U.P.B. Sci. Bull., Series C, Vol. 68, No. 3, 2006

ANALYTICAL APPROACH OF UNDETECTABLE BRIDGING FAULTS IN COMBINATIONAL CIRCUITS

I. I. BUCUR *

Defectele fizice cuprind defectele scurtcircuit, defectele întrerupere, tranzistor blocat la 1 şi tranzistor blocat la 0. Comparativ cu defectele tradiţionale „blocat la" introduse la nivelul de abstractizare poartă, defectele fizice reprezintă mult mai fidel defectele reale care apar la nivelele de reprezentare poartă şi tranzitor. Modelarea analitică pentru astfel de defecte, utilizată în proiectare şi testabiliate, este un domeniu încă nou şi în dezvoltare. Defectele scurtcircuit nedetectabile aparţin clasei defectelor dificil de detectat şi pot invalida anumite seturi complete de teste proiectate pentru defectele blocaj clasice. Această lucrare defineşte o caracterizare analitică a defectelor scurtcircuit nedetectabile prin utilizarea unei abordări analitice matematice discrete.

Physical faults include bridging faults, break (open) faults, transistor stuck-on and transistor stuck-off. Compared to traditional gate-level stuck-at faults, physical faults more closely represent realistic faults appearing at the gate level and transistor level. Analytical modeling for such faults, used for design and testability, is still a new and emerging area. Undetectable bridging faults belong to hard to detect faults class and can invalidate several sets of tests designed for classical stuck-at faults. This work defines an analytical characterization for undetectable bridging faults using discrete analysis mathematical approach.

Keywords: bridge defects, bridging fault models, resistive bridging faults, fault simulation, undetectable bridging faults, multiple bridging faults, stuck-at faults, induced wired-AND, induced wired-OR, nMOS circuits, TTL, Boolean differential calculus, exclusive-disjunctive expansion, discrete Taylor expansion.

Introduction

A key goal in manufacturing test is to maximize the quality of parts delivered to customers — ideally, shipping zero defective parts while reducing the cost of testing those parts. The purpose of testing is to develop confidence that a design or its specific implementation is functioning correctly or to identify the location and nature of the fault(s) within it (diagnosis). Because the numbers, types and locations of possible faults are huge, it is practical impossible that

* Lecturer, Dept. of Computer Science, University POLITEHNICA of Bucharest, Romania

testing can guarantee a perfect design. Many different faults, as types and/or locations, are often exhibiting similar symptoms involving that one test pattern could detect all of them. It makes facts more tractable but actual testing complexity is still very large. It was observed that some faults are more probable than others so a test for the most frequent faults will produce higher confidence (but not sureness) in the perfection of the design. The advent of deep-submicron (DSM) designs has created new difficulty in clock skew and power delivery, while the latest nanometer technologies have demonstrated that defects are located predominantly in routing [1]. Inductive fault analysis of actual circuits suggests that bridging faults account for thirty to fifty percent of all faults [2]. Other studies also show that most silicon defects exhibit bridging fault behavior and test strategies that use simple fault model, such as single stuck-at, do not satisfy the growing test quality requirements [3, and 4]. Test strategies that are using simple fault model, such as the single stuck-at model, are not satisfying growing test quality requirements. Other test approaches, such as delay and I_{DDQ} testing, are increasingly used for manufacturing testing of integrated circuits. A number of papers are showing that each such test policy detects some sole defects [5, and 6]. Although, I_{DDQ} testing has been successfully used for detection of shorts for a long time, the efficiency of this method in complex deep sub-micron circuits has recently become questionable. Consequently, a significant part of the testing must rely on other approaches such as voltage and delay testing.

In [7] is investigated a characterization of hard-to-detect bridging faults. For circuits with large numbers of lines (or nodes), this characterization can be used in selecting target faults for test generation when it is impractical to target all the bridging faults (or all the realistic bridging faults). Pomeranz *et al.,* [7], demonstrate that the faults selected based on the proposed characterization are indeed hard-to-detect by showing that the fault coverage of a given test set with respect to this subset is lower and more sensitive to the test set than the fault coverage obtained with respect to a random subset of the same size, with respect to the complete set of faults, and when possible, with respect to a subset of realistic bridging faults of the same size. Authors of [7] also demonstrated that a test set for the selected subset of faults detects other faults more effectively than when a test set is derived for a randomly selected subset of faults of the same size.

An efficient algorithm is presented in [8] for identification of bridges between two lines in combinational CMOS logic, such narrowing down the two-line bridge candidates based on tester responses for voltage tests. Due to the implicit enumeration of bridge sites, no layout extraction or precomputed stuck-at fault dictionaries are required. The bridge identification is easily refined using additional test pattern results when necessary. Vogels *et al.*, [8], did present results for benchmark circuits and four common fault models (dominant,

composite, wired-AND, and wired-OR), evaluating the diagnosis against other possible fault types, and summarizing the quality of their results.

