Composition of Switching Lattices and Autosymmetric Boolean Function Synthesis

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

Anna Bernasconi\textsuperscript{1}, Valentina Ciriani\textsuperscript{2}, Luca Frontini\textsuperscript{2}, Gabriella Trucco\textsuperscript{2}

\textsuperscript{1}Dipartimento di Informatica, Università di Pisa, Italy, anna.bernasconi@unipi.it

\textsuperscript{2}Dipartimento di Informatica, Università degli Studi di Milano, Italy, 
\{valentina.ciriani, luca.frontini, gabriella.trucco\}@unimi.it

NANO\textsuperscript{x}COMP
Introduction

CMOS technology

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

The Moore's Law era is coming to 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 is necessary, focused on the properties of the devices

**Synthesis efficiency can be the main factor for a technology choice**

We focus our work on Synthesis for Switching Lattices
Switching Lattices
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** $V_{in}$ voltage makes p-nanowires conductive and n-nanowires resistive
- **High** $V_{in}$ voltage makes n-nanowires conductive and p-nanowires resistive
The Switching Lattices

Switching Lattices are **two-dimensional** array 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**
- $f = x_1x_2x_3 + x_1x_2x_5x_6 + x_4x_5x_2x_3 + x_4x_5x_6$
From Crossbars to Lattices

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
  \( f = \overline{x_1} \overline{x_2} x_3 + x_1 x_2 + x_2 x_3 \)
- 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

\[
f = \overline{x}_8 \overline{x}_7 \overline{x}_6 x_3 \overline{x}_2 x_1 + \overline{x}_8 \overline{x}_7 x_5 x_3 \overline{x}_2 x_1 + x_4 x_3 \overline{x}_2 x_1
\]

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 $f^D$
To optimize lattice synthesis there are different approaches, but common goals:

- Produce optimal-size lattices
- Reduce synthesis time
- Create efficient methods for sub-optimal lattice synthesis

Use of sub-optimal lattices when optimal synthesis requires too much computing time or memory
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 or multiple lattices
- 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 preprocessing technique that exploits the properties of the **Autosymmetric** boolean functions.
Autosymmetric functions
Consider a Boolean function \( f : \{0, 1\}^n \rightarrow \{0, 1\} \): the function \( f \) is **closed under** a vector \( \alpha \in \{0, 1\}^n \), if for each vector \( w \in \{0, 1\}^n \), \( w \oplus \alpha \in f \) if and only if \( w \in f \).

The set \( L_f = \{ \beta : f \text{ is closed under } \beta \} \) is a **vector subspace** of \( (\{0, 1\}^n, \oplus) \). The set \( L_f \) is called the **vector space** of \( f \).

**Definition:** A completely specified Boolean function \( f \) is **\( k \)-autosymmetric** if its vector space \( L_f \) has dimension \( k \).

**Definition:** Let \( V \) be a vector subspace of \( (\{0, 1\}^n, \oplus) \). The set \( A = \alpha \oplus V \), \( \alpha \in \{0, 1\}^n \), is an **affine space** over \( V \) with translation point \( \alpha \).

The points of \( f \) can be partitioned into \( \ell = |f|/2^k \) disjoint sets, where \( |f| \) denotes the number of points of \( f \); all these sets are affine spaces over \( L_f \).

\[
f = \bigcup_{i=1}^{\ell} (w^i \oplus L_f)
\]
Autosymmetric functions can be reduced to “equivalent, but smaller” functions if $f$ is $k$-autosymmetric,

- $f_k$ is a function over $n - k$ variables, $y_1, y_2, \ldots, y_{n-k}$, such that
  \[ f(x_1, \ldots, x_n) = f_k(y_1, \ldots, y_{n-k}) \]
- $y_i$ is an EXOR combination of a subset of $x_i$'s.
- These combinations are $\text{EXOR}(X_i)$, where $X_i \subseteq X$
- $y_i = \text{EXOR}(X_i)$, $i = 1, \ldots, n - k$, are called reduction equations
- $f_k$ is called a restriction of $f$

$f_k$ is “equivalent” to, but smaller than $f$, and has $|f|/2^k$ points only.
Example of autosymmetric function decomposition

- \( f = \{0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1011, 1101, 1110\} \)
- Vector space \( L_f = \{0000, 0011, 0101, 0110\} \)
- Canonical variables \( x_2 \) and \( x_3 \) (independent variables on \( L_f \)).

