Efficient RT-level Fault Diagnosis Methodology *

Ozgur Sinanoglu and Alex Orailoglu

Computer Science and Engineering Department
University of California, San Diego
La Jolla, CA 92093

{ozgur, alex}@cs.ucsd.edu

ABSTRACT

Increasing IC densities necessitate diagnosis methodologies with enhanced defect locating capabilities. Yet the computational effort expended in extracting diagnostic information and the stringent storage requirements constitute major concerns due to the tremendous number of faults in typical ICs. In this paper, we propose an RT-level diagnosis methodology capable of responding to these challenges. In the proposed scheme, diagnostic information is computed on a grouped fault effect basis, enhancing both the storage and the computational aspects. The fault effect grouping criteria are identified based on a module structure analysis, improving the propagation ability of the diagnostic information through RT modules. Experimental results show that the proposed methodology provides superior speed-ups and significant diagnostic information compression at no sacrifice in diagnostic resolution, compared to the existing gate-level diagnosis approaches.

1. INTRODUCTION

Diagnosing IC defects is becoming more crucial as increasing chip size and complexity magnify the defect probabilities. Accurate pinpointing of the cause of the failure based on the defective chip responses yields significant financial benefits, both by ramping up yields rapidly for next generation ICs and by reducing chip returns.

The diagnosis process has been traditionally carried out by matching the defective chip responses with the precomputed faulty responses. Such a process is performed based on a specific fault model as taking into account any possible defect is apparently infeasible. Attaining certain diagnostic resolution necessitates the selection of a representative fault model; the defective chip behavior is thus attributed to the presence of modeled faults. Currently existing diagnosis methodologies are typically based on the utilization of diagnostic information on a per fault basis; the applicability of these techniques is typically limited by the storage and computational complexity stemming from the tremendous number of possible fault locations in ICs.

Two types of fault dictionaries are used in typical gate-level diagnosis schemes: full fault dictionary and pass-fail dictionary. A full dictionary stores for each test vector and for each fault the complete circuit responses. In the presence of R outputs, T test vectors and F faults, the size of a full fault dictionary is $O(R \cdot T \cdot F)$. Although a full dictionary delivers perfect diagnostic resolution, storing only the information regarding whether faults are detected by test vectors in a pass-fail dictionary provides at significantly lower cost almost as high a resolution as a full dictionary. The size of a pass-fail dictionary is still considerable, however, as $O(T \cdot F)$ bits are required. Besides their storage complexity, building these dictionaries demands significant computational effort due to the tremendous number of faults. In order to reduce the space needed to store a fault dictionary, numerous methods have been proposed [1, 2, 3, 4, 5]; a detailed summary can be found in [6]. Although these techniques alleviate the storage requirements slightly, the computational challenges remain yet unsolved.

An aspect that promises to cope with the aforementioned storage and computational challenges, yet overlooked by the previous diagnosis methodologies, is the utilization of RT-level techniques. Hierarchical approaches have been widely utilized in test generation [7] and fault simulation [8, 9], delivering computational speedups. Utilization of design hierarchy along with the exploitation of module functionality helps extract diagnostic information while satisfying storage and computational requirements.

The computation of diagnostic information on a per grouped faults basis rather than on a per fault basis attains significant storage compression. The utilization of RT module functionalities helps capture the grouped fault effect propagation behavior of the modules, enabling the tracking of the diagnostic information through RT modules. While the computation of diagnostic information regarding grouped fault effects delivers computational efficiency, rapid propagation through larger RT blocks further expedites the diagnostic information collection process compared to gate-level circuit traversing schemes.

The storage and computational benefits delivered by the tracking of information regarding grouped fault effects strongly depend on the fault effect grouping criteria; the mutual manifestation behavior of the faults in a group constitutes the diagnostic information to be tracked. In this work, we analyze the functionality of common RT-level modules so as to identify the fault effect grouping criteria that enable an increased propagation ability of the diagnostic information. We provide an analysis that captures and exploits the monotonic information flow inherent in various RT-level operators. Such a regularity helps model the diagnostic information propagation through the corresponding RT modules. Based on this analysis, fault effects are judiciously grouped, enabling the preservation of the group-characterizing information in propagation through RT-level modules.

In this paper, we propose an RT-level diagnosis methodology based on the computation of manifestation information regarding grouped fault effects. Not only the execution of the RT-level tool is significantly faster compared to that of the gate-level techniques, but also considerably smaller storage size is required for the output diagnostic information. The proposed diagnosis tool is able to deliver maximal possible diagnostic resolution; not only are the fault modules identified but furthermore the gate-level faults that account for the defective chip behavior are pinpointed accurately. The diagnosis methodology we propose is customizable to any fault model, promising to deliver further enhanced defect location capabilities.

2. RT-LEVEL APPROACH TO DIAGNOSIS

While attaining high diagnostic resolution is crucial in the identification of the cause of IC failures, computational and storage challenges complicate the diagnostic information computation process. An RT-level diagnosis approach is capable of responding to these challenges. Rapid propagation of the fault manifestation informa-

*The work of the first author is supported through a graduate fellowship by IBM.

tion through high level blocks alleviates the computational complexity of the diagnostic information extraction process. Furthermore, the grouping of fault effects and the computation of their manifestation information as a group rather than on a per fault basis not only delivers computational benefits but also results in succinct diagnostic information, satisfying the storage requirements simultaneously.

