Synthesis on Switching Lattices of Dimension-Reducible Boolean Functions

Marie Skłodowska-Curie grant agreement No 691178
(European Union’s Horizon 2020 research and innovation programme)

Anna Bernasconi¹, Valentina Ciriani², Luca Frontini², Gabriella Trucco²

¹Dipartimento di Informatica, Università di Pisa, Italy, anna.bernasconi@unipi.it
²Dipartimento di Informatica, Università degli Studi di Milano, Italy, {valentina.ciriani, luca.frontini, gabriella.trucco}@unimi.it

NANOxCOMP
Introduction

CMOS technology

- Transistor size has shrunk for decades
- The trend reached a critical point

The Moore's Law era is coming to an end

New emerging technologies

- Biotechnologies, molecular-scale self-assembled systems
- Graphene structures
- Switching lattices arrays

These technologies are in an early state

A novel synthesis approach should be focused on the properties of the devices

Synthesis efficiency can be an important factor for a technology choice

We focus on Synthesis for Switching Lattices
How Switching Lattices are made

**Nanowires** are one of the most promising technologies

- Nanowire circuits can be made with **self-assembled structures**
- **pn-junctions** are built crossing $n$-type and $p$-type nanowires
- **Low** $V_{\text{in}}$ voltage makes $p$-nanowires conductive and $n$-nanowires resistive
- **High** $V_{\text{in}}$ voltage makes $n$-nanowires conductive and $p$-nanowires resistive
Switching Lattices are **two-dimensional** arrays of **four-terminal** switches

- When switches are **ON** all terminals are connected, when **OFF** all terminals are disconnected
- Each switch is controlled by a boolean literal, 1 or 0
- The boolean function $f$ is the SOP of the literals along each path from **top** to **bottom**
- The function synthesized by the lattice is:
  \[ f = x_1 x_2 x_3 + x_1 x_2 x_5 x_6 + x_4 x_5 x_2 x_3 + x_4 x_5 x_6 \]
For an easier representation, the crossbars are converted to lattices:

- A ‘checkerboard’ notation is used
- Darker and white sites represent ON and OFF

a), b): the 4-terminal switching network and the lattice describing
\[ f = \overline{x_1}x_2\overline{x_3} + x_1x_2 + x_2x_3 \]

- c), d): the lattice evaluated on inputs (1,1,0) and (0,0,1)
The synthesis methods

**Altun-Riedel, 2012**

- Synthesizes $f$ and $f^D$ from **top to bottom** and **left to right**
- It produces lattices with size growing **linearly** with the SOP
- Time **complexity is polynomial** in the number of products

<table>
<thead>
<tr>
<th>TOP</th>
<th>LEFT</th>
<th>RIGHT</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_6$</td>
<td>$x_8$</td>
<td>$x_6$</td>
</tr>
<tr>
<td>$x_8$</td>
<td>$x_8$</td>
<td>$x_5$</td>
</tr>
<tr>
<td>$x_6$</td>
<td>$x_7$</td>
<td>$x_6$</td>
</tr>
<tr>
<td>$x_7$</td>
<td>$x_7$</td>
<td>$x_5$</td>
</tr>
<tr>
<td>$x_6$</td>
<td>$x_5$</td>
<td>$x_6$</td>
</tr>
<tr>
<td>$x_1$</td>
<td>$x_1$</td>
<td>$x_1$</td>
</tr>
<tr>
<td>$x_2$</td>
<td>$x_2$</td>
<td>$x_2$</td>
</tr>
<tr>
<td>$x_3$</td>
<td>$x_3$</td>
<td>$x_3$</td>
</tr>
</tbody>
</table>

**Gange-Søndergaard-Stuckey, 2014**

- $f$ is synthesized from **top to bottom**
- The synthesis problem is formulated as a **satisfiability problem**, then the problem is solved with a SAT solver
- The synthesis method searches for better implementations starting from an upper bound size
- The synthesis loses the possibility to generate both $f$ and $f^D$

<table>
<thead>
<tr>
<th>TOP</th>
<th>LEFT</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_4$</td>
<td>$x_6$</td>
</tr>
<tr>
<td>$x_2$</td>
<td>$x_5$</td>
</tr>
<tr>
<td>$x_1$</td>
<td>$x_6$</td>
</tr>
<tr>
<td>$x_3$</td>
<td>$0$</td>
</tr>
</tbody>
</table>

In both examples the synthesized function is:

$$f = \overline{x_8} \overline{x_7} x_6 x_3 \overline{x_2} x_1 + \overline{x_8} \overline{x_7} x_5 x_3 \overline{x_2} x_1 + x_4 x_3 \overline{x_2} x_1$$
The logic synthesis of 4-terminal switches can be very computational intensive.

