Simplification of Boolean Functions
Map Method

- Complexity of the digital logic gates that implement a Boolean function
  - directly related to the complexity of the algebraic expression implemented.
- Simplification by algebraic means is awkward
  - due to the lack of specific rules to predict succeeding steps.
- Map method provides
  - straightforward procedure for minimizing Boolean functions
  - Can be considered as pictorial form of a truth table or as an extension of the Venn diagram
- First proposed by Veitch (1) and slightly modified by Karnaugh (2),
  - known as the "Veitch diagram" or the "Karnaugh map."

Karnaugh Map

- Map is a diagram made up of squares.
  - Each square represents one minterm.
  - presents a visual diagram of all possible ways a function may be expressed in a standard form.
- By recognizing various patterns,
  - alternative algebraic expressions for the same function is derived.
  - Simplest expression is selected (with minimum literals- not necessarily unique).

Two variable map

- Useful way to represent any one of the 16 Boolean functions of two variables

Three variable map

- Minterms are arranged, not in a binary sequence, but in a sequence where,
  - only one bit changes from 1 to 0 or from 0 to 1.

Simplify the Boolean function:

$$F = x'yz + x'yz' + xy'z' + xy'z$$

- 1 marked in each square as needed to represent the function.
- Adjacent squares are grouped together (minterms differ by 1 variable).

Simplify the Boolean function:

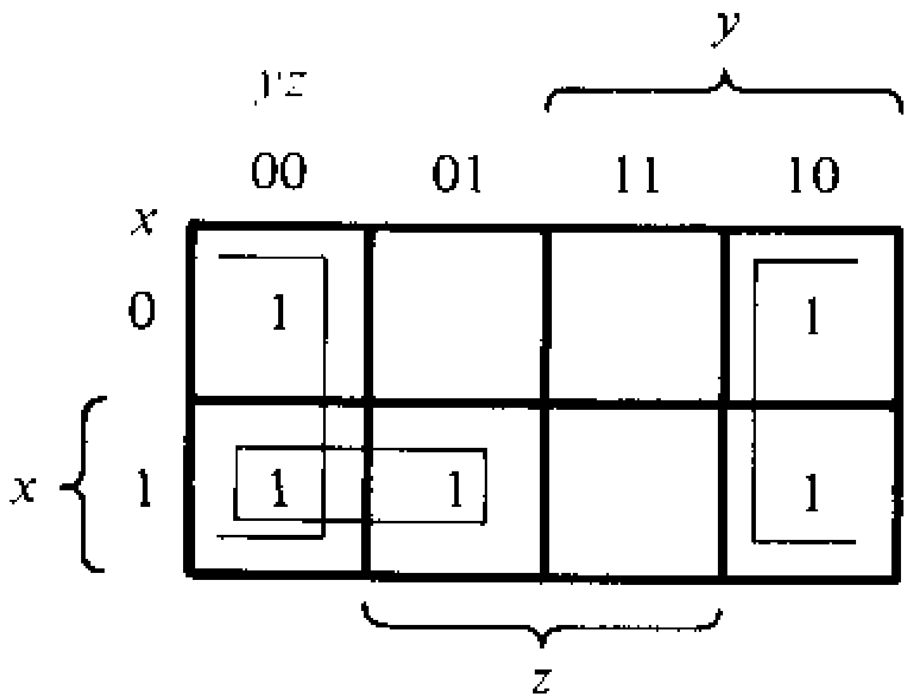
$$F = x'yz + x'yz' + xy'z' + xy'z$$

- Ans: $F = x'y + xy'$

Simplify $F = x'yz + xy'z' + xyz + xyz'$

Simplify $F = A'C + A'B + AB'C + BC$

Simplify $F(x, y, z) = \Sigma(0, 2, 4, 5, 6)$.



$$F = z' + xy'$$

Four variable map

- 16 minterms
- Rows and columns are numbered
  - with only one digit changing value between two adjacent rows or columns.
- Each square can be obtained from the concatenation of the row number with the column number.

Four variable map (continued..)

