The Economics of Defect-Based Testing

Chris Allsup
Synopsys, Inc. 700 E. Middlefield Road, Mountain View, CA 94043

***Abstract*—This paper formulates the economic trade-offs involved when multiple defect-based tests are generated to achieve high defect coverage for digital designs. Specifically, we define the criterion for determining the most "expensive" of these tests. Assuming all the other generated tests will be applied in their entirety, we show that there is a threshold tester memory level below which it is more cost-effective to compress and test only a subset of the patterns in the most expensive test. We also describe how to achieve even greater cost reduction by pruning all patterns that incrementally increase total cost. For either approach, the number of retained patterns is calculated by evaluating the cost trade-off between less scan compression, leading to marginally-lower test quality, and more scan compression, leading to marginally-higher test quality**.

I. INTRODUCTION

Several types of test, each targeting specific physical failure mechanisms, are needed to achieve high defect coverage of digital ICs [1, 2]. Semiconductor manufacturers now routinely rely on transition delay tests in addition to stuck-at tests to improve test quality. Other defect-based tests such as small delay defect tests and dynamic bridging tests have captured significant mindshare because they potentially achieve even higher quality. But the use of multiple test sets has significantly increased total pattern count even as the number of scan flops in designs has climbed to the level of millions. In response, designers have turned to on-chip compression techniques to stem the costs of test data volume inflation. Even so, ever higher compression levels are needed to reduce the amount of data associated with multiple test pattern sets, especially when the tester configuration has relatively limited available memory.

If the number of test pattern sets continues to increase, semiconductor firms may benefit from applying an economic model that evaluates the cost of adding more test patterns to marginally improve quality when tester memory is limited. In this paper, we present a framework for estimating the difference in cost between supplementing several ATPG pattern sets with one additional pattern set, versus the same set that has been pruned to a pattern level determined by the scan compression ratio that minimizes total cost.

What does the scan compression ratio have to do with the cost of adding test patterns? Increasing compression "squeezes" more patterns into available tester memory to *potentially* improve defect coverage, but at the same time potentially increases the die size and therefore the die cost. By analyzing the fault coverage-versus-pattern count characteristic of a test, it is possible to make an economically-viable trade-off between:

a) The incremental decrease in the cost of test escapes gained by adding enough compression to "squeeze" *all* the patterns in the test into limited tester memory, and

b) The incremental silicon area cost savings gained by *not* adding compression sufficient to accommodate *all* the patterns in the test.

Section 2 provides a brief overview of a scan compression cost model that considers the impact on test and manufacturing costs of on-chip compression circuits designed to reduce test data volume and test application time. We then expand this model to consider the more common scenario of generating, compressing, and applying multiple defect-based test pattern sets. Composite fault coverage, described in Section 3, is a key element of this analysis. Expressed initially as a function of pattern count in Section 4, then scan compression ratio in Section 5, the composite fault coverage reflects the combined quality contribution of multiple defect-based tests generated in serial fashion. Section 6 defines the criterion for determining the most expensive test. Section 7 shows how to calculate the compression level that minimizes the sum of test escape cost and silicon cost to determine if (and how many) patterns should be removed from the most expensive test. Section 8 describes how further cost reduction can be achieved by pruning *all* patterns that incrementally increase total test cost. Section 9 presents cost analysis results for industrial designs.

II. COMPRESSION COST MODEL

The compression cost model [3] for a single pattern set is consistent, in terms of some of the fundamental relations and parameters, with earlier economic models for test [4, 5]. The model assumes the cost of test escapes C_{esc}, the test execution cost C_{exec}, and the silicon area overhead cost of compression $C_{silicon}$ are the costs most sensitive to the level of scan compression. These cost components and their sum, C_{test}, referred to in this context as the "total costs of test," are shown in Figure 1. From an economic vantage point, the model considers test data volume reduction

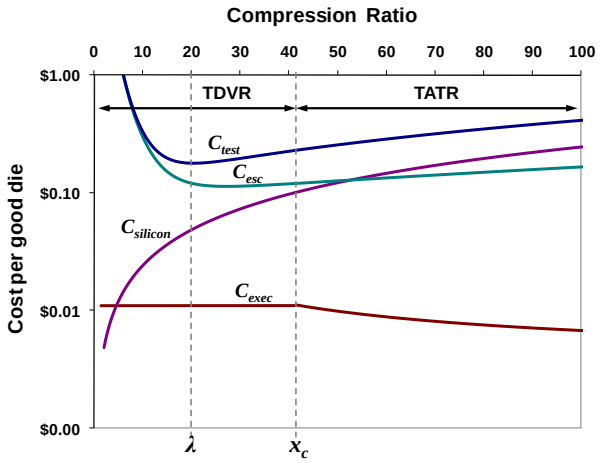

Fig 1. Component costs versus compression.

(TDVR) and test application time reduction (TATR) different compression *phases*, and the only reason to implement scan compression is to a) reduce the cost of escapes C_{esc} in the TDVR phase, and b) reduce the test execution cost C_{exec} in the TATR phase. Both TDVR and TATR compression increase the silicon area overhead cost of compression $C_{silicon}$ in proportion to the amount of compression [3, 4]:

$$C_{silicon}(x) = C_S\left(\frac{A(x)}{Y(x)} - \frac{A_0}{Y_0}\right) \quad x > 1 \qquad (1)$$

$$\text{where } A(x) = A_F + A_0\left(1 + \gamma x + \zeta x^2\right)$$

The independent variable x represents the ratio of the number of internal scan chains to the number of scan I/O channels (we refer to x as the "compression level" but keep in mind it is actually the compression ratio). C_s is the silicon area cost multiplier ($/cm^2), A_0 is the die size without compression (cm^2), Y_0 is the manufactured yield without compression, and A_F is the area of compression circuitry that is independent of compression level (cm^2). γ is the fractional increase in die size per unit increase in compression and depends on the area of compression circuits added per scan chain, the number of scan I/O channels, and the die size A_0. ζ is a second-order area-scaling coefficient that accounts for the effect of wire routing congestion, which increases the compression area by more than that described by γ. The yield $Y(x)$ is a user-defined function of the die size with compression $A(x)$ and the manufacturing defect density (defects/cm^2). The model assumes that any increase in silicon area due to compression increases the die size by the same amount according to (1).

