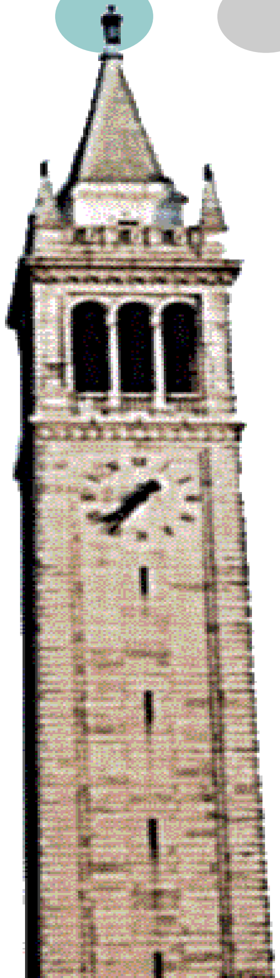


Fundamental Algorithms for System Modeling, Analysis, and Optimization

Edward A. Lee, Jaijeet Roychowdhury, Sanjit A. Seshia
UC Berkeley

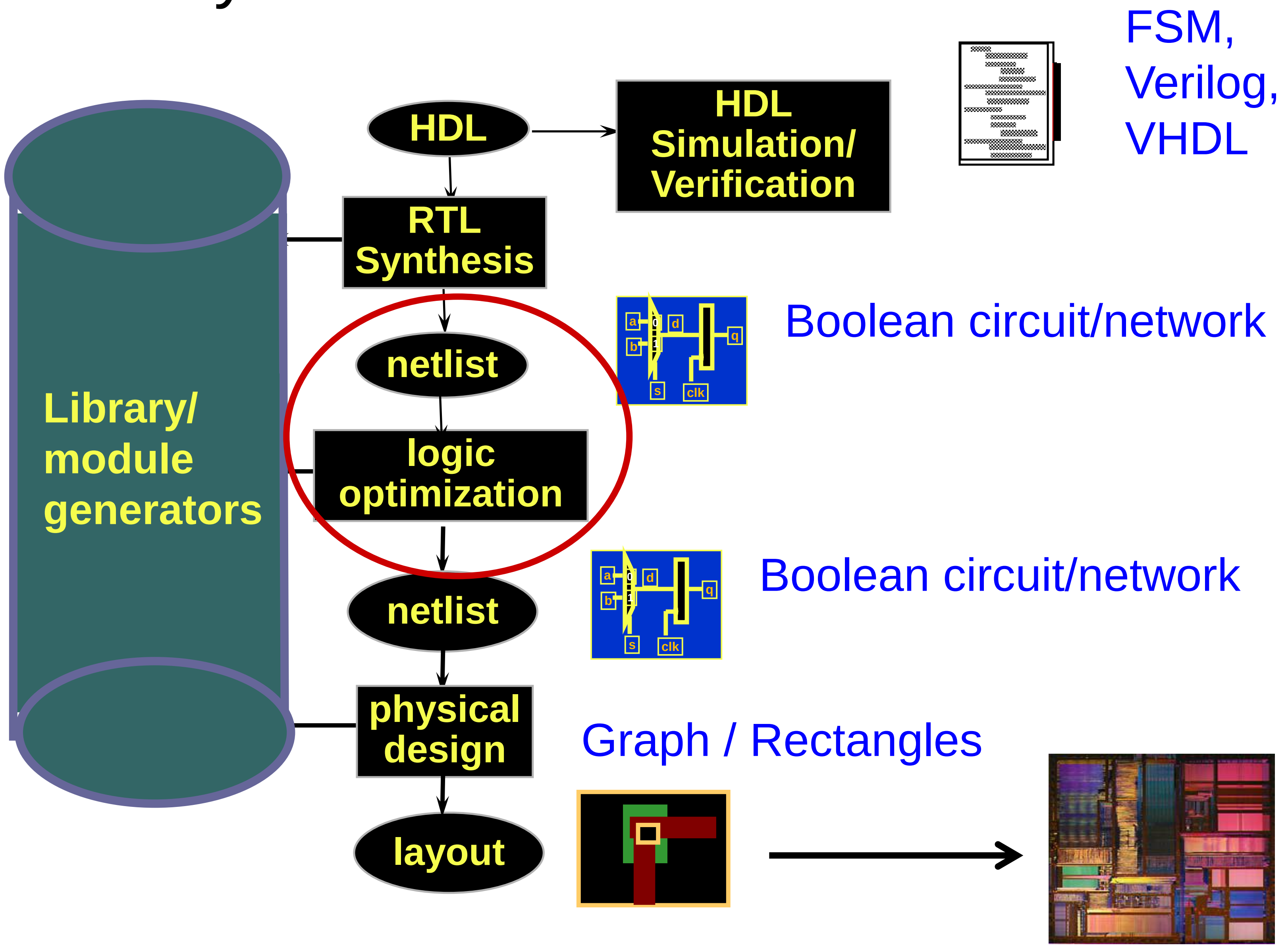
EECS 244 Fall 2016

Lecturer: Yu-Yun Dai

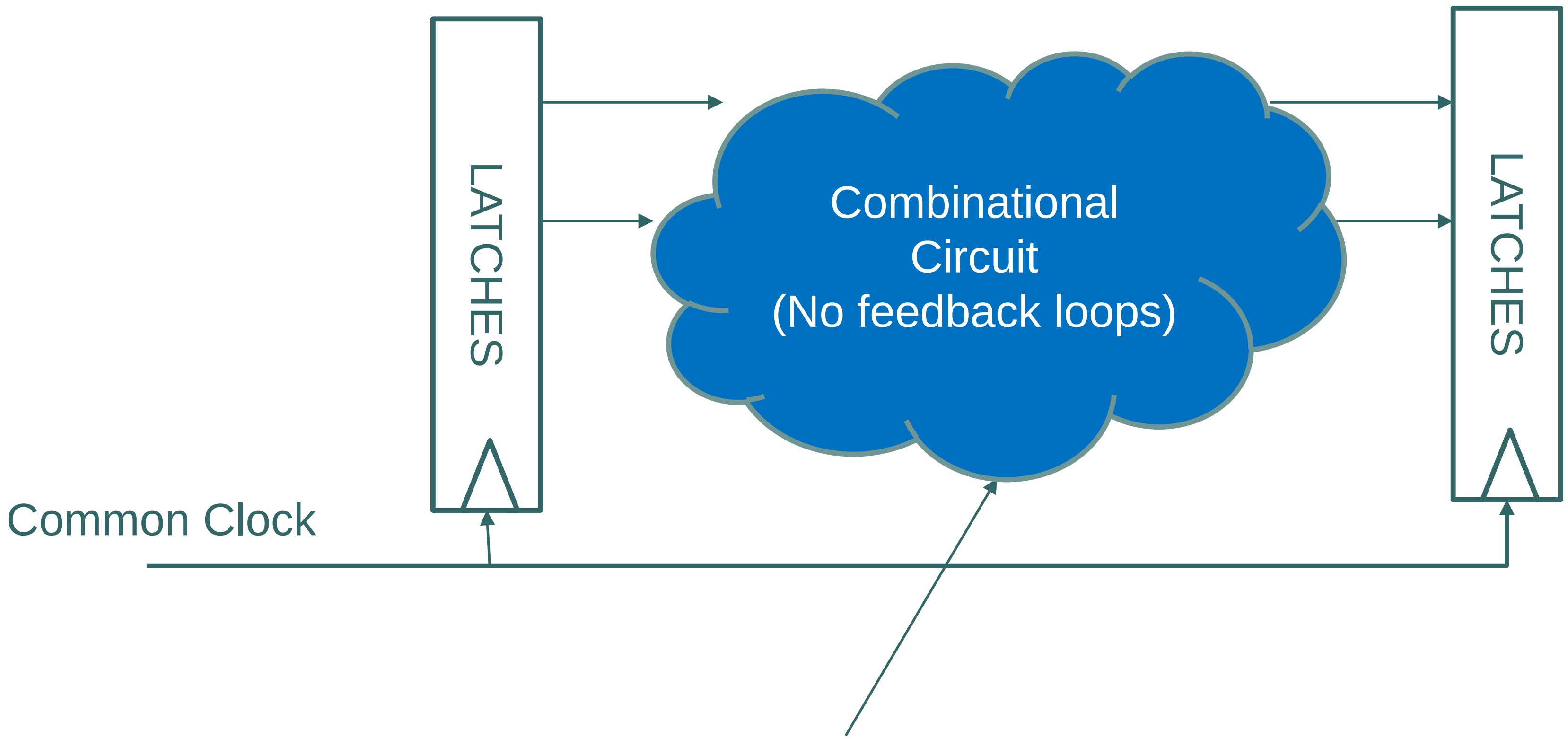
Boolean Algebra and Two-Level Logic Optimization

