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

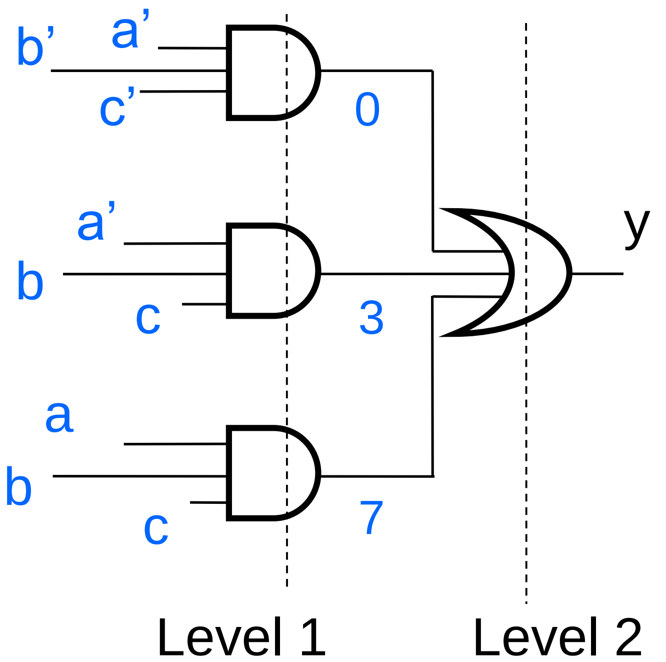
(K-map)

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

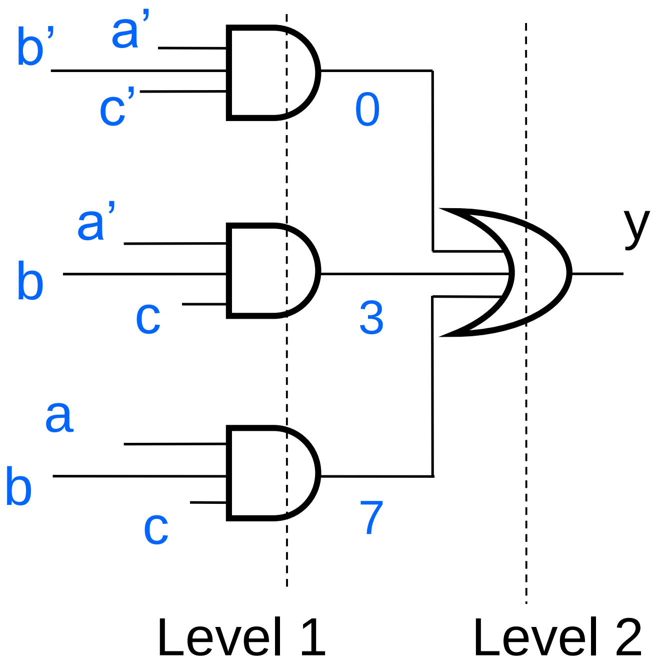


Row	a b c	Y
0	0 0 0	1
1	0 0 1	0
2	0 1 0	0
3	0 1 1	1
4	1 0 0	0
5	1 0 1	0
6	1 1 0	0
7	1 1 1	1

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

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

Example



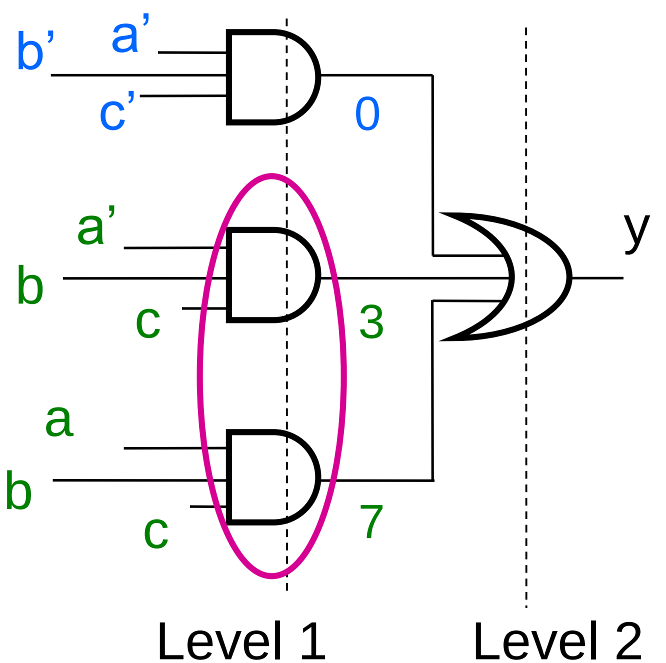
Row	a b c	Y
0	0 0 0	1
1	0 0 1	0
2	0 1 0	0
3	0 1 1	1
4	1 0 0	0
5	1 0 1	0
6	1 1 0	0
7	1 1 1	1

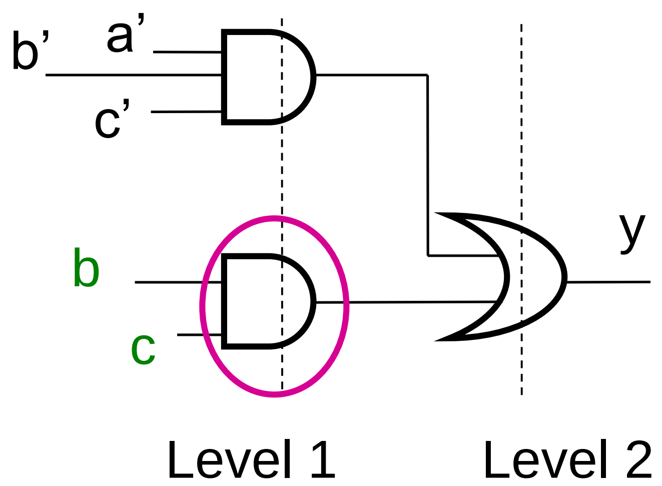
$Y = \sum a, b, c \quad (0, 3, 7)$

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

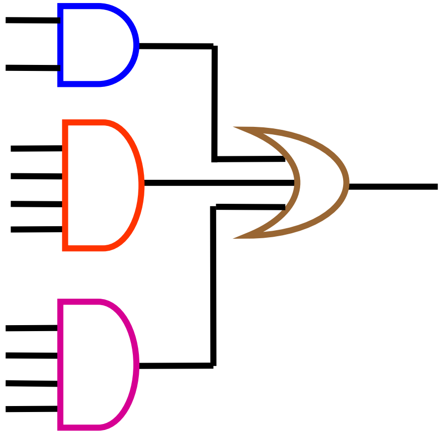
Example from Chapter 3:

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

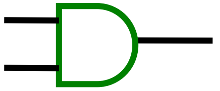
Left side: 2-input AND 4-input AND 4-input AND

3-input OR

Right side: 2-input AND



Left



Right

Huge Difference!

How to simplify?

- **Switching algebra?**

Powerful & flexible, but

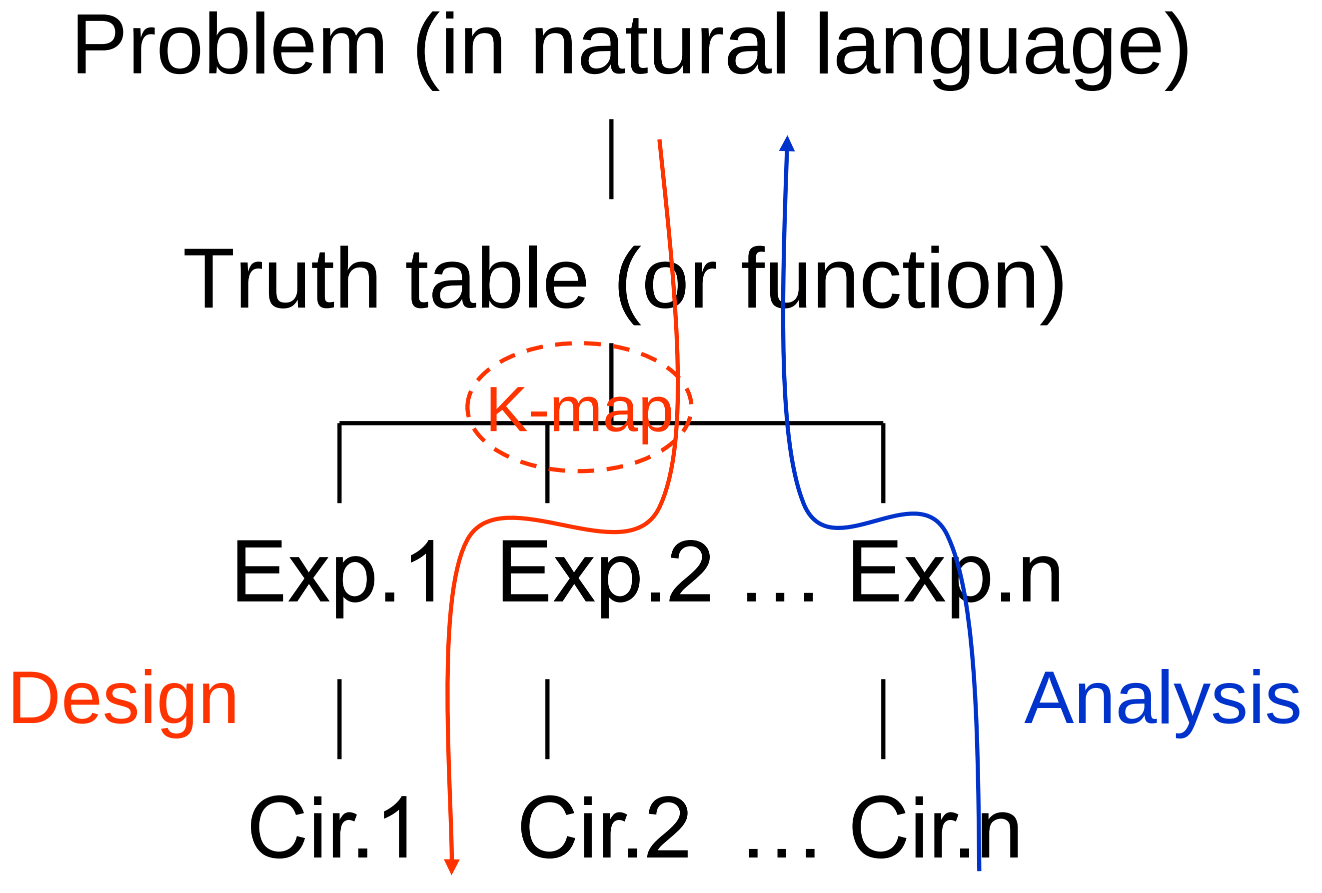
Not easy to apply manually.