Partitioning the faults in the Module Under Diagnosis (MUD), *i.e.* the module whose faults are targeted, into groups enables an RT-level approach to computing the signature of a fault group at the primary outputs based on the signature at the MUD outputs. Several MUD faults with a mutual manifestation behavior can be grouped together, giving rise to the propagation of the manifestation information from the MUD outputs to the primary outputs. Preservation of this information during propagation through other modules reflects into the delivery of high diagnostic resolution levels, as the fault group signatures at primary outputs help distinguish the fault pairs.

The mutual manifestation behavior corresponding to a group of faults delivers enhanced diagnostic information compared to the pass-fail dictionary approach; the manifestation information of a fault consists not only of the pass-fail information but furthermore of the bit location(s) wherein the fault manifests. Perfect preservation of the manifestation information regarding fault groups from the MUD output to the primary outputs is slated to deliver more enhanced diagnostic resolution levels compared to pass-fail dictionaries. The loss of the manifestation information during propagation through other modules, however, reflects into diagnostic resolution degradation. The fault grouping criterion and the propagation ability of the consequent manifestation information are crucial as they determine the diagnostic capabilities of an RT-level approach.

Similar to the dictionary-based approaches, diagnostic information should be computed and stored for every test vector individually; each test vector brings the circuit into a distinct state, imposing distinct fault sensitization conditions for the same fault. During the diagnosis application process, based on the defective responses, every test vector pinpoints a group of fault effects that accounts for the defective chip behavior; intersection of these groups yield the candidate fault(s). Activation of differing sensitization paths for distinct test vectors yields differing fault groupings, narrowing down the candidate fault set with the application of each test vector.

3. FAULT GROUPING CRITERION

The fact that typical RT-level designs consist of modules with monotonic information flow enables the utilization of this property in the efficient propagation of the fault manifestation information. The monotonic information flow in these modules can be observed and modeled so as to identify the conditions for the propagation of the fault manifestation information through these modules; typical module examples that exhibit monotonicity are adders and multiplers, widely utilized in arithmetic circuits. The identification of the propagation conditions, which may be used as a basis for grouping fault effects, helps compute the expected signature of the judiciously formed fault effect groups. The mutual manifestation behavior of the faults in a group, defined by the grouping criteria, constitutes the information to be tracked through the RT modules.

The independence of distinct output bits in a typical logic block significantly eases the trace of the fault manifestation information. The impact of a group of fault effects on the output bits can be identified based on their impact on the input bits due to the structural regularity. For modules such as logic arrays, the output bit in a certain position is computed based on only the input bits in the same position. Similarly, for shifters, an output bit depends only on a single input bit. The independence among bitwise output computations hints at the information to be utilized in grouping the fault effects and in tracing the fault manifestation information: faulty bit locations, *i.e.*, the bit positions in which the correct and the faulty responses differ. For a logic array module, the propagation of the faulty bit location information can be questioned by a simple comparison of the corresponding side input bit with the controlling value. Similarly, faulty bit location trace is straightforward through shifters as well; the faulty bit location information is retained unless the faults manifest in a bit that has been shifted out. For these modules, the propagation of the fault manifestation information can be thus summarily resolved as the faulty bit location information can be computed for the module output.

The arithmetic operators, on the other hand, typically fail to satisfy the aforementioned independence property, slightly complicating the identification of the appropriate fault grouping criteria. The associated information flow is typically unidirectional for common arithmetic operations, resulting in a monotonic dependence of the output bits on the input bits; an output bit in a certain position depends on only the input bits to either the right or the left of this position[1], but not both. This monotonicity results in the preservation of either the rightmost or the leftmost faulty bit location information depending on the monotonicity direction.

Commonly utilized arithmetic modules, such as adders and multipliers, typically exhibit a right-to-left monotonic information flow, enabling the utilization of the rightmost faulty bit location for tracing the fault manifestation information. For the adder module, one can observe this monotonicity in the information flow by examining the bitwise output computation; an output bit in a certain position is computed by XORing the input bits in the same position with the corresponding carry bit. In fact, accounting for the carry bit in the computation of the output bits distinguishes an addition operation from a bitwise XOR operation that satisfies the bitwise independence property. Due to the carry bits, a monotonic information flow exists from right to left, shattering the independence property in the addition operation; as a carry bit in a certain position depends on all the bits of the inputs to the right of this position, so does the corresponding output bit. The monotonicity property, which stems from the fact that an output (or carry) bit in a certain position depends on none of the bits to the left of this position, results in the preservation of the *Rightmost Faulty Bit Location* (RFBL) through an adder module. An analogous argument can be made for the subtractor module as the borrow bits exhibit monotonicity with a right-to-left flow as well. A group of fault effects having a certain RFBL on one of the inputs of an adder (subtractor) displays the same RFBL in the module output. The monotonicity property of the addition (subtraction) operation enables tracking the RFBL information through these modules, consequently.

As the multiplication operation can be considered as a series of addition operations, it satisfies the monotonicity property as well, expectedly resulting in RFBL information preservation. To justify this intuitive observation, we can examine the shift-and-add multiplication algorithm [10] for an example wherein the first input has an RFBL of r and the second input is nonzero and it has w consecutive 0's in its rightmost bit positions. In every step of the shift-and-add algorithm, A is shifted left by one; the decision on whether to add A to the accumulated sum depends on the bits of B. In each of the first w steps, A is shifted left, with no additions performed. Then A is added to the sum, which is 0; therefore, at that point, the sum has a faulty bit, whose location is $r + w$. In the subsequent steps, this sum is possibly added with the A bits shifted left whose faulty bit locations exceed $r + w$. Using the result of our analysis for the addition operation, we can con-

[1]We follow the convention that the rightmost bit corresponds to the least significant bit, and the leftmost bit, to the most significant bit.

