Syllabus Content:

15.2 Karnaugh Maps

- show understanding of (K-Maps) Karnaugh Maps
- show understanding of the benefits of using Karnaugh Maps
- solve logic problems using Karnaugh Maps

3.3.3 Karnugh Maps

So far we can see that applying Boolean algebra can be awkward in order to simplify expressions. Apart from being laborious (and requiring the remembering all the laws) the method can lead to solutions which, though they appear minimal, are not.

What are Karnaugh maps?
A Karnaugh map provides a pictorial method of grouping together expressions with common factors and therefore eliminating unwanted variables. The Karnaugh map can also be described as a special arrangement of a truth table.

The diagram below illustrates the correspondence between the Karnaugh map and the truth table for the general case of a two variable problem.

2 Variable Truth Table

	A	B	F
0	0	0	0
1	0	1	1
2	1	0	1
3	1	1	1

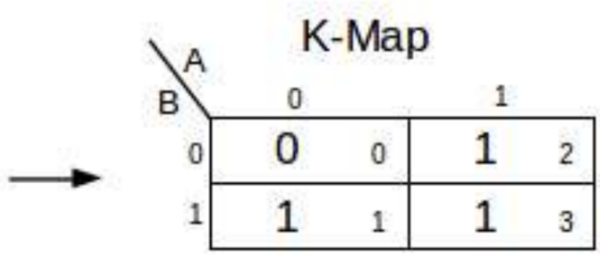
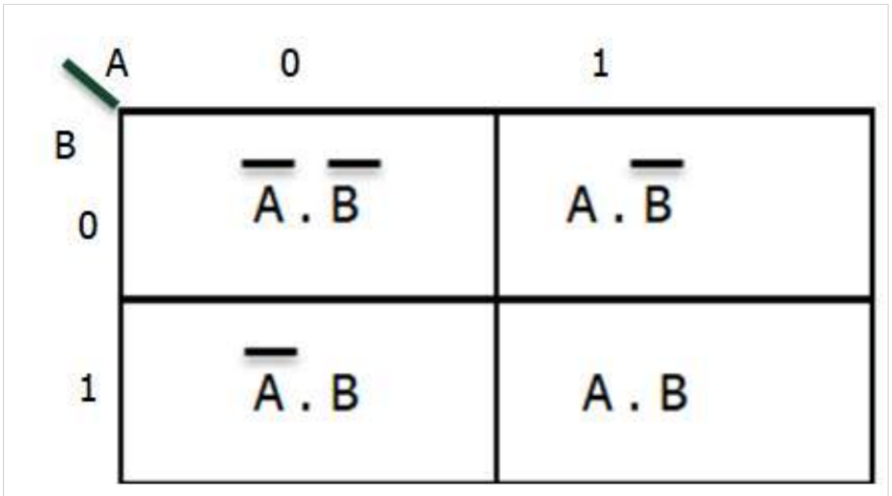
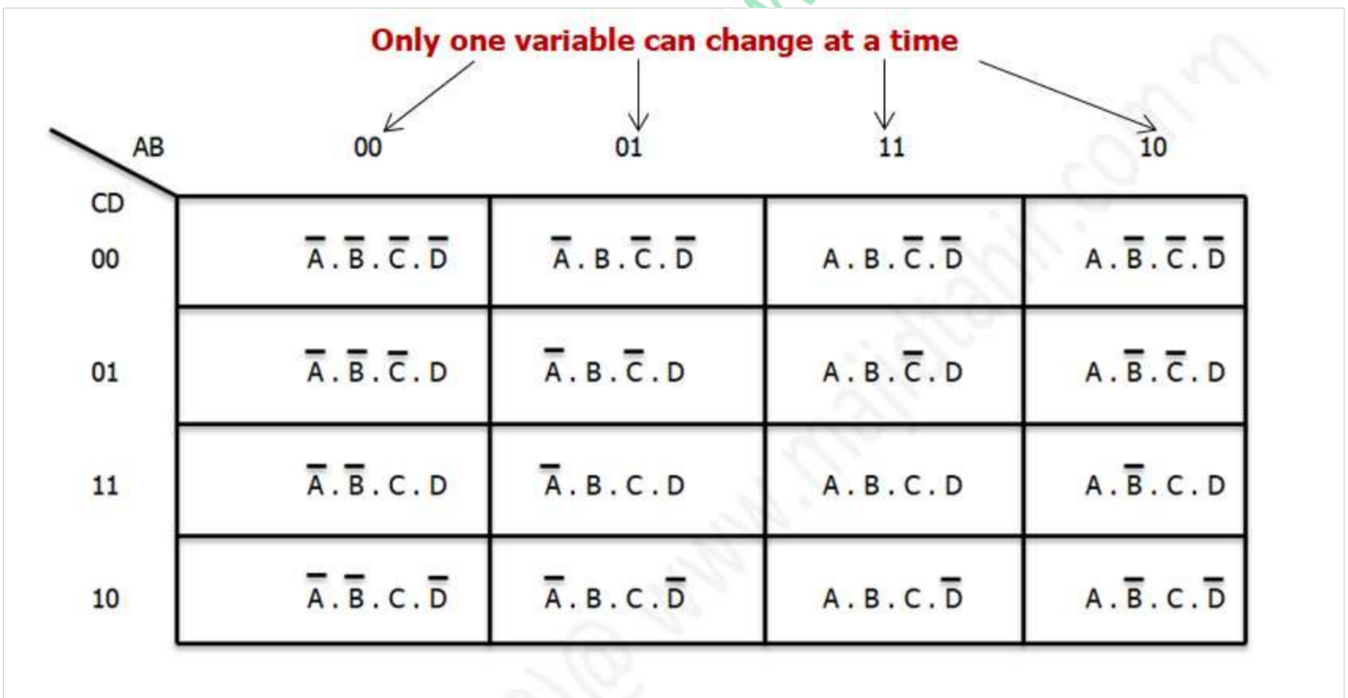


