Statistical Test To Uncover Process Variations

Vikram Iyengar[1] Jinjun Xiong[2] Subbayyan Venkatesan[3] Vladimir Zolotov[2]
David Lackey[1] Peter Habitz[1] Chandu Visweswariah[2]
[1,3]IBM Microelectronics [2]IBM Thomas J. Watson Research Center
[1]Essex Junction, VT [2]Yorktown Heights, NY [3]San Jose, CA
{*vikrami, jinjun, venki, zolotov, delacke, habitz, chandu*}@*us.ibm.com*

***Abstract*— Meeting the tight performance specifications mandated by the customer is critical for contract manufactured ASICs. To address this, at speed test has been employed to detect subtle delay failures in manufacturing. However, the increasing process spread in advanced nanometer ASICs poses considerable challenges to predicting hardware performance from timing models. Performance verification in the presence of process variation is difficult because the critical path is no longer unique. Different paths become frequency limiting in different process corners. In this paper, we present a novel variation-aware method based on statistical timing to select critical paths for structural test. Node criticalities are computed to determine the probabilities of different circuit nodes being on the critical path across process variation. Moreover, path delays are projected into different process corners using their linear delay function forms. Experimental results for three multimillion gate ASICs demonstrate the effectiveness of our methods.**

I. Introduction

In contract manufacturing of application-specific integrated circuits (ASICs), the customer exclusively contracts the ASIC manufacturer for a product. The ASIC netlist is owned and designed by the customer, implemented in the manufacturer's cell libraries, and fabricated and tested by the manufacturer. Satisfying the strict performance specifications mandated by the customer while meeting the contracted test cost is critical.

However, achieving the specified performance requirements for advanced nanometer ASICs poses considerable challenges. The effects of process and environmental variation are increasing with each new technology generation [1]. The number of significant sources of variation is also increasing, leading to a variational space of very high dimensionality and a wide distribution in ASIC performance [1]. Delay measurements of simple test structures, such as ring oscillators, can no longer guarantee adequate timing performance. Certain process and environmental parameters may in fact affect the ASIC logic differently than its test structures. Moreover, for contract manufactured ASICs, expert knowledge of the design's functionality is not available. The performance verification methodology must therefore be applicable to a wide variety of designs. Furthermore, the test cost must adhere to customer-established budgets.

At-speed test has become a popular low-cost approach to detect subtle delay failures in manufacturing [2], [3]. However, at-speed test for catastrophic defects does not address the problem of delays caused by process variation [4]. In advanced nanometer manufacturing, systematic delays owing to unpredictable process changes have assumed a greater share of customer fails than catastrophic defects [5]. Moreover, at-speed transition fault test is not guaranteed to exercise critical paths required to measure performance [6].

Path delay test has been proposed to verify performance [6], [7]. However, path selection based on deterministic timing cannot capture shifts in the critical path caused by process variation. Recently, several methods for critical path identification based on statistical modeling of delays have been proposed. These are either based on enumerating long paths through each gate [8], using worst case slacks to identify critical nodes [9], or modeling delay defect size as a random variable [4]. However, testing paths through each circuit gate is economically infeasible. Current 90 nm ASICs can have as many as 25 million gates, potentially having billions of paths. Methods based on worst case slacks or modeling delay defect size as a random variable do not consider process and environmental parameters. Overall, there is a pressing need for an integrated approach to variation-aware performance verification combined with a low-cost at-speed test methodology. Furthermore, the approach must address a wide variety of ASIC designs.

In this paper, we present an integrated approach to at-speed structural test (ASST) for performance verification, explicitly considering a multidimensional process space. The first objective of this work is to uncover performance violations in a defect-free chip arising from the systematic aggregation of very small delay changes on interconnects and gates affected by the same process variations. One set of applications is for speed sorting to cull high-performance products for high-value customers. The second objective of this work is to enrich the transition fault test suite with a few path delay test patterns targeting minute delay defects on critical paths. The third objective of this work is to provide performance feedback from manufacturing to reduce pessimism in timing models and to drive design changes based on observed performance. This would be used to increase the parametric yield. This paper represents the first reported successful integration of an industrial statistical timing tool with a proven low-cost at-speed test methodology that is applicable to a wide variety of contract manufactured ASICs.