Module	$RFBL_z$	Condition for propagation
Register	$RFBL_a$	Load = 1, Clear = 0
ADD/SUB	$\mathcal{S}(RFBL_a, RFBL_b)$	-
Multiplier	$\mathcal{S}(RFBL_a + \mathcal{W}(b),$ $RFBL_b + \mathcal{W}(a),$ $RFBL_a + RFBL_b)$	-
(N)AND/ (N)OR	$min(RFBL_a, RFBL_b)$	if $RFBL_a = RFBL_b$ $a[RFBL_a] = b[RFBL_b]$; else if $RFBL_a < RFBL_b$ $b[RFBL_a] = \overline{\mathcal{C}(Module)}$; else $a[RFBL_b] = \overline{\mathcal{C}(Module)}$
Shift-left	$RFBL_a + 1$	$RFBL_a \neq buswidth$
Shift-right	$RFBL_a - 1$	$RFBL_a \neq 0$
MUX	$RFBL_a$	select=0
	$RFBL_b$	select=1

Table 1: RFBL information propagation through RT modules

clude that the output has a rightmost faulty bit location of $r + w$, and that the RFBL information is propagated through the multiplier.

For the common RT modules, the RFBL information can be utilized in the computation of diagnostic information regarding a group of fault effects, as this information is typically preserved subsequent to propagation. Both computational and storage requirements can thus be satisfied in diagnosing large ICs while delivering high diagnostic resolution levels.

4. RT MODULE CHARACTERIZATION

The computation of RFBL-based diagnostic information necessitates the modeling of the RFBL information propagation behavior of RT modules. The propagation of the RFBL information regarding a group of faults in a certain module to the circuit primary outputs helps compute the expected faulty signature corresponding to these faults. In this section we define the RFBL propagation rules for various RT modules.

As the information traced consists of the rightmost faulty bit location, the failure of RFBL information propagation results in the loss of information regarding the corresponding fault effects. The only information that can be still utilized subsequent to the loss of the RFBL information is the fact that the bits to the right of the rightmost faulty bit location are fault-free. As the bits to the left of the rightmost faulty bit location may or may not be faulty, no information can be utilized regarding the bits in these locations. We therefore introduce the *strictly minimum function*, denoted by $\mathcal{S}()$, that takes as inputs two integers and returns the mathematical minimum of its arguments only if the two values differ; otherwise, it returns a symbolic value that denotes a number that is only known to be greater than its arguments. In the latter case, the information regarding the lower bound on the fault bit index is utilized to track the fault-free bits to the right of this index.

Let us consider an adder module with both of its inputs faulty. If the input RFBLs differ, the RFBL of the output equals the smaller input RFBL; otherwise, the exact RFBL information is lost as the faulty bits in the same bit locations of the two inputs cancel. In the latter case, the only information that can still be utilized is the fact that the adder output bits to the right of the input RFBL are fault-free. The RFBL information propagation behavior of an adder module can be modeled through the $\mathcal{S}$ operation, consequently.

Table 1 provides the RFBL information propagation through some common RT modules; the output RFBL, denoted as $RFBL_z$, based on the RFBL of the inputs, denoted as $RFBL_a$ and $RFBL_b$, and the condition to be satisfied to ensure the propagation of the RFBL information are shown. For some logic operations, the *controlling value of the operation*, denoted as the $\mathcal{C}()$ function, may determine the propagation of the RFBL information; the $\mathcal{C}()$ function takes as input a logic operation and returns the controlling value bit of this operation. The RFBL information propagation behavior of regular logic arrays can be modeled by the mathematical minimum operation, denoted as $min()$, due to the independence of output bit computations. The RFBL information propagation of the multiplier operation is modeled based on the $\mathcal{W}()$ function, which takes as input a bit string and returns an integer that denotes the number of consecutive 0's in the rightmost bit positions of the bit string.

The propagation conditions are defined for the cases wherein multiple module inputs manifest the fault effects; under the single faulty module assumption, multiple faulty inputs occur due to reconvergent fanouts or loops in the designs. In the typically more common case wherein only a single module input manifests the fault effects, the propagation conditions defined in table 1 are significantly relaxed, as the aliasing possibility, which is due to multiple fault effects canceling each other out, is eliminated; the rules defined as a min or a $\mathcal{S}$ function are simplified down to a single term consisting of the RFBL value of the faulty input only. Although in this work we define a diagnosis methodology under the single faulty module assumption, the proposed methodology can be extended to handle multiple faulty modules simultaneously due to the generality of the RFBL propagation rules defined in table 1.

5. DIAGNOSTIC INFORMATION

An efficient RT-level analysis can be carried out based on the aforementioned RFBL information propagation, enabling the computation of storage efficient diagnostic information. Although such a grouped fault effect based analysis attains a high diagnostic resolution in a computationally efficient manner, it is possible that certain fault pairs may still be indistinguishable based on the diagnostic information collected. To regain the maximum possible resolution, these indistinguishable fault pairs should be targeted individually.

The proposed diagnostic information computation process consists of two phases. In the first phase, an RT-level analysis is performed based on RFBL information propagation, attaining a certain diagnostic resolution. In this phase of the algorithm, the only assumption is the single faulty module assumption which can be eliminated as explained earlier; no assumptions regarding a certain fault model are made. The first phase of the algorithm can be customized with respect to any fault model, such as the single/multiple stuck-at/bridging fault models, as the manifestation behavior of any type of faults can be modeled by the RFBL information.