Figure 1: Variable Truth Table & K-Map



Two variable Karnugh map

We can extend this map designed to four variables as shown below:

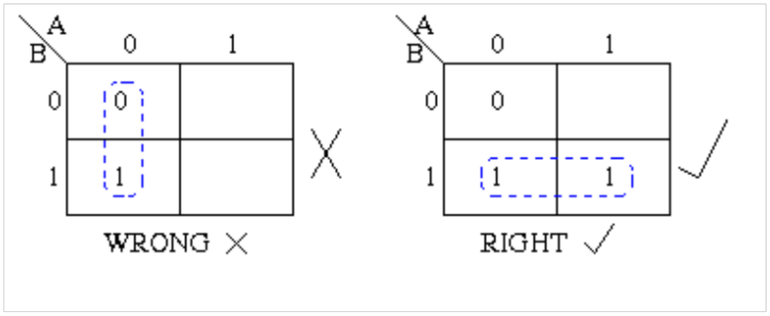


The order of the bit pair for the column and the rows is important. They use a system called **'Grey Code'**- each bit pair differs by only one bit change from the previous bit.

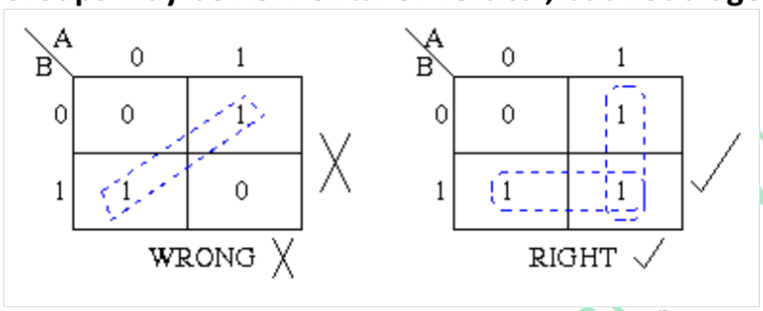
Karnaugh Maps - Rules of Simplification

The Karnaugh map uses the following rules for the simplification of expressions by *grouping* together adjacent cells containing *ones*

