JOURNAL OF ELECTRONIC TESTING: Theory and Applications, , 1–12 ()
© Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

On Efficiently Producing Quality Tests For Custom Circuits in PowerPC™* Microprocessors

LI-C. WANG
Somerset PowerPC Design Center, Motorola Inc.,
6200 Bridgepoint Parkway, Bldg 4, Austin, Texas 78730
lwang@ibmoto.com

MAGDY S. ABADIR
Somerset PowerPC Design Center, Motorola Inc.,
6200 Bridgepoint Parkway, Bldg 4, Austin, Texas 78730
abadir@ibmoto.com

Received February 10, 1999; Revised June 17, 1999

Editor: Anthony Ambler

Abstract.
Custom circuits, in contrast to those synthesized by automatic tools, are manually designed blocks of which the performance is critical to the full chip operation. Testing these blocks represents a major challenge and a crucial time-to-market factor in PowerPC microprocessor design flow. This paper investigates various methodologies for testing custom blocks. Issues of efficiently obtaining proper circuit model for ATPG tools as well as producing quality tests will be analyzed and discussed. Tradeoffs among various methods will be analyzed and compared. Experience and results based on recent PowerPC microprocessors will be reported.

Keywords: Custom Circuits, ATPG, High Level Circuit Extraction, DFT, Time-To-Market

1. Introduction

With the increasing complexity of today's microprocessors, the demands of highly effective test and design-for-testability techniques have attracted interests of every microprocessor design team. Among various design blocks in a microprocessor, custom circuits, such as embedded arrays and data computation units, represent one of the major testing challenges. Testing these blocks efficiently is critical since in modern RISC microprocessors they may occupy more than half of the on-chip area.

Custom circuits are designs of which the performance is critical to the speed of full chip operation. Usually, they are manually designed and tuned at the transistor level. This results in complex structures that can be hard to test, and also quite different from how they are specified at the RTL level.

Today, commercial ATPG tools operate mostly at the gate level. The required gate-level test models are typically derived from either transistor

*IBM and PowerPC are trademarks of IBM Corporation in the United States, or other countries, or both

level models or the corresponding RTL specification. Hence, with possible structural mismatch between the transistor level implementation and the RTL specification, different gate level models may be constructed. Then, it becomes questionable if one approach can produce better test models than the other with respect to test quality and the ATPG efficiency.

In the industry, test quality is commonly measured by stuck-at fault coverages, while it should actually be measured by realistic defect coverages. Previous research suggested that a high fault coverage may not necessarily imply a high defect coverage due to the mismatch between the stuck-at fault model and the actual defects (ex. [14]). Various ATPG approaches were thus proposed to overcome that problem, including using a combination of different metrics [12], detecting faults multiple times [13] [8], reducing test generation biases on a given fault model [18], etc. For custom circuits, the mismatch between faults and defects adds another dimension to the problem of producing quality tests [6], in addition to the possible structural mismatch already existing in the generation of gate-level test models.

This paper investigates the correlation between test model generation and test pattern generation with respect to the detection of *non-target* faults (faults that are not explicitly targeted for test generation) and implicitly to the detection of real defects. Although quality is our focus, important factors that affect time-to-market, such as engineering efforts, run time, test length, etc., will also be addressed and discussed. Tradeoffs among various approaches will be analyzed on custom circuits from recent PowerPC microprocessors. Based upon our results and experiences, a cost-effective methodology for testing custom circuits will be suggested.

The paper is organized as follows. In Section 2, we introduce basic design methodology for PowerPC microprocessor. In Section 3, we present two basic approaches for creating gate-level test models. Section 4 discusses critical factors that impact cost and time-to-market, such as ease of use, test model size, test length, and engineering efforts. Section 5 presents the initial comparison results. In Section 6, we analyze various issues associated with test quality in more detail, and in Section 7, we suggest several improvements to the ATPG process for enhancing test quality. Section 8 reports final experimental results and Section 9 summarizes the paper.

2. Design Methodology Overview

At Somerset, a design is represented by two different views (Figure 1). They are:

- the high-level RTL view, and
- the transistor-level schematic view.

Each of the two views represents design data for its particular set of tools and applications. For example, the RTL view is the most abstract, in the sense that it is the high-level specification of a design, where all functionality is described using high-level constructs. In a normal design flow, the RTL model for a module is first derived. Extensive functional simulation at the full-chip level is first conducted to verify the correctness of the RTL specification. Then, the actual design is implemented as a transistor-level schematic view. The implementation is usually carried out via synthesis tools. However, due to the limitations of the tools, the implementation may be done manually for certain selected timing-critical blocks.

