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 Bernasconi[1], Valentina Ciriani[2], *Luca Frontini*[2], Gabriella Trucco[2]

[1]Dipartimento di Informatica, Università di Pisa, Italy, anna.bernasconi@unipi.it

[2]Dipartimento 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

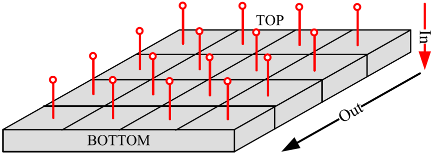
Synthesis efficiency can be an important factor for a technology choice

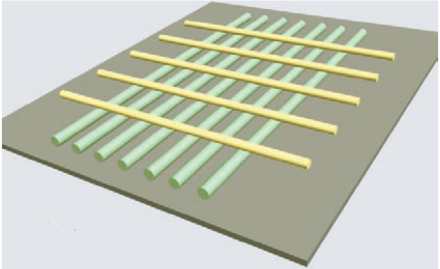
We focus on Synthesis for Switching Lattices

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_{in} voltage makes *p*-nanowires conductive and *n*-nanowires resistive
- **High** V_{in} voltage makes *n*-nanowires conductive and *p*-nanowires resistive

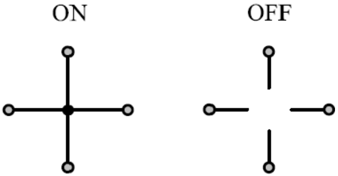


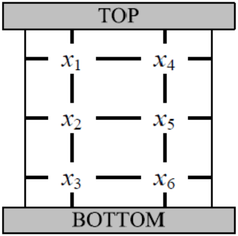


The Switching Lattices

Switching Lattices are **two-dimensional** 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:
  $f = x_1x_2x_3 + x_1x_2x_5x_6 +$
  $+x_4x_5x_2x_3 + x_4x_5x_6$

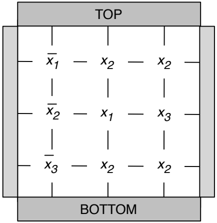




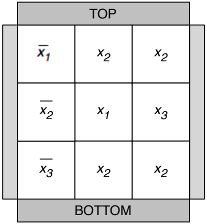
From Crossbars to Lattices

For an easier representation, the **crossbars** are converted to **lattices**:

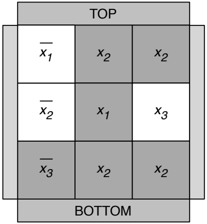
- A 'checkerboard' notation is used
- Darker and white sites represent **ON** and **OFF**
- a), b): the 4-terminal switching network and the lattice describing $f = \overline{x}_1\overline{x}_2\overline{x}_3 + x_1x_2 + x_2x_3$
- c), d): the lattice evaluated on inputs (1,1,0) and (0,0,1)



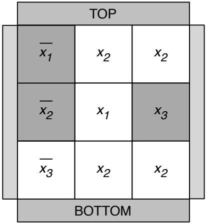
(a)



(b)



(c)

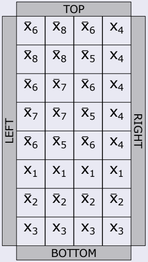


(d)

The synthesis methods

Altun-Riedel, 2012

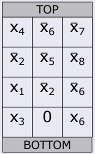
- Synthesizes f and f^D from **top to bottom** and **left to right**
- It produces lattices with size growing **linearly** with the SOP
- Time **complexity is polynomial** in the number of products



$\overline{x}_6$	$\overline{x}_8$	$\overline{x}_6$	x_4
$\overline{x}_8$	$\overline{x}_8$	$\overline{x}_5$	x_4
$\overline{x}_6$	$\overline{x}_7$	$\overline{x}_6$	x_4
$\overline{x}_7$	$\overline{x}_7$	$\overline{x}_5$	x_4
$\overline{x}_6$	$\overline{x}_5$	$\overline{x}_6$	x_4
x_1	x_1	x_1	x_1
$\overline{x}_2$	$\overline{x}_2$	$\overline{x}_2$	$\overline{x}_2$
x_3	x_3	x_3	x_3

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



x_4	$\overline{x}_6$	$\overline{x}_7$
$\overline{x}_2$	$\overline{x}_5$	$\overline{x}_8$
x_1	$\overline{x}_2$	$\overline{x}_6$
x_3	0	x_6

In both examples the synthesized function is:

$$f = \overline{x}_8\overline{x}_7\overline{x}_6x_3\overline{x}_2x_1 + \overline{x}_8\overline{x}_7\overline{x}_5x_3\overline{x}_2x_1 + x_4x_3\overline{x}_2x_1$$

