The Karnaugh Map Introduction (A)

Quite often, it is difficult to simplify a Boolean expression to its minimal expression due to its complexity. In addition to this difficulty, remember that the possibility of making a mistake goes up significantly with the increase in the complexity of the expression. Finally, it is quite often impossible to know for sure:

- **if the result of your Boolean algebra attempt is indeed the simplest answer, or if it is the simplest,**
- **are there other answers which are just as simple which might be of more use to the logic designer**.

At first, it might seem that the **Karnaugh Map** is just another way of presenting the information in a truth table. In one way that's true. However, any time you have the opportunity to use another way of looking at a problem, advantages can accrue to you. In the case of the **Karnaugh Map (K-Map)** the advantage is that it is designed to present the information in a way that allows easy grouping of terms that can be combined legally.

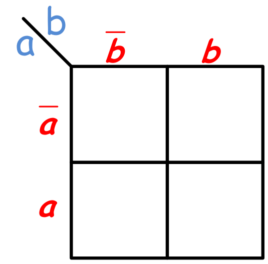
The **K-map technique converts each row in a truth table representing some Boolean expression into a single cell in a graphic map.** Each cell in this **K-map** is actually a **min-term** in a row of a truth table. Through this technique, we will be able to come to a minimal result faster and other minimal solutions will be more obvious. In addition, the designer can use this technique to expand algebraic expressions into their algebraic and/or numerical canonical forms easily with less chance of error.

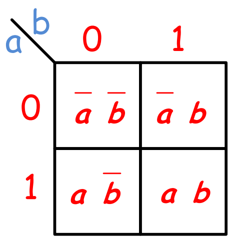
The 2-variable Karnaugh Map

In order to visualize the one-to-one correlation between the truth table and a K-map, let's observe the table to the right. Note that each row is in binary numerical order and can be associated with min-term numbers. Each table represents the output of a Boolean expression. We can take this table form of representing a Boolean expression and look at it graphically instead.

a	b	min– *terms*	
0	0	$\bar{a}\,\bar{b}$	m_0
0	1	$\bar{a}b$	m_1
1	0	$a\bar{b}$	m_2
1	1	ab	m_3

The designer begins by creating a graph with one cell for each row of the table. The rows are assigned to the **MSB** (in this case, a) and the columns are assigned to the **LSB, b.** Then all possibilities of the variables are listed along the top and the side as shown in the **K-map** to the right **(we will do this differently (and better) in a minute, but stick with me.)**

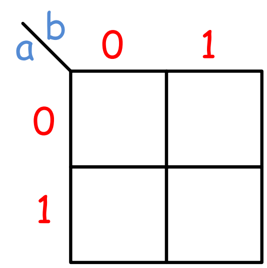




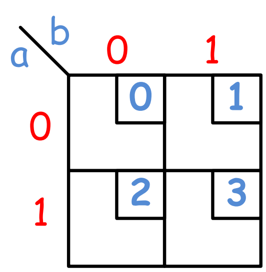
It's normally easier once you understand the meaning of the columns and the rows, to <u>replace the algebraic headings with their corresponding logic levels</u> as is demonstrated in the **K-map** to the left. <u>**This is the method you will be REQUIRED to use in this course.**</u>

a	*b*	min– *terms*	
0	0	$\bar{a}\,\bar{b}$	m_0
0	1	$\bar{a}b$	m_1
1	0	$a\bar{b}$	m_2
1	1	ab	m_3

As can be observed, each **min-term** in the table to the left can now be easily mapped into the associated **K-map** to the right.



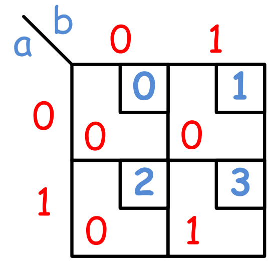
A very handy way to remember the min-term value of each cell is to record that value in the corner of each cell. This will be especially important when you have a **numerical canonical list** which you wish to map into the graph or you wish to expand an expression into its **numerical canonical form**. Again, this is the method that you will be **REQUIRED** to use in any course I teach.



The AND K-map

a	b	f(a,b)
0	0	0
0	1	0
1	0	0
1	1	1

We can now begin to map the truth table results into a **K-map**. You should recognize the truth table on the left as representing the AND **operation**. Each row can be mapped into the **K-map** to the right as shown. Care must be taken to map it in correctly.