This paper is organized as follows. In Section II, we review related prior work. In Section III, we introduce the integrated performance verification methodology. In Section IV, we describe the at-speed test architecture on which the integrated methodology is based. In Section V, we review parameterized statistical timing. In Section VI, we describe the computation of critical probabilities for paths and nodes. In Section VII, we present the proposed method to identify paths in the design that are critical across the entire process and environmental parameter space. A method to avoid false paths is described. Finally, in Section VIII, we present experimental results for three multimillion gate ASICs.

II. Prior work

At-speed test for delay faults is being widely carried out in the industry. In [10], methods for at-speed deterministic test are reported. In [3], the authors describe at-speed test for Freescale's e600 core. In [2], an ASST methodology for contract manufactured ASICs is presented. The new methods for performance verification presented in this paper are based on [2]. In general, the existing approaches are targeted at gross defect detection, and use the transition fault model for test generation. Critical paths are not guaranteed to be exercised. For performance verification, testing the critical paths is important to determine whether the manufactured ASICs meet the performance requirements.

Path selection for delay test has been rigorously investigated. In [11], a method based on graph traversal to identify the longest path through a gate is reported. In [6], critical path selection by ranking endpoints in order of slack is studied. In [7], the authors compare the delays of the longest sequential and combinational testable paths through each gate. In general, the prior work based on deterministic timing is not variation-aware. Hence, shifts in the critical path with process variation cannot be captured.

The delays of different paths on a chip have been observed to correlate if the paths pass through the same gates and interconnects [4]. Hence, recent methods for critical path selection have focused on statistical modeling of delays. In [8], a statistical fault coverage metric based on identifying the longest path per gate is presented. However, this would lead to an explosion in the number of paths tested for large circuits. Testing a large number of paths would exceed the test cost budget established by the customer. In [9], false-path-aware statistical timing is used to select critical paths. The method uses worst case slacks to identify critical nodes and does not consider process or environmental parameters. Next, in [4], a method to model delay defects as random variables is presented. Performance violation due to process variation in defect-free circuits is not investigated. In both [9] and [4], Monte-Carlo simulation is used to identify critical paths. This is inefficient for large circuits. Finally, in [12], a method to identify the longest paths through every gate under process variation is described. However, testing paths through each gate is infeasible in terms of test data volume and testing time for current 90 nm ASICs that can have millions of gates.

In timing analysis of advanced nanometer ASICs, it is important to consider process and environmental parameters such as transistor channel length, supply voltage, and metal thickness [1]. Statistical timing has been proposed to compute the statistical characteristics of arrival times (ATs), required arrival times (RATs), and slacks as a function of the process [13].

Research in statistical timing has also resulted in the concept of criticality [13]. The *criticality* of a node is the probability that this node lies on the critical path over the entire process space. More recently, an efficient algorithm to compute node criticalities has been presented [14].

In this paper, we use block-based statistical timing based on a detailed statistical model of gate delays, taking into account multiple process parameters exhibiting chip-to-chip and intra-chip variation with possible spatial correlation. We propose an efficient algorithm for computing the criticalities of circuit nodes using the results of statistical timing. Our performance verification methodology is based on testing critical paths passing through nodes of high criticality. To the best of our knowledge, this paper represents the first reported successful integration of an industrial statistical timing tool with a low-cost test architecture applicable to a wide variety of ASICs.

III. Integrated Methodology

The performance verification problem that we address in this paper can be formally expressed as follows.

Problem 1: *Given* an ASIC netlist, timing models for library cells, a frequency requirement for each clock domain, and a contracted test cost budget,

Create a methodology in which the performance of each clock domain is verified for chips from across the entire process and environmental parameter variational space,

Such that the test cost budget is not exceeded, and any hardware overhead and design-for-test (DFT) changes to the netlist are minimized.