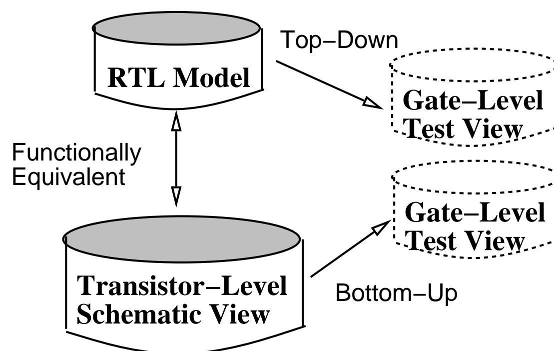

Fig. 1. Two Major Design Views

In this methodology, additional gate-level views are required for the purpose of automatic test pattern generation. For a manually-implemented custom block, the gate level model can be generated based upon either an RTL model or a transistor level schematic model.

Note that in a design cycle, it is imperative to maintain functional equivalence among all data views/models for a design, as each model may be modified and adjusted for fitting into its particular applications over time. For examples, a DFT engineer may modify a gate level view for achieving a higher fault coverage and a designer may adjust a schematic view for enhancing its performance. Maintaining functional consistency among the RTL, gate, and transistor-level schematic views relies on continuous application of boolean equivalence checking tools as well as other formal validation techniques [9] [10] [16].

3. Generation of Gate Level Models

Basically, two approaches are possible for the generation of gate-level test models. A *top-down* approach generates gate level model from the RTL view. At Somerset, RTL level designs are based upon a restricted *Verilog*-like language. Converting each RTL statement into a gate level netlist can be done in a rather straightforward fashion. In general, the synthesis process involves various optimization steps, such as using logic optimization techniques [2] for area and/or delay minimization. Furthermore, to avoid degradation of testability, testability/test-set preserving logic transformations can be also employed during the optimization process [1] [3].

However, for custom circuits in PowerPC microprocessors, we employ a rather straightforward synthesis process for the generation of gate level models from the RTL. The approach allows ATPG activities to start as soon as an RTL model becomes available. Also, there are no sufficient evidence indicating that sophisticated optimization techniques would result in superior gate level models with respect to ATPG efficiency and test quality. Therefore, it is more logical to adopt a simpler top-down approach so that valuable engineering time and efforts can be saved for other purposes.

Alternatively, a *bottom-up* approach obtains gate level model from the schematic view. Various methods for extracting high level models from transistor level circuits have been proposed by researchers, including the symbolic analysis approaches [4] [5] [17] and the path-oriented analysis approach [11]. For test, the key benefit offered by a bottom-up method is the structural consistency between the resulting gate level model and its actual implementation. Since the implementation of a custom circuit may be dramatically different from its RTL specification, one interesting question to ask would be "Does the structural consistency help to produce better quality tests (for defects)?"

Note that extraction to higher level models from transistor level circuits is crucial in many applications in addition to ATPG. These include applications in diagnosis, verification, and simulation. In diagnosis, the benefit of the structural consistency is clear since it allows working on a representation that is structurally resemble to the actual design, and can be more efficiently handled. In verification, such as equivalence checking where an RTL specification is checked to be equivalent to its transistor level implementation, a higher level model can be extracted from the transistor level implementation and then compared functionally to the RTL model. In simulation, an extracted gate level model can be processed more efficiently.

4. Factors That Affect Cost and Time-To-Market

This section analyzes the two fundamental approaches for obtaining gate-level test models in terms of several factors that impact cost and time-to-market. These factors are:

- the sizes of the generated gate level models and their resulting fault sizes, ATPG run time, and test lengths,
- the manual efforts involved (developing the test models, verifying the tests, DFT, etc.), and
- the stability of the methodology.

4.1. Size, Run Time, and Test Length

For custom blocks, it is not unusual that designers have to trade area (space) for time in order to reach high enough performance. Therefore, it is natural for an extracted gate level model to have a size much larger than a gate level model produced based upon its RTL specification.

Schematics usually contain many layers of hierarchy. In contrast, an RTL view for a custom block may not. Often, the extraction is done for each individual module independently, preserving the same hierarchy in the gate level model. The hierarchy facilitates the DFT work afterwards. Another reason why a flattened circuit is not preferred during extraction is that its size can increase dramatically, and incur significant overhead during the process.

There are other possibilities to incur unnecessary size overhead during the extraction process. For example, suppose the "path-oriented" extraction method is used. The following example illustrates one of the cases. Figure 2 shows a design for implementing a 1-bit adder. Without the precharging, the intended function for this design is specified below.

