International Journal of Advanced Technology and Innovative Research

www.ijatir.org

ISSN 2348–2370
Vol.09,Issue.05,
April-2017,
Pages:0661-0669

Register-Transfer Level Fault Modeling and Test Evaluation Techniques for VLSI Circuits

LEELA SANTHOSHI KURUMADDALI
Faculity, Dept of ECE, Padmavathi Mahila Vishwa Vidhyalayam, Tirupati, AP, India,
E-mail: kleelasanthoshi@gmail.com.

Abstract: Stratified fault sampling is used in RTL fault simulation to estimate the gate-level fault coverage of given test patterns. RTL fault modeling and fault injection algorithm are developed such that the RTL fault list of a module can be treated as a representative fault sample of the collapsed 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 VLSI system, consisting of several modules, the overall coverage is a weighted sum of RTL module coverages. Several techniques are proposed to determine these weights, known as stratum weights. For a system timing controller ASIC, the stratified RTL coverage of verification testbenches was estimated within 0.6% of the actual gate-level coverage. This ASIC consists of 40 modules (9,000 lines of Verilog HDL) that are synthesized into 17,126 equivalent logic gates by a commercial synthesis tool. Similar results on two other VLSI systems are reported.

Keywords: RTL, Verilog HDL, VLSI.

I. INTRODUCTION

Test patterns for large VLSI systems are often determined 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 Thatte and Abraham [23], Chakraborty and Ghosh [7], Ghosh [13], Ward and Armstrong [22], Armstrong et al. [4], Cho and Armstrong [10], and, Sanchez and Hidalgo [18]. None of these techniques establish the relationship between high-level fault coverage and gate-level fault coverage. Mao and Gulati [15] 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 indicated as much as a 10% error between the actual gate-level fault coverage and the RTL fault coverage. Hayne and Johnson [14] developed a fault model based on finding an abstraction of the industry standard single-stuckline 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 RT level 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 RT level since the RT level description is a higher level of abstraction and may not contain the low-level structural information needed to exactly replicate all gate level failures. Also, since the gate-level net list changes drastically with all logic synthesis iteration that efforts to model all possible gate faults at RT level 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 SSF model and mapping of RTL constructs onto gate level structures during logic synthesis.

The proposed RTL faults of a module have a distribution of detection probabilities similar to that for collapsed gate faults of a corresponding gate-level net list. 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 established for the random-sampling technique by Agrawal and Kato [3]. The effectiveness of the proposed RTL 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 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. This observation led us to consider a technique known as stratified sampling [11]. A VLSI system is divided into several non-overlapping strata according to RTL module boundaries. 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 2, 3 and 4 describe the research contribution of this paper. Section 2 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 3 presents the application of stratified sampling theory to RTL fault modeling. Section 4 outlines the stratum weight extraction techniques. Section 5 describes the experimental work and results generated for several real-life industry-application VLSI systems. Section 6 summarizes conclusion and outlines limitation of the proposed approach. Section 7 contains acknowledgement followed by a list of references in Section 8.

II. RTL FAULT MODEL, FAULT-INJECTION ALGORITHM AND FAULT SIMULATION

Hardware description languages (HDLs) are used to model VLSI circuits. HDL constructs are classified into three types: structural, register-transfer level (RTL) and behavioral [21]. RTL constructs represent a subset of HDL constructs with the corresponding design guidelines meant to ensure the consistent synthesis of gate-level net lists by logic synthesis tools. With event scheduling and resource allocation information built-in, an RTL model represents the micro-architecture of a circuit. Some of the previous research in the area of high-level fault modeling scoped its application to behavioral design modeling [13, 18], while the other research aimed towards RTL design modeling [14, 15]. The research presented in this paper focuses its application on RTL design modeling. In this paper, the Verilog HDL is used as a medium to explain the proposed RTL fault model. However, the concepts developed and described here can be applied to any other hardware description language. RTL constructs of the Verilog HDL are listed in [25]. A few clarifications on the terminology used in this paper are offered here: Language Operators: RTL language operators are classified into Boolean (&, |, ^, ~), synthetic (+, -, *, >=, <=, <, >, ==,!=), and logical (&&, ||, !) operators. A further classification of these operators, though used in other contexts, is unnecessary for the purpose of RTL fault model description. 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

RTL fault model and a fault-injection algorithm presented in this paper 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 [20] 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. [19] 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 this pattern set, the detection probability of the fault is given as k/n. When two fault-lists (an RTL and collapsed gate-level) 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 [3] 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 error bounds established for the random sampling technique by Agrawal and Kato [3]. This assumption is supported by the data given in a later sub-section. RT level design description dictates the micro-architecture of the gate-level representation. During logic synthesis, RTL operators map onto Boolean components of varying complexities, 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 net list although 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 SSF model in the RTL fault model. These properties are listed below.

