Efficient RTL Coverage Metric for Functional Test Selection

Jian Kang
jkang@cse.unl.edu

Sharad C. Seth
University of Nebraska - Lincoln, seth@cse.unl.edu

Vijay Gangaram
Intel Corporation, vijay.gangaram@intel.com

Efficient RTL Coverage Metric for Functional Test Selection

Jian Kang and Sharad C. Seth
Dept. of Computer Science & Engineering
University of Nebraska - Lincoln
Lincoln, NE, USA
jkang, seth@ cse.unl.edu

Vijay Gangaram
Design Technology Solutions
Intel Corporation
Folsom, CA, USA
vijay.gangaram@intel.com

Abstract

For performance-critical microprocessors, efficient test-selection methods are needed for reusing a subset of functional validation tests to detect manufacturing defects. Our new input/output transition fault-coverage metric (TRIO) at the register-transfer level is shown to perform much better than current metric in test selection at only an incrementally higher computational cost. TRIO may also be used for testability analysis early in the design cycle.

1. Introduction

Semiconductor devices are becoming increasingly complex in terms of transistor count, frequency and integration. Emerging design styles coupled with aggressive design methods pose significant challenges for testing. These challenges include testing for manufacturing defects as well as speed binning of devices such as microprocessors. Functional tests are derived manually or through automated test generation techniques [1-3] in design validation phase. They exercise the design and build confidence that the design matches specification. In addition to design validation, functional tests can also be reused for manufacturing test. Studies have shown additional fallout using functional tests even for test set with high structural coverage [4, 5].

The number of functional tests is normally large. Due to test time and tester memory limitations, only those tests that provide good manufacturing defect screening value are added to production test tape. The process of selecting a subset of tests from a pool of functional test sequences is called functional test selection.

Exact methods for test selection such as fault simulation of the entire test suite are not practical due to computational costs. Even though it is preferable to use register-transfer-level (RTL) coverage metrics, existing RTL metrics either do not establish the correlation with gate-level fault models [6-10], or require expensive fault simulation [11-14]. Current approaches to functional test selection for manufacturing testing are ad-hoc and often use structural coverage metrics such as toggle coverage [15], which gives suboptimal results. We propose a new RTL coverage metric which is simple yet very effective with a low computational overhead. This coverage metric can be used in evaluating functional tests for high volume manufacturing (HVM) as well as in early testability analysis. Another recently published functional coverage metric [16] monitors events during logic simulation. However, defining events comprehensively for adequate coverage by automated means is an unsolved problem.

The rest of the paper is organized as follows. Section 2 covers the necessary background on the test selection problem. Section 3 introduces the proposed coverage metric. Sections 4 and 5 show test selection results for ISCAS89 benchmarks and industrial circuits respectively. Section 6 concludes the paper and identifies future extensions.

2. Background

We assume that a pool of tests is available for selection, where each test in the pool consists of a sequence of test vectors. As mentioned, test selection is the problem of selecting a subset from the pool of tests. This is done with the help of some coverage metric (say, M) using a two-step process [17]:

1. Evaluate the coverage of each test in the pool according to M.
2. Select the smallest number of tests that cover all faults covered by the whole set.

An optimum selection in the second step involves solving the set-cover problem which is known to be

NP-complete, hence a greedy heuristic is used to iteratively select a test with the highest incremental coverage. When there is no further coverage improvement with respect to metric M, we say that test selection using M saturates. This is the most M could help in selecting tests. The outcome of the greedy approach is an ordered set of selected tests.

The goodness of the test selection can be measured along three dimensions: the quality of selected tests, the time spent in test selection, and the number of selected tests. The quality would ideally be measured as actual defect coverage but for practical reasons traditional measures, such as gate-level stuck-at or transition fault coverage, are used as proxies. Clearly, all three dimensions of goodness are important and can be traded off depending on the context. While test time and data volume are important concerns, they are typically optimized on the tester as long as they can fit within capacity targets. Generally, as long as the selection time is affordable, we prefer a metric that does not saturate too soon, and provides the highest defect coverage.

In addition, because functional tests are available for RTL designs, the metric should preferably be evaluated at the RTL, so that the test selection can take place before gate-level net lists are generated. Such an early coverage metric could also identify testability holes early in the design cycle.

