# Realizing High Test Quality Goals with Smart Test Resource Usage

**Xinli Gu, Cyndee Wang,**
**Abby Lee, Bill Eklow**

Cisco Systems, Inc.
170 W. Tasman Drive
San Jose, CA 95134, USA

**Kun-Han Tsai, Jan A. Tofte,**
**Mark Kassab, Janusz Rajski**

Mentor Graphics Corporation
8005 S.W. Boeckman Road
Wilsonville, OR 97070, USA

### *Abstract*

*Growing ASIC design sizes and advanced deep sub-micron technologies require new fault models and more test vectors to meet high test quality goals. To realize these goals within given test resources and cost constraints, new DFT techniques must be used. This paper reports test quality metrics and the test cost of industrial designs for different fault models using three DFT techniques: ATPG for deterministic patterns, Logic BIST for pseudo-random patterns, and EDT for compressed deterministic patterns. It is shown how these techniques can be used to achieve the high quality goals within the test resources currently available for stuck-at tests.*

## 1. INTRODUCTION

Manufacturing test programs include a number of test sets designed to effectively screen defective parts. The tests typically include functional tests, parametric tests, Iddq, logic test, and memory built-in self test (MBIST). Logic test is predominant in terms of tester memory and time requirements, and will be the focus of this paper.

Shrinking geometries make new types of defects, such as resistive vias and bridges between signal lines, more common [1,2]. This creates the need for additional tests targeting new fault models. At-speed tests targeting transition delay faults are effective at screening defects that affect design performance. Multiple-detect patterns, where multiple tests are generated for each fault, have also been proven in silicon to be effective at screening out bridging defects and unmodeled defects [3].

The ever-growing design sizes continue to increase the volume of test data and test time required to achieve high quality goals. The need for additional tests targeting the new fault models exacerbate the problem. Transition delay tests have been observed to contain at least two to five times as many test patterns as those required for stuck-at tests. Multiple-detect patterns can additionally multiply the scan data volume depending on the number of targeted detects.

In general, it is impossible to apply all aforementioned tests given test resource limitations. At-speed and multiple-detect test sets are often truncated or not applied at all, which may negatively impact quality.

This paper introduces a test generation flow that uses a combination of several DFT techniques: Logic Built-In Self Test (LBIST) for pseudo-random test and Embedded Deterministic Test (EDT) for compressed deterministic test [4,5]. An incremental ATPG approach with coverage gradient analysis for test vector distributions among all test sets is also presented to improve the efficient tester resource usage. Multiple-detect concept and the test quality metrics are applied to both stuck-at and transition fault models to achieve high test quality. It is shown that tester resource saving due to the use of the compressed deterministic test makes it possible to target new test models, such as at-speed transition faults and bridging faults within the available test resource constraints.

The paper is organized as follows: Section II covers the fault models used beyond stuck-at for high-quality tests. Section III presents the metrics which will be used to assess the quality of test sets. Section IV describes the test generation flow for different test and fault models such that the pattern count is optimized. Section V reports the experimental results.

## 2. FAULT MODELS FOR HIGH-QUALITY TESTS

### 2.1 Delay fault models

Research has shown that the traditional stuck-at fault test is no longer sufficient to maintain low DPM (defects per million) goals at deep sub-micron geometries as defects which require at-speed tests become more prevalent [1,6]. At-speed testing therefore becomes a necessity to achieve high test qualities for such deep sub-micron processes. Two fault models are commonly used for at-speed testing: the path delay fault model and the transition delay fault model. When using the path delay fault model, the user is required to specify paths to be targeted, usually a subset of critical paths derived from a static timing analysis tool. Path delay test is typically used for design performance characterization and speed binning [7,8]. Today's high-frequency designs involve aggressive optimization by the synthesis and layout tools to maximize the operating frequency. Consequently, a large portion of the design is in the critical-path domain. A subset of these paths is provided to ATPG as it is not practical to target all paths. Since the paths derived from static timing analysis may not be testable,


0-7803-8580-2/04 $20.00 Copyright 2004 IEEE 

iterations between the static timing analysis tool and the ATPG tool may be required to generate a path delay test set.

On the other hand, the transition fault model is more robust and easier to use since no path information needs to be provided. The ATPG tool targets each pin in the circuit and ensures that the pin is tested for slow-to-fall and slow-to-rise transitions that are observed by at least one observation point. The main drawback of the transition fault model compared to the path delay model is that it provides far more flexibility in choosing the fault propagation path, and does not guarantee that the path is long enough to detect a speed-related defect.

### 2.2 Beyond traditional fault models

In general, there are two directions to improve the test quality beyond the fault models described above. One direction is to enhance the fault model by describing the defect behavior and presenting it in a suitable form to an ATPG tool. In this case, the faults can be modeled more precisely, but the additional complexity may result in a much larger fault list. Advanced fault models, e.g. bridging faults and crosstalk effects, use physical layout information to compile a targeted fault list. A complete example of this approach is given in [9]. The possible bridges are identified by layout analysis using weighted critical areas, and their behavior is modeled by different types of faults and a special netlist. The experimental results show that patterns created using this approach detected unique defective parts that were not detected by a high-coverage stuck-at test set. This approach, however, requires substantial infrastructure and it is difficult to target all the bridging defects. In [9], only 10% of the top 400K bridges were targeted, achieving approximately 27% coverage of the node-to-node bridges. In addition, such a test generation approach creates dependency on the physical design and may cause additional delay before the design is taped out. The ATPG algorithms for such advanced fault models may be complex and may increase the test generation time significantly. Furthermore, the exact mechanism of various defects is very often unknown, which makes the modeling itself a challenging task. In the worst case, the ATPG tool may have very precise but insufficiently robust targets to work with.

