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

- RTL Synthesis
- HDL
- HDL Simulation/Verification
- Library/module generators
- Boolean circuit/network
- Graph / Rectangles
- Layout
- Physical design
- Logic optimization
- Netlist
- RTL Synthesis
- FSM, Verilog, VHDL

K. Keutzer
Sequential v.s. Combinational Synthesis/Logic Optimization

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

- Logic optimization
  - netlist
  - tech independent
  - tech dependent
  - 2-level Logic opt
  - multilevel Logic opt

- Library
- Generic Library
- Real Library

EECS 144/244, UC Berkeley: 4
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

$f: B^n \rightarrow Y^m$
Definition used in Logic Optimization

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 $\text{ff}$ (or ‘Boolean’ function, switching function) in $n$ inputs and $m$ outputs is a map $\text{ff}: B^n \rightarrow Y^m$
The Boolean $n$-Cube, $B^n$

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

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

\( x_1, x_2, \ldots \) 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 \( f^1 \)
A literal is a variable or its negation \( y, y' \). It represents a logic function.

Literal \( x_1 \) represents the logic function \( f = \{ x \mid x_1 = 1 \} \)

Literal \( x_1' \) represents logic function \( g = \{ x \mid x_1 = 0 \} \)
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

\[ f = x_1 X x_2' + x_1' X x_2 = (x_1 + x_2) X (x_1' + x_2') \]
\[ h = a + b X c = (a' X (b' + c'))' \]

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

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

There are $2^{2^n}$ 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

\[
\text{ff: } B^3 \rightarrow Y^2
\]

<table>
<thead>
<tr>
<th>(X_1)</th>
<th>(X_2)</th>
<th>(X_3)</th>
<th>(Y_1)</th>
<th>(Y_2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

\(Y_1:\)  
ON-SET\(_1\) = \{000, 001, 100, 101, 110\}  
OFF-SET\(_1\) = \{010, 011\}  
DC-SET\(_1\) = \{111\}
Operations on Logic Functions

(1) Complement: \( f \rightarrow \overline{f} (\neg f \text{ or } f') \)
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)

\[ C = xy' \text{ is a cube} \]

\[ C = (x=1)(y=0) \]
2-level Minimization: Minimizing SOP (DNF)

\[ F_1 = \overline{A} \overline{B} + \overline{A} \overline{B} D + \overline{A} B \overline{C} \overline{D} \\
+ A B C D + A B + A B D \]

\[ F_1 = B + D + A C + A C \] (minimum representation)

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 - -</td>
<td>1</td>
</tr>
<tr>
<td>0 1 - 1</td>
<td>1</td>
</tr>
<tr>
<td>0 1 0 0</td>
<td>1</td>
</tr>
<tr>
<td>1 1 1 0</td>
<td>1</td>
</tr>
<tr>
<td>1 0 - -</td>
<td>1</td>
</tr>
<tr>
<td>1 1 - 1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>- 0 - -</td>
<td>1</td>
</tr>
<tr>
<td>- - - 1</td>
<td>1</td>
</tr>
<tr>
<td>0 0 - 0</td>
<td>1</td>
</tr>
<tr>
<td>1 - 1 -</td>
<td>1</td>
</tr>
</tbody>
</table>

(number of cubes, literals)
PLA’s - Multiple Output Functions

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

$n=3, m=3$

Personality Matrix

<table>
<thead>
<tr>
<th>$abc$</th>
<th>$f_1f_2f_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>10-</td>
<td>1 - -</td>
</tr>
<tr>
<td>-11</td>
<td>1 - -</td>
</tr>
<tr>
<td>0-0</td>
<td>- 1 -</td>
</tr>
<tr>
<td>111</td>
<td>- 1 1</td>
</tr>
<tr>
<td>00-</td>
<td>- - 1</td>
</tr>
</tbody>
</table>
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 $f$

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

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$Y_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

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

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 $F \setminus \{c_i\} \not\supset f$

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

Not covered

$F \setminus \{ab\} \not\supset f$
Example Covers

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$Y_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>0 0 1</td>
<td>0 1 0</td>
<td>0 1 1</td>
</tr>
<tr>
<td></td>
<td>0 1 1</td>
<td>1 0 0</td>
<td>1 0 1</td>
</tr>
<tr>
<td></td>
<td>1 1 0</td>
<td>1 1 1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0 0 -</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 0 -</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 1 -</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$Y_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>0 0 1</td>
<td>0 1 0</td>
<td>0 1 1</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$Y_1$</th>
<th>00 -</th>
<th>10 -</th>
<th>11 -</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td>is a cover.</td>
<td>Is it prime?</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td>Is it irredundant?</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td>-0 -</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td></td>
<td></td>
<td>Is it minimum?</td>
</tr>
</tbody>
</table>
### Example Covers