Properties of the gate level SSF model:

- 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 single stuck fault (SSF) assumption, only one fault is applied at a time when a test set is either being created or evaluated.
- The fault-list is collapsed using the "check-point" theorem [1]. The collapsed fault-list of a module contains input as well as fan-out faults. Further collapsing may be done using structural equivalence and dominance relationships.

Properties of the RTL fault model:

- Language operators (which map onto Boolean components in the gate-level net list) are assumed to be fault-free.
- Variables (which map onto signal lines in the gate-level net list) 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
- 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 fan-out faults. RTL variables that are used more than once in executable statements or the instantiations of lower level modules of the design-hierarchy are considered to have fan-out.
- Input faults of a module at the RT level have one-to-one equivalence to input faults of the module at the gate-level. The fan-out faults of variables inside a module at the RT level represent a subset of the fan-out faults of a possible gate-level implementation.

The definition of the RTL fault model and the fault injection algorithm encompasses modeling of faults for synthetic, Boolean and logical operators, sequential elements and fan-out/stem variables, as well as the collapsing of RTL faults. RTL faults are depicted with crosses ("x") in Figure 1. When RTL constructs contain synthetic operators, faults are injected only on the input variables of such operators. During logic synthesis, the synthetic operators are replaced by combinational circuits implementing the respective functions, e.g., adder, subtractor, comparator, etc. The internal details of such functions represented by synthetic operators 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 using RTL constructs that contain Boolean operators, faults are injected on variables that form 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 fan-outs. The post-synthesis 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 post-synthesis design. When RTL constructs contain logical operators, faults are injected on variables that constitute inputs of such operators. Most often the post-synthesis 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 both 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 fan-out faults are unique since neither equivalence nor dominance relations exist between them. Stem and fan-out faults are treated as special cases in the RTL fault model as well. A separate RTL fault is injected on each fan-out 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 fan-out fault modeling. The properties described for stem and fan-out 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 [1]. 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.

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 single stuck-at fault assumption and simulated with a given set of test patterns. When the responses of a good circuit and a 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 fault 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 described in Section 2.1. For more details on the method of running RTL fault grading using Verifault-XL™, one may refer to Mao and Gulati [15] and Verifault-XL™ User's Guide [6]. Several circuits (see Table 1) were simulated using this method.

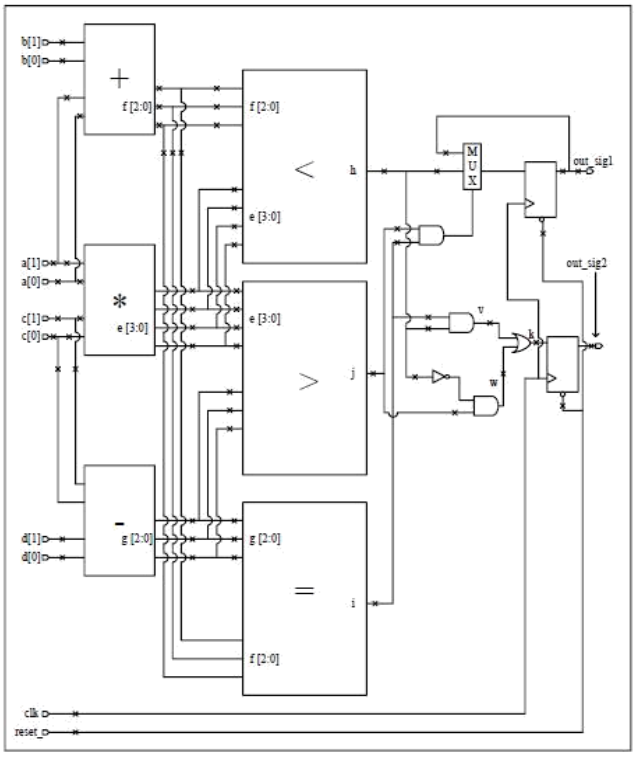

Figure1. RTL Faults in a Schematic Representation

Table 1 Design Data

	PI/PO/Bidi	Code Size (lines)	Area (gates)	No. of Faults		
				RTL	Gate	
					NCF	CF
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

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, in the input variables of the Boolean components, and on the fan-outs of Inter connecting signals between Boolean components. The RTL faults may also be placed on fan-outs 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 various internal faults. This is illustrated by the example provided in Figure 1 and Figure 2. Figure 1 contains a schematic representation of a hardware function. Figure 2 contains corresponding RTL description. The proposed RTL fault model and fault-injection algorithm when applied to the code, judiciously place faults in the RTL code. RTL faults are depicted with crosses ("x") in Figure 1. As can be observed in Figure 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 Figure 2 may produce many different gate-level implementations.

```
// 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

// 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
```

Figure2. RTL Code Example.

