Function Simplification

- Why simplify?
  - Simpler expression uses less logic gates.
  - Thus: cheaper, less power, faster (sometimes).
- Simplification techniques:
  - Algebraic Simplification.
    - simplify symbolically using theorems/postulates.
    - requires skill but extremely open-ended.
  - Karnaugh Maps.
    - diagrammatic technique using 'Venn-like diagram'.
    - easy for humans (pattern-matching skills).
    - simplified standard forms.
    - limited to not more than 6 variables.
  - Quine-McCluskey tabulation technique.

Algebraic Simplification (1/4)

- Algebraic simplification aims to minimise
  - (i) number of literals, and
  - (ii) number of terms
- But sometimes conflicting.
- Let's aim at reducing the number of literals.

Algebraic Simplification (2/4)

- Find minimal SOP and POS expressions of

$f(x,y,z) = x'.y.(z + y'.x) + y'.z$

$= x'.y.(z+y'.x) + y'.z$

$= x'.y.z + x'.y.y'.x + y'.z$ (distributivity)

$= x'.y.z + 0 + y'.z$ (complement, null element 0)

$= x'.y.z + y'.z$ (identity 0)

$= x'.z + y'.z$ (absorption)

$= (x' + y').z$ (distributivity)

Minimal SOP of *f* = **$x'.z + y'.z$** (2 2-input AND gates and 1 2-input OR gate)

Minimal POS of *f* = **$(x' + y').z$** (1 2-input OR gate and 1 2-input AND gate)

Algebraic Simplification (3/4)

- Find minimal SOP expression of

  f(a,b,c,d) = a.b.c + a.b.d + a'.b.c' + c.d + b.d'

  = a.b.c + a.b.d + a'.b.c' + c.d + b.d'

  = a.b.c + a.b + a'.b.c' + c.d + b.d' (absorption)

  = a.b.c + a.b + b.c' + c.d + b.d' (absorption)

  = a.b + b.c' + c.d + b.d' (absorption)

  = a.b + c.d + b.(c' + d') (distributivity)

  = a.b + c.d + b.(c.d)' (DeMorgan)

  = a.b + c.d + b (absorption)

  = b + c.d (absorption)

  Number of literals reduced form 13 to 3.

Algebraic Simplification (4/4)

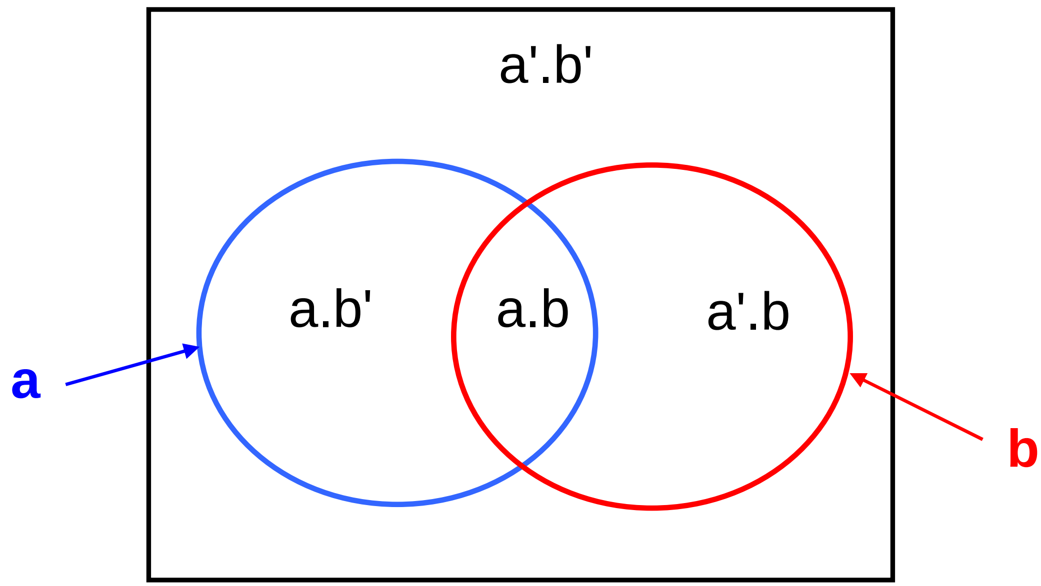
- Difficulty – needs good algebraic manipulation skills.
- Advantage – very open-ended (to your desired form!)

