Digital Logic Design

Combinational Logic

Minterms

- A **product term** is a term where literals are ANDed.
  - Example: x'y', xz, xyz, …
- A **minterm** is a product term in which all variables appear exactly once, in normal or complemented form
  - Example: F(x,y,z) has 8 minterms: x'y'z', x'y'z, x'yz', ...
- In general, a function with n variables has 2^n minterms
- A minterm equals 1 at exactly one input combination and is equal to 0 otherwize
  - Example: x'y'z' = 1 only when x=0, y=0, z=0
- A minterm is denoted as m_i where i corresponds the input combination at which this minterm is equal to 1

Minterms

Minterms for Three Variables

Src: Mano's book

X	Y	Z	Product Term	Symbol	m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
0	0	0	$\overline{X}\,\overline{Y}\,\overline{Z}$	m_0	1	0	0	0	0	0	0	0
0	0	1	$\overline{X}\,\overline{Y}Z$	m_1	0	1	0	0	0	0	0	0
0	1	0	$\overline{X}Y\overline{Z}$	m_2	0	0	1	0	0	0	0	0
0	1	1	$\overline{X}YZ$	m_3	0	0	0	1	0	0	0	0
1	0	0	$X\overline{Y}\,\overline{Z}$	m_4	0	0	0	0	1	0	0	0
1	0	1	$X\overline{Y}Z$	m_5	0	0	0	0	0	1	0	0
1	1	0	$XY\overline{Z}$	m_6	0	0	0	0	0	0	1	0
1	1	1	XYZ	m_7	0	0	0	0	0	0	0	1

Variable complemented if 0
Variable uncomplemented if 1

m_i indicated the i^{th} minterm
i indicates the binary combination
m_i is equal to 1 for ONLY THAT combination

Maxterms

- A **sum term** is a term where literals are ORed.
  - Example: x'+y', x+z, x+y+z, …
- A **maxterm** is a sum term in which all variables appear exactly once, in normal or complemented form
  - Example: F(x,y,z) has 8 maxterms: (x+y+z), (x+y+z'), (x+y'+z), ...
- In general, a function with n variables has 2^n maxterms
- A maxterm equals 0 at exactly one input combination and is equal to 1 otherwize
  - Example: (x+y+z) = 0 only when x=0, y=0, z=0
- A maxterm is denoted as M_i where i corresponds the input combination at which this maxterm is equal to 0

Maxterms

Src: Mano's book

Maxterms for Three Variables

X	Y	Z	Sum Term	Symbol	M_0	M_1	M_2	M_3	M_4	M_5	M_6	M_7
0	0	0	$X+Y+Z$	M_0	0	1	1	1	1	1	1	1
0	0	1	$X+Y+\overline{Z}$	M_1	1	0	1	1	1	1	1	1
0	1	0	$X+\overline{Y}+Z$	M_2	1	1	0	1	1	1	1	1
0	1	1	$X+\overline{Y}+\overline{Z}$	M_3	1	1	1	0	1	1	1	1
1	0	0	$\overline{X}+Y+Z$	M_4	1	1	1	1	0	1	1	1
1	0	1	$\overline{X}+Y+\overline{Z}$	M_5	1	1	1	1	1	0	1	1
1	1	0	$\overline{X}+\overline{Y}+Z$	M_6	1	1	1	1	1	1	0	1
1	1	1	$\overline{X}+\overline{Y}+\overline{Z}$	M_7	1	1	1	1	1	1	1	0

Variable complemented if 1
Variable not complemented if 0

M_i indicated the i^{th} maxterm
i indicates the binary combination
M_i is equal to 0 for ONLY THAT combination

Minterms and Maxterms

In general, a function of n variables has

- 2^n minterms: $m_0, m_1, \ldots, m_{2^n-1}$
- 2^n maxterms: $M_0, M_1, \ldots, M_{2^n-1}$

Minterms and maxterms are the complement of each other!

$$M_i = \overline{m_i} \qquad \forall i = 0,1,2,\ldots\ldots,(2^n - 1)$$

Example: F(X,Y):

$m_2 = XY' \Rightarrow m_2' = X'+Y = M_2$

Expressing Functions with Minterms

- A Boolean function can be expressed algebraically from a give truth table by forming the logical sum (OR) of ALL the minterms that produce 1 in the function

Example:

Consider the function defined by the truth table

$$F(X,Y,Z) = X'Y'Z' + X'YZ' + XY'Z + XYZ$$
$$= m_0 + m_2 + m_5 + m_7$$
$$= \Sigma m(0,2,5,7)$$

X	Y	Z	m	F
0	0	0	m_0	1
0	0	1	m_1	0
0	1	0	m_2	1
0	1	1	m_3	0
1	0	0	m_4	0
1	0	1	m_5	1
1	1	0	m_6	0
1	1	1	m_7	1

Expressing Functions with Maxterms

- A Boolean function can be expressed algebraically from a give truth table by forming the logical product (AND) of ALL the maxterms that produce 0 in the function

Example:

Consider the function defined by the truth table

$F(X,Y,Z) = \Pi\ M(1,3,4,6)$

X	Y	Z	M	F	F′
0	0	0	M_0	1	0
0	0	1	M_1	0	1
0	1	0	M_2	1	0
0	1	1	M_3	0	1
1	0	0	M4	0	1
1	0	1	M_5	1	0
1	1	0	M_6	0	1
1	1	1	M_7	1	0

Applying DeMorgan

$F' = m_1 + m_3 + m_4 + m_6$

$= \Sigma m(1,3,4,6)$

$F = F'' = [m_1 + m_3 + m_4 + m_6]'$

$= m_1'.m_3'.m_4'.m_6'$

$= M_1.M_3.M_4.M_6$

$= \Pi\ M(1,3,4,6)$

Note the indices in this list are those that are missing from the previous list in $\Sigma m(0,2,5,7)$

Sum of Minterms vs Product of Maxterms

- A Boolean function can be expressed algebraically as:
  - The sum of minterms
  - The product of maxterms
- Given the truth table, writing F as
  - $\sum m_i$ – for all minterms that produce 1 in the table, or
  - ΠM_i – for all maxterms that produce 0 in the table
- Minterms and Maxterms are complement of each other.

Example (Cont.)

Solution: **Method2_a**

$E = Y' + X'Z'$

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

$= (XY'+X'Y')(Z+Z') + X'YZ'+X'Z'Y'$

$= XY'Z+X'Y'Z+XY'Z'+X'Y'Z'+ X'YZ'+X'Z'Y'$

$= m_5 + m_1 + m_4 + m_0 + m_2 + m_0$

$= m_0 + m_1 + m_2 + m_4 + m_5$

$= \Sigma m(0,1,2,4,5)$

To find the form ΠMi, consider the remaining indices

$E = \Pi M(3,6,7)$

Solution: **Method2_b**

$E = Y' + X'Z'$

$E' = Y(X+Z)$

$= YX + YZ$

$= YX(Z+Z') + YZ(X+X')$

$= XYZ+XYZ'+X'YZ$

$E = (X'+Y'+Z')(X'+Y'+Z)(X+Y'+Z')$

$= M_7 . M_6 . M_3$

$= \Pi M(3,6,7)$

To find the form Σm_i, consider the remaining indices

$E = \Sigma m(0,1,2,4,5)$

Example

Question: F (a,b,c,d) = $\sum m(0,1,2,4,5,7)$, What are the minterms and maxterms of F and and its complement $\overline{F}$?

Solution:

F has 4 variables; $2^4 = 16$ possible minterms/maxterms

$$F(a,b,c,d) = \sum m(0,1,2,4,5,7) = \prod M(3,6,8,9,10,11,12,13,14,15)$$

$$\overline{F}(a,b,c,d) = \sum m(3,6,8,9,10,11,12,13,14,15) = \prod M(0,1,2,4,5,7)$$

Canonical Forms

The sum of minterms and the product of maxterms forms are known as the **canonical forms** (الصيغ القانونية) of a function.

Standard Forms

- Sum of Products (SOP) and Product of Sums (POS) are also standard forms
  - AB+CD = (A+C)(B+C)(A+D)(B+D)
- The sum of minterms is a special case of the SOP form, where all product terms are minterms
- The product of maxterms is a special case of the POS form, where all sum terms are maxterms

SOP and POS Conversion

SOP → POS

$F = AB + CD$

$= (AB+C)(AB+D)$

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

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

Hint 1: Use $X+YZ=(X+Y)(X+Z)$

Hint 2: Factor

POS → SOP

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

$= (A'+BC)(C+D)$

$= A'C+A'D+BCC+BCD$

$= A'C+A'D+BC+BCD$

$= A'C+A'D+BC$

Hint 1: Use i $(X+Y)(X+Z)=X+YZ$

Hint 2: Multiply

Question1: How to convert SOP to sum of minterms?
Question2: How to convert POS to product of maxterms?

