(Two-level) Logic Synthesis
PLAs and Two-level Logic Synthesis

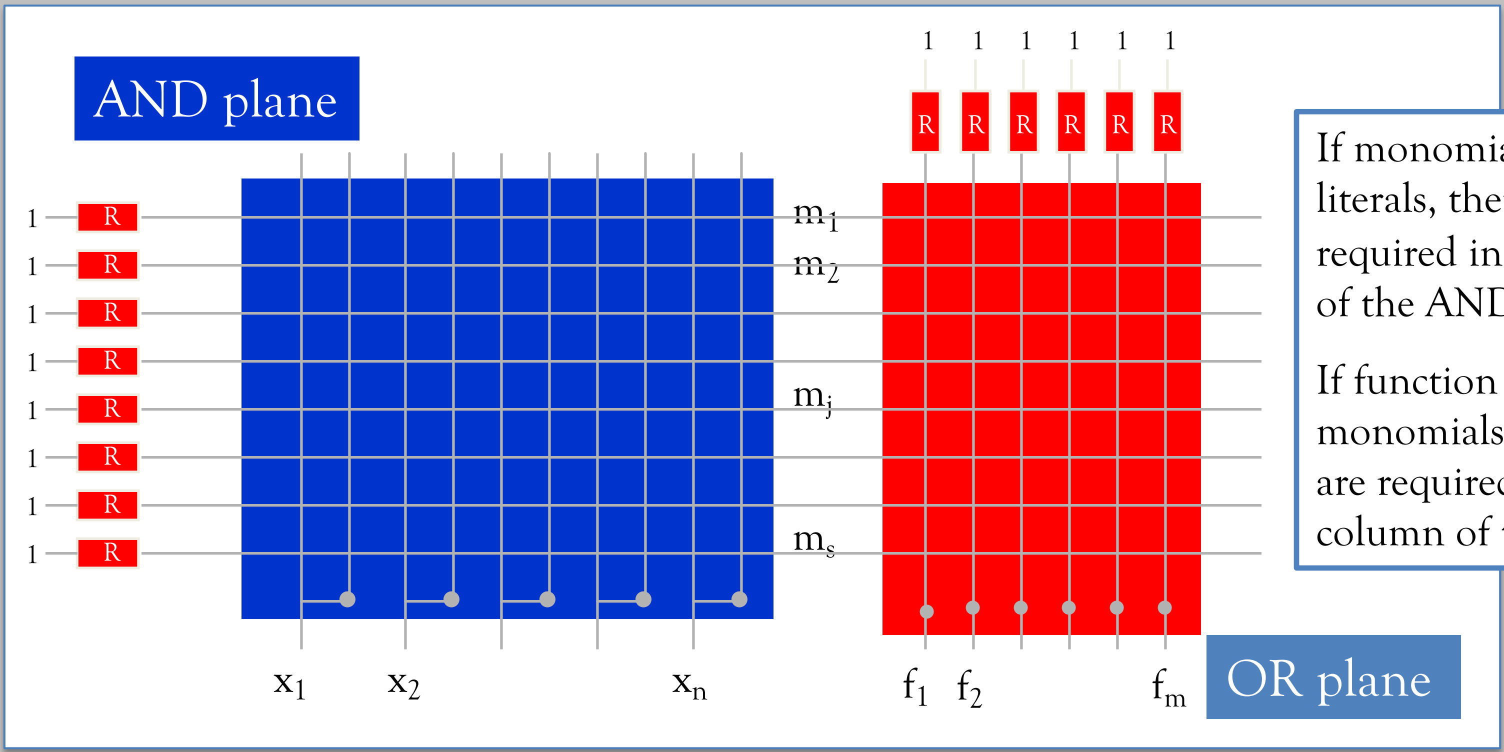
Becker/Molitor, Chapter 7.1

Jan Reineke
Universität des Saarlandes

Programmable Logic Arrays (PLA)