Introduction to K-maps

- Systematic method to obtain *simplified* sum-of-products (SOPs) or product -of-sums (POSs) Boolean expressions.
- It is a pictorial format to show relationship between logic inputs and the desired output.
- Objective: Minimum possible terms & with fewest possible number of literals in each term.
- Diagrammatic technique based on a special form of *Venn diagram.*
- Advantage: Easy with visual aid.
- Disadvantage: Limited to 5 or 6 variables.

Venn Diagrams (1/2)

- Venn diagram to represent the space of minterms.
- Example of 2 variables (4 minterms):



Venn Diagrams (2/2)

- Each set of minterms represents a Boolean function. Examples:

{ a.b, a.b' } → a.b + a.b' = a.(b+b') = a

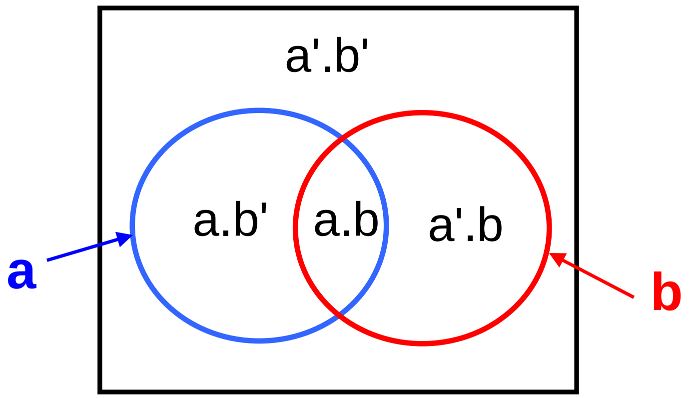
{ a'.b, a.b } → a'.b + a.b = (a'+a).b = b

{ a.b } → a.b

{ a.b, a.b', a'.b } → a.b + a.b' + a'.b = a + b

{ } → 0

{ a'.b',a.b,a.b',a'.b } → 1



2-variable K-maps (1/4)

- **Karnaugh-map** (K-map) is an abstract form of Venn diagram, organised as a matrix of squares, where
  - each square represents a minterm
  - adjacent squares always differ by just one literal (so that the theorem may apply:

    $a + a' = 1$)
- For 2-variable case (e.g.: variables a,b), the map can be drawn as:

2-variable K-maps (2/4)

- Alternative layouts of a 2-variable (a, b) K-map

Alternative 1:

		b
	a'b'	a'b
a	ab'	ab

OR

		b
	m0	*m1*
a	*m2*	*m3*

Alternative 2:

		a
	a'b'	ab'
b	a'b	ab

OR

		a
	m0	*m2*
b	*m1*	*m3*

Alternative 3:

	a	
b	ab	a'b
	ab'	a'b'

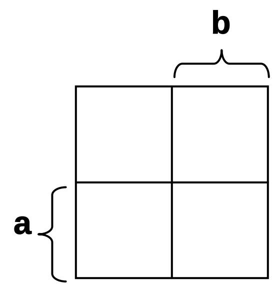
OR

	a	
b	*m3*	*m1*
	m2	*m0*

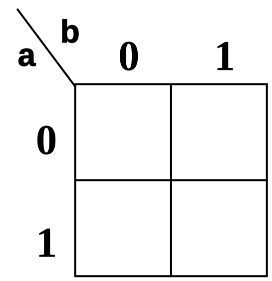
and others…

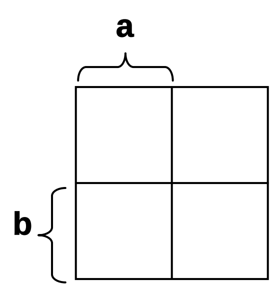
2-variable K-maps (3/4)

- Equivalent labeling:

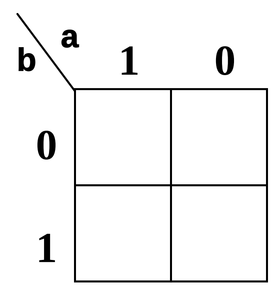


equivalent to**:**



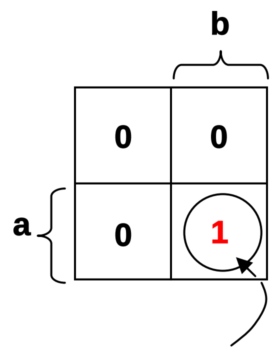


equivalent to:

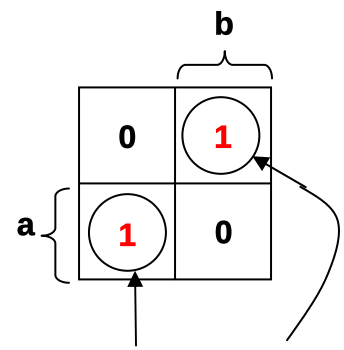


2-variable K-maps (4/4)

- The K-map for a function is specified by putting
  - a ‘1’ in the square corresponding to a minterm
  - a ‘0’ otherwise
- For example: Carry and Sum of a half adder.



$C = a.b$



$S = a.b' + a'.b$

3-variable K-maps (1/2)

- There are 8 minterms for 3 variables (a, b, c). Therefore, there are 8 cells in a 3-variable K-map.

a \ bc	00	01	11	10
0	a'b'c'	a'b'c	a'bc	a'bc'
1	ab'c'	ab'c	abc	abc'

OR

a \ bc	00	01	11	10
0	*m0*	*m1*	*m3*	*m2*
1	*m4*	*m5*	*m7*	*m6*

Above arrangement ensures that minterms of adjacent cells differ by only *ONE literal*. (Other arrangements which satisfy this criterion may also be used.)

3-variable K-maps (2/2)

- There is wrap-around in the K-map:
  - a'.b'.c' (*m0*) is adjacent to a'.b.c' (*m2*)
  - a.b'.c' (*m4*) is adjacent to a.b.c' (*m6*)

a \ bc	00	01	11	10
0	*m0*	*m1*	*m3*	*m2*
1	*m4*	*m5*	*m7*	*m6*

Each cell in a 3-variable K-map has 3 adjacent neighbours. In general, each cell in an *n*-variable K-map has *n* adjacent neighbours. For example, *m0* has 3 adjacent neighbours: *m1*, *m2* and *m4*.

Questions

The K-map of a 3-variable function *F* is shown below. What is the sum-of-minterms expression of *F*?

a \ bc	00	01	11	10
0	1	0	0	1
1	0	1	0	0

Draw the K-map for this function *A*:

$$A(x, y, z) = x.y + y.z' + x'.y'.z$$

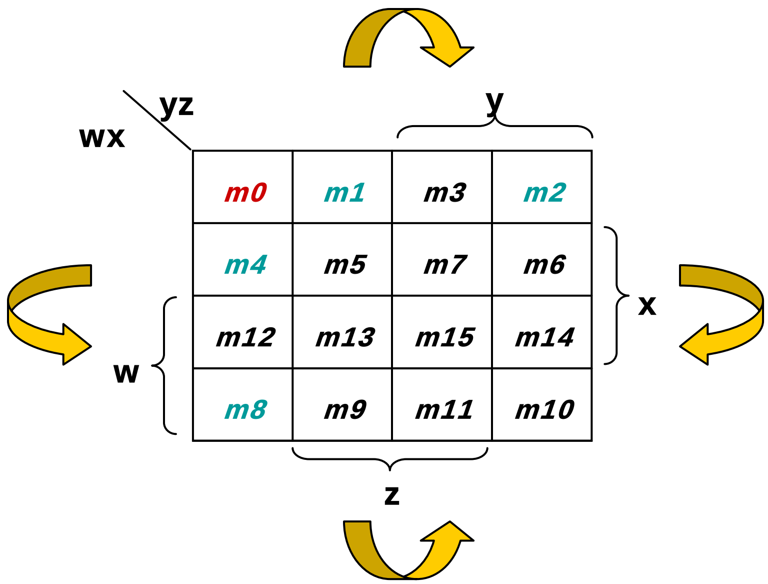
4-variable K-maps (1/2)