1. **Groups may not include any cell containing a zero**

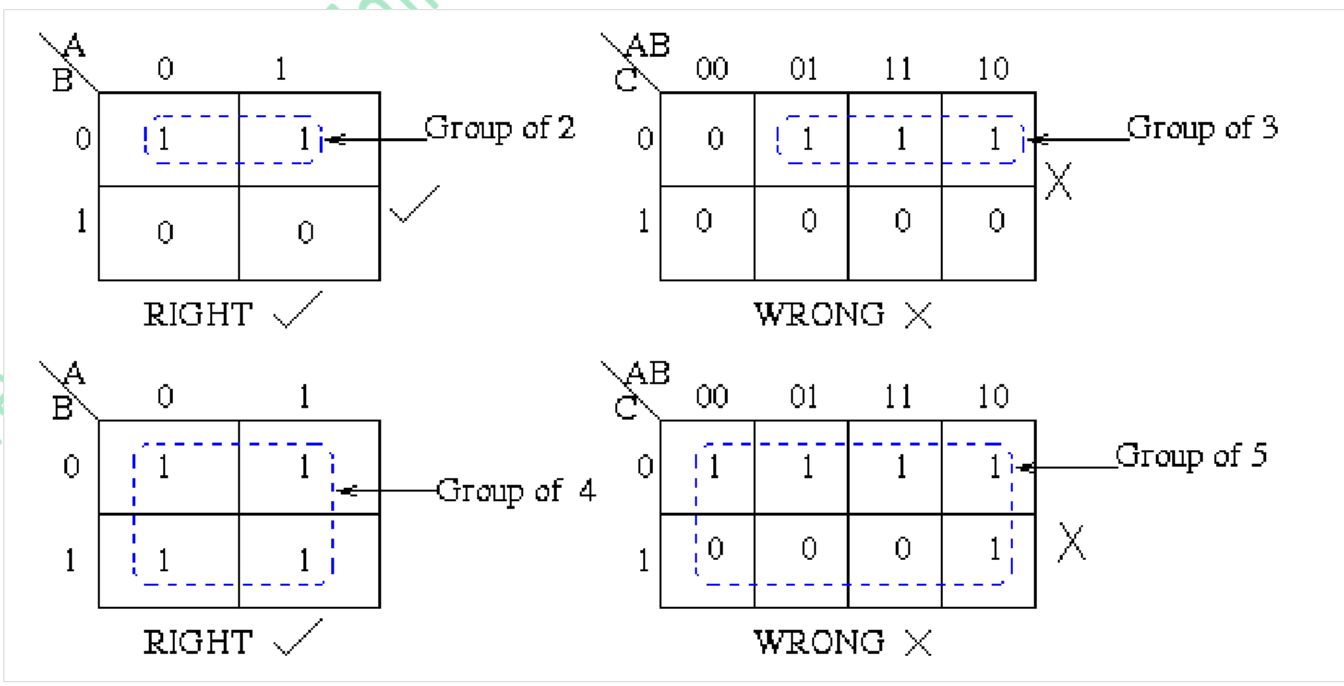


2. **Groups may be horizontal or vertical, but not diagonal.**

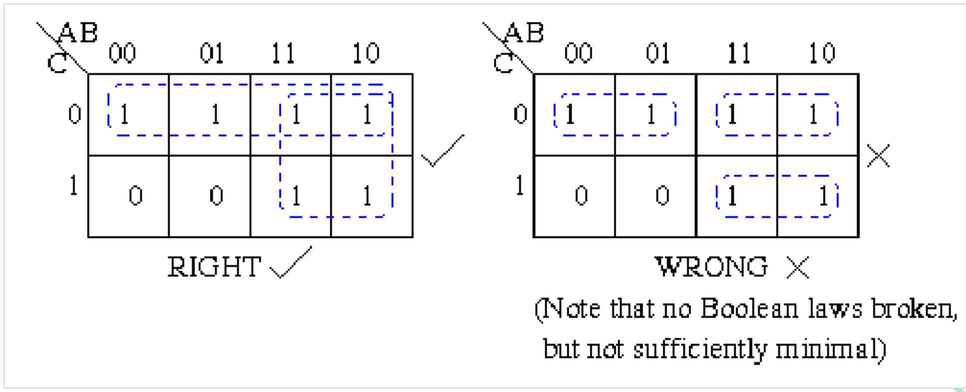


3. Groups must contain 1, 2, 4, 8, or in general 2^n cells. That is if n = 1, a group will contain two 1's since $2^1 = 2$.

