SUPERB: Simulator Utilizing Parallel Evaluation of Resistive Bridges

PIET ENGELKE
Albert-Ludwigs-University, Freiburg
BERND BECKER
Albert-Ludwigs-University, Freiburg
MICHEL RENOVELL
LIRMM – UMII, Montpellier
JUERGEN SCHLOEFFEL
Mentor Graphics Development, Hamburg
BETTINA BRAITLING
Albert-Ludwigs-University, Freiburg
ILIA POLIAN
Albert-Ludwigs-University, Freiburg

A high-performance resistive bridging fault simulator SUPERB (Simulator Utilizing Parallel Evaluation of Resistive Bridges) is proposed. It is based on fault sectioning in combination with parallel-pattern or parallel-fault multiple-stuck-at simulation. It outperforms a conventional interval-based resistive bridging fault simulator by three orders of magnitude while delivering identical results. Further competing tools are outperformed by several orders of magnitude. Industrial-size circuits, including a multi-million-gates design, could be simulated with run times within an order of magnitude of the run times for pattern-parallel stuck-at fault simulation.

Categories and Subject Descriptors: B.1.3 [**Control Structures and Microprogramming**]: Control Structure Reliability, Testing, and Fault-Tolerance; B.2.3 [**Arithmetic and Logic Structures**]: Reliability, Testing, and Fault-Tolerance; B.7.2 [**Integrated Circuits**]: Design Aids—*Simulation*; B.7.3 [**Integrated Circuits**]: Reliability and Testing—*Testability*; J.6 [**Computer-Aided Engineering**]: Computer-Aided Design

General Terms: Algorithms, Reliability

Additional Key Words and Phrases: Resistive bridging faults, bridging fault simulation, PPSFP, SPPFP, fault mapping

P. Engelke, B. Becker, B. Braitling and I. Polian are with Albert-Ludwigs-University of Freiburg, Georges-Köhler-Allee 51, D-79110 Freiburg i. Br., Germany (e-mail: {engelke|becker|braitlin|polian}@informatik.uni-freiburg.de). Michel Renovell is with LIRMM – UMII, 161 Rue Ada, 34392 Montpellier, France (e-mail: renovell@lirmm.fr). Juergen Schloeffel is with Mentor Graphics Development GmbH, Tempowerkring 1B, D-21079 Hamburg, Germany (e-mail: juergen_schloeffel@mentor.com).

Parts of this article were presented at the Asian Test Symposium 2007 and at the Design Automation and Test in Europe Conference 2008.

1. INTRODUCTION

Defect-based test (DBT) [Maly 1987; Sengupta et al. 1999] is a methodology to improve the quality of micro- and nanoelectronic products by employing accurate models of actual defects showing up in the silicon. DBT is used to complement standard test methods which are based on the stuck-at fault model. It has long been known that many defects are not adequately represented by the stuck-at fault model [Maly 1987; Ferguson and Larrabee 1991; Hawkins et al. 1994; Aitken 1995; Zachariah and Chakravarty 2000]. Although a large share of defect population is detected by stuck-at tests, the product quality level is often inadequate when only the stuck-at fault model is used. DBT helps to increase the quality level by explicitly targeting and detecting defects not covered by the stuck-at fault model.

The conventional approach to testing micro- and nanoelectronic circuits includes test generation using the stuck-at fault model and application of the generated test patterns to the circuit by the automatic test equipment. The list of faults to be considered is derived directly from the gate-level net-list of the circuit. In contrast, pattern generation for DBT may require additional information, such as circuit layout or technology parameters [Ferguson and Shen 1988; Konuk et al. 1995; Khare and Maly 1996].

Fault simulation is a key element of a DBT flow. Although test generation methods for non-standard fault models do exist [Ferguson and Larrabee 1991; Chang et al. 1999; Krsti et al. 2001; Nourani et al. 2005; Mitra et al. 2006], they do not always scale for industrial-size circuits. Hence, it is important to determine the coverage of realistic defects by the existing test pattern sets (where the test pattern sets might have been created using conventional fault models). Then, defects missed by that sets could be addressed pinpointedly using expensive defect-based test generation approaches. Alternatively, techniques such as n-detection [Ma et al. 1995] or its extensions [Grimaila et al. 1999; Polian et al. 2004] could be employed. These techniques increase the accidental detections of non-targeted defects by targeting the same stuck-at fault $n > 1$ times. Defect-based fault simulation is useful in determining the value of n which leads to an adequate defect coverage, as overestimating n would lead to large test patterns and increased cost of test application.

Resistive short defects have been an important defect class in the past [Rodríguez-Montañés et al. 1992], and their relevance continues to grow. Within a DBT framework, short defects are represented by bridging faults. Simple bridging fault models ignore the resistance of the defect [Mei 1974; Banerjee and Abraham 1985; Millman and Garvey 1991; Maxwell and Aitken 1993; Greenstein and Patel 1992; Ferguson and Larrabee 1991; Rearick and Patel 1993]. Resistive bridging faults (RBF) are modeling this aspect with a higher degree of accuracy [Renovell et al. 1994; 1995; Hao and McCluskey 1991; Favalli et al. 1993; Vierhaus et al. 1993; Liao and Walker 1996; Lee and Walker 2000; Sar-Dessai and Walker 1998; 1999; Renovell et al. 1999; Engelke et al. 2006b; Li et al. 2003; Polian et al. 2005]. Short defect resistance R_{sh} is a random parameter not known in advance. Hence, an RBF simulator calculates for a given fault the range of resistances in which a given test pattern set detects the fault. This range is called *analogue detectability interval* or ADI [Renovell et al. 1994; 1995].

An ADI is often of the shape $[0, R_{\max}]$, where $R_{\max}$ is the maximal R_{sh} value for which the fault is detected. However, an ADI can also be a union of disjoint intervals [Renovell et al. 1999]. Once the ADI is known for each fault in the fault list, the fault coverage is obtained by relating C-ADI, the range of resistances detected by any test pattern from the test pattern set, to G-ADI, the range of resistances detectable by any possible test pattern [Renovell et al. 1999; Engelke et al. 2006b]. This approach is different from the conventional bridging fault simulation, where any fault is either detected by the test pattern set or not and the fault coverage is the fraction of the detected faults. Calculation of G-ADI is NP complete [Engelke et al. 2006b]; it can be obtained by exhaustive simulation or an ATPG procedure [Engelke et al. 2006a], or approximated [Lee and Walker 2000; Engelke et al. 2006b].

In this article, we propose the high-performance resistive bridging fault simulator SUPERB (Simulator Utilizing Parallel Evaluation of Resistive Bridges). It uses the sectioning-based approach [Shinogi et al. 2001] in combination with a parallel multiple-stuck-at fault engine. For a bridging fault between circuit nodes a and b, a section $[R_1, R_2]$ denotes a range of bridging defect resistances for which the logic behavior of the circuit is identical. In particular, if a bridging defect between nodes a and b with a resistance R_{sh} from section $[R_1, R_2]$ is detected by a test vector, then all bridging defects between a and b with resistances between R_1 and R_2 will be detected by the same vector.

Given a section and an input vector, the faulty behavior of the circuit can be characterized by a multiple-stuck-at fault. Since the resistive bridging fault model considers pattern-dependency [Renovell et al. 1999; Engelke et al. 2006b], the faulty behavior of the same circuit under a different input vector may be described by a different multiple-stuck-at fault. We store the mapping of resistance sections to multiple-stuck-at faults considering the pattern dependency in a hash table which is calculated before actual simulation begins. It is possible to generate the hash table using SPICE simulations [Lee and Walker 2000] or electrical equations [Renovell et al. 1994; 1995; Polian et al. 2005]. Whenever the simulation arrives at a fault site, the adequate multiple-stuck-at fault is looked up in the hash table and handed to a multiple-stuck-at fault simulation engine which utilizes the usual speed-ups. We reiterate that there is no one-to-one mapping between a resistive bridging fault, even restricted to a section, and a multiple-stuck-at fault, because of pattern dependency.

