# Karnaugh Maps

Based on slides by Jared Moore

## Boolean Algebra

Equations that we have previously discussed are not necessarily the most efficient

$$Y = ABC + AB'C + AB + C'$$

*Key:* Not always a straightforward process to simplify.

## Karnaugh Maps

Graphical method for simplifying Boolean equations
(Up to 4 variables)

Essentially just a different way of writing out truth tables that make certain simplifications more obvious

Recall $PA + PA' = P$:  K-maps put terms that can be simplified like this next to one another in a grid

Consider the Boolean expression

$$A'B' + AB'$$

In this case, it’s relatively easy to see that we can use the previous rule to simplify this to $B'$

We are going to see how this same simplification shows up in a K-map

## $A'B' + AB'$

| A | B | Y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

First step to using K-map is to generate truth table

## A′B′ + AB′

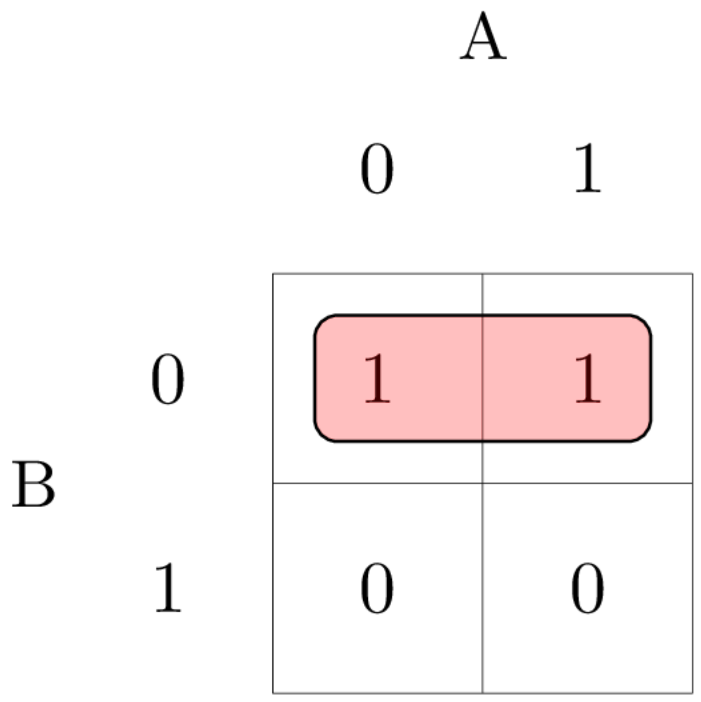
| A | B | Y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

$$\begin{array}{cc|c|c|}
 & & \multicolumn{2}{c}{A} \\
 & & 0 & 1 \\
\cline{3-4}
\multirow{2}{*}{B} & 0 & 1 & 1 \\
\cline{3-4}
 & 1 & 0 & 0 \\
\cline{3-4}
\end{array}$$

K-map is rearrangement of truth table

## A’B’ + AB’

| A | B | Y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |



When two 1s are next to one another in K-map, it is visual indication that simplification of form PA + PA’ = P applies

Consider this example. Clearly, when B is 0, A can be 0 or 1 and the result is 1, so A is not needed

## K-map: general idea vs process

Previous slides tried to get across intuition of *why* K-maps are helpful for simplifying

Even if that is not yet intuitive, we can treat K-maps as a rote, mechanical process for simplifying, which is what we do next

Most of the interesting things with K-maps happen in expressions with 3 or 4 variables

## K-map process

1. Create truth table
2. Convert truth table to K-map
3. Circle 1s according to rules (given later)
4. Convert circles back to simplified Boolean expression

