MULTIPLE-LEVEL LOGIC OPTIMIZATION

Giovanni De Micheli
Stanford University

Outline

- Algebraic methods.
- Rule-based methods.
- Taxonomy of optimization methods:
  - Goals: area/delay.
  - Algorithms: algebraic/Boolean.
- Representations.

GDM
Motivation

Circuit modeling

- Structural model.
  - Interconnection of logic gates.

  Bound (mapped) networks:

- Hybrid structural/behavioral model.
  - Interconnection of logic functions.

  Logic network:

Motivation

- Applicable to a variety of designs.
  • Better performance.
  • More flexibility.

- Gates versus macros (PLAs):

- Semi-custom libraries:
  • Multiple-level networks:
\[ n = z \]
\[ i = h \]
\[ s = x \]
\[ a = m \]
\[ p^p + p^q + pq + p^r = a \]
\[ p + q + c + d = n \]
\[ e + pq + cq + pq + cd = i \]
\[ f + s = a \]
\[ p + d = i \]
\[ q + v = b \]
\[ ep + ce = d \]
\[
\begin{bmatrix}
\varphi + q + a \\
pq + c + q + pv + \varphi q \\
p + c + q + p + q v \\
v p + p c + pq + qv
\end{bmatrix} = 1
\]
Network optimization

Minimize area estimate:
- subject to delay constraints.

Minimize maximum delay:
- subject to area constraints.

Maximize testability.

Minimize power.

Estimation:
- Sensitizable paths.
- Refined gate delay models.
- Number of stages.

Delay:
- Number of functions/gates.
- Number of literals.

Area:

- Sensitizable paths.
- Refined gate delay models.
- Number of stages.

Delay:
- Number of functions/gates.
- Number of literals.
**Problem analysis**

Multiple-level optimization is hard.

**Exact methods:**
- Exponential complexity.
- Impractical.

**Approximate methods:**
- Heuristic algorithms.
- Rule-based methods.

**Strategies for optimization:**

- Selection and order of transformations.
- Types of transformations.
- Methods differ in:
  - Preserve network behavior.
  - Circuit transformations.
  - Modify parts of the network one at a time.

- Improve circuit by step-wise transformations.

<table>
<thead>
<tr>
<th>Selection and order of transformations</th>
<th>Types of transformations</th>
<th>Methods differ in:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preserve network behavior</td>
<td>Circuit transformations</td>
<td>Modify parts of the network one at a time.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Improve circuit by step-wise transformations.</td>
</tr>
</tbody>
</table>
Perform variable substitution.

Example:

\[ f + p + d = s \iff - \]

\[ p + d = j \iff f + j = s \iff - \]

Eliminate one function from the network.

Example
Example of decomposition:

\[ \rho v + p \ell = a \]

\[ \rho + q + p = \ell \]

\[ \rho v + p \rho + pq + pr = a \]

Example:

- Break one function into smaller ones.
- Introduce new vertices in the network.
Example extraction

1. Find a common sub-expression of two or more expressions.
2. Extract sub-expression as new function.
3. Introduce new vertex in the network.

Example:

\[ r = p + a' \]
\[ s = r + b' \]
\[ t = ac + ad + bc + bd + e \]
\[ u = q'c + qc' + qc \]
\[ q = a + b \]

Example:

\[ x + y + qa = t \]
\[ x = d \]
\[ p + c = q \]

Example:

\[ (q + v)(p + c) = t \]
\[ (p + c) = d \]
\[ x + p q + q e + p q + q e + p q + a c + d e + c e = d \]

Example:

Introduce new vertex in the network.

Extract sub-expression as new function.

(or more) expressions.

Find a common sub-expression of two

Example extraction

Example
Example simplification

- $c + b = n \iff \neg c \land \neg b \land \neg n$
- $u = q'c + qc' + qc \land q = a + b$
- $p = ce + de$
- $w = a'd + bd + c'd + ae'$

- Example:
  - Simplify a local function.
Simplify a local function by using an additional input that was not previously in its support set.

Example:

\[ r = p + a' \]
\[ s = r + b' \]
\[ q = a + b \]
\[ u = q'c + qc' + qc \]
\[ k = c + d \]
\[ t = ka + kb + e \]
\[ p = ke \]
\[ v = a'd + bd + c'd + ae' \]
**Optimization approaches**

- **Algorithmic approach:**
  - Define an algorithm for each transformation type.
  - Algorithm is an operator on the network.