Lechner *et al.*, [9], discuss the significance of fault simulation techniques to investigate pragmatic circuit malfunction modes and test requirements. It is revealed for an ADC goal design that hard faults frequently cause minor rather than disastrous failure; hence have to be subject to test. In [10], Engelke *et al.* introduced the concept of a multiple-valued logic simulator that is able to more accurately determine the possible behavior of a circuit in the presence of a bridging fault. However, this approach needs user defined mapping of a range of voltages to a logic value and the simulator takes care of certain voltages more closely than common bridge fault simulators that map all voltages to either logic 1 or 0.

Bridging faults are hard to be studied at circuit's logic level because most of the information about circuit's layout and interconnects are missing [3, 8, and 9]. This information is available only after the technologically mapping and when all modules are placed and routed. For the past three decades, researchers have used the s*tuck-at* fault model to represent faults in digital circuits. The stuck-at fault model has been used satisfactory to model faults in *Transistor-Transistor-Logic* (TTL) and n-channel *Metal-Oxide-Semiconductor* (nMOS) circuits.

Most of the testing technology is still using traditionally developed tests using *stuck-at* model, because complete tests for this model are detecting most of the physical faults. However, bridging faults and other typical physical faults occurring on DSM circuits are hard to be modeled using *stuck at* concept.

Among the bridging faults, the undetectable ones are still more difficult to test, for obvious reasons, but their invalidating effect on complete sets of tests designed for stuck-at faults is making them separate target [11, and 12]. Undetectable faults are, generally, due to re-convergent fan-out and redundant logic. Hazard-free multiplexers, as an example, have undetectable faults due to re-convergent fanout and redundancy. Bridging input lines of an AND gate, in TTL technology, is equivalent of a wired-AND gate. Such a bridging fault is undetectable. If all possible bridging faults between any only two nodes are considered, for P nodes the number of bridging faults will be $P^2 - P$ (either model) [13, and 14]. In practice, only the adjacent or overlapping wires in the circuit layout need to be considered.

This paper will focus on the analysis of undetectable bridging faults, introducing accurate relationship between functions carried on p lines connected through a wired-OR or wired-AND bridging fault.

1. Bridging fault model

There are three advanced, deterministic fault models for DSM defect testing: the bridging fault model, the path-delay fault model the transition fault model, and the transition fault model [1]. The bridging fault model, assumes that two or more wires are connected, but not intended to, by a resistive short line between them, which could be caused by a piece of metal from the sputtering process (Fig. 1). For a shorted line *w*, one have to distinguish between the value one could actually observe on *w* and the value of *w* as determined by its source element; the latter is called *driven value*. Bridging faults models [15] are capturing device faults caused by short between circuit connections. In [15] there are presented three bridging fault models valid for different CMOS technologies. These models are partitioned into a general framework, shared by all three models, and a technology-specific part. The first model is based on Shockley equations and is valid for conventional but not deep submicron CMOS. The second model is obtained by fitting SPICE data. The third resistive bridging fault model uses Berkeley Predictive Technology Model and BSIM4. This third resistive bridging fault model is valid for CMOS technologies with feature sizes of 90nm and below, accurately describing non-trivial electrical behavior in those technologies. Polian *et al.*, [15], did obtain experimental results for ISCAS circuits showing that the test patterns obtained for the Shockley model are still valid for the Fitted model, but lead to coverage loss under the Predictive model.

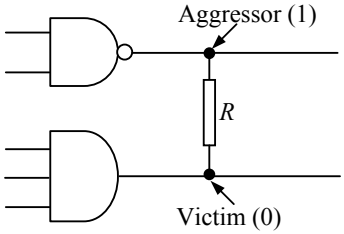

Fig. 1. Bridging fault becomes active when two or more nets connected by a resistive short line between them are driven to opposite values.

For most manufacturing processes, one could appreciate that defects between metal lines are the most plausible fault mechanism. Efficient and accurate bridging fault modeling are essential for reducing test costs and enhancing quality of generated test patterns.

There are two current models for wires shorted together:

- Wired-AND/Wired-OR fault model. Several classic studies suppose that bridging faults are pure-short defects, involving that bridge resistance is low or negligible.

- Dominant fault model. Bridging faults in CMOS technology require more complicated models. Whereas the stuck-at fault model assumes that a cell input or output is always tied to a fixed value, this bridging fault model assumes that one gate will dominate the value driven on the other net via the electrical path through the resistive short. If one net dominates, then the other may have an incorrect logic value at one or more of its fanouts.

