Testability of Switching Lattices in the Stuck at Fault Model

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

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

[2]Università degli Studi di Milano, Italy, {valentina.ciriani, luca.frontini}@unimi.it

NANOxCOMP

Abstract

Switching lattices are two-dimensional arrays of four-terminal switches. We analyze lattices testability under the stuck-at-fault model (SAFM). Than we identify and discus properties of fully-testable lattices.

Stuck-At-Fault Model (SAFM)

SAFM assumes that a defect causes input or output fixed to 0 or 1.

Definition: A *stuck-at fault* with fault location v is a tuple $(v[i], \epsilon)$ or $([i]v, \epsilon)$. $v[i]$ $([i]v)$ denotes the i-th input (output) pin of v, $\epsilon \in \{0, 1\}$ is the fixed constant value.

We consider stuck-at-0 (SA0) and stuck-at-1 (SA1) faults.

Definition: An input t to a combinational logic circuit C is a *test* for a fault f, iff the primary output values of C on applying t in presence of f are different from the output values of C in the fault free case.

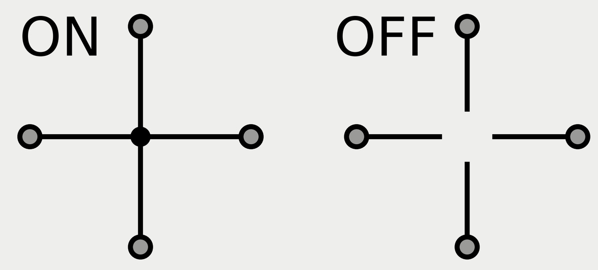
- We have to determine the not-testable faults.
- A node v in C is called *fully testable*, if there does not exist a redundant fault with fault location v.

- **If all nodes in C are fully testable, then C is called *fully testable*.**

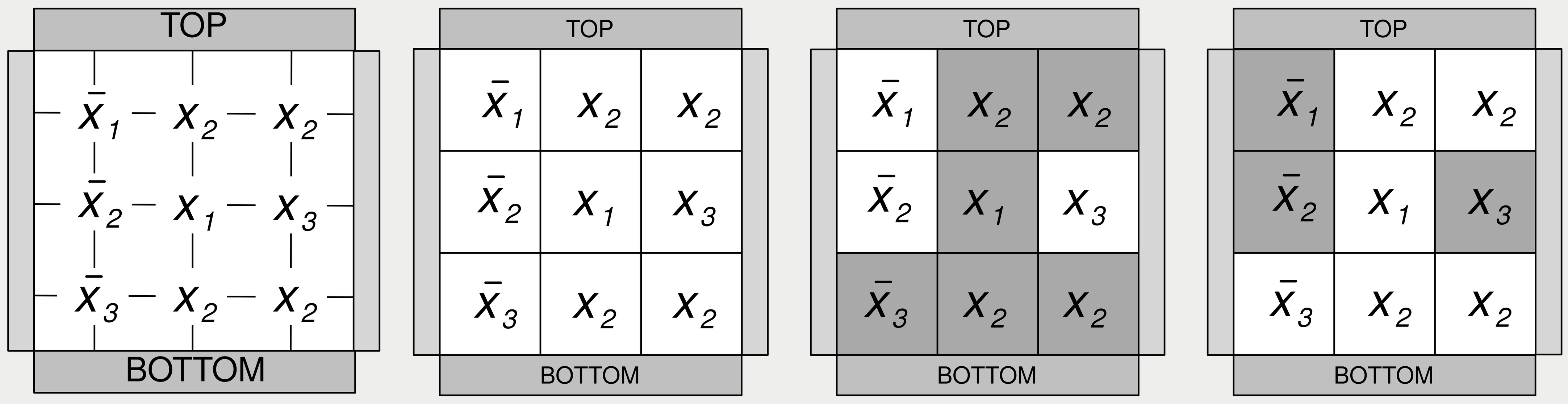
Switching Lattices

A switching lattice is a two-dimensional array of 4-terminal switches.
If there is a connection between top and bottom the lattice outputs 1, 0 otherwise.

- ON switch: all terminals connected
- OFF switch: all terminals disconnected
- Each switch is controlled by a boolean literal, 1 or 0.



