Digital Systems I

Instructor: Mohammad Ghamari

Electrical and Computer Engineering Department

These slides are modified version of the slides by Nozar Tabrizi, Introduction to Digital Circuits

Chapter 4

Logic Minimization

Using Karnaugh Maps

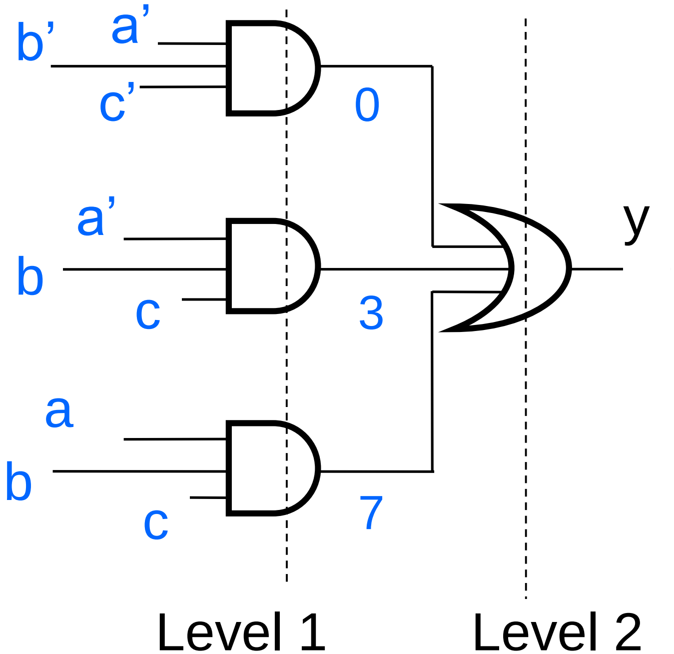
(K-map)

Example

Row	a b c	Y
0	0 0 0	1
1	0 0 1	0
2	0 1 0	0
3	0 1 1	1
4	1 0 0	0
5	1 0 1	0
6	1 1 0	0
7	1 1 1	1

What is Canonical SOP of this example?

Example

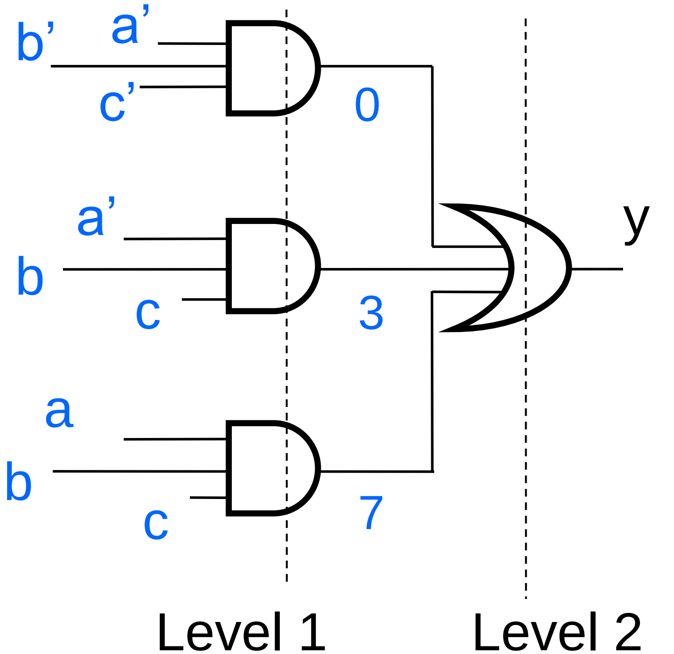


Row	a b c	Y
0	0 0 0	1
1	0 0 1	0
2	0 1 0	0
3	0 1 1	1
4	1 0 0	0
5	1 0 1	0
6	1 1 0	0
7	1 1 1	1

