Possible Functions

- n variable can give how many possible functions?
- Answer = 2^{2^n}

Other Logic Operations

- Given two Boolean variables:
  - When binary operators AND and OR are placed between two variables they form two Boolean functions x . y and x + y
  - there are $2^{2^2} = 16$ combinations of the two variables as there are 2^{2^n} possible functions for n binary variables (we will see the details of these 16 functions in next slides)
  - each combination of the variables can result in one of two values, 0 or 1, therefore there are $2^4=16$ functions (combinations of 0's and 1's for the four combinations, 00,01,10,11)
- AND and OR represent two of the 16 possible functions.

Function Combinations

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

- F_1 represents the AND Operation
- F_7 represents the OR Operation
- There are 14 other functions

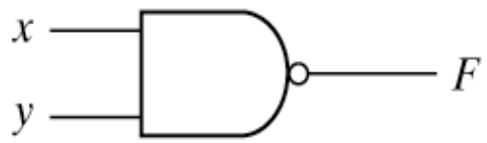
16 Two-Variable Functions

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Boolean Function	Operator Symbol	Name	Comments
$F_0 = 0$		NULL	binary constant 0
$F_1 = xy$	$x \cdot y$	AND	x and y
$F_2 = xy'$	x/y	Inhibition	x, but not y
$F_3 = x$		Transfer	x
$F_4 = x'y$	y/x	Inhibition	y, but not x
$F_5 = y$		Transfer	y
$F_6 = xy' + x'y$	$x \oplus y$	Exclusive-OR	x or y, but not both
$F_7 = x + y$	$x + y$	OR	x or y
$F_8 = (x + y)'$	$x \downarrow y$	NOR	not-OR
$F_9 = xy + x'y'$	$(x \oplus y)'$	Equivalence (XNOR)	x equals y
$F_{10} = y'$	y'	Complement	not y
$F_{11} = x + y'$	$x \subset y$	Implication	if y, then x
$F_{12} = x'$	x'	Complement	not x
$F_{13} = x' + y$	$x \supset y$	Implication	if x, then y
$F_{14} = (xy)'$	$x \uparrow y$	NAND	not-AND
$F_{15} = 1$		Identity	Binary constant 1

Function Gate Implementations Contd..

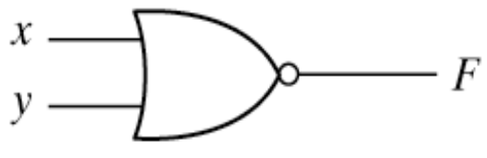
NAND



$F = (xy)'$

x	y	F
0	0	1
0	1	1
1	0	1
1	1	0

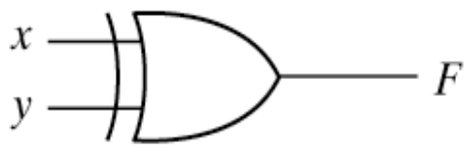
NOR



$F = (x + y)'$

x	y	F
0	0	1
0	1	0
1	0	0
1	1	0

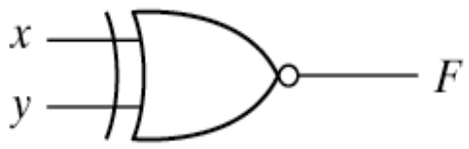
Exclusive-OR (XOR)



$F = xy' + x'y$
$= x \oplus y$

x	y	F
0	0	0
0	1	1
1	0	1
1	1	0

Exclusive-NOR or equivalence



$F = xy + x'y'$
$= (x \oplus y)'$

x	y	F
0	0	1
0	1	0
1	0	0
1	1	1

Function Gate Implementations

- It is easier to implement a Boolean function with these types of gates (as seen on last slide)
- Inverter (Complement), Buffer (transfer), AND, OR, NAND, NOR, X-OR, and XNOR (equivalence) are used as standard gates in digital design
- NAND and NOR are extensively used logic gates and are more popular than AND and OR gates because these gates are easily constructed with transistor circuits and digital circuits are easily implemented with them