RTL toggle coverage has been used as an approximation of gate-level fault coverage [18], and has been used for test selection because of its small overhead [17]. However, the toggle coverage does not take propagation into account and its correlation with stuck and transition fault models is limited. When used in test selection, it saturates too early and results in low coverage. VVG [12] extends the toggle metric to include propagation but requires RTL fault simulation which is expensive. Later works extend the fault model, by considering not only RTL line stuck-at faults, but also condition stuck faults [13], and additional stuck-at faults inside the blocks whose structures are unknown at RTL [14]. In this paper, we concentrate on an efficient technique that has the same order of complexity as logic simulation. These more expensive metrics could be used if our interest is in bringing in more exactness.

3. The proposed metric

Our proposed metric also extends the toggle at the RTL. However, in contrast to VVG, we do not introduce additional variables and add only partial observability. Further, unlike VVG, we preserve the transition aspect of the toggle so that a single measure can capture both the static and dynamic faults. Guided by these considerations, we formally define our extension next.

3.1 TRIO Metric

Definition 1: The Input/Output TRansition (TRIO) fault model is defined with respect to a subset S of the RTL variables of an RTL module. S consists of primary inputs, primary outputs, and state variables (i.e. registers and latches) of the module. A TRIO fault is a pair (<V_i, T_i>, <V_j, T_j>), where V_i is one bit of a primary input or a (current) state variable, V_j is one bit of a primary output or a (next) state variable, T_i is rising or falling transition on V_i and T_j is a rising or falling transition on V_j. Further, there exists a combinational path from V_i to V_j with the correct polarity so that the transition T_i on V_i can cause the transition T_j on V_j. In TRIO model, we ignore clock signals.

The TRIO faults can be represented in graphical form. In this graph, a node represents a bit with transition <V_i, T_i> and an edge from node <V_i, T_i> to <V_j, T_j> means that signal transition T_i on V_i could cause transition T_j on V_j. Each edge in the graph represents a TRIO fault.

Example 1: For the RTL circuit module shown in Figure 1, which is a modulo-4 counter of signal on input A, the bits of interest are: adder inputs A, S[1], S[0], and adder outputs N[1], N[0]. From the function of the circuit, we know that a rising transition on A will increment the counter state but a falling transition will not. Similarly two successive 1s on A will cause successive increments of the counter state. From this analysis we can obtain all possible TRIO faults in this circuit.

The TRIO graph for the example circuit is shown in Figure 2. There are a total of ten TRIO faults in this circuit, as represented by the ten edges in TRIO graph.

```
module count (reset, clk, A, S, N);
    input reset, clk, A;
    output [1:0] S;
    reg [1:0] S;
    wire [1:0] N;
    assign N = S + A;
    always @(posedge clk)
        if (reset = 1) S<=2'b00;
        else S <= N;
endmodule
```

Figure 1. RTL description of the example circuit

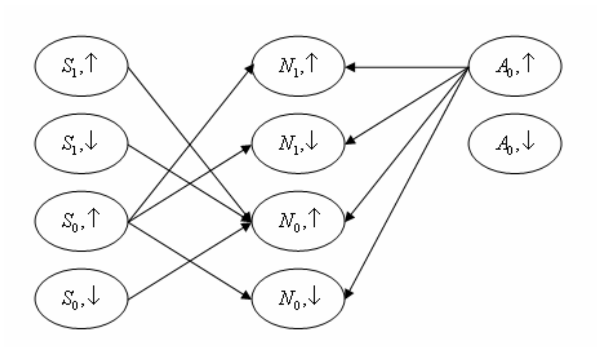

Figure 2. A graphical representation of TRIO faults

The TRIO graph in Figure 2 reflects the functional constraints of the circuit. For actual circuits, we may not know or be able to derive all functional constraints. Fortunately, for the test selection application it is sufficient to determine the absolute coverage of functional tests being compared, which does not require explicit construction of the TRIO graph. The absolute coverage of a test can be determined from analyzing the results of its functional simulation in conjunction with the knowledge of bits that are directly connected through a combinational path. The latter can be generated efficiently from the parse tree built by an RTL compiler.

Now, consider the coverage of TRIO faults by a test. For the example circuit, assume that the initial counter state to be zero when the test sequence 011110 is applied to input A of the circuit. The result of simulating this test on the circuit is shown in Table 1.

