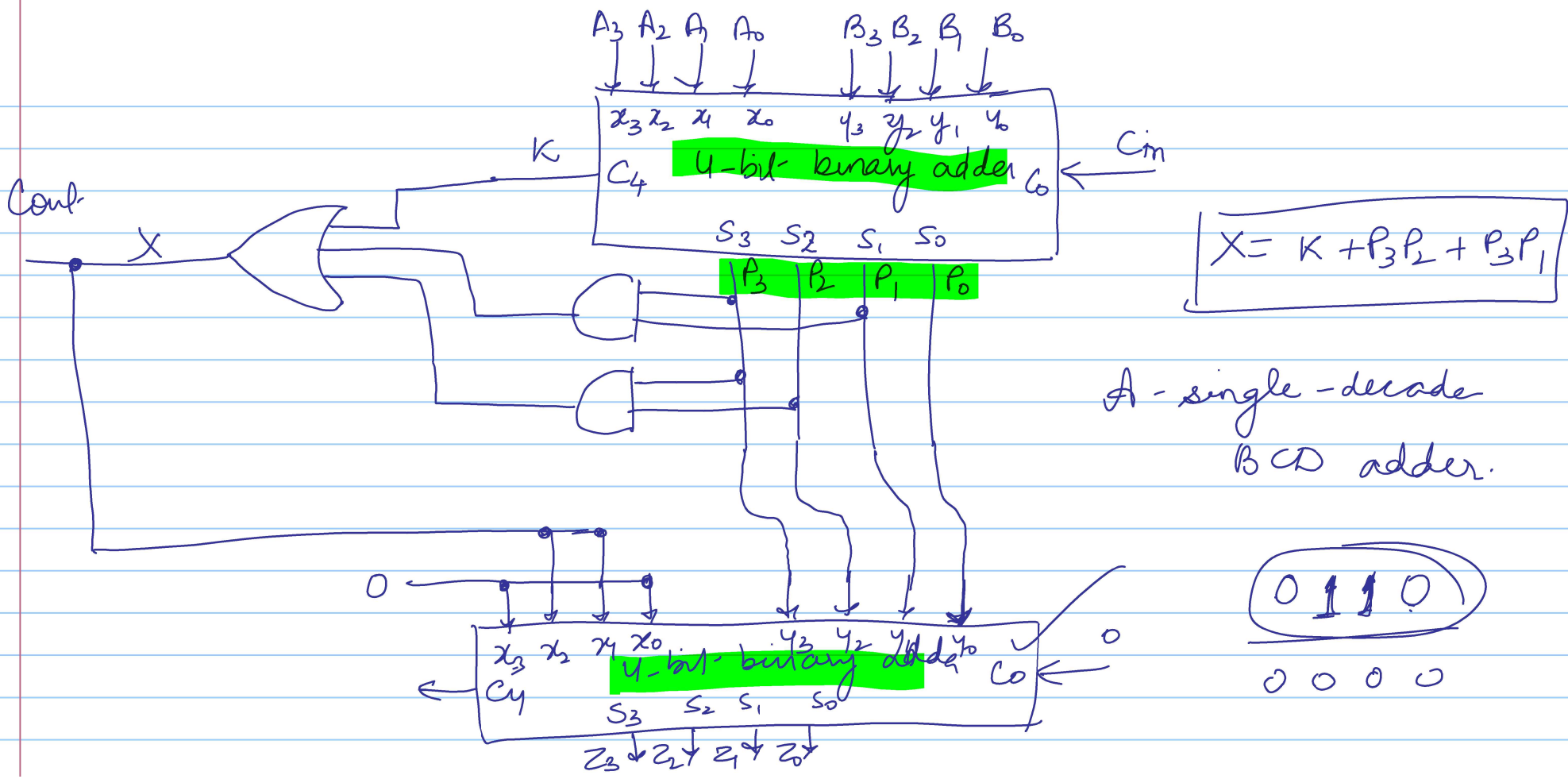
$1001 + 0110 = 10001$

---

(1) Add two 4-bit BCD code groups, using straight binary addition

(2) If the O/P is > 9 (1001), if it true Then add 110 (decimal 6) to this sum and generate a carry to the next decimal position.

$$\begin{array}{l} \overset{\nearrow 1}{} \quad \overset{\nearrow 0}{} \\ 01010 + 0110 = 10000 \end{array}$$



$$X = K + P_3P_2 + P_3P_1$$

A - single - decade BCD adder.

0 1 1 0

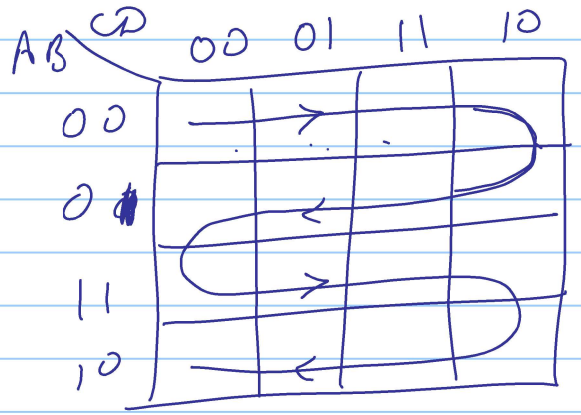
0 0 0 0

Code converters / 4-bit binary to Gray code

4-bit Binary				4-bit Gray			
B_3	B_2	B_1	B_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

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

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



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

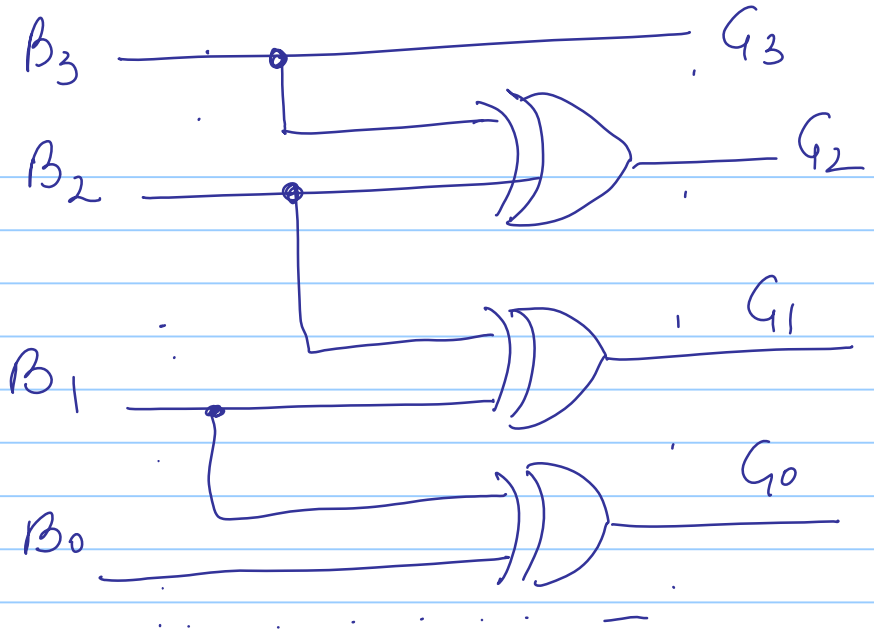
$G_3 = B_3$

$G_2 = B_3 \oplus B_2$

$G_1 = B_2 \oplus B_1$

$G_0 = B_1 \oplus B_0$

Draw K-map



4-bit Binary to Gray code converter

Ques Design 4-bit Gray to Binary CC

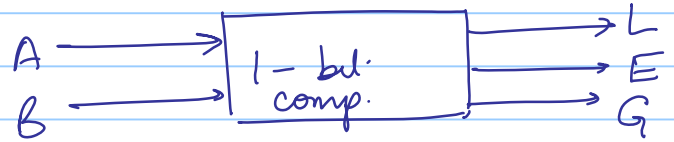
6 Binary to BCD CC

4 BCD to XS-3 CC

4 BCD to Gray CC

Comparators

Note Title 6/11/2021



1 - bit comparator

$A = A_0$, $B = B_0$

A	B	L	E	G
0	0	0	1	0
0	1	1	0	0
1	0	0	0	1
1	1	0	1	0

truth table

$L: (A < B) = \bar{A}B$

$E: (A = B) = A \odot B$

$G: (A > B) = A\bar{B}$

X-NOR

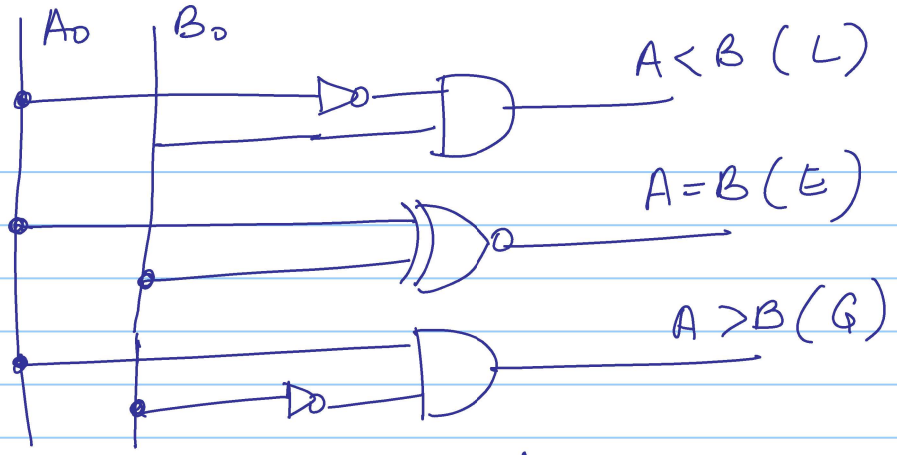
A	B	O/P
→ 0	0	1
0	1	0
1	0	0
→ 1	1	1

Coincidence gate.

$A > B$: G

$A = B$: E

$A < B$: L



Logic diagram

2-bit comparator

$A = A_1 A_0$ (MSB, LSB), $B = B_1 B_0$

Inputs

A_1	A_0	B_1	B_0	G	E	L
0	0	0	0			

(1) If $A_1 = 1$ and $B_1 = 0$, then $A > B$.

(2) If $A_1 = B_1$ and $A_0 = 1$ and $B_0 = 0$

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

then $A > B$

$$G(A>B) = A_1\bar{B}_1 + (A_1 \odot B_1)A_0\bar{B}_0$$

$$= A_1\bar{B}_1 + (A_1 \odot B_1)A_0\bar{B}_0$$

---

① If $A_1 = 0$ and $B_1 = 1$, then $A < B$

② If $A_1 = B_1$ and $A_0 = 0$ and $B_0 = 1$ then $A < B$

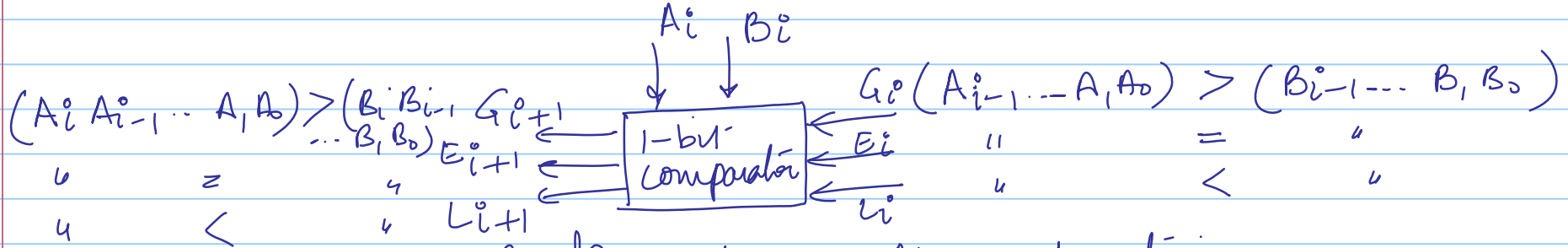
$$L(A<B) = \bar{A}_1 B_1 + (A_1 \odot B_1)\bar{A}_0 B_0$$

When $A = B$, $A_1 = B_1$ and $A_0 = B_0$

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

A_1
B_1
A_0
B_0
$\bar{A}_1$
B_1
$\bar{A}_0$
B_0
$\bar{B}_1$
A_1
$\bar{B}_0$
A_0
$L\,(A<B)$
$E\,(A=B)$
$G\,(A>B)$

$A = A_{n-1}\ A_{n-2} \cdots A_1 A_0$, $B = B_{n-1}\ B_{n-2} \cdots B_1 B_0$



Organization of 1-bit comparator

1-bit comp is a 5-input, 3-output n/w.

Fourth table

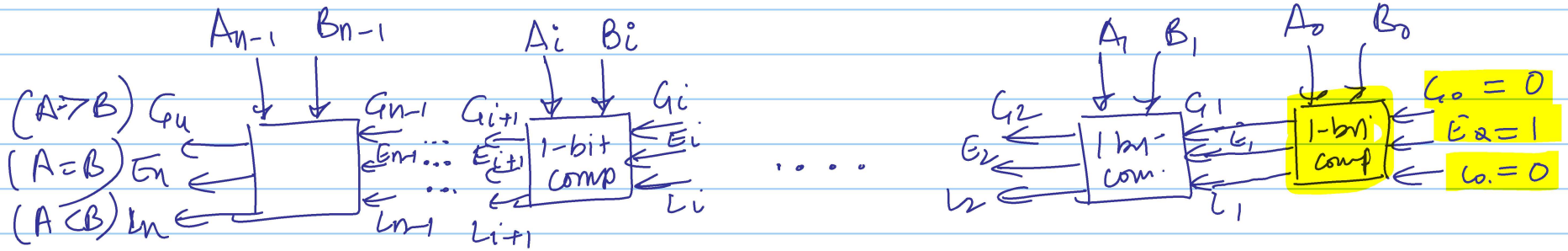
A_i	B_i	G_i	E_i	L_i	G_{i+1}	E_{i+1}	L_{i+1}
0	0	0	0	0	—	—	—
0	0	0	0	1	0	0	1
0	0	0	1	0	0	1	0
0	0	0	1	1	—	—	—
0	0	1	0	0	1	0	0
0	0	1	0	1	—	—	—
0	0	1	1	0	—	—	—
0	0	1	1	1	—	—	—
0	1	0	0	0	—	—	—
0	1	0	0	1	0	0	1
0	1	0	1	0	0	0	1
0	1	0	1	1	—	—	—
0	1	1	0	0	0	0	1
0	1	1	0	1	—	—	—
0	1	1	1	0	—	—	—
0	1	1	1	1	—	—	—

A_i	B_i	G_i	E_i	L_i	G_{i+1}	E_{i+1}	L_{i+1}
1	0	0	0	0	—	—	—
1	0	0	0	1	1	0	0
1	0	0	1	0	1	0	0
1	0	0	1	1	—	—	—
1	0	1	0	0	1	0	0
1	0	1	0	1	—	—	—
1	0	1	1	0	—	—	—
1	0	1	1	1	—	—	—
1	1	0	0	0	—	—	—
1	1	0	0	1	0	0	1
1	1	0	1	0	0	1	0
1	1	0	1	1	—	—	—
1	1	1	0	0	1	0	0
1	1	1	0	1	—	—	—
1	1	1	1	0	—	—	—
1	1	1	1	1	—	—	—

$$G_{i+1} = A_i \bar{B}_i + A_i G_i + \bar{B}_i G_i$$

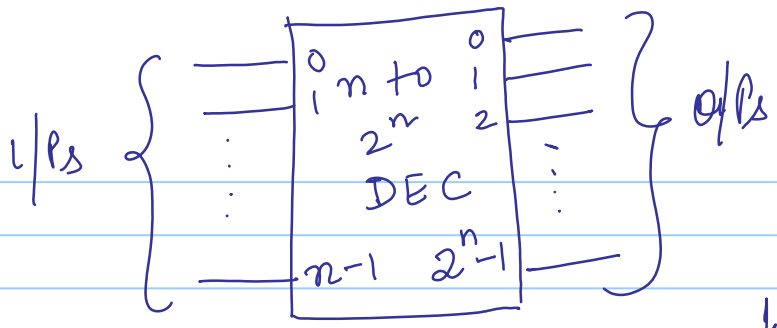
$$E_{i+1} = \bar{A}_i \bar{B}_i \bar{E}_i + A_i B_i E_i$$

$$L_{i+1} = \bar{A}_i B_i + B_i L_i + \bar{A}_i L_i$$



Decoders

n – to – 2^n line decoder



An n-to-2^n line decoder.