Fig. 2 shows a general model of a bridging fault between two lines *w* and *t,* inducing a wired-function [13]. Such bridging fault is noted by (*w, t*) and the function introduced by the bridging fault is noted by *Z*(*w, t*). The fanout of *Z* is the union of the fanouts of the shorted signals. Note that the values of *w* and *t* in this model are their driven values but these are not observable in the circuit.

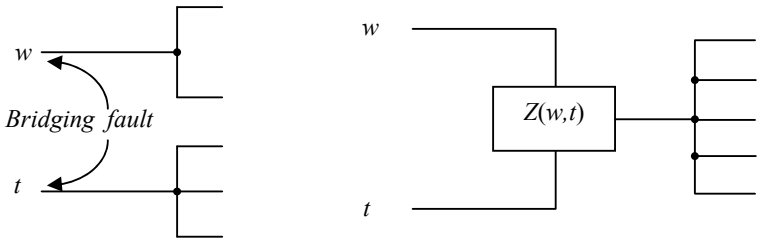

Fig. 2. Bridging fault model inducing a wired-function.

Induced wired-function *Z* has the property that *Z*(*a, a*) = *a*. What happens when *x* and *y* have opposite values value of *Z* depends only on the technology. For example, in *CMOS* the value of *Z*(*a, a'*) is, in general, non-determinate but in many technologies (such as TTL or ECL), when two shorted lines have opposite driven values, one value (the strong one) overrides the other. If $c \in \{0, 1\}$ is the strong value, then *Z*(0, 1) = *Z*(1, 0) = *c*, and the function introduced by the bridging fault is AND if *c* = 0 and OR if *c* = 1.

If there exists (at least) one path between *x* and *y*, then bridging fault (*x, y*) creates one or more feedback loops. Such a fault is referred to as a *feedback bridging fault*. A bridging fault that does not create feedback is referred to as a *nonfeedback bridging fault*. A feedback bridging fault transforms a combinational circuit into a sequential one. Moreover, if feedback loop involves an odd number of inversions, the circuit may oscillate. If the delay along the loop is small, the resulting oscillations have high frequency and may cause the affected signals to assume indeterminate logic values [13, and 14].

The multiple bridging fault models represents shorts involving more then two signal lines. A multiple bridging fault with $p > 2$ shorted lines can be modeled as being composed of p-1 bridging faults between two lines. Multiple bridging

faults among lines *r, s* and *t*, as an example, could be represented by the bridging faults (*r, s*) and (*s, t*). This model assumes that only one group of lines are shorted. Masking relations may occur among the components of a multiple bridging fault [15], but most multiple bridging faults are detected by the tests designed to detect their component bridging faults. The number of feasible bridging faults is potentially larger than the number of single stuck-at faults in the same circuit.

Stuck-at faults can be reduced, for testing but not for diagnosis, using functional (structural) equivalence and dominance relations. For bridging faults such similar collapsing techniques are not yet developed.

2. Analytical Model of Undetectable Bridging Faults

The concept of Boolean difference, as developed by J.P. Deschamps, M. Davio and G. Bioul [16], was introduced by Reed [17], and has been thoroughly investigated in papers due to A. Thayse and M. Davio [18], A. Thayse [19, and 20], etc. In these papers, various differential operators are introduced and described in connection with their application to switching functions. The Boolean differential calculus used here, in this paper, encompasses and generalizes the algebraic concepts introduced by these authors; it is considered here as a general method of analysis of an arbitrary Boolean function, and, as such, it could be applied to any classical switching problem. However, this representation tool seems particularly well suited for specific problems such as, for example, the diagnosis of switching circuits.

Following two propositions [21] are describing dependencies of *p* internal functions of scalar function *f* and involved in bridging fault inducing wired-AND, and respectively wired-OR.

Proposition 1. Let $l_{i_0}, l_{i_1}, \ldots, l_{i_{p-1}}$ be p lines in a combinational circuit C implementing scalar function f, and let note with $y_{i_0}, y_{i_1}, \ldots, y_{i_{p-1}}$ locale functions of these p lines. Then, a bridging fault inducing a wired AND between lines $l_{i_0}, l_{i_1}, \ldots, l_{i_{p-1}}$ is undetectable in C, if and only if:

$$\left(\bigoplus_{1 \le \mathbf{j} \le 2^p - 2} (f(\mathbf{x}, \mathbf{j}) \oplus f(\mathbf{x}, 0)) \cdot \mathbf{y}^{(\mathbf{j})} \right) = 0 . \tag{1}$$

Where it was noted with $f(\boldsymbol{x}, \boldsymbol{y})$ same function f but depending explicitly on both of primary variables $\boldsymbol{x}$ and internal variables $\boldsymbol{y}$, these internal variables being noted: $\boldsymbol{y} = (y_{i_0}, y_{i_1}, \ldots, y_{i_{p-1}})$.

