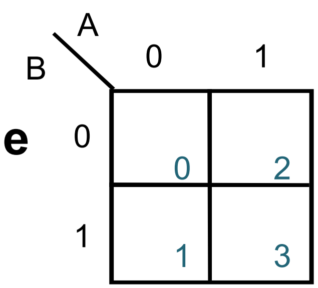
بهینه سازی با جدول کارنو

Karnaugh Map

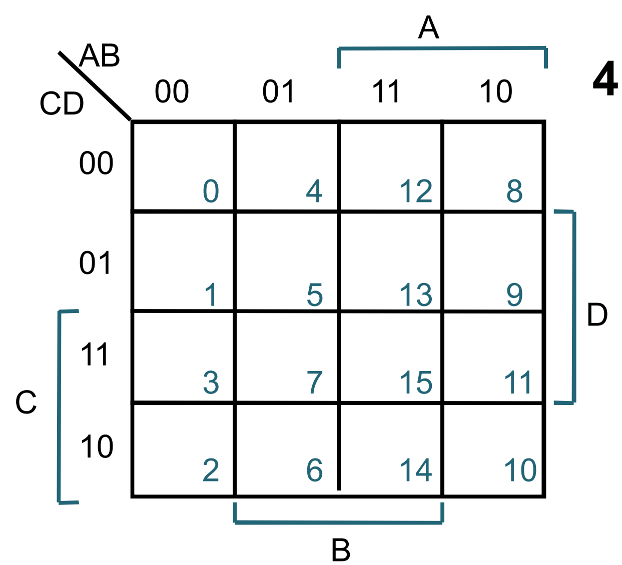
Karnaugh Map

- Method of graphically representing the truth table that helps visualize adjacencies

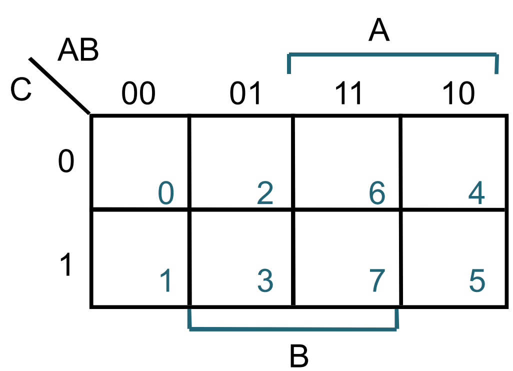
2-variable K-map



4-variable K-map

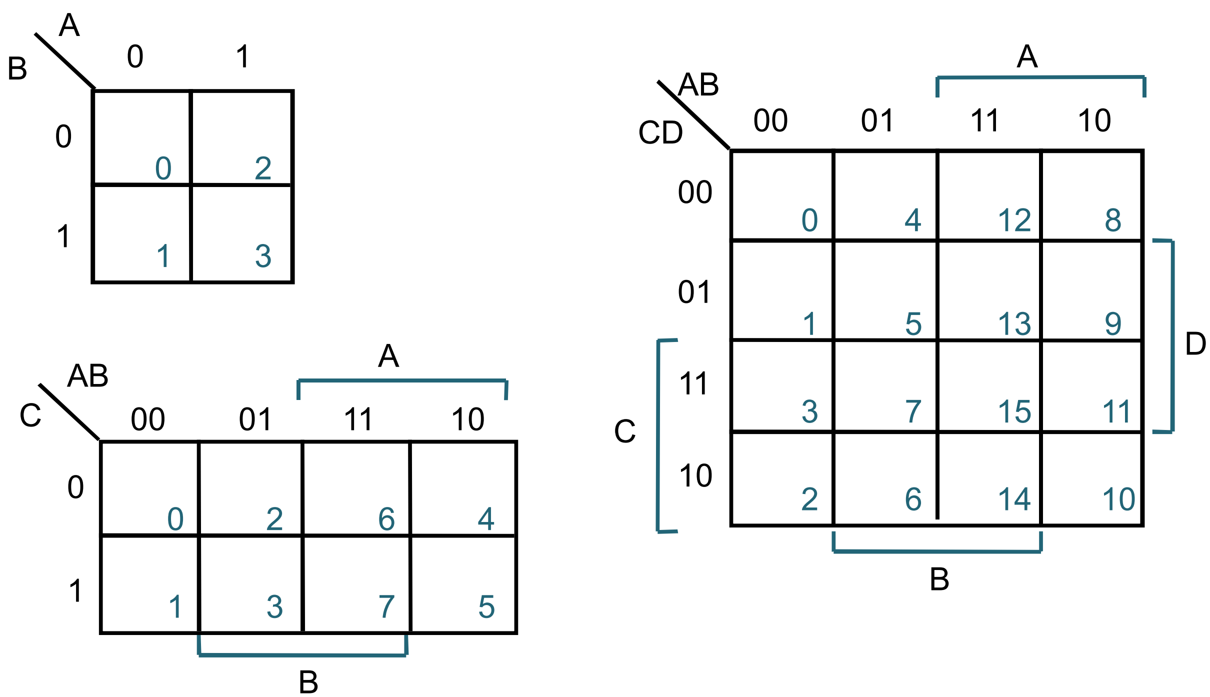


3-variable K-map



Karnaugh Map

- One cell in K-map = one row in truth table
- One cell = a minterm (or a maxterm)
- Multiple-cell areas = standard terms



Karnaugh Map

$$f_1(A,B,C) = m_1 + m_2 + m_4 + m_6$$

$$= A'B'C + A'BC' + AB'C' + ABC'$$

C \ AB	00	01	11	10
0	0 $_0$	1 $_2$	1 $_6$	1 $_4$
1	1 $_1$	0 $_3$	0 $_7$	0 $_5$

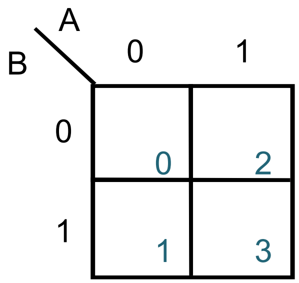
A: columns 11, 10; B: columns 01, 11

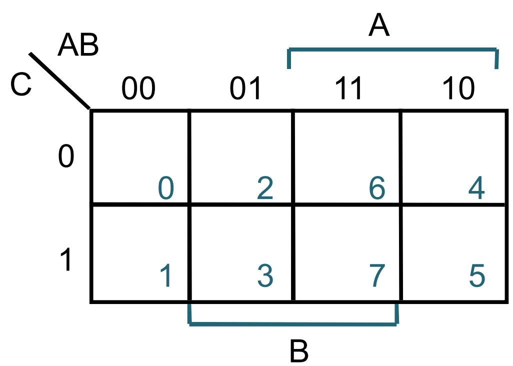
A	B	C		f_1
0	0	0	m0	0
0	0	1	m1	1
0	1	0	m2	1
0	1	1	m3	0
1	0	0	m4	1
1	0	1	m5	0
1	1	0	m6	1
1	1	1	m7	0

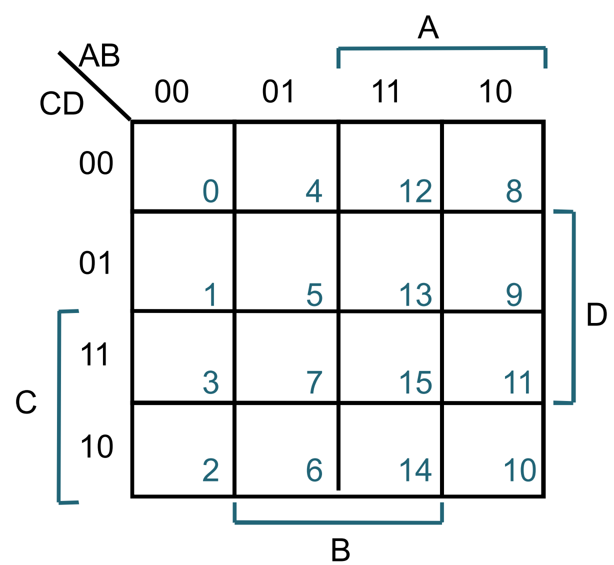
Karnaugh Map

- Numbering Scheme: 00, 01, 11, 10

  Gray Code: only a single bit changes from one code word to the next code word.