Special two-level circuits to implement

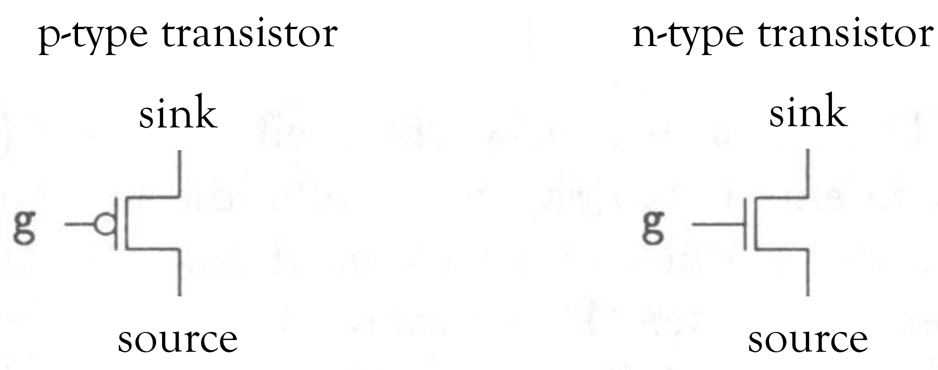
Boolean Polynomials $f_i = m_{i1} + m_{i2} + \ldots + m_{ik}$ with m_{iq} from $\{m_1, \ldots, m_s\}$



If monomial m_j consists of k literals, then k transistors are required in the respective row of the AND plane.

If function f_t consists of p monomials, then p transistors are required in the respective column of the OR plane.

Short excursion: Transistors



- A transistor can be seen as a voltage-controlled switch:
  - Gate g controls the conductivity between source and sink
- **n-type transistor**:
  - transmits, if gate is 1
  - disconnects, if gate is 0
- **p-type transistor**:
  - transmits, if gate is 0
  - disconnects, if gate is 1