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

A graphical representation for logic functions.

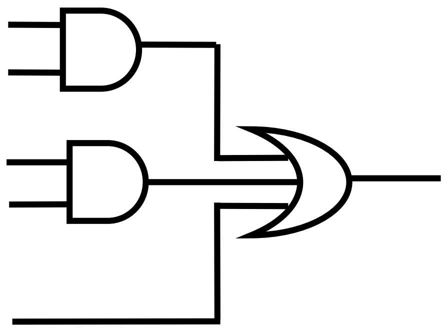
A two-dimensional version of truth table.

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

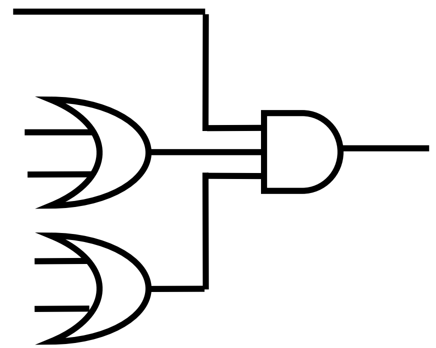

Problem (in natural language)
Truth table (or function)
K-map
Exp.1 Exp.2 … Exp.n
Design
Analysis
Cir.1 Cir.2 … Cir.n

What is a 2-level logic?

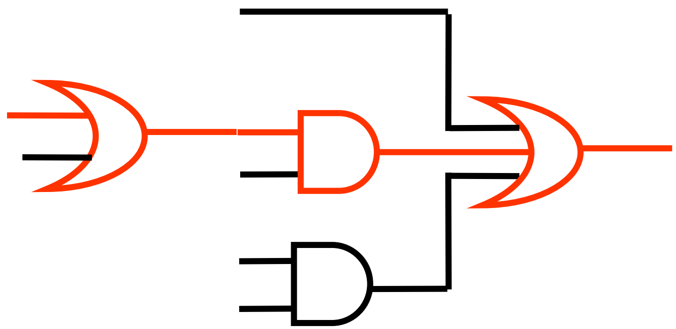
Each signal passes through 2 gates at the most to reach the output.



2-level



2-level



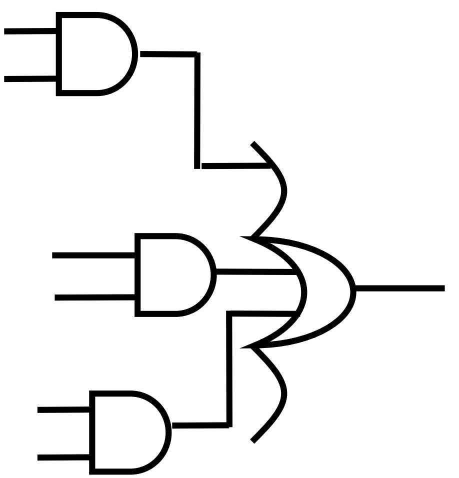
3-level

SOP and POS are 2-level logic

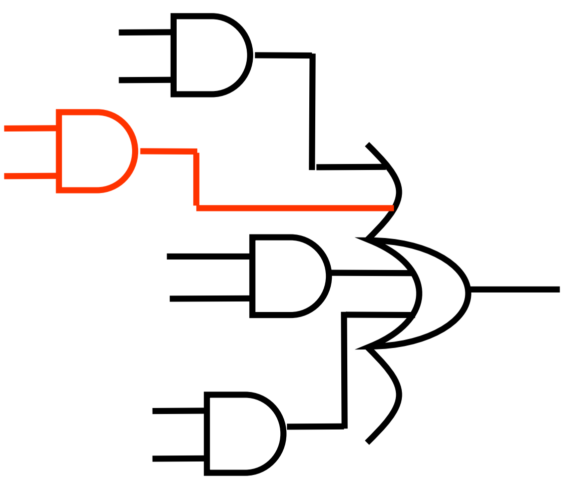
What is a minimal SOP?

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



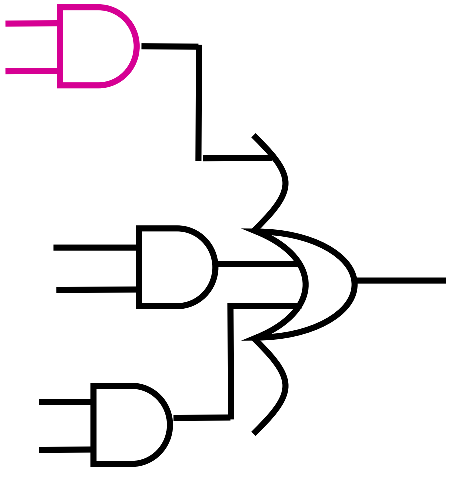
Not minimal

What is a minimal SOP? (Cont'd)

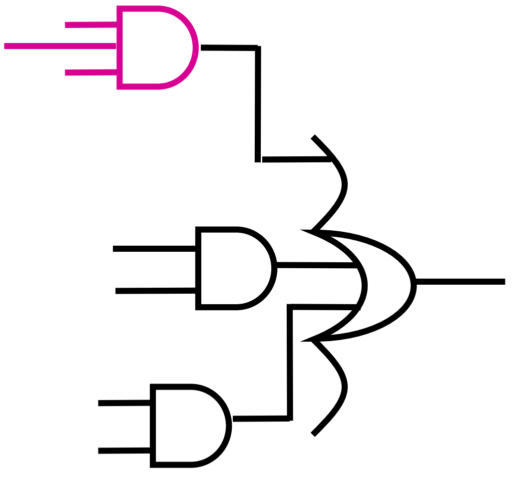
If there are 2 or more SOP expressions meeting this criterion, then the minimal SOP is the one with as few literals as possible.

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. $A' . B . C . D'$ & $A . B . C . D'$

Conclusion

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

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'

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 $a . b + a . b' = a$ $(a + b) . (a + b') = a$

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

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

Cells 2 and 6 are physically, hence logically adjacent

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

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

$A' \cdot B \cdot C' + A \cdot B \cdot C' = B \cdot C'$

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

Example 2. (p. 6)

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

The canonical SOP of this function is Y = A'.B.C' + A'.B.C

- These cells are physically, hence logically adjacent.
- To combine them, variable C is dropped. Therefore: **Y = A'.B**

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

Example 3. (p. 7)

Use a K-map to minimize $Y = \sum_{A, B, C, D} (6, 14)$.

Try to solve this

Example 3. (p. 7)

Minimize $Y = \sum_{A,\,B,\,C,\,D} (6, 14) = A' . B . C . D' + A . B . C . D'$

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

Example 3. (p. 7)

Minimize Y = $\sum_{A, B, C, D}$ (6, 14) = 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 1	14 1	10

Rule 1. Keep B,C&D.

(In coordinates of cells

B, C&D do not change.)

Rule 2. D is inverted,

but B & C are not.

So, Y = B . C . D'

Cell is doubled (1-Ecell)

(Extended Cell)

The algorithm to combine two adjacent 1-cells and obtain a p-term

- **Rule 1.** Obtain the right variables:

Determine the only variable that is not fixed in coordinates of the two 1-cells. Discard this variable and keep the rest.

- **Rule 2.** Obtain the right primes (or negations):

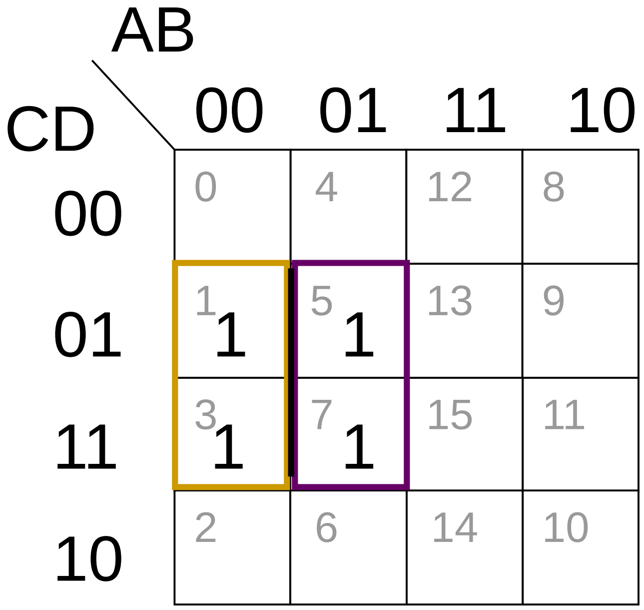
If the fixed value of a variable is 1, it will participate in the minimized p-term as a *non-inverted* variable; otherwise, the variable will be *inverted*. The resulting p-term represents a larger rectangular cell comprised of the two original 1-cells.