The synthesis problem on a lattice consists in finding an assignment of literals to switches.
We use two synthesis methods: Altun-Riedel (AR) and Gange-Søndergaard-Stuckey (GSS)



(a) (b) (c) (d)

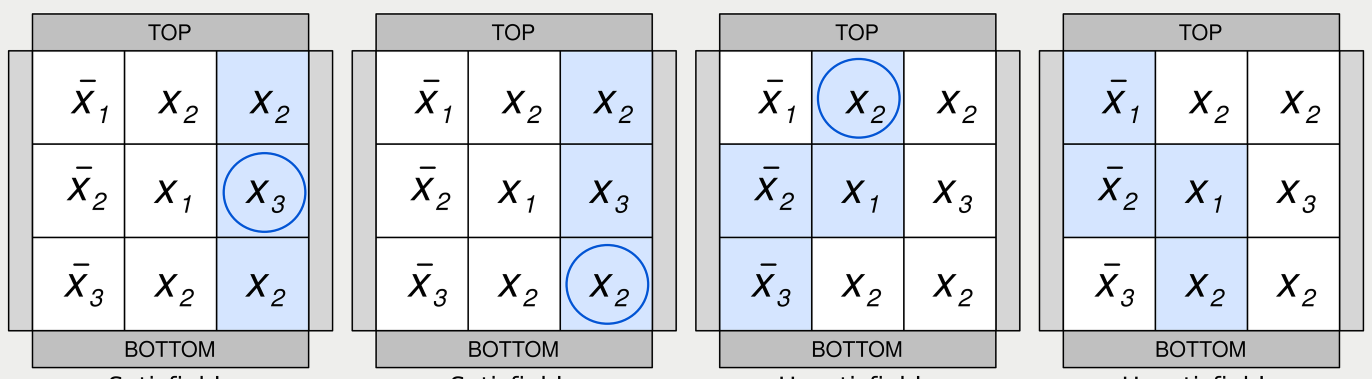
(a) A 4-terminal switching network of $f = \bar{x}_1\bar{x}_2\bar{x}_3 + x_1x_2 + x_2x_3$; (b) lattice form; (c-d) lattice with input $\{1,1,0\}$ and $\{0, 0, 1\}$. Grey and white squares represents ON and OFF switches.

Lattices: definitions and properties

A *path* is a list $l_1, l_2, \ldots, l_{m-1}, l_m$ of literals such that l_i and l_{i+1} (for $1 \le i < m$) are in adjacent cells.

Definitions:

- A path in a lattice is *unsatisfiable* (resp., *satisfiable*) if contains (resp., does not contain) both a variable x and its complement $\bar{x}$.
- The *product associated to a satisfiable path* is the conjunction of all literals of the path, without repetitions. The *product associated to an unsatisfiable path* is 0.
- An *accepting path* for a minterm v in a lattice is a satisfiable path whose associated product covers v.
- A path $l_1, \ldots, l_i, \ldots, l_m$ in a lattice L is *prime* w.r.t. a literal l_i $(1 \le i \le m)$, if the product associated to the sequence of literals obtained removing l_i from the path is not an implicant of the function implemented by L.



Satisfiable prime w.r.t. x_3 | Satisfiable not prime w.r.t. x_2 | Unsatisfiable prime w.r.t. x_2 | Unsatisfiable never prime

Proposition: The on-set of the function $f_{L^{c\leftarrow 1}}$ $(f_{L^{c\leftarrow 0}})$implemented by $L^{c\leftarrow 1}$ $(L^{c\leftarrow 0})$ is a superset (subset) of the on-set of f_L, i.e., $f_L^{on} \subseteq f_{L^{c\leftarrow 1}}^{on}$ $(f_{L^{c\leftarrow 0}}^{on} \subseteq f_L^{on})$.

Testability in the Stuck at Fault Model (SAFM)

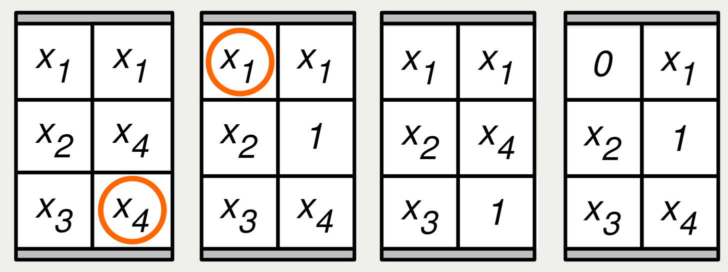
Definitions:

- A literal in a lattice's switch is *0-irredundant* (resp., *1-irredundant*) if it cannot be substituted by the constant 0 (resp., 1) without changing the function computed by the lattice.
- A lattice is *0-irredundant* (resp., *1-irredundant*) if any literal contained in it is 0-irredundant (resp., 1-irredundant).
- A lattice is *irredundant* if it is 0-irredundant and 1-irredundant.

Proposition: An irredundant lattice is fully testable with respect to the SAFM.

Theorems:

- A switching lattice L with a minimum number of literals is fully testable in the SAFM.
- A SA1 in a lattice cell c with literal l is testable if and only if there exists a path p that contains the cell c and is prime with respect to l.
- A SA0 in a lattice cell c is testable if and only if the cell c is essential.



