Multi-level logic optimization

• Representation
• Optimization
  – Goal: area/delay
  – Algorithmic: algebraic, Boolean
  – Rule-based methods
• Examples of transformations
Motivation

• Multiple-level networks:
  – Semi-custom libraries.
  – Gates versus macros (PLAs):
    – More flexibility.
    – Better performance.
  – Applicable to a variety of designs.
Circuit Modeling

• Logic network:
  – Interconnection of logic functions.
  – Hybrid structural/behavioral model.

• Bound (mapped) networks:
  – Interconnection of logic gates.
  – Structural model.
Multi-level vs. Two-level

- **PLA**
  - Control logic
  - Constrained layout
  - Highly automatic
  - Technology independent
  - Slower
  - Multi valued function
  - Input, output, state encoding

- **Multi-level**
  - All logic
  - General
  - Automatic
  - Partially tech dependent
  - Can be faster
  - Some results
  - Some results
Example of bound network

```
<table>
<thead>
<tr>
<th>a</th>
<th>p</th>
<th>x</th>
</tr>
</thead>
<tbody>
<tr>
<td>b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>q</td>
<td>y</td>
</tr>
</tbody>
</table>
```

```
va -> vp -> vx
vb -> vp
vc -> vq -> vy
```
Example of network

\[ p = ce + de \]
\[ q = a + b \]
\[ r = p + a' \]
\[ s = r + b' \]
\[ t = ac + ad + bc + bd + e \]
\[ u = q'c + qc' + qc \]
\[ v = a'd + bd + c'd + ae' \]
\[ w = v \]
\[ x = s \]
\[ y = t \]
\[ z = u \]
Example of network

\[
v = a'd + bd + c'd + ae'
\]
\[
p = ce + de
\]
\[
r = p + a'
\]
\[
s = r + b'
\]
\[
t = ac + bd + bc + bd + e
\]
\[
q = a + b
\]
\[
u = q'c + qc' + qc
\]

©GDM
Network optimization

• Minimize area (power)
  – Subject to delay constraint

• Minimize maximum delay
  – Subject to area (power) constraint

• Minimize power consumption
  – Subject to delay constraint

• Maximize testability
Estimation

• Area:
  – Number of literals
  – Number of functions/gates

• Delay
  – Number of stages
  – Refined gate delay models
  – paths
Problem analysis

• Multi-level optimization is hard
• Exact models
  – Exponential complexity
  – Impractical
• Approximate methods
  – Heuristics algorithms
  – Rule-based methods
Strategies for optimization

• Improve circuit step by step
  – Circuit transformation
• Preserve network behavior
• Methods differ in
  – Types of transformations
  – Selection and order of transformations
Eliminate

• Eliminate one function from the network
• Perform variable substitution
• Example:
  - \( s = r + b' \); \( r = p + a' \);
  - \( s = p + a' + b' \);
Example

\[ v = a'd + bd + c'd + ae' \]

\[ p = ce + de \]

\[ t = ac + bd + bc + bd + e \]

\[ q = a + b \]

\[ u = q'c + qc' + qc \]

\[ s = p + a' + b' \]

\[ w \]

\[ x \]

\[ y \]

\[ z \]
decomposition

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

\[ v = a'd + bd + c'd + ae' \]

\[ \rightarrow j = a'b + c'; \quad v = jd + ae'; \]
Example

\[ V = jd + ac' \]
\[ p = ce + de \]
\[ r = p + a' \]
\[ s = r + b' \]
\[ t = ac + bd + bc + bd + e \]
\[ q = a + b \]
\[ u = q'c + qc' + qc \]
Example extraction

• Find a common sub-expression of two or more expressions
• Extract sub-expression as new function
• Introduce new vertex in the network
• Example

\[ p = ce + de; \quad t = ac + ad + bc + bd + e; \]
\[ p = (c + d)e; \quad t = (c + d)(a + b) + e; \]
\[ k = c + d; \quad p = ke; \quad t = ka + kb + e; \]
Example

\[ v = a'd + bd + c'd + ae' \]
\[ p = k \cdot e \]
\[ r = p + a' \]
\[ s = r + b' \]
\[ t = k \cdot a + k \cdot b + e \]
\[ q = a + b \]
\[ u = q'c + qc' + qc \]

©GDM
Simplification

- Simplify a local function
- Example

\[ u = q'c + qc' + qc; \]

\[ \rightarrow \quad u = q + c \]
Example

\[ v = a'd + bd + c'd + ae' \]
\[ p = ce + de \]
\[ r = p + a' \]
\[ s = r + b' \]
\[ t = ac + bd + bc + bd + e \]
\[ q = a + b \]
\[ u = q + c \]

©GDM
substitution

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

\[
\begin{align*}
t &= ka + kb + e \\
t &= kq + e \\
\text{because } q &= a + b
\end{align*}
\]
Example

\[ v = a'd + bd + c'd + ae' \]

\[ p = ke \]

\[ r = p + a' \]

\[ s = r + b' \]

\[ K = c + d \]

\[ t = ka + kb + e \]

\[ q = a + b \]

\[ u = q'c + qc' + qc \]

\[ w \]

\[ x \]

\[ y \]

\[ z \]
Example

\[ v = a'd + bd + c'd + ae' \]

\[ p = ke \]

\[ r = p + a' \]

\[ s = r + b' \]

\[ q = a + b \]

\[ K = c + d \]

\[ t = kq + e \]

\[ u = q'c + qc' + qc \]

\[ w \]
Optimization approaches

• Algorithmic approach
  – Define an algorithm for each transformation
  – Algorithm is an operator on the network

• Rule-based approach
  – Rule-data base
    • Set of pattern pairs
  – Pattern replacement by rules
Algorithmic approach

• Each operator has well-defined properties
  – Heuristic methods
  – Weak optimality
• Sequence of operators
  – Defined by scripts in SIS
  – Based on experience

©GDM
Example of script in SIS

- sweep; eliminate -1
- simplify --m nocomp
- eliminate -1
- sweep; eliminate 5
- simplify --m nocomp
- resub --a
- fx
- resub --a;sweep...
Boolean and algebraic methods

• Boolean methods
  – Exploit properties of logic functions
  – Use don’t care conditions
  – Complex

• Algebraic methods
  – View functions as polynomials
  – Exploit properties of polynomial algebra
  – Simpler, faster but weaker
summary

• Multilevel logic synthesis is performed by step-wise transformations
• Algorithms are based on both the Boolean and the algebraic models
• Rule-based