Repetitive Combining:
An Extension to Single-Cell Combining

1-Ecell Combining

Two same-size E-cells are ***physically*** adjacent if they have (at least) one same-size side in common.

Therefore, different-size E-cells cannot be adjacent.

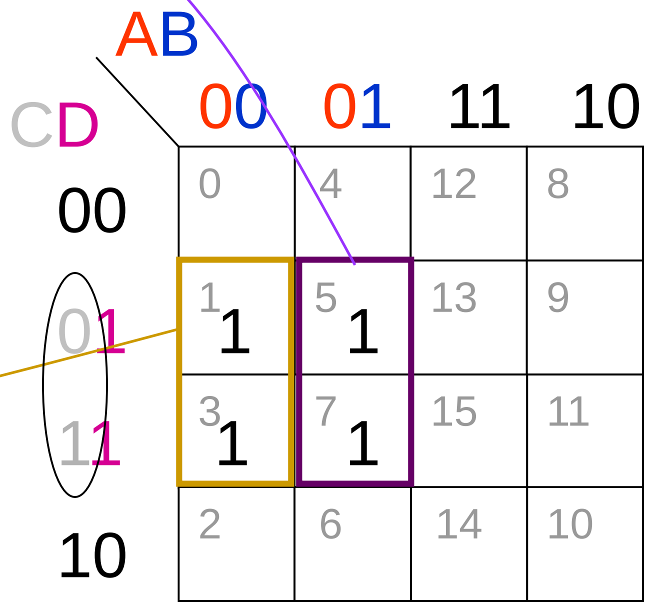


Two same-size 1-Ecells are ***logically*** adjacent if within the coordinates of these Ecells the value of ***only*** *one* variable changes (the remaining variables each stay at a fixed value).

*** Coordinates ***

ABD = 001 & 011

A & D have fixed values, but
B does not. (logically adjacent)



It can be shown that

Two physically adjacent Ecells are always logically adjacent as well, and vice versa.

In General

Two adjacent 1-Ecells can be combined.

The resulting 1-Ecell is

- twice as large &
- represented by a p-term comprised of all literals ***shared*** by original p-terms.

This combining procedure may continue <u>until no combining is possible anymore</u>.

<u>The more 1-Ecells are combined</u>, <u>the larger the resulting 1-Ecell</u>, hence the smaller (less expensive) the resulting p-term is.

Without loss of generality, a 1-cell may also be considered a (single-cell) 1-Ecell.

Definition: (p. 9) An on-set E-cell is called a *prime implicant* if it cannot grow anymore (in the K-map domain) .

Conclusion: All p-terms in a minimal SOP must be prime implicants.

Phase 1 of logic minimization using K-maps:

Obtain all prime implicants of function under consideration.

Example 6. (p. 10) Obtain prime implicant of

$Y = \sum A, B, C, D\ (8, 9, 10, 11, 12, 13, 14, 15)$.

Try to solve this

Example 6. (p. 10) Obtain prime implicant of

Y = ∑ A, B, C, D (8, 9, 10, 11, 12, 13, 14, 15).

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

(12,13):	A . B . C’	upper left E-cell
(8, 9):	A . B’ . C’	upper right E-cell
(14, 15):	A . B . C	lower left E-cell
(11,10):	A . B’ . C	lower right E-cell

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

Is A . B’ . C’ a prime implicant?

Is A . B’ . C a prime implicant?

Is A . C a prime implicant?

(12,13): A . B . C’
(8, 9): A . B’ . C’
} A . C’

(14, 15): A . B . C
(11,10): A . B’ . C
} A . C

A . C’, A . C } A

- **A** is prime implicant which represents the eight-cell 1-Ecell.
- Y = 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' + A.B.C.D = **A**

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

Is A . B' . C' a prime implicant?

Is A . B' . C a prime implicant?

Is A . C a prime implicant?

(12,13): A . B . C'
(8, 9): A . B' . C'
} A . C'

(14, 15): A . B . C
(11,10): A . B' . C
} A . C

} A

Definition: (p. 9) An on-set E-cell is called a *prime implicant* if it cannot grow anymore (in the K-map domain) .

Conclusion: All p-terms in a minimal SOP must be prime implicants.

Phase 1 of logic minimization using K-maps:

Obtain all prime implicants of function under consideration.

Notice that

Any rectangle with 2^k 1-cells is a 1-Ecell, where k is an integer.

For k = 0 the number of participating 1-cells in the 1-Ecell becomes 1, signifying a *single-cell* 1-Ecell.

A 1-Ecell is a prime implicant if the 1-Ecell cannot grow anymore.

Next slides show examples on a one-step procedure (shortcut) to obtain a prime implicant

Example 8a. prime implicant ?

$$Y1 = \sum A, B, C, D\ (0, 1, 4, 5)$$

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

$$Y = A' \cdot C'$$

Example 8d.

Obtain prime implicant of $Y4 = \sum A, B, C, D\ (1, 5, 9, 13)$

Try to solve this

Example 8d.

$$Y4 = \sum A, B, C, D\ (1, 5, 9, 13)$$

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

$$Y = C' . D$$

Example 9a.

Obtain prime implicant of $Y1 = \sum A, B, C, D\ (1, 3, 9, 11)$.

Try to solve this

Example 9a.

$$Y1 = \sum A, B, C, D\ (1, 3, 9, 11).$$

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

$$Y = B' . D$$

Example 10. (p. 13)

Obtain prime implicant of $Y1 = \sum A, B, C, D\ (8, 10)$.

Try to solve this

Example 10. (p. 13)

$Y1 = \sum A, B, C, D\ (8, 10).$

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

$Y = A \,.\, B' \,.\, D'$

Example 11.

Obtain prime implicant of $Y2 = \sum A, B, C, D\ (0, 2, 8, 10)$.

Try to solve this

Example 11.

$Y2 = \sum A, B, C, D\ (0, 2, 8, 10).$

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

$Y = B' \cdot D'$

Summary: To obtain the p-term of a prime implicant

(An extension to rules 1 & 2)

Locate a rectangle made up of 2^k 1-cells. The rectangle must not be able to grow anymore. Then follow the following guidelines to obtain the p-term:

- **Rule 3.** Obtain the right variables:

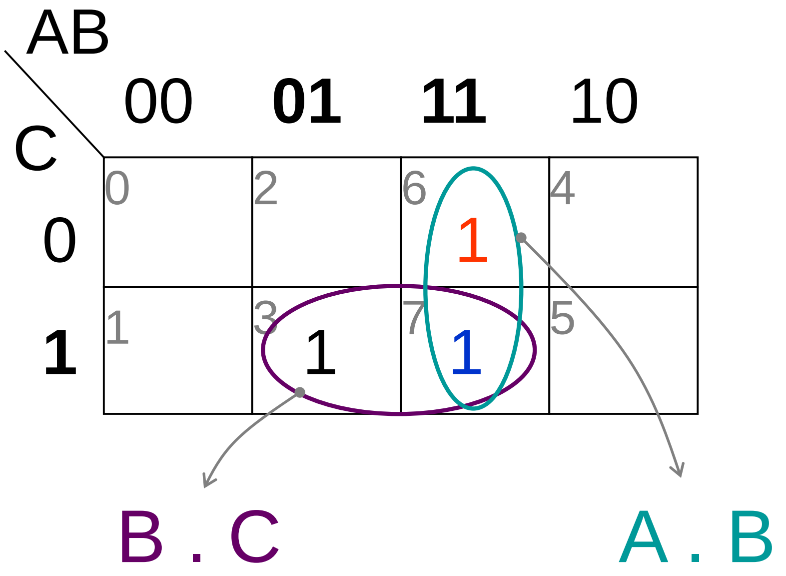
Determine variables each with a fixed value in coordinates of all participating 1-cells. Keep these variables and discard the rest.

- **Rule 4.** Obtain the right primes (negations):

If the fixed value of a variable (which was kept according to Rule 3) is 1, then that variable will participate in p-term as a *non-inverted* variable, otherwise the variable will be *inverted*.

Complete SOPs versus Minimal SOPs

Definition: The sum of all prime implicants of a function is called the *complete SOP*.



Prime implicant?

