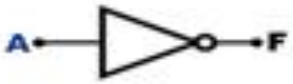
Lecture 4

- SOP
- POS
- K- MAP

AND  	A B F 0 0 0 0 1 0 1 0 0 1 1 1	74LS08 	F=AB
OR  	A B F 0 0 0 0 1 1 1 0 1 1 1 1	74LS32 	F=A+B
NOT  	A F 0 1 1 0	74LS04 	$F=\overline{A}$

Synthesis

Definitions

- Variable
  - Represents a quantity (0 or 1); typically inputs
- Literal
  - Appearance of a variable (repetition included)
- Product Term
  - Product of literals
- Minterm
  - product term whose literals include every variable of the function exactly once in true or complemented form
- Sum-of-Products (SOP) form
  - ORing of product terms; abc + abc’
  - Note: (a + b)c is not in SOP form

F(a,b,c)=a’b’c+ab

Variables: a,b,c

Literal: a,a’,b,b’,c

Product: a’b’c, ab

Minterm: a’b’c

$\cancel{ab} \rightarrow abc$

$\cancel{ab} \rightarrow abc'$

Sum-of-Products: a’b’c+ab

Synthesis

Sum-of-Products Form

- List of minterms for a 3-variable table
  - Rows numbered to easily identify minterms
  - Notation m_i denotes minterm for a particular row

Row number	x1	x2	x3	Minterm
0	0	0	0	$m_0 = x1'x2'x3'$
1	0	0	1	$m_1 = x1'x2'x3$
2	0	1	0	$m_2 = x1'x2x3'$
3	0	1	1	$m_3 = x1'x2x3$
4	1	0	0	$m_4 = x1x2'x3'$
5	1	0	1	$m_5 = x1x2'x3$
6	1	1	0	$m_6 = x1x2x3'$
7	1	1	1	$m_7 = x1x2x3$

Sum-of-Products Form

- Canonical sum-of-products form
  - If each product in sum-of-product form is a minterm
- Properties of canonical form
  - Exactly one canonical form
  - Objects that have the same canonical form are the same

Sum-of-Products Form

- Derive sum-of-product form for the following truth table

Row number	x1	x2	x3	F	Minterm
0	0	0	0	0	m_0 = x1′x2′x3′
1	0	0	1	1	m_1 = x1′x2′x3
2	0	1	0	0	m_2 = x1′x2x3′
3	0	1	1	0	m_3 = x1′x2x3
4	1	0	0	1	m_4 = x1x2′x3′
5	1	0	1	1	m_5 = x1x2′x3
6	1	1	0	1	m_6 = x1x2x3′
7	1	1	1	0	m_7 = x1x2x3

F = m1 · 1 + m4 · 1 + m5 ·1 + m6 · 1

F = m1 + m4 + m5 + m6

F = x1′x2′x3 + x1x2′x3′ + x1x2′x3 + x1x2x3′

F(x1, x2, x3) = Σ(m1, m4 , m5 , m6)

F(x1, x2,x3) = Σ(1, 4, 5, 6)

Maxterm

- Maxterms
  - Complement of minterm
    - m0′ = M0
  - DeMorgan’s Theorem – (xy)′ = x′ + y′
- List of maxterms for 3-input table listed below
  - Notation Mi denotes maxterm for a particular row

Row number	x1	x2	x3	Minterm	Maxterm
0	0	0	0	$m_0 = x1'x2'x3'$	$M_0 = x1 + x2 + x3$
1	0	0	1	$m_1 = x1'x2'x3$	$M_1 = x1 + x2 + x3'$
2	0	1	0	$m_2 = x1'x2x3'$	$M_2 = x1 + x2' + x3$
3	0	1	1	$m_3 = x1'x2x3$	$M_3 = x1 + x2' + x3'$
4	1	0	0	$m_4 = x1x2'x3'$	$M_4 = x1' + x2 + x3$
5	1	0	1	$m_5 = x1x2'x3$	$M_5 = x1' + x2 + x3'$
6	1	1	0	$m_6 = x1x2x3'$	$M_6 = x1' + x2' + x3$
7	1	1	1	$m_7 = x1x2x3$	$M_7 = x1' + x2' + x3'$

Product-of-Sum Form

- Product-of-sum form
  - Function can be represented as maxterms with corresponding value of f=0 ANDed together

Row number	x1	x2	F	Maxterm
0	0	0	1	$M_0 = x1 + x2$
1	0	1	0	$M_1 = x1 + x2'$
2	1	0	0	$M_2 = x1' + x2$
3	1	1	1	$M_3 = x1' + x2'$

$$F = M_1 \cdot M_2$$

$$F = (x1 + x2')(x1' + x2)$$

- More concise form uses row-number subscripts to specify a given function
  - Π symbol denotes logical sum

$$F(x1, x2) = \Pi(M_1, M_2)$$

$$F(x1, x2) = \Pi(1, 2)$$

Product-of-Sum Form

- Derive product-of-sum form for the following truth table

Row number	x1	x2	x3	F	Maxterm
0	0	0	0	0	$M_0 = x1 + x2 + x3$
1	0	0	1	1	$M_1 = x1 + x2 + x3'$
2	0	1	0	0	$M_2 = x1 + x2' + x3$
3	0	1	1	0	$M_3 = x1 + x2' + x3'$
4	1	0	0	1	$M_4 = x1' + x2 + x3$
5	1	0	1	1	$M_5 = x1' + x2 + x3'$
6	1	1	0	1	$M_6 = x1' + x2' + x3$
7	1	1	1	0	$M_7 = x1' + x2' + x3'$

$$F = M_0 \cdot M_2 \cdot M_3 \cdot M_7$$

$$F = (x1 + x2 + x3)(x1 + x2' + x3)$$
$$(x1 + x2' + x3')(x1' + x2' + x3')$$

$$F(x1, x2, x3) = \prod(M_0, M_2, M_3, M_7)$$
$$F(x1, x2, x3) = \prod(0,2,3,7)$$

Product-of-Sum Form

Is sum-of-products and product-of-sums really equivalent?

Row number	x1	x2	F	Minterm	Maxterm
0	0	0	0	$m_0 = x1'x2'$	$M_0 = x1 + x2$
1	0	1	0	$m_1 = x1'x2$	$M_1 = x1 + x2'$
2	1	0	1	$m_2 = x1x2'$	$M_2 = x1' + x2$
3	1	1	1	$m_3 = x1x2$	$M_3 = x1' + x2'$

According to our defenitions…

Sum-of-products

$F = m_2 + m_3$

$F = (x1x2') + (x1x2)$

Product-of-sums

$G = M_0 \cdot M_1$

$G = (x1 + x2)(x1 + x2')$

Complement of a function indicates when output is 0

$F' = m_0 + m_1$

$F' = (x1'x2') + (x1'x2)$

Complement of F′ should then indicate when output is 1

$F'' = ((x1'x2') + (x1'x2))'$

$F'' = (x1'x2')' (x1'x2)'$

$F'' = (x1 + x2)(x1 + x2')$

Matches

Remember DeMorgan′s Theorem

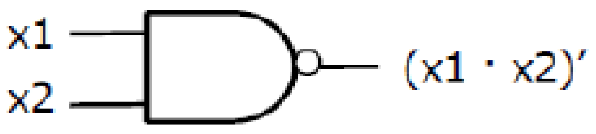
15a. $(x \cdot y)' = x' + y'$

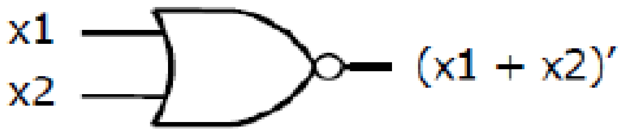
15b. $(x + y)' = x' \cdot y'$

NAND and NOR Logic Networks

NAND and NOR Truth Table

- NAND and NOR gate truth tables
- NAND and NOR gates are popular because the underlying implementation is simpler (we'll see in Chapter 3)





