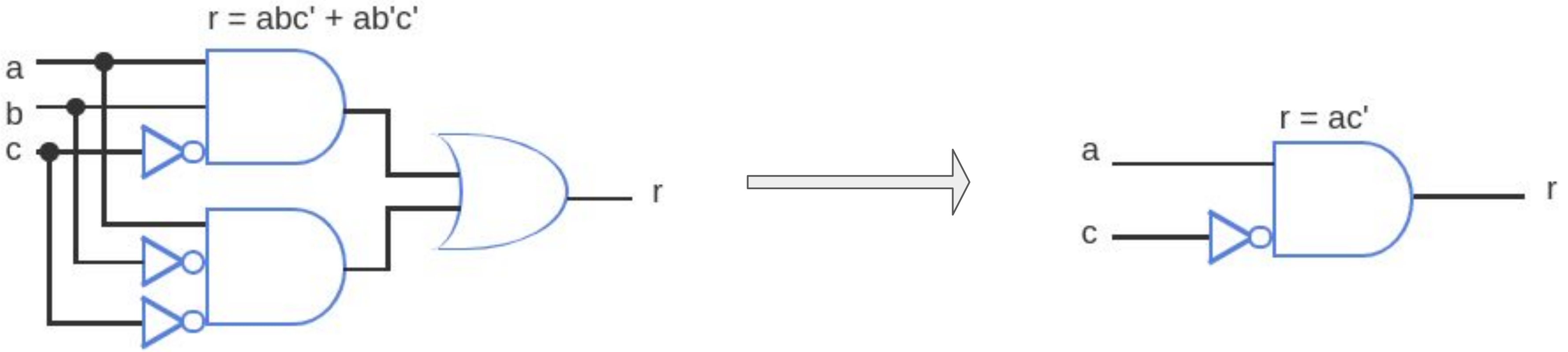
Simplification

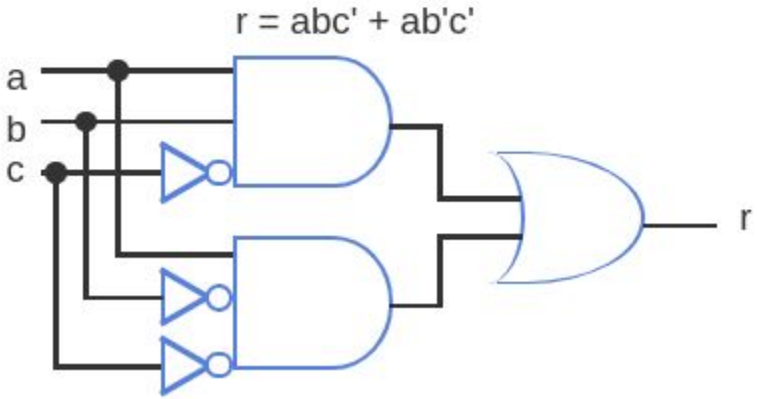
Two-level combinational logic simplification

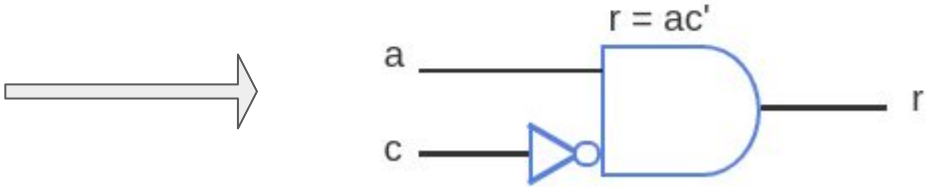
Logic simplification

Logic simplification (also referred to as logic minimization) means to simplify a Boolean expression before converting to a circuit to yield a smaller circuit.



