Modern Digital System Design

ECE 2372 / Fall 2018 / Lecture 04

Texas Tech University
Dr. Tooraj Nikoubin

Multi- Level Gate Circuits and Karnaugh Maps

Outline

- SOP & POS Implementation
- Boolean function: Representation
– SOM (Sum of Minterms), Canonical form
– SOP (Sum of Product)

- Karnough map simplification
– 2, 3 variable KMap
– 4 variable karnaugh map

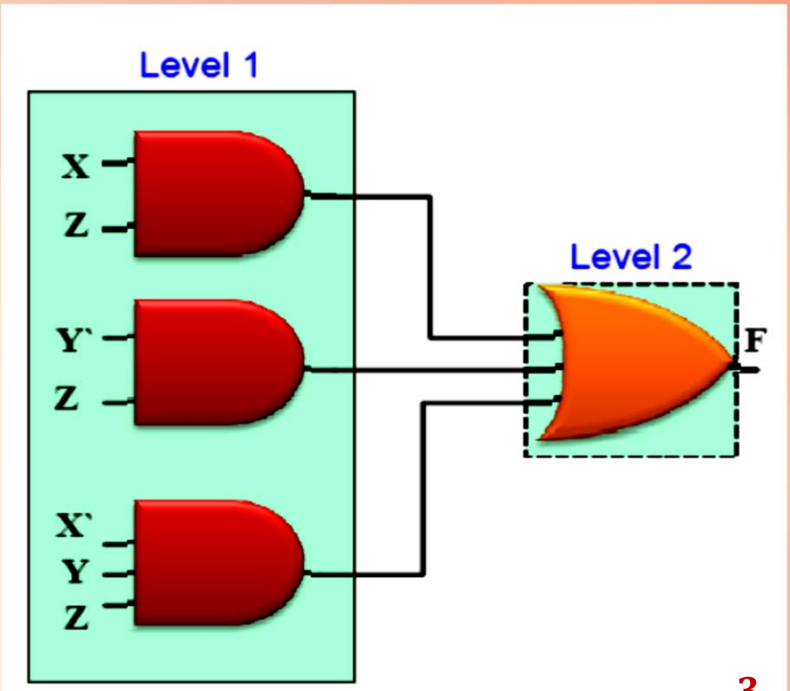
Implementation of SOP

$$F(X,Y,Z) = XZ+Y'Z+X'YZ$$

- Any SOP expression can be implemented using 2- levels of gates
- The 1st level consists of AND gates,

and the 2nd level consists of a single OR gate

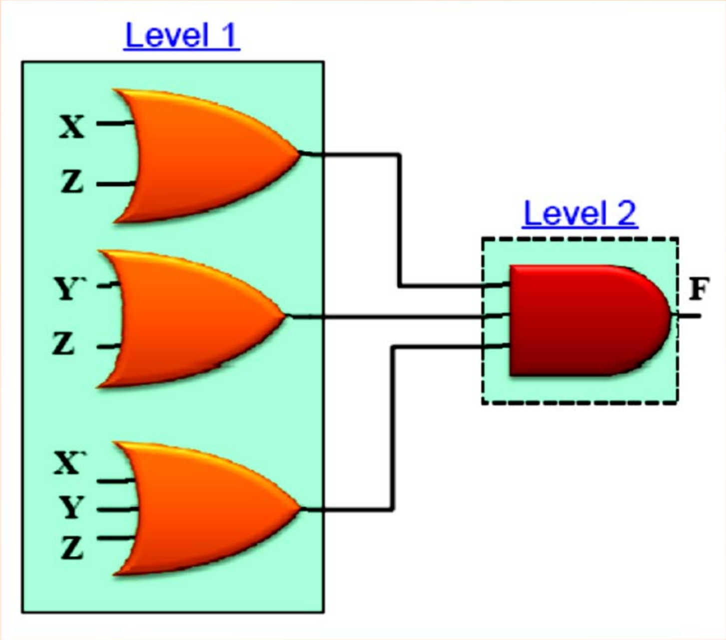
- Also called 2-level Circuit



Implementation of POS

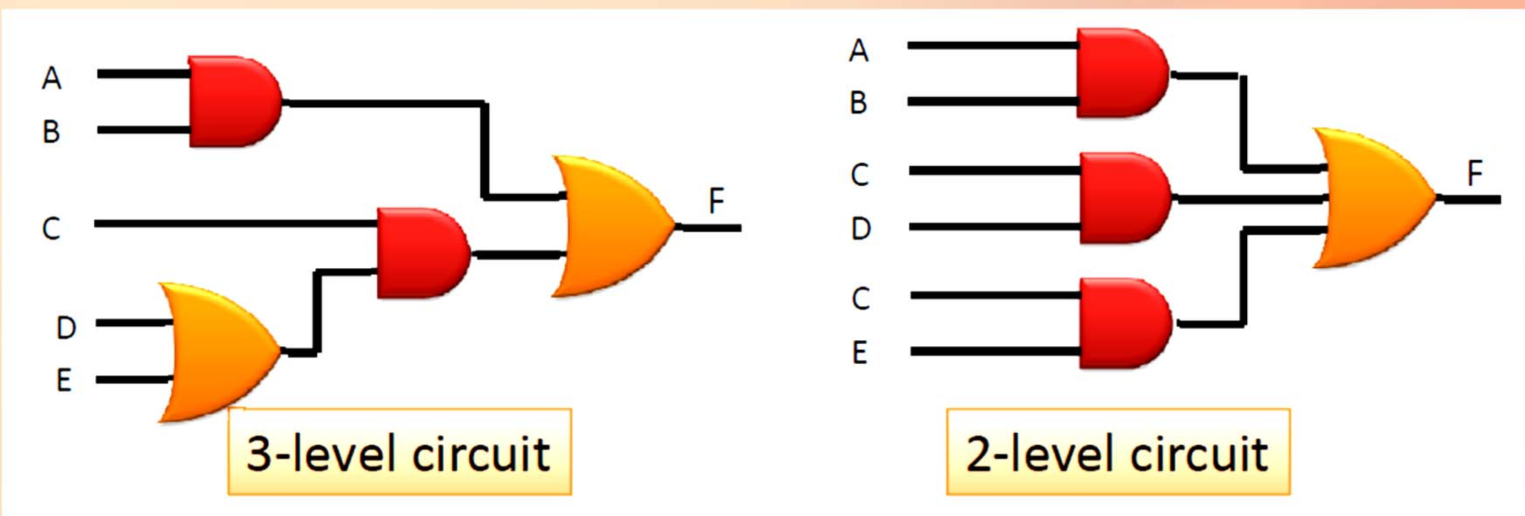
$$F(X,Y,Z) = (X+Z)(Y'+Z)(X'+Y+Z)$$

- Any POS expression can be implemented using 2- levels of gates
- The 1st level consists of OR gates, and the 2nd level consists of a single AND gate
- Also called 2-level Circuit



Implementation of SOP

- Consider F = AB + C(D+E)
- This expression is NOT in the sum-of-products form
- Use the identities/algebraic manipulation to convert to a standard form (sum of products), as in F = AB + CD + CE
- Logic Diagrams:



Canonical Forms

- It is useful to specify Boolean functions in a form that:

–Allows comparison for equality.

