Fig. 10. Measurement results: the output signal of the preamplifier and the gain control signal.
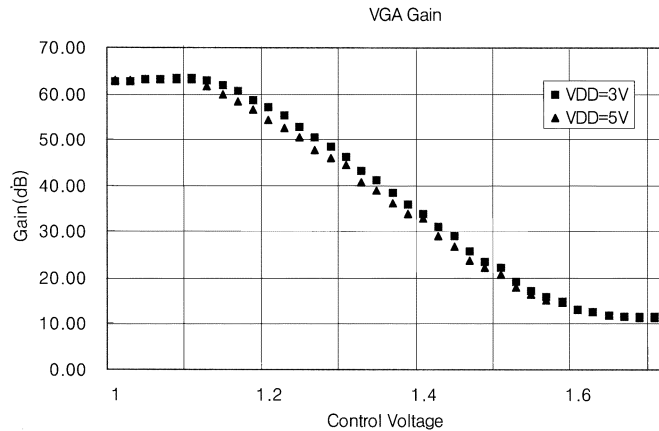

Fig. 11. Test result of the VGA. When the AGC loop is stable, the control voltage of the VGA is 1.28 V and the gain of the preamplifier is 50 dB similar to simulated one.

IV. CONCLUSION

In this paper, a transient noise model for frequency-dependent noise sources such as $1/f$ noise of G-R noise has been introduced. Therefore, we can make noise signals for transient simulation not only from white noise sources but also from frequency-dependent noise sources. By using the proposed noise model, we can simulate dynamic and exact noise performances of circuits including noise peak detector or large gain stage. Measured noise performances of designed OSP are almost the same as simulated ones. Using this model, we can reduce the design time because the transient noise performance can be simulated.

REFERENCES

[1] C. D. Motchenbacher and J. A. Connelly, *Low Noise Electronic System Design*. New York: Wiley, 1993, pp. 8–32.
[2] A. Van der Ziel and S. K. Kim, "Generation-recombination noise at 77 K in silicon bars and JFET's," *Solid-State Electron.*, vol. 22, no. 2-D, pp. 177–179, 1979.
[3] R. G. Meyer, L. Nagel, and S. K. Lui, "Computer simulation of $1/f$ noise performance of electronic circuits," *IEEE J. Solid-State Circuits*, vol. SC-6, pp. 237–240, June 1973.
[4] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes*. Cambridge, U.K.: Cambridge Univ. Press, 1986, pp. 202–203.
[5] A. Van der Ziel, "Thermal noise in field effect transistors," *Proc. IEEE*, vol. 50, pp. 1808–1812, Aug. 1962.
[6] A. R. Bennet, *Electronical Noise*. New York: McGraw-Hill, 1960, p. 42.
[7] P. Z. Peebles Jr. , *Probability, Random Variables, and Random Signal Processor*. New York: McGraw-Hill, 1993, pp. 147–150.
[8] A. B. Carson, *Communication Systems*. New York: McGraw-Hill, 1986, pp. 143–146.
[9] ——, *Communication Systems*. New York: McGraw-Hill, 1986, pp. 38–39.
[10] B. Porat, *A Course in Digital Signal Processing*. New York: Wiley, 1997, pp. 101–104.

A Test Evaluation Technique for VLSI Circuits Using Register-Transfer Level Fault Modeling

Pradip A. Thaker, Vishwani D. Agrawal, and Mona E. Zaghloul

Abstract—Stratified fault sampling is used in register transfer level (RTL) fault simulation to estimate the gate-level fault coverage of given test patterns. RTL fault modeling and fault-injection algorithms are developed such that the RTL fault list of a module can be treated as a representative fault sample of the collapsed gate-level stuck-at fault set of the module. The RTL coverage for the module is experimentally found to track the gate-level coverage within the statistical error bounds. For a very large scale integration system, consisting of several modules, the level of description may differ from module to module. Therefore, the stratified fault sampling technique is used to determine the overall coverage as a weighted sum of RTL module coverages. Several techniques are proposed to determine these weights, known as stratum weights. For a system timing controller application specific integrated circuit, the stratified RTL coverage of verification test-benches is estimated to be within 0.6% of the actual gate-level coverage of the synthesized circuit. This ASIC consists of 40 modules (consisting of 9000 lines of Verilog hardware description language) that are synthesized into 17 126 equivalent logic gates by a commercial synthesis tool. Similar results on two other systems are reported.

Index Terms—Design for test, RTL fault coverage, RTL fault modeling, stratified fault sampling.

I. INTRODUCTION

Test patterns for large very large scale integration (VLSI) systems are often derived from the knowledge of the circuit function. A fault simulator is then used to find the effectiveness of the test patterns in

detecting gate-level "stuck-at" faults. Existing gate-level fault-simulation techniques suffer prohibitively expensive performance penalties when applied to the modern VLSI systems of larger sizes. Also, findings of such test generation and fault-simulation efforts in the post logic-synthesis phase are too late in the design cycle to be useful for design-for-test (DFT)-related improvements in the architecture. Therefore, an effective register-transfer level (RTL) fault model is highly desirable.

Several high-level fault models and fault-simulation techniques have been proposed by many authors including Thatte and Abraham [1], Gosh and Chakraborty [2], Ghosh [3], Ward and Armstrong [4], Armstrong *et al.* [5], Cho and Armstrong [6], and Sanchez and Hidalgo [7]. None of these techniques establish the relationship between high-level fault coverage and gate-level fault coverage. Mao and Gulati [8] proposed an RTL fault model and a simulation methodology but did not establish the relationship of RTL faults to gate-level faults. Their approach also required one to run fault simulation twice (first in an optimistic and then in a pessimistic mode) and to use the average of the results to reduce the difference between the RTL and the gate-level fault coverages. This is an inefficient solution derived purely empirically. The authors did not establish any theoretical basis to generalize the application of their fault model. Their experimental data indicate as much as a 10% error between the actual gate-level fault coverage and the RTL fault coverage. Ferrandi *et al.* [9] presented stuck-at fault model along with binary decision diagram (BDD) approach to create faulty and fault-free control data-flow graphs (CDFGs) to guide the test vector generation process. Both approaches [8], [9] do not establish any predetermined error bounds between RTL coverage reported by their proposed fault model and actual gate-level fault coverage. Hayne and Johnson [10] developed a fault model based on an abstraction of the industry standard single-stuck-line faults in the behavioral domain. This fault model was developed such that for every possible gate-level fault in the circuit, there is a corresponding faulty RTL circuit. Similar efforts by others were focused on developing a model that, when applied to the RTL description, could produce the behavior of all possible gate-level single "stuck-at" faults. The RTL fault models that fall short of achieving this goal have been considered incomplete.

The procedure presented in this paper works on the premise that all hardware (gate-level) faults may not be represented at the RTL since the RTL description is a higher level of abstraction, which may not contain the low-level structural information implicit in many gate-level failures. Also, since the gate-level netlist can change drastically with every logic synthesis iteration, efforts to model all possible gate faults at the RTL are inefficient. Instead, in this paper, an RTL fault model and fault-injection algorithm are developed such that the RTL fault-list of a module becomes a representative sample of the corresponding collapsed gate-level fault-list. The proposed RTL fault model and the fault-injection algorithm are developed from an analysis of the properties of the gate-level single stuck-at fault (SSF) model and mapping of RTL constructs onto gate-level structures during logic synthesis.

In this paper, the problem of RTL fault modeling is addressed in two parts. First, RTL fault model and an injection algorithm is developed for single RTL module. Second, application of stratified sampling technique is proposed for VLSI system consisting of multiple modules.

