UNIVERSITI TEKNOLOGI MALAYSIA

www.utm.my

SCSR1013 DIGITAL LOGIC

MODULE 5:
COMBINATIONAL LOGIC CIRCUITS

FACULTY OF COMPUTING

Objectives:

1. To introduce AND-OR and AND-OR-Invert Logic
2. To illustrate the conversion between different representation of digital circuit
3. To introduce the universality of gate
4. To introduce a dual symbol in a digital circuit
5. To explain the design a combinational circuit

Basic Combinational Gates

Basic Combinational Logic

- AND-OR Logic
- AND-OR-Invert Logic
- Exclusive-OR (XOR) Logic
- Exclusive-NOR (XNOR) Logic

- In designing a circuit, we can make the output to become 0 (low) or 1 (high)

1

- If we want the output to be active high, then the final circuit will have a SOP expression. This kind of circuit can easily be implemented by **AND-OR Logic**

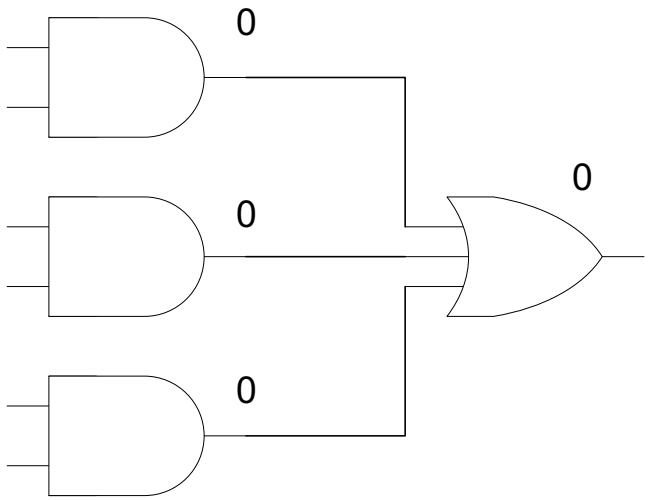
0

- To get an active low output, the output of the circuit with an active high output (AND-OR Logic) has to be inverted to logic 0 by adding the inverter, therefore the circuit becomes **AND-OR-Invert Logic.**

AND-OR Logic

- A circuit consisting of any number of AND gates and an OR gate.
- Example: SOP expressions (e.g. AB + CD)

- If **any** of the AND gates output are HIGH, the output in the OR gate is HIGH
- If **all** of the AND gates outputs are LOW, the output in the OR gate is LOW



Example: AND-OR Logic

Based on the truth table:

- derive the expression
- simplify using K-Map
- Implement using AND-OR logic

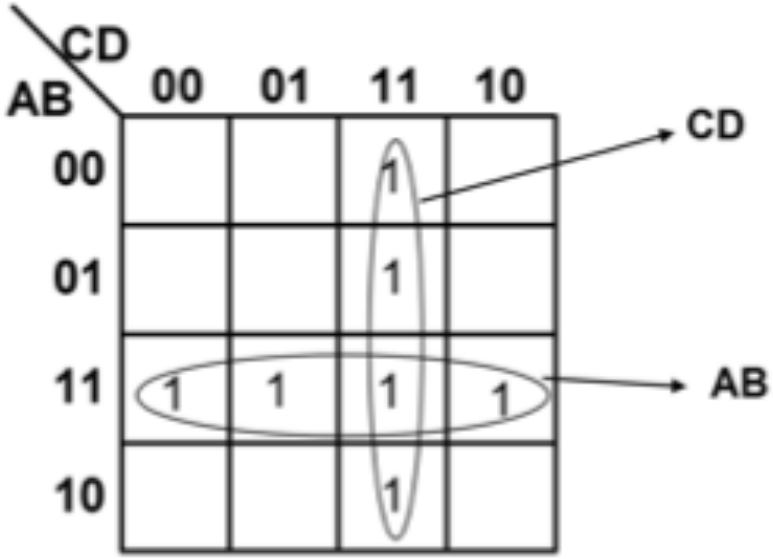
INPUTS						OUTPUT
A	B	C	D	AB	CD	X
0	0	0	0	0	0	0
0	0	0	1	0	0	0
0	0	1	0	0	0	0
0	0	1	1	0	1	1
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	0	1	1
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	0	1	1
1	1	0	0	1	0	1
1	1	0	1	1	0	1
1	1	1	0	1	0	1
1	1	1	1	1	1	1

Solution:

- Write the Boolean expression or map the 1's to the K-Map

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

- Simplify the output expression by using Boolean algebra or K -Map



D	AB	CD	OUTPUT X
0	0	0	0
1	0	0	0
0	0	1	0
0	0	1	1	1	0	1	1
0	1	0	0	0	0	0
0	1	0	1	0	0	0
0	1	1	0	0	0	0
0	1	1	1	0	1	1
1	0	0	0	0	0	0
1	0	0	1	0	0	0
1	0	1	0	0	0	0
1	0	1	1	0	1	1
1	1	0	0	1	0	1
1	1	0	1	1	0	1
1	1	1	0	1	0	1
1	1	1	1	1	1	1

Extra

Self-Test:

Simplify the expression using Boolean algebra.

Solution:

$$X = \overline{A}\overline{B}CD + \overline{A}BCD + A\overline{B}CD + AB\overline{C}\overline{D} + AB\overline{C}D + ABC\overline{D} + ABCD$$

(Associative Law) (Associative Law) (Associative Law)

$$X = (A'+A)B'CD + A'BCD + ABC'(D'+D) + ABC(D'+D)$$

(Rule 6) (Rule 6) (Rule 6)

$$X = B'CD + A'BCD + ABC' + ABC$$

(Associative Law)

$$X = B'CD + A'BCD + AB(C'+C)$$

(Rule 6)

$$X = B'CD + A'BCD + AB$$

(Associative Law)

continue...

$X = B'CD + (A'CD + A)B$

(Associative Law)

$X = B'CD + (A'CD + A)B$

(Rule 11)

$X = B'CD + (A + CD)B$

(Distributive Law)

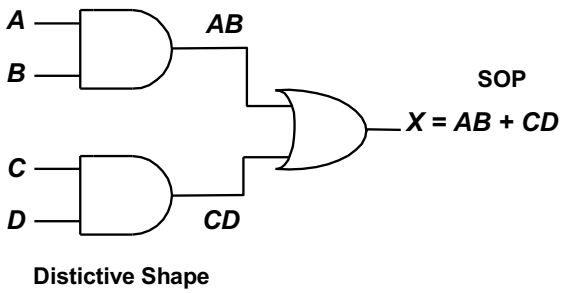
$X = B'CD + BCD + AB$

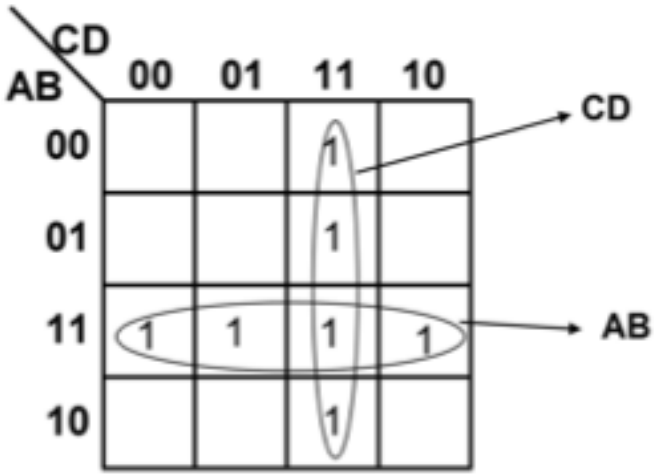
(Associative Law)

$X = (B'+B)CD + AB$

(Rule 6)

$X = CD + AB$



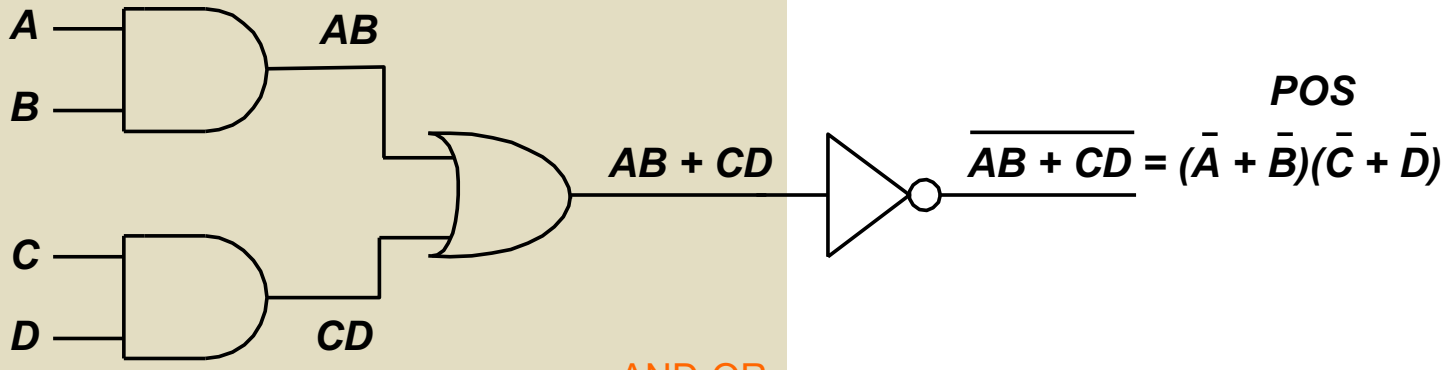


AND-OR-Invert Logic

- It is the complemented AND-OR circuit
- Example: POS expressions $(\bar{A} + \bar{B})(\bar{C} + \bar{D})$

$$\text{is actually } \overline{(\bar{A} + \bar{B})(\bar{C} + \bar{D})} = \overline{AB + CD}$$

- Results are the complement of AND-OR circuit.



$\overline{AB + CD}$

INPUTS							OUTPUT
A	B	C	D	AB	CD	AB + CD	X
0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	1
0	0	1	0	0	0	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	0	1
0	1	1	0	0	0	0	1
0	1	1	1	0	1	1	0
1	0	0	0	0	0	0	1
1	0	0	1	0	0	0	1
1	0	1	0	0	0	0	1
1	0	1	1	0	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	0
1	1	1	0	1	0	1	0
1	1	1	1	1	1	1	0

Truth table:
AND-OR-Invert

Complement of AB+CD

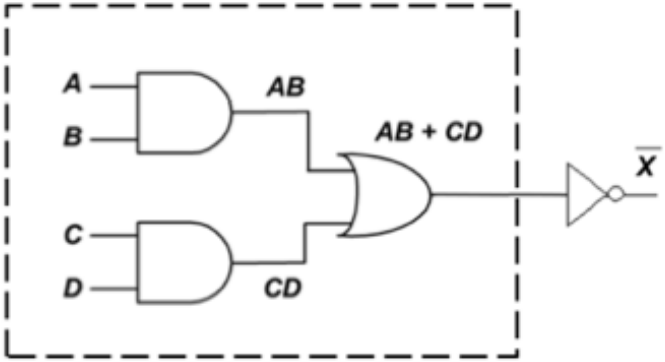
Exercise 5.1: Refer to previous example (slide 12), solve the problem for an active low output using AND-OR-Invert logic.

Solution: To implement an active low output, take the complement of X.

INPUTS A	B	C	D	AB	CD	OUTPUT X	Output X'
0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	1
0	0	1	0	0	0	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	0	0	1
0	1	0	1	0	0	0	1
0	1	1	0	0	0	0	1
0	1	1	1	0	1	1	0
1	0	0	0	0	0	0	1
1	0	0	1	0	0	0	1
1	0	1	0	0	0	0	1
1	0	1	1	0	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	0
1	1	1	0	1	0	1	0
1	1	1	1	1	1	1	0

Based on the truth table:

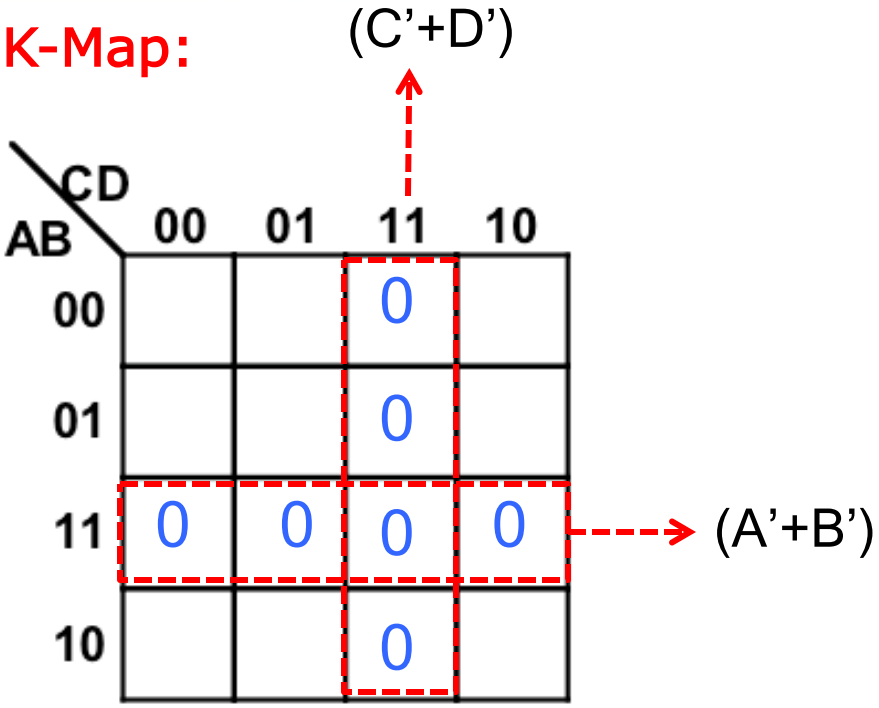
- derive the expression
- simplify using K-Map
- Implement using AND-OR-Invert logic.



- If we want active low, put an inverter at the output of the SOP circuit, it will produce a POS for $\overline{X}$ (*note: not POS for X*)

continue...

K-Map:



Truth table:

INPUTS A	B	C	D	AB	CD	Output X'
0	0	0	0	0	0	1
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	0	0	1
0	1	1	0	0	0	1
0	1	1	1	0	1	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	0	0	1
1	0	1	1	0	1	0
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	1	1	0

Expression (Truth table):

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

Expression (K-Map):

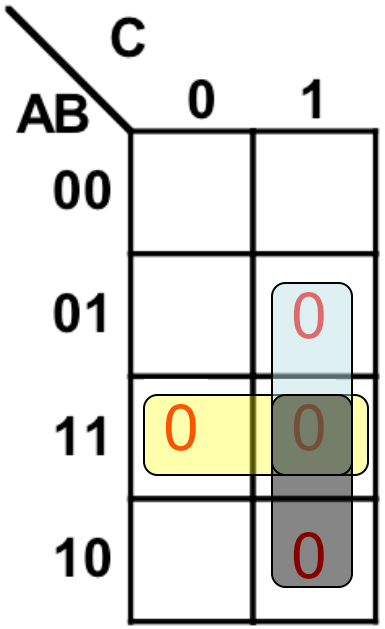
$$X = (A'+B')(C'+D')$$

Self-Test:

Based on the truth table below, implement the active low output by applying the AND-OR-Invert logic.

sensor inputs

A	B	C	Output	Output X'
0	0	0	0	1
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	0



Expression (K-Map):

$X' = (A'+B')(B'+C')(A'+C')$

The Universal Property of NAND and NOR Gates

The Universal Property of NAND and NOR Gates

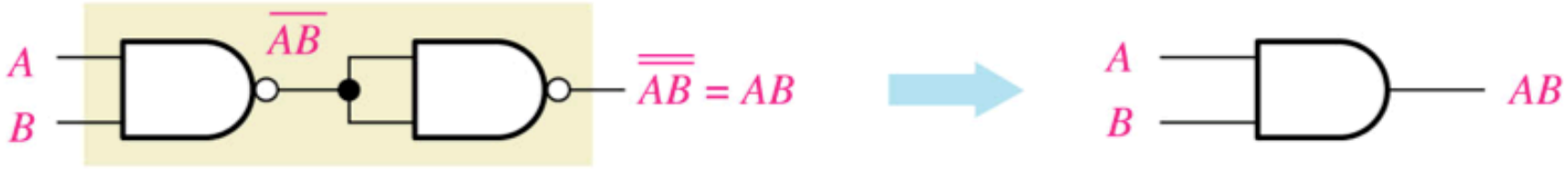
- If a gate can be converted and perform the function of a basic gates, then it can be used to implement any logic circuits. Therefore they are called universal gate.
- There are 2 common universal gates:
  - The NAND Gate
  - The NOR Gate

- It's a good thing to implement the final circuit using only a single type of gate. The advantages are
  - Have to stock only one type of gates
  - Buy only one type of gates in a volume so that, it can reduce cost
  - Can reuse the extra of unused gates to implement other gates

- This can be done by using a universal gate -- NAND or NOR
  - Usually SOP is implemented using NAND only
  - and POS implemented as NOR only

The NAND Gate as Universal Logic Element

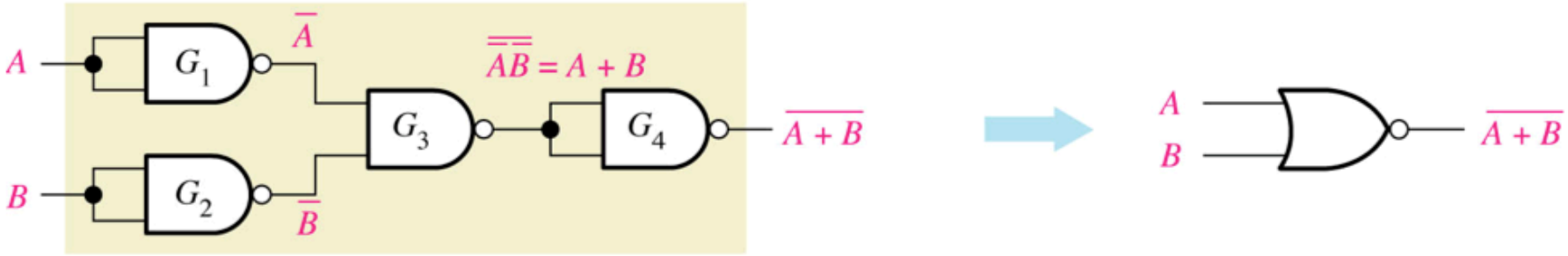
(a) A NAND gate used as an inverter



(b) Two NAND gates used as an AND gate

continue...

