Chapter3: Gate-Level Minimization

Lecture2- Five and Six-Variables Function Simplification using Map Method

Engr. Arshad Nazir, Asst Prof
Dept of Electrical Engineering
SEECS

Objectives

- Functions Simplification in Sum-of-Products (SOP) form using Five and Six-Variables Map
- Product of Sums Minimization
- Don’t Care Conditions

5-Variable Map Patterns

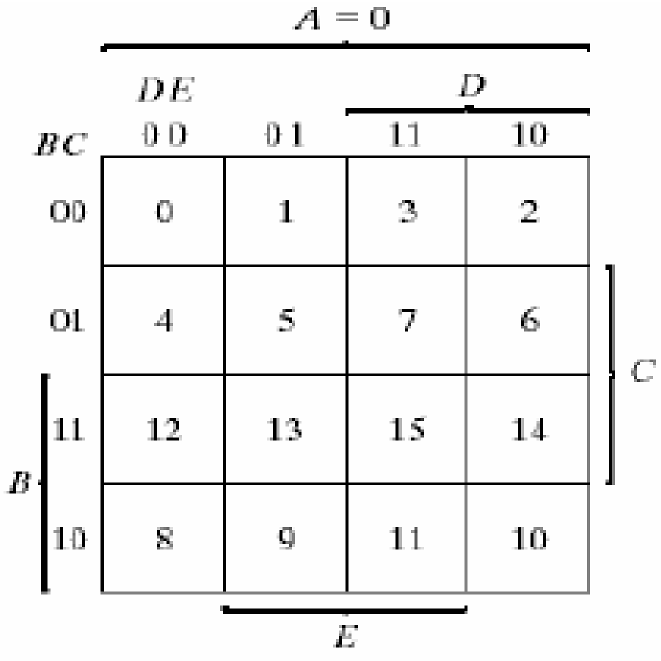
AB \ CDE	000	001	011	010	100	101	111	110
00								
01								
11								
10								

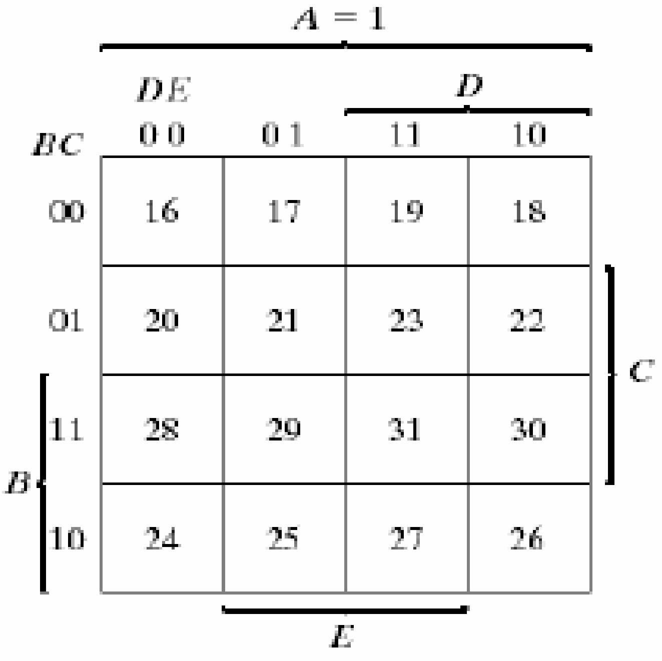
5- variable Karnaugh map (overlay)

AB \ CDE	000	001	011	010	110	111	101	100
00								
01								
11								
10								

5- variable Karnaugh map (Gray code)