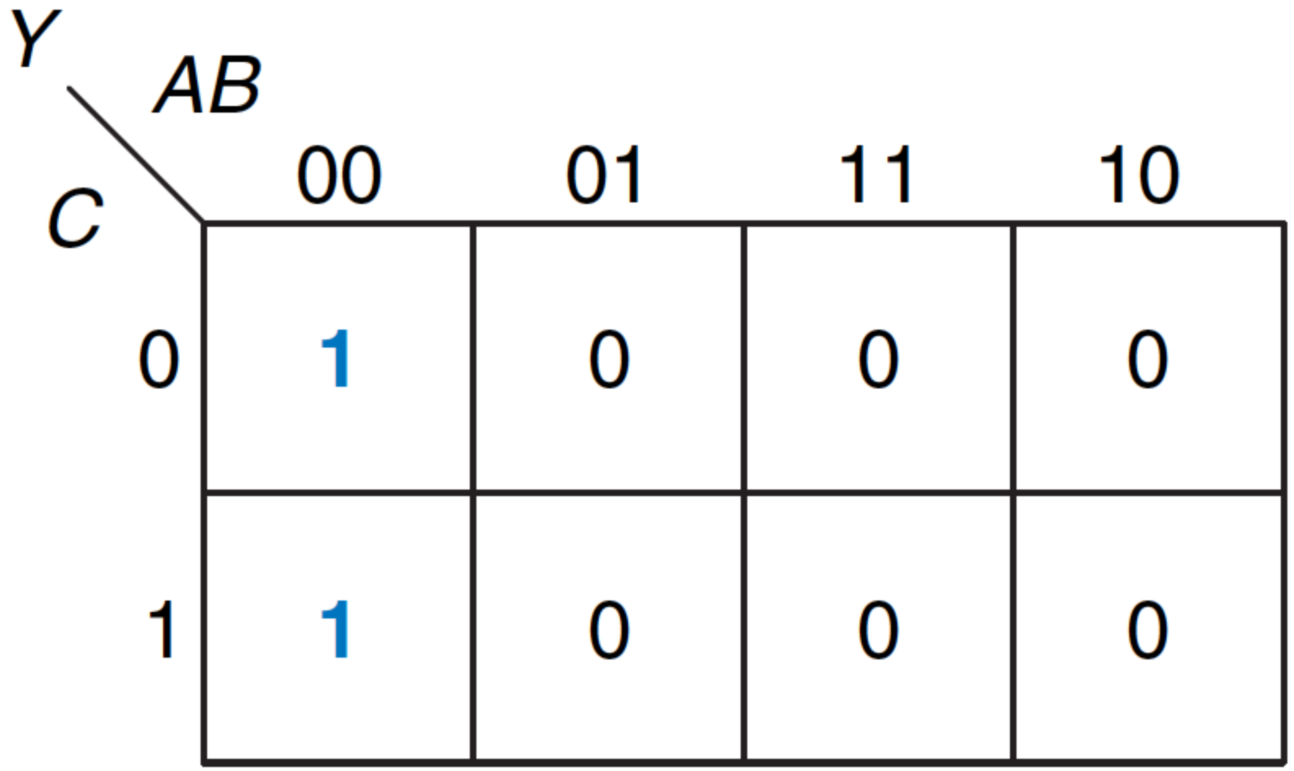
## Step 1

| A | B | C | Y |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

$Y = A'B'C' + A'B'C$

## Step 2

| A | B | C | Y |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |



Y = A′B′C′ + A′B′C

## Creating K-maps (Step 2)

Where you put each variable (along the top or down the side) does not matter

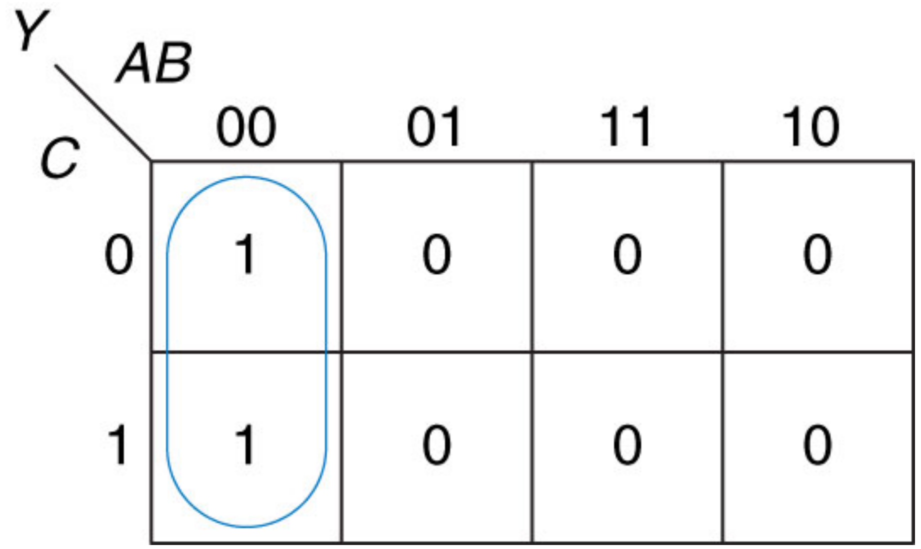
Key is that indices of each adjacent square should differ by exactly one bit flip

