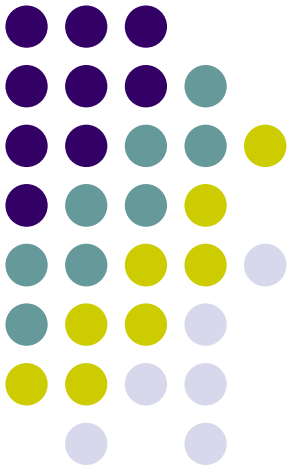
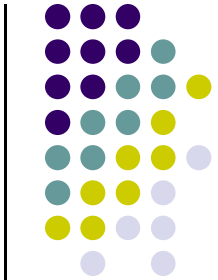
Digital Logic

From Boolean algebra to implementing logic circuits

Tutorials

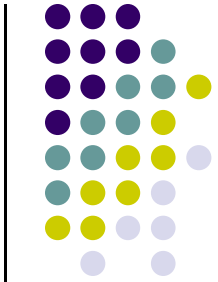


Outline



- Boolean algebra
- Logic gates
- Truth table
- Logic expressions
- Implementing digital logic using gates
- Karnaugh map – logic minimization
- Using NAND, NOR gates only.
- Binary numbers
- ADC – Analog to digital conversion

Boolean Algebra



- Boolean variables
  - Can have values of either TRUE (1), FALSE(0)
  - Can represent
    - either truth or falsehood of a statement
    - ON or OFF states of a switch
    - High(5V) or low(0V) of a voltage level

Boolean operation

AND (x.y)

x	y	x.y
0	0	0
0	1	0
1	0	0
1	1	1

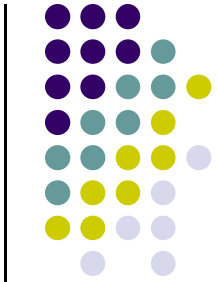
OR(x+y)

x	y	x+y
0	0	0
0	1	1
1	0	1
1	1	1

NOT(x)

x	$\bar{x}$
0	1
1	0

Basic laws of Boolean Algebra



$$0 + A = A$$

$$1 + A = 1$$

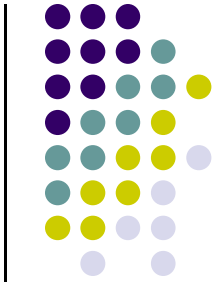
$$A + A = A$$

$$A + \overline{A} = 1$$

$$0.A = 0$$

$$1.A = A$$

Basic laws of Boolean Algebra



$$A.A = A$$

$$A.\overline{A} = 0$$

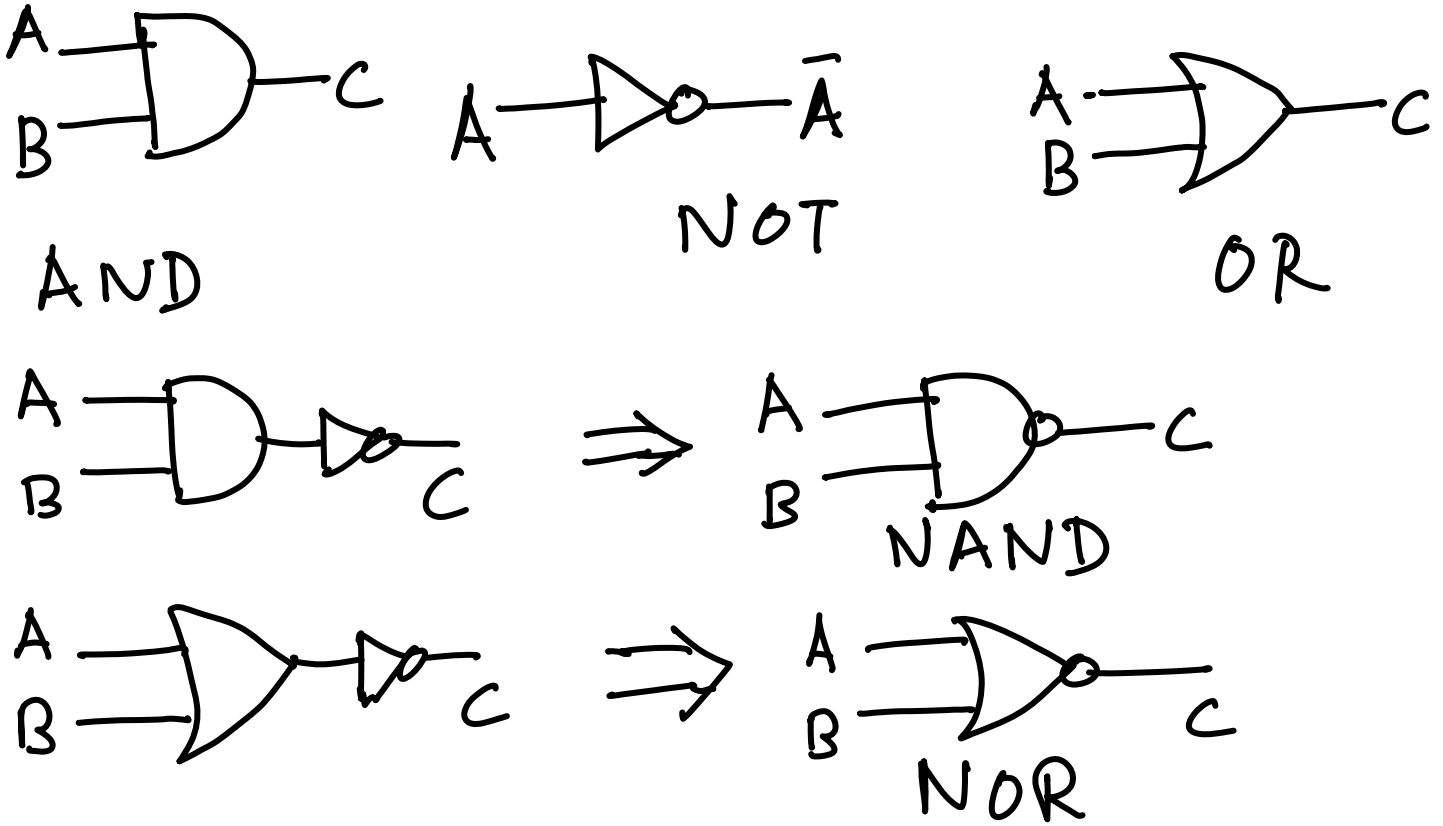
$$A = \overline{\overline{A}}$$

$$A + B = B + A$$

$$A.B = B.A$$

Logic Gates

- Electronic circuits used to perform boolean logic operations are known as logic gates.

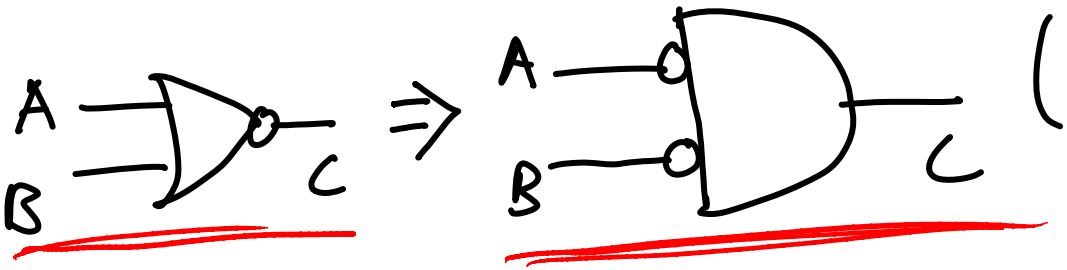


DeMorgan's Theorem

$\overline{A+B} = \overline{A}.\overline{B}$ → Inverting a sum results in product of inverted signals.