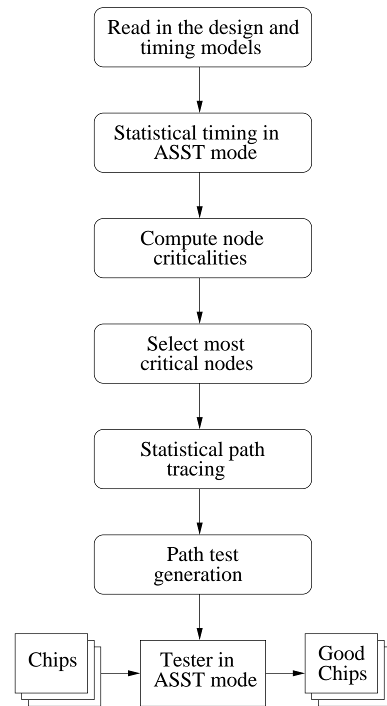

Fig. 1. Performance verification methodology.

To solve **Problem 1**, we propose the integrated methodology illustrated in Figure 1. The process begins by reading in the design netlist, timing models for library cells, and related timing assertions and constraints. We then run statistical timing in a special ASST mode to compute the signal arrival times as functions of random variables. Since we use a low-cost tester to satisfy the constraint on test cost in **Problem 1**, at-speed patterns cannot be applied to chip IOs from tester pins. Hence, only flop-to-flop paths are timed in this special mode. Moreover, certain timing constraints, such as clock-gating setup and hold cannot be tested by ASST. Hence, these checks are disabled for ASST timing. From the statistical distributions of the random variables, we can predict the probability of signals being correct at the flip-flops for a given clock frequency. This enables us to compute the criticality of every node in the design. The circuit nodes having the highest criticalities are selected and statistically-critical paths are traced through them. After tracing the critical paths, we perform test generation to obtain the test patterns to be applied for performance verification.

After path test generation, the test patterns obtained are loaded into the tester. For ASST, the low-cost tester scans each pattern into the chip at a low tester frequency and then hands over control to the on-chip functional clock generation logic. The phase-locked loops (PLLs) on the chip are activated to begin a launch–capture sequence, such that the path-under-test is clocked at-speed. The PLLs can be programmed to run faster or slower to determine the failing frequency of each path. After each pattern has been applied at-speed, the test responses are scanned out at a low tester frequency and compared to the expected results. In the following sections, we describe each step of the integrated methodology in detail.

IV. At-Speed Structural Test

Here, we introduce the ASST architecture [2] used for performance verification in our integrated methodology. ASST, illustrated in Figure 2, is used to exercise each clock domain of the ASIC at its

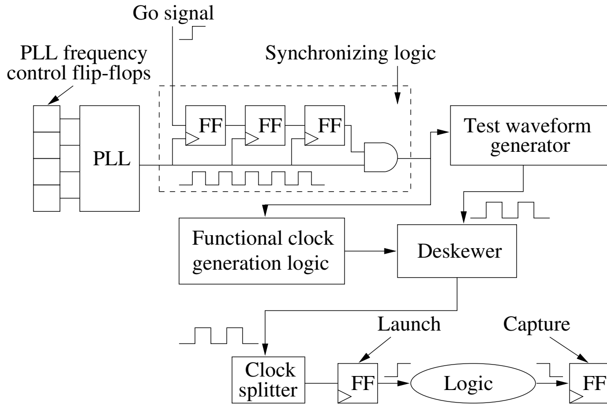

Fig. 2. Illustration of ASST architecture.

specified performance while using a low-cost tester. We first scan the functional programming values in to the PLL frequency control flip-flops and lock the PLLs for correct operation. Once the PLLs are operational, an asynchronous GO signal is asserted by the tester to begin an at-speed (or faster) functional clock sequence.

