

Karnaugh Maps

Digital Logic Design
Combinational Logic

Two-Level Simplification

Karnaugh Map Method

The K-map is a method of representing the truth table that helps visualize adjacencies in up to 6 dimensions

Beyond that, computer-based methods are needed

2-variable
K-map

3-variable
K-map

4-variable
K-map

Numbering Scheme: 00, 01, 11, 10
Gray Code — only a single bit changes from code word to next code word

2004 13

In the above slide, the k-maps are numbered like they would be in a truth table. From the truth table, you will take one output and place it inside the k-map. For example, the following 3 variable truth table would be represented with the k-map beside it. This ensures the inputs and outputs match up. The input values are on the outside of the k-map. Check to be sure the input combinations have the correct output.

| A | B | C | Output | Number |
|---|---|---|--------|--------|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 1 | 2 |
| 0 | 1 | 1 | 1 | 3 |
| 1 | 0 | 0 | 1 | 4 |
| 1 | 0 | 1 | 0 | 5 |
| 1 | 1 | 0 | 1 | 6 |
| 1 | 1 | 1 | 0 | 7 |

Next you will circle squares inside the k-map of adjacent minterms (ones). The following examples illustrate how to circle the minterms with different numbers of inputs. The general rule is you must have a **power of two** of minterms to circle. So 2, 4, 8, 16.

Two-Level Simplification

K-Map Method Examples

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

A asserted, unchanged
B varies

B complemented, unchanged
A varies

$F = A$

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

$G = B'$

How does this work?

With a group of four – we are applying the uniting theorem twice!

| | | | | | | |
|-----|---|---|----|----|----|----|
| | A | B | 00 | 01 | 11 | 10 |
| Cin | 0 | 0 | 0 | 0 | 1 | 0 |
| | 1 | 0 | 1 | 1 | 1 | 1 |

$Cout = A B + B Cin + A Cin$

| | | | | | | |
|---|---|---|----|----|----|----|
| | A | B | 00 | 01 | 11 | 10 |
| C | 0 | 0 | 0 | 0 | 1 | 1 |
| | 1 | 0 | 0 | 0 | 1 | 1 |

$F(A,B,C) = A$

Two-Level Simplification

More K-Map Method Examples, 3 Variables

| | | | | | | |
|---|---|---|----|----|----|----|
| | A | B | 00 | 01 | 11 | 10 |
| C | 0 | 1 | 0 | 0 | 1 | 1 |
| | 1 | 0 | 0 | 1 | 1 | 1 |

$F(A,B,C) = \Sigma m(0,4,5,7)$

$F = B' C' + A C$

In the K-map, adjacency wraps from left to right and from top to bottom

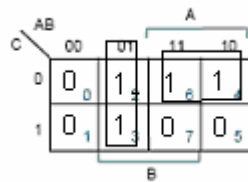
For F' simply circle 0's and write minterms

$F'(A,B,C) = \Sigma m(1,2,3,6)$

$F' = B C' + A' C$

Compare with the method of using DeMorgan's Theorem and Boolean Algebra to reduce the complement!

For our previous three variable k-map, it would be circled as below.



Generally, the fewer boxes you have, the more minimized the design is. Notice, however, that there are two boxes whose sides are adjacent. This can result in a static hazard. A static hazard is when the output momentarily jumps to the wrong output. For this example, when the input changes to 011 or 110 (the two adjacent squares) the output will jump to zero before going to the static one. This phenomenon can be explained by looking at the timing diagram of the final schematic.

To convert the squares into a usable logic equation, we will look at the change in the inputs. Look at the vertical box in our example. The inputs **A** and **B** stay constant inside the box. **A** is '0' and **B** is '1'. Therefore the first term will be

$$A'B$$

C changes values so it will not be used in the first term. For the second term, look at the horizontal box. **B** changes values so it will not be used. **A** stays at a constant '1'. **C** stays at '0'. Therefore the second term will be

$$AC'$$

The final equation will be

$$A'B + AC'$$

This process is illustrated again in the four variable k-map below.