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$Y_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

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

- - 0 - is a cover. Is it prime?
- - 1 1 - Is it irredundant?
- - 1 1 - 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 $f : \mathbb{B}^n \to \mathbb{B}$ 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 $x_i : \exists x_i. f = f_{x_i} \lor f_{x_i'}$
- Universal quantification over variable $x_i : \forall x_i. f = f_{x_i} \land f_{x_i'}$
Fundamental Theorem

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

**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 $\exists m \in \mathbb{B}^n, f_c(m) = 0$.
Find $m^\wedge$: Let $m_i^\wedge = m_i$, if $x_i \not\in c$ and $x_i' \not\in c$.
    or if $m_i = 0$, $x_i ' \in c$
    or $m_i = 1$, $x_i \in c$.

$m_i^\wedge = m_i'$ otherwise.
i.e. we make the literals of $m^\wedge$ agree with $c$, i.e. $m^\wedge \in c$.
    But then $f_c(m^\wedge) = f_c(m) = 0$, ($f_c$ is independent of literals $l \in c$)
Hence, $c(m^\wedge) = 1$
and $f_c(m^\wedge) c(m^\wedge) = 0$,
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

$$c_i \subseteq (G\{c_i\}) \cup D \equiv G_i$$  \hspace{1cm} (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 $c_i f_{ON} \subseteq G\{c_i\}$. Thus $f_{ON} \subseteq G\{c_i\}$.)
Checking for Prime and Irredundant

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

$c_i \subseteq G$ is redundant iff

$$c_i \subseteq (G \setminus \{c_i\}) \cup D \equiv G^i$$

(Since $c_i \subseteq G^i$ and $f_{\text{ON}} \subseteq G \subseteq f_{\text{ON}} + f_{\text{DC}}$ then $c_i \subseteq c_i f_{\text{ON}} + c_i f_{\text{DC}}$

and $c_i f_{\text{ON}} \subseteq G \setminus \{c_i\}$. Thus $f_{\text{ON}} \subseteq G \setminus \{c_i\}$.)

• A literal $l \in c_i$ is prime if $(c_i \setminus \{l\}) \ (= (c_i)_l)$ 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 $\iff (c_i)_l \subseteq f_{\text{ON}} + f_{\text{DC}}$ (2)
Note: Both tests (1) and (2) can be checked by tautology:

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

\[ F = acd + bcd + a'b'd' + a'c'd' + c'd + ac' + ad' + b'cd' + a'b'd + a'b'c \]

Is \( F = 1 \)? NOT EASY!!!
List of Cubes (Cover Matrix)

We often use a matrix notation to represent a cover:

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

\[
\begin{array}{c|cccc}
  & a & b & c & d \\
\hline
ac & 1 & 2 & 1 & 2 \\
\hline
\bar{c}d & 2 & 2 & 0 & 1 \\
\end{array}
\]

or

\[
\begin{array}{c|cccc}
  & a & b & c & d \\
\hline
1 & - & - & 1 & - \\
\hline
- & 0 & 1 & - & - \\
\end{array}
\]

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 \textbf{AND}(\text{List\_of\_Cubes } C_1, \text{List\_of\_Cubes } C_2) \{ \\
\quad C = \emptyset \\
\quad \text{foreach } c_1 \in C_1 \{ \\
\quad \quad \text{foreach } c_2 \in C_2 \{ \\
\quad \quad \quad c = c_1 \cap c_2 \\
\quad \quad \quad C = C \cup c \\
\quad \quad \} \\
\quad \} \\
\quad \text{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 \cup C₂
}

On-the-fly optimizations:
- remove cubes that are completely covered by other cubes
  - complexity is O(m²); m is length of list
- merge adjacent cubes
- remove redundant cubes?
  - complexity is O(2ⁿ); n is number of variables
  - too expensive for non-orthogonal lists of cubes
Naive implementation of COMPLEMENT operation

- apply De’Morgan’s law to SOP
- complement each cube and use AND operation

Example:

<table>
<thead>
<tr>
<th>Input</th>
<th>non-orth.</th>
<th>orthogonal</th>
</tr>
</thead>
<tbody>
<tr>
<td>01-10  =&gt; 1----</td>
<td>=&gt; 1----</td>
<td></td>
</tr>
<tr>
<td>-0----</td>
<td>00----</td>
<td></td>
</tr>
<tr>
<td>-0-0-</td>
<td>01-0-</td>
<td></td>
</tr>
<tr>
<td>----1</td>
<td>01-11</td>
<td></td>
</tr>
</tbody>
</table>
Generic Tautology Check