Table 1. Simulation trace for a given test

Time	0	1	2	3	4	5
A	0	1	1	1	1	0
S[1:0]	00	00	01	10	11	00
N[1:0]	00	01	10	11	00	00

There is a rising transition T_a on signal A from time frame 0 to 1, accompanied by a rising transition T_{n0} on signal N[0]. Because A is the only signal in the support set of N[0] that have changed during this cycle, transition T_{n0} must be caused by transition T_a. In this case, the cause effect relationship between $<A,\uparrow>$ and $<N_0,\uparrow>$ is trivial, the TRIO fault $(<A,\uparrow>,<N_0,\uparrow>)$ is covered.

In general, the cause-effect relationship is harder to deduce. Transition T_j may depend on multiple input transitions as well as the bits that remain stable. Because TRIO is intended to be a fault model at the RTL, the definition of TRIO coverage needs to be based on the function and not the structure of the circuit. Accordingly, the exact definition of the cause-relation relationship would be defined from a subset of inputs to an output. Every subset of changed inputs that could cause the Boolean difference on function V_j w.r.t. this subset to be true would be a cause for the change on V_j. However, for this computation, both the fault list and fault-evaluation time would be exponential. On the other hand, the alternative of crediting only single-input changes in V_j's fanin cone is unduly pessimistic, ruling out T_i causing the change in combination with other changing bits. In the current version of TRIO, we take an optimistic interpretation, by simply checking that V_i is in the support set of the function on V_j:

Definition 2: A TRIO fault ($<V_i, T_i>$, $<V_j, T_j>$) is covered if we see transitions T_j and T_i in the simulation trace, T_j occurs one cycle later than T_i if V_j corresponds to a state variable, otherwise, the two transitions occur in the same time frame, and V_i is in the support set of V_j.

Considering the simulation trace in Table 1, from time frame 2 to 3, there are simultaneous transitions in signals S[1] and S[0], and rising transition on N[0]. Under our definition, we say that both TRIO faults $(<S_1,\uparrow>,<N_0,\uparrow>)$ and $(<S_0,\downarrow>,<N_0,\uparrow>)$ are covered. For the given input sequence, 6 of the 10 TRIO faults are covered.

The definition of the TRIO fault model was guided by considerations that strike a balance between our desire for a functional metric at the RTL and the need for accuracy and computational feasibility. As compared to the computationally-efficient toggle fault model, TRIO is stricter in requiring not only that V_i toggles but also that the toggling be propagated to V_j. This eliminates the problem of early saturation with the toggle coverage. For example, the input sequence in Table 1 covers all the toggle faults in the circuit. TRIO does not stipulate toggle propagation all the way to primary outputs because this will be equivalent to defining an RTL stuck-fault model and require expensive fault simulation [12]. Consideration of implementation-independence guided us in restricting the TRIO definition to bits corresponding to primary inputs, primary outputs, and registers. At the same time, we require signal sensitization paths from every input of a combinational block to all reachable outputs, with the expectation of covering a large fraction of lines in any structural implementation. TRIO model also ensures that faults on these lines are further propagated to the block outputs. This results in a better correlation of the TRIO metric with structural models.

Two other fault models in the literature are apparently similar to TRIO. The double-transition fault (DTF) [19] approximates path delay faults by transitions between all pairs of $<g_1, g_2>$ of connected gates in the circuit. It requires robust path propagation of the transition from g_1 to g_2 and from g_2 to a primary output. These requirements limit the use of the DTF to the gate-level and to implicit evaluation of coverage because of the huge fault list.

The coupling fault (CF) model [20] is also defined by an input/output pair. However, detection of a CF requires application of all vectors that satisfy the Boolean difference of the output w.r.t. to the input, which is called the coupling test set (CTS). CF model is extended to cover delay faults by requiring that all adjacent pairs of vectors in the CTS must be applied. These pairs correspond to the subset of all single-input change (SIC) pairs in CTS that yield different outputs. The twin requirements of SIC and all pairs were shown to be useful in generating realization-independent robust path-delay tests, but they are unduly pessimistic for coverage evaluation.

3.2 Evaluation of TRIO Metric

TRIO-coverage evaluation could either be integrated tightly with or carried out after logic simulation. The second option may be slower but was preferred in our work because of its ease of implementation and independence from logic simulation. During logic simulation we capture the simulation trace on the bits of interest and post-process it to get the TRIO fault coverage. The latter involves determining at each signal-change step whether the associated TRIO fault is covered according to the cause-effect relationship described above. The total time for TRIO-based fault evaluation is the sum of the time for logic simulation and post-processing.