Two-Variable Map (cont.)

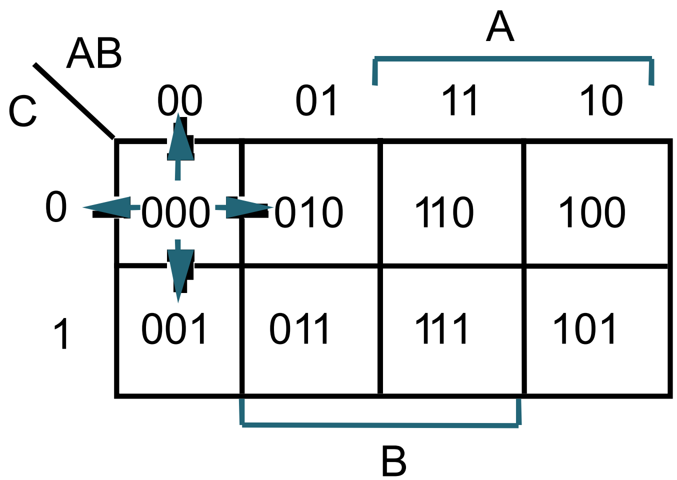
- Any two adjacent cells in the map differ by ONLY one variable, which appears complemented in one cell and uncomplemented in the other.

- **Example:**
  - m_0 (=A'B') is adjacent to m_1 (=A'B) and m_2 (=AB') but NOT m_3 (=AB)

B \ A	0	1
0	0	2
1	1	3

Karnaugh Map

- Adjacencies in the K-Map

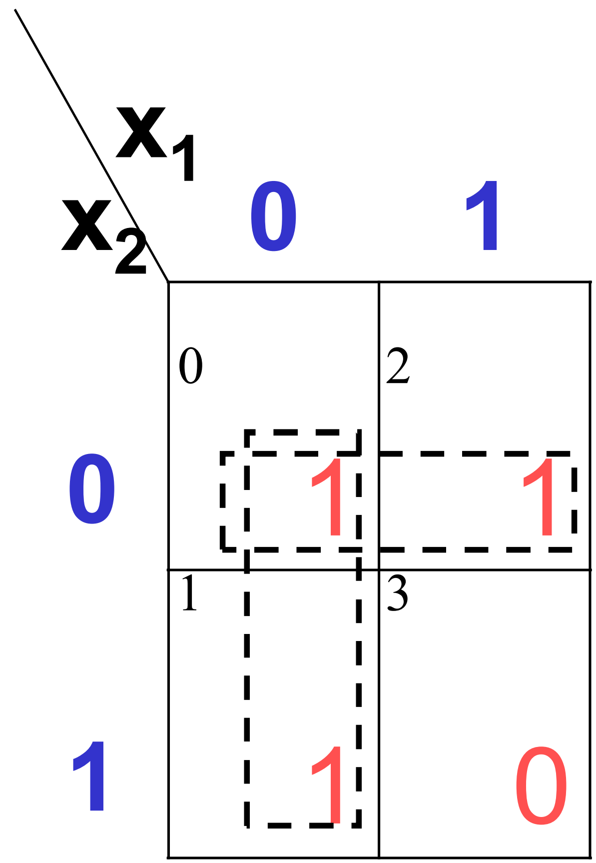


Wrap from first to last column

Top row to bottom row

2-Variable Map -- Example

- $f(x_1,x_2) = x_1'x_2' + x_1'x_2 + x_1x_2'$
  $= m_0 + m_1 + m_2$
- 1's placed in K-map for specified minterms m_0, m_1, m_2
- Grouping (ORing) of 1s allows simplification
- What (simpler) function is represented by each dashed rectangle?
  - $m_0 + m_1 = x_1'x_2' + x_1'x_2 =$
    $x_1'(x_2' + x_2) = x_1'$
  - $m_0 + m_2 = x_1'x_2' + x_1x_2' =$
    $x_2'(x_1' + x_1) = x_2'$
- Simplified form: $f(x_1,x_2) = x_1' + x_2'$
- Note m_0 is covered twice



Minimization as SOP using K-map

- Enter 1's in the K-map for each product term (minterm) in the function
- Group *adjacent* K-map cells containing 1's to obtain a product with fewer variables
  - Groups must be in power of 2 (2, 4, 8, …)
- Handle "boundary wrap"
- Answer may not be unique

Minimization as SOP

B \ A	0	1
0	0	1
1	0	1

A asserted, unchanged
B varies

B complemented, unchanged
A varies

B \ A	0	1
0	1	1
1	0	0

F = ?
F = A

G = ?
G = B'

Cin \ A B	00	01	11	10
0	0	0	1	0
1	0	1	1	1

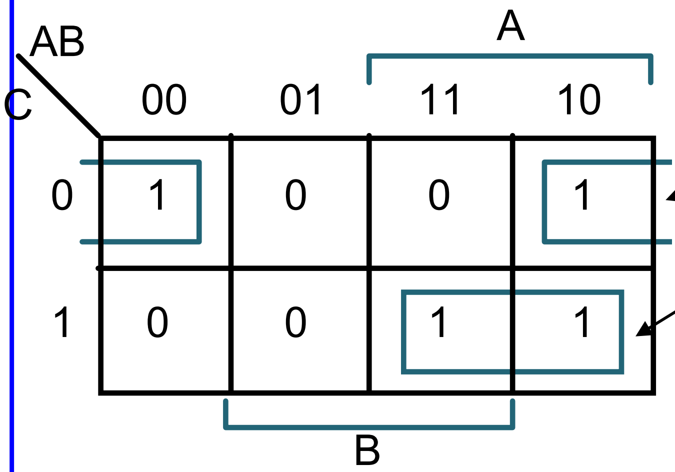
Cout = ?

Cout = AB + BCin + ACin

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

F(A,B,C) = ?
F(A,B,C) = A

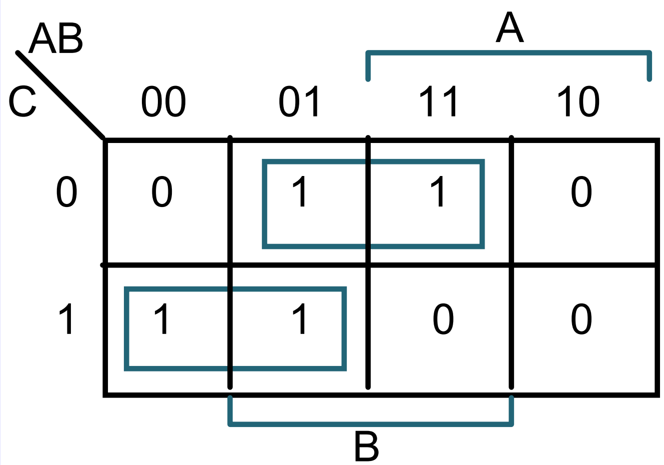
More Examples



Why not group m4 and m5?