The gate-level implementations derive the micro architecture from the RTL description. One of the gate level net lists 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 sub-set of the gate-level fault-list.

2. Comparison of RTL and Gate Fault Coverage

Since the proposed RTL fault model is expected to resemble statistical properties of the random sample, the difference between the RTL and the gate-level fault coverages of a module for a given set of test patterns is expected to be within the error bounds established for the random-sampling technique. Agrawal and Kato [3] established the range of coverage for the random sampling technique as, when N gate faults are sampled from a total of M gate faults in the circuit, obtained from the Gaussian probability distribution function for a confidence probability of 0.998, and c is the ratio of detected to total among sampled gate faults. When equation (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 a module and c represents the ratio of the number of detected RTL faults to all RTL faults.

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

In this section, the RTL and the gate 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 algorithm. RTL modules are then synthesized using the commercial logic synthesis tool Design Compiler™ [5] and a 0.35 micron 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 range with a confidence probability of 99.8% as per equation (1). All circuits used for experiments, M1-Counter (4-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 ASIC, contain sequential as well as combinational logic.

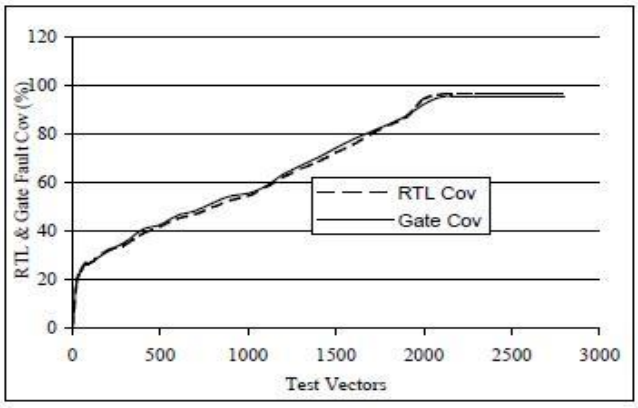

Figure 3 Module M3: RTL and Gate Cov. v/s Test Vectors.

Table 2 Empirical Data for Modules

Module	Test	c_m (%)	C_m (%)	c_m-C_m (%)	Estimation Error (%) (3σ bound)
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

Design data for these circuits (PI- No. of Primary Input signals, PO-No. of Primary Output signals, Bidi-No. of Bi-directional signals, Code Size- No. of lines of Verilog HDL code, and Area-Gate area measured as No. of two-input NAND CMOS equivalent gates) is provided in Table 1. Table 2 contains empirical data. For all except four data-points, RTL coverage is found tracking gate-level coverage within statistical error bounds. These four cases are being investigated and may contribute towards refinement of the proposed fault model. Figure 3 and Figure 4 show correlation between RTL and gate-level fault coverage across 31 data points for Module M3. Similar results are reported for other circuits (M1, M2 and M4) in [26]. From the experimental data (see Table 2), it can be conclusively derived 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 (4-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 characteristic is a function of the sample size as well as population size and it is present in the random sampling technique as well [2].

However, modern VLSI 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 presented in the previous section reveal that the RTL fault coverage provides a good estimate of the gate level fault coverage. The experimental data also reveals 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, for module M1, the number 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 defined 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 smaller gate-level implementation and subsequently make a smaller contribution of gate-level faults to the 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 larger gate-level implementation after logic synthesis, and thus contribute a larger percentage of faults to the overall gate-level fault-list of the system.