$Z_0 = \bar{x}_2 \bar{x}_1 \bar{x}_0 = m_0$

$Z_1 = \bar{x}_2 \bar{x}_1 x_0 = m_1$

$Z_2 = \bar{x}_2 x_1 \bar{x}_0 = m_2$

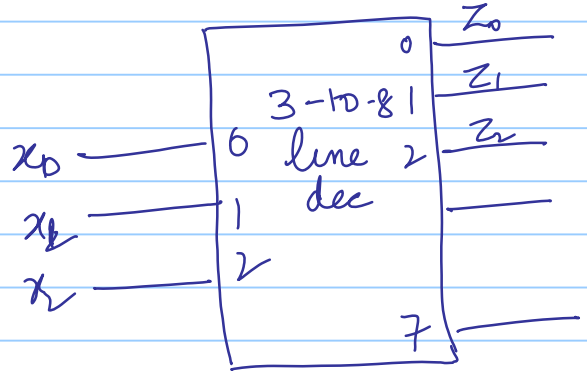
$Z_3 = \bar{x}_2 x_1 x_0 = m_3$

$Z_4 = x_2 \bar{x}_1 \bar{x}_0 = m_4$

$Z_5 = x_2 \bar{x}_1 x_0 = m_5$

$Z_6 = x_2 x_1 \bar{x}_0 = m_6$

$Z_7 = x_2 x_1 x_0 = m_7$

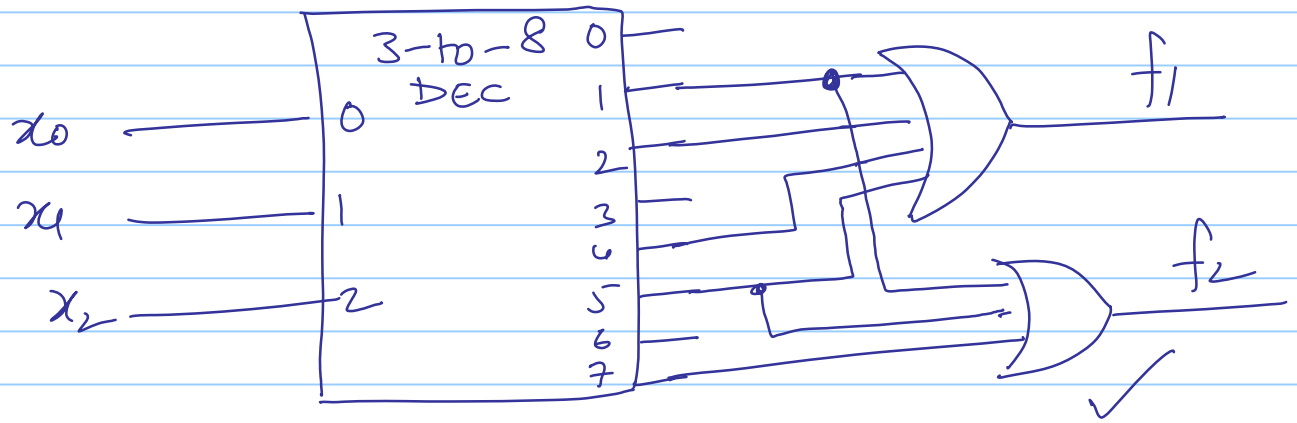


minterm generator

Inputs			Outputs							
x_2	x_1	x_0	Z_0	Z_1	Z_2	Z_3	Z_4	Z_5	Z_6	Z_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

$f_1(x_2, x_1, x_0) = \sum m(1, 2, 4, 5)$

$f_2(x_2, x_1, x_0) = \sum m(1, 5, 7)$



Note Title 6/12/2021

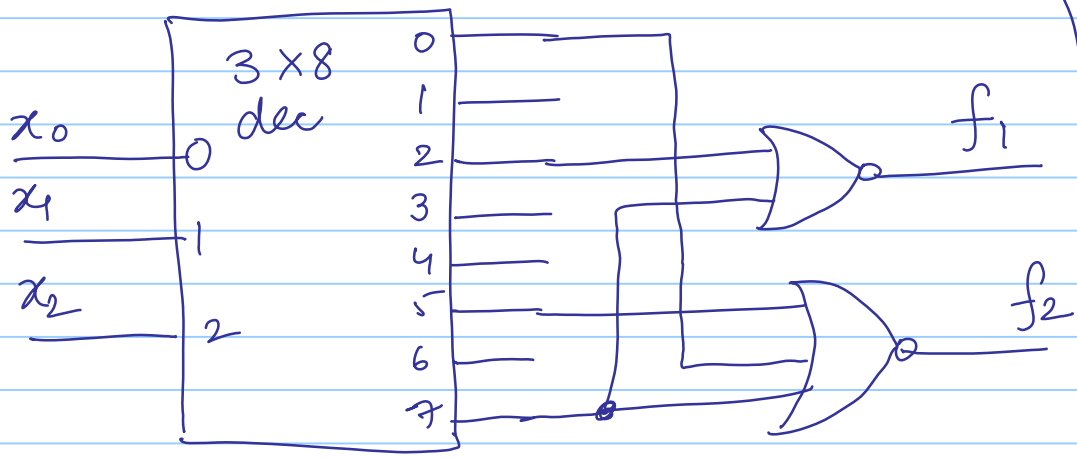
Ques $f_1(x_2, x_1, x_0) = \sum m(0, 1, 3, 4, 5, 6)$

$f_2(x_2, x_1, x_0) = \sum m(1, 2, 3, 4, 6)$

$\overline{f_1}(x_2, x_1, x_0) = \sum m(2, 7)$

$\overline{f_2}(x_2, x_1, x_0) = \sum (0, 5, 7)$

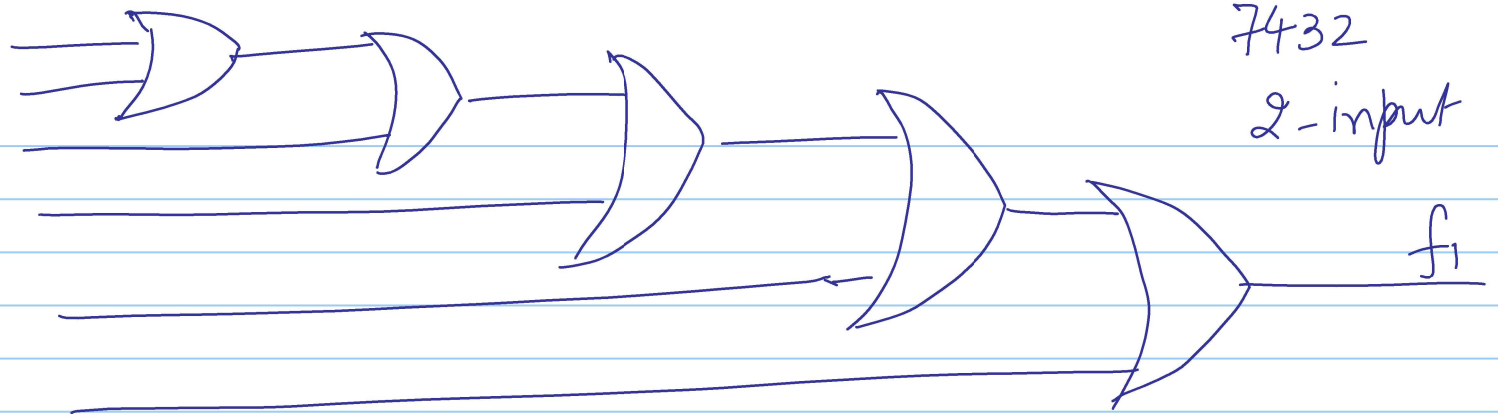
n — to — 2^n



Complementing above exp. using DeMorgan's law

$\overline{\overline{f_1}} = f_1 = \overline{\sum m(2, 7)}$

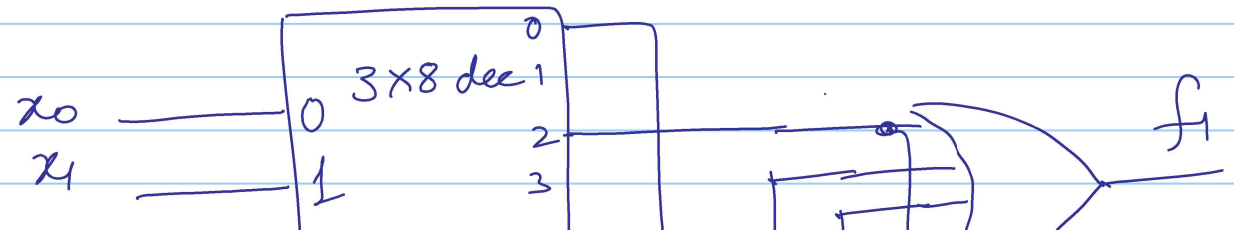
$\overline{\overline{f_2}} = f_2 = \overline{\sum m(0, 5, 7)}$

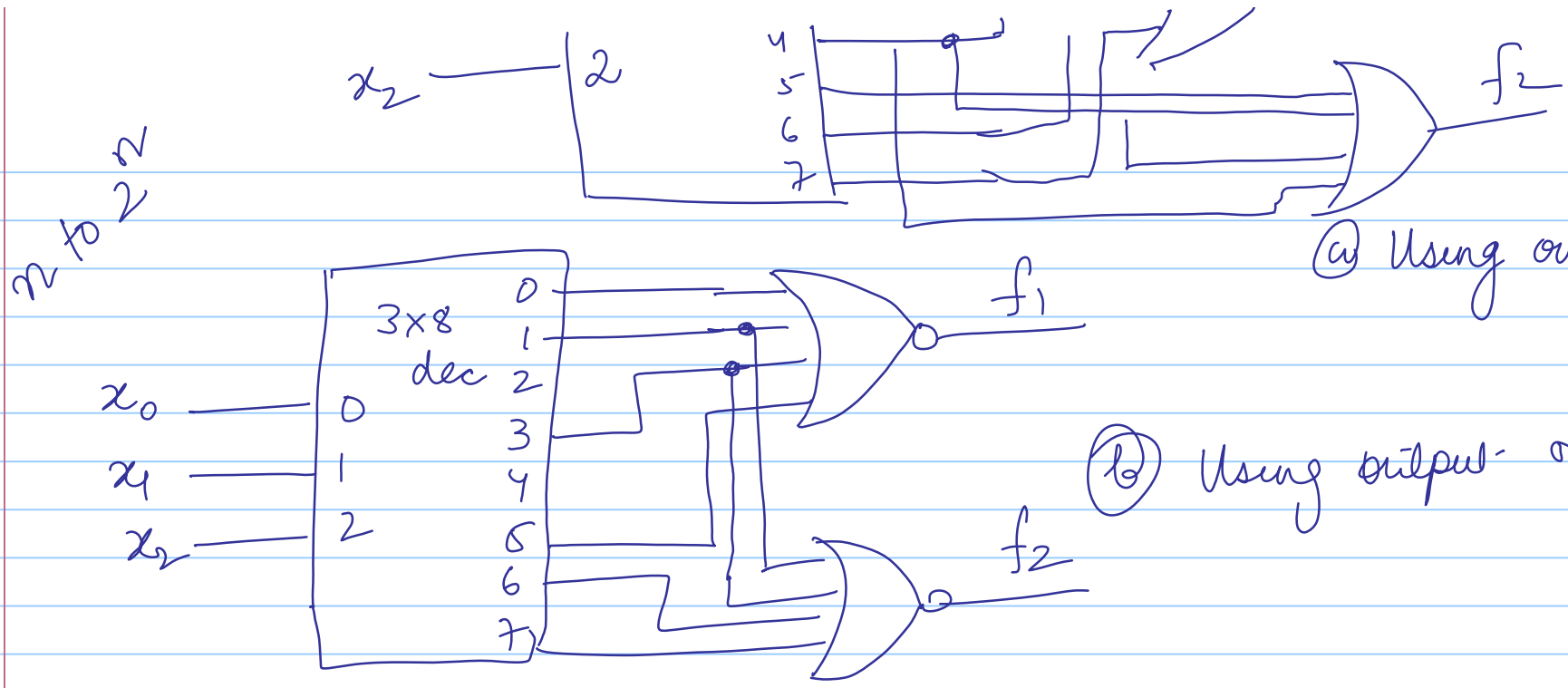


Ans

$f_1(x_2, x_1, x_0) = \Pi M(0,1,3,5) = \Sigma m(2,4,6,7)$

$f_2(x_2, x_1, x_0) = \Pi m(1,3,6,7) = \Sigma m(0,2,4,5)$



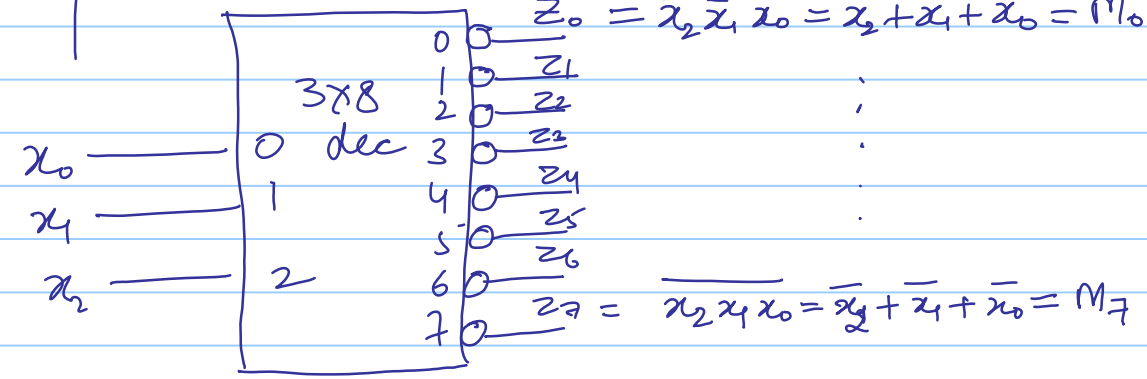


(a) Using output OR gates

(b) Using output NOR gates

$n \times 2^n$ decoders are constructed using NAND gates

Inputs			Outputs							
x_2	x_1	x_0	z_0	z_1	z_2	z_3	z_4	z_5	z_6	z_7
0	0	0	0	1	1	1	1	1	1	1
0	0	1	1	0	1	1	1	1	1	1
0	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	1	0	1	1	1
1	0	1	1	1	1	1	1	0	1	1
1	1	0	1	1	1	1	1	1	0	1
1	1	1	1	1	1	1	1	1	1	0



$$z_0 = \overline{\overline{x_2}\,\overline{x_1}\,\overline{x_0}}$$

$$= x_2 + x_1 + x_0 = M_0$$

$$z_1 = x_2 + x_1 + \overline{x_0} = M_1$$

$$z_2 = x_2 + \overline{x_1} + x_0 = M_2$$

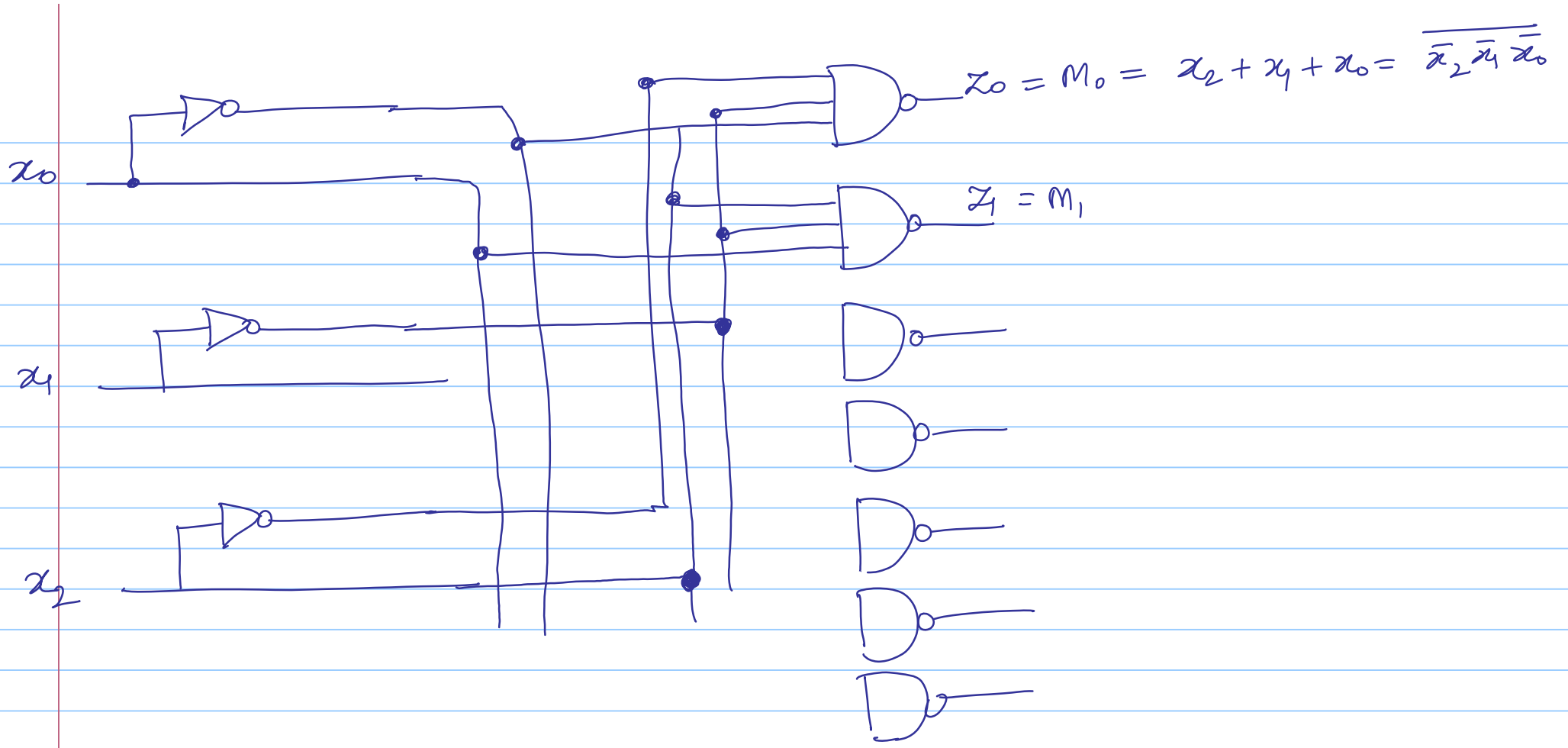
$$z_3 = x_2 + \overline{x_1} + \overline{x_0} = M_3$$