Considering surrogate faults, including multiple-stuck-at faults and multiple single-stuck-at faults, to represent complex resistive bridging defects has been proposed in the past. Maeda and Kinoshita utilized locally exhaustive testing of the bridge site [Maeda and Kinoshita 2000]. n-detection [Ma et al. 1995; Reddy et al. 1997] and its extensions [Grimaila et al. 1999; Polian et al. 2004] demanded application of multiple tests for one single-stuck-at fault. The Unified Fault Model [Chen et al. 2005] considered all possible multiple-stuck-at faults at the nodes succeeding the bridge site. In contrast to these approaches, we retain the modeling accuracy of the original resistive bridging fault model. The fault coverages returned by SUPERB are not over- or underapproximations, they correspond exactly to the numbers generated by an interval-based resistive bridging fault simulator.

We present results for large industrial circuits provided by NXP. While the simulation data reported earlier assumed 10,000 faults or less in the fault list, we employ fault lists containing $10 \times S$ faults where S is the number of gates in the circuit.

(This appears to be a realistic number of faults when inductive fault analysis [Ferguson and Shen 1988] with no fault list truncation is done; in our experiments this corresponded to some 25 million faults for the largest circuit.) We also present results for ISCAS85, ISCAS89 and ITC99 circuits using identical setup. For comparison, we report results for stuck-at fault simulation of the same circuits using the same simulation engine. It turns out that, while the complexity of the RBF simulation is higher than the complexity of the stuck-at fault simulation, the overhead is limited and does not differ significantly for different classes and sizes of circuits.

Two RBF simulators employing parallelism have been published so far. Lee and Walker proposed the interval-based simulator PROBE which processes pointers to intervals rather than intervals themselves [Lee and Walker 2000]. The pointers can be shared, reducing the memory requirements. Cheung and Gupta introduced (in October 2007, the month in which the first version of SUPERB was presented [Engelke et al. 2007]) a simulator which is based on a slightly different electrical model [Cheung and Gupta 2007]. We will discuss the relation of SUPERB to these simulators later in the article.

The remainder of the article is organized as follows. In Section 2 we briefly reiterate basic information about resistive bridging faults. In Section 3 we formalize the fault sectioning approach [Shinogi et al. 2001] within our framework and discuss its differences from the interval-based approach in context of simulation. In Section 4 we explain the simulator SUPERB which utilizes the sectioning technique to leverage the accelerations known for stuck-at fault simulation and relate it to previous work. In Section 5, experimental results are reported. They include, for the first time, the analysis of the probability that a single resistive bridging fault causes a double error. Section 6 concludes the article.

2. RESISTIVE BRIDGING FAULTS

In a fault-free circuit, the voltage level V_a at the output a of the logic gate A is either $V_a = V_{DD}$ if the logical value at a is 1 or $V_a = 0$V if the logical value at a is 0. If two lines (say, outputs a and b of gates A and B) are connected by a resistive short defect with a resistance R_{sh} and have opposing logic values (say, logic-1 at a and logic-0 at b), then V_a will assume some value below V_{DD} and V_b will assume some value above 0V. The exact values of V_a and V_b depend on the parameters of the transistors in gates A and B, the number of driving transistors (i.e., the logical values applied to the inputs of gates A and B) and R_{sh} [Renovell et al. 1994].

The intermediate voltages V_a and V_b will be interpreted by the gates succeeding lines a and b as either logic-1 or logic-0, depending on the logic threshold of the gates. The logic threshold depends on the parameters of the transistors within the succeeding gate; different inputs of a gate generally have different logic thresholds. The defect is detected if at least one succeeding gate interpreted the voltage on a line involved in the bridge as a faulty logical value and this value propagated to an output through a sensitized path. Multiple fault effects could be propagated through reconverging paths, resulting in fault effect cancellation for certain values of R_{sh} [Renovell et al. 1999]. Further pattern-dependent effects including the multiple strengths problem are described in [Engelke et al. 2006b].

3. FAULT SECTIONING

In this section, the fault sectioning technique from [Shinogi et al. 2001] is formalized within our framework and illustrated by an example.

A *bridging fault* is given by two bridged nodes a and b. A *bridging defect* is given by two nodes a and b and the defect resistance R_{sh}. A bridging fault corresponds to an infinite number of bridging defects having different resistances.

For a bridging fault or a bridging defect between a and b, the two gates which drive a and b are called *preceding gates* and the gates driven by either a or b (including all their fanout branches) are called *succeeding gates*. If an input vector implies opposite logic values on nodes a and b, then a bridging defect will result in intermediate voltages V_a and V_b on the respective lines, i.e., $0\text{V} < V_a, V_b < V_{DD}$. For every input of each succeeding gate, a single *logic threshold* is assumed such that every voltage below this threshold is interpreted as logic 0 and every voltage above is interpreted as logic 1. Although it is possible to consider two distinct logic thresholds per gate [Cheung and Gupta 2007], we did not implement this option to ensure identical fault simulation outcome to earlier interval-based simulators which assume a single threshold [Lee and Walker 2000; Shinogi et al. 2001; Engelke et al. 2006b]. For the same reason, we did not model process variations.

The resistance of a bridging defect which induces a voltage corresponding to a logic threshold of one of the inputs driven by the node is called *critical resistance.* An *analogue detection interval* (ADI) is the range of resistances for all bridging defects which are detected by a test vector, i.e., which produce a wrong logic value on at least one primary output (or other observable point). An ADI typically has the shape $[0, R]$ for some value of R but it can also be a union of disjoint sub-intervals [Renovell et al. 1999]. The boundaries of the ADIs are always critical resistances. The number of different critical resistances depends on the fault site. The ADI is calculated at the bridge site based on an electrical analysis and propagated to the observable points of the circuit.

The interval $[R_1, R_2]$ is called a *section* if R_1 and R_2 are critical resistances and there is no critical resistance R_3 such that $R_1 < R_3 < R_2$. The number of sections is given by the number of different critical resistances. Let $R_1, R_2, \ldots, R_m$ be all the critical resistances for the bridging fault between nodes a and b, sorted in ascending order. The detection status of a bridging fault can be represented by the detection statuses of the sections $[0, R_1]$, $[R_1, R_2]$, $\ldots$, $[R_{m-1}, R_m]$, $[R_m, \infty]$. The detection status is uniform for every bridge resistance within a section, i.e., either all bridging defects with a resistance from a section are detected by a test vector or none is detected. If the detection status of all the sections is known, the ADI can be constructed as the union of all detected sections. This approach results in the same ADI as the technique based on interval propagation.

For illustration, we consider the circuit from Figure 1 (a) [Engelke et al. 2006b] which is a part of a larger circuit shown in Figure 2. Solid curves in Figure 1 (b) show possible voltage characteristics as a function of bridge resistance R_{sh} when logic values 0011 are applied to gates A and B preceding the bridge and dashed curves show the characteristics when values 0111 are applied. Logic thresholds of the succeeding gates are Th_C, Th_D and Th_E. The fault is not excited and no detection is possible if nodes a and b have the same logic value. For the sake of

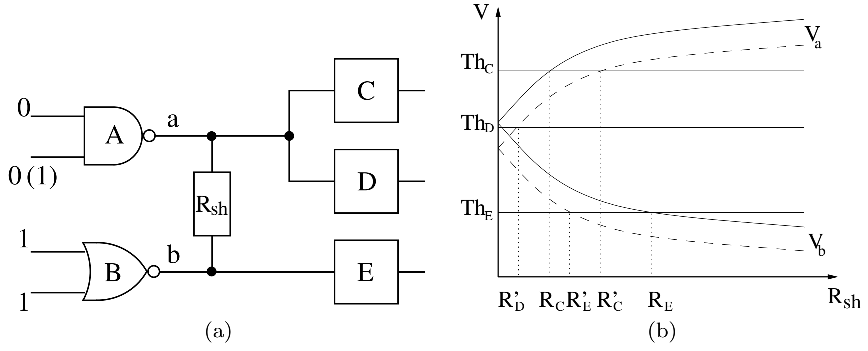


Fig. 1. Example circuit: bridge site (a); R_{sh}-V-diagram (b)