Proof: It's obvious that wired AND between lines $l_{i_0}, l_{i_1}, \ldots, l_{i_{p-1}}$ introduces new function in network *C*. Let $z = y_{i_0} y_{i_1} \ldots y_{i_{p-1}}$ be this wired-AND. It could be remarked that faulty function f^* has now this form:

$f^*(\boldsymbol{x}, y_{i_0} = z, y_{i_1} = z, \ldots, y_{i_{p-1}} = z) = f^*(\boldsymbol{x}, z)$. One can start the proof considering exclusive-disjunctive expansion of f [18]:

$$f(\boldsymbol{x}, \boldsymbol{y}) = \bigoplus_{0 \le \boldsymbol{j} \le 2^p - 1} (f(\boldsymbol{x}, \boldsymbol{j}) \cdot \boldsymbol{y}^{(\boldsymbol{j})}) \tag{2}$$

Outlining first term in above sum:

$$f(\boldsymbol{x}, \boldsymbol{y}) = \Big(\bigoplus_{0 \le \boldsymbol{j} \le 2^p - 1} (f(\boldsymbol{x}, \boldsymbol{j}) \cdot \boldsymbol{y}^{(\boldsymbol{j})}) \Big) \oplus f(\boldsymbol{x},0) \oplus f(\boldsymbol{x},0) \tag{3}$$

And using identity: $\bigoplus_{0 \le \boldsymbol{j} \le 2^p - 1} \boldsymbol{y}^{(\boldsymbol{j})} = 1$, previous expression becomes:

$$f(\boldsymbol{x}, \boldsymbol{y}) = \Big(\bigoplus_{0 \le \boldsymbol{j} \le 2^p - 1} (f(\boldsymbol{x}, \boldsymbol{j}) \cdot \boldsymbol{y}^{(\boldsymbol{j})}) \Big) \oplus f(\boldsymbol{x},0) \cdot \Big(\bigoplus_{0 \le \boldsymbol{j} \le 2^p - 1} \boldsymbol{y}^{(\boldsymbol{j})} \Big) \oplus f(\boldsymbol{x},0) \tag{4}$$

Factoring term $\mathbf{y}^{(\mathbf{j})}$ in (4), one can obtain:

$$f(\boldsymbol{x}, \boldsymbol{y}) = \Big(\bigoplus_{0 \le \boldsymbol{j} \le 2^p - 1} (f(\boldsymbol{x}, \boldsymbol{j}) \oplus f(\boldsymbol{x},0)) \cdot \boldsymbol{y}^{(\boldsymbol{j})} \Big) \oplus f(\boldsymbol{x},0) \tag{5}$$

Outlining last term in ring sum of (5):

$$f(\mathbf{x}, \mathbf{y}) = \Big(\bigoplus_{1 \le \mathbf{j} \le 2^p - 2} (f(\mathbf{x}, \mathbf{j}) \oplus f(\mathbf{x},0)) \cdot \mathbf{y}^{(\mathbf{j})} \Big) \oplus f(\mathbf{x},0) \oplus \oplus (f(\mathbf{x}, 2^p - 1) \oplus f(\mathbf{x},0)) \mathbf{y}^{2^p - 1} \tag{6}$$

Making obvious substitutions and re-arranging terms in (6), it becomes

$$f(\boldsymbol{x}, \boldsymbol{y}) = \Big(\bigoplus_{1 \le \boldsymbol{j} \le 2^p - 2} (f(\boldsymbol{x}, \boldsymbol{j}) \oplus f(\boldsymbol{x},0)) \cdot \boldsymbol{y}^{(\boldsymbol{j})} \Big) \oplus f(\boldsymbol{x},0,0,\ldots 0) \oplus \oplus (f(\boldsymbol{x},1,1,\ldots,1) \oplus f(\boldsymbol{x},0,0,\ldots,0)) \cdot y_{i_0} \cdot y_{i_1} \ldots \cdot y_{i_{p-1}} \tag{7}$$

Using previously introduced notation $z = y_{i_0} \cdot y_{i_1} \cdot \ldots \cdot y_{i_{p-1}}$, relation (7) could be re-written:

$$f(\boldsymbol{x}, \boldsymbol{y}) = \Big(\bigoplus_{1 \le \boldsymbol{j} \le 2^p - 2} (f(\boldsymbol{x}, \boldsymbol{j}) \oplus f(\boldsymbol{x},0)) \cdot \boldsymbol{y}^{(\boldsymbol{j})} \Big) \oplus f(\boldsymbol{x},0,0\ldots,0) \oplus \oplus (f(\boldsymbol{x},1,1,\ldots,1) \oplus f(\boldsymbol{x},0,0,\ldots,0)) \cdot z \tag{8}$$

