# Karnaugh Maps

An alternative representation of a switching function that is isomorphic to a truth table but more useful for optimization is a *Karnaugh map*.

The binary patterns of the possible assignments to $n$ variables are organized as a hypercube in tabular form (as opposed to graph form typically used. See textbook for a discussion of hypercubes.)

**A two−variable Karnaugh Map**

| | | X = 0 | X = 1 |
|---|---|---|---|
| Y | 0 | F0 | F2 |
| | 1 | F1 | F3 |

**there is an exact correspondence to the rows of the truth table, Fi is the output value for the i−th row of the truth table**

| | | X = 0 | X = 1 |
|---|---|---|---|
| Y | 0 | 1 | 1 |
| | 1 | 0 | 0 |

$$F(X,Y) = X'Y' + XY' = m_0 + m_2$$

**A three−variable Karnaugh Map**

| Z \ XY | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | F0 | F2 | F6 | F4 |
| 1 | F1 | F3 | F7 | F5 |

**Note the order of XY pairs and the resulting mapping to the Fi**

| Z \ XY | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 |

$$F(X, Y, Z) = X'Y'Z' + X'YZ' + XYZ$$
$$= m_0 + m_2 + m_7$$

**A four−variable Karnaugh Map**

| YZ \ WX | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | F0 | F4 | F12 | F8 |
| 01 | F1 | F5 | F13 | F9 |
| 11 | F3 | F7 | F15 | F11 |
| 10 | F2 | F6 | F14 | F10 |

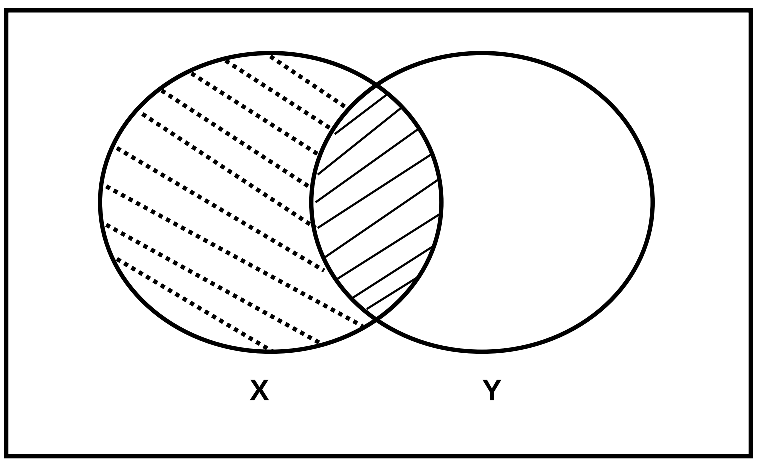
| YZ \ WX | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | 0 | 0 | 0 |
| 01 | 1 | 0 | 0 | 0 |
| 11 | 0 | 1 | 1 | 0 |
| 10 | 0 | 1 | 1 | 0 |

$$F(W, X, Y, Z) = W'X'Y'Z' + W'X'Y'Z + W'XYZ' + W'XYZ + WXYZ' + WXYZ$$
$$= m_0 + m_1 + m_6 + m_7 + m_{14} + m_{15}$$

**Basic Theorm of K−Maps**

$$X Y + X Y' = X$$

**keep the common expression of the two products**



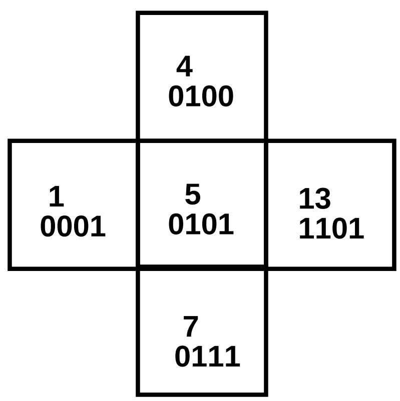

$m_2$ + $m_3$

10 11 differ by one binary digit

**K−maps are set up to reflect such single bit differences**

| YZ \ WX | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | F0 | F4 | F12 | F8 |
| 01 | F1 | F5 | F13 | F9 |
| 11 | F3 | F7 | F15 | F11 |
| 10 | F2 | F6 | F14 | F10 |

**cells are adjacent if they differ by one bit, i.e. moving up/down or right/left moves to a cell corresponding to a row with only one bit changed in the binary representation of the starting row index.**




**Therefore, we can use the Theorem and the single bit difference property to identify minterms that can be combined into one product term in a sum−of−products via several applications of the Theorem.**