An alternative approach is to utilize conventional fault models, such as stuck-at and transition delay fault models, and apply the same ATPG algorithm to generate different patterns to detect the same faults multiple times. This will increase the probability of detecting other unmodeled defects without using layout information. This approach is referred to as multiple-detect ATPG. There are several proposed methods that target stuck-at faults multiple times [10,11,12,13]. Multiple-detect test sets using the stuck-at fault model have been proven in silicon production to be effective at detecting defects that other tests, e.g., functional and Iddq tests, did not detect [3]. The study shows that the defects uniquely detected by the multiple-detect stuck-at test set are typically static bridging defects which require additional conditions to be activated.

### 2.3 Multiple-detect for at-speed testing

Multiple-detect ATPG can be extended to the at-speed transition delay model. By using random decision order during ATPG, a fault targeted multiple times will likely be observed through different paths. Timing-related defects caused by high resistance bridging faults or dynamic bridging faults can therefore benefit from multiple-detect ATPG. Figure 1. illustrates, as an example, the importance of using multiple-detect ATPG for at-speed testing. In this case, the faulty site $f_1$ has 5 observation points $O_1$, $O_2$, $O_3$, $O_4$, and $O_5$, where only $O_5$ has enough delay to observe the fault effect. For single-detect pattern generation of transition faults, the test generator only ensures that the fault effect propagates to a single observation point. If the selected observation point is not on a sufficiently delayed path, the fault effect will escape detection. Multiple-detect of at-speed transition delay fault testing increases the detection probability.

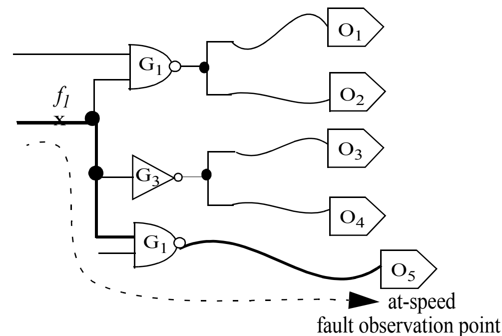


**Figure 1: Using multiple-detect ATPG for delay faults**

## 3. MEASURING THE QUALITY OF A TEST SET

Traditional ATPG test quality is measured by test coverage, or 1-detect test coverage. When multiple-detect test patterns are used, the number of detections should also be measured as part of the test quality. To correlate the test coverage and the number of detections to some fault models other than stuck-at and transition fault models, several metrics are introduced. They are Bridging Coverage Estimate (*BCE*) and Fault Observation Coverage (*FOC)* [3]. In [3], only *BCE* was used directly to compare the quality of test sets. The calculation of *FOC* is very time-consuming. It is only used to demonstrate that the multiple-detect test set generated using the random decision order ATPG can propagate a fault through different observation paths for each detection. For pseudo-random pattern fault simulation, where the number of patterns is very large, the calculation of *FOC* becomes a performance bottleneck.

In this paper, the quality of a test set is measured by *BCE* and *FOE* as described below, in addition to the traditional test coverage for a given fault model.

### 3.1 Metric for bridging defects

The first metric, *BCE*, is used for static defects as well as for at-speed defects. We assume that each defect has 50% probability of correlating a defect to a fault model to be used with multiple-detect pattern generation. It can be adjusted if the probability of the defect activation condition is known.

**Definition:** Given a test set *T* and target fault list *F*, the Bridging Coverage Estimate (*BCE)* is calculated as follows:

$$BCE = \sum_{i=1}^{n} \frac{f_i}{|F|} \cdot (1 - 2^{-i})$$

where $f_i$ is the number of faults detected *i* times by *T*, and $|F|$ is the total number of faults in the target fault list *F*. *n* is the maximum number of times that a fault can be detected by *T.* In practice, *n* is the maximum number of detections that an ATPG tool keeps track of. Once a fault is detected *n* times, it will be dropped from the target fault list. The calculated *BCE* could vary depending on the actual number of detections. For example, if *n* is 5, the upper bound of the *BCE* is 96.875%. if *n* is 10, the upper bound of the *BCE* becomes 99.9%, which should be accurate enough to judge the quality of the test set.

### 3.2 Metric for at-speed defects

**Definition:** Given a test set *T*, the Fault Observation Coverage (*FOC*) is defined as follows:

$$FOC = \left(\left(\sum_{f \in F} \frac{op(f)}{op_{max}(f)}\right) / |F|\right)$$

where $op(f)$ is the number of observation points utilized by *T* to observe a fault *f,* and $op_{max}(f)$ is the maximum number of possible observation points for fault *f.*

**Example:** Consider a circuit containing three faults $f_1$, $f_2$, and $f_3$, and each fault has five possible observation points. Suppose that faults $f_1$, $f_2$, and $f_3$ are detected at 5, 3, and 2 observation points respectively. The corresponding *FOC* is calculated as below:

*FOC* = (5/5 + 3/5 + 2/5) / 3 = 66.67%

The problem with this model is that the calculation of $op_{max}(f)$ is time consuming. Therefore, a new model called *Fault Observation Estimate* (*FOE*) is proposed to measure the observation quality for a test set. *FOE* can be derived from the multiple-detect coverage profile after fault simulation.

