Division of Computing Science and Mathematics, University of Stirling

CSCU9V4 Systems

Tutorial 1c: Karnaugh Map

Learning Objectives:

At the end of this tutorial you will be able to;

☑ Draw a Karnaugh map for a logic system with up to three inputs and use it to minimise the number of gates required;

Karnaugh maps

In this tutorial we are going to show you another method of simplification, which you will hopefully find much quicker and less prone to error, as in many cases the simplest Boolean expression is obtained immediately, without the need for any further manipulation using Boolean algebra.

This method of simplification of requires the production of a diagram to represent the contents of a truth table, (or a map). Let us look at a very simple example for a two input logic gate. From our previous work you will realise that a two input gate can have four different possible settings. The truth table looks like.

Inputs		Output
B	**A**	**Q**
0	0	0
0	1	1
1	0	1
1	1	1

What type of logic gate is this ? ....

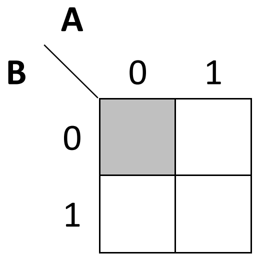
An alternative way of presenting this data is as follows

B \ **A**	0	1
0	0	1
1	1	1

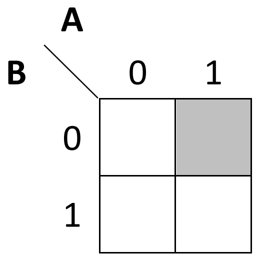
Can you see how the logic state of **Q** has been transferred to the diagram?

This diagram is in fact a Karnaugh map, each square or cell in the Karnaugh map corresponds to a cell in truth table, as shown below.

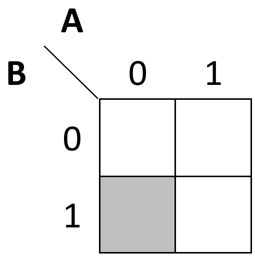
Inputs		**Output**
B	**A**	**Q**
0	0	
0	1	
1	0	
1	1	



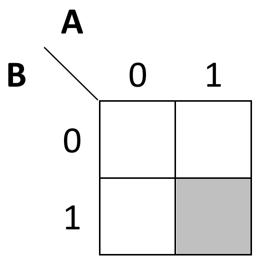
Inputs		**Output**
B	**A**	**Q**
0	0	
0	1	
1	0	
1	1	



Inputs		**Output**
B	**A**	**Q**
0	0	
0	1	
1	0	
1	1	



Inputs		**Output**
B	**A**	**Q**
0	0	
0	1	
1	0	
1	1	



Now you should be able to complete a Karnaugh map for any two input logic function, so let’s put that to the test.

Exercise 1: Complete the following Truth Tables and Karnaugh maps for the logic gates named.

1. AND gate:

Inputs		Output
B	**A**	**Q**
0	0	
0	1	
1	0	
1	1	

B \ A	0	1
0		
1		

2. NOR gate:

Inputs		Output
B	**A**	**Q**
0	0	
0	1	
1	0	
1	1	

B \ A	0	1
0		
1		

3. XOR gate

Inputs		Output
B	**A**	**Q**
0	0	
0	1	
1	0	
1	1	

B \ A	0	1
0		
1		

4. NAND gate

Inputs		Output
B	**A**	**Q**
0	0	
0	1	
1	0	
1	1	

B \ A	0	1
0		
1		

5. XNOR gate

Inputs		Output
B	**A**	**Q**
0	0	
0	1	
1	0	
1	1	

B \ A	0	1
0		
1		

An alternative way of looking at the cells in the Karnaugh map is shown below:

Inputs		Output
B	**A**	**Q**
0	0	$\overline{A}.\overline{B}$
0	1	$A.\overline{B}$
1	0	$\overline{A}.B$
1	1	$A.B$

B \ A	0	1
0	$\overline{A}.\overline{B}$	$A.\overline{B}$
1	$\overline{A}.B$	$A.B$

This should give you a clue as to how we can read a Karnaugh map, since each cell has it’s own Boolean expression!

So we can now fill in a Karnaugh map, but how do we read them to obtain a Boolean expression? Complete the Karnaugh map for the following truth table.