**Subcubes are of interest, i.e., rectangles that have dimensions that are powers of 2. E.G. 2 X 1, 1 X 2, 2 X 2, 4 X 1, etc.**

## A three−variable Karnaugh Map

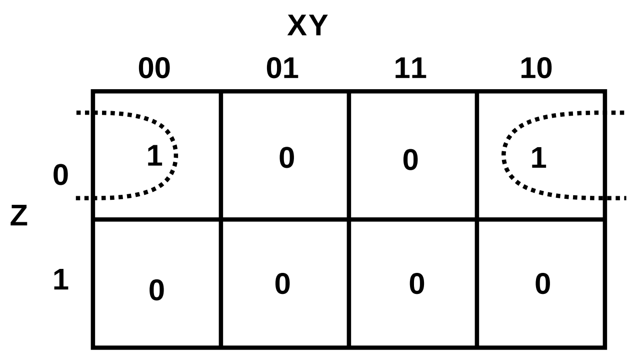
### Two−term combination

| Z \ XY | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 |

F ( X , Y , Z ) = X ’ Y ’ Z ’ + X Y ’ Z ’ (canonical sum)

= X ’ ( Y ’ Z ’ ) + X ( Y ’ Z ’ ) (associativity)

= Y ’ Z ’ (K−map Theorem)




This reduction can be done by inspection of the K−map and identifying a 1 X 2 subcube (note wraparound)

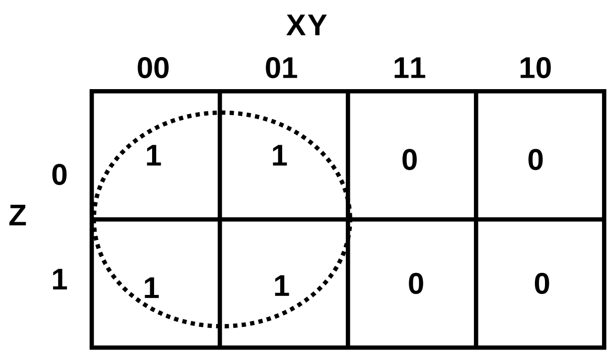
X ’ Y ’ Z ’ + X Y ’ Z ’ ⟶ Y ’ Z ’

000 100 *00

X changes, i.e., X and X’ present and Y’ and Z’ are constant in circled cells

**A three−variable Karnaugh Map**

**Four−term combination**




X’ Y’ Z’ + X’YZ’ + X’Y’Z + X’YZ ⟶ X’

000 010 001 011 0**

changing in Y and Z, constant in X’
over the cells grouped together

12 literals reduced to 1

This is equivalent to the following algebraic reduction via the K−map theorem

X’ Y’ Z’ + X’ Y Z + X’ Y Z’ + X’ Y’ Z

↓

X’ ( Y’ Z’ + Y Z ) + X’ ( Y Z’ + Y’ Z )

↓

X’ Q + X’ Q’

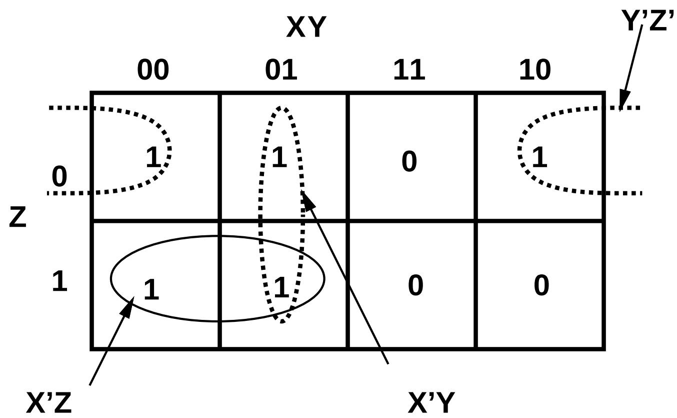
↓

X’