Complete SOP

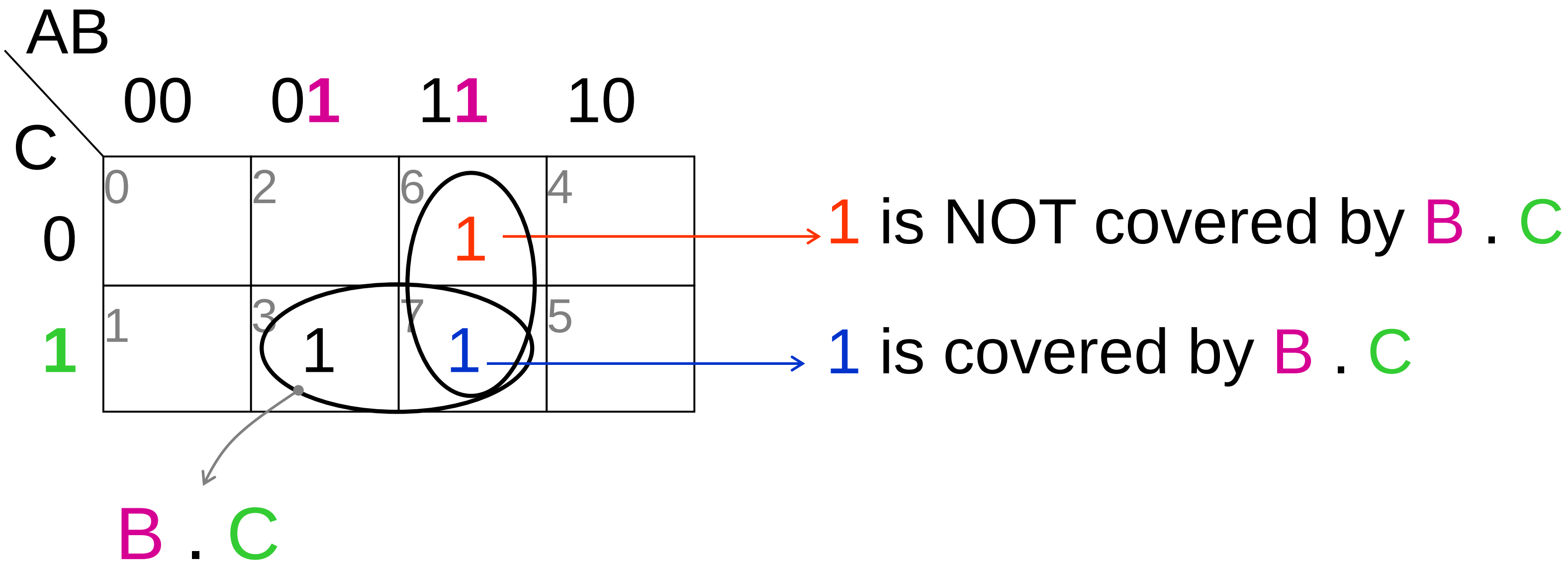
B . C + A . B

The complete SOP is always a correct algebraic expression to represent the corresponding function.

$$Y = B \,.\, C + A \,.\, B \quad \text{(complete SOP)}$$

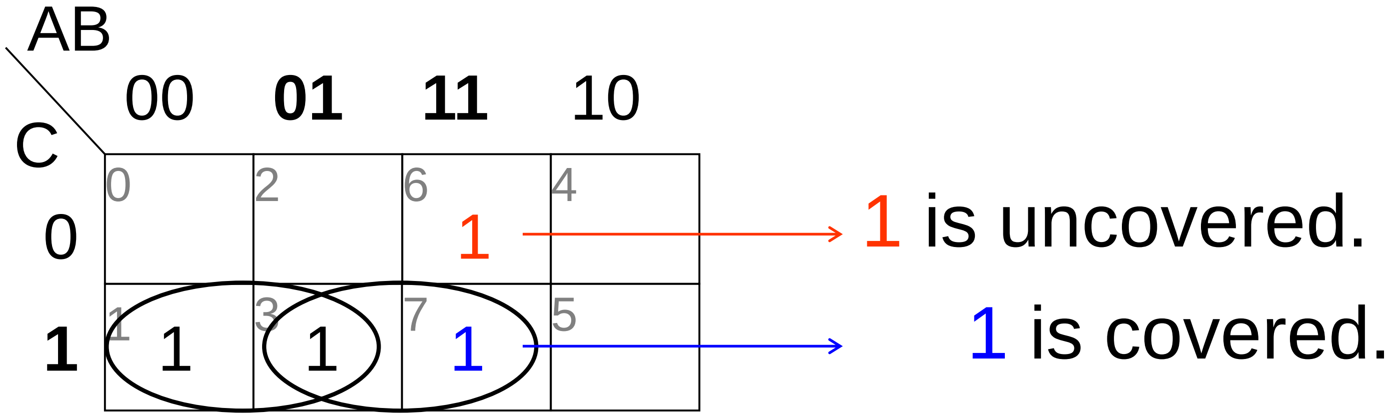
However, the complete SOP is not necessarily minimal, as we will see shortly.

Definition: A 1-cell is *covered* by a prime implicant if the 1-cell is a member of that prime implicant.



Definition:

A 1-cell is *uncovered* if it is not circled in the K-map



$Y = \sum_{A, B, C, D} (7, 9, 13, 15)$ (p. 15)

Prime implicants ?

$Y = \mathbf{A} . B' . \mathbf{C'} . \mathbf{D} + \mathbf{A} . B . \mathbf{C'} . \mathbf{D} + A . \mathbf{B} . \mathbf{C} . \mathbf{D} + A' . \mathbf{B} . \mathbf{C} . \mathbf{D}$

Canonical SOP

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

Prime implicant I: A.C'.D

Prime implicant II B.C.D

Prime implicant III: A.B.D

Apply T10-L

Use algebraic techniques

$Y = \mathbf{A} . \mathbf{C'} . \mathbf{D} + \mathbf{B} . \mathbf{C} . \mathbf{D}$

- These two p-terms correspond to PIs 1 and 2.
- Why PI no.3 is missing?
  - 1-cells 13 & 15 are also covered by PI no.1 & PI no.2
  - Thus, PI no.3 is **redundant** and **has to be removed** from the complete SOP to reach a minimal SOP.

Complete SOP ?

$Y = \mathbf{A} . \mathbf{C'} . \mathbf{D} + \mathbf{B} . \mathbf{C} . \mathbf{D} + A . B . D$

Complete SOP is NOT *necessarily* minimal.

Phase 1 of logic minimization using K-maps:

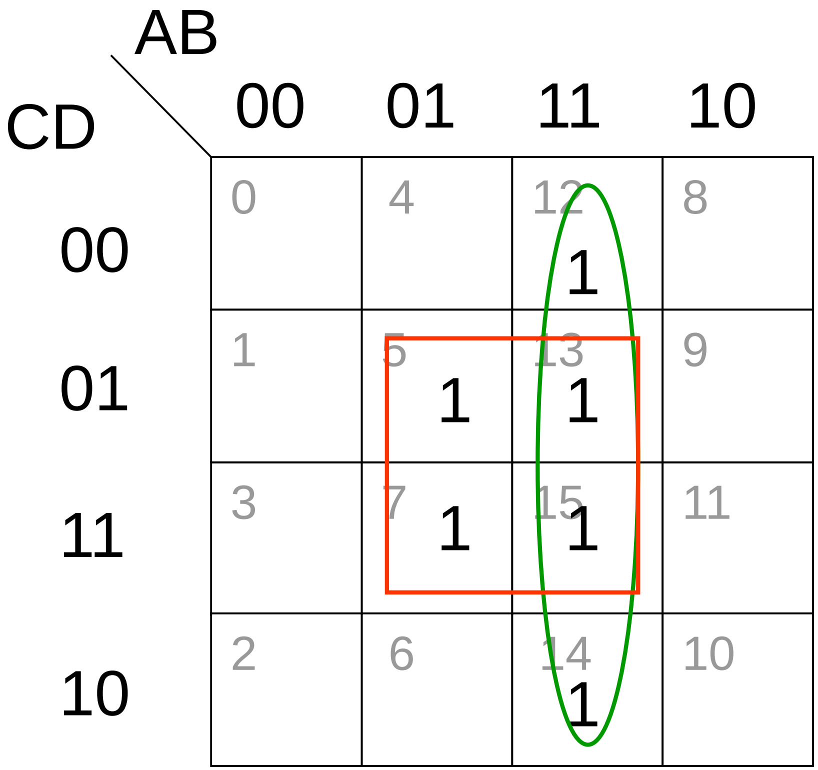
- Obtain all prime implicants of function under consideration

Now: Phase 2 of logic minimization using K-maps:(p. 15)

- Start with the set of all prime implicants
- Obtain a **minimum-size** subset of that set so that
  - each individual 1-cell will be covered by at least one prime implicant in the subset
- In case of multiple minimum-size subsets choose the one with the minimum number of literals

Example 13. Obtain a minimal SOP for

Y = ∑ A, B, C, D (5, 7, 12, 13, 14, 15).



- **There is** **no redundant prime implicant** **in the complete SOP because if either of these two prime implicant is left out, then two 1-cells will be left uncovered.**
- **Thus, the** **complete SOP is the minimal SOP** **as well.**

PIs: A . B, B . D

Y(A, B, C, D) = A . B + B . D (complete SOP is also minimal)

Example 14. (p. 16) Obtain a minimal SOP for

$Y = \sum A, B, C, D\ (1, 4, 5, 6, 7, 8, 10, 14)$.

Try to solve this

Example 14. (p. 16) Obtain a minimal SOP for

$Y = \sum A, B, C, D\ (1, 4, 5, 6, 7, 8, 10, 14)$.