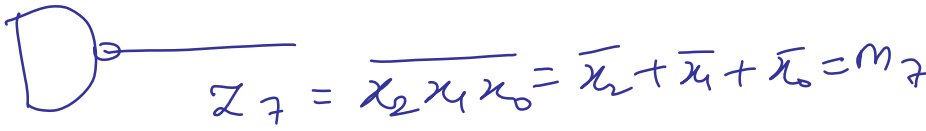
$$z_4 = \overline{x_2} + x_1 + x_0 = M_4$$

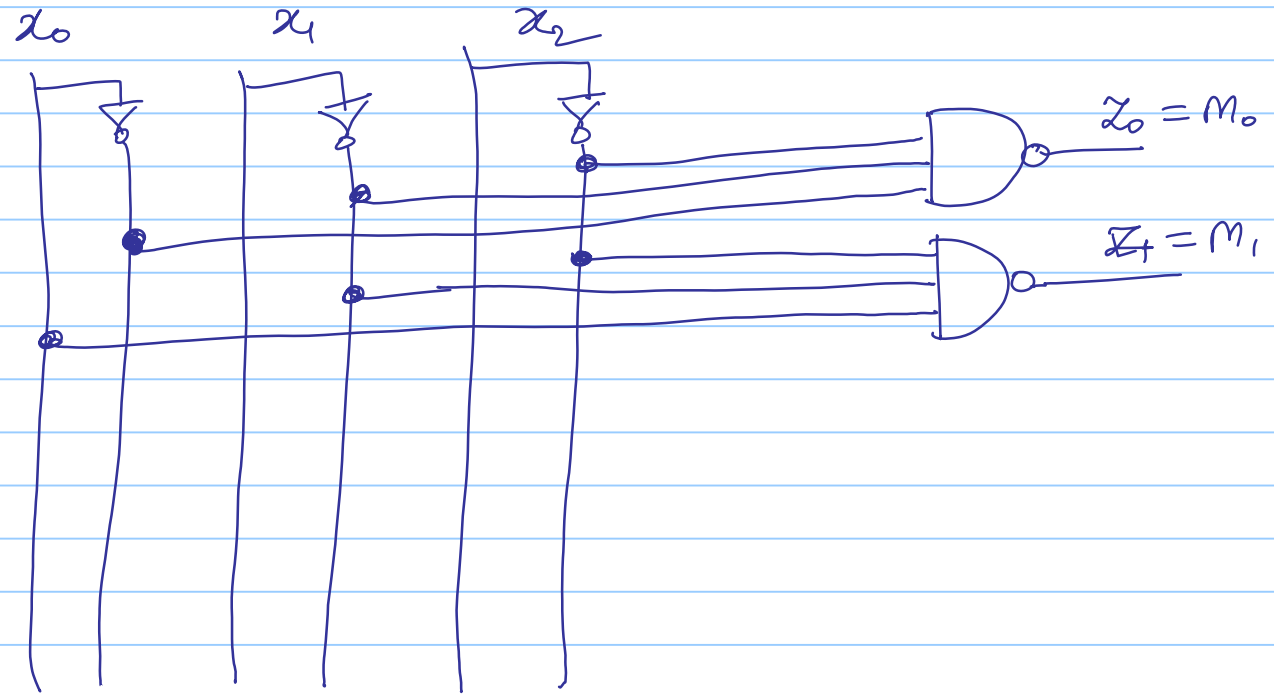
$$z_5 = \overline{x_2} + x_1 + \overline{x_0} = M_5$$

$$z_6 = \overline{x_2} + \overline{x_1} + x_0 = M_6$$

$$z_7 = \overline{x_2} + \overline{x_1} + \overline{x_0} = M_7$$


x_0
x_1
x_2
$z_0 = M_0 = x_2 + x_1 + x_0 = \overline{\bar{x}_2 \bar{x}_1 \bar{x}_0}$
$z_1 = M_1$


$Z_7 = \overline{x_2 x_1 x_0} = \bar{x}_2 + \bar{x}_1 + \bar{x}_0 = m_7$

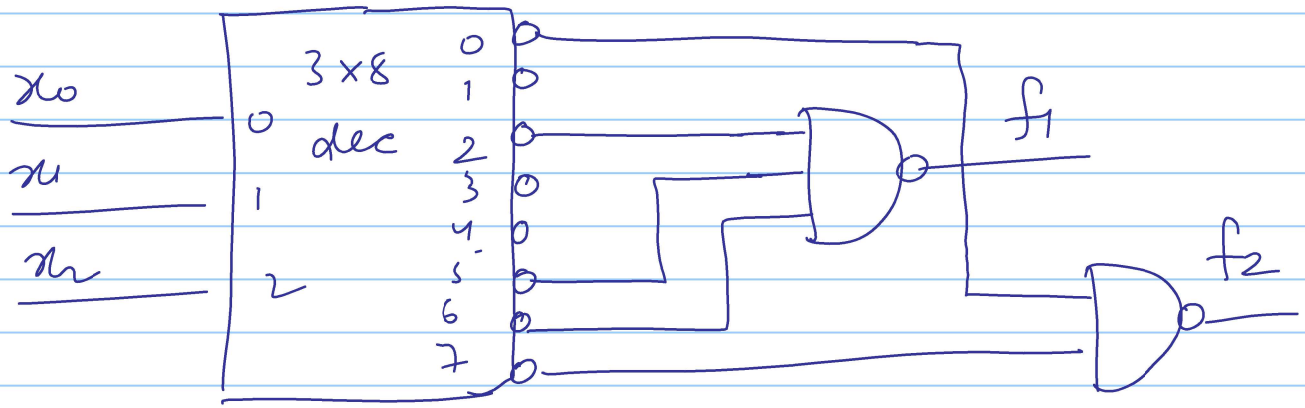

x_0
x_1
x_2
$Z_0 = m_0$
$Z_1 = m_1$

$f_1 = \Pi M(0,1,3,4,7)$

$f_2 = \Pi M(1,2,3,4,5,6)$

$\overline{f}_1(x_2, x_1, x_0) = f_1(x_2, x_1, x_0) = \overline{\Pi M(2,5,6)}$

$\overline{f}_2(x_2, x_1, x_0) = f_2(x_2, x_1, x_0) = \overline{\Pi M(0,7)}$



Decoders with an Enable input

enable input E, $\bar{E}$

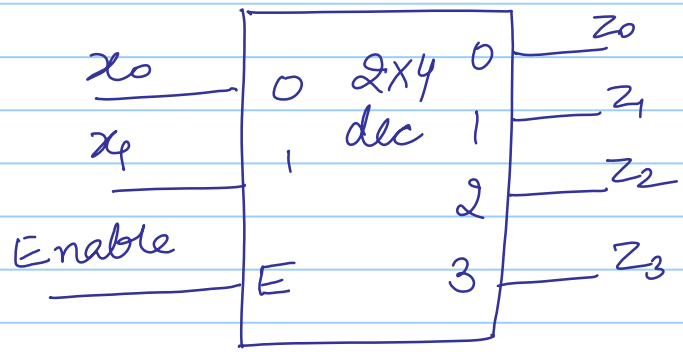
2×4 decoder

Compressed truth table

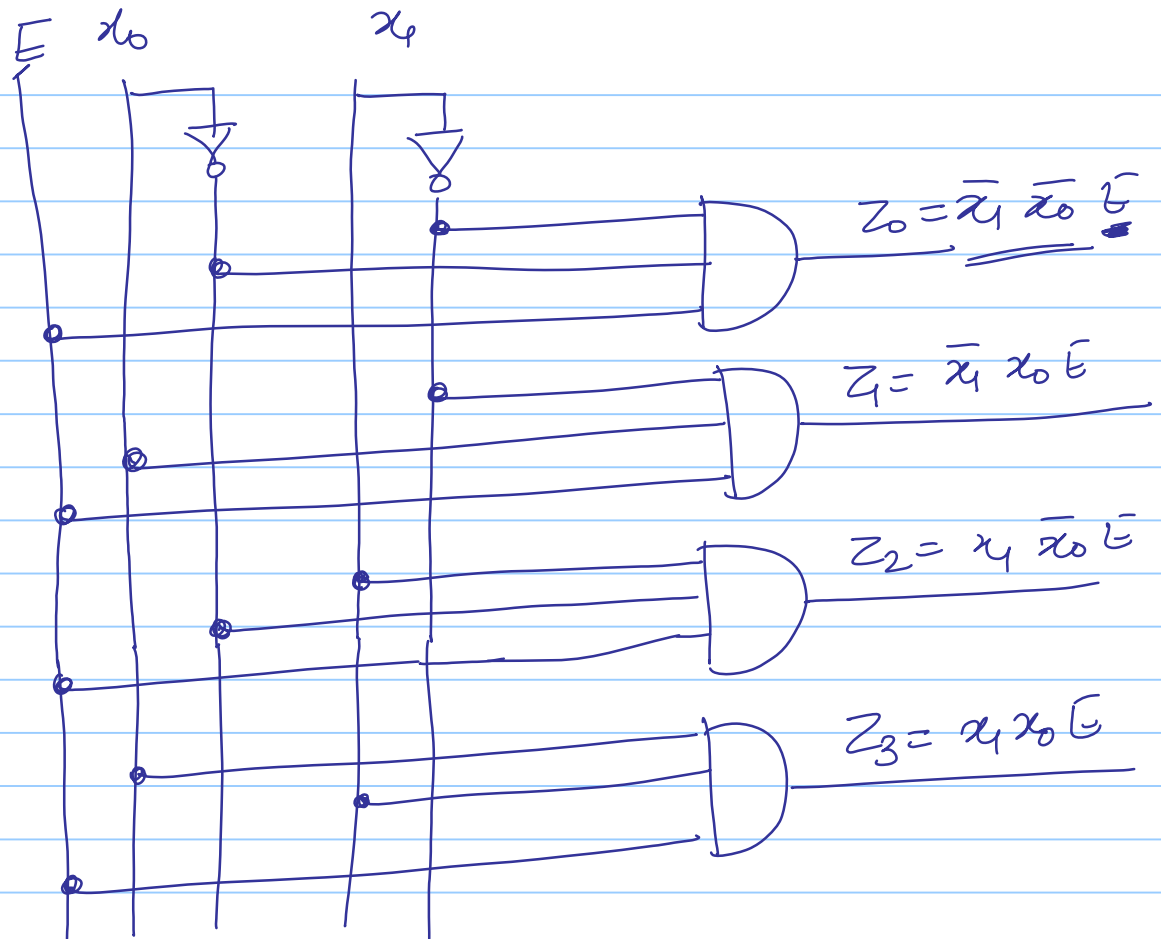
Input			output			
E	x_1	x_0	z_0	z_1	z_2	z_3
0	X	X	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1

$z_0 = \bar{x}_1 \bar{x}_0 E$ | $z_2 = x_1 \bar{x}_0 E$

$z_1 = \bar{x}_1 x_0 E$ | $z_3 = x_1 x_0 E$



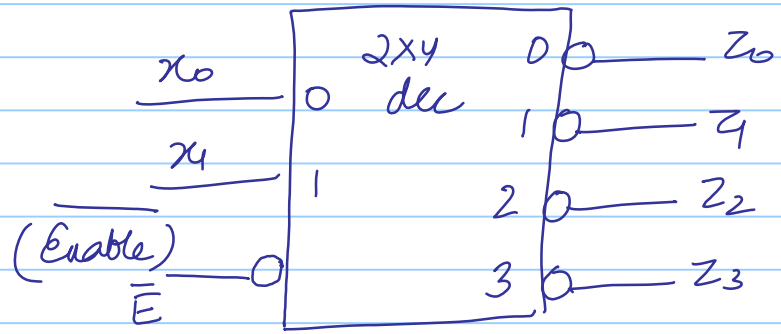
Symbol



Logic diagram

NAND gate 2x4 dec

Inputs			Outputs			
$\bar{E}$	x_1	x_0	z_0	z_1	z_2	z_3
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
1	X	X	1	1	1	1

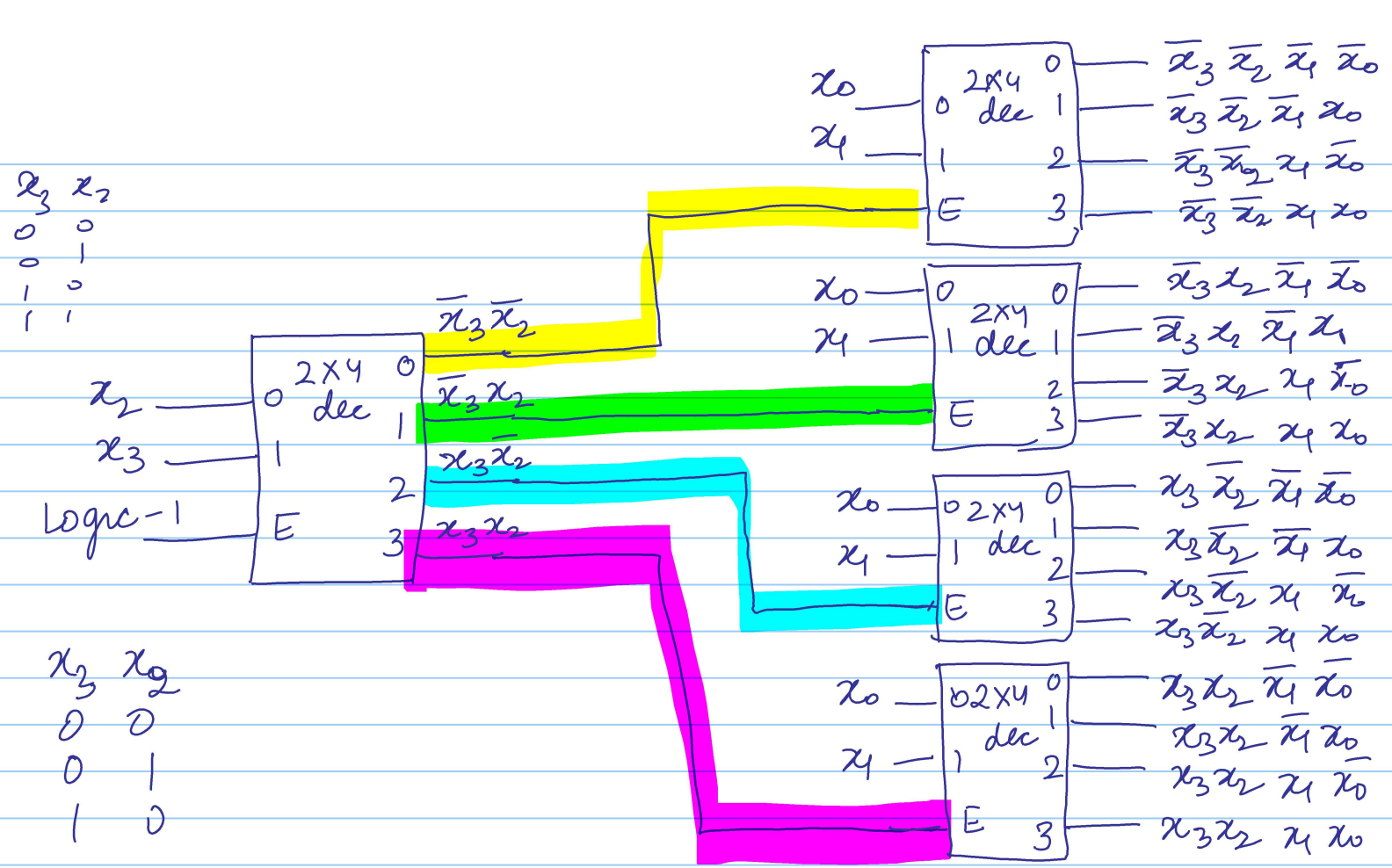


Decoders with enable i/p's are used to construct larger decoders from smaller decoders

E

x_3	x_2	x_1	x_0
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

x_3	x_2	x_1	x_0
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1


x_3 x_2
0 0
0 1
1 0
1 1
2X4 dec
x_2
x_3
Logic-1
E
0
1
2
3
$\bar{x}_3\bar{x}_2$
$\bar{x}_3 x_2$
$x_3\bar{x}_2$
$x_3 x_2$
x_0
x_1
$\bar{x}_3\bar{x}_2\bar{x}_1\bar{x}_0$
$\bar{x}_3\bar{x}_2\bar{x}_1 x_0$
$\bar{x}_3\bar{x}_2 x_1\bar{x}_0$
$\bar{x}_3\bar{x}_2 x_1 x_0$
$\bar{x}_3 x_2\bar{x}_1\bar{x}_0$
$\bar{x}_3 x_2\bar{x}_1 x_0$
$\bar{x}_3 x_2 x_1\bar{x}_0$
$\bar{x}_3 x_2 x_1 x_0$
$x_3\bar{x}_2\bar{x}_1\bar{x}_0$
$x_3\bar{x}_2\bar{x}_1 x_0$
$x_3\bar{x}_2 x_1\bar{x}_0$
$x_3\bar{x}_2 x_1 x_0$
$x_3 x_2\bar{x}_1\bar{x}_0$
$x_3 x_2\bar{x}_1 x_0$
$x_3 x_2 x_1\bar{x}_0$
$x_3 x_2 x_1 x_0$
x_3 x_2
0 0
0 1
1 0

Combinational logic implementation

2^n minterms for n input variables

m – func^n

3×8

f_1 f_2 $m=2$

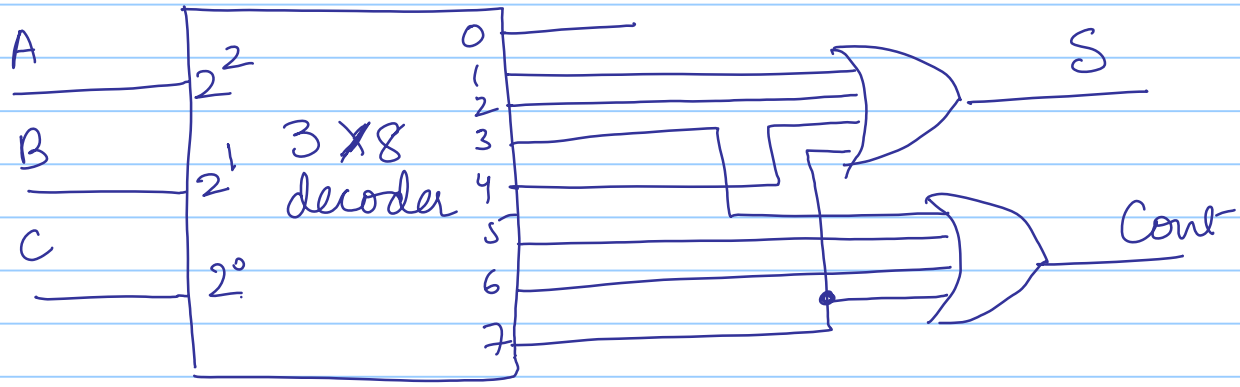
Any comb^n ckt with n inputs and m outputs can be implemented with an n to 2^n decoder and m OR gates

we need Boolean func^n to be expressed in SOP or minterm.