Inputs		Output
B	**A**	**Q**
0	0	0
0	1	1
1	0	0
1	1	0

B \ A	0	1
0		
1		

In this case, the truth table and the Karnaugh map do not match one of our standard logic gates, but we can extract the Boolean equation straight from the cell in the Karnaugh map containing a logic 1, i.e. $\mathbf{A.\overline{B}}$

So what, you might say, "there doesn't seem to be a lot of point in what we are doing here", because we could have obtained this directly from the truth table, without having to draw the Karnaugh map. Let us have a look at another example.

Inputs		Output
B	**A**	**Q**
0	0	1
0	1	1
1	0	0
1	1	1

B \ A	0	1
0	1	1
1	0	1

At first glance this might look like one of our standard logic gates, however it isn't. Now if we write down the Boolean expression for each logic 1 in the map we get the following:

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

This is not exactly the simplest of Boolean equations and we could have obtained the same result from the truth table – there still doesn't appear to be much of an advantage to the Karnaugh map.

Let us look at that expression and simplify it using Boolean Algebra to obtain the simplest solution.

$$\overline{A}.\overline{B} + A.\overline{B} + A.B$$
$$\overline{A}.\overline{B} + A.(\overline{B} + B)$$
$$\overline{A}.\overline{B} + A$$
$$\overline{B} + A$$

Now look at the Karnaugh map again

B \ A	0	1
0	1	1
1	0	1

If we think of the right hand side of the Karnaugh map and compare it to the map we introduced earlier showing the Boolean terms for each cell we obtain the following:

B \ A	0	1
0	1	1
1	0	1

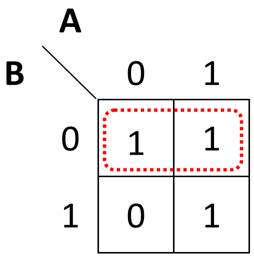
B \ A	0	1
0	$\overline{A}.\overline{B}$	$A.\overline{B}$
1	$\overline{A}.B$	$A.B$

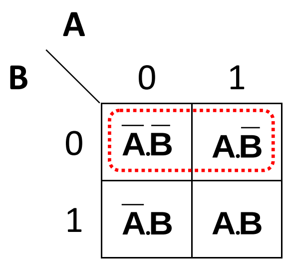
What this shows is that the Boolean terms we have grouped together are:

$$A.\overline{B} + A.B$$
$$A.(\overline{B} + B)$$
$$A.1$$
$$A$$

Using the Karnaugh map is a graphical method of finding the common factors in a Boolean expression. If we look at the Karnaugh map we can see that the group of 1's we have identified corresponds to when input **A** is a logic 1, which corresponds to the simplest term obtained via the Boolean simplification.

Now consider the following alternative grouping that could have been created.





Again this shows that the Boolean terms we have grouped together are:

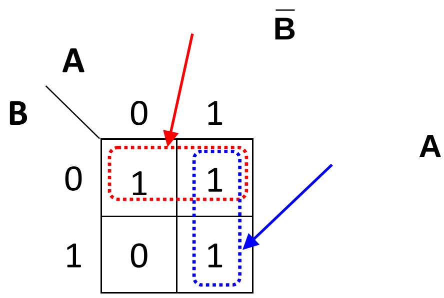
$$\overline{A}.\overline{B} + A.\overline{B}$$
$$\overline{B}.(\overline{A} + A)$$
$$\overline{B}.1$$
$$\overline{B}$$

If we look at the Karnaugh map we can see that the group of 1's we have identified corresponds to when input **B** is a logic 0, or $\overline{\mathbf{B}}$, which corresponds to the simplest term obtained via the Boolean simplification.

If we put the two results together we obtain the simplest solution as

$$A + \overline{B}$$

In practice we would perform these two simplifications in one step as shown below:



The advantage of a Karnaugh map is that we can quickly spot the common terms in a complicated expression. This advantage will be more obvious when we deal with 4 input expressions.

The only thing we have to be sure of is that every logic 1 in the map is included in at least one group.

Example 2: Try the following example:

i. Complete the Karnaugh map.
ii. Group the logic 1's together
iii. Write down the Boolean term for each group linked by the OR function '+'.
iv. Check your answer by simplifying the expression obtained from the truth table using the rules of Boolean Algebra.

Inputs		**Output**
B	**A**	**Q**
0	0	1
0	1	0
1	0	1
1	1	1

B \ **A**	0	1
0		
1		

Simplest Expression from map = ...

Boolean simplification from Truth Table.

....

....

....

....

Hopefully you are beginning to see the advantages of using a Karnaugh map, but for two input logic functions there is not a great deal of advantage to be gained. The true power of Karnaugh maps becomes much clearer when we look at three and four input logic systems, which is where we will go next.