(c) Three NAND gates used as an OR gate



(d) Four NAND gates used as a NOR gate

- To prove that a gate is universal – it must be able to perform as an AND, OR and Inverter (NOT) gate.
- To prove that NAND can function as an AND, OR and Inverter (NOT) gate
- (a) NAND to NOT:

  - tie both input NAND ➔ $\overline{A.A} = \overline{A}$

- (b) NAND to AND:

  - add an inverter at the output of the NAND, it will cancel out the effect of the bubble ➔ $\overline{\overline{AB}} = AB$

- (c) NAND to OR:

  1. from OR put the bubble at all input and output, change the input bubble to an inverter by using NAND gate
  2. change the OR symbol to AND ➔ $\overline{\overline{A}\,\overline{B}} = A + B$

- (d) NAND to NOR:

  1. from OR put the bubble at all input and output, change the input bubble to an inverter by using NAND gate
  2. change the OR symbol to AND ➔ $\overline{\overline{A}\,\overline{B}} = A + B$
  3. tie both input ➔ $\overline{A + B}$

(Module Page:155)

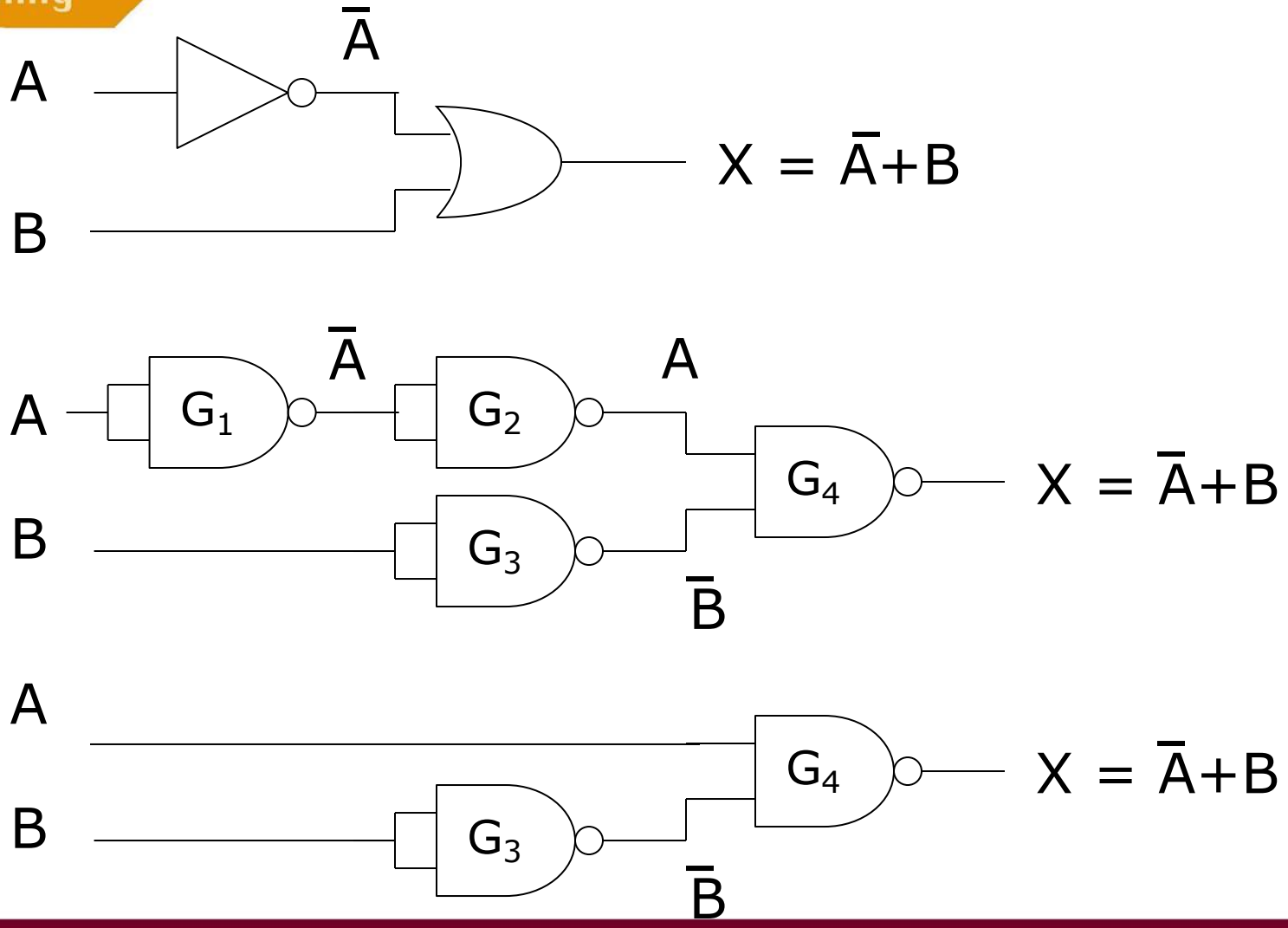
Exercise 5.2:
Use NAND gates to implement each expression.

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

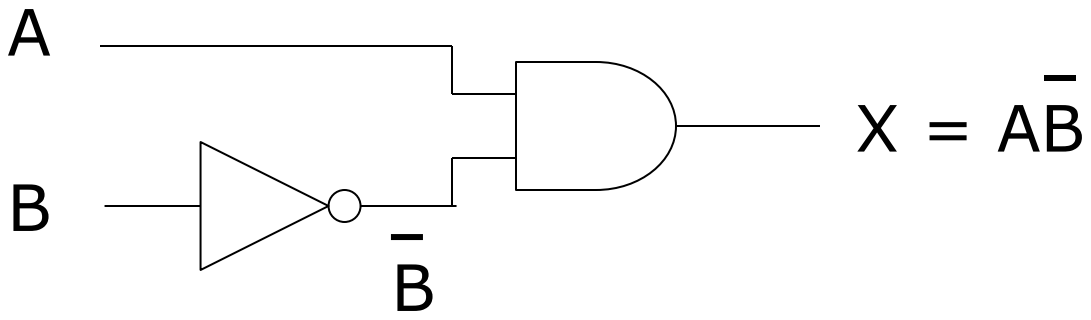
b. $X = A\overline{B}$

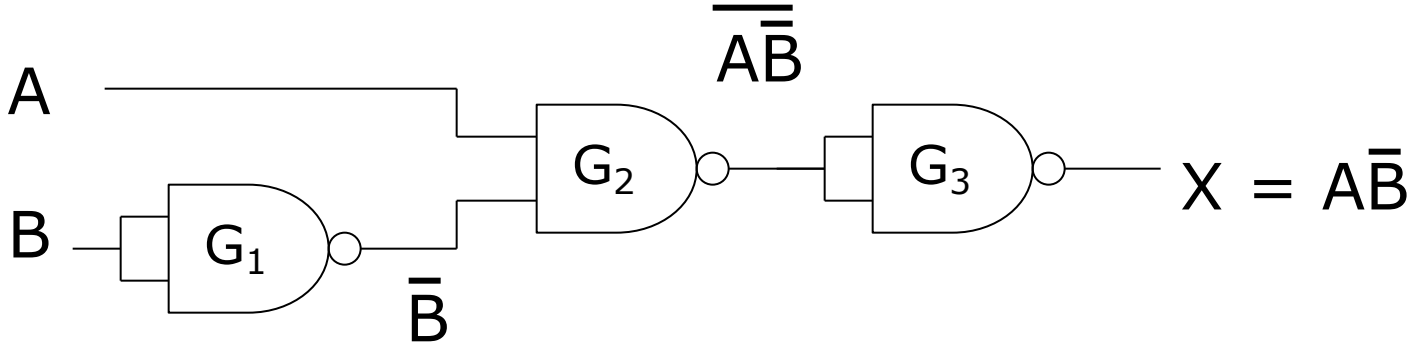
Solution:

4(a)


A
B
$\bar{A}$
X = $\bar{A}$+B
A
B
$\bar{A}$
A
G1
G2
G3
G4
$\bar{B}$
X = $\bar{A}$+B
A
B
G3
G4
$\bar{B}$
X = $\bar{A}$+B

4(b)



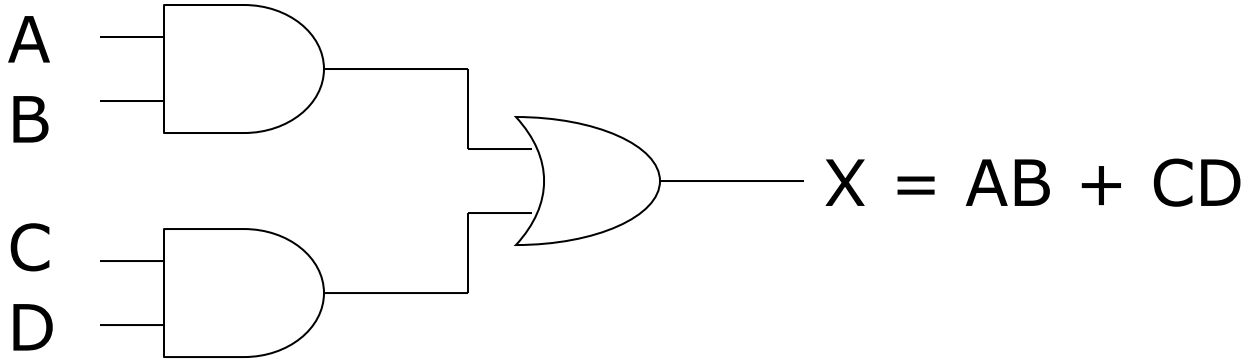


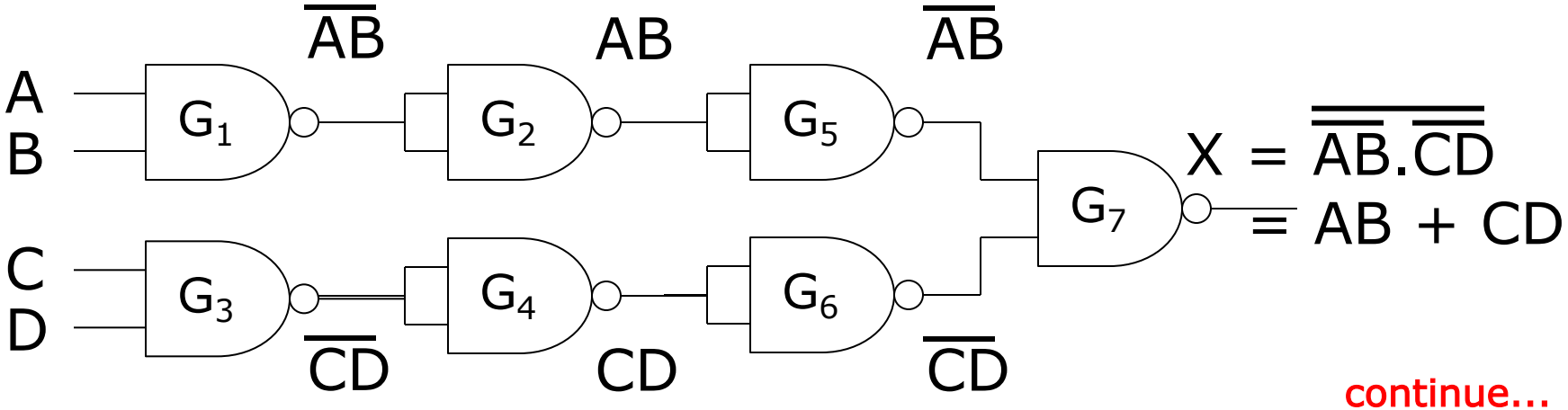
Extra

Exercise 5.3: Use NAND gates to implement this expression.

$X = AB + CD$

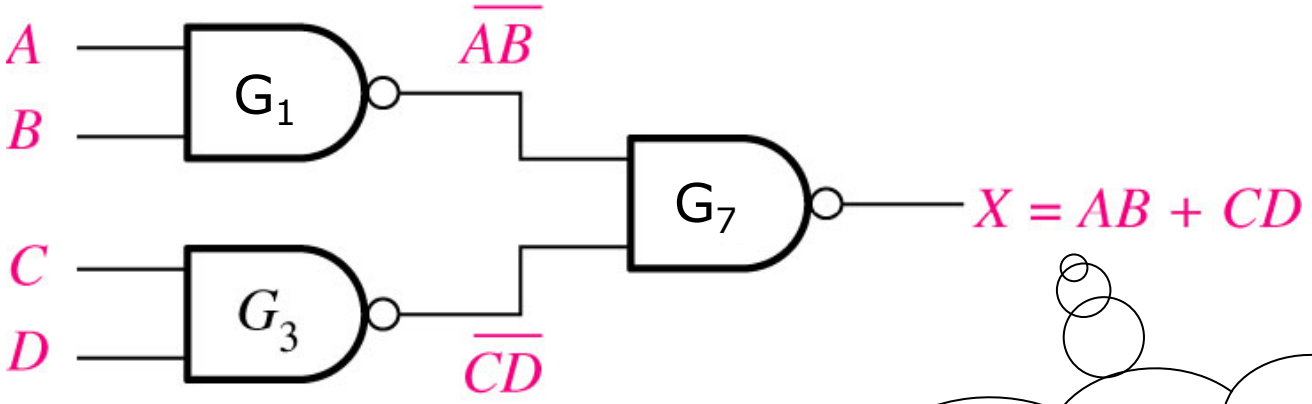
Solution:





continue...

Can discard G_2, G_4, G_5 and G_6



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

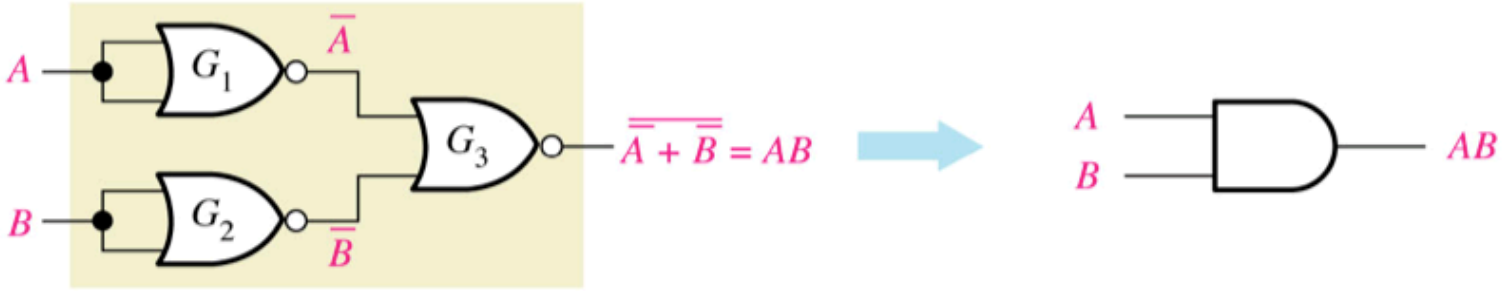
$$= AB + CD$$

The NOR Gate as Universal Logic Element

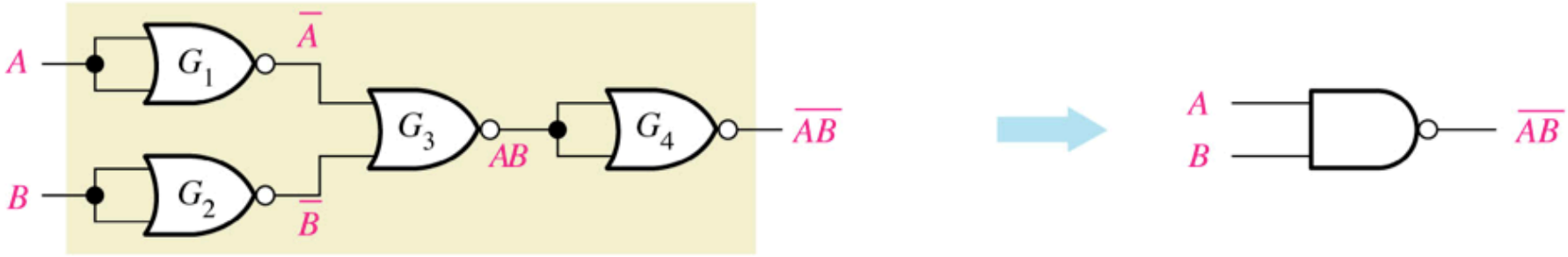
(a) A NOR gate used as an inverter

(b) Two NOR gates used as an OR gate

continue...