For a module, the proposed RTL faults are assumed to have a distribution of detection probabilities similar to that for collapsed gate faults of a corresponding gate-level netlist. Under this assumption, RTL faults are a "representative sample" of any gate-level fault-set, and the difference between RTL and gate-level fault coverages of a module for a given set of test patterns is expected to be within the error bounds for the random fault-sampling technique [11], [12]. The effectiveness of this RTL module fault model is verified using several real-life industry-application VLSI circuits.

It is observed that the total number of RTL faults in a module does not represent the size of the gate-level fault-list. This lack of a clearly defined relationship between the number of RTL faults and the number of possible gate-level faults presents a problem for a VLSI system, which consists of several modules. Although the RTL fault-list of each module is a representative sample of the corresponding gate-level fault-list of that module, the overall RTL fault-list of the multimodule system does not constitute a representative sample of the overall gate-level fault-list. This observation led us to consider a technique known as *stratified sampling* [13]. A VLSI system is divided along RTL module boundaries into several nonoverlapping blocks we call *strata*. The stratum weights for these modules are determined using any of the proposed techniques described in a later section. RTL fault coverages of modules are added according their respective stratum weights to determine the stratified RTL fault coverage for the VLSI system. The stratified RTL fault coverage serves as an estimate of the gate-level fault coverage of the VLSI system for the given set of test patterns. The error bounds for this estimate are statistically calculated. The stratified RTL fault coverages of several real-life industry-application VLSI systems are compared with the corresponding gate-level fault coverages for various test pattern sets.

Sections II–IV describe the research contribution of this paper. Section II contains a detailed description of the proposed RTL fault model, the fault-injection algorithm and the RTL fault-simulation methodology. The relationship of the proposed RTL faults and the traditional single stuck-at gate faults is elaborated using an example. Section III presents the application of stratified sampling theory to RTL fault modeling. Section IV outlines the stratum weight extraction techniques. Section V describes the experimental work and results generated for several real-life industry-application VLSI systems. Section VI summarizes and outlines limitations of the proposed approach.

II. RTL FAULT MODEL, INJECTION ALGORITHM, AND SIMULATION

Hardware description language (HDL) constructs are classified into three types: structural, RTL, and behavioral [14]. RTL constructs represent a subset of HDL constructs, which, with additional design guidelines, ensure proper synthesis of a gate-level netlist by logic synthesis tools. With event scheduling and resource allocation information built in, an RTL model represents the microarchitecture of a circuit.

Some of the previous research in the area of high-level fault-modeling scopes its application to behavioral design modeling [3], [7] while other research aims at RTL design modeling [8], [10]. The research presented in this paper focuses on RTL design modeling. Although specific Verilog constructs [15] are used as a medium to explain the proposed RTL fault model, the concepts developed here can be applied to any other hardware description language. A few clarifications on the terminology used in this paper are given below:

> *Language Operators*: RTL language operators are classified into Boolean ($\&$, $|$, $^\wedge$, $\sim$), synthetic ($+$, $-$, *, $\geq$, $\leq$, $<$, $>$, $==$, $!=$), and logical ($\&\&$, $\|$, $!$) operators. A further classification of these operators, as used in another context, is unnecessary for the purpose of fault modeling.
> *Identifiers*: Identifiers are the names that one gives to the objects like wires, gates, and functions in the circuit. All identifiers that specify signal names will be referred to as "variables" in this paper.

A. RTL Fault Model and Injection Algorithm

An RTL fault model and a fault-injection algorithm are developed such that the RTL fault-list of a module becomes a representative sample of the collapsed gate-level fault-list. The classical definition of

the term "representative sample" in the context of statistical theory is given by Stephan and McCarthy [16] as:

A representative sample is a sample which, for a specified set of variables, resembles the population from which it is drawn to the extent that certain specified analyses that are to be carried out on the sample (computation of means, standard deviations, etc., for particular variables) will yield results which will fall within acceptable limits set about the corresponding population values, except that in a small proportion of such analyses of samples (as specified in the procedure used to obtain this one) the results will fall outside the limits.

In order for the RTL fault-list of a module to be a representative sample of the collapsed gate-level fault-list, RTL faults should have a distribution of detection probabilities similar to that for collapsed gate faults. The detection probability of a fault is defined by Seth *et al.* [17] as the "probability of detecting a fault by a randomly selected pattern." In other words, if a given test set contains n patterns and a fault is detected k times during fault simulation using the test set, the detection probability of the fault is given as k/n. When two fault-lists (one containing RTL faults and the other gate-level collapsed faults) with similar detection probability distributions are simulated for a given set of test patterns, the respective fault coverages are expected to track each other closely within statistical error bounds. Agrawal and Kato [11], [12], established error bounds for the random fault-sampling technique in which detection probability distributions for a random sample and that for the entire gate-fault population are expected to be similar. If the RTL fault-list of a module is indeed a representative sample of the collapsed gate-level fault-list, the difference between RTL and gate-level fault coverages of a module for a given set of test patterns should be within the established error bounds [11], [12]. This is verified by the experimental data given in a later section.

RTL design description dictates the microarchitecture of the gate-level representation. During logic synthesis, RTL operators map onto Boolean components of varying complexity, e.g., Boolean and logical RTL operators map onto Boolean gates, synthetic operators map onto components such as adders, comparators, etc. The RTL variables map onto signal lines in the gate-level netlist, though the relationship may not be a one-to-one mapping. The goal of the proposed fault model is to judiciously place RTL faults in the design description of a module. This is assured by mirroring properties of the gate-level single SSF model in the RTL fault model. These properties are listed below.

Properties of the gate level SSF model include the following.

- Boolean components are assumed to be fault-free.
- Signal lines contain faults:
  - a stuck-at-zero (s-a-0) fault when the logic level is fixed at value 0;
  - a stuck-at-one (s-a-1) fault when the logic level is fixed at value 1.
- According to the SSF assumption, only one fault is applied at a time when a test set is either being created or evaluated.
- The fault list is reduced using the concept of equivalence or dominance fault collapsing or the checkpoint theorem [12], [18]. The collapsed fault-list of a module contains input as well as fanout faults.

The following are properties of the RTL fault model.

- Language operators (which map onto Boolean components in the gate-level netlist) are assumed to be fault-free.
- Variables (which map onto signal lines in the gate-level netlist) contain faults:
  - an s-a-0 fault when the logic level is fixed at value 0;
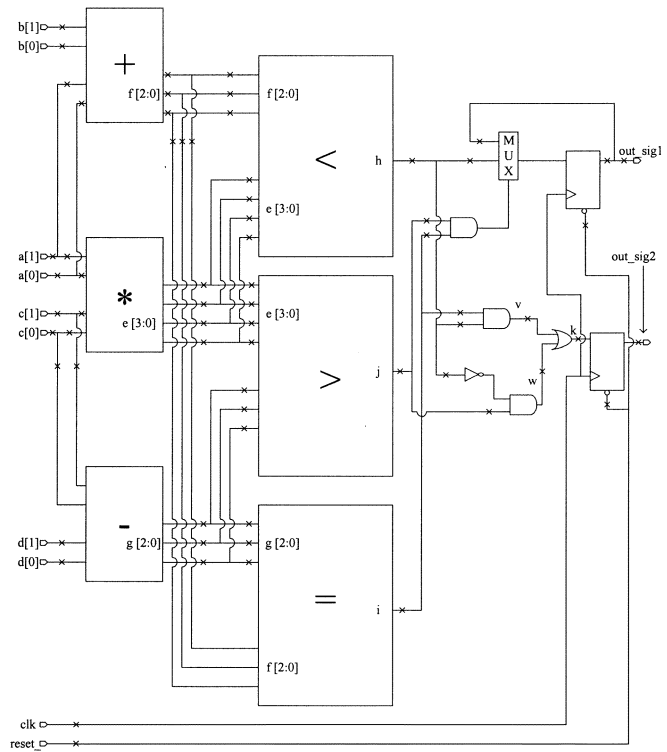 - an s-a-1 fault when the logic level is fixed at value 1.