If bridging fault between lines $l_{i_0}, l_{i_1}, \ldots, l_{i_{p-1}}$, equivalent of a wired-AND, is undetectable then, this identity, linking faulty function f^* and f, hold:

$$f(\boldsymbol{x}, \boldsymbol{y}) = f^*(\boldsymbol{x}, \boldsymbol{y} = (z \cdot z \cdot \ldots \cdot z)) = f^*(\boldsymbol{x}, z \cdot z \cdot \ldots \cdot z) = f^*(\boldsymbol{x}, z) \tag{9}$$

Making discrete Taylor expansion [19] for $f^*(\mathbf{x}, z)$, in $z = 0$:

$$f^*(\boldsymbol{x},z)=f^*(\boldsymbol{x},0)\oplus\frac{df^*(\boldsymbol{x},z)}{dz}\cdot z \tag{10}$$

And, yet using discrete derivative definition [16]:

$$f^*(\boldsymbol{x},z)=f^*(\boldsymbol{x},0,)\oplus(f(\boldsymbol{x},1,)\oplus f(\boldsymbol{x},0))\cdot z \tag{11}$$

Considering (8) and (10), and tacking in account that bridging fault is undetectable, it results:

$$\Big(\bigoplus_{1\le \boldsymbol{j}\le 2^p-2}(f(\boldsymbol{x},\boldsymbol{j})\oplus f(\boldsymbol{x},0))\cdot \boldsymbol{y}^{(\boldsymbol{j})}\Big)=0.$$

And it finishes proof of the *if* part of ***Proposition 1***.
If (1) is true, then it results that:

$$f(x,\ y)\ =\ f(x,0,\cdots,0)\ \oplus\ (f(x,1,\cdots,1)\ \oplus\ f(x,0,\cdots,0))y_{i_0}y_{i_1}\cdots y_{i_{p-1}} \tag{12}$$

And using notation $z=y_{i_0}\cdot y_{i_1}\cdot\ldots\cdot y_{i_{p-1}}$, one can deduce that $f(\boldsymbol{x},\boldsymbol{y})=f^*(\boldsymbol{x},z)$, *i.e.* bridging fault inducing a wired-AND, between lines $l_{i_0}, l_{i_1}, \ldots, l_{i_{p-1}}$, is undetectable. It finishes up second part (*only if*) of the proof.
Considering $p=2$, then (1) becomes:

$$\begin{aligned}&(f(\boldsymbol{x},y_{i_1}=1,y_{i_0}=0)\oplus f(\boldsymbol{x},y_{i_1}=0,y_{i_0}=0))\cdot y_{i_1}\cdot y_{i_0}{}'\oplus\\&\oplus(f(\boldsymbol{x},y_{i_1}=0,y_{i_0}=1)\oplus f(\boldsymbol{x},y_{i_1}=0,y_{i_0}=0))\cdot y_{i_1}'\cdot y_{i_0}=0.\end{aligned} \tag{13}$$

Using discrete derivative definition [16] one could write (13) as follows:

$$\left.\frac{df(\boldsymbol{x},\boldsymbol{y})}{dy_{i_1}}\right|_{y_{i_0}=0}\cdot y_{i_1}\cdot y_{i_0}'\oplus\left.\frac{df(\boldsymbol{x},\boldsymbol{y})}{dy_{i_0}}\right|_{y_{i_1}=0}\cdot y_{i_1}'\cdot y_{i_0}=0. \tag{14}$$

It's obvious that $\dfrac{df(\boldsymbol{x},\boldsymbol{y})}{dy_{i_0}}$ depends only on $\boldsymbol{x}$ and y_{i_1}, and $\dfrac{df(\boldsymbol{x},\boldsymbol{y})}{dy_{i_1}}$ depends only on $\boldsymbol{x}$ and y_{i_0}. Using an exclusive-disjunctive expansion one can obtain for both derivatives:

$$\frac{df(\boldsymbol{x},\ \boldsymbol{y})}{dy_{i_0}}\ =\ \left.\frac{df(\boldsymbol{x},\ \boldsymbol{y})}{dy_{i_0}}\right|_{y_{i_1}=0}\cdot\ y_{i_1}'\ \oplus\ \left.\frac{df(\boldsymbol{x},\ \boldsymbol{y})}{dy_{i_0}}\right|_{y_{i_1}=1}\cdot\ y_{i_1} \tag{15}$$

$$\frac{df(\boldsymbol{x},\ \boldsymbol{y})}{dy_{i_1}} = \frac{df(\boldsymbol{x},\ \boldsymbol{y})}{dy_{i_1}}\bigg|_{y_{i_0}=0} \cdot\ y'_{i_0} \oplus \frac{df(\boldsymbol{x},\ \boldsymbol{y})}{dy_{i_1}}\bigg|_{y_{i_0}=1} \cdot\ y_{i_0} \tag{16}$$