Decomposition Techniques and Lattices

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**:

$$f = \chi_A \cdot f_A$$

- χ_A is the characteristic function of A
- f_A is the projection of f onto A

D-reducible Boolean functions

A function $f : \{0,1\}^n \to \{0,1\}$ 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
- χ_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

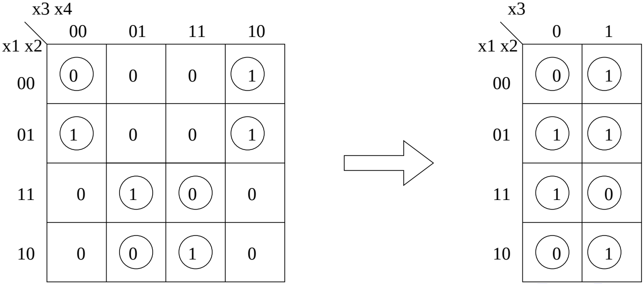
- the 70% of classical ESPRESSO benchmark suite have at least one output that is D-reducible
- we want to analyze how this decomposition can be exploited in the switching lattice synthesis process

Example of D-red function

$$f = \overline{x}_1 x_3 \overline{x}_4 + \overline{x}_1 x_2 \overline{x}_4 + x_1 \overline{x}_2 x_3 x_4 + x_1 x_2 \overline{x}_3 x_4$$

- 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
- $f_A = \overline{x}_2 x_3 + \overline{x}_1 x_2 + x_2 \overline{x}_3$ and $(x_1 \oplus \overline{x}_4)$ represents the the associated affine space A

$$f = (x_1 \oplus \overline{x}_4)(\overline{x}_2 x_3 + \overline{x}_1 x_2 + x_2 \overline{x}_3)$$



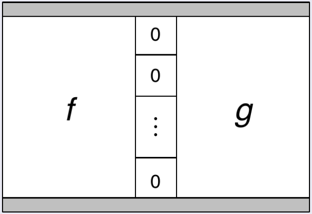
x1 x2 \ x3 x4	00	01	11	10
00	0	0	0	1
01	1	0	0	1
11	0	1	0	0
10	0	0	1	0

x1 x2 \ x3	0	1
00	0	1
01	1	1
11	1	0
10	0	1

Disjunction and conjunction of lattices

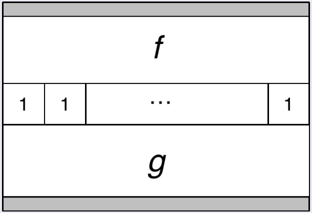
$f + g$

- 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



$f \cdot g$

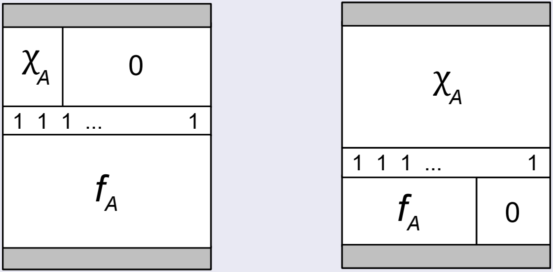
- any top-bottom path of f is joined to any top-bottom path of g
- add a row of 1s
- add padding columns of 0s if lattices have different number of columns



D-reducible functions lattice implementation

A lattice for a D-reducible function is obtained **merging the lattice of χ_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 χ_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**

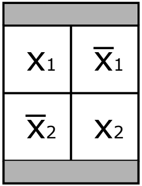
2D-Red: Two-variables EXOR

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 2×2

$$f_{EXOR} = x_1 \oplus x_2$$



x_1	$\overline{x}_1$
$\overline{x}_2$	x_2

Two-variables Exor and literals

For instance, $\chi_A = (x_1 \oplus \overline{x}_3) \cdot (x_2 \oplus x_4) \cdot \overline{x}_5 \cdot (x_1 \oplus x_8)$, subspace of $\{0,1\}^8$

$$\begin{cases} x_1 \oplus \overline{x}_3 & = 1 \\ x_2 \oplus x_4 & = 1 \\ \overline{x}_5 & = 1 \\ x_1 \oplus x_8 & = 1 \end{cases} \implies \begin{cases} x_1 & = x_3 \\ x_2 & = \overline{x}_4 \\ x_5 & = 0 \\ x_1 & = \overline{x}_8 \end{cases}$$