Thanks to S. Devadas, K. Keutzer, S. Malik, R. Rutenbar, R. Brayton, A. Kuehlmann for several slides

RTL Synthesis Flow

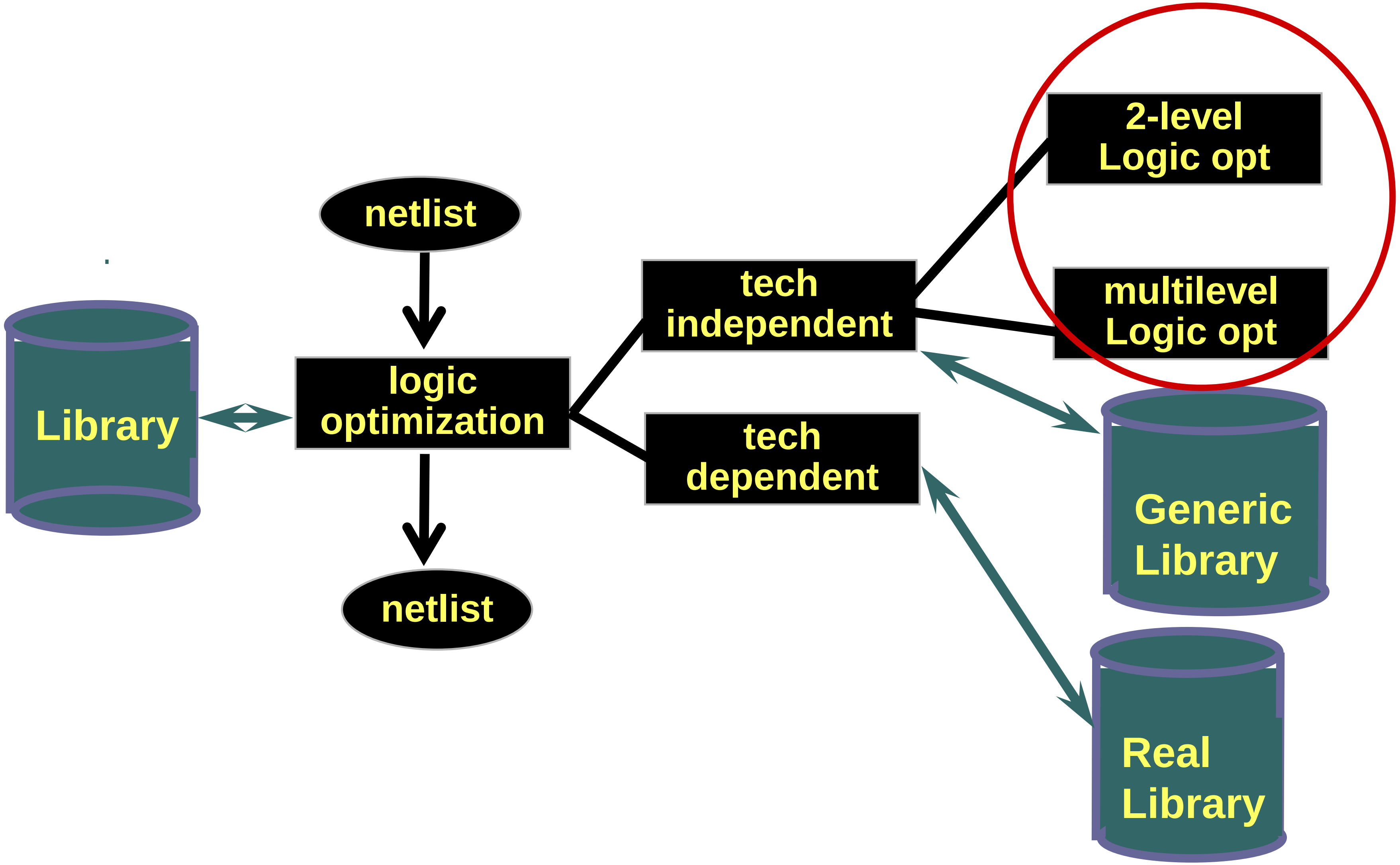


Sequential v.s. Combinational Synthesis/Logic Optimization



Optimize the size/delay/etc. of the combinational circuit
(viewed as a Boolean network)

Logic Optimization



Outline of Topics

Basics of Boolean functions

- Prime, Implicants, cubes
- Tautology checking

Two-level logic optimization

- Quine-McCluskey Method
- Espresso

Multi-level logic optimization

Definitions – 1: What is a Boolean function?

Let $B = \{0, 1\}$ and $Y = \{0, 1\}$

Input variables: $X_1, X_2 \ldots X_n$

Output variables: $Y_1, Y_2 \ldots Y_m$

A logic function *f* (or 'Boolean' function, switching function) in n inputs and m outputs is a map

$$f\colon B^n \longrightarrow Y^m$$

Definition used in Logic Optimization

don’t care – aka “X”

Let $B = \{0, 1\}$ and $Y = \{0, 1, 2\}$

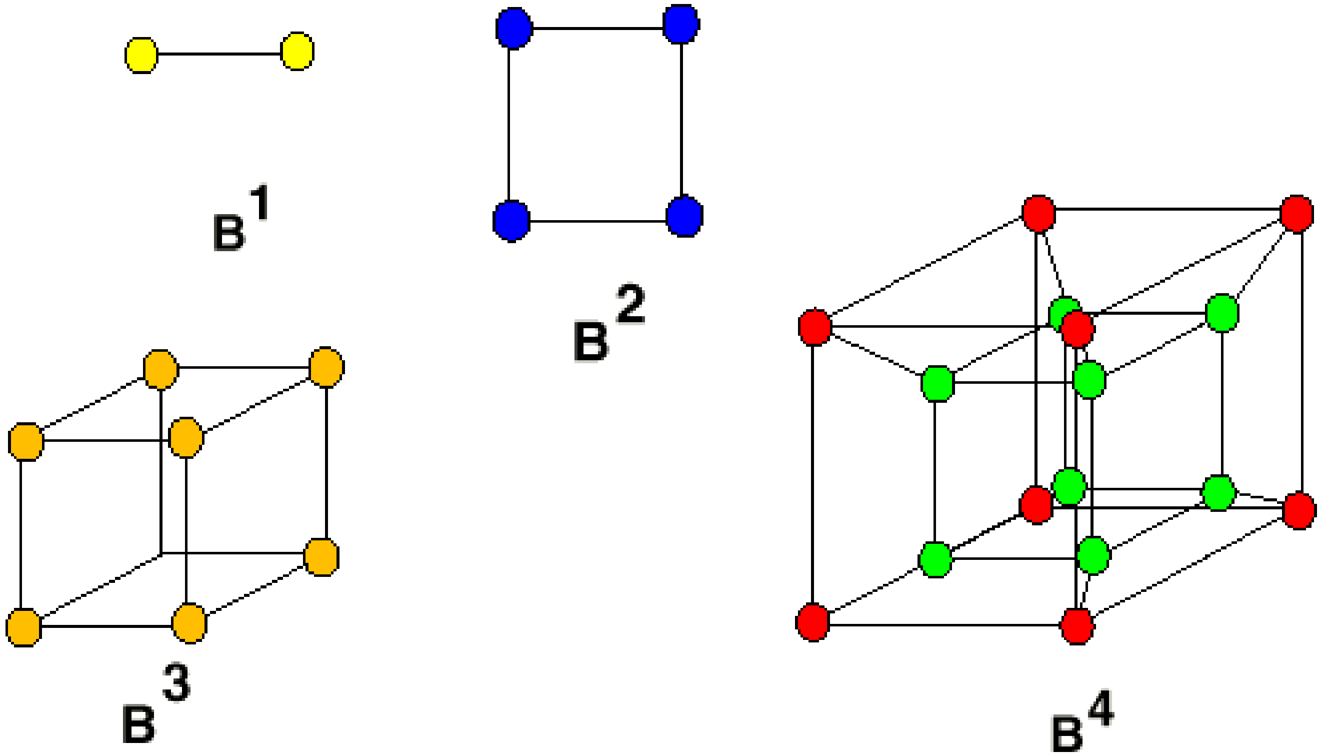
Input variables: $X_1, X_2 \ldots X_n$

Output variables: $Y_1, Y_2 \ldots Y_m$

A logic function ff (or ‘Boolean’ function, switching function) in n inputs and m outputs is a map

$$ff: B^n \longrightarrow Y^m$$

The Boolean n-Cube, B^n



- $\mathcal{B} = \{0, 1\}$
- $\mathcal{B}^2 = \{0, 1\} \times \{0, 1\} = \{00, 01, 10, 11\}$

