DIGITAL ELECTRONICS

CHAPTER THREE: *Simplification of Boolean Expressions and Combinational Logic Circuit Design (Part I)

By J. Mathenge

Previously on Boolean Algebra Simplification:

Example 6

- A student may register for course X only if he satisfies the following conditions:
- (1) Has completed at least 20 courses AND is a Computing student AND of good conduct, OR
- (2) Has completed at least 20 courses AND is a Computing student AND has departmental approval, OR
- (3) Has completed fewer than 20 courses AND is a Computing student AND not of good conduct, OR
- (4) Is of good conduct AND has departmental approval, OR
- (5) Is a Computing student AND does not have departmental approval.

Solution

Let the listed conditions be represented by the letters below:

- A: Has completed at least 20 courses
- B: Is a Computing student
- C: Is of good conduct
- D: Has departmental approval
- Y: Student may register for course X

Y=ABC+ABD+A`BC`+CD+BD`

We can now simplify this expression.

Solution

Y=ABC+ABD+A`BC`+CD+BD`

Y=ABC+A`BC`+CD+B{D`+DA}

Y=ABC+A`BC`+CD+B{D`+A}

Y=ABC+A`BC`+CD+BD`+BA

Y=AB{C+1}+A`BC`+CD+BD`

Y=AB+A`BC`+CD+BD`

Y=B{A+A`C`}+CD+BD`

Y=B{A+C`}+CD+BD`

Y=AB+BC`+CD+BD` ***Rewrite DC+D`B** to match theorem #8

Y=AB+BC`+DC+D`B+CB=AB+BC`+CD+BD`+CB

Y=AB+B{C`+C}+CD+BD`=AB+B+CD+BD`=B(A+1)+CD+BD`

Y=B+CD+BD`=B{D`+1}+CD = B+CD

Using the theorems

5: $A + \overline{A} \cdot B = A + B$

6: $A \cdot B + A \cdot \overline{B} = A$

8: $A \cdot B + \overline{A} \cdot C + B \cdot C = A \cdot B + \overline{A} \cdot C$

Hence a student may register for the course X if he is a Computing student OR he is of good conduct AND has departmental approval.

3.1.2 The Karnaugh Map

Introduction:

- This is a graphical method used to simplify a Boolean expressions. It represents the information in a truth-table in a different format. Each combination of inputs is represented by a cell in the map.
- Once a Karnaugh Map (K-map) has been filled with ones and zeros, the Sum of Products expression can be obtained by ORing together the those squares that contain 1s. The Product of Sums expression can be obtained by ANDing together those squares that contain 0s.

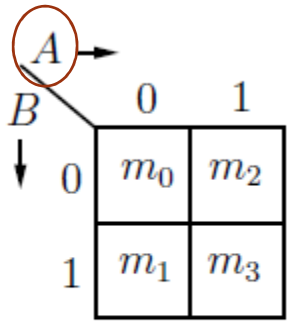
K-Map Illustration

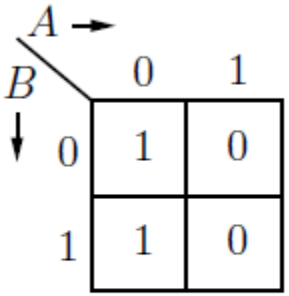
Two variable K-map

Consider the 2-variable truth-table shown below:

A	B	x	minterm
0	0	1	m_0
0	1	1	m_1
1	0	0	m_2
1	1	0	m_3

- There are four input combinations, so this truth-table can be converted to a K-map with 4 cells represented as:





Recommended Arrangement of Variables

Figure 3.3: Two variable K-map

K-Map Illustration

Or represented as:

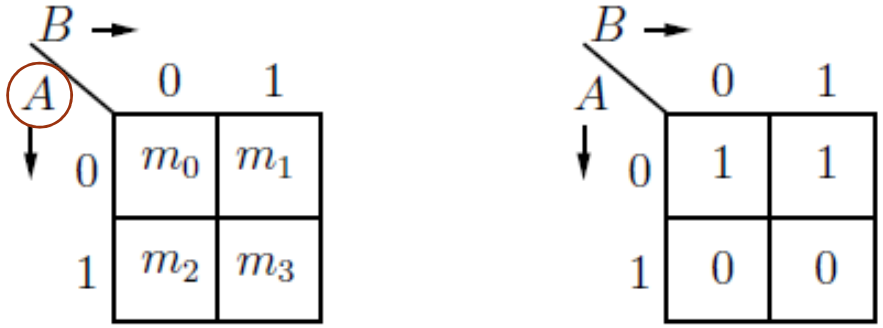


Figure 3.4: Two variable K-map: alternative representation

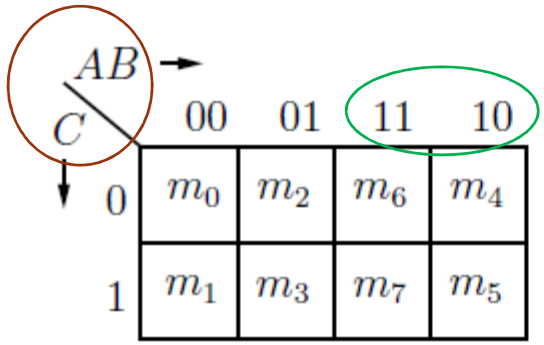
K-Map Illustration

Three variable K-map

Given the truth table:

