Today:

- Gate-level minimization – part 2
  - Four variable K-maps
  - POS implementation using K-maps
  - Incompletely specified functions & Don’t care
  - Ex-OR being an “Odd function”
  - Application: Parity generator & checker

Four variable K-map

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)

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

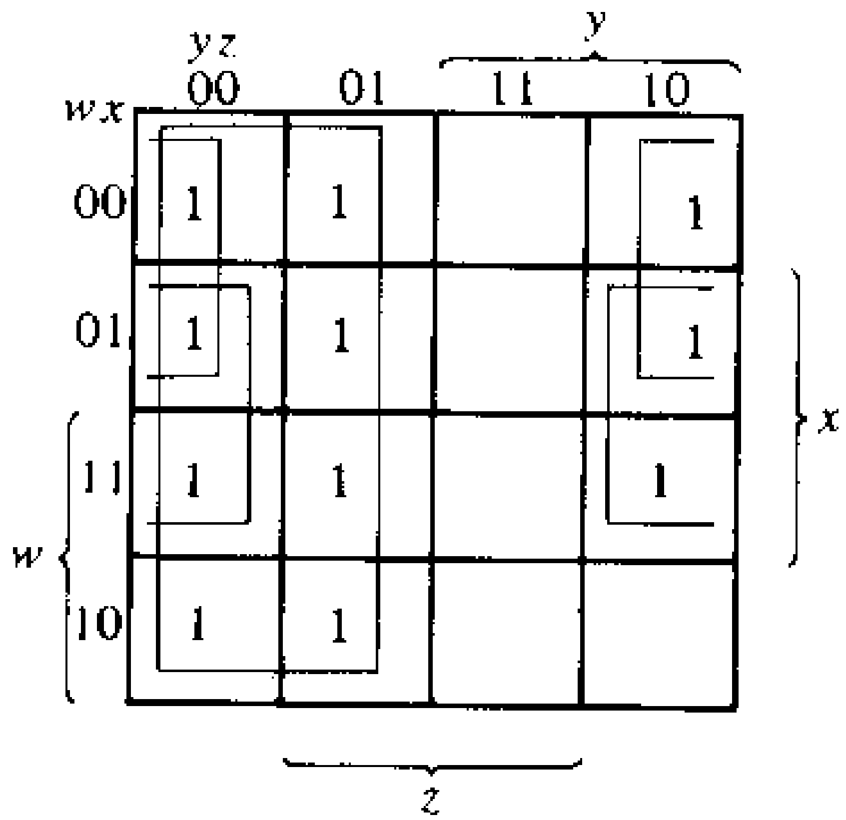
(b)

Remarks

One square represents one minterm, giving a term of four literals.
Two adjacent squares represent a term of three literals.
Four adjacent squares represent a term of two literals.
Eight adjacent squares represent a term of one literal.
Sixteen adjacent squares represent the function equal to 1.

Example

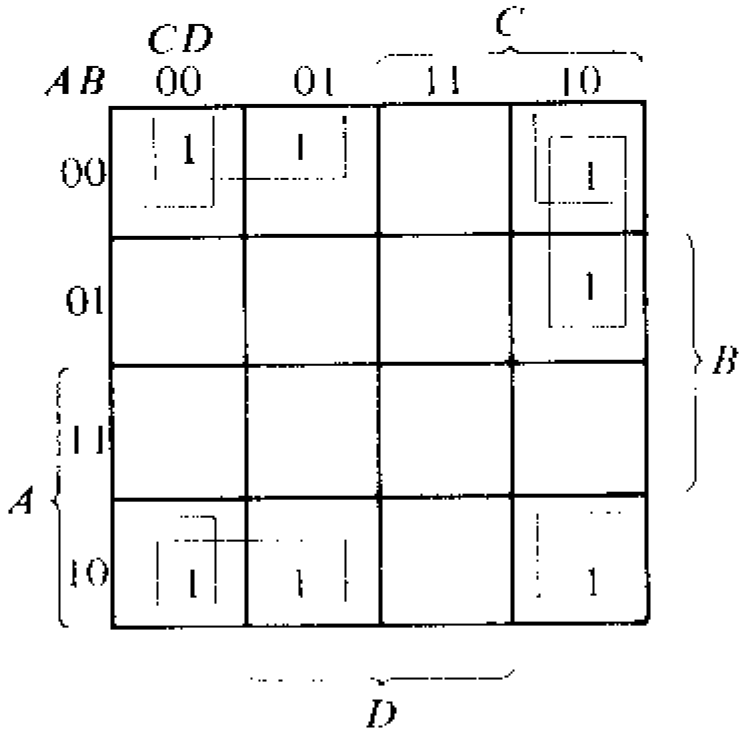
Given: $F(w, x, y, z) = \Sigma(0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)$