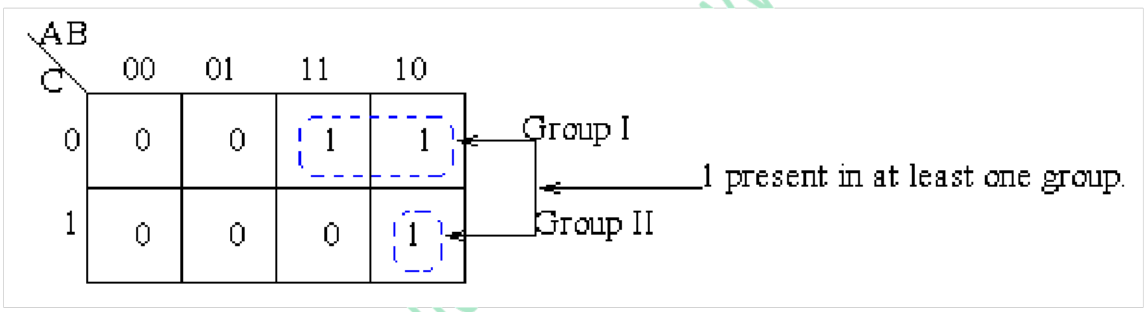
   If n = 2, a group will contain four 1's since $2^2 = 4$.



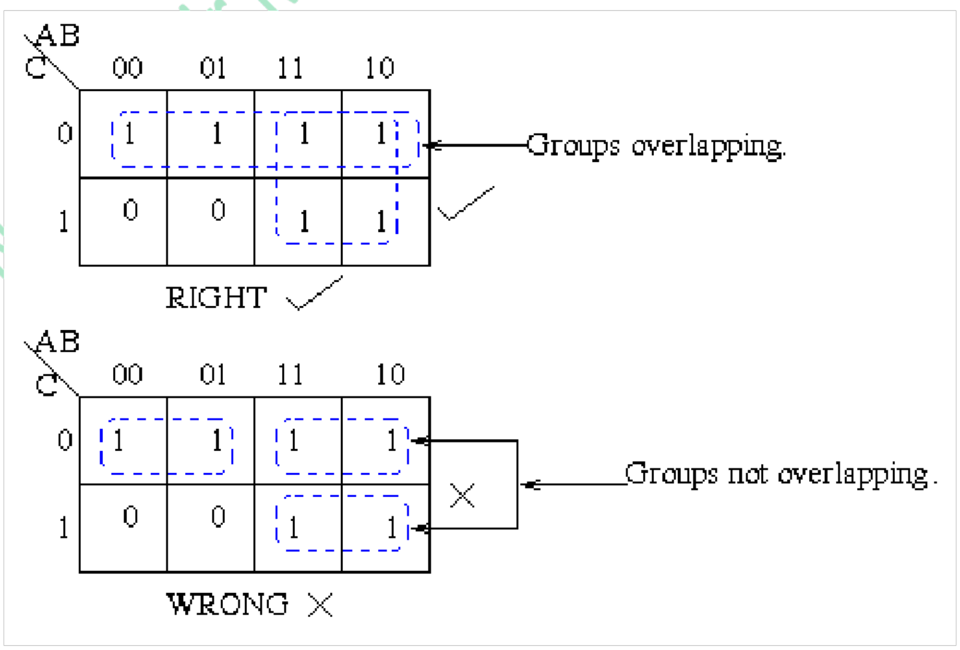
4. Each group should be as large as possible.