3.3 An Extension of TRIO

For comparison, we implement an extended version of TRIO, called E_TRIO, which employs a stricter cause-effect relationship in its fault definition and includes observability to a primary output. The following steps summarize the E_TRIO implementation.

1. From the circuit description, obtain the list of E_TRIO faults, which is identical to that of TRIO faults.
2. For each E_TRIO fault (<V_i, T_i >, < V_j, T_j >), inject a transition fault at V_i , according to the direction of T_i .
3. Using V_j as the observation point, do transition-fault simulation for the injected transition fault at V_i and record the cycles at which this transition fault is detected at V_j. This gathers the information about E_TRIO fault excitation.
4. Inject a transition fault at V_j in each cycle when the transition fault on V_i was detected at V_j and use a transition-fault simulator to determine if the newly added transition fault is detectable at an observable output. This step gathers information about E_TRIO fault propagation.
5. By combining the result of E_TRIO fault excitation and propagation, we determine if the E_TRIO fault is detected.

As the cost of E_TRIO evaluation is quite high it is only feasible for small circuits.

4. Experiments on ISCAS89 benchmarks

In the absence of an available test pool of functional tests for ISCAS89 sequential benchmark circuits, for each circuit we generated a test set using sequential ATPG [21] and augmented it with random tests. We used the test generation tool repeatedly, targeting at a single stuck fault each time, to generate a set of short tests instead of a single long one. The stuck fault coverage of our test set is not as high as reported using a single test [21], since each test in our test set starts from the unknown state, increasing the test generation difficulty. This was done to generate validation-like independent test sequences. For TRIO evaluation, every circuit was considered as a single block with the state and I/O signals visible. For the toggle coverage, we could have used the same signals, but chose the gate-level instead so as to compare TRIO against a measure that performed better in terms of not saturating too early and correlating better with the gate level fault models.

Test selection was carried out for the toggle, TRIO and E_TRIO metrics using the two-step process described in Section 2. In addition, to compare the quality of selected tests, we also did test selection using the reference metrics, i.e. gate-level stuck-at fault model and transition fault model. Tests selected using each of the above five metrics are then evaluated for their gate-level stuck-at coverage and transition coverage respectively. We didn't include big circuits because of the E_TRIO evaluation cost. Table 2 summarizes the results. The details of the test pools are shown in columns 2 to 4. Column 2 is the number of tests in each test pool. Columns 3 and 4 show the total stuck-at and transition coverage respectively of each test pool. Although the goal of test selection is to maximize the fault coverage on standard HVM fault models (e.g. stuck-at and transition), coverage loss can be expected for models that are different from the standard model used as the reference. Columns 5, 6 and 7 show the stuck-at coverage loss in test selection, respectively, for the toggle, TRIO and E_TRIO

Table 2. Test selection results for ISCAS89 benchmarks

Ckt	Test Pool			Stuck Fault Coverage Loss				Transition Fault Coverage Loss			
Name	# of tests	Stuck cov.	Tran cov	toggle	TRIO	E_ TRIO	Tran	toggle	TRIO	E_ TRIO	Stuck
s298	378	78.11	13.75	10.95	1.78	2.67	8.29	5.20	0.55	0.00	0.55
s820	1371	94.43	55.49	38.98	18.91	13.92	4.87	25.86	14.44	8.08	3.41
s832	1380	92.74	54.12	35.82	20.97	14.74	4.53	26.94	12.05	9.72	3.57
s1196	1735	96.80	96.04	24.46	9.07	3.52	0.78	42.92	14.99	7.03	7.96
s1238	1761	92.10	91.67	17.80	8.11	3.08	1.07	33.17	14.99	6.33	8.11
s1423	1379	83.36	33.58	16.15	2.88	0.66	4.08	13.22	1.91	1.61	3.11
s1488	1846	96.27	67.92	19.94	7.00	3.74	0.87	27.73	6.28	4.31	4.13
s1494	1833	95.59	67.07	20.79	5.06	2.76	0.52	24.15	7.01	5.03	2.98
AVG				**23.11**	**9.22**	**5.64**	**3.13**	**24.90**	**9.03**	**5.26**	**4.23**

metrics. For completeness we also show the stuck-at coverage loss for the transition fault model (column 8). Similarly, columns 9-12 display the loss in the transition fault coverage for the toggle, TRIO, E_TRIO and stuck-at metrics, respectively. In all cases TRIO achieved higher coverage (both stuck-at and transition) than toggle and, in many cases, TRIO did as well as E_TRIO.