The approach is highly scalable to any number of logic domains-under-test. Any number of on-chip PLLs can be accommodated. Moreover, PLL frequency programming is separated from test waveform generation; hence, PLLs can easily be reprogrammed to run faster or slower than at-speed to determine when the critical paths fail. This is important to provide feedback on how accurately the timing models match the manufactured performance. This feedback allows us to tweak timing models to account for observed process variation. *Any* functional clock waveform for n clock cycles can be derived from the PLL to be applied to different clock domains. The method uses a low-cost tester, requires no functional design changes by the customer, and imposes little hardware overhead, thus satisfying the constraints imposed by **Problem 1**. Moreover, each clock domain on the ASIC has its own deskewer. Critical paths between synchronous clock domains can therefore be tested without the need to use functional clock generation logic (customer IP not made available to contract manufacturers–see Figure 2).

V. STATISTICAL TIMING

In this section, we introduce parameterized statistical timing and its application to path selection for ASST. Deterministic timing models delays, ATs and RATs as simple scalar numbers. On the other hand, statistical timing models all time quantities as linear function canonical forms of variational parameters. For example, a timing quantity D in statistical timing may be expressed as

$$D = d_0 + \sum_{i=1}^{n} d_i X_i + \sum_{j=1}^{n} d_{s,j} X_{s,j} + d_r X_r. \quad (1)$$

Here, d_0 is the mean value of timing quantity D; d_i are D's sensitivities to random variables X_i that represent globally correlated chip-to-chip process variation; $d_{s,j}$ are D's sensitivities to random variables $X_{s,j}$ that model intra-chip spatial variation; and d_r is D's sensitivity to X_r that models uncorrelated random variation. Without loss of generality, in this work, we assume all sources of variation follow the Gaussian distribution. However, our approach is not limited to any particular variation model and can be easily extended to more general models [15].

Once delays are modeled in the canonical form of Equation (1), statistical timing is performed using statistical equivalents of the max, min, addition and subtraction operations for delays, ATs and RATs. By Monte Carlo simulation and hardware correlation, this method has been shown to be efficient and accurate.

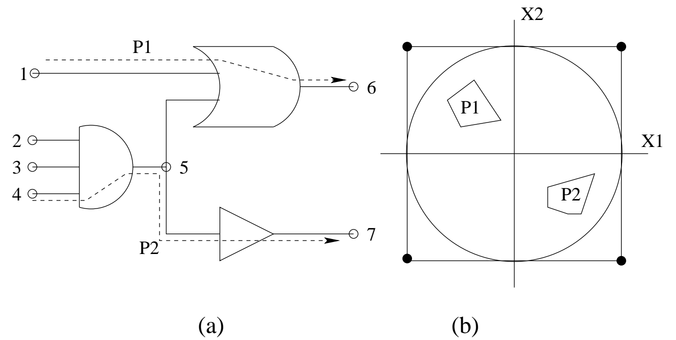

Fig. 3. Critical paths and process subspaces.

VI. CRITICALITY PROBABILITIES

In this section, we explain how statistical timing and the concept of criticality are used to identify circuit nodes that lie on the critical path.

The objective of our work is to verify the performance of manufactured ASICs over the entire process space. To do this, we must test the delays of critical paths. However, different paths may become frequency-limiting in different parts of the process space. For example, Figure 3(a) illustrates a circuit having two long paths labeled P1 and P2. The delays of paths P1 and P2 are functions of two process parameters X1 and X2, e.g., transistor channel length, metal thickness, etc. The two-dimensional process space for X1 and X2 is illustrated in Figure 3(b). Path P1 has the longer delay when the values of X1 and X2 lie in the area in the upper-left quadrant of the process space. However, Path P2 is the critical path when the values of X1 and X2 lie in the area in the lower-right quadrant. Hence, the problem of variation-aware performance verification becomes that of testing the paths that have the highest probability of being critical over the entire process space.