The TDVR phase is defined as the range of compression levels that extends up to the compression level x_c, which is the amount of compression needed to fit the pattern-inflated equivalent of all P_c patterns generated in standard scan mode into the amount of tester memory M allocated for digital stimulus and response patterns. The test execution time, and therefore test execution cost, is virtually constant during the TDVR phase because every unit increase in compression adds more patterns that must be tested, and this exactly offsets any potential reduction in test time. This test time is T_0, the time it takes to execute P_0=$M/(3F)$ patterns that can be loaded into tester memory without compression [3]:

$$T_0 \cong \frac{P_0 F}{C\,\mathrm{f}_{test}} \quad \text{for } M < 3P_c F \qquad (2)$$

where F is the number of scan flops in the design, C is the number of scan I/O channels, and f_{test} is the tester scan shift frequency. The coefficient "3" represents one scan stimulus bit and two response bits: one is the response bit itself and the other a mask or measure bit needed to determine if the response bit should be compared or not. Note that this third bit may not be needed for some designs. The formula assumes the internal scan chains are well-balanced.

Increasing compression above x_c has no economic benefit in terms of reducing the cost of escapes (the green curve in Figure 1) because all the test patterns have been loaded into memory and there are no more patterns to improve quality. In fact, if the total of escape cost and silicon cost is minimized at a ratio λ below x_c where the compression level is not enough to load all the patterns, then there is no economic benefit in increasing compression higher than λ. Above λ, the incremental area overhead cost of compression exceeds the incremental cost savings from the reduction in test escapes, increasing total cost above its minimum at λ, as shown in Figure 1.

The compression level in the TDVR phase is equivalent to the ratio of the number of patterns that can be loaded into memory at compression level x, to P_0, the number of patterns that can be loaded without compression. As x increases, a greater number of test patterns are needed to achieve the same fault coverage due to pattern inflation. For most designs, pattern count increases linearly across a wide range of compression levels, so that the rate of pattern inflation can be described using a single parameter, ε, which represents the fractional increase in pattern count per unit increase in compression ratio.

In many situations tester memory is not a limited resource, so the cost minimum instead occurs at a ratio λ above x_c in the TATR phase. In this range of compression levels there are *potential* cost savings from test application time reduction because the scan chain lengths continue to decrease and the tester time declines by a factor of $(x_c/x)(1+\varepsilon x)$, proportional to the bottom curve in Figure 1. Increasing compression above λ, however, has no economic benefit because above λ the incremental area overhead cost of compression exceeds the incremental cost savings from test time reduction.

The compression cost model predicts the optimal compression ratio λ that minimizes total costs C_{test} in the TDVR phase by combining the silicon area overhead cost

of compression defined in (1) with the cost formula (25) of Section 7 relating the cost of escapes to test escape rate. To estimate escape rate, we need to specify fault coverage as an input parameter to a defect model such as the one presented in equation (26) of Section 7. The next section describes an approach to determining a "composite" fault coverage used for this input parameter when more than one test pattern set is generated.

III. Composite Fault Coverage

In this section we derive the *composite fault coverage* associated with multiple ATPG pattern sets used to test a design, and the *composite fault coverage components* associated with each individual test. The component fault coverages are designed to reflect the individual tests' relative contribution to overall test quality.

Consider the multi-pass test flow of Figure 2 in which each of three tests T_1, T_2, and T_3 corresponds to an ATPG run using a different fault model. Test T_1 generates patterns using fault model M_1. These patterns are fault-simulated on a second fault model M_2, and then a second pattern set is generated using this model, targeting only faults that were undetected from the previous run. This process is repeated for the third test.

Figure 3 illustrates how each test affects the fault spaces. Each fault model M_k, $k=1,2,3$ has a fault space containing N_k faults, and the number of faults in the fault universe is $N_1+N_2+N_3$. Patterns from the first test T_1 detect D_{11} faults in the first space, D_{12} faults in the second space, and D_{13} faults in the third space. Patterns from T_2 detect D_{22} faults in the second space and D_{23} faults in the third space. Finally, patterns from T_3 detect D_{33} faults in the third space. The D_{jk} values represent the number of detected faults in fault model M_k *uniquely* attributed to test T_j, $j=1,2,3$. For example, test T_2 may detect faults in common with the D_{12} faults detected by T_1 but these are

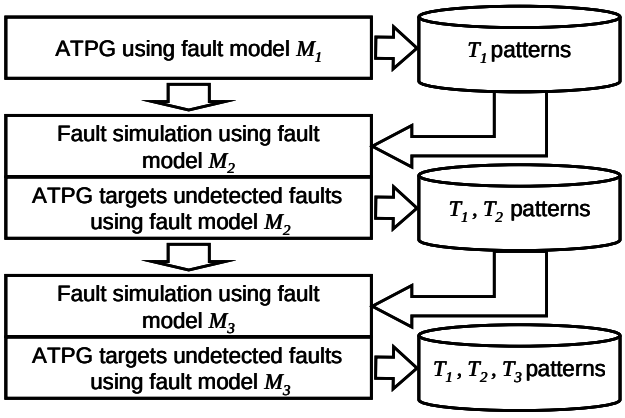

Fig. 2. Test flow in which three types of tests are generated to increase total test quality.

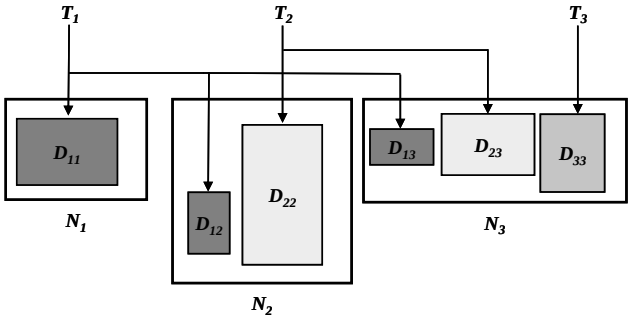

Fig. 3. Fault universe consisting of fault spaces formed by the test flow of Figure 2.

not included in D_{22} and we assume they were already designated detected by the fault simulation of the T_1 patterns.

We now define the composite fault coverage ${}^nf$ of tests $T_1 \ldots T_n$ as the sum of the composite fault coverage components F_j:

$$ {}^nf = \sum_{j=1}^{n} F_j \tag{3} $$

Each composite fault coverage component F_j is the fraction of total faults in the fault universe that are detected uniquely by test T_j. Referring to Figure 3 for three tests,

$$ \begin{aligned} F_1 &= \frac{D_{11}+D_{12}+D_{13}}{N_1+N_2+N_3} = w_1 f_{11} + w_2 f_{12} + w_3 f_{13} \\ F_2 &= \frac{D_{22}+D_{23}}{N_1+N_2+N_3} = w_2 f_{22} + w_3 f_{23} \\ F_3 &= \frac{D_{33}}{N_1+N_2+N_3} = w_3 f_{33} \end{aligned} \tag{4} $$

where

$$ \begin{aligned} w_j &= \frac{N_j}{\sum_{i=1}^{n} N_i} \\ f_{11} &= \frac{D_{11}}{N_1},\ f_{12} = \frac{D_{12}}{N_2},\ f_{13} = \frac{D_{13}}{N_3} \\ f_{22} &= \frac{D_{22}}{N_2},\ f_{23} = \frac{D_{23}}{N_3} \\ f_{33} &= \frac{D_{33}}{N_3} \end{aligned} \tag{5} $$

From now on, we will use the term "component coverage" when referring to component values F_j of the total composite fault coverage ${}^nf$.

