Co-factors: Re-cap

- A very useful operation on Boolean functions
  - Derived by fixing one of the variables to a constant (0 or 1)
- Applications of co-factors
  - Shannon's expansion – a way to recursively simplify or divide Boolean functions
  - Boolean difference ($f_x \oplus f_{x'}$)
  - Universal and Existential Quantification

Quantification

- Two more functions of Shannon co-factors
  - $f_{x_i} . f_{x_i'} = 1$ specifies when f = 1 independent of the value of x_i

    $f(x_1 \ldots x_{i-1}, \mathbf{x_i = 1}, x_{i+1} \ldots x_n) = 1$ AND

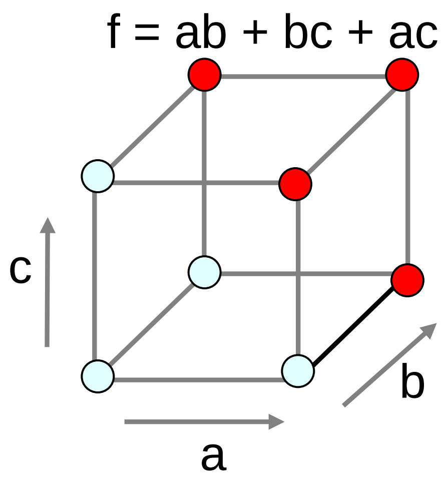
    $f(x_1 \ldots x_{i-1}, \mathbf{x_i = 0}, x_{i+1} \ldots x_n) = 1$
  - Called **Universal quantification** or **Consensus**

$$\boxed{\forall x(f) = f_x \cdot f_{x'}}$$

$$\boxed{C_a(f)}$$

Universal Quantification / Consensus

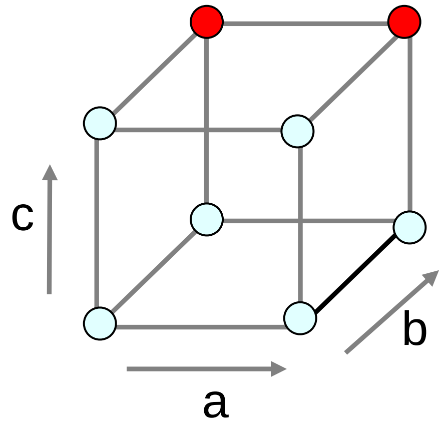
- Geometric interpretation using Boolean hypercube representation



$f_a = b + c$

$f_{a'} = bc$

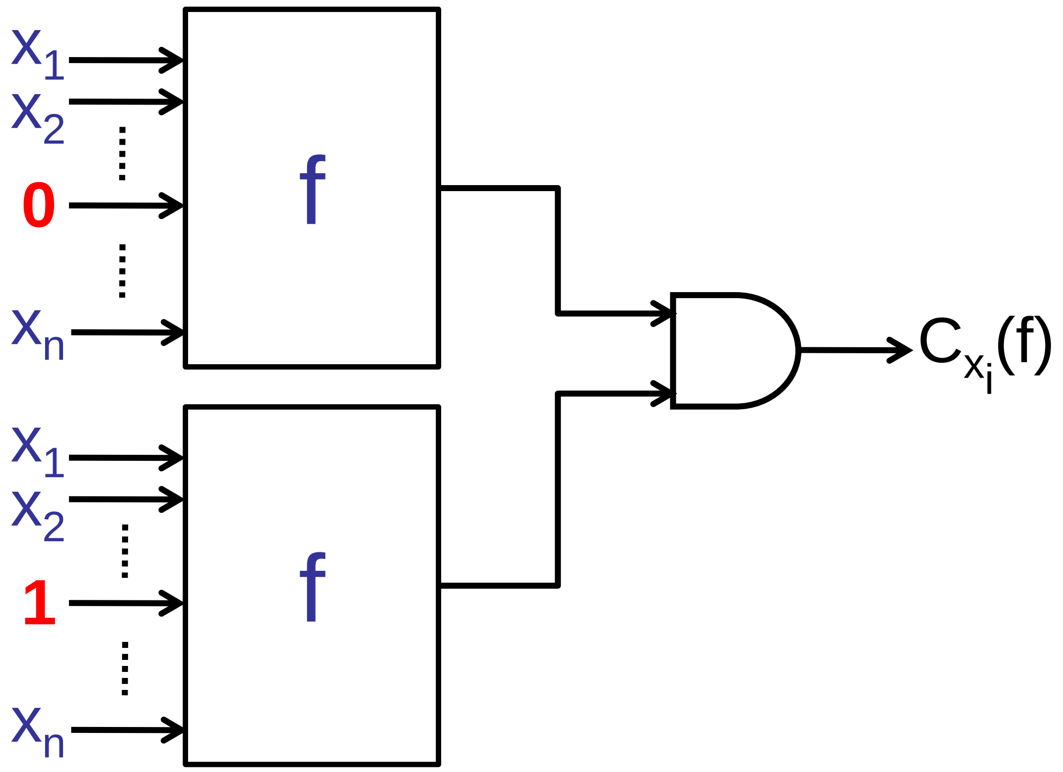
$\boxed{\forall a(f) = bc}$



Geometric interpretation: Keep vertices where f = 1 independent of a

Universal Quantification / Consensus

- Circuit interpretation



Quantification

- Two more functions of Shannon co-factors
  - $f_{x_i} \cdot f_{x_i'} = 1$ specifies when f = 1 independent of the value of x_i