$$p = ab,\ q = a \oplus b, \text{ and } z = \overline{a}\overline{b}$$

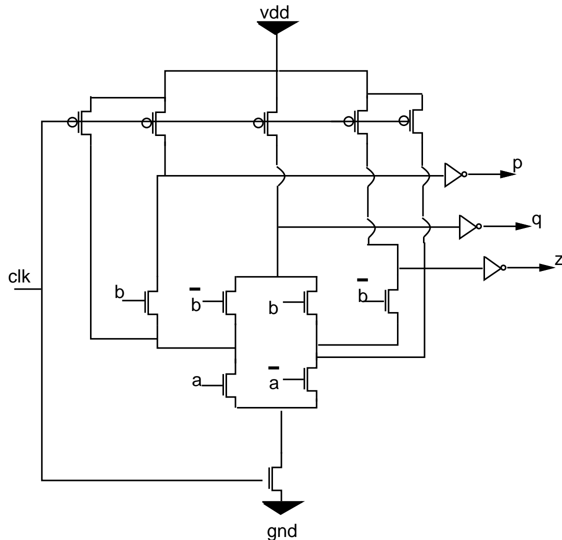

Fig. 2. Schematic for a 1-bit Add Element

To extract the gate level function for the output p, we enumerates all possible paths from p to the ground. This results in two different paths: $p \longrightarrow b \longrightarrow a \longrightarrow gnd$ and $p \longrightarrow \overline{b} \longrightarrow \overline{b} \longrightarrow b \longrightarrow \overline{a} \longrightarrow gnd$. Hence, the resulting gate level model would be "$p = (clk)ba + (clk)b\overline{b}\overline{a}$." Note that when analyzing only this particular cell, the extraction tool *has no way to know that* $\overline{a}$ *and* $\overline{b}$ *may actually be complemented from the signals* a *and* b*, respectively.* Therefore, the gate level models extracted would be the following.

From schematics:

$$p = (clk)ba + (clk)b\overline{b}\overline{a},$$
$$q = (clk)\overline{b}a + (clk)b\overline{a}, \text{ and}$$
$$z = (clk)\overline{b}\overline{a} + (clk)\overline{b}ba$$

From RTL:

$$q = (clk)ab,$$
$$q = (clk)(a \oplus b), \text{ and}$$
$$z = (clk)(\overline{a}\overline{b})$$

The above example indicates that extraction of independent cell in a hierarchy may introduce unnecessary extra logic due to its unawareness of the input constraints from the surrounding circuits. Hence, a sophisticated extraction tool needs to perform cross-cell analysis as well as some logic optimization to avoid this problem. This implies a higher complexity for tool development.

A larger gate level model typically results in more faults, a longer ATPG run time, and more test vectors produced. More faults and a longer run time may be worthwhile if a higher test quality is achieved. However, the restriction on the test length is more rigid. Test length is usually bounded by tester memory size that directly affects test cost. Given a design as big and complex as a microprocessor, our experiences show that efforts were always required to make a proper trade-off between test length and test cost. Hence, smaller gate-level test models may be preferred for that particular reason.

4.2. Manual Efforts

In a design cycle, manual DFT efforts are always required for achieving better gate level models with respect to increasing fault coverages and reducing test lengths, simply because no tool is perfect. No matter which way to take, top-down or bottom-up, manual efforts are required. However, intuitively a bottom-up approach represents

a much more complicated problem than the translation of RTL to gate. Hence, without a surprise, our experience indicates that a bottom-up approach requires much more manual efforts than the top-down method.

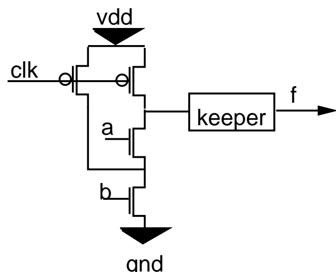

Fig. 3. Example of single end domino without foot device

For example, there can be cases where an extraction tool fails to produce proper gate level models. Typically, if extraction is done without flattening the circuit as stated above, the path-oriented approach would be used. First, sequential (state-holding) elements are required to be found and isolated. Moreover, consider an example circuit shown in Figure 3. Without knowing the relation between clk and a,b, it is not possible to determine that as $clk = 0$, if f would be precharged to 1 or not (since the values of a and b during precharge are unknown and they also depends on the strength configuration as well). Then, the best guess may be $f = \overline{ab}$ while in the RTL it is actually specified as $f = \overline{(clk)ab}$. The circuit is implemented this way because the designer has the knowledge that during precharge phase, $a = b = 0$. However, the extraction tool has no way to capture that input constraints automatically without performing a cross-cell analysis. If $f = \overline{ab}$ is produced for this cell, ATPG may fail to generate proper tests to detect stuck-at faults associated with a and b (depending on how the signal clk is set). As a result, manual intervention is required.

4.3. Methodology Stability

The top-down methodology is more stable. Test activities can start right after the RTL level view becomes available. On the other hand, with a bottom-up approach, DFT engineers have to wait until the actual implementation is ready. Although in both cases, changes made in their original views result in changes in the gate level models as well, RTL view tends to stabilize much earlier in a design cycle (it has to be, because the RTL serves as the functional specification of a design). On the contrary, last-minute changes can always happen on the actual implementation due to various requirements such as timing and power. Hence, more efforts will be required if a bottom-up approach is adopted.