F(A,B,C) = Σm(0,4,5,7)

F = B' C' + A C



F' simply replaces 1's with 0's and vice versa

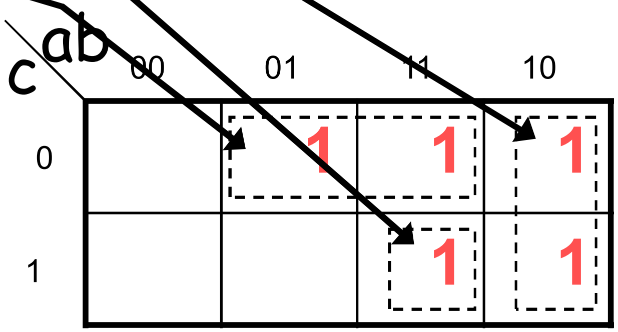
F'(A,B,C) = Σm(1,2,3,6)

F' = B C' + A' C

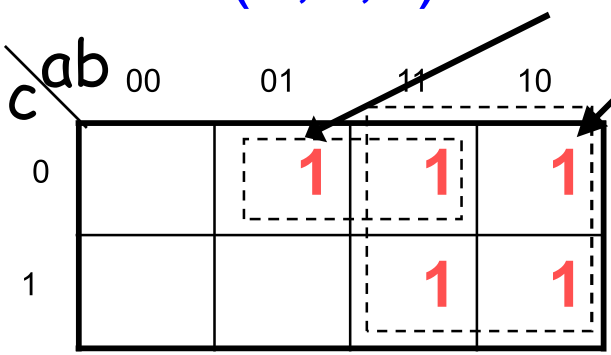
Simplification

- ➢ Enter minterms of the Boolean function into the map, then group terms

- **Example:**
  - ➢ $f(a,b,c) = bc' + abc + ab'$



➢ Result: $f(a,b,c) = bc' + a$



4-Variable Map

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

F(A,B,C,D) = Σm(0,2,3,5,6,7,8,10,11,14,15)

F = ?

CD \ AB	00	01	11	10
00	m_0	m_4	m_{12}	m_8
01	m_1	m_5	m_{13}	m_9
11	m_3	m_7	m_{15}	m_{11}
10	m_2	m_6	m_{14}	m_{10}

Four-variable Map Simplification

- One square represents a minterm of 4 literals.
- A rectangle of 2 adjacent squares represents a product term of 3 literals.
- A rectangle of 4 squares represents a product term of 2 literals.
- A rectangle of 8 squares represents a product term of 1 literal.
- A rectangle of 16 squares produces a function that is equal to logic 1.

4-Variable Map

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

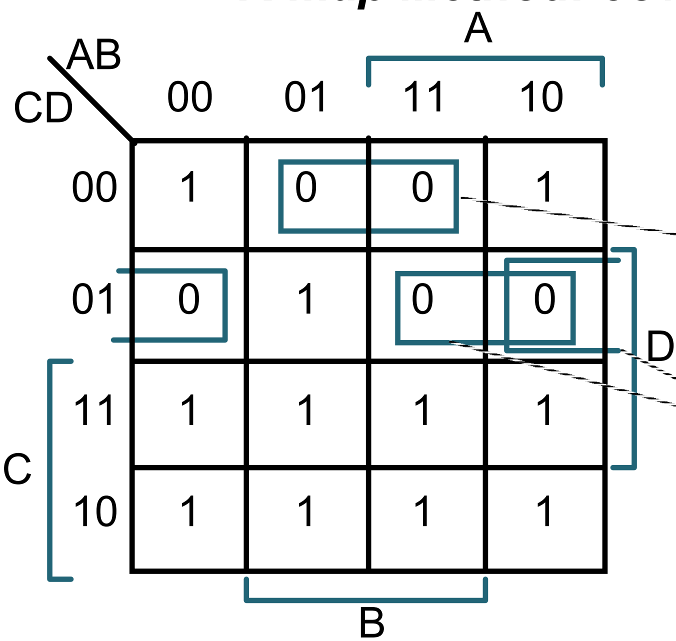
$$F(A,B,C,D) = \Sigma m(0,2,3,5,6,7,8,10,11,14,15)$$

$$F = C + A'BD + B'D'$$

- ➢Find the **smallest number** of the **largest possible subcubes** that cover the ON-set (for SoP)

Simplify for POS

K-map Method: covering zeros to get product of sums form



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

Alternative approach:

Replace F by F', 0's become 1's and vice versa

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

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

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

Don’t Cares

Don't Cares can be treated as 1's or 0's, whichever is more advantageous

CD \ AB	00	01	11	10
00	0	0	X	0
01	1	1	X	1
11	1	1	0	0
10	0	X	0	0

$F(A,B,C,D) = \Sigma m(1,3,5,7,9) + \Sigma d(6,12,13)$

$F = A'D + B'C'D$ **w/o don't cares**

$F = A'D + C'D$ **w/ don't cares**

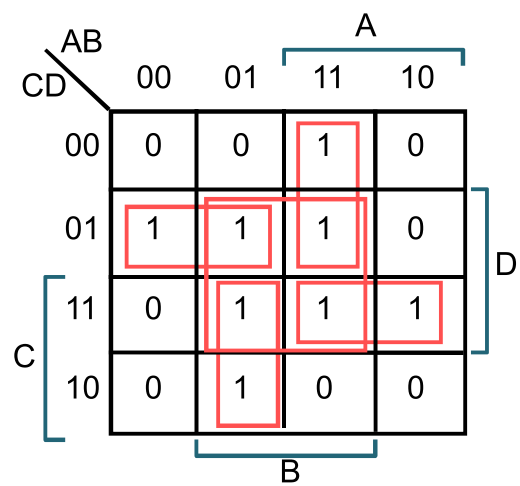
CD \ AB	00	01	11	10
00	0	0	X	0
01	1	1	X	1
11	1	1	0	0
10	0	X	0	0

In Product of Sums form: $F = D (A' + C')$

Same answer as above,
but fewer gates

Order Dependency

- **Order is important**



Fact: we will get different results depending on the order of groupings.

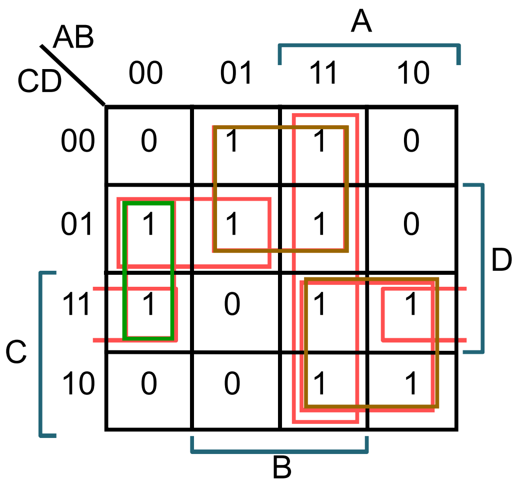
Question: can we make the approach order-independent?

Answer: yes, as can be seen in the next slides.

Definition of Terms

- **Implicant:**
  - single element of the ON-set or any group of elements that can be combined together in a K-map (= adjacency plane)
- **Prime Implicant (PI)** **(maximal PI):**
  - implicant (a circled set of 1-cells) satisfying the combining rule, such that if we try to make it larger (covering twice as many cells), it covers one or more 0s.
- **Distinguished 1-cell:**
  - an input combination that is covered by only one PI.
- **Essential Prime Implicant (EPI):**
  - a PI that covers one or more distinguished 1-cells.