To identify critical paths more efficiently, we introduce the metric of node criticality. We define *node criticality* as the probability that a certain node will lie on the critical path among all manufactured chips. In this section, we show that there exists an efficient algorithm to compute node criticalities.

In statistical timing, a circuit is modeled as a timing graph $G(N, E, n_s, n_f)$, where N is a set of nodes and E is a set of edges weighted by delays of gates and wires. Figure 4 shows an example combinational circuit and its timing graph. Graph G has a virtual source node n_s connected to all the primary inputs (PIs) by virtual edges. The delays on these edges are the ATs at the respective PIs. The virtual sink node n_f is connected to all the primary outputs (POs) by virtual edges. The delays of these edges are the negative of the RATs of the respective POs.

For any edge e of interest, we divide all the paths of the timing graph into the subset S_e passing through edge e, and the subset $\overline{S_e}$ not passing through edge e. The criticality of edge e is the probability that the maximum path delay over all the paths in S_e is larger than the maximum path delay over all the paths in $\overline{S_e}$. The maximum delay $d(G_e)$ of the paths passing through edge $e = (i, t)$ (see Figure 5) is called edge slack and is expressed as

$$d(G_e) = d_{to} + d(e) + d_{from}. \quad (2)$$

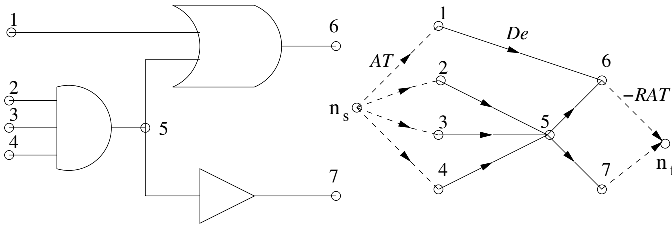

Fig. 4. Combinational circuit and timing graph.

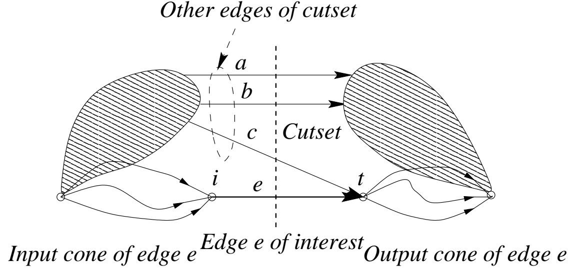

Fig. 5. Illustration of edge slack computation.

Here, d_{to} is the maximum delay from the source node to node i, $d(e)$ is the delay of edge e, and d_{from} is the maximum delay from node t to the sink node. Using ATs and RATs, Equation (2) can be rewritten as

$$d(G_e) = AT(i) + d(e) - RAT(t). \tag{3}$$

Here, $AT(i)$ is the AT at node i and $RAT(t)$ is the RAT at node t. Edge slack computation is illustrated in Figure 5. Statistical timing provides us with all ATs and RATs in parameterized canonical form. Therefore, we can compute the maximum path delay over all the paths passing though any edge in parameterized canonical form as well. To calculate the maximum delay of the paths not passing through edge e, we construct a mimimal cutset C_e containing edge e and separating nodes n_s and n_f from the timing graph. Each path from n_s to n_f has only one common edge with a miminmal cutset. This construction is illustrated in Figure 5.

Let $C_{\overline{e}} = C_e - \{e\}$ be a set of all cutset edges except e. Then the set of paths passing through the edges of $C_{\overline{e}}$ includes all the paths of the timing graph except the paths passing through the edge e, which is exactly the set of paths $\overline{S_e}$. The maximum delay of the paths passing through the set of edges $C_{\overline{e}}$ is expressed as

$$d(C_{\overline{e}}) = \max d(G_k). \tag{4}$$

Here, $d(G_k)$ is the maximum path delay over all paths passing through the kth edge of the set $C_{\overline{e}}$. The value of $d(G_k)$ can be computed in canonical form by Equation (3). Therefore, the maximum delay of the paths not passing through edge e can also be computed in canonical form. In [14], it is shown that this computation can be performed in linear time.