In the second phase of the algorithm, gate level information is utilized so as to distinguish the fault pairs that the first phase fails to distinguish; the fault pairs (based on the desired fault model) that the RFBL-based analysis of the first phase fails to distinguish are targeted individually, attaining maximal resolution. The amount of diagnostic information to be stored is proportional to the number of fault pairs to be handled in the second phase; the effectiveness in terms of storage of the proposed methodology strongly depends on the diagnostic resolution level attained at the end of the first phase and hence on the RFBL information propagation behavior of the RT modules.

5.1 RFBL Information Propagation

Based on the RFBL information propagation rules defined in table 1, an RT-level analysis can be performed to extract the diagnostic information for the RT modules in the design. In this analysis, each module is attacked individually; the RFBL information in the output of the targeted module, denoted as Module Under Diagnosis (MUD), is propagated through other modules, computing the RFBL signature of the MUD faults at the primary outputs.

Two integers are utilized to trace the RFBL information corresponding to the MUD; a symbol in the form of $i \rightarrow j$ denotes that an RFBL of i in the MUD output reflects into an RFBL of j in the current cir-

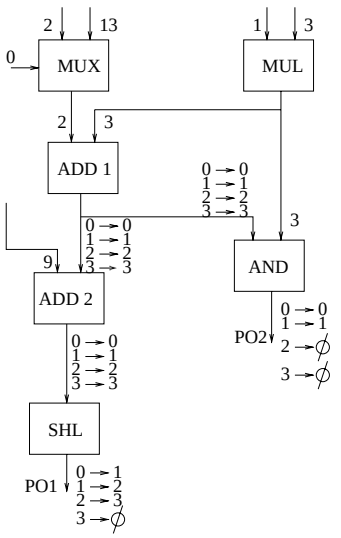

Figure 1: **RFBL information propagation for ADD1 module**

Module	PO_1	PO_2
MUX	$0 \rightarrow 1$	$0 \rightarrow 0$
	$1 \rightarrow 2$	$1 \rightarrow 1$
	$2 \rightarrow 3$	$2 \rightarrow \phi$
	$3 \rightarrow \phi$	$3 \rightarrow \phi$
MUL	$0 \rightarrow 1$	$0 \rightarrow 0$
	$1 \rightarrow 2$	$1 \rightarrow 1^+$
	$2 \rightarrow 3$	$2 \rightarrow 2^+$
	$3 \rightarrow \phi$	$3 \rightarrow 3$
ADD1	$0 \rightarrow 1$	$0 \rightarrow 0$
	$1 \rightarrow 2$	$1 \rightarrow 1$
	$2 \rightarrow 3$	$2 \rightarrow \phi$
	$3 \rightarrow \phi$	$3 \rightarrow \phi$
ADD2	$0 \rightarrow 1$	$0 \rightarrow \phi$
	$1 \rightarrow 2$	$1 \rightarrow \phi$
	$2 \rightarrow 3$	$2 \rightarrow \phi$
	$3 \rightarrow \phi$	$3 \rightarrow \phi$
AND	$0 \rightarrow \phi$	$0 \rightarrow 0$
	$1 \rightarrow \phi$	$1 \rightarrow 1$
	$2 \rightarrow \phi$	$2 \rightarrow 2$
	$3 \rightarrow \phi$	$3 \rightarrow 3$
SHL	$0 \rightarrow 0$	$0 \rightarrow \phi$
	$1 \rightarrow 1$	$1 \rightarrow \phi$
	$2 \rightarrow 2$	$2 \rightarrow \phi$
	$3 \rightarrow 3$	$3 \rightarrow \phi$

Figure 2: **Module signatures**

	t_1	t_2	t_3	t_4	t_5
f_1	0	0	1	0	1
f_2	0	0	0	1	1
f_3	1	0	0	1	1
f_4	0	0	1	0	1
f_5	1	1	1	0	1
f_6	0	1	1	1	1
f_7	0	1	0	1	0

Figure 3: **Pass-fail information**

	t_1	t_2	t_3	t_4	t_5
(f_1, f_2)			$\surd$	$\surd$	
(f_1, f_3)	$\surd$		$\surd$	$\surd$	
(f_2, f_3)	$\surd$				
(f_4, f_5)	$\surd$	$\surd$			
(f_6, f_7)			$\surd$		$\surd$

Figure 4: **Pairwise distinguishability table**

cuit line subsequent to propagating through numerous modules. The integer to the left of the '$\rightarrow$' symbol is denoted as the *characterizing RFBL*, as it represents a group of fault effects in the MUD and remains constant during the propagation of the symbol through other modules; the integer to the right of the '$\rightarrow$' symbol is denoted as the *reflected RFBL*. The characterizing and the reflected RFBLs attached to the same symbol may be distinct due to operations such as $shift$; these operations might lead to the manifestation of the fault effects at bit locations other than the ones at the MUD output line, possibly changing the reflected RFBL. The RFBL-based symbols in the primary output buses are computed for any possible MUD output bit; the number of symbols to be propagated equals the MUD output bus width. The diagnostic information in the form of RFBL reflections of MUD output bits to primary outputs is thus computed.

Figure 1 illustrates the proposed RT-level analysis for the extraction of the RFBL-based diagnostic information for a single test vector. The step-by-step RFBL information propagation corresponding to the $ADD1$ module is illustrated on the RT-level circuit; based on the propagation rules defined in table 1, the RFBL information is propagated through the RT modules. As all the circuit lines are four-bit wide, the number of symbols to be propagated is four. Although the addition operation ($ADD2$) retains the RFBL information, due to the SHL module, the reflected RFBL values exceed the characterizing RFBL values by 1 in every symbol in $PO1$; the fault effects on the leftmost bit, denoted by the characterizing RFBL of 3, are lost. As the side input of the AND module is 3, "0011" in binary representation, the fault effects with characterizing RFBL values of 2 or 3 are lost, denoted by the ϕ symbol; the two rightmost bits of $PO2$ still manifest the fault effects with characterizing RFBL values of 0 or 1. An $ADD1$ module fault that has an RFBL value of 0 in the $ADD1$ output, for instance, exhibits the RFBL values of 1 and 0 in $PO1$ and $PO2$, respectively. The RT-level analysis illustrated for the $ADD1$ module can be performed for the other modules as well, computing their signatures; figure 2 provides the RFBL information propagation corresponding to all the modules in the example RT-level circuit.

