DESIGN AUTOMATION FOR
REVERSIBLE AND ADIABATIC CIRCUITS

Robert Wille
Johannes Kepler University Linz
Software Competence Center Hagenberg GmbH (SCCH)

robert.wille@jku.at
https://iic.jku.at/eda/research/quantum/
SYNTHESIS OF REVERSIBLE CIRCUITS

\[ f: B^n \rightarrow B^m \]

Embedding

\[ f: B^k \rightarrow B^k \]

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
SYNTHESIS OF REVERSIBLE CIRCUITS

\[ f: B^n \rightarrow B^m \]

Embedding

\[ f: B^k \rightarrow B^k \]

Synthesis

Reversible Circuit

\[
\begin{array}{c|c}
 a & b & y \\
 0 & 0 & 0 \\
 0 & 1 & 0 \\
 1 & 0 & 0 \\
 1 & 1 & 1 \\
\end{array}
\]

\[
\begin{array}{ccc|ccc}
 x & y & z & x' & y' & z' \\
 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 1 & 0 & 0 & 1 \\
 0 & 1 & 0 & 0 & 1 & 0 \\
 0 & 1 & 1 & 1 & 1 & 1 \\
 1 & 0 & 0 & 1 & 0 & 0 \\
 1 & 0 & 1 & 1 & 1 & 0 \\
 1 & 1 & 0 & 1 & 1 & 0 \\
 1 & 1 & 1 & 0 & 1 & 1 \\
\end{array}
\]
THE EMBEDDING PROCESS

- Make output patterns distinguishable
  - Add 1 garbage output

- Adjust number of inputs and outputs
  - Insert 1 ancillary input

- Assign precise values
  - 192 possibilities

- Embedding is a coNP-hard problem

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>a</th>
<th>x'</th>
<th>y'</th>
<th>g</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>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</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>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</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>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
SYNTHESIS OF REVERSIBLE CIRCUITS

- Example: Transformation-based Synthesis
  - Transform outputs to inputs
  - Apply gates from right to left

<table>
<thead>
<tr>
<th>line (i)</th>
<th>input</th>
<th>output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000</td>
<td>000</td>
</tr>
<tr>
<td>1</td>
<td>001</td>
<td>001</td>
</tr>
<tr>
<td>2</td>
<td>010</td>
<td>010</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>011</td>
</tr>
<tr>
<td>4</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>5</td>
<td>101</td>
<td>101</td>
</tr>
<tr>
<td>6</td>
<td>110</td>
<td>110</td>
</tr>
<tr>
<td>7</td>
<td>111</td>
<td>111</td>
</tr>
</tbody>
</table>
SYNTHESIS OF REVERSIBLE CIRCUITS

\[ f : B^n \rightarrow B^m \]

Embedding

\[ f : B^k \rightarrow B^k \]

Synthesis

Reversible Circuit

Solution:
Skipping embedding
\( \rightarrow \) One-pass synthesis

Drawbacks:

- Embedding is expensive
- Degree of freedom is not exploited
- Exponential growth of representation
ONE-PASS DESIGN FLOW

- Example: Transformation-based Synthesis

<table>
<thead>
<tr>
<th>line $i$</th>
<th>input $xy$</th>
<th>output $xy$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>00</td>
<td>00✔</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>01✔</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>10✔</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>11✔</td>
</tr>
</tbody>
</table>

- Start synthesis without embedding
- Modify function if problem occurs
  - Store changes on buffer line
- Complete synthesis with „wrong“ function
- Revert changes after synthesis
  - One gate for each buffer line
DESIGN AUTOMATION FOR ADIABATIC CIRCUITS

Thus far:
- Assumed full reversibility (e.g. mapping to Toffoli gates)
- Caused overhead which is not necessarily needed
  - Conditional reversibility is sufficient

Possible two-stage approach:
1. Realize the function with respect to a certain logic gate library
2. Map the resulting netlist to an adiabatic circuit satisfying switching rules
1\textsuperscript{st} STEP: UTILIZE AND-INVERTER GRAPHS (AIGS)

- Graph-based representation of Boolean functions
- Nodes represent AND operations; edges can be inverted (denoted by black circle)
- Can easily be mapped to NAND circuits and, using DeMorgan, to NOR circuits

\begin{align*}
y_0 &= \overline{x}_2 x_1 + x_1 x_0 + x_2 \overline{x}_1 \overline{x}_0 \\
y_1 &= \overline{x}_2 \overline{x}_1 + \overline{x}_2 x_0 + \overline{x}_1 x_0
\end{align*}

\textbf{Applying DeMorgan}

\textbf{Mapping to NOR circuit}
2\textsuperscript{ND} STEP: MAP TO ADIABATIC CIRCUIT

- How to map gates?
- How to connect the gates to the power clocks?
- How to generate a corresponding waveform for these clocks?
- In all steps, switching rules need to be satisfied!
2\textsuperscript{ND} STEP: USING RETRACTILE CIRCUITS

- 1:1 mapping of OR gates to transmission gates
- Circuit is composed of four stages → four clocks are needed
- Clock signals trigger the computations through the stages
- Once stable, clocks trigger decomputations in reverse order
FURTHER READING: BROADENING DESIGN

Synthesis of Reversible Circuits

- One-pass Synthesis
  One-pass Design for Reversible Circuits: Combining Embedding and Synthesis for Reversible Logic, TCAD 2017

- Additionally Exploiting Coding Techniques
  Exploiting Coding Techniques for Logic Synthesis of Reversible Circuits. ASP-DAC 2018
  → https://iic.jku.at/eda/research/one_pass_design_of_reversible_circuits

Design Automation for Adiabatic Circuits

- Design Automation for Adiabatic Circuits, ASP-DAC 2018
  https://arxiv.org/abs/1809.02421

Efficient Representation of Reversible Logic

- Decision Diagrams
  QMDDs: Efficient Quantum Function Representation and Manipulation, TCAD, 2016
  → http://iic.jku.at/eda/research/quantum_dd/