Multiple Inputs

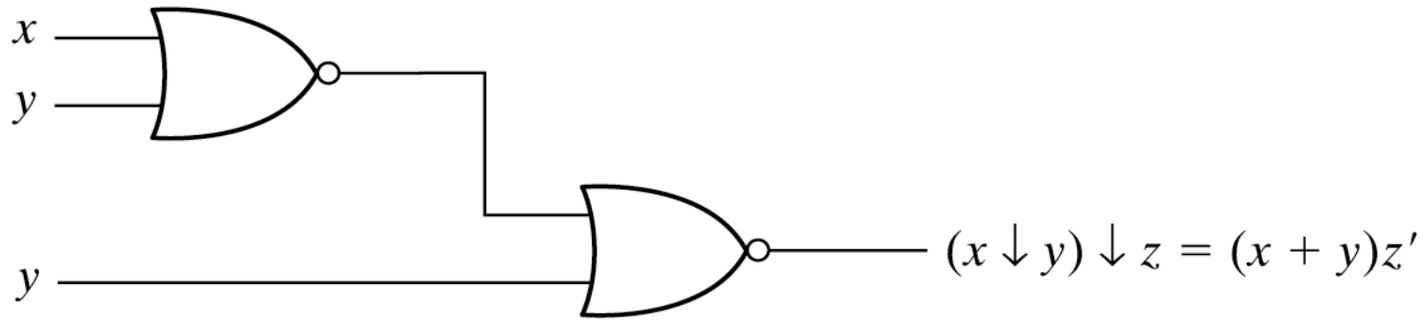
- All of the previous defined gates, with the exception of the inverter and the buffer, can have multiple inputs.
  - A gate can have multiple inputs provided it is a binary operation that is commutative

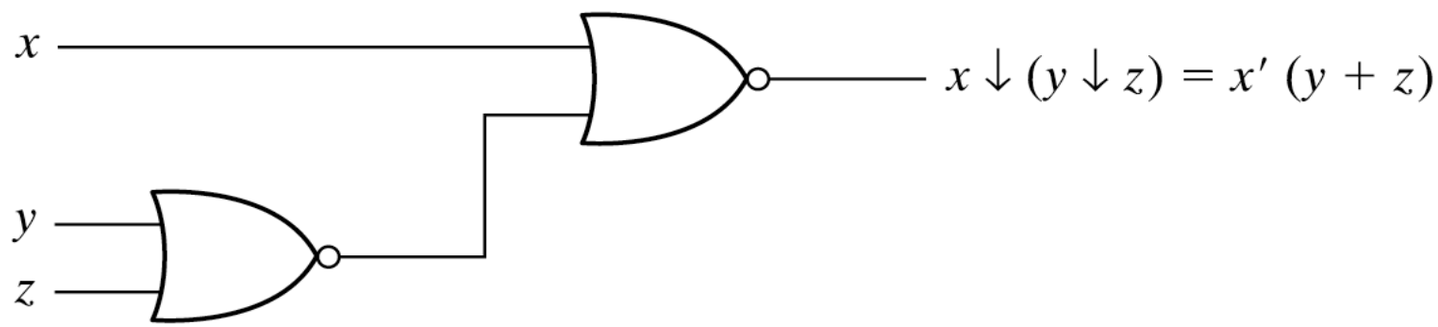
    (x + y = y + x and xy = yx)