- There are 16 cells in a 4-variable (w, x, y, z) K-map.

wx \ yz	00	01	11	10
00	*m0*	*m1*	*m3*	*m2*
01	*m4*	*m5*	*m7*	*m6*
11	*m12*	*m13*	*m15*	*m14*
10	*m8*	*m9*	*m11*	*m10*

4-variable K-maps (2/2)

- There are 2 wrap-arounds: a horizontal wrap-around and a vertical wrap-around.
- Every cell thus has 4 neighbours. For example, the cell corresponding to minterm *m0* has neighbours *m1*, *m2*, *m4* and *m8*.



Larger K-maps

- Maps of more than 4 variables are more difficult to use because the geometry for combining adjacent squares becomes more involved.
- For 5 variables, e.g. vwxyz, need $2^5 = 32$ squares.

- 6-variable K-map is pushing the limit of human "pattern-recognition" capability.
- K-maps larger than 6 variables are practically unheard of!
- Normally, a 6-variable K-map is organised as four 4-variable K-maps, which are mirrored along two axes.

Simplification Using K-maps (1/9)

- Based on the Theorem:

$$A + A' = 1$$

- In a K-map, each cell containing a ‘1’ corresponds to a minterm of a given function *F*.
- Each group of adjacent cells containing ‘1’ (group must have size *in powers of twos*: 1, 2, 4, 8, …) then corresponds to a simpler product term of *F*.
  - ❖ Grouping (Looping) 2 adjacent squares eliminates 1 variable, grouping 4 squares eliminates 2 variables, grouping 8 squares eliminates 3 variables, and so on. In general, grouping 2^n squares eliminates *n* variables.

Simplification Using K-maps (2/9)

- Group as many squares as possible.
  - The larger the group is, the fewer the number of literals in the resulting product term.
- Select as few groups as possible to cover all the squares (minterms) of the function.
  - The fewer the groups, the fewer the number of product terms in the minimized function.

Simplification Using K-maps (3/9)

- Example:

$$F\,(w,x,y,z) = w'.x.y'.z' + w'.x.y'.z + w.x'.y.z' + w.x'.y.z + w.x.y.z' + w.x.y.z$$

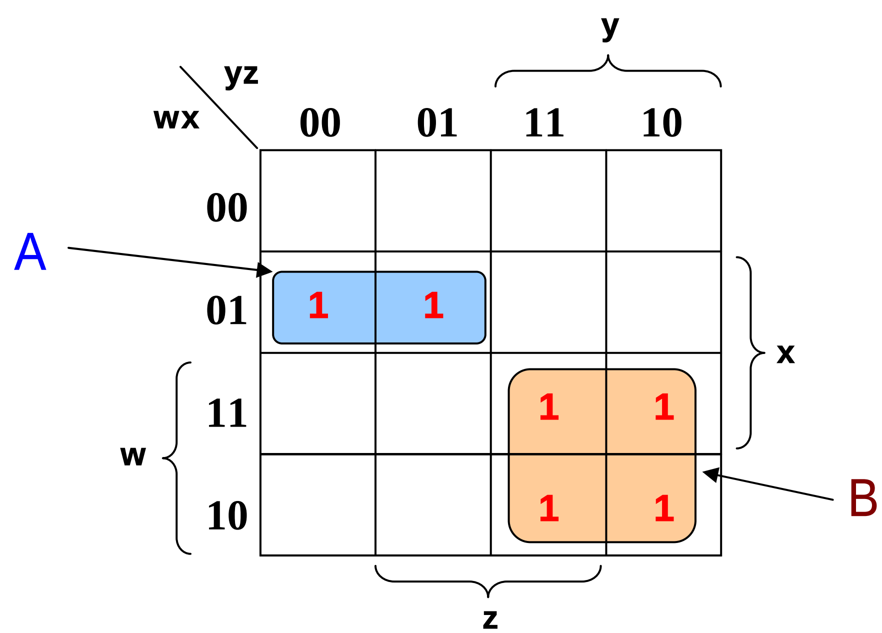
$$= \Sigma\, m(4, 5, 10, 11, 14, 15)$$

wx \ yz	00	01	11	10
00				
01	1	1		
11			1	1
10			1	1

(cells with ‘0’ are not shown for clarity)

Simplification Using K-maps (4/9)

- Each group of adjacent minterms (group size in powers of twos) corresponds to a possible product term of the given function.