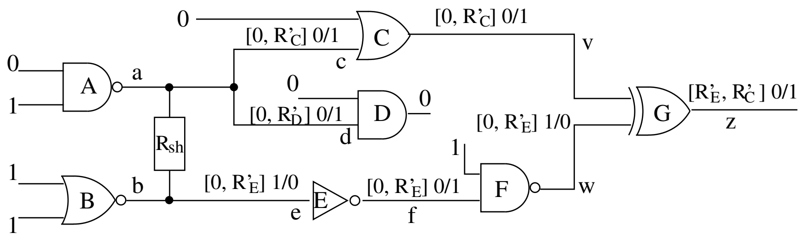


Fig. 2. Example circuit: interval-based propagation

simplicity, we assume that the only logic value assignments to the inputs of the preceding gates leading to a fault excitation are 0011 and 0111. (Assignment 1011 can be assumed to have effect identical to 0111.) Critical resistances are R_C, R_E (for value assignment 0011), R'_C, R'_D and R'_E (for value assignment 0111). Note that there is no critical resistance R_D. Sections are $[0, R'_D]$, $[R'_D, R_C]$, $[R_C, R'_E]$, $[R'_E, R'_C]$, $[R'_C, R_E]$ and $[R_E, \infty]$. Note that the fault-free value is always assumed in the last section such as $[R_E, \infty]$ in this example and this section does not need to be considered explicitly.

Figure 2 shows how the fault effect is propagated for input values 0111 using the interval-based technique. The interval assigned to a node is shown over the node. For instance, interval $[0, R'_C]0/1$ means that the node assumes logic value 0 if the bridge resistance is between 0Ω and R'_C, and logic value 1 otherwise. Similarly, interval $[0, R'_E]1/0$ means that the node assumes logic value 1 if the bridge resistance is between 0Ω and R'_E, and logic value 0 otherwise. The intervals are propagated through the circuit taking the types of the gates and the values on the side inputs into account. For instance, no interval is propagated through the AND gate D because its side input has controlling logic value 0. The interval is complemented by inverting gates such as E. If two intervals reconverge, i.e., arrive at two inputs of a single gate such as XOR gate G preceding the output, intervals not starting at 0Ω or even having 'holes' may be produced. In the example, the fault-free logic value is 1, so the fault is detected by the applied test vector if R_{sh} is between R'_E and R'_C.

Circuit	Fault-free	Value assumed in section				
node	value	$[0, R'_D]$	$[R'_D, R_C]$	$[R_C, R'_E]$	$[R'_E, R'_C]$	$[R'_C, R_E]$
c	1	0	0	0	0	1
d	1	0	1	1	1	1
e	0	1	1	1	0	0
f	1	0	0	0	1	1
v	1	0	0	0	0	1
w	0	1	1	1	0	0
z	1	1	1	1	0	1

Table I. Section-based propagation in circuit from Figure 2

The logic values assumed by a node for individual sections are shown in Table I. For example, the second input c of gate C is assumes logic value 0 for bridge resistances between 0Ω and R'_C, i.e., in sections $[0, R'_D]$, $[R'_D, R_C]$, $[R_C, R'_E]$, $[R'_E, R'_C]$, and logic value 1 in the remaining section $[R'_C, R_E]$ (section $[R_E, \infty]$ is not shown as explained above). The values for individual sections can be propagated through the circuit based on the logic functions of the gates. It is obvious that, if the sections are known, five bits representing the logic values for sections contain exactly the same information as the interval-based representation. In particular, the values at the output of the circuit are (1, 1, 1, 0, 1), i.e., only section $[R'_E, R'_C]$ assumes logic value 0 and all others assume logic value 1) corresponds to ADI $[R'_E, R'_C]0/1$ generated by the interval propagation.

Simulating a section under a given input vector (only the values assigned to the inputs of the preceding gates matter indeed) can be represented as simulating a multiple-stuck-at fault. In our example, simulating section $[0, R'_D]$ under input 0111 corresponds to simulating stuck-at faults c stuck-at-0, d stuck-at-0, and e stuck-at-1 injected simultaneously. Stuck-at faults corresponding to all sections and inputs are shown in Table II.

The union of all ADIs on all outputs for all vectors is called *covered ADI* or C-ADI. C-ADI includes all the bridge resistances for which the fault has been detected by at least one test vector. C-ADI for the exhaustive test set is called *global ADI* or G-ADI, it represents all the resistances for which the fault could be detected. G-ADI can be calculated using ATPG techniques [Engelke et al. 2006a]. The *global fault coverage* G-FC is defined as the probability that a detectable fault f is detected:

$$G\text{-FC}(f) = 100\% \cdot \left(\int_{C\text{-ADI}} \rho(r)dr\right) / \left(\int_{G\text{-ADI}} \rho(r)dr\right),$$

where $\rho(r)$ is the probability density function of the short resistance r which can be derived from manufacturing data. It is possible to approximate G-ADI by $[0, R_m]$, where R_m is the largest critical resistance [Lee and Walker 2000]. R_m is much easier to calculate than G-ADI. The fault coverage which uses this approximation is called E-FC in [Engelke et al. 2006b]:

$$E\text{-FC}(f) = 100\% \cdot \left(\int_{C\text{-ADI}} \rho(r)dr\right) / \left(\int_{0}^{R_m} \rho(r)dr\right).$$

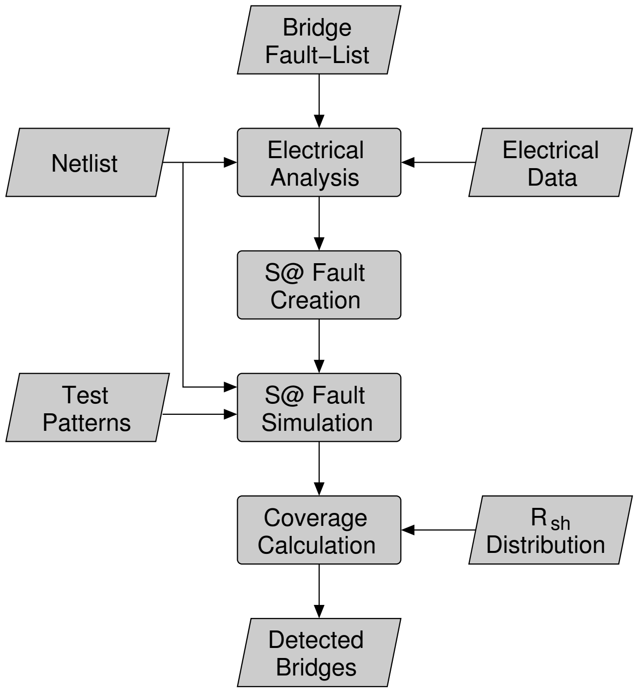


Fig. 3. Resistive bridging fault simulation flow

For multiple faults, average G-FC and E-FC are calculated. Redundant faults, i.e., faults with an empty G-ADI, are excluded.

Some sections are very small and their contribution to the fault coverage may be insignificant. It would be possible to achieve simulation speed-up by omitting such sections at the cost of some accuracy. We did not implement such techniques.

Input	Section	Stuck-at faults
0111	$[0, R'_D]$	c s-a-0, d s-a-0, e s-a-1
0111	$[R'_D, R_C]$	c s-a-0, e s-a-1
0111	$[R_C, R'_E]$	c s-a-0, e s-a-1
0111	$[R'_E, R'_C]$	c s-a-0
0111	$[R'_C, R_E]$	–
0111	$[R_E, \infty]$	–
0011	$[0, R'_D]$	c s-a-0, e s-a-1
0011	$[R'_D, R_C]$	c s-a-0, e s-a-1
0011	$[R_C, R'_E]$	e s-a-1
0011	$[R'_E, R'_C]$	e s-a-1
0011	$[R'_C, R_E]$	e s-a-1
0011	$[R_E, \infty]$	–

Table II. Multiple stuck-at faults corresponding to resistive bridging faults

4. SUPERB

Figure 3 shows the flow of SUPERB. First, SUPERB reads the list of resistive bridging faults and gate and process technology parameters. Electrical analysis is performed to obtain the critical resistances for all faults in the fault list [Renovell et al. 1994; 1995]. Then, the mapping of sections to multiple stuck-at faults is performed. For every section $[R_{i-1}, R_i]$, a hash table mapping possible input values of the gates driving the bridge I to the multiple stuck-at fault f_i^I is generated (explained in detail in Section 4.1). Then, 64-bit parallel-pattern or parallel-fault multiple-stuck-at fault simulation is performed (details can be found in Section 4.2). After the simulation has yielded the detection status of each section, the ADI is determined and aggregated to C-ADI and the fault coverage (G-FC or E-FC) is calculated.