(c) Three NOR gates used as an AND gate



(d) Four NOR gates used as a NAND gate

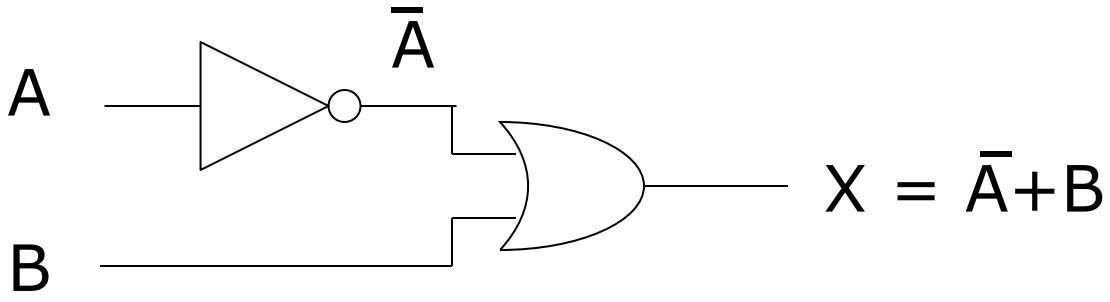
Exercise 5.4:
Use NOR gates to implement each expression.

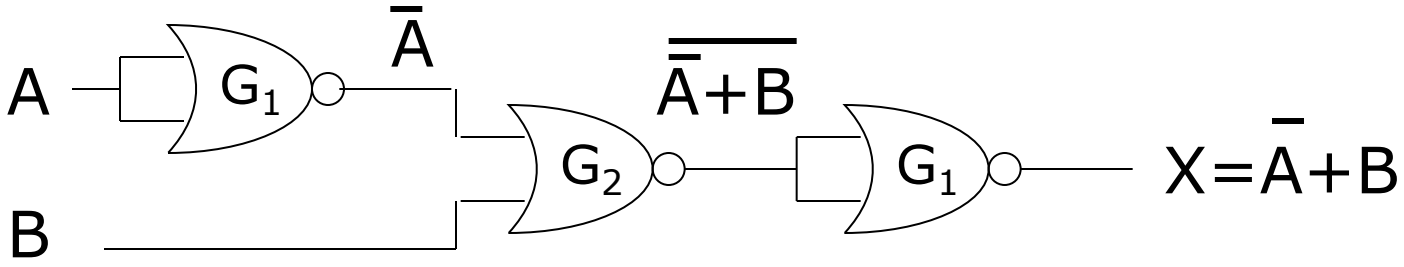
a. $X = \overline{A} + B$

b. $X = A\overline{B}$

Solution:

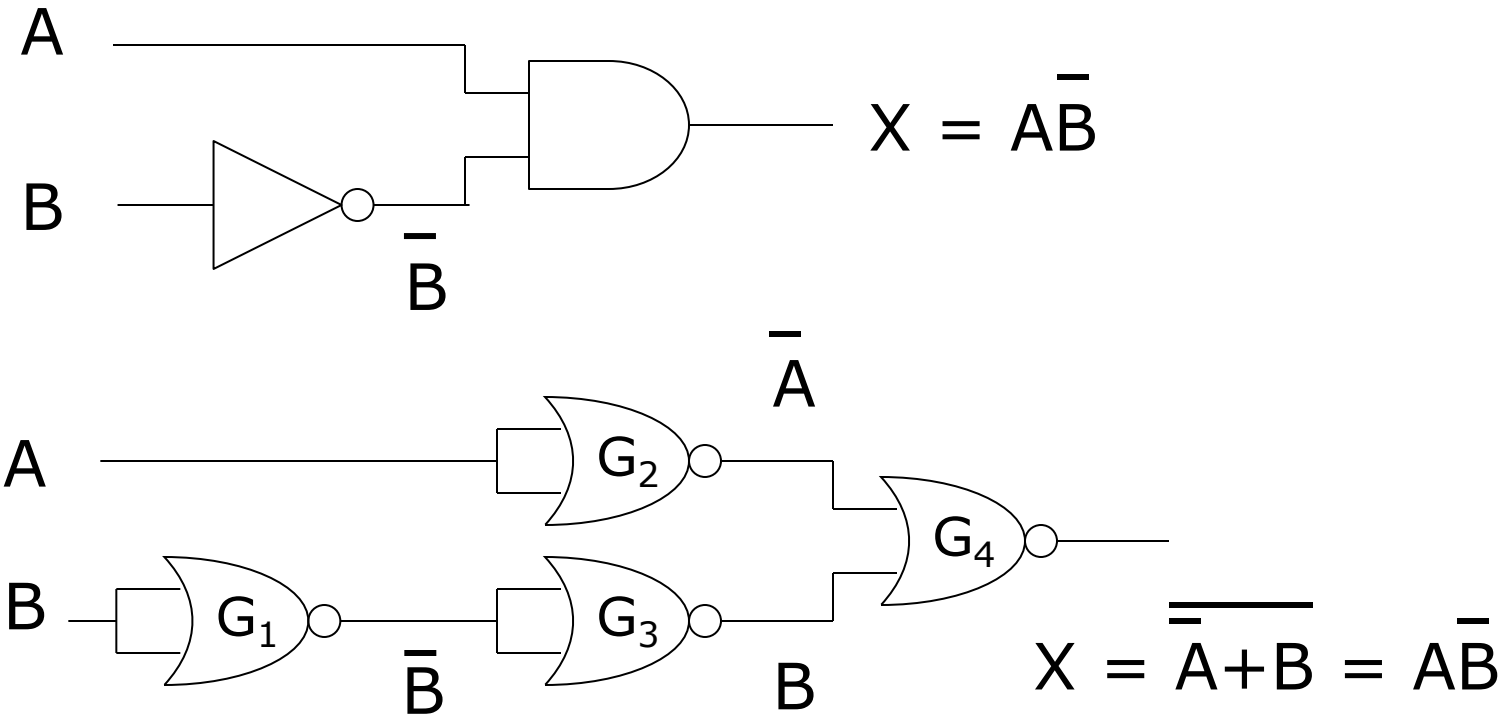
6(a)





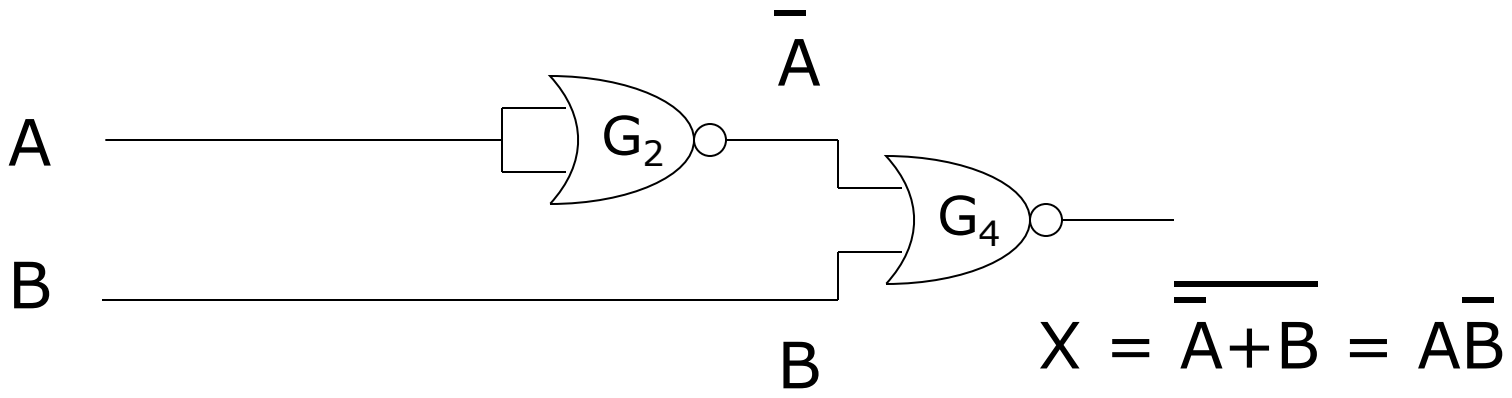
continue...

6(b)


A
B
$\bar{B}$
$X = A\bar{B}$
$\bar{A}$
A
G_2
G_4
B
G_1
G_3
$\bar{B}$
B
$X = \overline{\bar{A}+B} = A\bar{B}$

continue...

Can discard G_1, and G_3



Dual Symbol

Dual Symbol

NAND

Alternate symbol for NAND – Negative OR

NOR

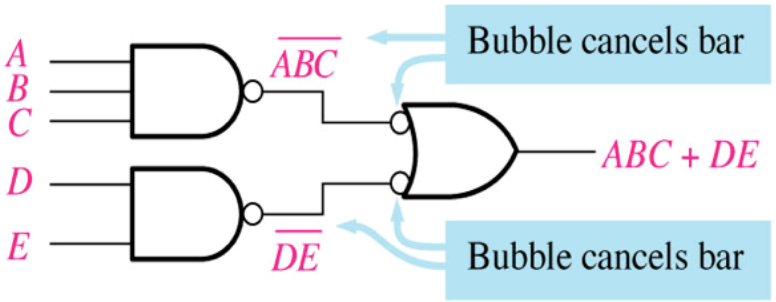
Alternate symbol for NOR – Negative AND

Converting to an alternate symbol

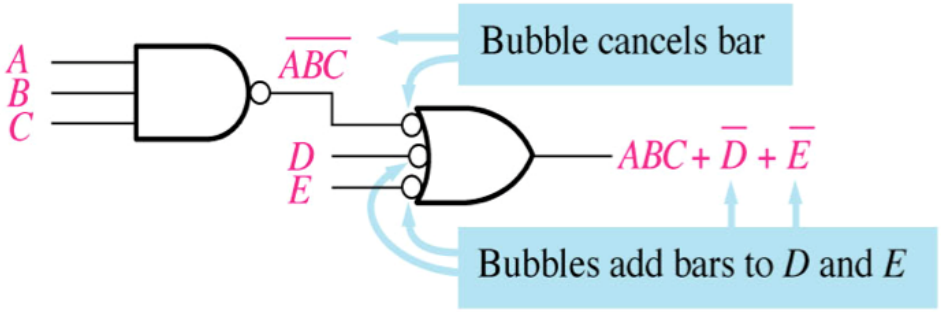
Step1 : Pull bubble at all input and output
Step 2 : Change the symbol AND to OR, OR to AND
Step 3 : Redraw the symbol, if a bubble connected to another bubble it will cancel out the bubble

Careful:

$$DE \neq \overline{D} + \overline{E}$$

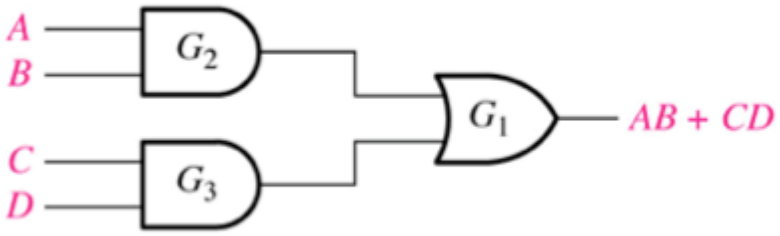


(a)



(b)

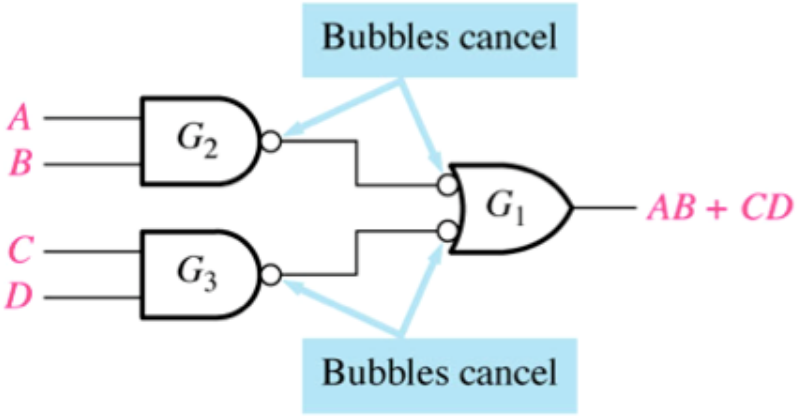
Converting SOP Circuit to NAND only

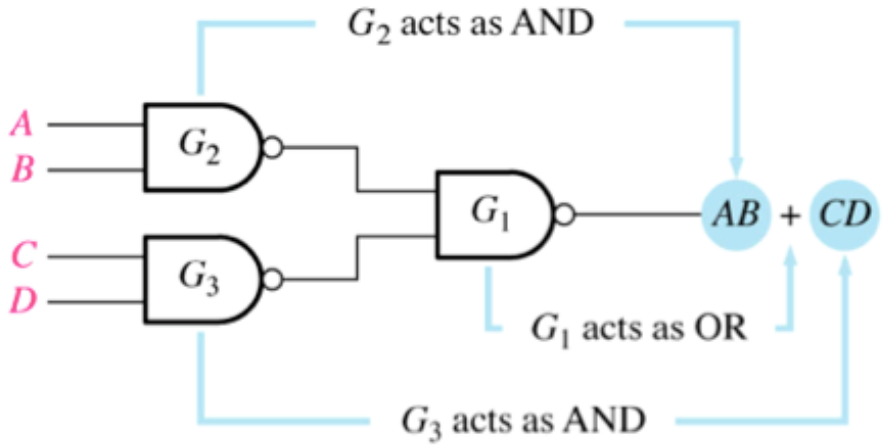


Step 1: Put the bubble at the input of right most OR.

Step 2: To compensate the effect of putting that bubble, add another bubble at the output of the AND gate.

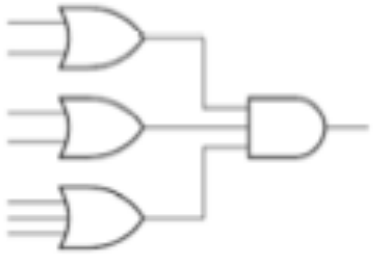
Step 3: Replace the negative-OR symbol with NAND symbol.