Two-level combinational logic simplification





$r = a\,b\,c' + a\,b'\,c'$	
$r = a\,c'\,(b + b')$	Distributive
$r = a\,c'\,(1)$	Complement
$r = a\,c'$	Identity

Two-level combinational logic simplification

Simplifying algebraically can be hard, a ***designer may not see any opportunity*** to simplify.

1) To help with this process, a designer converts to ***sum-of-minterms*** form.

2) designer must create i(j + j') opportunities by **replicating a term**.
Note: The need for such replication may not be obvious.

3) Then the designer must apply i(j + j') simplifications.
Again, the places where i(j + j') can be applied aren't obvious.

$ab+a'$	
$ab+a'(1)$	by Identity
$ab+a'(b+b')$	by Complement
$ab+a'b+a'b'$	by Distributive
$ab+a'b+a'b+a'b'$	by Idempotent
$(a+a')b+a'(b+b')$	by Distributive
$(1)b+a'(1)$	by Complement
$\boxed{a'+b}$	Identity and Commutative

K-map (Karnaugh map) simplification

help us to make $i(j + j')$ simplification opportunities obvious

K-map map simplification

A ***K-map*** is a graphical function representation that *eases the simplification process* for expressions involving a few variables, by adjacently placing minterms that differ in exactly one variable.

Let us consider a function of ***two*** variables a,b: y = a'b' + ab'

	a	b	y
a'b'	0	0	1
a'b	0	1	0
ab'	1	0	1
ab	1	1	0

Our K-map consists of four cells:

two rows (for a = 0 and a = 1).
two columns (for b = 0 and b = 1)

a \ b	0	1
0	1	0
1	1	0

K-map is short for *Karnaugh map* (Karnaugh wiki)

K-map map simplification

A ***K-map*** is a graphical function representation that *eases the simplification process* for expressions involving a few variables, by adjacently placing minterms that differ in exactly one variable.

Let us consider a function of ***two*** variables a,b: $y = a'b' + ab'$

$y = b'$

Our K-map consists of four cells:

two rows (for a = 0 and a = 1).
two columns (for b = 0 and b = 1)

a \ b	0	1
0	a'b'	a'b
1	ab'	ab

a \ b	0	1
0	1	0
1	1	0

b'

to simplify using K-map:

1) circle adjacent cells that have 1

2) write a product term with the *differing* variable omitted

Because adjacent minterm cells differ in exactly one literal, K-map's key benefit is to make i(j + j') simplification opportunities obvious

K-map map simplification

Practice: $y = a'b' + ab' + ab$

a \ b	0	1
0	1	0
1	1	1

a

b’

to simplify using K-map:

1) circle adjacent cells that have 1
2) write a product term with the differing variable omitted

a \ b	0	1
0	a’b’	a’b
1	ab’	ab

Therefore answer is $y = a + b'$

Expression simplification in programming

*Computer programmers sometimes simplify expressions using **K-maps**.*

For example: A program may make a decision based on an expression:

If ((isRed AND !isBlue) OR (!isRed AND isBlue) OR (isRed AND isBlue)) then take action X

The expression represents a function with two variables.
Using a two-variable K-map, the programmer simplifies the expression, yielding:
If (isRed OR isBlue) then take action X.

Therefore y = a + b

a = isRed
b = isBlue

→ f(a,b) = ab’ + a’b + ab

a \ b	0	1
0	a’b’	a’b
1	ab’	ab

	a \ b	0	1	
a’b’	0	0	1	a’b
ab’	1	1	1	ab

a

b

K-map map simplification

Let us consider a function of ***three*** variables a,b,c: $f(a,b,c) = a'b'c + ab'c + abc' + abc$

	a	b	c	f(a,b,c)
a’b’c’	0	0	0	0
a’b’c	0	0	1	1
a’bc’	0	1	0	0
a’bc	0	1	1	0
ab’c’	1	0	0	0
ab’c	1	0	1	1
abc’	1	1	0	1
abc	1	1	1	1

Our K-map will consist of 8 cells:

- two rows (for a = 0 and a = 1).
- four columns (for bc = 00, bc=01, bc=11, bc=10)

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

K-map map simplification

Let us consider a function of ***three*** variables a,b,c: f(a,b,c) = a'b'c + ab'c + abc' + abc

	a	b	c	f(a,b,c)
a'b'c'	0	0	0	0
a'b'c	0	0	1	1
a'bc'	0	1	0	0
a'bc	0	1	1	0
ab'c'	1	0	0	0
ab'c	1	0	1	1
abc'	1	1	0	1
abc	1	1	1	1

Our K-map will consist of 8 cells:

two rows (for a = 0 and a = 1).

four columns (for bc = 00, bc=01, bc=11, bc=01)

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

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

K-map map simplification

Let us consider a function of **three** variables a,b,c: f(a,b,c) = a'b'c + ab'c + abc' + abc

To simplify using K-map: we circle adjacent cells containing **1** to form groups for simplification.
But we will expand the rules that we used to group:

- size of the group must be a power of 2 → for function of 3 variable valid group sizes: 2^0, 2^1, 2^2,2^3
- Groups should be as large as possible. → start from highest power
- Every 1 must be in at least one group. → circle a single 1 if it is not adjacent
- Overlapping allowed.
- Wrap around allowed.
- fewest number of groups

Simplify:
write a product term with the differing variable omitted

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

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

ab

b'c

Therefore f(a,b,c) = ab + b'c

K-map map simplification

Let us consider a function of **three** variables a,b,c: f(a,b,c) = a'b'c + ab'c + abc' + abc

To simplify using K-map: we circle adjacent cells containing **1** to form groups for simplification.
But we will expand the rules that we used to group:

- size of the group must be a power of 2 → for function of 3 variable valid group sizes: 2^0, 2^1, 2^2,2^3
- Groups should be as large as possible. → start from highest power
- Every 1 must be in at least one group. → circle a single 1 if it is not adjacent
- Overlapping allowed.
- Wrap around allowed.
- fewest number of groups

Simplify:
write a product term with the differing variable omitted

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

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

Why is the above grouping not correct?

K-map map simplification

Let us consider a function of **three** variables a,b,c: f(a,b,c) = a'b'c + ab'c + abc’ + abc

To simplify using K-map: we circle adjacent cell containing **1** to form groups for simplification.
But we will expand the rules that we used to group:

- size of the group must be a power of 2 → for function of 3 variable valid group sizes: 2^0, 2^1, 2^2,2^3
- Groups should be as large as possible. → start from highest power
- Every 1 must be in at least one group. → circle a single 1 if it is not adjacent
- Overlapping allowed.
- Wrap around allowed.
- fewest number of groups

Simplify:
write a product term with the differing variable omitted

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

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

what is wrong with this grouping?

K-map map simplification

Let us consider a function of **three** variables a,b,c: f(a,b,c) = a'b'c + ab'c + abc’ + abc

To simplify using K-map: we circle adjacent cell containing **1** to form groups for simplification.
But we will expand the rules that we used to group:

- size of the group must be a power of 2 → for function of 3 variable valid group sizes: 2^0, 2^1, 2^2,2^3
- Groups should be as large as possible. → start from highest power
- Every 1 must be in at least one group. → circle a single 1 if it is not adjacent
- Overlapping allowed.
- Wrap around allowed.
- fewest number of groups

Simplify:
write a product term with the differing variable omitted

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

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

how about now, why is this not correct?