– Has a correspondence to the truth tables

- Canonical Forms in common usage:

– Sum of Minterms (SOM)

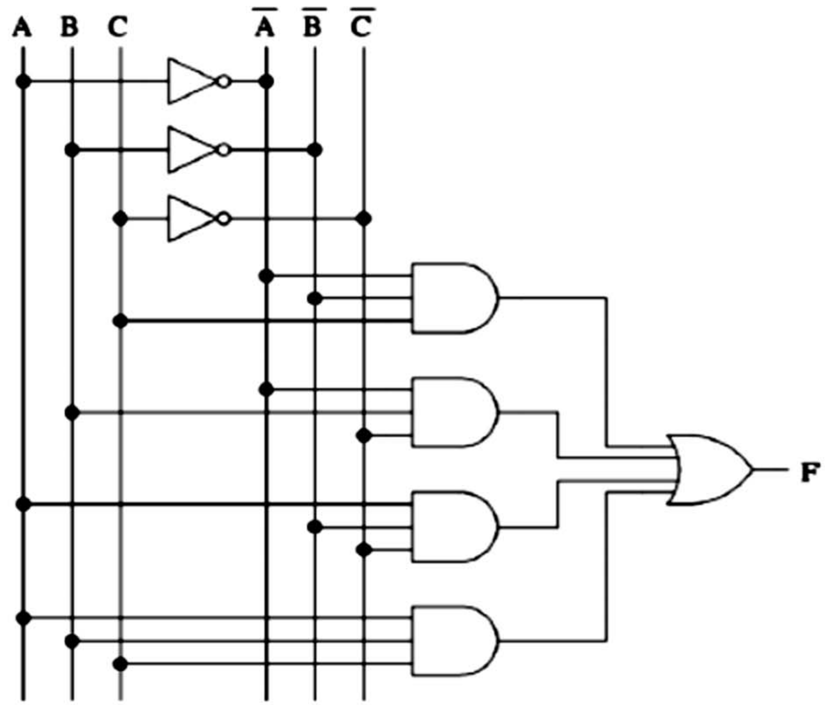
– Product of Maxterms (POM)

Brute Force Method of Implementation

3-input even-parity function

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

- **SOM implementation**

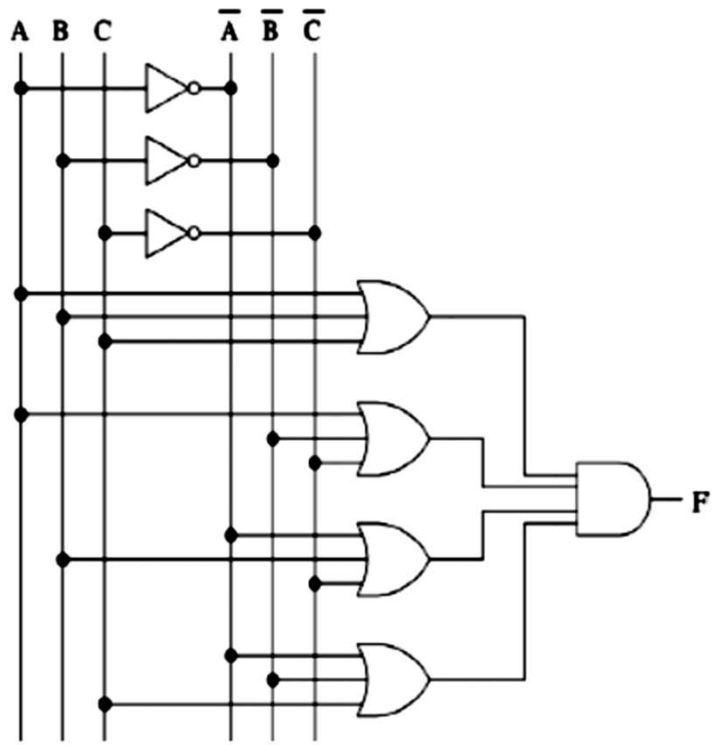


Brute Force Method of Implementation

3-input even-parity function

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

- **POM** implementation



Simplification: Theorem method

$$E = \sum m(0,1,2,4,5)$$
$$= m_0 + m_1 + m_2 + m_4 + m_5$$
$$= m_5 + m_1 + m_4 + m_0 + m_2 + m_0$$
$$= \mathbf{XY'Z+X'Y'Z+XY'Z'+X'Y'Z'+X'YZ' +X'Z'Y'}$$
$$= (XY'+X'Y')(Z+Z') + X'YZ'+ X'Z'Y'$$
$$= Y'(X+X')(Z+Z') + X'Z'(Y+Y')$$
$$= \mathbf{Y' + X'Z'}$$

Simplified one: Require less
Gates and faster
2 Level

Both are in SOP
format : 2 level

Simplification of Boolean Functions

- An implementation of a Boolean Function requires the use of logic gates.
- A smaller number of gates, with each gate (other then Inverter) having less number of inputs, may reduce the cost of the implementation.
- There are 2 methods for simplification of Boolean functions.

Simplification of Boolean Functions: Two Methods

- **Algebraic method** by using Identities & Theorem

- **Graphical method** by using Karnaugh Map method

–The K-map method is easy and straightforward.
–A K-map for a function of n variables consists of 2^n cells, and,
–in every row and column, two adjacent cells should differ in the value of only one of the logic Variables 9 variables.

Karnaugh Map Advantages

- Minimization can be done more systematically
- Much simpler to find minimum solutions
- Easier to see what is happening (graphical)
- Almost always used instead of boolean minimization.