    $f(x_1 \ldots x_{i-1}, \mathbf{x_i = 1}, x_{i+1} \ldots x_n) = 1$ AND

    $f(x_1 \ldots x_{i-1}, \mathbf{x_i = 0}, x_{i+1} \ldots x_n) = 1$

    $$\boxed{\forall x(f) = f_x \cdot f_{x'}}$$

    $\boxed{C_a(f)}$
  - Called **Universal quantification** or **Consensus**
  - $f_x + f_{x'} = 1$ specifies when f = 1 for at least one value of x_i

    $f(x_1 \ldots x_{i-1}, \mathbf{x_i = 1}, x_{i+1} \ldots x_n) = 1$ OR

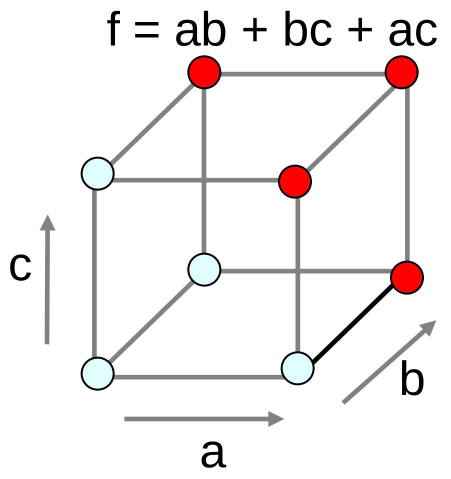
    $f(x_1 \ldots x_{i-1}, \mathbf{x_i = 0}, x_{i+1} \ldots x_n) = 1$

    $$\boxed{\exists x(f) = f_x + f_{x'}}$$

    $\boxed{S_a(f)}$
  - Called **Existential quantification** or **Smoothing**

Existential Quantification / Smoothing

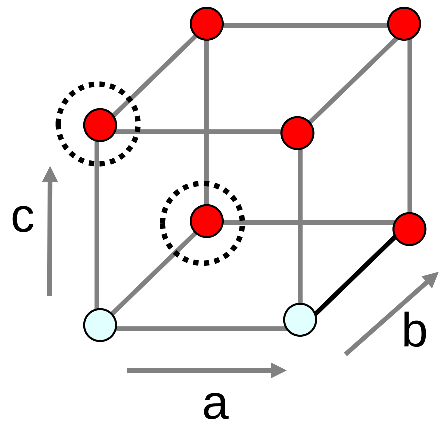
- Geometric interpretation using Boolean hypercube representation



$f_a = b + c$

$f_{a'} = bc$

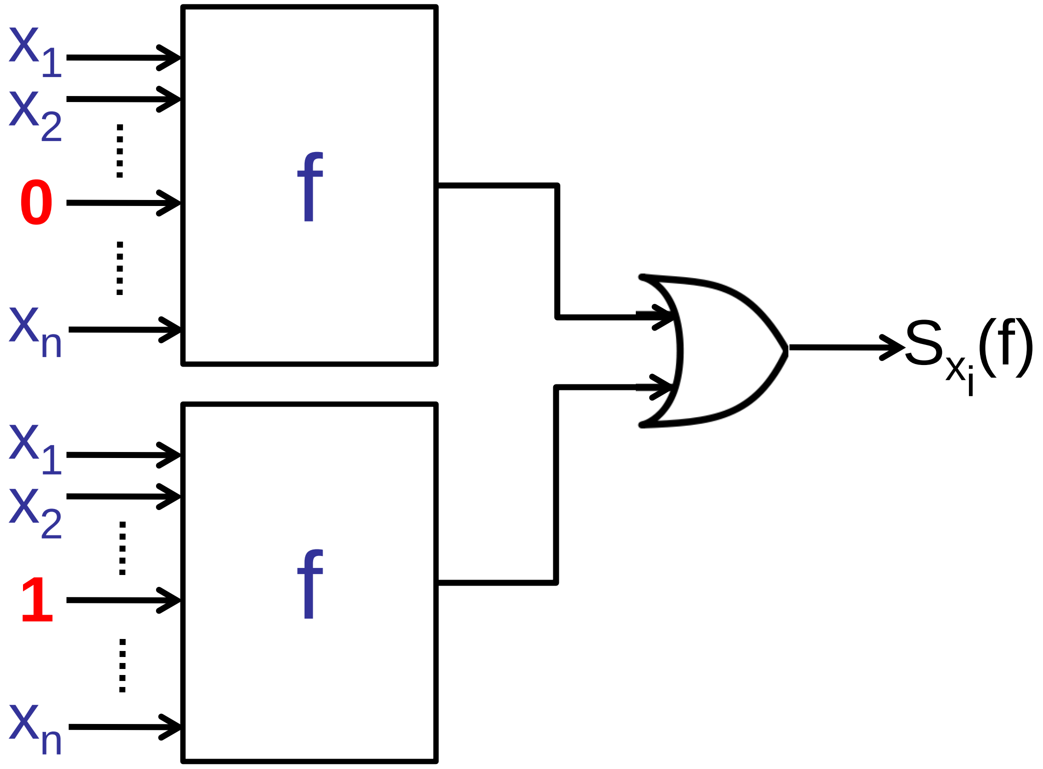
$\boxed{\exists a(f) = b + c}$



Geometric interpretation: If an off-set vertex has an on-set neighbor in the a-dimension, move it into the on-set

