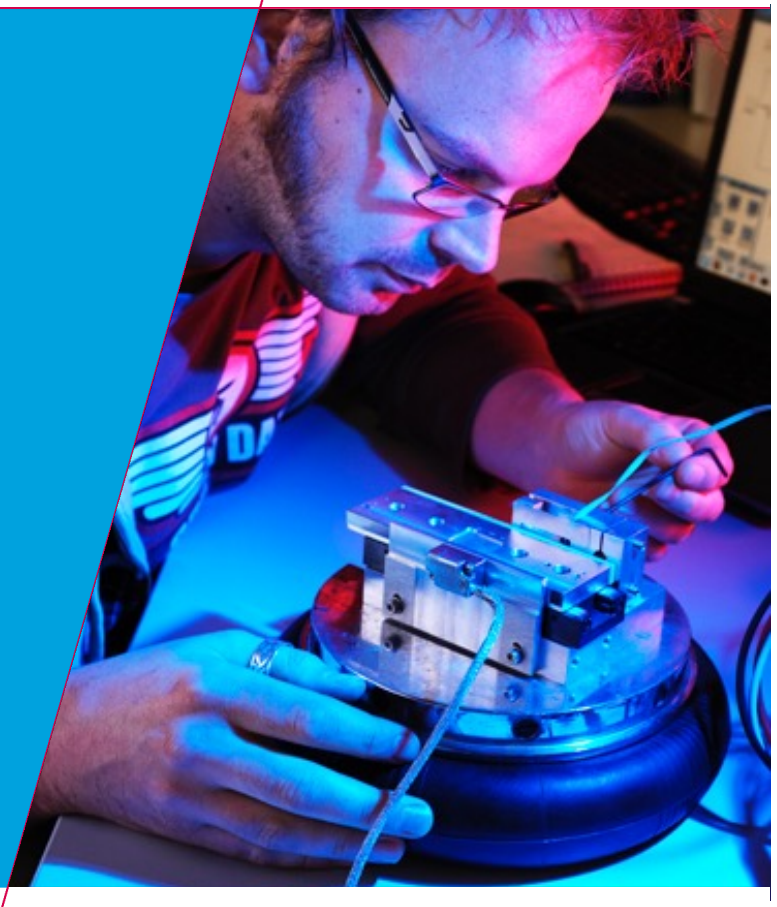
2IC30: Computer systems
Optimizing circuits
Karnaugh maps

Jan Friso Groote

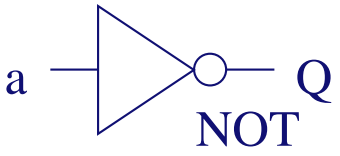


Technische Universiteit
Eindhoven
University of Technology

Where innovation starts

Logical gates I

Logical gates:

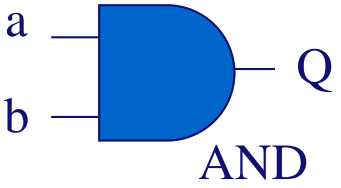


a	Q
0	1
1	0

$Q = \neg a$

$Q = a'$

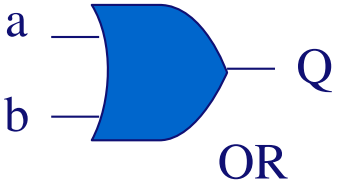
$Q = \overline{a}$



a	b	Q
0	0	0
0	1	0
1	0	0
1	1	1

$Q = a \wedge b$

$Q = a \bullet b$

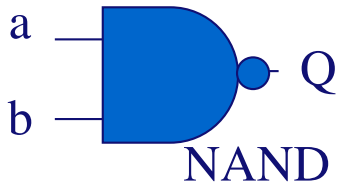


a	b	Q
0	0	0
0	1	1
1	0	1
1	1	1

$Q = a \vee b$

$Q = a + b$

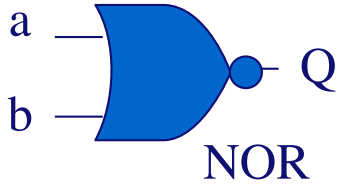
Logical gates II



a	b	Q
0	0	1
0	1	1
1	0	1
1	1	0

$Q = \neg(a \wedge b)$

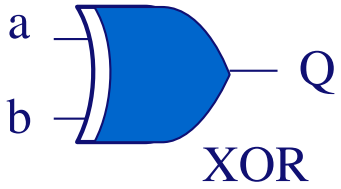
$Q = (a \bullet b)'$



a	b	Q
0	0	1
0	1	0
1	0	0
1	1	0

$Q = \neg(a \vee b)$

$Q = (a + b)'$

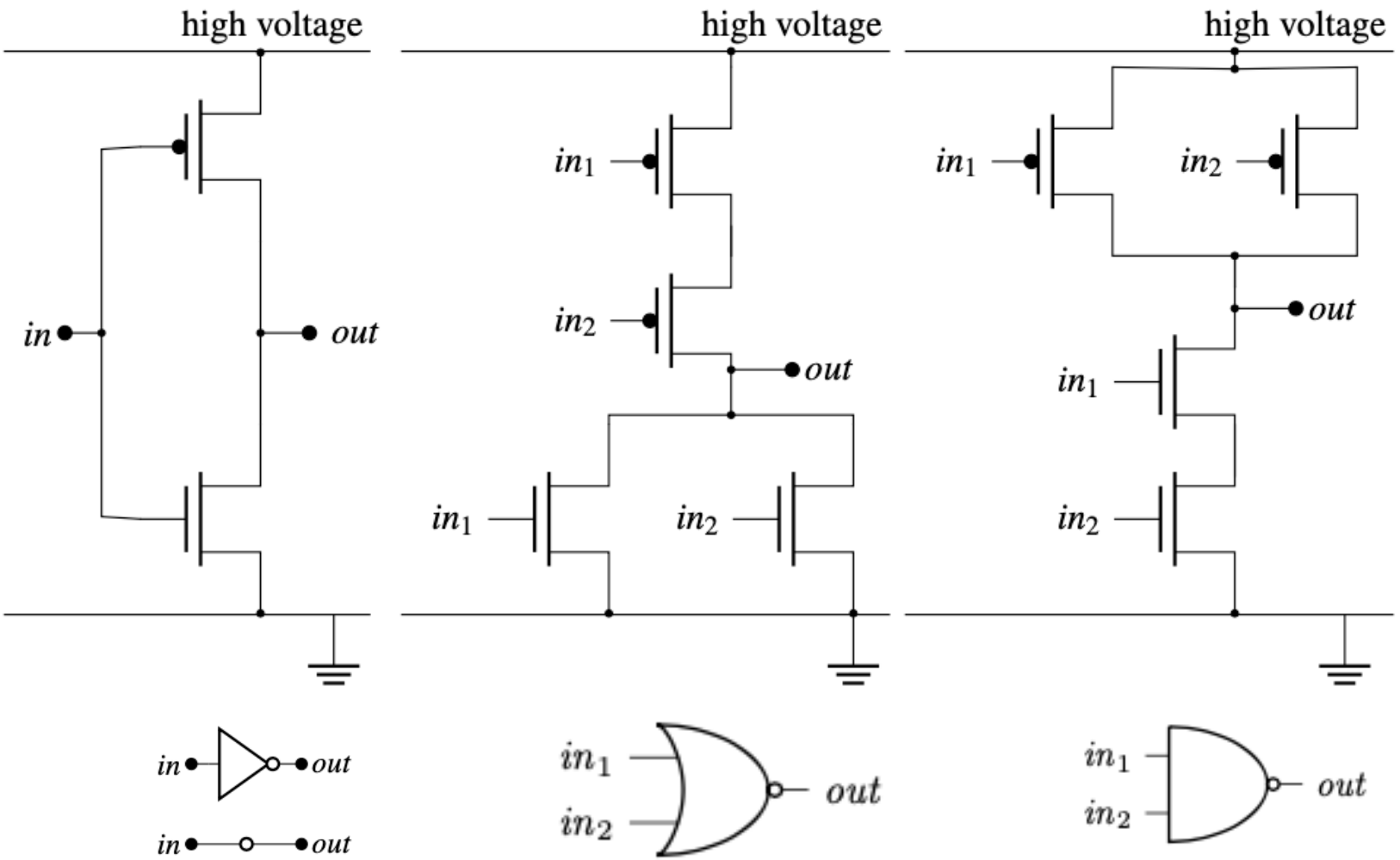


a	b	Q
0	0	0
0	1	1
1	0	1
1	1	0

$Q = a \not\leftrightarrow b$

$Q = (a \oplus b)$

Gates with Field Effect Transistors.



not gate **nor gate** **nand gate**

Duality principle

- 1 is the dual of 0 (*true* is the dual of *false*).
- $\wedge$ is the dual of $\vee$.

- The dual is obtained by applying negation. Properties:

- $\neg(x \vee y) = \neg x \wedge \neg y$
- $\neg(x \wedge y) = \neg x \vee \neg y$

- $x \vee (\neg x \wedge y) = x \vee y$
- $x \wedge (\neg x \vee y) = x \wedge y$

Rules for switching/boolean algebra (1)

- $x \vee x = x$
- $x \wedge x = x$

} Idempotence

- $x \vee 1 = 1$
- $x \wedge 0 = 0$

} Zero elements

- $x \vee 0 = x$
- $x \wedge 1 = x$

} Identity elements

- $\neg(\neg x) = x$

} Double negation

- $(x \vee y) \vee z = x \vee (y \vee z)$
- $(x \wedge y) \wedge z = x \wedge (y \wedge z)$

} Associativity

- $x \vee \neg x = 1$

} Excluded middle

- $x \wedge \neg x = 0$

} Contradiction

Red rules are complete, i.e., sufficient to derive all others (Ninomiya & Mukaidono, 2003).