4.4. The Rule-Of-Simplicity

In a microprocessor design center where profit and time-to-market are crucial, we have to follow the **rule-of-simplicity**" which states that *until a more complicated approach can be shown to yield significant advantages, the simpler one should be adopted.* Discussion in this section indicates that a bottom-up approach requires a sophisticated tool, often results in larger gate level models, and usually requires more engineering efforts. Hence, such an approach will not be preferred if it does not provide significant benefit to the enhancement of test quality.

5. Test Quality — Initial Experiments

For the initial experiments, six custom circuits (arbitrarily named as A,B,C,D,E,F) were selected from recent PowerPC microprocessors. For each case, a commercial ATPG tool was applied on both an extracted gate level model and an RTL-synthesized gate level model. During extraction, we used an in-house tool based on the path oriented approach and the extraction process preserved the design hierarchy. In the top-down method, we used another in-house tool which implemented the straightforward synthesis process mentioned above.

5.1. Fault Sizes

Table 1 shows the number of stuck-at faults for each circuit in each gate level model. From our previous analysis, it is not a surprise to see that

extracted gate level models (from the bottom-up method) tend to result in larger fault sizes. This also indicates that the resulting gate level models are larger.

Table 1. Fault (Circuit) Size (# of Faults)

Circuit	Top-Down	Bottom-Up
A	220	285
B	906	1233
C	4410	5441
D	1200	3015
E	2042	4610
F	6620	14980

Table 2. Initial Test Lengths

Circuit	Top-Down	Buttom-Up
A	20	28
B	37	60
C	75	106
D	28	132
E	31	190
F	87	595

5.2. Test Length and Stuck-At Fault Coverage

Table 2 shows the test lengths to achieve the corresponding stuck-at fault coverages in Table 4. In Table 4, the difference between fault and test coverages is that test coverage does not take the redundant faults into account. Since extracted gate level models contain much more faults, it is natural for the ATPG tool to generate more tests. Notice that extracted gate level models are in general harder to test (lower fault coverages), but the difference is not significant.

However, the difference in test length is significant. For the three large circuits, D,E,F, the difference can be as many as six times! This raises the concern of possibly exceeding the tester memory limitation and/or involving too much cost.

5.3. Quality Comparison at The Transistor Level

To evaluate test quality, we select stuck-at faults at the transistor level as our *non-target* faults. These faults are not explicitly targeted for test generation and hence, detecting them is fortuitous. This selection may be biased in favor of the extraction approach. However, the key idea here is that if the two approaches do not differ much with respect to this evaluation of quality, then based on the "rule-of-simplicity" described earlier, the top-down approach will be favored because it is simpler.

Note that although extracted gate level model is structurally closer to the real implementation, there are still many faults at the transistor level, which may not be modeled at the gate level. For example, consider the circuit in Figure 4. The stuck-at 0 fault shown in one of the precharge line cannot be properly modeled by a stuck-at fault in the gate level test view. This is because the extracted gate level test view would be combinatorial. However, to detect the particular stuck-at 0 fault shown, it requires first setting $clk = 0, a = 0, c = 0$ and then $clk = 1, b = 0, c = 1$. In the normal operation, the output z will have a value 0 (suppose the keeper inverts its incoming value) during the first pattern, and stay as 0 during the second pattern. In the faulty case, since node n will be charged to 0 instead of 1 during the first

Table 4. Fault Coverage

Circuit	Top-Down Method		Buttom-Up Method	
	Fault Coverage	Test Coverage	Fault Coverage	Test Coverage
A	84%	100%	96%	100%
B	97.6%	100%	80.0%	88.4%
C	98.1%	100%	92.7%	98.3%
D	85.5%	100%	81.9%	89.3%
E	82.0%	93.4%	81.2%	96.6%
F	81.1%	100%	82.5%	100%

pattern, as clk and c both turn to 1 in the second pattern, the value 0 at node n will trigger the keeper and result in a value 1 for z. This example illustrates the fact that *fortuitous detection* [6] is required to detect transistor level stuck-at faults.

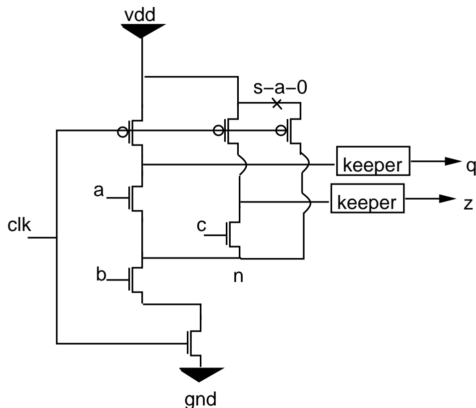

Fig. 4. Fault example that requires fortuitous detection

Table 3. Number of Transistor Level Faults

Circuit	# of Transistor Level Faults
A	1168
B	3636
C	12522
D	4976
E	9774
F	30084

Table 3 lists the number of transistor level faults in each circuit. and Table 5 presents the coverage results. Note that at transistor level, lower coverages are expected. Two types of the coverage are listed, one for true detection and the other for true plus potential detection. A true detection is defined the same way as in the traditional sense. For a potential detection, it means that in the faulty machine, one or more outputs have the unknown value X while in the good machine, they should be 0 or 1, or vice verse.