00 is one bit flip away from 01

00 is *not* one bit flip away from 11

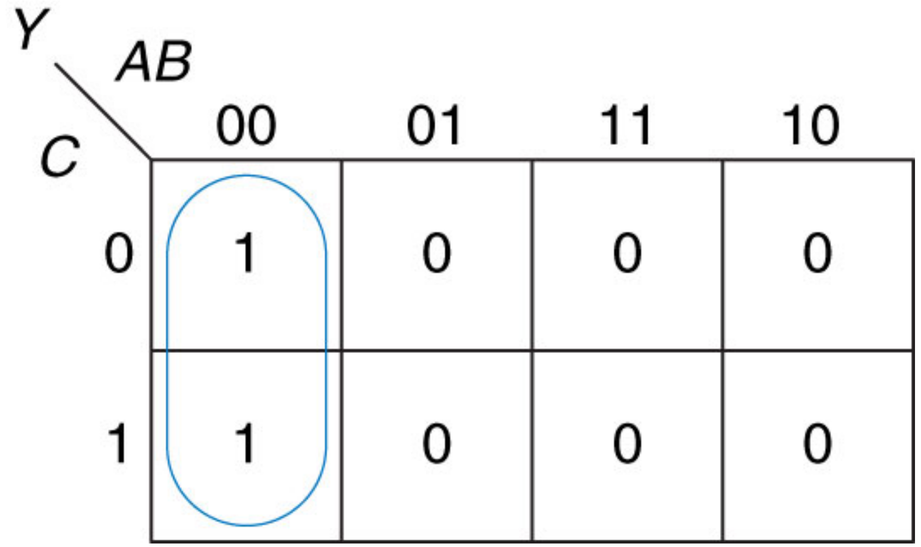
# Step 3 – general idea

1. Circle all the rectangular blocks of 1's in the map with the fewest number of circles.
2. Each circle should be as large as possible



# Step 4

1. Each circle is a term
2. For each term, include only literals that stay the same *everywhere* in the circle
3. All terms are OR'd together to get overall expression



A'B'

## K-map Circling Rules (Rules for Step 3)

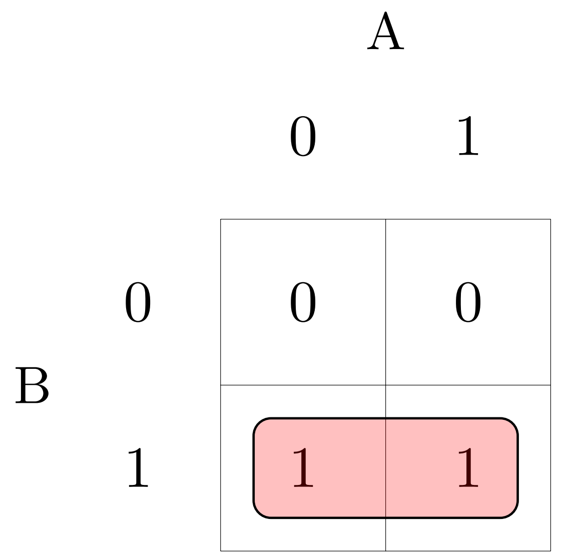
1. Use the fewest circles necessary to cover all the 1's.
2. All the squares in each circle must contain 1's.
3. Each circle must span a rectangular block that is a power of 2 (e.g. 1, 2, or 4) in each direction.
4. Each circle should be as large as possible.
5. A circle may wrap around the edges of the K-map.
6. A 1 in a K-map may be circled multiple times if doing so allows fewer circles to be used.

