Testability of Switching Lattices in the Cellular Fault Model

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

Anna Bernasconi[1], Valentina Ciriani[2], *Luca Frontini*[3]

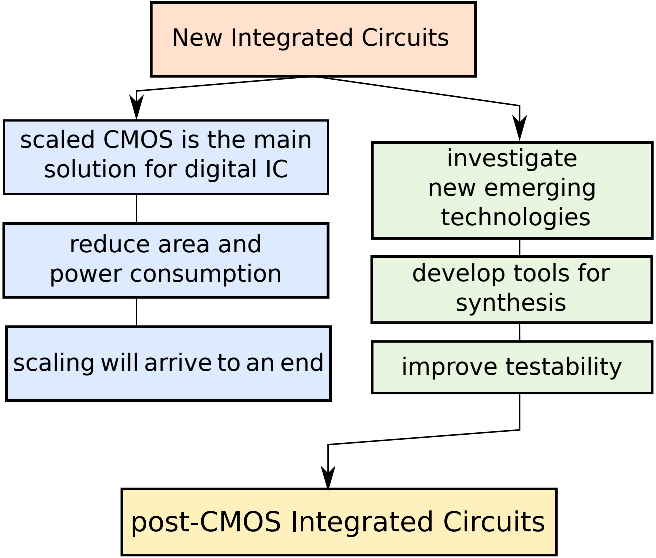
[1]Dipartimento di Informatica, Università di Pisa, Italy, anna.bernasconi@unipi.it

[2]Dipartimento di Informatica, Università degli Studi di Milano, Italy, valentina.ciriani@unimi.it

[3]I.N.F.N., sezione di Milano, Italy, luca.frontini@mi.infn.it

NANOxCOMP

Emerging Technologies

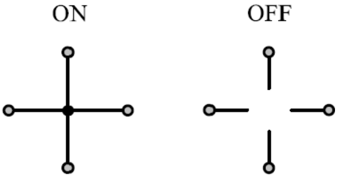


Overview

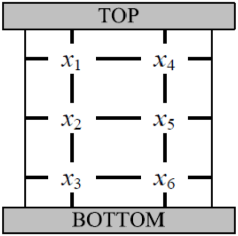
1. Preliminaries on **switching lattices** and on the **cellular fault model** (CFM) .
2. Analysis of **cellular fault testability**.
3. Two methods for **improving the testability** of adjacent cellular faults in a lattice.
4. Experimental results.
5. Conclusions.

The Switching Lattices

Switching Lattices are **two-dimensional** array of **four-terminal** switches. They are self-assembled devices fabricated with nano-fabrication techniques.



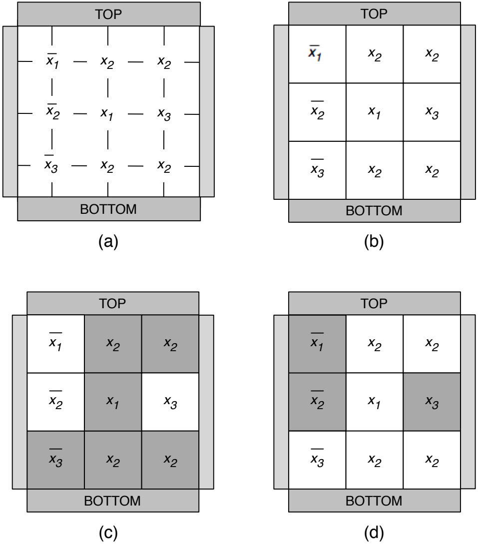
- When switches are ON all terminals are connected, when OFF all terminals are disconnected
- each switch is controlled by a boolean literal, 1 or 0
- the boolean function f is the SOP of the literals along each path from **top** to **bottom**
- $f = x_1x_2x_3 + x_1x_2x_5x_6 +$ $+x_4x_5x_2x_3 + x_4x_5x_6$