Removing patterns from T_n has no effect on the component coverages associated with $T_1 \ldots T_{n-1}$. This implies we can achieve cost minimization in the TDVR

phase as discussed in Section 2 by analyzing the fault coverage convergence characteristic of just this test to find the optimal compression level, knowing the other tests will be applied in their entirety. To perform the analysis, we first express the composite fault coverage as a function of pattern count.

IV. Composite Fault Coverage Versus Pattern Count

Fault coverage as a function of pattern count in a test, *F(P)*, can be represented as an exponential of the form [6]:

$$F(P) = F_c\left(1 - e^{-\eta P}\right) \tag{6}$$

where F_c represents the maximum fault coverage (i.e., the fault coverage that results from applying the complete pattern set, P_c) and η is the exponential constant of fault coverage convergence. η is inversely proportional to P_c and can be obtained in practice by curve-fitting *F(P)* to the ATPG fault coverage data.

To illustrate how composite fault coverage increases with pattern count, we assume that three hypothetical test pattern sets T_1, T_2, and T_3 are generated using the test flow of Figure 2, and produce the following example data for individual fault coverages, pattern counts, and fault coverage convergence constants:

$T_1: f_{11} = 90.0\%,\ f_{12} = 40.0\%,\ f_{13} = 20.0\%,\ P_1 = 3290,\ \eta_1 = 0.0028.$
$T_2: f_{22} = 55.0\%,\ f_{23} = 30.0\%,\ P_2 = 1056,\ \eta_2 = 0.0087.$
$T_3: f_{33} = 34.0\%,\ P_3 = 6950,\ \eta_3 = 0.0013.$
$w_1 = w_2 = w_3 = 33.3\%.$
From (3), $F_1 = 50.0\%,\ F_2 = 28.3\%,\ F_3 = 11.3\%.$

Notice there is only one value for the exponential constant of convergence η_j for each test, measured from ATPG fault coverage versus pattern count data for f_{11}, f_{22}, and f_{33}. Since the number of patterns in each test does not change as we fault-simulate them on successive fault models, our subsequent scaling of these individual fault coverages to obtain the component fault coverages F_j does not affect the exponential convergence constants. Specifically, we assume uniformity of convergence in the tail of the convergence curves across the fault models so that scaling (6) by a constant does not alter η as long as P_c is unchanged.

Referring to Figure 3, the individual fault coverages f_{11}, f_{22}, and f_{33} are obtained in practice by subtracting fault coverages measured from fault-simulating previous pattern sets from the measured ATPG fault coverage data. The fault coverages f_{12} and f_{13} are measured directly after fault-simulating the T_1 patterns. Fault coverage f_{23} is obtained by subtracting the measured fault coverage after fault-simulating the T_1 patterns from the fault coverage measured after fault-simulating the T_2 patterns.

Figure 4 graphs individual fault coverage as a function of pattern count for each test.

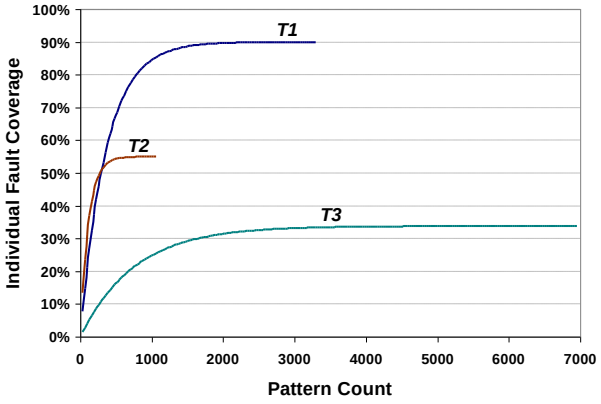

Fig. 4. Individual fault coverages f_{11}, f_{22}, f_{33} versus pattern count for three tests.

If we apply each test in sequence and sum the component fault coverages, then, adjusting the exponent for pattern count offset, the composite fault coverage as a function of pattern count, ${}^3f(P)$, is:

$$ {}^3f(P) = \begin{cases} F_1\left(1-\exp\left(-\eta_1 P\right)\right) & 0 \le P \le P_1 \\ F_1 + F_2\left(1-\exp\left(-\eta_2\left(P-P_1\right)\right)\right) & P_1 \le P \le P_1 + P_2 \\ F_1 + F_2 + F_3\left(1-\exp\left(-\eta_3\left(P-(P_1+P_2)\right)\right)\right) & \\ P_1 + P_2 \le P \le P_1 + P_2 + P_3 & \end{cases} \tag{7}$$

where P_j is the pattern count for test T_j and F_j its corresponding component coverage. Figure 5 graphs the composite fault coverage functions in (7) for the three tests of Figure 4.

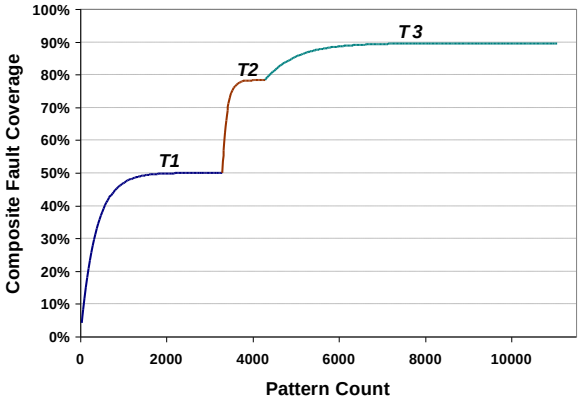

Fig. 5. Composite fault coverage versus pattern count for the three tests of Figure 4.

Because the composite fault coverage components for tests $T_1 \ldots T_{n-1}$ are not affected by the component coverage of test T_n, we can prune patterns *ex post facto* from T_n without impacting the component coverages of the other tests. Therefore, the last expression in (7), expanded for *n* tests, is sufficient for the cost analysis:

$$ {}^{n}f(P) = \sum_{j=1}^{n-1} F_j + F_n\left(1-\exp\left(-\eta_n P\right)\right) \quad P \le P_n \quad (8) $$

To simplify terms, P represents the P^{th} pattern out of P_n patterns in the last test.