Three Input Logic Systems

A three input logic system will have eight possible input combinations as shown below.

Inputs			Output
C	**B**	**A**	**Q**
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

We need a bigger Karnaugh map to accommodate the extra input details, and this is shown below:

C \ **BA**	00	01	11	10
0				
1				

Things to note:

i. the inputs **A** and **B** have been linked together on the top row of the Karnaugh map.

ii. the sequence of combinations for **A.B**, is not quite as you might expect. They do not increase numerically as they move across the table. This is deliberate, as it ensure that from one cell to the next only one variable changes. **This is essential if the Karnaugh map is to be read correctly.**

The completion of the Karnaugh map is carried out in a similar way to that of the two input map. In the 2 diagrams below, letters have been used to help you identify how the various rows in the truth table correspond to the cells on a Karnaugh map:

Inputs			Output
C	**B**	**A**	**Q**
0	0	0	**a**
0	0	1	**b**
0	1	0	**c**
0	1	1	**d**
1	0	0	**e**
1	0	1	**f**
1	1	0	**g**
1	1	1	**h**

C \ **BA**	00	01	11	10
0	**a**	**b**	**d**	**c**
1	**e**	**f**	**h**	**g**

Example :

Inputs			Output
C	**B**	**A**	**Q**
0	0	0	**1**
0	0	1	**1**
0	1	0	**1**
0	1	1	**0**
1	0	0	**0**
1	0	1	**0**
1	1	0	**1**
1	1	1	**0**

C \ **BA**	00	01	11	10
0	**1**	**1**	**0**	**1**
1	**0**	**0**	**0**	**1**

We use the same rules as before to group the Logic 1's together as a group as shown below.

C \ **BA**	00	01	11	10
0	**1**	**1**	**0**	**1**
1	**0**	**0**	**0**	**1**

Now write down the Boolean expression for each group. The rule is that we ignore any variable that changes between groups.

$B.\overline{A}$ {Note: **C** changes but not $B.\overline{A}$ }

$\overline{C}.\overline{B}$ {Note: **A** changes but not $\overline{C}.\overline{B}$ }

Now compare this to the Boolean Simplification.

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

So the Boolean Expression is $\overline{B}.\overline{C} + \overline{A}.B$.

Now consider the following example.

Inputs			Output
C	**B**	**A**	**Q**
0	0	0	**1**
0	0	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	0	**0**
1	0	1	**0**
1	1	0	**1**
1	1	1	**1**

C \ **BA**	00	01	11	10
0	**1**	**1**	**1**	**1**
1	**0**	**0**	**1**	**1**

In this case there are a large number of logic 1's in the Karnaugh map, we could group these into three groups of 2 as we have done previously.

C \ **BA**	00	01	11	10
0	**1**	**1**	**1**	**1**
1	**0**	**0**	**1**	**1**

Using our normal rules we obtain the following Boolean expression.

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

This does not look to be a very simple solution, and indeed we can use Boolean Algebra to simplify this further as follows:

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

If we reconsider the simpler solution we have just obtained with the original Karnaugh map we can identify why we did not get the simple answer straight away.

The simplest Boolean expression is $\overline{C}+B$. The Karnaugh map is:

$\overline{C}$ {Note: **A** and **B** change but not $\overline{C}$}

C \ BA	00	01	11	10
0	**1**	**1**	**1**	**1**
1	**0**	**0**	**1**	**1**

B {Note: **A** and **C** change but not **B**}

The Karnaugh map shows that the two terms in the simplest Boolean expression refer to groups of 4 logic 1's in the map. This gives us another rule for working with Karnaugh maps – **Group the logic 1's into the largest group possible, i.e. groups of 2 or 4 adjacent cells.**

Exercise 3: Derive the simplest Boolean expression for the logic function defined by the truth table.

Inputs			Output
C	**B**	**A**	**Q**
0	0	0	**0**
0	0	1	**1**
0	1	0	**0**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**0**
1	1	1	**1**

C \ BA	00	01	11	10
0				
1				

Simplest Boolean expression = ...

Exercise 4: Simplify the following Boolean Expressions using Karnaugh Map

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

C \ BA	00	01	11	10
0				
1				

Simplest Boolean expression = ...

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

C \ BA	00	01	11	10
0				
1				

Simplest Boolean expression = ...

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

C \ BA	00	01	11	10
0				
1				

Simplest Boolean expression = ...