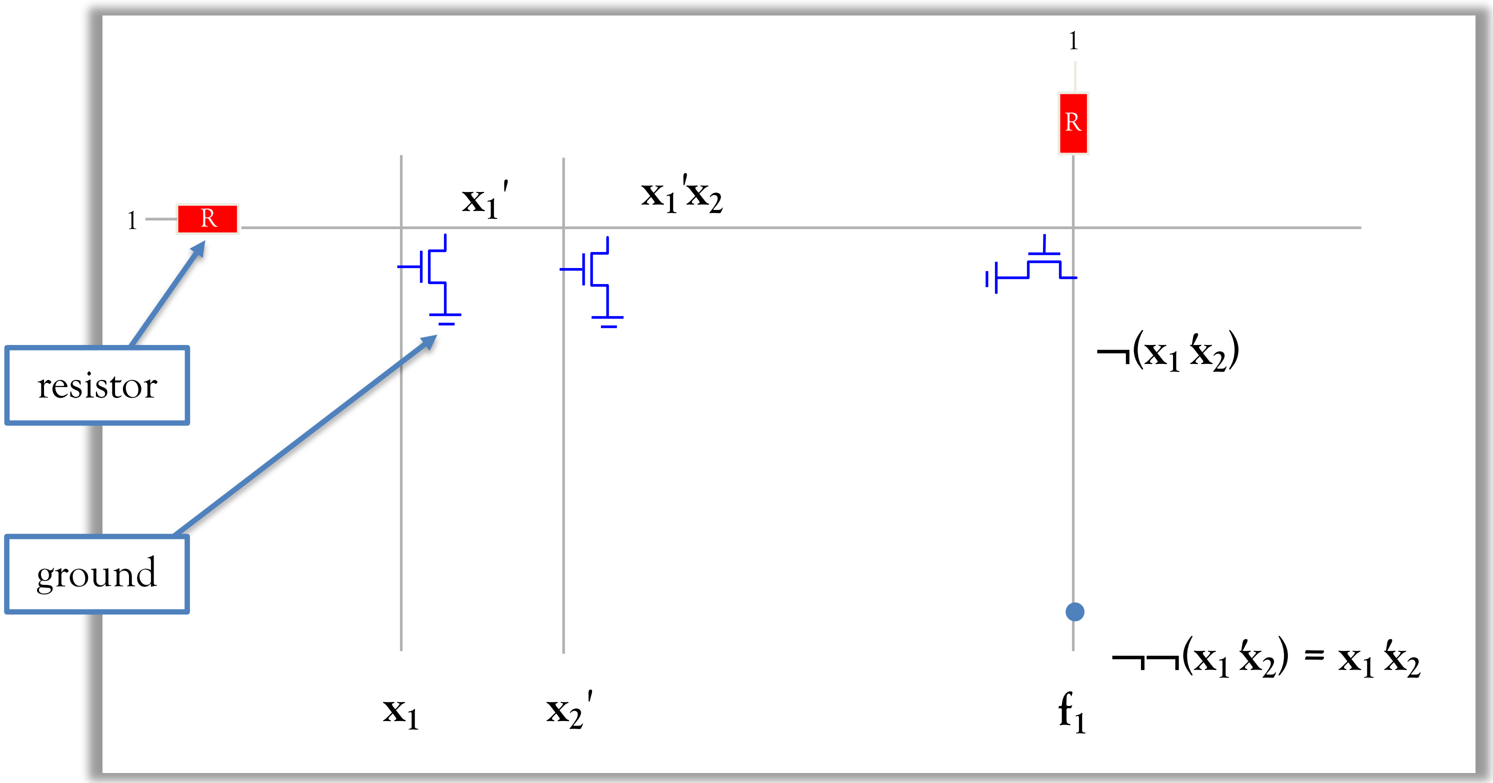
PLAs: Implementation of monomials

PLAs use n-type-transistors as switches:

If gate is 1,

the 1 at the source is pulled down to 0.

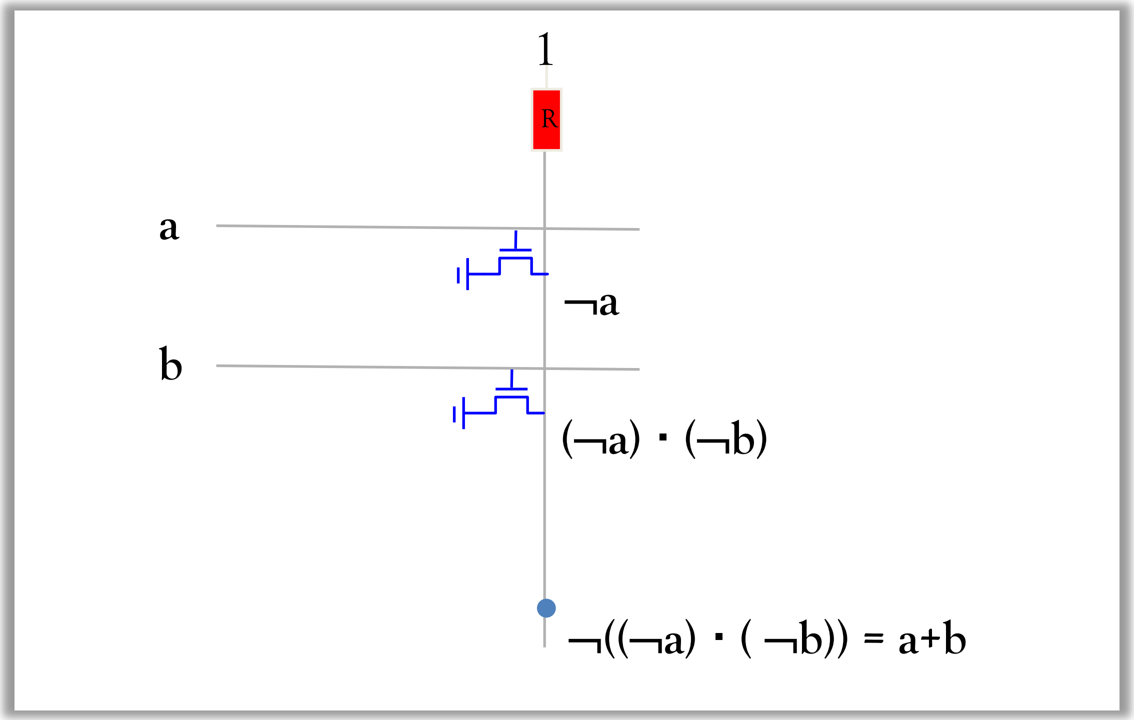
Computes the conjunction of the complements of the inputs.