Implicant, PI and EPI



6 Prime Implicants:

A' B' D, B C', A C, A' C' D, A B, B' C D

Essential

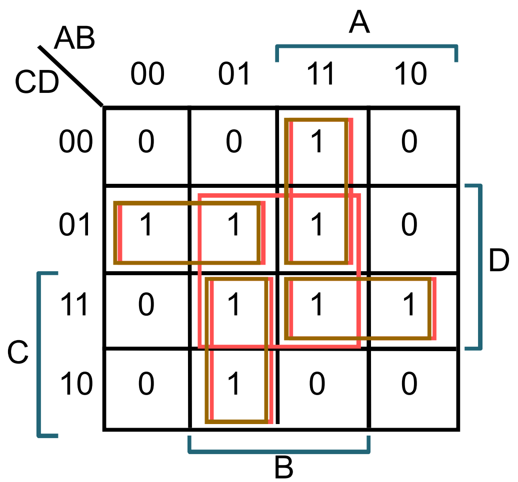
Minimum cover =

First: cover EPIs

Then: minimum number of PIs

= B C' + A C + A' B' D

Implicant, PI and EPI



5 Prime Implicants:

B D, A B C', A C D, A' B C, A' C' D

essential

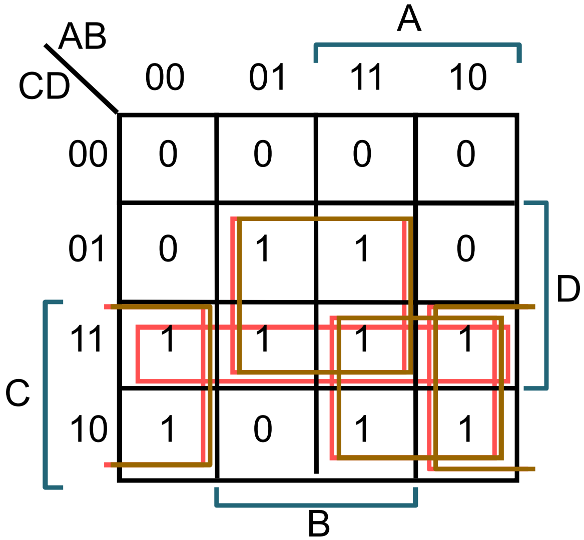
Minimum cover =

First: cover EPIs

Then: minimum number of PIs

= A B C’ + A C D + A' B C + A' C' D

More Examples



Prime Implicants:

B D, C D, A C, B' C

essential

= BD + AC + B' C

Example

What if there are don't cares in the k-map?

Example: $f(A,B,C,D) = m(4,5,6,8,9,10,13) + d(0,7,15)$

CD \ AB	00	01	11	10
00	X	1	0	1
01	0	1	1	1
11	0	X	X	0
10	0	1	0	1

Initial K-map

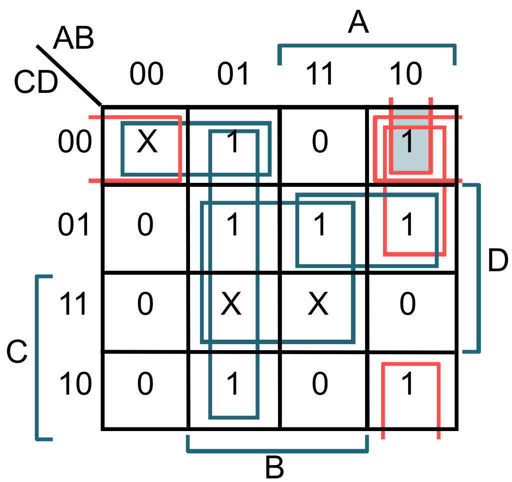
CD \ AB	00	01	11	10
00	X	1	0	1
01	0	1	1	1
11	0	X	X	0
10	0	1	0	1

Primes around A' B C' D'

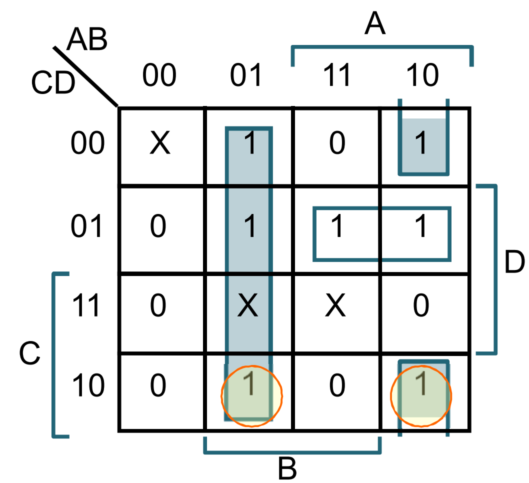
CD \ AB	00	01	11	10
00	X	1	0	1
01	0	1	1	1
11	0	X	X	0
10	0	1	0	1

Primes around A B C' D

Example: Continued

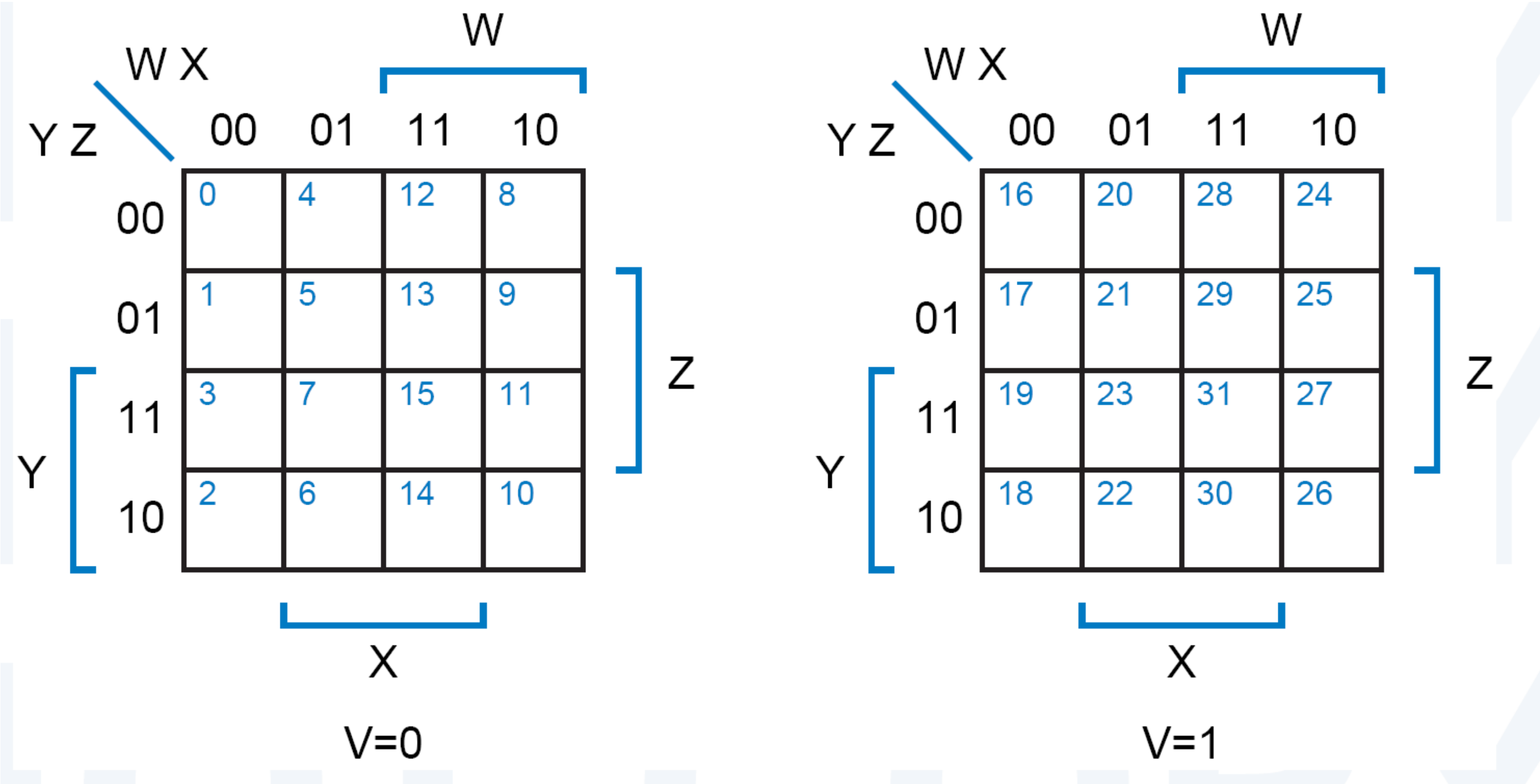


Primes around
A B' C' D'