Gray Codes

- Gray code is a binary value encoding in which adjacent values only differ by one bit

2-bit Gray Code
00
01
11
10

Truth Table Adjacencies

F = A'

A	B	F
0	0	1
0	1	1
1	0	0
1	1	0

These are *adjacent in a gray code sense*

- they differ by 1 bit

We can apply XY + XY' = X

A'B' + A'B = A'(B'+B) = A'(1) = A'

F = B

A	B	F
0	0	0
0	1	1
1	0	0
1	1	1

Same idea: A'B + AB = B

Key idea:

Gray code adjacency allows use of simplification theorems

Problem:
Physical adjacency in truth table does not indicate gray code adjacency

Logical Equivalence

- **Proving logical equivalence of two circuits**
  - **Derive the logical expression for the output of each circuit**
  - **Show that these two expressions are equivalent**

» **Two ways:**

– You can use the truth table method

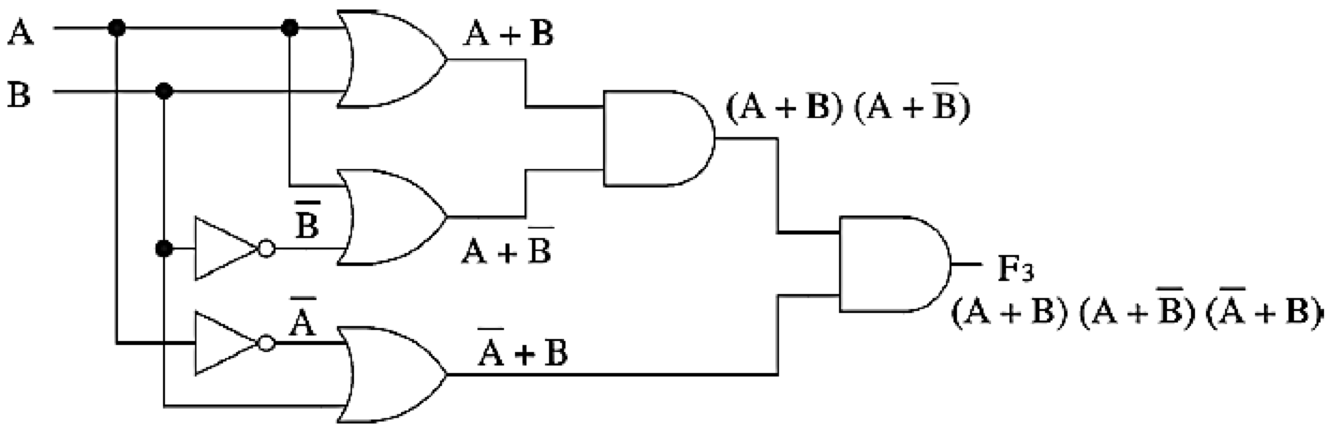
- For every combination of inputs, if both expressions
- yield the same output, they are equivalent Good for logical expressions with small number of variables

– You can also use algebraic manipulation

- Need Boolean identities

Logical Equivalence

- Derivation of logical expression from a circuit
  - Trace from the input to output
    - Write down intermediate logical expressions along the path



Logical Equivalence (cont'd)

- Proving logical equivalence: Truth table method

A	B	F1 = A B	F3 = $(A + B)(\overline{A} + B)(A + \overline{B})$
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

Karnaugh Map Method

• The truth table values are placed in the K map.

• **Adjacent K map square differ in only one variable both horizontally and vertically.**

• **The pattern from top to bottom and left to right must be in the form**

• A SOP expression can be obtained by ORing all squares that contain a 1.
A'B', A'B, AB, AB'
00, 01, 11, 10

Filling of Karnaugh Map

Why not: A'B', A'B, AB', AB

00, 01, 10, 11

Only two adjacent can be grouped

Group Reduce a variable: AB'+AB=A(B'+B)=A

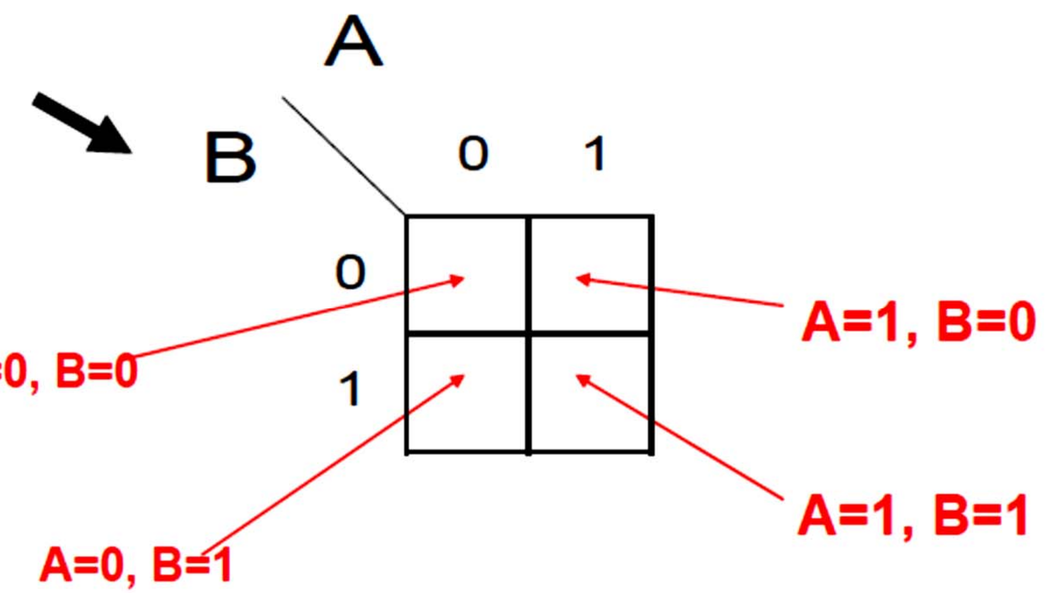
A'B', A'B, AB, AB'