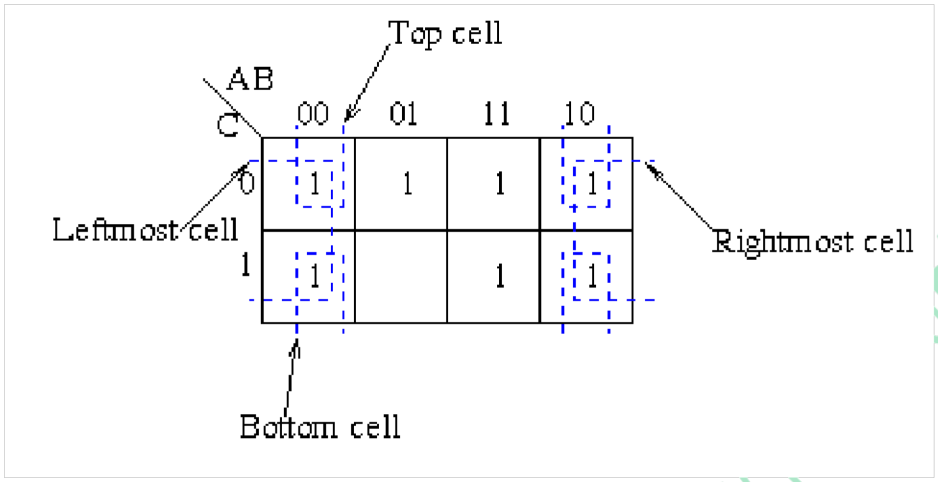
5. Each cell containing a *one* must be in at least one group.



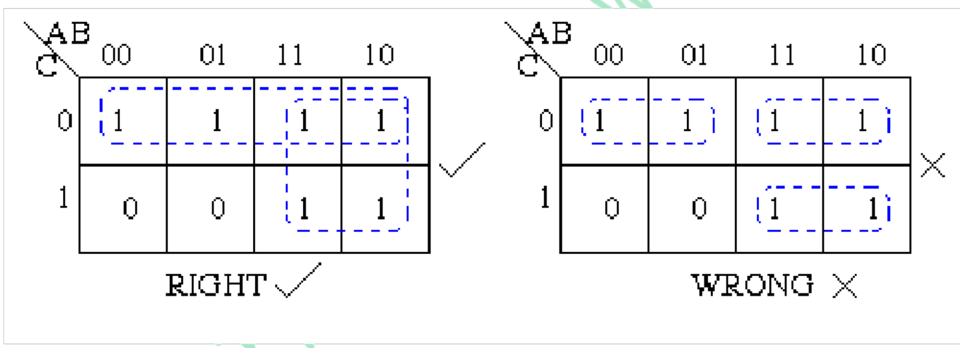
6. Groups may overlap.



7. **Groups may wrap around the table. The leftmost cell in a row may be grouped with the rightmost cell and the top cell in a column may be grouped with the bottom cell.**



8. **There should be as few groups as possible, as long as this does not contradict any of the previous rules.**



Summmary:

1. No zeros allowed.
2. No diagonals.
3. Only power of 2 number of cells in each group.
4. Groups should be as large as possible.
5. Every one must be in at least one group.
6. Overlapping allowed.
7. Wrap around allowed.
8. Fewest number of groups possible

A **Karnaugh map** is a method of creating a Boolean algebra expression from a truth table. It can make the process much easier than if you use sum-of-products to create minterms. The truth table for an OR gate, shown as Table 18.06, can be used to illustrate the method. Using the **sum-of-product** approach gives the following expression of **X.**

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Table 18.06 The truth table for the OR operand

$$X = \bar{A} . B + A . \bar{B} + A.B$$

This is not instantly recognisable as **A+B** but, with a little effort, using Boolean algebra laws it could be shown to be the same.

The Karnaugh map approach is simpler. The corresponding K-map is shown in Figure **18.08**. Each cell in a Karnaugh map shows the value of the output X for a combination of input values for A and B

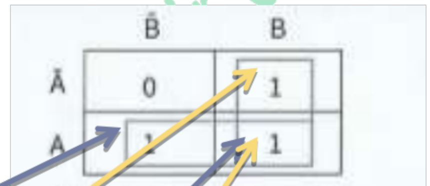


Figure 18.08 A K-map of the truth table in Table 18.06

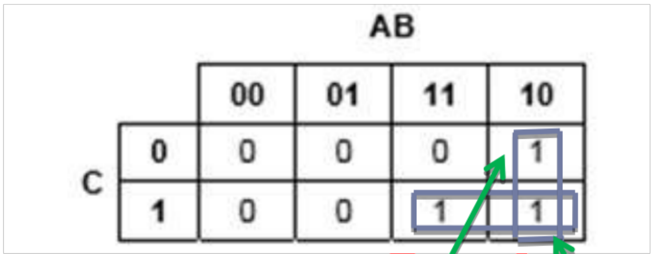
Now how this **A + B** came

- We draw the groups according to rules
- We analyse first group and see what variable doesn't change
- In first group second row **A=1, B=0** and second and then **A=1 and B=1**. The value A=1 remains same so we will take **A**
- In second group in second column A=0, B=1 and then **A=1 and B=1**, so B remains same and we take **B**
- Now because we have two groups one containing **A** and other containing **B** we we combine them and write **A+B**