A	B	C	x	minterm
0	0	0	1	m_0
0	0	1	1	m_1
0	1	0	1	m_2
0	1	1	0	m_3
1	0	0	0	m_4
1	0	1	0	m_5
1	1	0	1	m_6
1	1	1	0	m_7

**Note that the K- map cells are labelled in such a way that adjacent cells differ only in one variable.



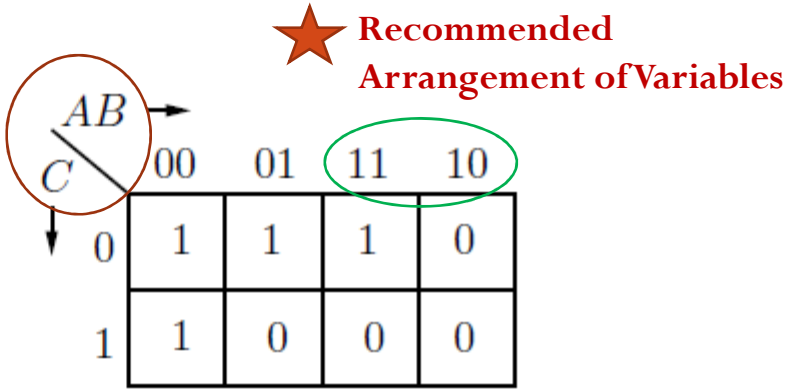


Figure 3.5: Three variable K-map

K-Map Illustration

An alternative representation of the 3 variable K-Map is as shown below:

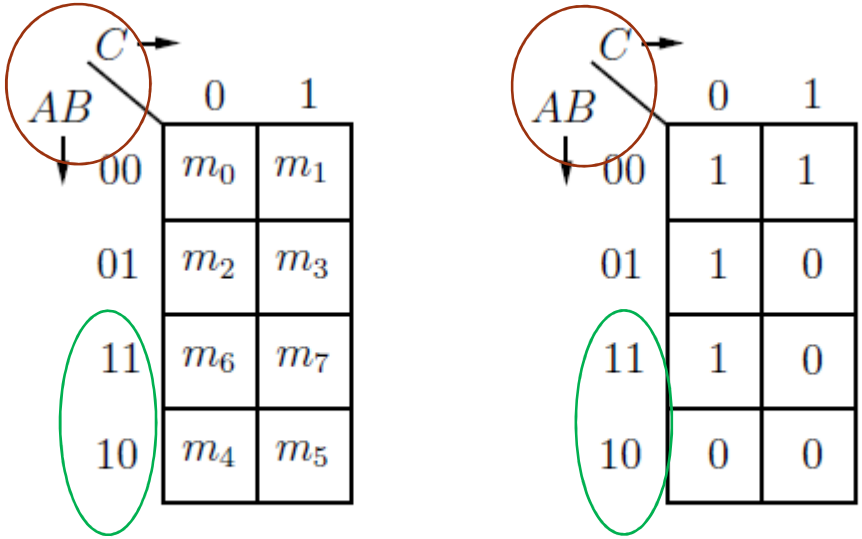


Figure 3.6: Three variable K-map: alternative representation

- The two representations shown (Figure 3.5 and Figure 3.6) are equivalent, and you may choose to work with whichever representation that you are most comfortable with.

K-Map Illustration

4-variable K-map

Consider the 4-input truth-table shown below:

A	B	C	D	x	minterm
0	0	0	0	0	m_0
0	0	0	1	1	m_1
0	0	1	0	0	m_2
0	0	1	1	0	m_3
0	1	0	0	0	m_4
0	1	0	1	1	m_5
0	1	1	0	0	m_6
0	1	1	1	0	m_7
1	0	0	0	0	m_8
1	0	0	1	0	m_9
1	0	1	0	0	m_{10}
1	0	1	1	0	m_{11}
1	1	0	0	0	m_{12}
1	1	0	1	1	m_{13}
1	1	1	0	0	m_{14}
1	1	1	1	1	m_{15}

Recommended Arrangement of Variables

CD \ AB	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}

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

Figure 3.7: Four variable K-map

K-Map Illustration

An alternative representation of the 4 variable K-Map is as shown below:

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

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

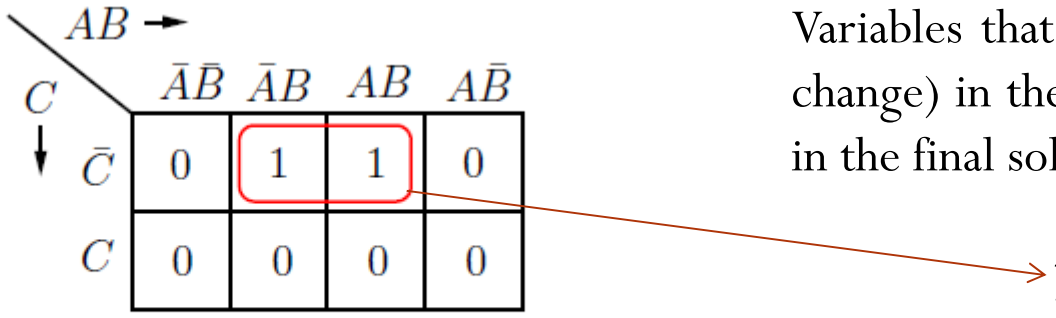
Figure 3.8: Four variable K-map: alternative representation

3.1.3 Looping

- A Logic function can be simplified by properly combining those squares in the that contain 1s.
- This process of combining 1s is called looping. The number of 1s that can be looped together should be *a power of 2* (1, 2, 4, 8, 16 etc.).