00, 01, 11, 01

All 4 Adjacent can be grouped

2-Variable Karnaugh Map

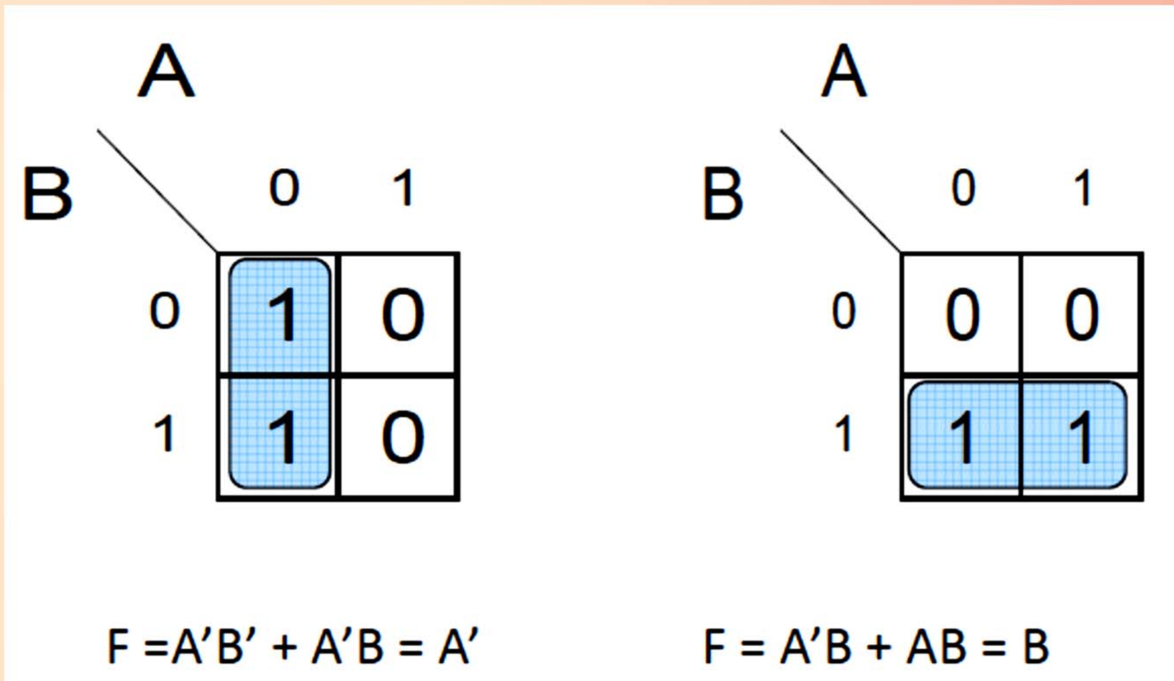
A	B	F
0	0	
0	1	
1	0	
1	1	



A different way to draw a truth table: by folding it

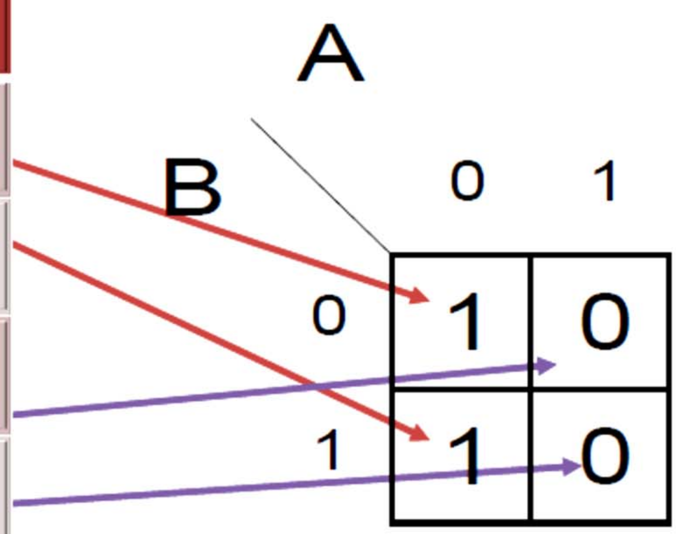
Karnaugh Map

- In a K-map, physical adjacency **does** imply gray code adjacency



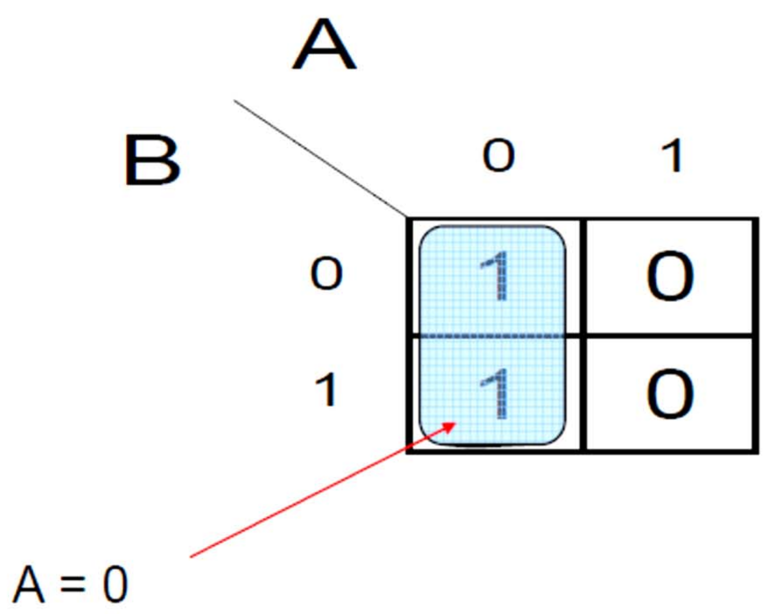
2-Variable Karnaugh Map

A	B	index	F
0	0	0	1
0	1	1	1
1	0	2	0
1	1	3	0



2-Variable K Map: Grouping

A	B	index	F
0	0	0	1
0	1	1	1
1	0	2	0
1	1	3	0



F = A'

2-Variable Karnaugh Map

A	B	index	F
0	0	0	1
0	1	1	1
1	0	2	0
1	1	3	0

B \ A	0	1
0	1	0
1	1	0

F = A’B’ + A’B = A’

Another Example

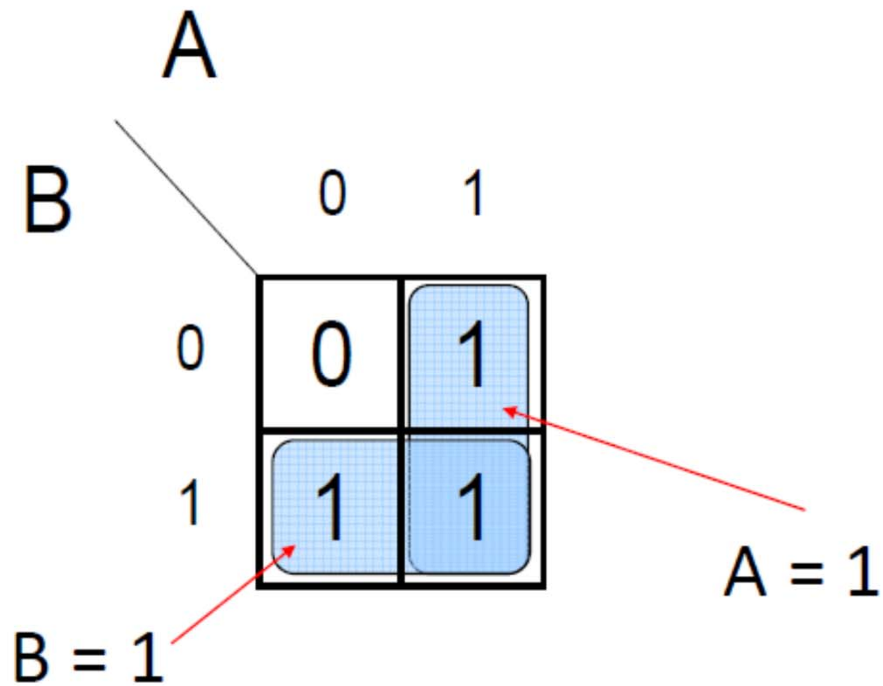
A	B	index	F
0	0	0	0
0	1	1	1
1	0	2	1
1	1	3	1