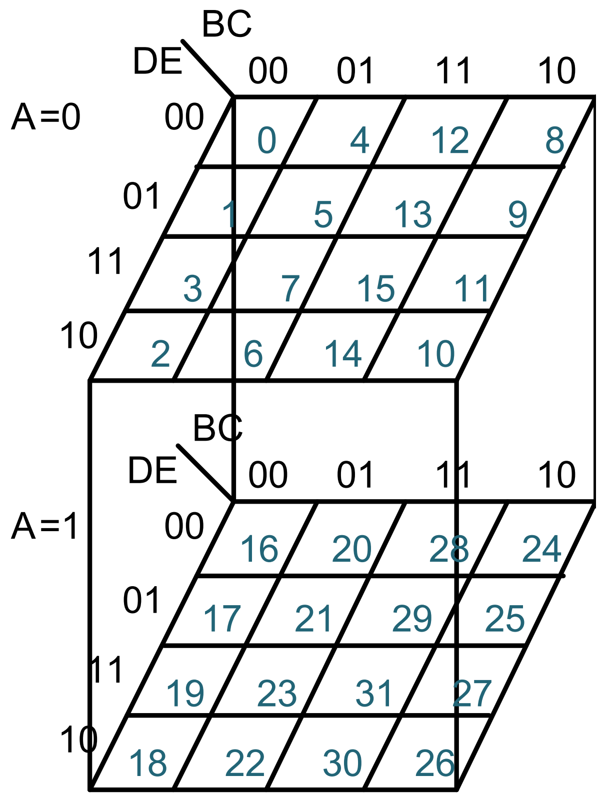


Essential Primes
with Min Cover
(each element covered once)

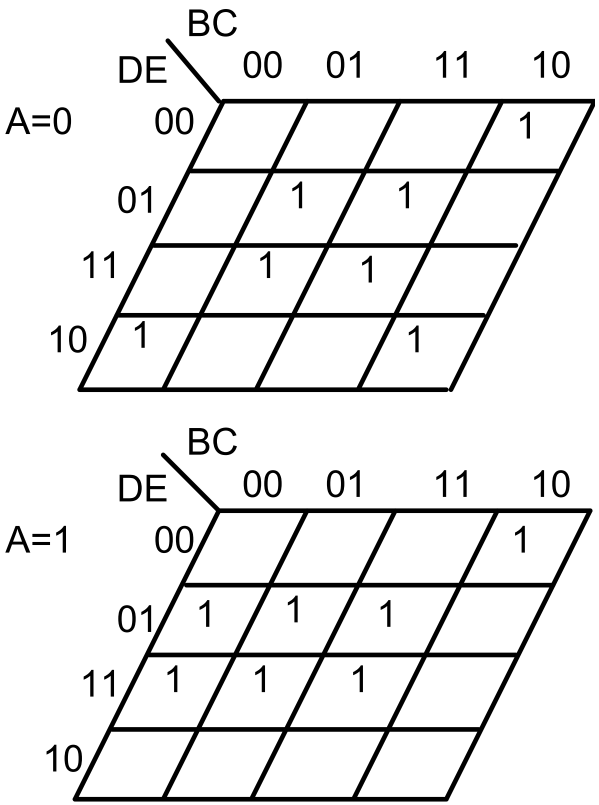
5-Variable K-Map

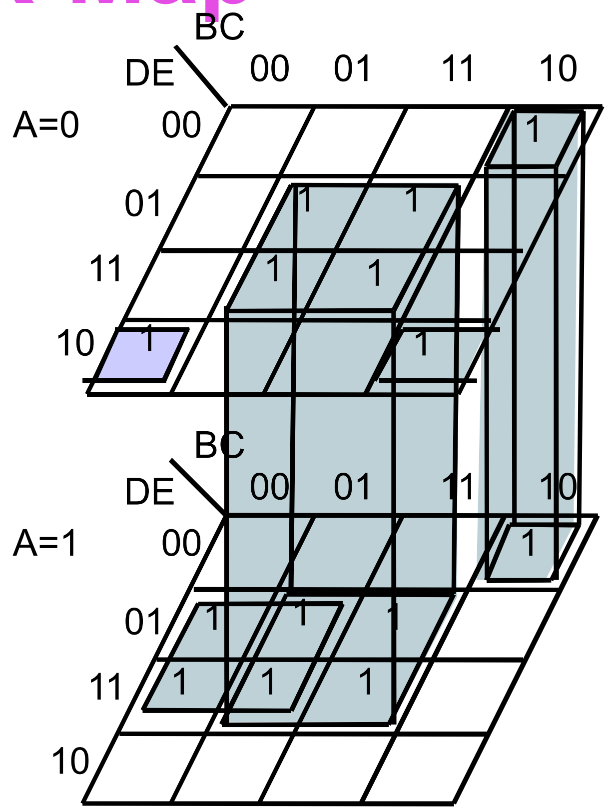


5-Variable K-Map



5-Variable K-Map

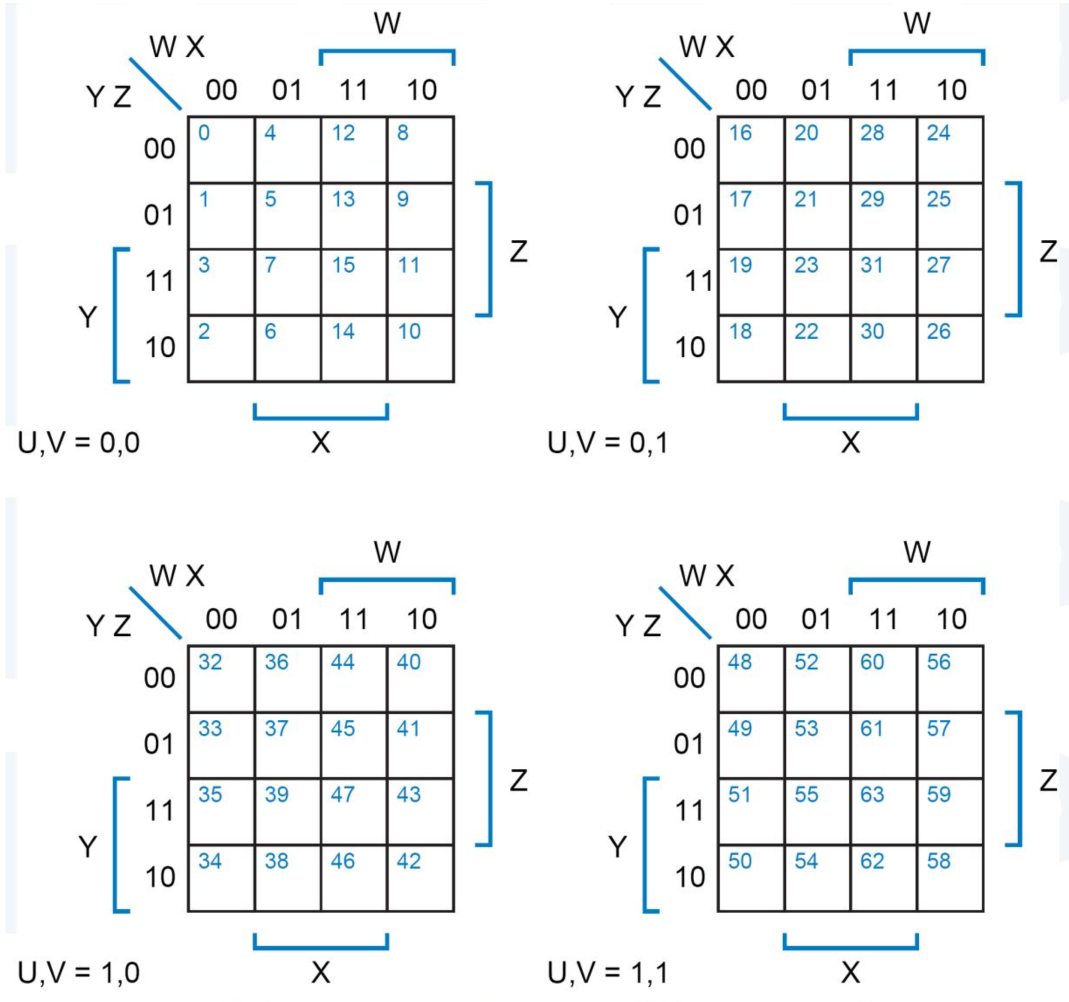




f(A,B,C,D,E) = Σm(2,5,7,8,10, 13,15,17,19,21,23,24,29,31)

= C E + A B' E + B C' D' E' + A' C' D E'

6-Variable K-Map



7-Variable K-Map

?