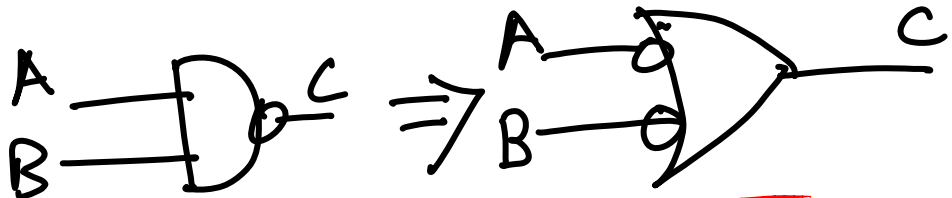
$\overline{A.B} = \overline{A}+\overline{B}$

NOR : $\overline{A+B} = \overline{A}\cdot\overline{B}$



(Alternate NOR)

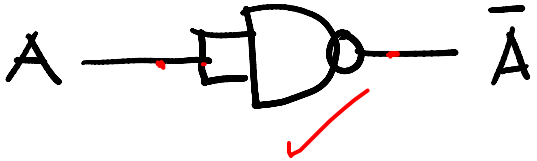
NAND $\overline{A.B} = \overline{A}+\overline{B}$



(Alternate NAND).

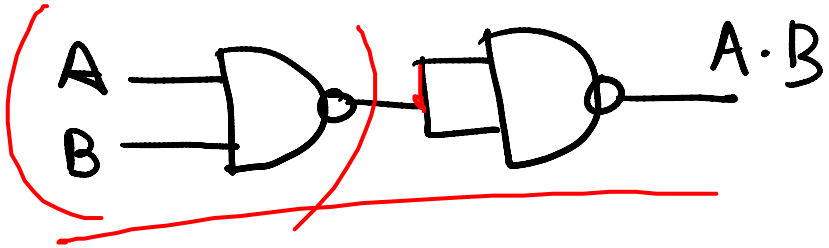
Universality of NAND gates

- AND, OR, NOT gates can be obtained from NAND gate

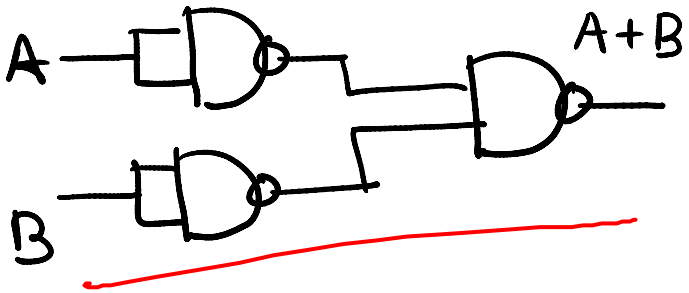


as $\overline{A \cdot A} = \bar{A}$

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



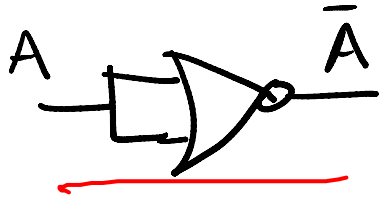
$A + B = \overline{\overline{A + B}} = \overline{\bar{A} \cdot \bar{B}}$



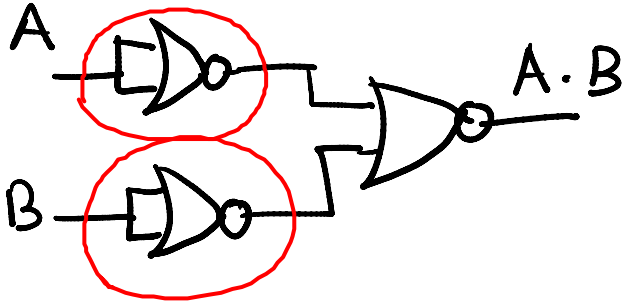
Universality of NOR gates

- AND, OR, NOT gates can be obtained from NOR gate

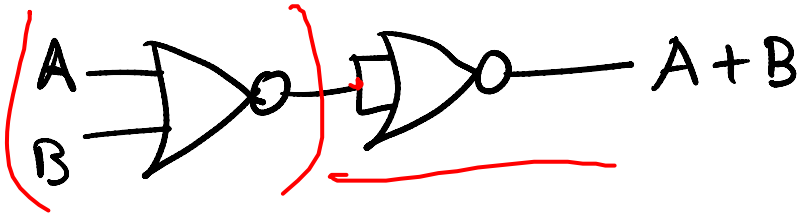
$\overline{A+A} = \bar{A}$ as $A+A=A$;



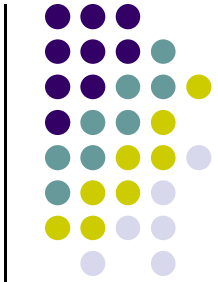
$A \cdot B = \overline{\overline{A \cdot B}} = \overline{\bar{A} + \bar{B}}$;



$A+B = \overline{\overline{A+B}}$;



Truth table and logic expression



- Consider a logic circuit having two inputs A, B and one output C.
- The output becomes 1 when only one input is TRUE (1), but is FALSE(0) other wise.

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

TRUTH table

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

SUM OF PRODUCTs expression

- For each row in the Truth table, with '1' at the output column, we can create a PRODUCT term of all inputs.
  - Take the input, if the entry for it is 1
  - Take the input with a bar, if the entry for it is 0
- The logic expression will be SUM of these PRODUCTs (SOP)

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

$\Rightarrow \bar{A} \cdot B$

$A \cdot \bar{B}$

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

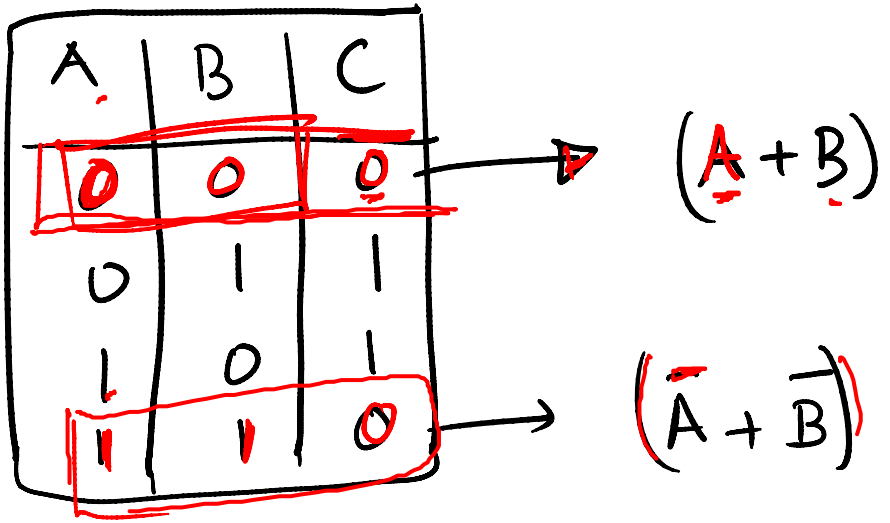
$C = \bar{A} \cdot \bar{B} + \bar{A} \cdot B$

SOP

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

PRODUCT OF SUMs terms

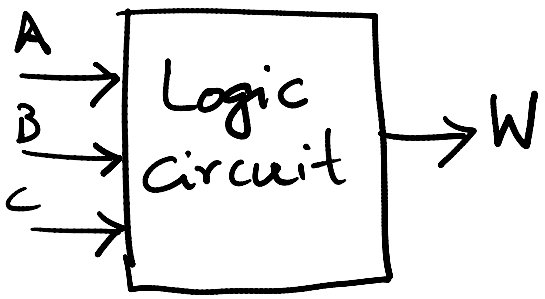
- For each row in the Truth table, with ‘0’ at the output column, we can create a SUM term of all inputs.
  - Take the input, if the entry for it is 0
  - Take the input with a bar, if the entry for it is 1
- The logic expression will be PRODUCT of SUMs (POS).