How to implement disjunctions?

Employ double negation and de Morgan:

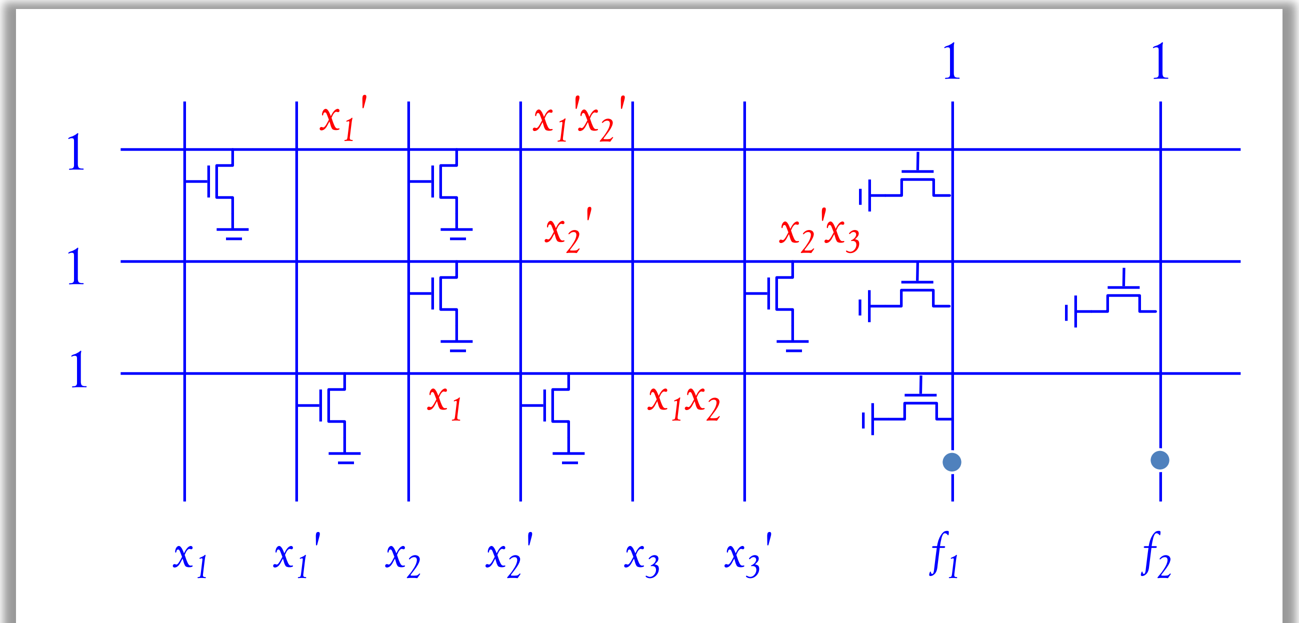
$$a + b \underset{\text{double negation}}{=} \neg\neg(a + b) \underset{\text{de Morgan}}{=} \neg((\neg a) \cdot (\neg b))$$



Programmable Logic Arrays: Example (1/2)