Fig. 1. RTL faults in a schematic representation.

- The proposed RTL fault model follows the single-fault assumption and, therefore, only one fault is applied at a time when a test set is evaluated.
- The RTL fault-list of a module contains input as well as fanout faults. RTL variables appearing more than once in executable statements or instantiations of lower level modules in the design hierarchy are considered to have fanout. Input faults of an RTL module have a one-to-one equivalence to input faults of the module at the gate level. The fanout faults of variables inside an RTL module represent a subset of the fanout faults of a possible gate-level implementation.

The above definition of the RTL fault model and the fault-injection procedure encompasses modeling of faults for synthetic, Boolean and logical operators, sequential elements, and fanout or stem variables, as well as the collapsing of RTL faults. RTL faults are depicted with crosses ("x") in Fig. 1.

When RTL constructs contain synthetic operators, faults are injected only on the input variables of such operators. During logic synthesis, synthetic operators are replaced by combinational circuits implementing the respective functions, e.g., adder, subtracter, comparator, etc. Internal details of such functions are not available at the RT level and, therefore, only the subset of the checkpoint faults of the gate-level representation of these operators, namely, the primary input faults are modeled.

When a function is represented by RTL constructs containing Boolean operators, faults are injected on variables that form the Boolean equations. Some internal signals of these constructs are available at the RT level and, therefore, RTL faults are placed at primary inputs and internal nodes including signal stems and fanouts. The postsynthesis gate-level representation of such a construct may be structurally different from the RTL Boolean representation. However, some RTL faults have equivalent faults in a collapsed gate-level fault-list of any postsynthesis design.

When RTL constructs contain logical operators, faults are injected on variables that constitute inputs of such operators. Most often the postsynthesis gate-level implementation of a function described using logical operators maintains the structure implied in the RTL description. In such cases, RTL faults have a one-to-one equivalence to the collapsed gate faults of the synthesized logic.

Hardware description languages support two types of sequential elements, latch as well as flip-flop. In both cases, RTL faults are placed on input ports of these components. In the case of the flip-flop, faults are placed on clock as well as the reset variables.

In the gate-level SSF model, stem and fanout faults are unique since neither equivalence nor dominance relations exist between them. Stem and fanout faults are treated as special cases in the RTL fault model as well. A separate RTL fault is injected on each fanout of each bit of the variable. Also, a unique RTL fault is placed on each stem. In a VLSI system, several modules are interconnected. The interconnecting signals between modules have similar issues about stem and fanout-fault modeling. The properties described for stem and fanout-fault modeling for RTL constructs are applied for interconnecting signals between modules as well.

The fault-collapsing technique is widely used during gate-level fault simulation to reduce the size of the fault-list [18], [12]. Smaller fault-lists require lesser resources, and improve run-time performance. Although it is desirable, fault collapsing is not performed at the RT level since the structural information needed to analyze the equivalence of faults is missing. At the minimum, the proposed RTL fault model inherently avoids generating duplicate faults.

Although there are several subtle and critical differences between RTL fault model proposed here and previous approaches [8]–[10], we do not intend to claim novelty of RTL faults described here. The novelty of our technique lies in the coverage evaluation methods for the proposed RTL fault model and the associated statistical tolerance. As the RTL model gets coarser, the result remains correct though within larger statistical tolerance bounds. These ideas are analyzed in the following sections.

B. RTL Fault-Simulation Method

The RTL fault simulator accepts the fault-injected RTL circuit description and the test pattern set as inputs. The RTL fault-simulation method is analogous to the gate-level approach, in which good and faulty circuits are created based on the SSF assumption and simulated with a given set of test patterns. When responses of a good circuit and the faulty circuit do not match, the fault is considered detected. Fault simulation is continued until all faults are evaluated for the given set of test patterns. At the completion of fault simulation, a report is generated which contains statistics and other information on detected as well as undetected RTL faults. The RTL fault coverage of a module is defined as the ratio of the number of detected RTL faults to all RTL faults.

The RTL fault simulator used in this research is *Verifault-XL*. *Verifault-XL* is suitable for use as an RTL fault simulator due to its capability of propagating fault effects through RTL circuit description. RTL faults are identified to *Verifault-XL* as zero-delay buffers inserted between variables and executable statements as per the fault-injection algorithm. For more details on the method of RTL fault grading using *Verifault-XL*, one may refer to Mao and Gulati [8] and *Verifault-XL User's Guide* [19]. Several circuits (see Table I) were simulated using this method. M1 through M4 are RTL modules with a single level of hierarchy. These are studied in this section. D1, D2, and D3 are multimodule designs. The technique of this section will apply to their internal modules. VLSI systems that comprise of several modules, however, require the special technique of Section III and so D1, D2, and D3 will be discussed later.

TABLE I
DESIGN DATA OF MODULES AND VLSI SYSTEMS USED IN EXPERIMENTS

	PI/PO/Bidi$^\eta$	Code Size (Verilog lines)	Area (gates)	Number of Faults		
				RTL	Gate-level	
					Uncollapsed	Collapsed
M1	8/4/0	28	55	24	72	62
M2	37/10/0	214	428	390	1,192	651
M3	312/68/0	819	981	976	4,100	2,262
M4	122/190/0	648	3,168	1,490	7,002	5,500
D1	36/71/0	673	1,170	678	2,254	1,332
D2	15/7/16	9,000	17,126	24,982	45,688	29,670
D3	34/51/10	8,580	104,881	27,108	107,290	73,374

Note: η: Bidi refers to bi-directional I/Os

C. Study of Efficacy of RTL Fault Model and Fault-Injection Algorithm

In this section, the effectiveness of the RTL fault model and fault-injection algorithm in modeling hardware faults at the RT level is discussed.

1) Comparison of RTL and Gate Fault Lists: In practice, an RTL module contains several interconnected Boolean components described using various constructs. RTL faults are judiciously placed at input ports of the module, on input variables of the Boolean components, and on fanouts of interconnecting signals between Boolean components. The RTL faults may also be placed on fanouts of variables internal to the Boolean components if they are described using Boolean RTL operators. Therefore, an RTL fault-list of a module contains not just pin faults but also some internal faults. This is illustrated by the example in Figs. 1 and 2. Fig. 1 contains a schematic representation of a hardware function. Fig. 2 contains corresponding RTL description. The proposed RTL fault model and fault-injection procedure, when applied to the code, judiciously place faults in the RTL code. RTL faults are depicted with crosses ("x") in Fig. 1. As can be observed in Fig. 1, the RTL fault set consists of pin faults as well as internal faults of the module. Constraint-driven logic synthesis of the RTL code given in Fig. 2 may produce many different gate-level implementations. The gate-level implementations derive the microarchitecture from the RTL description. One of the gate-level netlists was arbitrarily selected to demonstrate the relationship of RTL faults to the gate faults. An analysis of the RTL and gate-level fault-lists reveals that individual RTL faults, when applied one at a time to the RTL design, produce behaviors that match the corresponding behaviors of faulty gate-level circuits resulting from individual stuck-at gate faults applied one at a time. Such RTL and gate-faults are considered equivalent. The RTL and collapsed gate-level fault-lists contain 67 and 174 faults, respectively. Upon comparing the RTL fault-list to the collapsed gate-level fault-list of the selected implementation, it is found that each RTL fault is equivalent to a unique gate-level fault. There are 107 gate-faults that do not have equivalent RTL fault. In this case, an RTL fault-list is viewed as a representative subset of the gate-level fault-list.