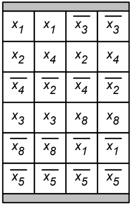
are derived the equalities:

$$\begin{aligned} x_1 &= x_3 \quad = \overline{x}_8 \\ x_2 &= \overline{x}_4 \\ x_5 &= 0\,, \end{aligned}$$

result: partition of a subset of the input variables:

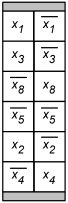
$$\{\{0, x_5\}, \{x_1, x_3, \overline{x}_8\}, \{x_2, \overline{x}_4\}\}\,,$$

standard synthesis:



x_1	x_1	$\overline{x}_3$	$\overline{x}_3$
x_2	x_4	x_2	x_4
$\overline{x}_4$	$\overline{x}_2$	$\overline{x}_4$	$\overline{x}_2$
x_3	x_3	x_8	x_8
$\overline{x}_8$	$\overline{x}_8$	$\overline{x}_1$	$\overline{x}_1$
$\overline{x}_5$	$\overline{x}_5$	$\overline{x}_5$	$\overline{x}_5$

2-EXOR synthesis:



x_1	$\overline{x}_1$
x_3	$\overline{x}_3$
$\overline{x}_8$	x_8
$\overline{x}_5$	$\overline{x}_5$
x_2	$\overline{x}_2$
$\overline{x}_4$	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 ℓ 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

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

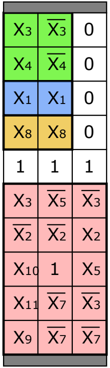
Synthesis example

$$\begin{aligned} f = \quad & x_1x_2\overline{x}_3\overline{x}_4x_5x_8x_9x_{10}x_{11} + x_2\overline{x}_2\overline{x}_3\overline{x}_4\overline{x}_5x_8x_9x_{10}x_{11} + x_1\overline{x}_2\overline{x}_3\overline{x}_4\overline{x}_5\overline{x}_7x_8 + \\ & + x_1\overline{x}_2x_3x_4\overline{x}_7x_8 + x_1\overline{x}_2\overline{x}_3\overline{x}_4\overline{x}_5\overline{x}_7x_8 + x_1\overline{x}_2x_3x_4\overline{x}_7x_8 \end{aligned}$$

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

$$\chi_A = x_1x_8\left(\overline{x_3 \oplus x_4}\right)$$

$\overline{x}_4$	$\overline{x}_2$	$\overline{x}_2$	$\overline{x}_2$	$\overline{x}_2$	$\overline{x}_2$
x_2	$\overline{x}_5$	$\overline{x}_5$	x_4	$\overline{x}_5$	$\overline{x}_5$
$\overline{x}_3$	$\overline{x}_3$	$\overline{x}_3$	x_4	$\overline{x}_3$	x_4
x_5	$\overline{x}_2$	$\overline{x}_2$	x_2	$\overline{x}_2$	$\overline{x}_2$
$\overline{x}_4$	$\overline{x}_4$	$\overline{x}_4$	x_3	$\overline{x}_4$	x_3
x_1	x_1	x_1	x_1	x_1	x_1
x_{11}	x_{11}	$\overline{x}_7$	$\overline{x}_7$	$\overline{x}_7$	$\overline{x}_7$
x_9	x_9	$\overline{x}_7$	$\overline{x}_7$	$\overline{x}_7$	$\overline{x}_7$
x_{10}	x_{10}	$\overline{x}_7$	$\overline{x}_7$	$\overline{x}_7$	$\overline{x}_7$
x_8	x_8	x_8	x_8	x_8	x_8



x_3	$\overline{x}_3$	0
x_4	$\overline{x}_4$	0
x_1	x_1	0
x_8	x_8	0
1	1	1
x_3	$\overline{x}_5$	$\overline{x}_3$
$\overline{x}_2$	$\overline{x}_2$	x_2
x_{10}	1	x_5
x_{11}	$\overline{x}_7$	$\overline{x}_3$
x_9	$\overline{x}_7$	$\overline{x}_7$

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

	not decomposed	decomposed		
	standard	**D-red**	χ_A	f_A
f-Name	Col×Row	Col×Row	Col×Row	Col×Row
amd(5)	**6**×**2**	2×8	2 × 6	2 × 2
amd(7)	5×5	**3**×**6**	1 × 1	3 × 5
exp(6)	**5**×**4**	3×7	1 × 2	3 × 5
exp(10)	6×12	**6**×**5**	1 × 2	5 × 4
in2(7)	17×26	17×26	1 × 1	17 × 25
t1(0)	6×9	**3**×**8**	1 × 1	3 × 7
t1(1)	7×9	7×9	1 × 1	7 × 8

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