Solution: $F = y' + w'z' + xz'$

Example

Given: $F = A'B'C' + B'CD' + A'BCD' + AB'C'$



Solution: $F = B'D' + B'C' + A'CD'$

Remarks

- Ensure following while choosing adjacent squares in a K-map:
  1. All the minterms of the function are covered when we combine the squares.
  2. The number of terms in the expression is minimized.
  3. There are no redundant terms, i.e., minterms already covered by other terms.

  Note: Sometimes there may be two or more expressions that satisfy the simplification criteria.

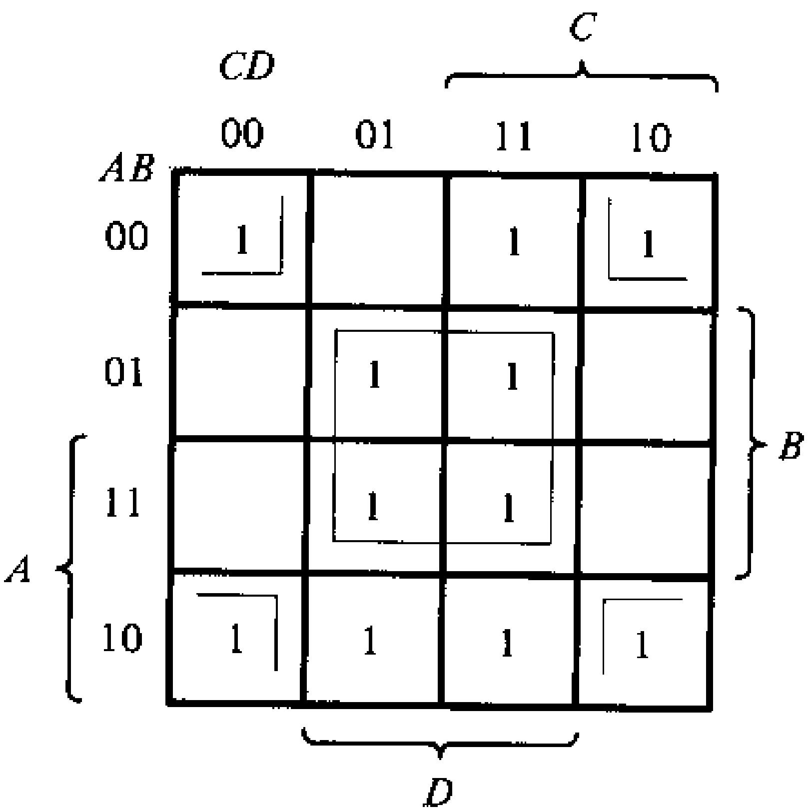
A more systematic way to combine the squares

- *Prime implicant* is a product term obtained by combining the maximum possible number of adjacent squares in the K-map.