NAND

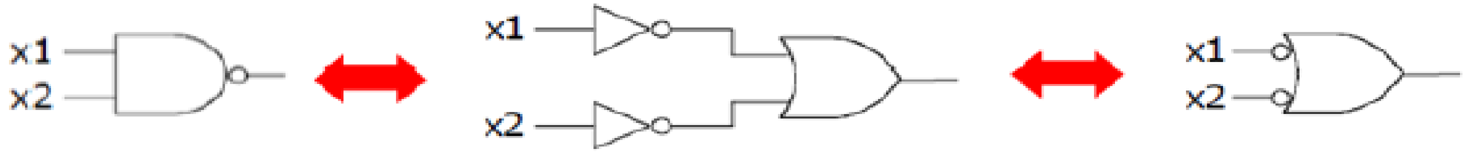
a	b	F
0	0	1
0	1	1
1	0	1
1	1	0

NOR

a	b	F
0	0	1
0	1	0
1	0	0
1	1	0

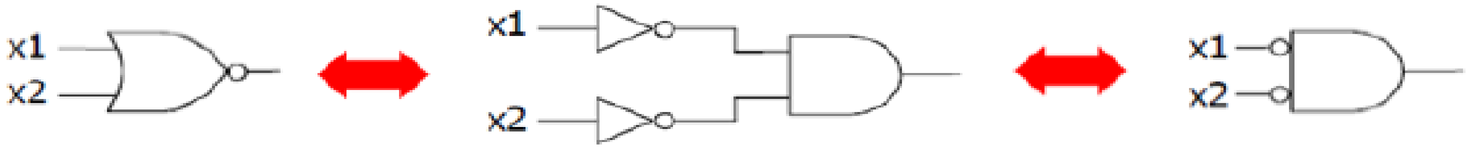
NAND and NOR Logic Networks
DeMorgan Theorem as Logic Circuits

- Can we use NAND and NOR gates to implement various logic circuits?
  - Let's look at logic gate implementation of DeMorgan's Theorem



NAND of two variables the same as complementing variables first, then ORing them

15a. (x1x2)'=x1'+x2'



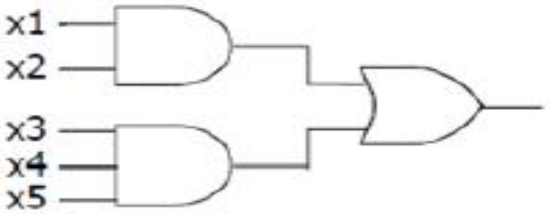
NOR of two variables: the same as complementing variables first, then ANDing them

15b. (x1+x2)'=x1'x2'

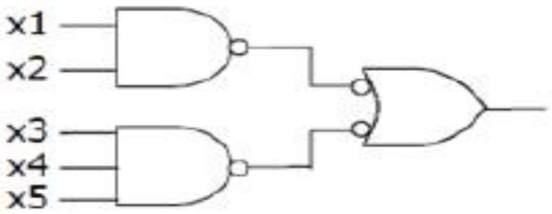
Note that the NOT gates are represented as inversion bubbles

NAND and NOR Logic Networks
DeMorgan Theorem as Logic Circuits

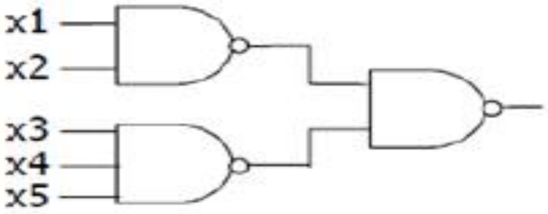
- Any function can be implemented in sum-of-products form
  - We can transform into a network using only NAND gates



general sum-of-products form



connections between AND and OR gates includes 2 INV gates –functionally equivalent 9. x'' = x

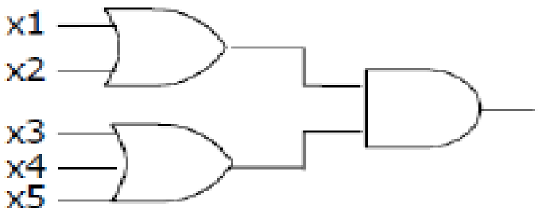


Network can be transformed into a network of NAND gates

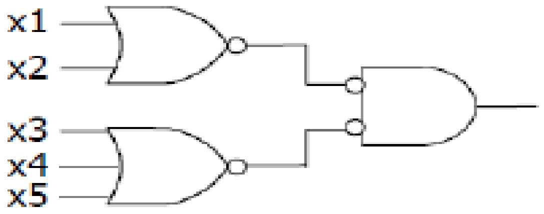
ust showed this equivalency – DeMorgan's Theorem

NAND and NOR Logic Networks
DeMorgan Theorem as Logic Circuits

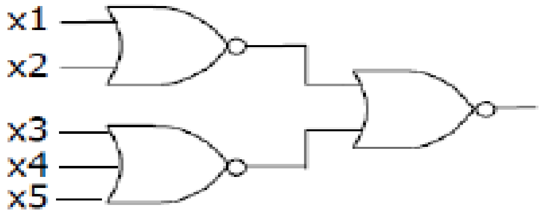
- Any function can be implemented in product-of-sums form
- We can transform into a network using only NOR gates



general product-of-sums form

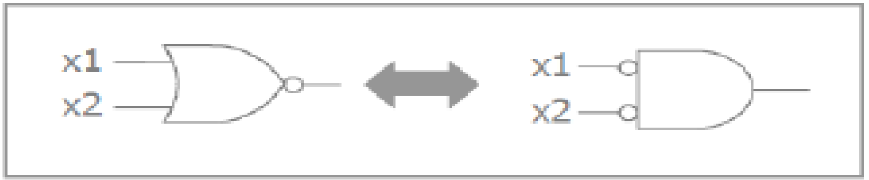


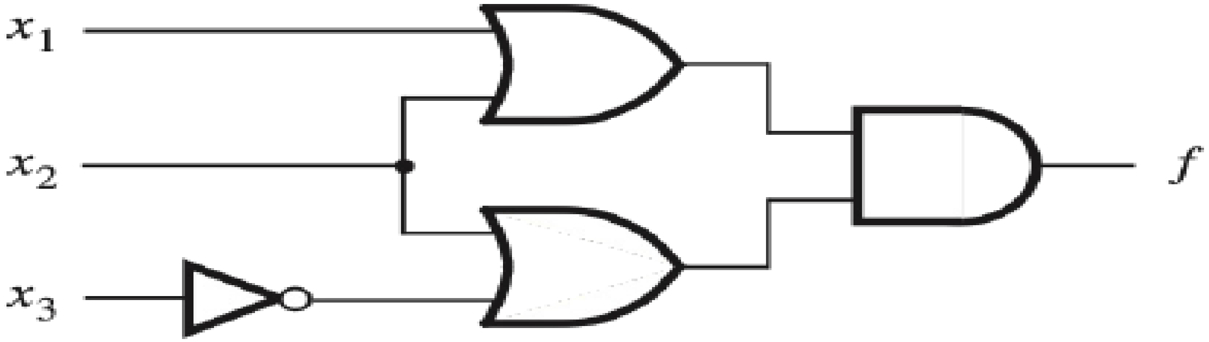
connections between OR and AND gates includes 2 INV gates –functionally equivalent
9. x'' = x