Boolean Functions

$B = \{0, 1\}$, $x = (x_1, x_2, \ldots, x_n)$

$x_1, x_2, \ldots$ are **variables**

$x_1, x_1', x_2, x_2', \ldots$ are **literals**

Each vertex of B^n is mapped to 0 , 1 or 2 (don't care)

the **onset** of f is $\{x | f(x)=1\} = f^{1} = f^{-1}(1)$

the **offset** of f is $\{x | f(x)=0\} = f^{0} = f^{-1}(0)$

if $f^{1} = B^n$, f is the **tautology**, i.e. $f \equiv 1$

if $f^{0} = B^n$ ($f^{1} = \emptyset$), f is **not satisfiable**

if $f(x) = g(x)$ for all $x \in B^n$, then f and g are **equivalent**

We write simply f instead of f^{1}

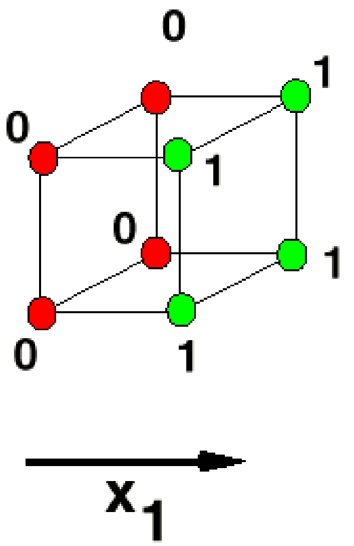
Literals

A literal is a variable or its negation

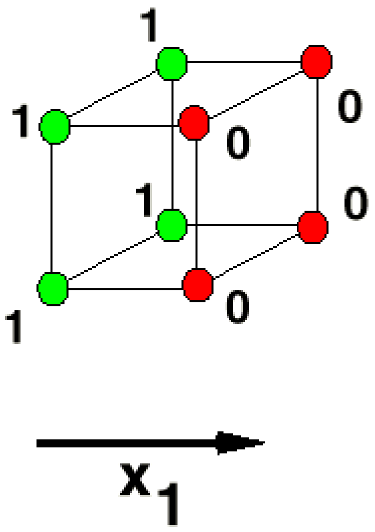
y, y'

It represents a logic function

Literal x_1 represents the logic function f, where $f = \{x | x_1 = 1\}$
Literal x_1' represents logic function g where $g = \{x | x_1 = 0\}$



$f = x_1$



$f = \overline{x}_1$

Green – ON-set
Red – OFF-set

Boolean Formulas -- Syntax

Boolean formulas can be represented by formulas defined as catenations of

- parentheses (,)
- literals x, y, z, x', y', z'
- Boolean operators + (OR), X (AND)
- complementation, e.g. (x + y)'

Examples

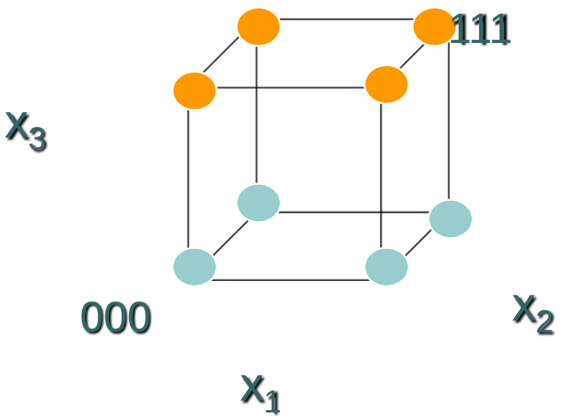
$$f = x_1 \text{ X } x_2' + x_1' \text{ X } x_2 = (x_1+x_2) \text{ X } (x_1'+x_2')$$

$$h = a + b \text{ X } c = (a' \text{ X } (b' + c'))'$$

We usually replace X by catenation, e.g. a X b→ab

Logic functions

There are 2^n vertices in input space B^n



000	1
001	0
010	1
011	0
100 ⇒	1
101	0
110	1
111	0

"truth table"

There are $\mathbf{2^{2^n}}$ distinct logic functions.

- How many logic formulae?

Each subset of vertices is a distinct logic function:

$f \subseteq B^n$

“Semantic” Description of Boolean Function

EXAMPLE: Truth table form of an incompletely specified function

$ff: B^3 \longrightarrow Y^2$

X_1	X_2	X_3	Y_1	Y_2
0	0	0	1	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	2
1	1	0	1	1
1	1	1	2	1

Y_1: ON-SET$_1$ = {000, 001, 100, 101, 110}

OFF-SET$_1$ = {010, 011}

DC-SET$_1$ = {111}

Operations on Logic Functions

(1) Complement: $f \longrightarrow \bar{f}$ ($\neg f$ or f')

interchange ON and OFF-SETS

(2) Product (or intersection or logical AND)

$h = f \cdot g$ (what happens to ON/OFF sets?)

(3) Sum (or union or logical OR):

$h = f + g$ (ON/OFF sets?)

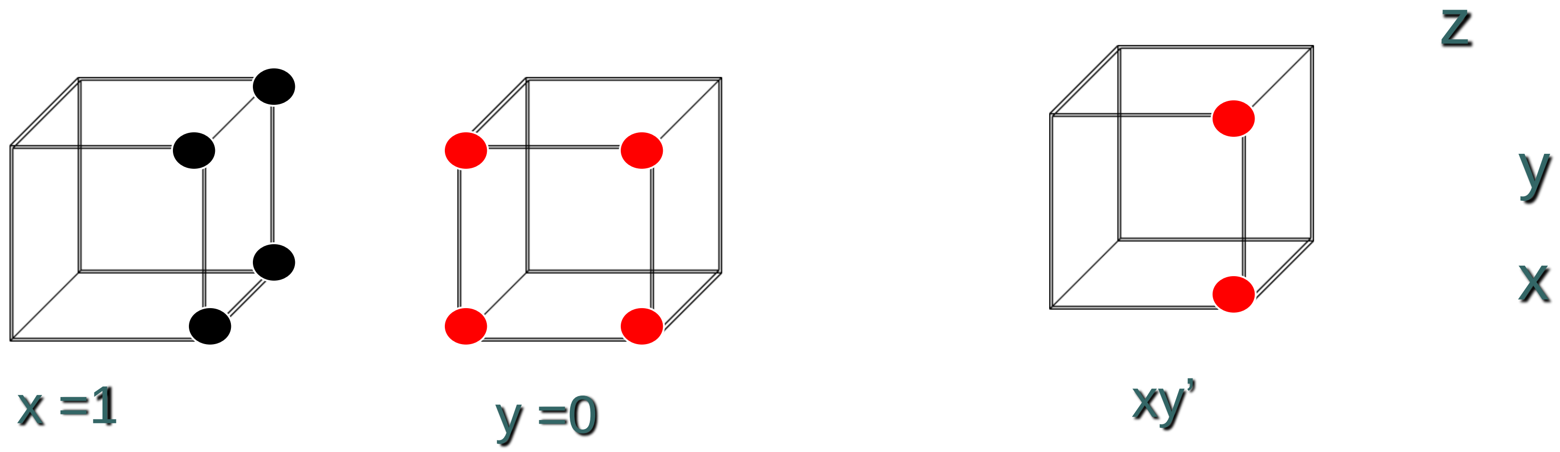
Cubes

The AND of a set of literal functions ("conjunction" of literals) is a cube

(also view as a set of minterms)