$f_1 = x_1'x_2' + x_2'x_3 + x_1x_2$
$f_2 = x_2'x_3$

Three monomials:
$x_1'x_2'$, $x_2'x_3$, *and* x_1x_2



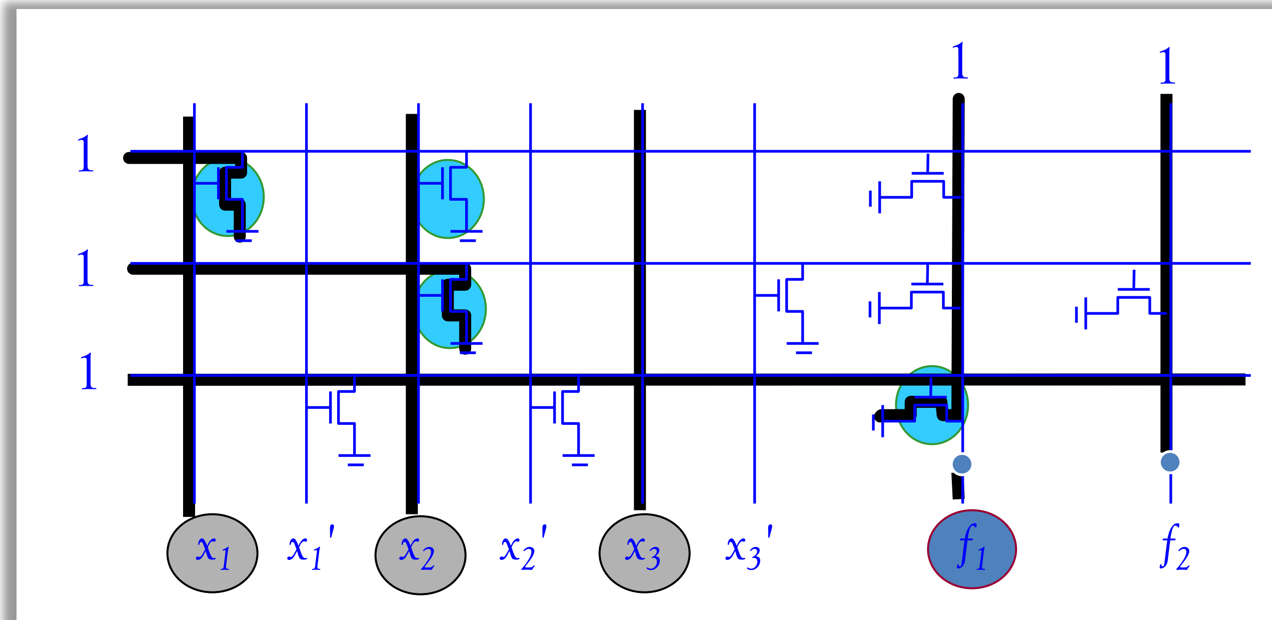
Programmable Logic Arrays: Example (2/2)

$f_1 = x_1'x_2'+x_2'x_3+x_1x_2$

$f_2 = x_2'x_3$

Three monomials: $x_1'x_2'$, $x_2'x_3$, *und* x_1x_2

Assume valuation $x_1=1$, $x_2=1$, $x_3=1$. Then we have:



Cost of monomials

Let $q = q_1 \cdot q_2 \cdot \ldots \cdot q_r$ be a monomial.

Then, the cost $|q|$ of q

is defined to be the number of transistors required to implement q in the PLA, so $|q| := r$.

Cost of polynomials

Let $p_1,\ldots,p_m$ be polynomials, and let $M(p_1,\ldots,p_m)$ denote the set of monomials occuring these polynomials.

- The **primary cost** $cost_1(p_1,\ldots,p_m)$ of a set of polynomials $\{p_1,\ldots,p_m\}$ is the number of required rows in a PLA to implement $p_1,\ldots,p_m$, and so
$$cost_1(p_1,\ldots,p_m) = |M(p_1,\ldots,p_m)|.$$
- The **secondary cost** $cost_2(p_1,\ldots,p_m)$ of a set of polynomials $\{p_1,\ldots,p_m\}$ is the number of transistors required, and so

$$cost_2(p_1,\ldots,p_m) = \sum_{q \in M(p_1,\ldots,p_m)} |q| + \sum_{i=1,\ldots,m} |M(p_i)|$$

Comparing costs

Let $cost = (cost_1, cost_2)$ be a cost function.