**Definition:** Given a test set *T*, the Fault Observation Estimate (*FOE*) is defined as follows:

$$FOE = \sum_{f \in F} \left(1 - \left(1 - \frac{1}{op_{max}(f)}\right)^n\right) / |F|$$

where *n* is the number of times the fault *f* is detected by *T,* and $op_{max}(f)$ is the maximum number of possible observation points for fault *f.*

Take the same example shown in figure 1, where fault $f_1$ has 5 observation points and only one of them has enough delay to observe the fault effect. If the test generator propagates the fault effect randomly, *FOE* assumes that it has a 20% chance to propagate the fault to the at-speed detectable observation point when $f_1$ is detected once. If $f_1$ is detected four times with random propagation, the probability of propagating the fault effect to the right observation point is $(1-(1-1/5)^4)$, or 59%. *FOE* is especially suitable for high-speed deep-submicron design where the combinational logic is shallow and the delays are dominated by wiring, which makes the probability of fault detection among the different paths more equal. In such design, *FOE* more accurately represents the average of the at-speed observation probability for all faults in the design. For at-speed deterministic test generation, where the timing information is not available, random decision order is used to propagate the fault effect through different observation points for the different detections to maximize *FOE*. *FOE* gives an indication of the average percentage of the observation points that are used by the test set. Note that the maximum *FOE* coverage may not be 1 since $op_{max}(f)$ is computed from the structural analysis of all reachable observation points of a fault site. A fault may not have a sensitization path to a reachable observation point. In general, a higher *FOE* number indicates better at-speed test quality.

## 4. A GENERAL TEST GENERATION FLOW

A good test pattern generation flow determines the test quality, the effectiveness of the test pattern set, and the efficiency of the test pattern generation run-time. Depending on design requirements, design limitations and available test resources, we can customize a test generation flow from a general flow. Listed in the following is a set of items that could affect this process:

1. Maximum number of test patterns that can be installed on a tester.
2. Test quality goal for standard logic: stuck-at test coverage and the number of detections, transition test coverage and the number of detections, number of paths and test coverage for path delay test.
3. Number of clock domains for transition test and path delay test.
4. Clock frequency requirement for transition test and path delay test of each clock domain.
5. Quality requirement for on-system test.

The tester memory size and the test quality goals determine if on-chip test compression technique is required. When more than one clock domain is required to have transition and/or path delay test coverage, we need to determine the tester memory distribution among these domains. We also need to decide the tester memory distribution between stuck-at test, transition test, and path delay test. On-system test is a feature that allows an ASIC to be tested

after it has been integrated on a PCB board. The application is useful for system level test, diagnosis, and in-field test. Logic BIST [9] needs to be implemented when this feature is required.

From the test generation flow point of view, we need to make sure that the desired test quality is met with given available tester memory. Several concepts will be used to describe the test generation flow. A *coverage gradient* of a test vector set is defined as an additional test coverage gain obtained from each additional test vector added to the existing test vector set. The optimum tester memory distribution should keep the coverage gradient among all test vector sets equal, and the sum of all test vector sets within the tester memory limit. In order to meet this optimum tester memory distribution, an initial ATPG run or test coverage estimation is required to determine the number of vectors for each test set. This does not affect the design schedule since the average number of ATPG iterations during the whole design cycle is typically around 30~50. The coverage gradient obtained from each ATPG run is used to determine the pattern size of each vector set in the following ATPG run. The coverage gradients tend to stabilize as the design cycle proceeds towards the final release.

Each test vector set is generated to target the different fault lists--transition faults, path-delay faults or stuck-at faults. The vector sets also target another fault list(s). For example, test vectors for path-delay faults also detect some transition faults and stuck-at faults. To improve the tester memory usage, an incremental ATPG approach is used [14]. The incremental ATPG run only targets the remaining faults which were not detected by other ATPG runs. In the case where multiple-detect is required, each ATPG run only targets those faults that have not reached the number of required detections. Both path-delay and transition ATPG runs may cross-detect faults in the other fault list, i.e., the path-delay test set may also detect transition faults, and the transition test set may also detect path-delay faults. Since transition ATPG tends to use short paths to propagate fault effect, the selected critical paths are unlikely to be detected by the transition test vectors. Therefore, we only use the path-delay test set to evaluate transition fault detection, but not vice versa. When logic BIST is used, all deterministic ATPG runs will only target the remaining faults not detected by the logic BIST run.

To summarize the incremental ATPG procedure, assume the initial fault lists for path-delay faults, transition faults, and stuck-at faults are $\mathbf{F_P}$, $\mathbf{F_T}$, and $\mathbf{F_S}$, respectively. When multiple-detection is required, we can duplicate the faults in the fault lists $\mathbf{F_P}$, $\mathbf{F_T}$, or $\mathbf{F_S}$ with the number of times equal to the number of detections required. Each time a fault is detected, it will be removed from the fault lists once. Another approach is to attach a number equal to the number of detections required to each fault as an attribute. Each fault detection will decrement this number until it becomes zero. The logic BIST run detects $\mathbf{F_{P\text{-}LBIST}}$, $\mathbf{F_{T\text{-}LBIST}}$, and $\mathbf{F_{S\text{-}LBIST}}$ faults from the path-delay, transition and stuck-at fault lists, respectively. The path-delay ATPG run detects $\mathbf{F_{P\text{-}path}}$, $\mathbf{F_{T\text{-}path}}$, and $\mathbf{F_{S\text{-}path}}$ faults from the path-delay, transition and stuck-at fault lists, respectively. Transition ATPG run detects $\mathbf{F_{T\text{-}trans}}$ and $\mathbf{F_{S\text{-}trans}}$ faults from the transition and stuck-at fault lists, respectively. Finally, stuck-at ATPG run detects $\mathbf{F_{S\text{-}stuck}}$ faults from the stuck-at fault list. In general, the multiple-detect ATPG is applied for the transition and stuck-at fault models, but not for the path delay fault model.