Knowing the canonical forms of the maximum delays $d(G_e)$ and $d(\overline{G_e})$, the edge criticality is the probability of $d(G_e) > d(\overline{G_e})$, or the *tightness probability* T_e expressed as

$$T_e = \Phi\left[\frac{s_{e,0} - s_{\overline{e},0}}{\sqrt{\sigma_e^2 + \sigma_{\overline{e}}^2 - 2\delta}}\right]. \tag{5}$$

Here, Φ is the cumulative probability distribution function of a standard normal distribution; $s_{e,0}$ and $s_{\overline{e},0}$, and σ_e^2 and $\sigma_{\overline{e}}^2$ are the mean and variance of $d(G_e)$ and $d(\overline{G_e})$, respectively; δ is the covariance between $d(G_e)$ and $d(\overline{G_e})$.

```
Procedure Critical_path_identification()
1 Let set of paths P be empty;
2 Identify a set of ASST-testable clock domains;
Identify critical paths
3 For each ASST-testable clock domain
4    Perform statistical timing in ASST mode;
5    Compute criticality probabilities for all nodes;
6    Select a set of nodes ranked most critical;
7    For each critical node N_i
8      Find all capture flops fed by this node;
9      For each capture flop F_j
10       Identify a number of least-slack flop-to-flop
           paths that pass through N_i and end at F_j;
Avoid false paths
11       For each path p identified in Line 10
12         If each node or flip-flop in p appears
            no more than r times in set P
13           Add path p to set P;
```

Fig. 6. Critical path identification.

VII. Statistical Path Tracing

Having identified the critical timing nodes in the circuit in Section VI, the next step is to trace the critical paths that pass through each of these nodes.

A. Critical paths

The variation-aware method for critical path identification is presented in the form of pseudocode in Figure 6. Here, we first describe Lines 1–10 of Figure 6 that are related to identifying critical paths.

In Line 1 of Figure 6, we initialize the set **P** of paths to empty. As critical paths are progressively identified, they are added to **P**. To meet the strict test cost budget established by the customer, we use low-cost testers. ASST operates at the functional speed of the ASIC, however, low-cost testers are unable to apply test patterns to chip IOs at functional speeds. Hence, only internal flop-to-flop paths are tested by ASST. Moreover, timing relationships between functional clock edges belonging to asynchronous PLLs cannot be predicted. Test across asynchronous clock domains is therefore not possible. To address this issue, we test only intra-domain critical paths for which the launch and capture flip-flops are clocked by the same deskewer. Critical path selection therefore begins in Line 2 of Figure 6 with the identification of the ASST-testable clock domains.

Not all the statistical timing tests are testable by ASST. For example, functional clock gating is disabled for ASST. Therefore, we enhance the timing tool with a special timing mode that analyzes only ASST-testable timing tests. In our special timing mode, three timing test types are enabled, namely, setup, hold, and end-of-cycle. In Line 4 of Figure 6, we perform statistical timing in this special ASST mode.

In Line 5, we compute the criticalities for all the nodes in a given clock domain, as described in Section VI. We then select a set of nodes ranked most critical in Line 6. In Lines 7-8, for each selected critical node N_i, we identify all the capture flops F_j at the endpoints of paths that pass through N_i. The rationale behind using node criticalities is that node criticality is a measure of the probability that the most critical path passes through this node. Hence, transitions

on all the paths that pass through this node must be captured by one of the flip-flops represented by F_j. By tracing paths in this manner, we select the most probable critical paths over the entire process space. Finally, in Line 10, for each such capture flop F_j, we trace a number of flop-to-flop paths that pass through node N_i and end at flip-flop F_j.