Rules for switching/boolean algebra (2)

- $x \vee y = y \vee x$
- $x \wedge y = y \wedge x$

} Commutativity

- $x \vee (x \wedge y) = x$
- $x \wedge (x \vee y) = x$

} Absorption

- $x \vee (\neg x \wedge y) = x \vee y$
- $x \wedge (\neg x \vee y) = x \wedge y$

} Complement Absorption

- $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
- $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$

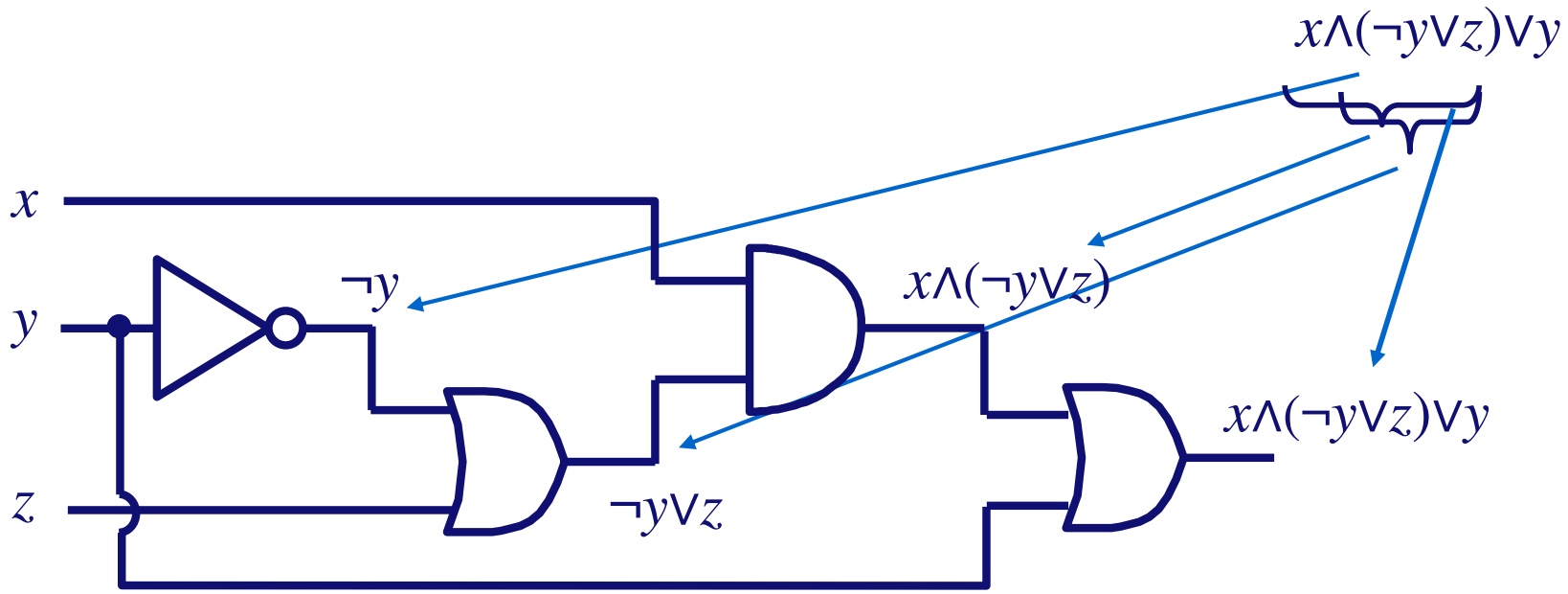
} Distributivity

- $\neg(x \vee y) = \neg x \wedge \neg y$
- $\neg(x \wedge y) = \neg x \vee \neg y$

} De Morgan

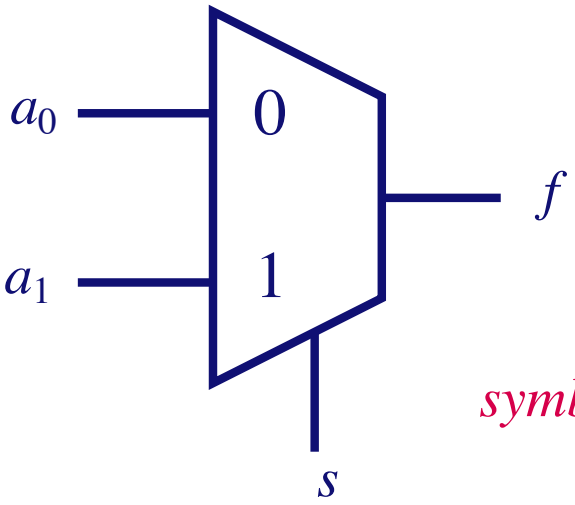
From gates to boolean expressions and v.v.

- Any switching function can be written as a boolean expression over the input signals.
- A boolean expression has a 1-1 correspondence with a gate implementation:



How to systematically design a circuit from a truth table?

2 input multiplexer.



symbol

multiplexer

s	a_0	a_1	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

2-input multiplexer

s	f
0	a_0
1	a_1

Minterms and maxterms

- Every line in the truth table where the function has value 1 corresponds to a (so-called) *minterm*;

Example:
if f =1 for a = 1, b = 0, and c = 1, the minterm is:
$a \wedge \neg b \wedge c$.

a	b	c	f	f '
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

Minterms and maxterms

- Every line in the truth table where the function has value 0 corresponds to a (so-called) *maxterm*;

Example:
if $f = 0$ for $a = 0, b = 1$, and $c = 0$, the maxterm is:
$a \vee \neg b \vee c$.

a	b	c	f	f '
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

Two Canonical Forms

- *Disjunctive Normal Form*: every boolean function can be written as the *disjunction* of a set of *minterms*.

- *Conjunctive Normal Form*: every boolean function can be written as the *conjunction* of a set of *maxterms*.

Disjunction of minterms (canonical)

$f = \neg a \wedge b \wedge c \vee a \wedge \neg b \wedge \neg c \vee a \wedge \neg b \wedge c \vee a \wedge b \wedge \neg c \vee a \wedge b \wedge c$

a	b	c	f	f′
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

$f' = \neg a \wedge \neg b \wedge \neg c \vee \neg a \wedge \neg b \wedge c \vee \neg a \wedge b \wedge \neg c$

Truth tables and disjunctions of minterms (in fixed order) are canonical representations

Unfortunately very verbose!

Rules for switching algebra (2)

- $x \vee y = y \vee x$
- $x \wedge y = y \wedge x$

} Commutativity

- $x \vee (x \wedge y) = x$
- $x \wedge (x \vee y) = x$

} Absorption

- $x \vee (\neg x \wedge y) = x \vee y$
- $x \wedge (\neg x \vee y) = x \wedge y$

} Complement Absorption

- $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
- $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$

} **Distributivity**

- $\neg(x \vee y) = \neg x \wedge \neg y$
- $\neg(x \wedge y) = \neg x \vee \neg y$

} De Morgan

Disjunction of minterms (canonical)

$f = \neg a \wedge b \wedge c \vee a \wedge \neg b \wedge \neg c \vee a \wedge \neg b \wedge c \vee a \wedge b \wedge \neg c \vee a \wedge b \wedge c$

a	b	c	f	f′
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

$f' = \neg a \wedge \neg b \wedge \neg c \vee \neg a \wedge \neg b \wedge c \vee \neg a \wedge b \wedge \neg c$

$f = a \wedge \neg b \wedge (c \vee \neg c) \vee \neg a \wedge b \wedge c \vee a \wedge b \wedge (\neg c \vee c)$

$= a \wedge \neg b \vee \neg a \wedge b \wedge c \vee a \wedge b$

$= a \wedge (\neg b \vee b) \vee \neg a \wedge b \wedge c$

$= a \vee \neg a \wedge b \wedge c$

$= a \vee b \wedge c$

Truth tables and disjunctions of minterms (in fixed order) are canonical representations

Unfortunately very verbose!

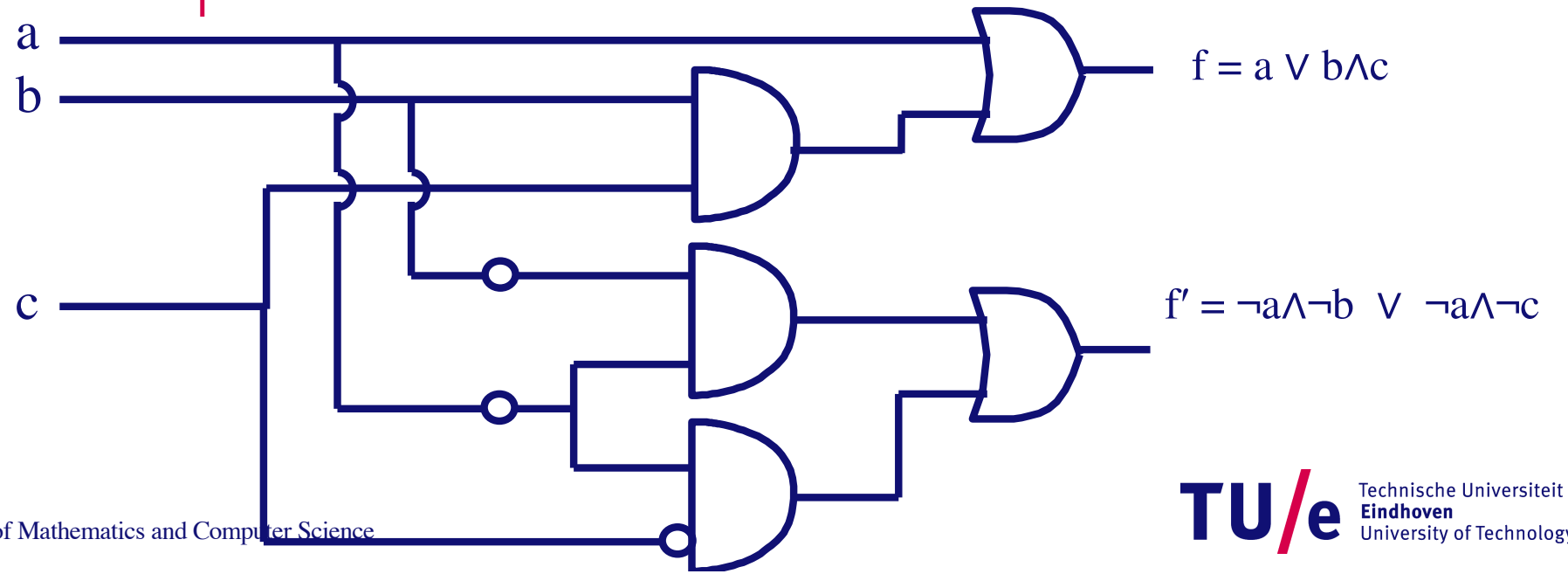
CHECK: $f' = \neg(a \vee b \wedge c) = \neg a \wedge (\neg b \vee \neg c) =$
$\neg a \wedge (\neg b \vee b \wedge \neg c) = \neg a \wedge \neg b \vee \neg a \wedge b \wedge \neg c =$
$\neg a \wedge \neg b \wedge (c \vee \neg c) \vee \neg a \wedge b \wedge \neg c =$
$\neg a \wedge \neg b \wedge c \vee \neg a \wedge \neg b \wedge \neg c \vee \neg a \wedge b \wedge \neg c$