2) Comparison of RTL and Gate-Fault Coverage: If, indeed, the proposed RTL fault model possesses the statistical properties of a random sample of gate-level faults, the difference between the RTL and the gate-level fault coverages of a module for a given set of test patterns should be within the error bounds for the random fault-sampling technique. Agrawal and Kato [11], [12] give the range of coverage for the random sampling technique as

$$c \pm \frac{\alpha^2 k}{2N} \sqrt{1 + \frac{4Nc(1-c)}{(\alpha^2 k)}} \qquad (1)$$

where $k = 1 - N/M$, when N faults are sampled from a total of M gate-level faults in the circuit, $\alpha = 3.00$ for the three-sigma confi-

```
// Module Name: Miscellaneous Function

module misc_func(out_sig1, out_sig2, a, b, c, d, clk, reset_);

// i/o declarations

input[1:0] a, b, c, d;
input clk, reset_;

output out_sig1, out_sig2;

// Data Type Declarations

wire[3:0] e;
wire[2:0] f;
wire[2:0] g;
wire[1:0] a, b, c, d;
wire v, w, k;

reg out_sig1, out_sig2;
reg h, i, j;

// Functionality Implementation

assign e[3:0] = a[1:0] * c[1:0];
assign f[2:0] = a[1:0] + b[1:0];
assign g[2:0] = c[1:0] - d[1:0];


always @(e or f)
if(e < f)
  h = 1'b1;
else
  h = 1'b0;

always @(f or g)
if(f == g)
  i = 1'b1;
else
  i = 1'b0;

always @(e or g)
if(e > g)
  j = 1'b1;
else
  j = 1'b0;

always @(posedge clk or negedge reset_)
if(!reset_)
  out_sig1 <= 1'b0;
else
  out_sig1 <= (i & j) ? h : out_sig1;

assign v = i & h;
assign w = (!h) & j;
assign k = v | w;

always @(posedge clk or negedge reset_)
if(!reset_)
  out_sig2 <= 1'b0;
else
  out_sig2 <= k;

endmodule
```

Fig. 2. RTL code example.

TABLE II
MEASURED FAULT COVERAGE DATA FOR MODULES

Module	Test	c_m (%) RTL Fault Coverage	C_m (%) Gate-level Coverage	c_m-C_m (%)	Estimation Error (%) (3σ bound) Eq. (1)
M1	T1	20.8	25.3	− 4.5	± 22.6
	T2	58.3	75.8	− 17.5	± 26.3
	T3	79.2	86.3	− 7.1	± 22.6
	T4	91.7	92.9	− 1.2	± 17.5
	T5	95.8	96.8	− 1.0	± 14.9
M2	T1	5.4	8.9	− 3.5	± 2.2
	T2	25.1	28.0	− 2.9	± 4.2
	T3	49.5	54.1	− 4.6	± 4.8
	T4	77.9	80.8	− 2.9	± 4.0
	T5	82.6	83.6	− 1.0	± 3.7
M3	T1	19.4	19.0	+ 0.4	± 2.9
	T2	42.0	42.8	− 0.8	± 3.6
	T3	52.9	54.6	− 1.7	± 3.6
	T4	65.7	67.2	− 1.5	± 3.4
	T5	75.8	78.0	− 2.2	± 3.0
	T6	96.6	95.5	+ 1.1	± 1.3
M4	T1	16.2	15.4	+ 0.8	± 2.5
	T2	24.2	23.9	+ 0.3	± 2.9
	T3	47.9	44.5	+ 3.4	± 3.3
	T4	70.7	68.9	+ 1.8	± 3.0
	T5	84.1	81.6	+ 2.5	± 2.4
	T6	87.3	84.5	+ 2.8	± 2.2

Legend:
c_m = RTL Coverage, C_m = Gate Coverage, c_m-C_m = Error

dence probability of 0.998, and c is the ratio of detected to total faults among the sampled faults. When (1) is used for error bounds of the gate-level fault coverage estimated by the proposed RTL fault-modeling technique, N represents the number of RTL faults in a module, M represents the number of gate faults in the module, and c represents the ratio of the number of detected RTL faults to all RTL faults.

RTL and gate-level fault coverages of several real-life industry-application circuits are compared. An RTL fault-list is created for each RTL module using the proposed fault model and fault-injection procedure. RTL modules are then synthesized using the commercial logic synthesis tool Design Compiler [20] and a 0.35-μm CMOS technology library. A gate-level implementation is arbitrarily selected to measure the gate-level fault coverage of the given test patterns. RTL and gate-

level circuits are simulated using the fault simulator *Verifault-XL*. Test pattern sets were written for circuits using their functional specifications. The error between RTL and gate-level fault coverage is expected to be within $\pm 3\sigma$ range with a confidence probability of 99.8% as per (1).

All circuits used for experiments, M1-Counter (four-bit) module, M2-Transmit Buffer module, M3-SDRAM Controller module, M4-DSP Interface module, D1-Frame Timing Control ASIC, D2-System Timing Controller ASIC, and D3-Digital Signal Processor (DSP) ASIC, contain sequential as well as combinational logic. Design data for these circuits (PI = number of primary input signals, PO = number of primary output signals, Bidi = number of bidiectional signals, Code Size = number of lines of Verilog HDL code, and Area = gate area measured as number of two-input NAND CMOS equivalent gates) are provided in Table I. As we pointed out earlier, D1, D2, and D3 are multimodule systems that require a different technique to be discussed in Section III. Only the single module circuits M1–M4 are discussed in this section.

In Table II, for all except four data-points, RTL coverage is found to track the gate-level coverage within statistical error bounds. These four cases are being investigated and may contribute toward refinement of the proposed fault model. Figs. 3 and 4 show correlation between RTL and gate-level fault coverage across 26 data points for Module M3. Similar results are reported for other circuits (M1, M2, and M4) in [21].

From the experimental data (see Table II), we conclude that the RTL fault coverage is a good estimate of the gate-level fault coverage for medium to large size modules. It is observed that for very small modules such as module M1 (four-bit counter), the error between the RTL and the gate-level fault coverages, though within statistically calculated error bounds, is large (more than 5%). This is due to the small sample size in the random-sampling technique [22]. However, modern VLSI
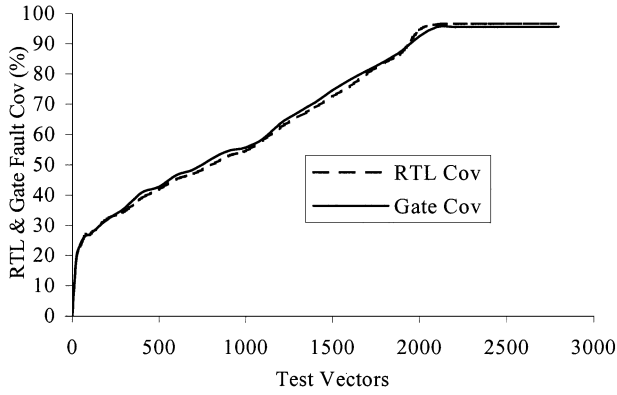
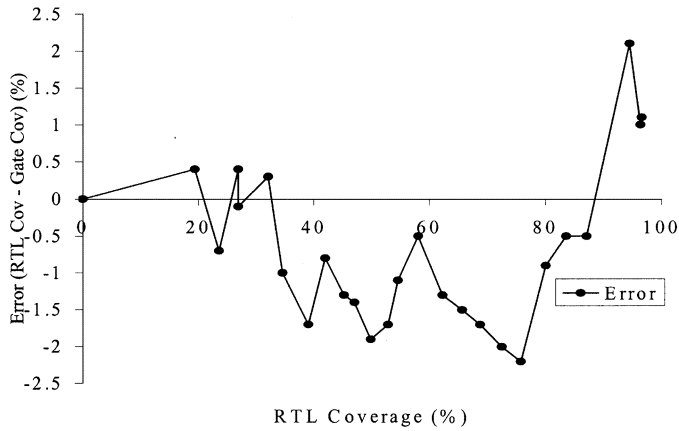Fig. 3.   Module M3: RTL and gate coverages versus test vectors.