Groups of 2 (Pairs)

- Two ones can be looped together if they are *horizontally* or *vertically* adjacent. CANNOT be looped diagonally.



Variables that are the same (do not change) in the loop are what appear in the final solution e.g.

$$Y = B\bar{C}$$

Figure 3.9: Looping two 1s which are horizontally adjacent

3.1.3 Looping

.

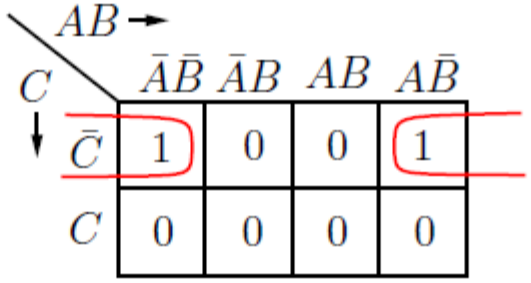
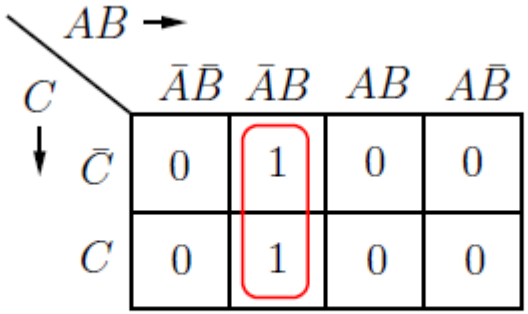


Figure 3.11: Looping two 1s which are horizontally adjacent

In this case,

$$x = \bar{B}\bar{C}$$

Looping two 1's that are vertically adjacent:



$$Y = \bar{A}B$$

Figure 3.10: Looping two 1s which are vertically adjacent

3.1.3 Looping

Groups of 4 (Quads)

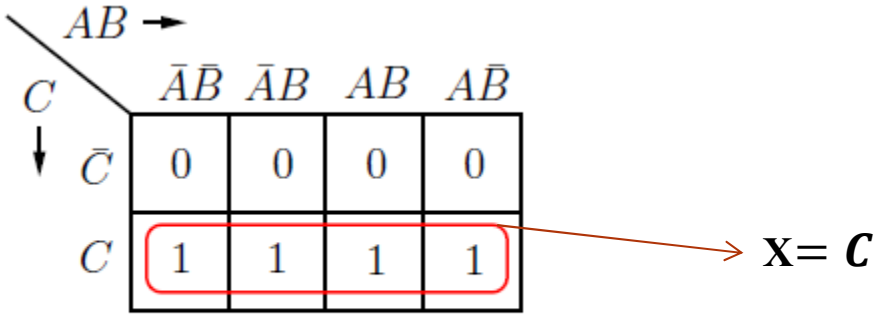


Figure 3.12: Looping four 1s which are horizontally adjacent

3.1.3 Looping

Example

Consider the K-map shown on Figure 3.13.

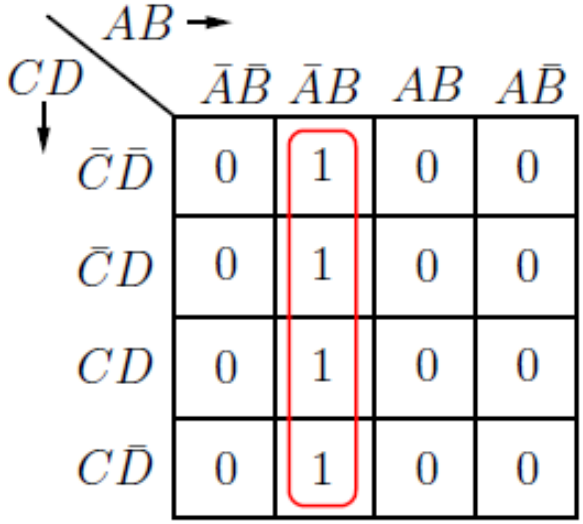


Figure 3.13: Looping four 1s which are vertically adjacent

The four 1s are vertically adjacent and are looped together to give:

$$x = \bar{A}B$$

3.1.3 Looping

Example

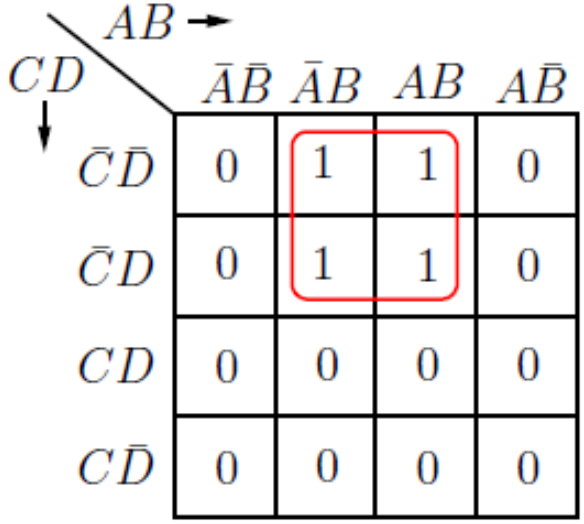
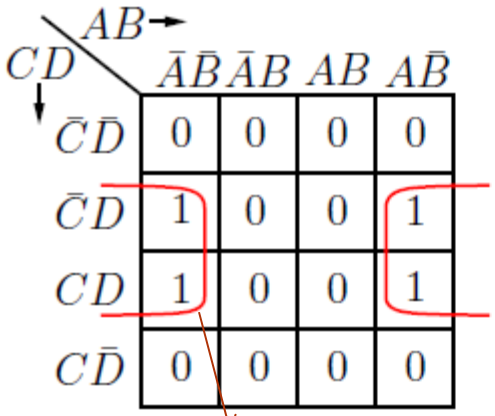