The only way to get comfortable with K-maps is to practice

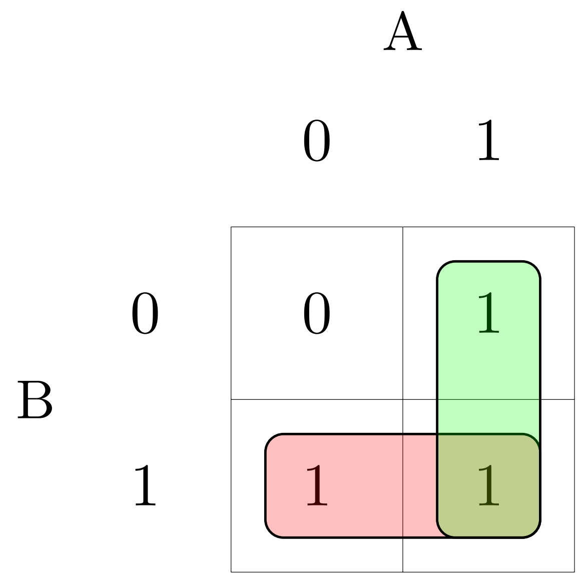
As you go through the following examples, it may be helpful to keep the slide of circling rules handy in a separate window so you have the rules for reference

|  |  | A |  |
|---|---|---|---|
|  |  | 0 | 1 |
| B | 0 | 0 | 0 |
|  | 1 | 1 | 1 |

|  |  | A |  |
|---|---|---|---|
|  |  | 0 | 1 |
| B | 0 | 0 | 1 |
|  | 1 | 1 | 1 |



$Y = B$



$Y = A + B$

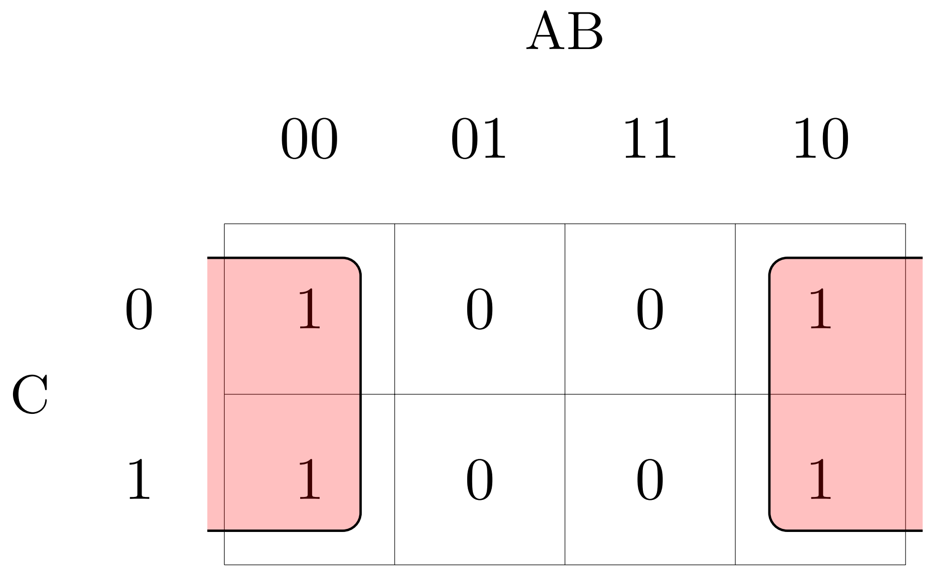
| | | AB | | | |
|---|---|---|---|---|---|
| | | 00 | 01 | 11 | 10 |
| C | 0 | 0 | 0 | 0 | 1 |
| | 1 | 1 | 0 | 0 | 0 |

$$\begin{array}{cc|c|c|c|c|}
 & & \multicolumn{4}{c}{AB} \\
 & & 00 & 01 & 11 & 10 \\ \hline
\multirow{2}{*}{C} & 0 & 0 & 0 & 0 & \boxed{1} \\ \hline
 & 1 & \boxed{1} & 0 & 0 & 0 \\ \hline
\end{array}$$

$$Y = AB'C' + A'B'C$$

$$Y = AB'C' + (A \text{ xor } B')A' + AB'C$$

|  |  | AB |  |  |  |
|---|---|---|---|---|---|
|  |  | 00 | 01 | 11 | 10 |
| C | 0 | 1 | 0 | 0 | 1 |
|  | 1 | 1 | 0 | 0 | 1 |