5-Variable Map Patterns





5-Variable Map Patterns

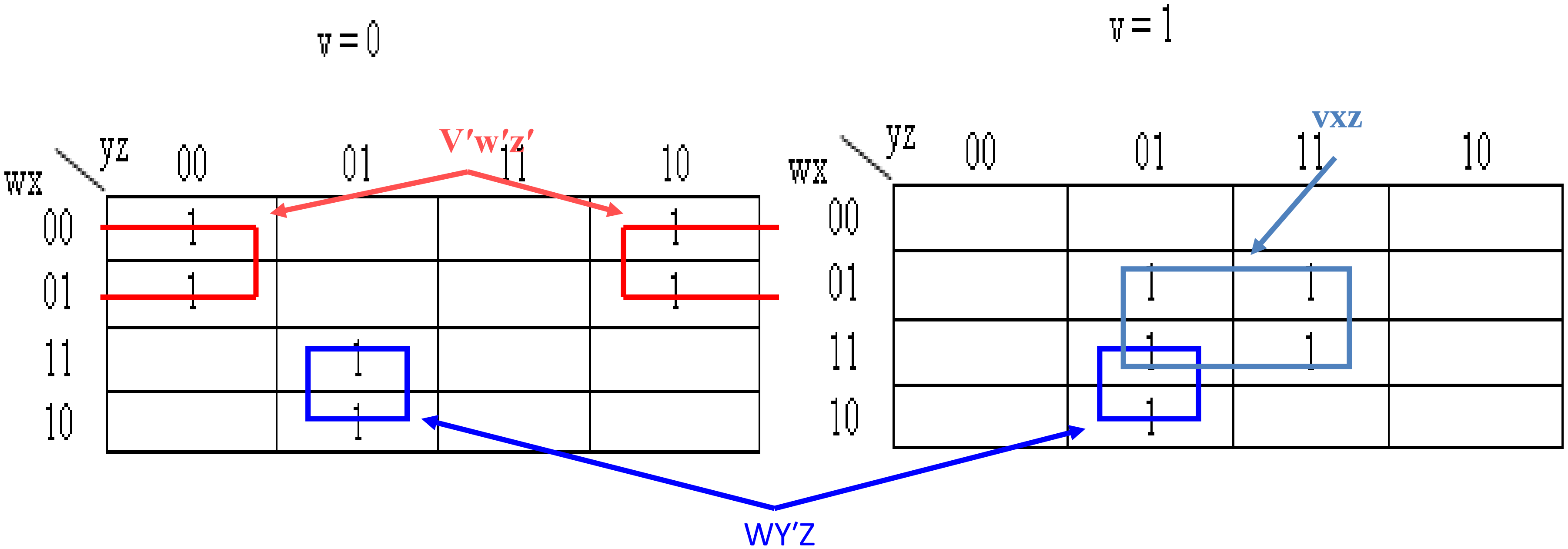
- The number of adjacent squares that may be combined always represent a number that is a power of 2 such as 1, 2, 4, 8, 16, and 32.
  - One square represents one minterm with five literals.
  - Two adjacent squares represents a term of four literals.
  - Four adjacent squares represents a term of three literals.
  - Eight adjacent squares represents a term of two literals.
  - Sixteen adjacent squares represents a term of one literal.
  - Thirty-two adjacent squares represents the entire map and produces a function that is always equal to 1.

Note that the squares on map can be combined horizontally or vertically but not diagonally since these differ by more than one variable.