- **Rule-based approach:**
  - Rule-base: Set of pattern pairs.
  - Define an algorithm for each transformation type.

---

**Example**

<table>
<thead>
<tr>
<th>q + b + c' = a</th>
</tr>
</thead>
<tbody>
<tr>
<td>e + b + a = n</td>
</tr>
<tr>
<td>q + p + aq = s</td>
</tr>
<tr>
<td>q + a = b</td>
</tr>
<tr>
<td>p + c = q</td>
</tr>
<tr>
<td>p + q + p = f</td>
</tr>
</tbody>
</table>
Algorithmic approach

- Heuristic methods still used.
- Weak optimality properties.
- Heuristic methods still used.
- Each operator has well-defined properties.

Example

Eliminate expressions if the increase in literals does not exceed the threshold.

Set a threshold \( t \) (usually 0).

Examine all expressions.

Example elimination algorithm:

Based on experience.

Defined by scripts.

Sequence of operators:

- Each operator has well-defined properties.
elimination algorithm

Example

repeat

c

GDM

sweep; eliminate -1

simplify-mnocomp

eliminate -1

result - a: sweep

result - a

eliminate -1

simplify-mnocomp

result - a

sweep: eliminate 5

eliminate -1

simplify-mnocomp

sweep: eliminate -1

eliminate -1

result - a

eliminate -1

full-simplify-mnocomp

eliminate -1: sweep

result - a: sweep
Boolean and algebraic methods

- Exploit properties of logic functions.
- Use don't care conditions.
- Complex at times.
- Simpler, faster but weaker.

GDM

Boolean methods:

- Exploit properties of polynomial algebra.

Algebraic methods:

- View functions as polynomials.

Example

\[ q + v = b \]

Because

\[ e + bq = 1 \]

\[ e + v = 1 \]

Algebraic substitution:

\[ \frac{e + v}{e} = (p + v)(p + v)q + v = e + bq + v = e + bq + v = q \]

Because

\[ e + bq + v = e \]

\[ ep + v = b \]

Boolean substitution:

Example and algebraic methods
Given two algebraic expressions: •

When:

Obtain dividend and divisor:

Division operator:

Complement:

Do not care sets:

Symmetric distribution laws:

Boolean functions — polynomials:

Algebraic methods:

- Boolean algebra:
  - Complement.
  - Symmetric distribution laws.
  - Don’t care sets.
  - Expressions (sum of product forms).

Algebraic division:

Algebraic model
Example

GDM

**Algebraic division:**

Let 
\[ \frac{\partial^1}{\partial x^2} \]

and 
\[ \frac{\partial^1}{\partial x^3} \]

Then 
\[ \frac{\partial^1}{\partial x^2} = \frac{\partial^1}{\partial x^3} \]

Let 
\[ \frac{\partial^1}{\partial x^2} = \frac{\partial^1}{\partial x^3} \]

Non-algebraic division:

\[ \emptyset \neq \{p, q, r\} \cup \{\ell f\} \]

\[ f = (p + q) \cdot (q + v) \]

\[ q + v = \ell f \]

and 
\[ q + v = \ell f \]

Quotient \( \bar{Q} \) and remainder \( R \) are sum of

(monomials) of the divisor:

\[ \{u, \ldots, z\} = B \]

(monomials) of the dividend:

\[ \{\ell, \ldots, \ell\} = A \]

An algorithm for division

\[ \emptyset \neq \{\ell f\} \cup \{p, q, r\} \]

\[ f = (p + q) \cdot (q + v) \]

Then 
\[ f = (p + q) \cdot (q + v) \]

Let 
\[ f = (p + q) \cdot (q + v) \]

Because 
\[ f = (p + q) \cdot (q + v) \]

Then 
\[ f = (p + q) \cdot (q + v) \]

Let 
\[ f = (p + q) \cdot (q + v) \]

Algebraic division:

Example
An algorithm for division

\[ q + a = q + \text{division remainder} \]
\[ p + c = p + \text{remainder} \]
\[ \{ q \} = H \quad \text{and} \quad \{ p, c \} = \emptyset \quad \text{\textbf{- Result:}} \]