Disjunction of minterms

a	b	c	f	f′
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	1	0

$f = a \vee b \wedge c$

$f' = \neg a \wedge \neg b \vee \neg a \wedge \neg c$



Conjunction of maxterms (canonical)

a	b	c	Maxterms	f	f′
0	0	0	$a \vee b \vee c = M_0$	0	1
0	0	1	$a \vee b \vee \neg c = M_1$	0	1
0	1	0	$a \vee \neg b \vee c = M_2$	0	1
0	1	1	$a \vee \neg b \vee \neg c = M_3$	1	0
1	0	0	$\neg a \vee b \vee c = M_4$	1	0
1	0	1	$\neg a \vee b \vee \neg c = M_5$	1	0
1	1	0	$\neg a \vee \neg b \vee c = M_6$	1	0
1	1	1	$\neg a \vee \neg b \vee \neg c = M_7$	1	0

Maxterm:

Disjunction of signals in which every variable or its complement occurs exactly once but not both!

Maxterm form for function f:

Lines with f = 0

Input 0 **: take variable**

Input 1 **: take complement of input**

Conjunction of maxterms

$f = (a \vee b \vee c) \wedge (a \vee b \vee \neg c) \wedge (a \vee \neg b \vee c)$

$f' = (a \vee \neg b \vee \neg c) \wedge (\neg a \vee b \vee c) \wedge (\neg a \vee b \vee \neg c) \wedge (\neg a \vee \neg b \vee c) \wedge (\neg a \vee \neg b \vee \neg c)$

Rules for switching algebra (2)

- $x \vee y = y \vee x$
- $x \wedge y = y \wedge x$

} Commutativity

- $x \vee (x \wedge y) = x$
- $x \wedge (x \vee y) = x$

} Absorption

- $x \vee (\neg x \wedge y) = x \vee y$
- $x \wedge (\neg x \vee y) = x \wedge y$

} Complement Absorption

- $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
- $x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$

} Distributivity

- $\neg(x \vee y) = \neg x \wedge \neg y$
- $\neg(x \wedge y) = \neg x \vee \neg y$

} **De Morgan**

Two level canonical forms and de Morgan

Disjunction of minterms of f' to the conjunction of maxterms of f

$$f' = \neg a \wedge \neg b \wedge \neg c \vee \neg a \wedge \neg b \wedge c \vee \neg a \wedge b \wedge \neg c$$

Using de Morgan to get f:

$$f = \neg(f') = \neg(\neg a \wedge \neg b \wedge \neg c \vee \neg a \wedge \neg b \wedge c \vee \neg a \wedge b \wedge \neg c)$$

$$= (a \vee b \vee c) \wedge (a \vee b \vee \neg c) \wedge (a \vee \neg b \vee c)$$

Conjunction of maxterms of f' to disjunction of minterms of f

$$f' = (a \vee \neg b \vee \neg c) \wedge (\neg a \vee b \vee c) \wedge (\neg a \vee b \vee \neg c) \wedge (\neg a \vee \neg b \vee c) \wedge (\neg a \vee \neg b \vee \neg c)$$

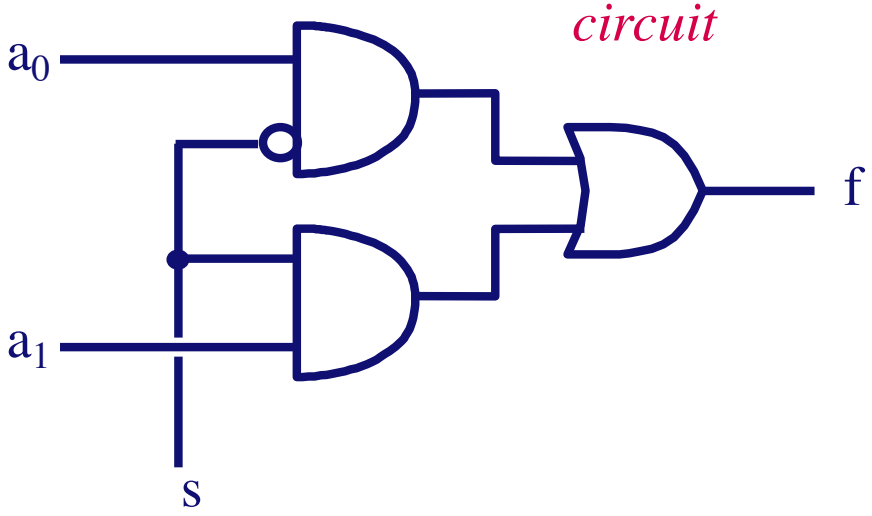
Using de Morgan to get f:

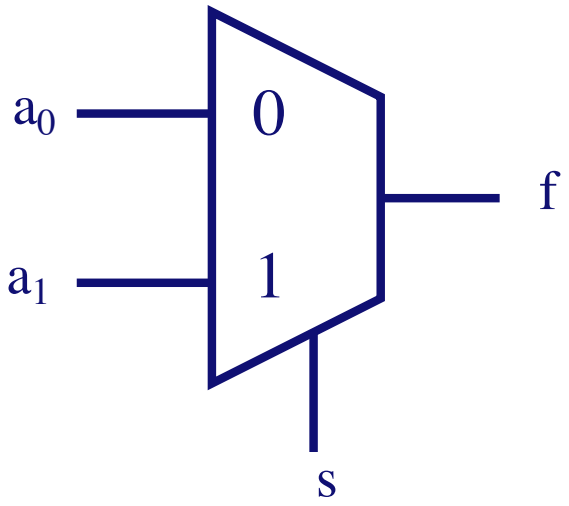
$$f = \neg(f') = \neg((a \vee \neg b \vee \neg c) \wedge (\neg a \vee b \vee c) \wedge (\neg a \vee b \vee \neg c) \wedge (\neg a \vee \neg b \vee c) \wedge (\neg a \vee \neg b \vee \neg c))$$

$$= \neg a \wedge b \wedge c \vee a \wedge \neg b \wedge \neg c \vee a \wedge \neg b \wedge c \vee a \wedge b \wedge \neg c \vee a \wedge b \wedge c$$

How to design a circuit from a truth table?

circuit





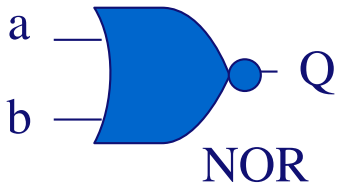
multiplexer

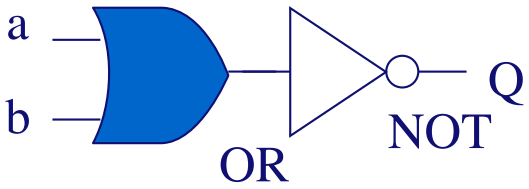
s	a_0	a_1	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Questions?

Functional completeness of a set of gates.

- Important question:
  - Which gates do we need to implement ALL possible switching functions?
  - Observe: the NOR is not necessary

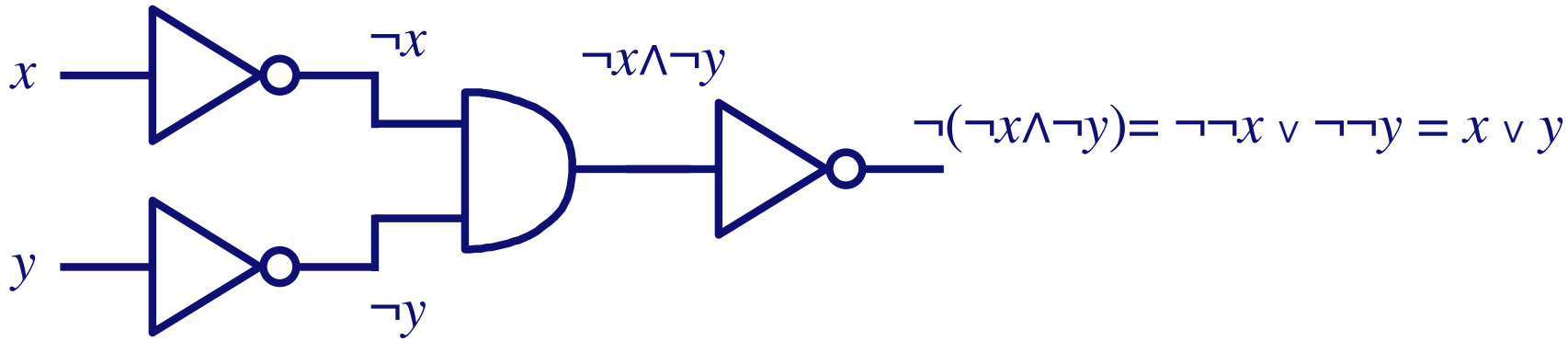




Completeness of a set of gates

- A set of gates is “complete” iff any switching function can be implemented by them.
- Of course { AND, OR, NOT } is functionally complete (consider min- and maxterm implementations).