## A three−variable Karnaugh Map

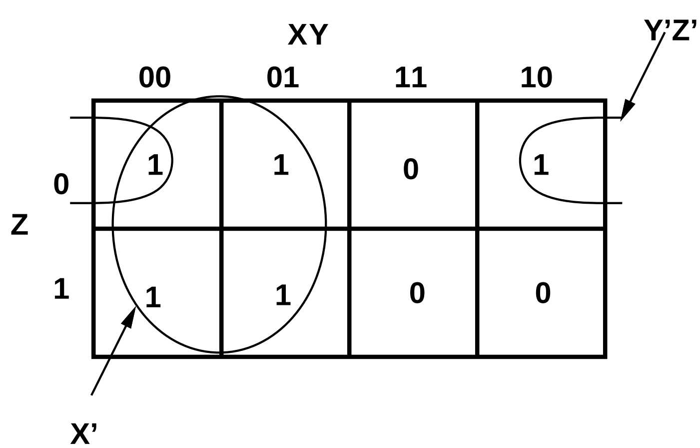
### Nested groupings

X’ Y’ Z’ + X’ Y’ Z + X’ Y Z’ + X’ Y Z + X Y’ Z’




F = X’Y + X’Z + Y’Z’

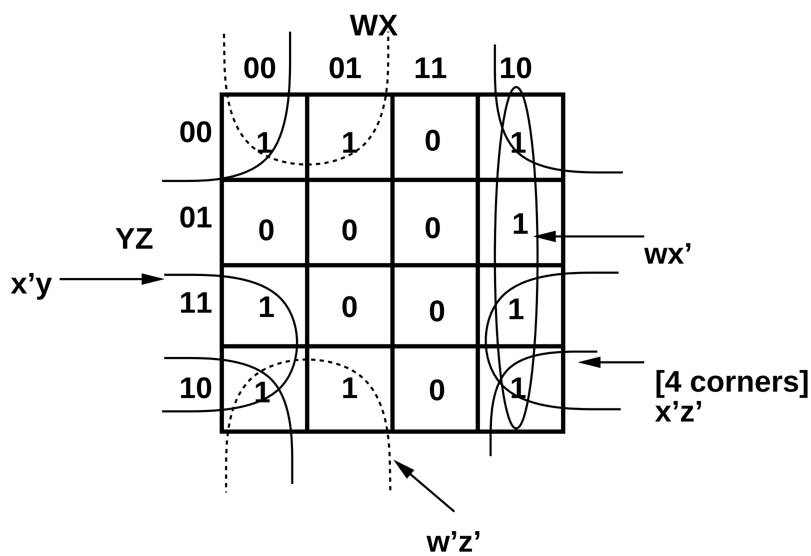
uses two−term combinations




F = X’ + Y’Z’

uses four−term combinations to
cover two−term combinations and
reduce the number of products needed

A four−variable Karnaugh Map
and redundant subcubes




F = w’z’ + wx’ + x’z’ + x’y

no nesting but there is redundancy in
the coverage of the minterms

the 4 corners term x’z’ only covers
minterms already covered by the others

F = w’z’ + wx’ + x’y

Note that none of these can be removed
without loss of coverage and therefore
a change in the switching expression value

Therefore the K-map design goal is to

> Find the smallest collection of subcubes that cover all minterms for which $f = 1$ in the K-map such that each subcube is as large as possible.

When the result is implemented in the standard AND-OR two-level network for a sum-of-products another form of this design goal is clear.

> Find an expression with the smallest number of products (objective 1); and with the smallest number of literals of all SOPs satisfying objective 1 (objective 2)

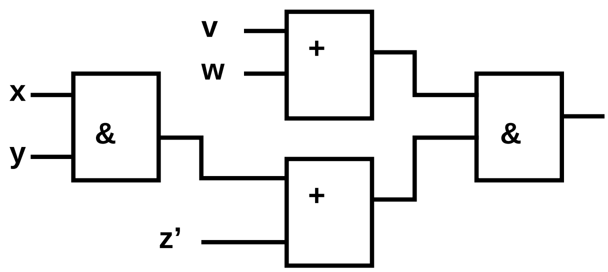
The first objective minimizes the number of AND gates needed; the second objective minimizes their complexity, i.e., the number of inputs required to those gates.

**Definition:** An SOP that satisfies the above condition is called a *minimal* SOP.

Note the term minimal is used since there may be more than one SOP that satisfies the conditions.