POS

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

Truth table – Logic expressions



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

$(A+B+C)$

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

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

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

$\overline{A}.\overline{B}.C$

$\overline{A}.B.\overline{C}$

$A.\overline{B}.\overline{C}$

$A.\overline{B}.C$

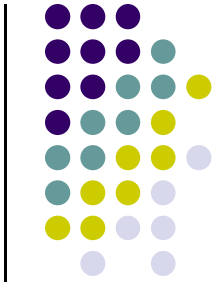
Sum of Products (SOP)

$$W = \overline{A}.\overline{B}.C + \overline{A}.B.\overline{C} + A.\overline{B}.\overline{C} + A.\overline{B}.C$$

Product of Sums (POS)

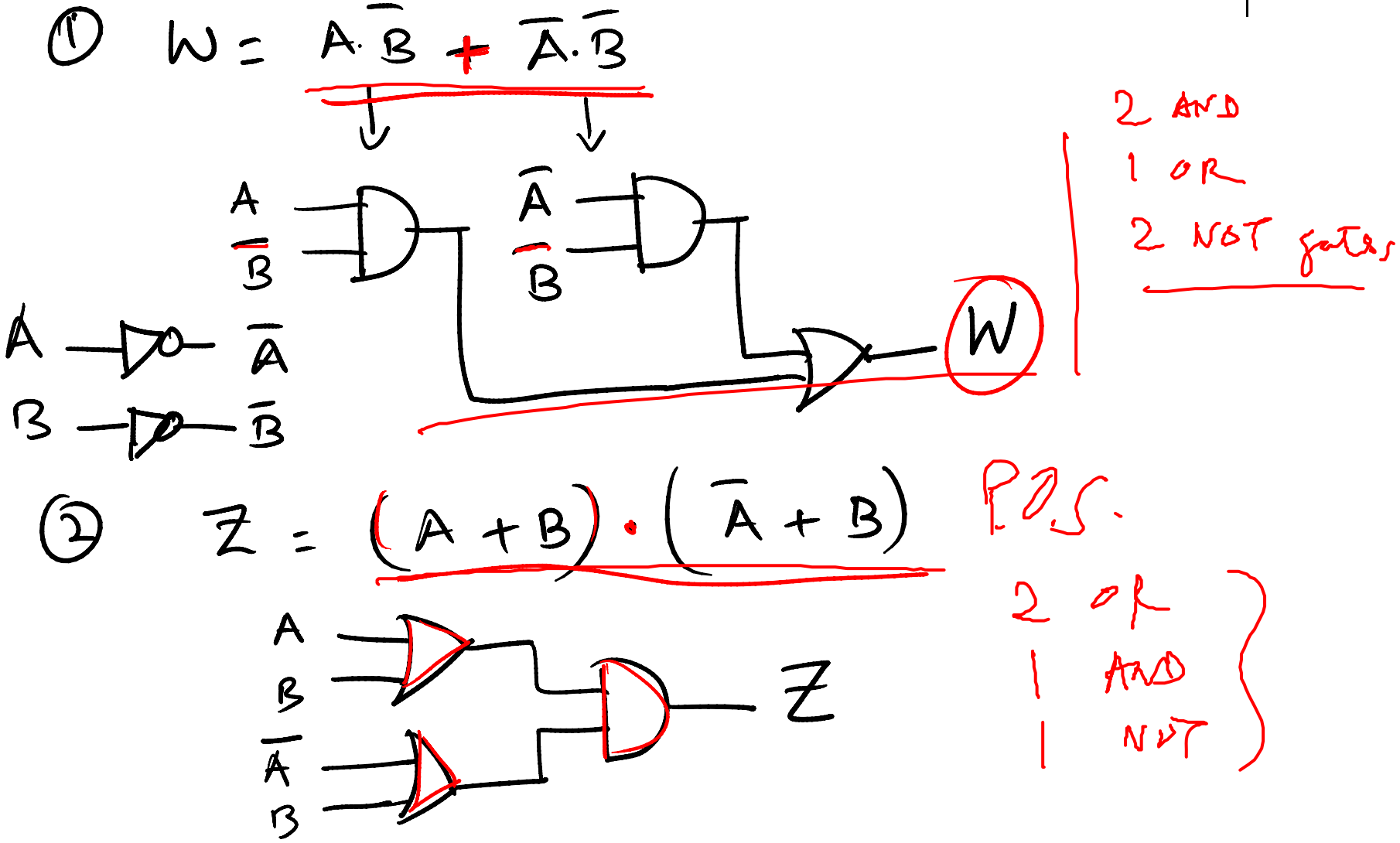
$$W = (A+B+C).(A+\overline{B}+\overline{C}).(\overline{A}+\overline{B}+C).(\overline{A}+\overline{B}+\overline{C})$$

Design logic circuits: Implementing logic expressions

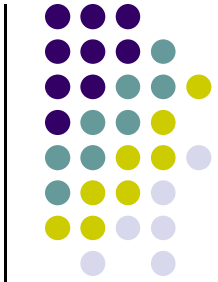


- Logic expressions are in either Sum of Product (SOP) or Product of Sums (POS) form
- Implement the sum term using OR gates and product term using AND gates
- Hence, using AND, OR and NOT gates we can implement any logic expressions

Implementing sop and pos



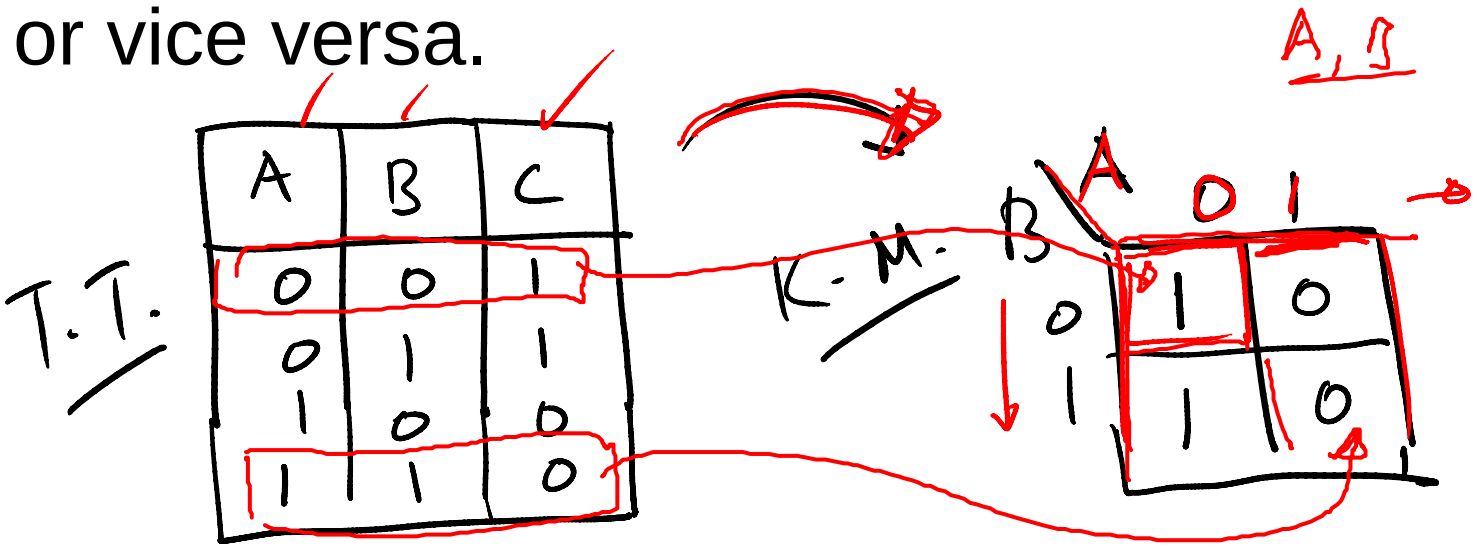
Design logic circuits: Implementing logic expressions



- As designer, our aim will be to use minimum number of gates.
- We need to minimize the SOP and POS expressions.
- There are two ways for this:
  - Boolean algebra laws
  - Graphical method – Karnaugh map

Karnaugh Maps

- It is a graphical representation of the truth table – each row in truth table corresponds to one cell in the K-map.
- The adjacent (horizontal or vertical) cells have only one variable changing from 0 to 1 or vice versa.