Boolean function decomposition techniques:
- decompose a function according to a given decomposition scheme
- implement the decomposed blocks into a single lattice
- decomposed functions have less variables and/or a smaller on-set
- the implementation may be smaller and the synthesis less computational intensive

We use a decomposition based on D-reducible functions:

\[ f = \chi_A \cdot f_A \]

- \( \chi_A \) is the characteristic function of A
- \( f_A \) is the projection of \( f \) onto A
A function $f : \{0, 1\}^n \rightarrow \{0, 1\}$ is *D-reducible* if its ON-set is contained in an affine space $A \subseteq \{0, 1\}^n$, of dimension strictly smaller than $n$.

A D-reducible function $f$ is contained in an affine space $A$ smaller than $\{0, 1\}^n$

$$f = \chi_A \cdot f_A$$

- A is the unique associated space
- $\chi_A$ is the characteristic function of A
- $f_A$ is the projection of $f$ onto A
- $f$ and $f_A$ have the same number of points, but the points of $f_A$ are compacted in a smaller space

- the 70% of classical Espresso benchmark suite have at least one output that is D-reducible
- we want to analyze how this decomposition can be exploited in the switching lattice synthesis process
Example of D-red function

\[ f = x_1 x_3 x_4 + x_1 x_2 x_4 + x_1 x_2 x_3 x_4 + x_1 x_2 x_3 x_4 \]

- \( f \) is D-reducible
- we can project it onto a space of dimension three.
- \( f \) and \( f_A \) have the same number of points, but the point of \( f_A \) are now compacted in a smaller space
- \( f_A = x_2 x_3 + x_1 x_2 + x_2 x_3 \) and \( (x_1 \oplus x_4) \) represents the the associated affine space \( A \)

\[ f = (x_1 \oplus x_4)(x_2 x_3 + x_1 x_2 + x_2 x_3) \]
### Disjunction and conjunction of lattices

<table>
<thead>
<tr>
<th><strong>f + g</strong></th>
<th><strong>f · g</strong></th>
</tr>
</thead>
</table>
| - separate the paths from top to bottom for \( f \) and \( g \)  
- add a column of 0s  
- add padding rows of 1s if lattices have different number of rows  | - any top-bottom path of \( f \) is joined to any top-bottom path of \( g \)  
- add a row of 1s  
- add padding columns of 0s if lattices have different number of columns |

**Example:**

<table>
<thead>
<tr>
<th>( f )</th>
<th>( g )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>( \vdots )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( f )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( g )</th>
</tr>
</thead>
</table>
A lattice for a D-reducible function is obtained merging the lattice of $\chi_A$ and the projection $f_A$, placed in physically separated regions of a single lattice.

$$f = \chi_A \cdot f_A$$

both $f_A$ and $\chi_A$ depends on fewer variables than $f$:
- the synthesis should be less computational intensive
- it is possible that the final lattice has a smaller area
2D-Red: Two-variables EXOR

2D-Reducible functions

- we focus our work on a subset of D-Red functions: 2D-Red
- the affine space of 2D-Red can be represented by products (AND) of two literals EXOR

- two-variable EXOR factors lattices are simple to synthesize
- the dimension of a two-variables EXOR lattice is $2 \times 2$

$$f_{\text{EXOR}} = x_1 \oplus x_2$$
Two-variables Exor and literals

For instance, \( \chi_A = (x_1 \oplus x_3) \cdot (x_2 \oplus x_4) \cdot \overline{x_5} \cdot (x_1 \oplus x_8) \), subspace of \( \{0, 1\}^8 \)

\[
\begin{align*}
    x_1 \oplus x_3 &= 1 \\
    x_2 \oplus x_4 &= 1 \\
    \overline{x_5} &= 1 \\
    x_1 \oplus x_8 &= 1
\end{align*}
\]

\[
\begin{align*}
    x_1 &= x_3 \\
    x_2 &= \overline{x_4} \\
    x_5 &= 0 \\
    x_1 &= \overline{x_8}
\end{align*}
\]

are derived the equalities:

\[
\begin{align*}
    x_1 &= x_3 = \overline{x_8} \\
    x_2 &= \overline{x_4} \\
    x_5 &= 0,
\end{align*}
\]

result: partition of a subset of the input variables:

\[
\{\{0, x_5\}, \{x_1, x_3, \overline{x_8}\}, \{x_2, \overline{x_4}\}\},
\]
Two literals Exor – theorem

**Theorem**

- Let $A$ be an affine subspace of $\{0, 1\}^n$ described by the product of single literals and two literals EXOR.
- Let $P_A$ be the partition of the subset of input variables that defines $A$, and let $n' \leq n$ be the number of distinct variables occurring in $P_A$.
- Suppose that $P_A$ contains $\ell$ subsets of literals, in addition to the subset $C$ with the constant 0.
- Let $c$ be the number of literals in $C$.