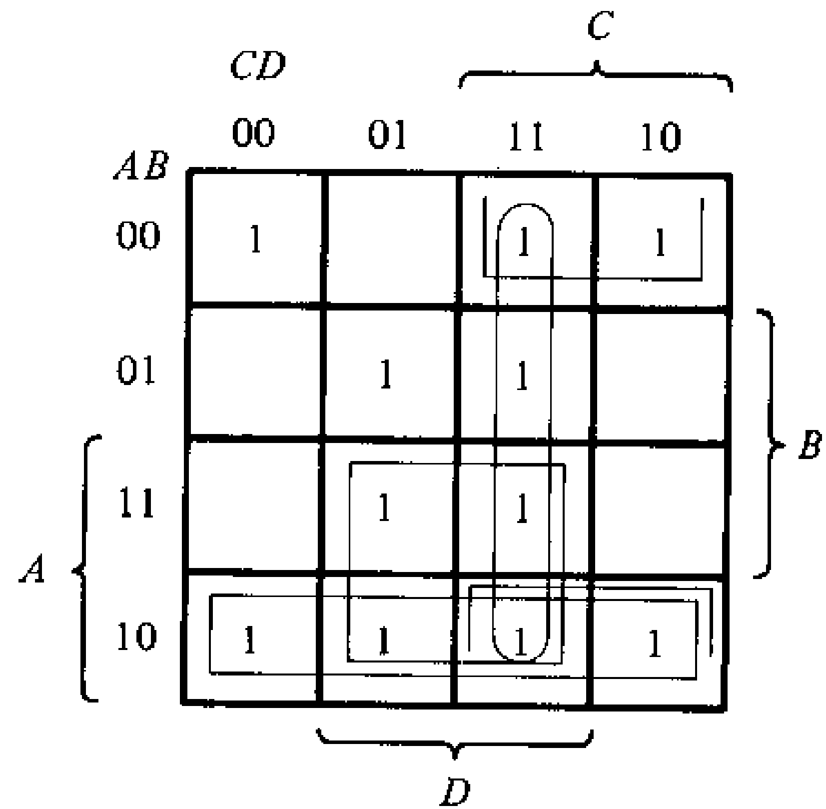
- If a minterm in a square is covered by only one prime implicant, that prime implicant is said to be *essential*.

Example

$$F(A, B, C, D) = \Sigma(0, 2, 3, 5, 7, 8, 9, 10, 11, 13, 15)$$



(a) Essential prime implicants BD and $B'D'$



(b) Prime implicants CD, $B'C$, AD, and AB'

…continued

Four possible ways to express the corresponding function:

$$
\begin{aligned}
F &= BD + B'D' + CD + AD \\
&= BD + B'D' + CD + AB' \\
&= BD + B'D' + B'C + AD \\
&= BD + B'D' + B'C + AB'
\end{aligned}
$$

Product-of-Sums (POS) Implementation using K-map

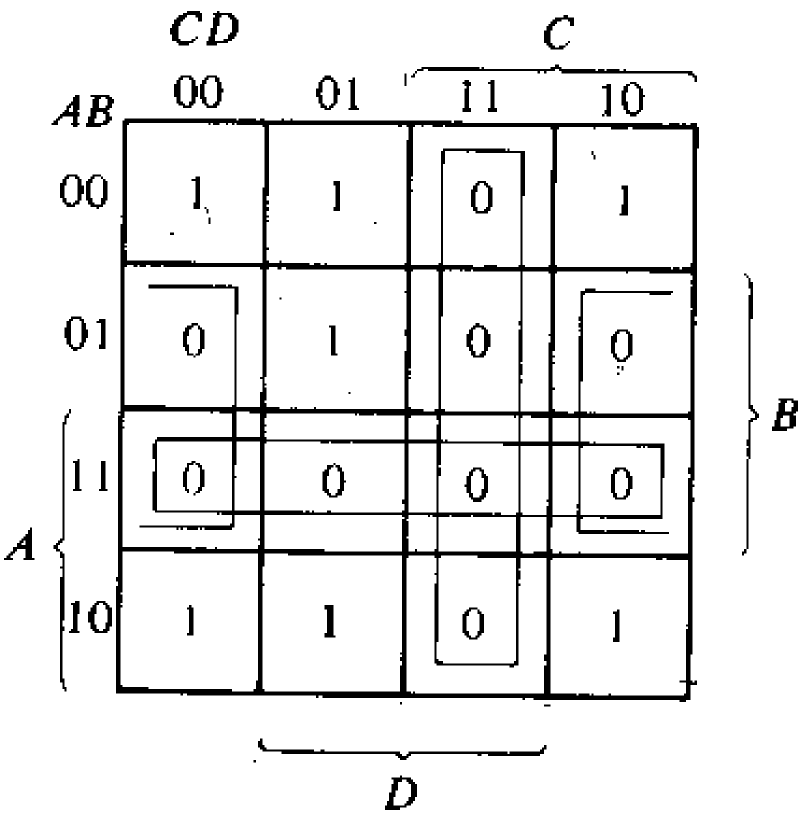
- Hint:

  Use DeMorgan’s property in the K-map and we can obtained the POS implementation instead of the SOP form.

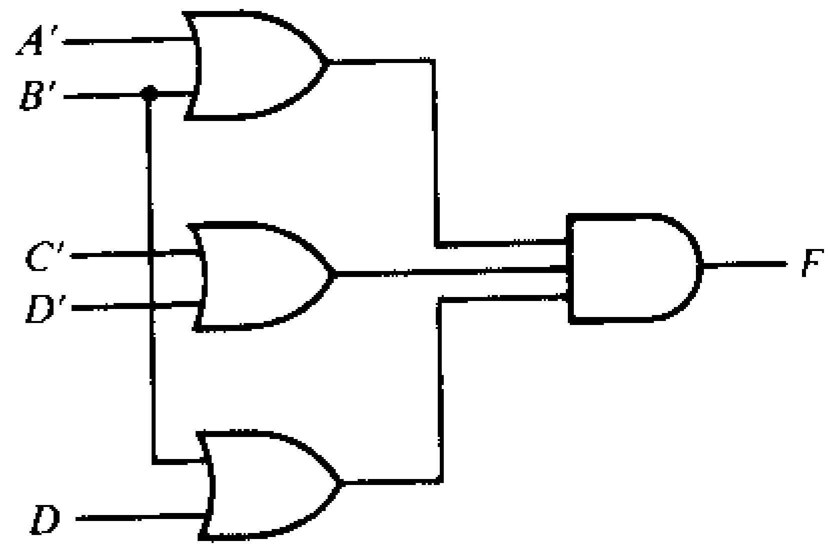
Find POS implementation

See that it is SOP equation

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



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



Find the SOP and POS implementations for given Truth-table

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

K-map representation

$x \backslash yz$	00	01	11	10
0	0	1	1	0
1	1	0	0	1

SOP expression

$$F(x, y, z) = \Sigma(1, 3, 4, 6)$$

POS expression

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

Incompletely specified functions

- Functions that have **unspecified** outputs for some input combinations are called *incompletely specified functions*.

  meaning of **<u>unspecified:</u>**

  e.g., in a design for an application, some combinations of the inputs are **not used** and hence considered as **unspecified.**

Don’t care conditions

- The unspecified minterms of a function called as *don’t-care conditions*.

  meaning:

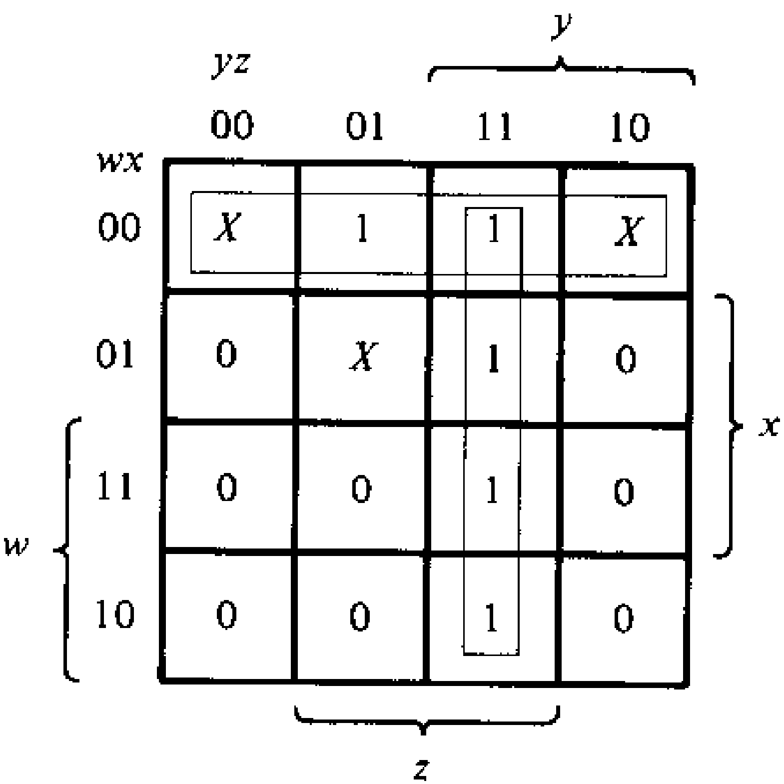
  we simply don’t care what value is assumed by such minterms, either 0 or 1.

- Symbol in the K-map: “X”

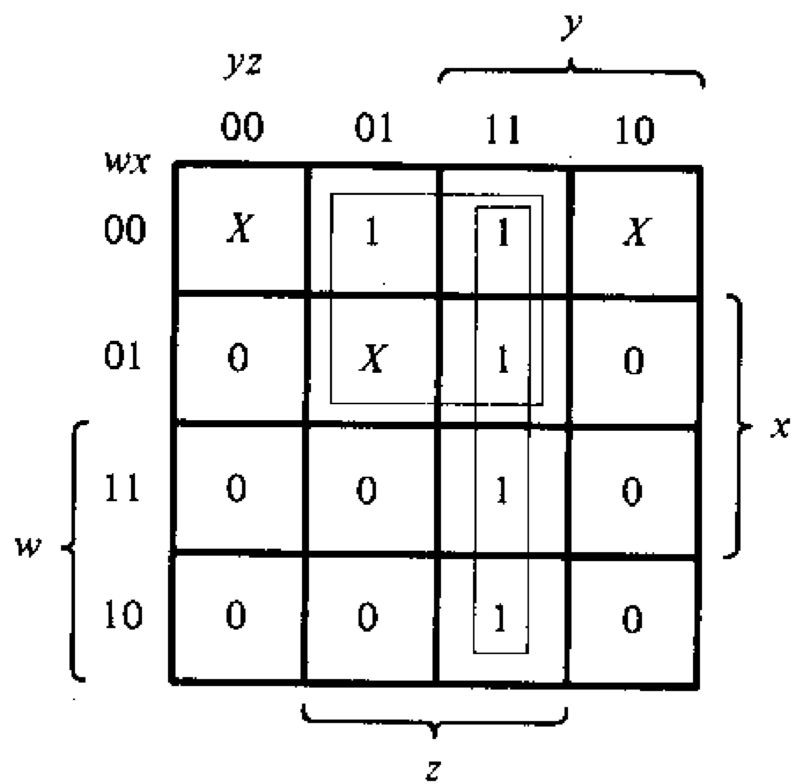
- Use in simplifying the Boolean functions.

Ex: function with don't-care conditions

$F(w, x, y, z) = \Sigma(1, 3, 7, 11, 15)$ **with** $d(w, x, y, z) = \Sigma(0, 2, 5)$



(a) $F = yz + w'x'$



(b) $F = yz + w'z$