8-Variable K-Map

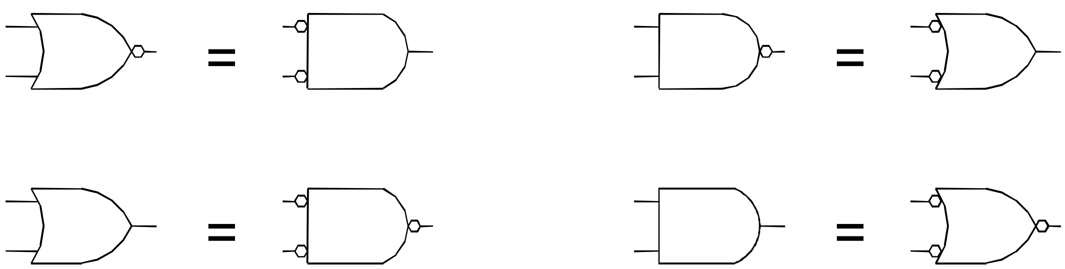
?

Implemention by NAND gates only

- **NAND:**
  - Universal gate
  - Can replace gates by equivalent NAND circuit.
    - Large circuit (many gates)

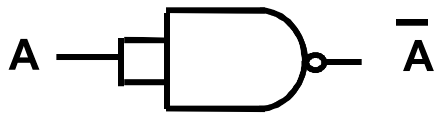
New Symbols for AND/OR

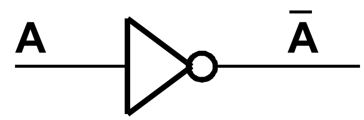
- **DeMorgan's Law:**
  - $(a + b)' = a' b'$ $(a b)' = a' + b'$
  - $a + b = (a' b')'$ $(a b) = (a' + b')'$



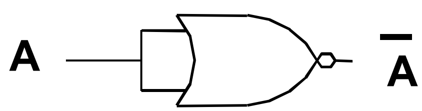
NAND-only and NOR-only implementations for NOT