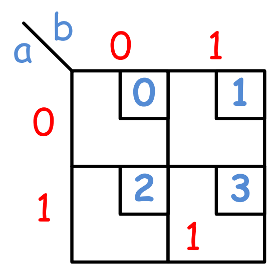


The **K-map** above demonstrated how to map a truth table output function column. Note that both the 1's and the 0's were mapped. **However, the normal method is to decide if you desire the result to be in SOP or in POS form.**

If an:

- SOP result is desired
  - **Map only** the 1's
- POS result is desired
  - **Map only** the 0's

The **K-map** is now redrawn with only the 1's from the table indicating that an **SOP result is desired**.

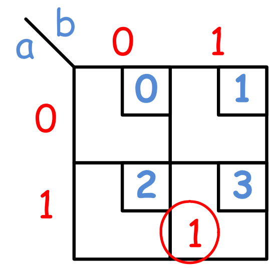


Grouping the min-terms

Remember that one of the biggest reasons for the **k-map** method was to be able to simplify the algebraic expression more easily. To accomplish that task we can now introduce the first "**grouping rule**."

Grouping Rule #1

Group the terms into groups of powers of 2, i.e., group sizes of 1, 2, 4, 8, 16,… In order for a term to be included in a group, it must "share an edge" with another term.



The K-map to the right has a single **min-term** in it so it is grouped into a "group of 1" (**1 is a power of 2**.) using the **Grouping Rule #1**. The result can now be expressed in algebraic terms by recording the column and row headings for that group.

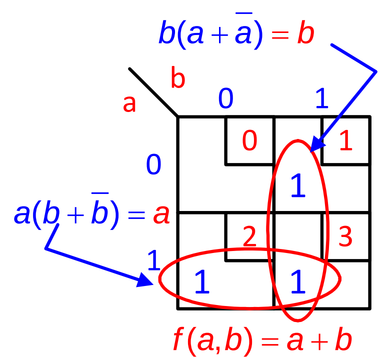
In this case, the **MSB (row)** heading is a **'1'** which is an **'a'** and the **LSB (column)** heading is a '**1**' which is a **"b"**. Therefore, the resulting term is:

$$f(a,b) = ab$$

The OR K-map

a	b	$f(a,b)$
0	0	0
0	1	1
1	0	1
1	1	1

Now let's look at the **OR operation** truth table and resulting **K-map**. If we apply "**grouping rules #1**" we can perform the simplification operation to determine the **minimal algebraic expression**.

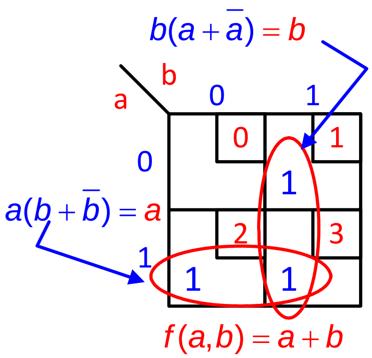


The key here is to note that if a row or column variable changes within a group, then that variable falls out of the answer!

The **K-map** above is a one-to-one representation of the **OR truth table**. As can be seen, the **truth table** has **1**'s in **rows 1, 2,** and **3**, and the **K-map** has 1's in **cells 1, 2,** and **3.** Note that now there are two groups with two **1**'s each. **Cell 1** shares an edge with **cell 3,** therefore they can be grouped together, while **cells 2** and **3** share an edge and can be grouped together as well.

Also note that a group of 2 IS a POWER OR TWO!!

Let's go a little deeper into the analysis of this **K-map** just in case the reader is still confused: Let's look at the **cell grouping (1,3).**



Vertically, it stays in the **"1" column**, so it can be represented by a '**(b)**', while horizontally it is in both the **0 and the 1 rows** for '**a**' and can be represented by a

$$\left(a+\overline{a}\right)$$

Remember from the **Boolean algebra** rules that: $\left(a+\overline{a}\right)=1$

For this reason, the expression becomes $b\left(a+\overline{a}\right)=b\left(1\right)=b$.

