

|  |
| --- |
| **Activity 2.2.1 Karnaugh Mapping** |

Introduction

At this point you have the ability to apply the theorems and laws of Boolean algebra to simplify logic expressions in order to produce simpler and more cost effective digital logic circuits. You may have also realized that simplifying a logic expression using Boolean algebra, though not terribly complicated, is not always the most straightforward process. There isn’t always a clear starting point for applying the various theorems and laws, nor is there a definitive end in the process.

Wouldn’t it be nice to have a process for simplifying logic expressions that was more straightforward, had a clearly defined beginning, middle, and end, and didn’t require you to memorize all of the Boolean theorems and laws? Well there is, and it’s called Karnaugh mapping. Karnaugh mapping, or K-Mapping, is a graphical technique for simplifying logic expressions containing up to four variables.

In this activity you will learn how to utilize the Karnaugh mapping technique to simplify two, three, and four variable logic expressions. Additionally, logic expressions containing *don’t care* conditions will be simplified using the K-Mapping process.

Equipment

* Paper and pencil

Procedure

Write the simplified sum-of-products (SOP) logic expression for the K-Maps shown below

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. F1 = A’  |  |  |  | | --- | --- | --- | |  |  |  | |  | 1 | 1 | |  | 0 | 0 | | 1. F2 = AC’+C  |  |  |  | | --- | --- | --- | |  |  |  | |  | 0 | 1 | |  | 0 | 0 | |  | 1 | 1 | |  | 1 | 0 | | 1. F3= A’C’D+A’CD’+AC’  |  |  |  |  |  | | --- | --- | --- | --- | --- | |  |  |  |  |  | |  | 0 | 1 | 0 | 1 | |  | 0 | 1 | 0 | 1 | |  | 1 | 1 | 0 | 0 | |  | 1 | 1 | 0 | 0 | |

After transferring the truth table data into the K-Maps, write the simplified sum-of-products

(SOP) logic expression for the K-Maps shown below.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. F4 =Q  |  |  |  | | --- | --- | --- | | Q | R | F4 | | 0 | 0 | 0 | | 0 | 1 | 0 | | 1 | 0 | 1 | | 1 | 1 | 1 | | |  |  |  | | --- | --- | --- | |  |  |  | |  | 0 | 0 | |  | 1 | 1 | |
| 1. F5 = QR’+QS’+RS’+Q’R’S  |  |  |  |  | | --- | --- | --- | --- | | Q | R | S | F5 | | 0 | 0 | 0 | 0 | | 0 | 0 | 1 | 1 | | 0 | 1 | 0 | 1 | | 0 | 1 | 1 | 0 | | 1 | 0 | 0 | 0 | | 1 | 0 | 1 | 1 | | 1 | 1 | 0 | 1 | | 1 | 1 | 1 | 0 | | |  |  |  | | --- | --- | --- | |  |  |  | |  | 0 | 1 | |  | 1 | 0 | |  | 1 | 0 | |  | 1 | 1 | |
| 1. F6 = RT+QR’S’+Q’RS+QR’ST’  |  |  |  |  |  | | --- | --- | --- | --- | --- | | Q | R | S | T | F6 | | 0 | 0 | 0 | 0 | 0 | | 0 | 0 | 0 | 1 | 0 | | 0 | 0 | 1 | 0 | 0 | | 0 | 0 | 1 | 1 | 0 | | 0 | 1 | 0 | 0 | 0 | | 0 | 1 | 0 | 1 | 1 | | 0 | 1 | 1 | 0 | 1 | | 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 | | |  |  |  |  |  | | --- | --- | --- | --- | --- | |  |  |  |  |  | |  | 0 | 0 | 0 | 0 | |  | 0 | 1 | 1 | 1 | |  | 0 | 1 | 1 | 0 | |  | 1 | 1 | 0 | 1 | |