C = xy' is a cube

C = (x=1)(y=0)



2-level Minimization: Minimizing SOP (DNF)

$$F1 = \overline{A}\,\overline{B} + \overline{A} B D + \overline{A} B \overline{C}\,\overline{D} + A B C \overline{D} + A \overline{B} + A B D$$

Inputs	Outputs
0 0 - -	1
0 1 - 1	1
0 1 0 0	1
1 1 1 0	1
1 0 - -	1
1 1 - 1	1

$$F1 = \overline{B} + D + \overline{A}\,\overline{C} + A C$$

Inputs	Outputs
- 0 - -	1
- - - 1	1
0 - 0 -	1
1 - 1 -	1

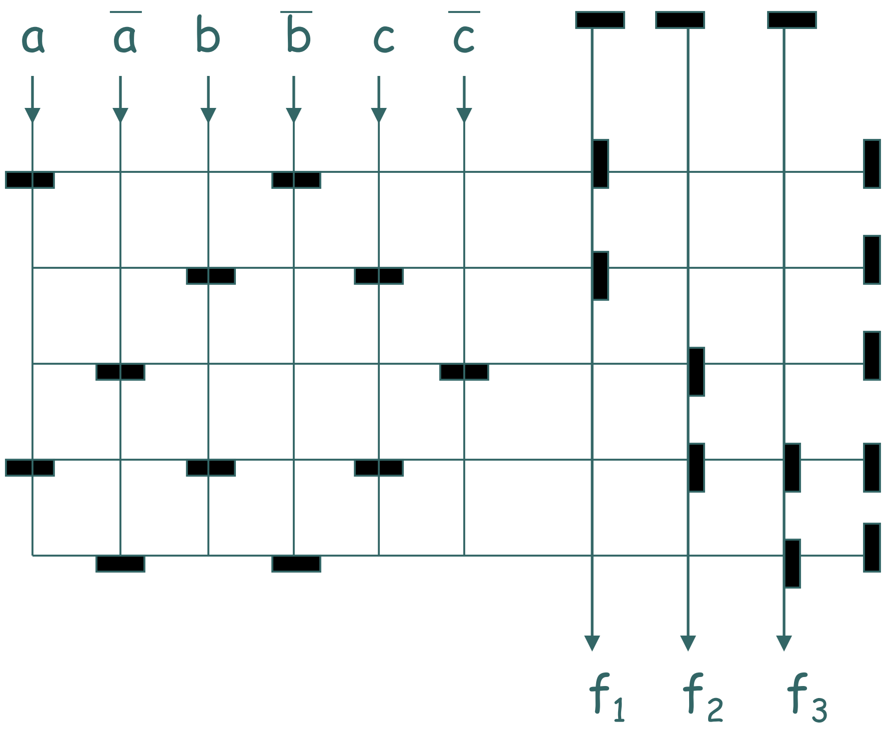
↑ **minimum representation**

(number of cubes, literals)

PLA's - Multiple Output Functions

A PLA is a function $f : B^n \rightarrow B^m$ represented in SOP form:

n=3, m=3



Personality Matrix

abc	$f_1 f_2 f_3$
10-	1 - -
-11	1 - -
0-0	- 1 -
111	- 1 1
00-	- - 1

PLA’s (cont.)

Each distinct cube appears just once in the AND-plane, and can be shared by (multiple) outputs in the OR-plane, e.g., cube (abc).

Extensions from single output to multiple output minimization theory are straightforward.

Multi-level logic can be viewed mathematically as a connection of single output functions.

Implicants

An *implicant* of a function f is a *cube* p that does not intersect the OFF-SET of f

$p \subseteq f_{ON} \cup f_{DC}$

Prime Implicants

An *implicant* of f is a *cube* p that does not intersect the OFF-SET of f

$p \subseteq f_{ON} \cup f_{DC}$

A ***prime implicant*** of f is an implicant p such that

(1) No other implicant q contains it

(i.e. $p \not\subset q$)

(2) $p \not\subset f_{DC}$

A ***minterm*** is a fully specified implicant

e.g., 011, 111 (not 01-)

Examples of Implicants/Primes

X_1	X_2	X_3	Y_1
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	2

000, 00- are implicants, but not primes (-0-)

How about 1-1 ? 0-0 ?

Prime and Irredundant Covers

A ***cover*** is a set of cubes C such that

$$C \supseteq f_{ON} \quad \text{and} \quad C \subseteq f_{ON} \cup f_{DC}$$

All of the ON-set is covered by C

C is contained in the ON-set and Don’t Care Set

A *prime cover* is a cover whose cubes are all prime implicants

An *irredundant cover* is a cover C such that removing any cube from C results in a set of cubes that no longer covers the function (ON-set)

A prime of f is essential (essential prime) if there is a minterm (essential vertex) in that prime but in no other prime.

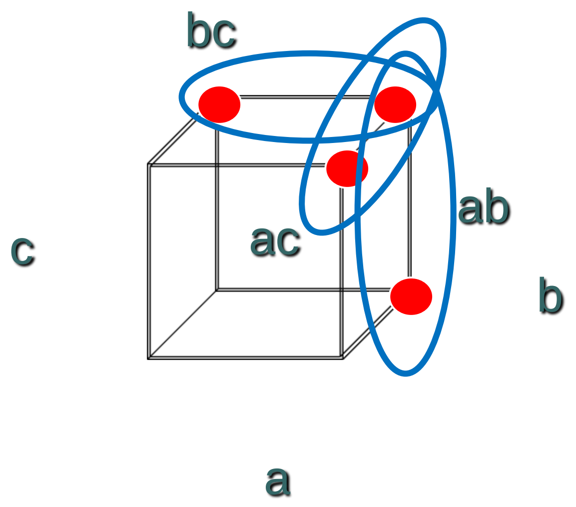
Irredundant

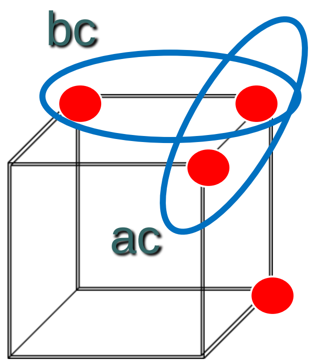
Let $F = \{c_1, c_2, \ldots, c_k\}$ be a cover for f.

$$f = \sum_{i=1}^{k} c_i$$

A cube $c_i \in F$ is irredundant if $F \backslash \{c_i\} \not\supset f$

Example 2: $f = ab + ac + bc$





Not covered

$F \backslash \{ab\} \not\supset f$

Example Covers

X_1	X_2	X_3	Y_1
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	2

0 0 -
1 0 -   is a cover. Is it prime?
1 1 -   Is it irredundant?

Minimum Covers

Definition: A *minimum cover* is a cover of minimum ***cardinality***

Theorem: There exists a minimum cover that is a prime and irredundant cover.

Why?

Minimum Covers

Defn: A *minimum cover* is a cover of minimum cardinality

Theorem: There exists a minimum cover that is a prime and irredundant cover.

Given any cover C

(a) if redundant, not minimum

(b) if any cube q is not prime, replace q with prime $p \supseteq q$ and continue until all cubes prime; it is a minimum prime cover

Example Covers

X_1	X_2	X_3	Y_1
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	2

0 0 -
1 0 - is a cover. Is it prime?
1 1 - Is it irredundant?

What is a minimum prime and irredundant cover for the function?

Example Covers

X_1	X_2	X_3	Y_1
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	2

0 0 -
1 0 - is a cover. Is it prime?
1 1 - Is it irredundant?

- 0 -
1 1 - is a cover. Is it prime?
Is it irredundant?
Is it minimum?

Example Covers

X_1	X_2	X_3	Y_1
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	2

0 0 -
1 0 - is a cover. Is it prime?
1 1 - Is it irredundant?

- 0 -
1 1 - is a cover. Is it prime?
Is it irredundant?
Is it minimum?

What about
- 0 -
1 - -

Checking for Prime and Irredundant

We will use Shannon (Boole's) Cofactor and Tautology Checking!