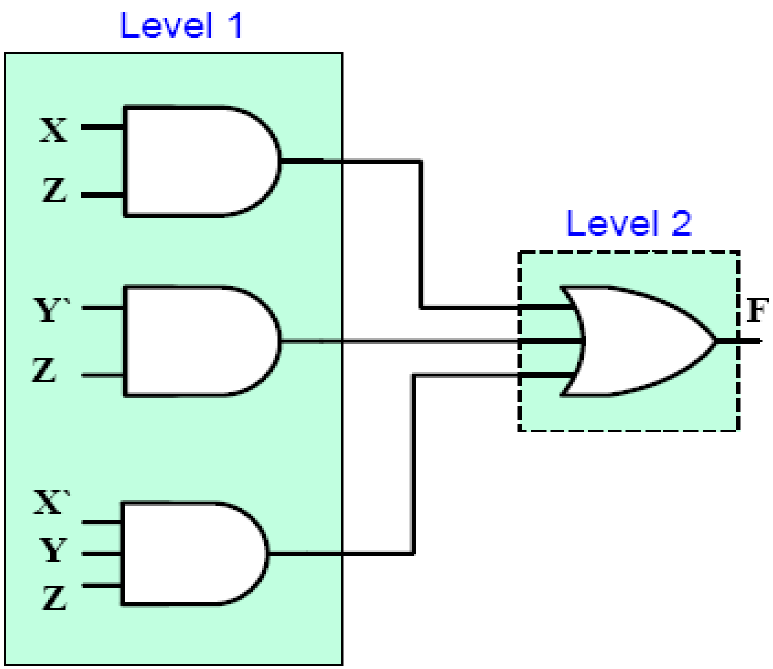
Implementation of SOP

Any SOP expression can be implemented using a

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



Two-Level Implementation (F = XZ + Y`Z + X`YZ)
Level-1: AND-Gates ; Level-2: One OR-Gate

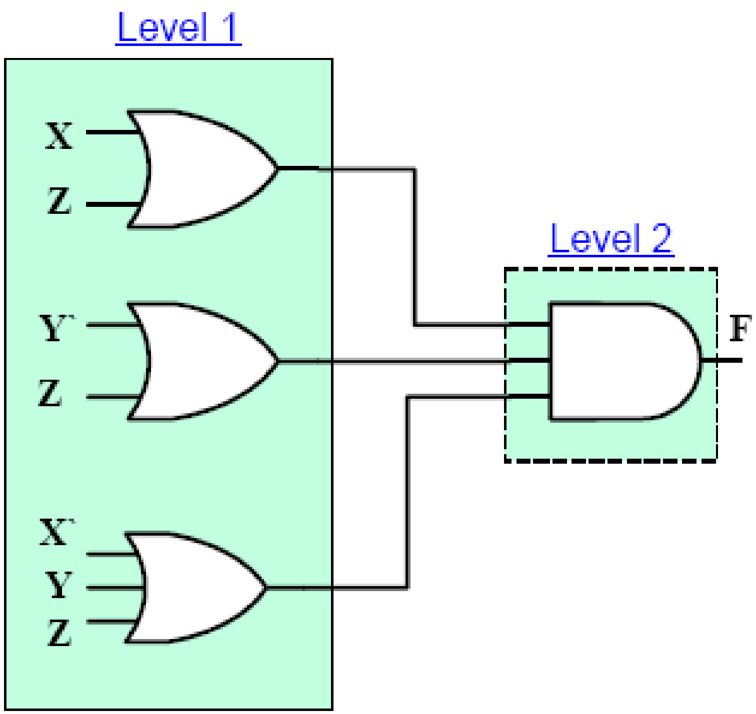
Implementation of POS

Any POS expression can be implemented using a

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



Two-Level Implementation {F = (X+Z)(Y`+Z)(X`+Y+Z)}

Level-1: OR-Gates ; Level-2: One AND-Gate

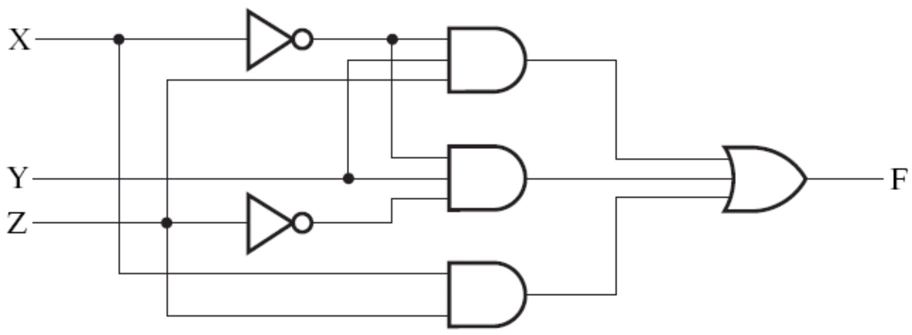
Simplification

- Simplification using Algebra
- Simplification using Karnaugh Maps (K-Maps)

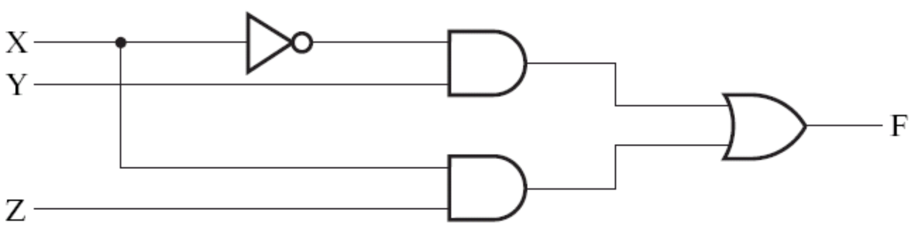
Simplification using Algebra

$$F = X'YZ + X'YZ' + XZ$$
$$= X'Y(Z+Z') + XZ$$
$$= X'Y.1 + XZ$$
$$= X'Y + XZ$$

- Simplification may mean different things
- here it means less number of literals



(a) $F = \overline{X}YZ + \overline{X}Y\overline{Z} + XZ$



(b) $F = \overline{X}Y + XZ$

Simplification Revisited

- Algebraic methods for minimization is limited:
  - No formal steps, need experience.
  - No guarantee that a minimum is reached
  - Easy to make mistakes
- Karnaugh maps (k-maps) is an alternative convenient way for minimization:
  - A graphical technique
  - Introduced by Maurice Karnaugh in 1953
- K-maps for up to 4 variables are straightforward to build
- Building higher order K-maps (5 or 6 variable) are a bit more cumbersome
- Simplified expression produced by K-maps are in SOP or POS forms

Gray Code & Truth Table Adjacencies

- Remember that Only one bit changes with each number increment in gray codes

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

These minterms are adjacent in a gray code sense – they differ by only one bit.

We can apply :

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

- When we group adjacent minterms (gray codes), we Keep common literal only!

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

Same idea:

F = A'B + AB = B

Keep common literal only!

K-Map

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

A different way to draw a truth table !

Take advantage of adjacency and gray codes

A \ B	0	1
0	0 A' B'	1 A' B
1	0 A B'	1 A B

F = A'B + AB = B

Keep common literal only!

Minimization (Simplification) with K-maps

1. Draw a K-map
2. Combine maximum number of 1's following rules:
    1. Only adjacent squares can be combined
    2. All 1's must be covered
    3. Covering rectangles must be of size 1,2,4,8, … 2^n
3. Check if all covering are really needed
4. Read off the SOP expression

2-variable K-map

- Given a function with 2 variables: F(X,Y), the total number of minterms are equal to 4:

  m_0, m_1, m_2, m_3

- The size of the k-map is always equal to the total number of minterms.
  - Each entry of the k-map corresponds to one minterm for the function:
  - Row 0 represents: X'Y', X'Y
  - Row 1 represents: XY', XY'

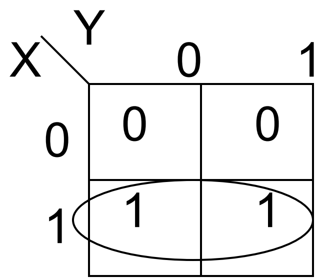
X \ Y	0	1
0	m_0	m_1
1	m_2	m_3

X \ Y	0	1
0	*0*	*1*
1	*2*	*3*

Example 1

For a given function F(X,Y) with the following truth table, minimize it using k-maps

X	Y	F
0	0	0
0	1	0
1	0	1
1	1	1



Combining all the 1's in only the adjacent squares

The final reduced expression is given by the common literals from the combination:

- Therefore, since for the combination, Y has different values (0, 1), and X has a fixed value of 1,