In fact, the composite fault coverage components are independent of each other from the standpoint that it is possible to target *any single test* in the test flow of Figure 2, not just the last test, for pattern pruning without impacting the component coverages of the other tests. To determine the optimal compression level, we can analyze the fault coverage convergence of the *targeted* test, which need not be the last test.

V. Composite Fault Coverage Versus Compression

At this point, we have described the composite fault coverage of an arbitrary number of tests in terms of their composite fault coverage components and the pattern count of the last test. But to perform a cost analysis, we need to transform the expression in (8) into a function of compression. To do this, we first need to describe how pattern inflation from scan compression affects pattern count when multiple tests are generated.

Pattern inflation increases with the compression ratio x so that one can describe the behavior as a linear relationship [3]:

$$ P'(x) = P\left(1+\varepsilon x\right) \quad x > 1 \quad (9) $$

where P is the non-inflated number of test patterns needed to achieve a given fault coverage without compression, $P'(x)$ is the greater (inflated) number of test patterns needed to achieve the same fault coverage when applying compression level x, and ε is the pattern inflation rate representing the fractional increase in test patterns per unit increase in compression. The compression cost model presented in Section 2 assumes a single test with a single pattern inflation rate, yet different tests may require different constraints that alter the number of unknown signals propagating through the logic. This leads to different pattern inflation rates among the tests.

Suppose two pattern sets are generated for a design in standard (uncompressed) scan mode. Test T_1 consists of P_1 patterns and test T_2 consists of P_2 patterns. We then configure the design into compression mode with compression ratio x and re-generate the pattern sets using the same ATPG constraints as before. If ε_1 is the pattern inflation rate for test T_1 and ε_2 is the pattern inflation rate for test T_2, then the total number of inflated test patterns is the sum of the inflated pattern counts for each test, $P_1'(x)$ and $P_2'(x)$:

$$ P'(x) = P_1'(x) + P_2'(x) = P_1\left(1+\varepsilon_1 x\right) + P_2\left(1+\varepsilon_2 x\right) \quad x > 1 \quad (10) $$

Now assume that when testing the part we load test T_1 into tester memory first, starting with the first pattern. If there is limited memory, only a subset of the inflated patterns from T_1 can be loaded and it is not possible to test the part with patterns further down in the pattern set. But if there is more than enough memory to load all the T_1 patterns, then it is possible to also load some patterns from test T_2 into tester memory, starting with the first pattern of T_2 and loading as many patterns further down in the pattern set as will fit into memory.

The fraction of T_1 patterns that can be loaded into memory is the amount of available memory M divided by the test data volume at compression level x, $TDV_1(x)$:

$$ \frac{M}{TDV_1(x)} = M \div \frac{3FP_1'(x)}{x} = M \div \left(\frac{3FP_1(1+\varepsilon_1 x)}{x}\right) = \frac{P_0 x}{P_1(1+\varepsilon_1 x)} \quad (11) $$

where $P_0=M/(3F)$ is the number of patterns that can be loaded into memory without compression and F is the number of scan flops in the design. Because pattern inflation is linear, the fraction in (11) also represents the fraction of total patterns P_1 in the original, uncompressed set that can be tested. Therefore, the number of non-inflated patterns that can be tested as a function of compression level x, $P_1(x)$, is P_1 multiplied by this fraction:

$$ P_1(x) = P_1 \frac{M}{TDV_1(x)} = \frac{P_0 x}{1+\varepsilon_1 x} \quad 1 < x \le x_1 \quad (12) $$

The compression level x_1 is the compression level at which all P_1 patterns can tested, determined by solving $P_1(x)=P_1$:

$$ x_1 = \frac{P_1}{P_0 - P_1\varepsilon_1} \quad (13) $$

Notice that the number of inflated patterns loaded into memory, $P'(x)$, can be expressed in terms of P_0:

$$ \begin{aligned} P'(x) = P_1'(x) &= P_1(x)(1+\varepsilon_1 x) = \frac{P_0 x}{1+\varepsilon_1 x}\left(1+\varepsilon_1 x\right) \\ &= P_0 x \quad 1 < x \le x_1 \end{aligned} \quad (14) $$

Equation (14) confirms our expectations that the required compression up to x_1 is the ratio of the total number of test patterns $P'(x)$ to the number of test patterns P_0 that can be loaded into memory without compression.

How many patterns in test T_2 can be tested at compression level x? The answer is zero if $x \le x_1$. At higher compression levels, the number depends on the amount of available memory. This time we must account for the data volume already occupied by patterns in test T_1, $TDV_1(x)$, which decreases with compression even as the number of patterns continues to inflate. The fraction of T_2 patterns that can be loaded into memory is the amount of available memory, $M–TDV_1(x)$, divided by the test data volume at compression level x, $TDV_2(x)$:

$$\frac{M - TDV_1(x)}{TDV_2(x)} = \left(M - \frac{3FP_1(1+\varepsilon_1 x)}{x}\right) \div \left(\frac{3FP_2(1+\varepsilon_2 x)}{x}\right) = \frac{P_0 x - P_1(1+\varepsilon_1 x)}{P_2(1+\varepsilon_2 x)} \quad (15)$$

The number of non-inflated patterns that can be tested, $P_2(x)$, is P_2 multiplied by this fraction:

$$P_2(x) = P_2 \frac{M - TDV_1(x)}{TDV_2(x)} = \frac{P_0 x - P_1(1+\varepsilon_1 x)}{1+\varepsilon_2 x} \quad x_1 < x \le x_2 \quad (16)$$

The compression level x_2 is the compression level at which all P_2 patterns can tested, determined by solving $P_2(x)=P_2$:

$$x_2 = \frac{P_1 + P_2}{P_0 - P_1\varepsilon_1 - P_2\varepsilon_2} \quad (17)$$

The inflated pattern count $P'(x)$, expressed in terms of P_0, is:

$$\begin{aligned} P'(x) &= P_1'(x) + P_2'(x) \\ &= P_1(1+\varepsilon_1 x) + \frac{P_0 x - P_1(1+\varepsilon_1 x)}{1+\varepsilon_2 x}(1+\varepsilon_2 x) \\ &= P_1(1+\varepsilon_1 x) + P_0 x - P_1(1+\varepsilon_1 x) = P_0 x \quad 1 < x \le x_2 \end{aligned} \quad (18)$$