    and associative

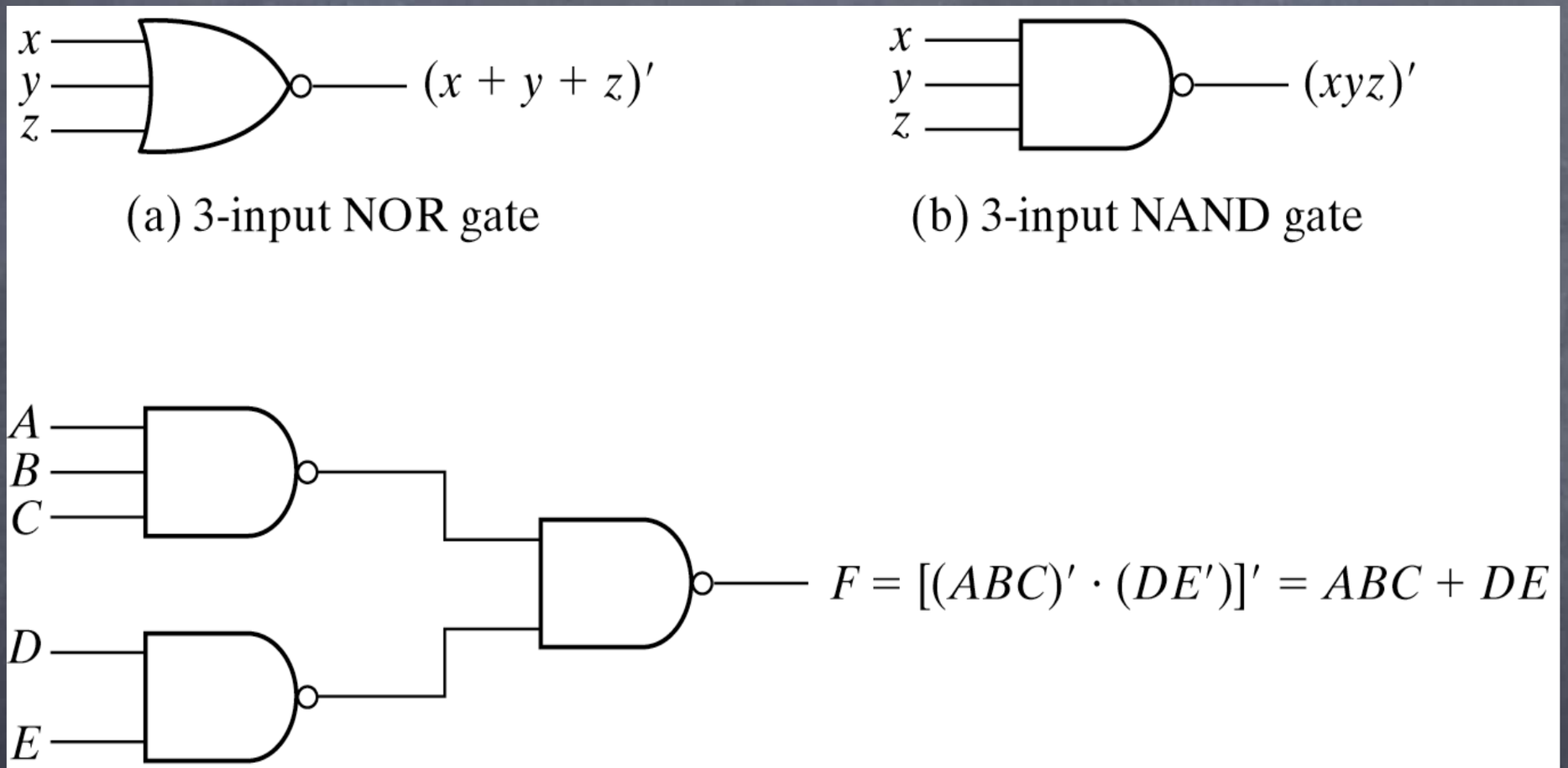
    (x + (y + z) = (x + y) + z and x(yz) = (xy)z)
  - NAND and NOR functions are commutative but not associative
  - To overcome this difficulty we define multiple NOR (or NAND) gate as a complemented OR (or AND) gate e.g., as (x+y+z)’ or (xyz)’

Multiple Inputs (Non-associative NOR operation)