The reduced function is: F(X,Y) = X

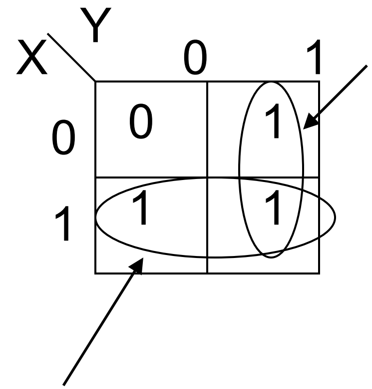
Example 2

Q. Simplify the function F(X,Y) = $\sum m(1,2,3)$

Sol. This function has 2 variables, and three 1-squares (three minterms where function is 1)

$F = m_1 + m_2 + m_3$

Note: The 1-squares can be combined more than once



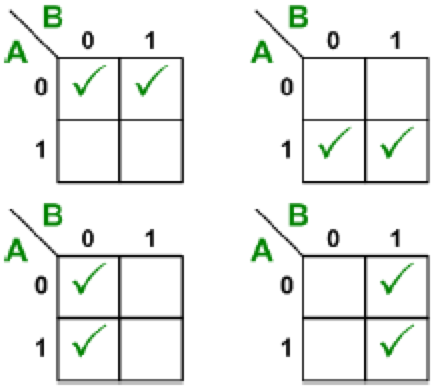
Y is the common literal in the adjacent 1-squares

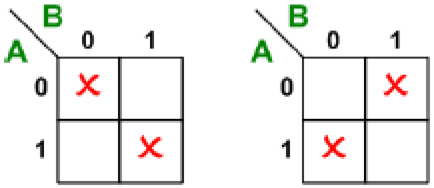
X is the common literal

Minimized expression: F = X + Y

2 variable K-Maps (Adjacency)

In an n-variable k-map, each square is adjacent to exactly *n* other squares





Q: What if you have 1 in all squares?

The boolean function does not depend on the variable , so it is a fixed logic 1

3-variable K-maps

- For 3-variable functions, the k-maps are larger and look different.
- Total number of minterms that need to be accommodated in the k-map = 8

To maintain adjacency neighbors don’t have more than 1 different bit

A \ BC	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

3-variable K-maps

A \ BC	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

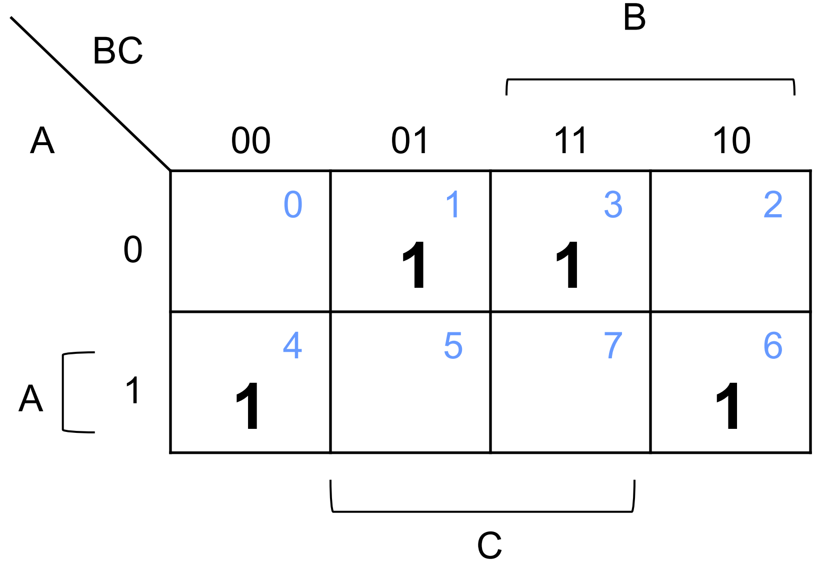
A \ BC	00	01	11	10
0	$\mathbf{m_0}$	m_1	m_3	$\mathbf{m_2}$
1	$\mathbf{m_4}$	m_5	m_7	$\mathbf{m_6}$

Note: You can only combine a power of 2 adjacent 1-squares. For e.g. 2, 4, 8, 16 squares. You cannot combine 3, 7 or 5 squares

- Minterms m_o, m_2, m_4, m_6 can be combined as m_0 and m_2 are adjacent to each other, m_4 and m_6 are adjacent to each other
- m_o and m_4 are also adjacent to each other, m_2 and m_6 are also adjacent to each other

Example 1

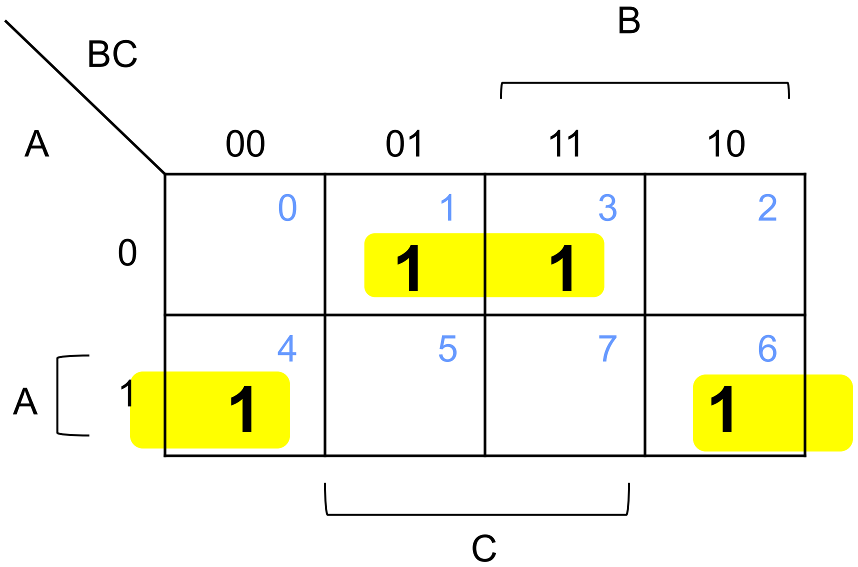
Simplify F = $\sum m(1, 3, 4, 6)$ using K-map



Example 1

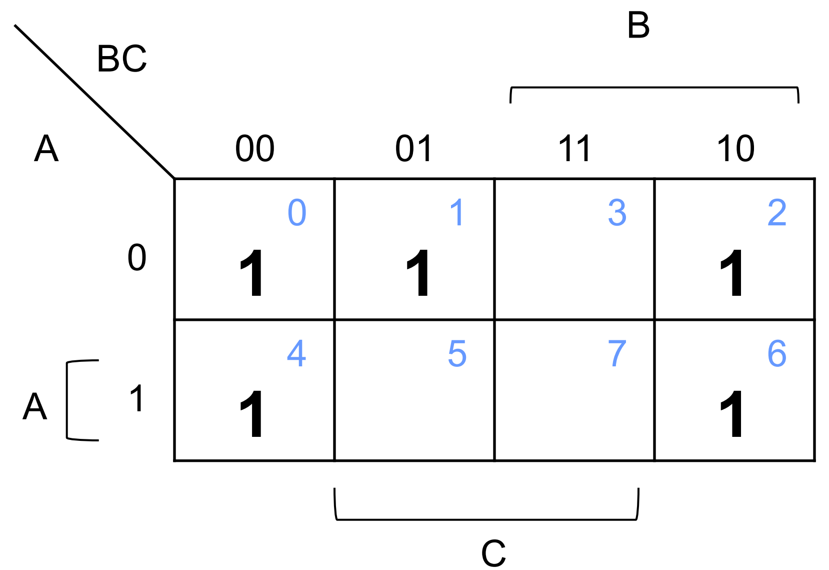
Simplify $F = \sum m(1, 3, 4, 6)$ using K-map

$F = A'C + AC'$



Example 2

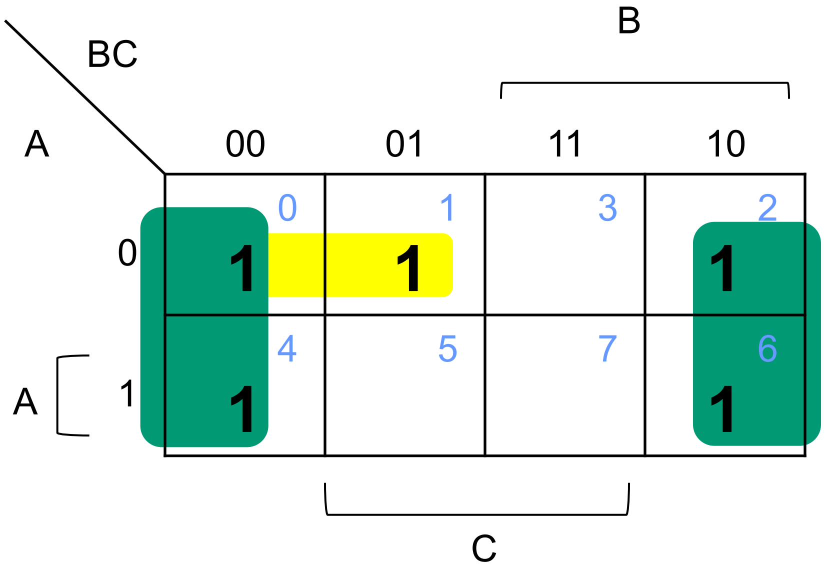
Simplify F = $\sum m(0,1, 2, 4, 6)$ using K-map