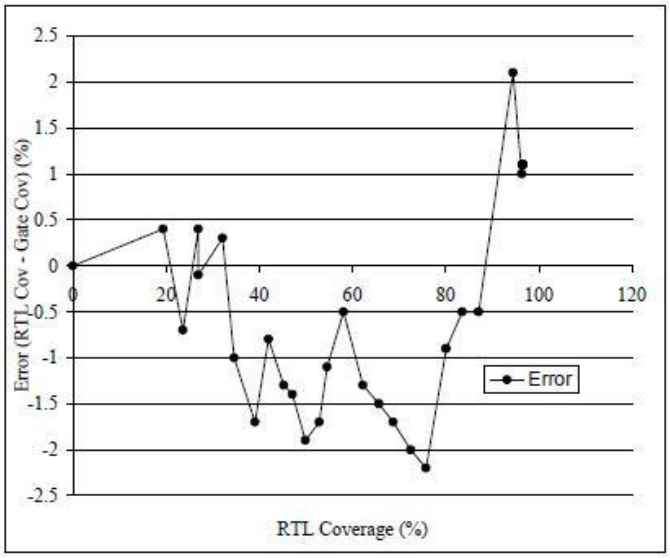

Figure 4 Module M3: Error v/s Fault Coverage Table

3 Inaccurate Estimation of Gate Fault Coverage

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 3 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. 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. The above observation leads us to consider a technique known as stratified sampling [22].

Table1.

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

According to stratified sampling technique, the fault population is divided according to 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 Gm gate-level faults. Then,

$$G = \sum_{m=1}^{M} G_m \tag{2}$$

and,

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

From the mth module, rm 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, two coverages for the mth module are defined as

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

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

Where ymi are random variables whose values are determined by fault simulation: ymi = 1 if the ith sampled fault in mth module is detected, or ymi = 0 if that fault is not detected. Cm is an unbiased estimate [26] for the gate-level fault coverage of the module m. Cm is the true gate-level fault coverage in the mth module which provides the total gate-level coverage of the 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 3. 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 C is different from the overall non-stratified RTL fault coverage, which is given by

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

In general, the non-stratified RTL coverage can be quite different from the true gate-level coverage. The variance of stratified RTL fault coverage is given as

$$\sigma^2 = \sum_{m=1}^{M} \frac{W_m^{\,2}}{r_m - 1} c_m (1 - c_m) \qquad (9)$$

For a given confidence probability, the range of coverage is given as [26]

$$\overline{C} \pm t\sigma \qquad (10)$$

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. It is evident from equations (7), (9) and (10), the parameters that are crucial in determining stratified RTL fault coverage and error bounds do not require knowledge of the absolute values of Gm or G. They require the ratio of the two quantities in the form of stratum weights. Techniques developed to determine the stratum weights of modules in a given VLSI system are described next.

IV. Stratum Weight Extraction Techniques

Approaches proposed for determining the stratum weights of modules of a VLSI system are classified in three categories:

- Logic synthesis based weight extraction,
- Entropy-measure based weight extraction [8, 16], and
- Floor-planning based weight extraction.

For more details on these proposed techniques, one may refer to [26]. Accurate stratum weights are only available after final logic synthesis. However, the techniques proposed above allow one to estimate stratum weights of modules during early phases of the design cycle. The impact of error in weight estimation on the overall stratified RTL fault coverage of the VLSI system is observed to be negligible [16].

V. EXPERIMENTAL RESULTS

Several industry-application VLSI systems ranging in size from 1,000 to 105,000 gates were used for empirical studies. The experimental procedure is as follows,

- The RTL code is run through C++ parser. This parser processes the RTL code and generates the fault-injected RTL code without altering circuit behavior. The parser is developed according to the proposed fault model and fault-injection algorithm.
- 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 proposed techniques.
- Stratified RTL fault coverage is computed using equation (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 equation (10).

The experimental procedure described above was carried out using commercial electronic design automation tools. The estimates of the gate-level fault coverages were compared with the non-stratified RTL fault coverages calculated using equation (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 net list. Gate-level fault simulations were pre formed using Verifault-XL. The gate-level net list for each VLSI system was obtained by performing logic synthesis of the RTL code for an arbitrary set of optimization constraints. Design Compiler was used for logic synthesis [5]. 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 4 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 equations (9) and (10). As can be noted in data for system D3 (Table 4), the error bounds for test pattern sets T1, T5, T6 and T7 are significantly wider than those for T2, T3 and T4. The RTL fault simulations for test pattern sets T1, T5, T6 and T7 were performed using a very small random sample of the overall RTL fault-list. As per equation (9), 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 4. 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.

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 [9, 10, 12, 14, 17]. 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. [24] with empirical data, the architectural testability properties and the subsequent test weaknesses of a gate-level net list 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.

Table 4 Empirical Data for Systems

	Test	C_{RTL} (%)	$\overline{C}$ (%)	C (%)	$\overline{C} - C$ (%)	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	19.4	18.0	– 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

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 thesis holds promise for significantly reducing the performance penalties of the gate-level fault simulation approach. 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. 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.

VII. ACKNOWLEDGMENT

The authors thank Natalya Dvorson, Arthur Friedman, and John Vogel for their help.