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

PIs:

$A' . B$

$A' . C' . D$

$A . B' . D'$

$B . C . D'$

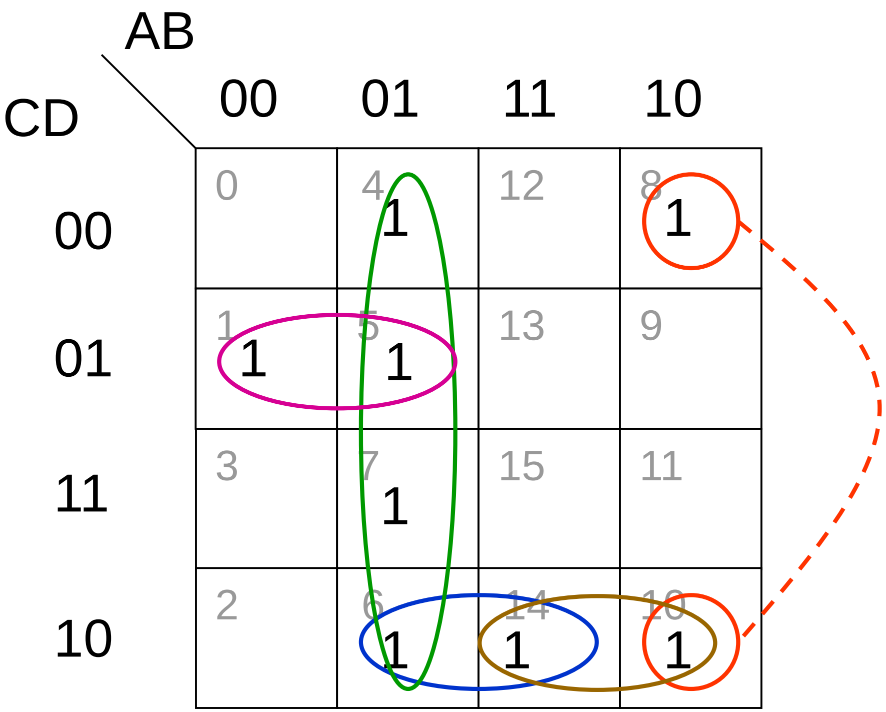
$A . C . D'$

Complete SOP: $Y(A, B, C, D) =$

$A' . B + A' . C' . D + A . B' . D' + B . C . D' + A . C . D'$

Example 14. (p. 16) Obtain a minimal SOP for

Y = ∑ A, B, C, D (1, 4, 5, 6, 7, 8, 10, 14).



PIs:

A’ . B

A’ . C’ . D

A . B’ . D’

B . C . D’

~~A . C . D’~~

Complete SOP: Y(A, B, C, D) =

A’ . B + A’ . C’ . D + A . B’ . D’ + B . C . D’ + ~~A . C . D’~~

Example 14. (p. 16) Obtain a minimal SOP for

$Y = \sum A, B, C, D\ (1, 4, 5, 6, 7, 8, 10, 14)$.

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

PIs:

A’ . B

A’ . C’ . D

A . B’ . D’

B . C . D’

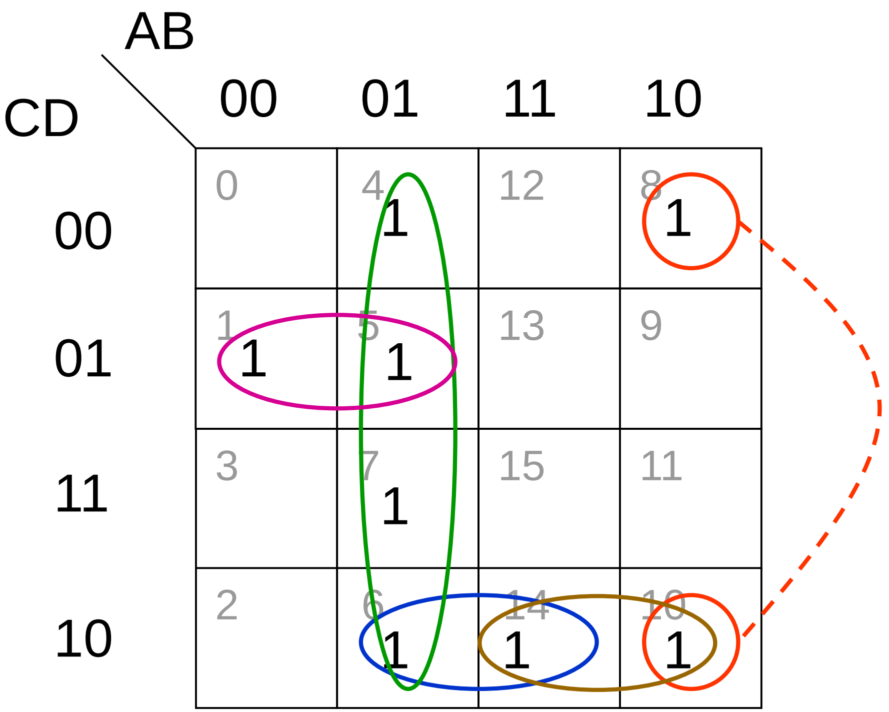
~~A . C . D’~~

Minimal SOP: Y(A, B, C, D) =

A’ . B + A’ . C’ . D + A . B’ . D’ + B . C . D’

Example 14. (p. 16) Obtain a minimal SOP for

Y = ∑ A, B, C, D (1, 4, 5, 6, 7, 8, 10, 14).



PIs:

A’ . B

A’ . C’ . D

A . B’ . D’

~~B . C . D’~~

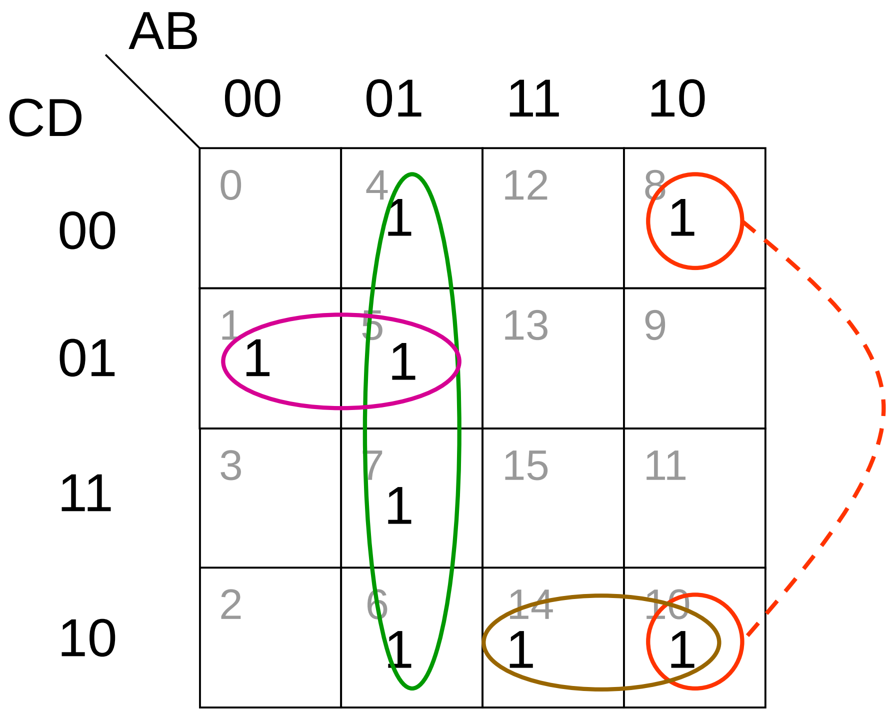
A . C . D’

Complete SOP: Y(A, B, C, D) =

A’ . B + A’ . C’ . D + A . B’ . D’ + ~~B . C . D’~~ + A . C . D’

Example 14. (p. 16) Obtain a minimal SOP for

$Y = \sum A, B, C, D\ (1, 4, 5, 6, 7, 8, 10, 14)$.



PIs:

$A' . B$

$A' . C' . D$

$A . B' . D'$

$\cancel{B . C . D'}$

$A . C . D'$

Two minimal SOPs

Minimal SOP: $Y(A, B, C, D) =$

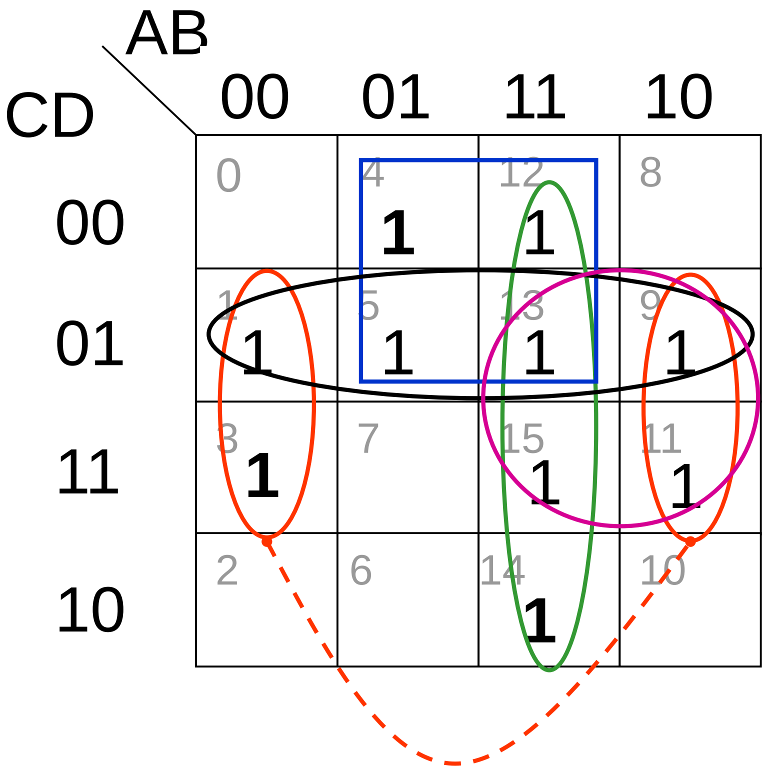
$A' . B + A' . C' . D + A . B' . D' + A . C . D'$