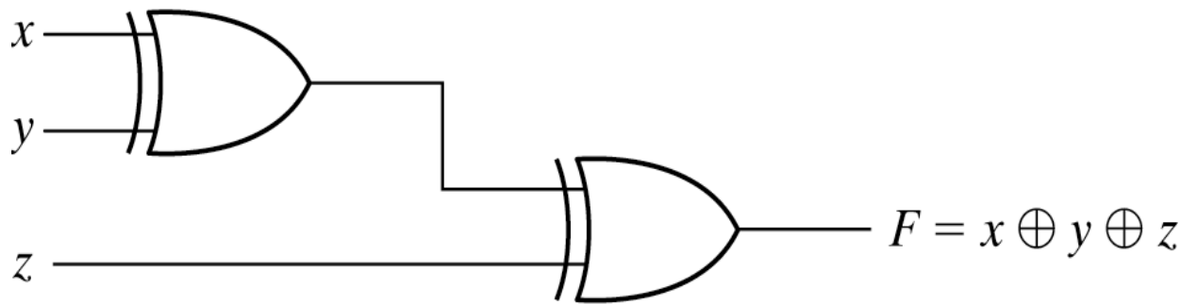




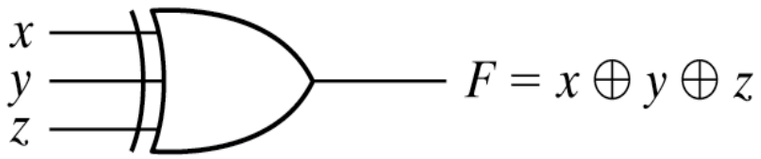
Multiple Inputs NOR and NAND gates



Multiple Inputs XOR gate



(a) Using 2-input gates



(b) 3-input gate

x	y	z	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

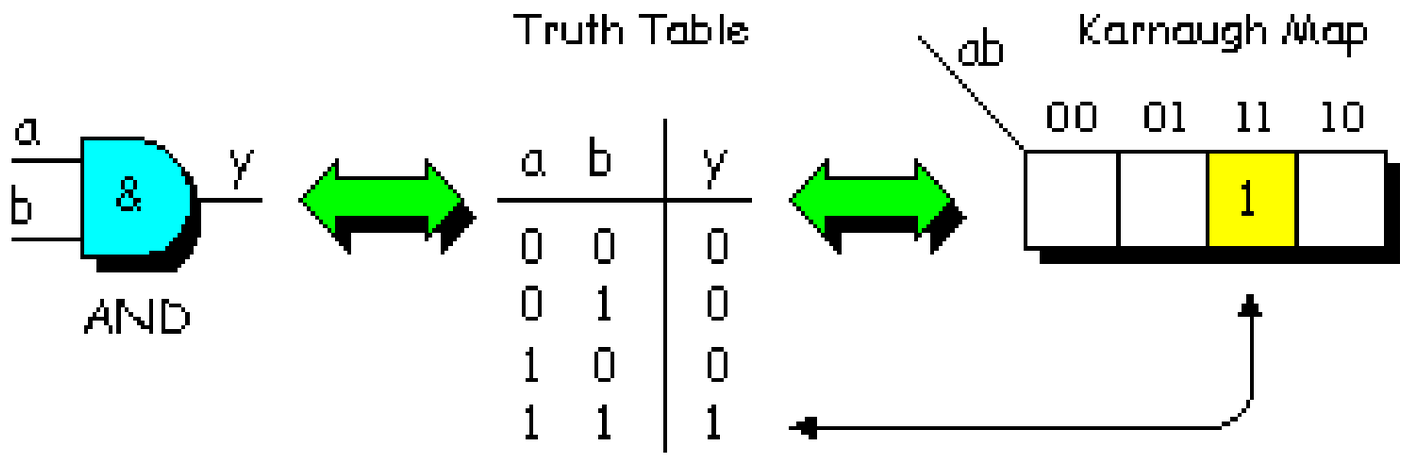
(c) Truth table

Learning Objective

- Given a function (completely or incompletely specified) of three to five variables, **plot it on a Karnaugh map** and use to **simplify Boolean expressions**. The function may be given in minterms, maxterms, or algebraic form
- Reference Text Book section 3.1 to 3.3