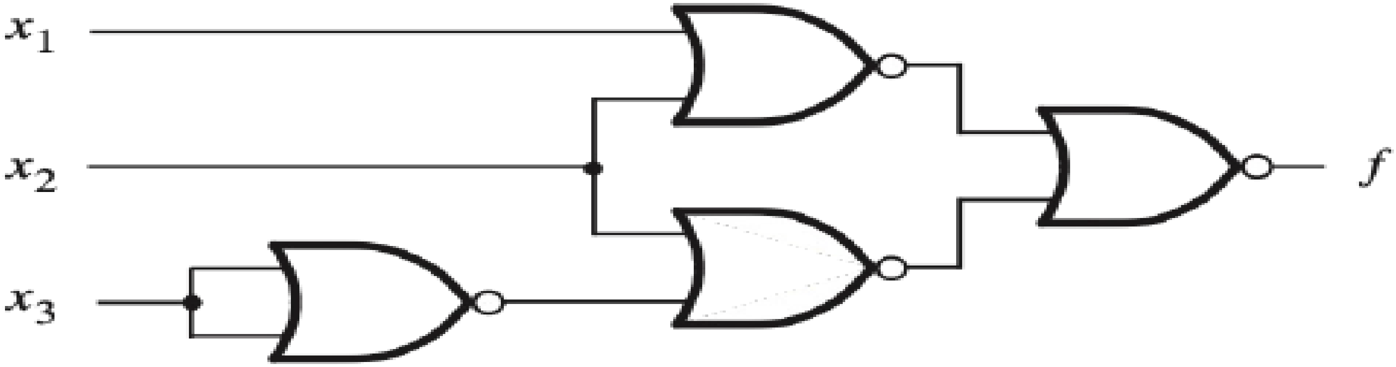
Network can be transformed into a network of NOR gates

Just showed this equivalency – DeMorgan's Theorem





(a) POS implementation



(b) NOR implementation

Figure 2.29 NOR-gate realization of the function in Example 2.13.

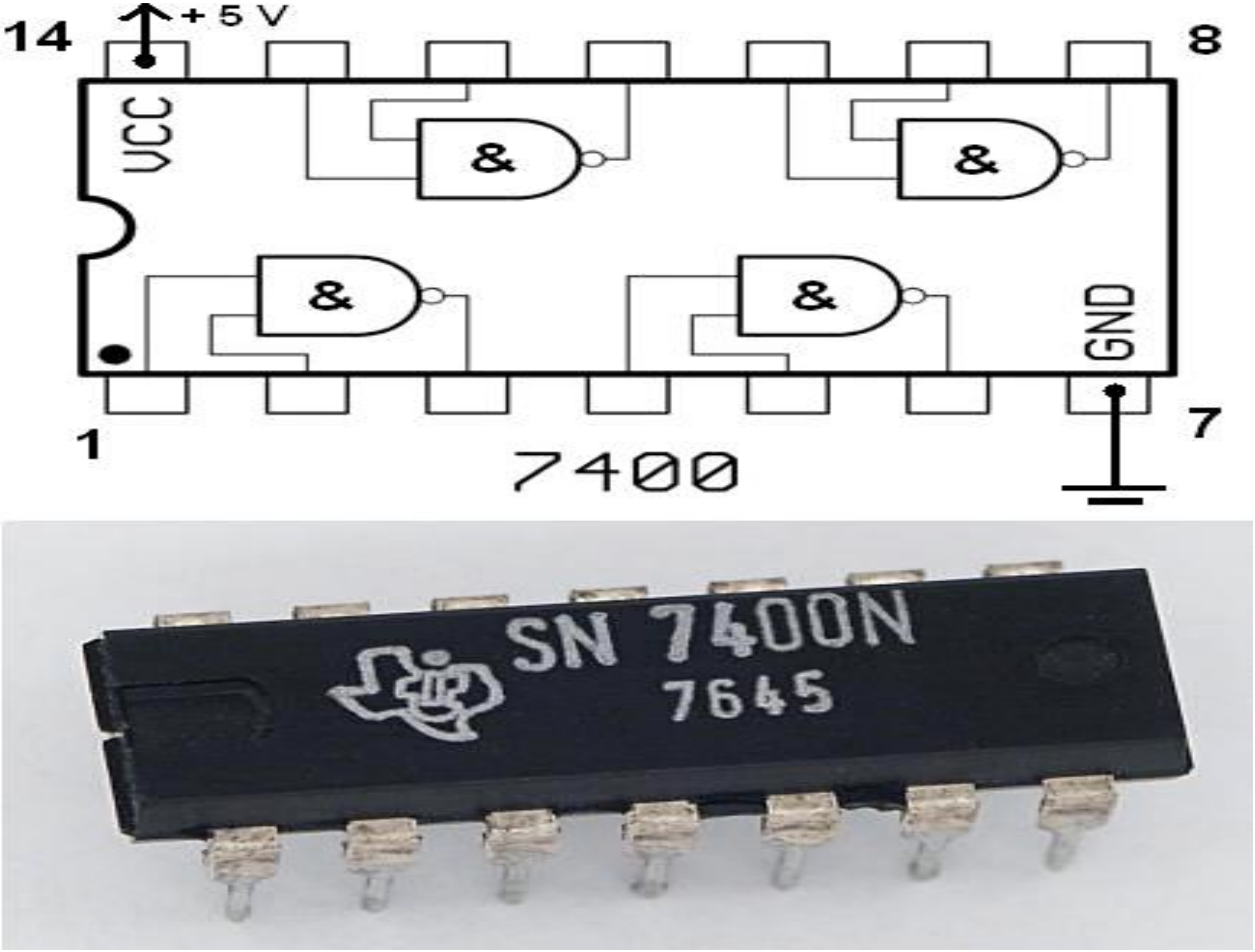
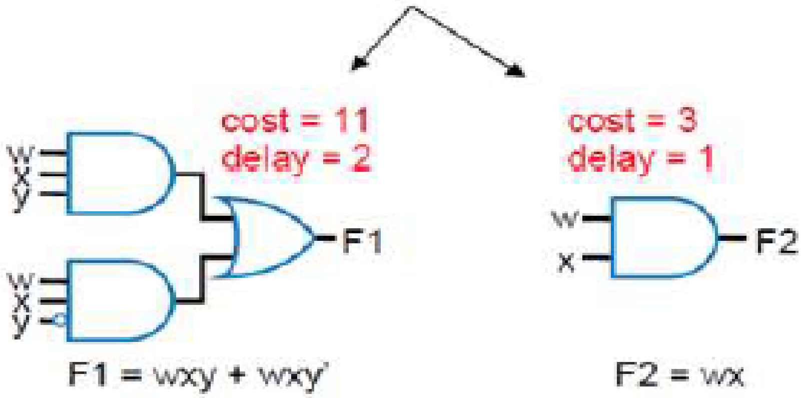


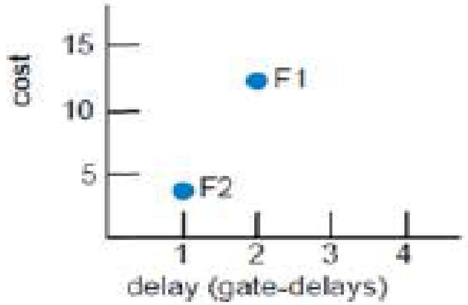
Figure 2.30 NAND-gate realization of the function in Example 2.10.