Then $A$ can be implemented with a lattice of area $r \times 2$, where the number $r$ of rows is given by

$$
r = \begin{cases} 
n' & \text{if } c \geq \ell - 1 \\
n' + \ell - 1 - c & \text{if } c < \ell - 1
\end{cases}
$$
Synthesis example

\[ f = x_1x_2\overline{x}_3\overline{x}_4x_5x_8x_9x_{10}x_{11} + x_2\overline{x}_2\overline{x}_3\overline{x}_4\overline{x}_5x_8x_9x_{10}x_{11} + x_1\overline{x}_2\overline{x}_3\overline{x}_4\overline{x}_5\overline{x}_7x_8 + \]
\[ + x_1\overline{x}_2x_3x_4\overline{x}_7x_8 + x_1\overline{x}_2\overline{x}_3\overline{x}_4\overline{x}_5\overline{x}_7x_8 + x_1\overline{x}_2x_3x_4\overline{x}_7x_8 \]

\[ f_A = \overline{x}_2x_3\overline{x}_7 + \overline{x}_2\overline{x}_5\overline{x}_7 + x_2\overline{x}_3x_5\overline{x}_6 + \overline{x}_2x_3x_9x_{10}x_{11} + x_2\overline{x}_3x_5x_9x_{10}x_{11} \]

\[ \chi_A = x_1x_8(\overline{x}_3 \oplus x_4) \]
Experiments

- Benchmarks are taken from LGSynth93
- Each benchmark output is considered as a separate boolean function
- A total of 385 functions
- We evaluate the results just for two literals EXOR
- We use a collection of Python scripts and a SAT solver to perform the Gange-Søndergaard-Stuckey synthesis

- The algorithm has been implemented in C
- The experiments have been run on a machine with 16 CPU @2.5 GHz, running Centos 6.6
The Experiments

<table>
<thead>
<tr>
<th>f-Name</th>
<th>not decomposed</th>
<th>decomposed</th>
<th>decomposed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>standard</td>
<td>D-red</td>
<td>(\chi_A)</td>
</tr>
<tr>
<td></td>
<td>Col × Row</td>
<td>Col × Row</td>
<td>Col × Row</td>
</tr>
<tr>
<td>amd(5)</td>
<td>6 × 2</td>
<td>2 × 8</td>
<td>2 × 6</td>
</tr>
<tr>
<td>amd(7)</td>
<td>5 × 5</td>
<td>3 × 6</td>
<td>1 × 1</td>
</tr>
<tr>
<td>exp(6)</td>
<td>5 × 4</td>
<td>3 × 7</td>
<td>1 × 2</td>
</tr>
<tr>
<td>exp(10)</td>
<td>6 × 12</td>
<td>6 × 5</td>
<td>1 × 2</td>
</tr>
<tr>
<td>in2(7)</td>
<td>17 × 26</td>
<td>17 × 26</td>
<td>1 × 1</td>
</tr>
<tr>
<td>t1(0)</td>
<td>6 × 9</td>
<td>3 × 8</td>
<td>1 × 1</td>
</tr>
<tr>
<td>t1(1)</td>
<td>7 × 9</td>
<td>7 × 9</td>
<td>1 × 1</td>
</tr>
</tbody>
</table>
Results of the Experiments

Decomposing the D-reducible functions we obtain:

- More compact area in 15% of cases
- Average area reduction of about 24%
- Average computing time reduction of about 50%

- In many cases the method Gange-Søndergaard-Stuckey fails in computing a result in a reasonable time
- We set a maximum of ten minutes for each SAT execution
- If synthesis is stopped we use the synthesis method by Altun-Riedel
Conclusions

- A new method for the synthesis of lattices with reduced size
- Based on decomposition of D-reducible function
- The lattice synthesis benefits from this decomposition:
  - smaller lattices: at least 24% of area reduction in 15% of functions
  - average reduction of computing time by 50%

In future works we will apply more complex type of decompositions
- considering D-reducible functions, with affine spaces described with EXOR factors of fan-in greater than two
- other decomposition methods