From Crossbars to Lattices

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

- a), b): the 4-terminal switching network and the lattice describing $f = \overline{x}_1\overline{x}_2\overline{x}_3 + x_1x_2 + x_2x_3$
- 'checkerboard' notation: darker and white sites represent ON and OFF
- c), d): the lattice with input (1,1,0) and (0,0,1)



- 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

TOP

LEFT | RIGHT

$\overline{x}_6$	$\overline{x}_8$	$\overline{x}_6$	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

BOTTOM

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

Given a **Boolean function** f and its dual function f^D:

1. find an **irredundant SOP representation for f and $\mathbf{f^D}$**:
   $SOP(f) = p_1 + p_2 + \dots p_s$,
   $SOP(f^D) = q_1 + q_2 + \dots q_r$;
2. form a **r $\times$ s switching lattice** and randomly assign each product p_j of $SOP(f)$ to a column and each product q_i of $SOP(f^D)$ to a row;
3. for all $1 \leq i \leq r$ and all $1 \leq j \leq s$, randomly **assign to the lattice cell $\mathbf{c_{i,j}}$ one literal that is shared by $\mathbf{q_i}$ and $\mathbf{p_j}$**.

Synthesis methods: Gange-Søndergaard-Stuckey, 2014

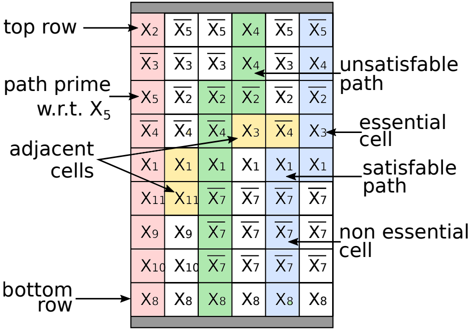
- f is synthesized from **top to bottom**
- the synthesis problem is formulated as a **satisfiability problem**, then the problem is solved with a SAT solver
- the synthesis method searches for better implementations starting from an upper bound size
- the synthesis loses the possibility to generate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

TOP		
$\bar{x}_4$	x_6	x_7
x_2	x_5	x_8
$\bar{x}_1$	x_2	x_6
$\bar{x}_3$	0	$\bar{x}_6$
BOTTOM		

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

Definitions

- A path is **unsatisfiable** if contains both a variable x and $\overline{x}$.
- The **product associated to a satisfiable path** is the conjunction of all literals of the path.
- An **accepting path** for a minterm v in a lattice is a satisfiable path whose associated product covers v.
- A path is **prime** w.r.t. a literal l_i, if the product obtained removing l_i from the path is not an implicant of the function.
- The cell c is **essential** if there exists at least a minterm v in the on-set whose accepting paths always contain c.



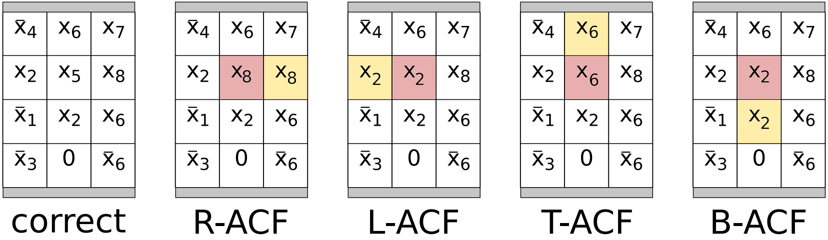
$$
\begin{aligned}
f = \; & x_1 x_2 \overline{x}_3 \overline{x}_4 x_5 x_8 x_9 x_{10} x_{11} + \\
& + x_1 \overline{x}_2 \overline{x}_3 \overline{x}_4 \overline{x}_5 x_8 x_9 x_{10} x_{11} + \\
& + x_1 \overline{x}_2 \overline{x}_3 \overline{x}_4 \overline{x}_5 \overline{x}_7 x_8 + x_1 \overline{x}_2 x_3 x_4 \overline{x}_7 x_8 + \\
& + x_1 \overline{x}_2 \overline{x}_3 \overline{x}_4 \overline{x}_5 \overline{x}_7 x_8 + x_1 \overline{x}_2 x_3 x_4 \overline{x}_7 x_8
\end{aligned}
$$