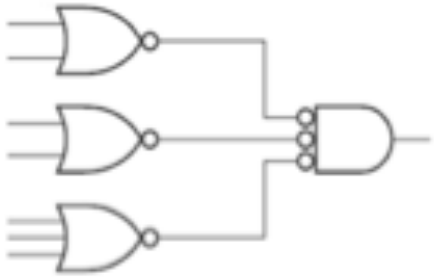
Converting POS circuit to NOR only

Original POS circuit

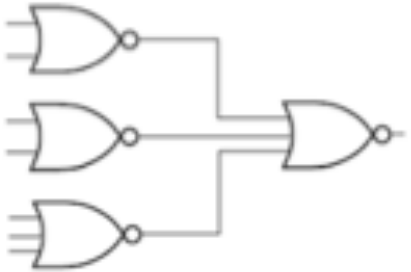


Step1: Put the bubble at the input of the right most AND

Step2: to compensate the effect of putting that bubble, add another bubble at the output of the OR gate

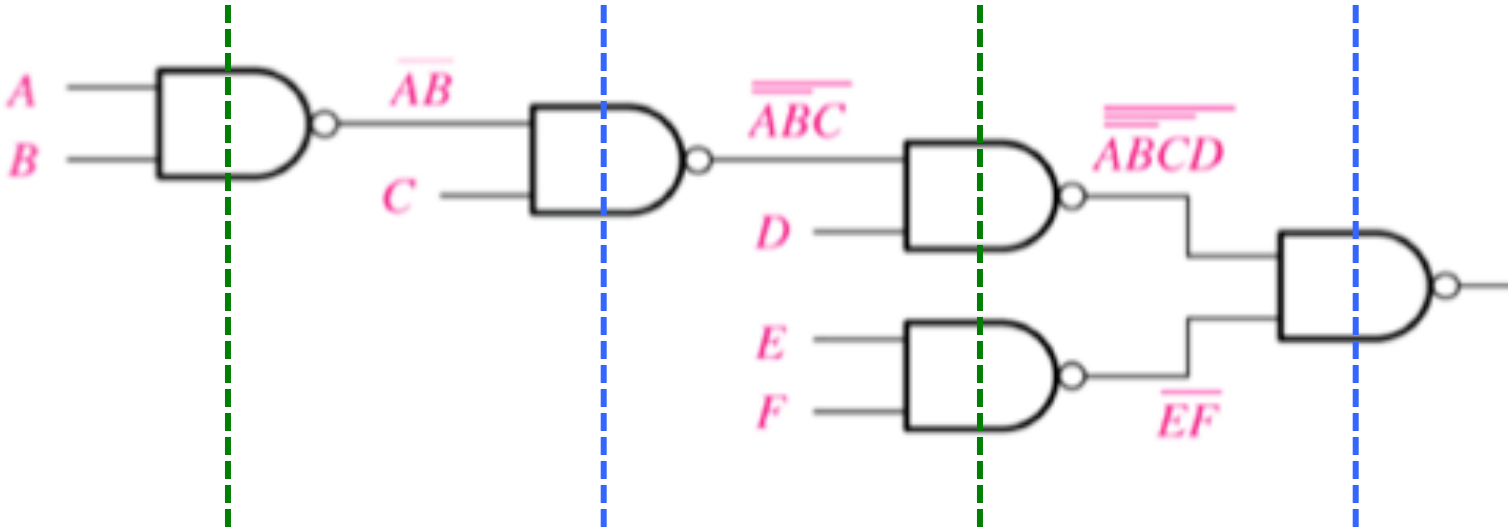


Step 3: Replace the negative OR symbol with a NOR symbol (no need, just to show NOR with the same symbol)

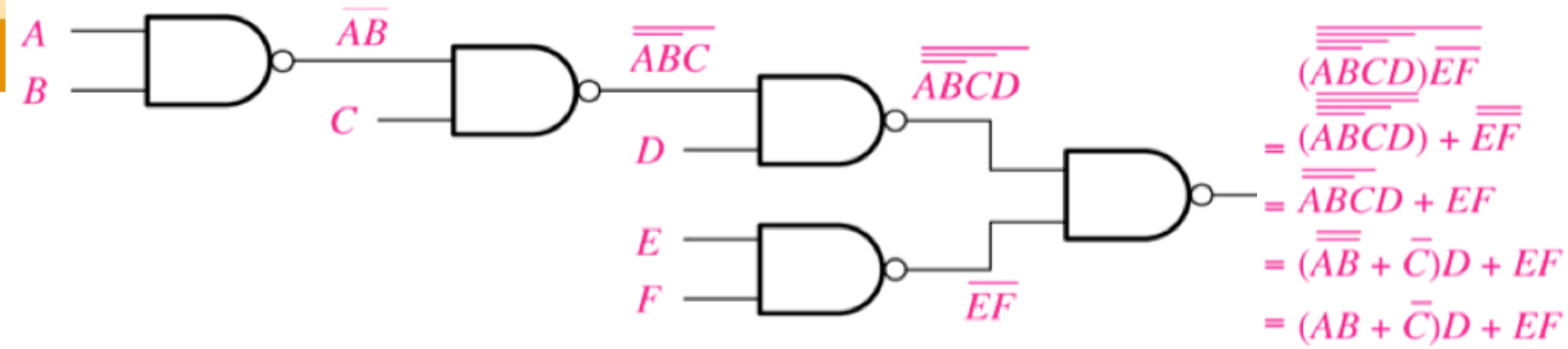


Dual Symbols: NAND Only (1)

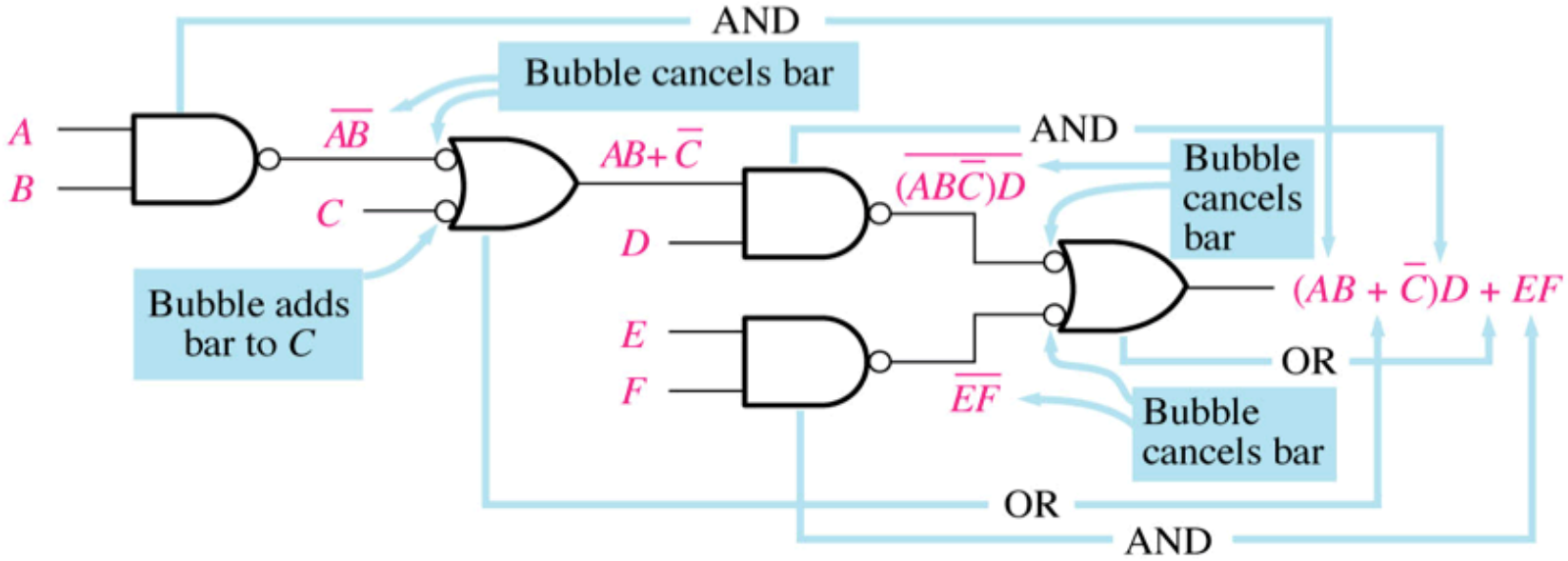
- Dual symbol is needed to simplify the "reading" of the schematic
  - can determine how the circuits works – very fast
- Replace the NAND gate with its dual symbol which is Negative-OR
- Choose which level that you want to replace the alternate symbol - odd or even
  - If you choose odd – only replace the odd level with an alternate symbol



Dual Symbols: NAND Only (2)

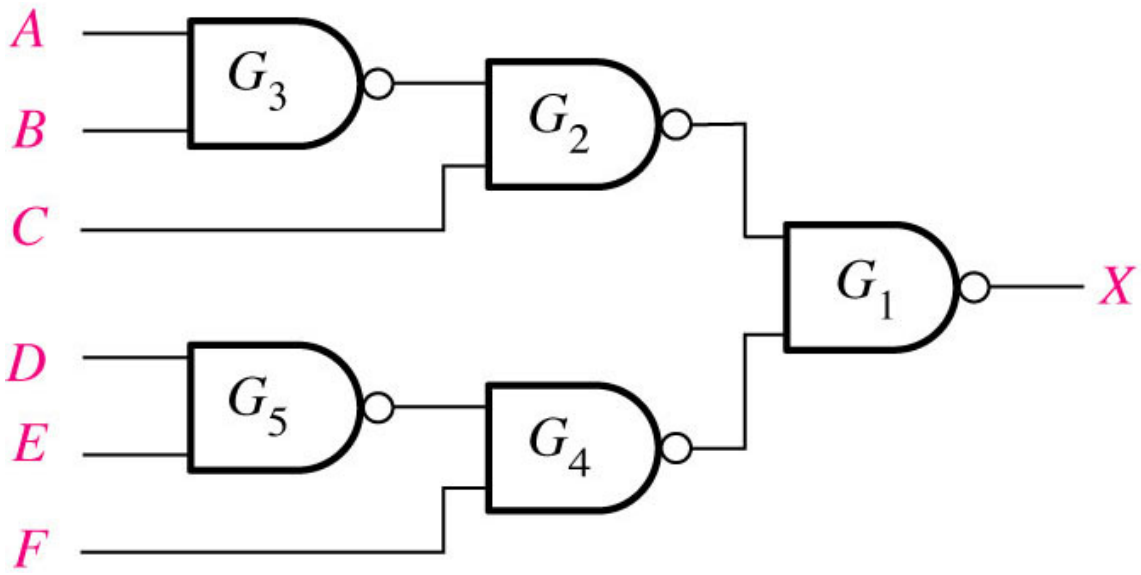


(a) Several Boolean steps are required to arrive at final output expression.

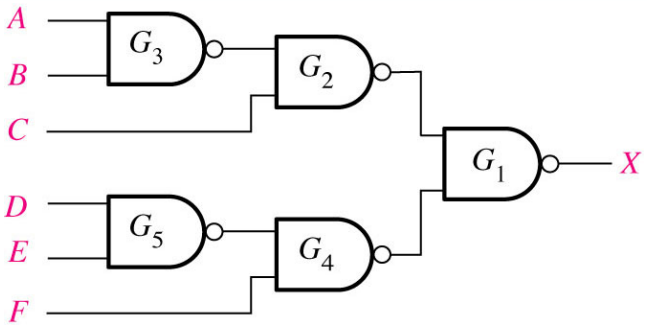


(b) Output expression can be obtained directly from the function of each gate symbol in the diagram

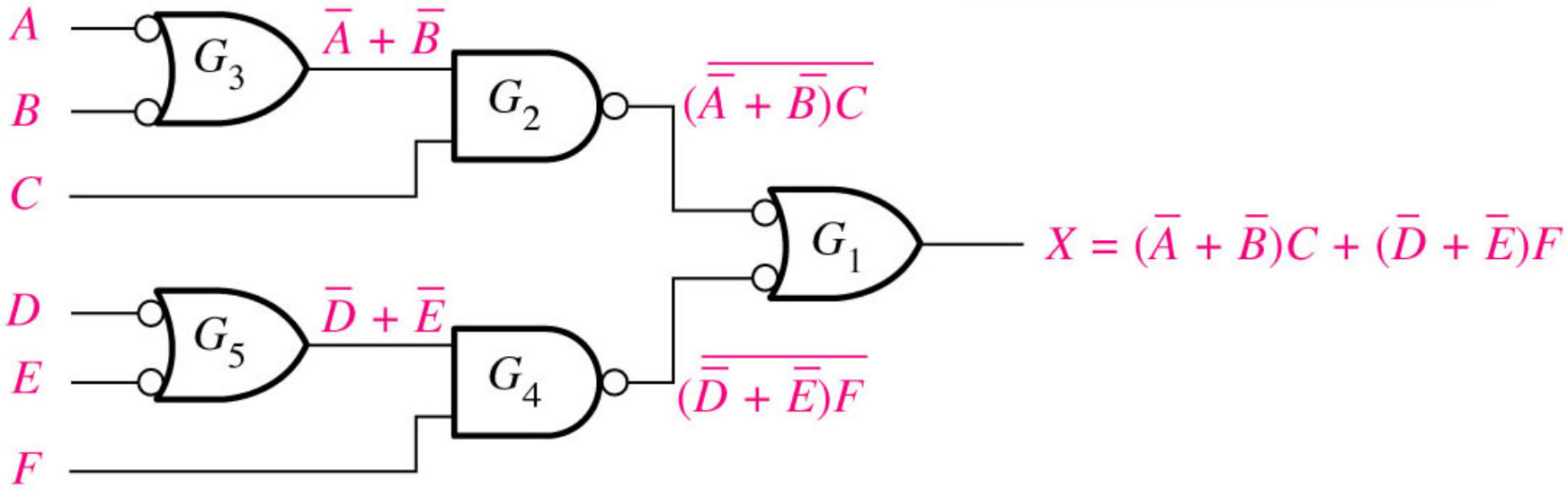
Example: Redraw the logic diagram and develop the output expression for the circuit using the appropriate dual symbols (odd level).



continue...



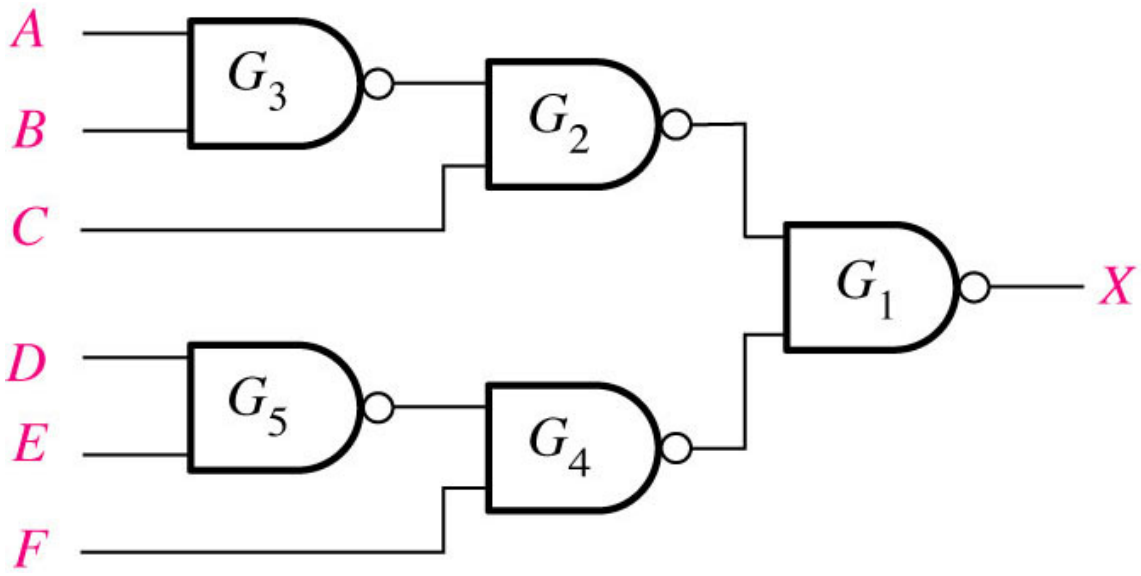
Solution:



(Odd level) (Even level) (Odd level)

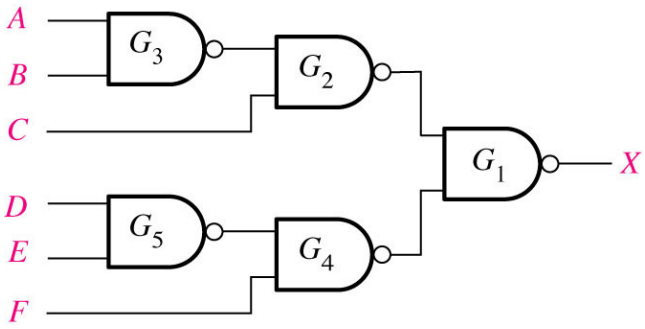
Extra

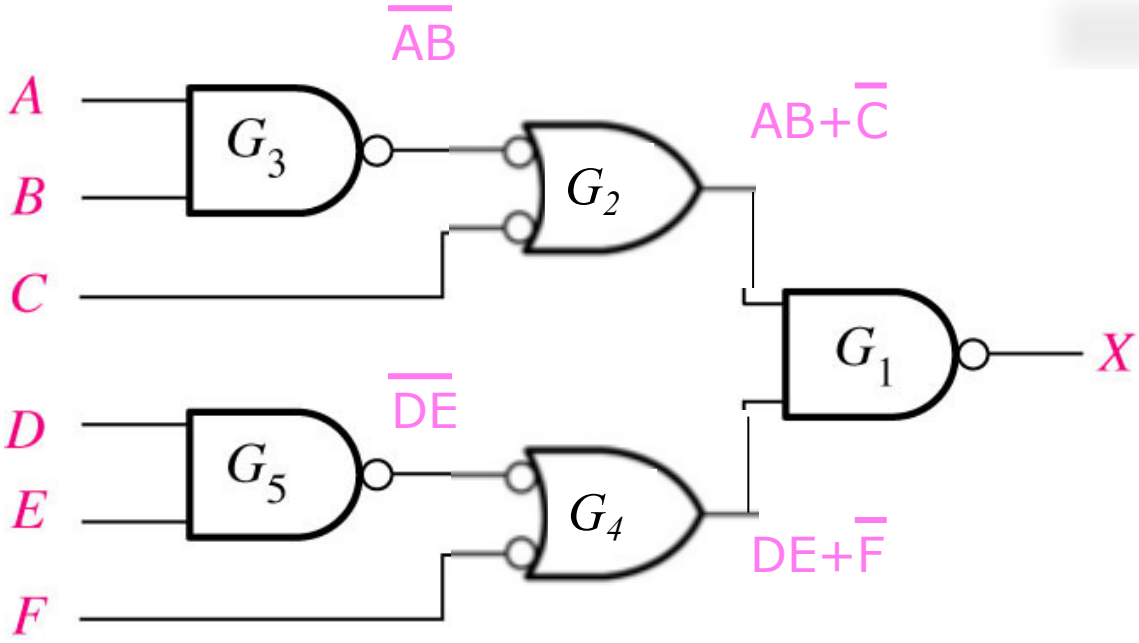
Exercise 5.5: Redraw the logic diagram and develop the output expression for the circuit using the appropriate dual symbols (even level).



continue...