Example 2

Simplify F = $\sum m(0,1,2,4,6)$ using K-map

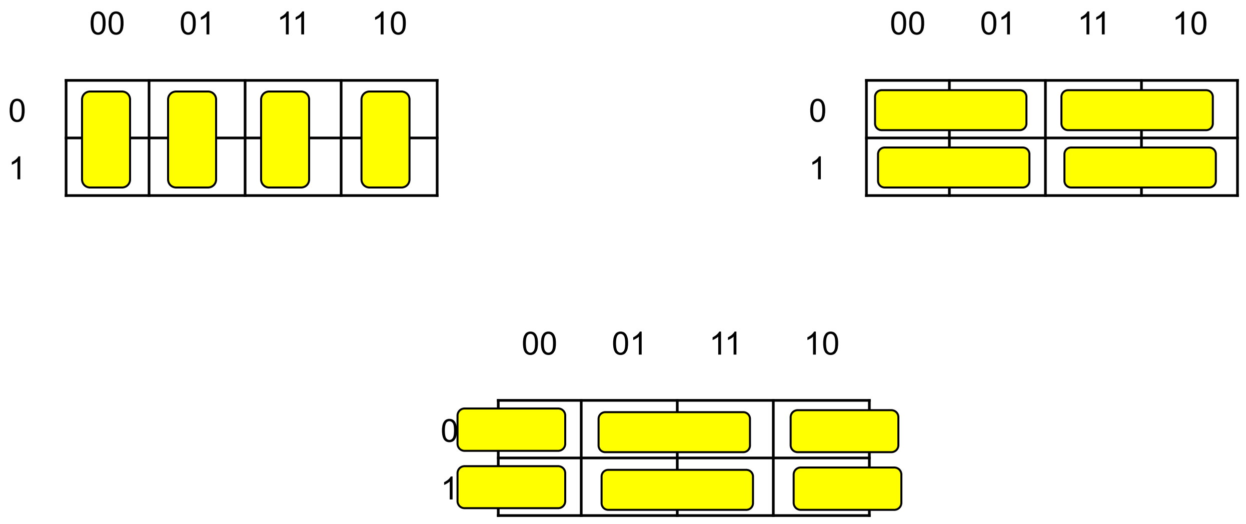
F = A' B' + C'



3 variable K-Maps (Adjacency)

A 3-variable map has 12 possible groups of 2 minterms

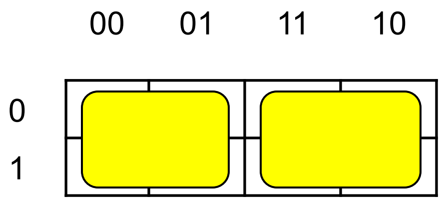
They become product terms with 2 literals

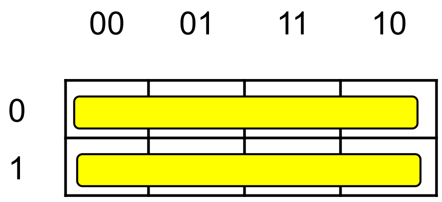


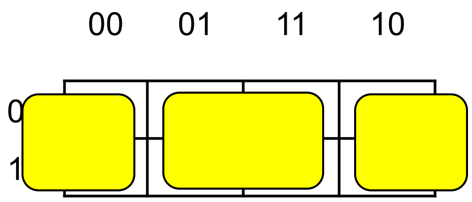
3 variable K-Maps (Adjacency)

A 3-variable map has 6 possible groups of 4 minterms

They become product terms with 1 literals







4-variable K-maps

A 4-variable function will consist of 16 minterms and therefore a size 16 k-map is needed

Each square is adjacent to 4 other squares

A square by itself will represent a minterm with 4 literals

Combining 2 squares will generate a 3-literal output

Combining 4 squares will generate a 2-literal output

Combining 8 squares will generate a 1-literal output

C

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

B

A

D

4-variable K-maps (Adjacency)

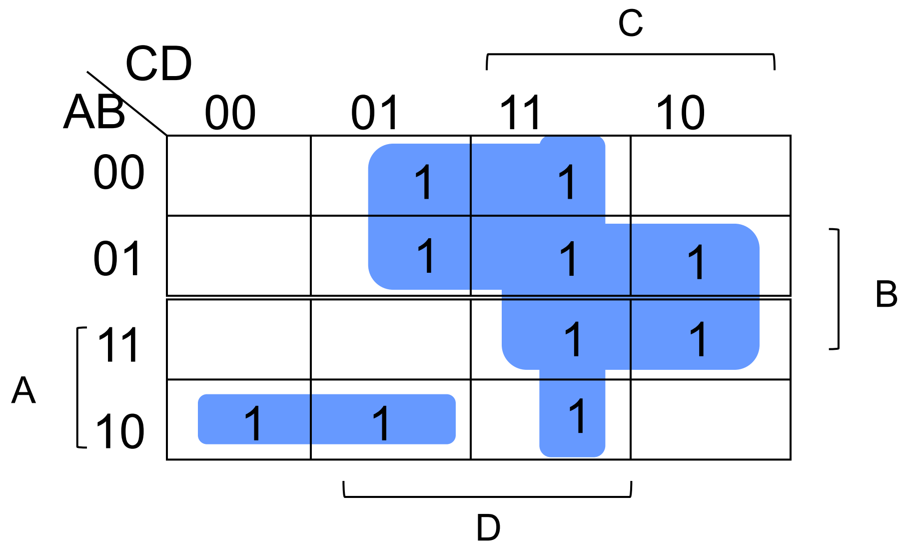
AB \ CD	00	01	11	10
00	m_0	m_1	m_3	m_2
01	m_4	m_5	m_7	m_6
11	m_{12}	m_{13}	m_{15}	m_{14}
10	m_8	m_9	m_{11}	m_{10}

Note: You can only combine a power of 2 adjacent 1-squares. For e.g. 2, 4, 8, 16 squares. You cannot combine 3, 7 or 5 squares

- Right column and left column are adjacent; can be combined
- Top row and bottom column are adjacent; can be combined
- Many possible 2, 4, 8 groupings

Example

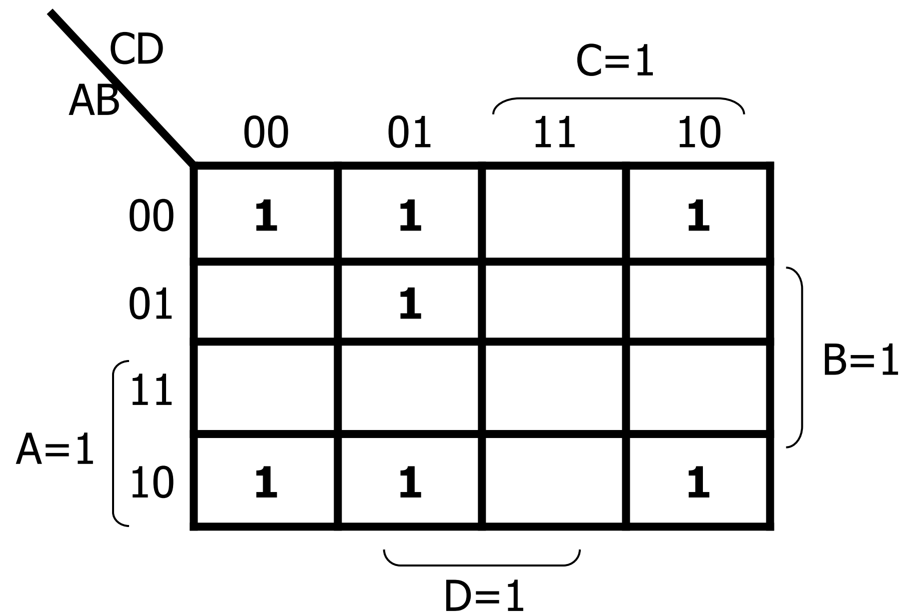
Minimize the function F(A,B,C,D)=∑m(1,3,5,6,7,8,9,11,14,15)



F = CD + A'D + BC + AB'C'

Example

F(A,B,C,D) = Σm(0,1,2,5,8,9,10)