Introduction to Optimization and Trade-off

- We now know how to build digital circuits
  - How can we build better circuits?
- Let's consider two important design criteria
  - Delay – the time from inputs changing to new correct stable output
    - Every gate has delay of "1 gate-delay"
    - Ignore inverters
  - Size – the number of transistors
    - Every circuit has cost
      = number of gates + number of gate inputs
    - Ignore inverters

$F = wx(y+y') = wx$

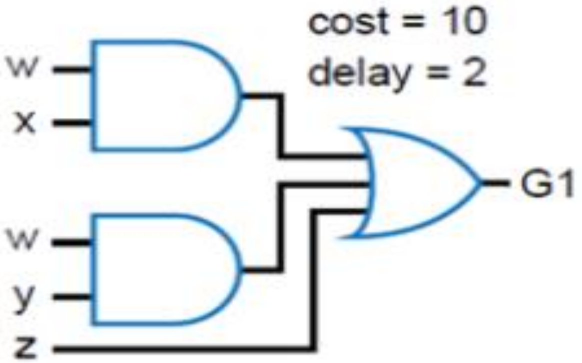




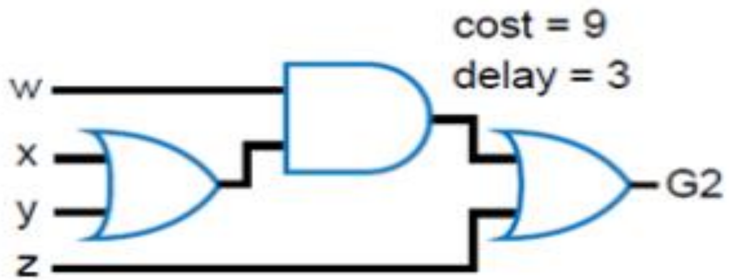
Transforming F1 to F2 represents an ***optimization***: Better in all criteria of interest

Introduction to Optimization and Trade-off

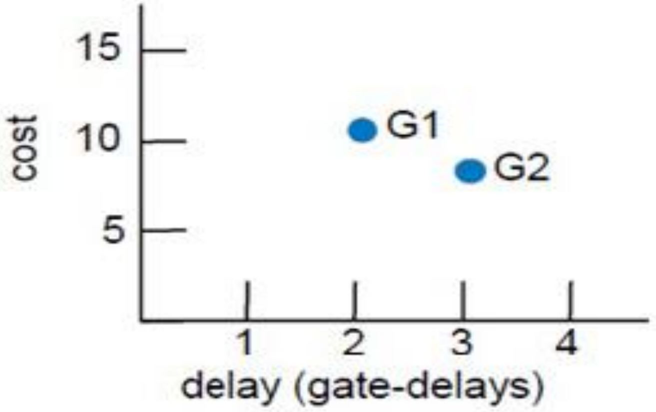
- Trade-off
  - Improves some, but worsens other, criteria of interest



G1 = wx + wy + z



G2 = w(x+y) + z



Transforming G1 to G2 represents a ***tradeoff***: Some criteria better, others worse.

Optimization through Algebraic Manipulation

- Algebraic manipulation
  - Multiply out to sum-of-products, then, apply following as much possible
    - $ab + ab' = a(b + b') = a*1 = a$
    - Combining terms to eliminate a variable
    - (Formally called the "Uniting theorem")
  - Duplicating a term sometimes helps
    - Note that doesn't change function
    - $c + d = c + d + d = c + d + d + d + d \ldots$

$$F = xyz + xyz' + x'y'z' + x'y'z$$
$$F = xy(z + z') + x'y'(z + z')$$
$$F = xy*1 + x'y'*1$$
$$F = xy + x'y'$$

$$F = x'y'z' + x'y'z + x'yz$$
$$F = x'y'z' + x'y'z + x'y'z + x'yz$$
$$F = x'y'(z+z') + x'z(y'+y)$$
$$F = x'y' + x'z$$

- Algebraic Manipulation
  - Which "rules" to use and when?
  - Easy to miss "seeing" possible opportunities to combine terms

- K-map contains 2^n cells (or squares) for n variables.
- Each cell in K-map corresponds to one row of the Truth Table and one minterm of the canonical Boolean function.
- Cells are identified by labelling the edges of the map in such a way as to give a unique combination of variables & their inverses for each cell.

General K-map Method

General K-map method

1. Convert the function's equation into sum-of-products form (or truth table)
2. Place 1s in the appropriate K-map cells for each term
3. Cover all 1s by drawing the fewest largest circles, with every 1 included at least once; write the corresponding term for each circle
4. OR all the resulting terms to create the minimized function.

Generalized two Variables K-map

Two Variable K-map

x_1	x_2	
0	0	m_0
0	1	m_1
1	0	m_2
1	1	m_3

(a) Truth table

$x_2 \backslash x_1$	0	1
0	m_0	m_2
1	m_1	m_3

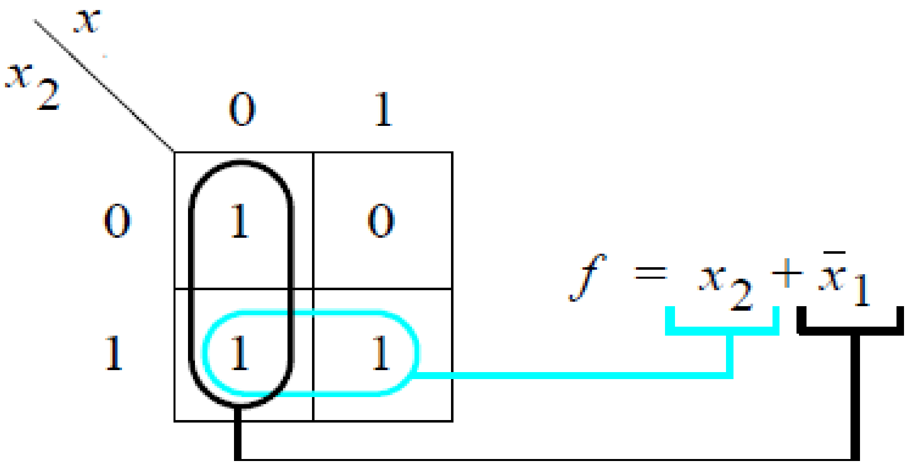
(b) Karnaugh map

Two Variables K-map -Example

- Fill in each cell with corresponding value of F
- Draw circles around adjacent 1's
  - Groups of 1, 2 or 4
- Circle indicates optimization opportunity
  - We can remove a variable
- To obtain function OR all product terms contained in circles
  - Make sure all 1's are in at least one circle

x_1	x_2	f
0	0	1
0	1	1
1	0	0
1	1	1

Truth table



Generalized Three Variables K-map

Three Variable K-map

x_1	x_2	x_3	
0	0	0	m_0
0	0	1	m_1
0	1	0	m_2
0	1	1	m_3
1	0	0	m_4
1	0	1	m_5
1	1	0	m_6
1	1	1	m_7