Truth table to K-map more examples

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

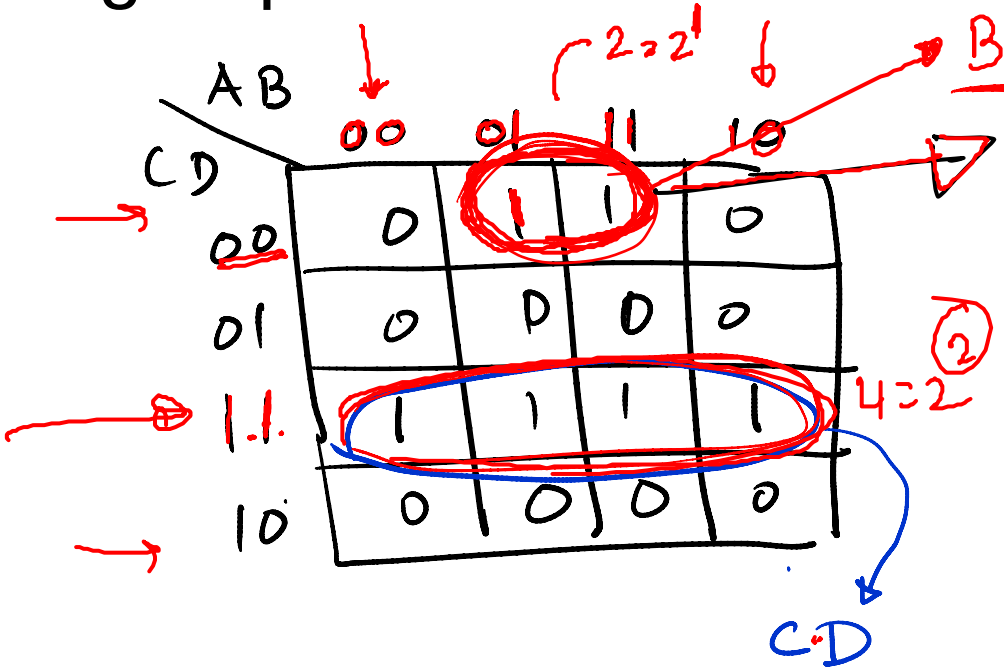
AB \ C	00	01	11	10
0	0	0	0	1
1	0	1	0	1

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

AB
0 0
0 1
1 0
1 1

Grouping adjacent cells

- Group adjacent cells containing same values
- Group in powers of 2 (2^n) i.e. in 2, 4 ,8 etc.
- The bigger the group smaller the term – a group with 2^n terms has n terms less



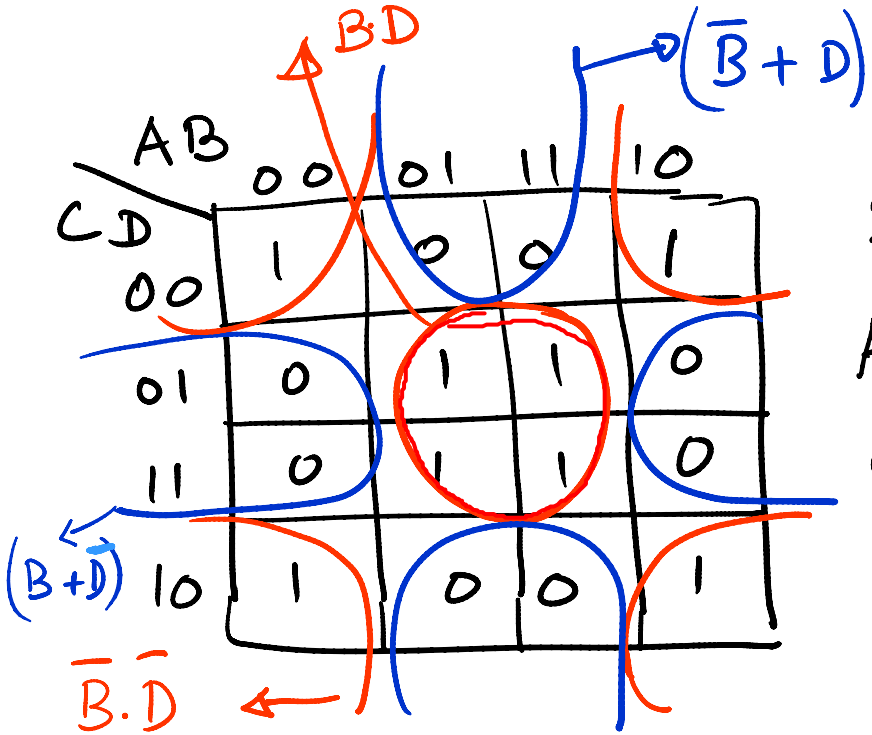
$\bar{A}B\bar{C}\bar{D} + AB\bar{C}\bar{D} =$

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

$= 1 \cdot B\bar{C}\bar{D} = B\bar{C}\bar{D}$

Variables with both 0 and 1 go.

Adjacent cells



Leftmost column AB = 00

Rightmost column AB = 10

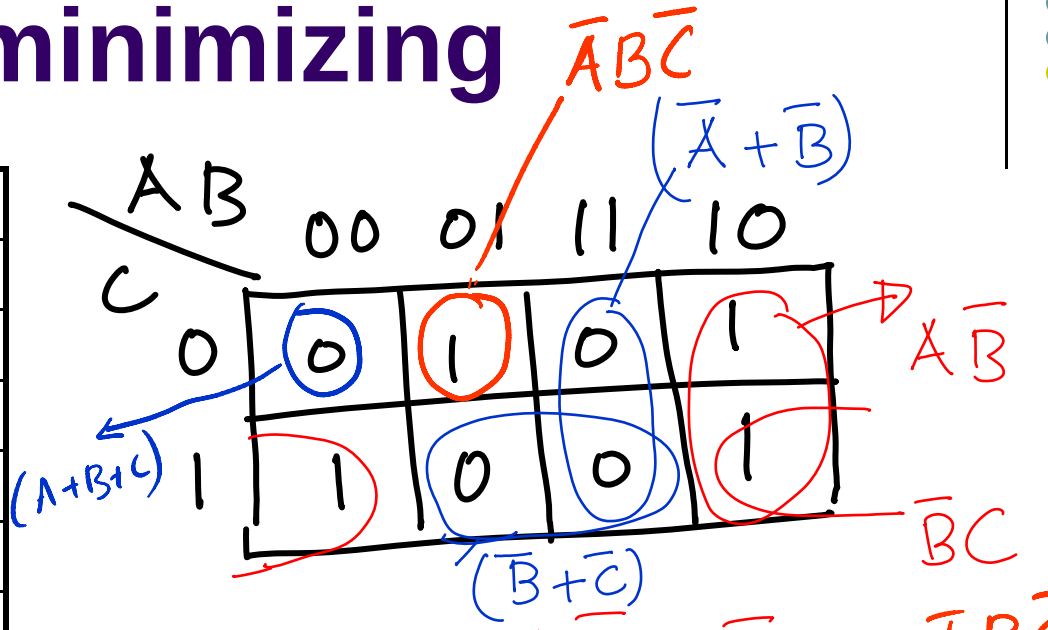
As only A changes from 0 to 1, the left most column and right most column are adjacent