Figure 3.14: Looping four 1s which form a square

The four 1s in Figure 3.14 form a square and are looped together to give:

$$x = B\bar{C}$$

3.1.3 Looping



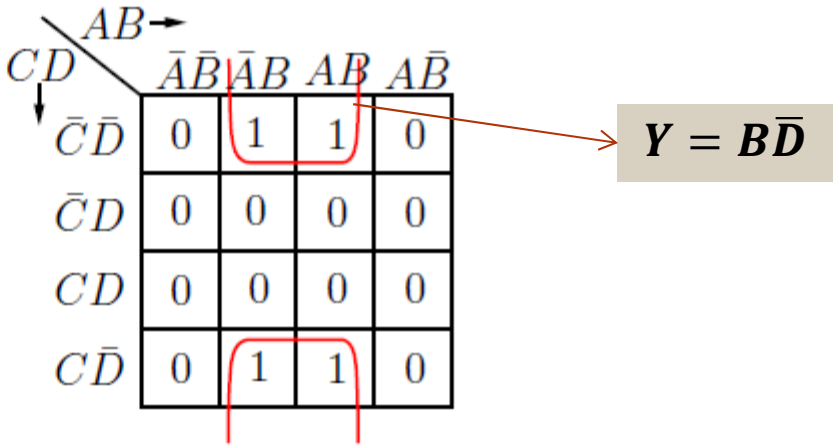
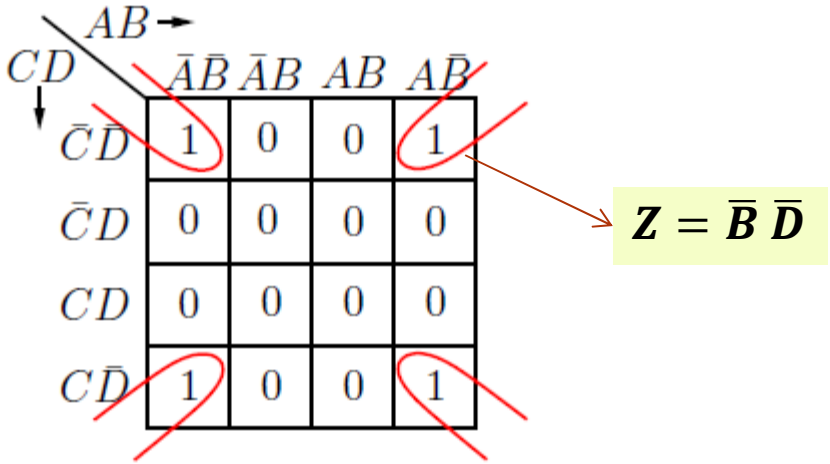


Figure 3.15: Looping four 1s



3.1.3 Looping

Groups of 8 (octets)

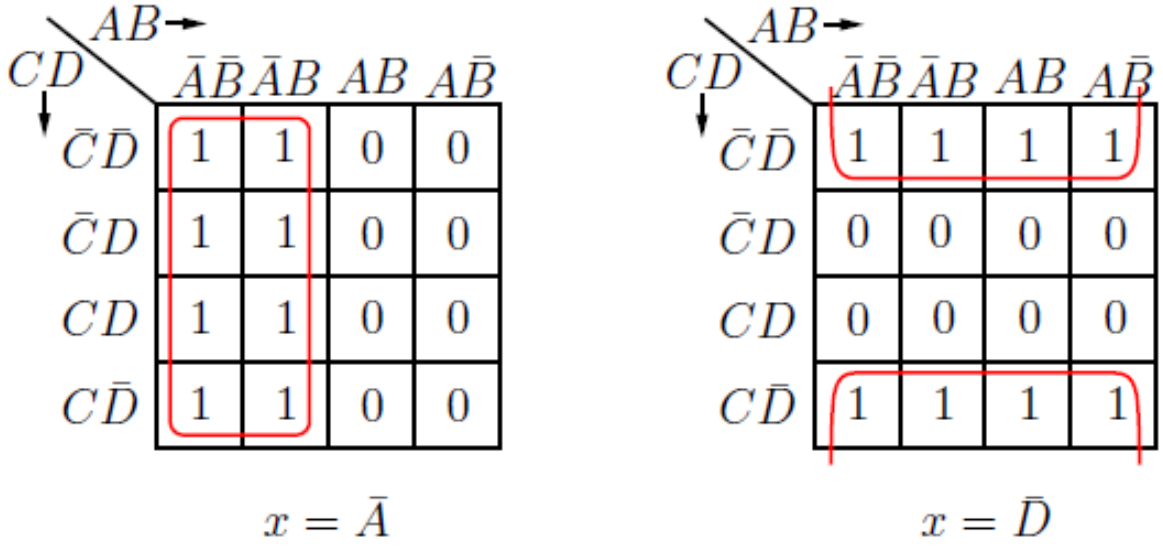


Figure 3.17: Looping eight 1s (octets)