Existential Quantification / Smoothing

- Circuit interpretation



Properties of Consensus and Smoothing

- Can be applied w.r.t. multiple variables
  - $C_{xy}(f) = C_x(C_y(f)) = C_y(C_x(f))$
  - $S_{xy}(f) = S_x(S_y(f)) = S_y(S_x(f))$

- Containment properties
  - Consensus of a function f w.r.t. variable x is contained in f
  - Smoothing of a function f w.r.t. variable x contains f

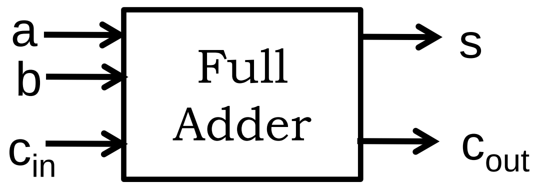
$$\boxed{\forall x(f) \subseteq f \subseteq \exists x(f)}$$

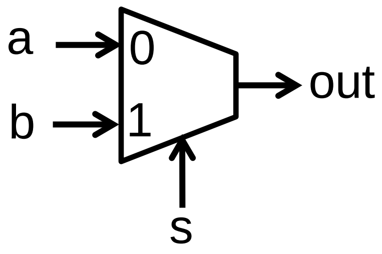
$$\boxed{C_x(f) \subseteq f \subseteq S_x(f)}$$

Hint: In this context, think of a function in terms of it's on-set

Quantification Examples

- Examples:





$\forall a(s) =$

$\forall c_{in}(s) =$

$\exists a(c_{out}) =$

$\exists c_{in}(c_{out}) =$

$\forall a(out) =$

$\forall s(out) =$

$\exists a(out) =$

$\exists s(out) =$

Lectures 3-5 Summary

- Boolean Algebra: Quick Review
- Advanced Boolean Algebra
  - Boolean spaces and functions
  - Representations of Boolean functions
  - Operations on Boolean functions
  - Co-factors and their applications
    - Shannon’s expansion
    - Boolean difference
    - Existential and Universal Quantification
      - a.k.a., Smoothing and Consensus

Reading for Lectures #5-8

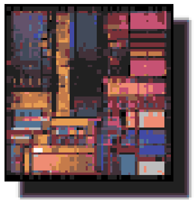
- Next Topic: Two-level synthesis
  - De Micheli, Chapter 7.1-7.4, 7.7
  - Hachtel & Somenzi, Chapter 4, Chapter 5

Acknowledgments

- Prof. Sharad Malik, Princeton
- Prof. Rob Rutenbar, CMU
- Prof. Maciej Ciesielski, U. Mass.

ECE 595Z
Digital Logic Systems Design Automation

Module 3 (Lectures 6-9): Two-level Logic Synthesis
Lecture 6



Anand Raghunathan

MSEE 348

raghunathan@purdue.edu

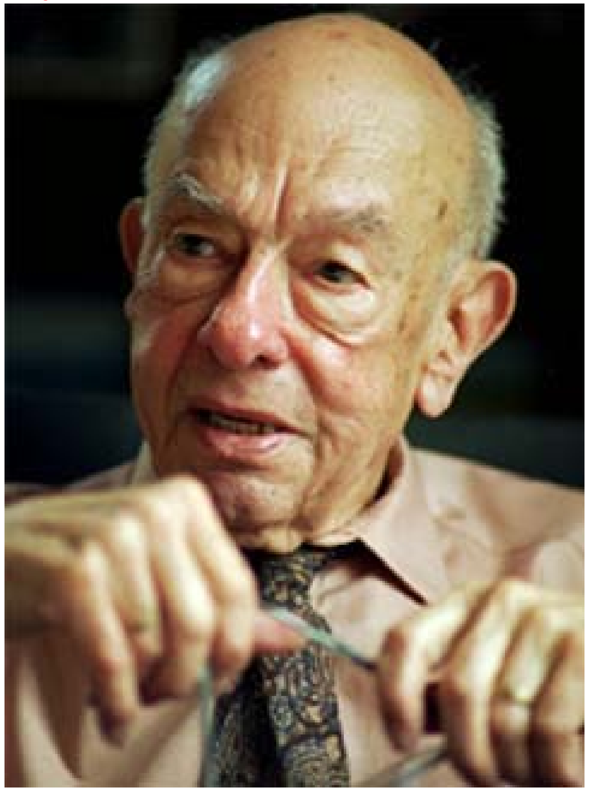
Re-visiting a Classic Problem

- What you know from your Digital Design class
  - Truth tables and Karnaugh maps
  - How to manually create minimal two-level (Sum-of-Products) implementations
- What we will discuss here
  - How the same thing can be done automatically
  - Computational strategies and scalability

Two-level Logic Minimization : A Brief History

- Initial ideas from Willard Quine
  - **Quine, W.V.** The Problem of Simplifying Truth Functions. In American Mathematical Monthly, Volume 59, Number 8 (1952), pp. 521-531.
  - **Quine, W.V.** A Way to Simplify Truth Functions. In American Mathematical Monthly, Volume 62, Number 9 (1955), pp. 627-631.