4.1 Hash table generation

There is one hash table for every section of every considered resistive bridging fault. Let the fault f between nodes driven by gates a and b have a section $[R_L, R_U]$. The hash table for section $[R_L, R_U]$ of fault f contains all the logic value assignments to the inputs of gates a and b under which a resistive defect with a resistance between R_L and R_U has any faulty effect. Every such input value assignment serves as a key to look up the equivalent multiple-stuck-at faults. In the example from Table II, the hash table for section $[0, R'_D]$ would have two entries: $(0111 \mapsto \{c$ s-a-0, d s-a-0, e s-a-1$\})$ and $(0011 \mapsto \{c$ s-a-0, e s-a-1$\})$. The hash table for section $[R'_C, R_E]$ would have one entry $(0011 \mapsto \{e$ s-a-1$\})$.

The identification of section boundaries (i.e., critical resistances) and the calculation of equivalent multiple stuck-at faults is done using the electrical equations from [Renovell et al. 1994; 1995]. It would also be possible to use SPICE simulations instead such as done in [Lee and Walker 2000]. The keys, i.e., the input value assignments, are represented as 32-bit unsigned integers. This is valid because the length of a key cannot exceed $2 \cdot I_{\max}$ where $I_{\max}$ is the maximal number of inputs of a logic gate in the library and $I_{\max}$ is not supposed to exceed 16.

The maximal theoretical number of entries in a hash table is $2^{2 \cdot I_{\max}}$ but in practice the hash tables turn out to be quite small. The number of hash tables is the number of faults in the fault list multiplied by the number of sections per fault. The latter is typically a one-digit number, so the memory overhead for storing the hash tables is limited.

4.2 Fault simulation

SUPERB supports both parallel-pattern single-fault processing (PPSFP) and single-pattern parallel-fault processing (SPPFP). In PPSFP simulation, one section is fault-simulated for 64 test vectors $t_1, \ldots, t_{64}$ simultaneously. For this purpose, every node j is assigned a 64-bit string B_j represented using a machine word. The ith position of B_j corresponds to the logic value of node j under vector t_i with fault injected. If node j is an input of a gate succeeding the bridged lines, the (multiple-stuck-at) fault injection is done using two 64-bit masks: AND mask A_j and OR mask O_j. The ith position of A_j is set to 0 if a stuck-at-0 fault is injected at node j under test vector t_i and to 1 otherwise. The ith position of O_j is set to

1 if a stuck-at-1 fault is injected at node j under test vector t_i and to 0 otherwise.

The circuit is processed in topological order. A logic gate driving node j is simulated by applying its bit-wise logic function to the bit-strings of its inputs. In case of gates succeeding the bridged lines, a bit-wise AND operation with A_j and a bit-wise OR operation with O_j is performed first. For example, if a NOR3 gate succeeding the bridge drives node n and the gate's input bit-strings are B_k, B_l and B_m, then B_n is obtained as

$$B_n = \neg\,((B_k \wedge A_k \vee O_k) \vee (B_l \wedge A_l \vee O_l) \vee (B_m \wedge A_m \vee O_m))$$

where $\neg$, $\vee$ and $\wedge$ represent bit-wise NOT, OR and AND operations, respectively.

The generation of masks A_j and O_j is done for all inputs of gates succeeding the bridging fault to which the simulated section belongs, using the hash table of that section. For a bit position i, the input value assignments of the preceding gates are evaluated and the corresponding key of the hash table is looked up. The ith component of A_j is set to 0 if the hash table entry contains a stuck-at-0 fault on node j. The ith component of O_j is set to 1 if the hash table entry contains a stuck-at-1 fault on node j. Note that, in contrast to traditional pattern-parallel simulation, the masks do not always contain identical values. This is because the faults to be injected depend on the input value assignments of the preceding gates.

In parallel-fault (SPPFP) simulation, an input vector is simulated under 64 sections, not necessarily belonging to the same fault. The AND and OR masks are set at all the succeeding nodes of the resistive bridging faults to which the 64 sections belong. This is done by looking up the 64 hash tables for the individual sections using the key corresponding to the input value assignment induced by the simulated vector. The processing of individual gates is identical to the parallel-pattern case.

Figure 4 (a) illustrates the strings B and the masks A and O for four-bit PPSFP simulation of section $[R'_E, R'_C]$ in circuit from Figure 2 for input vectors 1111, 0000, 0111 and 0011. (Note that we assume the side inputs of gates C and D to be constantly at logic-0 and the side input of gate F to be constantly at logic-1.) Figure 4 (b) shows the information for four-bit SPPFP simulation of the same circuit for input vector 0111 and four sections shown in the figure. Note that SUPERB uses 64 bit parallelism and could also simulate sections which belong to different faults in one pass.

4.3 Relation to previous work

The authors of PROBE [Lee and Walker 2000] refer to their technique as to PPSFP simulation. Their approach is interval-based. They maintain a list of pointers to the intervals, and if an interval to be created already exists, they store only the pointer, avoiding to copy the interval. In contrast to SUPERB, they do not employ bit-parallelism for fault effect propagation. The simulator is implemented in tcl/tk and does not include advanced optimization techniques.

Cheung and Gupta use a slightly different electrical model which considers two logic thresholds (V_{IL} and V_{IH}) for each gate [Cheung and Gupta 2007]. Voltages between V_{IL} and V_{IH} are interpreted as the unknown value X. In this way, the simulator can, to some extent, account for noise margins and process variations. In contrast, our focus was on improving the performance of the existing (single-threshold) simulation [Engelke et al. 2006b] rather than on refining the model. The

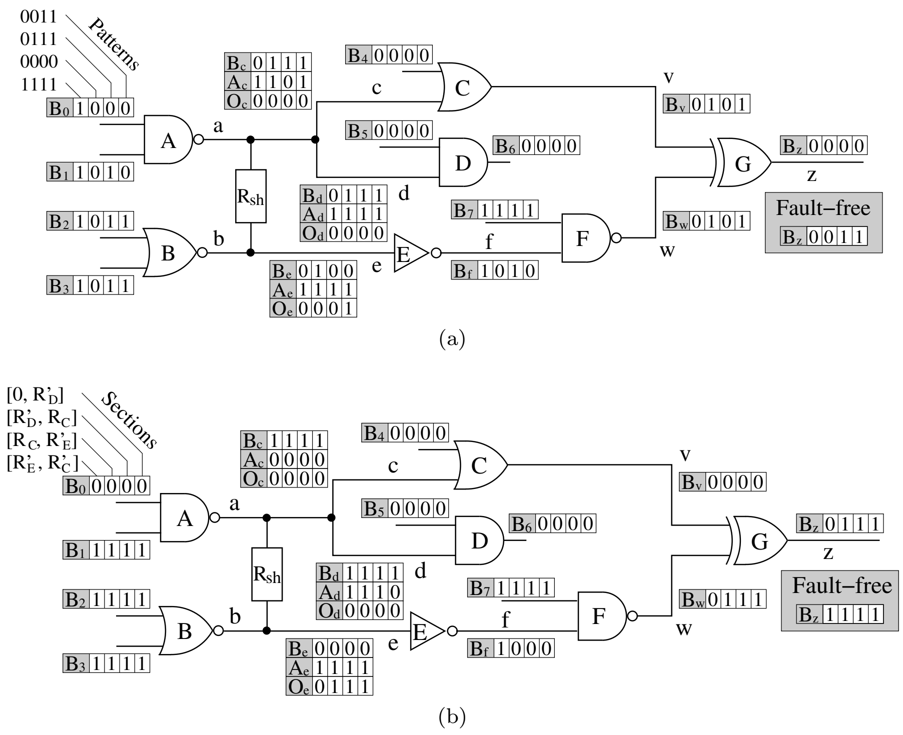


Fig. 4. Four-bit parallel-pattern simulation of section $[R'_E, R'_C]$ (a); four-bit parallel-fault simulation of input vector 0111 for circuit from Figure 2

fault coverage figures produced by SUPERB always match the numbers yielded by the simulator from [Engelke et al. 2006b]. SUPERB's optimization techniques not found in Cheung and Gupta's simulator are summarized below.