Minimization Example of 5-Variable Map

Example 3-7: Simplify the Boolean function

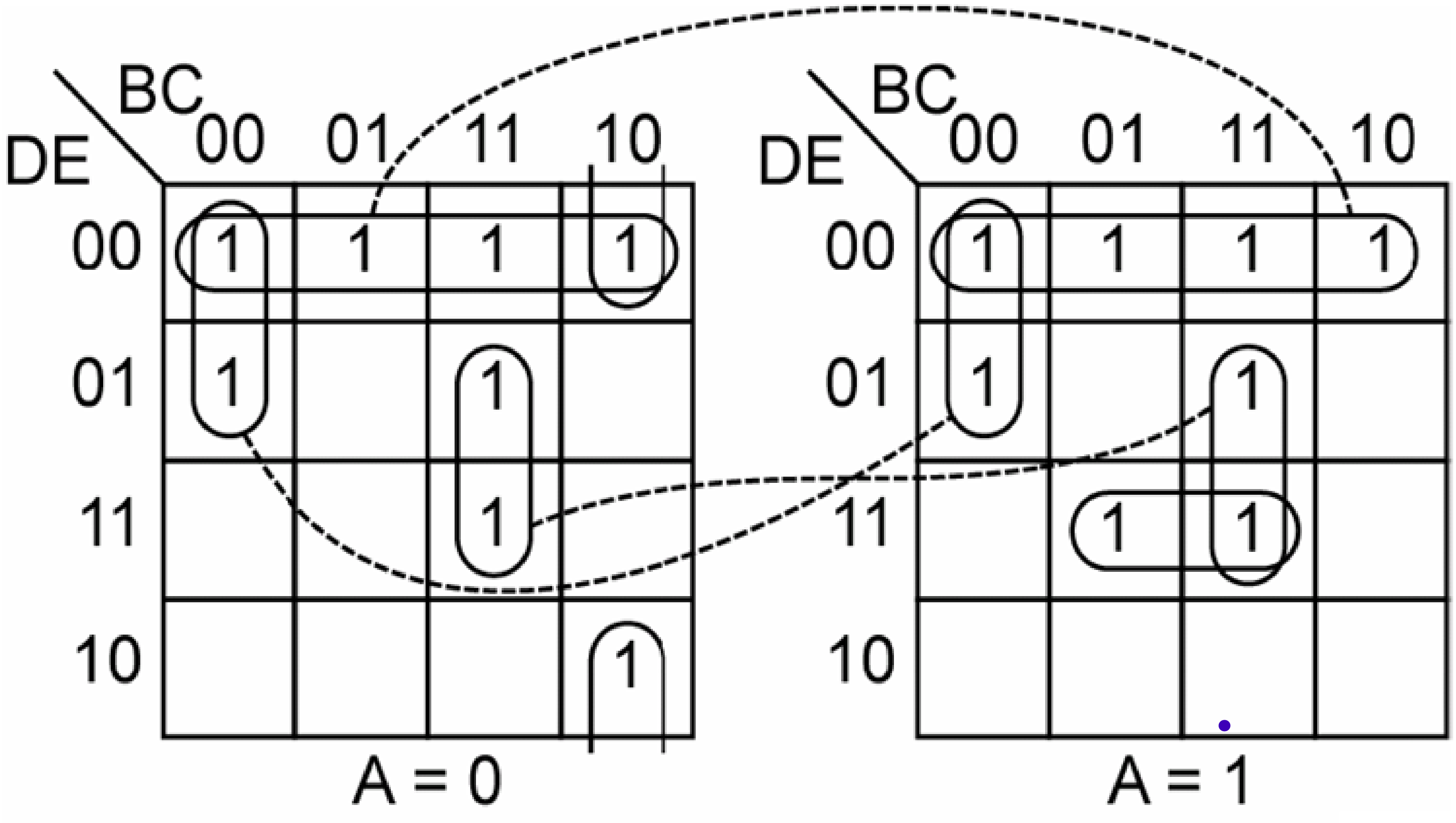
F(V,W,X,Y,Z) = $\sum(0,2,4,6,9,13,21,23,25,29,31)$