Clearly, ATPG based on the bottom-up extracted model tends to achieve better results for most circuits. However, this is *not* without the additional cost — test length, extra effort, etc. Note that in the case of circuit B, ATPG on the RTL gate model was able to get a higher coverage.

6. Factors That Affect Test Quality

Issues of quality for testing custom circuits are threefold: *the fault coverage that can be achieved*, *the mismatch between faults and defects*, and *the mismatch between gate level model and the actual implementation*. Historically, the single stuck-at fault model is used for test pattern generation. Today, this fault model remains the most cost-effective solution for testing microprocessor designs. Based upon this fact, two factors are crucial for affecting the outcome of test quality. First, due to the mismatch between faults and defects, catching defects relies on *fortuitous detection* [6]. Because of that it was suggested in the previous research that more "randomness" should be given to the test selection process in order to enhance the fortuitous detection of non-target faults and defects [7] [8] [13]. Another important observation is that a gate level representation can bias the test selection process. To illustrate the idea, consider an example shown in Figure 6. The intended function for this transistor network is the following.

$$F = (clk)(a + d)(b + e)(c + f)$$

Table 5. Transistor Level Fault Coverage Results

	Top-Down Method	Bottom-Up Method
A	16.5%, 34.5%	19.5%, 47.8%
B	45.1%, 73.6%	35.4%, 66.2%
C	28.9%, 58.2%	29.5%, 62.8%
D	14.3%, 30.3%	18.5%, 44.9%
E	10.0%, 33.2%	18.1%, 48.6%
F	9.40%, 29.9%	13.3%, 48.1%

Note: data X,Y represent "true detection", "true + potential detection"

Suppose a straightforward path-oriented analysis generates a gate level view as below without optimization (the RTL gate level model would be the same as the RTL specification above).

$$F = (clk)abc + (clk)abf + (clk)aec + (clk)aef + (clk)dbc + (clk)dbf + (clk)dec + (clk)def$$

Consider the fault "a stuck-at 0" shown in Figure 6. In the RTL gate model, possible tests to detect the fault are shown below.

a	d	b	e	c	f
1	0	0	1	0	1
1	0	0	1	1	0
1	0	0	1	1	1
1	0	1	0	0	1
1	0	1	0	1	0
1	0	1	0	1	1
1	0	1	1	0	1
1	0	1	1	1	0
1	0	1	1	1	1

In the un-optimized extracted gate model, since a has four fanouts, there are four stuck-at 0 faults on the fanouts. Those faults requires tests as the below.

a	d	b	e	c	f
1	0	0	1	0	1
1	0	0	1	1	0
1	0	1	0	0	1
1	0	1	0	1	0

Note that in the extracted model, detection of any one of the stuck-at 0 fault on a fanout of a will also detect the stuck-at 0 fault on the stem a. Hence, stuck-at faults on the stem a may never be considered for test generation. As a result, for the four stuck-at 0 faults on the four fanouts of a, exactly the same four tests shown above will be produced.

Observe from the above analysis that there are 9 possible tests if the RTL gate level model is used. Without further restriction, each of the nine tests will have equal probability to be selected. On the other hand, by using the extracted gate level model, only 4 out of the nine tests are possible to be selected. For example, it will never consider the test $abcdef = 101111$ (which is actually a test for the a stem). In other words, the test selection process will be more *biased.* For defects that require the five unreachable tests, fortuitous detection never happen. On the other hand, if the same number of tests (4) are allowed on the RTL gate model for a stuck-at 0, then each defect will have a probability of 0.38 $(1-(8/9)^4)$ if each test is selected with an equal chance from all 9 tests, or 0.44 $(1-5/9)$ if no test is selected more than once.

7. ATPG Adjustments For RTL-Generated Gate Level Model

Recall that a bottom-up extraction approach results in larger models and more tests. Suppose the same number of tests is allowed for each of the corresponding RTL gate model test cases. Is it possible that we can further improve the ATPG process on the RTL gate models so that test quality can be brought up to the same levels? For that purpose, we suggest two simple ATPG improvements.

Multiple Stuck-at Fault Detection

As described earlier, multiple stuck-at fault detection has been shown to have the advantage of detecting additional non-target faults/defects [13]. Thus, a straightforward improvement would be repeating the ATPG process (but make random decisions to produce different tests for the same fault) to achieve multiple fault detection.

Unbiased Testing

Another approach could be adopting the "unbiased" ATPG approach presented in [18]. The main idea is that during the ATPG process, the fault that has been detected the least number of times so far should have a greater chance to be selected as the target fault for test generation in the next round [8]. Also, during the test generation processes, special effort is made to maintain a balanced distribution of 0's or 1's across all sites, i.e. for each site the probability of receiving a 0's (or 1's) is made as close to 0.5 as possible.