However:

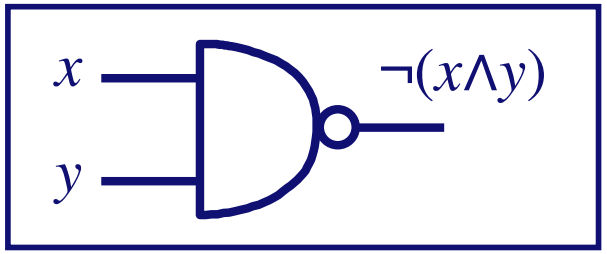


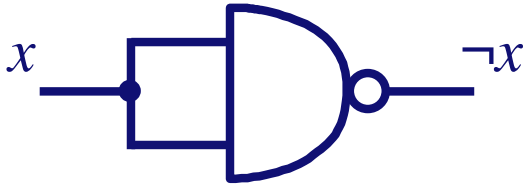
- Hence, one can build OR using only AND and NOT.

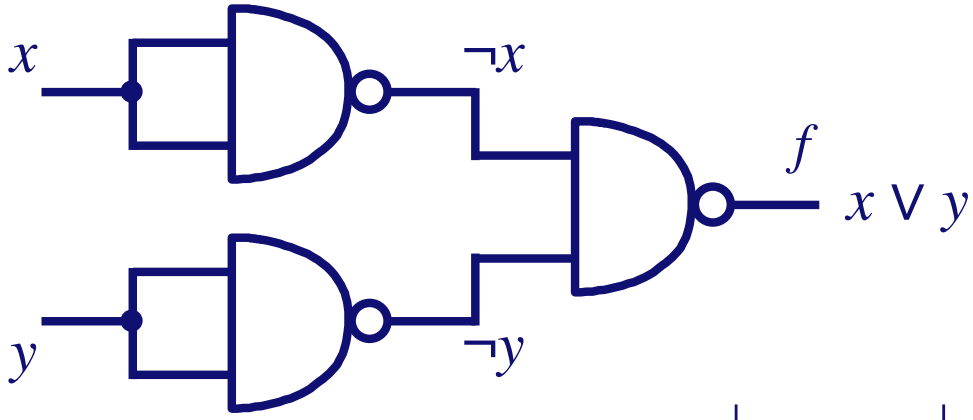
{ AND, NOT } is functionally complete
(idem: { OR, NOT } is functionally complete).

Completeness of sets of gates

A NAND gate output is 0 if and only if all inputs are 1:







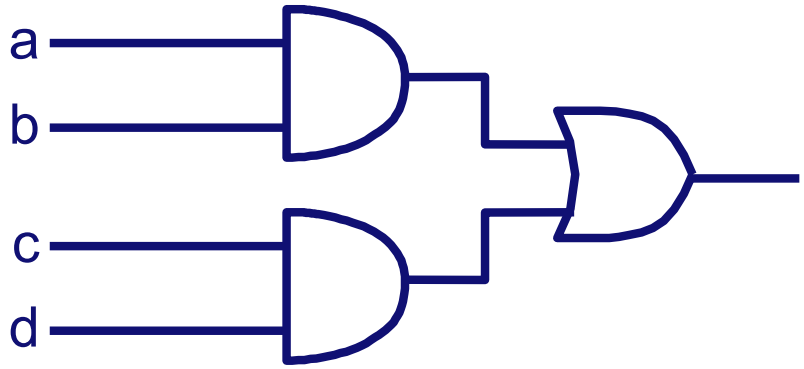
x	y	$\neg x$	$\neg y$	f
0	0	1	1	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	1

The NOT, OR and AND can be built with NAND gates!

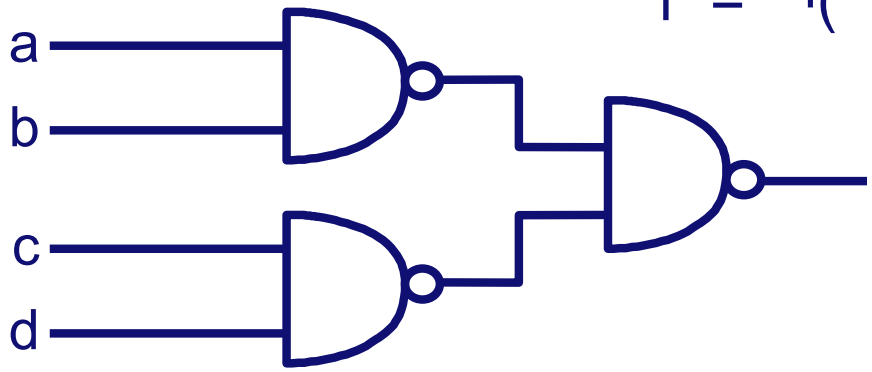
{ NAND } is functionally complete (idem: { NOR } is functionally complete).

Two-level AND-OR equals NAND-NAND

$$f = a \wedge b \vee c \wedge d$$



$$f = \neg(\neg(a \wedge b) \wedge \neg(c \wedge d))$$



Questions?

Karnaugh maps: optimising circuits.

- Each cell corresponds to one row from the truth table.
- The upper line and left column are “neighbors” from the bottom line and right column respectively.
- Two neighbors differ on exactly one input bit.

a	b	c	d	f
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0

cd \ ab	00	01	11	10
00	0	0	0	0
01	0	1	0	0
11	1	1	1	1
10	1	0	0	1

Karnaugh maps: optimising circuits.

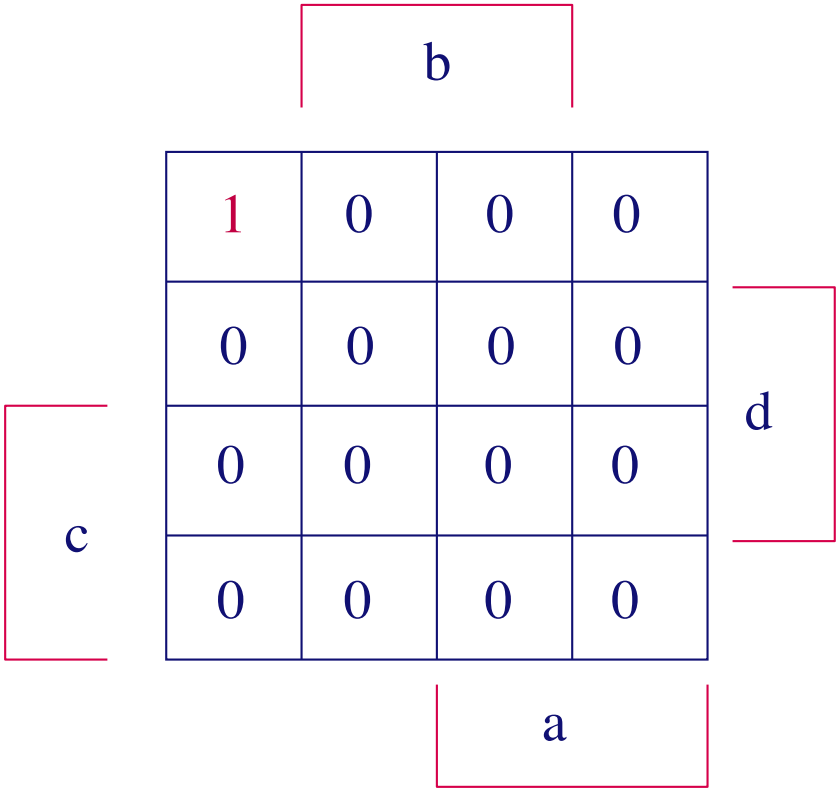
a	b	c	d	f
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

cd \ ab	00	01	11	10
00	0	0	0	0
01	0	1	0	0
11	1	1	1	1
10	1	0	0	1

Karnaugh maps

Minterms in Karnaugh maps

cd \ ab	00	01	11	10
00	1	0	0	0
01	0	0	0	0
11	0	0	0	0
10	0	0	0	0

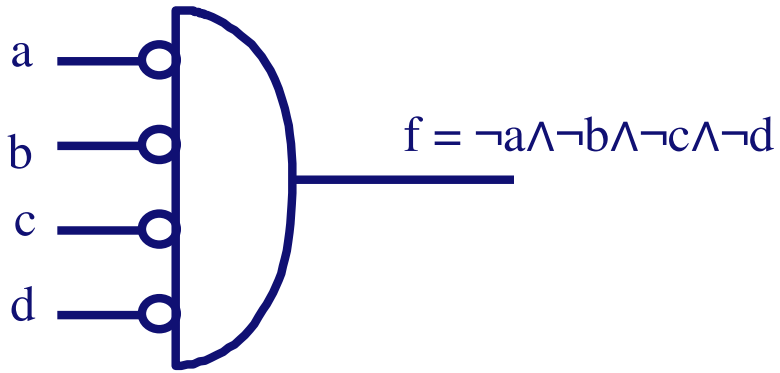


minterm = ¬a∧¬b∧¬c∧¬d
(that is where output is 1)

Karnaugh maps

Minterms in Karnaugh maps

cd \ ab	00	01	11	10
00	1	0	0	0
01	0	0	0	0
11	0	0	0	0
10	0	0	0	0

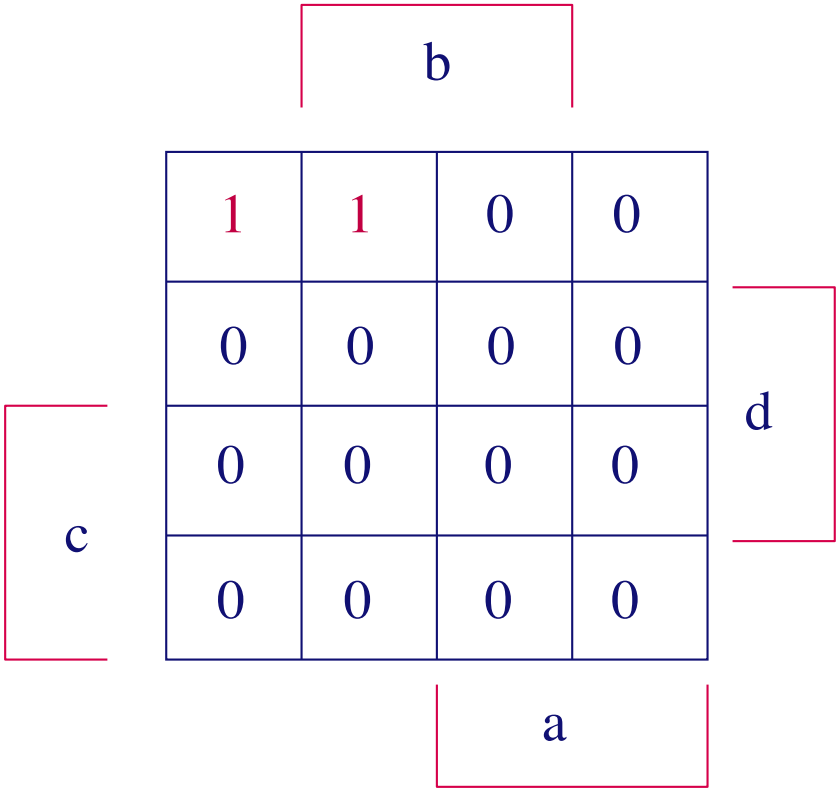


minterm = ¬a∧¬b∧¬c∧¬d
(that is where output is 1)