The same thing can be done with the **cell (2,3) grouping**. The group occupies only the **'1'(a)** row but occupies both the **'0'**$\left(\overline{b}\right)$ and the **'1'**$\left(b\right)$ columns. Therefore, we now have:

$$a(b+\overline{b})=a(1)=a$$

Now, since we mapped the '**1**'s, we desire an **SOP** expression, so we connect the two expressions by an **OR** and we get:

$$f\left(a,b\right)=a+b$$

Before we go any further, there's something that you should observe from what just happened. As hinted at earlier:

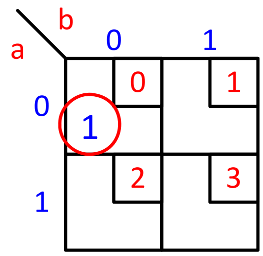
"**Anytime a row or column variable changed within a group, it fell out of the final expression.**"

Based on this, we could easily have looked at the **K-map** and seen that the '**a**' **dropped out** of the **(1,3) grouping** and that '**b**' **dropped out** of the **(2,3) grouping** without having to worry about the $x+\overline{x}=1$ rule. Using this quality of the **K-map**, we can now find the **K-map** representations for all the other basic **Boolean** operations by OBSERVATION!

The rest of the basic Boolean operation K-maps

The NOR operation:

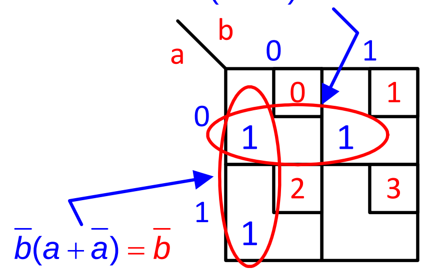
a	b	$f(a,b)$
0	0	1
0	1	0
1	0	0
1	1	0



$$f(a,b) = \overline{a}\,\overline{b}$$
$$= \overline{a+b}$$

The NAND operation:

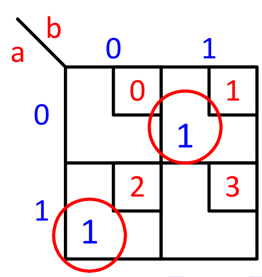
a	b	$f(a,b)$
0	0	1
0	1	1
1	0	1
1	1	0



$$f(a,b) = \overline{a} + \overline{b}$$
$$= \overline{ab}$$

The EXCLUSIVE OR (XOR) operation

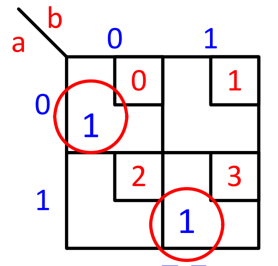
a	b	$f(a,b)$
0	0	0
0	1	1
1	0	1
1	1	0



$$f(a,b) = \overline{a}b + a\overline{b}$$
$$= a \oplus b$$

The EXCLUSIVE NOR (XNOR) operation

a	b	$f(a,b)$
0	0	1
0	1	0
1	0	0
1	1	1

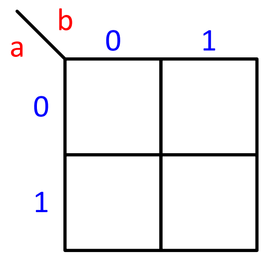


$$f(a,b) = \bar{a}\,\bar{b} + a\,b$$
$$= \overline{a \oplus b}$$

For the sake of completeness, there are two other **K-maps** which should be shown:

The '0' K-map.

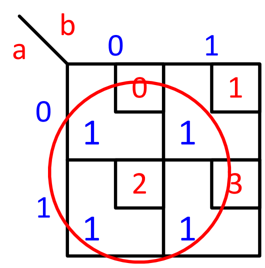
a	b	$f(a,b)$
0	0	0
0	1	0
1	0	0
1	1	0



$$f(a,b) = 0$$

The '1' K-map

a	b	$f(a,b)$
0	0	1
0	1	1
1	0	1
1	1	1



$$f(a,b) = 1$$

Introduction Summary

To summarize what we have learned:

- Fill in the **K-map** with either the 1's or the **0**'s depending on if you want an **SOP** expression or a **POS** expression (we will cover the 0's later).

- Group the **adjacent entries** into group sizes of "**powers of two**" (obviously the largest group obtainable with a **two-variable K-map** is a **group of 4**). **(WE WILL LEARN MORE GROUPING RULES IN THE NEXT FEW SECTIONS)**

- Drop any variable which changes within a group out of the expression.

- For an **SOP** expression, **OR** the resulting terms together.