Path tracing is achieved by a depth-first traversal of the timing graph. As the traversal proceeds, a tree data structure is maintained, and the most promising side branch is explored at each node in order to list paths in slack order. Since slacks in statistical timing are probability distributions, a projected slack measure is employed to determine which side branch is most promising. Projection computes the value of a canonical form at some point of the process variation space by substituting the coordinates of this point into Equation (1). For example, timing slack D can be projected onto its worst process corner by

$$D = d_0 - \sum_{i=1}^{n} k|d_i| - \sum_{j=1}^{n} k|d_{s,j}| - k|d_r|. \quad (6)$$

Here, k is the deviation of process parameters expressed in sigmas. In order to avoid paths that start at PIs or terminate at POs, which are not testable by ASST, we find all the flip-flops feeding the node of interest or all the flip-flops fed by the node of interest, and then iterate over such flops while listing paths that either originate or terminate at the flip-flop and also pass through the critical node of interest. The number of critical nodes chosen, and the number of paths listed through each of these nodes is user-selected.

We next describe Lines 11–13 of Figure 6 that are related to avoiding false paths.

B. Avoiding false paths

An inherent problem in selecting critical paths from timing is that a large number of the paths identified by timing tools are false and cannot be sensitized. False paths can be classified as Boolean untrue paths and functional false paths [6]. A *Boolean untrue* path is an impossible propagation path that cannot be exercised either during functional operation or test, while a *functional false* path has no importance for functional operation, however, it may be exercised in test.

In an early implementation of our path tracing method, several functional false paths were found that included flip-flops not allowed to launch or capture in ASST. Such flip-flops are shut down in ASST for several reasons: (i) PLL control flip-flops (see Figure 2) are not allowed to be disturbed, (ii) DFT logic, e.g., boundary scan is not designed to operate at-speed, and (iii) certain flip-flops that are clocked by customer IP cores instead of PLLs may be shut down [2]. These flip-flops are shut down for ASST by using the *ASST* signal (of Figure **??**) to gate off their C (capture) clocks at the inputs of the clock splitter as shown in Figure 7. In level-sensitive scan design (LSSD), a clock splitter is used to select between test clocks (A, B and C) and the functional clock (OSC) shown in Figure 7. A combination of *ASST* and the scan enable (SE) signal is used to gate off the B (shift) clocks so that the flip-flops can still be scanned [2].

To avoid these functional false paths, we use the method of constant propagation to perform statistical timing in the special ASST mode (Line 4 of Figure 6). In this mode, signals such as *ASST* that have special values are asserted so that only the ASST-testable logic is timed. Hence, flip-flops that are not clocked during ASST do not appear in the critical paths identified.

A large number of identified critical paths were also found to be Boolean untrue because they pass through reconvergent logic. For example, the path illustrated in Figure 8 passes through the Data 1 input of a multiplexer, whose select pin is set to 0 to sensitize the path at the downstream AND gate. The timing tool does not perform logic simulation, and hence cannot identify this logic segment as untestable. Moreover, in several instances, a large number of nodes within the same reconvergent logic segment, such as around the mux and AND gate in Figure 8, were deemed critical by the timing tool; hence a large number of paths having the same reconvergent logic segment appeared in set **P**.

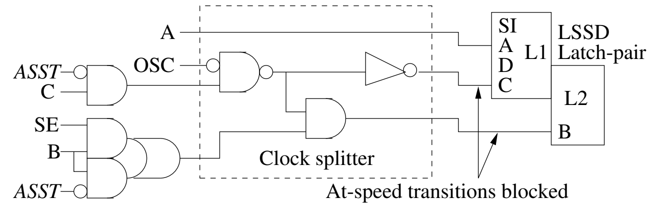

Fig. 7. Flip-flops that are shut down for ASST.

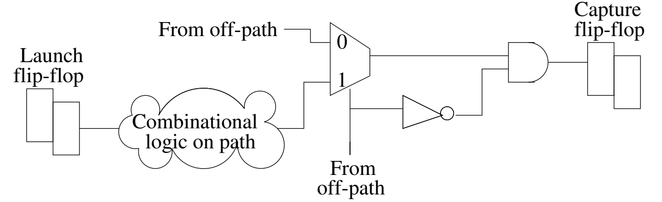

Fig. 8. Reconvergent logic segment.