Karnaugh maps

Minterms and maxterms in Karnaugh maps

cd \ ab	00	01	11	10
00	1	1	0	0
01	0	0	0	0
11	0	0	0	0
10	0	0	0	0

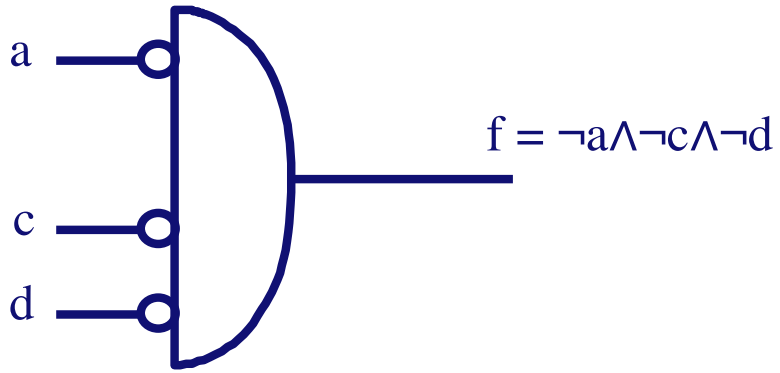


minterm = ¬a∧¬c∧¬d
(that is where output equals 1)

Karnaugh maps

Minterms and maxterms in Karnaugh maps

cd \ ab	00	01	11	10
00	1	1	0	0
01	0	0	0	0
11	0	0	0	0
10	0	0	0	0

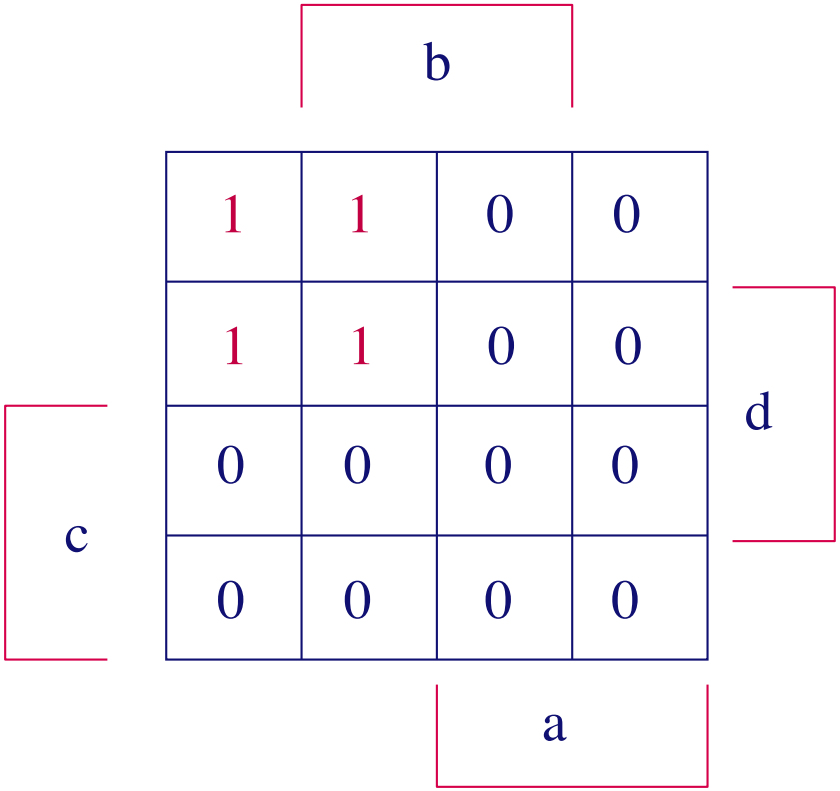


minterm = ¬a∧¬c∧¬d
(that is where output equals 1)

Karnaugh maps

Minterms and maxterms in Karnaugh maps

cd \ ab	00	01	11	10
00	1	1	0	0
01	1	1	0	0
11	0	0	0	0
10	0	0	0	0

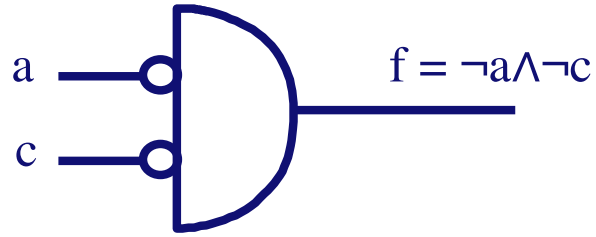


minterm = ¬a∧¬c
(that is where output equals 1)

Karnaugh maps

Minterms and maxterms in Karnaugh maps

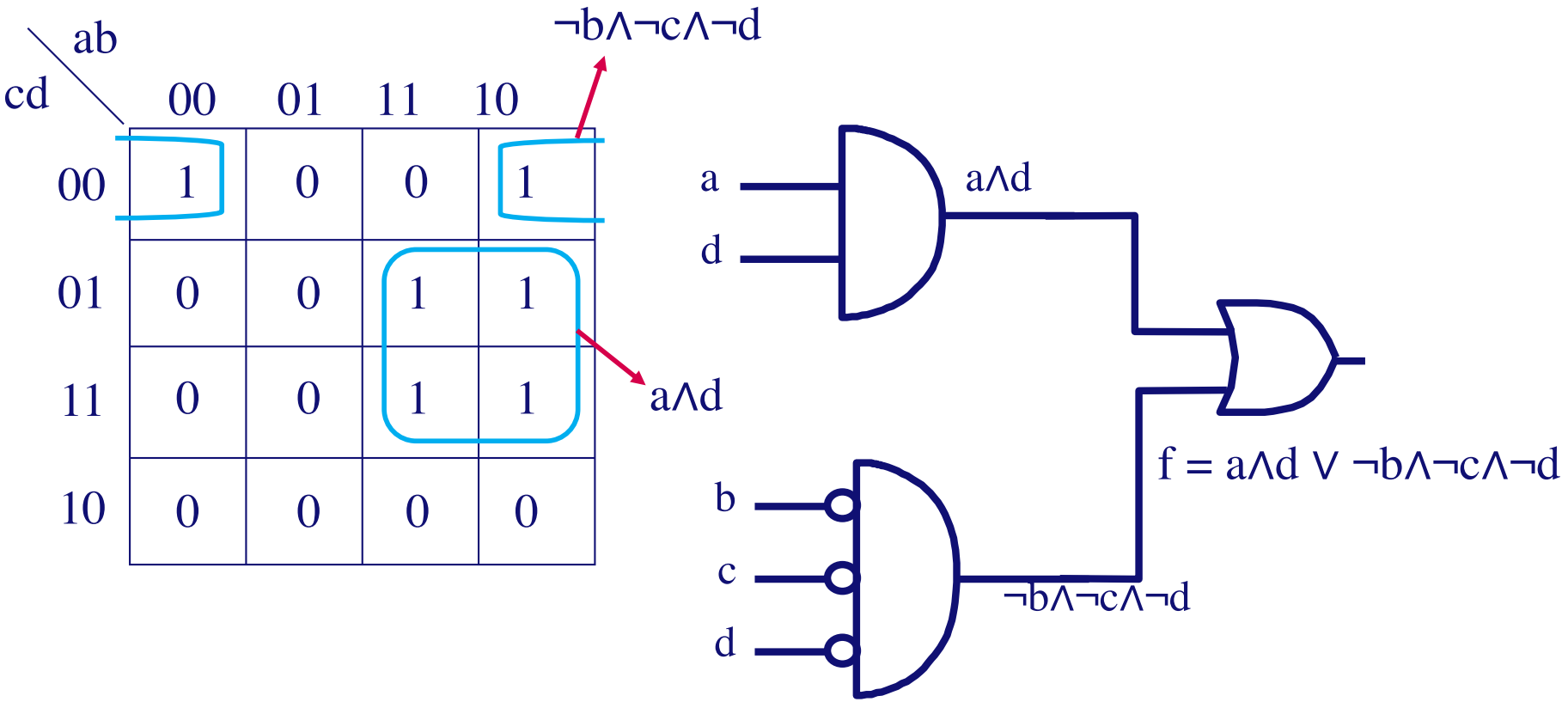
cd \ ab	00	01	11	10
00	1	1	0	0
01	1	1	0	0
11	0	0	0	0
10	0	0	0	0



minterm = ¬a∧¬c
(that is where output equals 1)