….. Quine never wrote on a computer, always preferring the 1927 Remington typewriter that he first used for his doctoral thesis. Because that project contained so many special symbols, he had to have the machine adjusted by removing the second period, the second comma and the question mark.
"You don't miss the question mark?" a reporter once asked him.
"Well, you see," he replied, "I deal in certainties."

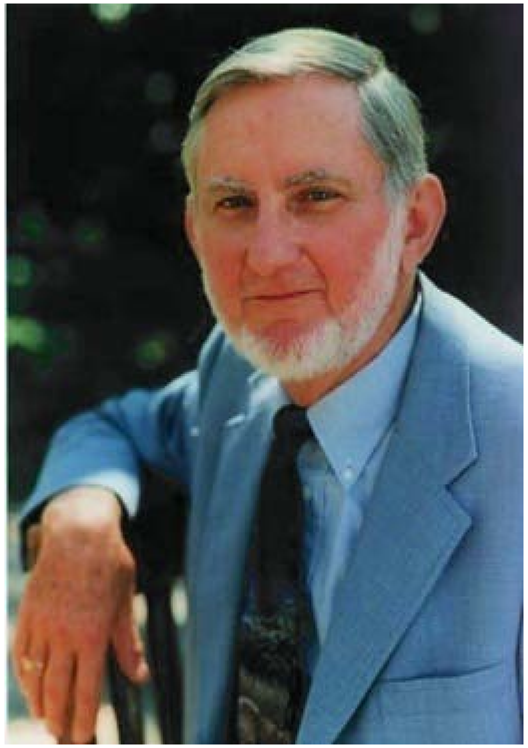

Willard Van Orman Quine
(1908-2000)
Analytic philosopher and logician

Source: New York Times, December 29, 2000

Two-level Logic Minimization : A Brief History

- Quine's ideas were refined by E. J. McCluskey
  - E. J. *McCluskey*, "*Minimization of Boolean functions*," The Bell System Technical Journal, vol. 35, no 5, pp 1417-1444, 1956.
- First algorithm suitable for implementation as a computer program

$$\begin{tikzcd}
\text{E. J. McCluskey} \arrow[d] \\
\text{J. Abraham} \arrow[d] \\
\text{N. K. Jha} \arrow[d] \\
\text{A. Raghunathan}
\end{tikzcd}$$



Edward J. McCluskey
Professor of EE and CS
Stanford University

http://www-crc.stanford.edu/users/ejm/McCluskey_Edward.html

Two-level Logic Minimization : A Brief History

- The Q-M method (with minor improvements) remained the state-of-the-art in exact two-level minimization for over two decades
- The increasing scale of integration (and circuit complexity) created a need for new (more efficient) algorithms
- Renewed research effort on two-level minimization, leading to new algorithms and software programs
  - Hong, S. J. and Cain, R. G. and Ostapko, D. L, "*Mini: A heuristic approach for logic minimization,*" IBM Journal of Research and Development, 1974.
  - Brayton, R. K., Sangiovanni-Vincentelli, A. L., McMullen, C. T., and Hachtel, G. D. 1984 *Logic Minimization Algorithms for VLSI Synthesis.* Kluwer Academic Publishers. (**ESPRESSO-II**)
  - M. R. Dagenais, V. K. Agarwal, and N. C. Rumin, "The McBOOLE logic minimizer. In *Proceedings of the 22nd ACM/IEEE Conference on Design Automation*, 1985. (**McBOOLE**)
  - M. Bartholomeus and H. D. Man, "Prestol-11: Yet Another Logic M' imizer for Programmed Logic Arrays". *Proc. Int. Symp. Circuits & Systems, June 1985.* (**Prestol-11**)

Two-level Logic Minimization : A Brief History

- With the advent of new symbolic analysis and optimization techniques (BDDs), new advances in the 1990s brought exact two-level minimization within the reach of larger circuits
- Heuristic techniques (ESPRESSO) still remain the de-facto standard

The Two-level Minimization Problem

- Given a Boolean function, derive a minimum two-level (SOP) implementation
- What does "minimum" mean?
  - Fewest product terms
  - Fewest literals
    - Literal : Appearance of a variable in true or complemented form

Example:

f = ab' + ab'c + abc' + acd

How many product terms?

How many literals?