After labeling the K-Map and transferring the truth table data into it, write the simplified

sum-of-products (SOP) logic expression for the K-Maps shown below.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. F7 =W+W’X  |  |  |  | | --- | --- | --- | | W | X | F7 | | 0 | 0 | 0 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 1 | | |  |  |  | | --- | --- | --- | |  | X’ | X | | W’ | 0 | 1 | | W | 1 | 1 | |
| 1. F8 = Y’+W’X’+WX  |  |  |  |  | | --- | --- | --- | --- | | W | X | Y | F8 | | 0 | 0 | 0 | 1 | | 0 | 0 | 1 | 1 | | 0 | 1 | 0 | 1 | | 0 | 1 | 1 | 0 | | 1 | 0 | 0 | 1 | | 1 | 0 | 1 | 0 | | 1 | 1 | 0 | 1 | | 1 | 1 | 1 | 1 | | |  |  |  | | --- | --- | --- | |  | Y’ | Y | | W’X’ | 1 | 1 | | W’X | 1 | 0 | | WX | 1 | 1 | | WX’ | 1 | 0 | |
| 1. F9 = Y’Z+W’X’Y’+W’XYZ’+WX’YZ’  |  |  |  |  |  | | --- | --- | --- | --- | --- | | W | X | Y | Z | F9 | | 0 | 0 | 0 | 0 | 1 | | 0 | 0 | 0 | 1 | 1 | | 0 | 0 | 1 | 0 | 0 | | 0 | 0 | 1 | 1 | 0 | | 0 | 1 | 0 | 0 | 0 | | 0 | 1 | 0 | 1 | 1 | | 0 | 1 | 1 | 0 | 1 | | 0 | 1 | 1 | 1 | 0 | | 1 | 0 | 0 | 0 | 0 | | 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 | 0 | | |  |  |  |  |  | | --- | --- | --- | --- | --- | |  | Y’Z’ | Y’Z | YZ | YZ’ | | W’X’ | 1 | 1 | 0 | 0 | | W’X | 0 | 1 | 0 | 1 | | WX | 0 | 1 | 0 | 0 | | WX’ | 0 | 1 | 0 | 1 | |

Write the simplified sum-of-products (SOP) logic expression for the K-Maps shown below.

Be sure to take advantage of any *don’t care* conditions.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1. F10 = K’+L’  |  |  |  | | --- | --- | --- | |  |  |  | |  | X(1) | 1 | |  | 1 | 0 | |
| 1. F11 = L  |  |  |  | | --- | --- | --- | |  |  | M | |  | 0 | 0 | |  | 1 | X(1) | |  | 1 | 1 | |  | X(0) | 0 | |
| 1. F12 = M’N+K’L’N+K’LM’+KMN’  |  |  |  |  |  | | --- | --- | --- | --- | --- | |  |  |  |  |  | |  | 0 | 1 | 1 | 0 | |  | 1 | 1 | 0 | X(0) | |  | 0 | X(1) | 0 | X(1) | |  | 0 | 1 | 0 | 1 | |