Example 15. (p. 16) Obtain a minimal SOP for

$Y(A, B, C, D) = \sum (1, 3, 4, 5, 9, 11, 12, 13, 14, 15)$

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

Try to solve this



Phase I.

all prime implicants

B . C’

B’ . D

A . B

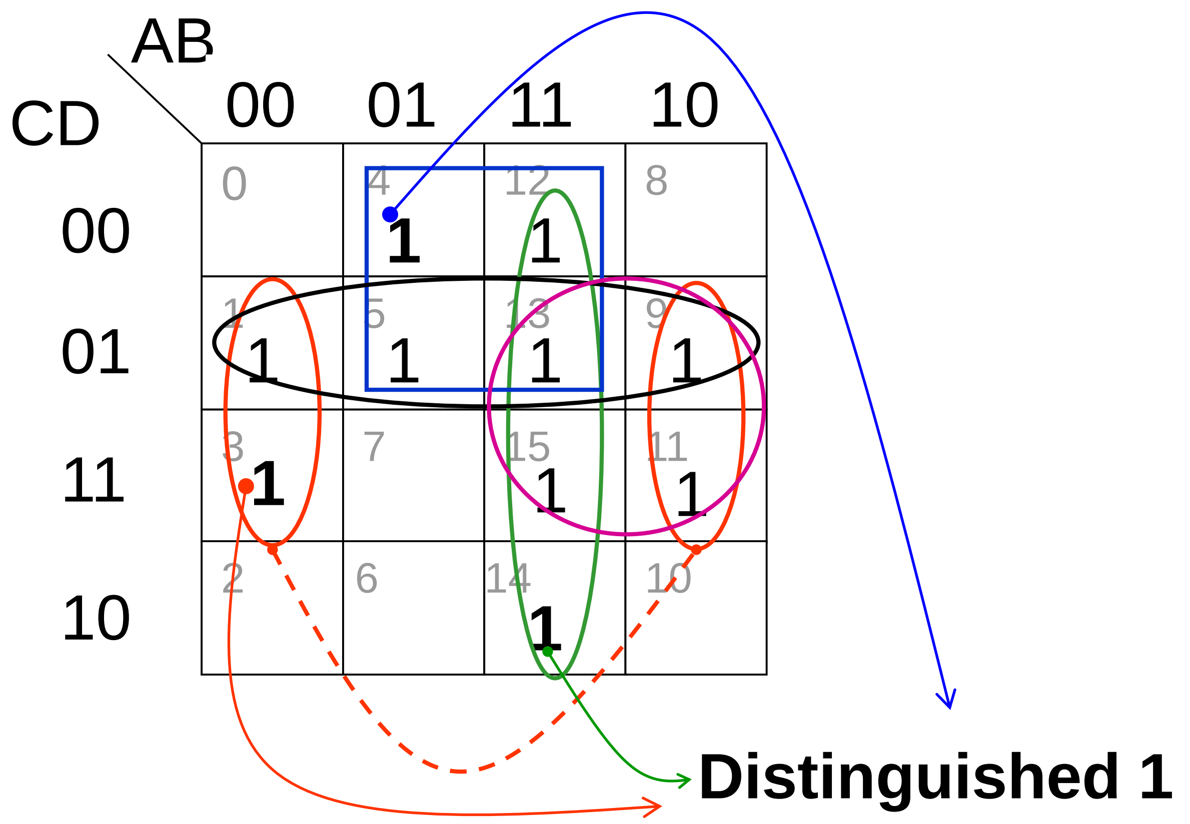
C’ . D

A . D

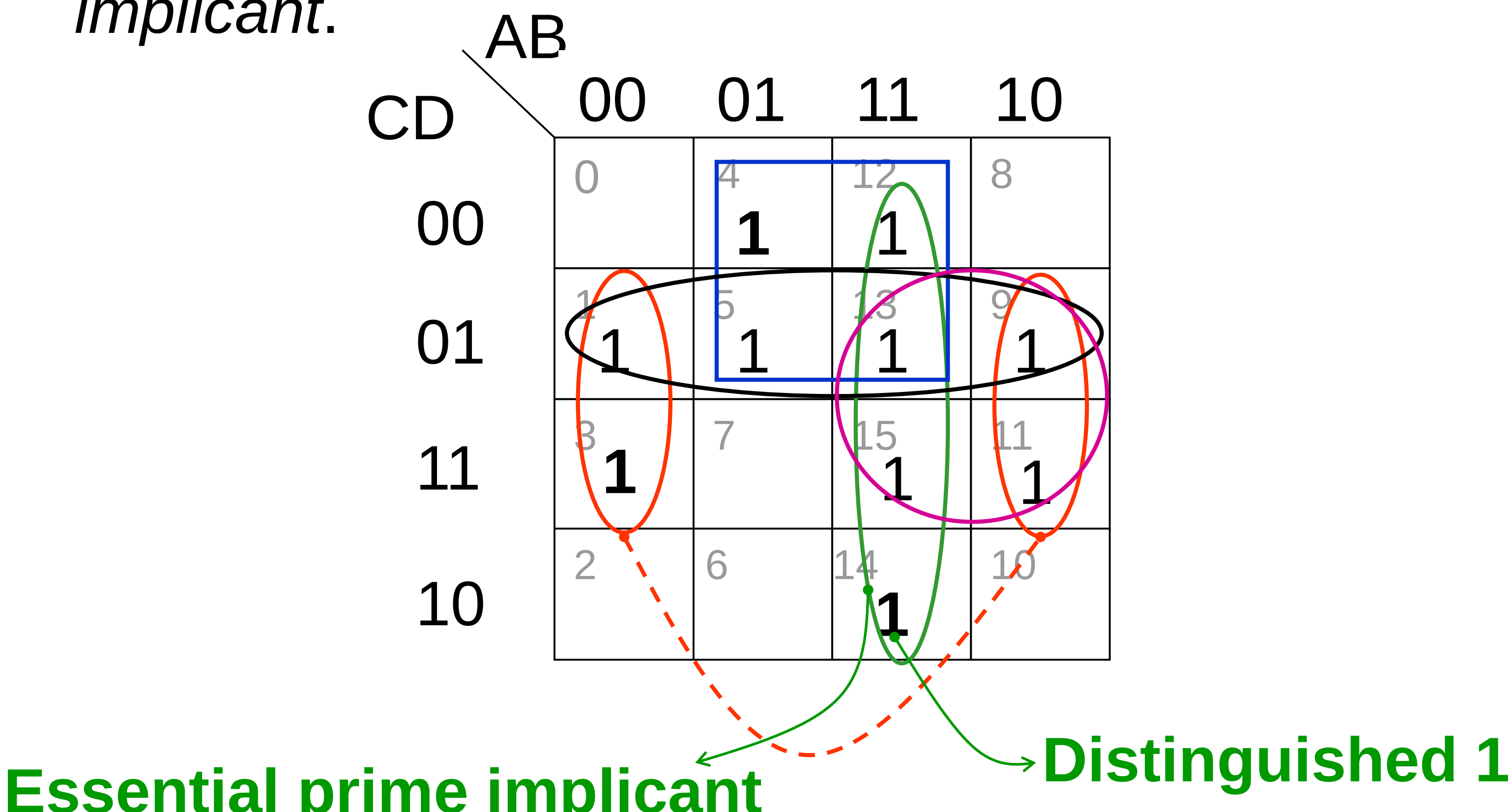
However, it is not that easy to apply **phase II**.

Let’s lower the number of valid choices that we have.

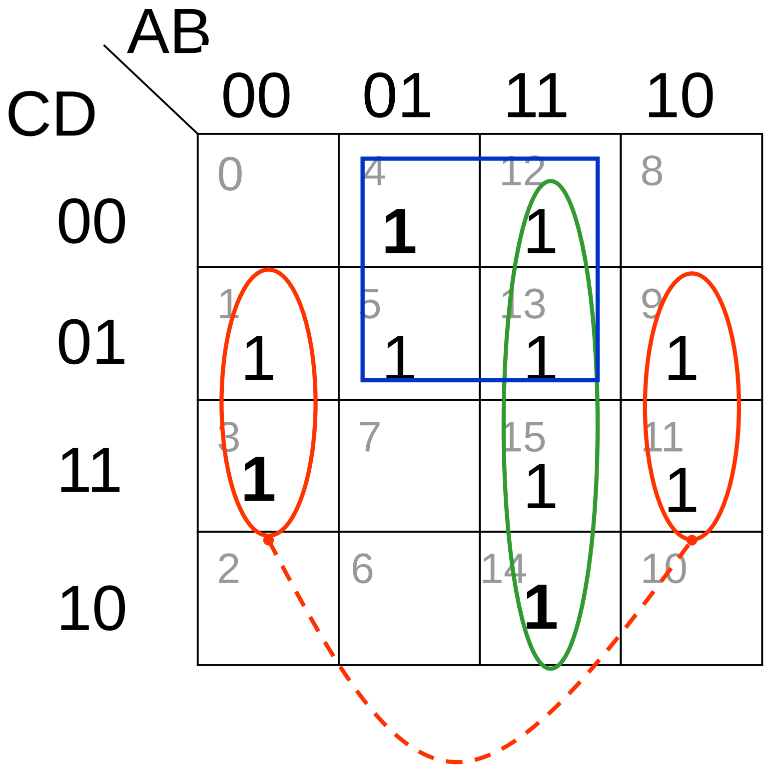
Definition: A *distinguished* 1-cell is a 1-cell that can be covered by only one prime implicant.