Solution:



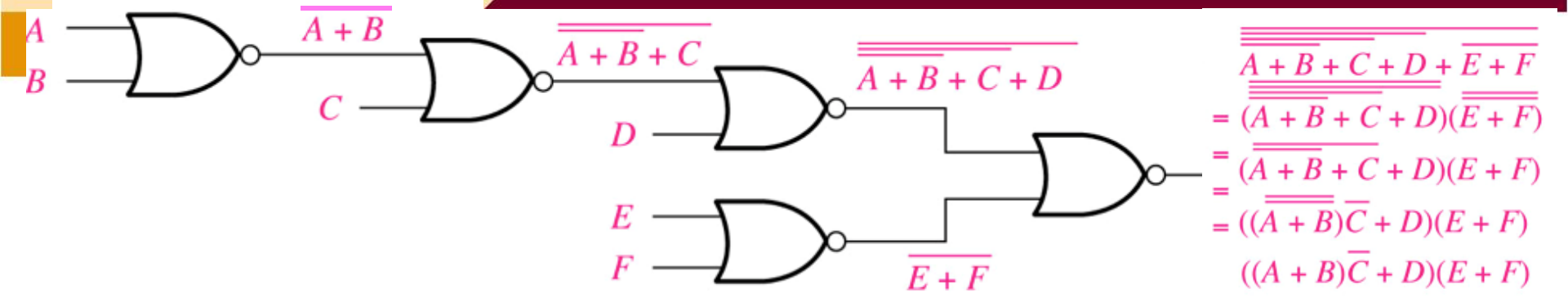


Expression X=?

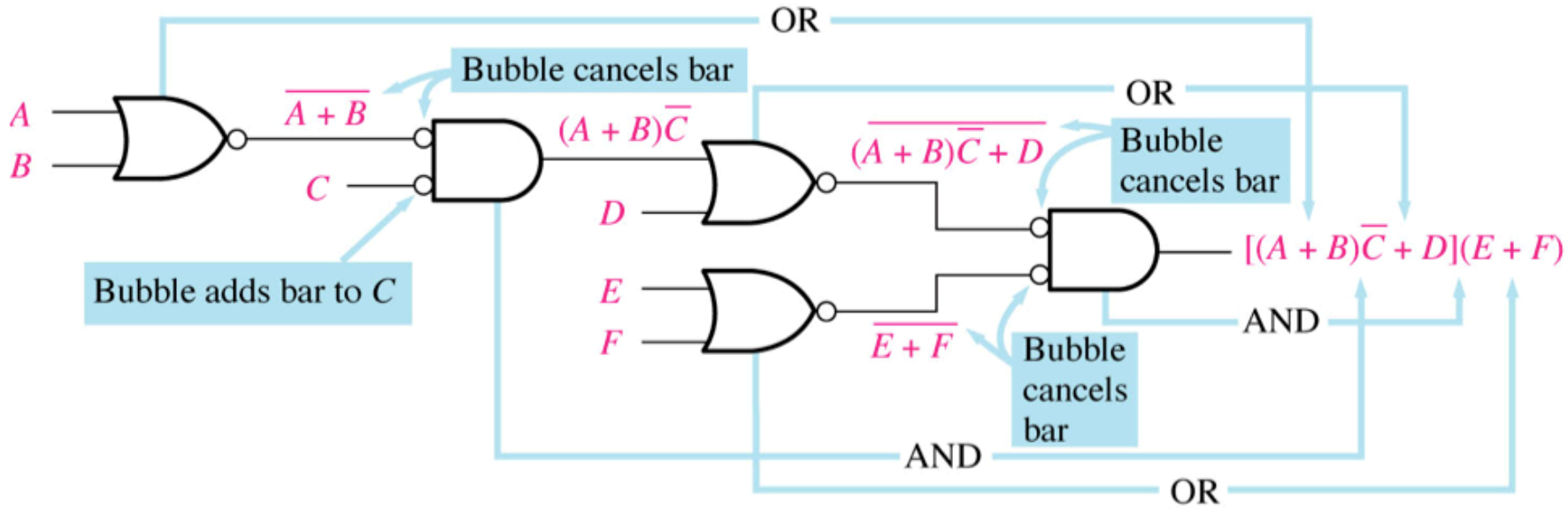
$$= \overline{(AB+\overline{C})(DE+\overline{F})}$$
$$= \overline{(AB+\overline{C})}+\overline{(DE+\overline{F})}$$
$$= (\overline{AB}.\overline{\overline{C}})+(\overline{DE}.\overline{\overline{F}})$$
$$= (\overline{A}+\overline{B})C + (\overline{D}+\overline{E})F$$

Dual Symbols: NOR Only

Example:



(a) Final output expression is obtained after several Boolean steps.



(b) Output expression can be obtained directly from the function of each gate symbol in the diagram.

Dual Symbols: NOR Only

Example:

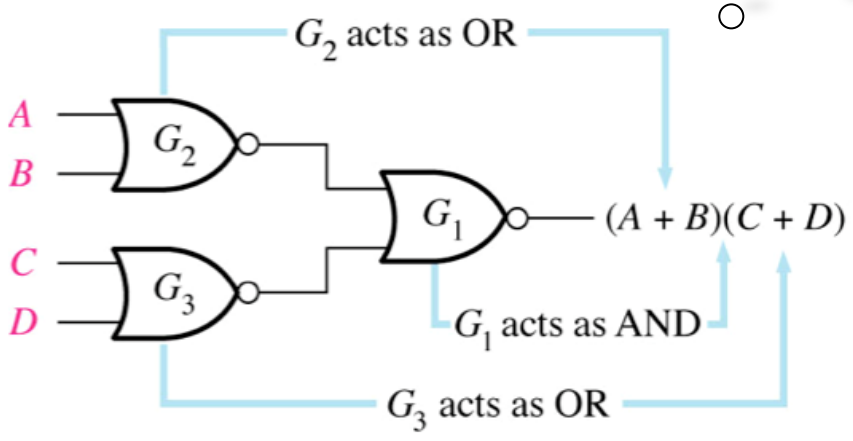
Implement the expression with NOR logic using appropriate dual symbol.

$(A+B)(C+D)$

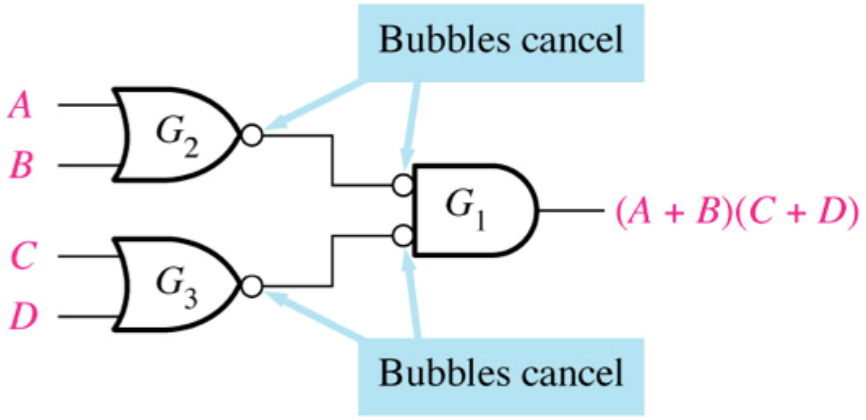
$$\overline{\overline{A+B}+\overline{C+D}} = \overline{\overline{A+B}}\,\overline{\overline{C+D}}$$

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

Solution:

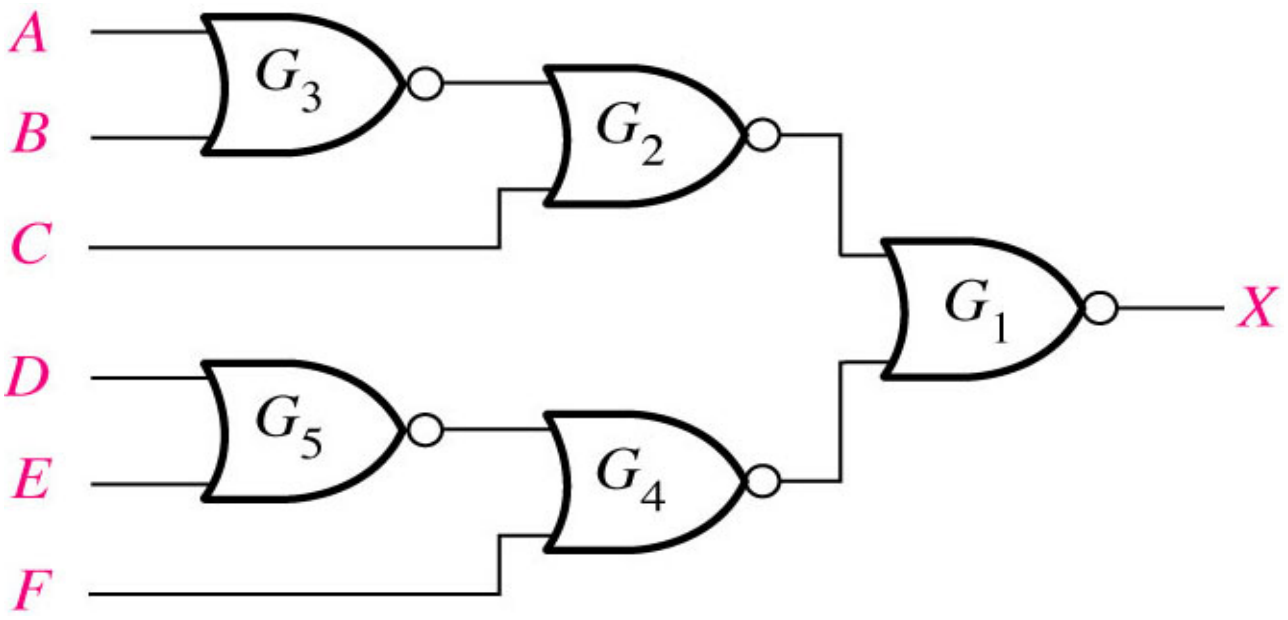


(a)



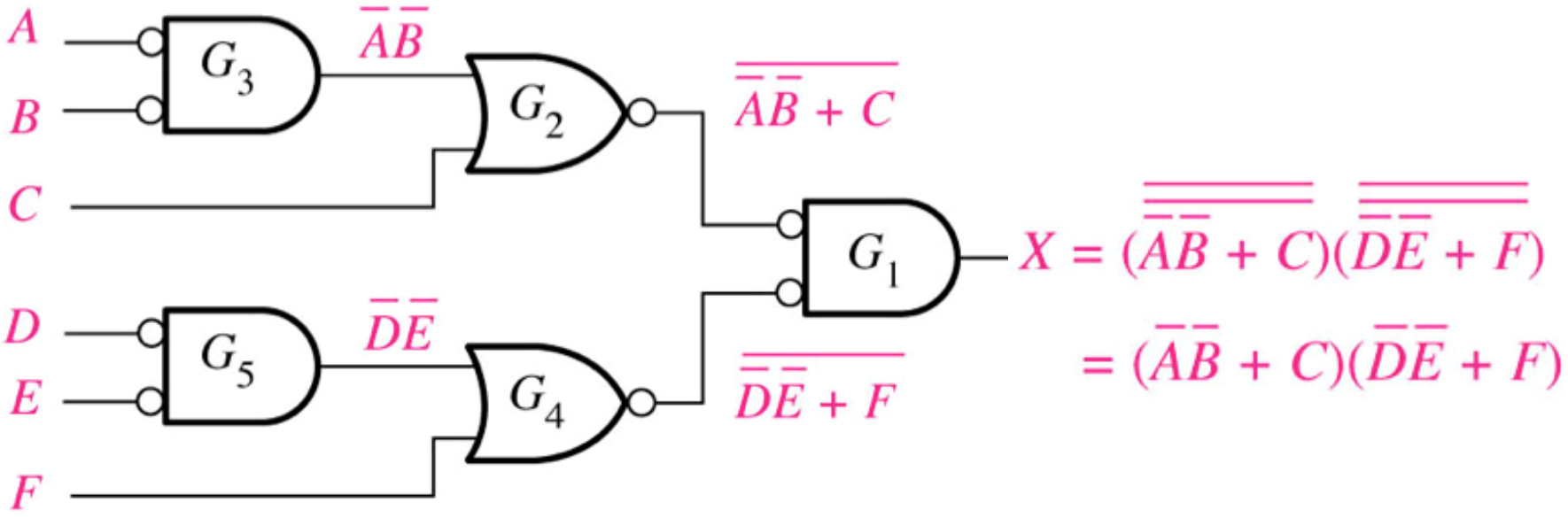
(b)

Example: Convert the NOR only circuit to an equivalent circuit using dual symbol. Prove the two circuits are equivalent by using Boolean algebra and DeMorgan's theorem. (Odd level)



continue...

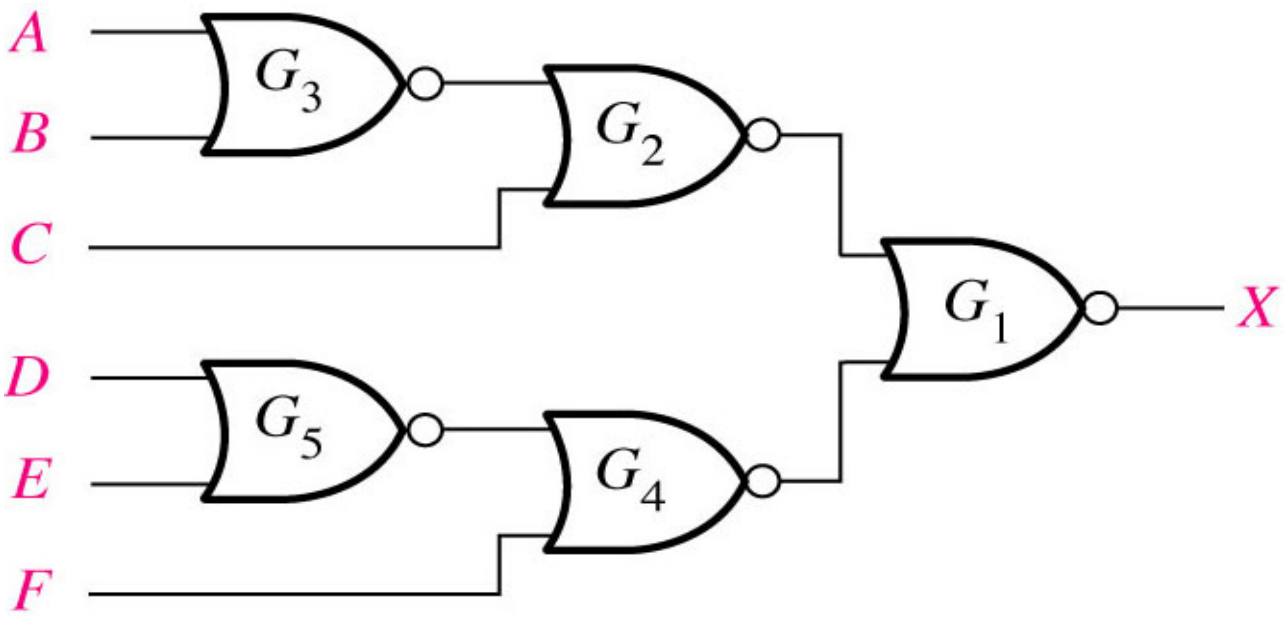
Solution:



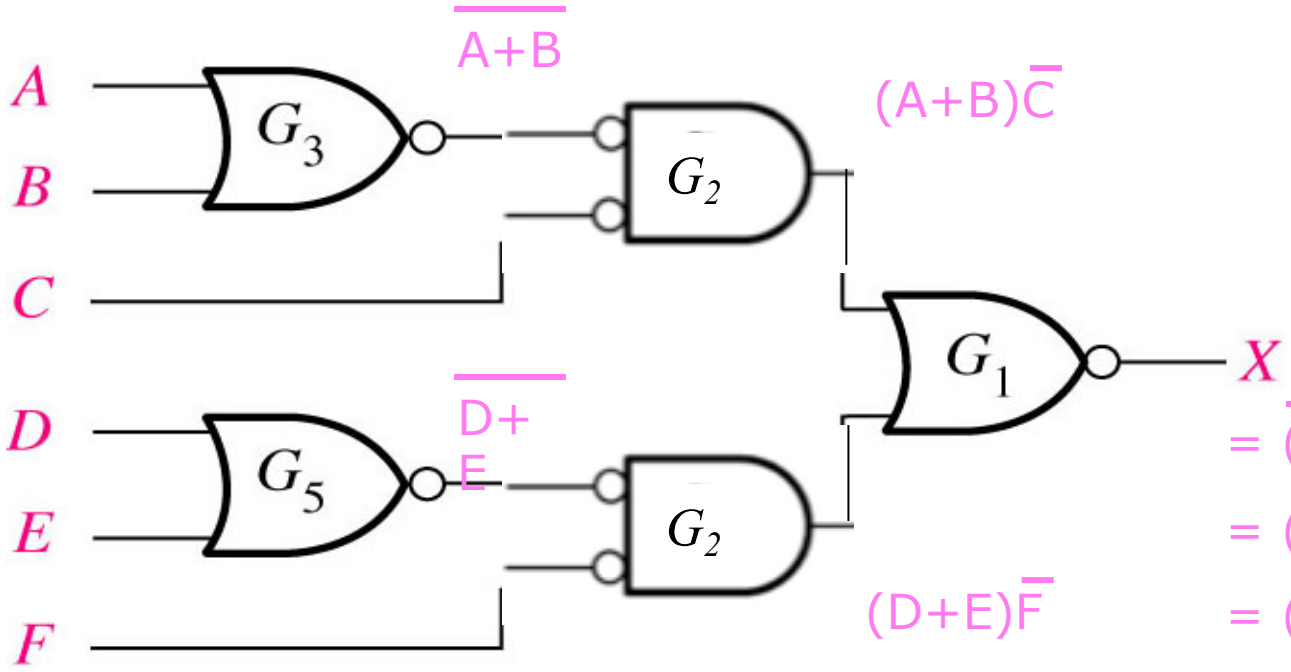
(Odd level) (Even level) (Odd level)

Extra

Exercise 5.6: Redraw the logic diagram and develop the output expression for the circuit using the appropriate dual symbols (even level).