Cellular fault model in a Lattice

Let $l_{i,j}$ be the literal in the cell $c_{i,j}$:

- **R-ACF** Right Adjacent Cellular Fault is the cellular fault $(c_{i,j}, l_{i,j}, l_{i,j+1})$;
- **L-ACF** Left Adjacent Cellular Fault is the cellular fault $(c_{i,j}, l_{i,j}, l_{i,j-1})$;
- **T-ACF** Top Adjacent Cellular Fault is the cellular fault $(c_{i,j}, l_{i,j}, l_{i-1,j})$;
- **B-ACF** Bottom Adjacent Cellular Fault is the cellular fault $(c_{i,j}, l_{i,j}, l_{i+1,j})$.

T^L, T^R, T^B, and T^T are the number of testable cells with a L-ACF, R-ACF, B-ACF, and T-ACF.



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

Testability in cellular fault model

- **CF** $(\mathbf{c}, \mathbf{l_c}, \mathbf{l_f})$ in cell c with controlling literal l_c and faulty literal l_f.
- The **test set** of the CF is the set $T_{(l_c \leftarrow l_f)}$ of all input vectors that give an uncorrected output on the faulty lattice.
- $T_{(l_c \leftarrow l_f)}$ are called **test vectors**.
- A fault is **testable if and only if its test set is not empty**.

TOP		
$\overline{x}_1$	x_2	x_2
$\overline{x}_2$	x_1	x_3
$\overline{x}_3$	x_2	x_2
BOTTOM		

Correct

TOP		
$\overline{x}_1$	x_2	$\overline{x}_1$
$\overline{x}_2$	x_1	x_3
$\overline{x}_3$	x_2	x_2
BOTTOM		

a)

TOP		
$\overline{x}_1$	x_2	x_3
$\overline{x}_2$	x_1	x_3
$\overline{x}_3$	x_2	x_2
BOTTOM		

b)

Example: $f = \overline{x}_1\overline{x}_2\overline{x}_3 + x_1x_2 + x_2x_3$.

a) $l_f = \overline{x}_1$, test vector $\overline{x}_1x_3x_2$, the fault is testable;

b) $l_f = x_2$, no test vectors, the fault is not testable.

Proposition 1 and 2: testability of CFM

- A $\mathbf{CF(c, l_c, l_f)}$ in a lattice cell c with literal l_c **is testable if and only if the** $\mathbf{CF(c, l_c, \overline{l_c})}$ **is testable** and the test set $T_{(l_c \leftarrow \bar{l}_c)}$ contains at least one input vector where l_f and l_c assume different values.

Correct:

TOP		
$\bar{x}_1$	x_2	x_2
$\bar{x}_2$	x_1	x_3
$\bar{x}_3$	x_2	x_2
BOTTOM		

a):

TOP		
$\bar{x}_1$	x_2	$\bar{x}_2$
$\bar{x}_2$	x_1	x_3
$\bar{x}_3$	x_2	x_2
BOTTOM		

b):

TOP		
$\bar{x}_1$	x_2	$\bar{x}_1$
$\bar{x}_2$	x_1	x_3
$\bar{x}_3$	x_2	x_2
BOTTOM		

a) $\text{CF}(c, x_2, \overline{x}_2)$, b) $\text{CF}(c, x_2, \overline{x}_1)$. a) is testable with test vector $\overline{x}_1 x_2 x_3$

- A **CF** $\mathbf{(c, l_c, \bar{l}_c)}$ **cannot be tested if** for each path p through c, the subpath $\mathbf{p' = p \setminus \{c\}}$ **is unsatisfiable**.