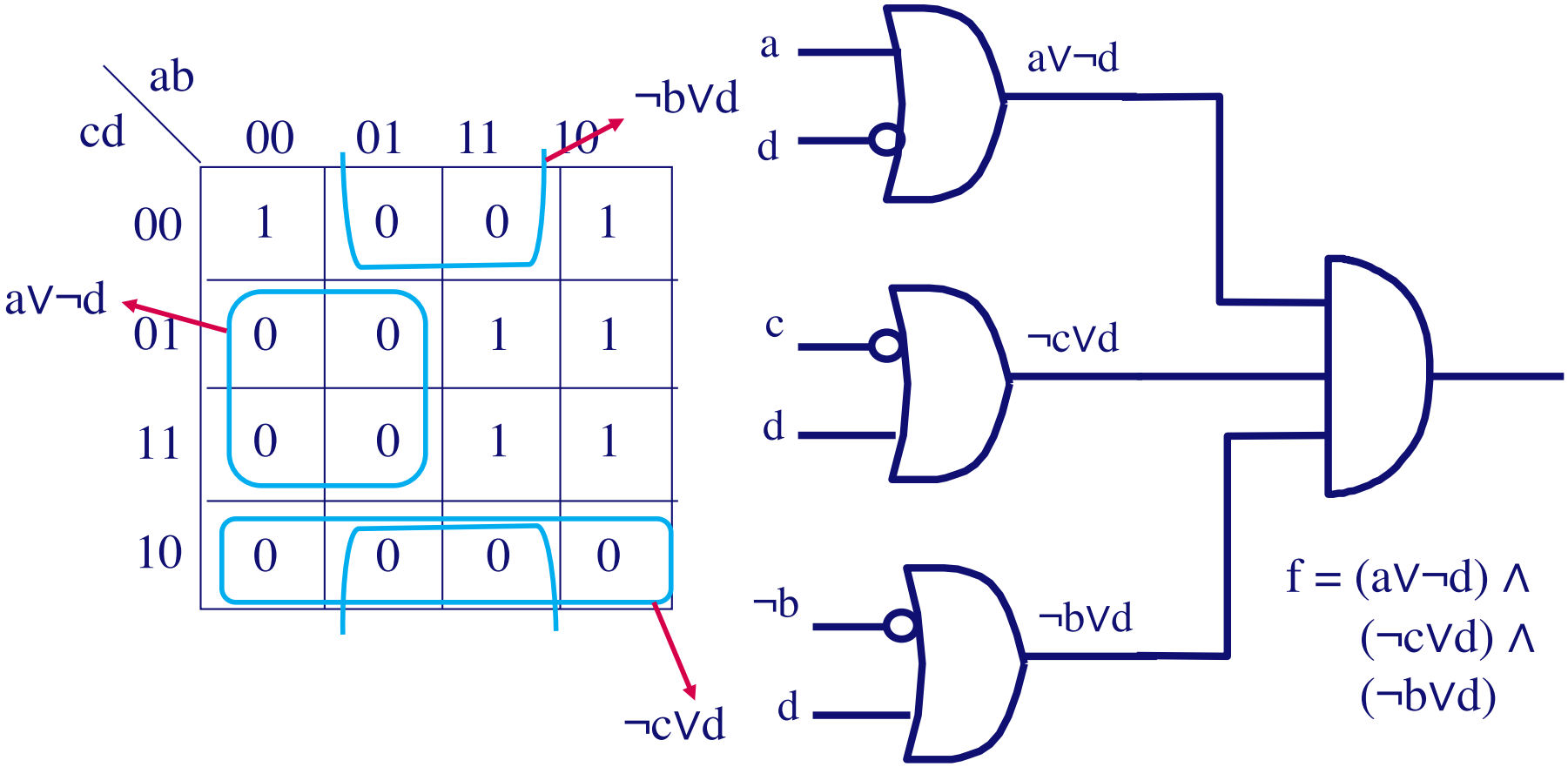
Minimal 2 level expressions

In a Karnaugh map, group rectangular areas of size $2^N \times 2^M$:



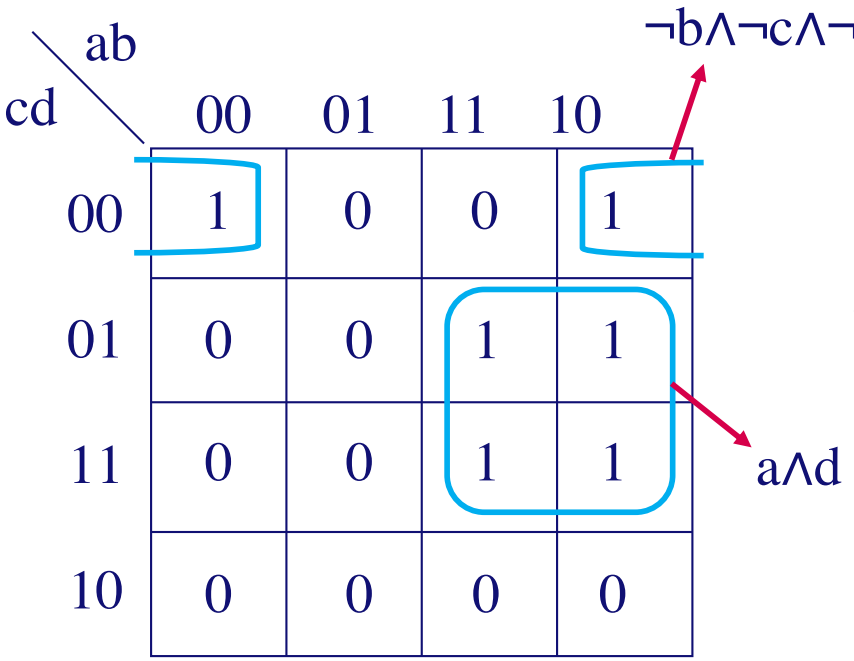
Karnaugh maps with maxterms.

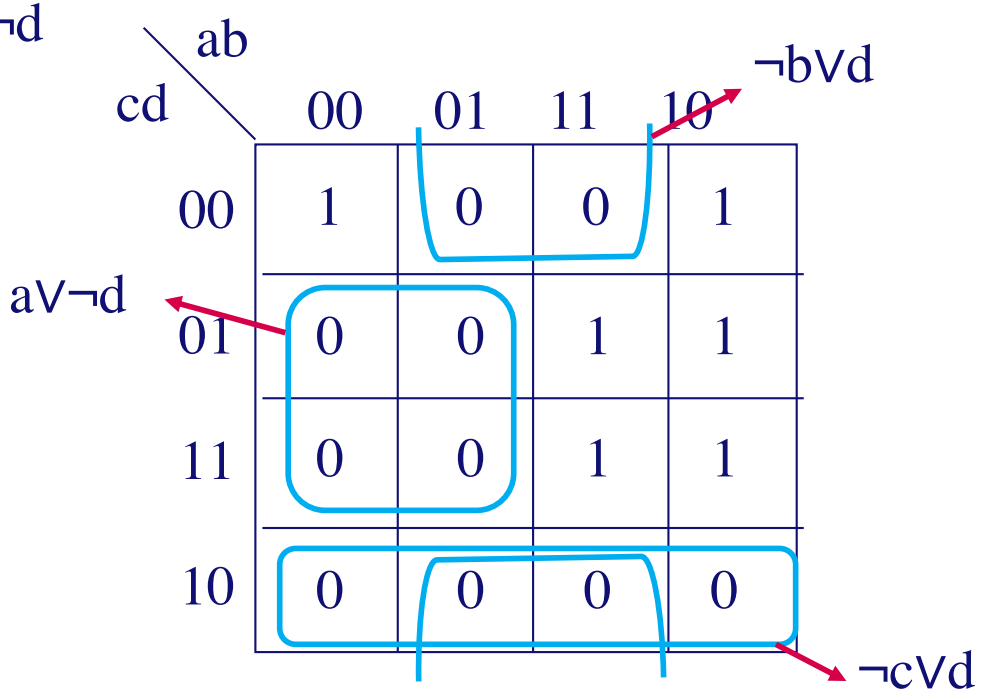
Dual approach: using maxterms we indicate where the output must be 0.



Minimal 2 level expressions

In a Karnaugh map, group rectangular areas of maximal size $2^N \times 2^M$:





$f = a \wedge d \vee \neg b \wedge \neg c \wedge \neg d$

minimal disjunction of minterms

$f = (a \vee \neg d) \wedge (\neg c \vee d) \wedge (\neg b \vee d)$

minimal conjunction of maxterms

Everybody makes mistakes. Double checking is a good idea!!!!

‘‘Don’t cares’’

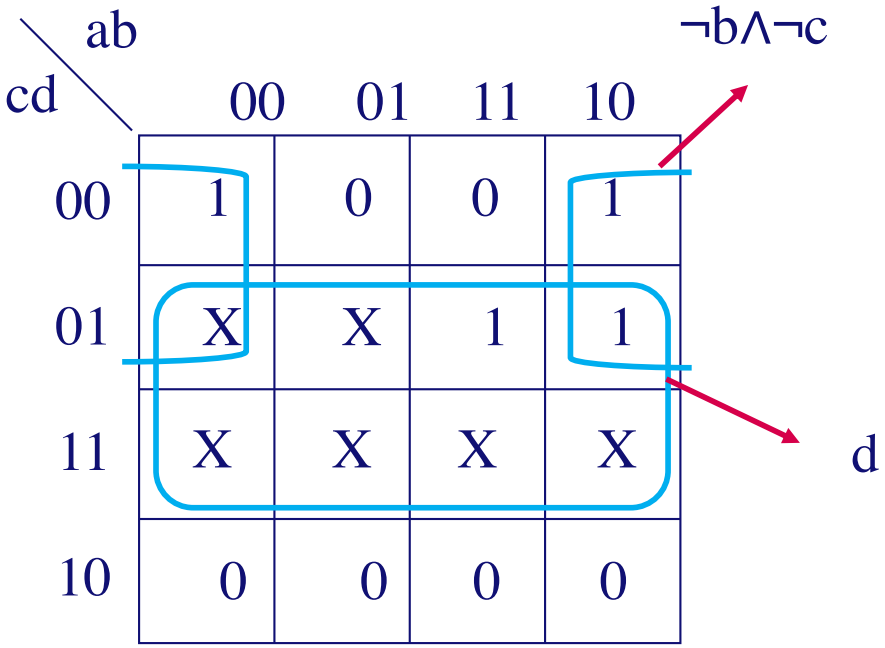
‘‘Don’t cares’’ are function values that are irrelevant:
they may be chosen by the circuit designer