In general, for n tests, the number of non-inflated patterns in test T_n that can be tested as a function of compression level x is:

$$\begin{aligned} &P_n(x) = \frac{P_0 x - P_1(1+\varepsilon_1 x) - \cdots - P_{n-1}(1+\varepsilon_{n-1} x)}{1+\varepsilon_n x} \\ &x_{n-1} < x \le x_n, \quad \varepsilon_j < P_0 / P_j \\ &\text{and } x_n = \frac{P_1 + \cdots + P_n}{P_0 - P_1\varepsilon_1 - \cdots - P_n\varepsilon_n} \end{aligned} \quad (19)$$

The inflated pattern count as a function of compression level x is:

$$P'(x) = \begin{cases} P_0 x & 1 < x \le x_n \\ P_1(1+\varepsilon_1 x) + \cdots + P_n(1+\varepsilon_n x) & x > x_n \end{cases} \quad (20)$$

Now that we have defined the equations needed to account for the effects of pattern inflation from compressing multiple pattern sets, we can describe the composite fault coverage as a function of compression level ${}^n f(x)$ for test T_n. We replace the variable for pattern count P in the composite fault coverage function (8) with the expression for pattern count $P_n(x)$ in (19):

$${}^n f(x) = \begin{cases} \sum_{j=1}^{n-1} F_j(1-\varphi_j x) + F_n(1-\phi_n x)\left[1 - \exp(-\eta_n P_n(x))\right] \\ x_{n-1} < x \le x_n \\ \sum_{j=1}^{n-1} F_j(1-\varphi_j x) + F_n(1-\phi_n x)\left[1 - \exp(-\eta_n P_n)\right] \\ x > x_n \end{cases} \quad (21)$$

The pattern count numbers P_j reflect the number of ATPG patterns generated for each test in uncompressed mode. The $(1-\varphi_j x)$ multipliers account for any loss in fault coverage not directly related to pattern inflation. We must consider that in the absence of ideal X-tolerance, or when there is very high data dependency in the decompressors, fault coverage *might* decrease with compression even after inflation is factored in. This can be observed when the ATPG tool reaches successively lower maximum fault coverage as more compression is applied. We can describe this coverage loss for each test T_j using a simple linear approximation $\varphi_j(x)=\varphi_j x$, where φ_j is the fractional decrease in component coverage F_j per unit increase in scan compression that is independent of pattern inflation. Each φ_j is calculated from the individual fault coverages. For a three-test scenario,

$$\begin{aligned} \varphi_1 &= w_1\varphi_{11} + w_2\varphi_{12} + w_3\varphi_{13} \\ \varphi_2 &= w_2\varphi_{22} + w_3\varphi_{23} \\ \varphi_3 &= w_3\varphi_{33} \end{aligned} \quad (22)$$

VI. The Most Expensive Test

The trade-off central to this analysis is the trade-off between more compression required to load *all* test patterns of a test into limited tester memory, and less compression required to load a *subset* of test patterns of the same test into the same amount of memory. It seems reasonable to consider tests with the largest data volume and the highest fault coverage loss as candidates for pattern removal. We will refer to this category of tests as more "expensive" than the others. Based on the criterion for selecting *the most expensive test* developed in this section, removing patterns from this test achieves greater cost savings than removing patterns from any of the other tests.

A factor to consider beyond just the number of patterns in a test is the tendency for the component coverages F_j to decrease in magnitude going from T_1 to T_n when the number of faults associated with each fault model are similar. This occurs because patterns of tests closer the beginning of the test flow of Figure 2 are fault-graded on more fault models and thus tend to have more of an impact on total quality as measured by the proportion of total faults in the fault universe detected by their patterns. Patterns of tests closer to the end of the test flow typically have less *measurable* impact on total quality. For example, assuming both pattern counts are the same, more patterns must be pruned from test T_n (requiring less compression) than test T_1 to achieve the same incrementally-lower quality. Stated in economic terms, pruning patterns from T_n results in a lower silicon area overhead cost of compression than pruning patterns from T_1 to achieve the same incrementally-higher cost of escapes (refer to equation (27), below).

In light of these observations, we define the most expensive test as the test with the smallest incremental increase in quality (or, considering fault coverage loss at high compression levels, the largest incremental decrease in quality) at the extreme end of its component fault coverage convergence tail. From a notational perspective, it is convenient to temporarily denote a given test as T_n and then apply the criterion to that test. The incremental change in quality is proportional to the derivative of the composite fault coverage ${}^{n}f(x)$ at the compression level x_n required to compress and test all patterns. The derivative of the first expression of (21) at $x=x_n$ is:

$$\frac{d}{dx}\,{}^{n}f(x=x_n) = F_n\left[\frac{\eta_n P_0}{(1+\varepsilon_n x_n)^2}\exp(-\eta_n P_n)(1-\varphi_n x_n)-\varphi_n(1-\exp(-\eta_n P_n))\right] - \sum_{j=1}^{n-1} F_j\varphi_j \quad (23)$$

where x_n is defined in (19). The most expensive test is the test with derivative of lowest value calculated from (23).

It is important to realize that the most expensive test may not be the last test in the ATPG flow described in Section 3. Consider a simple two-pass sequence where T_1 consists of stuck-at (SA) patterns and T_2, the most expensive test, consists of transition delay (TD) patterns. The composite fault coverage from (4) is:

$${}^{2}f = \frac{1}{2}(f_{11}+f_{12})+\frac{1}{2}f_{22} \quad (24)$$

Because SA patterns normally do not detect transition delay faults, f_{12}=0 and the tests are essentially generated independently using their own fault models. We can reduce the total number of patterns by reversing the order of the tests: TD patterns are generated first, and then the undetected faults are targeted by the SA patterns (slow-to-rise faults are mapped to stuck-at-0 faults and slow-to-fall faults are mapped to stuck-at-1 faults). In this flow, $f_{11}=f_{12}$ because the total number of faults is the same for both models, so the component coverages are $F_1 = f_{11}$ and $F_2 = \frac{1}{2}f_{22}$. For the design examples of Section 9 we will use this flow, in which the most expensive test, consisting of TD patterns, is the first test in the ATPG sequence.

VII. The Optimal Compression Level

Now that we have defined the composite fault coverage as a function of compression ratio for the most expensive test, we substitute the first expression in (21) into the equation for the cost of test escapes C_{esc} in the TDVR phase [3, 4]:

$$C_{esc}(x) = M_{cost}(x)\alpha_{esc}E(Y(x),{}^{n}f(x)) \quad x_{n-1} < x \le x_n \quad (25)$$

This expression represents the cost of escapes going from one stage of the test process to the next (for example, wafer probe to system level test). M_{cost} is the cost to test and manufacture each good die, α_{esc} is the escape rate multiplier [4] representing the full impact of test escapes relative to the unit cost that went into their manufacture and test, and E is the test escape rate, measured in defective parts per million (DPPM). For the design examples of Section 9 we will use the Agrawal-Seth defect model [7] and the exponential yield equation [8] to predict the escape rate as a function of the manufactured yield $Y(x)$ and the fault coverage ${}^{n}f(x)$ of the applied test patterns:

$$E(Y,f) = \frac{(1-f)(1-Y)e^{-(n_0-1)f}}{Y+(1-f)(1-Y)e^{-(n_0-1)f}}, \quad Y = \frac{1}{1+AD} \quad (26)$$

where n_0 is the average number of faults per defective die, A is the die size (cm^2), and D is the manufacturing defect density (defects/cm^2).