Full adder

$$S = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C}_{in} + A\bar{B}\bar{C}_{in} + ABC_{in} = A \oplus B \oplus C = \sum m(1,2,4,7)$$

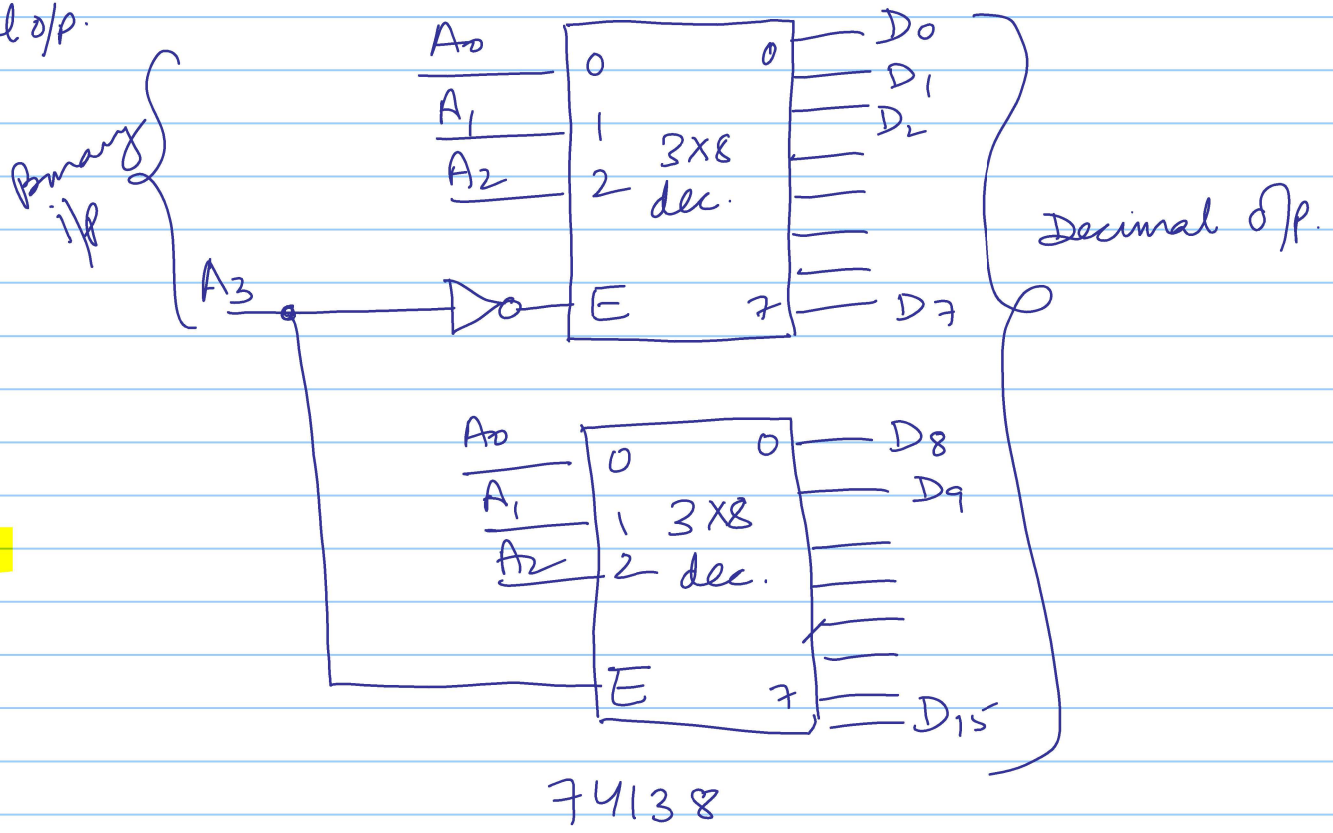
$$Cout = \bar{A}BC_{in} + A\bar{B}C_{in} + AB\bar{C}_{in} + ABC_{in} = AB + (A \oplus B)C_{in} = \sum m(3,5,6,7)$$



Logic diagram of a full adder using a decoder

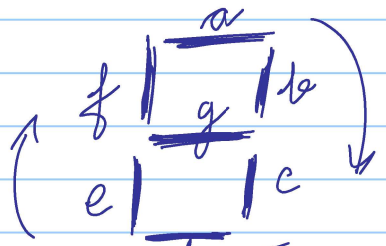
Implement 4 X16 decoder using 3 x8 decoders (two decoders)
function table

Binary I/Ps				decimal o/p.
A_3	A_2	A_1	A_0	
0	0	0	0	D_0
0	0	0	1	D_1
0	0	1	0	D_2
0	0	1	1	D_3
0	1	0	0	D_4
0	1	0	1	D_5
0	1	1	0	D_6
0	1	1	1	D_7
1	0	0	0	D_8
1	0	0	1	D_9
1	0	1	0	D_{10}
1	0	1	1	D_{11}
1	1	0	0	D_{12}



1	1	0	1	D_{13}
1	1	1	0	D_{14}
1	1	1	1	D_{15}

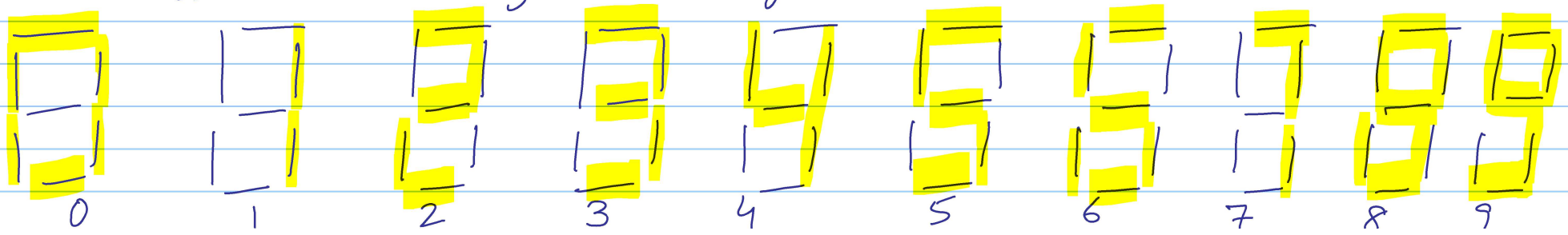
BCD to Seven-Segment Decoders



seven seg. display

LED, LCD

letters used to designate the segment



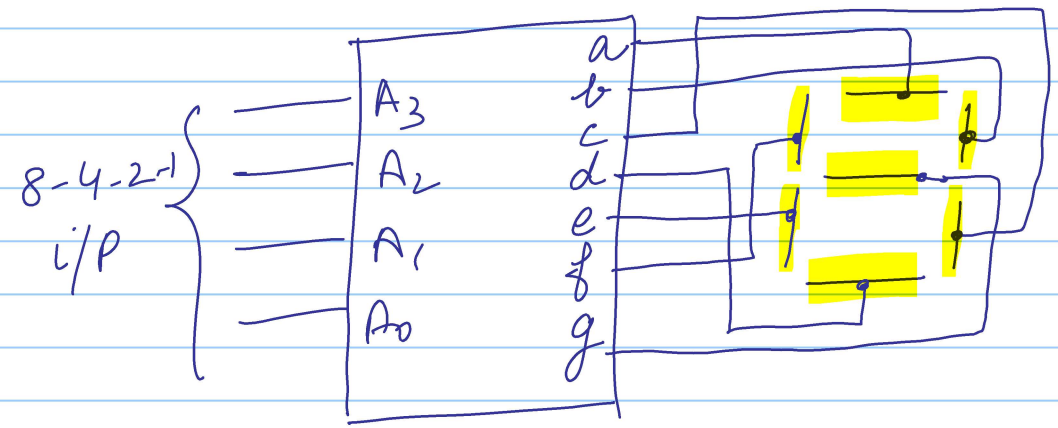
decimal digit	8-4-2-1 BCD				Seven-segment code						
	A_3	A_2	A_1	A_0	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1

$$b = \sum m(0, 1, 2, 3, 4, 7, 8, 9) + d \sum m(10, 11, 12, 13, 14, 15)$$

b

A_3A_2 \ A_1A_0	00	01	11	10
00	1 (0)	1 (1)	1 (3)	1 (2)
01	1 (4)	0 (5)	1 (7)	0 (6)
11	X (12)	X (13)	X (15)	X (14)
10	1 (8)	1 (9)	X (11)	X (10)

$$b = \bar{A}_2 + \bar{A}_1\bar{A}_0 + A_1A_0$$



a =

c =

d =

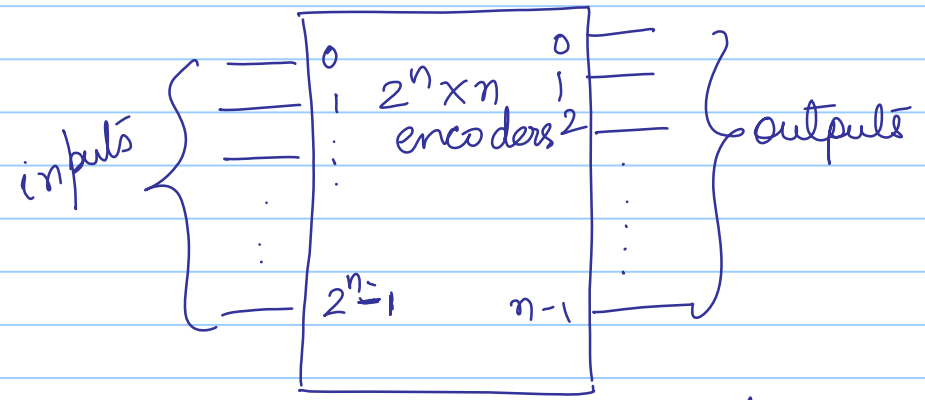
Encoders

Note Title 6/15/2021

Conversion of binary information from one form to another.

Reverse of decoders. ($n \times 2^n$)

Encoders ($2^n \times n$) i/p lines > o/p lines.



A $2^n \times n$ encoder symbol

Octal-to-Binary Encoder

(8 line to 3 line encoder)

$2^n = 8$ (0-7)

$n = 3$

Binary $A = A_2 A_1 A_0$

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

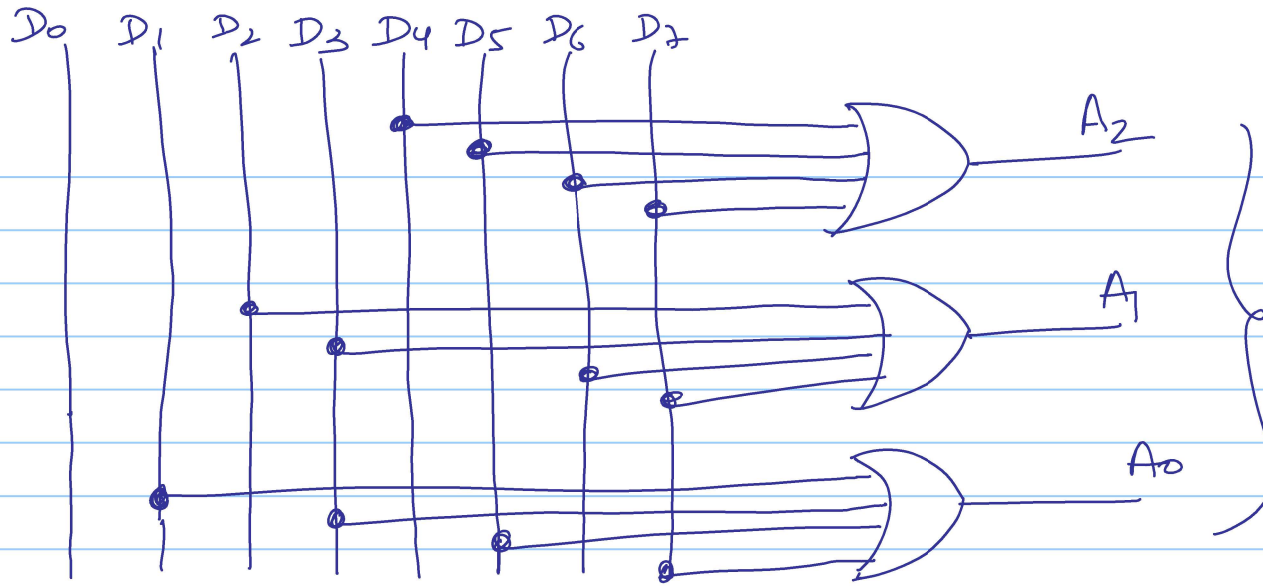
Truth table

$A_2 = D_4 + D_5 + D_6 + D_7$

$A_1 = D_2 + D_3 + D_6 + D_7$

$A_0 = D_1 + D_3 + D_5 + D_7$

$A_2 = 1$ if D_4 (+ OR) D_5 (+ OR) D_6 (+ OR) D_7



Logic diagram of octal-to-binary encoder

Priority Encoder

Have lunch → highest
Senior citizen
18-45
Children → lowest

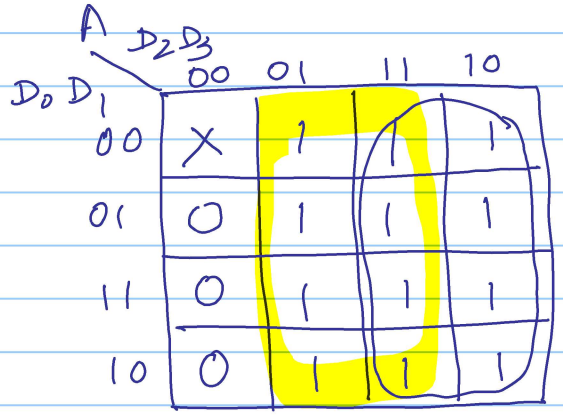
4-bit priority encoder

$V = D_0 + D_1 + D_2 + D_3$

Inputs				Outputs		
D_0	D_1	D_2	D_3	A	B	V
0	0	0	0	X	X	0
→ 1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

Condensed truth table

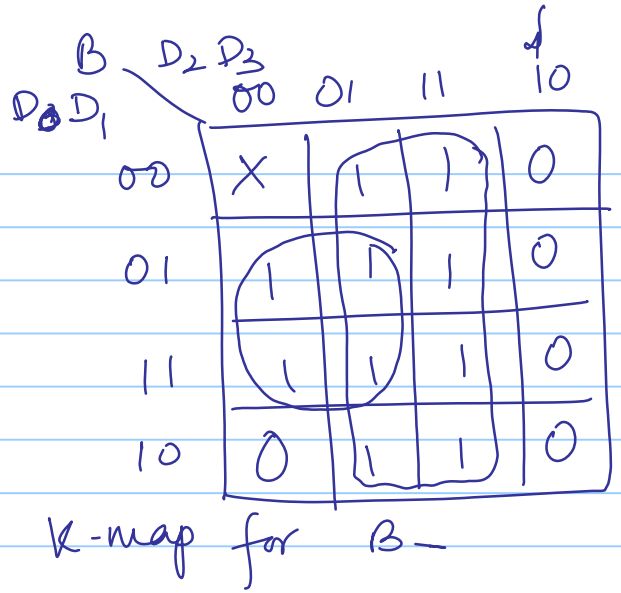
0	1	0	0	A=0
1	1	0	0	
0	0	1	0	A=1
0	1	1	0	
1	0	1	0	
1	1	1	0	
X	X	X	1	A=1



$A = D_3 + D_2$

K-map for A.