Multiplying (15) and (16) with $y'_{i_1}y_{i_0}$ and $y_{i_1}y'_{i_0}$, respectively one get:

$$\frac{df(\boldsymbol{x},\ \boldsymbol{y})}{dy_{i_0}} \cdot\ y'_{i_1}y_{i_0} = \frac{df(\boldsymbol{x},\ \boldsymbol{y})}{dy_{i_0}}\bigg|_{y_{i_1}=0} \cdot\ y'_{i_1}y_{i_0} \tag{17}$$

$$\frac{df(\boldsymbol{x},\ \boldsymbol{y})}{dy_{i_1}} \cdot\ y_{i_1}y'_{i_0} = \frac{df(\boldsymbol{x},\ \boldsymbol{y})}{dy_{i_1}}\bigg|_{y_{i_0}=0} \cdot\ y_{i_1}y'_{i_0} \tag{18}$$

This way relation (14) is rewritten:

$$\frac{df(\boldsymbol{x},\boldsymbol{y})}{dy_{i_0}}y'_{i_1}y_{i_0} \oplus \frac{df(\boldsymbol{x},\boldsymbol{y})}{dy_{i_1}}y_{i_1}y'_{i_0} = 0. \tag{19}$$

or, using exclusive-or properties:

$$\frac{df(\boldsymbol{x},\boldsymbol{y})}{dy_{i_0}}y'_{i_1}y_{i_0} = \frac{df(\boldsymbol{x},\boldsymbol{y})}{dy_{i_1}}y_{i_1}y'_{i_0} \tag{20}$$

For $p = 2$ relation (20) represents *necessary and sufficient* conditions of bridging fault undetectably when bridging fault, inducing an equivalent wired-AND, occurs between only *two internal lines* in circuit *C*.

Similar conditions holds for bridging fault, inducing wired-OR, between *p* internal lines [21] in combinational circuit C:

Proposition 2. Let $l_{i_0}, l_{i_1}, \ldots, l_{i_{p-1}}$ be *p* lines in a combinational circuit C implementing scalar function *f* and let note with $y_{i_0}, y_{i_1}, \ldots, y_{i_{p-1}}$ locale functions of these *p* lines. Then, a bridging fault inducing a wired OR between lines $l_{i_0}, l_{i_1}, \ldots, l_{i_{p-1}}$ is undetectable in C, iff:

$$\Big(\bigoplus_{1 \le \boldsymbol{j} \le 2^p - 2} (f(\boldsymbol{x}, \boldsymbol{j}) \oplus f(\boldsymbol{x}, \boldsymbol{1}))\boldsymbol{y}^{(\boldsymbol{j})} \Big) = 0 \tag{21}$$

Where it was noted with $f(\boldsymbol{x}, \boldsymbol{y})$ function *f* depending both of primary variables $\boldsymbol{x}$ and internal variables $\boldsymbol{y} = (y_{i_0}, y_{i_1}, \ldots, y_{i_{p-1}})$.

Considering $p = 2$, in (21), then it is expanded as follows:

$$(f(\boldsymbol{x}, y_{i_1} = 0, y_{i_0} = 1) \oplus f(\boldsymbol{x}, y_{i_1} = 1, y_{i_0} = 1)) \cdot y'_{i_1} \cdot y_{i_0} \oplus$$
$$(f(\boldsymbol{x}, y_{i_1} = 1, y_{i_0} = 0) \oplus f(\boldsymbol{x}, y_{i_1} = 1, y_{i_0} = 1)) \cdot y_{i_1} \cdot y'_{i_0} = 0 \quad (22)$$

Using Boolean derivative definition, expression (22) becomes:

$$\frac{df(\boldsymbol{x}, \boldsymbol{y})}{dy_{i_1}}\bigg|_{y_{i_0}=1} \cdot y'_{i_1} \cdot y_{i_0} \oplus \frac{df(\boldsymbol{x}, \boldsymbol{y})}{dy_{i_0}}\bigg|_{y_{i_1}=1} \cdot y_{i_1} \cdot y'_{i_0} = 0. \quad (23)$$

Remarking that:

- Expression $\frac{df(\boldsymbol{x}, \boldsymbol{y})}{dy_{i_0}}$ depends on $\boldsymbol{x}$ and y_{i_1}, and
- Expression $\frac{df(\boldsymbol{x}, \boldsymbol{y})}{dy_{i_1}}$ depends on $\boldsymbol{x}$ and y_{i_0}.