1. Initial fault lists: ( $\mathbf{F_P}$, $\mathbf{F_T}$, $\mathbf{F_S}$ )
2. Logic BIST run detects path-delay, transition and stuck-at faults $\mathbf{F_{P\text{-}LBIST}}$, $\mathbf{F_{T\text{-}LBIST}}$ and $\mathbf{F_{S\text{-}LBIST}}$, respectively.
3. Remaining fault lists:
   ( $\{\mathbf{F_P} - \mathbf{F_{P\text{-}LBIST}}\}$, $\{\mathbf{F_T} - \mathbf{F_{T\text{-}LBIST}}\}$, $\{\mathbf{F_S} - \mathbf{F_{S\text{-}LBIST}}\}$ )
4. Path-delay ATPG targets fault list: $\{\mathbf{F_P} - \mathbf{F_{P\text{-}LBIST}}\}$, and also detects path-delay, transition and stuck-at faults $\mathbf{F_{P\text{-}path}}$, $\mathbf{F_{T\text{-}path}}$ and $\mathbf{F_{S\text{-}path}}$, respectively.
5. Remaining fault lists:
   ( $\{\mathbf{F_P} - \mathbf{F_{P\text{-}LBIST}} - \mathbf{F_{P\text{-}path}}\}$,
   $\{\mathbf{F_T} - \mathbf{F_{T\text{-}LBIST}} - \mathbf{F_{T\text{-}path}}\}$,
   $\{\mathbf{F_S} - \mathbf{F_{S\text{-}LBIST}} - \mathbf{F_{S\text{-}path}}\}$ )
6. Multiple-detect ATPG targets the transition fault list:
   $\{\mathbf{F_T} - \mathbf{F_{T\text{-}LBIST}} - \mathbf{F_{T\text{-}path}}\}$,
   and also detects transition and stuck-at faults $\mathbf{F_{T\text{-}trans}}$ and $\mathbf{F_{S\text{-}trans}}$, respectively
7. Remaining fault lists:
   ( $\{\mathbf{F_P} - \mathbf{F_{P\text{-}LBIST}} - \mathbf{F_{P\text{-}path}}\}$,
   $\{\mathbf{F_T} - \mathbf{F_{T\text{-}LBIST}} - \mathbf{F_{T\text{-}path}} - \mathbf{F_{T\text{-}trans}}\}$,
   $\{\mathbf{F_S} - \mathbf{F_{S\text{-}LBIST}} - \mathbf{F_{S\text{-}path}} - \mathbf{F_{S\text{-}trans}}\}$ )
8. Multiple-detect ATPG targets stuck-at faults $\{\mathbf{F_S} - \mathbf{F_{S\text{-}LBIST}} - \mathbf{F_{S\text{-}path}} - \mathbf{F_{S\text{-}trans}}\}$, and also detects stuck-at faults $\mathbf{F_{S\text{-}stuck}}$
9. Final undetected fault lists:
   ( $\{\mathbf{F_P} - \mathbf{F_{P\text{-}LBIST}} - \mathbf{F_{P\text{-}path}}\}$,
   $\{\mathbf{F_T} - \mathbf{F_{T\text{-}LBIST}} - \mathbf{F_{T\text{-}path}} - \mathbf{F_{T\text{-}trans}}\}$,
   $\{\mathbf{F_S} - \mathbf{F_{S\text{-}LBIST}} - \mathbf{F_{S\text{-}path}} - \mathbf{F_{S\text{-}trans}} - \mathbf{F_{S\text{-}stuck}}\}$ )

From this ATPG flow, we can see that stuck-at faults can be detected by LBIST, path-delay test, transition test, and stuck-at test.

Multiple system clock-domains are often used in ASIC designs. These system clocks can be asynchronous to each other and running at different frequencies. The latencies and skews of these clocks can also be different. Since path-delay test and transition test are applied at-speed, it is critical to make sure that test timing is correct. A separate test vector set is generated for each clock domain with a unique timing template. Test vectors may still fail at some flip-flops during timing simulation due to multi-cycle paths in a design. One common approach to handle this situation is to mask these flip-flops that failed in timing simulation, so that captured data in these flip-flops are never compared. This loss in test coverage may be considerable, since these flip-flops not only capture the faults on multi-cycle paths, but also capture many faults on other single-cycle paths,

which will not cause timing simulation failure. The approach used in our design is to mask these flip-flops only in the test vectors that capture faults from the multi-cycle paths, i.e. each test vector masks a different set of failing flip-flops.

## 5. EXPERIMENTAL RESULTS

The design used to evaluate the three different DFT techniques contains 925K gates and 77K scan cells. The design includes a Logic BIST implementation required for system-level test, where a chip tester environment is not available. To simplify design timing closure and reduce hardware overhead, no test points were inserted in the design [15].

Table I shows the scan design information for three different DFT techniques used: ATPG for deterministic patterns (*ATPG*), EDT configured for up to 10X compression of deterministic patterns (*EDT10X*), and logic BIST for pseudo-random patterns (*LBIST*). The columns in the table provide the following data for each technique: a) the number of external scan chains to be connected to a tester, b) the number of internal scan chains within the design that can be loaded/unloaded in parallel, c) the number of shift cycles to fully load all scan chains, and d) the additional hardware cost to implement the techniques. Note that the number of internal scan chains used for *EDT10X* and *LBIST* are ten times as many as the chains used for *ATPG*. Consequently, they require about 10 times fewer cycles than ATPG to load/unload each test pattern. Due to the initialization of the hardware decompressor, *EDT10X* requires 4 extra shift cycles per pattern.

**Table I: Scan design characteristics**

| DFT method | # external chains | # internal chains | # shift cycles | overhead (K gate) |
|---|---|---|---|---|
| **ATPG** | 10 | 10 | 8530 | None |
| **EDT10X** | 10 | 100 | 780 | 2.5 |
| **LBIST** | None | 100 | 776 | 10 |

### 5.1 DFT techniques comparison

The test results of these three DFT techniques are reported in this section. Table II contains the stuck-at fault test results for the three DFT techniques. For uncompressed (*ATPG*) and compressed (*EDT10X*) deterministic test, multiple-detect up to 5-detect ATPG runs are shown. Note that for ATPG patterns, all available software compression techniques were used; the compression mentioned refers to the hardware decompression and compaction used by EDT but not by traditional ATPG.