K-map map simplification

Let us consider a function of **three** variables a,b,c: f(a,b,c) = ab’c’ + a’b’c’

Therefore f(a,b,c) = b’c’

- size of the group must be a power of 2 → for function of 3 variable valid group sizes: 2^0, 2^1, 2^2,2^3
- Groups should be as large as possible. → start from highest power
- Every 1 must be in at least one group. → circle a single 1 if it is not adjacent
- Overlapping allowed.
- Wrap around allowed.
- fewest number of groups

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

b’c’

Simplify:
write a product term with the differing variable omitted

K-map map simplification

Let us consider a function of **three** variables a,b,c: f(a,b,c) = a’b'c’ + a’bc’

Therefore f(a,b,c) = a’c’

- size of the group must be a power of 2 → for function of 3 variable valid group sizes: 2^0, 2^1, 2^2,2^3
- Groups should be as large as possible. → start from highest power
- Every 1 must be in at least one group. → circle a single 1 if it is not adjacent
- Overlapping allowed.
- Wrap around allowed.
- fewest number of groups

Simplify:
write a product term with the differing variable omitted

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

a’c’

K-map map simplification

Let us consider a function of **three** variables a,b,c: $f(a,b,c) = a'b'c' + ab'c' + a'bc' + abc'$

Therefore $f(a,b,c) = c'$

- size of the group must be a power of 2 → for function of 3 variable valid group sizes: 2^0, 2^1, 2^2,2^3
- Groups should be as large as possible. → start from highest power
- Every 1 must be in at least one group. → circle a single 1 if it is not adjacent
- Overlapping allowed.
- Wrap around allowed.
- fewest number of groups

Simplify:
write a product term with the differing variables omitted

c’

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

K-map map simplification

Let us consider a function of **three** variables a,b,c: $f(a,b,c) = a'b'c + a'bc + ab'c + abc$

- size of the group must be a power of 2 → for function of 3 variable valid group sizes: 2^0, 2^1, 2^2,2^3
- Groups should be as large as possible. → start from highest power
- Every 1 must be in at least one group. → circle a single 1 if it is not adjacent
- Overlapping allowed.
- Wrap around allowed.
- fewest number of groups

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

Simplify:
write a product term with the differing variables omitted

K-map map simplification

Consider a function of **three** variables a,b,c given as truth table:

a	b	c	f(a,b,c)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

simplify using K-map

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

K-map map simplification

Let us consider a function of **four** variables a,b,c,d: $y = a'b'c'd' + a'b'cd' + ab'c'd' + ab'cd'$

- size of the group must be a power of 2 → for function of 4 variable valid group sizes: 2^0, 2^1, 2^2,2^3,2^4
- Groups should be as large as possible. → start from highest power
- Every 1 must be in at least one group. → circle a single 1 if it is not adjacent
- Overlapping allowed.
- Wrap around allowed.
- fewest number of groups

ab \ cd	00	01	11	10
00	1			1
01				
11				
10	1			1

Simplify:
write a product term with the differing variables omitted

K-map map simplification

Let us consider a function of **four** variables a,b,c,d: y = a'bc'd + a'bcd + abc'd + abcd

- size of the group must be a power of 2 → for function of 4 variable valid group sizes: 2^0, 2^1, 2^2,2^3,2^4
- Groups should be as large as possible. → start from highest power
- Every 1 must be in at least one group. → circle a single 1 if it is not adjacent
- Overlapping allowed.
- Wrap around allowed.
- fewest number of groups

ab \ cd	00	01	11	10
00				
01		1	1	
11		1	1	
10				

Simplify:
write a product term with the differing variables omitted

K-map map simplification

Let's consider f(a,b,c,d) : $y = a'b'c'd + a'b'cd + a'bc'd' + a'bcd' + abc'd' + abcd' + ab'c'd + ab'cd$

- size of the group must be a power of 2 → for function of 4 variable valid group sizes: 2^0, 2^1, 2^2,2^3,2^4
- Groups should be as large as possible. → start from highest power
- Every 1 must be in at least one group. → circle a single 1 if it is not adjacent
- Overlapping allowed.
- Wrap around allowed.
- fewest number of groups

ab \ cd	00	01	11	10
00		1	1	
01	1			1
11	1			1
10		1	1	

Simplify:
write a product term with the differing variables omitted