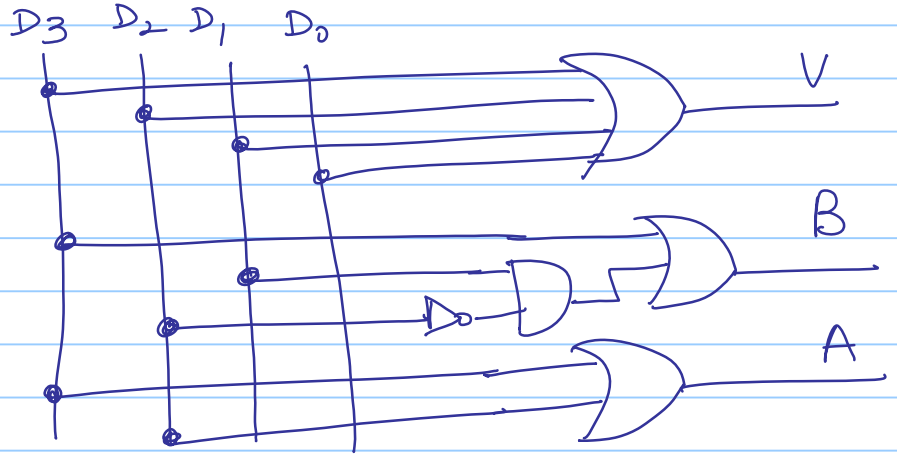
0	0	0	1	A=1
0	0	1	1	
0	1	0	1	
0	1	1	1	
1	0	0	1	
1	0	1	1	
1	1	0	1	
1	1	1	1	



K-map for B –

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

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



logic diag.

4- bit priority encoder

Decimal - to - BCD priority encoder

H. W

Multiplexer (MUX)

Multiplexing means sharing.

multiplexing < time
freq.

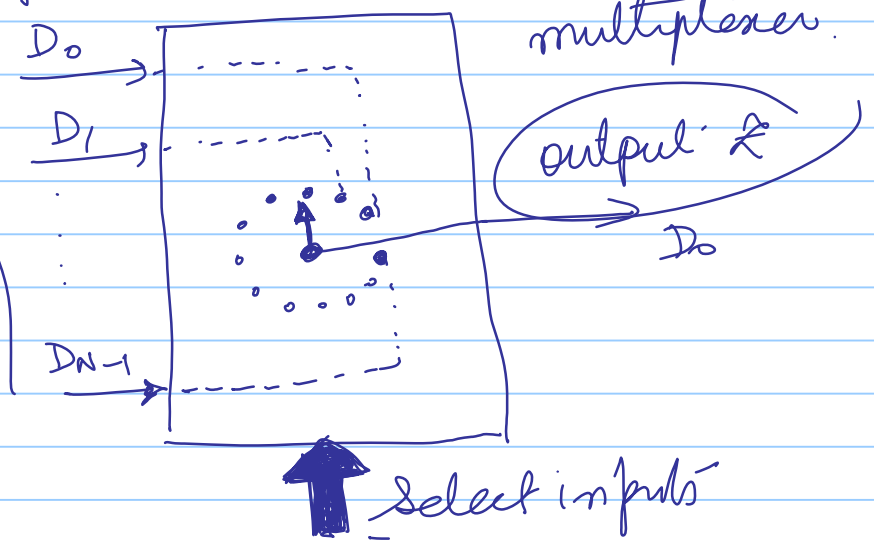
Multiplexer / Data selector

- $Z = D_0$ (Select i/p 1)
- $Z = D_1$ (" " 2)
- $Z = D_2$ (" " 3)

Data i/p

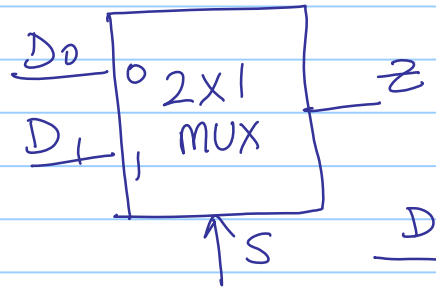
bus

functional diagram of a digital multiplexer.

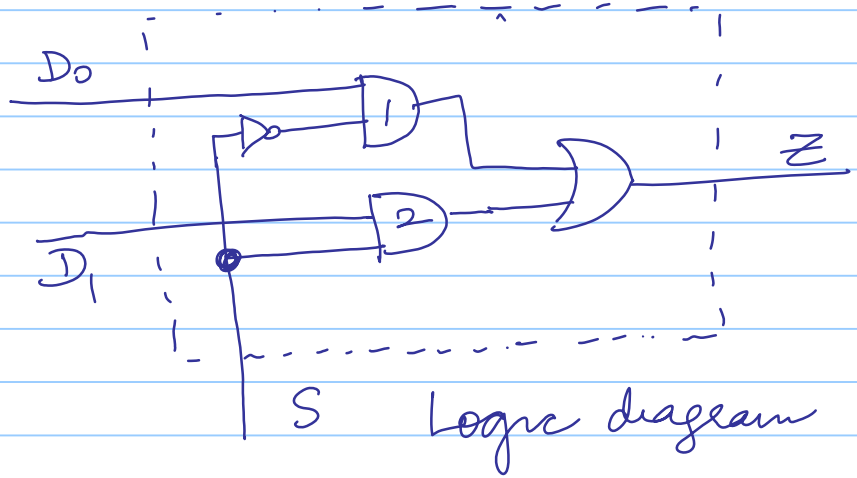


Basic 2-input multiplexer

data inputs D_0 & D_1, data select input S



S	output Z
0	D_0
1	D_1



Logic diagram

Note Title 6/16/2021

42, 48, 60, 78, 84,

116, 118, , 86, 96, 104, 106

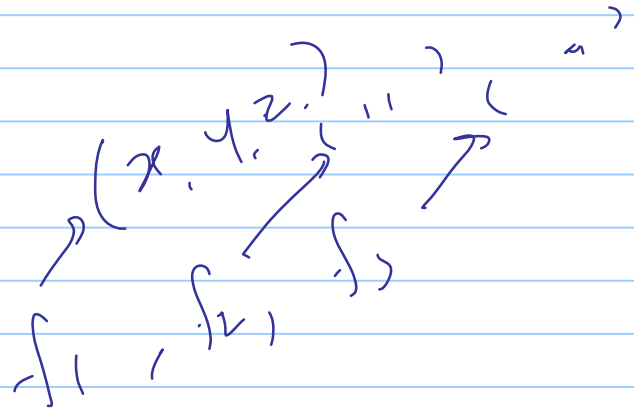
Roll no. 116, 8, 20

24, 28, 32, 38

Absent 62, 6, 14,

46, 62, 68, 70,

90, 100, 116



Even 34, 52, 54, 56,

64, 66, 72, 84,

92, 102, 114, 120,

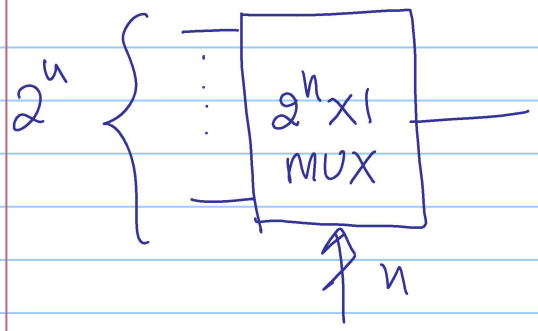
Basic 2-input multiplexer

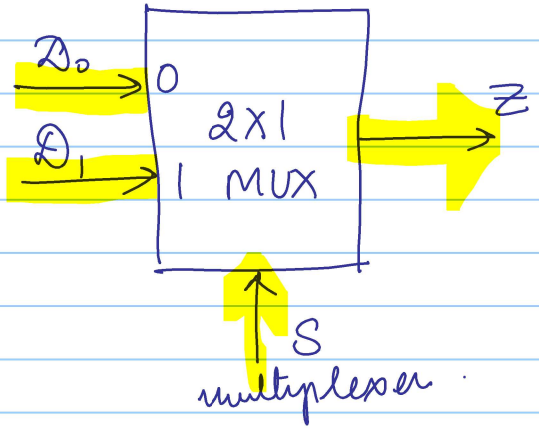
Sharing < time / frequency domain

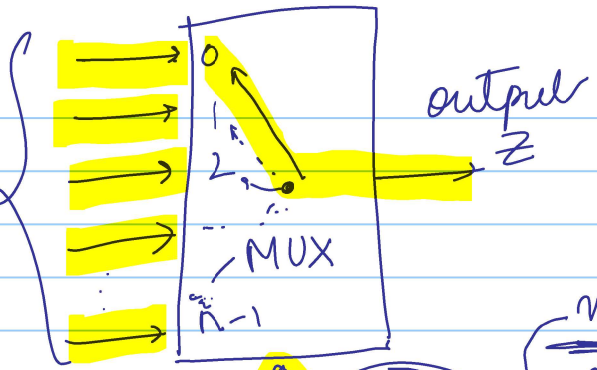
data selectors

2-i/p multiplexer

$n \rightarrow 2^n = m$







n - sel. lines

2^n → data lines

1 → output line

S	Z
0	D_0
1	D_1

function table

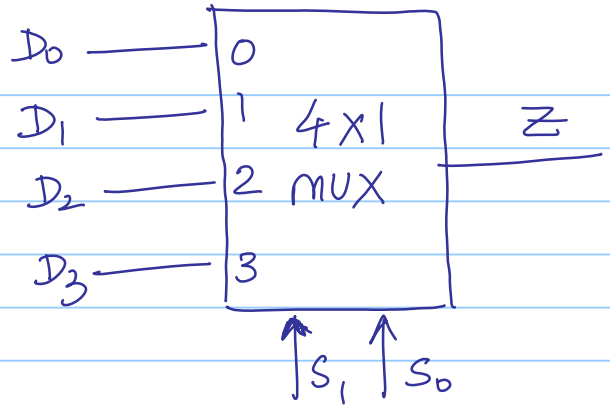
$Z = \bar{S} D_0 + S D_1$

2×1, 4×1, 8×1, 32:1

the 4-input multiplexer

2:1, 4:1, 8:1, 32:1

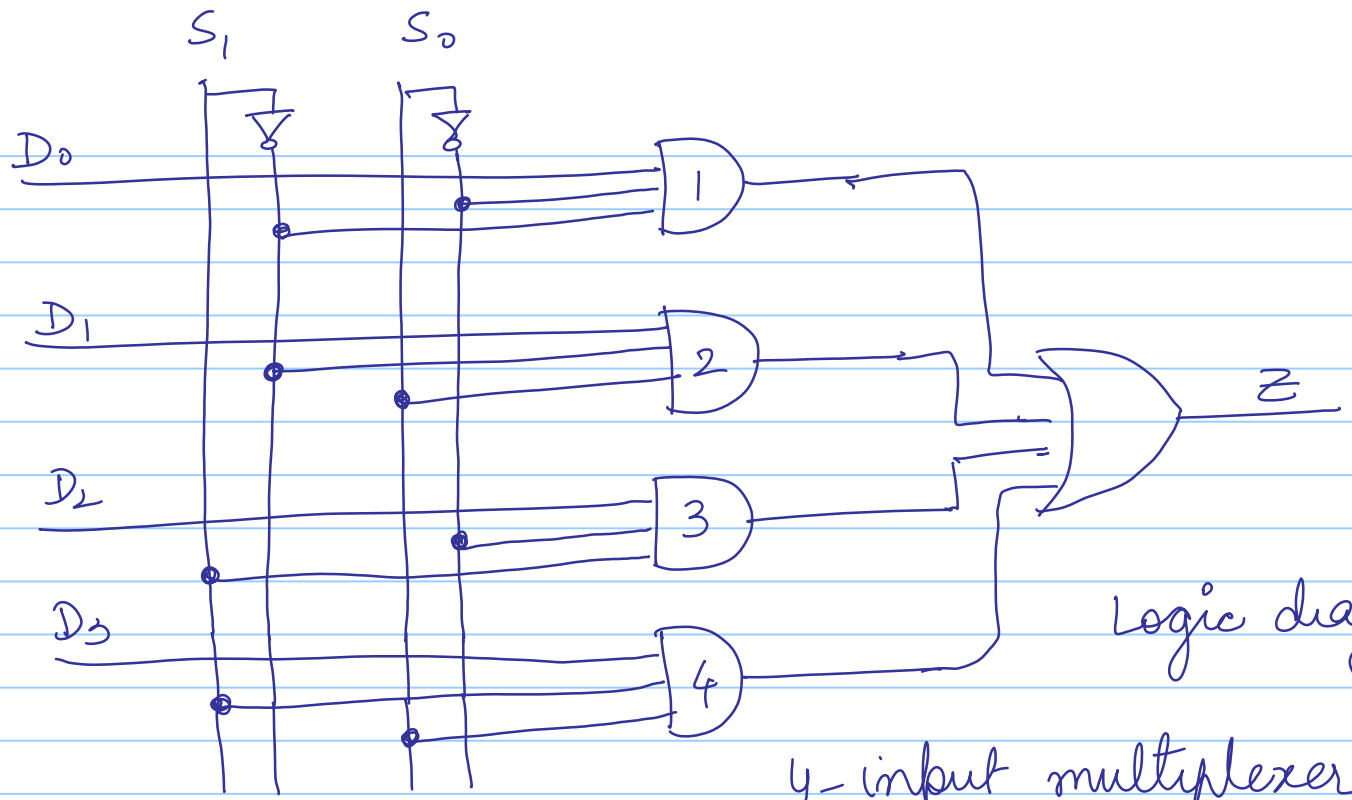
$2^n = 4 \Rightarrow n = 2$



deci	Select input S_1	S_0	Output Z
0	0	0	D_0
1	0	1	D_1
2	1	0	D_2
3	1	1	D_3

$$Z = \overline{S_1}\,\overline{S_0}\,D_0 + \overline{S_1}\,S_0\,D_1 + S_1\,\overline{S_0}\,D_2 + S_1 S_0 D_3$$

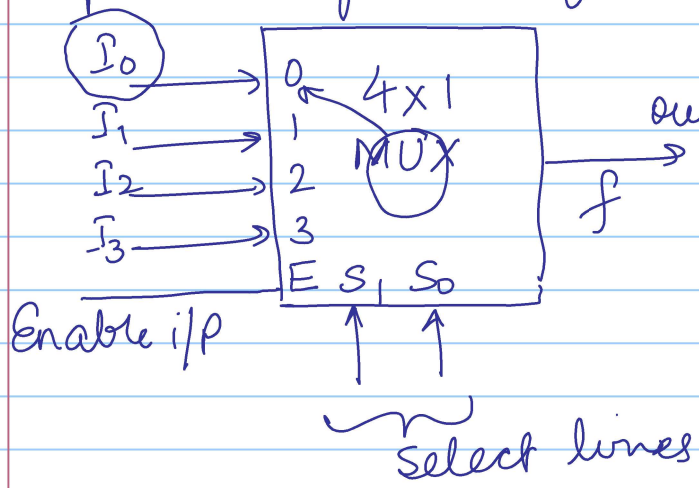
	4
74157	2X1
74153	dual 4X1 MUX



Logic diagram.

4-input multiplexer.

Additional line is called enable, or strobe line is to provide greater flexibility as in the case of a decoder.