- We can build \( f_2 \) by taking \( f_{x_2=0, x_3=0} = \{00, 01, 10\} \): \( f_2(y_1, y_2) = \overline{y_1 y_2} \).
- The homogeneous system whose solutions are \( \{0000, 0011, 0101, 0110\} \) is:

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

Autosymmetric boolean functions have already **studied and algebraically characterized**

The space \( L_f \), the function \( f_k \) and the reduction equation can be **calculated in a polynomial time**
Example of autosymmetric function decomposition

- $f = \{0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1011, 1101, 1110\}$
- Vector space $L_f = \{0000, 0011, 0101, 0110\}$
- Canonical variables $x_2$ and $x_3$ (independent variables on $L_f$).

Thus the reduction equations are given by

\[
\begin{align*}
y_1 &= x_1 \\
y_2 &= x_2 \oplus x_3 \oplus x_4.
\end{align*}
\]

$f$ can be represented as:

\[
f(x_1, x_2, x_3, x_4) = f_2(y_1, y_2) = \overline{y_1y_2} = \overline{x_1(x_2 \oplus x_3 \oplus x_4)}.
\]
Lattice implementation of autosymmetric functions

\[ y_1 = \text{EXOR} (X_1) \]

\[ y_i = \text{EXOR} (X_i) \]

\[ y_j = x_t \]

(a) (b)
**Disjunction and conjunction of lattices**

**$f + g$**
- 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

**$f \cdot g$**
- 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
Lattices of EXOR functions

EXOR factors lattices are simple to synthesize

- the dimension of a two-variables EXOR lattice is $2 \times 2$
- the dimension of a three-variables EXOR lattice is $4 \times 3$
Autosymmetric function: example

- \( f(x_1, x_2, x_3, x_4) = x_1 \oplus x_2 \oplus x_3 \oplus x_4. \)
- decomposing: \( f = g(y_1, y_2) = y_1 \oplus y_2, \) where \( y_1 = x_1 \oplus x_2 \) and \( y_2 = x_3 \oplus x_4 \)
- Multi-lattice: the sum of the areas of the lattices is smaller than the area of the optimum single-lattice

|   |   |   |   |
|---|---|---|---|
|   | \( x_1 \) | 0 | \( x_1 \) |
| \( x_2 \) | \( x_2 \) | 0 | \( x_2 \) |
| \( \bar{x}_1 \) | \( \bar{x}_2 \) | 0 | \( \bar{x}_2 \) |
| 1 | 1 | 0 | 1 |
| \( x_3 \) | \( x_3 \) | 0 | \( x_3 \) |
| \( x_4 \) | \( x_4 \) | 0 | \( x_4 \) |

\( x_1 \) \( x_2 \) \( y_1 \) \( y_2 \)
Experiments

- Benchmarks are taken from LGSynth93
- Each benchmark output is considered as a separate boolean function
- A total of 607 functions including 53 autosymmetric functions
- 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
## Auto-symmetric functions decomposition results

| F-Name       | Altun-Riedel | Gange-Søndergaard-Stuckey |
|--------------|--------------|----------------------------|
|              | standard     | Decomposed                 | standard     | Decomposed                 |
|              | Row×Col      | Row×Col + XOR area         | Row×Col      | Row×Col + XOR area         |
| add6(0)      | 2×2          | 1×1 + 4                    | 2×2          | 1×1 + 4                    |
| add6(1)      | 6×6          | 3×3 + 4                    | 5×3          | 3×3 + 4                    |
| dekoder(0)   | 4×2          | 3×1 + 4                    | 4×2          | 3×1 + 4                    |
| dekoder(1)   | 3×2          | 2×1 + 4                    | 3×2          | 2×1 + 4                    |
| rd53(1)      | 10×10        | 6×5 + 16                   | –            | 4×3 + 16                   |
| sqn(0)       | 17×16        | 7×7 + 8                    | –            | 3×5 + 8                    |

– : Gange-Søndergaard-Stuckey synthesis does not finish in 10min
Conclusions

- **Smaller lattices**: at least 53% of area reduction in 48% of functions.
- **Affordable computing time**, in some cases is possible to find a solution in less time than the optimum one.
- Some decomposed functions has **smaller total area** w.r.t. the lattice size in optimum case.
- Increase the number of lattices and the final lattice has more complex signal routing.
Thank you!