$F(v, w, x, y, z) = \sum(0, 2, 4, 6, 9, 13, 21, 23, 25, 29, 31)$

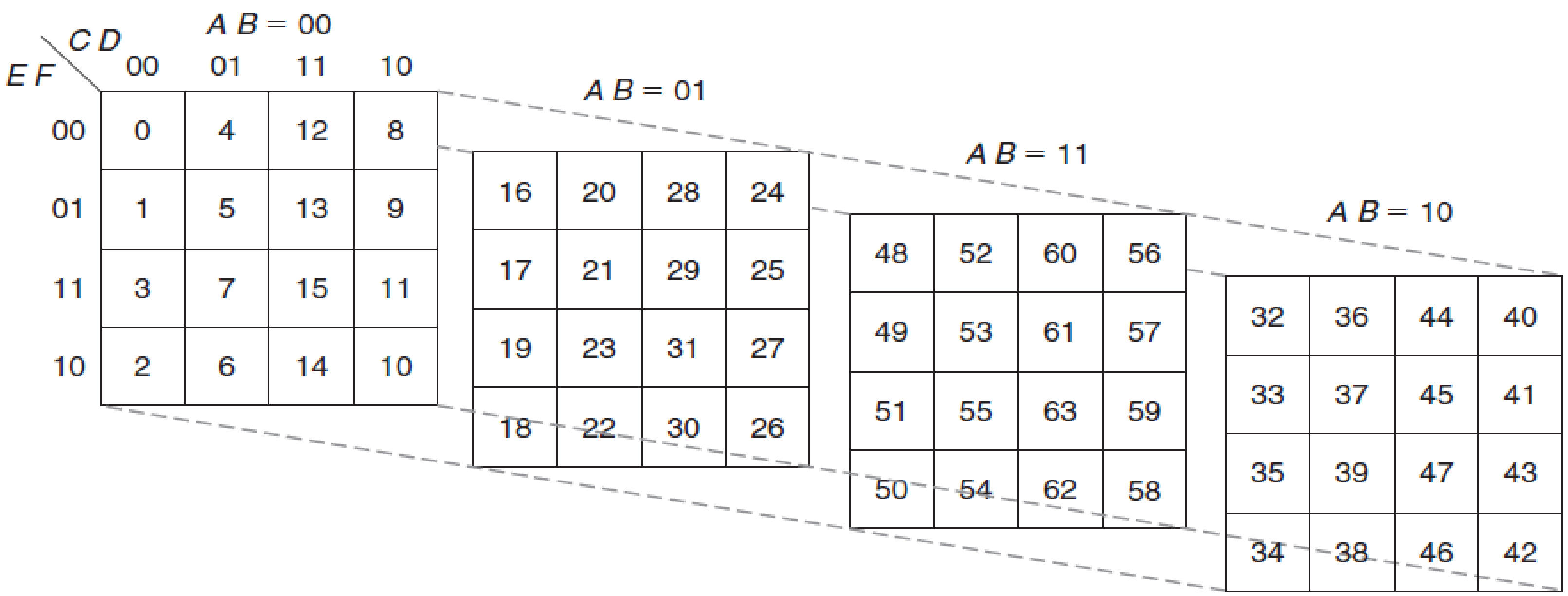
F = v′w′z′ + wy′z + vxz

Minimization Example of 5-Variable Map



F = D'E' + B'C'D' + BCE + A'BC'E' + ACDE

6-Variable Map Patterns



- A six-variable map holds sixty four minterms for six variables.
  - We use four four variable maps with two of the variables distinguishing between the four.
  - Each square in the first map is adjacent to the corresponding square in the second map (i.e. 4 and 20 are adjacent) and in the fourth map(i.e. 4 and 36 are adjacent). It is just like placing one map on the top of the other.

6-Variable Map Patterns

	D'E'F'	D'E'F	D'EF	D'EF'	DEF'	DEF	DE'F	DE'F'
A'B'C'	m_0	m_1	m_3	m_2	m_6	m_7	m_5	m_4
A'B'C	m_8	m_9	m_{11}	m_{10}	m_{14}	m_{15}	m_{13}	m_{12}
A'BC	m_{24}	m_{25}	m_{27}	m_{26}	m_{30}	m_{31}	m_{29}	m_{28}
A'BC'	m_{16}	m_{17}	m_{19}	m_{18}	m_{22}	m_{23}	m_{21}	m_{20}
ABC'	m_{48}	m_{49}	m_{51}	m_{50}	m_{54}	m_{55}	m_{53}	m_{52}
ABC	m_{56}	m_{57}	m_{59}	m_{58}	m_{62}	m_{63}	m_{61}	m_{60}
AB'C	m_{40}	m_{41}	m_{43}	m_{42}	m_{46}	m_{47}	m_{45}	m_{44}
AB'C'	m_{32}	m_{33}	m_{35}	m_{34}	m_{38}	m_{39}	m_{37}	m_{36}

6-Variable Map Patterns

- The number of adjacent squares that may be combined always represent a number that is a power of 2 such as 1, 2, 4, 8, 16, 32, and 64.
  - One square represents one minterm with six literals.
  - Two adjacent squares represent a term of five literals.
  - Four adjacent squares represent a term of four literals.
  - Eight adjacent squares represent a term of three literals.
  - Sixteen adjacent squares represent a term of two literals.
  - Thirty-two adjacent squares represents a term of one literal.
  - Sixty-four adjacent squares represent the entire map and produce a function that is always equal to 1.