$Y = \sum a, b, c \quad (0, 3, 7)$

Canonical SOP: Y = a’ . b’. c’ **+** a’ . b . c **+** a . b . c

Example



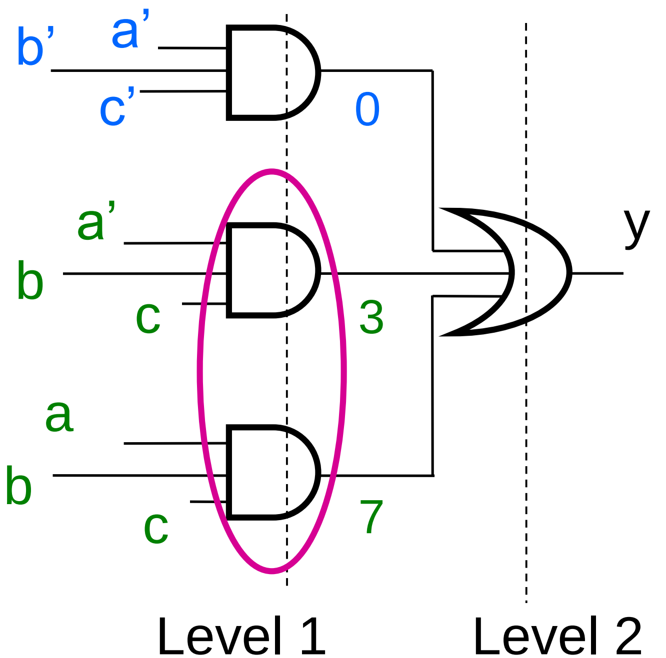
Row	a b c	Y
0	0 0 0	1
1	0 0 1	0
2	0 1 0	0
3	0 1 1	1
4	1 0 0	0
5	1 0 1	0
6	1 1 0	0
7	1 1 1	1

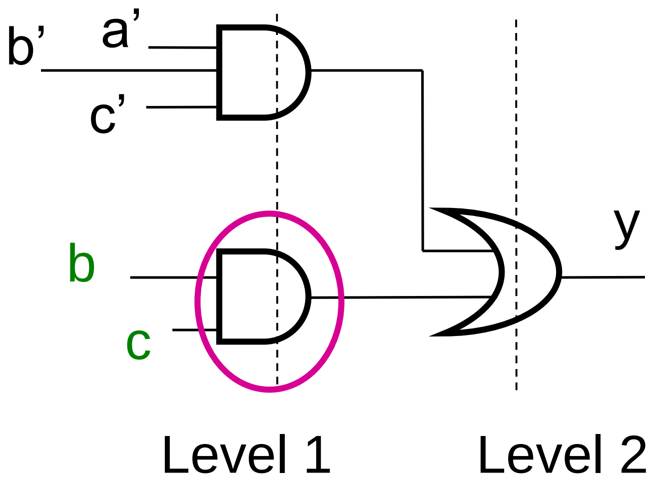
$Y = \sum a, b, c \;\; (0, 3, 7)$

Canonical SOP: $Y = a' . b' . c' + a' . b . c + a . b . C$

Now please use switching algebra to simplify this circuit.

Example





Row	a b c	Y
0	0 0 0	1
1	0 0 1	0
2	0 1 0	0
3	0 1 1	1
4	1 0 0	0
5	1 0 1	0
6	1 1 0	0
7	1 1 1	1

$Y = \sum a, b, c \quad (0, 3, 7)$

Combining(T10): a . b + a . b' = a

Canonical SOP: $Y = a'.b'.c' + a'.b.c + a.b.c$

Combining theorem: $Y = a'.b'.c' + b.c$

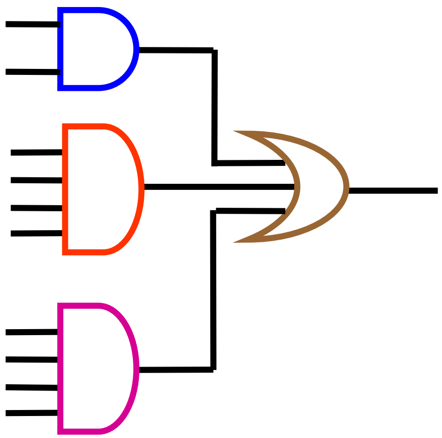
Example from Chapter 3:

$$(a \,.\, b) + (a \,.\, b \,.\, c' .\, d) + (a \,.\, b \,.\, d \,.\, e') = a \,.\, b$$

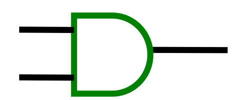
Left side: 2-input AND 4-input AND 4-input AND

3-input OR

Right side: 2-input AND



Left



Right

Huge Difference!