The optimal compression level λ in the TDVR phase occurs at the compression level where the rate of increase in the silicon area overhead cost of compression $C_{silicon}$ equals the rate of decrease in the cost of escapes C_{esc}:

$$\frac{d}{dx}C_{silicon} = -\frac{d}{dx}C_{esc} \quad x_{n-1} < x \le x_n \quad (27)$$

Compressing a design at this level ensures that all but the last $P_0\cdot(x_n-\lambda)$ patterns in the most expensive test can be loaded into tester memory. The complexity of formula (26) and nonlinearities in the die size function $A(x)$ (see equation (1)) preclude an explicit formulation for λ in the TDVR phase. The most practical way to calculate it is to simply identify the compression level corresponding to the total test cost minimum.

It will be beneficial in Section 9 also to determine the optimal compression level in the TATR phase. This occurs at the level where the rate of increase in the silicon area overhead cost of compression $C_{silicon}$ equals the rate of decrease in test execution cost C_{exec}:

$$\frac{d}{dx}C_{silicon} = -\frac{d}{dx}C_{exec} \quad x > x_n \quad (28)$$

The solution to (28) is derived in the appendix assuming $A(x)$ is linear and $Y(x) \approx Y_0$:

$$\lambda \cong \sqrt{\frac{R_{act}\alpha_0 F(P_1+\cdots+P_n)}{C_s A_0 \gamma(2-Y_0)\mathrm{Cf}_{test}}} \quad \lambda > x_n \quad (29)$$

Notice that the optimal compression level in the TATR phase is virtually independent of the pattern inflation rates of the tests.

VIII. Pruning Patterns From More Than One Test

Greater cost reduction can be achieved by pruning *all* patterns that incrementally increase total test cost. The trade-off between test escape cost and area overhead cost of compression can be performed for each test, not just the most expensive test, to determine if it is cost-effective to prune patterns from the tail of its convergence curve. Although this approach offers greater cost savings, it requires up to n times more measurements.

We begin by ordering the tests from the least to the most expensive. We apply the criterion in Section 6 to all n tests to determine the most expensive test. We then temporarily remove this test from the set and apply the criterion on the reduced set to determine which of the remaining tests is the most expensive. This process is repeated until only the least expensive test remains. The tests are then ordered from the least expensive test, T_1, to most expensive test, T_n.

Referring to (21), the composite fault coverage function for test T_1 is:

$$ {}^{1}f(x) = F_1(1-\varphi_1 x)\left[1-\exp\left(\frac{-\eta_1 P_0 x}{1+\varepsilon_1 x}\right)\right] \quad 1 < x \le x_1 \qquad (30) $$

We can now determine λ_1, the optimal compression level for T_1, using the methodology presented in the previous section. There may not be a local cost minimum in this compression range, in which case $\lambda_1=x_1$ and no patterns should be removed. However, when $\lambda_1 < x_1$, the last $P_0\cdot(x_1-\lambda_1)$ inflated patterns in T_1 contribute to a net increase in total cost. Pruning these patterns effectively reduces P_1, the total number of non-inflated patterns in T_1 used for calculating the composite fault coverage, the cost of test escapes, and ultimately the optimal compression level for the succeeding pattern set.

The pared-down, non-inflated pattern count $P(\lambda_1)$ is derived by recognizing that the number of pruned patterns is equivalent to the difference in the inflated pattern count at x_1 based on the complete pattern set, P_1, and the inflated pattern count at λ_1 based on the pared-down set, $P(\lambda_1)$:

$$ P_0(x_1-\lambda_1) = P_1(1+\varepsilon_1 x_1) - P(\lambda_1)\cdot(1+\varepsilon_1\lambda_1) \qquad (31) $$

Solving for $P(\lambda_1)$ yields:

$$ P(\lambda_1) = \frac{P_1(1+\varepsilon_1 x_1) - P_0(x_1-\lambda_1)}{1+\varepsilon_1\lambda_1} \qquad (32) $$

We need to account for this reduced pattern count when calculating $P_2(x)$ and x_2 to determine the optimal compression level for the second test. From (16) and (17),

$$ P_2(x) = \frac{P_0 x - P(\lambda_1)\cdot(1+\varepsilon_1 x)}{1+\varepsilon_2 x} \quad \lambda_1 < x \le x_2, $$
$$ \text{where } x_2 = \frac{P(\lambda_1)+P_2}{P_0 - P(\lambda_1)\varepsilon_1 - P_2\varepsilon_2} \qquad (33) $$

The adjusted composite fault coverage for test T_2 is:

$$ {}^{2}f(x) = F_1(1-\varphi_1 x)\left[1-\exp\left(-\eta_1 P(\lambda_1)\right)\right] + $$
$$ F_2(1-\varphi_2 x)\left[1-\exp\left(\frac{-\eta_2\left(P_0 x - P(\lambda_1)\cdot(1+\varepsilon_1 x)\right)}{1+\varepsilon_2 x}\right)\right] \qquad (34) $$
$$ \lambda_1 < x \le x_2 $$

We now calculate λ_2 and continue this process for the remaining pattern sets. The last λ_n is the actual compression ratio used for the design. As we might expect, it is different from the one calculated using the method that removes patterns only from the most expensive test. When generating the final test program, ATPG for each test is halted at pattern number $P(\lambda_j)\cdot(1+\varepsilon_j\lambda_n)$. Alternatively, if ATPG is run to completion, $P_j(1+\varepsilon_j x_n)-P(\lambda_j)(1+\varepsilon_j\lambda_n)$ patterns are pruned from the end of each set.

IX. Design Examples

Equipped with the mathematical framework described in the preceding sections, we performed the cost trade-off on two test pattern sets applied to two industrial designs. Transition delay patterns (T_1) and stuck-at patterns (T_2) were generated as in the flow of Figure 2 using the Synopsys TetraMAX ATPG product, version 2006-06. The transition delay test was confirmed to be the most expensive test based on the criterion of Section 6. (To avoid confusion concerning the indices, "x_c" is used in this section to represent the compression level required to load all test patterns from both tests into memory.)