* Topmost row and bottom most row are also adjacent.
* The four corners can be grouped together

$MSOP = B.D + \bar{B}.\bar{D}$, $MPOS = (\bar{B}+D).(B+\bar{D})$

K-Map for minimizing

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



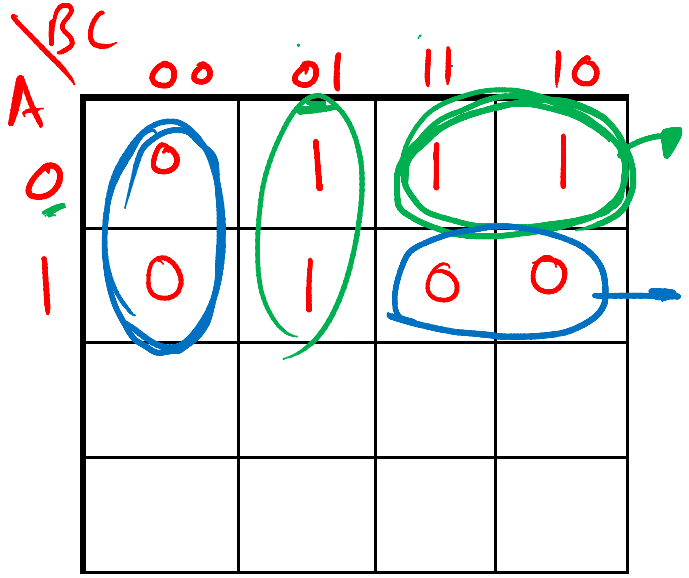
$MSOP = \bar{A}\bar{B} + \bar{B}C + \bar{A}B\bar{C}$

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

* Try to form as big a group as possible
* One cell can be part of more than one group if necessary.

Example : Truth table to Karnaugh Map

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

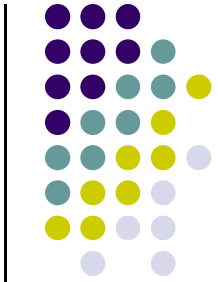


MSOP, MPOS

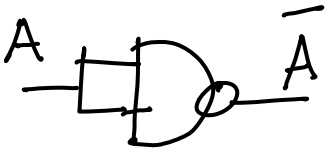
$Z_{MSOP} = \bar{A} \cdot B + \bar{B} \cdot C$

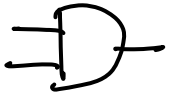
$Z_{MPOS} = (\bar{A} + \bar{B}) \cdot (B + C)$

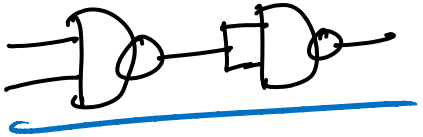
Logic design using only NAND or only NOR gates

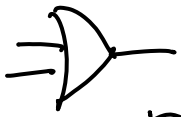


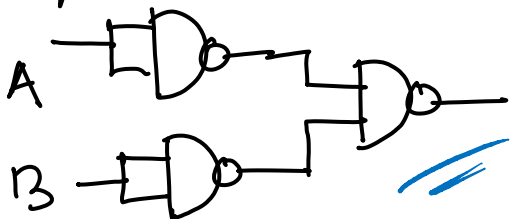
- NAND, NOR gates are universal gates
- The AND, OR, NOT can be converted into NAND/NOR gates

NOT can be made of NAND gate 

$A \cdot B = \overline{\overline{A \cdot B}}$  can be replaced by

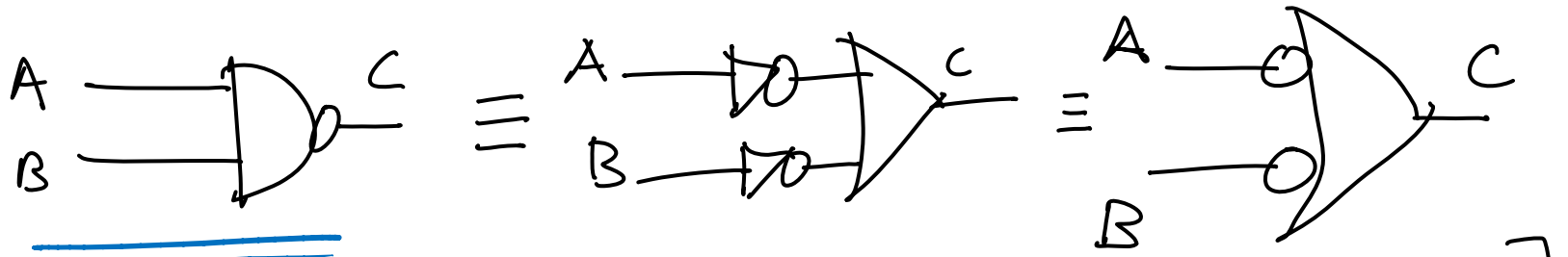


$A + B = \overline{\overline{A+B}} = \overline{\bar{A} \cdot \bar{B}}$  can be replaced by



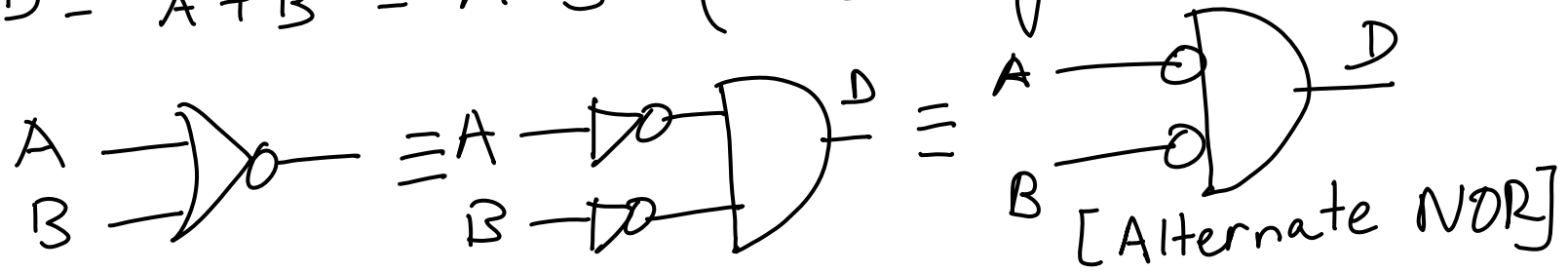
Alternate NAND NOR symbols

$C = \overline{A \cdot B} = \bar{A} + \bar{B}$ (DeMorgan's law).