Definition: A prime implicant that covers one or more distinguished 1-cells is called an *essential prime implicant*.



Conclusion: <u>All the essential prime implicants must be included in the minimal SOP</u> (i.e., they are necessary, but may or may not be sufficient.)



Essential PIs:

{B . C', B' . D, A . B}

In this example essential PIs cover all the 1-cells

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

We do not need any non-essential PI.

What about C’ . D and A . D ?

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

Non-essential PI: C’ . D , A . D

Example:

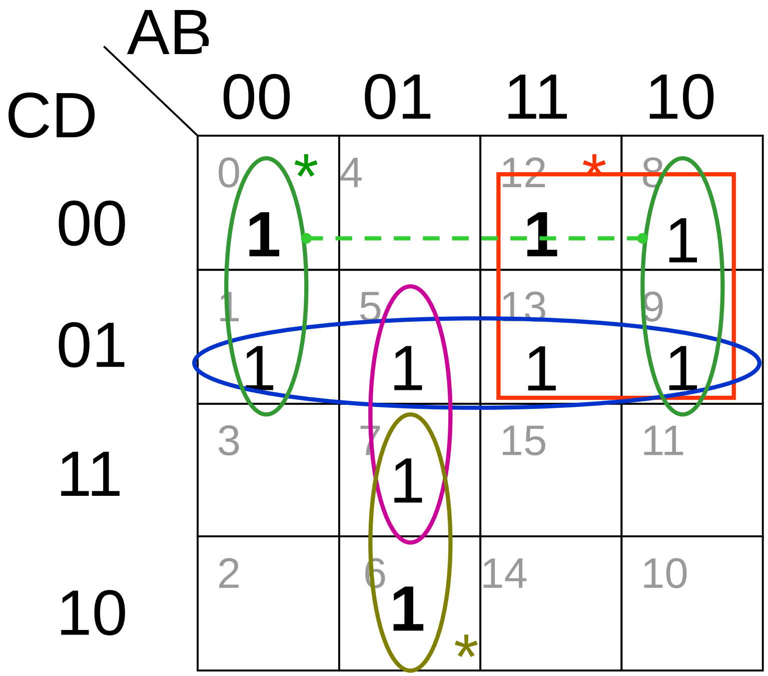
Obtain a minimal SOP for Y (A, B, C, D) = $\sum(0,1,5,6,7,8,9,12,13)$

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

Try to solve this

Example

Obtain a minimal SOP for Y (A, B, C, D) = ∑ (0,1,5,6,7,8,9,12,13)



Prime Implicants:

B’ . C’,

A . C’,

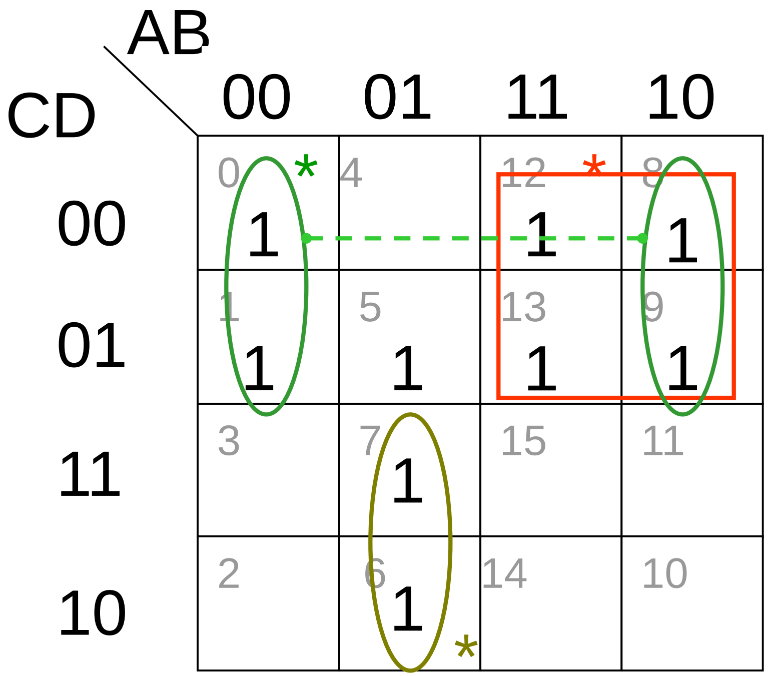
A’ . B . C,

C’ . D,

A’ . B . D

* marks distinguished 1-cells.

Example (Cont'd)



Essential PIs:

B' . C',

A . C',

A' . B . C

1-cell 5 is not covered by essential PIs. But it is covered by either of non-essential PIs.

Example (Cont'd)

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

Non-essential PIs:

$C' . D$,

$A' . B . D$

Which one would you choose?

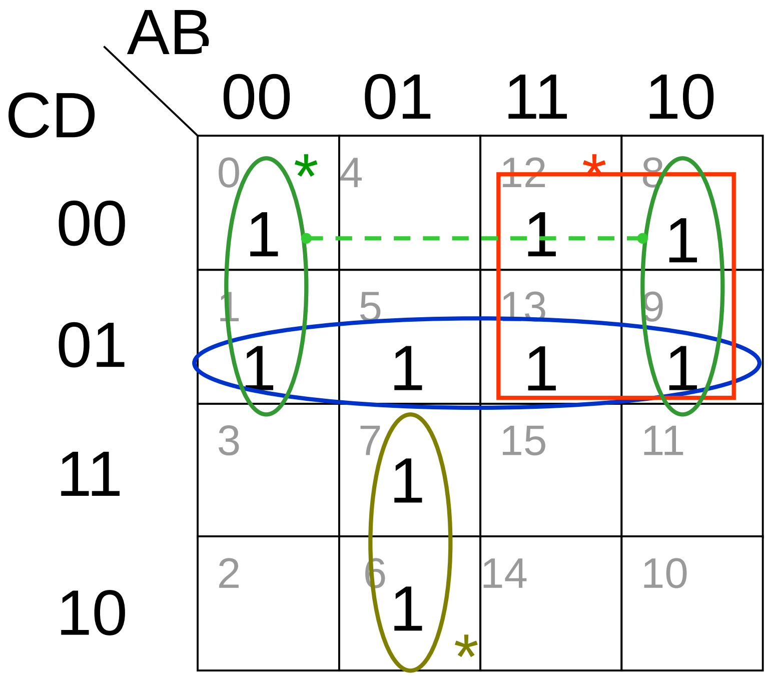
or

Answer:

The largest PI

(fewest literals)

Example (Cont'd)



Minimal SOP

Y =

B' . C' +

A . C' +

A' . B . C +

C' . D

Minimal POS

Combine 0s instead of 1s

We may reuse all procedures, rules and terms defined in the previous sections but need to replace

- minterms with maxterms,
- p-terms with sum-terms (or s-terms for short),
- on-sets with off-sets
- 1s with 0s.
- Therefore, **Rules 3 & 4** now become **3’ & 4’** in page 20 of text book

Example 19. Obtain a minimal POS:

SOP or POS, which one would you prefer?

$Y(A, B, C) = \prod(0, 1, 3, 5, 7) = \sum(2, 4, 6)$

Try to solve this

Example 19. Obtain a minimal POS:

SOP or POS, which one would you prefer?

Y(A, B, C) = ∏ (0, 1, 3, 5, 7) = ∑(2, 4, 6)

In this example POS is more cost-effective.

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

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

one 2-input OR &

one 2-input AND

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

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

one 2-input OR &

two 2-input AND

Example 21. Obtain a minimal SOP and a minimal POS

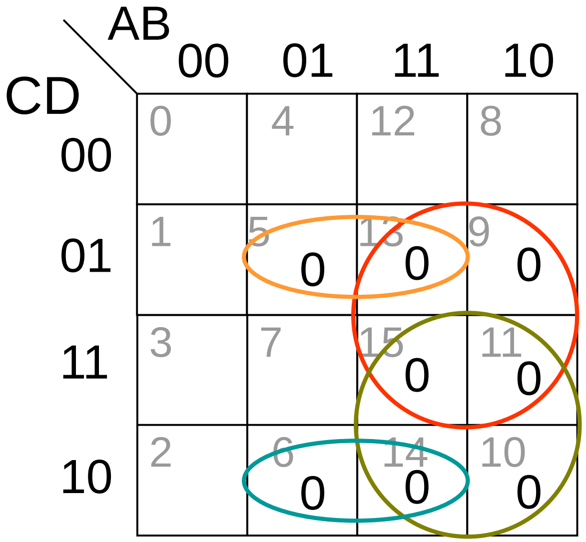
Which one, minimal SOP or minimal POS, would you prefer?

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

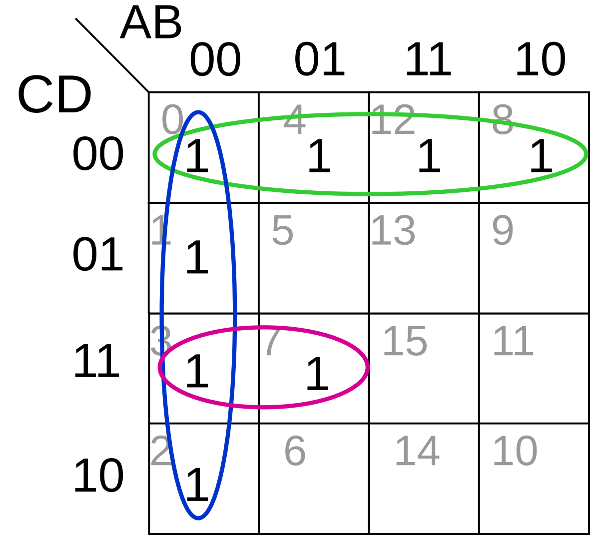
CD \ AB	00	01	11	10
00	(0) 1	(4) 1	(12) 1	(8) 1
01	(1) 1	(5)	(13)	(9)
11	(3) 1	(7) 1	(15)	(11)
10	(2) 1	(6)	(14)	(10)

Example 21. Obtain a minimal SOP and a minimal POS

Which one, minimal SOP or minimal POS, would you prefer?



