Concurrent Extraction of Partial Kernels with Timing Consideration

Peyman Rezvani, Massoud Pedram
University of Southern California
Los Angeles, CA

International Workshop on Logic Synthesis
June 14, 2001

Outline

- Introduction
- Concurrent Extraction of Partial Kernels
- Timing Consideration
- Experimental Results
- Conclusion
Introduction

- Logic optimization objectives
  - Area
    - Estimate: Literal count
  - Delay
  - Power consumption
  - Testability
- Logic optimization techniques
  - Two-level
  - Multi-level
    - Logic Extraction

Logic Extraction

- Exploiting commonality among different expressions to reduce literal count

[Diagram showing logic expressions and literal count: 13 9]
Kernels (Brayton & McMullen, ‘82)

- Kernel
  - A cube-free divisor resulting in single-cube quotient
  - Much fewer than general divisors
- Theorem:
  - Two expressions $f$ and $g$ have a common multiple-cube divisor iff some kernel of $f$ and some kernel of $g$ have more than one cube in common.
- Drawback
  - Number of kernels can grow exponentially.

Fast Extract (Rajski et al., ‘92)

- Only consider double-cube (and two-literal single-cube) divisors.
- Objects of size 2 $\Rightarrow$ Polynomial domain
- Theorem:
  - Two expressions have a common multiple-cube divisor iff they have a common double-cube divisor
Fast Extract – cont’d

for each node \( n_i \) in the network:
  for each cube pair \((c_i, c_j)\) in \( n_i \):
    \( k_{ij} \) = kernel corresponding to \((c_i, c_j)\);
    update the weight for \( k_{ij} \);
  while there exists a kernel with positive weight:
    select the kernel \( k_{ij} \) with maximum weight;
    extract \( k_{ij} \) from all its cube pairs;
    update other kernels’ weights;

- Drawback: *Greedy* and *total*
- Improvement: *Concurrent partial* extraction

Motivational Example

\[
\begin{align*}
  w &= 4 \\
  K_2 &= a + c \\
  n_j &= ab + bc + cde \\
  n_2 &= aefg + cefg \\
  n_3 &= bf + def \\
  K_1 &= b + de \\
  w &= 3
\end{align*}
\]

Fast Extract:
- Extract \( K_2 \) from \( n_j, n_2 \)
- Saving: 4 literals

Optimum Solution:
- Extract \( K_2 \) from \( n_2 \)
- Extract \( K_1 \) from \( n_j, n_3 \)
- Saving: 5 literals
Definitions

- Candidate Cube Pair (CCP):
  - Any two cubes belonging to the same node
  - Associated with a double-cube kernel

- Compatible CCPs:
  - Two CCPs corresponding to the same kernel
  - Example: \((ab, bc) \sim (aefg, cefg)\)

- Conflicting CCPs:
  - Two CCPs having a common cube
  - Example:
    \[
    n_1 = ab + bc + cde
    \]
    \[
    (a+c) \quad (b+de)
    \]

Definitions – cont’d

- CCP Graph
  - Vertices \(\leftrightarrow\) CCPs
  - Conflict (red) edge \(\leftrightarrow\) Conflict relation
  - Compatibility (green) edge \(\leftrightarrow\) Compatibility relation

\[
\begin{align*}
\text{CCP}_1 & : ab + bc \\
\text{CCP}_2 & : bc + cde \\
\text{CCP}_3 & : ab + cde \\
\text{CCP}_4 & : aefg + cefg \\
\text{CCP}_5 & : bf + def
\end{align*}
\]
Definitions – cont’d

- **CCP Class**
  - All CCPs corresponding to the same kernel
  - Makes a clique with compatibility edges
  - Sub-clique ↔ Partial extraction
- **Gain**
  - Defined for each CCP, is the literal saving upon substitution of a variable for the kernel in that CCP
- **Cost**
  - Defined for each CCP class (clique), is the number of literals in kernel
- **Value:**
  - Defined for each clique/sub-clique, is:
    \[
    value(c) = \sum_{i:v \in c} gain(v_i) - cost(c)
    \]