Making Taylor expansion for both expressions, it results:

$$\frac{df(\boldsymbol{x}, \boldsymbol{y})}{dy_{i_0}} = \frac{df(\boldsymbol{x}, \boldsymbol{y})}{dy_{i_0}}\bigg|_{y_{i_1}=0} \cdot y'_{i_1} \oplus \frac{df(\boldsymbol{x}, \boldsymbol{y})}{dy_{i_0}}\bigg|_{y_{i_1}=1} \cdot y_{i_1} \quad (24)$$

$$\frac{df(\boldsymbol{x}, \boldsymbol{y})}{dy_{i_1}} = \frac{df(\boldsymbol{x}, \boldsymbol{y})}{dy_{i_1}}\bigg|_{y_{i_0}=0} \cdot y'_{i_0} \oplus \frac{df(\boldsymbol{x}, \boldsymbol{y})}{dy_{i_1}}\bigg|_{y_{i_0}=1} \cdot y_{i_0} \quad (25)$$

Multiplying (24) and (25) with $y'_{i_1} \cdot y_{i_0}$ and, respectively, with $y_{i_1} \cdot y'_{i_0}$ it yields:

$$\frac{df(\boldsymbol{x}, \boldsymbol{y})}{dy_{i_0}} \cdot y_{i_1} \cdot y'_{i_0} = \frac{df(\boldsymbol{x}, \boldsymbol{y})}{dy_{i_0}}\bigg|_{y_{i_1}=1} \cdot y_{i_1} \cdot y_{i_0}' \quad (26)$$

$$\frac{df(\boldsymbol{x}, \boldsymbol{y})}{dy_{i_1}} \cdot y'_{i_1} \cdot y_{i_0} = \frac{df(\boldsymbol{x}, \boldsymbol{y})}{dy_{i_1}}\bigg|_{y_{i_0}=1} \cdot y'_{i_1} \cdot y_{i_0} \quad (27)$$

This way expression (22) becomes:

$$\frac{df(\boldsymbol{x}, \boldsymbol{y})}{dy_{i_0}} \cdot y_{i_1} \cdot y'_{i_0} \oplus \frac{df(\boldsymbol{x}, \boldsymbol{y})}{dy_{i_1}} \cdot y'_{i_1} \cdot y_{i_0} = 0. \quad (28)$$

Equivalent to:

$$\frac{df(\boldsymbol{x},\ \boldsymbol{y})}{dy_{i_0}} \cdot y_{i_1} \cdot y'_{i_0} = \frac{df(\boldsymbol{x},\ \boldsymbol{y})}{dy_{i_1}} \cdot y'_{i_1} \cdot y_{i_0}. \tag{29}$$

Conclusions

Expressions (20) and (29) derived from (1) and respectively (21) are same as formulas suggested in [11] by K.L. Kodandapani and D.K. Pradhan, but presented without proofs. It makes both ***Proposition 1*** and ***Proposition 2*** generalizations [21] of formulas in [11]. This analytical tool will be continued with application intended to efficiently compute expressions involved in it. Known methods are suitable for low complexity bridging faults (low value of parameter p). Real situations however are involving bridging faults between arbitrary numbers of lines. Performances of such application will make it usable for analysis in practical and manufacturing situations. Using and developing algorithms called pseudo canonical expansions having at most as many terms as the best canonical ones [21, and 22], make calculus of expressions (1) and (21) feasible for more then 20 variables. Undetectable bridging faults, once detected, may require special design modifications including circuits partitioning [23, and 24].

Other way of extending this tool will consider different type of circuits including CMOS. Modeling physical faults involves closer relationship with semiconductor processes and appropriate statistics, and behavior of such faults, but not limited to them.

R E F E R E N C E S

1. *R. Mattiuzzo, L. Tarantini, and C. Hay* "DSM fault models," http://www.reed-electronics.com/tmworld/article/CA489463?ref=nbcs.
2. *J.P. Shen, W. Maly, and F.J. Ferguson,* "Inductive fault analysis of MOS integrated circuits," in IEEE Design and Test Computers, **Vol. 2,** No. 6, Dec. 1985, pp. 13-26, 1985.
3. *D.B. Lavo, T. Larrabee, J.F. Ferguson, B. Chess, J. Saxena, and K. M. Butler*, "Bridging Fault Diagnosis in the Absence of Physical Information," in Proceedings of the International Test Conference, IEEE, 1997, pp. 887-893, 1997.
4. *J. Saxena, K.M. Butler, H. Balachandran, D.B. Lavo, B. Chess, T. Larrabee, and F.J. Ferguson,* "On Applying Non-Classical Defect Models to Automated Diagnosis," in Proceedings of International Test Conference 1998, pp. 748-757, 1998.
5. *E. Isern and J. Figueras,* "Analysis of I_{DDQ} detectable bridges in combinational CMOS circuits," in Proceedings of the Twelfth IEEE VLSI Test Symposium, pp. 368-373, April 1994.
6. *D.B. Lavo*, "Comprehensive Fault Diagnosis of Combinational Circuits," PHD dissertation, University of California Santa Cruz, September 2002.