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

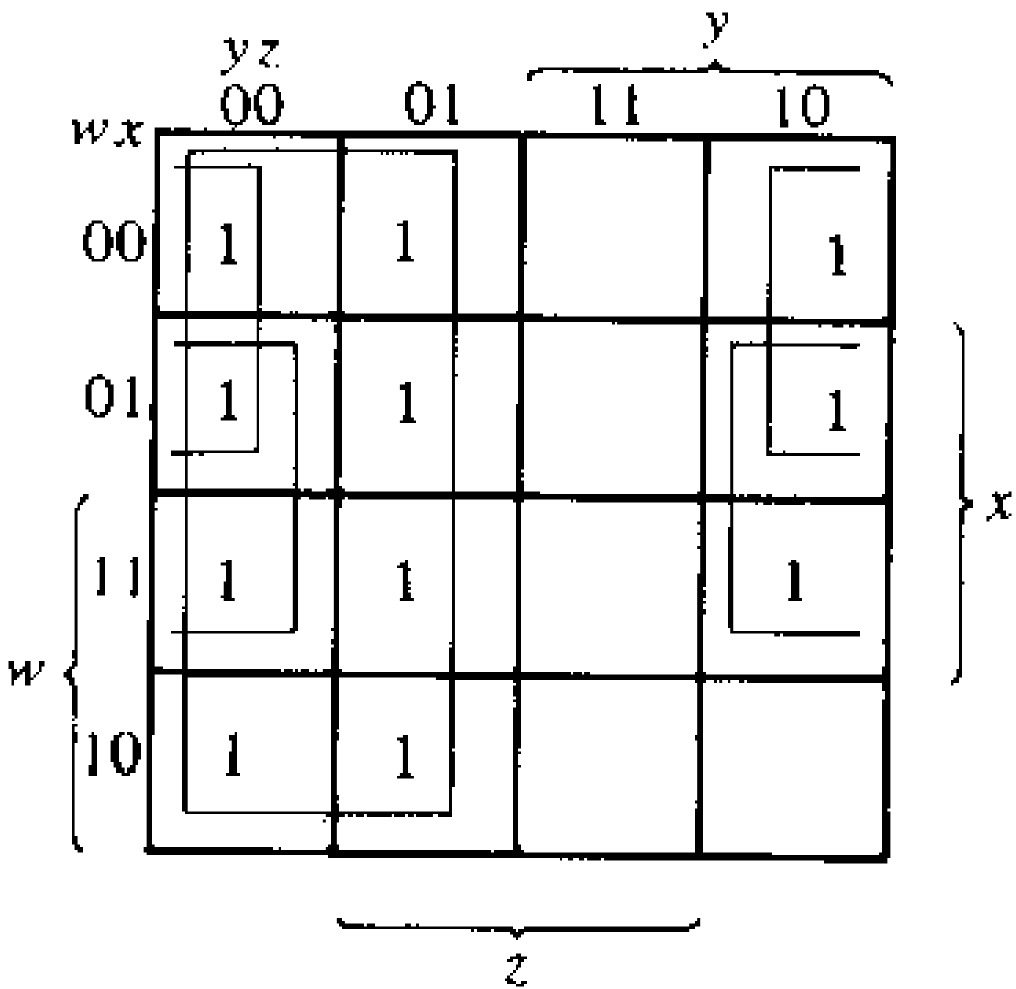
(a)

$wx \backslash yz$	00	01	11	10
00	$w'x'y'z'$	$w'x'y'z$	$w'x'yz$	$w'x'yz'$
01	$w'xy'z'$	$w'xy'z$	$w'xyz$	$w'xyz'$
11	$wxy'z'$	$wxy'z$	$wxyz$	$wxyz'$
10	$wx'y'z'$	$wx'y'z$	$wx'yz$	$wx'yz'$

y (columns 11, 10); x (rows 01, 11); w (rows 11, 10); z (columns 01, 11)

(b)