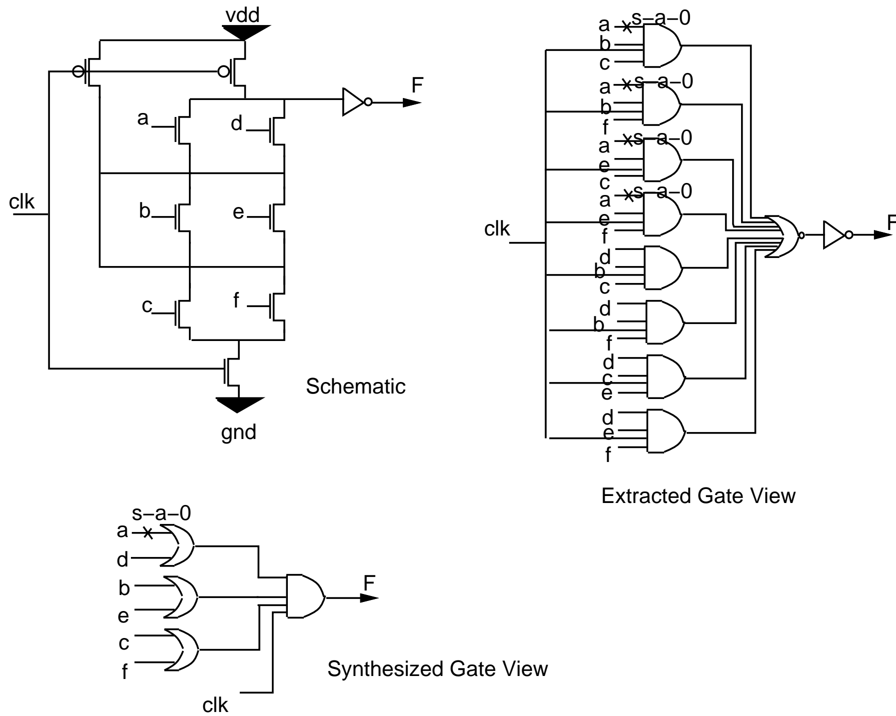

Fig. 6. Illustration of Representation Bias

7.1. Test Generation Methods

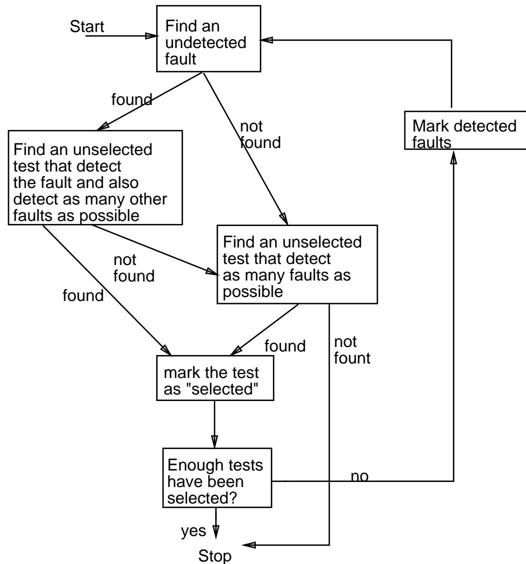

Fig. 5. Illustration of postprocessing step

Based on the two ideas above, four additional methods are implemented and experimented.

ATPG-dual: In this method, we use an additional commercial ATPG tool and append the two test sets. Note that test lengths remain smaller than those associated with the bottom-up method. Also, duplicate tests may exist in the appended test set.

ATPG-Post: The purpose of this method is to force a commercial ATPG tool to produce more tests. It contains the following four steps.

1. Run ATPG tool with one fault at a time.
2. Remove duplicated vectors.
3. Perform fault simulation without fault dropping. This gives information on fault detection for each test.
4. (Postprocessing) Follow the method in Figure 5 to select tests up to the desired length.

ATPG-UB: The commercial ATPG tools we use do not provide enough flexibility to emulate an unbiased ATPG approach described in [18]. For

that purpose, we used an in-house experimental ATPG tool called ATPG-UB.

ATPG-UB$\frac{1}{2}$: In this case, we still apply ATPG-UB but the test length is restricted to only half of that originally produced in ATPG-UB.

8. Final Results and Analysis

Table 6 presents the final results. For comparison purpose, we copy the results on the extracted models from Table 5 to Table 6. Columns 2 and 3 show results from ATPG-Post and the unbiased ATPG approach, ATPG-UB, where *test lengths are the same as those used for column 5.* In column 4 test sizes are reduced by half from column 3. In the table, entries with "—" indicate experiments that were not run because the test length differences shown in Table 2 are not significant.