The first column in the table indicates the DFT technique used. The second column shows the number of multiple-detect targeted. For logic BIST, only the test coverage results are listed for various numbers of pseudo-random patterns. The multiple-detect results will be discussed later in Figure 4. The third column shows the total number of test patterns, and the fourth column shows the total number of test cycles, which is a metric for test volume and test application time. Note that the test cycles for logic BIST only relates to test application time since there is no need to store the test data on a tester. The fifth, sixth, and seventh columns show three test quality measurements: test coverage, bridging coverage estimate (*BCE*), and fault observation estimate (*FOE*), respectively.

For *ATPG* and *EDT10X*, each additional multiple-detect run shows an increase in test quality measured by *BCE* and *FOE*, at the expense of increased test volume and test application time. Multiple-detect for both *ATPG* and *EDT10X* requires multiple times the corresponding single-detect test data volume to be stored on a tester. The relation is almost linear. For *LBIST*, the test quality increases with the increased number of pseudo-random patterns, at the expense of increased test application time. But the test quality that can be achieved by *LBIST* alone is limited. With 128K pseudo-random patterns, *LBIST* can achieve 92.39% test coverage. The *ATPG* and *EDT10X* tests with single-detect can achieve higher test coverage, 98.75%. Similarly, the *ATPG* and *EDT10X* test sets also achieved higher *BCE* and *FOE*. The reasons for the limited test quality that *LBIST* can achieve is the random pattern resistant faults, and that as mentioned earlier, no test points were added to the design.

A uniform distribution of multiple-detect throughout the fault population is also important to achieve high *BCE* and *FOE*. The large number of pseudo-random patterns applied by *LBIST* results in more detections for pseudo-random testable faults, while the pseudo-random resistant faults have few or no detection. The deterministic test pattern sets for both ATPG and EDT distribute the multiple detections across more testable fault sites. This can be reflected by *BCE*, *FOE* and test coverage numbers. These results show that when *LBIST* is used, it is necessary to use deterministic patterns (ATPG or EDT) to increase the test quality.

**Table II: Stuck-at fault test results**

| | n-det. | # pat. | # test cycles | test cov. | BCE | FOE |
|---|---|---|---|---|---|---|
| **ATPG** | 1 | 2144 | 18.3M | 98.75% | 96.22% | 59.11% |
| | 2 | 4036 | 34.4M | 98.76% | 97.70% | 60.02% |
| | 3 | 5847 | 49.9M | 98.76% | 98.24% | 60.43% |
| | 4 | 7688 | 65.6M | 98.76% | 98.47% | 60.69% |
| | 5 | 9498 | 81.0M | 98.76% | 98.58% | 60.86% |
| **EDT 10X** | 1 | 2272 | 1.8M | 98.73% | 96.16% | 59.04% |
| | 2 | 4190 | 3.3M | 98.74% | 97.64% | 59.91% |
| | 3 | 6061 | 4.7M | 98.74% | 98.20% | 60.33% |
| | 4 | 7925 | 6.2M | 98.75% | 98.44% | 60.59% |
| | 5 | 9832 | 7.7M | 98.75% | 98.56% | 60.78% |
| **LBIST** | | 8K | 6.4M | 89.77% | 88.49% | 56.56% |
| | | 16K | 12.7M | 90.64% | 89.62% | 57.22% |
| | | 32K | 25.4M | 91.36% | 90.49% | 57.78% |
| | | 64K | 50.9M | 91.94% | 91.20% | 58.29% |
| | | 96K | 76.3M | 92.21% | 91.54% | 58.55% |
| | | 128K | 101.8M | 92.39% | 91.76% | 58.71% |

When comparing the different deterministic techniques, both *ATPG* and *EDT10X* give the same level of high test quality across the same multiple-detect test set. However, due to hardware compression, the test data volume and test application time are significantly less for *EDT10X*, as indicated by the lower number of test cycles in the table. For example, to achieve the test quality of 5-detect, the test data volume for *EDT10X* is less than one tenth that of 5-detect ATPG, and less than half that of 1-detect ATPG. This implies that much higher test quality can be achieved by using compressed multiple-detect patterns when tester resources cannot accommodate uncompressed multiple-detect ATPG patterns.

One interesting observation for the deterministic test sets is that the test coverage grows slightly (less than 0.02% from 1-detect to 5-detect) when the multiple-detect test sets are applied. One might expect that the test coverage should remain the same since there is no credit for test coverage beyond the 1-detect. In practice, the slight increase in test coverage is due to random effects on some aborted faults. When targeting these aborted faults multiple times by ATPG using the random decision order, some of them are detected, causing a slight increase in test coverage.

Figure 2 depicts the *BCE* and *FOE* coverages as a function of test cycles for the three different DFT techniques, using the stuck-at fault model. Note that the x-axis displaying the test cycles has logarithmic scale. *EDT10X* has the same scan configuration as *LBIST*. 5-detect ATPG runs were used to display the curves. The BCE and FOE curves reveal the test quality that can be achieved with the number of test cycles used. The cost for *ATPG* and *EDT10X* is tester memory to accommodate the test cycles, as well as the test application time for applying the test cycles. For *LBIST*, the cost is only test application time, since tester resources for LBIST are negligible. Figure 2 demonstrates that the compression technology can achieve the highest test quality at a low cost. Note that *LBIST* initially has a higher test quality than *ATPG* for the same number of test cycles, but as the number of test cycles increases, *ATPG* prevails. This is because the scan chains for *LBIST* are ten times shorter than for *ATPG*, which means that *LBIST* applies ten times more test patterns than *ATPG* for the same number of test cycles. The test coverage that *LBIST* can achieve levels out around 90%.