Fig. 4.   Module M3: Error versus RTL fault coverage.

TABLE III
INACCURATE ESTIMATION OF GATE FAULT COVERAGE

Modeling Level	m1 Faults		m2 Faults		m1+m2
	Total	Covered	Total	Covered	Coverage
RTL	100	91%	100	39%	65%
Gate	150	90%	400	40%	54%

overall gate-level fault-list of that VLSI system. Similarly, a module that contributes fewer faults to the RTL fault-list of the VLSI system may result in a relatively large gate-level implementation after logic synthesis, and thus, contribute a larger percentage of faults to the overall gate-level fault-list of the system. Therefore, although the RTL fault-list of each module in a VLSI system is a representative sample of the corresponding gate-level fault-list of that module, the overall RTL fault-list of the system does not constitute a representative sample of the overall gate-level fault-list. Table III illustrates this problem for a hypothetical VLSI system consisting of two modules, m1 and m2. Based on the observation of the experiments in the previous section, it is assumed that the measured RTL fault coverage is close to the gate-level fault coverage in each module. The overall RTL fault coverage of the system is obtained as $(91 \times 100 + 39 \times 100)/200 = 65\%$. The overall gate-level fault coverage of the system is calculated as $(90 \times 150 + 40 \times 400)/550 = 54\%$. It is found that the overall coverages are quite different. This is because the two modules, although equal in size at the RT level, translate into quite different sizes at the gate-level. In order to find the gate-level fault coverage, modules' RTL coverages should be weighted according to their relative gate-level sizes. Weighted RTL fault coverage can be obtained as $91 \times (150/550) + 39 \times (400/550) = 53\%$. The above observation leads us to consider a technique known as *stratified sampling* [23].

According to the stratified-sampling technique, the fault population is divided along RTL module boundaries. Thus, each module is considered a stratum. Within a stratum, the RTL faults are considered as a sample of all (i.e., gate-level) faults. The ratio of the number of RTL faults to the number of gate-level faults will be the sampling fraction for the stratum. In general, if the number of RTL faults in a module is larger, then a random sample of those can also be used. Suppose, a VLSI system has G gate-level faults distributed among M modules (or strata). The mth module has G_m gate-level faults. Then

$$G = \sum_{m=1}^{M} G_m$$

and

$$\text{Weight of } m\text{th module}, \quad W_m = \frac{G_m}{G}. \tag{3}$$

From the mth module, r_m RTL faults are simulated. These can either be all RTL faults (100% sample) or a random sample of all RTL faults in the mth module.

Further, for a given set of test patterns two coverages for the mth module are defined as

$$C_m = \frac{\text{Detected gate faults in } m\text{th module}}{G_m} \tag{4}$$

$$c_m = \frac{1}{r_m} \sum_{i=1}^{r_m} y_{mi} \tag{5}$$

where y_{mi} are random variables whose values are determined by fault simulation: $y_{mi} = 1$ if the ith sampled fault in mth module is detected,

III. APPLICATION OF STRATIFIED SAMPLING

systems contain many modules of a variety of sizes and large estimation errors in a few small modules are insignificant while calculating the overall fault coverage.

The main conclusion of this section is that the proposed RTL fault model can be used to estimate the gate-level fault coverage of a module within statistical error bounds.

III. APPLICATION OF STRATIFIED SAMPLING

The results of Section II show that the RTL fault coverage provides a statistical estimate for the gate-level fault coverage. The experimental data also reveal that there is no straightforward relationship between the number of RTL faults and that of gate-level faults. Considering the fault-modeling method, the number of RTL faults is a measure of the size of the RTL description of the module. However, this number does not correlate with the gate count. For example (see Table I), for module M1, the numbers of gates and gate-faults are more than twice the number of RTL faults. For modules M2 and M3, the gate count is closer to the number of RTL faults, but there are almost twice as many gate-level faults. The main conclusion of these experiments is that the proposed RTL fault model can be used to estimate the gate-level fault coverage of a module. But, the total number of RTL faults in the module does not represent the size of the gate-level fault-list.

In general, a large VLSI system consists of many modules. The lack of a definite relationship between the number of RTL faults and the number of possible gate-level faults presents a problem. A module with a large contribution of RTL faults to the overall RTL fault-list of the VLSI system may get synthesized into a relatively small gate-level implementation and consequently make a smaller contribution to the

or $y_{mi} = 0$ if that fault is not detected. According to Section II, the RTL coverage c_m is an unbiased estimate for the gate-level fault coverage C_m of module m [21], [24]. C_m, being the true gate-level fault coverage of the mth module, provides the total gate-level coverage of the multimodule VLSI system as

$$C = \frac{\sum_{m=1}^{M} G_m C_m}{G} = \sum_{m=1}^{M} W_m C_m. \qquad (6)$$

Here, module coverages have been weighted according to their sizes to eliminate the type of error illustrated in Table III. The estimated value for the true gate-level coverage C of the VLSI system is called stratified RTL fault coverage and is obtained as

$$\overline{C} = \frac{\sum_{m=1}^{M} G_m c_m}{G} = \sum_{m=1}^{M} W_m c_m. \qquad (7)$$

Notice that the stratified RTL fault coverage $\overline{C}$ is different from the overall nonstratified RTL fault coverage, which is given by

$$C_{\text{RTL}} = \frac{\sum_{m=1}^{M} r_m c_m}{\sum_{m=1}^{M} r_m}. \qquad (8)$$

The discussion and empirical data provided in Section II show that the RTL fault coverage of a module is a close estimate of the gate-level coverage of that module. That is, given a vector set, almost identical fractions of the two types of faults are detected, or a randomly selected RTL fault has the same probability of being detected as a gate-level fault. By taking the statistical expectation of (7), it can be shown that $\overline{C}$ is an unbiased estimate of the true coverage C, or

$$E(\overline{C}) = \sum_{m=1}^{M} W_m E(c_m) = \sum_{m=1}^{M} W_m C_m = C. \qquad (9)$$

The variance of the estimate is evaluated as

$$\sigma^2 = \sigma^2(\overline{C}) = E(\overline{C} - C)^2. \qquad (10)$$

On simplification [13], this leads to

$$\sigma^2 = \frac{1}{G^2} \sum_{m=1}^{M} \frac{G_m(G_m - r_m)\sigma_m^2}{r_m} \qquad (11)$$

where σ_m^2 is the unbiased estimate for the variance of y_{mi}, given by

$$\sigma_m^2 = \frac{1}{r_m - 1} \sum_{i=1}^{r_m} (y_{mi} - c_m)^2. \qquad (12)$$

In (12), $y_{mi} = 1$ for $c_m r_m$ detected faults and $y_{mi} = 0$ for $(1 - c_m)r_m$ undetected faults. Therefore,

$$\sigma_m^2 = \frac{1}{r_m - 1}\left(c_m r_m (1 - c_m)^2 + (1 - c_m)r_m c_m^2\right)$$
$$= \frac{r_m c_m (1 - c_m)}{r_m - 1}. \qquad (13)$$

Equation (11) can be expanded as

$$\sigma^2 = \sum_{m=1}^{M} \left(\frac{G_m^2}{G^2} - \frac{G_m r_m}{G^2}\right)\frac{\sigma_m^2}{r_m}$$
$$= \sum_{m=1}^{M} \left(W_m^2 - \frac{W_m r_m}{G}\right)\frac{\sigma_m^2}{r_m}. \qquad (14)$$

Since an RTL module description is significantly more compact compared with the gate-level description, $r_m \ll G_m$. Therefore, simplifying (14) based on the approximation, we obtain

$$\sigma^2 = \sum_{m=1}^{M} \frac{W_m^2 \sigma_m^2}{r_m}. \qquad (15)$$

Substituting (13) into (15), the variance of stratified RTL fault coverage is given as

$$\sigma^2 = \sum_{m=1}^{M} \frac{W_m^2}{r_m - 1} c_m(1 - c_m). \qquad (16)$$

For a given confidence probability, the range of coverage is given as [21]

$$\overline{C} \pm t\sigma \qquad (17)$$

where t is the limit for which the area of the normalized (zero mean and unity variance) Gaussian probability density equals some given confidence probability. The values of t can be selected from tables of normal distribution, or taken as three for the popular *three-sigma estimate*.