- $(a \cdot a)' = a'$



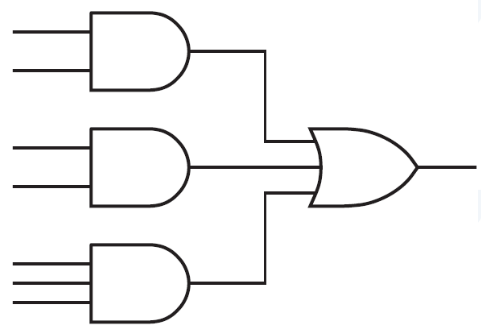


- $(a + a)' = a'$

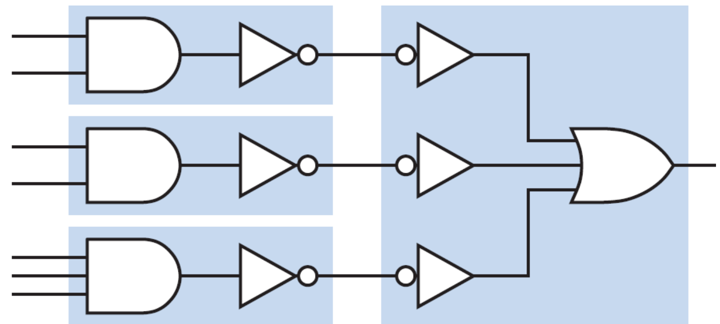


Finding NAND-only Implementation

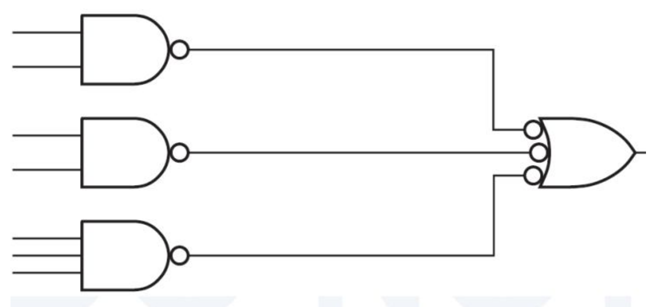
- 1st step: Find Sum-of-Product form



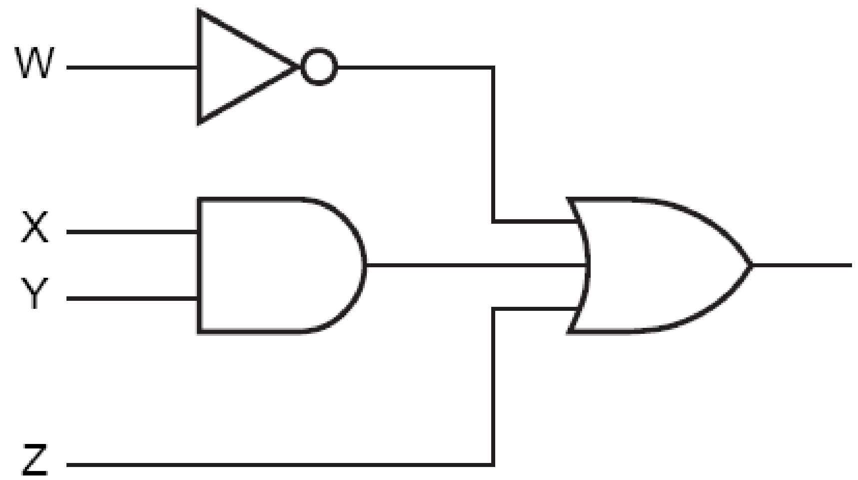
- 2nd step: Add double inverters

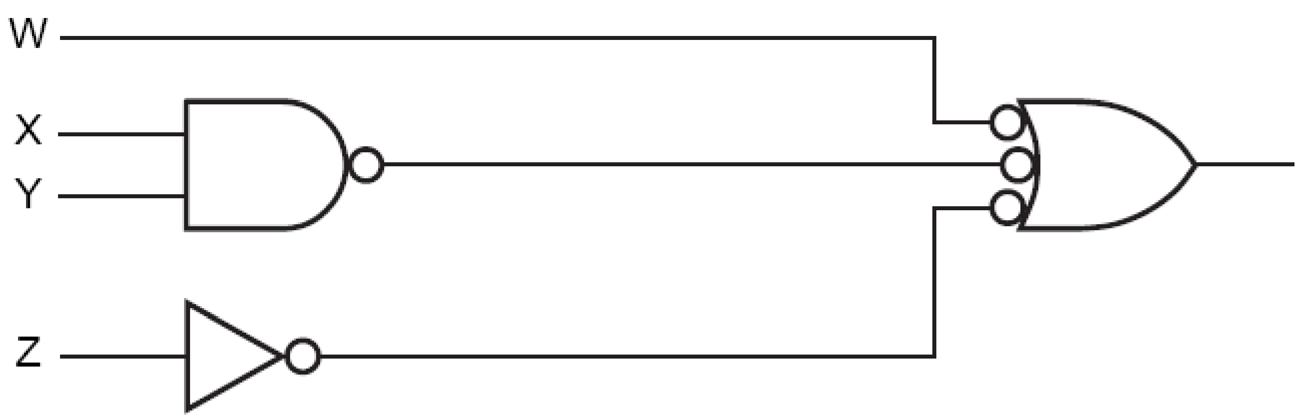


- 3rd step: Identify equivalent NANDs



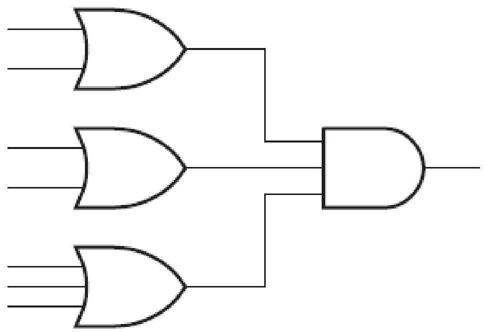
Another Example



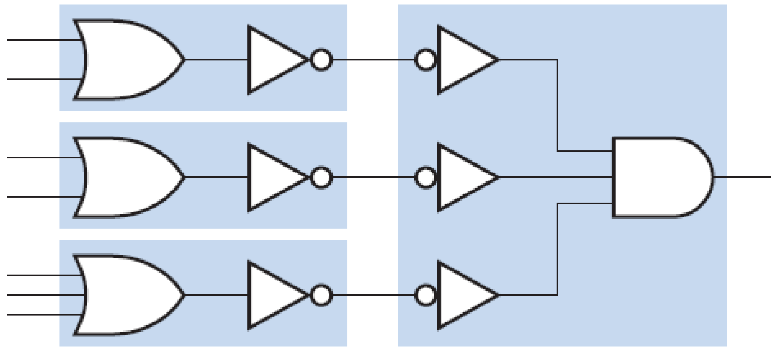


NOR-Only Implementation

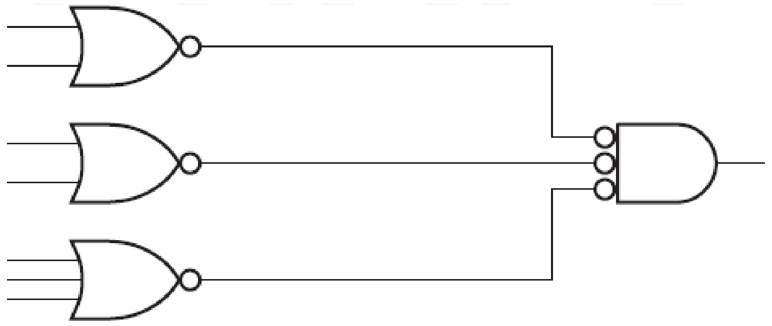
- 1st step: Find Product-of-Sums form



- 2nd step: Add double inverters



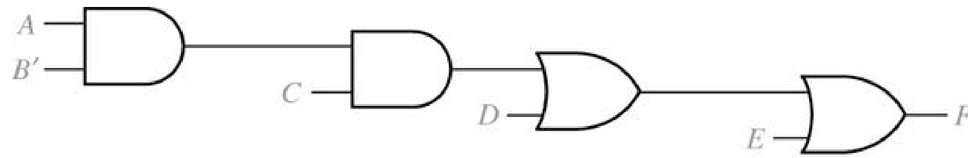
- 3rd step: Identify equivalent NORs



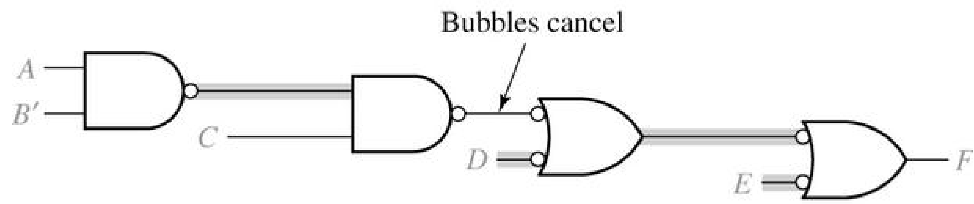
NAND-Only Implementation

• Multi-Level Circuits

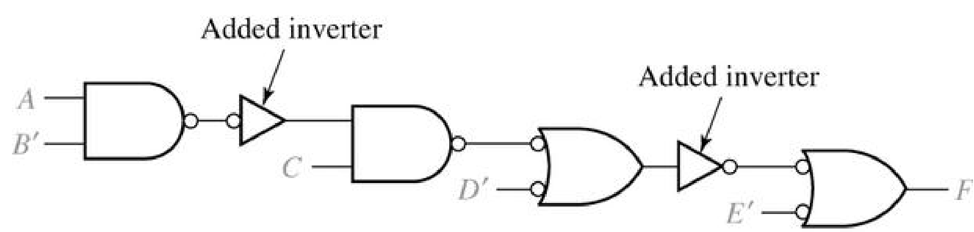
- ➢ Convert AND/OR gates to proper NAND gates
  - ➢AND → AND-NOT symbol
  - ➢OR → NOT-OR symbol
- ➢ Bubbles must cancel each other;
- ➢ otherwise, insert a NAND inverter.
- ➢ Take care of appropriate input literals.