Solution:



Expression X=?

$= \overline{(A+B)\overline{C} + (D+E)\overline{F}}$

$= \overline{((A+B)\overline{C})}\,\overline{((D+E)\overline{F})}$

$= ((\overline{A+B})+\overline{\overline{C}})((\overline{D+E})+\overline{\overline{F}})$

$= (\overline{A}\,\overline{B}+C)(\overline{D}\,\overline{E}+F)$

(Odd level) (Even level) (Odd level)

Exercise 5.7:

Question 1: Implement the expression using NAND logic and produce the answer using dual symbol.

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

Exercise 5.8:

Question 2: Implement the expression using NOR logic and produce the answer using dual symbol.

$$X = \overline{\overline{A}\overline{B}\overline{C} + (D + E)}$$

Problem Statements

Designing a Combinational circuit

Deriving a Boolean Expression

Steps in designing a combinational circuits

1. Understand the problem to be solved
2. Convert the problem statement to a Boolean expression or truth table, whichever is most natural for the given problem
3. Simplify the output by using Boolean algebra, K- Map or other technique
4. Draw the circuit from the simplified output
5. Convert the simplified circuit to universal gates only
   - SOP to NAND only
   - POS to NOR only
6. Redraw the final circuit using dual symbol

continue...

The most critical stage is step 1. Understand

Determine 3 things

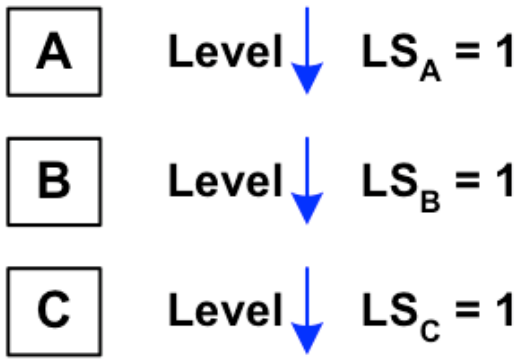
1. How many **inputs** and give it a variable name
2. How many **outputs** and give it a variable name
   - If more than one output treat each output separately, design a circuit for each output and finally combine them together
3. What is the **relationship** between output(s) and the inputs

Example:

- In a certain chemical-processing plant, a liquid chemical is used in a manufacturing process. The chemical is stored in three different tanks. A level sensor in each tank produces a HIGH voltage when the level of chemical in the tank drops below a specified point.

- Design a circuit that monitors the chemical level in each tank and indicates when the level in any two of the tanks drops below the specified point.

3 Tanks A, B, C



Solution:

Step1:

Input – 3 tanks – call it A,B,C - if drop below specified point it will input a HIGH (1) else LOW (0)

Output – 1 – call it X -- produces HIGH (1) if active

Relationship – any two tanks drop below specified point (has logic 1) the output active (HIGH)

continue...

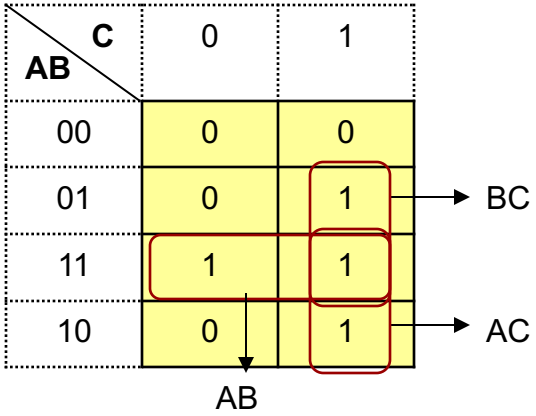
Step2:

Convert to standard form :

Truth table – the easiest in this case

Step3:

Map to K-Map then simplify



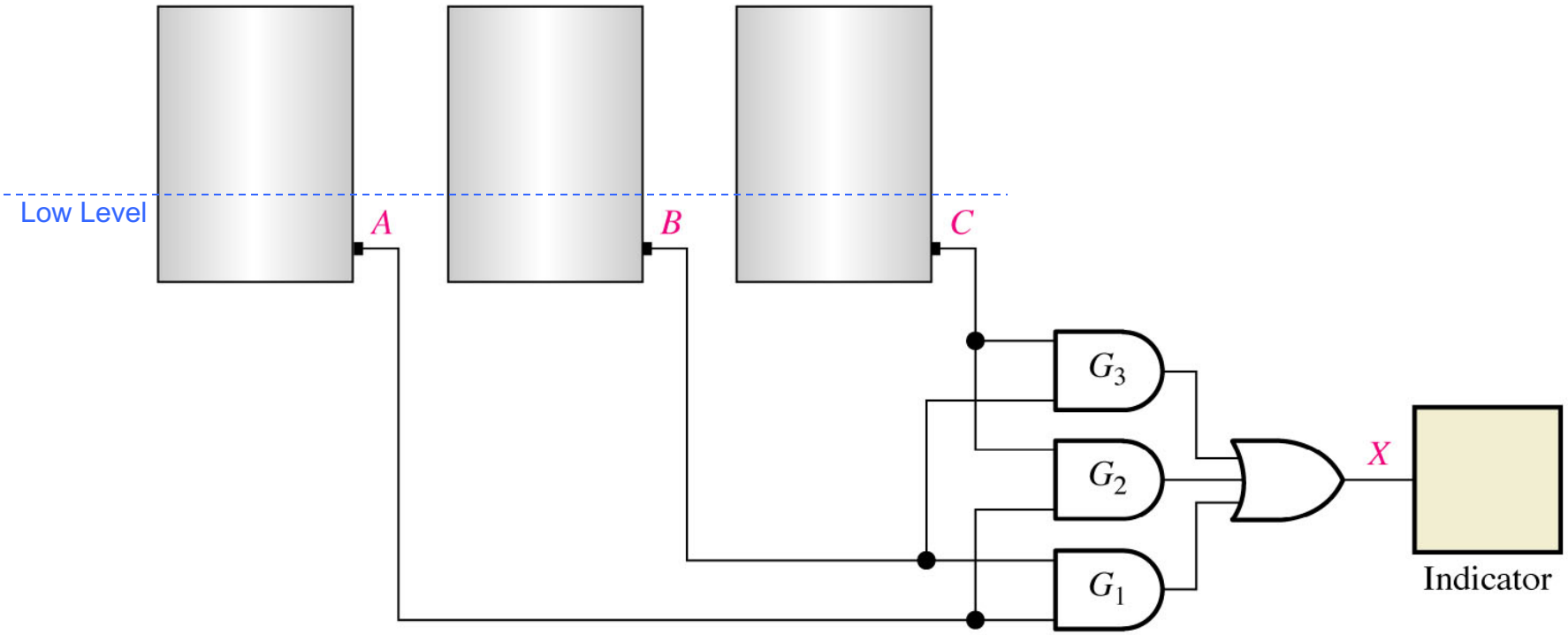
Truth table:

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

Simplified Equation: $X = AB + AC + BC$

continue...

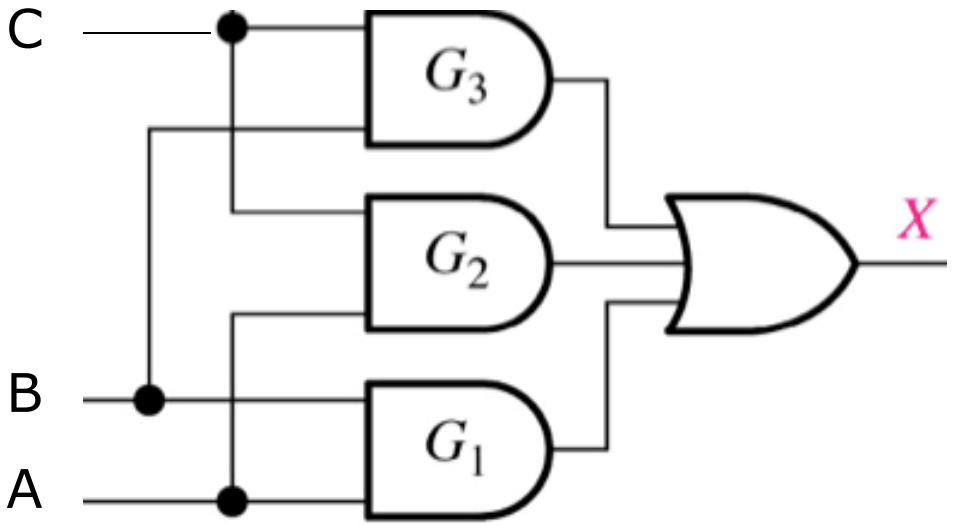
Step4: Draw the circuit for the simplified output $X = AB + AC + BC$



Exercise 5.9: (From previous example).

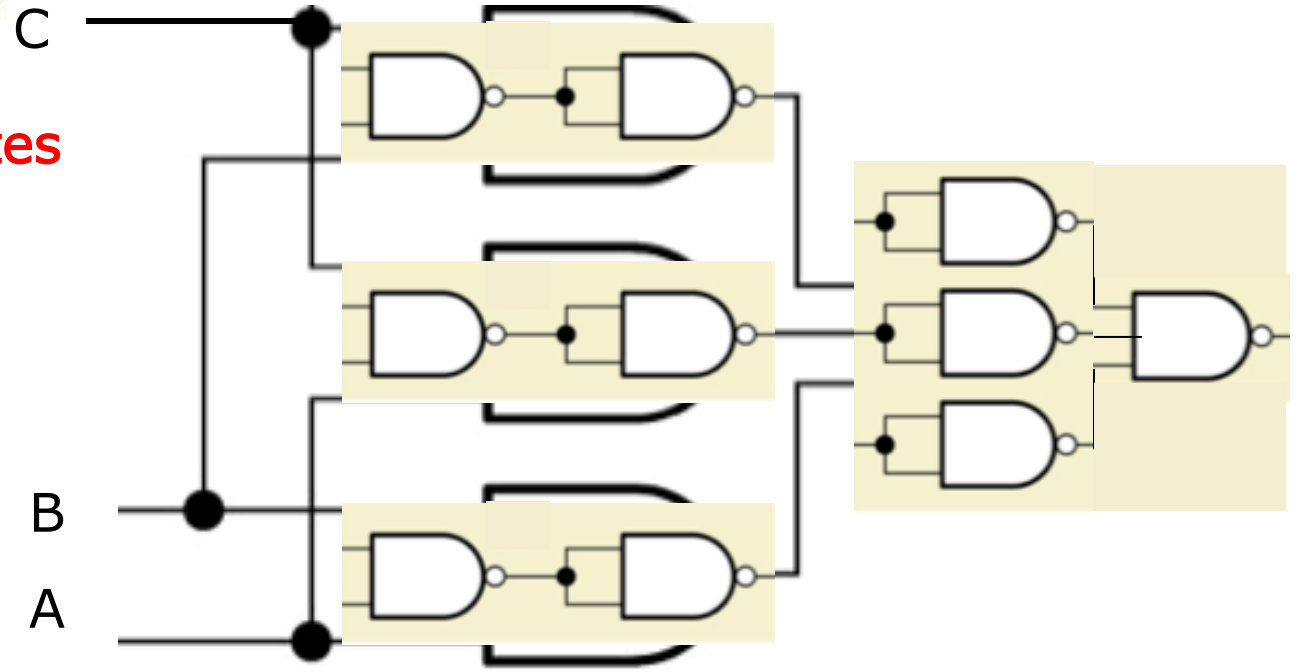
(a) Implement the circuit using universal gates

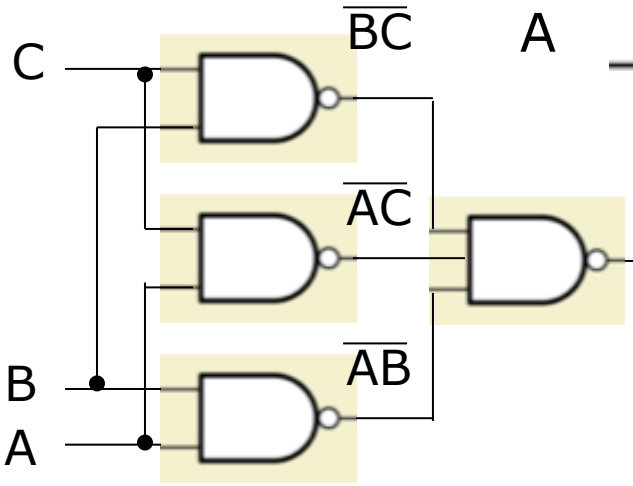
(b) Draw the final circuit using dual symbol



Solution:

5.9(a) NAND gates





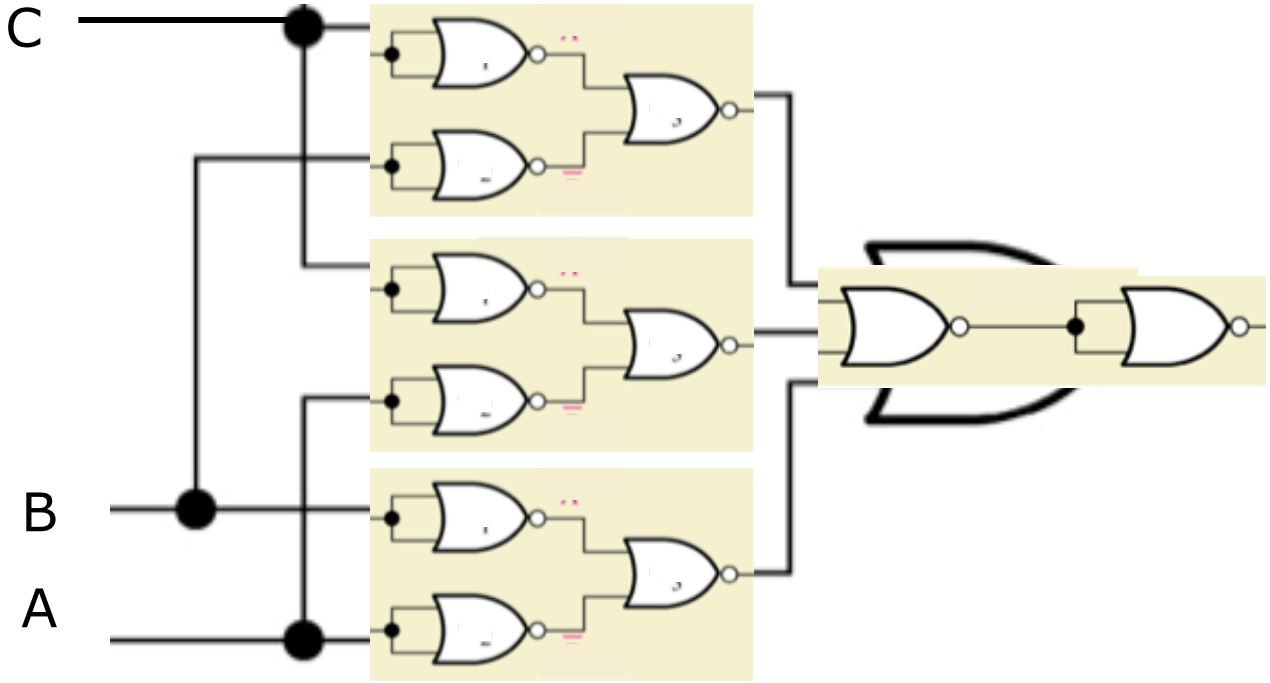
$$X = \overline{\overline{BC}.\overline{AC}.\overline{AB}}$$