How to simplify?

- **Switching algebra?**

Powerful & flexible, but

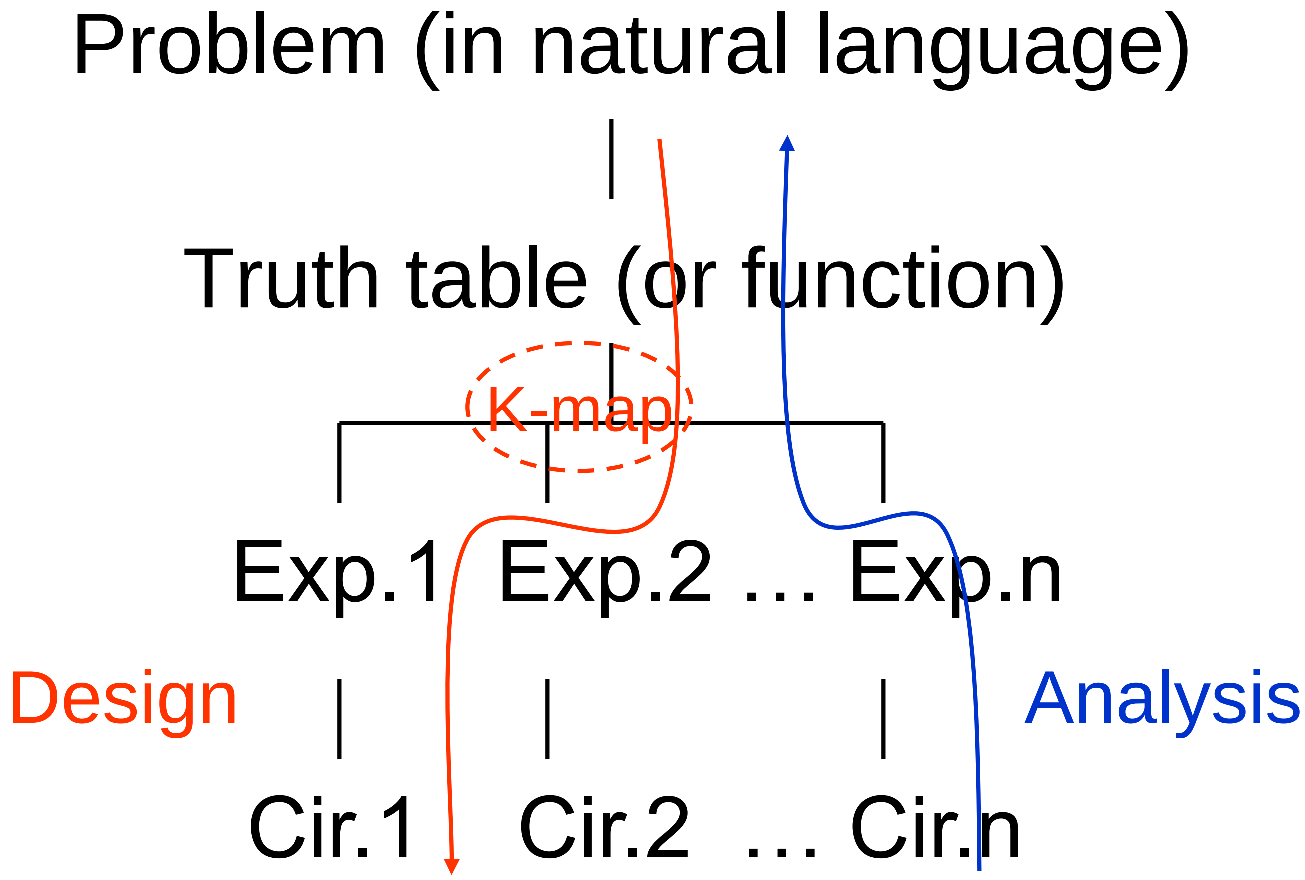
Not easy to apply manually.