B \ A	0	1
0	0	1
1	1	1

$$
\begin{aligned}
F &= A'B + AB' + AB \\
&= (A'B + AB) + (AB' + AB) \\
&= A + B
\end{aligned}
$$

Another Example

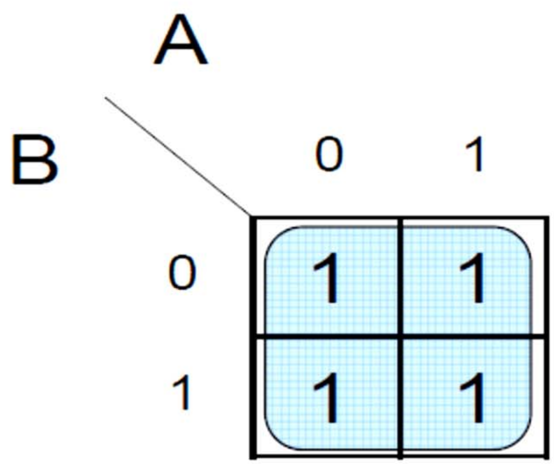
A	B	index	F
0	0	0	0
0	1	1	1
1	0	2	1
1	1	3	1



F = A + B

Another Example

A	B	index	F
0	0	0	1
0	1	1	1
1	0	2	1
1	1	3	1



F = 1

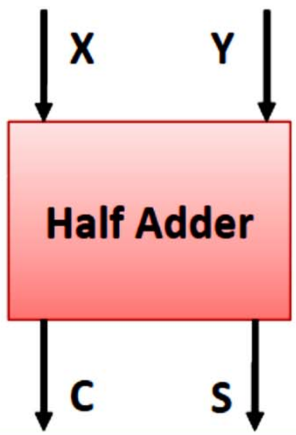
Groups of more than two 1's can be combined

HALF ADDER: One bit adder

X	Y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Y \ X	0	1
0	0	1
1	1	0

Y \ X	0	1
0	0	0
1	0	1



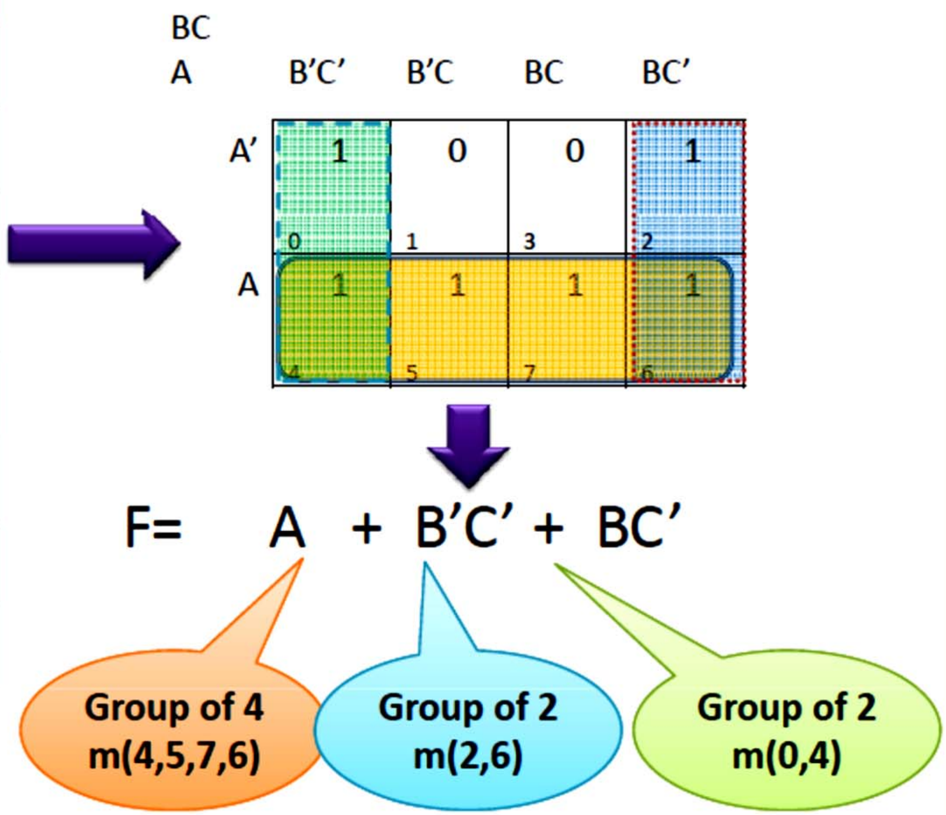
S = A'B + AB'
= A XOR B

C = AB

K-Map of three variable

A	B	C	index	F
0	0	0	0	1
0	0	1	1	0
0	1	0	2	1
0	1	1	3	0
1	0	0	4	1
1	0	1	5	1
1	1	0	6	1
1	1	1	7	1

$F = \sum m(0,2,4,5,6,7)$



3-Variable Karnaugh Map Showing Minterm Locations

Note the order of the B C variables:

0 0
0 1
1 1
1 0

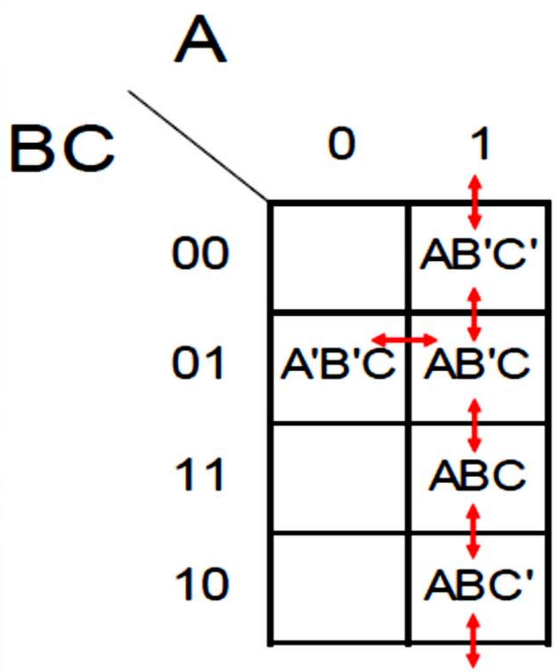
BC \ A	0	1
00	m0	m4
01	m1	m5
11	m3	m7
10	m2	m6