The RT-level diagnostic information to be stored consists of the reflected RFBL values at the primary output bus(es) for every module output bit. Compared to the gate-level diagnostic information that is stored on a per fault basis, significant storage benefits are delivered by the proposed RT-level analysis due to the information storage on a per fault group basis. Specifically, the storage complexity of the information per test vector can be denoted as $O(\sum_{i=1}^{M} |ModuleOut_i| \cdot \sum_{i=1}^{PO} log(|PO_i|))$, wherein M and PO denote the number of RT modules and primary output buses, respectively, while the bus width of the i^{th} module output and the i^{th} primary output are denoted by $|ModuleOut_i|$ and $|PO_i|$, respectively. The storage requirement of the RT-level diagnostic information delivered by the proposed methodology can be easily contrasted with that of the gate-level schemes, wherein a full dictionary necessitates the storage of $O(f \cdot \sum_{i=1}^{PO} |PO_i|)$ bits; the tremendous number of faults, denoted by f, typically magnifies considerably the storage required for fault-based dictionaries in gate-level approaches.

5.2 Diagnostic Resolution Restoration

The RFBL-based diagnostic information delivered by the first phase of the algorithm is fault model independent; the signatures for the fault groups (from any fault model) are computed and utilized for diagnosis purposes. In the second phase of the algorithm, described in this section, the diagnostic resolution gap between that attained in the first phase and the maximal possible resolution is restored. The fault pairs, in the desired fault model that the RFBL-based diagnostic information fails to distinguish are individually targeted to identify the minimally sized gate-level diagnostic information. In minimizing the amount of diagnostic information, the second phase of the proposed algorithm benefits from the fact that only a small number of faults need to be distinguished.

The pass-fail information regarding a minimal set of test vectors is computed by the second phase of the algorithm. A tabular structure, denoted as the *pairwise distinguishability table*, is utilized, with the rows denoting the indistinguishable fault pairs and the columns, the test vectors. An entry in this table is marked only if the faults in the corresponding fault pair can be distinguished by the corresponding test vector; a pair of faults can be distinguished by a test vector only if the vector detects one of the faults and fails to detect the other. A minimal set cover algorithm is executed on this table to identify the minimal number of test vectors that cover all the fault pairs, distinguishing the faults in any of the fault pairs listed in the table. The diagnostic information with respect to a small subset of fault pairs and a small subset of test vectors helps attain maximal possible resolution with significantly reduced storage size.

An example pass-fail information is provided in Figure 3; in the pass-fail table, a 1-entry denotes that the corresponding fault is detected by the corresponding test vector. For the RFBL information based indistinguishable fault pairs, (f_1, f_2), (f_1, f_3), (f_2, f_3), (f_4, f_5), and (f_6, f_7), the pairwise distinguishability table is provided in Figure 4. The execution of a minimal set cover algorithm on the pairwise distinguishability table identifies the minimal set of test vectors that distinguish all the faults listed in the table; the utilization of the pass-fail information of only t_1 and t_3 for these seven faults helps restore the diagnostic resolution loss of the first phase.