The hash tables employed by SUPERB have been heavily optimized. Although Cheung and Gupta give no details on the implementation of the fault characterization tables (their mechanism to relate the input values on the preceding gates and the sections with the logical values interpreted), we assume that our hash tables provide faster translation from sections to multiple stuck-at faults. Furthermore, the fault lists in Cheung and Gupta's simulator contain bridging faults; the simulator has to generate sections for a given bridging fault before running the parallel-pattern simulation. SUPERB's fault lists consist of sections, which have a one-to-one correspondence to multiple stuck-at faults for a fixed input vector. This simplifies the handling of the fault list by existing (multiple) stuck-at simulation engines. In addition, this allows both parallel-pattern and parallel-fault simulation, as 64 faults handled by the stuck-at simulation engine may correspond to sections that belong to different bridging faults. Finally, SUPERB supports event-driven simulation while Cheung and Gupta's simulator traverses the full fanout cone of the bridged signals.

Circuit	Cells	PI	PO	Resistive bridging fault simulation							Stuck-at fault sim.		
				Faults	*E*-FC	*G*-FC	Preproc. [s]	Sim. [s]	T/BV [ms]	Mem [MB]	Faults	FC	Time [s]
ISCAS 85 circuits													
c2670	1,566	233	140	15,660	96.65	97.70	0.12	7.26	0.00005	46	2,747	88.31	0.08
c3540	1,741	50	22	17,410	98.81	99.94	0.31	5.67	0.00003	58	3,428	95.83	0.04
c5315	2,608	178	123	26,080	99.65	100.00	0.31	3.81	0.00002	69	5,350	98.90	0.05
c6288	2,480	32	32	24,800	91.65	100.00	0.22	39.33	0.00016	63	7,744	99.56	0.17
c7552	3,827	207	108	38,270	99.04	99.53	0.38	12.79	0.00003	94	7,550	94.29	0.14
Average (all ISCAS 85 circuits)					97.64	99.72	0.14	6.57	0.00011			97.44	0.05
ISCAS 89 circuits (combinational cores)													
cs01488	692	14	25	6920	97.94	99.96	0.06	2.03	0.00003	27	1,486	99.80	0.01
cs01494	686	14	25	6860	98.12	99.96	0.07	2.25	0.00003	27	1,506	99.00	0.01
cs05378	3,221	214	228	32210	99.17	99.80	0.23	8.53	0.00003	77	4,603	98.18	0.06
cs09234	6,094	247	250	60940	95.56	96.95	0.34	36.92	0.00006	128	6,927	84.55	0.48
cs13207	9,441	700	790	94410	98.09	98.40	0.55	37.15	0.00004	194	9,815	90.92	0.46
cs15850	11,067	611	684	110670	98.26	98.81	0.61	34.67	0.00003	221	11,725	91.57	0.59
cs35932	19,876	1,763	2,048	198760	96.40	100.0	1.40	132.28	0.00007	418	39,094	89.81	2.03
cs38417	25,585	1,664	1,742	255850	96.00	98.11	1.61	159.89	0.00006	524	31,180	88.75	1.69
cs38584	22,447	1,464	1,730	224470	95.87	97.64	1.87	129.01	0.00006	479	36,303	94.52	1.57
Average (all ISCAS 89 circuits)					96.10	98.74	0.25	19.72	0.00008			95.16	0.23
ITC 99 circuits (combinational cores)													
b14	5,923	277	299	59,230	96.85	97.81	0.95	81.57	0.00014	159	16,167	88.87	0.96
b15	8,026	485	519	80,260	89.56	90.88	1.16	230.41	0.00029	206	21,282	73.64	6.10
b17	25,719	1,451	1,511	257,190	87.30	87.93	4.03	963.14	0.00038	671	68,207	71.10	25.36
b18	76,513	3,307	3,293	765,130	89.14	–	12.01	3826.46	0.00050	1996	206,812	75.80	137.65
b20	12,991	522	512	129,910	97.67	98.34	2.00	163.44	0.00013	342	35,731	91.65	2.27
b21	13,168	522	512	131,680	97.51	98.16	1.88	167.47	0.00013	345	36,058	90.48	2.69
b22	18,789	735	725	187,890	97.89	98.47	2.90	223.87	0.00012	490	51,341	92.01	3.82
Average (all ITC 99 circuits)					94.76	96.80	1.81	416.09	0.00016			90.51	13.49

Table III. Experimental results for ISCAS 85 and combinational cores of ISCAS 89 and ITC 99 circuits, 10,000 test vectors

Circuit	Cells	PI	PO	Resistive bridging fault simulation						Stuck-at fault simulation		
				Faults	*E*-FC	Preproc. [s]	Sim. [s]	*T*/*BV* [ms]	Mem [GB]	Faults	FC [s]	Time
p35k	48,927	2,912	2,229	489,270	82.44	3.50	922.17	0.00019	1.0	67,733	58.68	14.49
p45k	46,075	3,739	2,550	460,750	97.74	3.55	360.13	0.00008	1.0	68,760	93.04	9.91
p77k	75,033	3,487	3,400	750,330	78.50	6.14	24,942.20	0.00332	1.5	120,348	59.51	2,635.85
p78k	80,875	3,148	3,484	808,750	97.85	7.00	519.27	0.00007	1.7	163,310	100.00	3.24
p81k	96,722	4,029	3,952	967,220	87.17	12.81	1,728.02	0.00018	2.2	204,174	69.27	18.14
p89k	92,706	4,683	4,557	927,060	92.00	7.96	1,426.09	0.00015	1.9	150,538	75.89	26.91
p100k	102,443	5,902	5,829	1,024,430	98.28	8.88	824.15	0.00008	2.1	162,129	93.57	27.46
p141k	185,360	11,290	10,502	1,853,600	98.02	14.77	987.11	0.00005	3.8	282,428	91.50	24.71
p267k	296,404	17,332	16,621	2,964,040	97.10	20.49	2,075.97	0.00007	5.8	366,871	90.41	42.35
p269k	297,497	17,333	16,621	2,974,970	97.12	21.14	2,115.01	0.00007	5.8	369,055	90.59	43.00
p295k	311,901	18,508	18,521	3,119,010	90.80	24.28	4,214.95	0.00014	6.3	472,022	77.63	70.04
p330k	365,492	18,010	17,468	3,654,920	96.12	29.20	2,706.34	0.00007	7.3	540,758	86.66	61.06
p378k	404,367	15,732	17,420	4,043,670	97.96	35.81	2,702.50	0.00007	8.3	816,534	100.00	23.02
p388k	506,034	25,005	24,065	5,060,340	98.87	42.40	2,223.50	0.00004	10.3	881,417	96.06	71.84
p469k	49,771	635	403	497,710	98.43	8.74	22,091.80	0.00444	1.3	142,751	98.53	2,840.19
p951k	1,147,491	92,027	104,747	11,474,910	99.01	93.78	4,535.13	0.00004	22.9	1,557,914	95.32	127.63
p1522k	1,193,824	71,414	68,035	11,938,240	93.26	99.36	15,775.47	0.00013	23.9	1,697,662	80.91	287.23
p2927k	2,539,052	101,844	95,143	25,390,520	96.57	184.06	27,668.16	0.00011	48.8	3,527,607	88.56	1,100.29
Average					94.29	34.66	6,545.44	0.00052			85.90	412.63

Table IV. Experimental results for combinational cores of NXP circuits, 10,000 test vectors