Minimization Example of 6-Variable Map

Example: Simplify the Boolean function

F(A,B,C,D,E,F)=
∑(9,11,13,15,16,18,19,25,27,29,31,41,43,45,47,50,51,57,59,61,63)

K-maps become 3D for 5 & 6 variables

AB = 00

EF \ CD	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	0	0	1	1
10	0	0	0	0

AB = 01

EF \ CD	00	01	11	10
00	1	0	0	0
01	0	0	1	1
11	1	0	1	1
10	1	0	0	0

AB = 11

EF \ CD	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	1	0	1	1
10	1	0	0	0

AB = 10

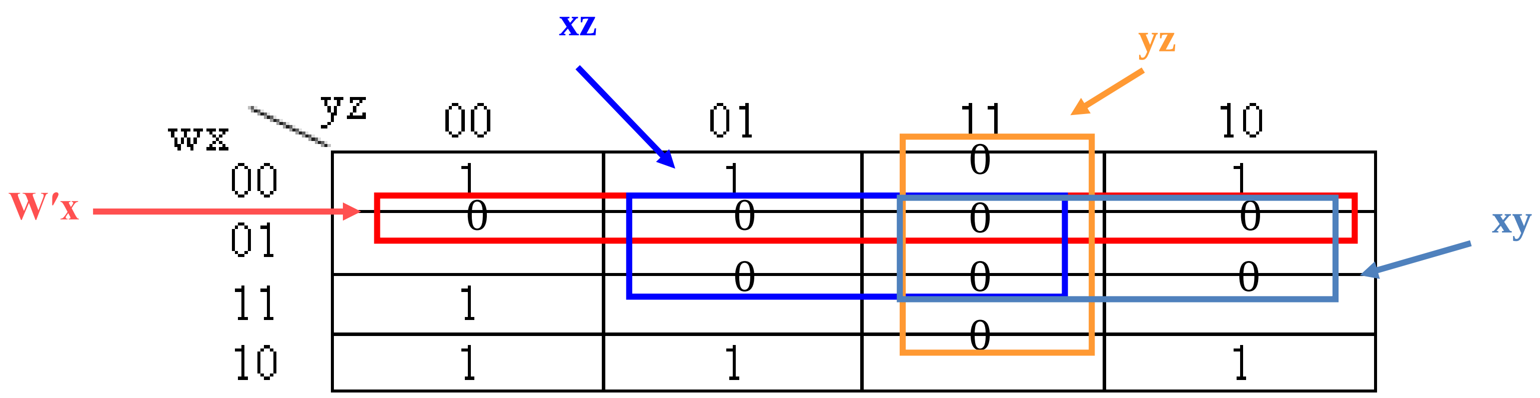
EF \ CD	00	01	11	10
00	0	0	0	0
01	0	0	1	1
11	0	0	1	1
10	0	0	0	0

OUTPUT =
A'BC'D'F' +
CF + BC'D'E

Product of Sums Minimization

- By definition, all the squares in a map that are not marked with a 1 represent the complement of the function.
  - If we mark the empty squares with 0s and then combine the zeros into valid adjacent squares, we obtain a simplified expression of the complement of the function i.e., F′
  - The complement of F′ [as (F′)′ = F] by DeMorgan's theorem (by taking the dual and complementing each literal, section 2-4), gives us the product of sums form

POS Minimization Example



$$F(w, x, y, z) = \sum(0, 1, 2, 8, 9, 10, 12)$$

$F' = w'x + yz + xz + xy$

$F = (F')' = (w'x + yz + xz + xy)' = (w + x')(y' + z')(x' + z')(x' + y')$

Example 3-8

- $F = \Sigma(0,1,2,5,8,9,10)$

  Simplify the function in

  - sum of products (SOP)
  - Product of sums (POS)