Table 3 shows the number of tests selected by each metric. Due to early saturation, the toggle metric selects the least number of tests in all cases. For example, for s298, toggle only selects three tests, while both TRIO and E_TRIO select closer to the number of tests selected by the reference (stuck-at) metric.

Table 3. Number of selected tests

Name	stuck	tran	E_TRIO	TRIO	toggle
s298	9	4	10	9	3
s820	56	45	37	20	5
s832	55	48	39	23	5
s1196	54	89	82	22	5
s1238	58	97	82	23	7
s1423	37	37	82	99	6
s1488	31	42	53	19	3
s1494	36	42	51	18	3
AVG	**42.00**	**50.50**	**54.50**	**29.13**	**4.63**

5. Experiments on industrial circuits

TRIO was also evaluated against toggle for test selection on two real industrial circuits, called F and S. These circuits are data path blocks in the execution cluster of an x86 CPU design. The whole cluster is about 20 times the size of block S. Table 4 shows the number of collapsed stuck-at faults for each circuit.

The test pool consisted of 1,010 functional tests for the whole cluster derived from micro-architectural and system level validation. The length of each test ranges from 100K to a few million vectors. Many of these tests have good fault coverage in some portion of the cluster. The goal of the experiment was to see if the TRIO metric could effectively mine this test pool and select a subset of tests with only a small coverage loss for faults in these blocks. Although the computational cost of gate-level fault simulation was quite high (see Table 5), we again wanted to include the test-selection results for stuck-at and transition faults as reference. However, E_TRIO runs could not be finished on these circuits because of excessive time. Both logic and fault simulation runs were performed at the cluster level on dual-core Pentium machines running on Linux.