Simplify $F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$



Simplify $F = A'B'C' + B'CD' + A'BCD' + AB'C'$

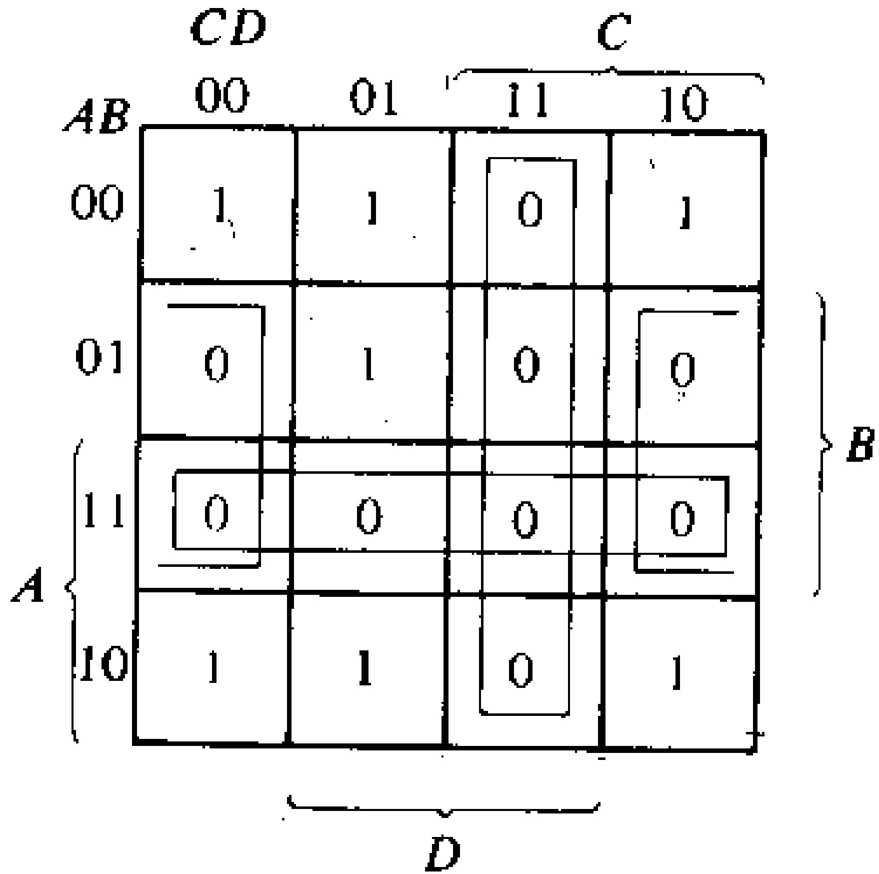
Product of Sums Simplification

- All problems were on sum of products.
- Write the equation for F′.
- Find its’s complement F″ to get the equation of the function in the POS form.

Simplify $F(A, B, C, D) = \Sigma(0, 1, 2, 5, 8, 9, 10)$

(a) sum of products

(b) product of sums.



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

Simplify $F(x, y, z) = \Pi(0, 2, 5, 7)$

Don’t care Conditions

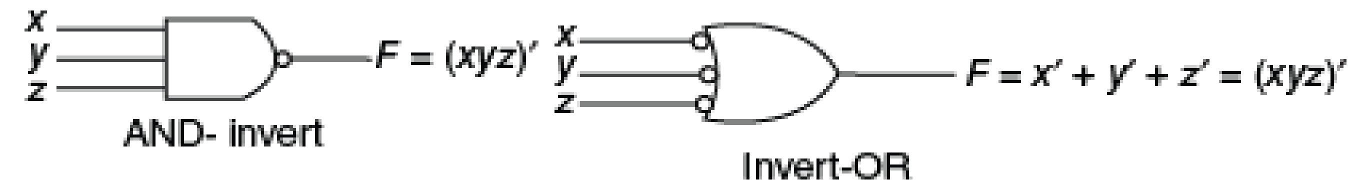
- A four-bit decimal code has six combinations which are not used.
- Digital circuit using this code operates under the assumption that
  - unused combinations never occurs when the system works properly.
- Hence, we don’t care what the function o/p is for these combinations.
  - can be used for further simplification of the function.
- It is marked as ‘X’ to distinguish it from 1’s and 0’s.
  - For simplification, it can assume either 0 or 1, whichever gives the simplest expression.

Simplify

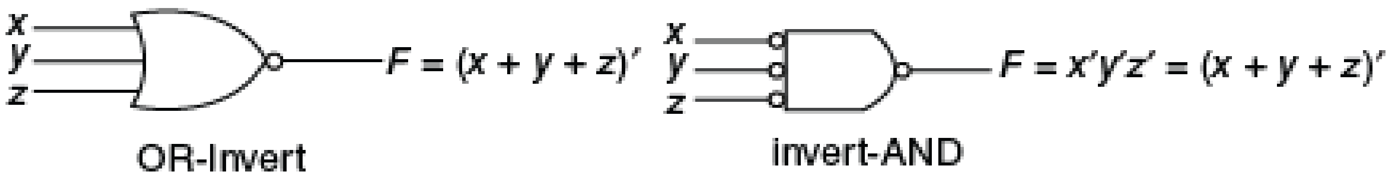
$F(w, x, y, z) = \Sigma(1,3, 7, 11, 15)$ and the don't-care conditions:

$$d(w, x, y, z) = \Sigma(0, 2, 5)$$

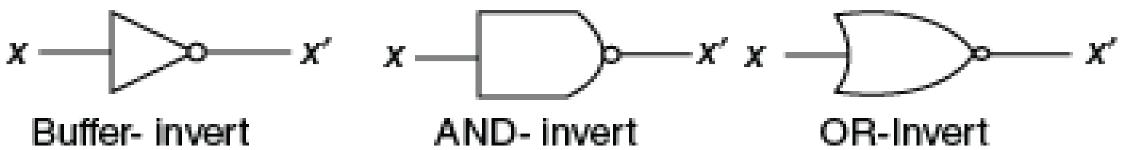
NAND and NOR Implementation



Two grapic symbol for NAND gate



Two grapic symbols for NOR gate.



Three grapic symbols for inverter.

Implement $F = AB + CD + E$

a) AND and OR gates.

b) Using NAND gates.

Implement with NAND gates

- $F(x, y, z) = \Sigma\,(0, 6)$
- Implement F′.

NOR implementation

TABLE 3-3
Rules for NAND and NOR Implementation

Case	Function to simplify	Standard form to use	How to derive	Implement with	Number of levels to F
(a)	F	Sum of products	Combine 1's in map	NAND	2
(b)	F'	Sum of products	Combine 0's in map	NAND	3
(c)	F	Product of sums	Complement F' in (b)	NOR	2
(d)	F'	Product of sums	Complement F in (a)	NOR	3

Implement with NOR gates

- $F(x, y, z) = \Sigma\,(0, 6)$
- Implement F′.

Review

- Map method for 3 and 4 variables.
- Don’t care condition
- NOR and NAND implementation