a	b	c	d	f
0	0	0	0	1
0	0	0	1	X
0	0	1	0	0
0	0	1	1	X
0	1	0	0	0
0	1	0	1	X
0	1	1	0	0
0	1	1	1	X
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	X
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	X

cd \ ab	00	01	11	10
00	1	0	0	1
01	X	X	1	1
11	X	X	X	X
10	0	0	0	0

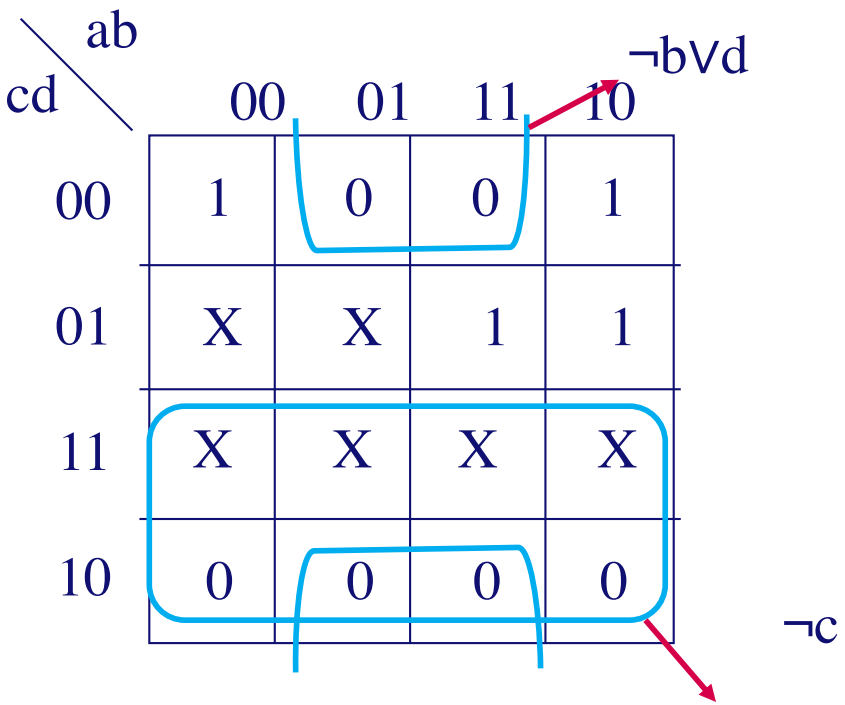
Karnaugh maps with ‘‘don’t cares’’

‘‘Don’t cares’’ are function values that are irrelevant:
they may be chosen by the circuit designer



$f = d \vee \neg b \wedge \neg c$

minimal disjunction of minterms

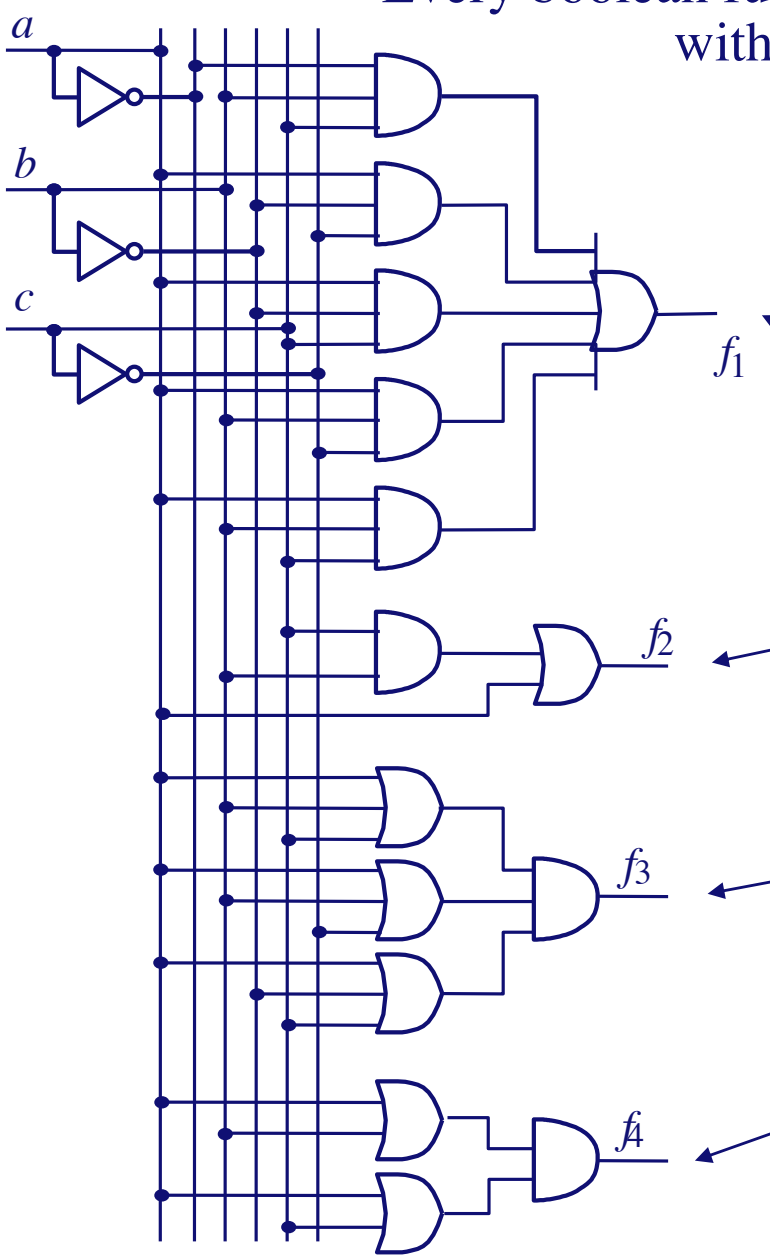


$f = \neg c \wedge (\neg b \vee d)$

minimal conjunction of maxterms

Two level minimisation

Every boolean function can be reduced to a two-level form with a minimum number of terms



Four two level implementations of

$f_1 = \neg a \wedge b \wedge c \vee a \wedge \neg b \wedge \neg c \vee a \wedge \neg b \wedge c \vee a \wedge b \wedge \neg c \vee a \wedge b \wedge c$

1) canonical disjunction of minterms

$f_2 = a \vee b \wedge c$

2) minimal disjunction of minterms

$f_3 = (a \vee b \vee c) \wedge (a \vee b \vee \neg c) \wedge (a \vee \neg b \vee c)$

3) canonical conjunction of maxterms

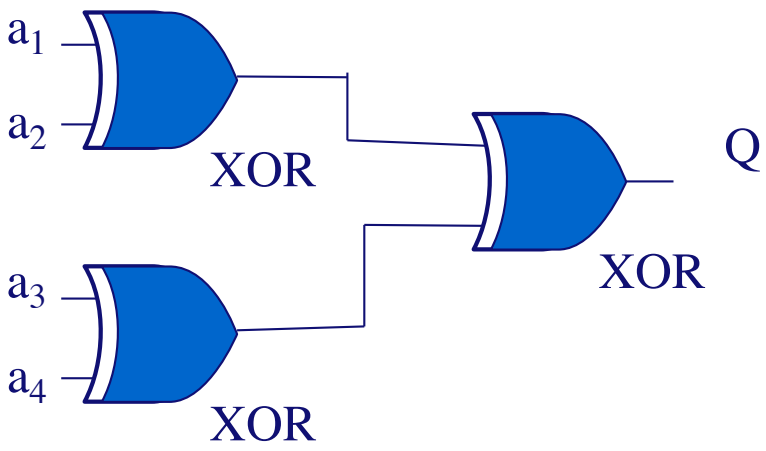
$f_4 = (a \vee b) \wedge (b \vee c)$

4) minimal conjunction of maxterms

A circuit to determine the parity.

The parity of the input is 1 if the number of inputs that is 1 is odd.