ABC = 101

ABC = 010

Adjacencies

Adjacent squares differ by exactly one variable



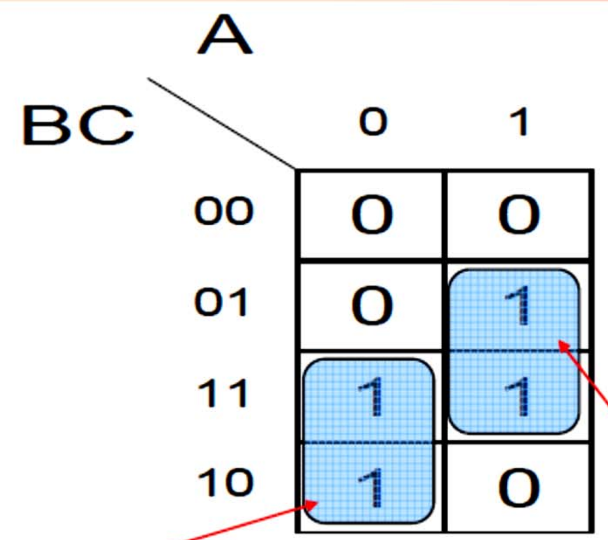
There is wrap-around:
top and bottom rows are adjacent

Truth Table to Karnaugh Map

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

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

Minimization Example



$A'BC+A'BC' = A'B$

$AB'C+ABC = AC$

$$F = A'B + AC$$

Minterm Expansion to K-Map

$$F = \sum m(1, 3, 4, 6)$$

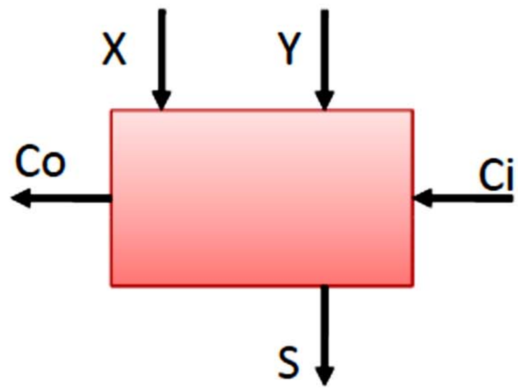
BC \ A	0	1
00	m0	m4
01	m1	m5
11	m3	m7
10	m2	m6

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

Minterms are the 1's, everything else is 0

Full Adder Example: Minterms

Ci	X	Y	index	S	Co
0	0	0	0	0	0
0	0	1	1	1	0
0	1	0	2	1	0
0	1	1	3	0	1
1	0	0	4	1	0
1	0	1	5	0	1
1	1	0	6	0	1
1	1	1	7	1	1



$$S = \sum m(1, 2, 4, 7)$$

$$Co = \sum m(3, 5, 6, 7)$$

Full Adder Output

$$S = \sum m(1, 2, 4, 7)$$

XY \ Ci	0	1
00	0	1
01	1	0
11	0	1
10	1	0

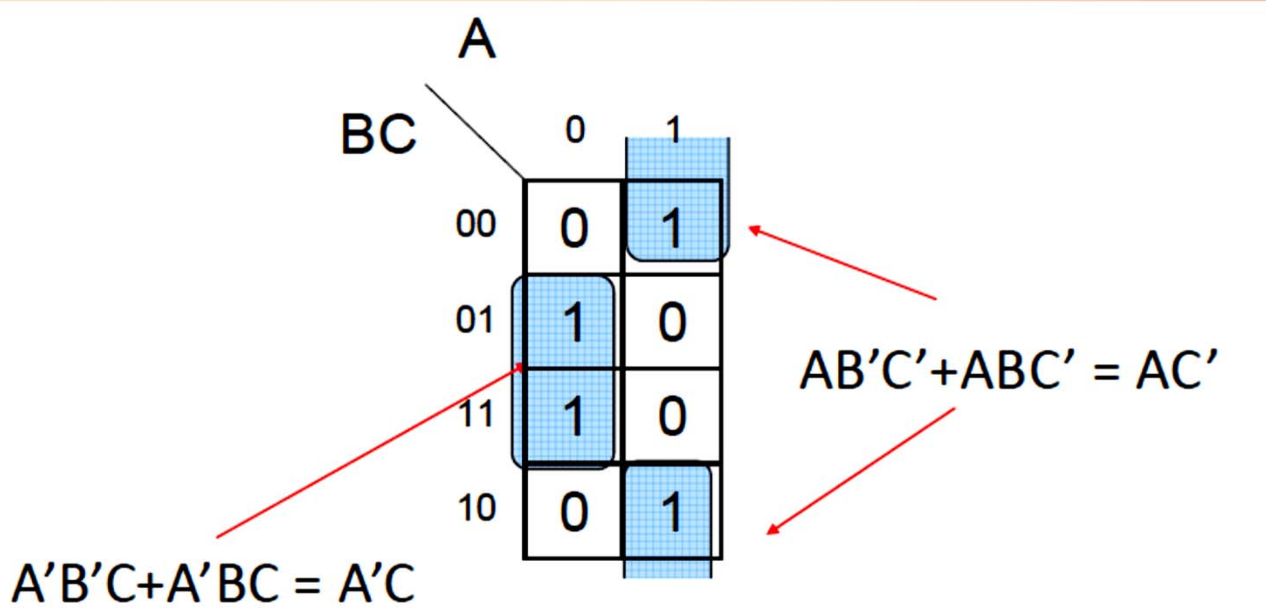
$$S = Ci'X'Y + Ci'XY' + CiX'Y' + CiXY$$

$$Co = \sum m(3, 5, 6, 7)$$

XY \ Ci	0	1
00	0	0
01	0	1
11	1	1
10	0	1

$$Co = XY + CiX + CiY$$

Another Example



$$F = A'C + AC' = A \oplus C$$

Maxterm Expansion to KMap

$$F = \prod M(0, 2, 5, 7)$$

BC \ A	0	1
00	M0	M4
01	M1	M5
11	M3	M7
10	M2	M6

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

Maxterms are the 0's, everything else is 1

Another Example

2^n 1's can be circled at a time

1, 2, 4, 8, … OK

3 not OK

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

A'B'C'+AB'C'+
A'B'C+AB'C = B'

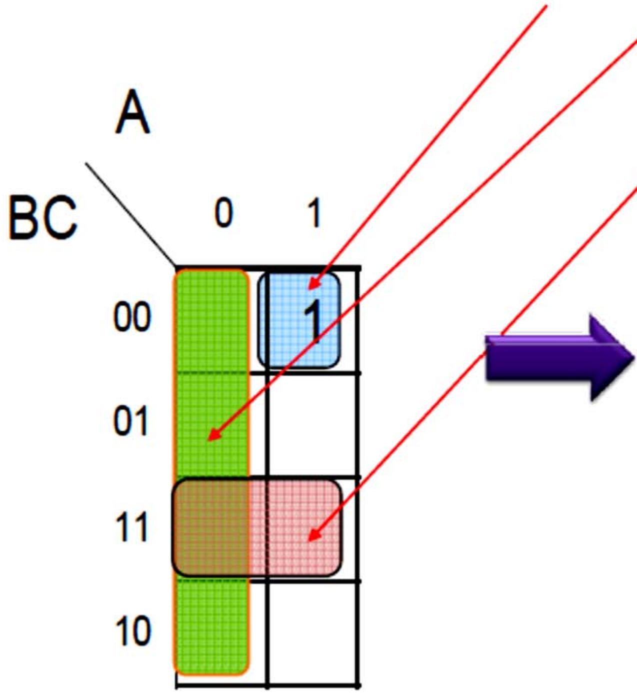
AB'C+ABC = AC

F = B' + AC

The larger the group of 1's the simpler the resulting product term

Boolean Algebra to Karnaugh Map

Plot: ab’c’ + a’ + bc



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

Remaining spaces are 0

Boolean Algebra to Karnaugh Map

Plot: ab'c' + bc + a'