- **Karnaugh maps**, or **K-maps** for short

A graphical representation for logic functions.

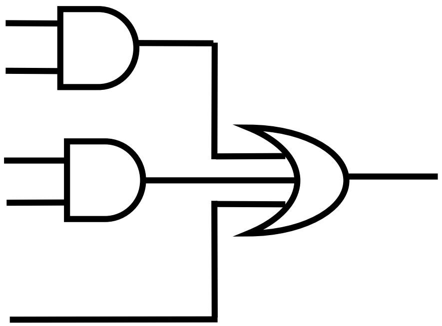
A two-dimensional version of truth table.

K-map-based procedure is able to obtain a *minimal* (2-level) SOP (& POS) for any switching function.

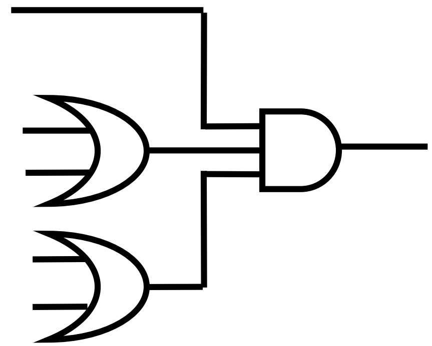

Problem (in natural language)
Truth table (or function)
K-map
Exp.1 Exp.2 … Exp.n
Design
Analysis
Cir.1 Cir.2 … Cir.n

What is a 2-level logic?

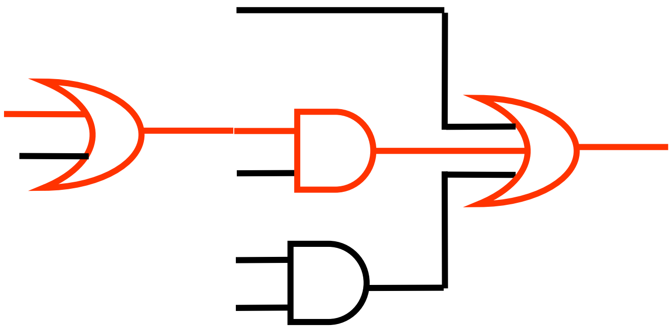
Each signal passes through 2 gates at the most to reach the output.



2-level



2-level



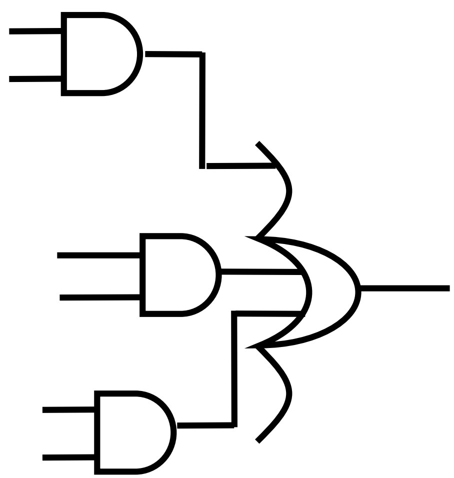
3-level

SOP and POS are 2-level logic

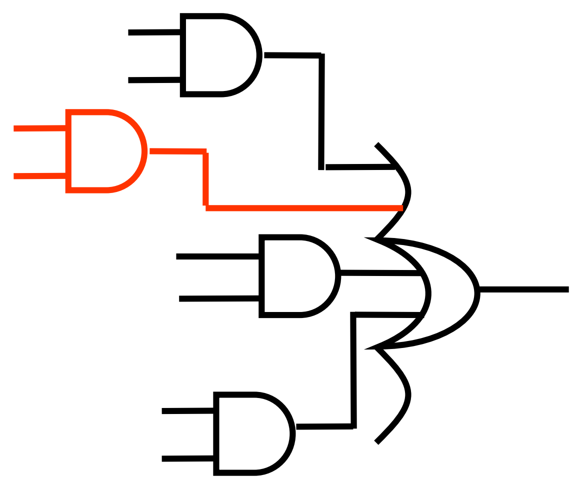
What is a minimal SOP?