The same tests are performed on the transition fault model for the three DFT techniques. The same conclusions as for the stuck-at tests are obtained for the transition tests as shown in table III and figure 3. All three test quality measurements (test coverage, BCE, and FOE) indicate that *EDT10X* achieves the highest test quality with low test cost. Because the deterministic transition test set is more than two times larger than the deterministic stuck-at test set, it becomes even more important to apply compression technique to reduce the test cost.

Figure 4 shows the multiple-detect test coverage profile for *EDT10X* with single-detect, *EDT10X* with five-detect, and *LBIST* with 128K pseudo-random patterns. The bars in

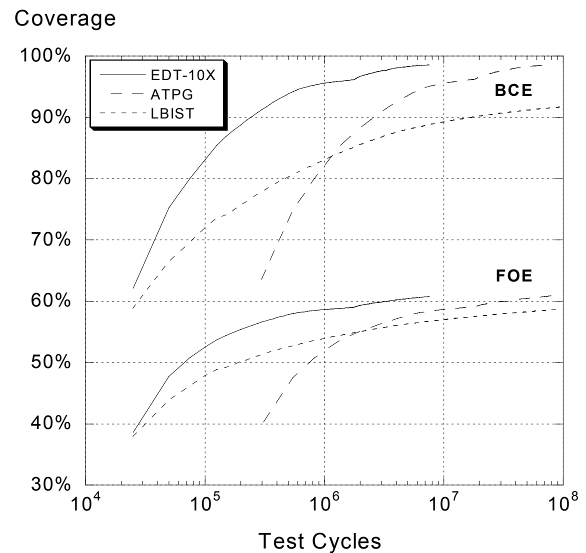


**Figure 2: BCE and FOE comparison of different techniques for stuck-at faults**

**Table III: Transition fault test results**

| | **n-det.** | **# pat.** | **test cycles** | **test cov.** | **BCE** | **FOE** |
|---|---|---|---|---|---|---|
| **ATPG** | 1 | 5082 | 43.3M | 97.94% | 94.91% | 57.65% |
| | 2 | 9400 | 80.2M | 98.03% | 96.74% | 58.82% |
| | 3 | 13396 | 114.3M | 98.06% | 97.41% | 59.38% |
| | 4 | 17354 | 148.0M | 98.07% | 97.71% | 59.73% |
| | 5 | 21221 | 181.0M | 98.08% | 97.85% | 59.97% |
| **EDT 10X** | 1 | 5657 | 4.4M | 97.87% | 94.80% | 57.46% |
| | 2 | 10867 | 8.5M | 97.98% | 96.64% | 58.62% |
| | 3 | 15834 | 12.4M | 98.01% | 97.33% | 59.20% |
| | 4 | 20678 | 16.1M | 98.02% | 97.65% | 59.58% |
| | 5 | 25396 | 19.8M | 98.04% | 97.80% | 59.83% |
| **LBIST** | | 8K | 6.4M | 85.96% | 83.69% | 53.01% |
| | | 16K | 12.7M | 87.60% | 85.89% | 54.18% |
| | | 32K | 25.4M | 88.88% | 87.53% | 55.10% |
| | | 64K | 50.9M | 89.91% | 88.78% | 55.88% |
| | | 96K | 76.3M | 90.48% | 89.43% | 56.31% |
| | | 128K | 101.8M | 90.84% | 89.84% | 56.59% |

the figure indicate the test coverage, i.e. the percentage of testable faults that are detected at least *n* times, where *n* has a range from 1 to 10 on the x-axis. Most of the faults, which are easy to detect, are detected 10 times or more by both the 1-detect *EDT10X* test set and the *LBIST* test set. This phenomenon has been observed for many full scan designs. The major difference between the 1-detect deterministic test patterns and the *LBIST* test set is that the deterministic test can achieve high test coverage within the targeted number of detects. *LBIST* does not achieve very high coverage. However, the coverage is fairly stable with a gradual drop off for higher numbers of multiple-detect.

In Figure 4, the number of faults detected at least 5 times by LBIST is higher than that by 1-detect determinis-

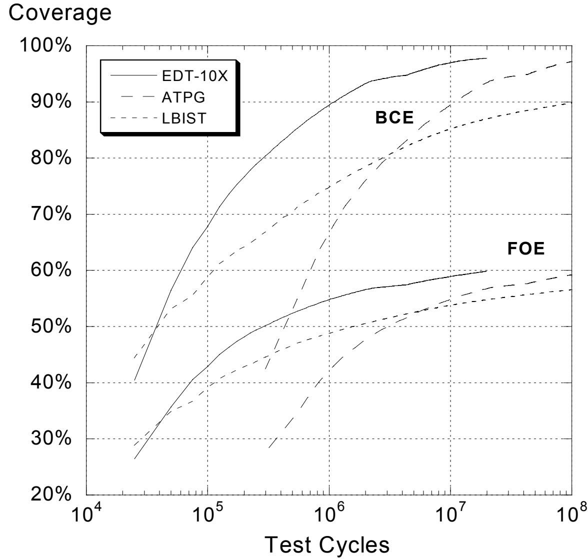


**Figure 3: BCE and FOE comparison of different techniques for transition faults**

tic test due to the large number of pseudo-random patterns used. On the other hand, the 1-detect deterministic test set has a larger number of faults detected for 1, 2, 3, and 4 times.