- Let $f : B^n \rightarrow B$ be a Boolean function, and $x = (x_1, x_2, \ldots, x_n)$ the variables in the support of f. The cofactor f_a of f by a literal $a=x_i$ or $a=x_i'$ is

$$f_{x_i}(x_1, x_2, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$$

$$f_{x_i'}(x_1, x_2, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$$

- Tautology: find a truth assignment to the inputs making a given Boolean formula false

Shannon (Boolean) Cofactor

The cofactor f_C of f by a cube C is f with the fixed values indicated by the literals of C, e.g. if $C=x_i x_j'$, then $x_i=1$, and $x_j=0$.

If $C= x_1 x_4' x_6$, f_C is just the function f restricted to the subspace where $x_1=x_6=1$ and $x_4=0$.

As a function, f_C does not depend on x_1, x_4 or x_6

(However, we still consider f_C as a function of all *n* variables, it just happens to be independent of x_1, x_4 and x_6).

$x_1 f \neq f_{x_1}$

Example: $f= ac + a'c'$, $af = ac$, $f_a=c$

Cofactor and Quantification

Let $f : B^n \rightarrow B$ be a Boolean function, and $x= (x_1, x_2, \ldots, x_n)$ the variables in the support of f.

- Positive cofactor $f_{x_i} (x_1, x_2, \ldots, x_n) = f (x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$
- Negative cofactor $f_{x_i'} (x_1, x_2, \ldots, x_n) = f (x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$
- Existential quantification over variable x_i : $\exists x_i.f = f_{x_i} \vee f_{x_i'}$
- Universal quantification over variable x_i : $\forall x_i.f = f_{x_i} \wedge f_{x_i'}$

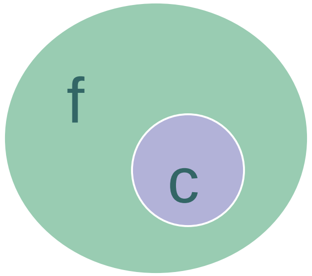
Fundamental Theorem

Theorem 1 Let c be a cube and f a function. Then $c \subseteq f \Leftrightarrow f_c \equiv 1$.

Proof. We use the fact that $xf_x = xf$, and f_x is independent of x.

If: Suppose $f_c \equiv 1$. Then $cf=f_c c=c$. Thus,

$c \subseteq f$.



Proof (contd)

Only if. Assume $c \subseteq f$

Then $c \subseteq cf = cf_c$. If $f_c \neq 1$, then $\exists\, m \in B^n$, $f_c(m)=0$.

Find m^: Let $m_i\hat{} = m_i$, if $x_i \notin c$ and $x_i' \notin c$.

or if $m_i=0$, $x_i' \in c$

or $m_i=1$, $x_i \in c$.

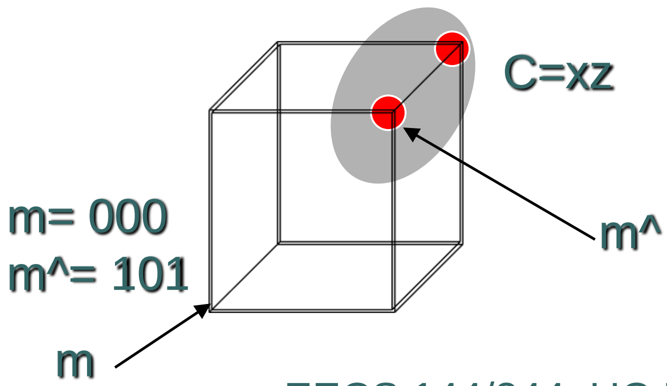
$m_i\hat{} = m_i'$ otherwise.

i.e. we make the literals of m^ agree with c, i.e. $m\hat{} \in c$. But then $f_c(m\hat{}) = f_c(m) = 0$,(f_c is independent of literals $l \in c$)

Hence, $c(m\hat{})=1$

and $f_c(m\hat{})\, c(m\hat{}) = 0$,

contradicting $c \subseteq cf_c$.



Checking for Prime and Irredundant

- Let $G=\{c_i\}$ be a cover of $F=(f_{ON}, f_{DC}, f_{OFF})$. Let D be a cover for f_{DC}.

$c_i \subseteq G$ is redundant iff

$$c_i \subseteq (G \backslash \{c_i\}) \cup D \equiv G^i \qquad (1)$$

(Since $c_i \subseteq G^i$ and $f_{ON} \subseteq G \subseteq f_{ON} + f_{DC}$ then $c_i \subseteq c_i\, f_{ON} + c_i\, f_{DC}$ and $c_i\, f_{ON} \subseteq G\backslash\{c_i\}$. Thus $f_{ON} \subseteq G\backslash\{c_i\}$.)

Checking for Prime and Irredundant

- Let $G=\{c_i\}$ be a cover of $F=(f_{ON}, f_{DC}, f_{OFF})$. Let D be a cover for f_{DC}.

$c_i \subseteq G$ is redundant iff

$$c_i \subseteq (G \backslash \{c_i\}) \cup D \equiv G^i \qquad (1)$$

(Since $c_i \subseteq G^i$ and $f_{ON} \subseteq G \subseteq f_{ON} + f_{DC}$ then $c_i \subseteq c_i f_{ON} + c_i f_{DC}$ and $c_i f_{ON} \subseteq G \backslash \{c_i\}$. Thus $f_{ON} \subseteq G \backslash \{c_i\}$.)

- A literal $l \in c_i$ is prime if $(c_i \backslash \{ l \})$ ($= (c_i)_l$) is not an implicant of F.

A cube c_i is a prime of F iff all literals $l \in c_i$ are prime.

Literal $l \in c_i$ is not prime $\Leftrightarrow (c_i)_l \subseteq f_{ON} + f_{DC} \qquad (2)$

Note: Both tests (1) and (2) can be checked by tautology:

1) $(G^i)_{c_i} \equiv 1$ (implies c_i redundant)
2) $(F \cup D)_{(c_i)_l} \equiv 1$ (implies l not prime)

Tautology Checking

F = acd + bcd + a’bd’ + a’c’d’ +c’d + ac’+ ad’ + b’cd’ + a’b’d + a’b’c

Is F = 1? NOT EASY!!!

F=

1211
2111
0120
0200
2201
1202
1220
2010
0021
0012

== 1?

List of Cubes (Cover Matrix)

We often use a matrix notation to represent a cover:

Example: $F = ac + \bar{c}d =$

	a	b	c	d		a	b	c	d
$ac \rightarrow$	1	2	1	2	or	1	-	1	-
$\bar{c}d \rightarrow$	2	2	0	1		-	-	0	1

Each row represents a cube

1 means that the positive literal appears in the cube

0 means that the negative literal appears in the cube

The 2 (or -) here represents that the variable does not appear in the cube. It implicitly represents both 0 and 1 values.

Operations on Lists of Cubes

AND operation:

- take two lists of cubes
- computes pair-wise AND between individual cubes and put result on new list
- represent cubes as pairs of computer words
- set operations are implemented as bit-vector operations

```
Algorithm AND(List_of_Cubes C₁,List_of_Cubes C₂) {
  C = ∅
  foreach c₁ ∈ C₁ {
    foreach c₂ ∈ C₂ {
      c = c₁ ∩ c₂
      C = C ∪ c
    }
  }
  return C
}
```

Operations on Lists of Cubes

OR operation:

- take two lists of cubes
- computes union of both lists

Naive implementation:

```
Algorithm OR(List_of_Cubes C1, List_of_Cubes C2) {
  return C1 ∪ C2
}
```

On-the-fly optimizations:

- remove cubes that are completely covered by other cubes
  - complexity is $O(m^2)$; m is length of list
- merge adjacent cubes
- remove redundant cubes?
  - complexity is $O(2^n)$; n is number of variables
  - too expensive for non-orthogonal lists of cubes

Operation on Lists of Cubes

Naive implementation of COMPLEMENT operation

- apply De'Morgan's law to SOP
- complement each cube and use AND operation
- Example:

Input		non-orth.		orthogonal
01-10	=>	1----	=>	1----
		-0---		00---
		---0-		01-0-
		----1		01-11

