A Novel Synthesis Algorithm for Reversible Circuits

Mehdi Saeedi, Mehdi Sedighi*, Morteza Saheb Zamani

Email: {msaeedi, msedighi, szamani}@aut.ac.ir

Quantum Design Automation Lab, Computer Engineering Department
Amirkabir University of Technology
Tehran, Iran

ICCAD 2007
Outline

- Introduction
- Basic Concept
- Previous Work
- Synthesis Algorithm
- Experimental Results
- Future Works
- Conclusions
Introduction

- Boolean reversible functions
  - n-input, n-output,
  - Unique output assignment
  - Example: a 3-input, 3-output function (0,1,2,7,4,5,6,3)

<table>
<thead>
<tr>
<th>a_0</th>
<th>a_1</th>
<th>a_2</th>
<th>f_0</th>
<th>f_1</th>
<th>f_2</th>
<th>F</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>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
Power dissipation

- Landauer’s paper
  - Every lost bit causes an energy loss
  - When a computer erases a bit of information, the amount of energy dissipated into the environment is at least $k_B T \ln 2$

- Bennett’s paper
  - To avoid power dissipation in a circuit, the circuit must be built with reversible gates
Applications of reversible circuits

- Low power CMOS design
  - Reversible 4-bit adder
    - 384 transistors with **no power rails**
- Optical computing
- Quantum computing
  - Each unitary quantum gate is intrinsically reversible
Basic Concept

- Reversible gate
- Various reversible gates
  - CNOT-based gates
    - NOT, CNOT, C^2NOT (Toffoli), …
  - Generalized Toffoli gate
    - Positive controls
    - Negative controls
Reversible Circuits

High-level Description

Gate-level circuits

Physical Implementation

Synthesis

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0
\end{bmatrix}
\]
Synthesis Algorithms Categories

- Transformation-based algorithms [11]-[13]
  - Used to improve the cost of circuit
  - Applied on the results of other algorithms
  - Usually use templates to optimize a circuit
Synthesis Algorithms Categories (Cnt’d)

- Constructive algorithms [15]-[19]
  - Construct a circuit from a given specification (i.e. truth table, PPRM expansion, decision diagrams, …)
  - The resulted cost may not be optimized
  - The time complexity of the algorithm may be too high
Synthesis Algorithms Categories (Cnt’d)

- A transformation based algorithm [18]
  - Basic algorithm
    - Uses row-based operations
  - Output permutation
    - Tries all $n!$ output permutations to simplify the result
  - Control input reduction
    - To reduce the number of control qubits
Synthesis Algorithms Categories (Cnt’d)

- Bidirectional algorithm
  - To apply the method in both directions simultaneously
- Template matching
  - A template consists of a sequence of gates to be matched and the sequence of gates to be substituted when a match is found
  - A time consuming procedure
Synthesis Algorithms Categories (Cnt’d)

- Search-based methods [15],[17]
  - Also called substitution-based methods
  - Use common sub-expressions to simplify the input function
  - All possible gates should be evaluated at each step
  - The best possible gates are selected based on a predefined function
  - The algorithm convergence is not guaranteed
  - An extensive exploration is required
  - A time consuming procedure
Synthesis Algorithms Categories (Cnt’d)
The Proposed Algorithm

- Definition: Output Translation
  - The application of a reversible CNOT-based gate at the output side of a reversible specification \( F \)
  - The result of using an output translation will also be reversible
  - Only one function is changed at a time after using an output translation
The Goal of the Algorithm

- To generate a set of ordered output translations
- When applied to the reversible specification $F$, generates $a_i$ from $f_i$
Applying an Output Translation

- Lemma 1 explains the results of using an output translation on a given specification:
  - (a) Applying an output translation, exchanges the location of $2^k$ minterm pairs where $k \leq n-1$

<table>
<thead>
<tr>
<th>a₀</th>
<th>a₁</th>
<th>a₂</th>
<th>f₀</th>
<th>f₁</th>
<th>f₂</th>
<th>F</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>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>7</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
(b) Exchanging the location of $2^{k-1}$ ($k=n-m+1$) minterm pairs produces the same result as applying an output translation if:

- All $2^k$ minterms have the same value on $m-1$ particular bit locations
- The two minterms of each pair differ only in one bit position
The Proposed Algorithm

1. Select the \( i^{th} \) minterm of output functions
2. Mark it as visited
3. If its \( b^{th} \) variable is not correct
   - Find a minterm which differs from it in its \( b^{th} \) variable
   - If the new minterm is below the current minterm
     - Exchange their locations
   - Mark it as visited
The Proposed Algorithm

If the new minterm is above the current minterm
If the new minterm is not in the right locations
Exchange their locations
Mark it as visited

Repeat the previous steps for all minterms and all variables until \( a_k = f_k \) for each \( k \)
Example
Gate Extraction Method

\[
\begin{align*}
    f_2(\text{new}) &= f_2 \oplus f_1 f_3 \\
    f_3(\text{new}) &= f_3 \oplus f_1 f_2' \\
    f_2(\text{new}) &= f_2 \oplus f_1 f_3
\end{align*}
\]