Circuit	[Shinogi et al. 2001]			PROBE [Lee and Walker 2000]			SUPERB (PPSFP)		
	B	T [s]	T/BV [ms]	B	T [s]	T/BV [ms]	B	T [s]	T/BV [ms]
c432	n/a	n/a	n/a	157	780.00	0.49602	5,253	1.24	0.00002
c499	n/a	n/a	n/a	136	900.00	0.66071	8,985	0.76	0.00001
c880	1,000	18.00	0.00180	949	6,960.00	0.73223	10,000	3.70	0.00004
c1355	1,000	114.00	0.01140	639	12,960.00	2.02493	10,000	7.17	0.00007
c1908	2,000	78.00	0.00390	1,662	37,680.00	2.26353	10,000	2.97	0.00003
c2670	5,000	840.00	0.01680	4,294	151,020.00	3.51138	10,000	4.69	0.00005
c3540	5,000	720.00	0.01440	4,431	202,860.00	4.57089	10,000	3.18	0.00003
c5315	8,000	540.00	0.00675	7,121	418,500.00	5.86760	10,000	1.34	0.00001
c6288	4,000	1,800.00	0.04500	3,216	603,240.00	18.72750	10,000	13.77	0.00014
c7552	13,000	2,700.00	0.02077	12,106	1,194,480.00	9.85108	10,000	3.29	0.00003
Σ, $\emptyset$		6,810.00	0.01510		2,629,380.00	4.87059		42.11	0.00004

Table V. Comparison with previous approaches

5. EXPERIMENTAL RESULTS

5.1 Performance of SUPERB

We applied 10,000 random test vectors to the ISCAS 85 circuits and the combinational cores of ISCAS 89 and ITC 99 circuits and industrial circuits by NXP. The number of faults considered equaled the number of cells in the circuit multiplied by 10. We selected the faults randomly. The faults could also be selected based on layout analysis [Ferguson and Shen 1988]. The number of faults was chosen to be close to typical numbers of faults obtained by layout-based selection procedures. Since SUPERB currently does not handle oscillation, we considered only non-feedback faults (simulation techniques for feedback faults are described in [Polian et al. 2005]).

We employed the density function ρ derived from one used in [Lee and Walker 2000] (using a uniform distribution [Spica et al. 2001] increased the fault coverage by 0.65% on average). We calculated G-ADI for ISCAS 85 and 89 and all except one ITC 99 circuits using the tool from [Engelke et al. 2006a]; it was not applicable to larger circuits, so no G-FC can be reported for them. Note that the tool from [Engelke et al. 2006a] is a test generation tool and not part of SUPERB.

Columns 1 through 5 of Tables III and IV contain the name of the circuit, the number of cells, inputs and outputs of its combinational core and the number of simulated resistive bridging faults. The subsequent columns report the obtained RBF coverage (E-FC and G-FC for ISCAS and ITC 99 circuits, only E-FC for NXP circuits), the time for preprocessing, the simulation time, the time per bridging fault and vector (in milliseconds), and the memory demand. For comparison, stuck-at fault simulation has been performed for the same 10,000 random vectors using the same simulation engine. The final three columns of Table IV contain the number of stuck-at faults, the achieved stuck-at fault coverage and the simulation time. The last row of the table details average numbers. The measurement was performed on a 2.8 GHz AMD Opteron Linux machine with 16 GB RAM.

Table III does not quote results for all the benchmarks, skipping the smaller circuits. Note that the average numbers refer to the complete respective benchmark suite, including the skipped circuits. The run times reported here differ slightly compared to [Engelke et al. 2008] because we repeated all measurements to ensure comparability with earlier approaches.

The RBF coverage achieved by 10,000 random vectors tends to be lower for larger circuits, although there appear to be smaller circuits with many random-pattern resistant faults such as p77k. The coverage is typically higher than the stuck-at fault coverage, but it does not track for all circuits. For instance, the RBF coverage of p100k significantly exceeds its stuck-at coverage while for p78k the opposite is the case. For circuits for which G-FC could be calculated it often significantly exceeds E-FC (more details on approximations of G-FC can be found in [Engelke et al. 2006a; 2006b]).

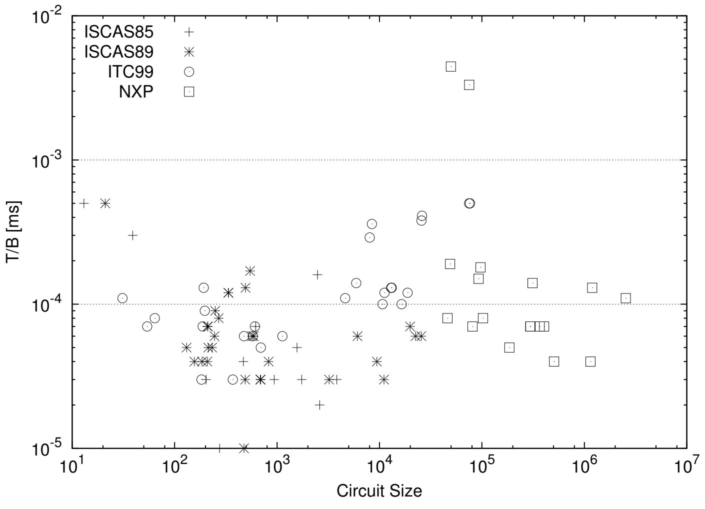
The largest simulation time of approximately 8 hours is required for the 2.5 million gate design p2927k. The average simulation time of the RBF simulation is approximately 19 times larger than that of the stuck-at fault simulation (this ratio is higher for smaller circuits). Given that the number of faults is approximately five times larger, the average simulation speed is competitive. Note that the actual number of multiple stuck-at faults simulated during RBF simulation is still higher because one RBF consists of multiple sections and every section is mapped to one pattern-dependent multiple stuck-at fault.

Figure 5 (a) shows the simulation time per bridging fault as a function of the circuit size. It can be seen that, with exception of two outliers, the time per fault is almost independent of the circuit size. Figure 5 (b) plots the RBF simulation time against the stuck-at simulation time. The dependency is almost perfectly linear, with exception of the smallest circuits.

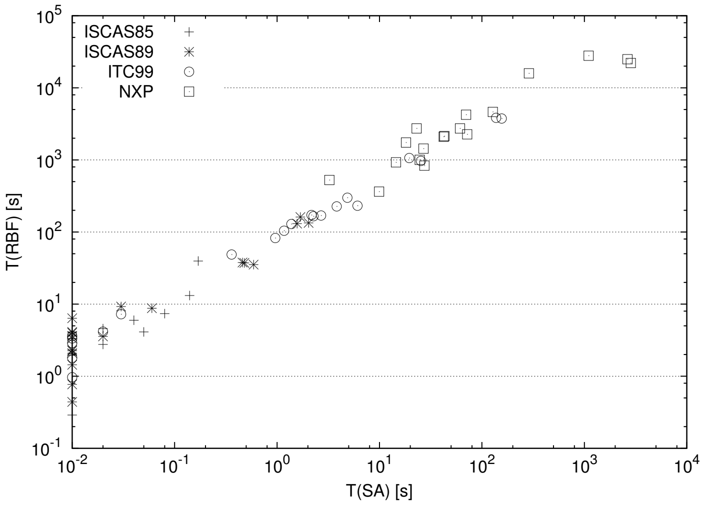
In addition to the memory usage figures reported here, we investigated the memory consumption as the function of the number of faults in the fault list. We found that SUPERB allocates some fixed amount of memory for the circuit and the test vectors. The remaining memory usage was almost perfectly linear in the number of faults. For example, the ITC 99 circuit b17 required some 37 MB with an empty fault list and some 2.5 MB for every 1,000 faults in the fault list. This memory is consumed by the hash tables and further fault-related data structures. SUPERB's total memory requirements are thus sub-linear in number of faults. If the available memory is not sufficient, SUPERB can be applied to portions of the fault list, and the final simulation outcome can be composed of the individual results.