Naive implementation of TAUTOLOGY check

- complement function using the COMPLEMENT operator and check for emptiness

Generic Tautology Check

```
Algorithm CHECK_TAUTOLOGY(List_of_Cubes C) {
  if(C == ∅)       return FALSE;
  if(C == {-...-})return TRUE; // cube with all '-'
  xi = SELECT_VARIABLE(C)
  C0 = COFACTOR(C, xi')
  if(CHECK_TAUTOLOGY(C0) == FALSE) {
     print xi = 0
     return FALSE;
  }
  C1 = COFACTOR(C,xi)
  if(CHECK_TAUTOLOGY(C1) == FALSE) {
     print xi = 1
     return FALSE;
  }
  return TRUE;
}
```

Improvements

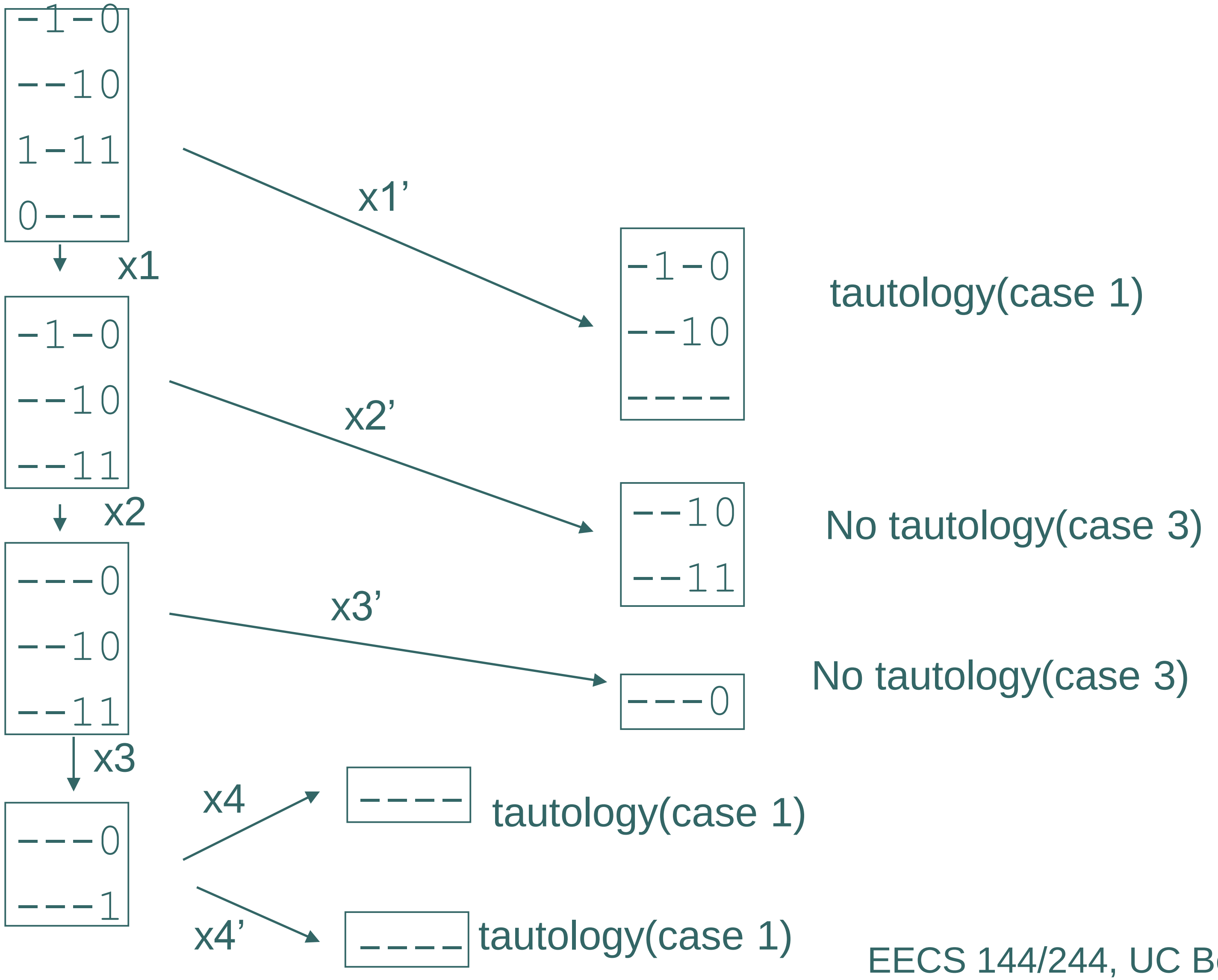
Variable ordering:

- pick variable that minimizes the two sub-cases ("-"s get replicated into both cases)

Quick decision at leaf:

- return TRUE if C contains at least one complete "-" cube among others (case 1)
- return FALSE if number of minterms in onset is < 2^n (case 2)
- return FALSE if C contains same literal in every cube (case 3)

Example



-1-0
--10
1-11
0---
x1
x1'
-1-0
--10
----
tautology(case 1)
-1-0
--10
--11
x2
x2'
--10
--11
No tautology(case 3)
---0
--10
--11
x3
x3'
---0
No tautology(case 3)
---0
---1
x4
----
tautology(case 1)
x4'
----
tautology(case 1)

The Quine-McCluskey Method: Exact Minimization

Given G' and D (covers for $F=(f_{ON}, f_{DC}, f_{OFF})$.
and f_{DC}), find a minimum cover G of primes where:

$$f \subseteq G \subseteq f_{ON} + f_{DC} \text{ (G is a prime cover of } F \text{)}$$

Step 1: List all minterms in ON-SET and DC-SET

Step 2: Use a prescribed sequence of steps to find all the prime implicants of the function

Step 3: Construct the prime implicant table

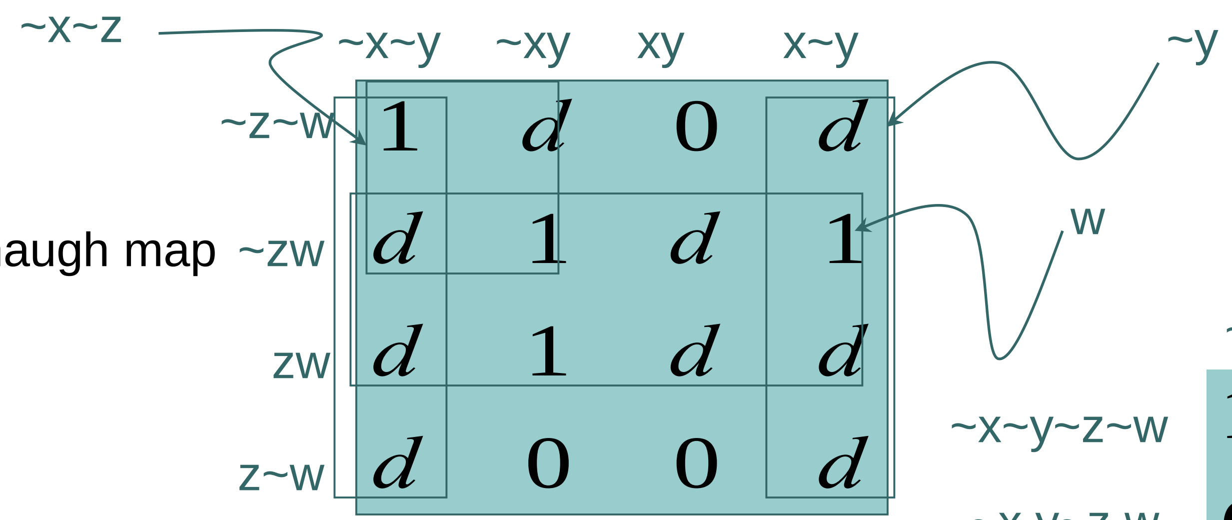
Step 4: Find a minimum set of prime implicants that cover all the minterms

Example

$$F = \overline{x}\,\overline{y}\,\overline{z}\,\overline{w} + \overline{x}y\overline{z}w + x\overline{y}\,\overline{z}w + \overline{x}yzw$$

$$D = \overline{y}z + xyw + \overline{x}\,\overline{y}\,\overline{z}w + x\overline{y}\,\overline{w} + \overline{x}y\overline{z}\,\overline{w}$$

Karnaugh map



	~y	w	~x~z
~x~y~z~w	1	0	1
~x y~z w	0	1	1
x~y~z w	1	1	0
~x y z w	0	1	0

Primes: ~y + w + ~x~z

Covering Table

Solution: {1,2} ⇒ ~y + w is minimum prime cover. (also w+ ~x~z)

Generating Primes - single output func.