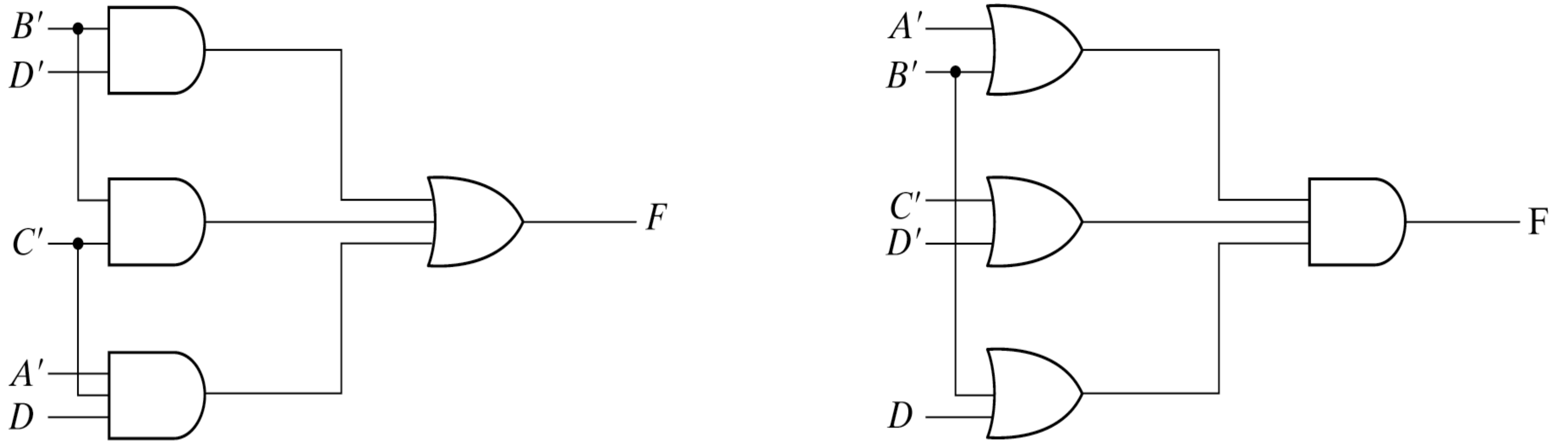
- Solution:
  - The squares marked with 1's represents minterms and are combined to form simplified function in sum of products (SOP). $F=B'D'+B'C'+A'C'D$
  - If the squares marked with 0's are are combined we obtain the simplified complemented function $F'=AB+CD+BD'$
  - Applying DeMorgan's theorem we obtain the simplified function in product of sum form (POS) $F=(A'+B')(C'+D')(B'+D)$

AB \ CD	00	01	11	10
00	1	1	0	1
01	0	1	0	0
11	0	0	0	0
10	1	1	0	1

Fig. 3-14 Map for Example 3-8; $F(A,B,C,D) = \Sigma(0,1,2,5,8,9,10)$
$= B'D' + B'C' + A'C'D = (A' + B')(C' + D')(B' + D)$

SOP and POS Gate Implementation

Two-level logic diagrams



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

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

Fig. 3-15 Gate Implementation of the Function of Example 3-8

Listing Truth Table using SOP and POS

A	B	C	D	F_1	F_2
1	1	1	1	0	0
1	1	1	0	0	0
1	1	0	1	0	0
1	1	0	0	0	0
1	0	1	1	0	0
1	0	1	0	1	1
1	0	0	1	1	1
1	0	0	0	1	1
0	1	1	1	0	0
0	1	1	0	0	0
0	1	0	1	1	1
0	1	0	0	0	0
0	0	1	1	0	0
0	0	1	0	1	1
0	0	0	1	1	1
0	0	0	0	1	1

$\mathbf{F_1} = B'D' + B'C' + A'C'D$

$\mathbf{F_2} = (A' + B')(C' + D')(B' + D)$

Working With Maxterms

- At times, we may be required to work with maxterms.
  - ➢ The previous process actually worked with minterms. Remember that the numbers used for minterms are the opposites of the numbers used for maxterms:
    - o $F(w, x, y, z) = \sum(0, 1, 2, 8, 9, 10, 11)$, uses minterms
    - o $F(w, x, y, z) = \pi(3, 4, 5, 6, 7, 12, 13, 14, 15)$, uses maxterms
  - ➢ If you are given minterms, fill in 1's for the minterms and then fill the remaining cells with 0's
  - ➢ If you are given maxterms, fill in 0's for the maxterms and then fill the remaining cells with 1's
  - ➢ For SOP simplification, solve the map for the 1's
  - ➢ For POS simplification, solve the map for the 0's to get complemented function. Taking the complement of this complemented function we obtain function in POS form