(a) Truth table

x_3 \ x_1x_2	00	01	11	10
0	m_0	m_2	m_6	m_4
1	m_1	m_3	m_7	m_5

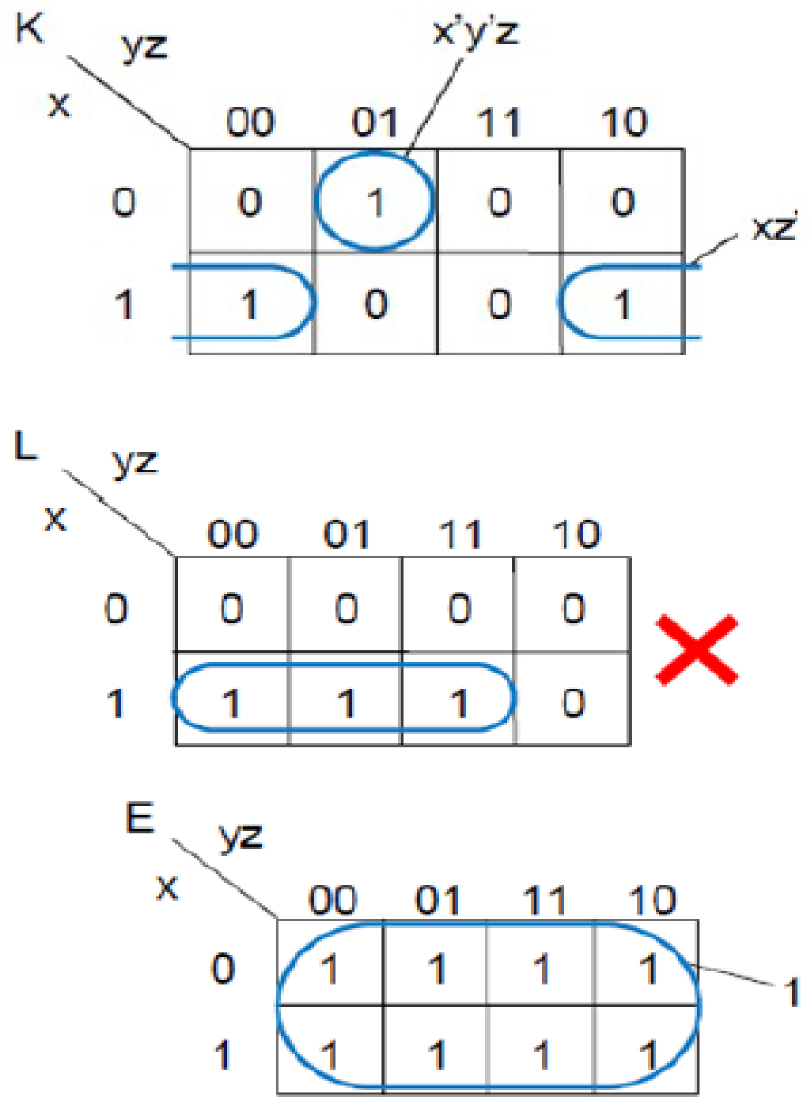
(b) Karnaugh map

REMEMBER: K-map graphically place minterms next to each other when they differ by one variable

- m2 cannot be placed next to m4 (x1’x2x3’, x1x2’x3’)
- m2 can be placed next to m6 (x1’x2x3’, x1x2x3’)
- m6 can be placed next to m4 (x1x2x3’, x1x2’x3’)

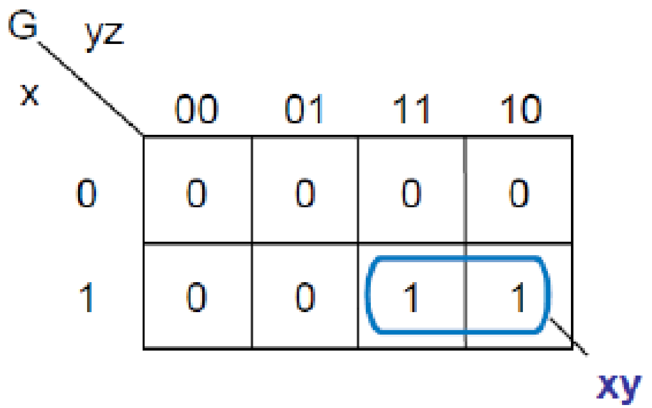
Generalized Three Variables K-map

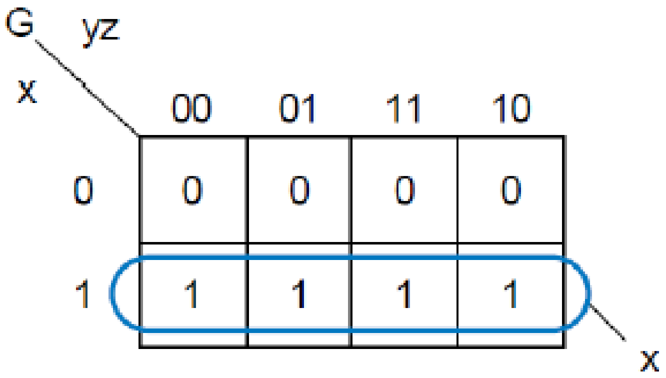
- Circles can cross left/right sides
  - Remember, edges are adjacent
  - Minterms differ in one variable only
- Circles must have 1, 2, 4, or 8 cells – 3,5, or 7 not allowed
  - 3/5/7 doesn't correspond to algebraic transformations that combine terms to eliminate a variable
- Circling all the cells is OK
  - Function just equals 1



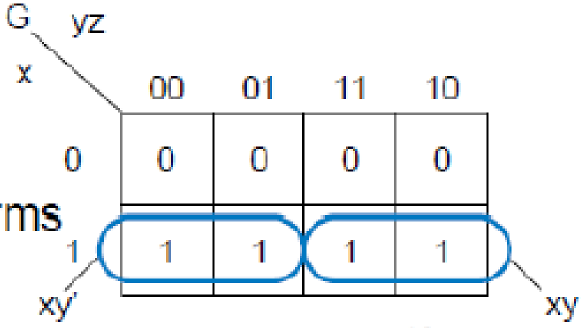
Generalized Three Variables K-map

- Two adjacent 1s means one variables can be eliminated
  - Same as in two-variable K-maps
- Four adjacent 1s means two variables can be eliminated
  - Makes intuitive sense – those two variables appear in all combinations, so one must be true
  - Draw one big circle –shorthand for the algebraic transformations above