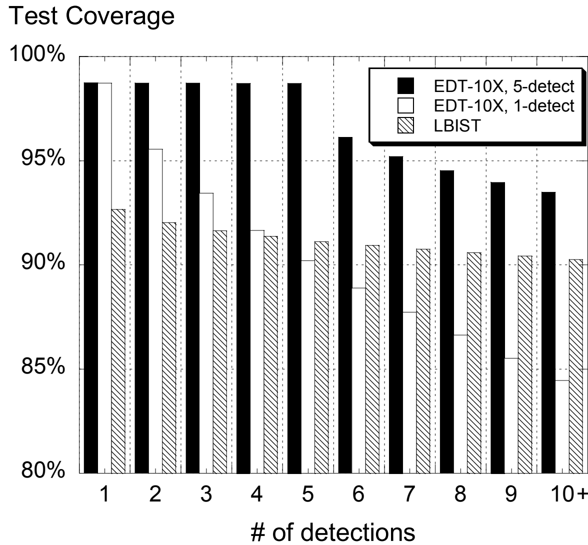


**Figure 4: Multiple-detect test coverage comparison between LBIST and EDT patterns for stuck-at faults**

## 5.2 Incremental ATPG flow

Experiments were also performed to evaluate the cost saving when an incremental ATPG generation flow is used. In the incremental ATPG generation flow, transition test ATPG was performed for the fault list within which the faults that have been already detected by LBIST are excluded. Similarly, the stuck-at ATPG was performed for the fault list within which the faults that are detected by LBIST and the transition test sets are excluded. The experiment with pseudo-random patterns uses an LBIST implementation with 32K patterns and two at-speed capture pulses. The test quality criteria is to achieve 5-detect for all testable stuck-at and transition faults. $P_{tr}$ and $P_{st}$ represent the test patterns sets for the transition and stuck-at tests, respectively. As shown in table IV, the incremental ATPG flow without any pseudo-random pattern reduces the test data volume by 19.0%. If 32K pseudo-random patterns are applied initially before the incremental ATPG flow, it further reduces the test data volume to 21.1% as compared to the basic flow.

**Table IV: Test results for incremental ATPG flow**

| | | 1-det | 2-det | 3-det | 4-det | 5-det |
|---|---|---|---|---|---|---|
| **Basic flow** | # $P_{tr}$ | 5,082 | 9,400 | 13,396 | 17,354 | 21,221 |
| | # $P_{st}$ | 2,144 | 4,036 | 5,847 | 7,688 | 9,498 |
| | **Total pattern count** | | | | | 30,719 |
| **inc. ATPG without LBIST** | # $P_{tr}$ | 5,082 | 9,400 | 13,396 | 17,354 | 21,221 |
| | # $P_{st}$ | 781 | 1,483 | 2219 | 2,925 | 3,659 |
| | **Total pattern count** | | | | | 24,880 |
| | **Test data reduction** | | | | | **19.0%** |
| **inc. ATPG with LBIST** | # $P_{tr}$ | 5,067 | 9,185 | 13,043 | 16,873 | 20,612 |
| | # $P_{st}$ | 773 | 1,498 | 2,208 | 2,918 | 3,622 |
| | **Total pattern count** | | | | | 24,234 |
| | **Test data reduction** | | | | | **21.1%** |

## 5.3 Industrial design case analysis

An industrial design case will be presented in this section. The design contains 10.4 million standard logic gates, 432K flip-flops and more than 100 clock domains. The main clock domain is running at 250 MHz. Logic BIST was implemented in this design. No test points were inserted in the design. In the scan test, there are 15 external scan chains that can be connected to the tester. In the LBIST test mode, there are 1,005 internal scan chains that can be loaded in parallel with pseudo-random patterns generated internally. When the EDT deterministic test is used, the 15 external scan chains are connected to the tester. The EDT decompressor will decompress 15 bits scan test data obtained from the tester via 15 external scan chains to 1,005 bits scan test data and feed into 1,005 internal scan chains in one cycle. The longest scan chain for scan test has 29,563 flip-flops, and the longest scan chain for LBIST and EDT has 442 flip-flops. It is obvious that the number of test cycles needed for each normal scan pattern is much larger than that for each LBIST or EDT test patterns.

Table V lists 1-detect ATPG results with no EDT deterministic logic and non-incremental ATPG. With 24,512 pseudo-random patterns, LBIST can achieve 93.65% stuck-at test coverage and 77.62% transition test coverage. The total number of test cycles used in LBIST test is

10,834K. Transition ATPG and stuck-at ATPG are performed separately and their coverages are 83.68% and 99.09%, respectively. Transition test ATPG is only performed for one clock domain, the 250MHz major clock domain. The actual numbers of test cycles for transition test and stuck-at test are much larger for ATPG than for LBIST. This is due to the maximum scan chain length in LBIST mode being much shorter than that for normal scan mode.

Table VI lists 1-detect pattern generation results using EDT instead of traditional ATPG, and using the incremental pattern generation flow. LBIST results are the same as listed in table V. The last two columns report the compression in tester cycles and tester memory, respectively, compared to the results in Table V. Transition pattern generation achieved 83.54% test coverage with 7,340 test vectors. The transition test vector set also detected many stuck-at faults. In this case, it achieved 85.38% stuck-at test coverage. On top of that, we ran stuck-at pattern generation and got 98.83% stuck-at test coverage. From this design, we observed that the compressed deterministic ATPG with the incremental pattern generation flow achieved slightly lower test coverage compared to the non-compressed deterministic ATPG and non-incremental pattern generation flow. However, that coverage difference is mainly due to additional constrains added during the EDT run which artificially gives up credit for faults which will be detected in reality. For example, a stuck-in-system-mode fault on the first scan cell of every chain is reported as being untestable in EDT. However, a fault on that line would be detected since it prevents scan loading from working (the scan chain would no longer be sensitized in the presence of this fault). Secondly, the EDT pattern count is higher than that of normal ATPG. However, due to the test cycle reduction for scan loading, the final number of test cycles for EDT is still much lower than that of normal scan test. Thus it saves tester memory and tester time. The total reduction in the number of tester cycles is 23x (508,201/22,031), which includes LBIST and deterministic patterns. The tester scan volume reduction is 44x (the ratio of the test cycles of ATPG vs. EDT).