7. *I. Pomeranz, S. Reddy, and S. Kundu,* **"**On the Characterization of Hard-to-Detect Bridging Faults," in Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE'03), pp. 11012-11019, 2003.
8. *T. J. Vogels, W. Maly, and R.D. (Shawn) Blanton,* "Progressive Bridge Identification," in ITC'03 International Test Conference, pp: 309-318, 2003.
9. *A. Lechner, A. Richardson and B. Hermes.* "Short Circuit Faults in State-Of-The-Art ADCs - Are They Hard or Soft?," in 10th Asian Test Symposium (ATS'01), Kyoto, Japan, pp. 417-422, 19-21 November 2001.
10. *P. Engelke, B. Becker, and M. Keim,* "A Parameterizable Fault Simulator for Bridging Faults," In The Fifth IEEE European Test Workshop (ETW'00), pp. 63 – 68, 2000.
11. *K.L. Kodandapani and D.K. Pradhan*, "Undetectability of Bridging Faults and validity of Stuck-at Fault Test Sets," in IEEE Transactions on Computers, **Vol. C-29,** No. 1, pp. 55-59, January 1980.
12. *I. Bucur*, "Contributions Concerning the Characterization of Undetectable Faults of Short Circuit Type in Combinational Digital Circuits," in Economic Computation and Economic Cybernetics and Research, **Vol. XXI,** No. 4, 1986.
13. *M. Abramovici, M.A. Breuer, A.D. Friedman*, Digital System Testing and Testable Design, Wiley-IEEE Press, September 1994.
14. *S.D. Millman, E.J. McCluskey, and J.M. Acken*, "Diagnosing CMOS bridging faults with stuck-at fault dictionaries," in *Proceedings of the International Test Conference,* IEEE,
15. *I. Polian, S. Kundu, J.-M. Galliere, and P. Engelke*. "Resistive Bridge Fault Model Evolution from Conventional to Ultra Deep Submicron Technologies,**"** in Proceedings of the 23rd IEEE VLSI Test Symposium, (VTS'05), pp 343-348, 2005.
16. *G. Bioul, M. Davio, and J.P. Deschamps*, "Minimization of Boolean Functions," in Philips Research Reports, **Vol. 28,** No. 816. pp. 17-36, 1973.
17. *I.S. Reed*, "Boolean Difference Calculus and Fault Finding," in SIAM Journal of Applied Mathematics, **Vol. 24**, No. 1, pp. 134-143, January 1973.
18. *A. Thayse, and M. Davio,* "Boolean Differential Calculus and Its Application to Switching Theory," in IEEE Transaction on Computers, **Vol. C-22,** no. 4, pp. 409-420, April 1973.
19. *A. Thayse,* "On Some Iteration properties of Boolean Functions," in Philips Research Reports, **Vol. 28,** No. 823, pp. 107-119, 1973.
20. *A. Thayse*, "Disjunctive and Conjunctive Operators for Boolean Functions," in Philips Research Reports, **Vol. 28,** No. 823, pp. 1-16, 1973.
21. *I. Bucur,* "Analiza şi sinteza subsistemelor numerice cu comportament optimal", PhD Thesis, University "Politehnica" of Bucharest, February 1999.
22. *M. Davio,* "Ring-Sum Expansions of Boolean Functions," In Proceedings of the Symposium on Computers and Automata Polytechnic Institute of Brooklyn, pp. 411- 418, April 13-15, 1971.
23. *I. Bucur*, "Undetectable Bridging Faults in Combinational Circuits," In Proceedings the 15th International Conference on Control Systems and Computer Science, CSCS-15, May 25-27, 2005, Romania, Bucharest. pp 762-767, ISBN: 973-8449-89-8, ISBN: 973-8449-91-X.
24. *I. Bucur,* "Partitioning Combinational Circuits," In Proceedings of the 15th International Conference on Control Systems and Computer Science, CSCS-15, May 25-27, 2005, Romania, Bucharest. pp 768-773, ISBN: 973-8449-89-8, ISBN: 973-8449-91-X.