We define the following total order on costs as follows:

We have $cost(p_1, \ldots, p_m) \leq cost(q_1, \ldots, q_m)$, if

- $cost_1(p_1, \ldots, p_m) < cost_1(q_1, \ldots, q_m)$ or
- $cost_1(p_1, \ldots, p_m) = cost_1(q_1, \ldots, q_m)$ and $cost_2(p_1, \ldots, p_m) \leq cost_2(q_1, \ldots, q_m)$.

I.e. costs are lexicographically ordered.

Two-level logic minimization

Given:

A Boolean function $f = (f_1,\ldots,f_m)$ in n variables and m outputs represented via

- a truth table of size $m2^n$ or
- a set of m polynomials $\{p_1,\ldots,p_m\}$ with $\psi(p_i)=f_i$.

Wanted:

A set of polynomials $\{g_1, \ldots, g_m\}$, such that

- $\psi(g_i)=f_i$ for all i,
- $cost(g_1, \ldots, g_m)$ is **minimal**.

In the following, for simplicity, we will only consider **total Boolean functions** with **a single output**.

Illustration of monomials and polynomials

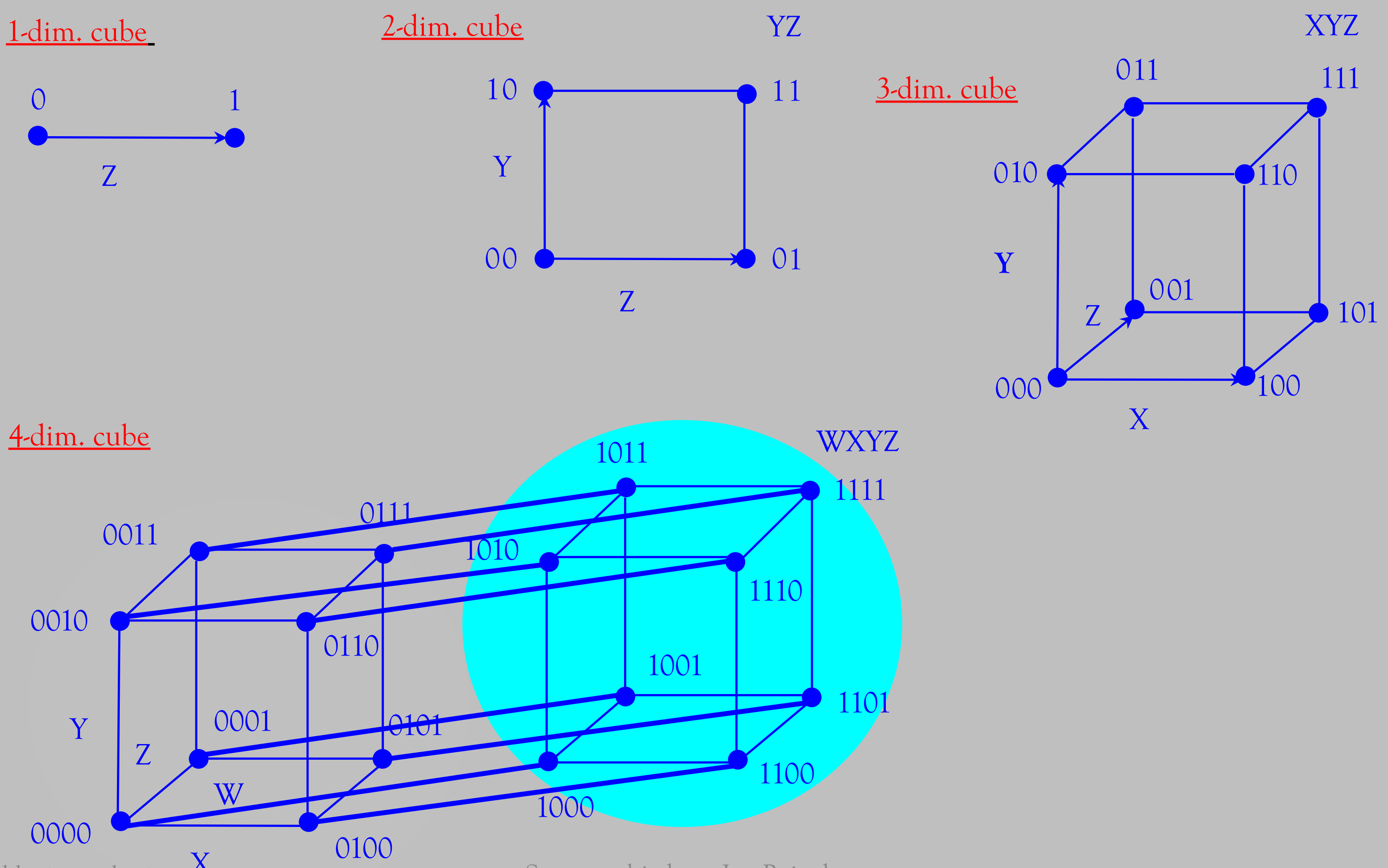


Illustration via hypercubes (1/2)

Every Boolean function f in *n* variables and a single output, can be specified by marking its on-set ON(f).

Example:

$$f(x_1,x_2,x_3,x_4)$$

$$= \quad x_1x_2$$

$$+ \; x_1'x_2'x_3'$$

$$+ \; x_1x_2'x_3'x_4$$

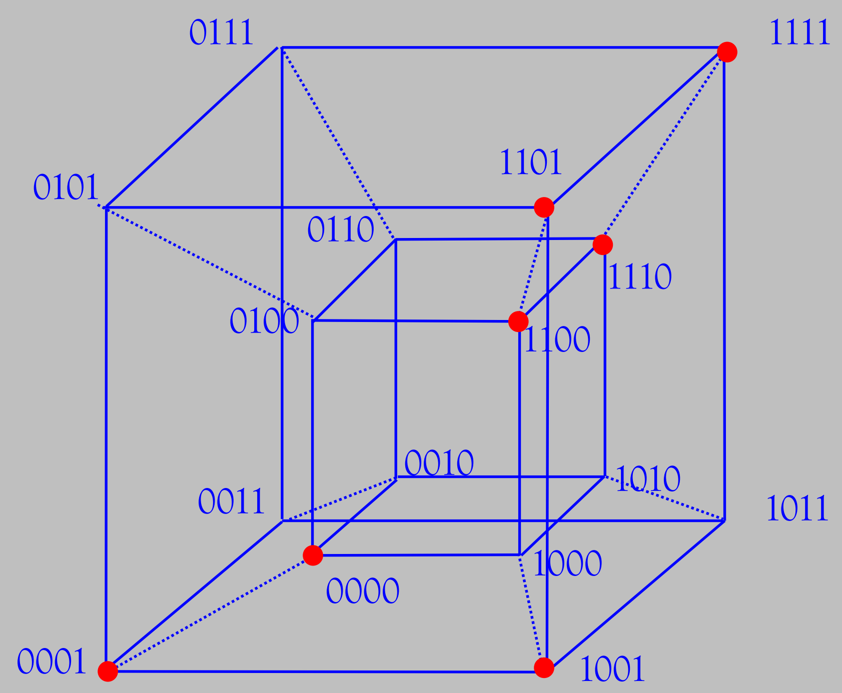
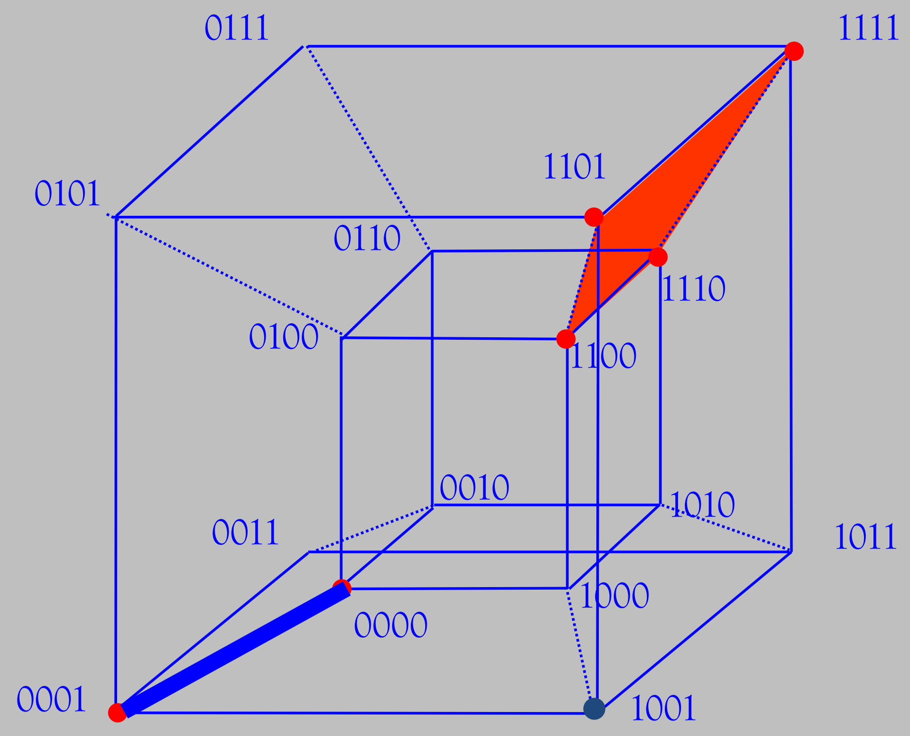