By a minimal SOP we mean a SOP expression with as few product terms (AND terms) as possible.

If these are 2 choices, which one is minimal ?



Minimal



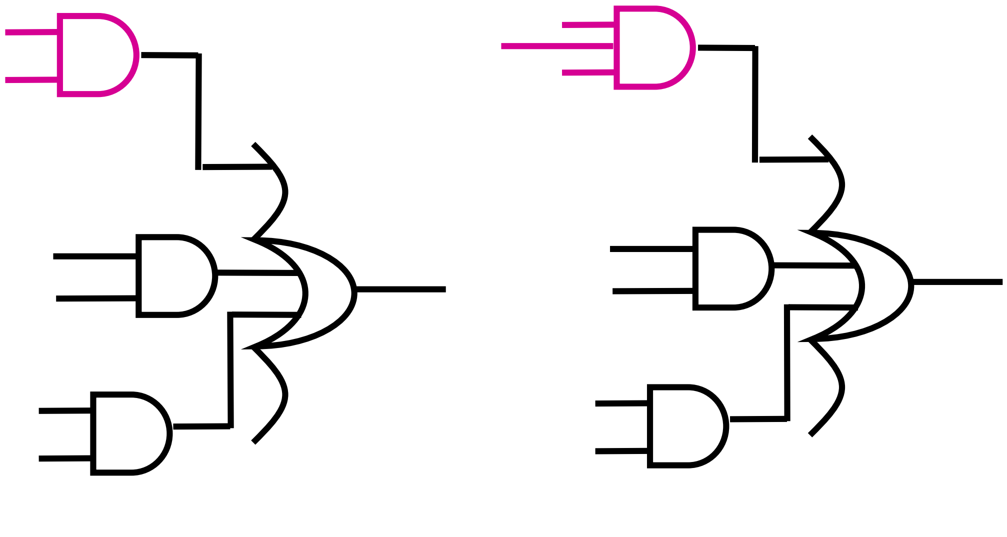
Not minimal

What is a minimal SOP? (Cont'd)

If there are 2 or more SOP expressions meeting this criterion, then the minimal SOP is the one with as few literals as possible.

If these are 2 choices, which one is minimal

Minimal SOP may not be unique.



Minimal Not minimal

To avoid confusion, first consider minimal SOP.

Then concepts developed for SOP will easily be extended to POS.

K-maps: two-dimensional truth tables

C \ AB	00	01	11	**10**
0	0	2	6	**4**
1	1	3	7	5

3-variable K-map

Row	ABC	Y
0	0 0 0	
1	0 0 1	
2	0 1 0	
3	0 1 1	
4	**1 0 0**	
5	1 0 1	
6	1 1 0	
7	1 1 1	

3-variable TT

Each box in K-map corresponds to one minterm

C \ AB	00	01	11	**10**
0	0 A’B’C’	2 A’BC’	6 ABC’	**4 AB’C’**
1	1 A’B’C	3 A’BC	7 ABC	5 AB’C

3-variable K-map

Row	A B C	Minterm
0	0 0 0	A’B’C’
1	0 0 1	A’B’C
2	0 1 0	A’BC’
3	0 1 1	A’BC
4	**1 0 0**	**AB’C’**
5	1 0 1	AB’C
6	1 1 0	ABC’
7	1 1 1	ABC

3-variable TT

Transfer output column to K-map

C \ AB	00	01	11	**10**
0	0 1	2 0	6 0	4 0
1	1 0	3 1	7 1	5 0

K-map representation

Cell 3: 1-cell or on-set cell
Cell 5: 0-cell or off-set cell

Row	ABC	Y
0	0 0 0	1
1	0 0 1	0
2	0 1 0	0
3	0 1 1	1
4	1 0 0	0
5	1 0 1	0
6	1 1 0	0
7	1 1 1	1

TT representation

2- & 4-variable K-maps

B \ A	**0**	**1**
0	0	2
1	1	3

2-variable K-map

CD \ AB	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

4-variable K-map

Look at horizontal & vertical code words

CD \ AB	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

00, 01, 10, 11 Normal binary

00, 01, 11, 10 Gray code

Question 1.