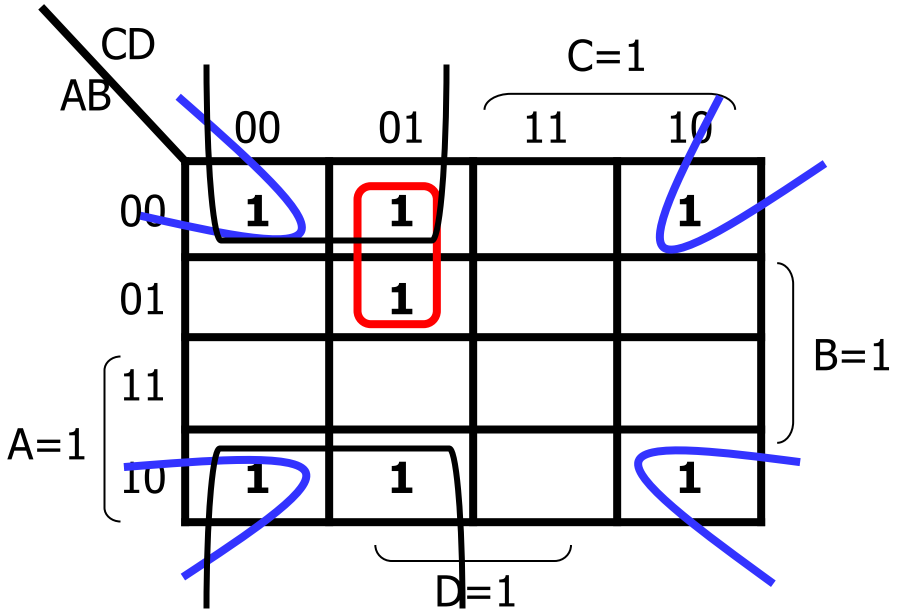


Example

F(A,B,C,D) = Σm(0,1,2,5,8,9,10)

Solution:

F = B’ D’ + B’ C’ + A’ C’ D



Using (POS)

F(A,B,C,D) = Σm(0,1,2,5,8,9,10)

Write F in the simplified product of sums (POS) not (SOP)

Two methods?

You already know one!

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

C=1 (columns 11, 10); D=1 (columns 01, 11); B=1 (rows 01, 11); A=1 (rows 11, 10)

Using (POS)

F(A,B,C,D) = Σm(0,1,2,5,8,9,10)

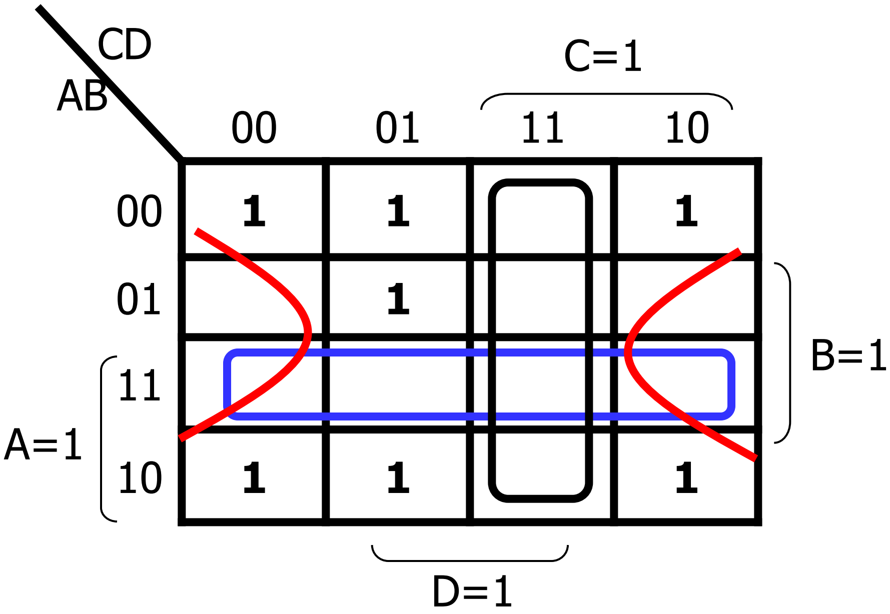
Write F in the simplified product of sums (POS) not (SOP)

- Follow same rule as before but for the ZEROs

F' = AB + CD + BD'

- Therefore,

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



Don't Cares

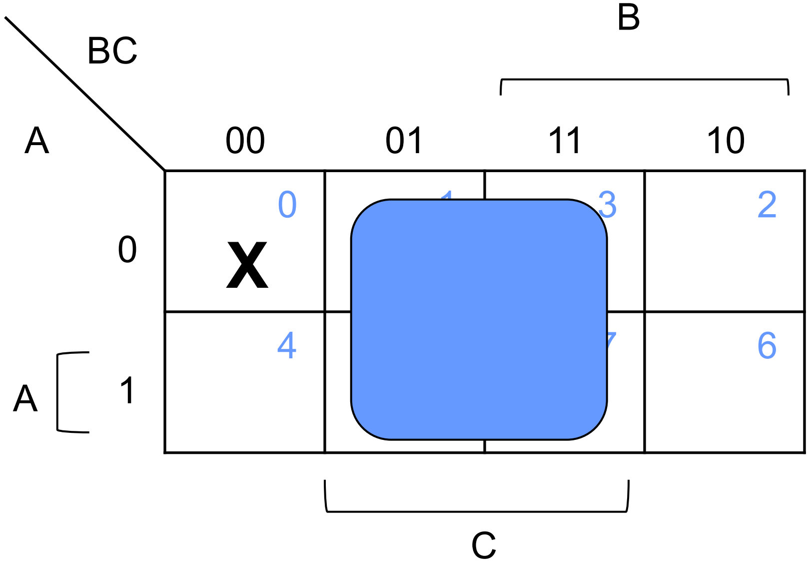
- In some cases, the output of the function (1 or 0) is not specified for certain input combinations either because
  - The input combination never occurs (Example BCD codes), or
  - We don't care about the output of this particular combination
- Such functions are called incompletely specified functions
- Unspecified minterms for these functions are called don't cares
- While minimizing a k-map with don't care minterms, their values can be selected to be either (1 or 0) depending on what is needed for achieving a minimized output.

Example

$F = \sum m(1, 3, 7) + \sum d(0, 5)$

Circle the x's that help get bigger groups of 1's (or 0's if POS).

Don't circle the x's that don't help.

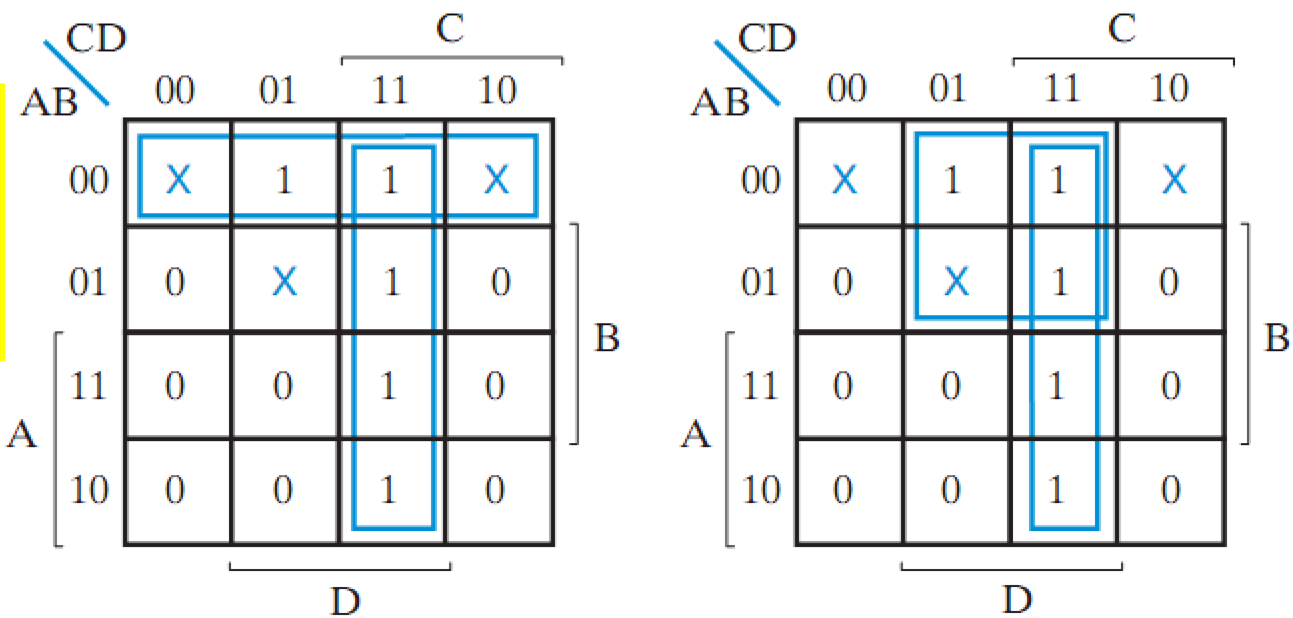


F = C

Example 2

$F(A, B, C, D) = \sum m(1, 3, 7, 11, 15) + \sum d(0, 2, 5)$

- Two possible solutions!
- Both acceptable.
- All 1's covered



(a) $F = CD + \overline{A}\,\overline{B}$ (b) $F = CD + \overline{A}D$

Src: Mano's Textbook

5-variable K-maps

A=0

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

A=1

BC \ **DE**	00	01	11	10
00	m_{16}	m_{17}	m_{19}	m_{18}
01	m_{20}	m_{21}	m_{23}	m_{22}
11	m_{28}	m_{29}	m_{31}	m_{30}
10	m_{24}	m_{25}	m_{27}	m_{26}