Now consider we are given a truth table to solve question and simplify the logic circuit: See question below

A	B	C	Working Space	X
0	0	0		0
0	0	1		0
0	1	0		0
0	1	1		0
1	0	0		1
1	0	1		1
1	1	0		0
1	1	1		1

Our Karnugh-map becomes



Now the Boolean expression

- **A=1B=1C=1** and **A=1B=0C=1 (A** and **C** remain **1** so we write **A.C)**
- **A=1 B=0 C=0** and **A=1 B=0 C=1 (A** remains **1** and **B** remain **0)** so we write **$A.\overline{B}$**
- **Now expression combines $A.C + A.\overline{B}$**
- **Answer becomes $X = A.C + A.\overline{B}$**

For further understanding, see the video on site. Video link given below:

https://drive.google.com/file/d/1M5C7rEzLOk9Kyg3OCoRnuna3XO-6m77L/view

Sum of Products: Boolean expression from a truth table

Consider the following Truth table:

INPUT			OUTPUT
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

We consider the **Output** column only and **1's** only in it to make our Boolean expression:

$$\overline{A}.B.C + A.\overline{B}.C + A.B.C$$

(9608/33/M/J/18)

4 (a) A Boolean expression produces the following truth table.

INPUT			OUTPUT
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

(i) Write the Boolean expression for the truth table as a sum-of-products.

X = ...[2]

(ii) Complete the Karnaugh Map (K-map) for the truth table in part (a)(i).

AB

		00	01	11	10
C	0				
	1				

[1]

The K-map can be used to simplify the function in part (a)(i).

(iii) Draw loop(s) around appropriate group(s) of 1s to produce an optimal sum-of-products for the table in part (a)(ii). [2]

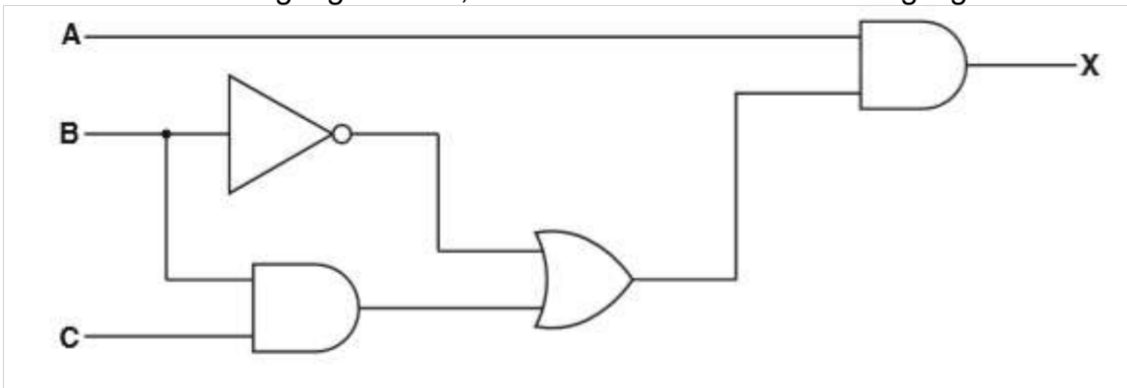
(iv) Write the simplified sum-of-products expression for your answer to part (a)(iii).
X = ... [2]