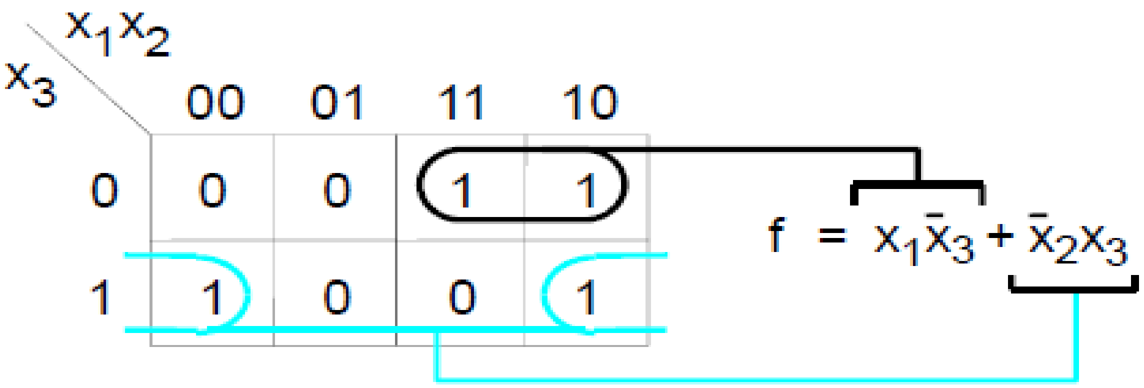
Draw the biggest circle possible, or you'll have more terms than really needed



Three Variables K-map – example -1

Let us try an example

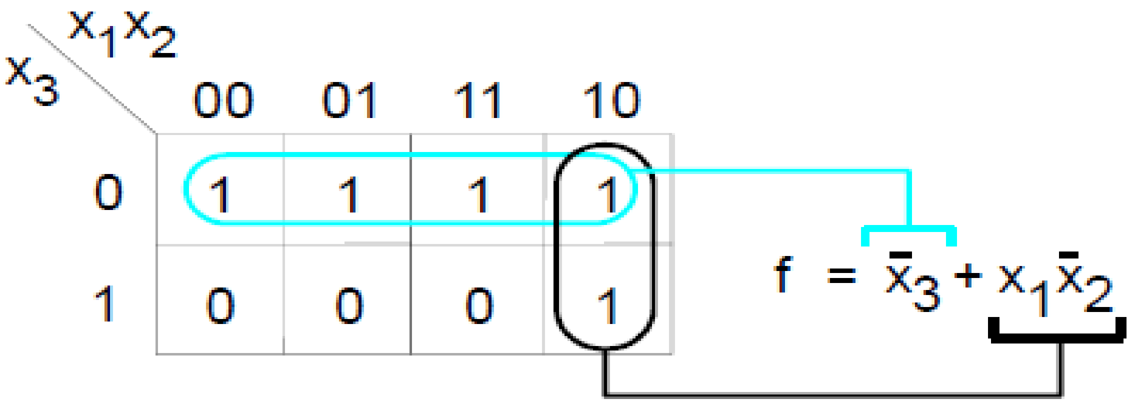
Row number	x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0



Three Variables K-map – Example -2

Let us try another example

Row number	x_1	x_2	x_3	f
0	0	0	0	1
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0



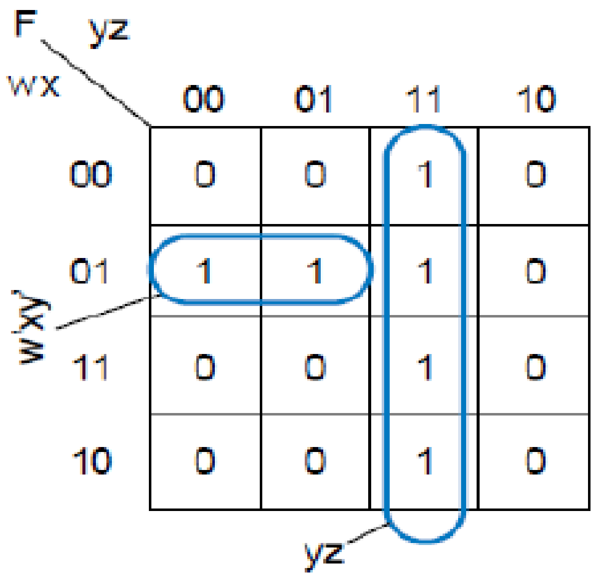
Generalized four Variables K-map

x_1	x_2	x_3	x_4	
0	0	0	0	m_0
0	0	0	1	m_1
0	0	1	0	m_2
0	0	1	1	m_3
0	1	0	0	m_4
0	1	0	1	m_5
0	1	1	0	m_6
0	1	1	1	m_7
1	0	0	0	m_8
1	0	0	1	m_9
1	0	1	0	m_{10}
1	0	1	1	m_{11}
1	1	0	0	m_{12}
1	1	0	1	m_{13}
1	1	1	0	m_{14}
1	1	1	1	m_{15}

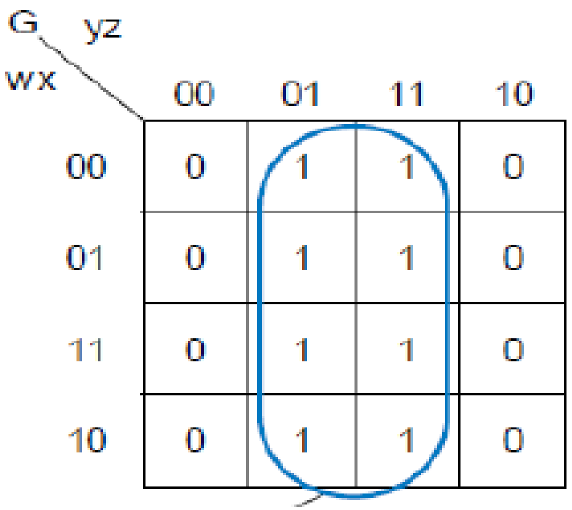
x_3x_4 \ x_1x_2	00	01	11	10
00	m_0	m_4	m_{12}	m_8
01	m_1	m_5	m_{13}	m_9
11	m_3	m_7	m_{15}	m_{11}
10	m_2	m_6	m_{14}	m_{10}

Generalized Four Variables K-map

- Four-variable K-map follows same principle
  - Left/right adjacent
  - Top/bottom also adjacent
  - Adjacent cells differ in one variable
  - Two adjacent 1's mean one variable can be eliminated
- Four adjacent 1s means two variables can be eliminated
  - Eight adjacent 1s means three variables can be eliminated