\{ p, c \} = \{ p, c \} \cup \{ p, c \} = \emptyset \quad \text{\textbf{- Then}} \]
\[ \{ p, c \} = \emptyset \quad \text{\textbf{- Then}} \]
\[ \{ p, c \} = \emptyset \quad \text{\textbf{- Then}} \]
\[ \{ q, a \} = \emptyset \quad \text{\textbf{- Result:}} \]

Example:

\[
\begin{align*}
\text{Result:} & \quad A = \emptyset \\
\text{else} & \quad 1 = 1 \\
& \quad \text{if } \exists (H, \emptyset) \text{ dropped} \quad \text{with var. in } S \text{ such that } \emptyset = A \\
& \quad \text{return } (A) \quad \forall \emptyset = \emptyset \\
& \quad \text{for } (u \text{ to } 1 = 1) \\
& \quad \text{end } \text{ÆBERNACCI DIVISION}(A, B) \\
& \quad \text{end } \text{gcm} \quad \text{gcm} \quad \text{gcm}
\end{align*}
\]
Theorem

Given $f_i$ and $f_j$, then $f_i/f_j$ is empty when:

- $f_j$ contains a variable not in $f_i$.
- $f_i$ contains a cube whose support is not contained in that of any cube of $f_j$.
- $f_i$ contains more terms than $f_j$.
- The count of any variable in $f_i$ than in $f_j$.

Substitution

Consider expression pairs.

Apply division (in any order).

Substitute $f_i$ divided by $f_j$ quotient $+$ remainder.

where $j = \frac{f_i}{f_j}$

Use filters to reduce divisions.

Consider expression pairs.
Extraction algorithm:

- Search for appropriate divisors.
- Multiple-cube (kernel) extraction.
- Single-cube extraction: monomial.
- Search for common sub-expressions:

Single-cube extraction:
- Monomial.

Multiple-cube (kernel) extraction:
- Divisors.

Substitution algorithm:

For a substitution to be favorable,

1. If substitution is favorable

   \[ f \in GDM \]

2. If \( f \) does not pass the filter test

   \( f \in \text{FILTERED}\)

3. For \( i = 1, \ldots, N \)

   \( f \in \text{SUBSTITUTION}\)
Definitions

**Cube-free expression:**
- Cannot be factored by a cube.

**Kernel of an expression:**
- Cube-free quotient of the expression divided by a cube, called co-kernel.

**Kernel set** of an expression:
- Set of kernels.

---

**Example**

$\{(b + d + e\ c e + q + c\ e + q + a\ c\ e + q + a)\} = (x f)$

Expression $f$ has a kernel of its own because cube free.

- Divide $f$ by $g$. Get 1. Not cube free.
- Divide $f$ by $e$. Get $a + c + q + d$. Cube free. Kernel.
- Divide $f$ by $c$. Get $q + a + e + c$. Cube free. Kernel.
- Divide $f$ by $q + a + e$. Get $c$. Cube free. Not cube free.
- Divide $f$ by $c$. Get $q + a + e + c$. Cube free. Not cube free.

---

Cube-free expression.
- Cannot be factored by a cube.

---

Divide $f$ by $g$. Get 1. Not cube free.

- Divide $f$ by $e$. Get $a + c + q + d$. Cube free. Kernel.
- Divide $f$ by $c$. Get $q + a + e + c$. Cube free. Kernel.
- Divide $f$ by $q + a + e$. Get $c$. Cube free. Not cube free.
- Divide $f$ by $c$. Get $q + a + e + c$. Cube free. Not cube free.

---

Example
Theorem

(Brayton and McMullen)
c + GDM

Two expressions have a common multiple-cubed divisor 1 if and only if:

- there exist kernels 3 4 5 6 7 8 9 and :

\[ \begin{align*}
\alpha \delta &= \gamma f \\
\epsilon \delta + \epsilon\epsilon + \epsilon \epsilon + \epsilon p \epsilon &= \delta f \\
\delta + \epsilon \epsilon + \epsilon \epsilon &= \chi f \\
\eta + \nu &= \eta f
\end{align*} \]

Select intersection (\( \eta + \nu \))