$$= \overline{BC} + \overline{AC} + \overline{AB}$$

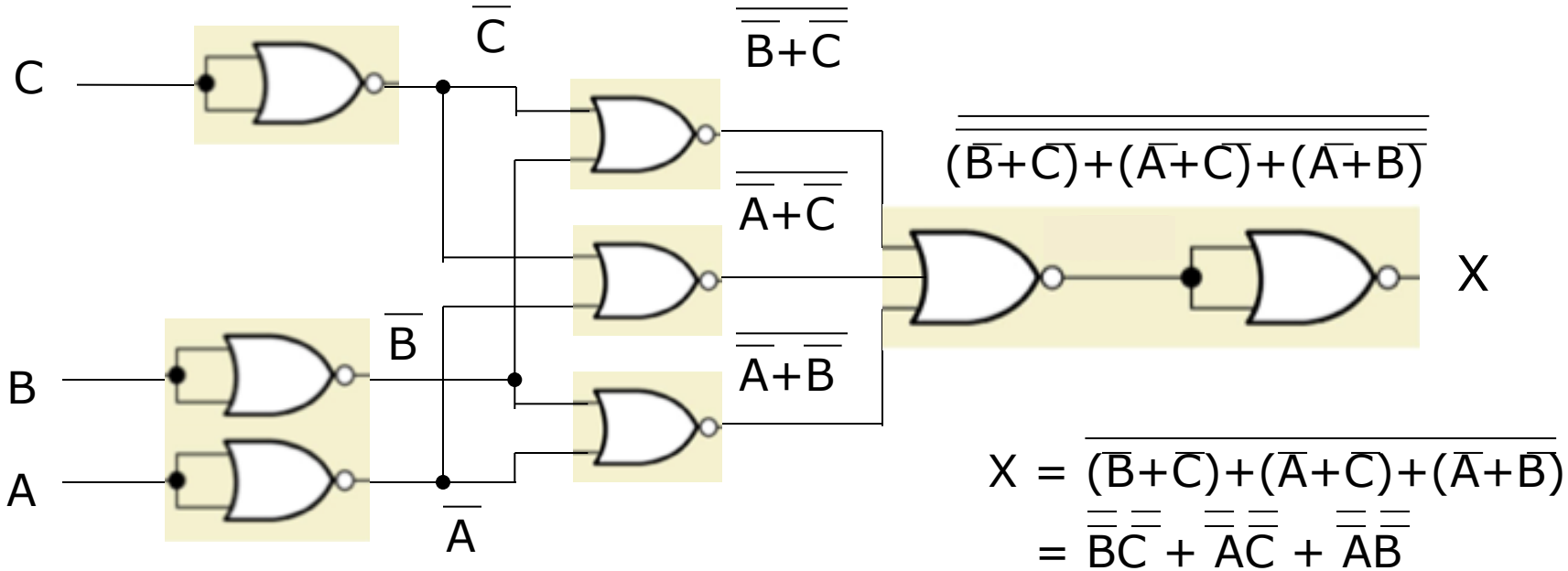
$$= BC + AC + AB$$

continue...

5.9(a) NOR gates



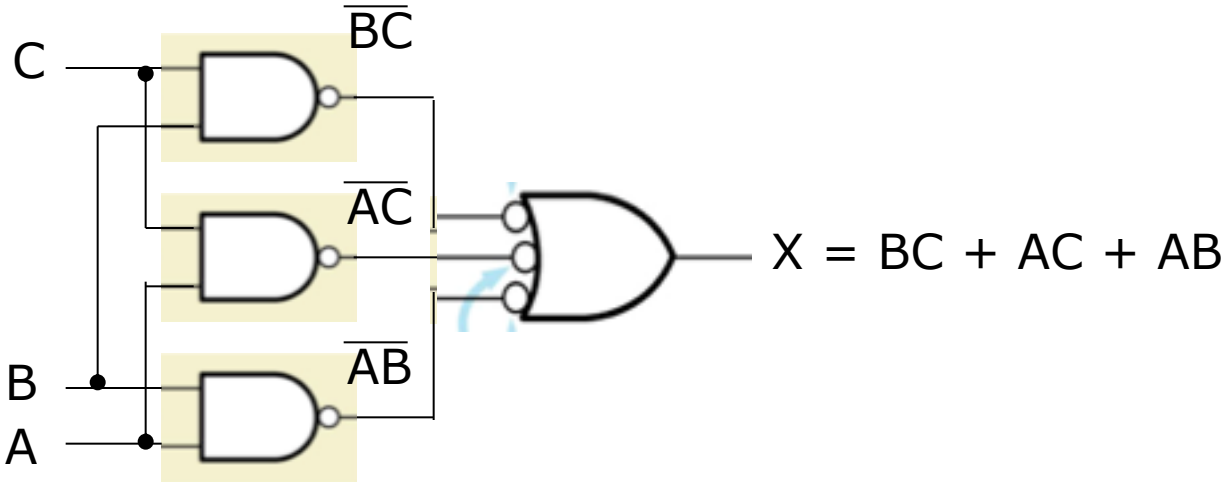
continue...



$$X = \overline{(\overline{B}+\overline{C})+(\overline{A}+\overline{C})+(\overline{A}+\overline{B})}$$
$$= \overline{B}\,\overline{C} + \overline{A}\,\overline{C} + \overline{A}\,\overline{B}$$
$$= BC + AC + AB$$

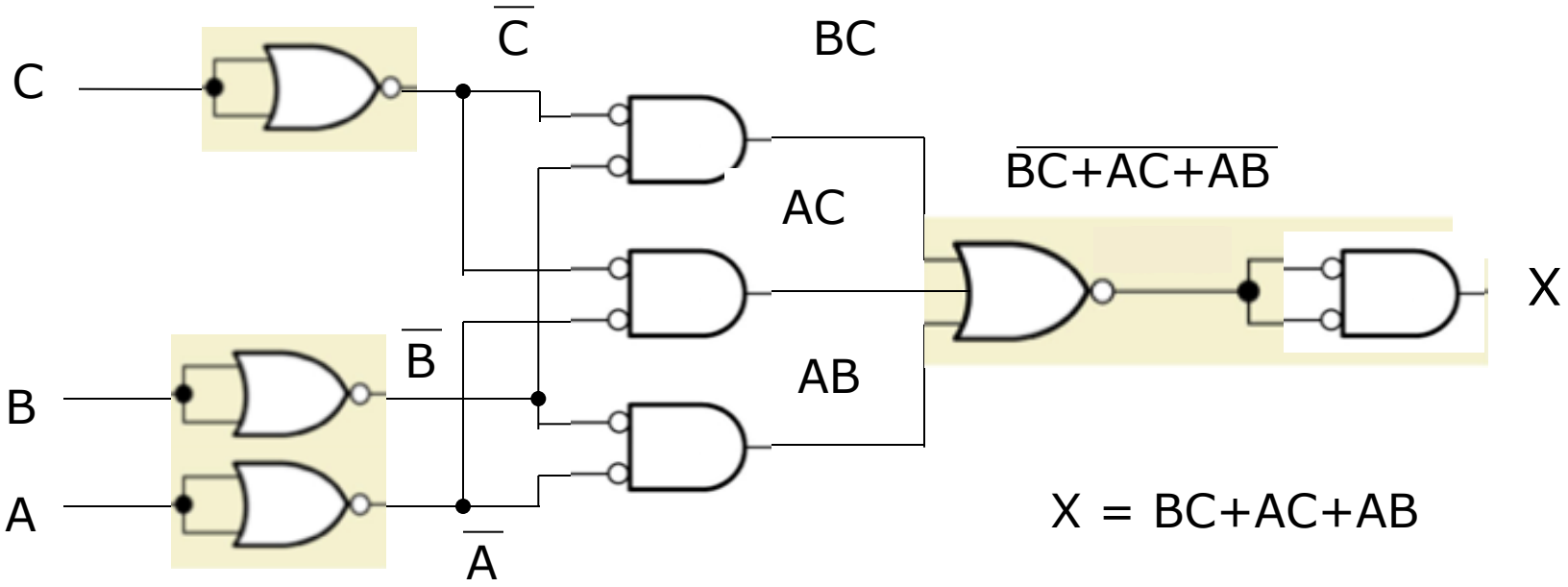
continue...

5.9(b) Dual symbol: NAND only



continue...

5.9(b) Dual symbol: NOR only



Example: multiple output..

Design a combinational circuit with 3 inputs A, B and C and 2 outputs X and Y.

When the binary input is 0,1, or 2 the binary output is one greater than the input.

When the binary input is 3, 4, 5, or 6 the binary output is three less than the input.

The binary input will never be 7.

Solution:

Step1:

Input – 3 – which is A, B AND C

Output – 2 – which is X and Y

Note: *for N number of output, treat each output separately, go through the normal design process, it will produce N different circuits, finally combine all N circuits into one*

Relationship

for 0,1,2 the output will be 1,2,3

for 3,4,5,6 the output will be 0,1,2,3

for 7, it will never exist therefore treat it as don't care

Note: input A, B, and C will be combined and treated as 3 bits binary number (ABC), X and Y combined together and treated as a binary number (XY)

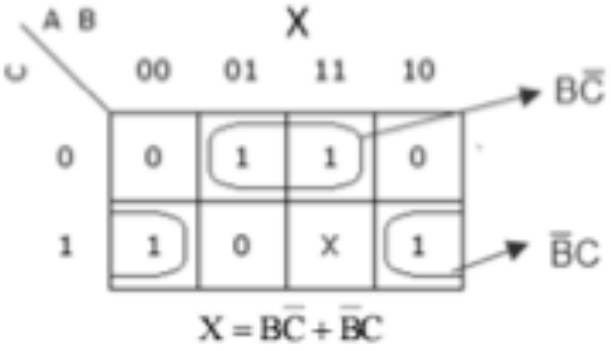
continue...

Step2:

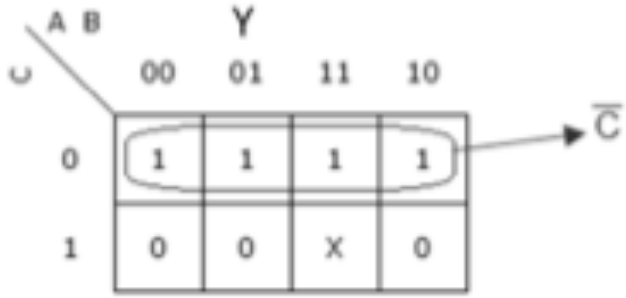
Use Truth table – the easiest in this case

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

Step 3 : Map the truth table to K-Map and simplify



$X = B\overline{C} + \overline{B}C$

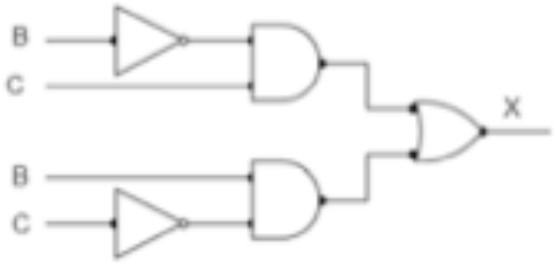


$Y = \overline{C}$

continue...

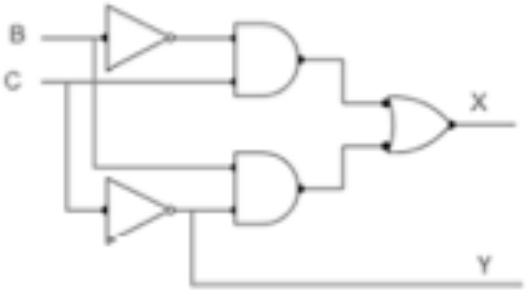
Step4: Draw the simplified circuit

circuit for X



circuit for Y

combined circuit



Exercise 5.10: **Step 5:** Implement the circuit using universal gates

Step 6: Draw the final circuit using dual symbol

Example: direct to Boolean expression

- Design a logic circuit for a car alarm that fulfill the following requirement:
  - The alarm will go off if the alarm system is activated and any of the two doors in the trunk are open , or if the vibration sensor is activated and the key is not in the ignition

Step1:

Input – determine a factor that will trigger an alarm, there are 6

Input	Condition when HIGH
AlarmActivated	activated
DriverDoorOpen	Open
PassengerDoorOpen	Open
TrunkOpen	Open
Vibration	Vibrate
KeyIgnition	in the ignition

Output – 1 output – signal to trigger an alarm ➔ 0 – alarm off, 1- alarm on, call it X

Relationship :

look for the word AND, OR , NOT (complement, invert, not the same) that connect between the input variables

The alarm will go off if the alarm system is activated ***and*** *any of the two doors* ***or*** *the trunk are open ,* ***or*** *if the vibration sensor is activated* ***and*** *the key is* ***not*** *in the ignition*

..Example: direct to Boolean expression

Step2:

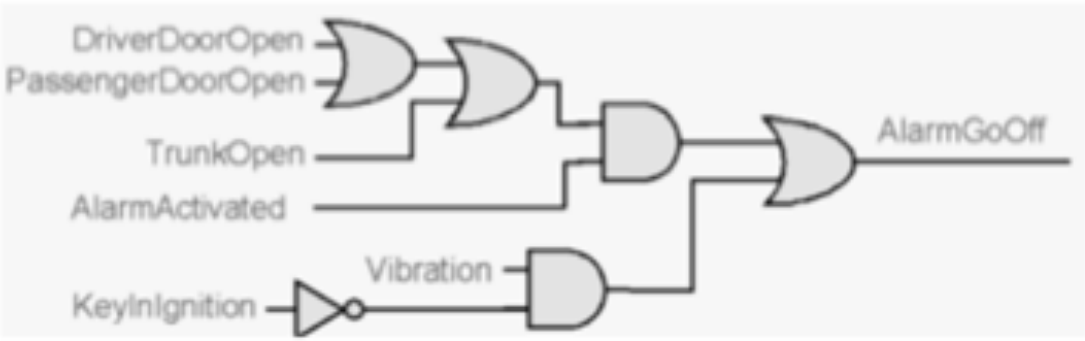
This kind of problem should not be converted to K-Map because it has 6 input and K-Map with variable more than 5 will be difficult to simplify manually.

It is much more easier to convert the problem statement directly to Boolean expression

$$AlarmGoOff = AlarmActivated.(DriverDoorOpen+PassengerDoorOpen+TrunkOpen)+Vibration.\overline{KeyInIgnition}$$

Step3: Simplify the Boolean expression if possible. In this case it is not possible

Step 4: Draw the Circuit



** Note: using only 2 input gate

..Example: direct to Boolean expression

Example 1. "I'll go to lunch if Maisarah goes OR Anis goes, AND Sabri does not go."

Let F represent my going to lunch (1 means I go, 0 I don't go)

Likewise, M for Maisarah going, A for Anis, and S for Sabri

Then **F = (M OR A) AND NOT(S)**

Example 2: An alarm system should triggered if a movement is sensed and the system is set to enabled.

Let Boolean variable

M represent "movement is sensed,"

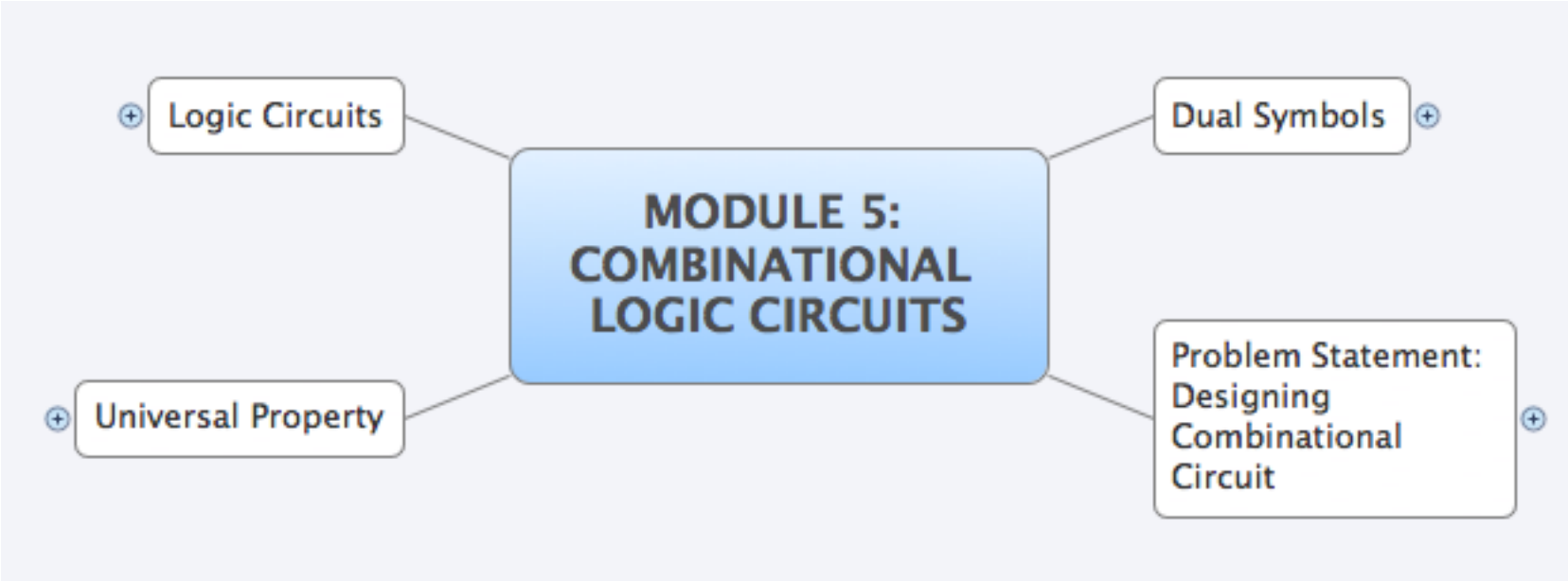
E represent "enabled,"

F represent "alarm triggered."