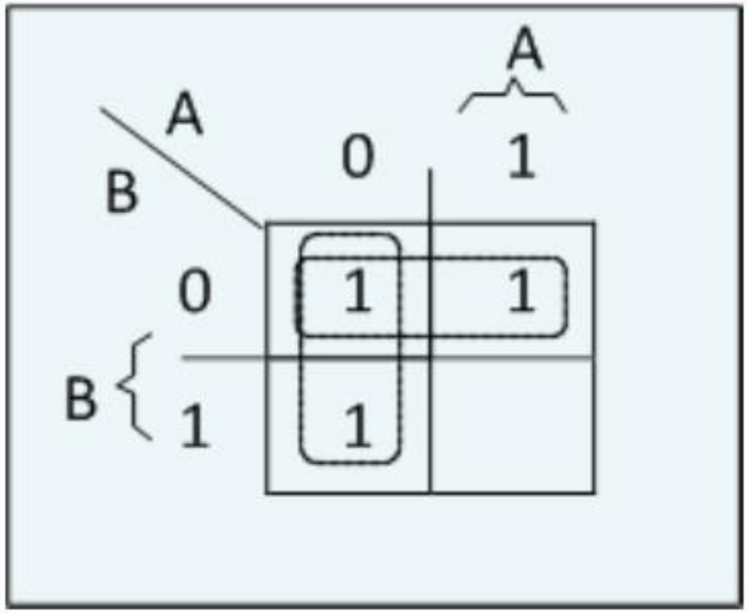


F=w'xy'+yz



G=z

Example 1: $f(A, B) = \Sigma(0, 1, 2)$



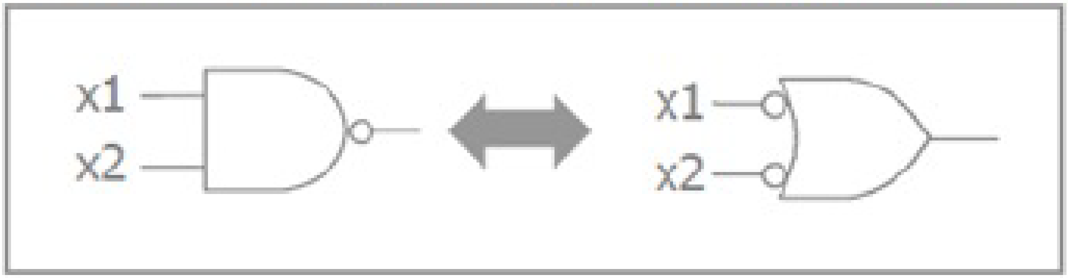
$$F(A,B) = A'B' + A'B + AB'$$

$$f(A, B) = \overline{A} + \overline{B}$$

normal way using algebraic manipulation

$$= A'(B'+B) + AB'$$

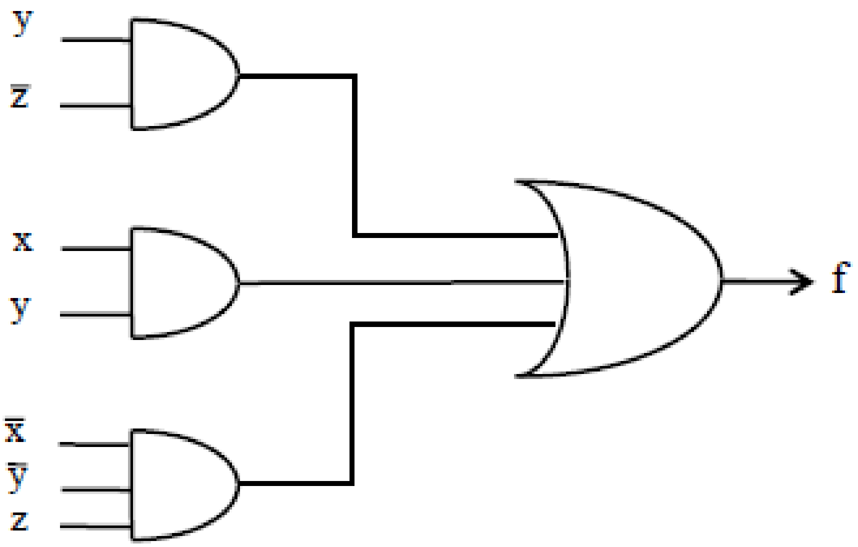
$$= A'.1 + AB' = A' + B'$$

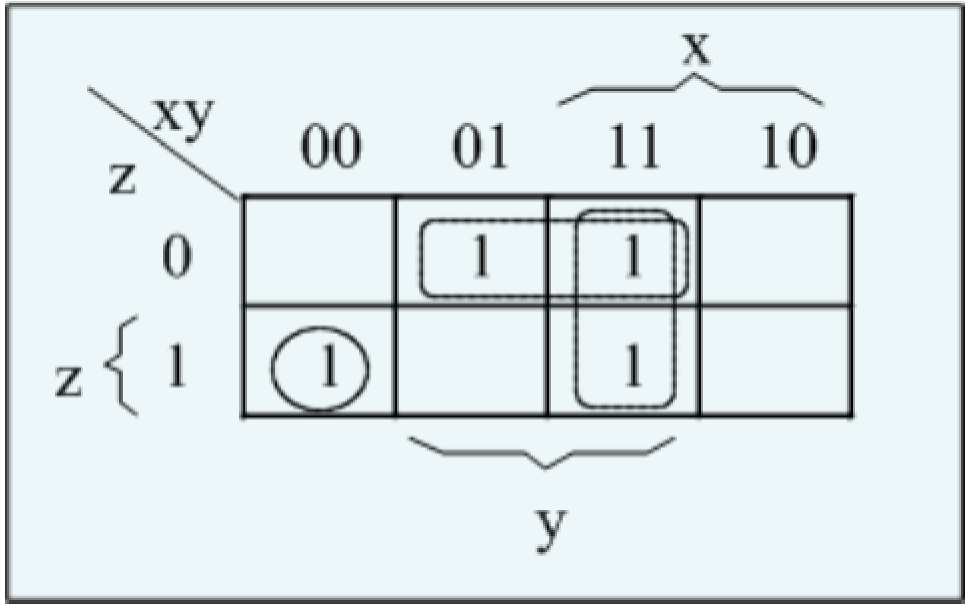


It is a NAND Gate

Example 2: $f(x,y,z)=\Sigma(1,2,6,7)$

$$f=y\bar{z}+xy+\bar{x}\bar{y}z$$

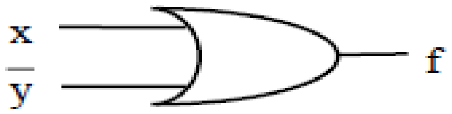


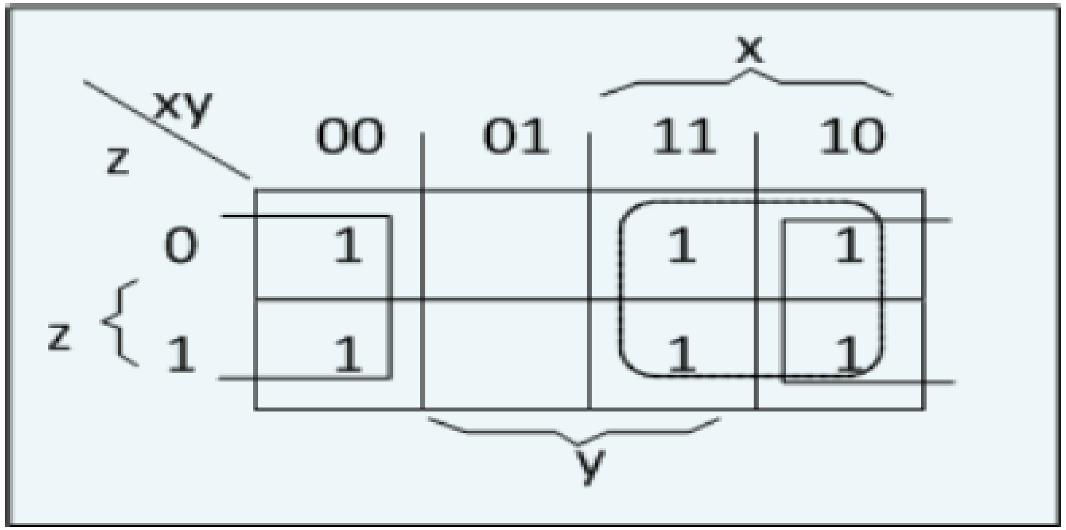


K-map – Examples – minimize from function