Table 4. Industrial circuit blocks

Block Name	# of collapsed stuck-at faults
F	5599
S	27754

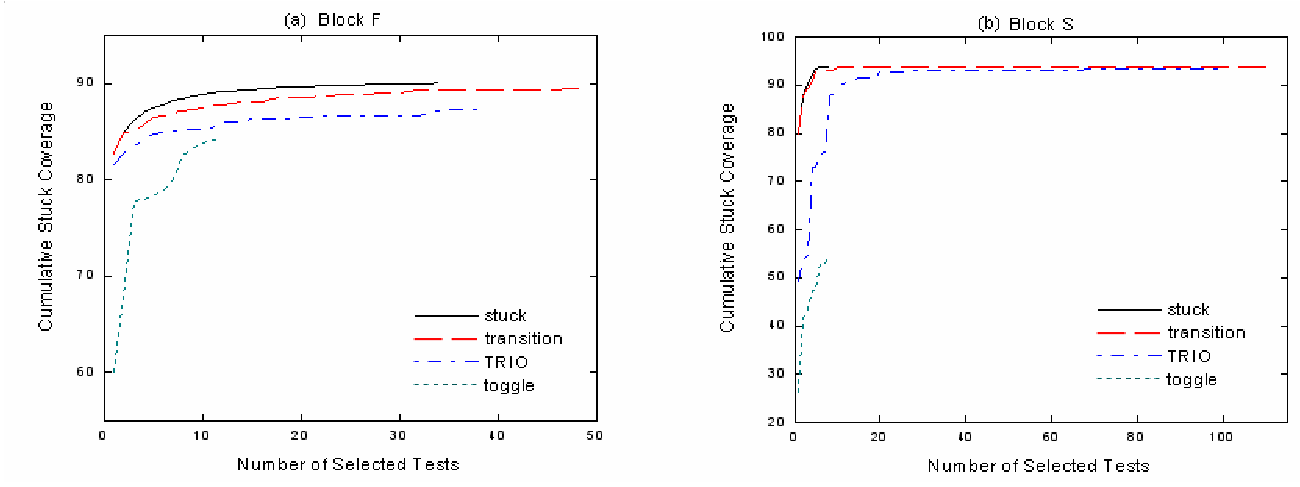

Figure 3. Test selection for stuck coverage

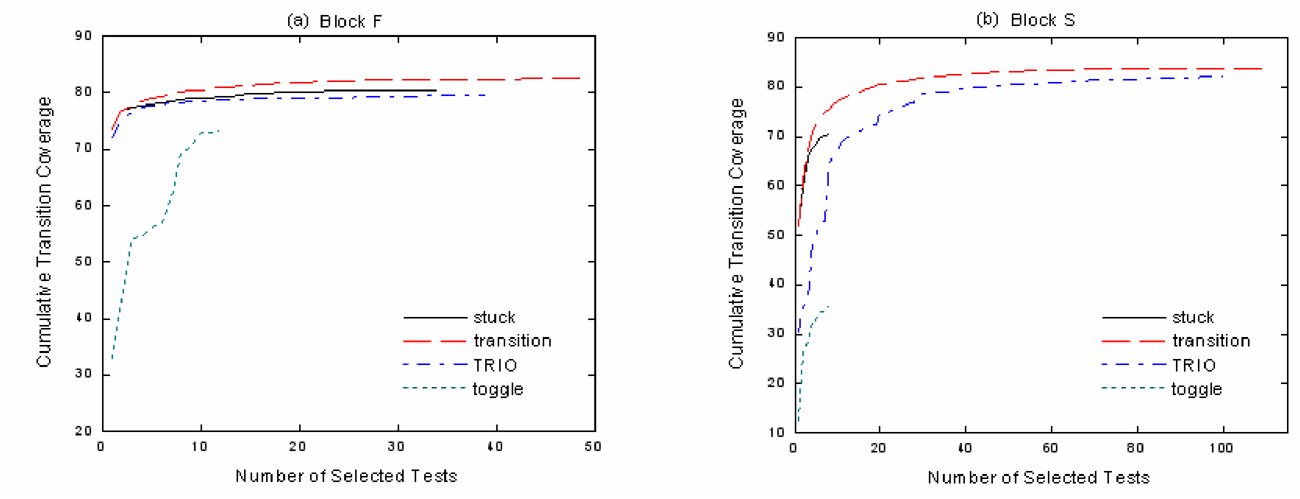

Figure 4. Test selection for transition coverage

Figures 3 and 4 show the cumulative stuck-at and transition fault coverage of tests selected by stuck-at, transition, TRIO and toggle. Note that the tests selected by a particular metric is invariant across the two figures; only the evaluation criterion (stuck-at vs. transition fault coverage) is changed. The results are consistent with those observed on ISCAS89 circuits. The TRIO based test selection achieves higher stuck-at and transition fault coverage than toggle based test selection. The issue of premature saturation with toggle metric is again apparent on both the circuits. Furthermore, tests selected using TRIO consistently have a higher coverage than toggle-selected tests, for any given number of tests. While the TRIO based selection achieves most of the stuck-at and transition fault coverage, the stuck based selection on circuit S fails to reach high transition fault coverage. The results on both circuits highlight the effectiveness of TRIO over toggle. Further, TRIO is almost as effective as stuck-at (transition) for selecting tests with high transition (stuck-at) coverage.

Table 5 shows the average and maximum time to evaluate functional tests. Since the simulation is performed at cluster level, logic simulation time is about the same for both blocks, where the difference is caused by tracing and dumping different set of signals. The TRIO metric involves logic simulation and post processing the simulation trace on RTL signals. Transition fault simulation run times are similar to that of stuck-at fault simulation. The toggle simulation time is roughly the same as for logic simulation. As can be seen, TRIO evaluation has very small computational overhead and can be subsumed as part of RTL simulation that is part of design validation effort. On the other hand, the fault-simulation for block S is already high, therefore, fault simulating the whole cluster or circuit would be impractical.

Table 5. Computational cost

Block Name		F	S
Avg Time (sec)	Stuck Fault Sim	5715.14	33539.25
	Logic Sim	1222.01	1196.63
	Post - process	1.68	5.51
Max Time (sec)	Stuck Fault Sim	73161.00	525169.00
	Logic Sim	15341.00	15839.00
	Post-process	21.00	112.00

6. Conclusion and future work

We have described an efficient RTL coverage metric, TRIO, and shown its effectiveness for solving a practical problem of functional test selection for high volume manufacturing. The proposed metric has very small computational overhead and it is easy to incorporate into existing RTL simulation flows used in design validation. Results on both ISCAS89 and industrial circuits show that it can be used to effectively mine validation tests for HVM.