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>4(a)(i)</td><td>2 marks all products correct, 1 mark 2 or 3 products correct
$X = \bar{A}.B.\bar{C} + \bar{A}.B.C + A.\bar{B}.\bar{C} + A.\bar{B}.C$</td><td>2</td></tr>
<tr><td>4(a)(ii)</td><td>1 mark for all correct bits
AB
<table><tr><td></td><td></td><td>00</td><td>01</td><td>11</td><td>10</td></tr><tr><td>C</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td></td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr></table></td><td>1</td></tr>
<tr><td>4(a)(iii)</td><td>1 mark for each correct loop
AB
<table><tr><td></td><td></td><td>00</td><td>01</td><td>11</td><td>10</td></tr><tr><td>C</td><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr><tr><td></td><td>1</td><td>0</td><td>1</td><td>0</td><td>1</td></tr></table></td><td>2</td></tr>
<tr><td>4(a)(iv)</td><td>1 mark per bullet – allow follow through from 4(a)(iii)
□ $\bar{A}.B$
□ $+A.\bar{B}$
$X = \bar{A}B + A\bar{B}$</td><td>2</td></tr>
</table>

(9608/31/M/J/17)

Further explanation via exam questions below

3. Consider the following logic circuit, which contains a redundant logic gate.



(a) Write the Boolean algebraic expression corresponding to this logic circuit:

....[3]

(b) Complete the truth table for this logic circuit.

A	B	C	Working space	X
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

[2]

(c) (i) Complete the Karnaugh Map (K-map) for the truth table in part (b).

AB

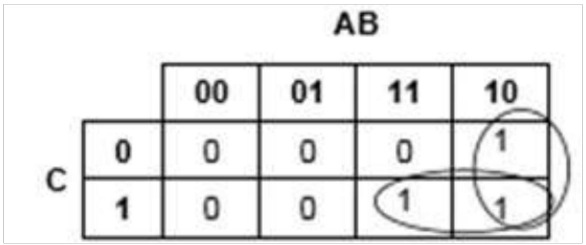
C		00	01	11	10
	0				
	1				

[1]

The K-map can be used to simplify the expression in **part (a)**.

(ii) Draw loop(s) around appropriate groups to produce an optimal sum-of-products. [2]

Answer



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

(iii) Write a simplified sum-of-products expression, using your answer to part (ii).

X = A.C ....[2]

(d) One Boolean identity is:

$$A + \overline{A} . B = A + B$$

Simplify the expression for X in part (a) to the expression for X in part (c)(iii). You should use the given identity.

....

....

.... [2]

Answer:

<table>
<tr><th>Question</th><th>Answer</th><th>Marks</th></tr>
<tr><td>3(a)</td><td>$X = A.(\overline{B} + (B . C))$
B.C 1
$\overline{B}$ + B.C 1
A. 1</td><td>3</td></tr>
<tr><td>3(b)</td><td>
<table>
<tr><th>A</th><th>B</th><th>C</th><th>Working Space</th><th>X</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td></td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td></td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td></td><td>0</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td></td><td>0</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td></td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td></td><td>1</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td></td><td>0</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td></td><td>1</td></tr>
</table>
1 mark first four entries, 1 mark for the last four entries</td><td>2</td></tr>
<tr><td>3(c)(i)</td><td>
<table>
<tr><th></th><th></th><th colspan="4">AB</th></tr>
<tr><th></th><th></th><th>00</th><th>01</th><th>11</th><th>10</th></tr>
<tr><th rowspan="2">C</th><th>0</th><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><th>1</th><td>0</td><td>0</td><td>1</td><td>1</td></tr>
</table></td><td>1</td></tr>
<tr><td>3(c)(ii)</td><td>
<table>
<tr><th></th><th></th><th colspan="4">AB</th></tr>
<tr><th></th><th></th><th>00</th><th>01</th><th>11</th><th>10</th></tr>
<tr><th rowspan="2">C</th><th>0</th><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><th>1</th><td>0</td><td>0</td><td>1</td><td>1</td></tr>
</table></td><td>2</td></tr>
<tr><td>3(c)(iii)</td><td>$X = A.\overline{B} + A.C$
1 1</td><td>2</td></tr>
<tr><td>3(d)</td><td>$X = A.(\overline{B} + (B . C))$
$X = A.(\overline{B} + C)$ 1
$X = A.\overline{B} + A.C$ 1 (dependent mark – must be correct outcome from previous line)</td><td>2</td></tr>
</table>

References:

- http://www.ee.surrey.ac.uk/Projects/Labview/minimisation/karrules.html
- Computer Science Course Book by Sylvia Langfield & Dave Duddell
- Teacher Support Guide (CIE Resource)
- Computer Science Revision Guide by Tony Piper.