As it can be observed, the original ATPG process based on the extracted gate level models does not effectively utilize the additional test vectors for detecting more transistor level stuck-at faults. Data in columns 2 to 4 demonstrate that *by improving the ATPG process slightly, it is possible to achieve better quality based upon the much simpler top-down generated gate level models.*

Note that for the bottom-up cases, using any of the improved ATPG methods does not really make sense because that would further increase the test lengths significantly. Before all faults are detected at least once, ATPG-UB behaves more like a regular ATPG. Hence, in order to guarantee multiple detection for all faults, more tests will be required.

8.1. Note on ATPG-UB

One of the key feature in ATPG-UB is that it directs any additional test(s) to the currently hard-to-detect faults. In other words, while regular ATPG on the extracted gate model spends its effort equally on all faults from a larger fault set, ATPG-UB works harder on a smaller fault set in order to allocate more resources to the hard-to-detect faults. Figure 7 illustrates the key difference between these two approaches. The figure includes two curves of the fault detection statistics. If we consider faults appearing on the right-hand side as hard-to-detect faults, then the figure tells that the average detection times of those hard-to-detect faults (in the RTL gate model) by ATPG-UB is higher than that by a regular ATPG (in the extracted gate model).

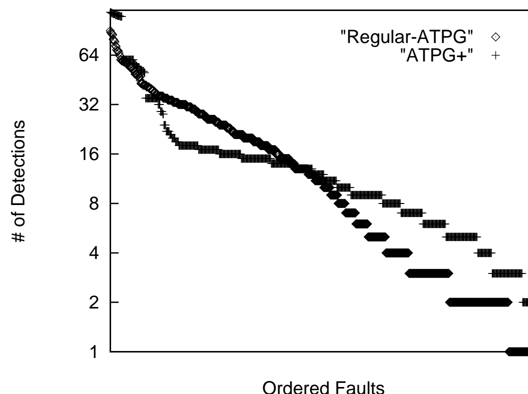

Fig. 7. Distribution of Fault Detection

9. Conclusion

We conclude that although a bottom-up extracted gate level model can structurally resemble to the actual implementation, tests generated based on this model may not necessarily be more effective. With minor adjustments in the ATPG, we demonstrate that a functionally-equivalent, top-down generated gate level model is able to produce more effective tests. A bottom-up approach requires higher tool complexity, greater test generation and application cost, and more additional engineering efforts. Following the "rule-of-simplicity," for testing custom circuits we thus suggest to use a top-down approach in conjunction with one of the ATPG improvements presented for achieving maximal cost effectiveness. Success of such a methodology has been demonstrated through experiments on custom circuits from recent PowerPC microprocessors.

Note that the top-down approach is not applicable if a synthesizable RTL level model does not exist in a design flow. Then, generating a proper RTL level requires additional cost and efforts, which may negate the benefits of the top-down method. However, this is not the case in our design methodology.

Table 6. Transistor Level Fault Coverage Results

	ATPG with Top-Down Method				ATPG with Bottom-Up Method
	Col. 1 ATPG-Dual	Col. 2 ATPG-Post	Col. 3 ATPG-UB*	Col. 4 ATPG-UB$\frac{1}{2}$**	Col. 5 ATPG-Bottom-Up*
A	—	—	26.2%, 51.2%	—	19.5%, 47.8%
B	—	—	48.2%, 74.7%	—	35.4%, 66.2%
C	—	—	46.2%, 70.4%	—	29.5%, 62.8%
D	15.2%, 34.9%	19.4%, 47.8%	45.9%, 61.3%	41.9%, 57.0%	18.5%, 44.9%
E	12.9%, 38.7%	17.0%, 46.2%	42.9%, 57.1%	39.2%, 52.8%	18.1%, 48.6%
F	10.2%, 32.2%	16.5%, 46.0%	37.2%, 55.8%	36.6%, 54.0%	13.3%, 48.1%

Note: data X,Y represent "true detection", "true + potential detection"
*Test lengths in ATPG-Post and ATPG-UB are the same as ATPG-Bottom-Up
**ATPG-UB$\frac{1}{2}$ means reducing the test lengths by half from ATPG-UB

Future work can be directed into various areas. First, a more detailed comparison can be made based on other types of fault model, such as bridgings at the transistor level. In additions, we will be working with commercial ATPG vendors to allow more randomness during the test selection process (i.e. producing different tests for the same fault). Also, for circuits with performance driven redundancy, an extracted gate level model will preserve the redundancy while an RTL synthesized gate level model may not. Impact of such redundancy on test quality should be carefully analyzed further.