It is evident from (7), (16), and (17) that the relevant parameters, which determine stratified RTL fault coverage and error bounds, do not require the knowledge of the absolute values of G_m or G. They require the ratio of the two quantities in the form of stratum weights. Techniques for determining the stratum weights of modules in a given VLSI system are outlined next.

IV. STRATUM WEIGHT-EXTRACTION TECHNIQUES

The accurate stratum weights of the modules of a VLSI system can be obtained only after final logic synthesis. However, since the goal of the proposed RTL fault model and test evaluation technique is to enable an accurate estimation of the gate-level fault coverage early in the design cycle, alternative approaches are proposed here.

- Preliminary (early) logic synthesis based weight extraction.
- Entropy-measure-based weight extraction [25], [26].
- Use of early area (gate-count) estimates (using built-in RTL-synthesis technology in formal verification tools such as Verplex [27]) to determine weights. It is assumed here that the size of the gate-level fault-list of each module is proportional to the area (gate-count) of the corresponding module and thus stratum weight W_m can be approximated to be the ratio of A_m and A instead of G_m and G, where $A_m = \text{area}$ (gate-count) of the module, and $A = \text{total area}$ (gate-count) of the VLSI system.

These techniques allow one to estimate stratum weights of modules during early phases of the design cycle without depending upon generation of the final gate-level netlist. The weights obtained by these techniques may be different from those obtained from the final gate-level netlist. In the analysis presented in Section III, error resulting from such estimation is ignored with the assumption that even though the absolute values of G_m and G may vary significantly for different gate-level netlists, their ratio represented as stratum weights does not vary much due to large denominator G. The impact of difference between the estimated and actual stratum weights on the stratified RTL fault coverage and error bounds is assumed to be insignificant. These assumptions are verified in Section V.

V. EXPERIMENTAL RESULTS

A. Estimation of the Gate-Level Fault Coverage of a VLSI System

Several industry-application VLSI systems (see design data in Table I) ranging in size from 1 000 to 105 000 gates were used for empirical studies. The experimental procedure is as follows.

- A C++ parser processes the RTL code and generates fault-injected RTL code without altering the circuit behavior. The parser implements the fault-injection procedure for the proposed fault model.
- From each module of the VLSI system, a set of RTL faults (all or a random subset) is selected. These faults are simulated for the given set of test patterns and module coverages are determined.
- Stratum weights for modules are computed using logic-synthesis-based weight-extraction technique.
- Stratified RTL fault coverage is computed using (7). The stratified RTL fault coverage serves as an estimate of the gate-level fault coverage of the VLSI system. The range of the estimated coverage is obtained using (17).

The experimental procedure described above was carried out using commercial electronic design-automation tools. The RTL fault simulation was performed using the *Verifault-XL* simulator. The estimates of the gate-level fault coverages were compared with the nonstratified RTL fault coverages calculated using (8) as well as the actual gate-level fault coverages. The actual gate-level fault coverage of each VLSI system was obtained by fault grading the gate-level netlist. Gate-level fault simulation, also preformed by *Verifault-XL*, used gate-level netlist for the VLSI system obtained by logic synthesis of the RTL code for an arbitrary set of optimization constraints. *Design Compiler* was used for logic synthesis [20]. Test patterns (test-benches) used for the RTL and gate-level fault simulations were manually generated for design verification using functional specifications of the systems.

Experimental data provided in Table IV indicates that the stratified RTL fault coverage is a good estimate of the gate-level fault coverage. In all except four cases, the true error is within statistical error bounds determined from (16) and (17). As can be noted in data for system D3 (Table IV), the error bounds for testbenches T1, T5, T6, and T7 are significantly wider than those for T2, T3, and T4. The RTL fault simulation for testbenches T1, T5, T6, and T7 was performed using a very small random sample of the overall RTL fault-list. This was done to maintain reasonable CPU time for all test sets given that T1, T5, T6, and T7 are significantly larger than T2, T3, and T4. As per (16), the size of the RTL fault sample in each module and the error bounds of the overall stratified RTL coverage of the system are related. The smaller the number of RTL faults used in simulation, the wider the error bounds. This is confirmed by the data presented in Table IV. Therefore, in order to estimate the gate-level fault coverage within narrow error bounds, a reasonable number of RTL faults should be selected during simulations. Figs. 5 and 6 show correlation between stratified RTL and gate-level fault coverage across several data points for Design D3.

B. Stratum Weight Estimation

The goal of the proposed RTL fault model and coverage evaluation technique is to enable early estimation of gate-level coverage by running RTL fault simulations. Some of the key parameters needed to determine stratified RTL fault coverage are stratum weights of all modules that constitute a VLSI system. The actual stratum weights can be determined only after final logic synthesis is completed and final gate-level netlist is available. However, use of final gate-level netlist defeats the purpose of the technique presented here. Thus, for an early estimation of stratum weights (without using final netlist), several alternative approaches are presented in Section IV with an assumption that the impact of difference between the estimated (using proposed tech-

niques) and actual stratum weights on the stratified RTL fault coverage and error bounds is insignificant. This assumption is verified here with experimental data.

For the VLSI system D3, several unique gate-level netlists were obtained by logic synthesis. Each netlist is generated using a different set of optimization constraints, e.g., area optimization (netlist 1), speed optimization (netlist 2), and combined area and speed optimizations (netlists 3 and 4). Gate-level fault-lists were generated from each netlist and several sets of stratum weights were extracted (one from each netlist). A set of stratum weights was also extracted using area-estimation technique with a formal verification tool. Fig. 7 indicates the data points (number of gate-faults and area) for each module in D3 (total 12 modules constitute Design D3) for each of

TABLE IV
FAULT COVERAGE DATA FOR VLSI SYSTEMS

	Test	C_{RTL} (%) (Eq. 8)	$\overline{C}$ (%) (Eq. 7)	C (%) (Eq. 6)	$\overline{C} - C$ (%)	Estimation Error (%) (3σ Bound)
D1	T1	24.3	23.8	25.6	− 1.8	± 5.1
	T2	35.8	27.7	29.3	− 1.6	± 4.6
	T3	44.1	32.9	35.4	− 2.5	± 4.5
	T4	42.6	32.2	35.9	− 3.7	± 4.6
	T5	43.4	35.3	38.9	− 3.6	± 5.2
	T6	87.3	81.9	82.0	− 0.1	± 4.8
	T7	84.8	82.0	82.7	− 0.7	± 4.8
D2	T1	48.9	52.6	51.7	+ 0.9	± 0.9
	T2	39.7	42.6	43.9	− 1.3	± 1.0
	T3	29.0	30.6	31.5	− 0.9	± 1.0
	T4	55.2	59.5	60.2	− 0.7	± 0.9
	T5	52.3	55.9	54.7	+ 1.2	± 0.9
	T6	56.8	60.9	61.5	− 0.6	± 0.9
D3	T1	56.4	55.1	57.7	− 2.6	± 8.0
	T2	23.9	18.0	19.4	− 1.4	± 1.7
	T3	57.7	59.2	61.4	− 2.2	± 2.4
	T4	48.7	50.2	50.9	− 0.7	± 2.1
	T5	59.4	57.2	57.7	− 0.5	± 7.9
	T6	68.8	69.8	70.9	− 1.1	± 9.3
	T7	73.3	74.9	72.7	+ 2.2	± 8.8