Illustration via hypercubes (2/2)

- Monomials of length *k* correspond to (*n*-*k*)-dimensional subcubes!
- A polynomial corresponds to the union of subcubes.

Example:

$$f(x_1,x_2,x_3,x_4) = \boxed{x_1x_2} + \boxed{x_1'x_2'x_3'} + \boxed{x_1x_2'x_3'x_4}$$

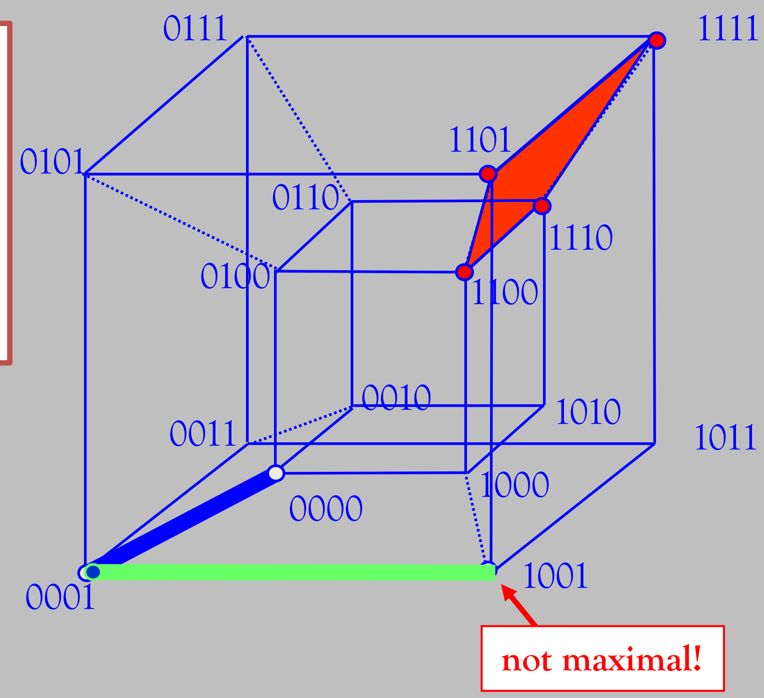


Formulation as a covering problem

Given:
A Boolean function f = (f_1,…,f_m) in *n* variables and a single output represented via a marked n-dimensional hypercube

Wanted:
A minimal covering of the marked nodes
via maximal subcubes
in the *n*-dimensional hypercube.

Minimal = with a minimal number of subcubes



… corresponds to the minimal polynomial:

$$\boxed{x_1 x_2} + \boxed{x_1' x_2' x_3'} + \boxed{x_2' x_3' x_4}$$