Simplification Using K-maps (5/9)

- There are 2 groups of minterms: A and B, where:

A $= w'.x.y'.z' + w'.x.y'.z$

$= w'.x.y'.(z' + z)$

$= w'.x.y'$

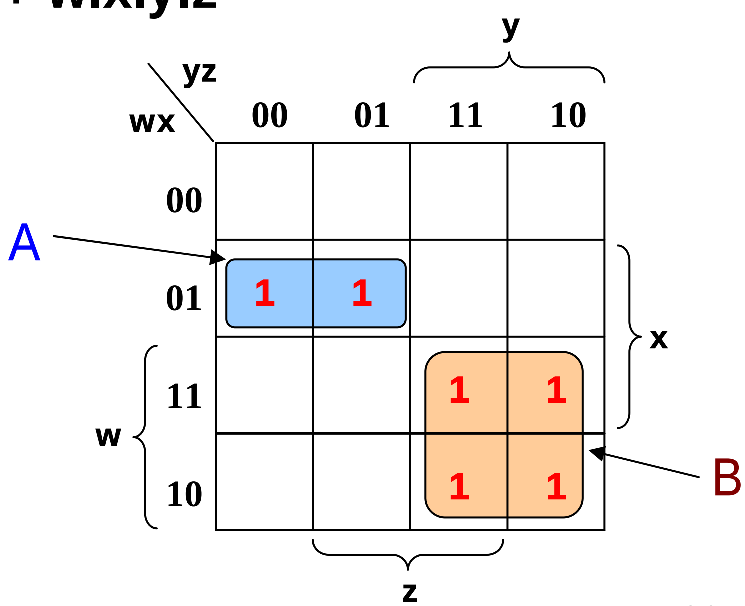
B $= w.x'.y.z' + w.x'.y.z + w.x.y.z' + w.x.y.z$

$= w.x'.y.(z' + z) + w.x.y.(z' + z)$

$= w.x'.y + w.x.y$

$= w.(x'+x).y$

$= w.y$



Simplification Using K-maps (6/9)

- Each product term of a group, w'.x.y' and w.y, represents the *sum of minterms* in that group.
- Boolean function is therefore the sum of product terms (SOP) which represent all groups of the minterms of the function.

$$F(w,x,y,z) = A + B = w'.x.y' + w.y$$

Simplification Using K-maps (7/9)

- Larger groups correspond to product terms of fewer literals. In the case of a 4-variable K-map:

singlets, 1 cell= 4 literals, e.g.: w.x.y.z, w'.x.y'.z

pair, 2 cells = 3 literals, e.g.: w.x.y, w.y'.z'

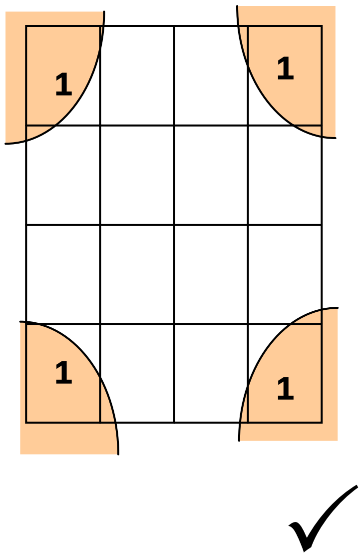
quads, 4 cells = 2 literals, e.g.: w.x, x'.y