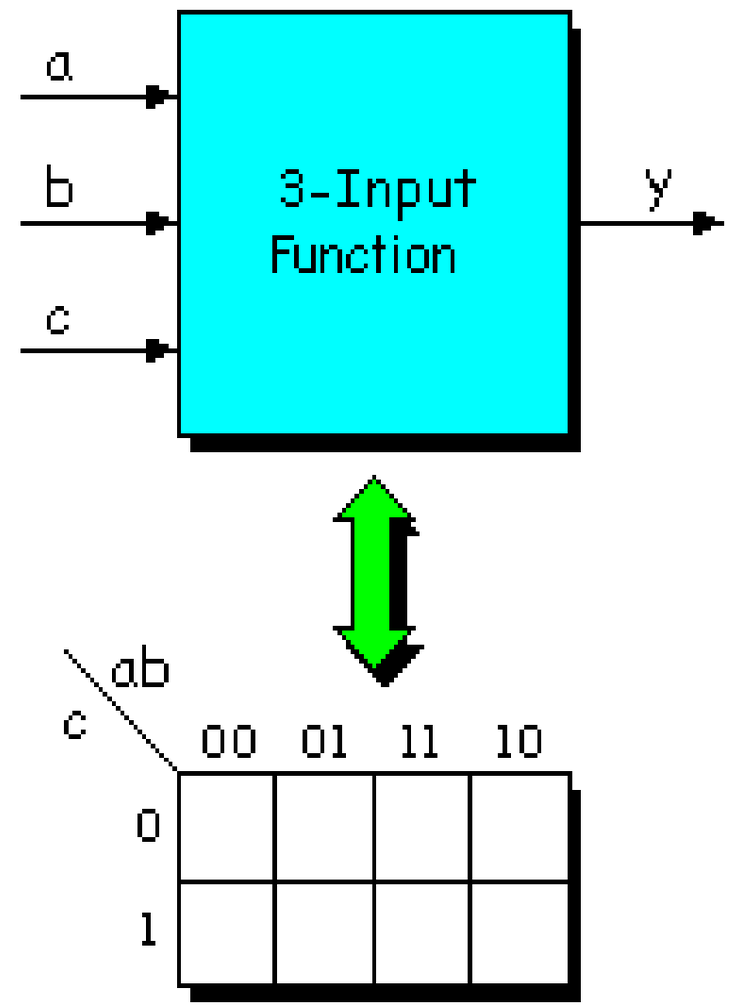
Karnaugh Map Method

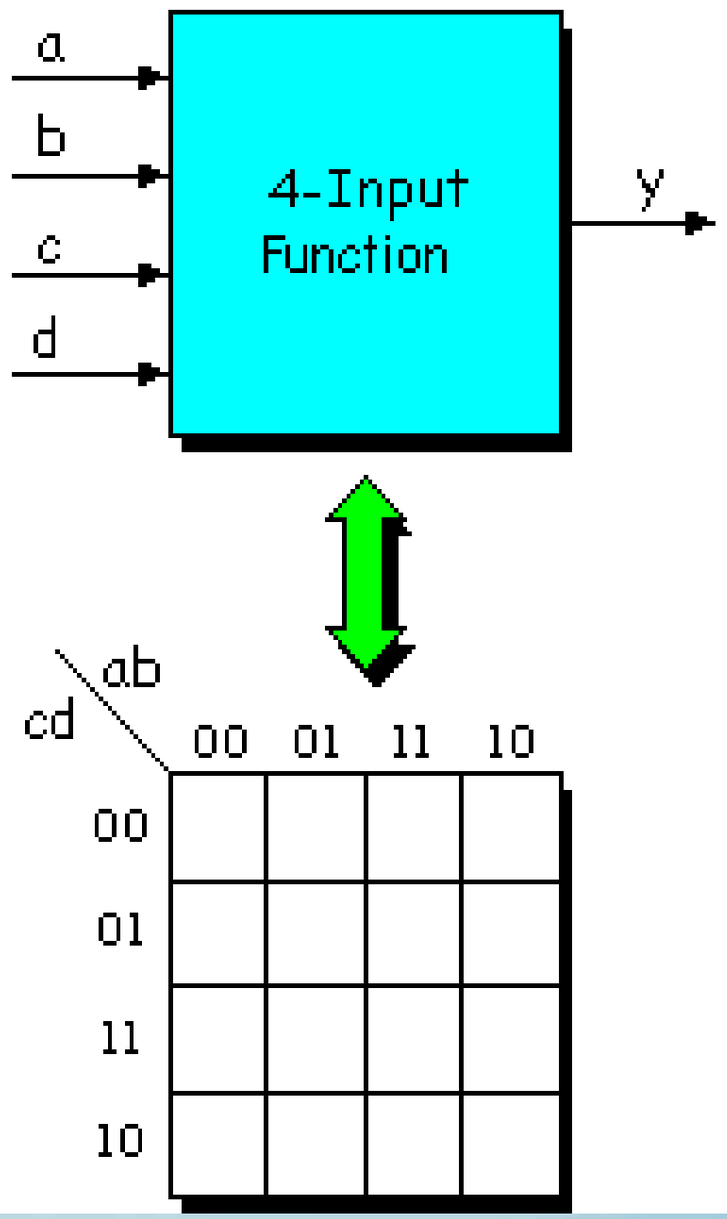
- Provides an alternative technique for representing Boolean functions
- One Box/square on map is for each row i.e minterm or maxterm of the function in Truth Table
- Karnaugh map's input values must be ordered such that the values for adjacent columns vary by only a single bit, for example, 00, 01, 11, and 10. This is necessary to observe the variable transitions
- Known as a G*ray code*