The kernel set of \( \eta \) is empty

\[ \{ (\alpha \delta + \epsilon \epsilon + \epsilon \epsilon + \epsilon p \epsilon) : (\delta + \epsilon \epsilon + \epsilon \epsilon + \epsilon p \epsilon) : (\eta + \nu) \} = (\eta f) \]

\[ \{ (\delta + \epsilon \epsilon + \epsilon \epsilon + \epsilon p \epsilon) : (\delta + \epsilon \epsilon + \epsilon \epsilon + \epsilon p \epsilon) : (\eta + \nu) \} = (\eta f) \]

Example

For common sub-expression can be dropped:

- if kernel intersection is void, then the search

Consequence:

\[ p f \]

- there exist kernels \( y \) such that

\[ (p f) \subseteq Y \]

cube divisor if and only if:

- two expressions have a common multiple-

(Reyes and McMullen)

Theorem
Kernel set computation

```
( (f/s)f, (c)f )

Naive method:

- Divide function by elements in power set of its support set.

Smart way:

- Use recursion:
  - Kernel of kernels are kernels.

Recursive kernel computation:

- Exploit commutativity of multiplication.

foreach variable x:

if (x f) has such that largest cube containing x:

{ foreach variable x ∈ S:

  if (x f) is non-cube-free quotient:
    return the cubes of whose support includes C:

  return

  \[
  f \cap K = K
  \]

  \[
  \{ (f/s)f, (c)f \}
  \]

Kernels of kernels are kernels.

return the cubes of whose support includes C:
```
Some computation may be redundant:

- Example:
  - Divide by \( g \) and then by \( a \).
  - Divide by \( a \) and then by \( g \).

**Improvement:**
- Keep a pointer to literals used so far.
- Obtain duplicate kernels.

**Example:**
- Some computation may be redundant.

---

Recursive kernel computation

---

Analysis
A co-kernel corresponds to a prime rectangle.

- \( C \) are the columns not in \( C \).
- Co-rectangle (\( H \) of a rectangle (\( H \), \( C \)):
- Prime rectangle:
  - with all entries equal to 1.
  - Subset of rows and columns
  - Rectangle (\( H \), \( C \)):
    - Boolean matrix:

Matrix representation of kernels

\[
\begin{align*}
\{(q + a) \cdot (p + q + a e + 9) + p + q + a e + g e + a e\} &= \emptyset \\
\text{Adds } a e + g e + a e + p + q + a e \text{ to the kernel set.} \\
\text{Literal } q. \text{ No action required.}
\end{align*}
\]

\[
\begin{align*}
\{\} &= \emptyset \\
\emptyset \text{ to the kernel set at the last step.} \\
\text{Call considers variable } q. \text{ No kernel.} \\
\text{Recursive call with arguments: } a e + g e + p + q + a e \text{ and pointer—} & = 5 + 1. \\
\text{Literal } q. \text{ Select cube } e:
\end{align*}
\]

\[
\begin{align*}
\emptyset &= \emptyset \\
\text{No action required.} \\
\text{Recursive call with arguments: } a e + g e + p + q + a e \text{ and pointer—} & = 5 + 1. \\
\text{Literal } q. \text{ No action required.} \\
\text{Recursive call with arguments: } a e + g e + p + q + a e \text{ and pointer—} & = 5 + 1. \\
\text{Literal } q. \text{ Select cube } e:
\end{align*}
\]

\[
\begin{align*}
\emptyset &= \emptyset \\
\text{No action required.} \\
\text{Recursive call with arguments: } a e + g e + p + q + a e \text{ and pointer—} & = 5 + 1. \\
\text{Literal } q. \text{ No action required.} \\
\text{Recursive call with arguments: } a e + g e + p + q + a e \text{ and pointer—} & = v + q = 3. \\
\text{Literal } v. \text{ No action required.} \\
\text{Literal } v \text{ or } q. \text{ No action required.}
\end{align*}
\]

\[
\begin{align*}
\emptyset &= \emptyset \\
\emptyset &= \emptyset \\
\text{Literal } q. \text{ No action required.}
\end{align*}
\]

Example

\[
\begin{align*}
\emptyset &= \emptyset \\
\emptyset &= \emptyset \\
\text{Literal } q. \text{ No action required.}
\end{align*}
\]
- Kernel $C$.