PLA implementation of two-level functions

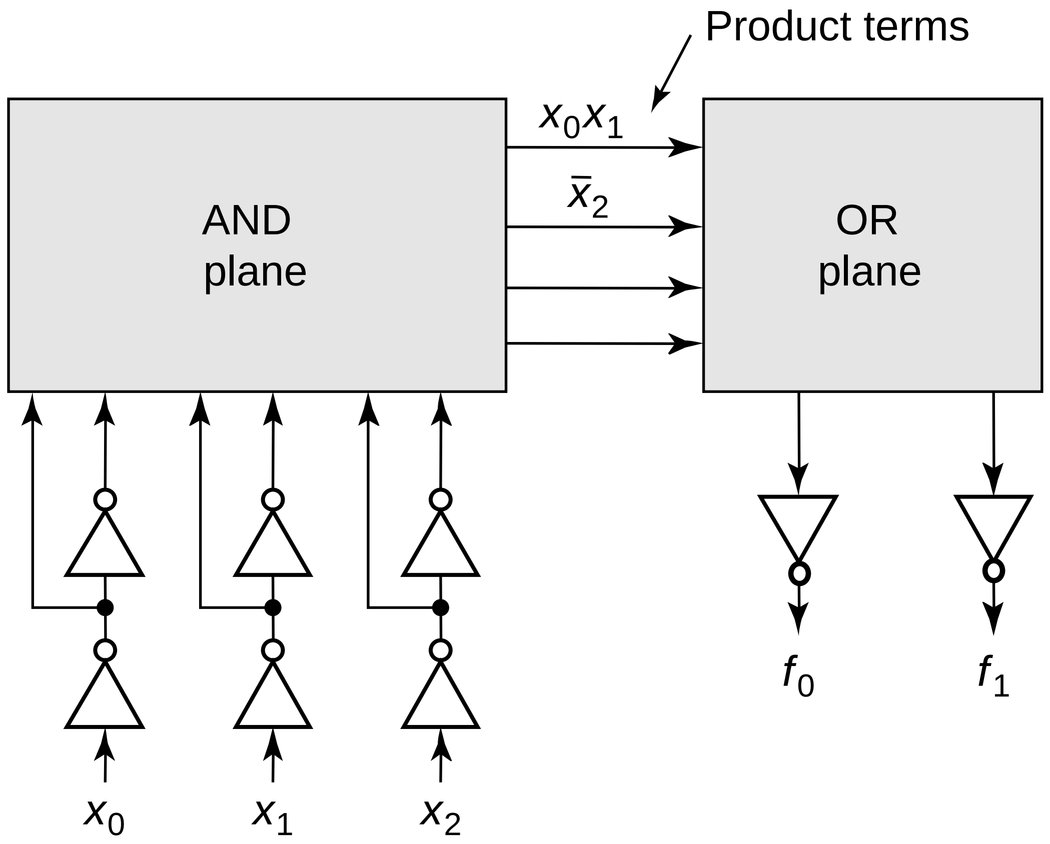
- Programmable Logic Array
  - Regular layout structure

$$f_0 = x_0x_1 + \overline{x_2}$$

$$f_1 = x_ox_1x_2 + \overline{x_2} + \overline{x_0}x_1$$

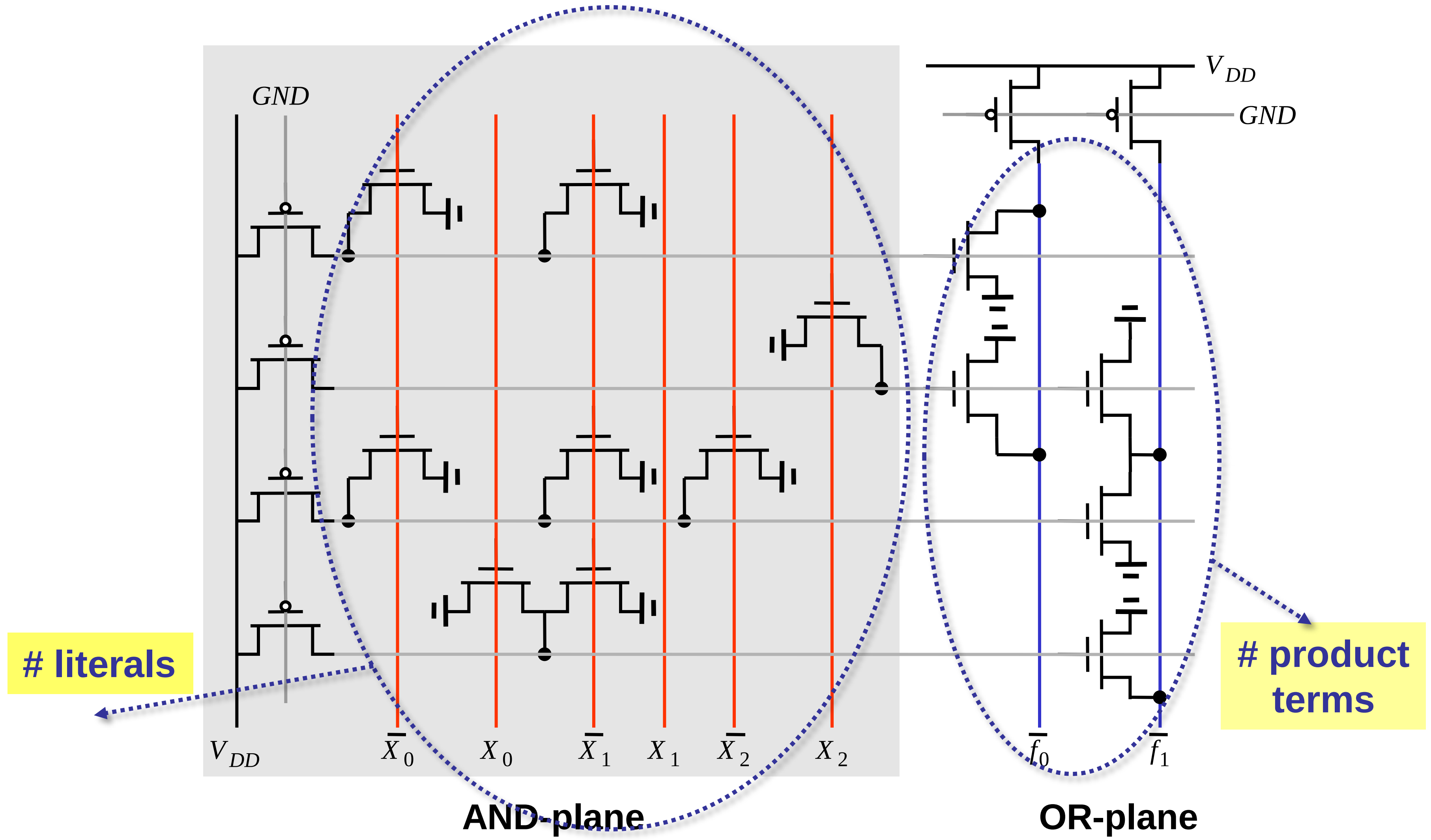
$$\overline{f_0} = \overline{\overline{(\overline{x_0} + \overline{x_1})} + \overline{x_2}}$$