5.2 Comparison with previous approaches

Table V compares the 32-bit parallel version of SUPERB with the sectioning-based tool from [Shinogi et al. 2001] and the interval-based tool PROBE [Lee and Walker 2000]. We do not compare with the tool from [Cheung and Gupta 2007] because it is based on a different electrical model and requires more processing to simulate the unknown values. For each tool, the number of bridging faults B (not sections) and the reported run time T in seconds are given. Note that the original papers [Shinogi et al. 2001; Lee and Walker 2000] report the run time in minutes. The number of used test vectors V was always 10,000 for the tool from [Shinogi et al. 2001] and SUPERB and 10,016 for PROBE. To allow a comparison, we calculate the normalized run time as run time T divided by the product of the number of bridges B and the number of vectors V and report the result (in milliseconds) in columns T/BV.



(a)



(b)

Fig. 5. Simulation time per resistive bridging fault as function of circuit size (a); simulation time for resistive bridging faults vs. stuck-at faults (b)

The final row contains the sums of all run times and the average normalized run times. SUPERB is approximately two orders of magnitude faster than the tool from [Shinogi et al. 2001] and five orders of magnitude faster than PROBE [Lee and Walker 2000]. Note that the experiments in [Shinogi et al. 2001] were run on a 850 MHz Pentium III, which may be three to five times slower than the computer we used. Moreover, the average number of sections per fault for the circuits in Table V was 3.61 for the tool from [Shinogi et al. 2001] and 3.11 for SUPERB, which may account for some 15% of the acceleration. The measurement in [Lee and Walker

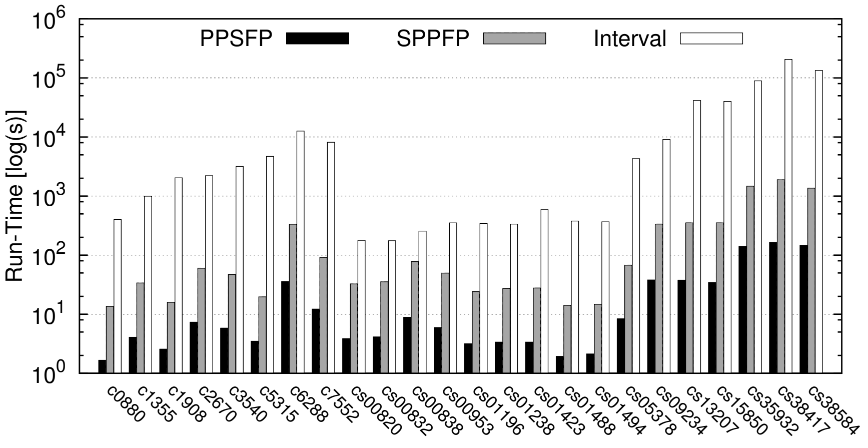


Fig. 6. Run time of SUPERB vs. interval-based approach (logscale)

2000] was done on a Sun SPARC 5 with an unspecified clock frequency which should not have resulted in a slowdown of more than two orders of magnitude. The remaining difference must be attributed to the interval-based simulation (despite the acceleration through the shared pointers) and the implementation of PROBE in tcl/tk.

Figure 6 shows the run time of SUPERB in PPSFP mode, SPPFP mode and the interval-based tool from [Engelke et al. 2006b] for identical fault lists and 10,000 identical test vectors on the same machine, in logscale form. On average, the PPSFP mode outperforms the SPPFP mode by approximately a factor of 10. We observed similar acceleration on combinational cores of ITC 99 and NXP circuits not reported here. The average speed-up of SUPERB compared to the interval-based tool is approximately 800. The interval-based tool did not terminate for industrial circuits in feasible time.

5.3 Double Errors Induced by Resistive Bridging Faults

A resistive bridging fault between circuit nodes a and b may induce erroneous logical values on both nodes simultaneously. For instance, suppose that node a drives a logic gate with logic threshold Th_1 and that the fault-free logical value on a is 1. Node b drives a gate with threshold $Th_2 < Th_1$ and b's fault-free value is 0. A defect with a fixed resistance R_{sh} results in node a having voltage level $V_a(R_{sh})$ and node b having voltage level $V_b(R_{sh}) \leq V_a(R_{sh})$. If $V_a(R_{sh}) < Th_1$ and $V_b(R_{sh}) > Th_2$, the gates driven by a and b will both interpret erroneous logical value. This behavior is not reflected by simple bridging fault models in which one bridged node changes its value while the other node's value stays unchanged.

We refer to the situation when both bridged nodes changing their logical value simultaneously as the *RBF-induced double error.*[1] Probability of such errors is

[1]If nodes a or b drive several gates, some of the driven gates may interpret the fault-free value

relevant for the analysis of fault-tolerance techniques which often assume that a single source of errors is present in the circuit. This assumption is not valid for RBF-induced double errors.

We investigate the probability that an RBF-induced double error shows up in a general circuit. For this purpose, we apply the local analysis employed by SUPERB for hash table generation to all possible resistive bridging faults. We exhaustively enumerate all combinations of gates driving the bridged nodes, values on their inputs, and gates driven by the bridged nodes. We consider only gates which show up in at least one ISCAS 85, ISCAS 89, ITC 99 or NXP circuit. For each combination, we extract resistance intervals (sections) in which a double error as defined above shows up.

In total, we evaluated 35,721,000 combinations. Out of them, RBF-induced double errors were possible for at least one bridge resistance in 61,792 cases (0.172985 per cent). For every such case we related the resistance interval in which double errors were observable to the maximum detectable resistance range to account for the fault coverage impact in a manner similar to E-FC. This yielded an average coverage of 18.8481 per cent. This means that the total double error probability is $0.00172985 \cdot 0.188481 = 0.0003260438$, or approximately 0.03 per cent. In conclusion, RBF-induced double errors are possible but quite rare. It appears that this mechanism can be safely ruled out as a source of double errors when designing fault-tolerance techniques or assessing their efficiency.

6. CONCLUSIONS

We presented an approach for fast simulation of resistive bridging faults. Sections of bridge resistances are mapped to multiple-stuck-at faults, thus enabling to employ acceleration techniques developed for stuck-at fault simulation. The mapping takes the pattern-dependent circuit behavior into account. While retaining the accuracy of the interval-based simulation, speed-ups of several orders of magnitude are achieved compared to both interval-based and sectioning-based tools proposed before. We also demonstrated the applicability of SUPERB to large industrial circuits with fault lists of realistic size. The simulation effort is not far away from the effort for stuck-at fault simulation, and the gap decreases for larger circuits. Based on the developed methodology, we analyzed the probability of RBF-induced double errors and found it to be low, yet not zero. Possible extensions are support of features present in industrial circuits such as Z values and incorporation of the model for feedback faults from [Polian et al. 2005].

Acknowledgments

Parts of this work are supported by the German Research Foundation under grant BE 1176/14-1. We are grateful to Prof. Gössel of the University of Potsdam for pointing us to the double-error issue investigated in Section 5.3.

while others interpret the erroneous value, depending on their threshold. In our definition of double error, we require that at least one successor of both a and b interprets the erroneous value. The situation when several gates driven by one node but none driven by the other node interpret the erroneous value is not considered double error.

REFERENCES

AITKEN, R. C. 1995. Finding defects with fault models. In *Int'l Test Conf.* 498–505.

BANERJEE, P. AND ABRAHAM, J. A. 1985. A multivalued algebra for modeling physical failures in mos vlsi circuits. *IEEE Trans. on CAD 4,* 5, 312–321.

CHANG, Y.-S., GUPTA, S. K., AND BREUER, M. A. 1999. Test generation for ground bounce in internal logic circuitry. In *VLSI Test Symp.* 95–104.

CHEN, G., REDDY, S. M., POMERANZ, I., RAJSKI, J., ENGELKE, P., AND BECKER, B. 2005. An unified fault model and test generation procedure for interconnect opens and bridges. In *European Test Symposium*. 22–27.

CHEUNG, H. AND GUPTA, S. K. 2007. Accurate modeling and fault simulation of byzantine resistive bridges. In *IEEE Int'l Conf. on Computer Design*. 347–353.

ENGELKE, P., BRAITLING, B., POLIAN, I., RENOVELL, M., AND BECKER, B. 2007. SUPERB: Simulator utilizing parallel evaluation of resistive bridges. In *Asian Test Symp.* 433–438.

ENGELKE, P., POLIAN, I., RENOVELL, M., AND BECKER, B. 2006a. Automatic test pattern generation for resistive bridging faults. *Jour. Electronic Testing: Theory and Applications 22,* 1 (February), 61–69.