Module	t_1		t_2		t_3		t_4		t_5	
	PO_1	PO_2	PO_1	PO_2	PO_1	PO_2	PO_1	PO_2	PO_1	PO_2
MUX	$0 \rightarrow 1$	$0 \rightarrow 0$	$0 \rightarrow 1$	$0 \rightarrow 0$	$0 \rightarrow 1$	$0 \rightarrow \phi$	$0 \rightarrow 1$	$0 \rightarrow \phi$	$0 \rightarrow 1$	$0 \rightarrow \phi$
	$1 \rightarrow 2$	$1 \rightarrow 1$	$1 \rightarrow 2$	$1 \rightarrow \phi$	$1 \rightarrow 2$	$1 \rightarrow 1$	$1 \rightarrow 2$	$1 \rightarrow \phi$	$1 \rightarrow 2$	$1 \rightarrow 1$
	$2 \rightarrow 3$	$2 \rightarrow \phi$	$2 \rightarrow 3$	$2 \rightarrow 2$	$2 \rightarrow 3$	$2 \rightarrow 2$	$2 \rightarrow 3$	$2 \rightarrow \phi$	$2 \rightarrow 3$	$2 \rightarrow \phi$
	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow 3$
MUL	$0 \rightarrow 1$	$0 \rightarrow 0$	$0 \rightarrow 1$	$0 \rightarrow 0$	$0 \rightarrow 1$	$0 \rightarrow 0^+$	$0 \rightarrow 1$	$0 \rightarrow 0$	$0 \rightarrow 1$	$0 \rightarrow 0$
	$1 \rightarrow 2$	$1 \rightarrow 1^+$	$1 \rightarrow 2$	$1 \rightarrow 1$	$1 \rightarrow 2$	$1 \rightarrow 1$	$1 \rightarrow 2$	$1 \rightarrow 1$	$1 \rightarrow 2$	$1 \rightarrow 1$
	$2 \rightarrow 3$	$2 \rightarrow 2^+$	$2 \rightarrow 3$	$2 \rightarrow 2$	$2 \rightarrow 3$	$2 \rightarrow 2$	$2 \rightarrow 3$	$2 \rightarrow 2$	$2 \rightarrow 3$	$2 \rightarrow 2^+$
	$3 \rightarrow \phi$	$3 \rightarrow 3$	$3 \rightarrow \phi$	$3 \rightarrow 3$	$3 \rightarrow \phi$	$3 \rightarrow 3$	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow 3$
ADD1	$0 \rightarrow 1$	$0 \rightarrow 0$	$0 \rightarrow 1$	$0 \rightarrow 0$	$0 \rightarrow 1$	$0 \rightarrow \phi$	$0 \rightarrow 1$	$0 \rightarrow \phi$	$0 \rightarrow 1$	$0 \rightarrow \phi$
	$1 \rightarrow 2$	$1 \rightarrow 1$	$1 \rightarrow 2$	$1 \rightarrow \phi$	$1 \rightarrow 2$	$1 \rightarrow 1$	$1 \rightarrow 2$	$1 \rightarrow \phi$	$1 \rightarrow 2$	$1 \rightarrow 1$
	$2 \rightarrow 3$	$2 \rightarrow \phi$	$2 \rightarrow 3$	$2 \rightarrow 2$	$2 \rightarrow 3$	$2 \rightarrow 2$	$2 \rightarrow 3$	$2 \rightarrow \phi$	$2 \rightarrow 3$	$2 \rightarrow \phi$
	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow 3$
ADD2	$0 \rightarrow 1$	$0 \rightarrow \phi$	$0 \rightarrow 1$	$0 \rightarrow \phi$	$0 \rightarrow 1$	$0 \rightarrow \phi$	$0 \rightarrow 1$	$0 \rightarrow \phi$	$0 \rightarrow 1$	$0 \rightarrow \phi$
	$1 \rightarrow 2$	$1 \rightarrow \phi$	$1 \rightarrow 2$	$1 \rightarrow \phi$	$1 \rightarrow 2$	$1 \rightarrow \phi$	$1 \rightarrow 2$	$1 \rightarrow \phi$	$1 \rightarrow 2$	$1 \rightarrow \phi$
	$2 \rightarrow 3$	$2 \rightarrow \phi$	$2 \rightarrow 3$	$2 \rightarrow \phi$	$2 \rightarrow 3$	$2 \rightarrow \phi$	$2 \rightarrow 3$	$2 \rightarrow \phi$	$2 \rightarrow 3$	$2 \rightarrow \phi$
	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow \phi$	$3 \rightarrow \phi$
AND	$0 \rightarrow \phi$	$0 \rightarrow 0$	$0 \rightarrow \phi$	$0 \rightarrow 0$	$0 \rightarrow \phi$	$0 \rightarrow 0$	$0 \rightarrow \phi$	$0 \rightarrow 0$	$0 \rightarrow \phi$	$0 \rightarrow 0$
	$1 \rightarrow \phi$	$1 \rightarrow 1$	$1 \rightarrow \phi$	$1 \rightarrow 1$	$1 \rightarrow \phi$	$1 \rightarrow 1$	$1 \rightarrow \phi$	$1 \rightarrow 1$	$1 \rightarrow \phi$	$1 \rightarrow 1$
	$2 \rightarrow \phi$	$2 \rightarrow 2$	$2 \rightarrow \phi$	$2 \rightarrow 2$	$2 \rightarrow \phi$	$2 \rightarrow 2$	$2 \rightarrow \phi$	$2 \rightarrow 2$	$2 \rightarrow \phi$	$2 \rightarrow 2$
	$3 \rightarrow \phi$	$3 \rightarrow 3$	$3 \rightarrow \phi$	$3 \rightarrow 3$	$3 \rightarrow \phi$	$3 \rightarrow 3$	$3 \rightarrow \phi$	$3 \rightarrow 3$	$3 \rightarrow \phi$	$3 \rightarrow 3$
SHL	$0 \rightarrow 0$	$0 \rightarrow \phi$	$0 \rightarrow 0$	$0 \rightarrow \phi$	$0 \rightarrow 0$	$0 \rightarrow \phi$	$0 \rightarrow 0$	$0 \rightarrow \phi$	$0 \rightarrow 0$	$0 \rightarrow \phi$
	$1 \rightarrow 1$	$1 \rightarrow \phi$	$1 \rightarrow 1$	$1 \rightarrow \phi$	$1 \rightarrow 1$	$1 \rightarrow \phi$	$1 \rightarrow 1$	$1 \rightarrow \phi$	$1 \rightarrow 1$	$1 \rightarrow \phi$
	$2 \rightarrow 2$	$2 \rightarrow \phi$	$2 \rightarrow 2$	$2 \rightarrow \phi$	$2 \rightarrow 2$	$2 \rightarrow \phi$	$2 \rightarrow 2$	$2 \rightarrow \phi$	$2 \rightarrow 2$	$2 \rightarrow \phi$
	$3 \rightarrow 3$	$3 \rightarrow \phi$	$3 \rightarrow 3$	$3 \rightarrow \phi$	$3 \rightarrow 3$	$3 \rightarrow \phi$	$3 \rightarrow 3$	$3 \rightarrow \phi$	$3 \rightarrow 3$	$3 \rightarrow \phi$

Table 2: RFBL-based dictionary

6. DIAGNOSIS APPLICATION PROCESS

Subsequent to the identification of the defective chips through the manufacturing test application, the test responses of the defective chips can be utilized to pinpoint the cause of the failure. In this section, we present how the diagnosis application process is to be carried out based on the diagnostic information computed through the proposed analysis described in the previous section.