In this example SOP is more cost-effective.



Y(A, B, C) =

(A’ + D’) . (A’ + C’) . (B’ + C + D’) . (B’ + C’ + D)

two 3-input OR, two 2-input OR, one 4-input AND

Y(A, B, C) =

C’ . D’ + A’ . B’ + A’ . C . D

one 3-input AND, two 2 input AND, 1 3-input OR

Which one to choose: minimal SOP or minimal POS?

The 2 choices to realize every function (minimal SOP and minimal POS) can be compared from 2 different point of views:

1- Number of transistors (in CMOS technology)

2- Number of terms

These two constraints may or may not be satisfied together:

In Example 19 the two choices have the same number of terms, but the minimal POS needs fewer transistors

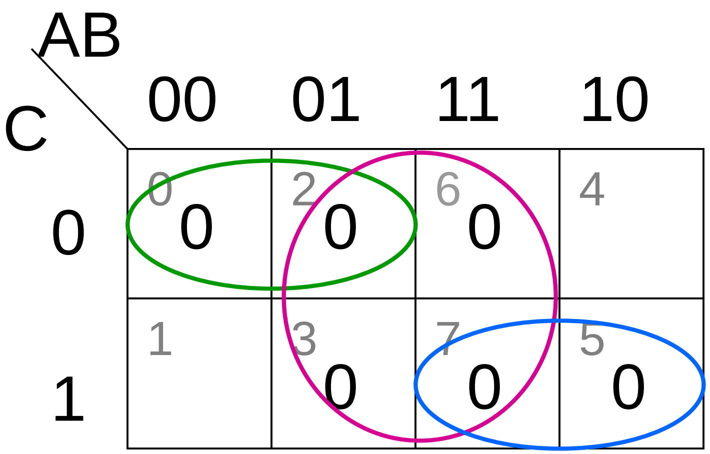
In Example 21 the minimal SOP has fewer terms and needs fewer transistors as well.

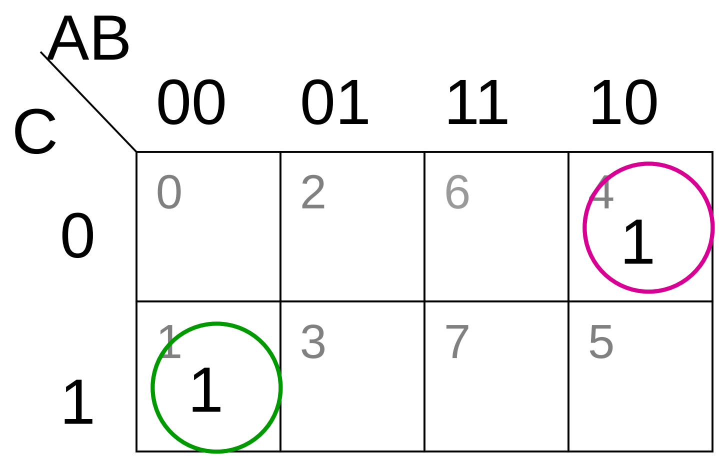
Now consider Y = ∑(1, 4)

Y = A’B’C + AB’C’   minimal SOP

Y = B’ (A + C) (A’ + C’)   minimal POS

In this example the minimal SOP has fewer terms but needs more transistors.





Incompletely Specified Circuits

“Don’t cares” in output columns

Example 22: The Dean’s List **…**

Senior students with GPAs above 90%

Junior students with GPAs above 95%

There are four variables in this problem:

G95 = 1: GPA above 95%

G90 = 1: GPA above 90%

S =1: Senior student

J = 1: Junior student

Output Y is pulled up if the student is on the list; otherwise the student is not on the list.

Row	G90 G95 J S	Y	Row	G90 G95 J S	Y
				.	
				.	

“don’t care”, x, in output: an impossible input combination

Row	G90 G95 J S	Y	Row	G90 G95 J S	Y
0	0 0 0 0	0	8	1 0 0 0	0
1	0 0 0 1	0	9	1 0 0 1	1
2	0 0 1 0	0	10	1 0 1 0	0
3	0 0 1 1	x	11	1 0 1 1	x
4	0 1 0 0	x	12	1 1 0 0	0
5	0 1 0 1	x	13	1 1 0 1	1
6	0 1 1 0	x	14	1 1 1 0	1
7	0 1 1 1	x	15	1 1 1 1	x

“don’t care”, x, in output: an impossible input combination

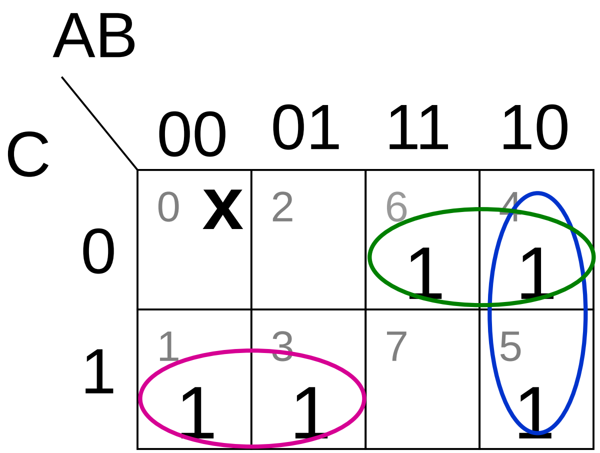
For minimization purposes:

- **We can replace each don’t care with a 0 or 1, whichever results in a more simplified expression**

Example 23. (p. 23) Obtain a minimal SOP for

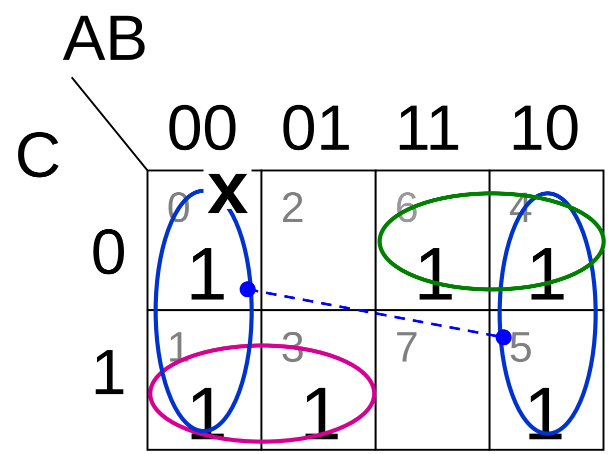
Y(A, B, C) = ∑(1, 3, 4, 5, 6) + d(0)

Assign 0 to don’t care



Y = A . B’ + A’ . C + A . C’

Assign 1 to don’t care



Y = B’ + A’ . C + A . C’

2nd assignment is more cost-effective

Example 23. (Cont'd) Obtain a minimal POS for

$Y(A, B, C) = \prod (2, 7).D(0)$

Assign 0 to don't care

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

Assign 1 to don't care

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

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

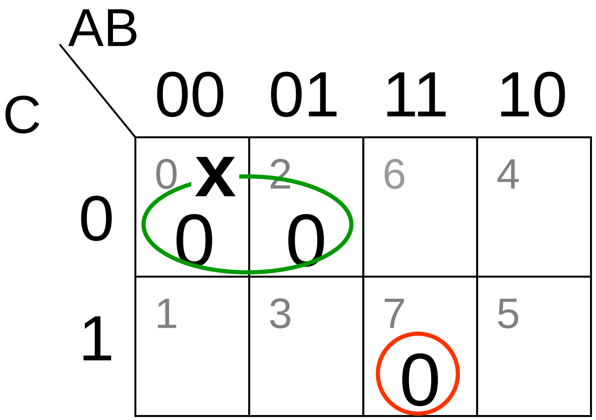
$Y = (A + B' + C) . (A' + B' + C')$

1st assignment is more cost-effective

Example 23. (Cont'd)

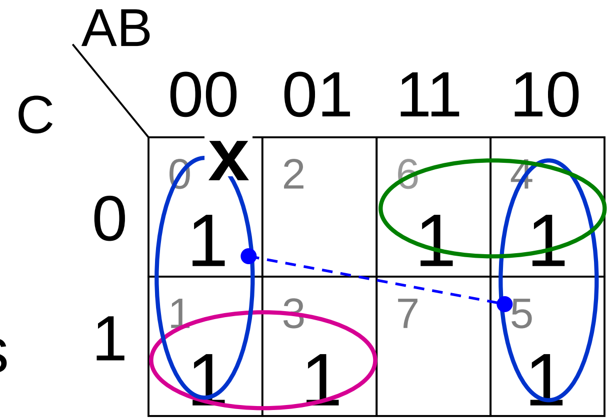
Determine the most cost-effective design.

Assign 0 to don't care



Same # of transistors

Assign 1 to don't care



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

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

2 2-input,

1 3-input

2 2-input,

1 3-input

On the other hand,

POS has fewer terms