Logic Design Background

Boolean Algebra (B,+,0,1), B={0,1}

Boolean function $f\colon B^n \rightarrow B$

Incompletely specified, multiple output

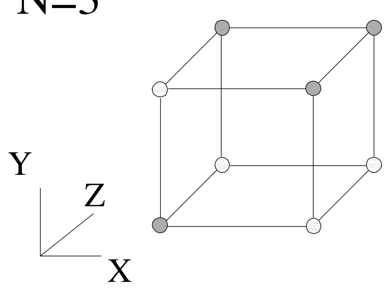
$$f\colon B^n \rightarrow \{0, 1, *\}^m$$

ON, OFF and DC sets: subsets of domain

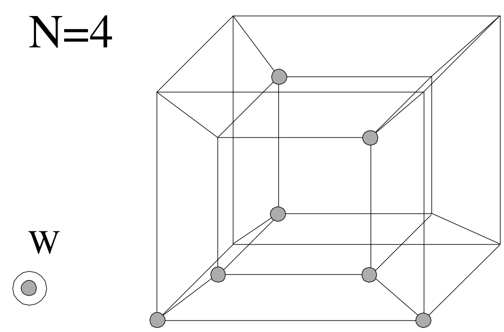
Boolean n-Cube

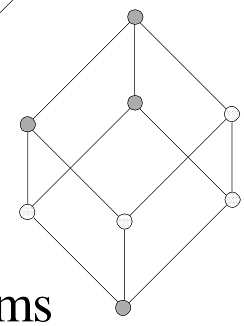
Completely specified function examples

N=3



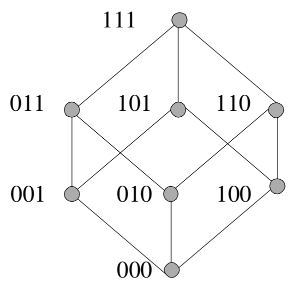
N=4

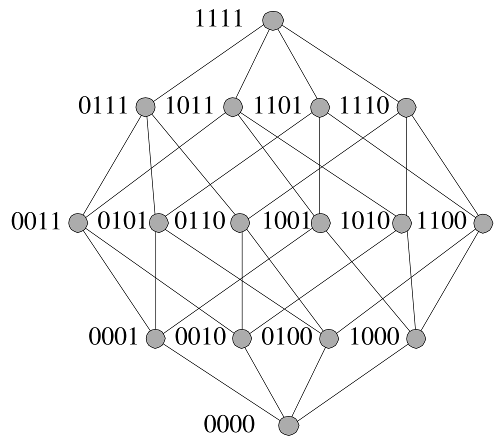




- Represent any Boolean function
- Vertex = **minterm**
- **Tautology**: function is 1 for all minterms

N-cube and Boolean Lattice





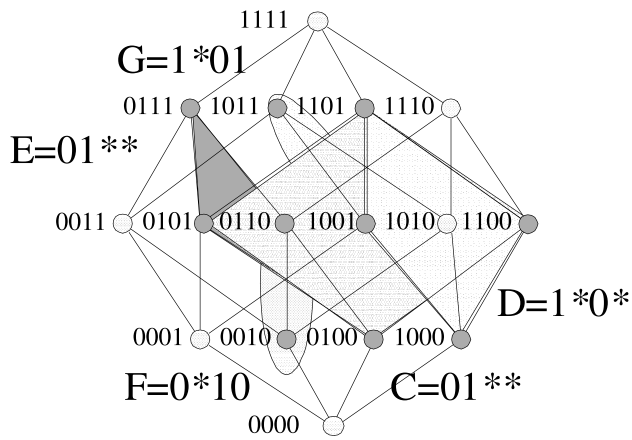
- Can draw any n-cube
  - Preserve neighbourhood relations
- Relations: partial order between points
- Counting:
  - Nodes at a level, indegree, outdegree

Definitions

- **Variable**: *x, y, z*
- **Literal**: *x, x', y, y', ...*
- **Product** (**cube**): $c = xy'z$
- **Implicant**: product c s.t. $f(c)=1$, or $c=>f$
- **Prime** Implicant: c no implicant d s.t. $c=>d$
- Truth table: values of *f* for all inputs
- **Cover** (implicant table): cubes for $f=1$
- **Irredundant** cover: no cube can be omitted

Definitions at Work

- Vertices: 2^n, logic functions: 2^{2^n}
- Cube with k literals: $2^{(n-k)}$ vertices
- Relation: $\subseteq$
  - Cube containment



Max. # Implicants?
XOR(parity): $2^n/2$

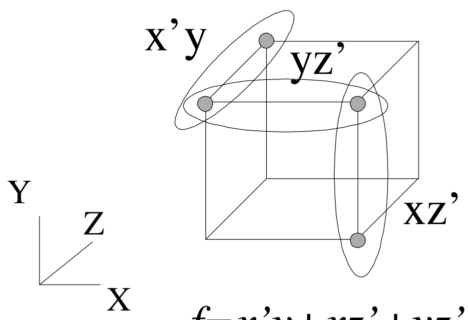
Truth Table vs. Logic Cover

x	y	z	f	g
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	0
1	0	1	0	1
1	1	0	1	1
1	1	1	1	1

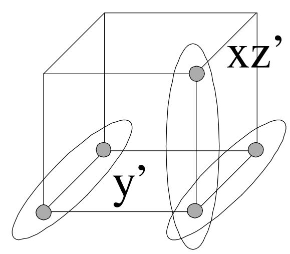
x	y	z	f	g
0	0	1	1	0
*	1	1	1	1
1	0	1	0	1
1	1	*	1	1

- Canonical (unique) or not
  - Truth table is better
- Reduced or not (tractable)
  - Implicant table (logic cover) is better

Example Covers



$f = x'y + xz' + yz'$



$g = x'y' + xy' + xz' = y' + xz'$

Why is f redundant?