- $C$-co-rectangle: $\{1,2,1,2,4,6\}$.  

- $C$-co-kernel $C$.

\[
\begin{array}{c|c|c}
\text{Rectangle (prime): } & \{1,2\} & \{3,5\} \\
\hline
\text{Kernel} & \text{co-rectangle: } \{1,2\} & \{3,5\} \\
\end{array}
\]

Single-cube-extraction

\[
\begin{array}{c|c|c|c|c|c|c}
\text{cube} & \text{var} & 1 & 2 & 3 & 4 & 5 & 6 \\
\hline
b & d & e & a & c & p & q & e \\
\end{array}
\]

\[
g + d e + a c e = x f
\]
Single-cube extraction:

- Cube intersection from different functions.
- Cube from matrix representation.
- Cube from matrix representation.
- Cube from matrix representation.

Prime rectangle: \{(1, 2, 3, 5, 6, 7) \}

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
\begin{array}{cccccc}
   a & q & c & b & e & d \\
\hline
   4 & 3 & 2 & 1 & 0 & 5 \\
   6 & 5 & 4 & 3 & 2 & 1 \\
\end{array}
\]

\[
q + cde + b + de + e = x_{non}f -
\]

Auxiliary function:

\[
q + cde = x_{non}f -
\]

\[
b + de + e = x_{non}f -
\]

Expressions:

- Cube from matrix representation.
- Cube from matrix representation.
- Cube from matrix representation.

Example:

- Cube from matrix representation.
- Cube from matrix representation.
- Cube from matrix representation.
Cubeextractionalgorithm

\begin{align*}
\text{w} &= a + b \\
\text{z} &= abc \\
\text{x} &= ace + bce + de + g \\
\text{y} &= ad + bd + cde + ge \\
\text{f} &= wde + de + g \\
\text{g} &= wde + cde + ge \\
\end{align*}

{ 
\text{By quotient + remainder, replace all functions } f \text{, where } f \text{ is a divisor.} \\
\text{Add } \text{to the network with expression } f.C. \\
\text{Generate new label } C. \\
\text{while (some favorable common cube exists) }
\} 

\text{Cube extraction algorithm}
Example

\[ \{(p+x)^2 \cap x \}: \{(q+x^2) \} = (bf)M * \]
\[ \{(q+x^2) \} = (df)M * \]

\[ p = p' x' + \forall q = \forall q' x' \cap \forall q = q' x' \cap \forall q = q' x' = q' x' = \forall x \]

Relabeling:

\[ \{(p + \forall q + \forall v) \cap (q + v) \} = (bf)M - \]
\[ p + \forall q + \forall v = bf \]
\[ \{(q + v) \} = (df)M - \]
\[ \forall p + \forall q + \forall e q + \forall e = df \]

Extended cube intersection algorithm:

- Sum of all kernels.
- Form auxiliary function:
- Relabeling:
- Kernels by cubes.
- Cubes by new variables.

Relabeling:

We need a kernel/cube matrix.

Multiple-cube extraction: any
Kernelextractionalgorithm

Example (2)
Decomposition

- Divide expression recursively.
- Kernel-based decomposition.
- NAND/NOR decomposition.
- Method of Asenhus and Curtis.

Different ways:

- Kernel-based decomposition.

\[ x = a + b \]

\[ t = ac + bc + d \]

\[ s = a + b \]

\[ x = tc + g \]

\[ x = ace + bce + de + g \]

Different ways:

- Kernel-based decomposition.

\[ x = tc + g \]

\[ x = ace + bce + de + g \]

Different ways:

- Kernel-based decomposition.
Decomposition Algorithm

\[
\text{Decompose expression } x \text{ if } \psi = x_n \\
\text{if } \phi \neq x_n \\
\text{whose size is above } x_n \\
\text{return } x_n \\
\text{selected vertex with expression } x_n \\
\text{repeat } \\
\text{Decompose (V', E')} \\
\]

Example

\[
q + a = x_f \\
p + c + e = q + c_e + e + a_e = x_f \\
\]

- Select kernel a
- Decompose •
- Recur on the quotient •
- Select kernel ac + e
- Decompose •
View Boolean functions as algebraic expressions.

Fast manipulation algorithms.

Some optimality lost.

Useful to reduce large networks.