**Table V: 1-detect with no EDT and non-incremental flow**

| | Trans cov. | Stuck-at cov. | # Pattern | # Test cyc. (K) |
|---|---|---|---|---|
| **LBIST** | 77.62% | 93.65% | 24,512 | 10,834 |
| **Trans. ATPG** | 83.68% | -- | 4,889 | 144,533 |
| **Stuck-at ATPG** | -- | 99.09% | 11,935 | 352,834 |
| **Total** | **83.68%** | **99.09%** | **41,336** | **508,201** |

**Table VI: 1-detect with EDT and incremental flow**

| | Trans cov. | Stuck-at cov. | # Pattern | # Test cyc. (K) | Compression | |
|---|---|---|---|---|---|---|
| | | | | | Cycles | Mem. |
| **LBIST** | 77.62% | 93.65% | 24,512 | 10,834 | 1x | N/A |
| **Trans. EDT** | 83.54% | 85.38% | 7,340 | 3,244 | 44x | 44x |
| **Stuck-at EDT** | -- | 98.83% | 17,994 | 7,953 | 44x | 44x |
| **Total** | **83.54 %** | **98.83 %** | **49,846** | **22,031** | **23x** | **44x** |

## 6. CONCLUSIONS

Test methods beyond the traditional stuck-at fault model are needed to address new types of defects for design processes at 130nm and below. In this paper, multiple-detect for both stuck-at and transition faults was used to increase the test quality for advanced deep-submicron designs. Two statistical metrics (*BCE* and *FOE*) were used to assess the quality of the different test sets considered.

The introduction of multiple-detect for stuck-at and transition tests dramatically increases the test data volume and test application time of scan patterns. Hence, applying the higher-quality test sets may be infeasible due to test resource constraints. This paper presented two techniques to reduce test data volume. Using Embedded Deterministic Test (EDT) to apply compressed deterministic patterns can reduce the test data volume and tester time by 10X or more as compared to traditional ATPG. The incremental test flow when generating patterns for multiple fault models can reduce the test set size further. Logic BIST, which is commonly used for system test, can be applied to increase multiple-detect coverage for detections beyond those targeted by the deterministic patterns.

Using the discussed techniques has been shown to allow the higher quality test sets, including transition tests and multiple-detect, to be applied within the tester constraints currently available for stuck-at tests.

## 7. REFERENCES


[1] B. R. Benware, R. Madge, C. Lu, and Dr. R. Daasch, "Effectiveness Comparisons of Outlier Screening Methods for Frequency Dependent Defects on Complex ASICs", VTS, pp. 39-46, 2003.

[2] W. Needham, C. Prunty, E. H. Yeoh, "High Volume Microprocessor Test Escapes, an Analysis of Defects our Tests are Missing", ITC, pp. 25-34, 1998.

[3] B. Benware, C. Schuermyer, S. Ranganathan, R. Madge, P. Krishnamurthy, N. Tamarapalli, K.-H. Tsai, J. Rajski, "Impact of Multiple-Detect Test Patterns on Product Quality", ITC, pp. 103-1040, 2003

[4] J. Rajski, M. Kassab, N. Mukherjee, N. Tamarapalli, J. Tyszer, J. Qian, "Embedded deterministic test for low-cost manufacturing", IEEE Design & Test of

Computers, Volume 20, Issue 5, pp. 58-66, Sept.-Oct. 2003

[5] F. Poehl, M. Beck, R. Arnold, P. Muhmenthaler, N. Tamarapalli, M. Kassab, N. Mukherjee, J. Rajski, "Industrial Experience with Adoption of EDT for Low-Cost Test without Concessions", ITC, pp. 1211-1220, 2003

[6] R. Wilson, "Delay-Fault Testing Mandatory, Author Claims," *EE Design*, Dec. 4, 2002.

[7] Bruce D. Cory, Rohit Kapur and Bill Underwood, "Speed binning with Path Delay Test in 150-nm Technology", *IEEE Design & Test of Computers*, Sept-Oct. 2003

[8] Kee Sup Kim, Subhasish Mitra, and Paul G. Ryan, "Delay Defect Characteristics and Testing Strategies", *IEEE Design & Test of Computers*, Sept-Oct. 2003

[9] S. Chakravarty, A. Jain, N. Radhakrishnan, E. W. Savage, and S. T. Zachariah, "Experimental Evaluation of Scan Tests for Bridges", *Proc. ITC*, pp. 688-695, 2002.

[10] M. R. Grimaila, Sooryong Lee; J. Dworak, K.M. Butler, B. Stewart, H. Balachandran, B. Houchins, V. Mathur, Ja-hong Park, L.-C.Wang, M. R. Mercer, "REDO-random excitation and deterministic observation-first commercial experiment", *Proc. VTS*, pp. 268 –274, 1999.

[11] J. Dworak, J. Wicker, S. Lee, M.R. Grimaila, K.M. Butler, B. Stewart, L.C.Wang and M.R. Mercer, "Defect-oriented testing and defective part level prediction for commercial sub-micron ICs", *IEEE Design and Test of Computers,* pp. 31-41, Jan.-Feb. 2001.

[12] S. Lee, B. Cobb, J. Dworak, M. R. Grimaila and M. R. Mercer "A New ATPG Algorithm to Limit Test Set Size and Achieve Multiple Detections of all Faults", *Proc. DATE*, pp.94-99, 2002.

[13] E. J. McCluskey, Chao-Wen Tseng, "Stuck fault tests vs. actual defects", *Proc. ITC*, pp. 336-342, 2000.

[14] Lin, X.; Press, R.; Rajski, J.; Reuter, P.; Rinderknecht, T.; Swanson, B.; Tamarapalli, N., "High-frequency at-speed scan testing", *Design & Test of Computers, IEEE*, pp. 17-25, Sept.-Oct. 2003.

[15] Xinli Gu, Sung S. Chung, Frank Tsang, and Jan A. Tofte, "An Effort-Minimized Logic BIST Implementation Method", *Proc. ITC*, pp 1002-1010, 2001.