Conclusion

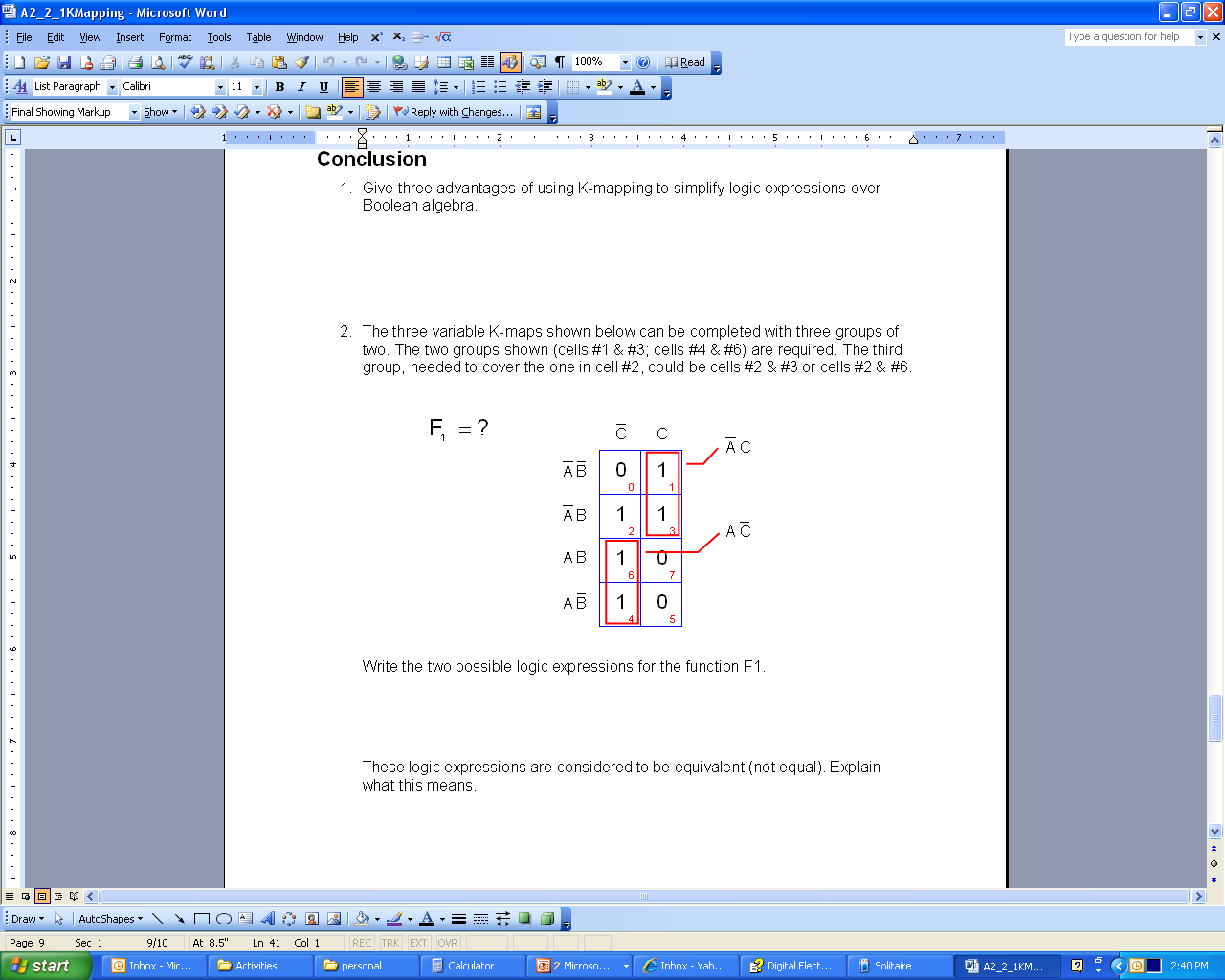
1. Give three advantages of using K-mapping to simplify logic expressions over Boolean algebra.

K-Mapping provides an easier and faster way to simplify the expressions.

They provide a visual aid so you can see what part of the expression is not needed.

You are not required to calculate because the expression is already simplified.

1. The three variable K-maps shown below can be completed with three groups of two. The two groups shown (cells #1 & #3; cells #4 & #6) are required. The third group, needed to cover the one in cell #2, could be cells #2 & #3 or cells #2 & #6.



Write the two possible logic expressions for the function F1.

1 A’C+AC’+A’B

2 A’C+AC’+BC’

1. These logic expressions are considered equal and equivalent but they do not look the same. Explain why these two expressions can be considered equal and equivalent even though they are not identical.

When used they will lead to the same result but the combination of the last gate is different.

**Going Further – Optional**

1. The following four variable K-Maps can be solved using the traditional method of grouping the ones (Identify the 3 groups of 8).

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  |  |  | | --- | --- | --- | --- | --- | |  |  |  |  |  | |  | 1 | 1 | 1 | 1 | |  | 1 | 0 | 0 | 1 | |  | 1 | 1 | 1 | 1 | |  | 1 | 1 | 1 | 1 | |

1. Rather than taking this approach, let’s get creative and take advantage of the fact that the K-Map contains only two zeros. Group these zeros and write the logic expression.

(A’BD)’

Since you grouped the zeroes, this is the logic expression for .

1. Now apply DeMorgan’s Theorem to get the logic expression for .

A+B’+D’

1. What is the advantage of taking this approach over the traditional approach of circling the ones?

It makes it a little more simple using the DeMorgan’s Theorem.

And it is a simple as breaking the line and changing the sign.

1. Are there any disadvantages?

**No**