Tabular method

(based on *consensus* operation):

Start with all minterm canonical form of *F*

Group *pairs* of adjacent minterms into cubes

Repeat merging cubes until no more merging possible; mark (√) + remove all covered cubes.

Result: set of *primes* of *f.*

Example:

$$F = x'y' + wxy + x'yz' + wy'z$$

$F = x'y' + wxy + x'yz' + wy'z$

$w'x'y'z'$ √	$w'x'y'$ √ $w'x'z'$ √ $x'y'z'$ √	$x'y'$ $x'z'$
$w'x'y'z$ √ $w'x'yz'$ √ $wx'y'z'$ √ $wx'y'z$ √	$x'y'z$ √ $x'yz'$ √ $wx'y'$ √ $wx'z'$ √ $wy'z$	
$wx'yz'$ √ $wxyz'$ √ $wxy'z$ √	wyz' wxy wxz	
$wxyz$ √		

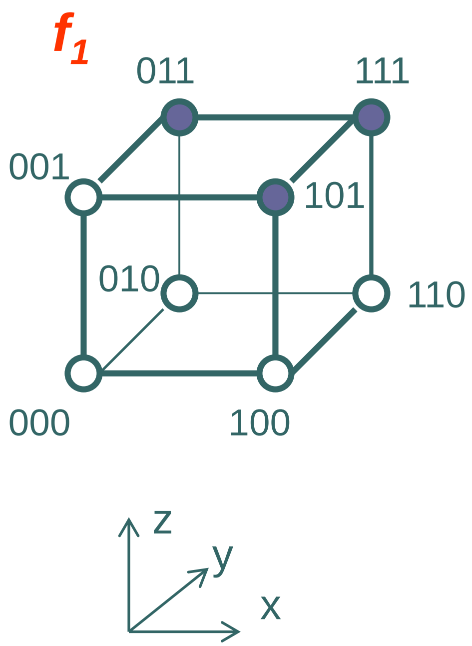
Courtesy: Maciej Ciesielski, UMASS

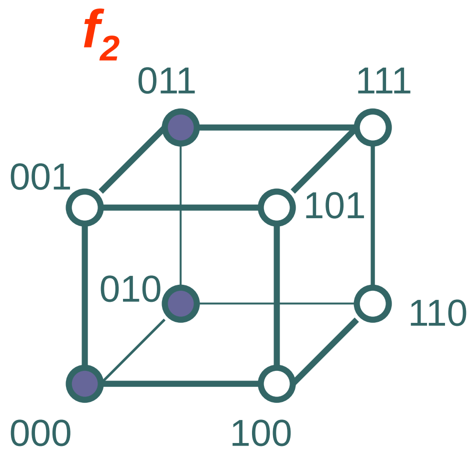
Generating Primes – multiple outputs

Procedure similar to *single-output* function, except:

- include also the primes of the products of individual functions

Example:





$x\,y\,z$	$f_1\,f_2$
0 – 0	0 1
0 1 1	1 1
1 – 1	1 0

Can also represent it as:

$x\,y\,z$	$f_1\,f_2$
0 – 0	0 1
0 1 –	0 1
– 1 1	1 0
1 – 1	1 0

Generating Primes - example

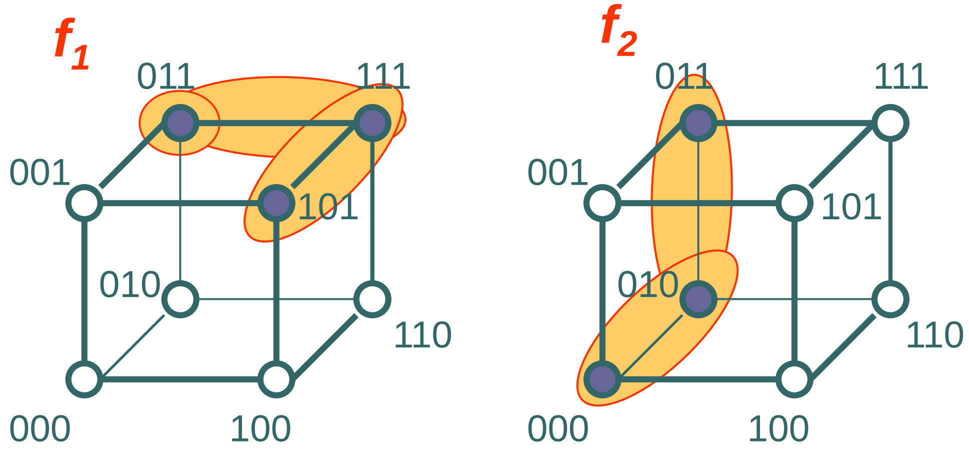
Modification (w.r.t single output function):

- When two adjacent implicants are merged, the output parts are intersected

x y z	f_1 f_2
0 – 0	0 1
0 1 1	1 1
1 – 1	1 0

000 | 01 √	0 – 0 | 01
010 | 01 √	0 1 – | 01
011 | 11	– 1 1 | 10
101 | 10 √	1 – 1 | 10
111 | 10 √	

There are five primes listed for this two-output function.
- What is the min cover ?



Minimize multiple-output cover - example

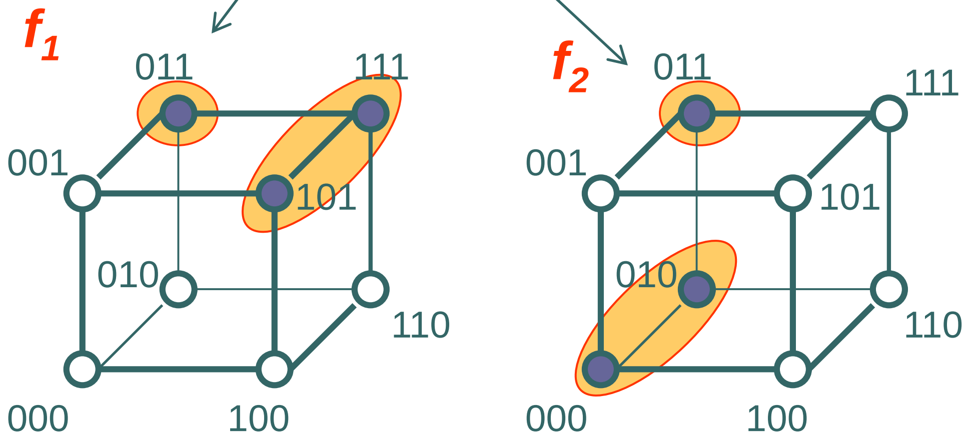
List multiple-output primes

- Create a covering table, solve

p_1 = 0 1 1 | 11
p_2 = 0 – 0 | 01
p_3 = 0 1 – | 01
p_4 = – 1 1 | 10
p_5 = 1 – 1 | 10

	p_1	p_2	p_3	p_4	p_5
000 | 01	0	1	0	0	0
010 | 01	0	1	1	0	0
011 | 01	1	0	1	0	0
011 | 10	1	0	0	1	0
101 | 10	0	0	0	0	1
111 | 10	0	0	0	1	1

listed twice



Min cover has 3 primes:
F = { p_1, p_2, p_5 }

Covering Table

Primes of f+d

Minterms of f

	~y	w	~x~z
~x~y~z~w	1	0	1
~xy~zw	0	1	1
X~y~zw	1	1	0
~xyzw	0	1	0

Row singleton (essential minterm)

Essential prime

Definition: An essential prime is any prime that uniquely covers a minterm of f.

Row and Column Dominance

Definition: A row i_1 whose set of primes is contained in the set of primes of row i_2 is said to dominate i_2.

Example:

i_1	011010
i_2	011110

i_1 dominates i_2

We can remove row i_2, because we have to choose a prime to cover i_1, and any such prime also covers i_2. So i_2 is automatically covered.

Row and Column Dominance

Definition: A column j_1 whose rows are a superset of another column j_2 is said to **dominate** j_2.