3.1.3 Looping: Examples

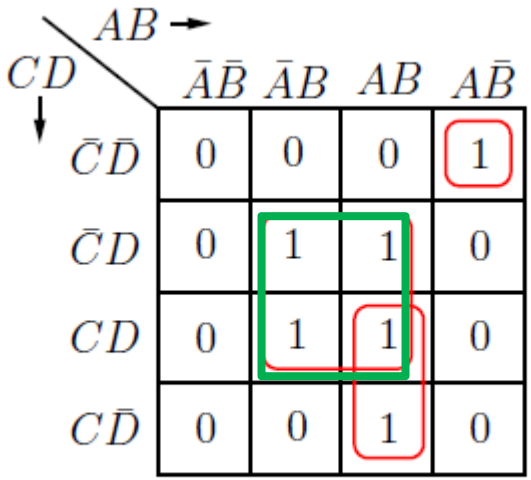


Figure 3.18: Example

From Figure 3.18, $x = A\bar{B}\bar{C}\bar{D} + ABC + BD$

3.1.3 Looping: Examples

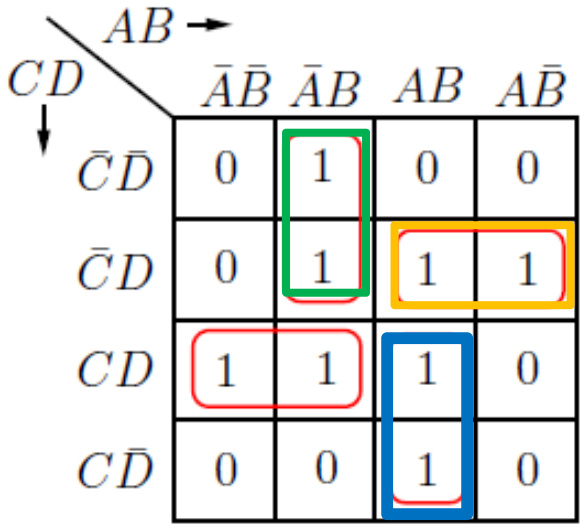
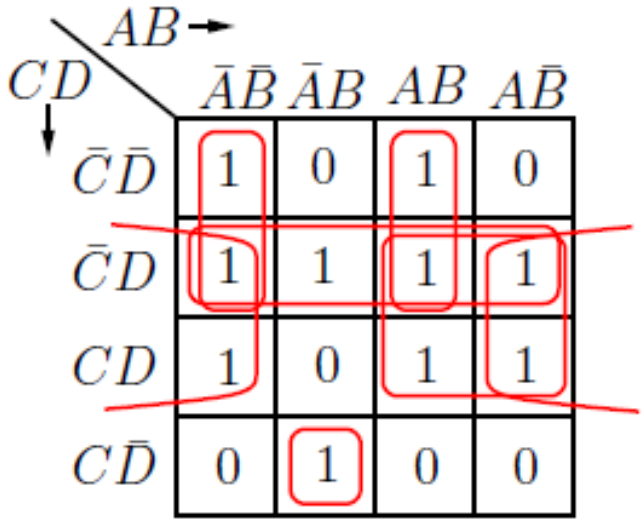


Figure 3.19: Example

From Figure 3.19, $x = \bar{A}B\bar{C} + A\bar{C}D + ABC + \bar{A}CD$

3.1.3 Looping: Examples

- We have another example as below: (*Draw Here*)



From Figure 3.20, $x = \bar{A}BC\bar{D} + \bar{A}\bar{B}\bar{C} + AB\bar{C} + \bar{C}D + \bar{B}D + AD$

3.1.4 Complete Simplification Procedure

1. Construct the K-map and place 1s in those cells corresponding to 1s in the truth-table. Place 0s in the other squares.
2. Identify those 1s that can be looped in groups of 16.
3. Repeat the same for groups of 8 then 4 and finally 2.
4. Loop any pairs necessary to include any 1s that have not yet been looped, making sure to use the minimum number of loops.
5. Examine the map and loop those 1s which are not adjacent to any other 1s.these are called *isolated 1s*.
6. Form the OR sum of all the terms generated by each loop.

Example 1

- Minimize the following Boolean function:

$F(A,B,C,D)=\{m(0,1,2,5,7,8,9,10,13,15)\}$

A	B	C	D	OUTPUT
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	1

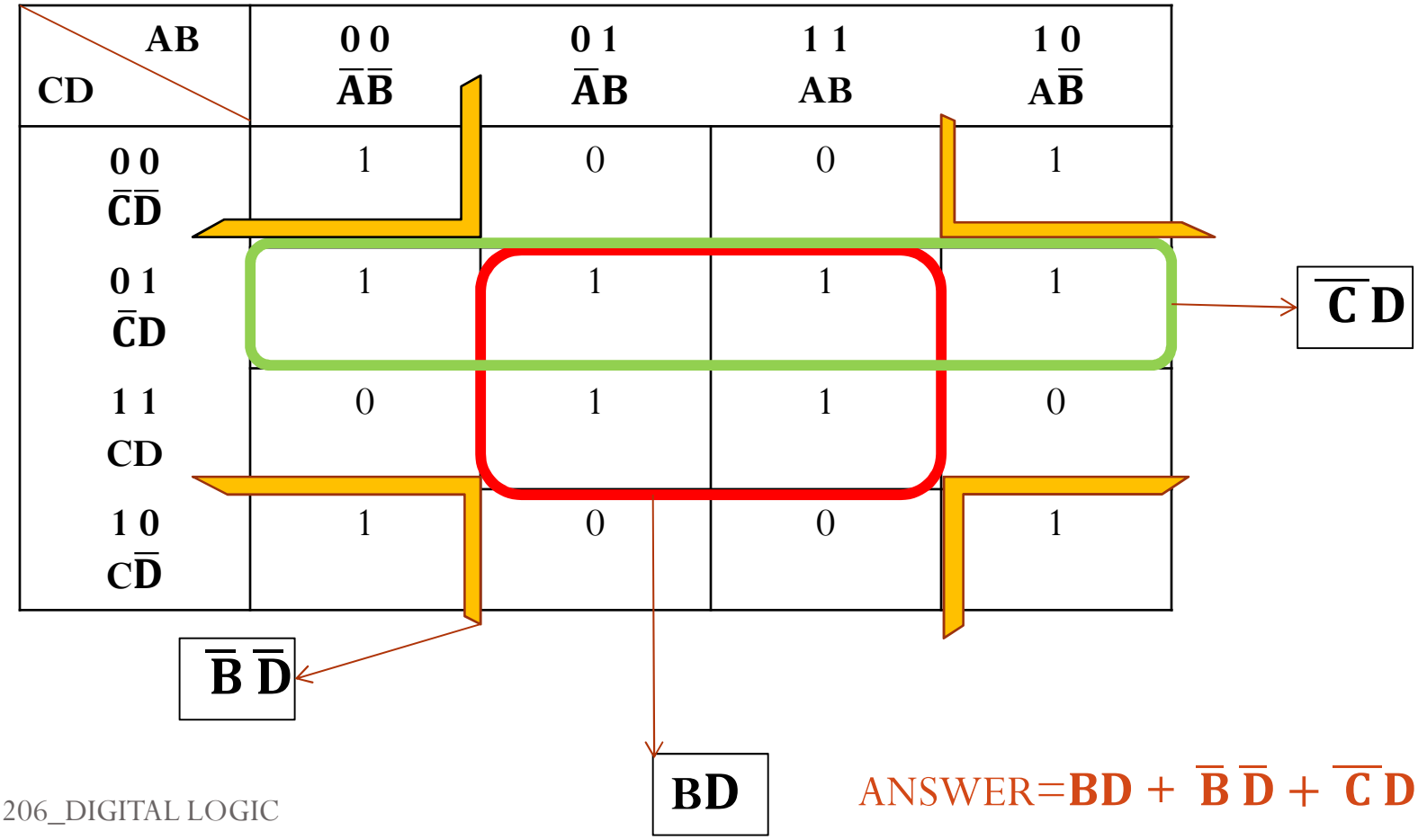
Solution

Since the expression has 4 variables, we draw a $2^n = 2^4$ K-map. Fill in the 1's as per the truth table.

CD \ AB	0 0 $\bar{A}\bar{B}$	0 1 $\bar{A}B$	1 1 AB	1 0 $A\bar{B}$
0 0 $\bar{C}\bar{D}$	1	0	0	1
0 1 $\bar{C}D$	1	1	1	1
1 1 CD	0	1	1	0
1 0 $C\bar{D}$	1	0	0	1

Solution

Looping…



ANSWER=$BD + \bar{B}\bar{D} + \bar{C}D$

Example 2

- Minimize the following Boolean function:

$F(A,B,C,D)=\{m(0,1,3,5,7,8,9,11,13,15)\}$

A	B	C	D	OUTPUT
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

Solution

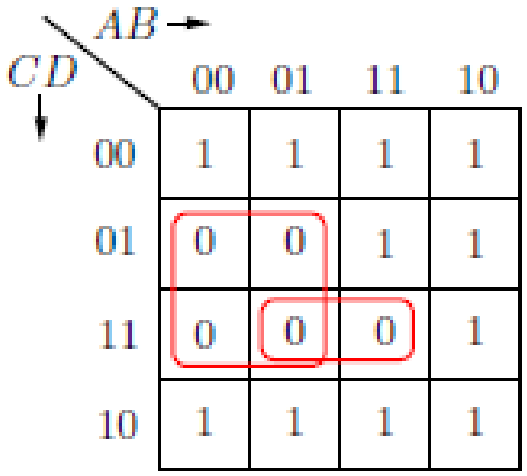
Since the expression has 4 variables, we draw a $2^n = 2^4$ K-map. Fill in the 1's as per the truth table.