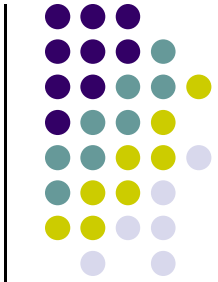
[Alternate NAND]

$D = \overline{A + B} = \bar{A} \cdot \bar{B}$ (DeMorgan's law)



[Alternate NOR]

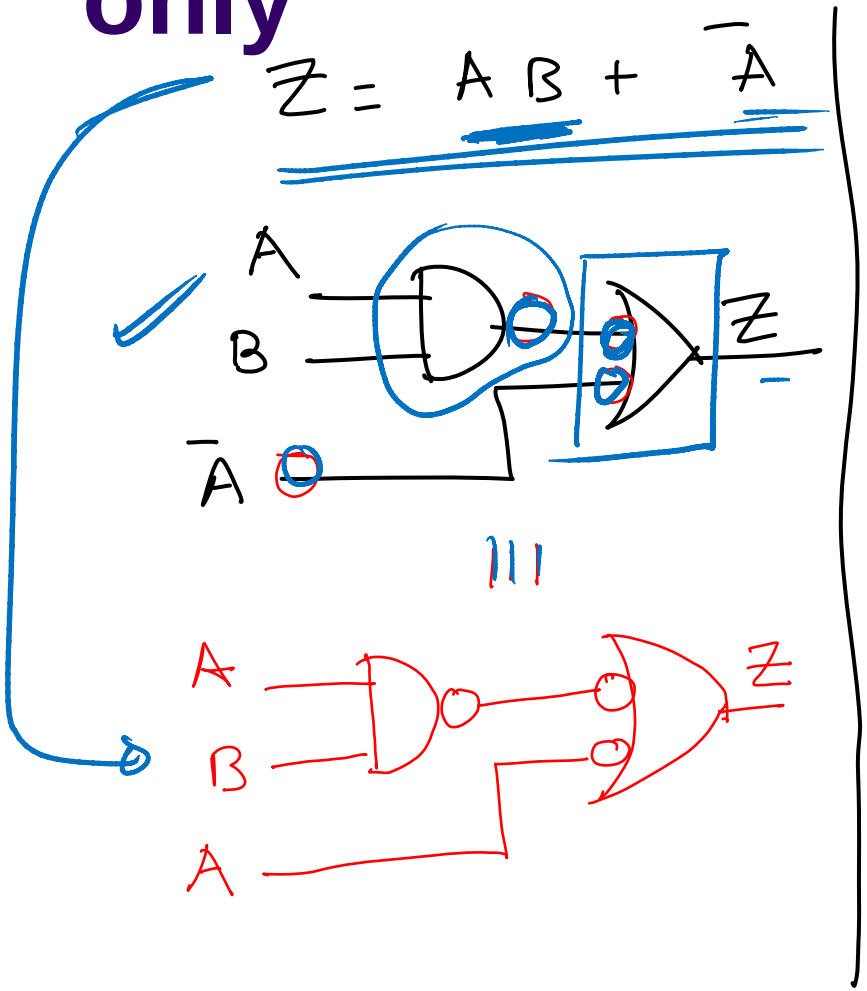
Converting Circuit with AND, OR to NAND or NOR only



- We shall use the alternate gate representation where ever necessary to do the conversion

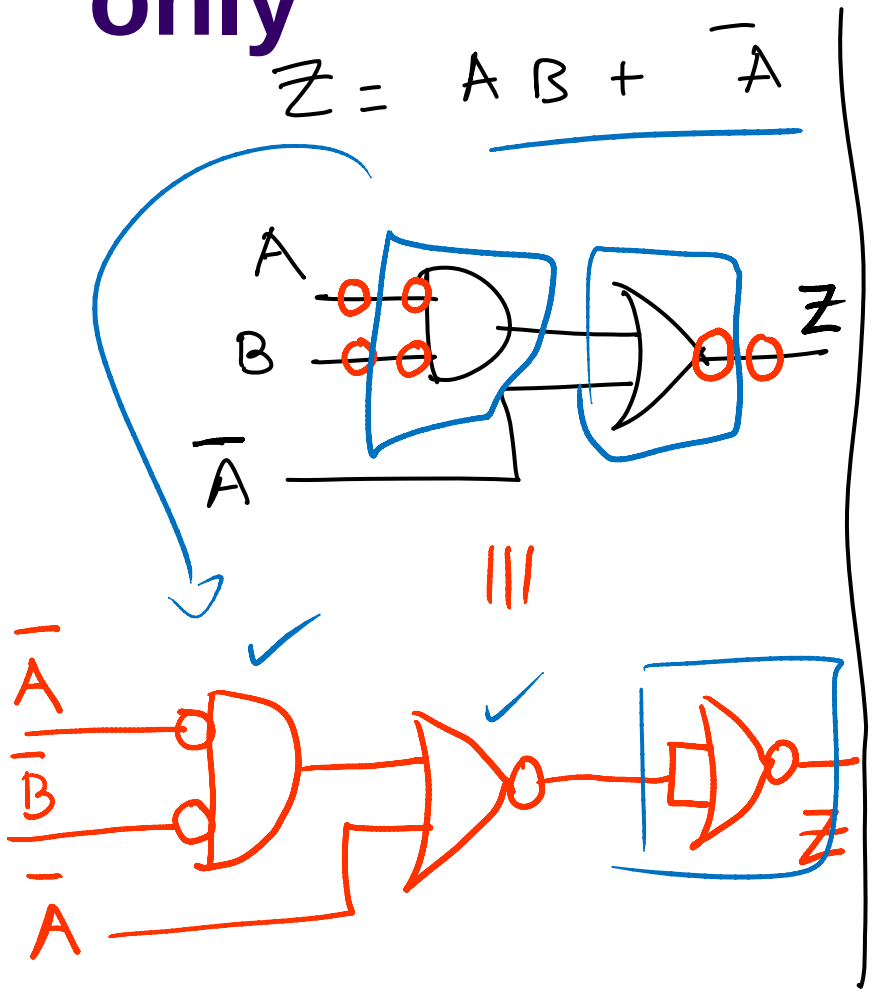
- Add bubbles(inverters) – can put two bubbles at any point without changing the logic
  - at the input of AND gate to convert AND to NOR
  - at the output of AND to convert it to NAND
  - at the input of OR gate to convert OR to NAND
  - at the output of OR to convert OR to NOR

From AND, OR , NOT to NAND only



- For AND gate, put an inverter (bubble) at its output.
- For OR gate, put bubbles at the inputs.
- At any point, we can put two bubbles without changing anything.

From AND, OR , NOT to NOR only



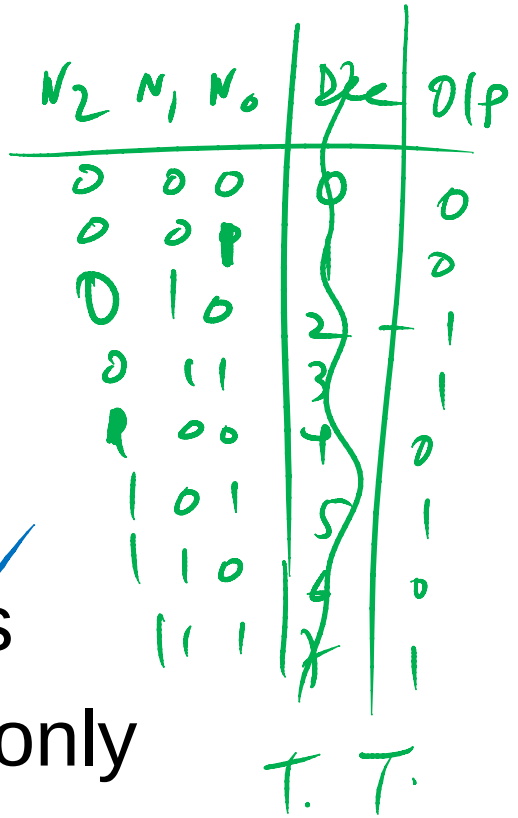
- For OR gate, put an inverter (bubble) at its output.
- For AND gate, put bubbles at the inputs.
- At any point, we can put two bubbles without changing anything.

Tutorial Q1

Design a circuit which outputs 1 when a 3 bit input represents a prime number and outputs 0 otherwise.

