Effect of BDD Optimization on Synthesis of Reversible and Quantum Logic

Robert Wille, Rolf Drechsler

Institute of Computer Science
University of Bremen, Germany
{rwille,drechsle}@informatik.uni-bremen.de
Outline

• Motivation and Background
• BDD-based Synthesis
• Exploiting BDD-optimization
  – Shared Nodes
  – Complement Edges
  – Reordering
• Experimental Results
• Conclusions
Reversible Logic

- Applications in
  - Quantum Computing
  - Low-Power Design
  - Optical Computing
  - DNA Computing
  - ...

Toffoli gate
Quantum Logic

- Is inherently reversible
- Signals represented by qubits (i.e. non-Boolean values)
- Value of each qubit is restricted to 0, 1, $V_0$ or $V_1$

- NOT: Performs inversion
- CNOT: Controlled inversion
- $V$: ‘square root’ of NOT
- $V^+$: Inverse of $V$
Synthesis Problem

- Given: Rev. function to be synthesized

<table>
<thead>
<tr>
<th>c</th>
<th>b</th>
<th>a</th>
<th>$\varphi_3$</th>
<th>$\varphi_2$</th>
<th>$\varphi_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<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>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<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>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- Task: Find network (i.e. a cascade of gates)

- Previous Work:
  - Often rely on truth table (or similar) description
  - Only applicable to small functions

$\Rightarrow$ No fanouts, no feedback
Outline

• Motivation and Background

• BDD-based Synthesis

• Exploiting BDD-optimization
  – Shared Nodes
  – Complement Edges
  – Reordering

• Experimental Results

• Conclusions
Binary Decision Diagrams (BDDs)

- Data structure for efficient representation and manipulation of Boolean functions
- Rooted, directed, acyclic graph, which consists of decision nodes and two terminal nodes (leafs)
- Each decision node is labeled by a Boolean variable and has two child nodes (low and high)

\[ f = x_1 \oplus x_2 \cdot x_3 \]
BDD-based Synthesis #1

1. Build BDD for function $f$ using existing techniques
2. Substitute each BDD node by a cascade of gates
BDD-based Synthesis #2
Example (XOR function)
BDD-based Synthesis #3

- Linear worst case behavior regarding run-time and space requirements
- Resulting circuits are bounded by BDD size
- BDD optimization can be exploited
Outline

- Motivation and Background
- BDD-based Synthesis
- Exploiting BDD-optimization
  - Shared Nodes
  - Complement Edges
  - Reordering
- Experimental Results
- Conclusions
Shared Nodes

- Used to represent a sub-formula more than once
- Need to preserve node values
  (requires additional line)
Complement Edges

- Allows to represent a function as well as its negation by a single node only.
Reordering

Can be directly applied
(no further adjustments)

\[ f = x_1 \cdot x_2 + x_3 \cdot x_4 + \cdots + x_{n-1} \cdot x_n \]
Outline

• Motivation and Background
• BDD-based Synthesis
• Exploiting BDD-optimization
  – Shared Nodes
  – Complement Edges
  – Reordering
• Experimental Results
• Conclusions
Experimental Setup

- Implemented on the top of CUDD
- Benchmarks from RevLib (www.revlib.org) and LGSynth package

Objectives:
- Circuit lines
- Number of Toffoli gates
- Quantum Cost
- Run-time (often negligible)
Results (selected)
Comparison to Previous Work

- RMRLS: Gupta et al. @ TCAD, 2006
- RMS: Maslov et al. @ TODAES, 2007

- Significant run-time for both RMRLS and RMS
- Most of the functions aborted after 500 CPU seconds
Outline

• Motivation and Background
• BDD-based Synthesis
• Exploiting BDD-optimization
  – Shared Nodes
  – Complement Edges
  – Reordering
• Experimental Results
• Conclusions
Conclusions

• BDD-based synthesis has been introduced

• Effect of BDD optimization
  – Shared Nodes: Always yields better results
  – Compl. Edges: Better results in most cases
  – Orderings: Best results with exact ordering, but Sifting also yields good circuits

• Comparison to Previous Work:
  – Larger functions can be handled
  – Significant improvements in quantum cost
  – More circuit lines needed
Effect of BDD Optimization on Synthesis of Reversible and Quantum Logic

Robert Wille, Rolf Drechsler

Institute of Computer Science
University of Bremen, Germany
{rwille,drechsle}@informatik.uni-bremen.de