References

1. M. J. Batek and J. P. Hayes, *Test-Set Preserving Logic Transformations*, Proc. 29th Design Automation Conference, 1992, pp. 454-457.
2. R. K. Branton, R. Rudell, A. Sangiovanni-Vincentelli, A. R. Wang, *MIS: A Multiple-Level Logic Optimization System*, IEEE Trans. on CAD, Vol 6., November 1987, pp. 1062-1081.
3. M. J. Bryan, S. Devadas, and K. Keutzer, *Testability-Preserving Circuit Transformations*, Proc. ICCAD, 1990, pp 456-459.
4. R. E. Bryant, *Extraction of Gate Level Models from Transistor Circuits by Four-Valued Symbolic Analysis*, Proc. ICCAD 1991, pp. 350-353.
5. R. E. Bryant, *Boolean Analysis of MOS circuits*, in IEEE Trans. on CAD, July, 1992, pp. 434-469.
6. K. M. Bulter and M. R. Mercer, *Quantifying Non-Target Defect Detection by Target Fault Test Sets*, Proc European Test Conference, 1991, pp. 91-100.
7. E. B. Eichelberger et. al., *Structured Logic Testing,* Prentice Hall, 1991, Sec. 15.
8. M. R. Grimaila et. al., *REDO — Random Excitation and Deterministic Observation — First Commercial Experiment*, in Proc. VLSI Test Symposium 1999.
9. Neeta Ganguly, Magdy S. Abadir, and Manish Pandey, *PowerPC Array Verification Methodology Using Formal Verification Techniques*, International Test Conference, Washington DC., 1996. pp. 857-864.
10. Charles H. Malley and M. Dieudonne, *Logic Verification Methodology for PowerPC Microprocessors*, 32nd Design Automation Conference, 1995, pp. 234-240.
11. Sandip Kundu, *GateMaker: A Translator to Gate Level Model Extractor for Simulation, Automatic Test Pattern Generation, and Verification*, in Proc. International Test Conference 1998, pp. 372-381
12. P. C. Maxwell, R. C. Aitken, V. Johansen, and I. Chiang, *The Effectiveness of Iddq, Functional, and Scan Tests: How Many Fault Coverages Do We Need?*, Proc. International Test Conference, 1992, pp. 168-177.
13. S. C. Ma, P. Franco, and E. J. McCluskey, *An Experimental Chip To Evaluate Test Techniques, Experiment Results*, in Proc. International Test Conference 1995, pp. 663-672.
14. J. Park, M. Naivar, R. Kapur, M. R. Mercer, and T. W. Williams, *Limitations in Predicting Defect Level Based on Stuck-at Fault Coverage*, Proc. VLSI Test Symposium, 1994.
15. Carol Pyron, Javier Prado, and James Golab, *Next Generation PowerPC Microprocessor Test Strategy Improvements*, in Proc. International Test Conference 1997, pp. 414-423
16. C,J. H. Seger and Randal E. Bryant, *Formal Verification By Symbolic Evaluation of Partially-Ordered Trajectories*, Formal Methods in System Design 6, 1995, pp. 147-189.
17. K. J. Singh and P. A. Subrahmanyam, *Extracting RTL Models from Transistor Netlists*, Proc. ICCAD 1995, pp. 11-17.
18. Li-C. Wang, Ray Mercer, and T.W. Williams, *Using Target Faults To Detect Non-Target Defects*, in Proc. International Test Conference 1996, pp. 629-638

Li-C. Wang received the B.S. degree with honors in Computer Engineering from National Chiao-Tung University, Taiwan in 1986, the M.S. degree in Computer Sciences in 1991 and Ph.D. degree in Computer and Electrical Engineering in 1996, both from the University of Texas at Austin. Currently, he is an assistant professor in the Computer Engineering Group of Department of Electrical Engineering at Texas A & M University, College Station. He also works part-time as a member of the tool and methodology technical staff at the Somerset PowerPC Design Center, Motorola, Inc. Prior to that, Dr. Wang worked at Somerset, Motorola for 3 years and worked at the Mathematics Research Center of Bell Labs, Murray Hill, New Jersey for the summers of 1991 to 1995. Dr. Wang's research interests lie in the areas of design for testability, high-level test generation, design validation, and formal verification.

Magdy S. Abadir received the B.S. degree with honors in Computer Science from the University of Alexandria, Egypt in 1978, the M.S. degree in Computer Science from the University of Saskatchewan, Saskatoon, Canada, in 1981, and the Ph.D. degree in Electrical Engineering from the University of Southern California, Los Angeles, in 1986. Currently he is the Chief Technologist for verification and Manager of the Test and Logic Verification Methodology and Tools group at Motorola's PowerPC Design Center (Somerset) in Austin, Texas. Prior to that he was the General Manager of Best IC Labs in Austin Texas (a Burn-in and Test Engineering firm). From 1986 to 1994 he worked at the Microelectronics and Computer Technology Corporation (MCC) as a senior member of the technical staff. Dr. Abadir has co-founded and chaired a series of international workshops on the economics of design, test and manufacturing and on microprocessor test and verification. He has co-edited several books on those subjects, and he also published over 70 technical journal and conference papers in the areas of test economics, design for test, computer-aided design, high-level test generation, and design verification and economics.