What is the point in using Gray code in K-maps?

Wait …

Definition (in K-map domain)

Two cells are *logically adjacent* if their coordinates are different in **exactly** one bit.

e.g. cells 6 & 14:

ABCD = 0110 & ABCD = 1110.

CD \ AB	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

Definition (in algebraic domain)

Two minterms are logically adjacent if they differ in **only** one variable.

e.g. **A’** . B . C . D’ & **A** . B . C . D’

CD \ AB	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

A’BCD’ ABCD’

Conclusion

Two minterms are logically adjacent if they belong to two logically adjacent cells and vice versa.

Example 1. Use switching algebra to simplify

$Y(A, B, C) = \sum(2, 6)$

$Y(A, B, C) = A' . B . C' + A . B . C'$

Apply T10-L to the two product terms:

C \ AB	00	01	11	10
0	0: 0	2: 1 A'BC'	6: 1 ABC'	4: 0
1	1: 0	3: 0	7: 0	5: 0

Combining

T10 $\quad a . b + a . b' = a \qquad (a + b) . (a + b') = a$

$Y = A' . B . C' + A . B . C' = B . C'$

Original circuit: two 3-input AND, one 2-input OR

Simplified circuit: one 2-input AND

Minterms A' . B . C' & A . B . C' are logically adjacent, because they differ in only one variable.

Conclusion

Two **logically adjacent** minterms, hence two **logically adjacent** 1-cells can be combined resulting in a simpler logic circuit.

Therefore

To minimize a logic circuit we need to identify all **logically adjacent** minterms or **logically adjacent 1-cells**.

Intermediate goal:

Identify all logically adjacent 1-cells.

Definition

Physically adjacent cells: 2 cells with 1 common side (edge)

CD \ AB	00	**01**	**11**	10
00	0	4	**12**	8
01	**1**	5	13	**9**
11	3	**7**	**15**	11
10	2	6	**14**	10

CD \ AB	00	**01**	**11**	10
00	0	4	AB C'D'	8
01	A'B' C'D	5	13	AB' C'D
11	3	A'B CD	AB CD	11
10	2	6	AB CD'	10

Assume

2 top & bottom sides are the same,

Also 2 right & left sides are the same.

Some examples

Question 1. (now we are ready to answer)

What is the point in using Gray code in K-maps?

00, 01, 10, 11 Normal binary

00, 01, 11, 10 Gray code

Answer 1. By using Gray code, *physically* adjacent cells become *logically* adjacent as well, and vice versa.

Question 2. Why is it important to make *physically adjacent* cells *logically adjacent* as well, and vice versa?

Answer 2.

Remember our intermediate goal:

Identify all logically adjacent 1-cells.

On the other hand,

Physically-adjacent 1-cells are identified at a glance.

Therefore, logically adjacent minterms are identified at a glance as well.

We have reached our intermediate goal!

CD \ AB	00	**01**	**11**	10
00	0	4	**12**	8
01	**1**	5	13	**9**
11	3	**7**	**15**	11
10	2	6	**14**	10

Back to **Example 1**. Y(A, B, C) = ∑(2, 6)

Cells 2 and 6 are physically, hence logically adjacent

So the corresponding minterms (A' . B . C' , A . B . C') can be combined.

A is different in 2 minterms, drop it; keep B & C':

A' . B . C' + A . B . C' = B . C'

C \ AB	00	**01**	**11**	10
0	0 0	2 A'BC' **1**	6 A BC' **1**	4 0
1	1 0	3 0	7 0	5 0

In Summary

Two adjacent minterms can be combined to produce one single *p-term* with *one variable fewer* than each minterm has.

To combine them, drop the only variable that appears as two different literals in the two minterms & keep the remaining literals.

$$A' . B . C' + A . B . C' = B . C'$$

Example 2. (p. 6)

Use K-map to minimize $Y = \sum_{A, B, C} (2, 3)$.

Try to solve this