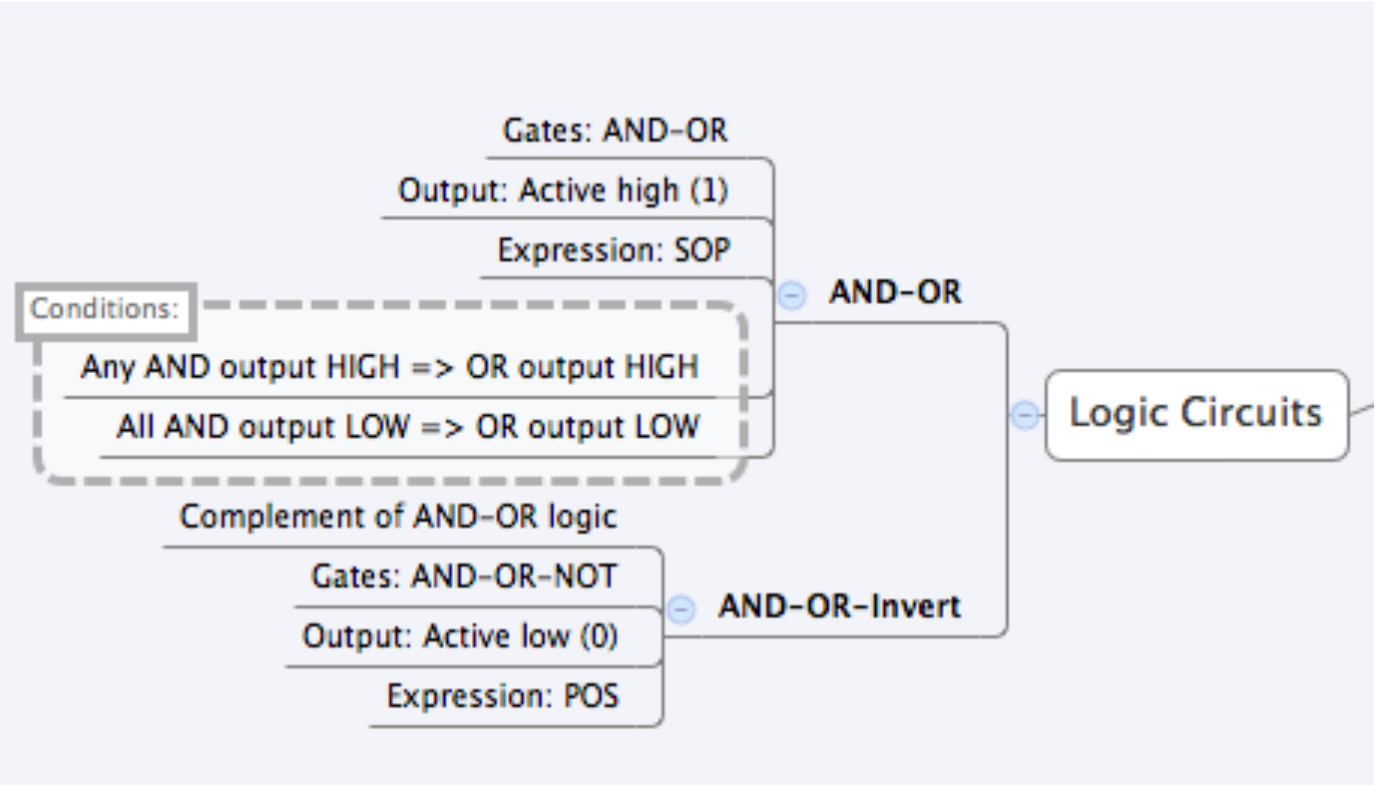
Then an equation is: **F = M AND E**

Summary

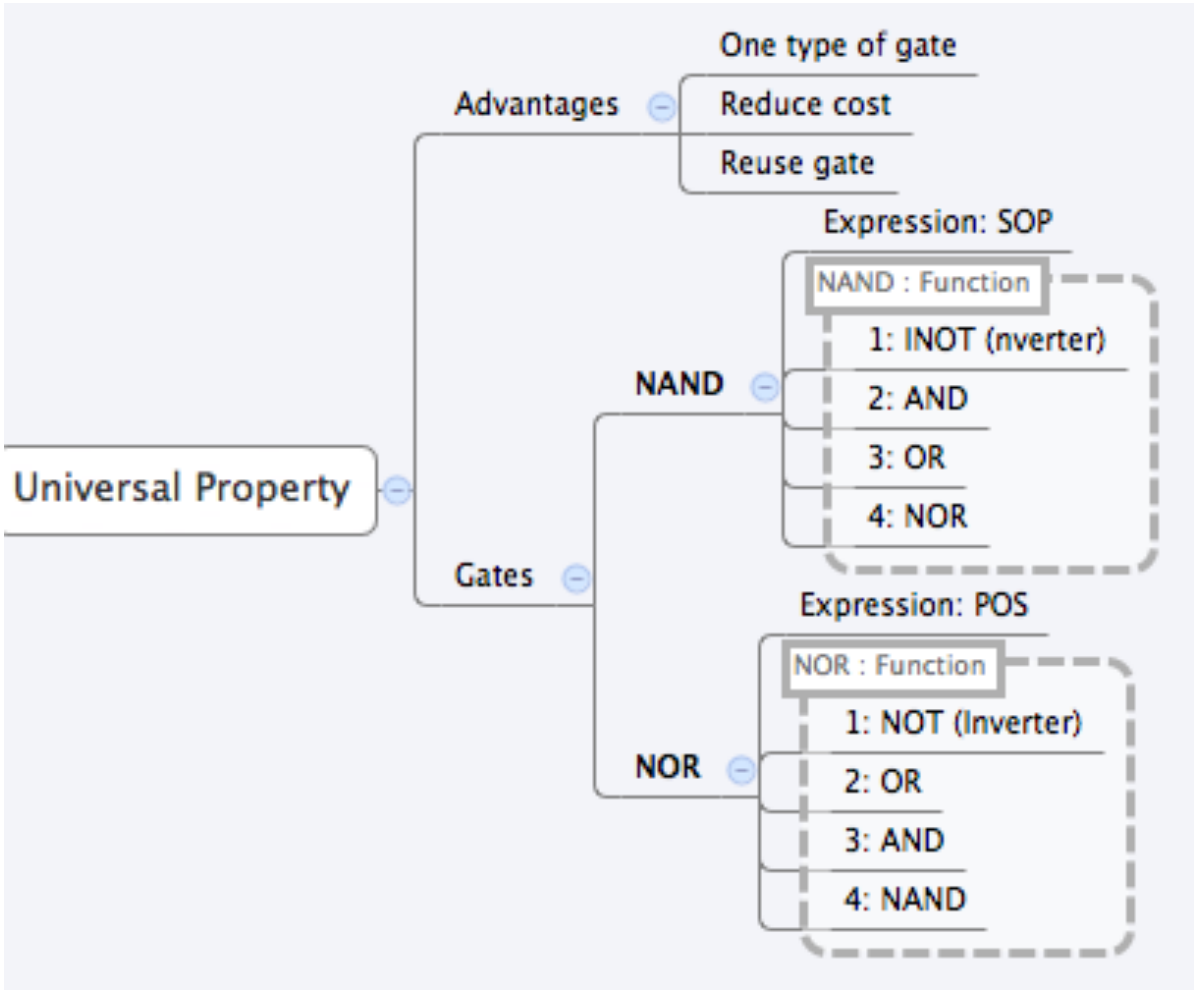
Extra



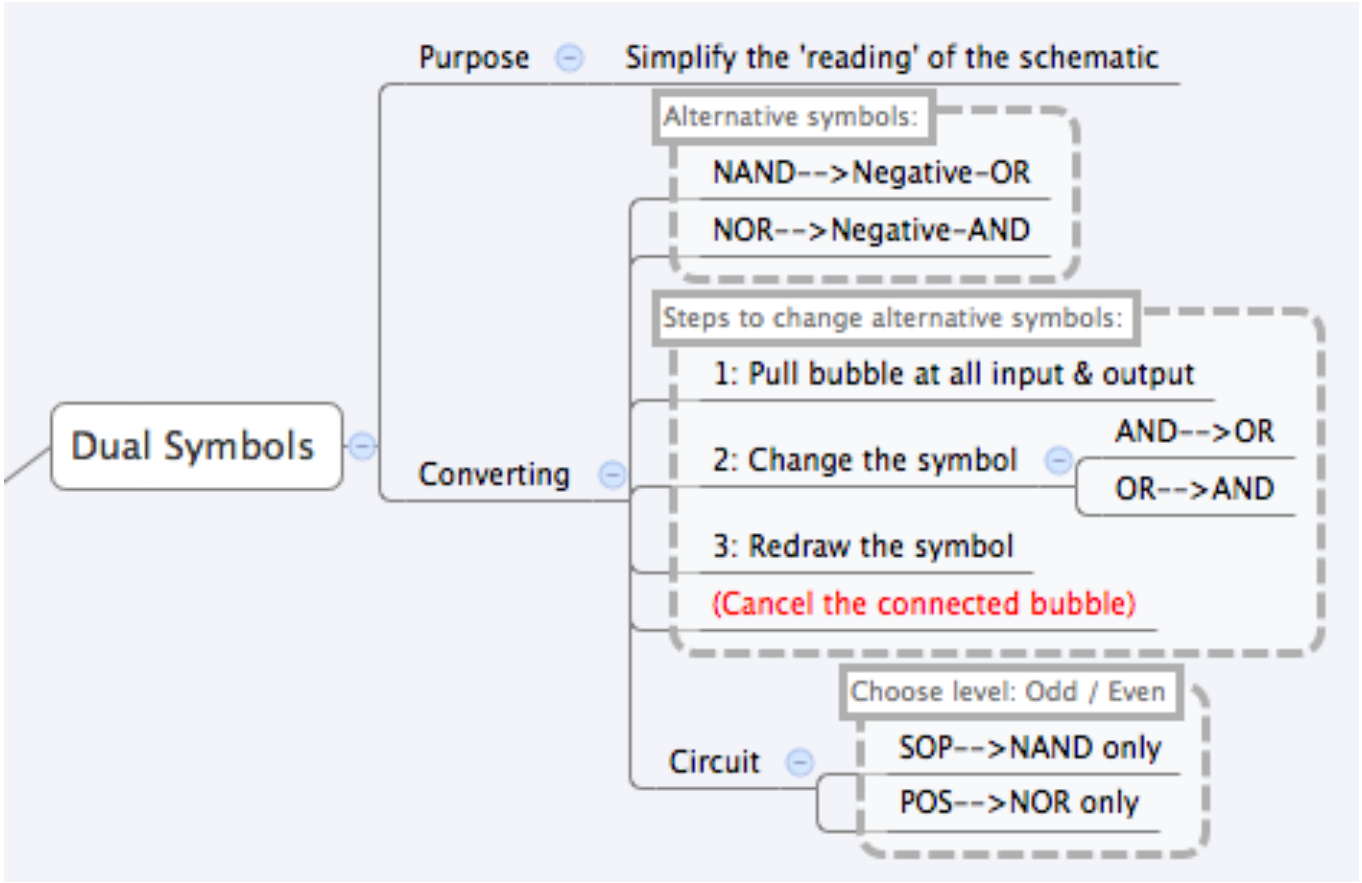
Extra


Logic Circuits
AND-OR
Gates: AND-OR
Output: Active high (1)
Expression: SOP
Conditions:
Any AND output HIGH => OR output HIGH
All AND output LOW => OR output LOW
AND-OR-Invert
Complement of AND-OR logic
Gates: AND-OR-NOT
Output: Active low (0)
Expression: POS

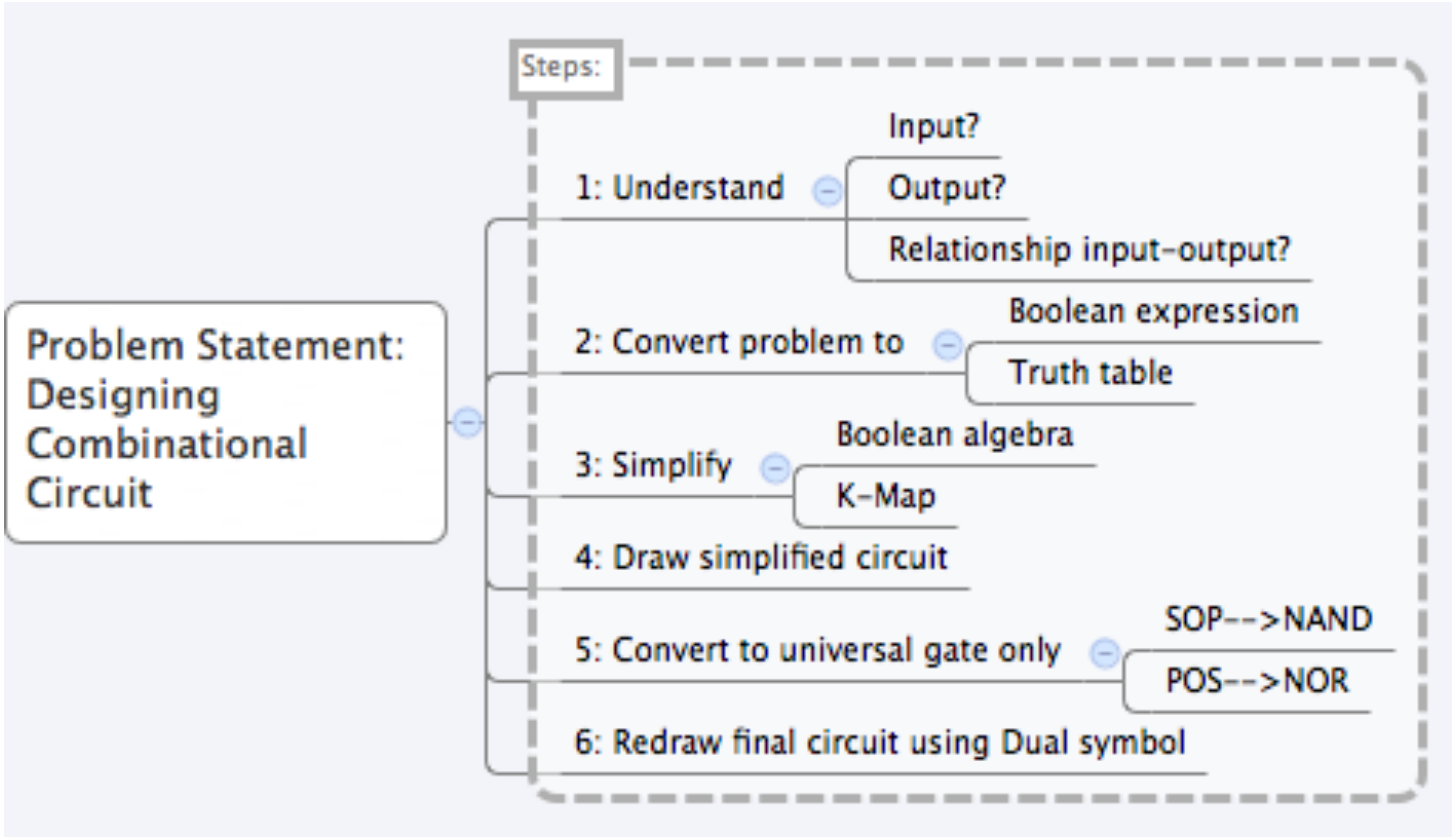
Extra


Universal Property
Advantages
One type of gate
Reduce cost
Reuse gate
Gates
NAND
Expression: SOP
NAND : Function
1: INOT (nverter)
2: AND
3: OR
4: NOR
NOR
Expression: POS
NOR : Function
1: NOT (Inverter)
2: OR
3: AND
4: NAND

Extra


Dual Symbols
Purpose
Simplify the 'reading' of the schematic
Converting
Alternative symbols:
NAND-->Negative-OR
NOR-->Negative-AND
Steps to change alternative symbols:
1: Pull bubble at all input & output
2: Change the symbol
AND-->OR
OR-->AND
3: Redraw the symbol
(Cancel the connected bubble)
Circuit
Choose level: Odd / Even
SOP-->NAND only
POS-->NOR only

Extra


Problem Statement: Designing Combinational Circuit
Steps:
1: Understand
Input?
Output?
Relationship input-output?
2: Convert problem to
Boolean expression
Truth table
3: Simplify
Boolean algebra
K-Map
4: Draw simplified circuit
5: Convert to universal gate only
SOP-->NAND
POS-->NOR
6: Redraw final circuit using Dual symbol