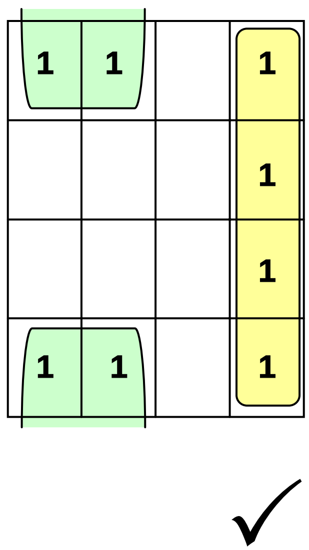
octets, 8 cells = 1 literal, e.g.: w, y', z

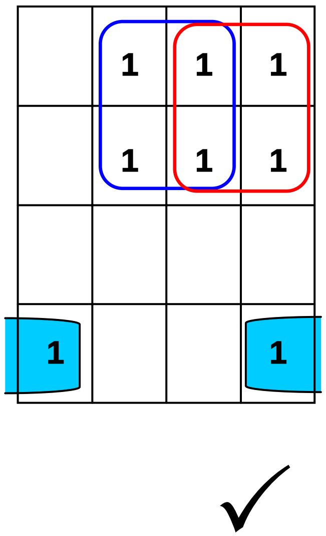
16 cells = no literal, e.g.: 1

Simplification Using K-maps (8/9)

- Other possible valid groupings of a 4-variable K-map include:







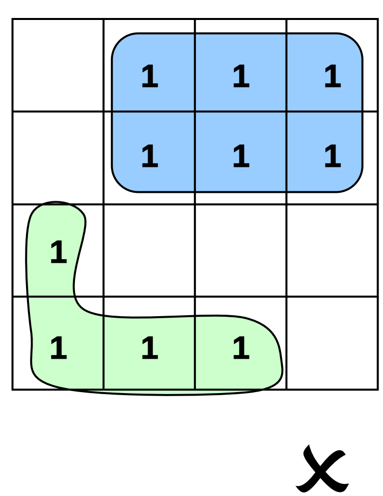
Simplification Using K-maps (9/9)

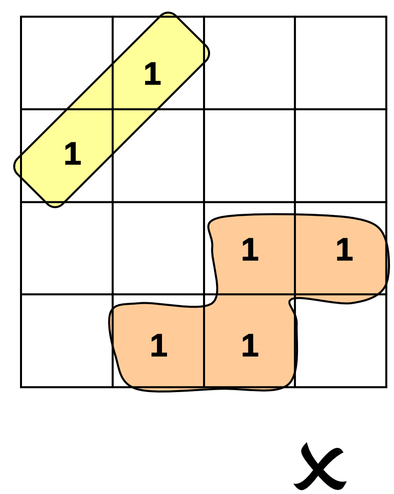
- Groups of minterms must be

  (1) rectangular, and

  (2) have size in powers of 2's.

  Otherwise they are *invalid* groups. Some examples of *invalid groups*:





Converting to Minterms Form (1/2)

- The K-map of a function is easily drawn when the function is given in canonical sum-of-products, or sum-of-minterms form.
- What if the function is not in sum-of-minterms?
  - ❖ Convert it to sum-of-products (SOP) form.
  - ❖ Expand the SOP expression into sum-of-minterms expression, or fill in the K-map directly based on the SOP expression.

Converting to Minterms Form (2/2)

- Example: **f(A,B,C,D) = A.(C+D)'.(B'+D') + C.(B+C'+A'.D)**

  = A.(C'.D').(B'+D') + B.C + C.C' + A'.C.D

  = A.B'.C'.D' + A.C'.D' + B.C + A'.C.D

A.B'.C'.D' + A.C'.D' + B.C + A'.C.D

= A.B'.C'.D' + A.C'.D'.(B+B') + B.C + A'.C.D

= A.B'.C'.D' + A.B.C'.D' + A.B'.C'.D' + B.C.(A+A') + A'.C.D

= A.B'.C'.D' + A.B.C'.D' + A.B.C + A'.B.C + A'.C.D

= A.B'.C'.D' + A.B.C'.D' + A.B.C.(D+D') + A'.B.C.(D+D') + A'.C.D.(B+B')

= A.B'.C'.D' + A.B.C'.D' + A.B.C.D + A.B.C.D' + A'.B.C.D + A'.B.C.D' + A'.B'.C.D