ENGELKE, P., POLIAN, I., RENOVELL, M., AND BECKER, B. 2006b. Simulating resistive bridging and stuck-at faults. *IEEE Trans. on CAD 25,* 10 (October), 2181–2192.

ENGELKE, P., POLIAN, I., SCHLÖFFEL, J., AND BECKER, B. 2008. Resistive bridging fault simulation of industrial circuits. In *Conf. on Design, Automation and Test in Europe*. 628–633.

FAVALLI, M., DALPASSO, M., OLIVO, P., AND RICCÒ, B. 1993. Analysis of dynamic effects of resistive bridging faults in CMOS and BiCMOS digital ICs. In *Int'l Test Conf.* 865–873.

FERGUSON, F. J. AND LARRABEE, T. 1991. Test pattern generation for realistic bridge fault in CMOS ICs. In *Int'l Test Conf.* 492–499.

FERGUSON, F. J. AND SHEN, J. 1988. Extraction and simulation of realistic CMOS faults using inductive fault analysis. In *Int'l Test Conf.* 475–484.

GREENSTEIN, G. S. AND PATEL, J. H. 1992. E-PROOFS: a CMOS bridging fault simulator. In *Int'l Conf. on CAD*. 268–271.

GRIMAILA, M. R., LEE, S., DWORAK, J., BUTLER, K. M., STEWART, B., BALACHANDRAN, H., HOUCHINS, B., MATHUR, V., PARK, J., WANG, L.-C., AND MERCER, M. R. 1999. REDO-random excitation and deterministic observation–first commercial experiment. In *VLSI Test Symp.* 268–274.

HAO, H. AND MCCLUSKEY, E. J. 1991. Resistive shorts within CMOS gates. In *Int'l Test Conf.* 292–301.

HAWKINS, C. F., SODEN, J., RIGHTER, A., AND FERGUSON, F. J. 1994. Defect classes - an overdue paradigm for CMOS IC testing. In *Int'l Test Conf.* 413–425.

KHARE, J. AND MALY, W. 1996. *From contamination to defects, faults and yield loss.* Kluwer Academic Publisher.

KONUK, H., FERGUSON, F. J., AND LARRABEE, T. 1995. Accurate and efficient fault simulation of realistic CMOS network breaks. In *ACM IEEE Design Automation Conf.* 345–351.

KRSTI, A., JIANG, Y. M., AND CHENG, K.-T. 2001. Pattern generation for delay testing and dynamic timing analysis considering power-supply noise effects. *IEEE Trans. on CAD 20,* 3, 416–425.

LEE, C. AND WALKER, D. M. H. 2000. PROBE: A PPSFP simulator for resistive bridging faults. In *VLSI Test Symp.* 105–110.

LI, Z., LU, X., QIU, W., SHI, W., AND WALKER, D. M. H. 2003. A circuit level fault model for resistive bridges. *ACM Trans. on Design Automation of Electronic Systems 8,* 4 (October), 546–559.

LIAO, Y. AND WALKER, D. M. H. 1996. Fault coverage analysis for physically-based CMOS bridging faults at different power supply voltages. In *Int'l Test Conf.* 767–775.

MA, S. C., FRANCO, P., AND MCCLUSKEY, E. J. 1995. An experimental chip to evaluate test techniques experimental results. In *Int'l Test Conf.* 663–672.

MAEDA, T. AND KINOSHITA, K. 2000. Precise test generation for resistive bridging faults of CMOS combinational circuits. In *Int'l Test Conf.* 510–519.

MALY, W. 1987. Realistic fault modeling for VLSI testing. In *ACM IEEE Design Automation Conf.* 173–180.

MAXWELL, P. C. AND AITKEN, R. C. 1993. Biased voting: A method for simulating CMOS bridging faults in the presence of variable gate logic thresholds. In *Int'l Test Conf.* 63–72.

MEI, K. C. Y. 1974. Bridging and stuck-at faults. *IEEE Trans. on CAD C-23,* 7 (July), 720–727.

MILLMAN, S. D. AND GARVEY, S. J. P. 1991. An accurate bridging fault test pattern generator. In *Int'l Test Conf.* 411–418.

MITRA, D., BHATTACHARJEE, S., SUR-KOLAY, S., BHATTACHARYA, B. B., ZACHARIAH, S. T., AND KUNDU, S. 2006. Test pattern generation for power supply droop faults. In *VLSI Design*.

NOURANI, M., TEHRANIPOUR, M., AND AHMED, N. 2005. Pattern generation and estimation for power-supply noise analysis. In *VLSI Test Symp.* 439–444.

POLIAN, I., ENGELKE, P., RENOVELL, M., AND BECKER, B. 2005. Modeling feedback bridging faults with non-zero resistance. *Jour. Electronic Testing: Theory and Applications 21,* 1 (February), 57–69.

POLIAN, I., KUNDU, S., GALLIERE, J.-M., ENGELKE, P., RENOVELL, M., AND BECKER, B. 2005. Resistive bridge fault model evolution from conventional to ultra deep submicron technologies. In *VLSI Test Symp.* 343–348.

POLIAN, I., POMERANZ, I., REDDY, S. M., AND BECKER, B. 2004. On the use of maximally dominating faults in n-detection test generation. *IEE Proceedings Computers and Digital Techniques 151,* 3 (May), 235–244.

REARICK, J. AND PATEL, J. H. 1993. Fast and Accurate CMOS Bridging Fault Simulation. In *Int'l Test Conf.* 54–62.

REDDY, S. M., POMERANZ, I., AND KAJIHARA, S. 1997. Compact test sets for high defect coverage. *IEEE Trans. on CAD 16*, 923–930.

RENOVELL, M., AZAÏS, F., AND BERTRAND, Y. 1999. Detection of defects using fault model oriented test sequences. *Jour. Electronic Testing: Theory and Applications 14*, 13–22.

RENOVELL, M., HUC, P., AND BERTRAND, Y. 1994. CMOS bridge fault modeling. In *VLSI Test Symp.* 392–397.

RENOVELL, M., HUC, P., AND BERTRAND, Y. 1995. The concept of resistance interval: A new parametric model for resistive bridging fault. In *VLSI Test Symp.* 184–189.

RODRÍGUEZ-MONTAÑÉS, R., BRULS, E. M. J. G., AND FIGUERAS, J. 1992. Bridging defects resistance measurements in a CMOS process. In *Int'l Test Conf.* 892–899.

SAR-DESSAI, V. AND WALKER, D. M. H. 1998. Accurate fault modeling and fault simulation of resistive bridges. In *Int'l Symp. Defect and Fault Tolerance in VLSI Systems.* 102–107.

SAR-DESSAI, V. AND WALKER, D. M. H. 1999. Resistive Bridge Fault Modeling, Simulation and Test Generation. In *Int'l Test Conf.* 596–605.

SENGUPTA, S., KUNDU, S., CHAKRAVARTY, S., PARAVATHALA, P., GALIVANCHE, R., KOSONOCKY, G., RODGERS, M., AND MAK, T. M. 1999. Defect-based test: A key enabler for successful migration to structural test. *Intel Technology Journal 1.*

SHINOGI, T., KANBAYASHI, T., YOSHIKAWA, T., TSURUOKA, S., AND HAYASHI, T. 2001. Faulty resistance sectioning technique for resistive bridging fault ATPG systems. In *Asian Test Symp.* 76–81.

SPICA, M., TRIPP, M., AND ROEDER, R. 2001. A new understanding of bridge defect resistances and process interactions from correlating inductive fault analysis predictions to empirical test results. In *IEEE Int'l Workshop on Current and Defect-Based Testing.* 11–16.

VIERHAUS, H., MEYER, W., AND GLÄSER, U. 1993. CMOS bridges and resistive transistor faults: IDDQ versus delay effects. In *Int'l Test Conf.* 83–91.

ZACHARIAH, S. T. AND CHAKRAVARTY, S. 2000. A scalable and efficient methodology to extract two node bridges from large industrial circuits. In *Int'l Test Conf.* 750–759.

Received Month Year