Table 1 summarizes the parameters used in the analyses. Estimates for initial die size, yield, and cost infrastructure parameters were shared among the design examples: A_0=1.0 cm^2, Y_0=77%, C_s=\$4.00 cm^{-2}, f_{test}=10 MHz, R_{act}=\$0.06 sec^{-1}, α_{esc}=20. The variables that impact compression cost savings—ε_j, φ_j, and γ—were extracted from least-squares fitting of data sets reflecting varying amounts of scan compression using the Synopsys DFT MAX adaptive scan compression product, version 2006-06. The exponential constant of convergence η_1 was extracted by curve-fitting $f_{11}(P)$ to the ATPG fault coverage-versus-pattern count data of the uncompressed designs. The composite fault coverages were calculated as described in Sections 4 and 6. The cost analysis of the design examples used the Agrawal-Seth model in (26) for escape rate assuming an average of n_0=4.0 faults per defective die for designs A and B. n_0 was chosen so the defect levels were approximately 700–1000 DPPM, assuming all patterns were compressed and used for testing the parts.

TABLE 1.

COST ANALYSIS OF THE DESIGN EXAMPLES

	Input Parameters					*Extracted Parameters*					*Results*			
Design	G_0 *	F	C	P_1 P_2	f_{11} ** f_{22}	η_1	ε_1 ε_2	φ_1 φ_2	γ	M	λ / x_c	Δ Cost	Δ DPPM	Discarded Patterns
A	5.697M	102,645	28	12,390 637	89.92% 10.02%	0.0007 -	0.012 0.005	0.00003 0.00244	0.00038	128 256 512 1024 2048	20 / 42 11 / 17 6 / 8 10 / 4 10 / 2	23% 8% 2% 18% 43%	0% 2% 1% 0% 0%	51.8% 36.4% 25.4% - -
B	1.485M	25,123	14	18,372 409	92.80% 7.05%	0.0004 -	0.012 0.035	0.00095 0.00333	0.00060	64 128 256 512 1024 2048	17 / 32 9 / 14 5 / 6 8 / 3 8 / 2 8 / 1	28% 11% 3% 16% 44% 68%	-9% 1% 2% 0% 0% 0%	48.7% 33.3% 19.9% - - -

* Proxy for die size, measured in Design Compiler area units, used to calculate γ. Each design referenced a separate DC library.
** Fault coverage = detected faults/detectable faults for uncompressed design. Faults in compression logic were excluded in compression runs.

We begin by noting that for sufficiently large tester memory M, additional cost savings from compression will always occur in the TATR phase where the optimal compression level λ is virtually unaffected by pattern inflation. Equation (29) predicts $\lambda \cong 11$ for design A, and the data in Table 1 indicates that if the tester has at least M=1 Gb available for digital testing, we do not need to consider truncating patterns to reduce cost. However, in circumstances where test data volume is relatively high compared with the amount of available tester memory, to minimize total cost we will need to examine the fault coverage convergence characteristic of the most expensive test in more detail. To give us a better perspective of what happens when we vary the amount of tester memory, Figure 6 displays the compression curves of design A using M as the independent variable.

The graph shows two curves: λ, the optimal compression level, and x_c, the compression level needed to fully load all test pattern sets into tester memory. Values of M to the left of the vertical line correspond to savings from test data volume reduction. When M is at a low value such as 128 Mb, a relatively high amount of compression x_c=42 is required to load all the patterns. With only this much memory, the total costs of test, C_{esc} and $C_{silicon}$, are minimized by discarding patterns at the tail of the fault coverage convergence curve of the transition delay test, corresponding to compression levels above λ=20 (Figure 1 graphs the test costs for this scenario).

As more memory becomes available for testing, both curves decline and the gap between them diminishes to its minimum near M=512 Mb. Just above this level, λ is closest to x_c and virtually all the patterns can be loaded onto the tester at the compression level that minimizes test costs. At incrementally higher memory, the optimal compression level no longer depends on fault coverage and it increases to the level of λ=10, where the sum of C_{exec} and $C_{silicon}$ are minimized. With more memory, savings from test application time reduction are possible and so the gap between λ and x_c increases as x_c declines asymptotically.

The gap between λ and x_c in Figure 6 corresponds to cost savings incurred by compressing at the optimal level instead of increasing (or decreasing) compression to the level at which all test patterns can be loaded into tester memory. The wider this gap, the greater the cost savings, and we see that the widest gaps occur at the extreme ends of M values. Table 1 displays cost savings—the difference in costs at λ and x_c expressed as a percentage of the total test costs at x_c—from compression optimization. For design A using M=128 Mb tester memory, 23% lower test costs are possible by implementing compression of λ=20 instead of x_c=42. This means that 51.8% of the total test patterns, all from the most expensive test set, are discarded, resulting in a negligible increase in the defect level.

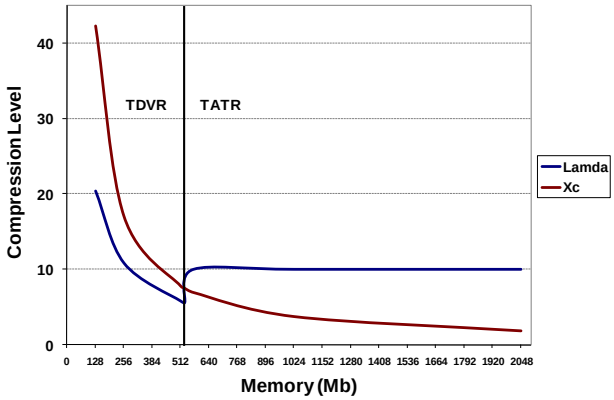

Fig. 6. Compression signatures for design A.

For both designs, and especially for design B, the model predicts a decline in defect level for the lowest M value when truncating patterns, compared with loading the complete pattern set. This is due to fault coverage loss that occurs at the highest compression levels. The transition delay test must accommodate many unknown logic states in design B, resulting in a φ_1 that is more than 30 times higher than that of design A.

Also note that values of the area scaling coefficients γ in Table 1 were extracted by measuring the fractional increase in gate count with compression. Area measurements of the physical circuits were needed to estimate the second-order area scaling coefficients ζ. Since these measurements were not made, we assumed ζ were negligible for this analysis. However, keep in mind this is typically not the case; for higher compression levels, significant wire routing congestion can increase the silicon area overhead nonlinearly and thus lower λ for all memory levels, increasing the cost savings gap between λ and x_c in the TDVR phase and decreasing it in the TATR phase.