The diagnosis application process consists of two steps: the identification of the faulty module and the identification of the candidate fault set. In the first step, the modules that account for the RFBL in the defective responses are identified; in this step, only the RFBL-based diagnostic information is utilized. In the second step, the faults in the module that have been identified to be faulty are focused upon; based on both the RFBL-based and pass-fail diagnostic information computed by the proposed methodology, the faults that account for the defective behavior are identified.

6.1 Faulty Module Identification

The comparison of the precomputed RFBL-based module signatures with the actual primary output RFBLs of the defective chip helps pinpoint the faulty module. The candidate faulty module set is identified for every test vector; the intersection of the candidate faulty module sets constitutes the module containing the defect.

Table 2 provides the RFBL-based dictionary for the RT-level circuit in Figure 1. Let us consider that PO_1 and PO_2 RFBL values corresponding to the actual defective chip responses are 1 and 0 for the first test vector, 1 and 1 for the third vector, and ϕ and 3 for the fifth vectors, respectively, and that the second and the fourth vectors yield fault-free responses. The dictionary entries consistent with the RFBL values of the defective chip responses are provided in a larger font size. By referring to the reflected RFBL values (the integers to the right of the '$\rightarrow$' symbol) in the dictionary, the candidate faulty module set can be identified to consist of the MUX, MUL and $ADD1$ modules based on the defective response of the first vector, while the MUX, MUL, $ADD1$, and AND modules account for the defective response of the fifth vector. The third test vector, on the other hand, yields the primary output RFBLs that narrow down the candidate faulty module set to a singleton set consisting of MUL only, pinpointing MUL as the defective module.

6.2 Candidate Fault Set Identification

Subsequent to the identification of the faulty module, the faults in this module that account for the defective chip behavior should be pinpointed. In this process, both the precomputed RFBL-based and the pass-fail diagnostic information are utilized. The reader should note that the RFBL-based candidate fault set identification can be customized to any fault model through the integration of the corresponding fault simulator.

Initially the candidate fault set is computed by referring to the RFBL-based dictionary. Every test vector that yields a defective behavior pinpoints a group of faults that accounts for the defective behavior. The actual RFBL of the defective response is compared to the reflected RFBL values in the dictionary; the characterizing RFBL value of the grouped fault effects that account for the defective behavior for a test vector is thus identified. The actual faults are identified by performing a fault simulation on the corresponding module only; the faults that display an RFBL value on the module output that matches the characterizing RFBL value are placed in the candidate fault set. The candidate fault sets corresponding to the defect manifesting test vectors are intersected so as to identify the faults that account for the cause of the failure. Furthermore, the information regarding the test vectors that yields fault-free responses should be utilized as well for refining the candidate fault set. Based on the dictionary entries corresponding to these vectors, any fault that definitely reflects to at least one primary output bus should be excluded from the candidate fault set.

The example in the previous subsection helps illustrate the identification of the candidate fault set based on the RFBL-based dictionary. As the module that contains the defect is identified to be the MUL module, the target faults consist of the faults in this module only. By referring to the boxed dictionary entry corresponding to the MUL module and t_1 in Table 2, the characterizing RFBL value is identified as 0. Any MUL fault that is detected by the first vector and that has an RFBL value of 0 in the MUL output is placed in the candidate fault set, consequently; these faults can be identified by performing a fault simulation targeting only the MUL module. Similarly, the candidate fault sets are formed for the third and the fifth test vectors by focusing on the faults with an RFBL value of 0 and 3, respectively. The final candidate set is formed by intersecting these three candidate fault sets. It should be noted that the second and the fourth test vec-

Circuit	Proposed		Complete pass-fail
	RFBL info.	Compressed pass-fail	
GCD1	1.74	1.63	1.63
GCD2	1.23	1.16	1.16
MAC1	1.52	1.24	1.24
MAC2	1.56	1.31	1.31

Table 3: Diagnostic resolutions

tors yield fault-free responses. The dictionary entries corresponding to the second test vector and the MUL module suggest that the fault that accounts for the defect should be dormant under the application of t_2, as any activated fault in MUL manifests in at least one primary output bus when t_2 is applied. The faults that are detected by the second test vector at the MUL output are eliminated from the candidate fault set, consequently. Based on the dictionary entries corresponding to the fourth test vector and the MUL module, on the other hand, one can observe that a fault with an RFBL value of 3 manifests in none of the primary outputs. Any fault that yields an RFBL value differing from 3 is to be excluded from the candidate fault set.

If the resulting candidate set is a singleton set, the diagnosis application process terminates successfully, with the fault being reported as the cause of the failure; otherwise, further refinement is necessitated through the utilization of the precomputed pass-fail information. In the example illustrated in section 5.2, for instance, the candidate fault set may consist of f_1, f_2, and f_3 subsequent to the refinement based on the RFBL information. Based on the pass-fail information regarding t_1 and t_3 in Figure 3, one of these faults is identified as the actual cause of the failure, resulting in the successful termination of the diagnosis application process.

7. EXPERIMENTAL RESULTS

The proposed fault diagnosis methodology has been applied to the datapath portions of the MAC [11] and GCD [12] circuits, and the diagnostic resolution, storage size and run-time results have been compared to those of a pass-fail dictionary. Although the proposed methodology is independent of any specific fault model, for experimental purposes, diagnostic resolution levels are reported based on the stuck-at fault model.