Digital Logic Design
Combinational Logic

Two-Level Simplification
K-map Method Examples: 4 variables

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

$F(A,B,C,D) = \Sigma m(0,2,3,5,6,7,8,10,11,14,15)$

$F = C + A'B'D + B'D'$

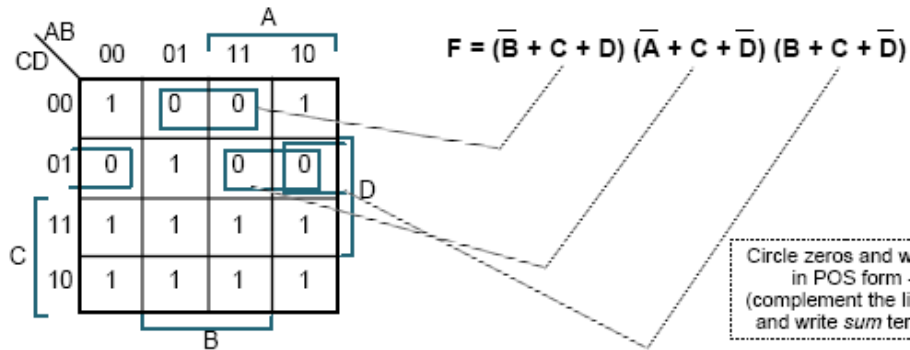
Find the smallest number of the largest possible *subcubes* that cover the ON-set

With a group of eight – we are effectively applying the uniting theorem three times!

2004 17

Two-Level Simplification

K-map Method: Circling Zeros



Replace F by \bar{F} , 0's become 1's and vice versa

$$\bar{F} = B \bar{C} \bar{D} + A \bar{C} \bar{D} + \bar{B} \bar{C} \bar{D}$$

$$\bar{F} = \bar{B} \bar{C} \bar{D} + A \bar{C} \bar{D} + B \bar{C} \bar{D}$$

$$F = (\bar{B} + C + D) (\bar{A} + C + \bar{D}) (B + C + \bar{D})$$

← Check the solution

By circling zeros, one can come up with the POS (product of sums) form.

Don't Cares

The next example explains how to use don't cares in a k-map. **The general rule:** if the X gives you a bigger square, circle it. If it doesn't, there is no need for it. Using don't cares will help minimize designs more fully.

Two-Level Simplification

K-map Example: Don't Cares

Don't Cares can be treated as 1's or 0's if it is advantageous to do so

| | | | | | | |
|---|----|----|----|----|----|---|
| | | AB | | A | | |
| | | 00 | 01 | 11 | 10 | |
| C | CD | 00 | 0 | 0 | X | 0 |
| | 01 | 1 | 1 | X | 1 | |
| | 11 | 1 | 1 | 0 | 0 | |
| | 10 | 0 | X | 0 | 0 | |

$F(A,B,C,D) = \Sigma m(1,3,5,7,9) + \Sigma d(6,12,13)$

$F = A'D + B'C'D$ w/o don't cares

$F = C'D + A'D$ w/ don't cares

By treating this DC as a "1", a 2-cube can be formed rather than one 0-cube

| | | | | | | |
|---|----|----|----|----|----|---|
| | | AB | | A | | |
| | | 00 | 01 | 11 | 10 | |
| C | CD | 00 | 0 | 0 | X | 0 |
| | 01 | 1 | 1 | X | 1 | |
| | 11 | 1 | 1 | 0 | 0 | |
| | 10 | 0 | X | 0 | 0 | |

In POS form: $F = D(A' + C')$

Same answer as above,
but fewer literals

2004
21

Extended Example: Comparator

Two-Level Simplification
Design Example: Two Bit Comparator

| A | B | C | D | F ₁ | F ₂ | F ₃ |
|---|---|---|---|----------------|----------------|----------------|
| 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 |

Block Diagram and Truth Table

A 4-Variable K-map for each of the 3 output functions

A two-bit comparator is an arithmetic circuit that compares the size of two binary numbers, each two bits in length.

2004
22

Two-Level Simplification
Design Example: Two Bit Comparator

K-map for F₁

K-map for F₂

K-map for F₃

Digital Logic Design
Combinational Logic

F₁ = A' B' C' D' + A' B C' D + A B C D + A B' C D'

F₂ = A' B' D + A' C + B' C D

F₃ = B C' D' + A C' + A B D'

2004
23