We are investigating ways to improve the accuracy of TRIO metric without adding substantially to the cost of evaluation. We may be able to improve upon the criterion for fault excitation when multiple inputs in the support set of an output have transitions. First, we could implement a more sophisticated linear-time cause-effect analysis than used in TRIO. Second, we could assign weights, based on functional characteristics, to edges in the TRIO graph and estimate the TRIO coverage as a weighted sum. We may be able to include inexpensive graph-based measures to add fault propagation to TRIO that is not as expensive as E_TRIO. We also plan to study the relationship of TRIO with other delay fault models such as robust path delay model. Finally, we would like to explore other applications of TRIO, including early testability analysis.

References

[1] P. Parvathala, K. Maneparambil, and W. Lindsay, "FRITS - a microprocessor functional BIST method," in International Test Conference, 2002.

[2] F. Fummi, G. Pravadelli, and F. Toto, "Coverage of formal properties based on a high-level fault model and functional ATPG," in European Test Symposium, 2005.

[3] M. B. Santos, F. M. Goncalves, I. C. Teixeira, and J. P. Teixeira, "RTL-based functional test generation for high defects coverage in digital SOCs," in European Test Workshop, 2000.

[4] P. Maxwell, I. Hartanto, and L. Bentz, "Comparing functional and structural tests," in International Test Conference, 2000.

[5] P. Nigh, W. Needham, K. Butler, P. Maxwell, and R. Aitken, "An experimental study comparing the relative effectiveness of functional, scan, IDDq and delay-fault testing," in VLSI Test Symposium, 1997.

[6] C. Chang Hyun and J. R. Armstrong, "B-algorithm: a behavioral test generation algorithm," in International Test Conference, 1994.

[7] Y. V. Hoskote, D. Moundanos, and J. A. Abraham, "Automatic extraction of the control flow machine and application to evaluating coverage of verification vectors," in International Conference on Computer Design, 1995.

[8] G. Al Hayek and C. Robach, "From specification validation to hardware testing: a unified method," in International Test Conference, 1996.

[9] M. Karunaratne, A. Sagahayroon, and S. Prodhuturi, "RTL fault modeling," in Midwest Symposium on Circuits and Systems, 2005, Vol. 2.

[10] F. Fallah, S. Devadas, and K. Kuetzer, "OCCOM: efficient computation of observability-based code coverage metrics for functional verification," in Design Automation Conference, 1998.

[11] W. Mao and R. K. Gulati, "Improving gate level fault coverage by RTL fault grading," in International Test Conference, 1996.

[12] P. A. Thaker, V. D. Agrawal, and M. E. Zaghloul, "Validation vector grade (VVG): a new coverage metric for validation and test," in Proceedings 17th IEEE VLSI Test Symposium, Dana Point, CA, USA, 1999.

[13] F. Fummi, C. Marconcini, and G. Pravadelli, "Functional fault coverage: the chamber of secrets or an accurate estimation of gate-level coverage?," in Ninth IEEE European Test Symposium, 2004.

[14] M. B. Santos, F. M. Goncalves, I. C. Teixeira, and J. P. Teixeira, "Implicit functionality and multiple branch coverage (IFMB): a testability metric for RT-level," in International Test Conference, 2001.

[15] C. J. Stoucny, R. Davies, P. McKernan, and T. Truong, "Alpha 21164 manufacturing test development and coverage analysis," Design & Test of Computers, IEEE, vol. 15, 1998.

[16] S. Park, L. Chen, P. Parvathala, S. Patil, and I. Pomeranz, "A functional coverage metric for estimating the gate-level fault coverage of functional tests," in International Test Conference, 2006.

[17] V. Gangaram, D. Bhan, and J. K. Caldwell, "Functional Test Selection Using Dynamic Untestability Analysis," in International Workshop on Microprocessor Test and Verification, 2006.

[18] D. Drako and P. Cohen, "HDL Verification Coverage," in Integrated System Design, 1998.

[19] I. Pomeranz, S. M. Reddy, and J. H. Patel, "On double transition faults as a delay fault model," in Great Lakes Symposium on VLSI, 1996.

[20] J. Yi and J. P. Hayes, "The Coupling Model for Function and Delay Faults," Journal of Electronic Testing: Theory and Applications, vol. 21, 2005.

[21] L. Xijiang, I. Pomeranz, and S. M. Reddy, "MIX: a test generation system for synchronous sequential circuits," in International Conference on VLSI Design, 1998.