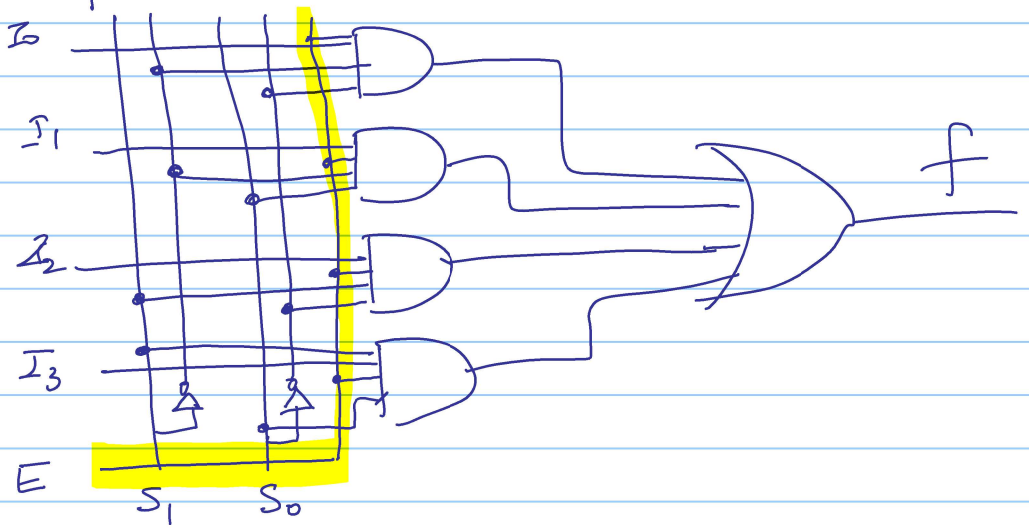


4:1 ← o/p

output line

$m = 2^n \Rightarrow n = \log_2 m$

$$f = (\bar{S}_1 \bar{S}_0 I_0 + \bar{S}_1 S_0 I_2 + S_1 \bar{S}_0 I_3 + S_1 S_0 I_4) E$$



compressed truth table for 4X1 line multiplexer

E	S_1	S_0	I_0	I_1	I_2	I_3	f
0	X	X	X	X	X	X	0
1	0	0	0	X	X	X	0
1	0	0	1	X	X	X	1
1	0	1	X	0	X	X	0
1	0	1	X	1	X	X	1
1	1	0	X	X	0	X	0
1	1	0	X	X	1	X	1
1	1	1	X	X	X	0	0
1	1	1	X	X	X	1	1

function table for a 4X1 line MUX

E	S_1	S_0	f
0	X	X	0
1	0	0	I_0
1	0	1	I_1
1	1	0	I_2
1	1	1	I_3

$$f = \left(I_0 \bar{S}_1 \bar{S}_0 + I_1 \bar{S}_1 S_0 + I_2 S_1 \bar{S}_0 + I_3 S_1 S_0 \right) E$$

Construct 16-to-1 MUX using 4X1 MUX

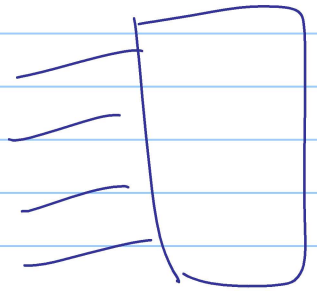
16 to 1		4 to 1 MUX	
Data lines	16	D.L	4
Select lines	4	S.L	2
Output line	1	O.L	1

16X1 MUX
16:1 MUX
16-to-1 MUX

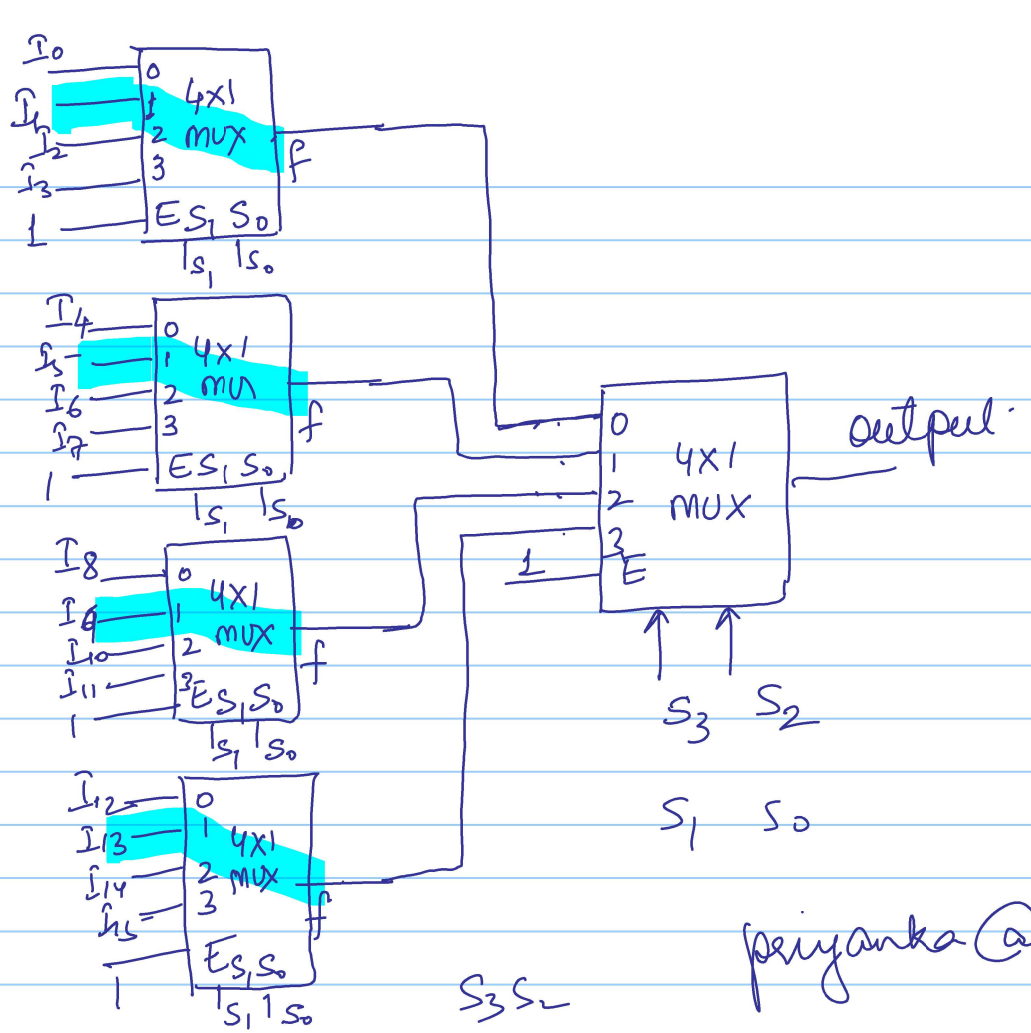
S_3	S_2	S_1	S_0	f
0	0	0	0	I_0
0	0	0	1	I_1
0	0	1	0	I_2
0	0	1	1	I_3
0	1	0	0	I_4
0	1	0	1	I_5
0	1	1	0	I_6
0	1	1	1	I_7
1	0	0	0	I_8
1	0	0	1	I_9
1	0	1	0	I_{10}
1	0	1	1	I_{11}
1	1	0	0	I_{12}

0 - 15

$I_0, I_1, I_2, \ldots, I_{15}$



16



1	1	0	1	I_{13}
1	1	1	0	I_{14}
1	1	1	1	I_{15}

S_3	S_2	S_1	S_0	
0	0	0	1	I_1
0	1	0	1	I_5
→ 1	0	0	1	I_9
1	1	0	1	I_{13}

S_3 S_2 S_1 S_0

MSB ↑ LSB

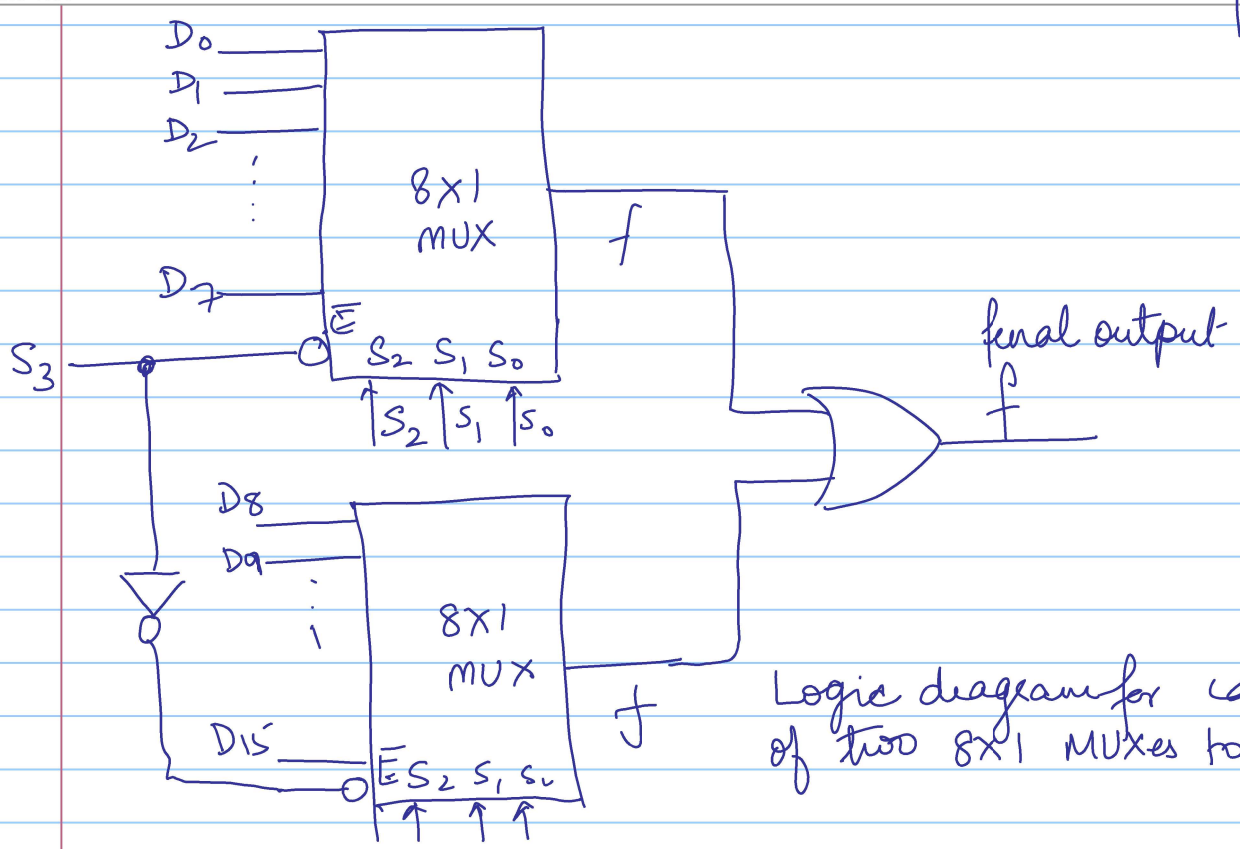
priyanka@iiitdm.ac.in

Note Title 6/17/2021

The 16-input MUX from two 8×1 MUX

I/P lines :- 16, O/P = 1
select lines = 4

I/P lines :- 8 O/P = 1
select lines = 3



Logic diagram for cascading of two 8×1 MUXes to get a 16×1 MUX

S_3	S_2	S_1	S_0	f
0	0	0	0	D_0 ✓
0	0	0	1	D_1 ✓
0	0	1	0	D_2 ✓
0	0	1	1	D_3
0	1	0	0	D_4
		⋮		⋮
0	1	1	1	D_7
1	0	0	0	D_8
		⋮		⋮
1	1	0	1	D_{13}
1	1	1	0	D_{14}
1	1	1	1	D_{15}

Applications of MUXes

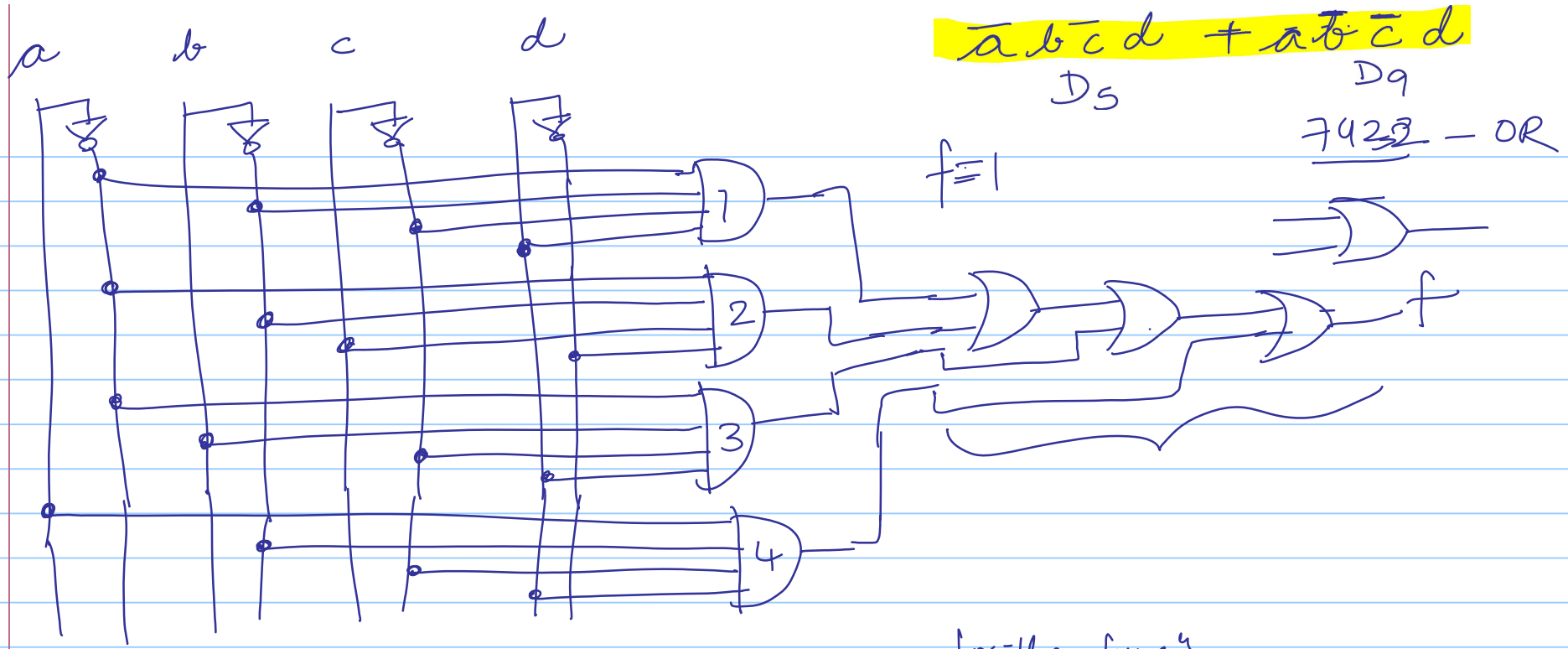
Various applications in digital systems of all types:

- data selection
- data routing
- operation sequencing
- parallel to serial conversion.
- waveform generation
- logic function generation.

Logic function generator

A MUX can be used in place of logic gates to implement a logic expression.

$$f = \Sigma m(1, 2, 5, 9) = \underbrace{\bar{a}b\bar{c}d}_{\rightarrow D_1} + \underbrace{a\bar{b}\bar{c}d}_{\rightarrow D_2} +$$



Steps ① Construct a truth table for the func^n to be implemented.

② Connect logic 1 to each data i/p of the MUX corresponding to each combination of the input variables.

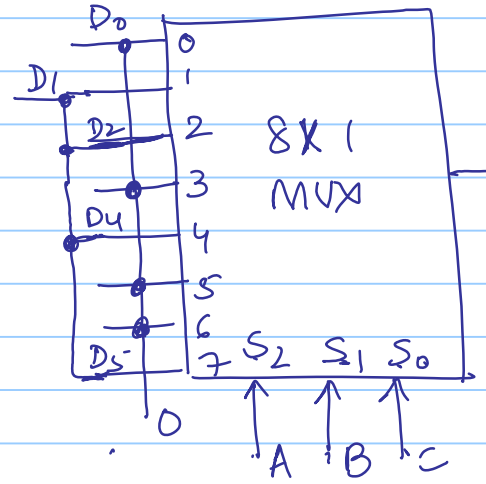
Ques: Use a MUX to implement the logic func^n $F = A \oplus B \oplus C$.

Sol There are 3 i/p variables $\therefore$ we need $2^3 \times 1$ MUX $\equiv$ 8×1 mux

S_2	S_1	S_0	$F = A \oplus B \oplus C$
A	B	C	
0	0	0	0 D_0
0	0	1	1 D_1
0	1	0	1 D_2
0	1	1	0 D_3
1	0	0	1 D_4
1	0	1	0 D_5
1	1	0	0 D_6
1	1	1	1 D_7

$\oplus \rightarrow$ odd $\rightarrow$ 1

$\odot \rightarrow$ even $\rightarrow$ 1



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

(1, 2, 4, 7)

$F = A \oplus B \oplus C$

o/p ↑ — 1/p

A	B	C	F	
0	0	0	0	
0	0	1	1	→ $\bar{A}\bar{B}C$
0	1	0	1	→ $\bar{A}B\bar{C}$
0	1	1	0	
1	0	0	1	→ $A\bar{B}\bar{C}$
1	0	1	0	
1	1	0	0	
1	1	1	1	→ ABC

A, B, C → Binary language

→ 0, 1

0 1

$\bar{A}$ A

F n=3

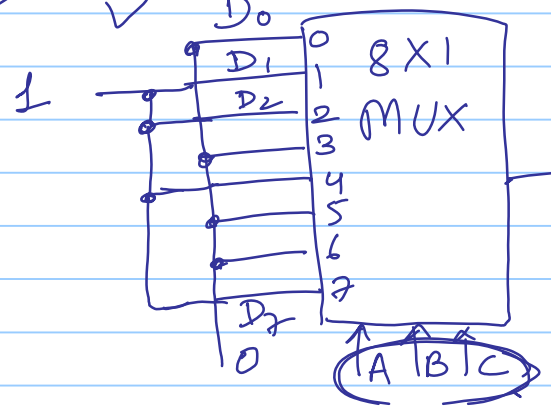
A, B, C → MUX → F

D_1 OR D_2 OR

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

D_1 D_2 D_4 D_7

D_0



Note Title 6/18/2021

Ques Implement the following func^n using **8 × 1 MUX**

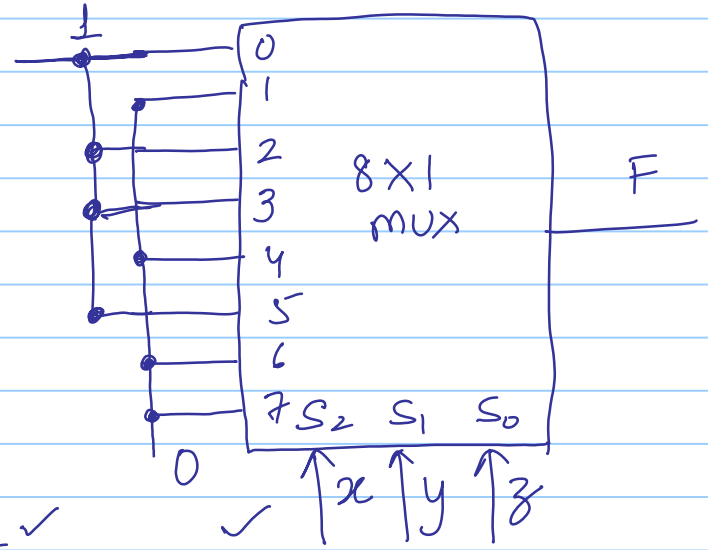
$$F(x, y, z) = \sum m(0, 2, 3, 5)$$

$8 \rightarrow 2^n \Rightarrow n = 3$

Select lines = 3

1 → o/p line

	S_2	S_1	S_0	
	x	y	z	F
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0



$$F = \bar{x}\bar{y}\bar{z} + \bar{x}y\bar{z} + \bar{x}yz + x\bar{y}z$$

In general, a multiplexer with n-data select i/Ps can implement any function of (n+1) variables

The key to this design is to use first n variables of the func^{n} as the select i/Ps and to use the least significant I/P variable and its complement to drive some of the data i/Ps.