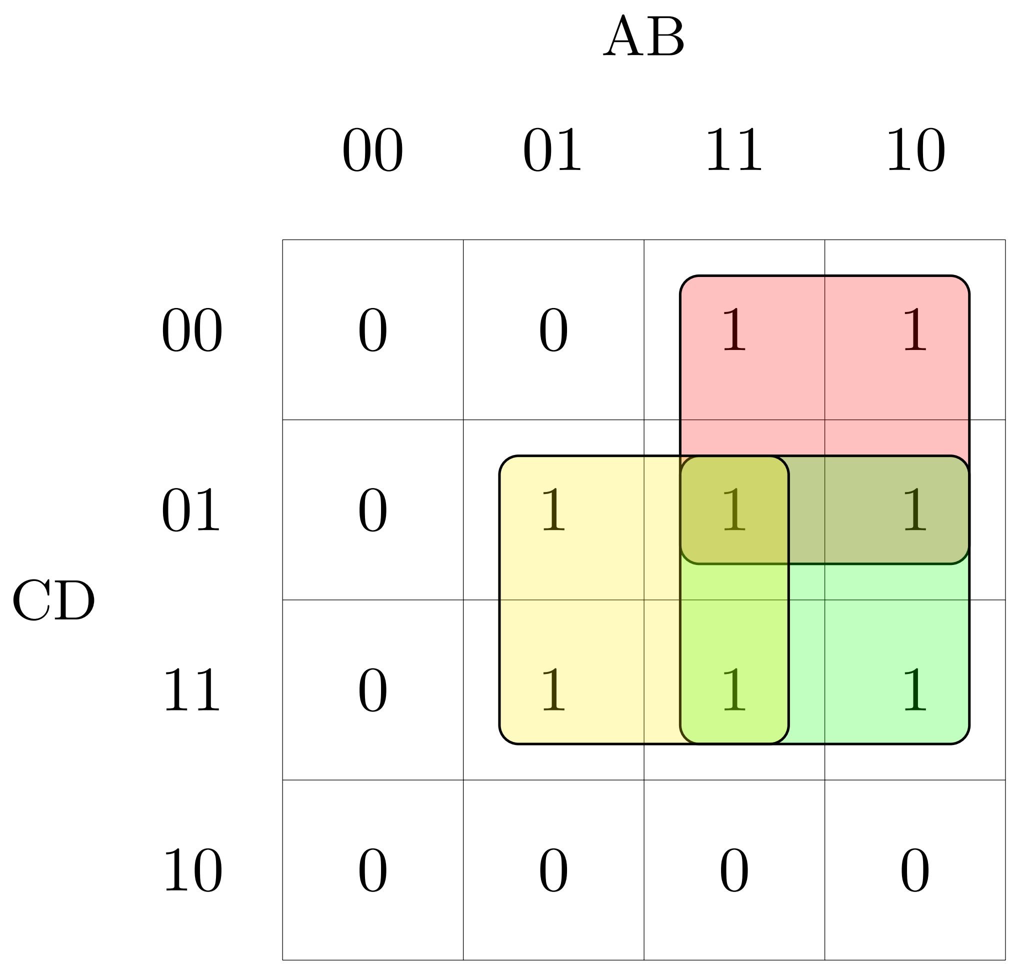


$Y = B'$

$$Y = A'BC'D + ABC'D' + ABC'D + AB'C'D' + A'BCD + ABCD + AB'CD + AB'C'D$$

AB

| CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 1 | 1 |
| 01 | 0 | 1 | 1 | 1 |
| 11 | 0 | 1 | 1 | 1 |
| 10 | 0 | 0 | 0 | 0 |



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

$Y = AC' + BD + AD$