- 32 minterms require 32 squares in the k-map
- Minterms 0-15 belong to the squares with variable A=0, and minterms 16-32 belong to the squares with variable A=1
- Each square in A' is also adjacent to a square in A (one is above the other)
- Minterm 4 is adjacent to 20, and minterm 15 is to 31

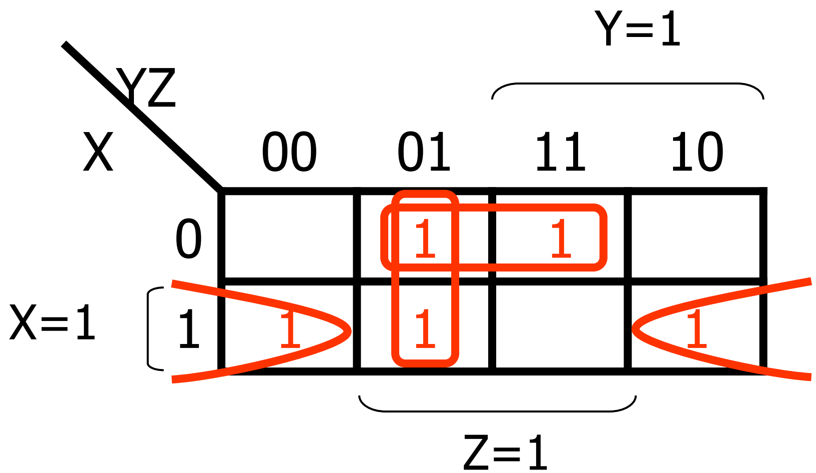
Finding the minimum SOP

Definitions

- An **implicant** is a product term of a function
  - Any group of 1's in a K-Map
- A **prime implicant** is a product term obtained by combining the maximum possible number of adjacent 1's in a k-map
  - Biggest groups of 1's
  - Not all prime implicants are needed!
- If a minterm is covered by exactly one prime implicant then this prime implicant is called an **essential prime implicant**

Example

Consider F(X,Y,Z) = Σm(1,3,4,5,6)



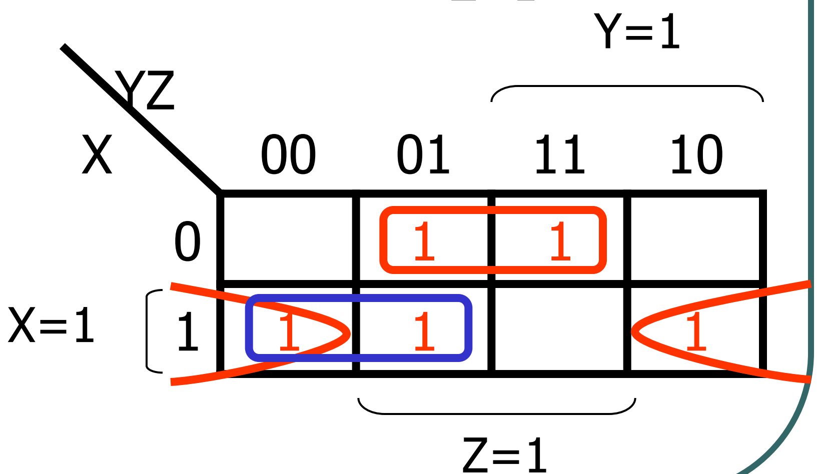
List all implicants, prime implicants and essential prime implicants

Solution:

Implicants: XY'Z', XZ', XY', XY'Z, X'Y'Z, Y'Z, ………

Prime Implicants: XY', XZ', Y'Z, X'Z

EPIs: X'Z, XZ'



The simplest expression is NOT unique!

Finding minimum SOP

1. Find each essential prime implicant and include it in the solution
2. If any minterms are not yet covered, find minimum number of prime implicants to cover them (minimize overlap).

Example 2

Simplify F(A, B, C, D) = ∑ m(0, 1, 2, 4, 5, 10,11,13, 15)

Note:

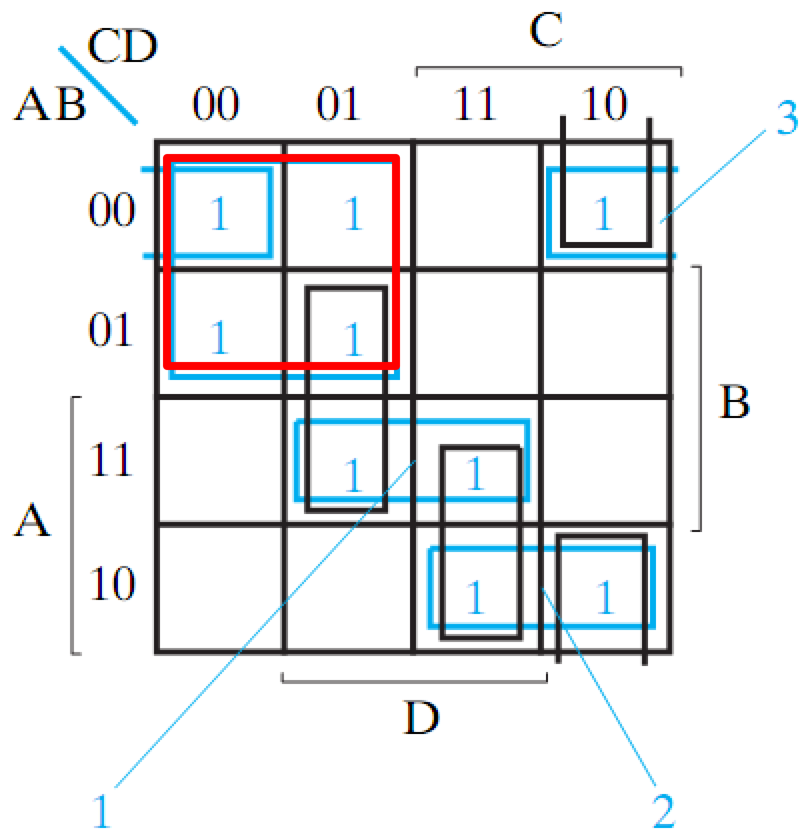
-Only A'C' is E.P.I

-For the remaining minterms:

-Choose 1 and 2 (minimize overlap)

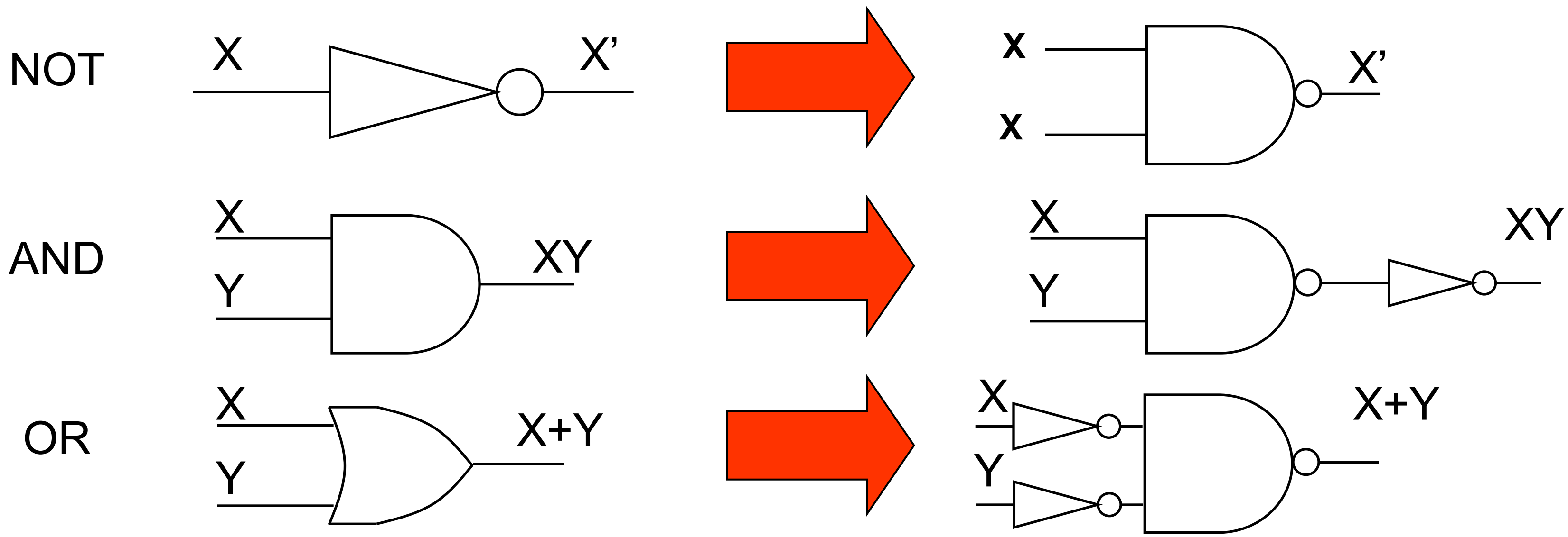
-For m_2, choose either A'B'D' or B'CD'