VIII. REFERENCES

[1] M. Abramovici, M. A. Breuer, and A. D. Friedman, Digital Systems Testing and Testable Design, IEEE Press, New York, NY, 1990.

[2] V. D. Agrawal, "Sampling Techniques for Determining Fault Coverage in LSI Circuits," J. Digital Systems, vol. 5, Sept 1981, pp. 189-202.

[3] V. D. Agrawal and H. Kato, "Fault Sampling Revisited," IEEE Design & Test of Computers, vol. 7, August 1990, pp. 32-35.

[4] J. R. Armstrong, F. S. Lam, and P. C. Ward, "Test Generation and Fault Simulation for Behavioral Models," Performance and Fault Modeling with VHDL, Prentice-Hall, Englewood Cliffs, NJ, 1992, pp. 240-303.

[5] H. Bhatnagar, Advanced ASIC Chip Synthesis, Kluwer Academic Publishers, Boston, MA, 1999.

[6] Cadence Design Systems, Inc., Verifault-XL User's Guide, San Jose, CA, 1997.

[7] T. Chakraborty and S. Ghosh, "On Behavior Fault-Modeling for Combinational Digital Designs," Proc. International Test Conference, Sept 1988, pp. 593- 600.

[8] K. T. Cheng and V. D. Agrawal, "An Entropy Measure for the Complexity of Multi-output Boolean Functions," Proc. 27th Design Automation Conference, June 1990, pp. 302-305.

[9] S. Chiusano, F. Corno, and P. Prinetto, "RT-level TPG Exploiting High-Level Synthesis Information," Proc. 17th IEEE VLSI Test Symposium, April 1999, pp. 341-346.

[10] C. H. Cho and J. R. Armstrong, "B-algorithm: A Behavioral Test Generation Algorithm," Proc. International Test Conference, Oct 1994, pp. 968-979.

[11] W. G. Cochran, Sampling Techniques, John Wiley & Sons, Inc., New York, NY., 1977.

[12] F. Corno, P. Prinetto, and M. S. Reorda, "Testability Analysis and ATPG on Behavioral RT-level VHDL," Proc. International Test Conference, Nov 1997, pp. 753-759.

[13] S. Ghosh, "Behavioral-level Fault Simulation,"IEEE Design & Test of Computers, vol. 5, no. 3, June 1988, pp. 31-42.

[14] R. J. Hayne and B. W. Johnson, "Behavioral Fault Modeling in a VHDL Synthesis Environment," Proc. 17th VLSI Test Symposium, April 1999, pp. 333-340.

[15] W. Mao and R. Gulati, "Improving Gate Level Fault Coverage by RTL Fault Grading," Proc. International Test Conference, Oct 1996, pp. 150-159.

[16] N. Pippenger, "Information Theory and the Complexity of Boolean Functions," Mathematical Systems Theory, vol.10, 1977, pp.129-167.

[17] 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 European Design Automation and Test Conference, Feb 1998.

[18] P. Sanchez and I. Hidalgo, "System Fault Simulation," Proc. International Test Conference, Oct 1996, pp. 732-740.

[19] S. C. Seth, V. D. Agrawal, and H. Farhat, "A Statistical Theory of Digital Circuit," IEEE Trans. On Computers, vol. 39, no. 4, April 1990, pp. 582-586.

[20] F. Stephan and P. McCarthy, Sampling Opinions: An Analysis of Survey Procedure, John Wiley & Sons, Inc., New York, NY, 1958.

[21] E. Sternheim, R. Singh, R. Madhavan, and Y. Trivedi, Digital Design and Synthesis with Verilog HDL, Automata Publishing Company, San Jose, CA., 1993.

[22] A. Stuart, The Ideas of Sampling, Charles Griffin and Company, Ltd., High Wycombe, Great Britain, 1984.

[23] S. M. Thatte and J. A. Abraham, "Test Generation for Microprocessors," IEEE Transactions on Computers, vol. C-29, June 1980, pp. 429-441.

[24] P. A. Thaker, M. E. Zaghloul, and M. B. Amin, "Study of Correlation of Testability Aspects of RTL Description and Resulting Structural Implementations," Proc. 12th International Conference on VLSI Design, Jan 1999, pp. 256-259.

[25] P. A. Thaker, V. D. Agrawal, and M. E. Zaghloul, "Validation Vector Grade (VVG): A New Coverage Metric for Validation and Test, "Proc. 17th IEEE VLSI Test Symposium, April 1999, pp. 182-188.