Multiple Inputs K-Map

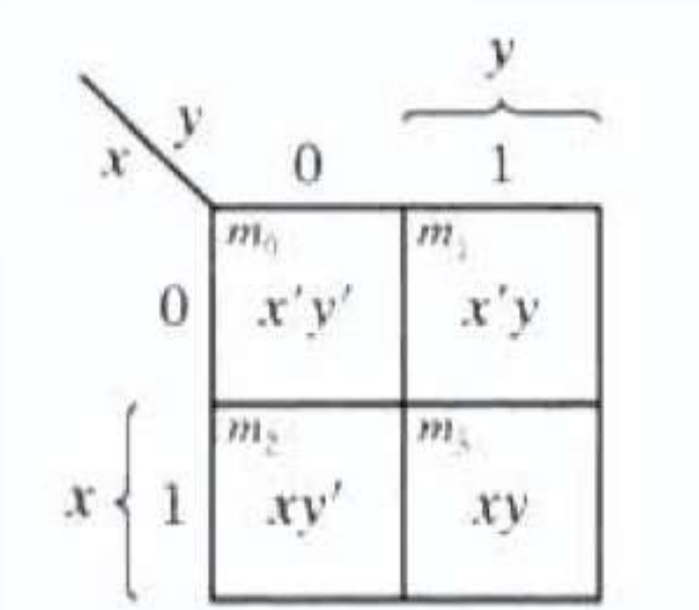
- *Three and Four-variable maps*

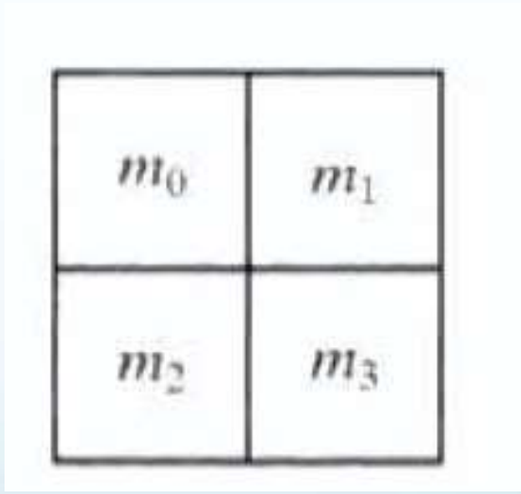




Two-Variable Map

- A two-variable map holds four minterms for two variables.
  - Again, we mark the squares of the minterms that belong to a given function.
  - Note that the sequence is not arranged in a binary way.
  - The sequence used, similar to Gray code, allows only one bit to change from column to column and row to row.





Three-Variable Map

- A three-variable map holds eight minterms for three variables.
  - Again, we mark the squares of the minterms that belong to a given function.
  - Note that the sequence is not arranged in a binary way.
  - The sequence used, similar to Gray code, allows only one bit to change from column to column and row to row.

x \ yz	00	01	11	10
0	x'y'z'	x'y'z	x'yz	x'yz'
1	xy'z'	xy'z	xyz	xyz'

x \ yz	00	01	11	10
0	m_0	m_1	m_3	m_2
1	m_4	m_5	m_7	m_6

Three-Variable Map

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

(a)

x \ yz	00	01	11	10
0	$x'y'z'$	$x'y'z$	$x'yz$	$x'yz'$
1	$xy'z'$	$xy'z$	xyz	xyz'

(y spans 11 and 10; z spans 01 and 11)

(b)

- Correction: Columns are yz and not xz in fig 3-3 (book)

The End