The GCD and MAC circuits consist of 9 and 20 high level modules, respectively. By adjusting the bus width of these circuits, two distinct versions for each of the GCD and MAC circuits are designed; the benchmarks utilized in our experiments consist of GCD1 (8-bit buses), GCD2 (64 bit buses), MAC1 (4 bit buses) and MAC2 (16 bit buses) circuits. These benchmarks are selected for our experiments as they consist of datapath dominated arithmetic circuits.

Table 3 provides the diagnostic resolutions attained by the proposed methodology and a gate-level pass-fail dictionary; the diagnostic resolutions attained by the RFBL-based information only, RFBL-based information along with the compressed pass-fail information, and the complete pass-fail information are shown in the second, third and the fourth columns, respectively. It should be noted that even the sole use of the RFBL-based diagnostic information delivers near-perfect diagnostic resolution results. As the diagnostic loss of the RFBL-based information can be successfully restored by the compressed pass-fail information, the overall resolution attained by the proposed methodology is identical to that of the complete pass-fail dictionary.

Table 4 summarizes the comparison of our scheme and the gate level pass-fail dictionary approach; the storage sizes required for diagnostic information and computation times are compared. For the smaller circuits, the storage size required for a pass-fail dictionary doubles the size required for the proposed scheme. For the larger circuits, on the other hand, the compression ratios attained by the proposed methodology significantly increase; for the circuits MAC1 and MAC2, the ratio of the storage size corresponding to the proposed

Circuit	Pass-fail		Proposed	
	Size (bits)	Time (s.)	Size (bits)	Time (s.)
GCD1	$1.55 \cdot 10^5$	$1.62 \cdot 10^2$	$7.36 \cdot 10^4$	$2.30 \cdot 10^1$
GCD2	$3.92 \cdot 10^6$	$2.32 \cdot 10^3$	$1.85 \cdot 10^6$	$1.67 \cdot 10^2$
MAC1	$2.37 \cdot 10^6$	$6.86 \cdot 10^3$	$1.16 \cdot 10^5$	$1.84 \cdot 10^2$
MAC2	$1.41 \cdot 10^7$	$1.04 \cdot 10^5$	$1.21 \cdot 10^6$	$1.65 \cdot 10^3$

Table 4: Storage and runtime comparisons

scheme to that corresponding to a pass-fail dictionary equals 20.45 and 11.70, respectively. Even for the smaller circuits, significant speedups are attained by the proposed methodology over the pass-fail dictionary approach. The run-time speedups for the four circuits are 7.04, 13.90, 37.29 and 63.25, respectively. As the circuit sizes increase, the proposed RT-level diagnosis methodology is slated to outperform the gate-level diagnosis approaches, consequently.

8. CONCLUSIONS

In this paper, we have proposed an RT-level diagnosis methodology. The methodology we propose is not only computationally efficient but it also delivers diagnostic information that satisfies storage requirements as well. The proposed methodology is in compliance with any fault model, resulting in enhanced defect locating capabilities.

We present a module structural analysis for common RT modules that enables the identification of the appropriate fault grouping criteria. The fault effects are propagated through RT modules all in one scoop by tracing only the information representing them. Computation of the diagnostic information on a grouped fault effect basis promises to deliver not only computational efficiency but also diagnostic information of compressed sizes.

Perfect diagnostic resolution is ensured by the proposed methodology by storing the pass-fail information regarding a subset of the test set. The fact that the minimally sized subset is identified alleviates the storage requirements significantly.

We have applied the outlined methodology to several RT-level benchmark circuits. The experimental results indicate that the proposed grouping of fault effects delivers significant storage and runtime benefits and that these benefits are magnified as circuit sizes increase. The proposed RT-level fault diagnosis methodology therefore constitutes a highly promising solution for the next generation ICs.

9. REFERENCES

[1] V. Boppana and W. K. Fuchs, "Fault Dictionary Compaction by Output Sequence Removal", in *ICCAD*, pp. 272–279, 1992.

[2] V. Boppana, I. Hartanto and W. K. Fuchs, "Full Fault Dictionary Storage Based on Labeled Tree Encoding", in *VTS*, pp. 174–179, 1996.

[3] B. Chess and T. Larrabee, "Creating Small Fault Dictionaries", *TCAD*, vol. 18, n. 3, pp. 346–356, March 1999.

[4] D. B. Lavo and T. Larrabee, "Making Cause-effect Cost Effective: Low-resolution Fault Dictionaries", in *ICCAD*, pp. 272–279, 1992.

[5] I. Pomeranz and S. M. Reddy, "On the Generation of Small Dictionaries for fault location", in *ICCAD*, pp. 272–279, 1992.

[6] P. G. Ryan, W. K. Fuchs and I. Pomeranz, "Fault Dictionary Compression and Equivalence Class Computation for Sequential Circuits", in *ICCAD*, pp. 508–511, 1993.

[7] J. Lee and J. H. Patel, "An architectural level test generator for a hierarchical design environment", in *FTCS*, pp. 44–51, 1991.

[8] O. Sinanoglu and A. Orailoglu, "RT-level Fault Simulation Based on Symbolic Propagation", in *VTS*, pp. 240–245, 2001.

[9] J. Lee, E. M. Rudnick and J. H. Patel, "Architectural-level fault simulation using symbolic data", in *EDAC*, pp. 437–442, 1993.

[10] J. P. Hayes, *Computer Architecture and Organization*, McGraw-Hill Inc., 1998.

[11] P. Ashenden, *The Designer's Guide to VHDL*, Morgan-Kaufmann Publishers Inc., 1996.

[12] Y. Makris, I. Bayraktaroglu and A. Orailoglu, "Invariance-Based On-line Test for RTL Controller-Datapath Circuits", in *VTS*, pp. 459–464, 2000.