Legend:

C_{RTL}=Non-stratified RTL coverage, $\overline{C}$ = Stratified RTL Coverage
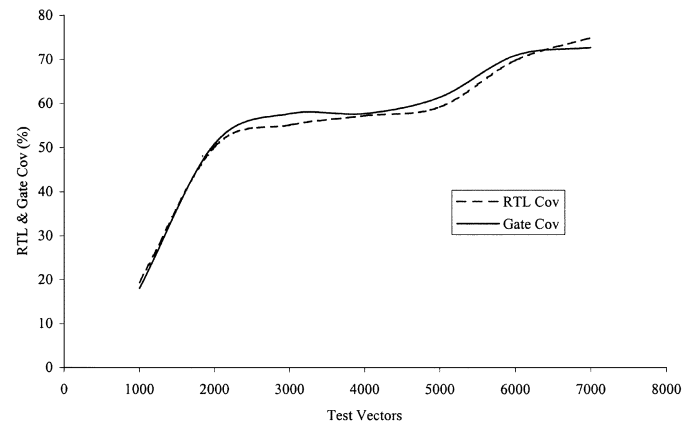
C = Gate-level fault coverage

Fig. 5.   Design D3: Stratified RTL and gate-coverages versus test vectors.
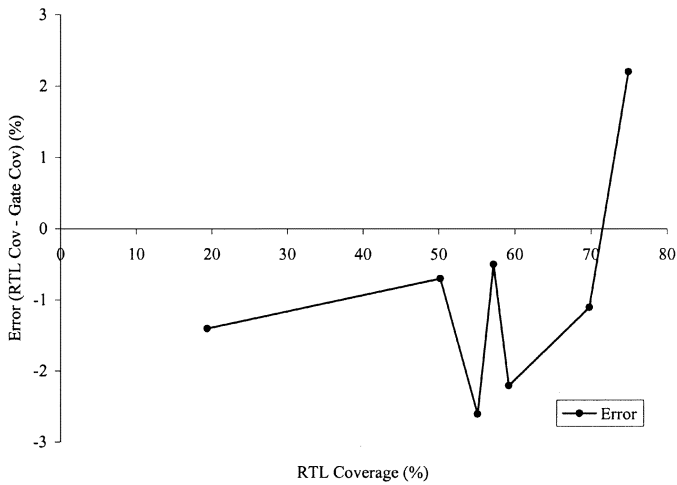
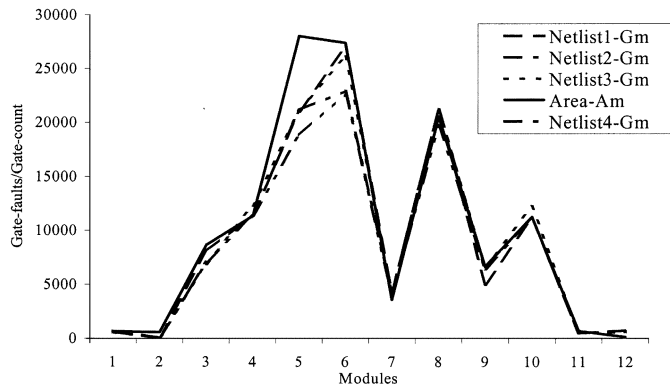Fig. 6.   Design D3: Error versus fault coverage.

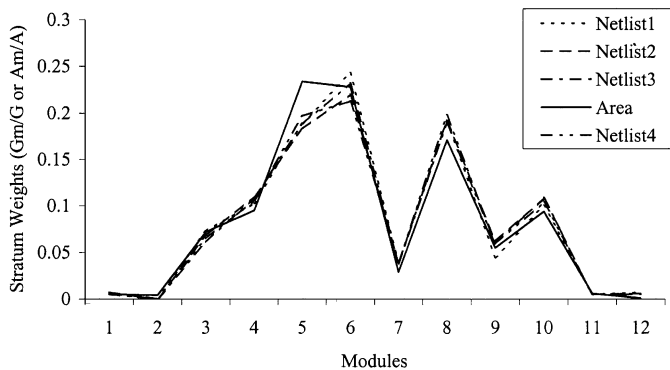Fig. 7.   Design D3: Gate-count and area distribution.

Fig. 8.   Design D3: Stratum weight distribution.

these approaches. Fig. 8 indicates the distribution of stratum weights across all modules for each set. Figs. 9 and 10 show stratified RTL fault coverage and error bounds determined using stratum weights from each of these approaches.

It can be observed from the figures here, that the absolute values of G_m and G may vary significantly for different gate-level netlists. However, their ratios represented as stratum weights W_m do not vary as much. Also, the impact of small variations in stratum weights on the stratified RTL fault coverage and error bounds is very small. These experimental results validate the use of early/preliminary netlist or area-estimation-based techniques for stratum weight extraction. For more information, please refer to [21].
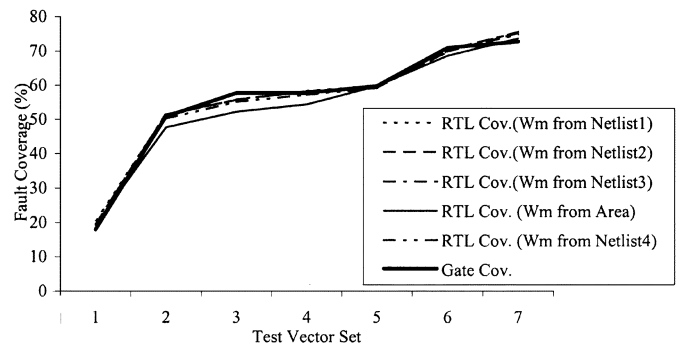

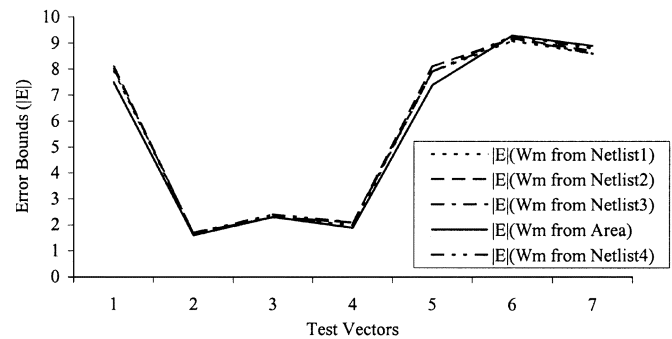Fig. 9.   Design D3: Stratified RTL fault coverage.

Fig. 10.   Design D3: Error bounds.

VI. CONCLUSION

In this paper, a novel procedure that supports RTL fault simulation and generates an estimate of the gate-level fault coverage for a given set of test patterns is proposed along with experimental results from several real-life industry-application VLSI systems. The proposed procedure can be used as an integral part of the high-level test generation systems [6], [10], [28]–[30]. The results of test-evaluation using the proposed procedure can provide feedback for further improving the quality of test patterns.