(a) AND-OR network



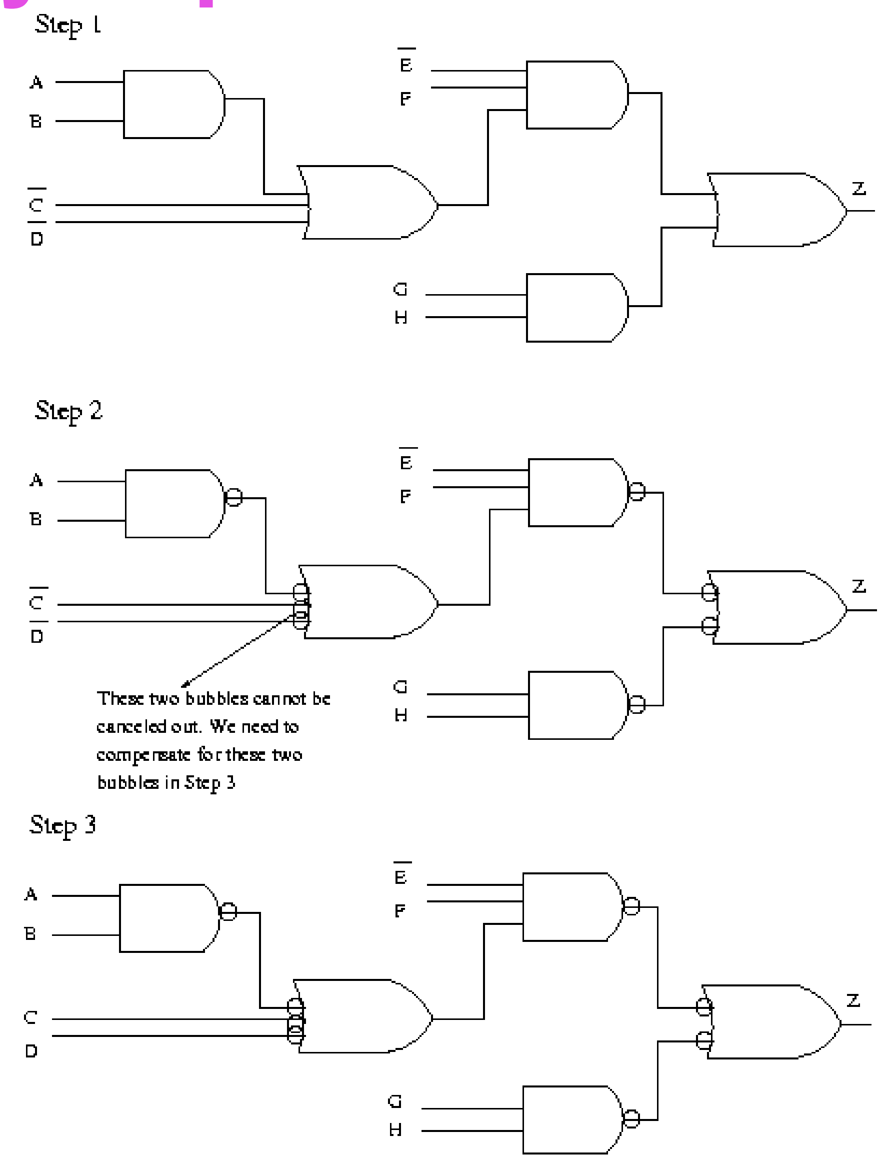
(b) First step in NAND conversion



(c) Completed conversion

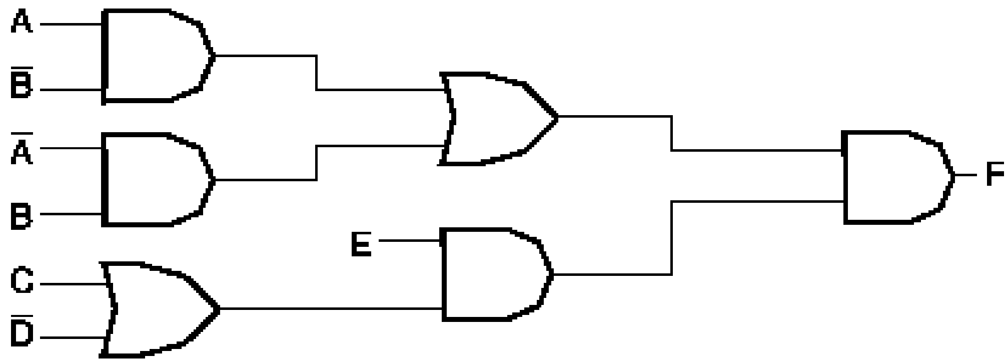
NAND-Only Implementation

- **Example:**

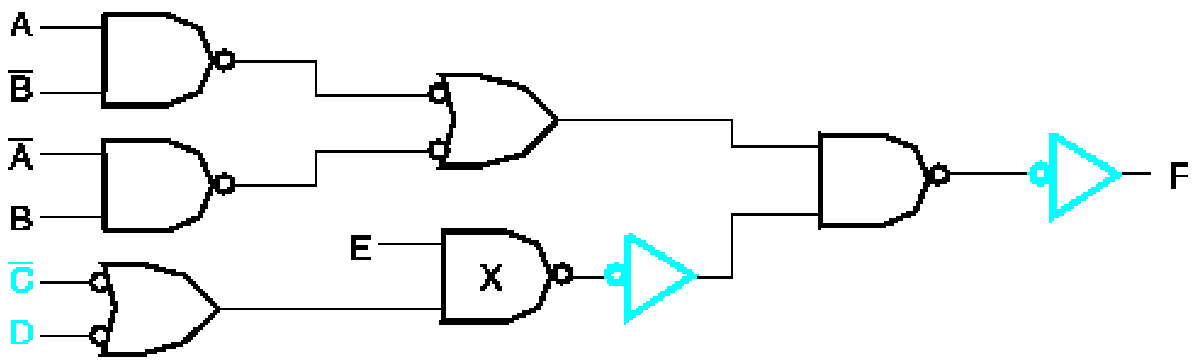


NAND-Only Implementation

- **Another Example:**



(a) AND – OR gates

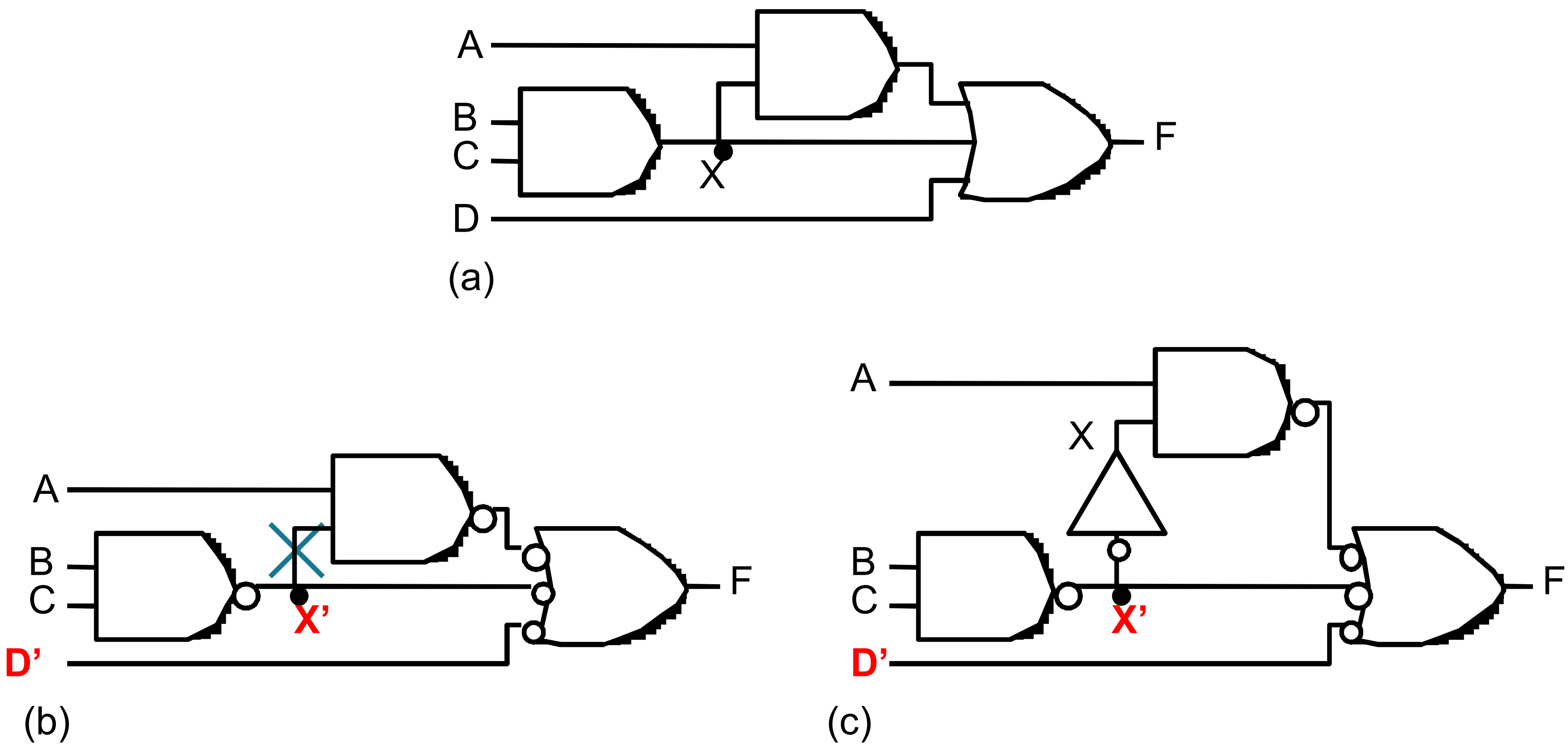


(b) NAND gates

Fig. 2-32 Implementing $F = (A\overline{B} + \overline{A}B)E(C + \overline{D})$

NAND-Only Implementation

- **Be careful about branches:**
  - Gates with multi-fanouts



NOR-Only Implementation

- **NOR-Only:**
  - Use "Duality" for the last several slides.