Correct:

TOP		
$\bar{x}_1$	x_2	x_2
$\bar{x}_2$	$\bar{x}_2$	x_3
$\bar{x}_3$	x_2	x_2
BOTTOM		

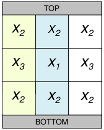
a):

TOP		
$\bar{x}_1$	x_3	x_2
$\bar{x}_2$	$\bar{x}_2$	x_3
$\bar{x}_3$	x_2	x_2
BOTTOM		

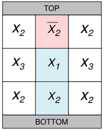
The blue subpath is unsatisfable, so CF (c, x_2, x_3) is cannot be tested.

Testability of the CF$(c, l_c, \bar{l}_c)$

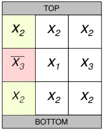
- The **CF $(\mathbf{c}, \mathbf{l_c}, \bar{\mathbf{l}}_\mathbf{c})$ can be tested on a path** $p = p' \cup \{l_c\}$, where p' is satisfiable and contains an **occurrence of $\bar{\mathbf{l}}_\mathbf{c}$** if and only if **p is prime** with respect to l_c.



Correct



a)

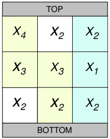


b)

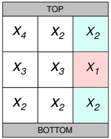
The blue path is prime, the green is not prime.

In a) the CF$(c, x_2, \overline{x}_2)$ is testable, in b) the CF$(c, x_3, \overline{x}_3)$ is not testable.

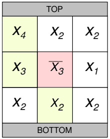
- The **CF** $(c, l_c, \bar{l}_c)$ **can be tested on a path** $p = p' \cup \{l_c\}$, where p' is satisfiable and contains an **occurrence of $\mathbf{l_c}$** if and only if **c** is **essential**.



Correct



a)

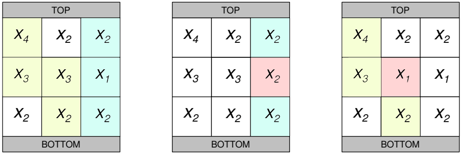


b)

The cell c in a) is essential and the CF is testable, the cell c in b) is not essential and the CF not testable.

Testability of the CF$(c, l_c, \bar{l}_c)$ and Theorem

- The CF $(c, l_c, \bar{l}_c)$ can be tested on a path $p = p' \cup \{l_c\}$, where p' is satisfiable and does not contain l_c or $\bar{l}_c$ if and only if p is prime with respect to l_c or the cell c is essential.



Correct a) b)

The cell c in a) is essential and the CF is testable, the cell c in b) is not essential

Theorem

For any literal $\mathbf{l_f}$ **different form** $\mathbf{l_c}$*, the CF* $(\mathbf{c}, \mathbf{l_c}, \mathbf{l_f})$ *in a lattice cell* c *with controlling literal* l_c **is testable if and only if** $\mathbf{T}_{(\mathbf{l_c} \leftarrow \bar{\mathbf{l}}_c)} \cap \mathbf{B}_{\mathbf{l_c} \neq \mathbf{l_f}} \neq \emptyset$.

$B_{l_c \neq l_f}$ is subset of the space $\{0, 1\}^n$ where $l_c \neq l_f$ assume different values

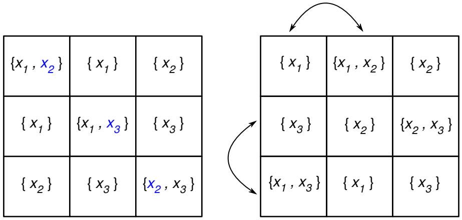
- With the test set for the fault $(c, l_c, \bar{l}_c)$ **it is possible to derive the test sets of all the other** $2n - 2$ **cellular faults** (c, l_c, l_f), where $l_f \neq l_c$ and $l_f \neq \bar{l}_c$

Improving the testability in ACFM

- Improve testability of Adjacent Cellular Faults in lattices synthesized with **Altun-Riedel (AR) synthesis method**.
- AR defines many equivalent lattices $r \times s$ for the same function f.