(a) (b) (c) (d)

Four different mininum size lattices implementing $f = x_1x_2x_3 + x_1x_4$.
(a) 0-irredundant, but not 1-irredundant lattice
(b) 1-irredundant, but not 0-irredundant lattice
(c) a fully testable lattice
(d) a fully testable lattice with a minimum literal number

Algorithm for irredundancy test

Algorithm for the testing of the 0-irredundancy of a cell c:
0-irredundant (cell c)
INPUT: A cell c (containing the literal l) in a lattice L
OUTPUT: **true** if c is 0-irredundant in L, **false** otherwise
forall sub-path p_T from a top cell of L to c $(c \notin p_T)$
  if (p_T contains $\bar{l}$) discard p_T;
  if (p_T contains x and $\bar{x}$) discard p_T;
  else
    forall sub-path p_B from c to a bottom cell of L $(c \notin p_B)$
      if (p_B contains $\bar{l}$) discard p_B;
      if (p_Tp_B contains x and $\bar{x}$) discard p_B;
      else forall minterm m of the product associated to p_T, l, p_B
        if m is not in the on-set of $L^{c\leftarrow 0}$ **return true** ;
return false;

Algorithm for the testing of the 1-irredundancy of a cell c.
1-irredundant (cell c)
INPUT: A cell c (containing the literal l) in a lattice L
OUTPUT: **true** if c is 1-irredundant in L, **false** otherwise
forall sub-path p_T from a top cell of L to c $(c \notin p_T)$
  if (p_T contains l) discard p_T;
  if (p_T contains x and $\bar{x}$) discard p_T;
  else
    forall sub-path p_B from c to a bottom cell of L $(c \notin p_B)$
      if (p_B contains l) discard p_B;
      if (p_Tp_B contains x and $\bar{x}$) discard p_B;
      else forall minterm m of the product associated to $p_T, \bar{l}, p_B$
        if m is not in the on-set of L **return true** ;
return false;

- The final test can be implemented using OBDD.
- The OBDDs represent f and all the products associated to paths c.
- The time complexity is polynomial in OBDDs and G_L graph size.

Experimental Results

- The experiments are done substituting, a single cell literal with a SA1 or a SA0.
- The substitution is repeated for each lattice cell
- The benchmarks functions are taken from a subset of LGSynth93 (580 functions).

Synthesis Method	Average area	$(R_0$/area)%	$(R_1$/area)%
AR12	30	20%	29%
GSS14	15	4.5%	4.5%

	Altun-Riedel				Gange-Søndergaard-Stuckey			
name	col × row	area	$(R_0$ / area)%	$(R_1$ / area)%	col × row	area	$(R_0$ / area)%	$(R_1$ / area)%
addm4 (6)	10×11	110	49%	79%	6×4	24	0%	0%
b11 (3)	3×6	18	22%	56%	3×4	12	8%	8%
b7 (27)	2×5	10	0%	30%	3×3	9	22%	0%
bench (3)	4×6	24	8%	58%	4×3	12	8%	0%
dc2 (1)	7×12	84	40%	62%	6×4	24	4%	13%
ex5 (34)	10×4	40	8%	53%	6×4	24	0%	8%
exps (32)	2×7	14	43%	29%	2×5	10	10%	0%
m3 (3)	5×4	20	10%	55%	5×3	15	7%	7%
m3 (4)	8×6	48	27%	42%	7×3	21	0%	0%
max128 (23)	11×12	132	33%	82%	–	–	–	–
newtag (0)	8×4	32	13%	69%	6×3	18	0%	0%
newxcpla1 (18)	10×7	70	44%	71%	3×7	9	0%	0%
p3 (10)	6×10	60	10%	67%	4×5	20	0%	15%
p82 (13)	5×7	35	29%	34%	3×5	15	0%	0%
rd53 (1)	10×10	100	18%	80%	–	–	–	–
risc (21)	2×5	10	20%	20%	2×4	8	13%	0%
root (1)	8×8	64	36%	73%	6×4	24	8%	8%
sex (4)	3×5	15	40%	27%	3×4	12	17%	17%
tms (0)	4×11	44	32%	41%	3×6	18	0%	0%

Conclusion

- We have analyzed the testability of switching lattices under the SAFM, and characterized the properties of fully testable lattices.
- We have proposed an algorithm to detect redundancies.

- Future work includes the design of a method to transform non testable lattices into testable ones, by replacing some literals with a constant value.

This work is part of a project that has received funding from the European Union's H2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 691178.

VLSI-SoC 2018, Verona, Italy