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



Src: Mano's Textbook

NAND Gate is Universal

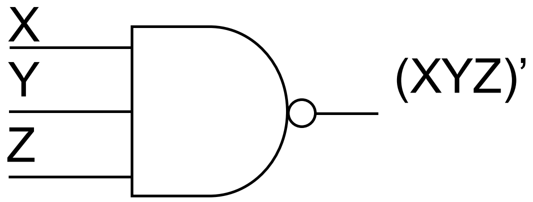


- Therefore, we can build all functions we learned so far using NAND gates ONLY *(Exercise: Prove that NOT can be built with NAND)*
- NAND is a UNIVERSAL gate

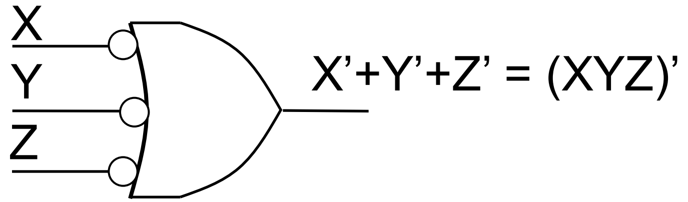
Graphic Symbols for NAND Gate

Two equivalent graphic symbols or shapes for the SAME function

AND-NOT



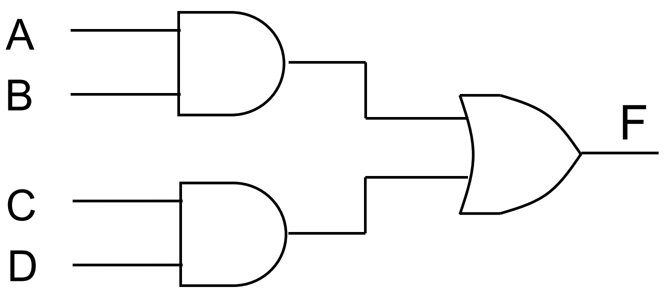
NOT-OR

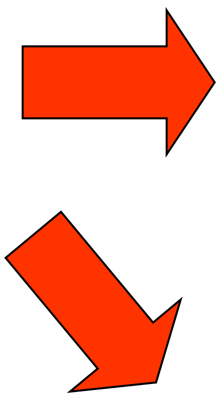


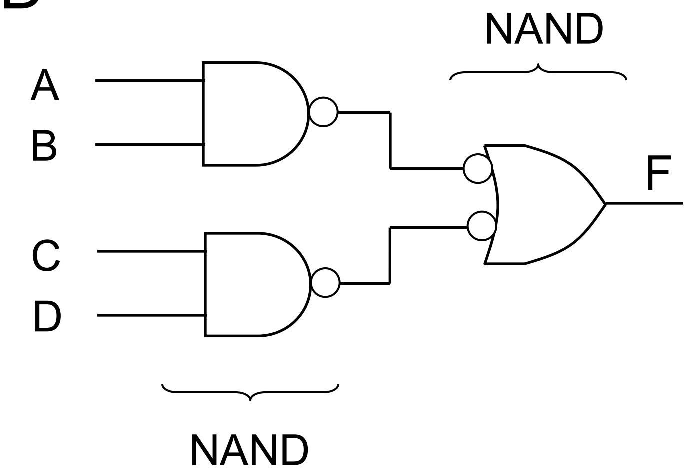
AND-NOT = NOT-OR

Implementation using NANDs

Example: Consider F = AB + CD





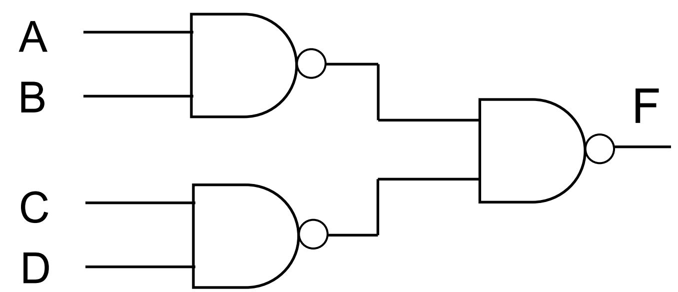


Proof:

F = F′′= ((AB)′.(CD)′)′

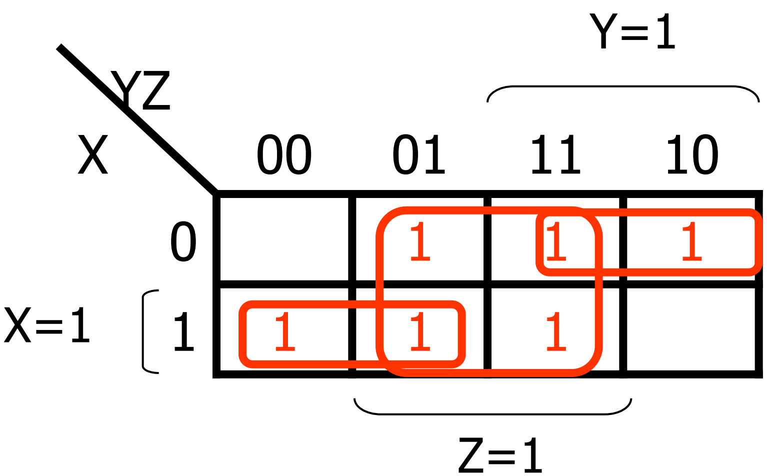
= ((AB)′)′ + ((CD)′)′

= AB + CD

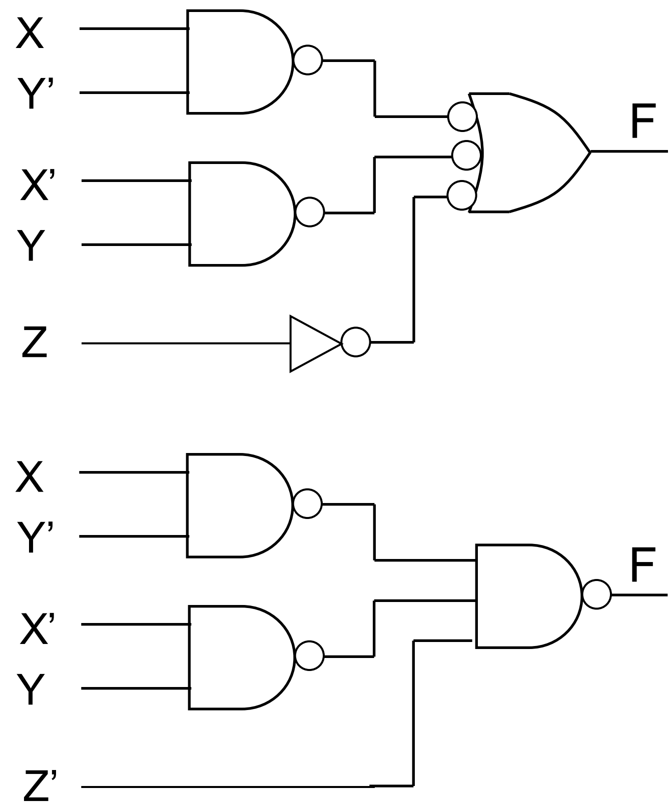


Implementation using NANDs

Consider F =Σm(1,2,3,4,5,7) – Implement using NAND gates



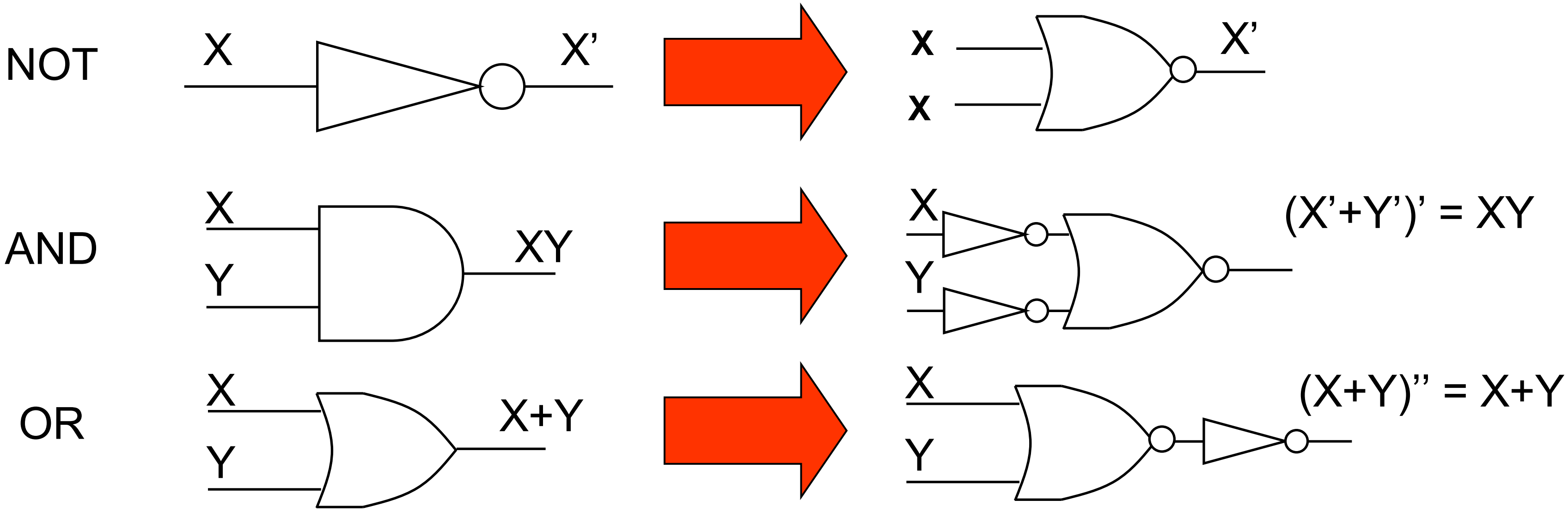
F(X,Y) = Z+XY′+X′Y



Rules for 2-Level NAND Implementations

1. Simplify the function and express it in <u>sum-of-products form</u>
2. Draw a NAND gate for each product term (with 2 literals or more)
3. Draw a single NAND gate at the 2nd level (in place of the OR gate)
4. A term with single literal requires a NOT

NOR Gate is Universal

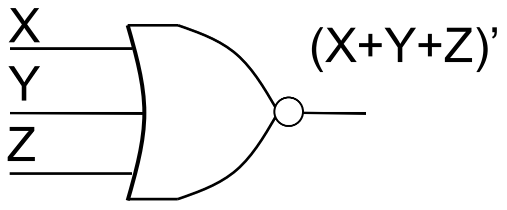


- Therefore, we can build all functions we learned so far using NOR gates ONLY *(Exercise: Prove that NOT can be built with NOR)*
- NOR is a UNIVERSAL gate

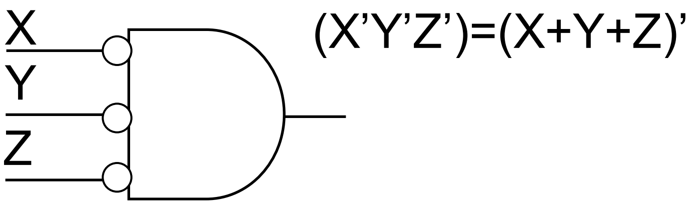
Graphic Symbols for NOR Gate

Two equivalent graphic symbols or shapes for the SAME function

OR-NOT



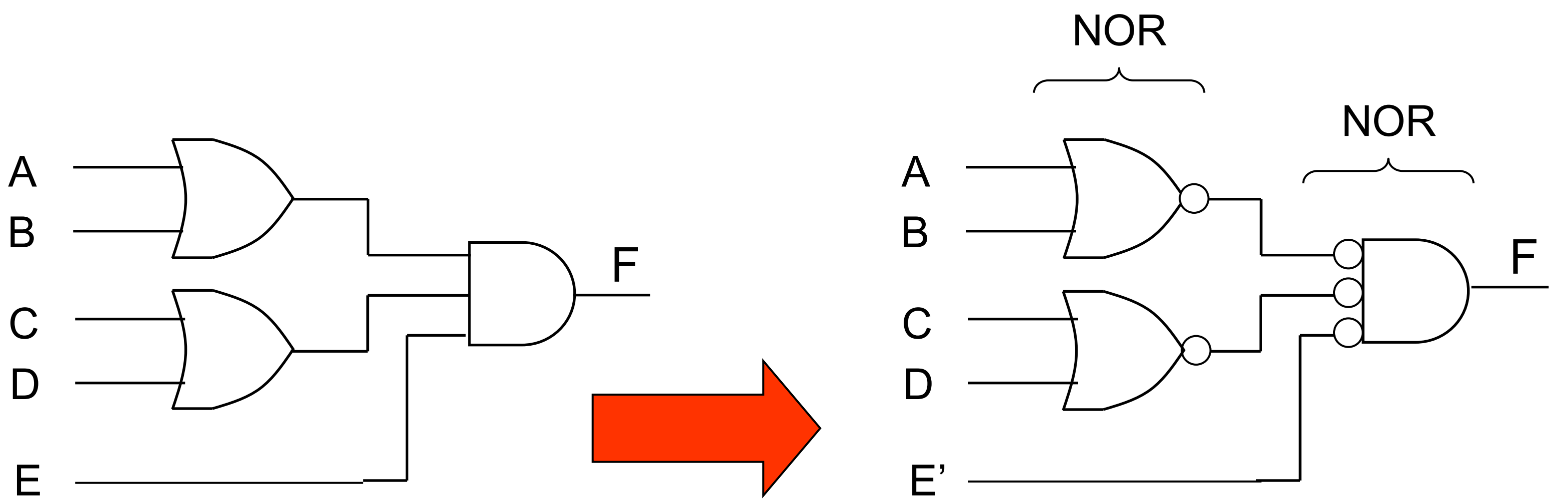
NOT-AND



OR-NOT = NOT-AND

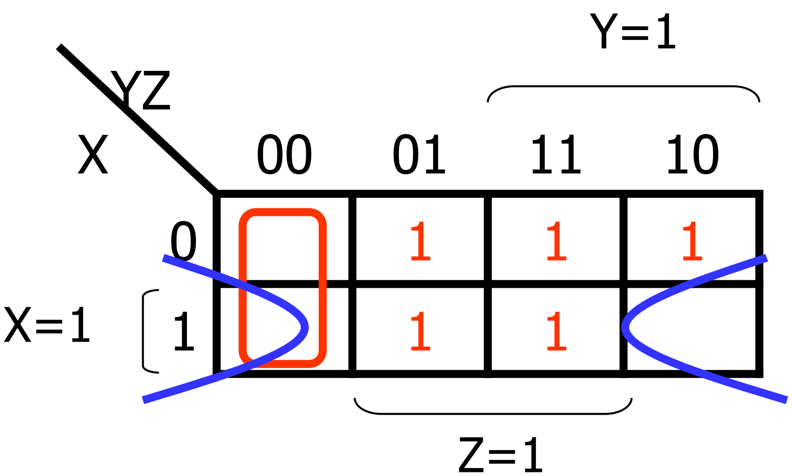
Implementation using NOR gates

Consider F = (A+B)(C+D)E

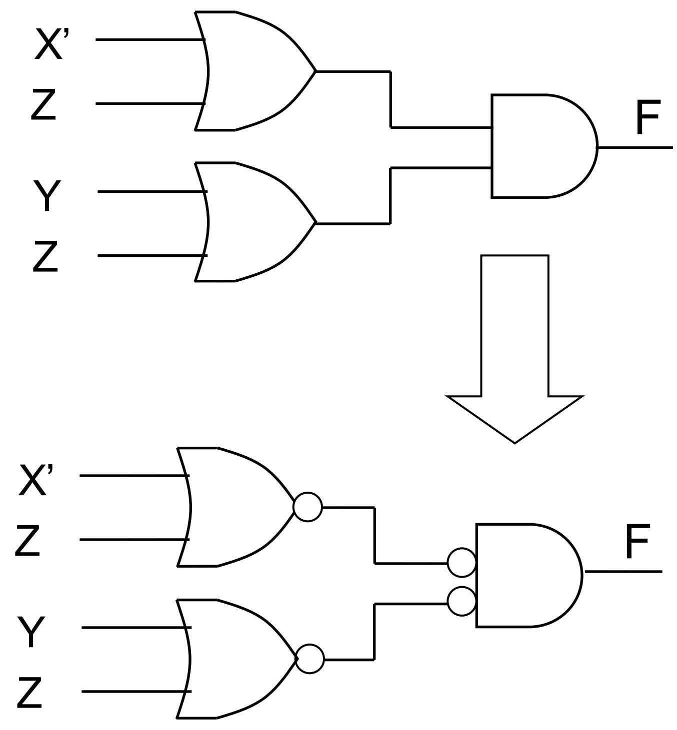


Implementation using NOR gates

Consider F =Σm(1,2,3,5,7) – Implement using NOR gates



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



Rules for 2-Level NOR Implementations

1. Simplify the function and express it in product of sums form
2. Draw a NOR gate (using OR-NOT symbol) for each sum term (with 2 literals or more)
3. Draw a single NOR gate (using NOT-AND symbol) the 2nd level (in place of the AND gate)
4. A term with single literal requires a NOT