$$\overline{f_1} = \overline{\overline{(\overline{x_0} + \overline{x_1} + \overline{x_2})} + \overline{x_2} + \overline{(x_0 + \overline{x_1})}}$$



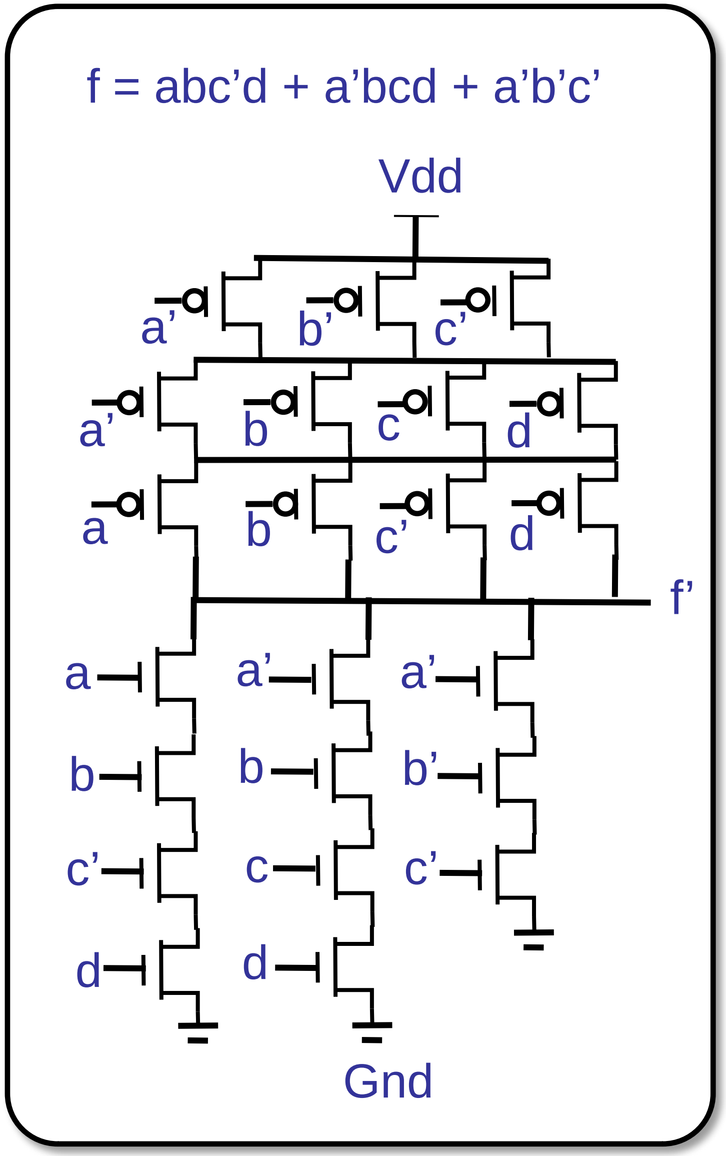
PLA implementation of two-level functions