What are primes of g?

Implicant table for both:

x	y	z	f	g
0	1	*	1	0
*	0	*	0	1
1	*	0	1	1

- Why is f redundant?
- What are primes in g?
- Implicant table for f and g ?

Cofactors

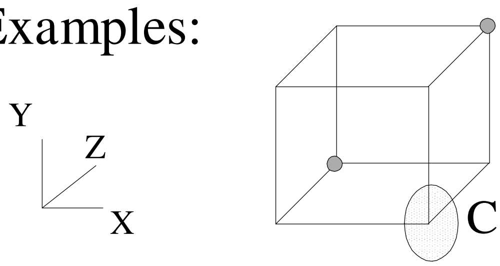
Shannon (Boole) expansion: $f = xf_x + x' f_{x'}$

Expansion tree

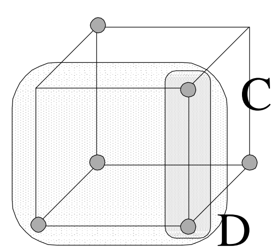
Cofactor by a cube: f_C =(f restricted to C)

Theorem: $c \subseteq f \Leftrightarrow f_C = 1$

Examples:



$f = xyz + x'y'z$, $C = xy'z'$
$fc = 0$



$f = y' + x z' + x'yz$, $C = z'$, $D = xz'$
$fc = y' + x$, $f_D = 1$

Operations on Functions

- **Boolean difference** w.r.t. x:

$$\partial f/\partial x = f_x \oplus f_{x'}$$

- **Consensus** w.r.t. x:

$$\mathrm{C_x} = f_x \cdot f_{x'}$$

- **Smoothing** w.r.t. x:

$$\mathrm{S_x} = f_x + f_{x'}$$

- Relation between the operators?

Concepts Introduced

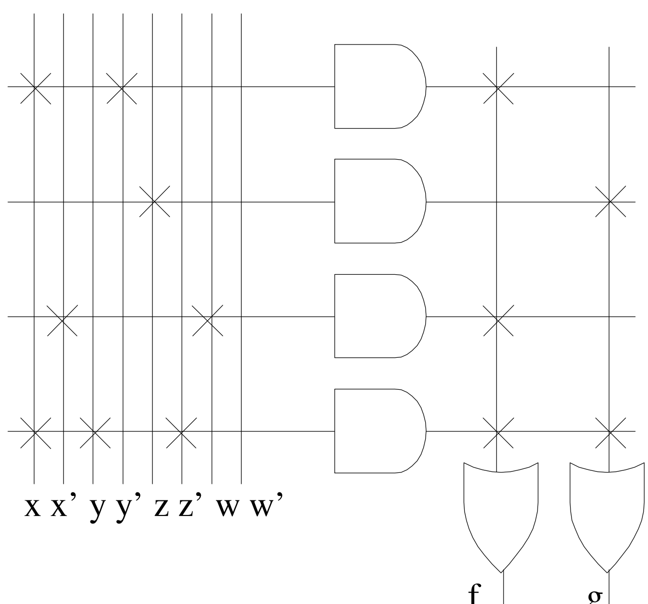
- Role of Logic Synthesis
- Logic design styles
- Boolean function representations
  - Java applets with n-cubes
- Literal, minterm, cube (product term)
- Logical covers, primality and irredundancy

Two-Level Logic

- Sum-of-products (SOP): two-levels
- Obvious way of Boolean function representation
  - SOP and POS
- Physical Implementations: PALs and PLAs
- Used in larger structures
  - Multi-level logic
  - Sequential logic

Programmable Logic Arrays

PLAs:



f=xy'+x'w+xyz'
g=z+xyz'

PALs: fixed OR array

Two-Level Optimization

Criteria:

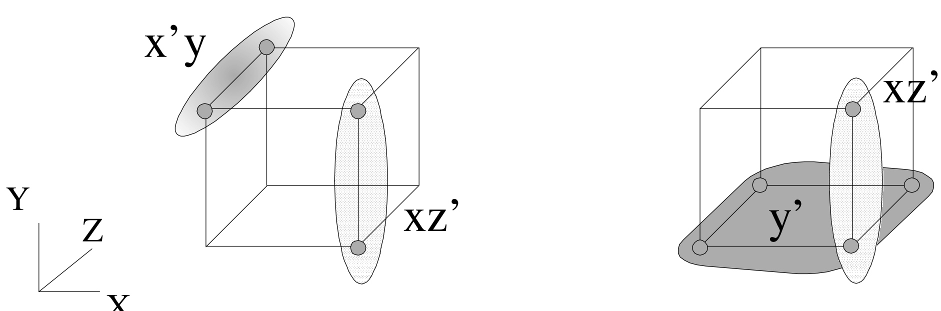
Reduce number of implicants
Reduce number of literals (secondary)

Rationale:

#(implicants)=#(rows)
Literals: transistors, fanin, loading

Cover with minimal number of implicants

Example Covers



f=x'y+xz'   *g=y'+xz'*

Minimum and irredundant prime cover

Essential prime implicants: *y', xz'* and *x'y*

Implicant table cannot be simplified