a_1	a_2	a_3	a_4	f
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

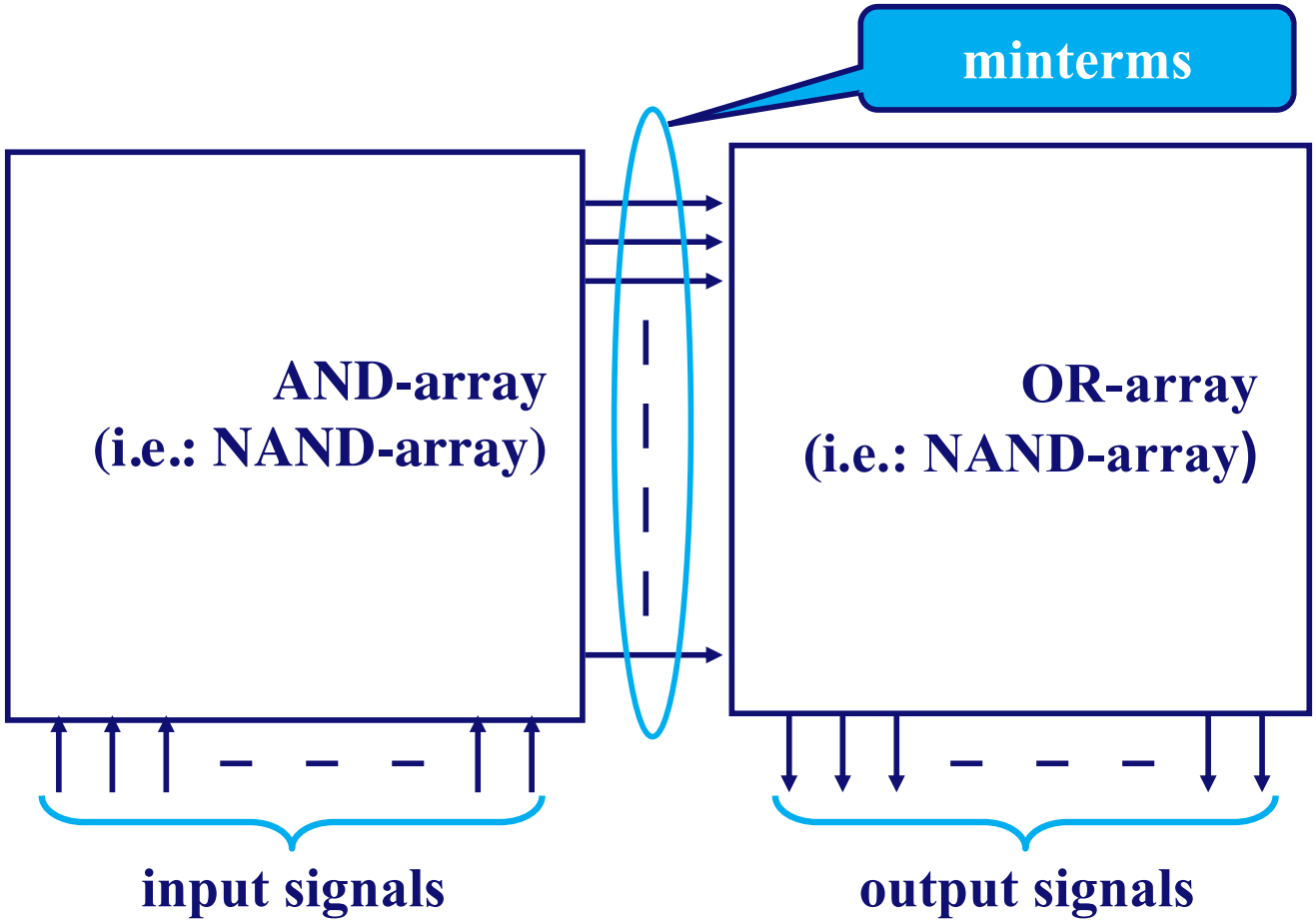


For inputs $a_1,\ldots,a_n$ there are n-1 XOR gates with a depth of log(n) rounded up.

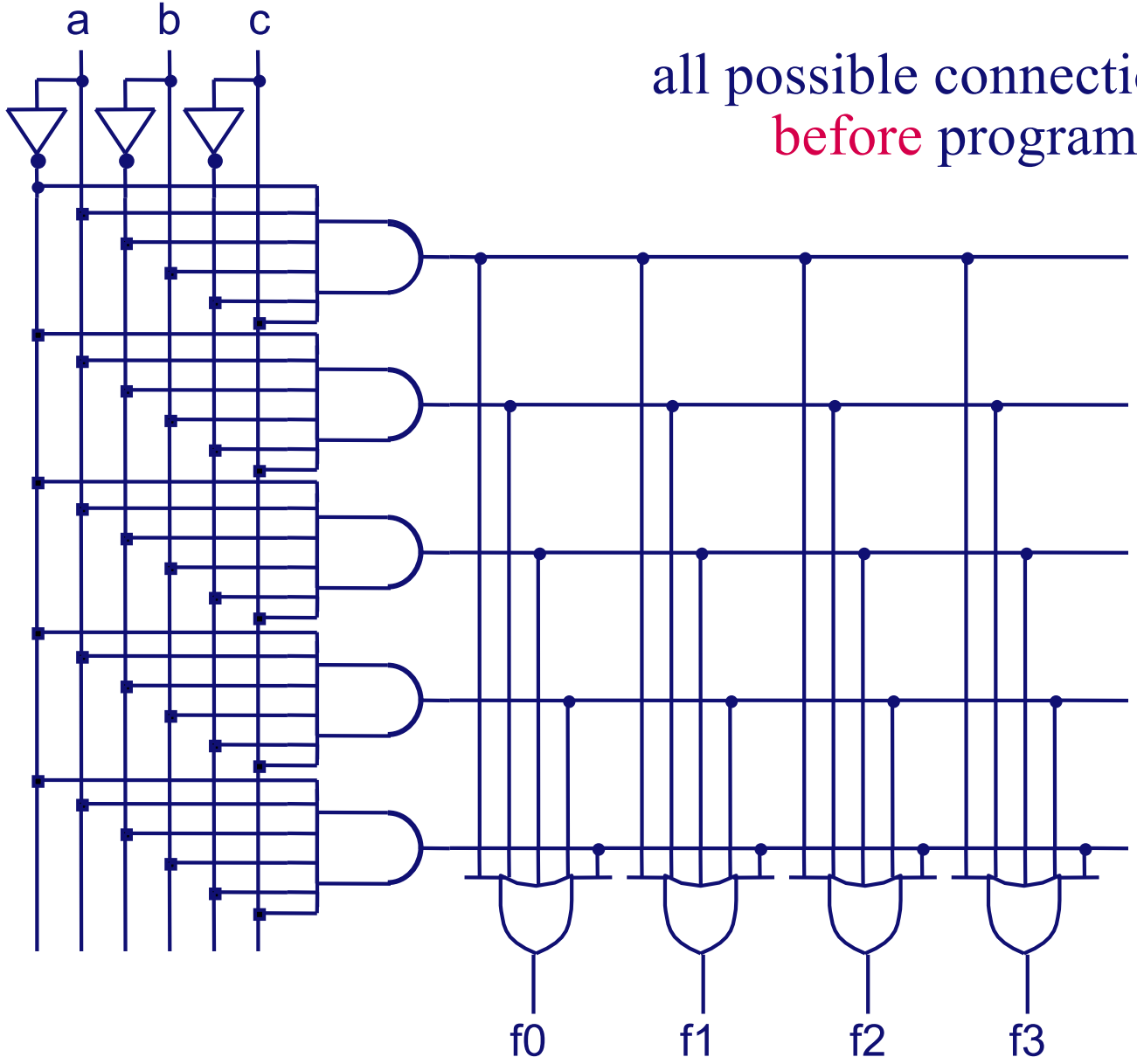
For inputs $a_1,\ldots,a_n$ a two layer solution needs an exponential number of gates.

Questions?

Programmable Logic Array (PLA)

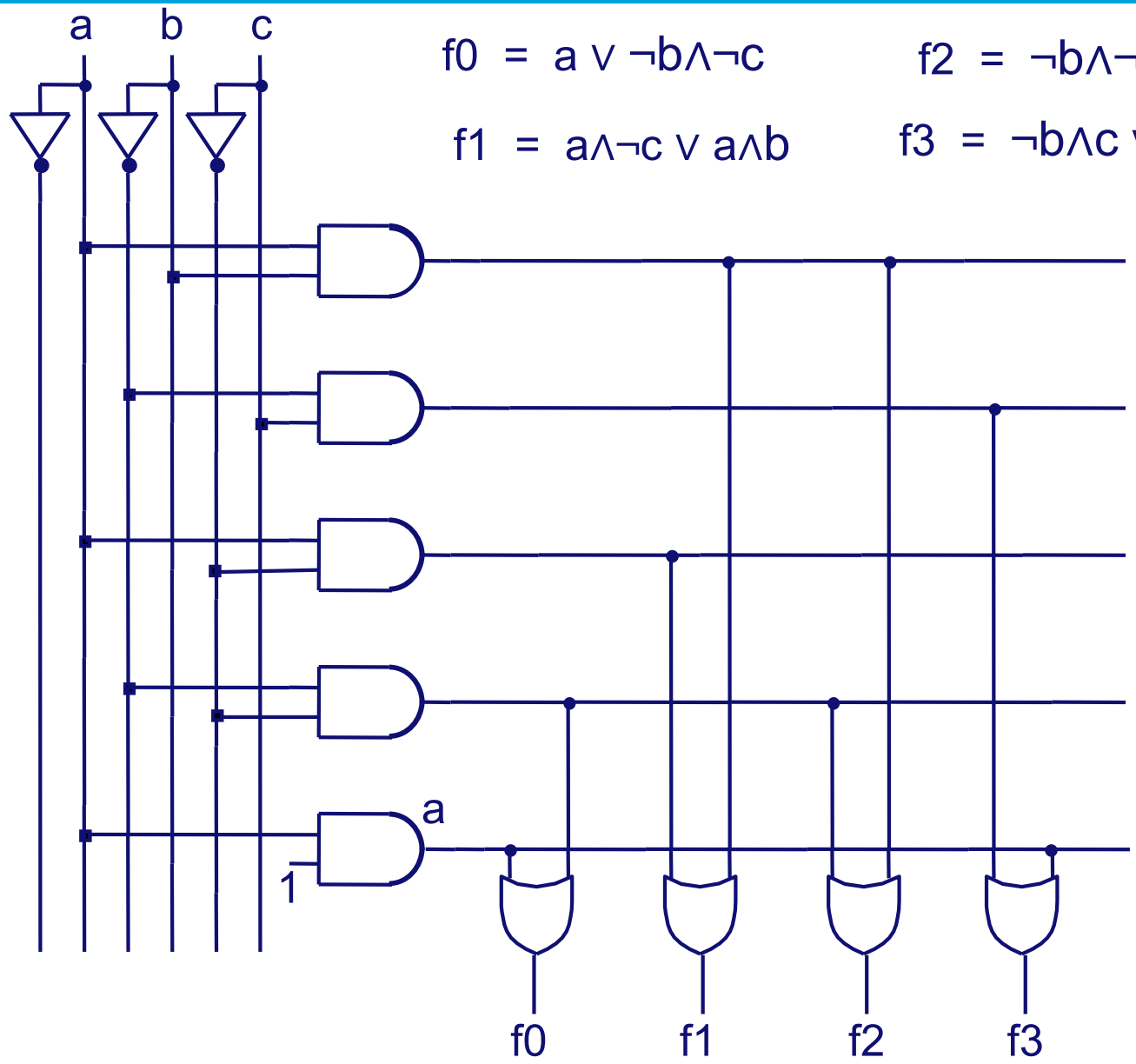


Programmable Logic Array (PLA)



all possible connections exists
before programming

Programmed Logic Array: connections removed



Summary?

Summary

What did you learn:

- Maxterms, minterms, conjunctive and disjunctive normal form.
- How to translate a circuit to a formula and truth table, and vice versa.
- How to design a minimal circuit; use Karnaugh diagrams and boolean algebra.
- Know that there are several ways to optimize circuits. There is not always a single good way.