MULTIPLE-LEVEL LOGIC OPTIMIZATION

©Giovanni De Micheli

Stanford University
Outline

- Representations.

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

- Examples of transformations.

- Boolean and algebraic models.
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.
Example of bound network
Example of network

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

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

\[ p = ce + de \]

\[ r = p + a' \]

\[ s = r + b' \]

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

\[ q = a + b \]

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

\[ p = ce + de \]

\[ r = p + a' \]

\[ s = r + b' \]

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

\[ q = a + b \]

\[ u = q'c + qc' + qc \]
Example
circuit terminal behavior

\[ f = \begin{bmatrix} a'd + bd + c'd + ae' \\ a' + b' + c + d \\ ac + ad + bc + bd + e \\ a + b + c \end{bmatrix} \]
Network optimization

- Minimize area (power) estimate:
  - subject to delay constraints.

- Minimize maximum delay:
  - subject to area (power) constraints.

- Minimize power consumption.
  - subject to delay constraints.

- Maximize testability.
Estimation

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

- Delay:
  - Number of stages.
  - Refined gate delay models.
  - Sensitizable paths.
Problem analysis

- Multiple-level optimization is hard.

- Exact methods:
  - Exponential complexity.
  - Impractical.

- Approximate methods:
  - Heuristic algorithms.
  - Rule-based methods.
Strategies for optimization

- Improve circuit step by step.
  - Circuit *transformations*.

- Preserve network behavior.

- Methods differ in:
  - *Types* of transformations.
  - *Selection* and *order* of transformations.
Example elimination

- Eliminate one function from the network.

- Perform variable substitution.

- Example:

  \[- s = r + b' ; \ r = p + a' \]

  \[- \Rightarrow s = p + a' + b'. \]
Example
elimination

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

\[ w \]
\[ x \]
\[ y \]
\[ z \]
Example decomposition

- Break one function into smaller ones.

- Introduce new vertices in the network.

- Example:

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

  \[ \Rightarrow j = a' + b + c'; v = jd + ae' \]
Example decomposition

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

\[ p = ce + de \]

\[ r = p + a' \]

\[ s = r + b' \]

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

\[ q = a + b \]

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

\[ j = a' + b + c' \]

\[ v = j d + ae' \]

\[ w \]

\[ x \]

\[ y \]

\[ z \]
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:

\[-p = ce + de; \quad t = ac + ad + bc + bd + e;\]

\[-p = (c + d)e; \quad t = (c + d)(a + b) + e;\]

\[-\Rightarrow k = c + d; \quad p = ke; \quad t = ka + kb + e;\]
Example simplification

- Simplify a local function.

- Example:

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

  \[\Rightarrow u = q + c;\]
Example simplification
Example
substitution

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

• Example:
  
  \[- t = ka + kb + e.\]
  
  \[- \Rightarrow t = kq + e\]
  
  – Because \( q = a + b \).
Example substitution
Example
sequence of transformations

\[
\begin{align*}
  j &= a' + b + c' \\
  k &= c + d \\
  q &= a + b \\
  s &= ke + a' + b' \\
  t &= kq + e \\
  u &= q + c \\
  v &= jd + ae'
\end{align*}
\]
Optimization approaches

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

- **Rule-based** approach:
  - Rule-data base:
    * Set of pattern pairs.
  - Pattern replacement driven by rules.
Algorithmic approach

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

- Sequence of operators:
  - Defined by *scripts*.
  - Based on experience.
Example elimination algorithm

- Set a threshold $k$ (usually 0).

- Examine all expressions.

- Eliminate expressions if the increase in literals does not exceed the threshold.
Example elimination algorithm

\[ ELIMINATE( G_n(V, E) , k) \{
    \text{repeat } \{ \\
    v_x = \text{selected vertex with value} < k; \\
    \text{if } (v_x = \emptyset) \text{ return}; \\
    \text{replace } x \text{ by } f_x \text{ in the network}; \\
    \}
\} \]
Example
MIS/SIS rugged script

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

- Boolean methods:
  - Exploit properties of logic functions.
  - Use *don’t care* conditions.
  - Complex at times.

- Algebraic methods:
  - View functions as *polynomials*.
  - Exploit properties of polynomial algebra.
  - Simpler, faster but weaker.
Example

• Boolean substitution:
  
  - $h = a + bcd + e; \; q = a + cd$
  
  - $\Rightarrow h = a + bq + e$
  
  - Because $a + bq + e = a + b(a + cd) + e = a + bcd + e$.

• Algebraic substitution:
  
  - $t = ka + kb + e$.
  
  - $\Rightarrow t = kq + e$
  
  - Because $q = a + b$. 
Summary

- Multilevel logic synthesis is performed by step-wise transformations.

- Algorithms are based on both the Boolean and the algebraic models.

- Rule-based systems.