Show the:

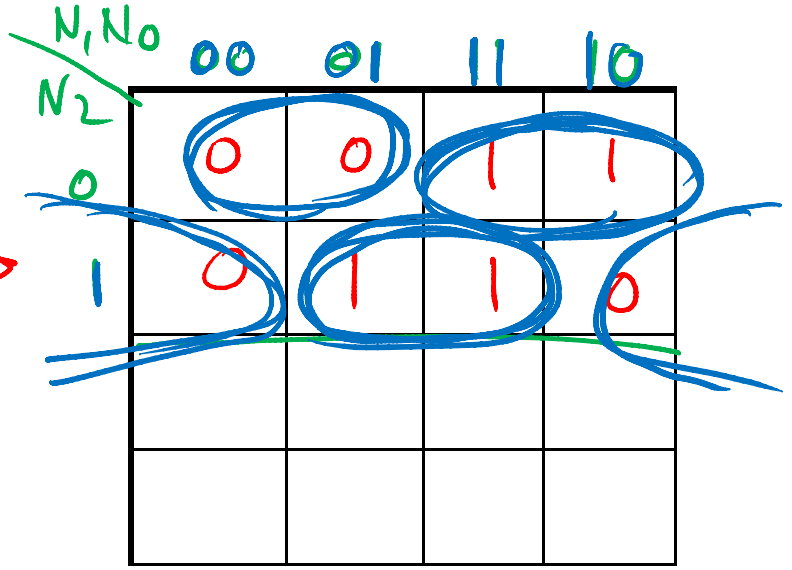
a) TRUTH table
b) SOP and POS expressions
c) Karnaugh map
d) MSOP and MPOS expressions
e) Implement using NAND gates only



T. T.

	N_2	N_1	N_0	Z
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	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	

3 bit . prime number



$$SOP = \bar{N_2} N_1 \bar{N_0} + \bar{N_2} \cdot N_1 \cdot N_0 + N_2 \cdot \bar{N_1} N_0 + N_2 \cdot N_1 \cdot N_0$$

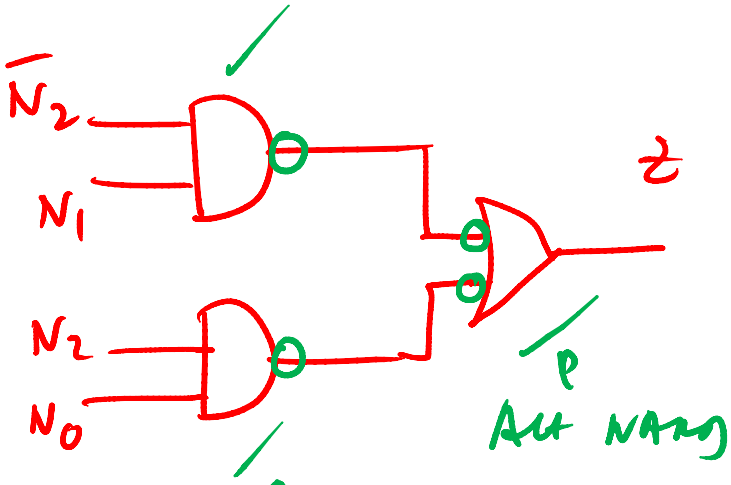
$$POS = (N_2 + N_1 + N_0) \cdot (N_2 + N_1 + \bar{N_0}) \cdot (\bar{N_2} + N_1 + N_0) \cdot (\bar{N_2} + \bar{N_1} + N_0)$$

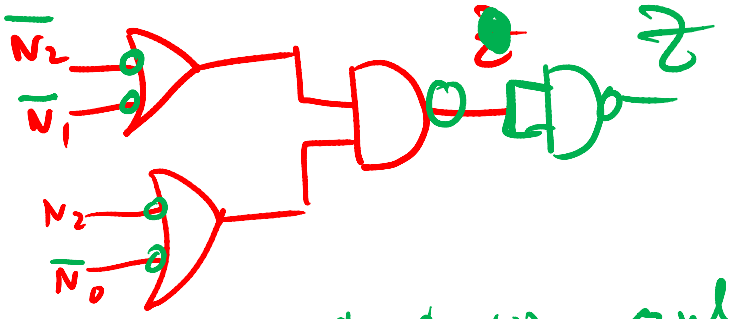
$$MSOP = \bar{N_2} \cdot N_1 + N_2 \cdot N_0$$

$$MPOS = (N_2 + N_1) \cdot (\bar{N_2} + N_0)$$

$Z_{mSOP} = \bar{N}_2 N_1 + N_2 \cdot N_0$ $\quad Z_{mPOS} = (N_2 + N_1) \cdot (\bar{N}_2 + N_0)$

NAND only





Conventional NAND

NAND only

Tutorial Q2

Design a circuit which outputs 0 when a 4 bit input represents a number greater than 5 and less than 13, and outputs 1 otherwise.

Show the:

o/p = 0 NTS a/w < 13

= 1 otherwise.

a) TRUTH table

b) Karnaugh map

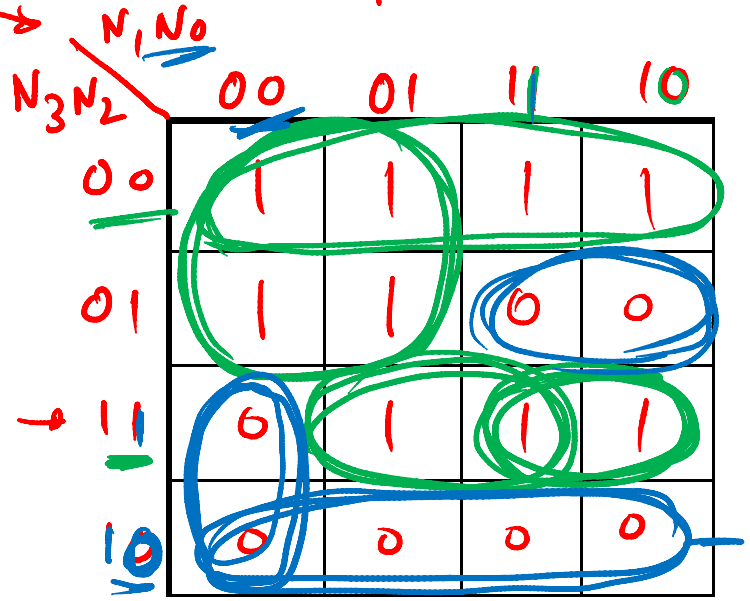
c) MSOP and MPOS expressions

d) Implement using NOR gates only

T. T.

	N_3	N_2	N_1	N_0	Z
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	1
3	0	0	1	1	1
4	0	1	0	0	1
5	0	1	0	1	1
6	0	1	1	0	0
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	0
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	1