- Pseudo-NMOS PLA : NOR-NOR implementation



Complex CMOS gate implementation

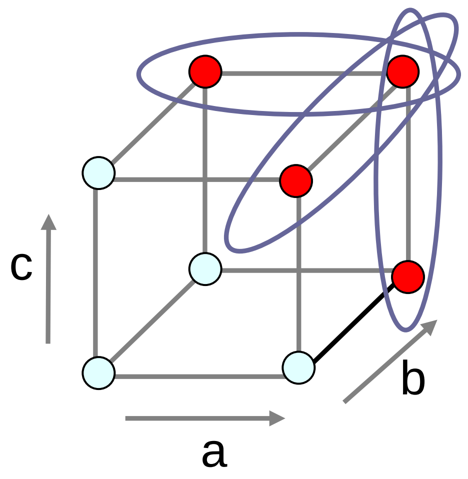
- SOP expressions can also be directly translated into complex CMOS gate implementations
  - Not scalable beyond a few inputs
  - # transistors = ??



Cubes, Implicants, and Covers (Recap)

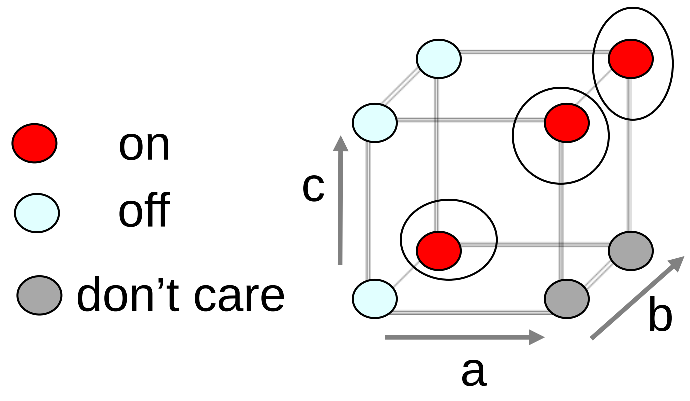
- Cube: Conjunction of literals
  - sub-space of a Boolean space B^n
- Implicant of a function f: Cube that is contained in the on-set of f
- Cover of a function f: Set of cubes that contains all minterms from the on-set of f, but does not contain any minterms from it's off-set
  - Represents a SOP implementation (each cube is a product term)

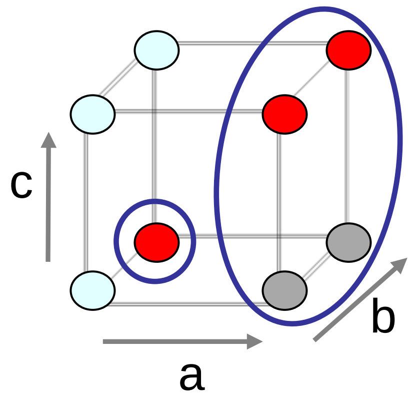
F = {ab, bc, ac}



Incompletely Specified Functions

- $f : B^n \rightarrow \{0, 1, *\}$
  - where * represents a *don't care.*
- Partition of B^n into on-set (f^1), off-set(f^0), and dc-set (f^* or f^{DC})
- Implicants and covers of f may include don't care vertices
  - But not off-set vertices





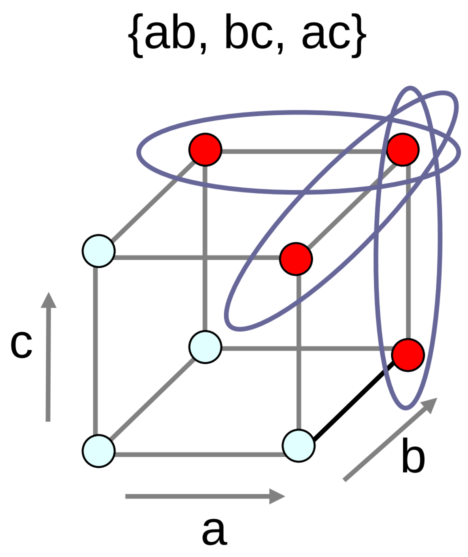
Prime Implicants and Covers

- An implicant (or cube) c_i of f is prime if it is not contained in any other implicant of f

  for all $c_i^* \supset c_i$

  $f + c_i^* \neq f$

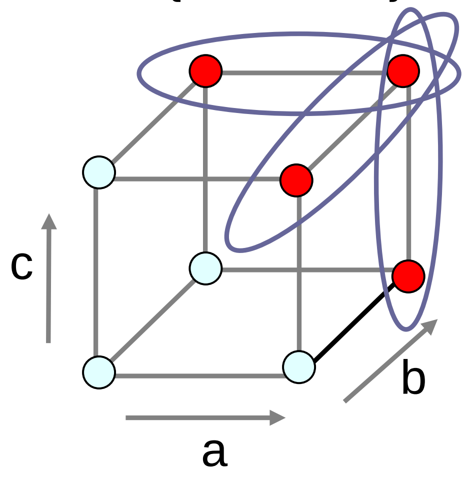
- Geometric interpretation: Expanding c_i in any dimension will make it include an off-set vertex
- Example:
  - ab is prime in $f = ab + ac + bc$
- A cover is prime if all of its cubes are prime