To **improve the lattice testability** reducing the number of adjacent cell with the same literal we propose to:

1. **choose the best controlling literal** for each cell,
2. permute lattice **columns and rows**.



1) 2)

$$f = x_1x_2 + x_1x_3 + x_2x_3$$

Choose the best controlling literal

- **Heuristic algorithm** for choosing the controlling literal in $S_{i,j}$ for the cell $c_{i,j}$.
- The algorithm **try to avoid to choice of controlling literals occurring in adjacent cells**.

$\{x_1, x_2\}$	$\{x_1\}$	$\{x_2\}$
$\{x_1\}$	$\{x_1, x_3\}$	$\{x_3\}$
$\{x_2\}$	$\{x_3\}$	$\{x_2, x_3\}$

a)

$\{x_2\}$	$\{x_1\}$	$\{x_2\}$
$\{x_1\}$	$\{x_3\}$	$\{x_3\}$
$\{x_2\}$	$\{x_3\}$	$\{x_2\}$

b)

$\{x_3\}$	$\{x_1\}$	$\{x_2\}$
$\{x_1\}$	$\{x_1\}$	$\{x_3\}$
$\{x_2\}$	$\{x_3\}$	$\{x_2\}$

c)

$\{x_3\}$	$\{x_1\}$	$\{x_2\}$
$\{x_1\}$	$\{x_3\}$	$\{x_3\}$
$\{x_2\}$	$\{x_3\}$	$\{x_2\}$

d)

ControllingLiterals (lattice L)
INPUT: a lattice L ($r \times s$) and, for each cell $c_{i,j}$, the set $S_{i,j}$ of its possible controlling literals
OUTPUT: a lattice L' where each cell $c'_{i,j}$ contains exactly one controlling literal $l'_{i,j}$

for $i = 1$ **to** $r - 1$
  for $j = 1$ **to** $s - 1$
    $S = S_{i,j} \setminus (S_{i+1,j} \cup S_{i,j+1})$
    if ($S \neq \emptyset$) choose randomly $l'_{i,j} \in S$;
    else
      $S = S_{i,j} \setminus S_{i+1,j}$
      if ($S \neq \emptyset$) choose randomly $l'_{i,j} \in S$;
      else
        $S = S_{i,j} \setminus S_{i,j+1}$
        if ($S \neq \emptyset$) choose randomly $l'_{i,j} \in S$;
        else choose randomly $l'_{i,j} \in S_{i,j}$;
for $i = 1$ **to** $r - 1$ // last column
  $S = S_{i,s} \setminus S_{i+1,s}$
  if ($S \neq \emptyset$) choose randomly $l'_{i,s} \in S$;
  else choose randomly $l'_{i,s} \in S_{i,s}$;
for $j = 1$ **to** $s - 1$ // last row
  $S = S_{r,j} \setminus S_{r,j+1}$
  if ($S \neq \emptyset$) choose randomly $l'_{r,j} \in S$;
  else choose randomly $l'_{r,j} \in S_{r,j}$;
choose randomly $l'_{r,s} \in S_{r,s}$;

Permute lattice columns and rows

- Each product of the **SOP for f is assigned to a column**.
- Each product of the **SOP for f^D is assigned to a row**.
- Any **permutation** of the products in $SOP(f)$ and in $SOP(f^D)$ gives rise to a **correct lattice for f**.

- It is possible to **permute columns and rows** in order to **minimize the number of adjacent cells containing the same literal**.
- If two adjacent cells contain exactly the same literal, the corresponding ACF cannot be tested.