Example 3: $f(x, y, z) = \bar{x}\bar{y} + xy\bar{z} + xyz + x\bar{y}$

$$f = x + \bar{y}$$

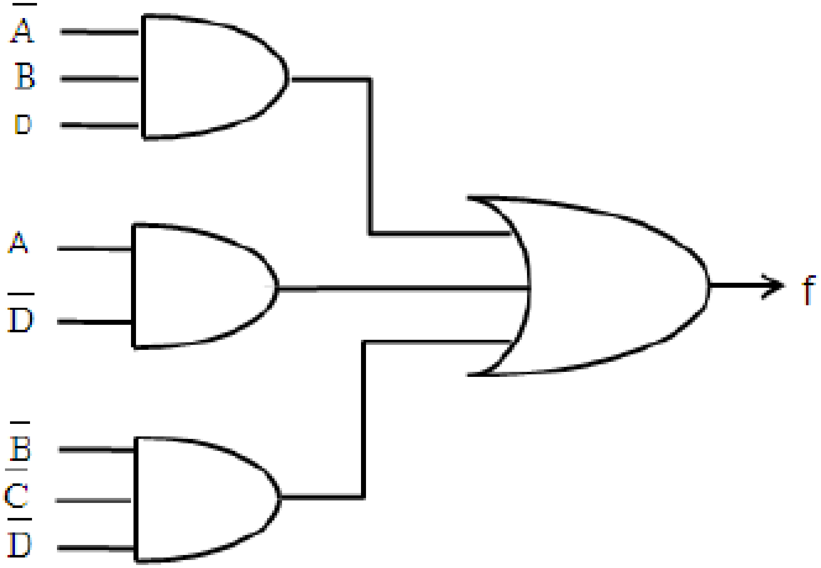


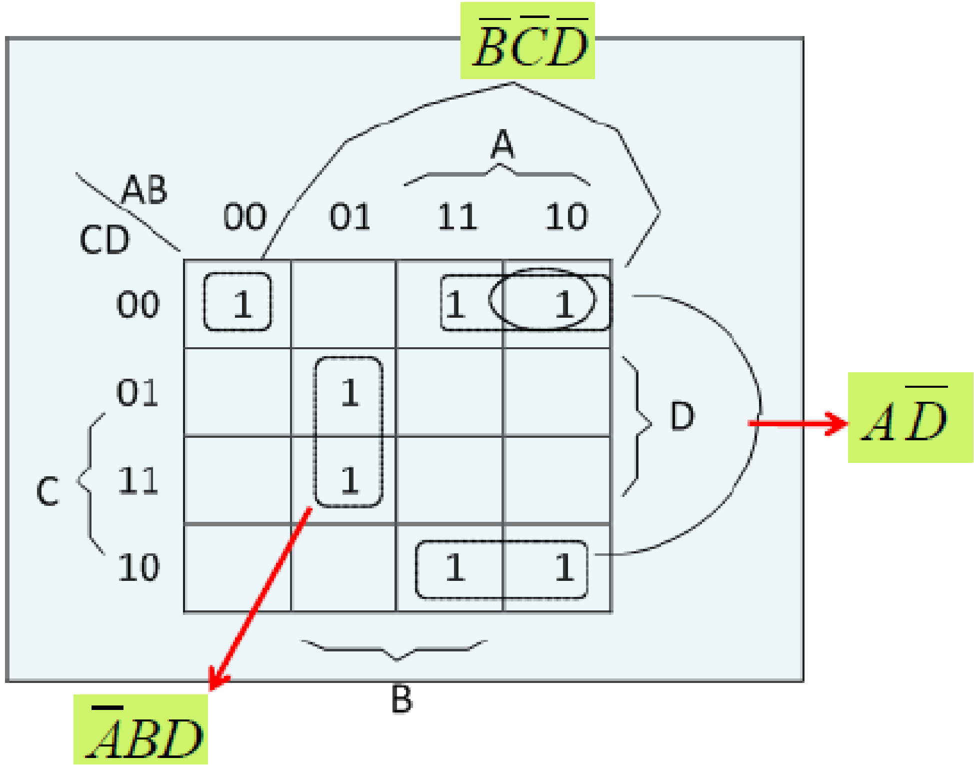


Example 4: $f(A, B, C, D) = \Sigma(0, 5, 7, 8, 10, 12, 14)$

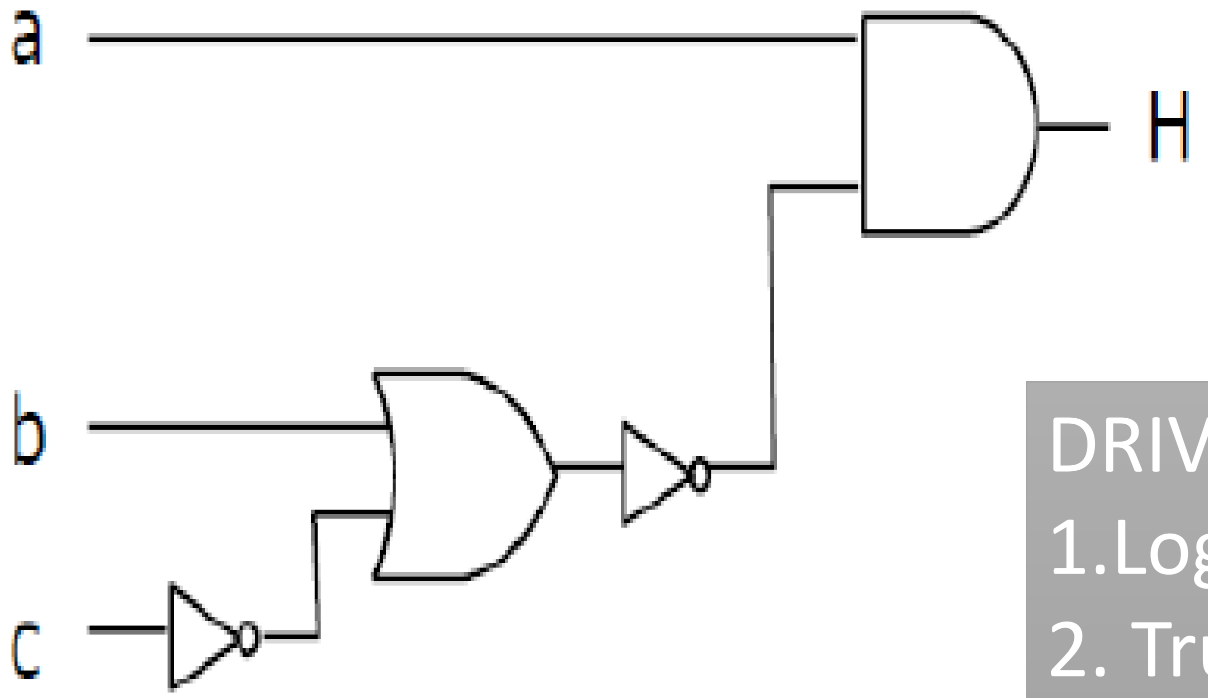
$$f(A, B, C, D) = \bar{A}\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{D} + AB\bar{C}\bar{D} + ABC\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD + A\bar{B}C\bar{D}$$

$$f = \bar{A}BD + A\bar{D} + \bar{B}\bar{C}\bar{D}$$





Exercise :



DRIVE THE FOLLOWING :

1.Logical function

2. Truth table

3.SOP ,POS minimization

4.K-map minimization