Algorithm CHECK_TAUTOLOGY(List_of_Cubes C) {
    if(C == ∅) return FALSE;
    if(C == {...}) return TRUE; // cube with all ‘-’
    $x_i = \text{SELECT_VARIABLE}(C)$
    $C_0 = \text{COFACTOR}(C, x_i')$
    if(CHECK_TAUTOLOGY($C_0$) == FALSE) {
        print $x_i = 0$
        return FALSE;
    }
    $C_1 = \text{COFACTOR}(C, x_i)$
    if(CHECK_TAUTOLOGY($C_1$) == FALSE) {
        print $x_i = 1$
        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

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)

No tautology (case 3)
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:

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

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

\[
F = xyzw + xyzw + xyzw + xyzw \\
D = yz + yzw + yzw + yzw + yzw
\]

Primes: \( \sim y + w + \sim x \sim z \)

Covering Table
Solution: \( \{1, 2\} \Rightarrow \sim y + w \) is minimum prime cover. (also \( w + \sim x \sim z \))
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:

\[
F = x' y' + wx y + x' yz' + wy'z
\]

<table>
<thead>
<tr>
<th>( w'x'y'z' )</th>
<th>✓</th>
<th>( w'x'y'z' )</th>
<th>✓</th>
<th>( x'y'z' )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( w'x'y'z )</td>
<td>✓</td>
<td>( x'y'z' )</td>
<td>✓</td>
<td>( x'yz' )</td>
</tr>
<tr>
<td>( wx'y'z' )</td>
<td>✓</td>
<td>( wx'y'z' )</td>
<td>✓</td>
<td>( wx'z' )</td>
</tr>
<tr>
<td>( wx'y'z )</td>
<td>✓</td>
<td>( wx'y'z )</td>
<td>✓</td>
<td>( wx'z )</td>
</tr>
<tr>
<td>( wx'yz )</td>
<td>✓</td>
<td>( wx'yz )</td>
<td>✓</td>
<td>( wx )</td>
</tr>
<tr>
<td>( wxyz' )</td>
<td>✓</td>
<td>( wxyz' )</td>
<td>✓</td>
<td>( wxy )</td>
</tr>
</tbody>
</table>

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:

Can also represent it as:
Generating Primes - example

Modification (w.r.t single output function):
- When two adjacent implicants are merged, the output parts are intersected

<table>
<thead>
<tr>
<th>x y z</th>
<th>$f_1$</th>
<th>$f_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>0 1</td>
<td></td>
</tr>
<tr>
<td>0 1 1</td>
<td>1 1</td>
<td></td>
</tr>
<tr>
<td>1 1</td>
<td>1 0</td>
<td></td>
</tr>
</tbody>
</table>

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

\[
\begin{align*}
p_1 &= 0 1 1 | 11 \\
p_2 &= 0 - 0 | 01 \\
p_3 &= 0 1 - | 01 \\
p_4 &= - 1 1 | 10 \\
p_5 &= 1 - 1 | 10
\end{align*}
\]

Min cover has 3 primes:
\[F = \{ p_1, p_2, p_5 \}\]
**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:

\[
\begin{array}{c}
  i_1 & 011010 \\
  i_2 & 011110 \\
\end{array}
\]

$i_1$ 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:

\[
\begin{array}{cc}
  j_1 & j_2 \\
  1 & 0 \\
  0 & 0 \\
  1 & 1 \\
  0 & 0 \\
  1 & 1 \\
\end{array}
\]

$j_1$ dominates $j_2$

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 $j_1$. 

EECS 144/244, UC Berkeley: 54
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 | 34567 |
| 1100001 | 10000 |
| 0110000 | 11100 |
| 0011100 | 01011 |
| 0001011 | 00110 |
| 0000110 | 01101 |
| 0001101 | 11100 |
| 0001110 | 10000 |

Essential Prime and Column Dominance
$G = P_1$

Cyclic Core

Row dominance

| 456 |
| 101 |
| 011 |
| 110 |

| 456 |
| 101 |
| 011 |
| 110 |

| 111 |

n-ary Essential Prime and Column Dominance
$G = P_1 + P_3$

EECS 144/244, UC Berkeley: 56
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 $\exists j$ such that $B_{i_1j} = B_{i_2j} = 1$. (They have no column in common)

Example: Covering matrix $B$ rearranged with independent sets first.

\[ B = \begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 0 \\
A \\
C
\end{bmatrix} \]

Independent set  = I
of rows
Solving the Cyclic Core

Lemma:

$|\text{Solution of Covering}| \geq |I|$
**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 \( j \)

\( I \leftarrow I \setminus \{I_i\} \)

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 \( (f_{\text{ON}}, f_{\text{DC}}) \)  

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

- https://webdocs.cs.ualberta.ca/~amaral/courses/329/webslides/Topic5-QuineMcCluskey/sld001.htm