Problem Formulation

- Concurrent Partial Kernel Extraction (CPKE): Find a set of cliques (not necessarily maximal) in the CCP graph to maximize the total value in such a way that no conflict edge exists between two selected vertices.

\[
\begin{align*}
ab + bc &= bK_2 \\
ac + efg + cefg &= efgK_2 \\
b + def &= fK_1 \\
bc + cde &= cK_1 \\
aefg + cefg &= efgK_2
\end{align*}
\]

- Theorem: CPKE problem is NP-Hard.
Proposed Approach

- Formulate the CPKE problem as the maximum-weight independent set problem (MWIS).
  - MWIS is NP-Hard, however, it is a well-studied problem and a number of effective heuristics exist for it.
  - MWIS formulation also enables us to consider other cost functions.

Yet More Definitions!

- CCPs are of two types:
  - Essential (green): no conflict
  - Optional (red): in conflict with some CCP
- Partial CCP class
  - Subset of a CCP class containing all essential CCPs
  - Corresponds to a partial kernel extraction
  - Associated with a value as: \( value = \sum (gain - cost) \)
- Conflicting Partial CCP classes
  - Partial CCP classes that have some CCP in conflict
Example

\[ p_3 = \{ CCP_3 \} \]
\[ p_4 = \{ CCP_4 \} \]
\[ p_{14} = \{ CCP_1, CCP_4 \} \]
\[ p_5 = \{ CCP_3 \} \]
\[ p_{25} = \{ CCP_2, CCP_3 \} \]

Last Definition!

- Dominating partial CCP class

\[ P_2 \subset P_1 \]

- Prime partial CCP class
  - A non-dominated partial CCP class
New Problem Formulation

- Build Conflict Graph:
  - Vertices ⇔ Prime partial CCP classes
  - Edges ⇔ Partial CCP class conflict

- Find max-weight independent set of vertices

Timing Consideration

- Objective:
  - Minimize literal count while not increasing delay
  - Trade off delay to further decrease area

- Delay estimate:
  - Unit delay model
  - Unit-Fanout delay model
  - Library delay model
Timing Consideration

Algorithm *Timing_Analyzer (S)*

for each prime partial CCP class \( p \) in \( S \):
   for each \( CCP \) in \( p \):
      mark the node corresponding to \( CCP \);
      traverse network in DFS order:
         arrival \( (n) = 1 + \max (\text{arrival} (\text{fanin} (n))) \);
         if \( n \) is marked then
            arrival \( (n) = \text{arrival} (n) + 1; \)
         return \( \max (\text{arrival} (\text{PO})) \);

- To speedup, operate on critical sub-network

Timing Consideration

- Extraction \( E_1 \) is superior to \( E_2 \) if \( E_1 \) results in:
   - Larger or equal literal saving and smaller level increase, or
   - Larger literal saving and smaller or equal level increase

- To prune a sub-tree of search space, use upper-bound for literal saving and lower-bound for level increase.
Experimental Results - I

Literal count improvement
Max: 30%
Ave: 5%

Experimental Results - II

Area improvement
Max: 27%
Ave: 5%
Experimental Results - III

Delay improvement
Max: 32%
Ave: 13%

Experimental Results - IV

Benchmark: 9symml

<table>
<thead>
<tr>
<th>Area</th>
<th>124743</th>
<th>117454</th>
<th>115337</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay</td>
<td>1.73</td>
<td>1.80</td>
<td>1.85</td>
</tr>
</tbody>
</table>

June 14, 2001
International Workshop on Logic Synthesis
Conclusion

- Introduced and formally defined the problem of concurrent extraction of partial kernels.
- Proved the problem to be NP-Hard.
- Formulated the problem as the MWIS problem in a graph.
- Proposed a method to control delay increase while doing extraction.