The results suggest the proposed technique for performing cost trade-off analysis of multiple test pattern sets *may* lower test costs. A major assumption in the analysis is that the composite fault coverage of multiple tests correlates reasonably well with test escapes. Empirical data is needed to support this assumption. Moreover, the compression cost model assumes it is possible to reliably estimate the cost of test escapes, which may not be feasible in certain situations. Finally, the author does not recommend pruning transition delay patterns per se; the analysis is intended to be performed in the presence of *many* pattern sets when tester memory is a constrained resource.

X. Conclusions

We presented a mathematical framework for evaluating the economic impact of adding more defect-based test pattern sets to incrementally improve test quality. We demonstrated that when the volume of test data is large enough relative to the amount of available tester memory to require a high compression level to store all the patterns, then *the most cost-effective compression strategy is to compress and test only a subset of the generated patterns.* Removing patterns in the fault coverage convergence tail of the most expensive test, or a combination of tests, reduces the impact on test quality; the number removed is determined by the compression level at which the incremental area overhead cost of compression equals the incremental saving due to quality improvement.

We believe this methodology may be beneficial in the years ahead as more tests are applied and total test data volume increases. However, the notion of discarding test patterns is an anathema to many designers because it violates a longstanding assumption that the highest possible quality is always better if it is easily achieved. Indeed, for many applications, highest quality *is* always better, if not always more *cost-effective*. And since there is uncertainty associated with defect level in the tail of the fault coverage convergence curve—and assignment of a dollar cost to that defect level—designers may elect not to prune patterns when the incremental costs of their inclusion are judged to be relatively small compared with the indeterminate costs of their exclusion [9].

Even so, firms that value test quality already place practical limits on maximum quality as reflected by their level of investment in DFT resources dedicated to improving test quality for each design. Our analysis formalizes this already implicit trade-off for high-volume, cost-driven environments. In fact, even if the sensitivity of a firm to test escapes is exceptionally high—a scenario characterized by the test escape multiplier α_{esc} approaching infinity—*the optimal compression level will always be less than x_c if test data volume is high enough relative to tester memory resources.* In other words, for any design there is a certain amount of available tester memory below which additional cost is incurred by adding patterns to improve test quality above the level provided by optimal compression. And if memory is not a factor, significant cost savings from test time reduction are attainable by increasing compression from x_c to the optimal level that satisfies (29).

XI. Appendix

Derivation of λ in the TATR Phase

It is possible to solve the equivalency of (28) explicitly if we assume that $Y(x) \approx Y_0$ and there is negligible non-linear area overhead of compression ($\zeta \approx 0$ in the formulation for $A(x)$ in equation (1)). Under these conditions, the area overhead cost of compression is described by [3]:

$$C_{silicon}(x) \cong \frac{C_s A_0}{Y_0}(2 - Y_0)(A_F/A_0 + \gamma x) \quad x > 1 \qquad (35)$$

Test execution cost as a function of compression level is directly proportional to the test execution time $T_{test}(x)$ required to test $P'(x)$ inflated patterns [3]:

$$C_{exec}(x) = \frac{R_{act}}{Y(x)} T_{test}(x) = \frac{R_{act}}{Y(x)} \frac{T_0}{P_0} \alpha(x) \left(\frac{P'(x)}{x} \right) \quad x > 1 \quad (36)$$

where R_{act} is the cost per second of active testers and T_0 is the time to execute all P_0 patterns without compression, given by (2). The multiplier $\alpha(x)$ is used to account for a slight decrease on average in the test execution time due to less time spent testing failing die ($Y(x) \le \alpha(x) \le 1$). If we assume that $\alpha(x) \approx \alpha_0$ and $Y(x) \approx Y_0$, then substituting the expression for $P'(x)$ in (20), we have:

$$C_{exec}(x) = \frac{R_{act}}{Y_0} \frac{T_0}{P_0} \alpha_0 \left(\frac{P_1(1+\varepsilon_1 x) + \cdots + P_n(1+\varepsilon_n x)}{x} \right) \quad x > x_n \quad (37)$$

Differentiating the formulas for $C_{silicon}(x)$ and $C_{exec}(x)$ and solving for x, we obtain the approximation for λ in the TATR phase:

$$\lambda \cong \sqrt{\frac{R_{act}\alpha_0}{C_s A_0 \gamma (2 - Y_0)} \left(\frac{T_0}{P_0} \right) (P_1 + \cdots + P_n)} \qquad (38)$$

The expression in (29) makes use of the relation between T_0 and P_0 defined in (2).

ACKNOWLEDGEMENTS

The author would like to thank Tim Ayres, Synopsys principal engineer, for his valuable suggestions on ways to improve this content, and Felix Ng, Synopsys corporate applications engineer, for his assistance in preparing the design data.

REFERENCES

[1] P.C. Maxwell, V. Johansen, I. Chiang, "The Effectiveness of I_{DDQ}, Functional and Scan Tests: How Many Fault Coverages Do We Need?" *Proc. Int'l Test Conf.*, pp. 168-177, 1992.

[2] P. Maxwell, R. Aitken, V. Johansen, C. Inshen, "The Effect of Different Test Sets on Quality Level Prediction: When is 80% Better than 90%?" *Proc. Int'l Test Conf.*, pp. 358-364, 1991.

[3] C. Allsup, "The Economics of Implementing Scan Compression to Reduce Test Data Volume and Test Application Time," *Proc. Int'l Test Conf.*, Lecture 2.2, 2006.

[4] S. Wei, P.K. Nag, R.D. Blanton, A. Gattiker and W. Maly, "To DFT or Not to DFT?" *Proc. Int'l Test Conf.*, pp. 557-566, 1997.

[5] I.D. Dear, C. Dislis, A.P. Ambler, J. Dick, "Economic Effects in Design and Test," *IEEE Design & Test of Computers*, Volume 8, Issue 4, Dec. 1991, pp. 64-77.

[6] C. Allsup, "Optimizing Compression in Scan-Based ATPG DFT Implementations," *Test & Measurement World*, March 2007.

[7] V.D. Agrawal, S.C. Seth, P. Agrawal, "Fault coverage requirement in production testing of LSI circuits," *IEEE Journal of Solid-State Circuits*, Volume 17, Issue 1, Feb. 1982, pp. 57-61.

[8] T.M Michalka, R.C. Varshney, J.D. Meindl, "A Discussion of Yield Modeling with Defect Clustering, Circuit Repair, and Circuit Redundancy," IEEE Transactions on Semiconductor Manufacturing, Vol. 3, No. 3, Aug. 1990, pp. 116-127.

[9] S. Davidson, A. Ambler, H. Davidson, "Behavioral Test Economics," *Proc. Int'l Test Conf.*, Paper 1.3. 2006.