As established by Thaker *et al.* [31] with empirical data, the architectural testability properties and the subsequent test weaknesses of a gate-level netlist are derived from the RTL description and remain unchanged by the constraint-driven logic synthesis. The RTL fault simulation using the proposed fault model provides means to identify testability problems at the RT level prior to logic synthesis. The circuits that do not attain high fault coverage even with a large number of test patterns indicate test-related flaws inherent in the design architecture. The undetected RTL faults indicate hard-to-test functional areas of the design. The identification of these test problems early in the design cycle prompts necessary architectural changes prior to logic synthesis, reducing the impact on time-to-market.

Existing fault-simulation techniques, which are based on gate-level SSF models, require a large memory and a lot of CPU time. The RTL fault-simulation approach presented in this paper holds promise for significantly reducing the performance penalties of the gate-level fault-simulation approach. The experiments conducted to compare the speed of RTL and gate-level fault simulations for an equal number of faults, show that the RTL fault simulations runs 2 to 6 times faster than the gate-level fault simulation. However, the commercial fault simulator used for these experiments is designed for optimum performance at the gate level, while supporting RTL fault simulation as only a secondary feature. A true comparison of RTL and gate-level fault-simulation performances can be obtained if fault simulators

optimized for RTL as well as gate-level fault models were used. Currently, such option is not available. In future, a fault simulator designed to work optimally with the proposed RTL fault model may offer larger performance improvements.

The limitation of the proposed procedure is that it requires one to preserve the partitioning of the VLSI systems across RTL module-boundaries in order to be able to use stratum weights reliably. Some of the advanced optimization techniques recommend flattening of the RTL structure/partitions during logic synthesis to maximize the area reduction. This and other similar logic synthesis/optimization techniques that require removal or restructuring of design partitions built into the RTL description as modules, should be avoided if the proposed procedure is to be effectively used.

REFERENCES

[1] S. M. Thatte and J. A. Abraham, "Test generation for microprocessors," *IEEE Trans. Comput.*, vol. C-29, pp. 429–441, June 1980.
[2] S. Ghosh and T. J. Chakraborty, "On behavior fault-modeling for combinational digital designs," *J. Electron. Testing: Theory Applicat.*, vol. 2, no. 2, pp. 135–151, 1991.
[3] S. Ghosh, "Behavioral-level fault simulation," *IEEE Design & Test Comput.*, vol. 5, pp. 31–42, June 1988.
[4] P. C. Ward and J. R. Armstrong, "Behavioral fault simulation in VHDL," in *Proc. 27th Design Automation Conf.*, June 1990, pp. 587–593.
[5] J. R. Armstrong, F. S. Lam, and P. C. Ward, "Test generation and fault simulation for behavioral models," in *Performance and Fault Modeling with VHDL*. Englewood Cliffs, NJ: Prentice-Hall, 1992, pp. 240–303.
[6] C. H. Cho and J. R. Armstrong, "B-algorithm: A behavioral test generation algorithm," in *Proc. Int. Test Conf.*, Oct. 1994, pp. 968–979.
[7] P. Sanchez and I. Hidalgo, "System fault simulation," in *Proc. Int. Test Conf.*, Oct. 1996, pp. 732–740.
[8] W. Mao and R. Gulati, "Improving gate level fault coverage by RTL fault grading," in *Proc. Int. Test Conf.*, Oct. 1996, pp. 150–159.
[9] F. Ferrandi, F. Fummi, and D. Sciuto, "Test generation and testability alternatives exploration of critical algorithms for embedded applications," *IEEE Trans. Comput.*, vol. 51, pp. 200–215, Feb. 2002.
[10] R. J. Hayne and B. W. Johnson, "Behavioral fault modeling in a VHDL synthesis environment," in *Proc. 17th VLSI Test Symp.*, Apr. 1999, pp. 333–340.
[11] V. D. Agrawal and H. Kato, "Fault sampling revisited," *IEEE Design & Test of Comput.*, vol. 7, pp. 32–35, Aug. 1990.
[12] M. L. Bushnell and V. D. Agrawal, *Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits*. Norwell, MA: Kluwer, 2000, pp. 78–124.
[13] W. G. Cochran, *Sampling Techniques*. New York: Wiley, 1977, pp. 10–47.
[14] E. Sternheim, R. Singh, R. Madhavan, and Y. Trivedi, *Digital Design and Synthesis with Verilog HDL*. San Jose, CA: Automata Pub., 1993, pp. 14–63.
[15] P. A. Thaker, V. D. Agrawal, and M. E. Zaghloul, "Validation vector grade (VVG): A new coverage metric for validation and test," in *Proc. 17th IEEE VLSI Test Symp.*, Apr. 1999, pp. 182–188.
[16] F. Stephan and P. McCarthy, *Sampling Opinions: An Analysis of Survey Procedure*. New York: Wiley, 1958, pp. 30–78.
[17] S. C. Seth, V. D. Agrawal, and H. Farhat, "A statistical theory of digital circuit," *IEEE Trans. Comput.*, vol. 39, pp. 582–586, Apr. 1990.
[18] M. Abramovici, M. A. Breuer, and A. D. Friedman, *Digital Systems Testing and Testable Design*. New York: IEEE Press, 1990, pp. 116–117.
[19] *Verifault-XL™ User's Guide*. San Jose, CA: Cadence Design Systems, Inc., 1997.
[20] H. Bhatnagar, *Advanced ASIC Chip Synthesis*. Norwell, MA: Kluwer, 1999, pp. 105–150.
[21] P. A. Thaker, "Register-Transfer Level Fault Modeling and Test Evaluation Technique," Ph.D disertation, Dept. of Elect. and Comput. Eng., George Washington Univ., Washington, DC, 2000.
[22] V. D. Agrawal, "Sampling techniques for determining fault coverage in LSI circuits," *J. Digital Syst.*, vol. 5, no. 3, pp. 189–202, 1981.
[23] A. Stuart, *The Ideas of Sampling*. High Wycombe, U.K.: Griffin and Co., 1984, pp. 149–155.
[24] P. A. Thaker, V. D. Agrawal, and M. E. Zaghloul, "Register-transfer level fault modeling and test evaluation techniques for VLSI circuits," in *Proc. Int. Test Conf.*, Oct. 2000, pp. 940–949.
[25] K. T. Cheng and V. D. Agrawal, "An entropy measure for the complexity of multi-output boolean functions," in *Proc. 27th Design Automation Conf.*, June 1990, pp. 302–305.
[26] N. Pippenger, "Information theory and the complexity of boolean functions," *Math. Syst. Theory*, vol. 10, pp. 129–167, 1977.
[27] *Verplex Formal Verification: Logic Equivalency Checker User's Guide*, Malpitas, CA, 2002.
[28] S. Chiusano, F. Corno, and P. Prinetto, "RT-level TPG exploiting high-level synthesis information," in *Proc. 17th IEEE VLSI Test Symp.*, Apr. 1999, pp. 341–346.
[29] F. Corno, P. Prinetto, and M. S. Reorda, "Testability analysis and ATPG on behavioral RT-level VHDL," in *Proc. Int. Test Conf.*, Nov. 1997, pp. 753–759.
[30] E. M. Rudnick, R. Vietti, A. Ellis, F. Corno, P. Prinetto, and M. S. Reorda, "Fast sequential circuit test generation using high-level and gate-level techniques," *Proc. IEEE Eur. Design Automation and Test Conf.*, pp. 570–576, Feb. 1998.
[31] P. A. Thaker, M. E. Zaghloul, and M. B. Amin, "Study of correlation of testability aspects of RTL description and resulting structural implementations," in *Proc. 12th Int. Conf. VLSI Design*, Jan. 1999, pp. 256–259.