If we assume a single variable D, each data I/P of the MUX will be D, $\bar{D}$, 1, 0.

A, B C, D

→

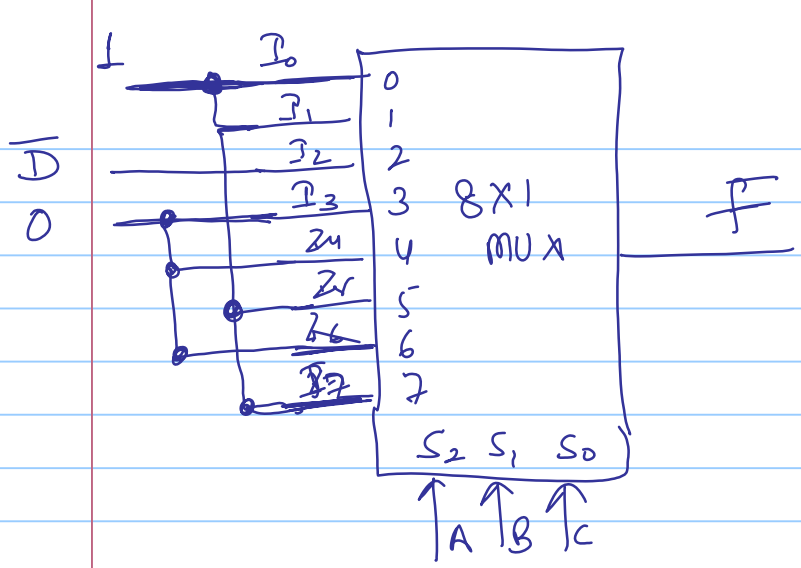
Ques Use a multiplexer having 3 data select i/Ps to implement the logic for the function given below. Also realize the same using 16 x 1 mux

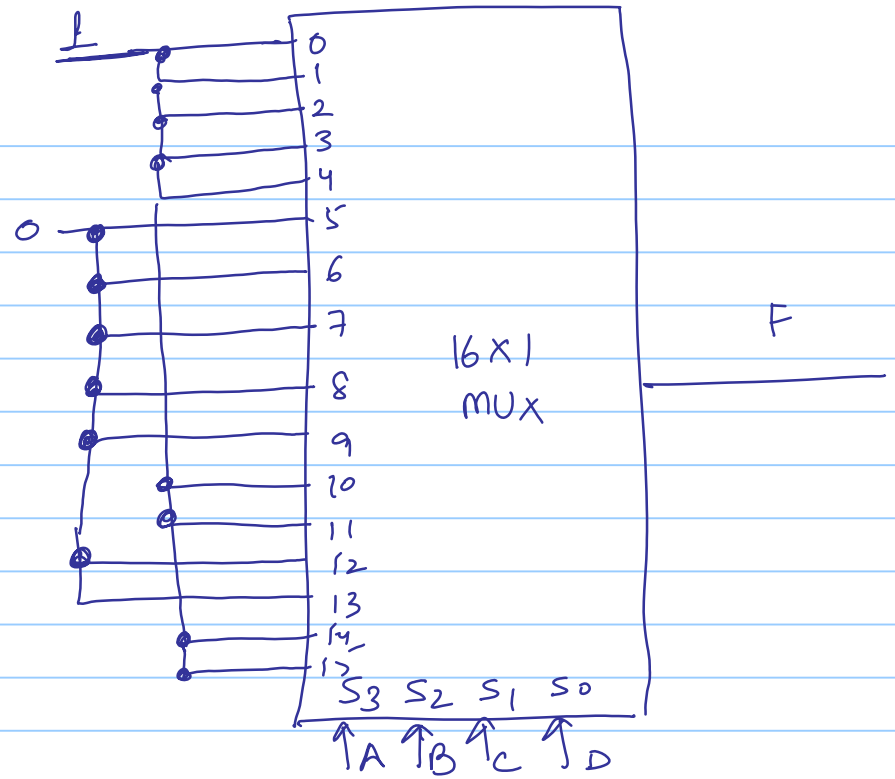
$$F = \sum m(0, 1, 2, 3, 4, 10, 11, 14, 15)$$

Sol The given func^n is of four variables (We can implement using (8x1 mux)

S_2	S_1	S_0		F	
A	B	C	D		
0	0	0	0	1	$F=1$ } I_0
0	0	0	1	1	
0	0	1	0	1	$F=1$ } I_1
0	0	1	1	1	
0	1	0	0	1	$F=\overline{D}$ } I_2
0	1	0	1	0	
0	1	1	0	0	$F=0$ } I_3
0	1	1	1	0	

S_2	S_1	S_0	S	F	
A	B	C	D	0	
1	0	0	0	0	$F=0$ } I_4
1	0	0	1	0	
1	0	1	0	1	$F=1$ } I_5
1	0	1	1	1	
1	1	0	0	0	$F=0$ } I_6
1	1	0	1	0	
1	1	1	0	1	$F=1$ } I_7
1	1	1	1	1	


1
I_0
I_1
I_2
I_3
I_4
I_5
I_6
I_7
0
1
2
3
4
5
6
7
8X1
MUX
F
S_2 S_1 S_0
A
B
C
$\overline{D}$
0

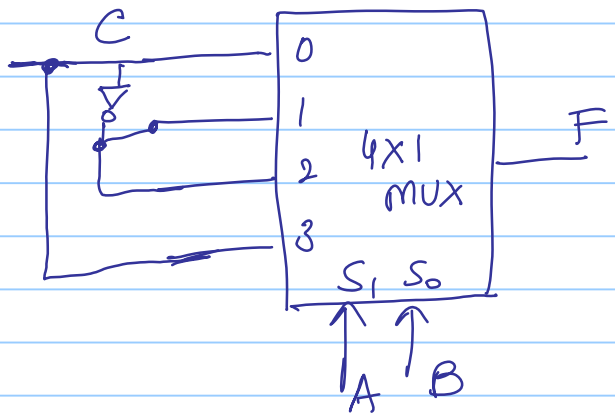

1
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16X1
MUX
F
S_3 S_2 S_1 S_0
A
B
C
D

Ques Use a 4x1 MUX to implement $F(A, B, C) = \sum m(1, 2, 4, 7)$

3 i/p variable

4x1 MUX → I_0, I_1, I_2, I_3

A	B	C	F	
0	0	0	0	$F = C \quad I_0$
0	0	1	1	
0	1	0	1	$F = \bar{C} \quad I_1$
0	1	1	0	
1	0	0	1	$F = \bar{C} \quad I_2$
1	0	1	0	
1	1	0	0	$F = C \quad I_3$
1	1	1	1	

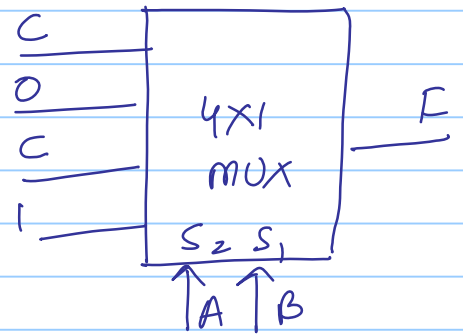


MUX implementation

Ques Implement the func^n $F(a,b,c) = ab + \bar{b}c$ using 4x1 MUX

$$F(a,b,c) = ab(c+\bar{c}) + (a+\bar{a})\bar{b}c$$
$$= abc + ab\bar{c} + a\bar{b}c + \bar{a}\bar{b}c$$
$$= \sum m(1,5,6,7)$$

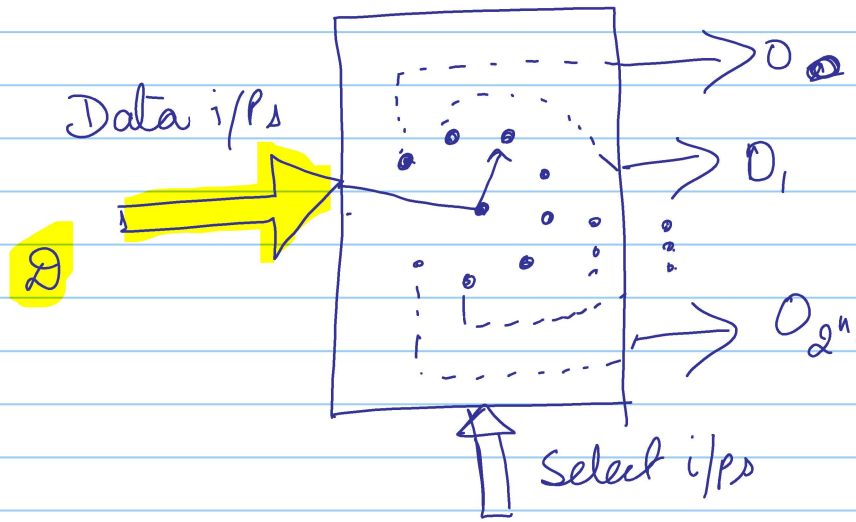
S_2 A	S_1 B	S_0 C	F	
0	0	0	0	F = C
0	0	1	1	
0	1	0	0	F = 0
0	1	1	0	
1	0	0	0	F = C
1	0	1	1	
1	1	0	1	F = 1
1	1	1	1	



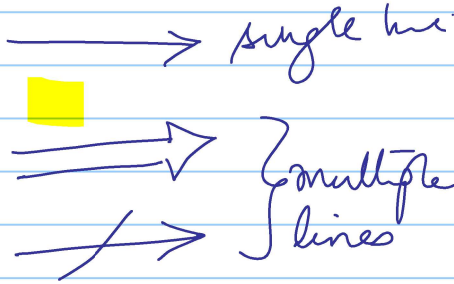
Demultiplexers (Data distributors)

$2^n \times 1 \rightarrow$ MUX

$1 \times 2^n \rightarrow$ DEMUX.



functional dy of a DEMUX



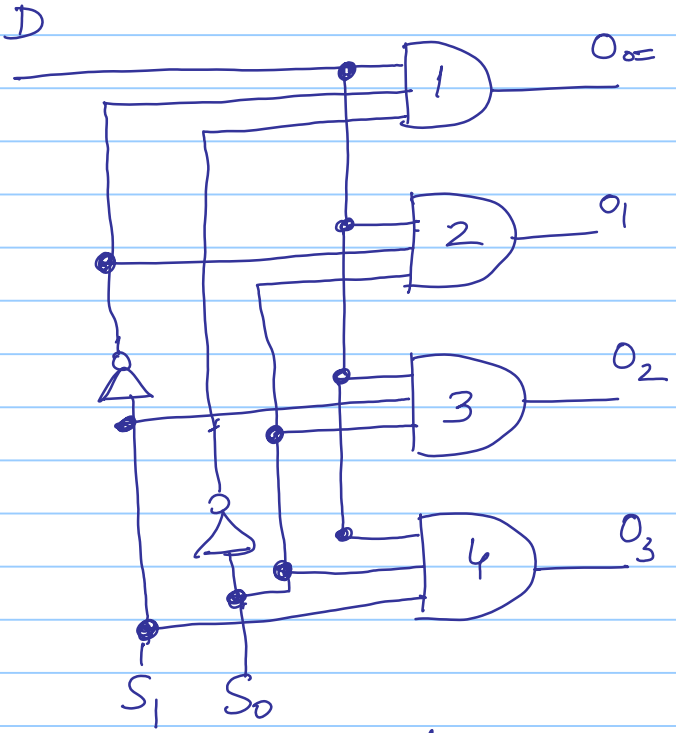
~~demux~~

1 line to 4 line Demultiplexer

Select code		Output			
S_1	S_0	O_3	O_2	O_1	O_0
0	0	0	0	0	D
0	1	0	0	D	0
1	0	0	D	0	0
1	1	D	0	0	0

truth table of 1×4 DEMUX

$$O_0 = D\,\bar{S}_1\,\bar{S}_0$$
$$O_1 = D\,\bar{S}_1\,S_0$$
$$O_2 = D\,S_1\,\bar{S}_0$$
$$O_3 = D\,S_1\,S_0$$

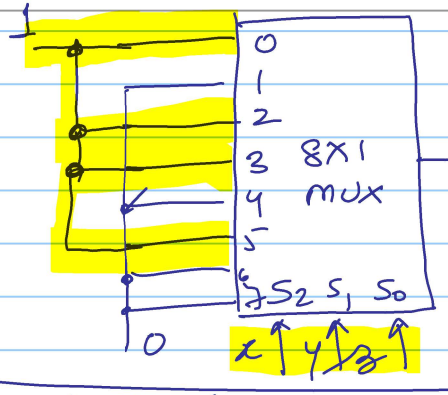


Logic diagram

Ques Implement $f(x, y, z) = \Sigma m(0, 2, 3, 5)$

Note Title 6/19/2021

Select lines 8 : 1 MUX



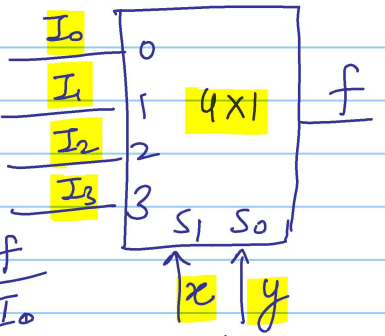
x y z

Implement 4 : 1 MUX

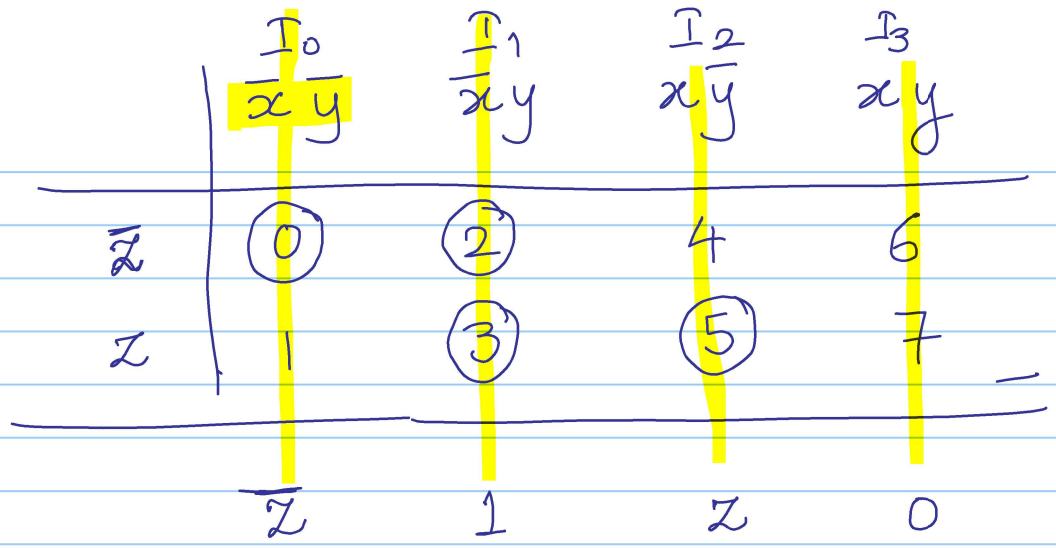
	x	y	z	f	
0	0	0	0	1	$\bar{z}$
1	0	0	1	0	
2	0	1	0	1	1
3	0	1	1	1	
4	1	0	0	0	z
5	1	0	1	1	
6	1	1	0	0	0
7	1	1	1	0	