CD \ AB	0 0 $\overline{A}\overline{B}$	0 1 $\overline{A}B$	1 1 AB	1 0 $A\overline{B}$
0 0 $\overline{C}\overline{D}$				
0 1 $\overline{C}D$				
1 1 CD				
1 0 $C\overline{D}$				

ANSWER= $\overline{B}\,\overline{C} + D$

3.1.5 Obtaining Product of Sums Expressions from K-maps

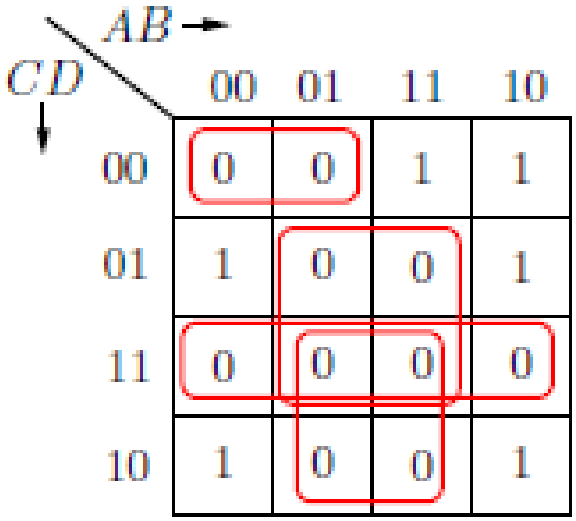
- So far, we have talked about obtaining Sum of Products expressions from K-maps. It is also possible to obtain Product of Sums expressions . only this time we loop the zeros together, not the ones. The examples below illustrate the procedure.



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

ANSWER=(A+D)(B+C+D)

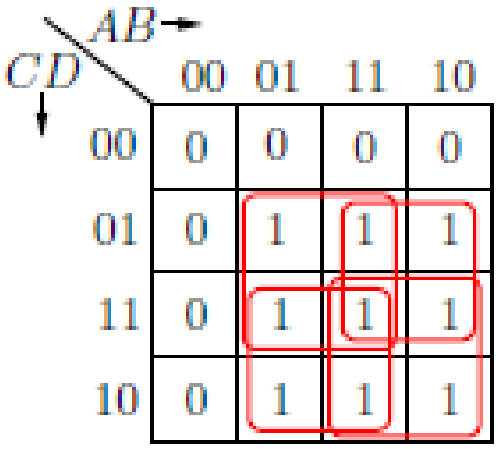
3.1.5 Obtaining Product of Sums Expressions from K-maps



ANSWER=(A+C+D)($\bar{B}$+$\bar{D}$)($\bar{C}$+$\bar{D}$)($\bar{B}$+$\bar{C}$)

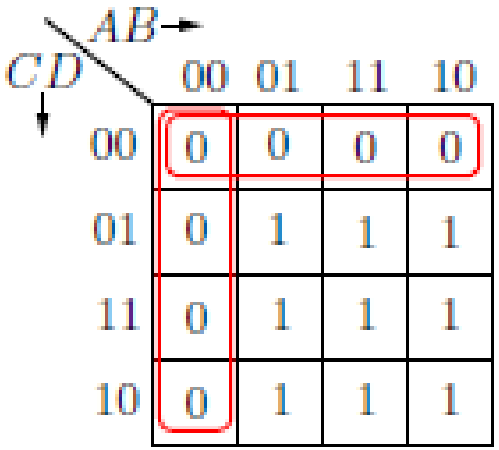
3.1.5 Obtaining Product of Sums Expressions from K-maps

- Note that for a given K-map, looping the 1s and looping the zeros gives the same results, only that the results are expressed in different ways. Consider the example shown below where in one case the 1s are looped and in the other case the 0s are looped.



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

ANSWER=AD+BD+BC+AC



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

ANSWER=(C+D)(A+B)

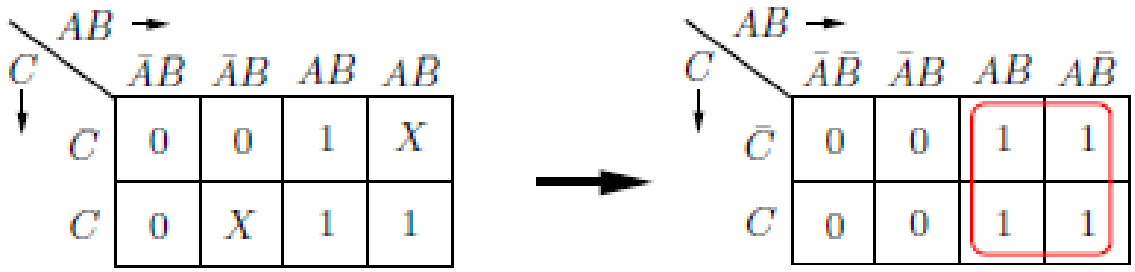
3.2 Don't Care Terms

- These are also referred to as Unused terms, Forbidden terms or Redundant terms. These terms describe combinations of variables which never occur in practice.
- In a truth-table or a K-map, these inputs are represented by an X, an R or a d.
- Example, suppose that we have a digital circuit with three inputs and one output, and the input combinations 000, 001 and 010 give an output 0, input combinations 101, 110 and 111 give an output 1, and input combinations 011 and 100 never occur in practice.
- Such a truth table is represented as:

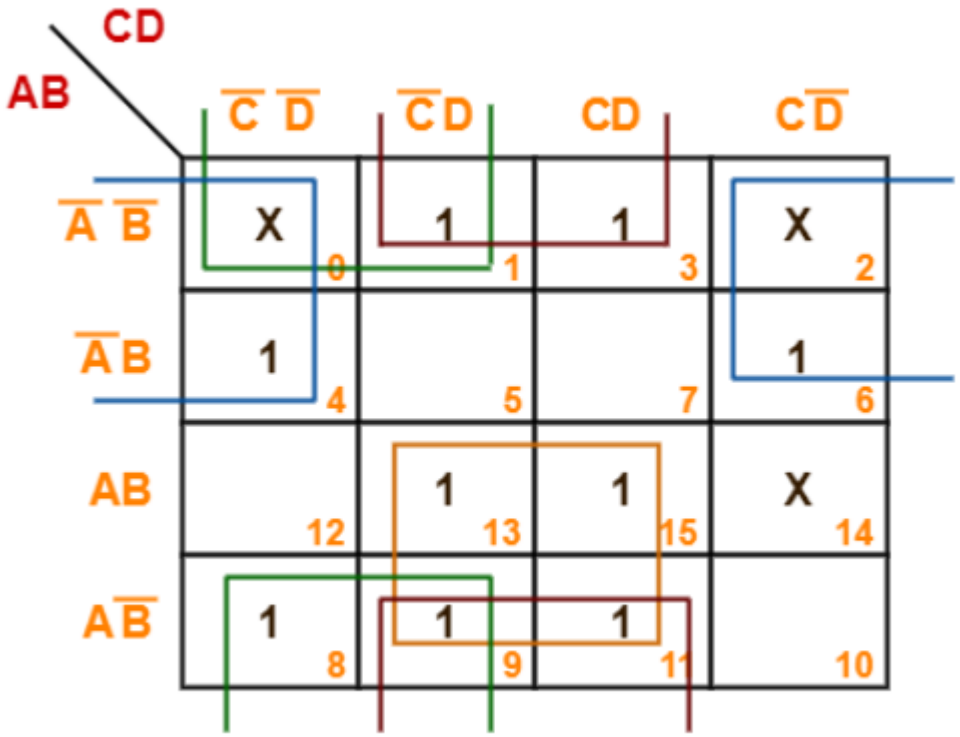
A	*B*	*C*	*output*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	*X*
1	0	0	*X*
1	0	1	1
1	1	0	1
1	1	1	1

3.2 Don't Care Terms

- When designing with K-maps containing don't care variables, the designer can make the output of any don't care condition either a 1 or a 0 in order to produce the simplest output expression.
- Example:



3.2 Don't Care Terms



F(A, B, C, D) = AD + B'D + B'C' + A'D'

3.2 Don't Care Terms

- Minimize the below Boolean expression:

$F(A, B, C) = \Sigma m(0, 1, 6, 7) + \Sigma d(3, 5)$

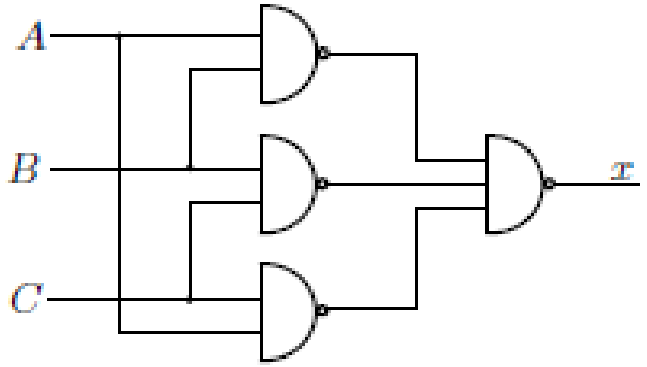
3.3 NAND/NOR gate circuit implementation

- To implement a logic circuit using NAND gates only:

1. Derive the minimized expression for the function in *sum of products form* (obtained by minimizing Boolean expressions algebraically or by looping 1s in a K-map).

2. Apply double negation and De Morgan's theorem to convert the expression in a form suitable for NAND gate implementation.

- Suppose a design problem results in a *minimized sum of products* expression: **AB+BC+AC**
- $X = AB + BC + AC = \overline{\overline{AB + BC + AC}} = \overline{\overline{AB} \cdot \overline{BC} \cdot \overline{AC}}$



Example: implementation using NAND gates only

3.3 NAND/NOR gate circuit implementation

- To implement a logic circuit using NOR gates only:

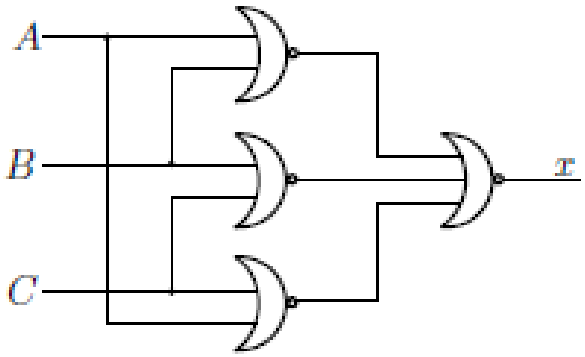
1. Derive the minimized expression for the function in product of sums form (obtained by looping the 0s in a K-map).

2. Apply double negation and De Morgan's theorem to convert the expression in a form suitable for NOR gate implementation.

- As an example, suppose a design problem resulted in a minimized product of sums expression: **(A+B) (B+C) (A+C)**

$$\mathbf{Y=(A+B)\,(B+C)\,(A+C)=\overline{\overline{(A+B)\,(B+C)\,(A+C)}}}$$

$$Y=\overline{\overline{(A+B)}+\overline{(B+C)}+\overline{(A+C)}}$$



Example: implementation using NOR gates only

End of session

Questions….?