Example:

j_1 dominates j_2

j_1	j_2
1	0
0	0
1	1
0	0
1	1

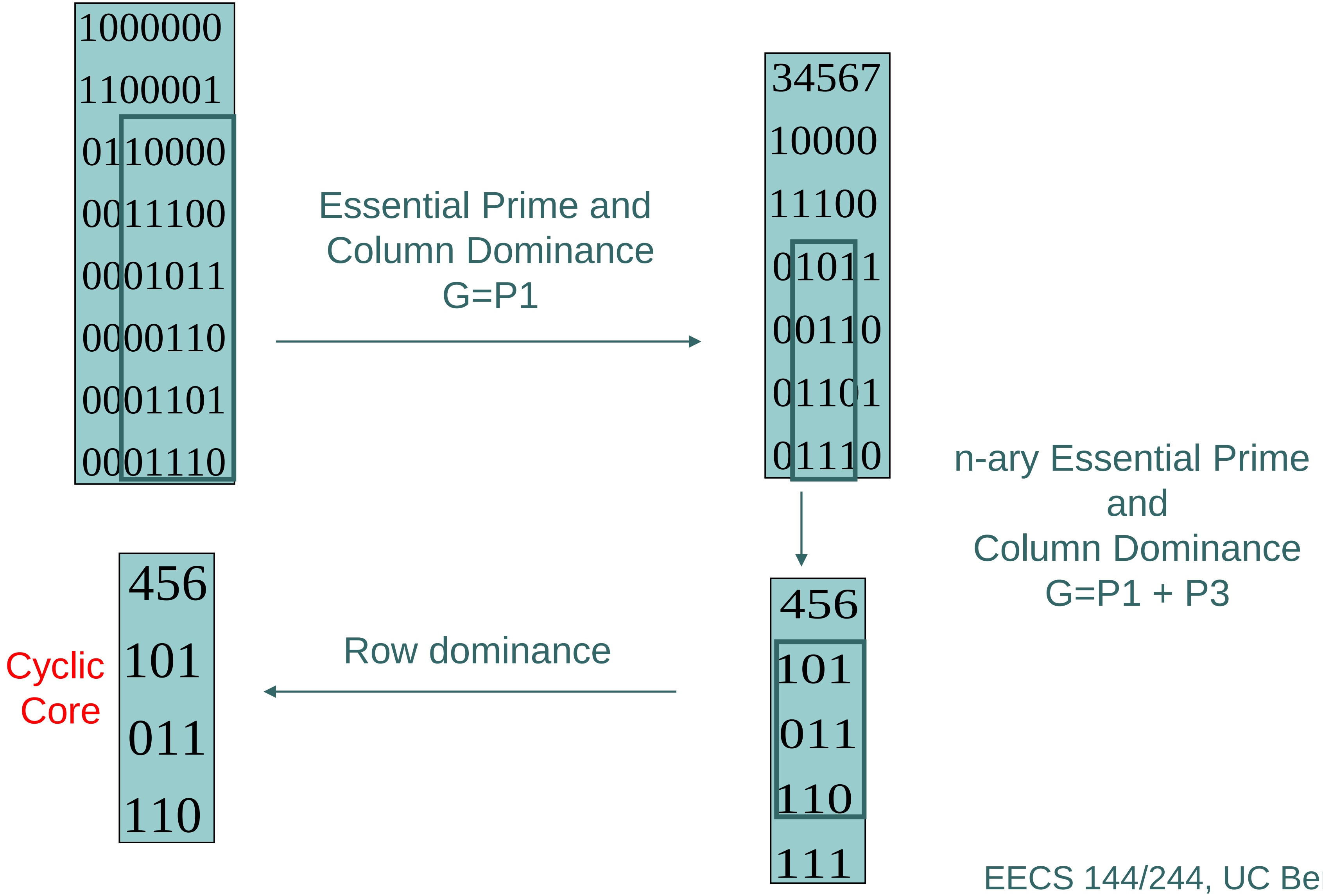
We can remove column j_2 since j_1 covers all those rows and more. We would never choose j_2 in a minimum cover since it can always be replaced by j_1.

Pruning the Covering Table

1. Remove all rows covered by essential primes (columns in row singletons). Put these primes in the cover G.
2. Group identical rows together and remove dominated rows.
3. Remove dominated columns. For equal columns, keep one prime to represent them.
4. Newly formed row singletons define n-ary essential primes.
5. Go to 1 if covering table decreased.

The resulting reduced covering table is called the cyclic core. This has to be solved (unate covering problem). A minimum solution is added to G - the set of n-ary essential primes. The resulting G is a minimum cover.

Example


1000000
1100001
0110000
0011100
0001011
0000110
0001101
0001110
Essential Prime and
Column Dominance
G=P1
34567
10000
11100
01011
00110
01101
01110
n-ary Essential Prime
and
Column Dominance
G=P1 + P3
456
101
011
110
111
Row dominance
Cyclic
Core
456
101
011
110

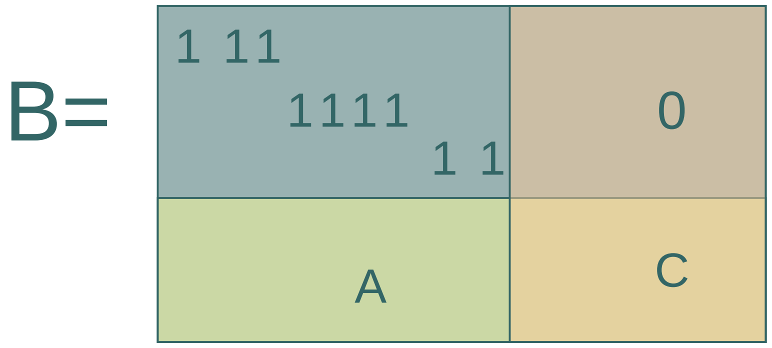
Solving the Cyclic Core

Best known method (for unate covering) is branch and bound with some clever bounding heuristics.

Independent Set Heuristic:

Find a maximum set of "independent" rows I. Two rows B_{i_1}, B_{i_2} are independent if $\nexists j$ such that $B_{i_1 j} = B_{i_2 j} = 1$. (They have no column in common)

Example: Covering matrix B rearranged with independent sets first.

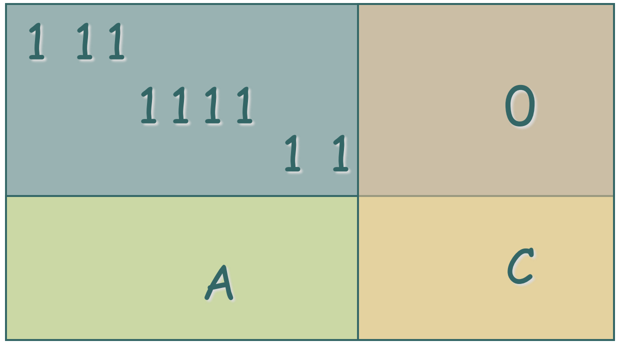


Independent set of rows = I

Solving the Cyclic Core

Lemma:

|Solution of Covering| ≥ |I|



Heuristic

Let $I=\{I_1, I_2, \ldots, I_k\}$ be the independent set of rows

choose $j \in I_i$ which covers the most rows of A. Put $j \rightarrow J$

eliminate all rows covered by column j

$I \leftarrow I \backslash \{I_i\}$

go to 1 if $|I| > 0$

If B is empty, then done (in this case we have the guaranteed minimum solution - IMPORTANT)

If B is not empty, choose an independent set of B and go to 1

Espresso Algorithm: Heuristic Minimization

ESPRESSO (f_{ON}, f_{DC}) {

F is ON-SET, DC is Don't Care Set

1. $R = U - (F \cup DC)$ U is universe cube
2. $n = |F|$
3. F = *Reduce* (F, DC); // reduce implicants in F to non-prime cubes
4. F = *Expand* (F, R); // expand cubes to prime implicants
5. F = *Irredundant* (F, DC); // extract minimal cover of prime implicants
6. If $|F| < n$ goto 2, else, post-process & exit

}

Bibliography

- https://webdocs.cs.ualberta.ca/~amaral/courses/329/webslides/Topic5-QuineMcCluskey/sld001.htm
- R.K. Brayton, C. McMullen, G.D. Hachtel and A. Sangiovanni-Vincentelli, Logic Minimization Algorithms for VLSI Synthesis. Kluwer Academic Publishers, 1984.