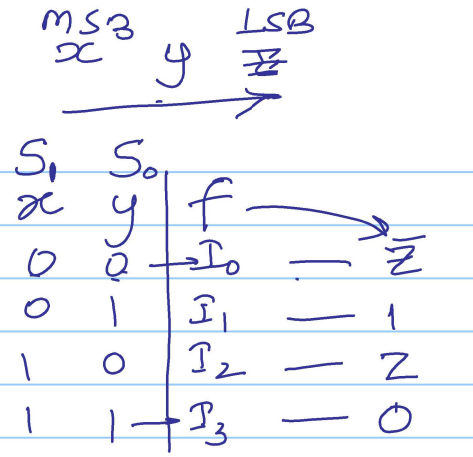
If x, y are the select lines

	I_0	I_1	I_2	I_3
	$\bar{x}\bar{y}$	$\bar{x}y$	$x\bar{y}$	xy
$\bar{z}$	0	2	4	6
z	1	3	5	7
	$\bar{z}$	1	z	0



x	y	f
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

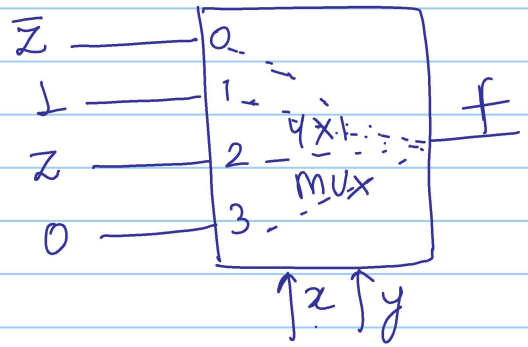




$f = \Sigma(0, 2, 3, 5)$

$z + \bar{z} = 1$

$= \bar{z}$



x y

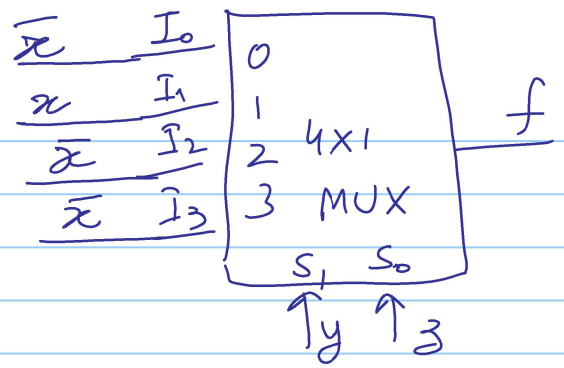
If $y, 3$ are select lines

	I_0 $\bar{y}\bar{3}$	I_1 $\bar{y}3$	I_2 $y\bar{3}$	I_3 $y3$
$\bar{x}$	(0)	1	(2)	(3)
x	4	(5)	6	7
	$\bar{x}$	x	$\bar{x}$	$\bar{x}$

	I_0 $\bar{x}\bar{z}$	I_1 $\bar{x}z$	I_2 $x\bar{z}$	I_3 xz
$\bar{y}$	(0)	1	4	(5)
y	(2)	(3)	6	7
	1	y	0	$\bar{y}$

~~$y3$~~ ~~S_1S_0~~

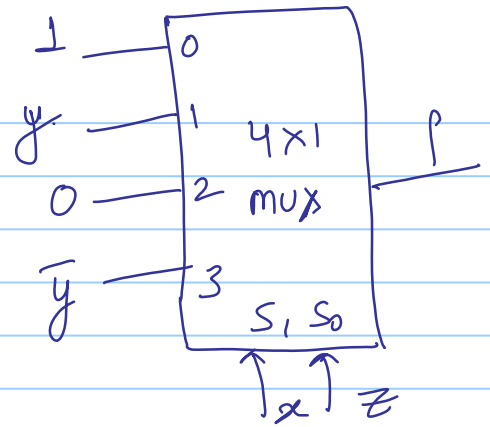
$x\ y\ 3$



y	3	f
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

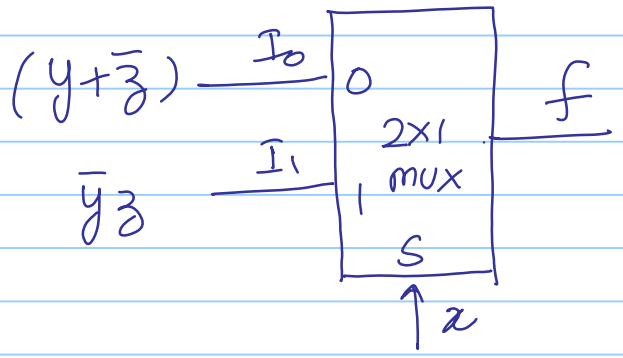
$x\ y3$

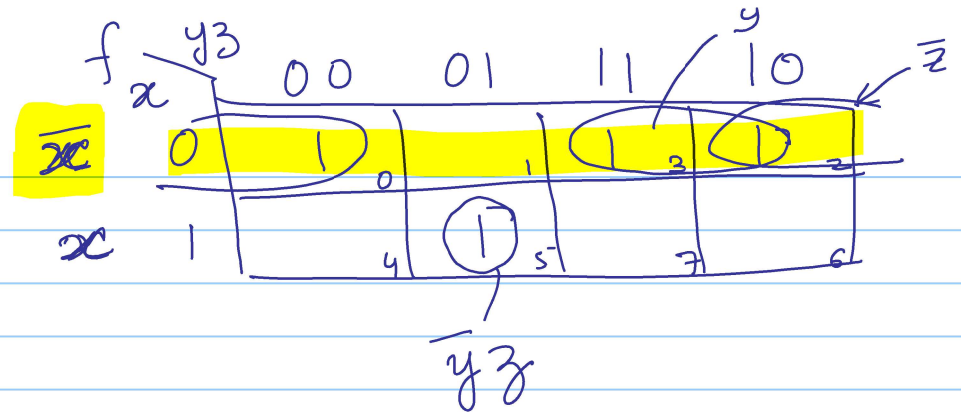
S_1	S_0	f
x	z	
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3



Implement using 2×1 mux

select lines :- 1
i/p lines :- 2
o/p line :- 1





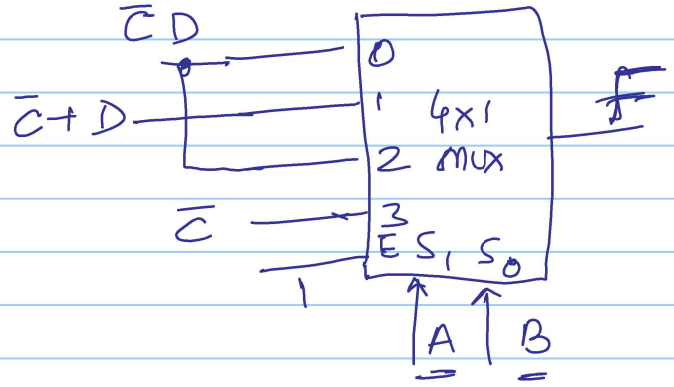
$f = \Sigma m(0, 2, 3, 5)$

$I_0 = y + \bar{z}$

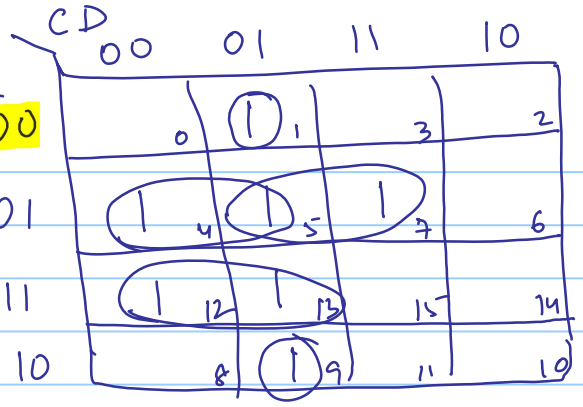
$I_1 = \bar{y}z$

Ques $F(A, B, C, D) = \Sigma m(1, 4, 5, 7, 9, 12, 13)$ using 4×1 mux

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



$S_1 S_0 = AB$



$I_0 = \bar{C}D$

$I_1 = \bar{C} + D$

$I_2 = \bar{C}D$

$I_3 = \bar{C}$

H.W Implement full adder using 4×1 MUX

$S_1 S_0 = AB$

S_1 $I_0 = C_i$

$I_1 = \bar{C}_i$

$I_2 = \bar{C}_i$

$I_3 = C$

C_o $I_0 = 0$

$I_1 = C$

$I_2 = C$

$I_3 = 1$

Note Title 6/22/2021

Programmable logic devices (PLDs)

Fixed func^{n}. ← Encoder, Decoder, MUX, DEMUX

(1)

1) Large board space requirements
2) Large power requirements
3) Lack of security
4) Additional cost, space, power req. etc.

(2)

Application specific Integrated circuits (ASICs)

To overcome the disadvantages of fixed-func^{n} ICs.

Advantages

1. Reduced power and space requirement
2. Considerable cost reduction if produced in large volumes
3. Large reduction in size
4. A- Impossible to copy

Disadvantages are

1. Initial development cost may be high
2. Testing method will ↑ cost & time.

Programmable logic devices

PLDs are programmable logic device is an IC that is user configurable and is capable of implementing logic func^{n}.

LSI chip that contains a 'regular' str. and allow the designer to customize it for any specific app.

- Large no of gates
- flip-flop
- register

Read-Only Memory (ROM)

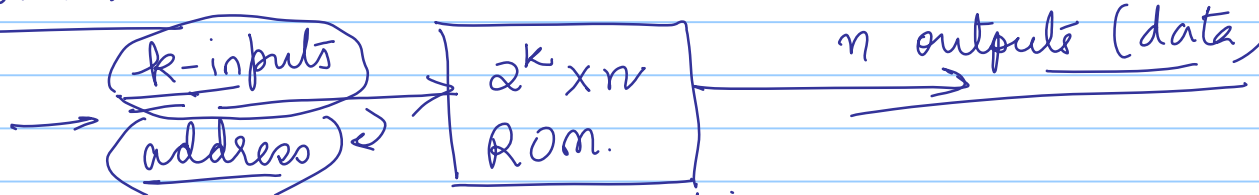
permanent binary information is stored.

A ROM which can be programmed is called a PROM

programming.

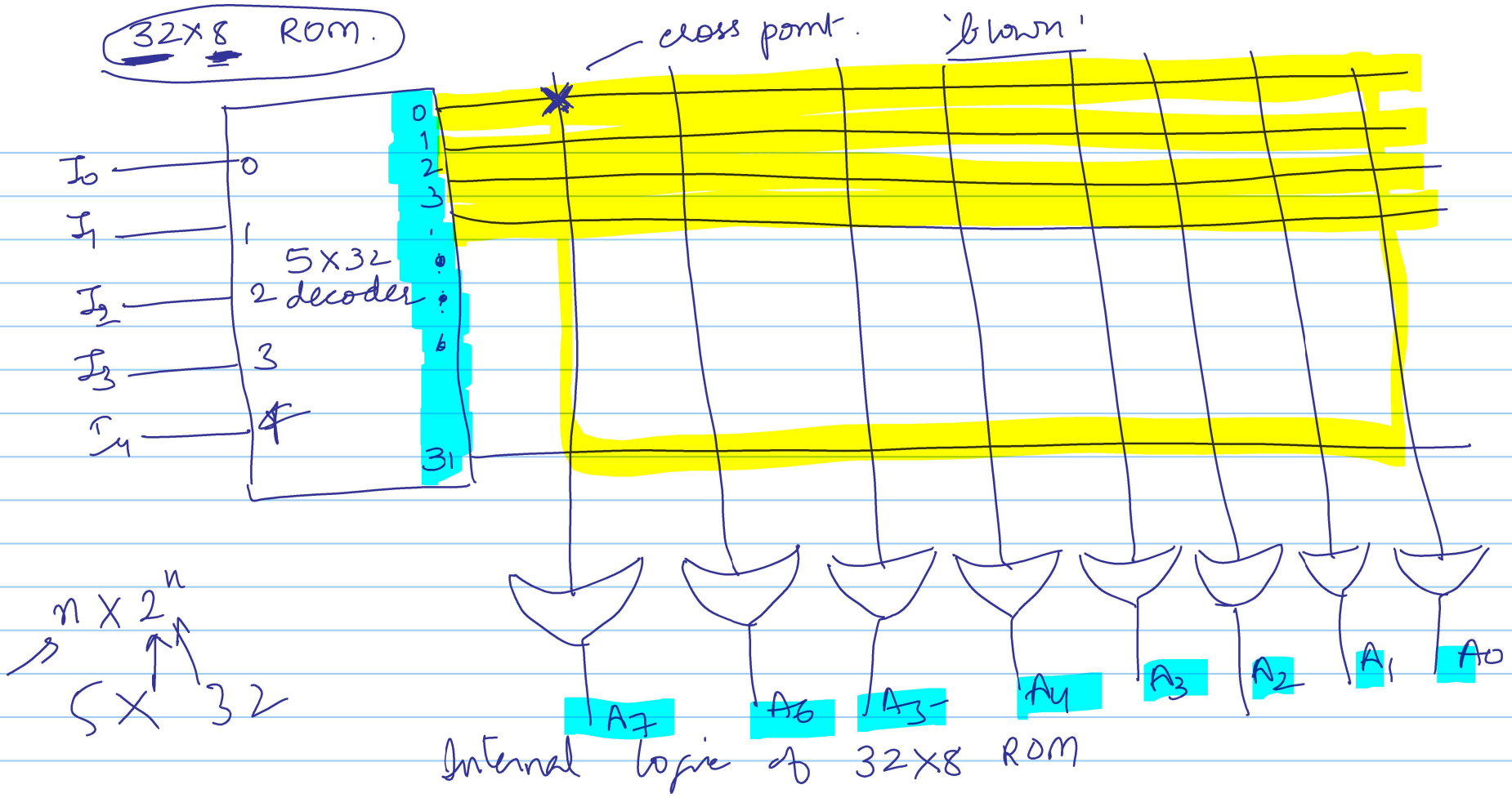
A	B	C	
0	0	0	→ word - Hell
0	0	1	→ word. Wrld
0	1	0	- Hi
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Organization



ROM block diagram.

ROM → No data inputs


32x8 ROM.
cross point.
'blown'
I_0
I_1
I_2
I_3
I_4
0
1
2
3
5x32 decoder
31
$n \times 2^n$
5 x 32
A_7
A_6
A_5
A_4
A_3
A_2
A_1
A_0
Internal logic of 32x8 ROM

$32 \times 8 = 256$ internal connections

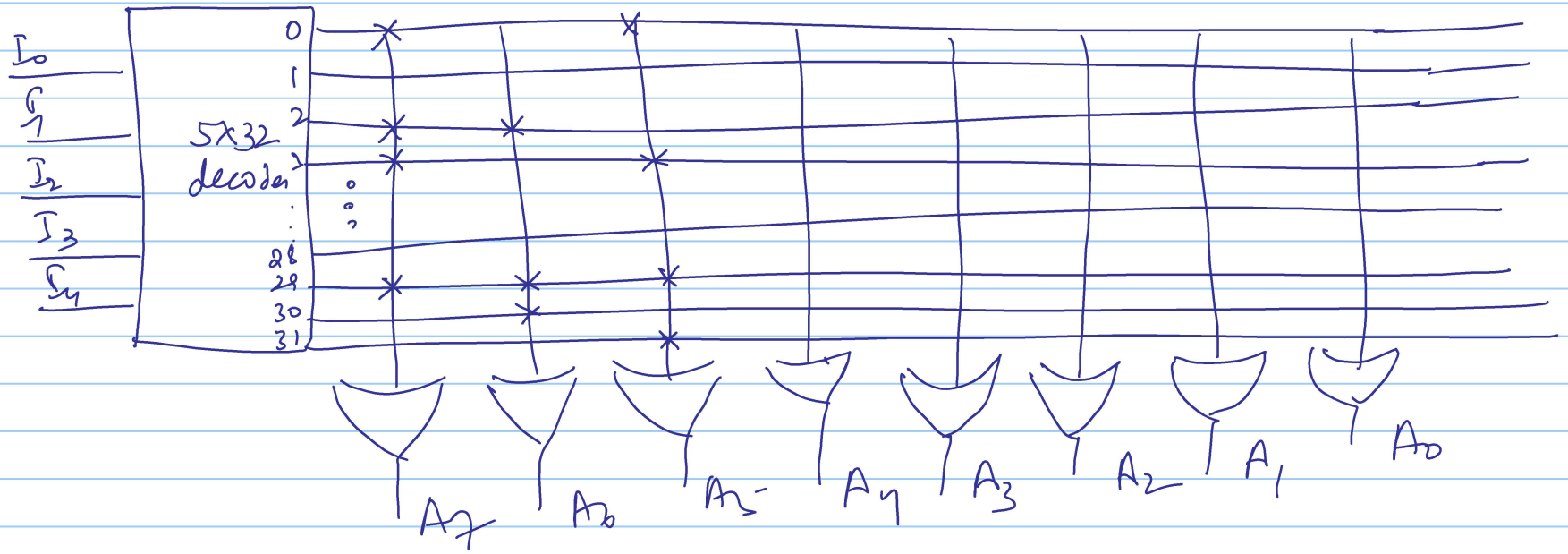
In general $2^k \times n$ ROM will have an internal $k \times 2^k$ decoder and n OR gates.

Each OR gate has 2^k inputs.

ROM truth table (Partial)

Inputs					Outputs							
I_4	I_3	I_2	I_1	I_0	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0
0	0	0	0	0	1	0	1	1	0	1	1	0
0	0	0	0	1	0	0	0	1	1	1	0	1
0	0	0	1	0	1	1	0	0	0	1	0	1
0	0	0	1	1	1	0	1	1	0	0	1	0
			⋮									

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



(1) Impl. of FA using MUX 8×1
4×1
2×1

(2) " " DEMUX

(3) " F.S " MUX
DEMUX

(4)

(5) Design a combination ckt using a ROM. The circuit accepts a 3-bit number and generate an output binary number equal to the square of the i/p no.

(6) Give the logic implementation of a 32×4 bit ROM using a

decoder of a suitable size.

(7) a 8 x 4 bit ROM

2	49	1
2	24	0
	12	0
	6	0
	3	1
	1	

110001 — 6 bits

3 → i/p

6 → o/p bits

ROM

010 → 000100

011 → 001001

111 → 110001