$\{x_1\}$	$\{x_1\}$	$\{x_2\}$
$\{\overline{x}_1\}$	$\{x_1\}$	$\{x_3\}$
$\{x_2\}$	$\{x_3\}$	$\{x_2\}$

not testable

$\{x_1\}$	$\{x_2\}$	$\{x_1\}$
$\{\overline{x}_1\}$	$\{x_3\}$	$\{x_1\}$
$\{x_2\}$	$\{x_2\}$	$\{x_3\}$

testable

A new version of Altun-Riedel algorithm

We propose a **new version of Altun-Riedel algorithm** in order to avoid some possible non-testable ACFs.

Step 1: find an irredundant, or a minimal, SOP representation for f and f^D: $SOP(f) = p_1 + p_2 + \ldots p_s$ and $SOP(f^D) = q_1 + q_2 + \ldots q_r$;

Step 2: form a $r \times s$ switching lattice and assign each product p_j $(1 \leq j \leq s)$ of $SOP(f)$ to a column and each product q_i $(1 \leq i \leq r)$ of $SOP(f^D)$ to a row;

Step 3: for all $1 \leq i \leq r$ and all $1 \leq j \leq s$, assign to the switch on the lattice site (i, j) one literal that is shared by q_i and p_j **following the strategy described in the Algorithm**;

Step 4: **permute rows and columns in order to minimize the number of adjacent cells containing the same literal**.

Experiments

- Benchmarks are taken from LGSynth93
- Each benchmark output is considered as a separate boolean function
- A total of 520 functions, we consider lattices with a number of variables lower than 6
- We compare the testability of ACFs for lattices obtained with Altun-Riedel (2012) and Gange-Søndergaard-Stuckey (2014) synthesis methods
- We evaluate the effect of the proposed lattice restructuring methods on the testability of lattices obtained with Altun-Riedel synthesis methods.
- We use a collection of Python scripts and a SAT solver to perform the Gange-Søndergaard-Stuckey synthesis.
- To compute the best permutation of rows and columns we use the linear optimizer GLPK (GNU Linear Programming Kit)

- The algorithm has been implemented in C
- The experiments have been run on a machine with 16 CPU @2.5 GHz, running Centos 7

Testability of lattices with different synthesis methods

- We **compare the number of testable cells** for each between lattices synthesized with **Altun-Riedel** and **Gange-Søndergaard-Stuckey** methods.
- The lattice synthesized with **Gange-Søndergaard-Stuckey** method contains a higher percentage of testable cells than Altun-Riedel in **more than 70%** of benchmarks.

		Altun-Riedel							Gange-Søndergaard-Stuckey						
name	n	r	s	area	$\%T^R$	$\%T^L$	$\%T^T$	$\%T^B$	r	s	area	$\%T^R$	$\%T^L$	$\%T^T$	$\%T^B$
add6(1)	4	6	6	36	53%	42%	67%	42%	5	3	15	**87%**	**93%**	**100%**	**100%**
addm4(6)	5	10	11	110	55%	47%	24%	24%	6	4	24	**100%**	**100%**	**96%**	**100%**
bench(7)	6	4	6	24	**100%**	**100%**	**100%**	**100%**	3	5	15	**100%**	**100%**	**100%**	**100%**
ex5(35)	6	7	3	21	**90%**	86%	57%	57%	6	3	18	89%	**89%**	**72%**	**67%**
exp(13)	6	2	5	10	90%	**100%**	**100%**	**100%**	2	4	8	**100%**	**100%**	**100%**	**100%**
fout(1)	6	9	10	4	**100%**	**100%**	**100%**	**100%**	6	4	24	87%	92%	**100%**	**100%**
fout(7)	6	8	10	80	64%	64%	27%	40%	6	4	24	**92%**	**92%**	**96%**	**100%**
fout(8)	6	9	10	90	61%	56%	27%	33%	6	4	24	**96%**	**92%**	**87%**	**96%**
risc(21)	5	2	5	10	80%	80%	90%	80%	2	4	8	**100%**	**100%**	**100%**	**100%**
Z5xp1(5)	5	10	10	100	30%	29%	23%	28%	4	5	20	**100%**	**100%**	**100%**	**95%**