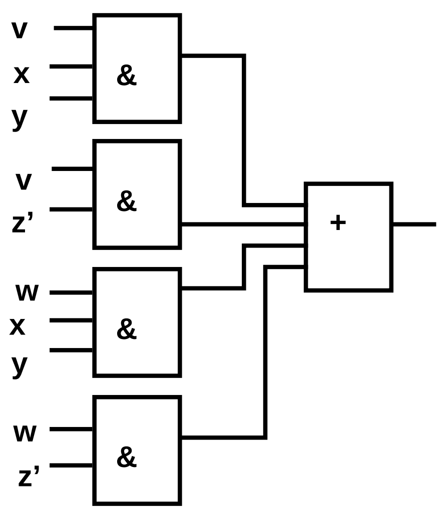
In general, the design tradeoffs tend to be delay vs. gates. Ideally an implementation of a switching function is fast (small delay) and inexpensive (few gates therefore low area)

( v + w ) ( xy + z' )




4 gates 3 levels delay

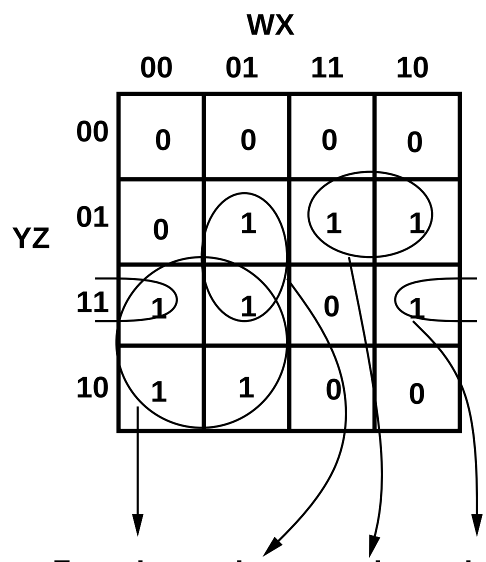
vxy + vz' + wxy + wz'




5 gates 2 levels delay

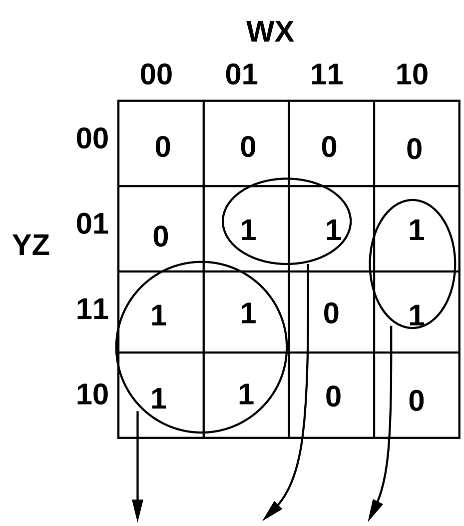
(although note the more complex gates and therefore potential difference in delay per level)

## Irredundant Sums




F = w'y + w'xz + wy'z + x'yz

**irredundant. i.e., no redundant terms can be removed, but it is not minimal**




F = w'y + xy'z + wx'z

**minimal sum arises from different covering which is also irredundant**

**Example:**

An insurance company will issue you a policy if you are

- married, male and had a defensive driving course, or
- married, male and never had an accident, or
- married, female and had accidents, or
- never had an accident and had a defensive driving course, or
- married and had no defensive driving course.

Define the switching variables

- $A = 1$ if married, 0 if not married
- $B = 1$ if male, 0 if female
- $C = 1$ if never had an accident, 0 if not
- $D = 1$ if had a defensive driving course, 0 if not
- $P = 1$ if policy issued, 0 if not

We can write a SOP expression directly from the conditions

$$P = ABD + ABC + AB'C' + CD + AD'$$

$$\begin{aligned}
P &= ABD + ABC + AB'C' + CD + AD' \\
Comm. &= ABD + AD' + ABC + AB'C' + CD \\
Dist. &= A(BD + D') + ABC + AB'C' + CD \\
No-Name &= A(B + D') + ABC + AB'C' + CD \\
Dist. &= AB + AD' + ABC + AB'C' + CD \\
Comm. &= AB + ABC + AD' + AB'C' + CD \\
Dist. &= AB(1 + C) + AD' + AB'C' + CD \\
Ident. &= AB + AD' + AB'C' + CD \\
Comm. &= AB + AB'C' + AD' + CD \\
Dist. &= A(B + B'C') + AD' + CD \\
No-Name &= A(B + C') + AD' + CD \\
2Dist. &= AB + A(C' + D') + CD \\
DeMor. &= AB + A(CD)' + CD \\
No-Name &= AB + A + CD \\
Absorp. &= A + CD
\end{aligned}$$

$$\begin{aligned} P &= ABD + ABC + AB'C' + CD + AD' \\ &= A + CD \end{aligned}$$

SOP yields 2 levels and 5 AND gates(multi-input) and 1 OR gate (multi-input)

Equivalent expression SOP yields 2 levels 1 2-input AND 1 2-input OR.

**A four−variable Karnaugh Map solution to the insurance problem**