Obtained gates should be applied in the reverse order
The Algorithm Convergence

- Theorem 1: The proposed algorithm will converge to a possible implementation after several steps
  - Each output translation does not change the results of the previous ones
  - Only one function is changed at a time after using an output translation
The Time Complexity

- **Assumption:** At most \( h \) gates are needed
- **Search-based method**
  - \( n \times 2^{n-1} \) gates must be evaluated to select the best possible gates at each step

\[
C_n^1 + 2 \times C_n^2 + n \times (C_{n-1}^3 + \ldots + C_{n-1}^{n-1}) = n \times 2^{n-1}
\]

- At most \((n \times 2^{n-1})^h\) gates should be evaluated
- The proposed algorithm needs \( O(h \times 2^n) \) steps to reach a result
Search-based Tree
## Experimental Results

<table>
<thead>
<tr>
<th>Ckt #</th>
<th>Specification</th>
<th>Number of Gates</th>
<th>Number of Searched Nodes &amp; Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Proposed Algorithm (Basic)</td>
<td>[15],[17]</td>
</tr>
<tr>
<td>1</td>
<td>(1,0,3,2,5,7,4,6)</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>(7,0,1,2,3,4,5,6)</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>(0,1,2,3,4,6,5,7)</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>(0,1,2,4,3,5,6,7)</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>(0,1,2,3,4,5,6,8,7,9,10,11,12,13,14,15)</td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td>6</td>
<td>(1,2,3,4,5,6,7,0)</td>
<td>3</td>
<td>3</td>
</tr>
</tbody>
</table>
### Experimental Results (Cnt’d)

<table>
<thead>
<tr>
<th>Specification</th>
<th>Number of Gates</th>
<th>Searched Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Proposed Algorithm (Basic)</td>
<td>[15], [17]</td>
</tr>
<tr>
<td>7 (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,0)</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>8 (0,7,6,9,4,11,10,13,8,15,14,1,12,3,2,5)</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>9 (3,6,2,5,7,1,0,4)</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>10 (1,2,7,5,6,3,0,4)</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>11 (4,3,0,2,7,5,6,1)</td>
<td>8</td>
<td>7</td>
</tr>
<tr>
<td>12 (7,5,2,4,6,1,0,3)</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>13 (6,2,14,13,3,11,10,7,0,5,8,1,15,12,4,9)</td>
<td>23</td>
<td>15</td>
</tr>
<tr>
<td>Average</td>
<td>7.46</td>
<td>5.76</td>
</tr>
</tbody>
</table>
## Experimental Results (Cnt’d)

<table>
<thead>
<tr>
<th>Circuit #</th>
<th>Specification</th>
<th>Number of Gates</th>
<th>Proposed Algorithm (Bidirectional)</th>
<th>[18] (Bidirectional)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(1,0,3,2,5,7,4,6)</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(7,0,1,2,3,4,5,6)</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(0,1,2,3,4,6,5,7)</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>(0,1,2,4,3,5,6,7)</td>
<td>5</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>(0,1,2,3,4,5,6,8,7,9, 10,11,12,13,14,15)</td>
<td>7</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>(1,2,3,4,5,6,7,0)</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
# Experimental Results (Cnt’d)

<table>
<thead>
<tr>
<th>Specification</th>
<th>Number of Gates</th>
<th>Proposed Algorithm (Bidirectional)</th>
<th>[18] (Bidirectional)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7 (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,0)</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>8 (0,7,6,9,4,11,10,13,8,15,14,1,12,3,2,5)</td>
<td>4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>9 (3,6,2,5,7,1,0,4)</td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>10 (1,2,7,5,6,3,0,4)</td>
<td>6</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>11 (4,3,0,2,7,5,6,1)</td>
<td>5</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>12 (7,5,2,4,6,1,0,3)</td>
<td>5</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>13 (6,2,14,13,3,11,10,7,0,5,8,1,15,12,4,9)</td>
<td>9</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>4.92</strong></td>
<td><strong>6.72</strong></td>
<td></td>
</tr>
</tbody>
</table>
Experimental Results (Cnt’d)

- All possible 3-input/3-output reversible circuits (8! = 40320) are synthesized
3-input/3-output reversible circuits

- Average number of gates per circuit
  - The proposed algorithm: 7.28
- Average number of steps per circuit = 63.87
- It takes about 4 minutes to synthesize all circuits
  - 0.006 seconds for each circuit on average
Future Directions

- Working on the improvement of the resulting synthesized circuit
  - By combining the proposed approach and the search-based methods
  - By selecting the best possible variable at each step
Conclusions

- A new non-search based synthesis algorithm was proposed
- Several examples taken from the literature are used
- The proposed approach guarantees a result for any arbitrarily complex circuit
- It is much faster than the search-based ones
Thank you for your attention!