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

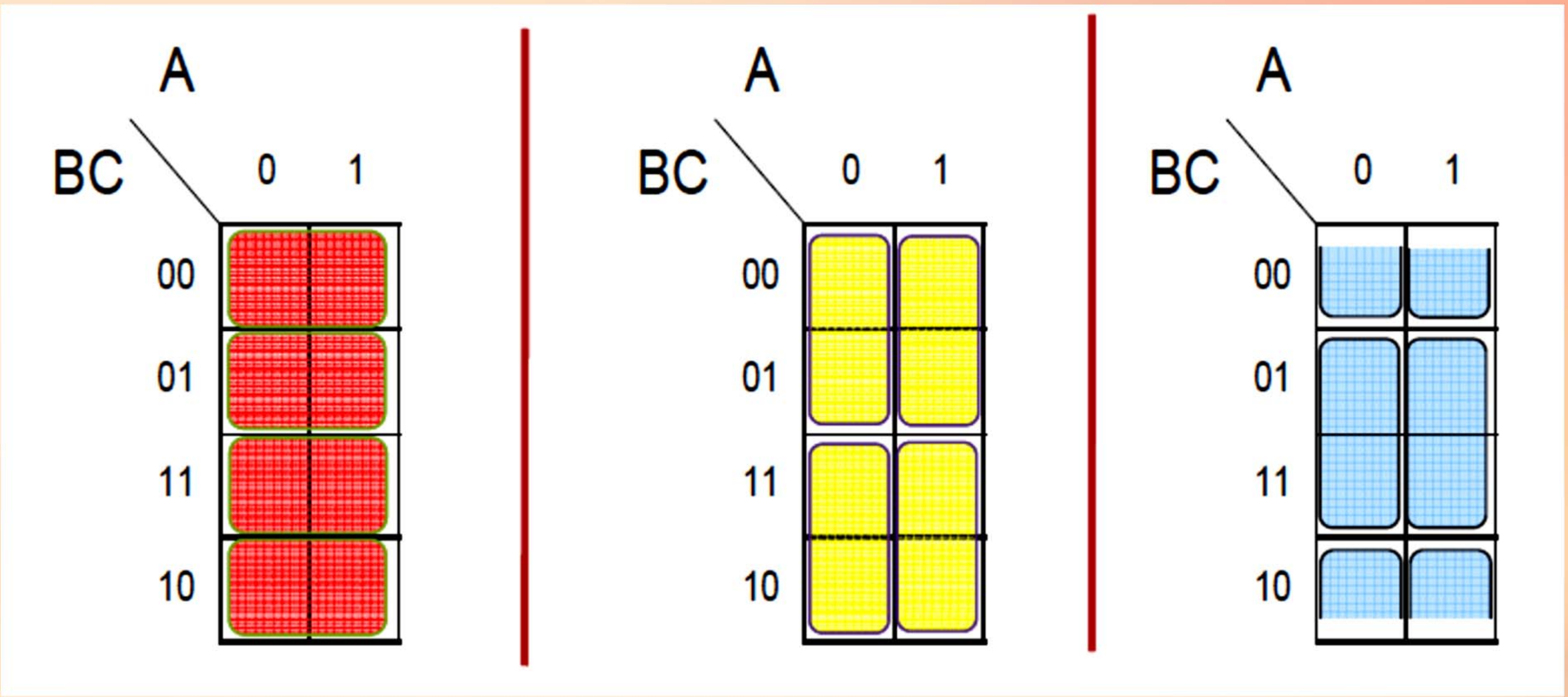
Now minimize

B'C' + BC + A'

This is a simpler equation than we started with.
Do you see how we obtained it?