Don’t Care Conditions

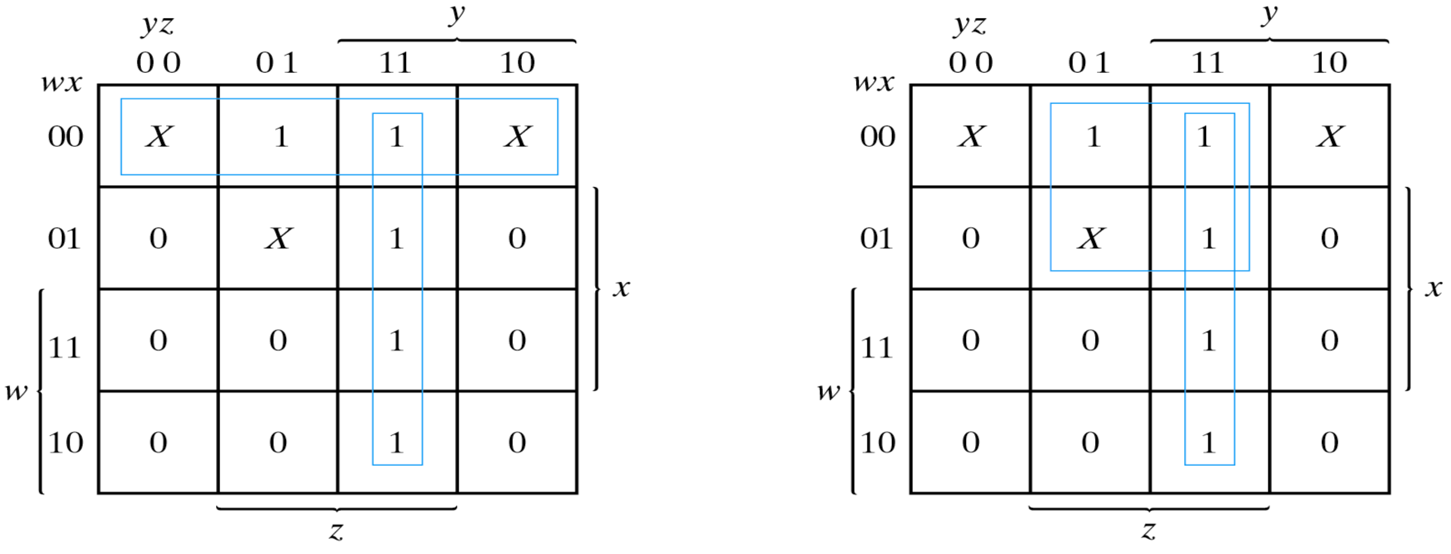
- So far, we have always assumed that all combinations of the input values are necessary in our expressions.
- Sometimes there are unspecified combinations within a function.
    - For example, four bit binary coded decimal code has six combinations that are not used.
- Functions that have unspecified outputs for some input combinations are called incompletely specified functions.
    - These are called don’t care conditions because in most applications, we do not care what the specification of the combination is and not concerned about the function output for these combinations..

Indicating Don’t Care Conditions

- A don’t care condition cannot be specified with a 1 because it would require the function to always be 1 for the combination.
- Likewise, a don’t care condition cannot be specified with a 0 because it would require the function to always be 0 for the combination.
- To specify don’t care conditions in a map, we use the letter ‘x’ or ‘d’.
  - ➢ When we choose adjacent squares to simplify the map, the don’t care minterms can be assumed to be 0 or 1, whichever leads to the simplest expression.

Simplification with Don’t Care Conditions

- Example 3.9: Simplify the Boolean function: $F(w,x,y,z) = \sum_m(1,3,5,9,13)+\sum_d(0,2,7)$



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

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

Fig. 3-17 Example with don’t-care Conditions

$F_1 = w'x'+y'z = \sum_m(0, 1, 2, 3, 5, 9, 13)$

$F_2 = w'z+y'z = \sum_m(1, 3, 5, 7, 9, 13)$

More Examples with Don’t Care

AB \ CD	00	01	11	10
00	0	1	0	0
01	x	x	x	1
11	1	1	1	x
10	x	0	1	1

$F = A'C'D + B + AC$

AB \ CD	00	01	11	10
00	0	1	0	0
01	x	x	x	1
11	1	1	1	x
10	x	0	1	1

$F = A'B'C'D + ABC' + BC + AC$

The End