Synthesis Method	Average area	$(T^R/\text{area})\%$	$(T^L/\text{area})\%$	$(T^T/\text{area})\%$	$(T^B/\text{area})\%$
GSS	12	95.6%	95.7%	95.8%	95.4%
AR	27	69.1%	67.9%	68.1%	69.2%

Improving the testability of with Altum-Riedel method

Comparison between literal chosen randomly and with the proposed Algorithm

name	n	col	row	area	Arbitrary $\%T^R$	$\%T^L$	$\%T^T$	$\%T^B$	Proposed Algorithm $\%T^R$	$\%T^L$	$\%T^T$	$\%T^B$
add6(2)	6	16	16	256	**20%**	20%	20%	21%	**20%**	**22%**	**24%**	**23%**
b12(0)	6	4	6	24	50%	37%	**58%**	**75%**	**62%**	**54%**	46%	68%
jbp(32)	5	2	4	8	**100%**	**100%**	**62%**	**62%**	**100%**	**100%**	**62%**	**62%**
m4(8)	6	1	6	6	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
m181(0)	6	4	6	24	50%	37%	**58%**	**75%**	**62%**	**54%**	46%	58%
mish(1)	6	5	6	30	60%	67%	67%	63%	**63%**	**70%**	**63%**	**70%**
shift(1)	5	4	4	16	62%	**44%**	**44%**	**62%**	**62%**	**44%**	**44%**	**62%**

Row and column permutations to minimize the number of adjacent cells that contain the same literal

name	Col	Row	Area	n	ordered $\%T^R$	$\%T^L$	$\%T^T$	$\%T^B$	randomly chosen $\%T^R$	$\%T^L$	$\%T^T$	$\%T^B$
add6(1)	6	6	36	4	**69%**	**72%**	**42%**	**47%**	53%	42%	33%	42%
alcom(2)	2	4	8	5	**62%**	**62%**	**100%**	**100%**	**62%**	**62%**	**100%**	**100%**
b12(0)	4	6	24	6	**68%**	**62%**	**58%**	**75%**	50%	37%	**58%**	**75%**
dc1(2)	4	4	16	4	**75%**	**75%**	**94%**	69%	62%	56%	56%	**81%**
inc(8)	2	3	6	4	**67%**	**67%**	**100%**	**100%**	**67%**	**67%**	**100%**	**100%**
mish(1)	5	6	30	6	**77%**	**80%**	**70%**	**67%**	60%	67%	67%	63%
radd(1)	6	6	36	4	**69%**	**72%**	**42%**	**47%**	53%	42%	33%	42%

	R-ACF			L-ACF			T-ACF			B-ACF		
	$(T^R/\text{area})\%$	% of improved lattices	% of increase of T^R	$(T^L/\text{area})\%$	% of improved lattices	% of increase of T^L	$(T^T/\text{area})\%$	%of improved lattices	% of increase of T^T	$(T^B/\text{area})\%$	% of improved lattices	% of increase of T^B
	83.9%	–	–	83.5%	–	–	85.5%	–	–	85.5%	–	–
with Algorithm	84.6%	12%	16%	84.2%	12%	15%	85.8%	9%	6%	85.9%	8%	3%
with permutations	88%	22%	52%	88%	23%	54%	89%	18%	40%	90%	21%	40%

Conclusions

- we have **extended the notion of cellular faults** to switching lattices
- we have proved that the testability of a general cellular fault is related to the **testability of the inverted literal fault**
- We have exploited this result for **simplifying the testability analysis of CFs**
- We proposed some techniques for **improving the testability** of a lattice for adjacent cellular fault **without increasing lattice dimension**.

Future works

- we will study different fault models for lattices
- we will improve the testability of lattices synthesized with the method of Gange-Søndergaard-Stuckey

Thank you!