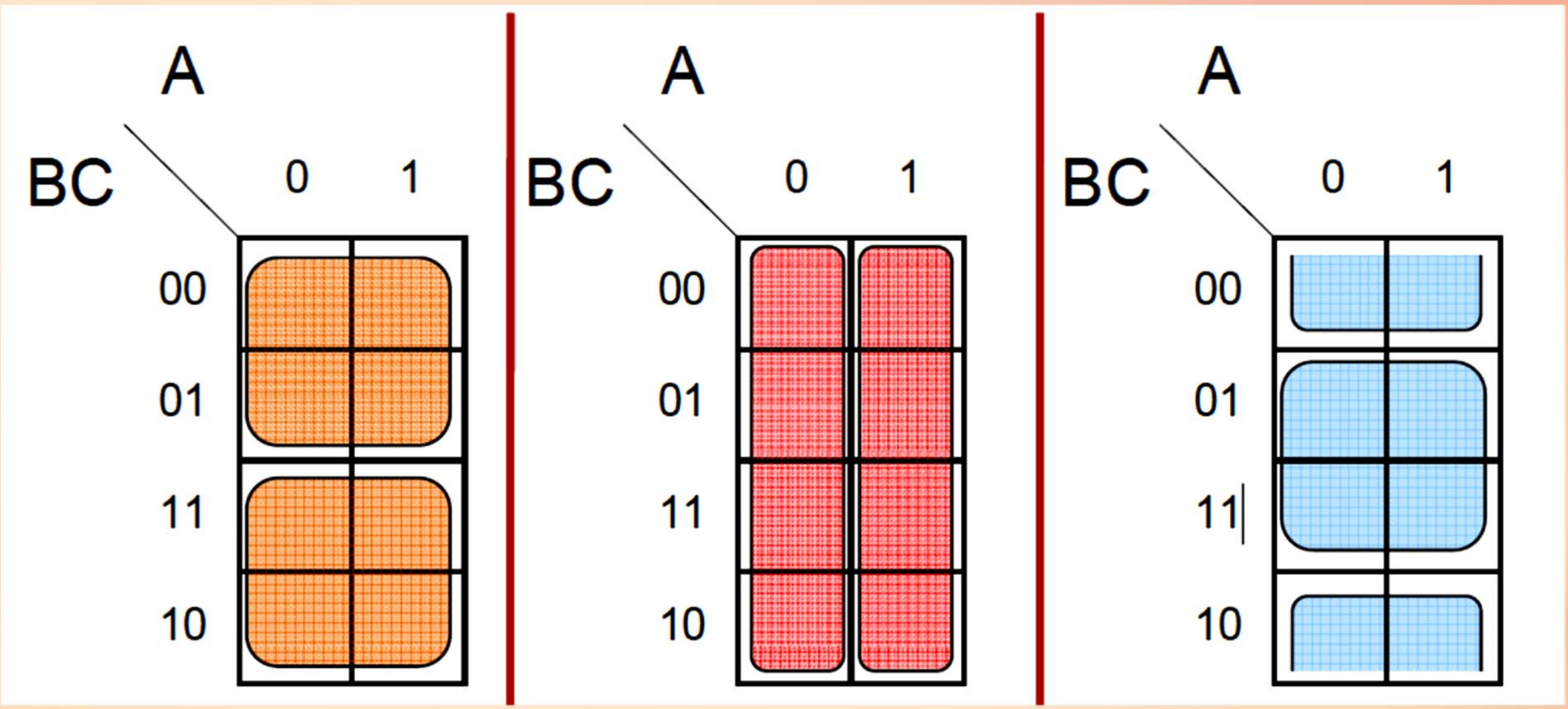
Mapping Sum of Product Terms

The 3-variable map has 12 possible groups of 2 spaces
These become terms with 2 literals



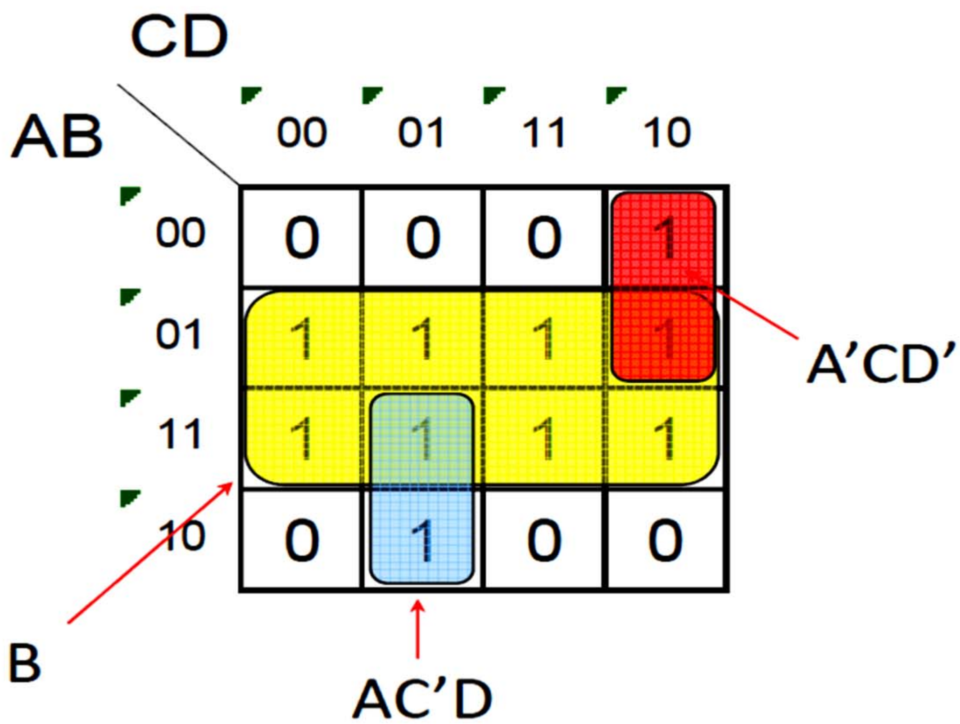
Mapping Sum of Product Terms

The 3-variable map has 6 possible groups of 4 spaces
These become terms with 1 literal



4-Variable Karnaugh Map

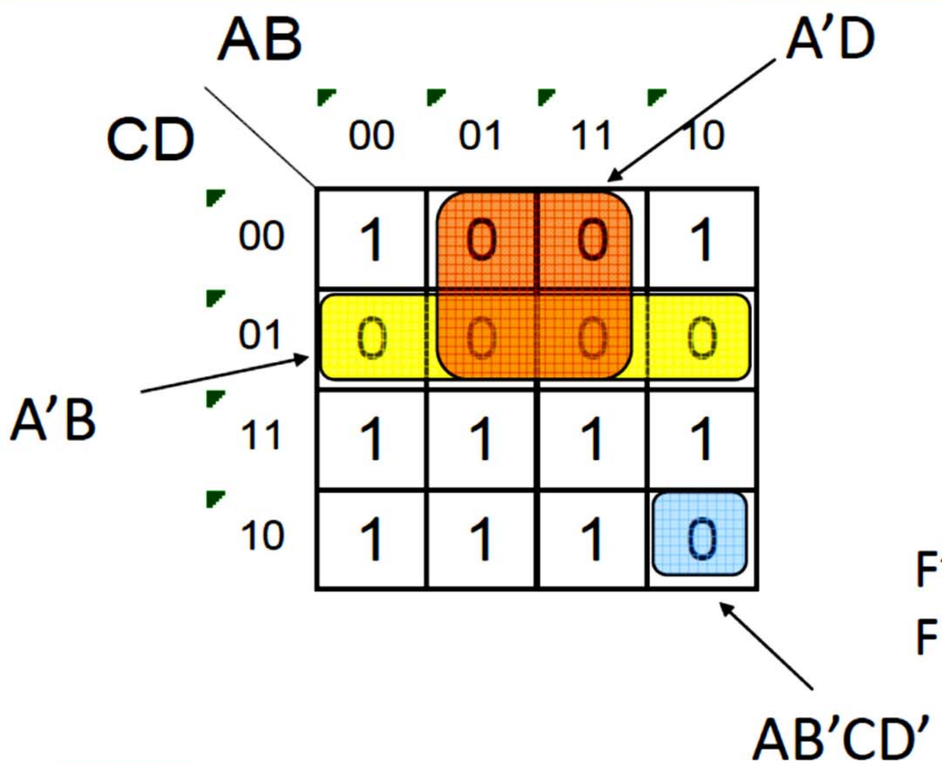
AB \ CD	00	01	11	10
00	m0	m1	m3	m2
01	m4	m5	m7	m6
11	m12	m13	m15	m14
10	m8	m9	m11	m10



F = AC’D + A’CD’ + B

Note the row and column orderings. Required for adjacency

Find a POS Solution



F' = A'D + A'B + AB'CD'

F = (A+D)(A+B')(A'+B+C'+D)

Find solutions to groups of 0's to find F' Invert to get F then use DeMorgan's

Thank You