Moving Forward: A Non-Search Based Synthesis Method towards Efficient CNOT-Based Quantum Circuit Synthesis Algorithms

Mehdi Saeedi, Morteza Saheb Zamani, Mehdi Sedighi

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

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

ASPDAC 2008
Outline

- Introduction
- Basic Concept
- Previous Work
- Synthesis Algorithm (MOSAIC)
- Experimental Results
- Future Works
- Conclusions
Quantum Computing

- The fundamental limits of CMOS technology
- The enormous amount of required processing power for future applications
- New computational models
- Quantum computing
Synthesis

- Quantum information processing is in the preliminary state
- No mature synthesis method for quantum circuit synthesis has been proposed yet
- A systematic algorithm for Boolean reversible circuit synthesis
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>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>
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, C²NOT (Toffoli), …
  - Generalized Toffoli gate
    - Positive controls
    - Negative controls
Matrix representation

- An n-qubit gate has a unitary $2^n \times 2^n$ matrix, QMatrix, describing its functionality.
- The QMatrix of an n-qubit quantum circuit is well-formed if it has the following two conditions:
  - Matrix elements can only be zeros or ones.
  - Each column or row has exactly one element with a value of 1.
- CNOT-based quantum circuits & Boolean reversible circuits have well-formed QMatrices
Reversible Circuits

Gate-level circuits

Physical Implementation

High-level Description

Synthesis

\[
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
\]

<table>
<thead>
<tr>
<th>c_n</th>
<th>a</th>
<th>b</th>
<th>g_1</th>
<th>g_2</th>
<th>s</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>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</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>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Synthesis Algorithms Categories

- **Transformation-based algorithms [12]-[15]**
  - 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 [6], [7], [17], [18]
  - 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
The Proposed Algorithm

Definition: $L_k$ QTranslation

- The application of a $k$-qubit gate with matrix $G$ on a quantum circuit with a QMatrix $M$
- The result of using an $L_k$ QTranslation is the same as multiplication of $M$ by $G$, i.e. $MG$
- The result of using an $L_k$ QTranslation is also well-formed
The Proposed Algorithm

- Definition: Quantum pair (QPair$_{i,j}$)
  - Two rows form a quantum pair (QPair$_{i,j}$) if the numbers $i$ and $j$ differ in only one bit position

- Definition: C$^k$QPair
  - The $2^k$ rows of a QMatrix the row numbers of which have the same value on their $n$-$k$ bit locations form a single group called C$^k$QPair
The Goal of the Algorithm

- The goal of MOSAIC is to decompose a given QMatrix into several elementary QMatrices of CNOT-based gates efficiently.
  - By generating a set of ordered $L_k$ QTranslation
  - When applied to the QMatrix $M$, generates an identity matrix $I$
Applying an $L_k$ QTranslation

- Lemma 1 and Lemma 2 explain the results of using an $L_k$ QTranslation on a given QMatrix $M$
The MOSAIC Algorithm

Select the $c^{th}$ column of the given QMatrix

set $r$ to be the $c$ row number which has a value of 1

if the $r^{th}$ row is not marked as visited

if the $b^{th}$ bits of $r$ and $c$ are not equal

find the number $p$ which differs with $r$ in its $b^{th}$ bit
The MOSAIC Algorithm

1. set q to be the column number of row p which has a value of 1
2. if q != p and p >= r
3. exchange the locations of the p^{th} and r^{th} rows
4. mark the p^{th} and r^{th} rows as visited
5. Repeat the previous steps for all columns and all bits until M has been changed to identity matrix
Example (1)

- $b=0; c=0$
- $r=7$ (111)
  - $p=110$ (6)
  - $q=7$
- $\{7,6\}$
- Brown box: $c$
- Green Box: $p$
### Example (2)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</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>0</td>
<td>0</td>
<td>1</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>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</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>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- **b=0; c=1**
- **r=0 (000)**
- **p=001 (1)**
- **q=2**
- **{0,1}**
Example (3)

\[
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

- **b=0; c=2**
- **r=1 (visited)**
Example (4)

\[
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

- \( b=0; c=3 \)
- \( r=2 \ (010) \)
- \( p=011 \ (3) \)
- \( q=4 \)
- \( \{2,3\} \)
Example (5)

\[
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

- \( b=0; c=4 \)
- \( r=3 \) (visited)
Example (6)
Example (7)

\[
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

- \( b=0; c=6 \)
- \( r=5 \) (visited)
Example (8)

\[
\begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{bmatrix}
\]

- \( b=0; c=7 \)
- \( r=6 \) (visited)

- \( \{0,1,2,3,4,5,6,7\} \)
Example (After the first step)

```
<table>
<thead>
<tr>
<th>0 0 1 0 0 0 0 0 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 0 1 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 1 0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 1 0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0 0 1 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0 1 0 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 0 0 0 0 0 1</td>
</tr>
</tbody>
</table>
```

Right locations
After the last step

Each row exchange corresponds to a gate (Lemma 1 and Lemma 2)

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]
The Algorithm Convergence

- Theorem 1: The MOSIC algorithm will converge to a possible implementation after several steps.
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
  
  $$C_n^1 + 2 \times C_n^2 + n \times (C_{n-1}^3 + \ldots + C_{n-1}^{n-1}) = n \times 2^{n-1}$$
  
  - $O(n \times 2^n)^h$ gates should be evaluated
- The MOSAIC algorithm needs $O(h \times 2^n)$ steps to reach a result
### 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>1</td>
<td>(1,0,3,2,5,7,4,6)</td>
<td>4</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>9</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>
<tr>
<td>7</td>
<td>(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</td>
<td>(0,7,6,9,4,11,10,13,8,15,14,1,12,3,2,5)</td>
<td>4</td>
<td>4</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>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>6</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>19</td>
<td>15</td>
</tr>
<tr>
<td>14 (9,7,13,10,4,2,14,3,0,12,6,8,15,11,1,5)</td>
<td>23</td>
<td>14</td>
</tr>
<tr>
<td>15 (6,4,11,0,9,8,12,2,15,5,3,7,10,13,14,1)</td>
<td>21</td>
<td>17</td>
</tr>
<tr>
<td>16 (13,1,14,0,9,2,15,6,12,8,11,3,4,5,7,10)</td>
<td>29</td>
<td>16</td>
</tr>
<tr>
<td>Average</td>
<td>9.81</td>
<td>7.62</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
## Different size QMatrices

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Number of Steps</th>
<th>Number of Gates</th>
<th>CPU Time (seconds)</th>
<th>Inputs</th>
<th>Number of Steps</th>
<th>Number of Gates</th>
<th>CPU Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>7</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>34</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>155</td>
<td>9</td>
<td>0.01</td>
</tr>
<tr>
<td>5</td>
<td>624</td>
<td>17</td>
<td>0.05</td>
<td>6</td>
<td>2265</td>
<td>30</td>
<td>0.17</td>
</tr>
<tr>
<td>7</td>
<td>7731</td>
<td>55</td>
<td>0.51</td>
<td>8</td>
<td>24422</td>
<td>84</td>
<td>1.65</td>
</tr>
<tr>
<td>9</td>
<td>72960</td>
<td>133</td>
<td>5.46</td>
<td>10</td>
<td>225280</td>
<td>206</td>
<td>17.27</td>
</tr>
<tr>
<td>11</td>
<td>581632</td>
<td>259</td>
<td>45.39</td>
<td>12</td>
<td>1277952</td>
<td>312</td>
<td>61.50</td>
</tr>
</tbody>
</table>
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!