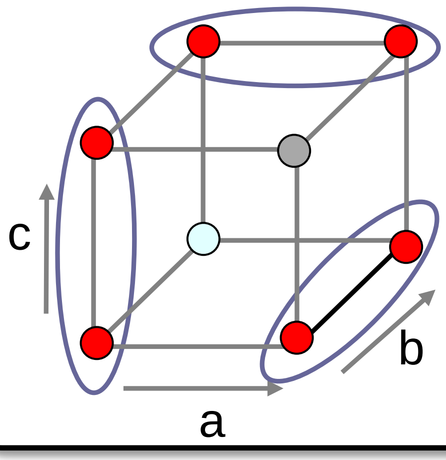
Irredundant Covers

- Let $C = \{c_1, c_2, \ldots c_k\}$ be a cover for f.
  - $f = c_1 + c_2 + \ldots c_k$
- A cube c_i is irredundant in C if $C - c_i$ is not a cover for f.
  - $f - c_i \neq f$
- Example:
  - ab is irredundant in ab + ac + bc
- A cover is irredundant if all of its cubes are irredundant

{ab, bc, ac}



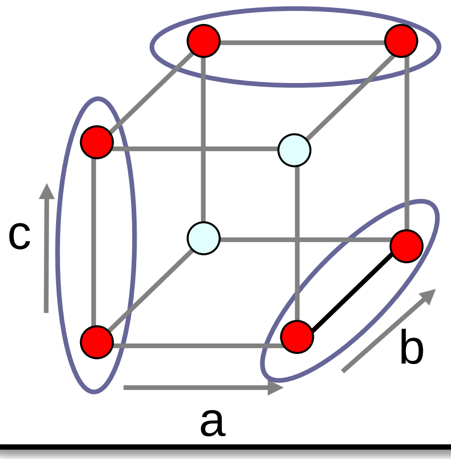
Is this cover prime and irredundant?



Essential Implicants

- A prime is essential if there is a minterm that is covered by that prime and no other prime of the function.

Which of these primes are essential?



Quine-McCluskey

Theorem [Quine]: There exists a minimum cover for f that is a prime cover.

- Why is this useful?
  - How many possible implicants for a function?
  - How many possible primes for a function?

Classical Quine-McCluskey Algorithm:

```
QM(f){
      P = $\cup_i$ $p_i$; /* $p_i$ is a prime of f */
      C = minimum_cover(P) ;
}
```

- Two major steps
  - Prime generation
  - Selection of a minimum subset of primes

General Principle

- Reduce the search space as much as possible even before you start the search!

Quine-McCluskey : Covering Table

- How do we get the minimum sub-set of primes?
- Formulate as a “covering” problem
- Covering table provides the relationship between primes and on-set minterms

Example of a Covering Table

	p1	p2	p3	p4	p5	p6	p7	p8
m1	1	0	0	0	0	0	0	0
m2	1	1	0	0	0	0	0	1
m3	0	1	1	0	0	0	0	0
m4	0	0	1	1	1	0	0	0
m5	0	0	0	0	1	0	1	1
m6	0	0	0	0	0	1	1	0
m7	0	0	0	0	1	1	0	1
m8	0	0	0	0	1	1	1	0

primes

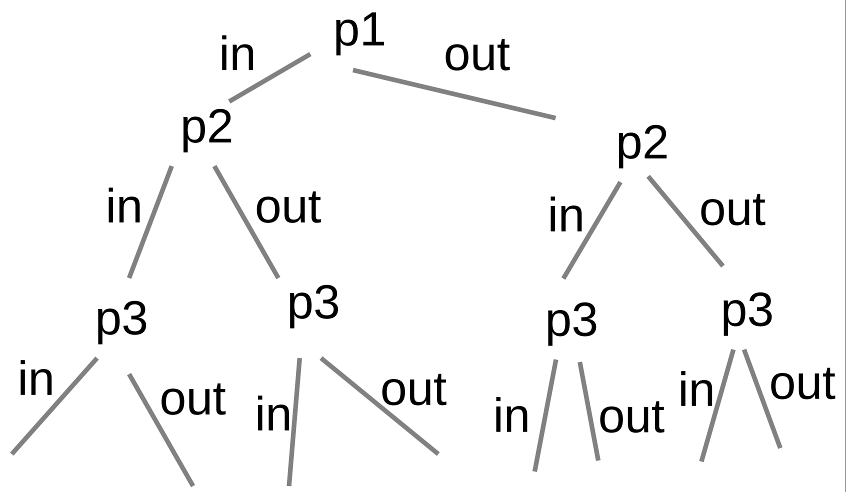
minterms

1: minterm covered by prime
0: minterm not covered by prime

Quine-McCluskey : Column Covering Problem

- Find the minimum subset of columns that *covers* all rows.
  - A column j covers row i if T(i,j) = 1
  - NP-complete problem!
- Brute force technique:
  - Consider all elements P
    - Each p_i may be included or excluded in the cover.
    - $2^{|P|}$ possibilities - $2^{|P|}$ leaves in the search tree
- **Need to do better!**

Search tree for column covering


p1
in
out
p2
in
out
p2
in
out
p3
in
out
p3
in
out
p3
in
out
p3
in
out