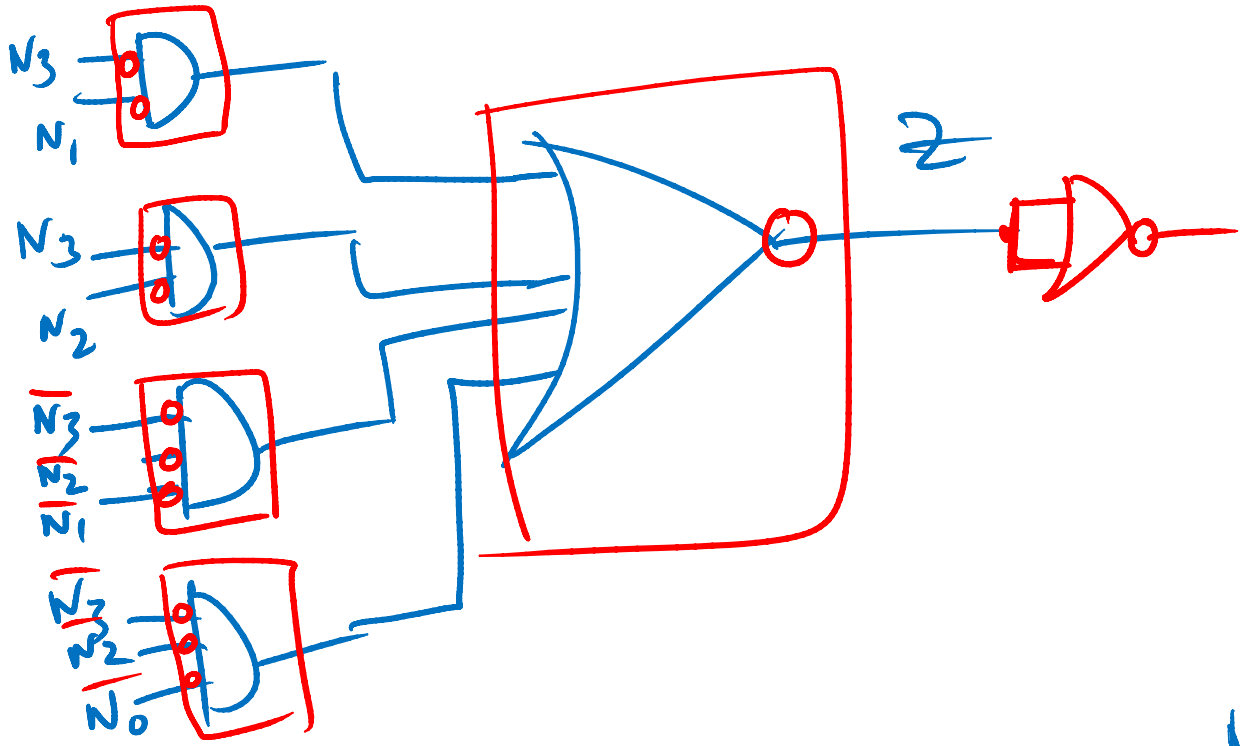
K - map



$$Z_{msop} = \overline{N_3} \cdot \overline{N_1} + \overline{N_3} \cdot \overline{N_2} + N_3 N_2 N_1 + N_3 N_2 N_0$$

$$Z_{mpos} = (\overline{N_3} + N_2) \cdot (\overline{N_3} + N_1 + N_0) \cdot (N_3 + \overline{N_2} + \overline{N_1})$$

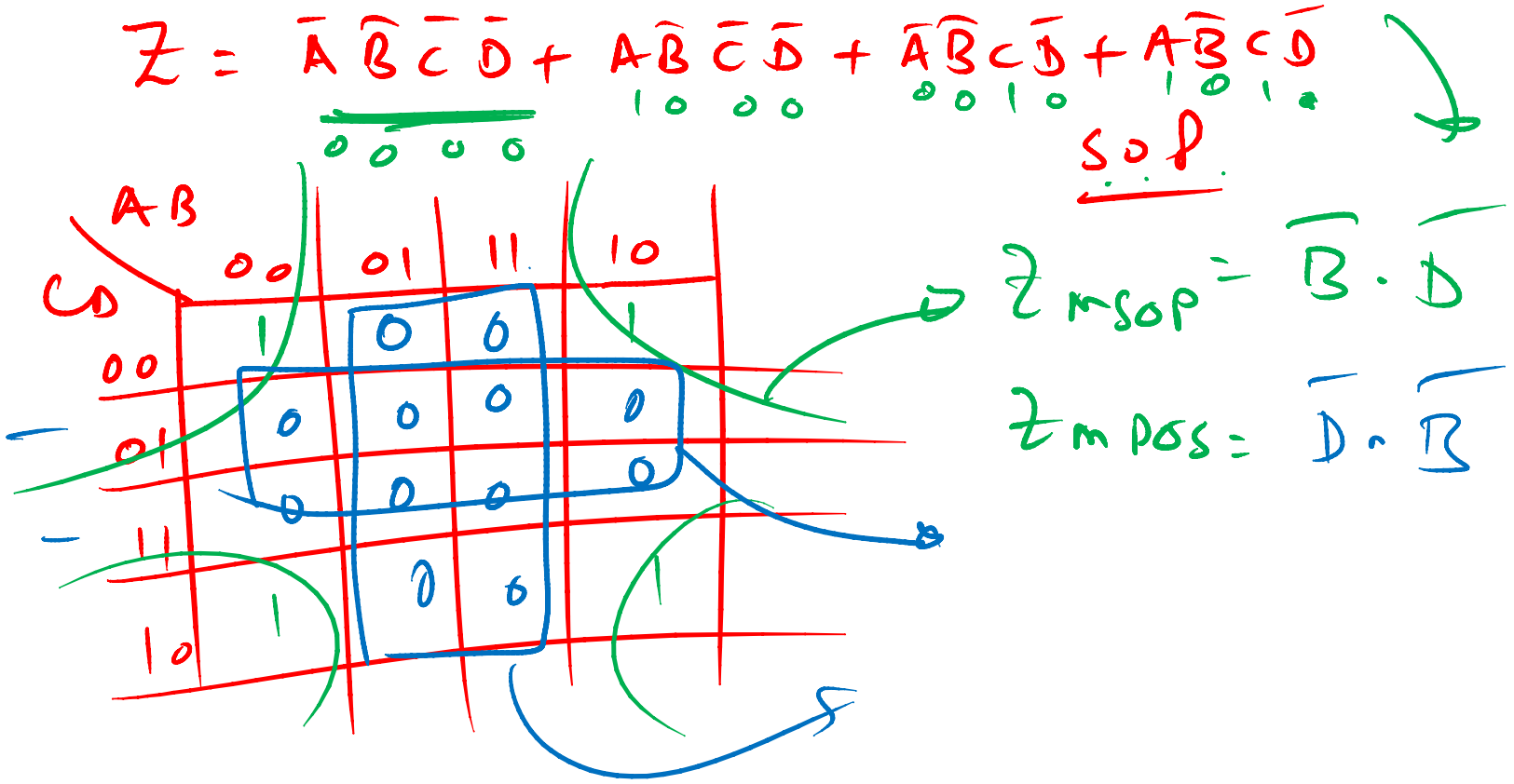
$$Z = \bar{N}_3 \bar{N}_1 + \bar{N}_3 \bar{N}_2 + N_3 N_2 N_1 + N_3 N_2 N_0$$



NOR

Tutorial Q3

- Minimization of logic expression using K-map

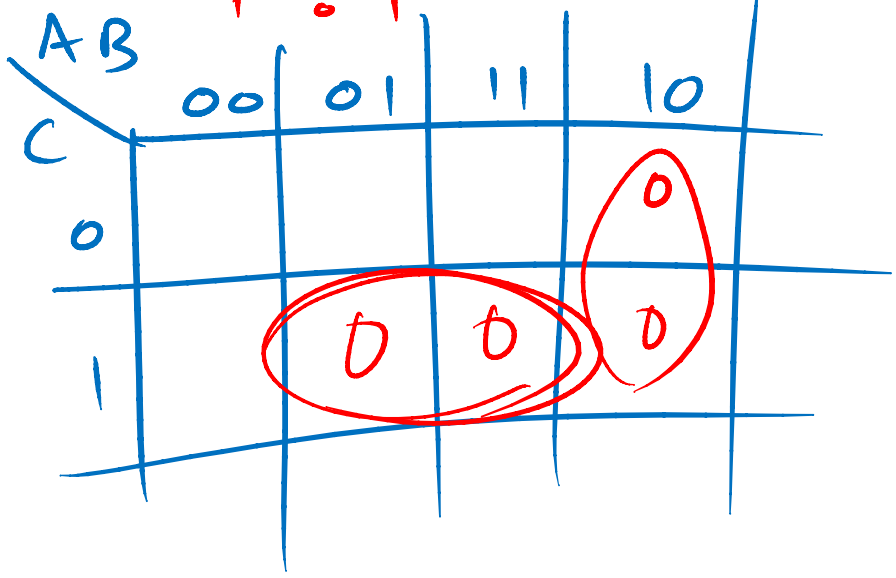


Tutorial Q4

- Minimization of logic expression using K-map

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

1 0 0 1 1 1 0 1 1



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