To minimize the number of Boolean untrue paths, in Lines 11–13 of Figure 6, we add a new path to set **P** only if each node or flip-flop in the new path has appeared no more than r times already in set **P**. This filtering mechanism has two advantages. Firstly, it ensures that the set of critical paths identified is not dominated by a few repeating untestable logic segments. Secondly, the topological coverage of the critical path set is improved by spreading the critical paths out over a larger number of nodes and flip-flops. Moreover, a node or flip-flop is not filtered out until r paths including it have been identified. Hence, the path coverage per critical node remains high. The number r is user-selected.

VIII. Experimental Results

In this Section, we present experimental results on the integrated performance verification methodology for three multimillion gate ASICs called Chip S, Chip SCR and Chip MCR. A description of the ASICs is provided in Table I. Chip S is a 130 nm ASIC and has 8 asynchronous clock domains ranging in frequency from 125-312 MHz. It has 24.7 M internal nodes and approximately 8.1 M gates. Chips SCR and MCR are 90 nm ASICs. Chip SCR has 5 asynchronous clock domains ranging in frequency from 125-312 MHz. It has 31.9 M internal nodes and approximately 11 M gates. Chip MCR has 3 asynchronous clock domains ranging in frequency from 125-281 MHz. It has 23.5 M internal nodes and approximately 8.1 M gates. From the large number of internal nodes in each ASIC, it is clear that methods proposed in the literature to test the longest path through each node would be economically infeasible for circuits of this size. The EncounterTest tool from Cadence [16] was used to perform test generation for all the experiments in this paper. The maximum frequency of the low-cost tester used is 250 MHz.

TABLE I

ASIC CHIPS S, SCR AND MCR.

Circuit data	Chip S	Chip SCR	Chip MCR
Primary inputs	46	97	48
Primary outputs	1	74	36
Bidirectional IOs	475	85	192
Internal nodes	24.7 M	31.9 M	23.5 M
Flip-flops	548 K	633 K	938 K
Gates (approx.)	8.1 M	11 M	8.1 M
Clock domains	8	5	3

TABLE II

QUALITY OF DETERMINISTIC PATHS.

Clock domain	Specified frequency (Mhz)	Observed maximum frequency (MHz)	
		Transition fault patterns	Deterministic path test patterns
1	125	250	340
2	250	464	712
3	156	312	400
4	312	878	800
5	156	310	370
6	312	784	830
7	166	211	300
8	125	250	250

A. Worst case deterministic paths

To motivate the use of variation-aware statistical timing to identify critical paths, we first investigate the quality of critical paths obtained from standard deterministic timing. Chip S was used for this experiment. We first generated a suite of 30 K transition fault test patterns for Chip S; these patterns have 78% transition fault coverage. No use of long paths or timing information was made for test generation. Next, for each clock domain in Chip S, 5 K worst case least-slack paths were obtained from deterministic timing. Path delay test generation was performed on these paths to obtain a second test suite. Both test suites were applied to Chip S on the tester. For both test suites, the frequency of the PLLs used was progressively increased until the first failing pattern was observed. In Table II, we compare the resulting maximum frequencies that the test patterns in each test suite could be run at for each clock domain.

For 6 of the 8 clock domains, the path delay patterns could be applied at a higher frequency than the transition fault patterns. This clearly demonstrates that the worst case paths obtained from deterministic timing are less critical than even the paths exercised by transition fault tests, which are generated with no consideration of long paths.

The patterns run faster than the specified frequency because of the inherent pessimism in timing closure. Timing closure takes into account several effects, such as negative bias temperature instability, hot carrier degradation, and worst case noise; hence ASICs are timed to operate faster than their specified frequency. Moreover, as noted in [6], we find that a large number of the least-slack paths reported by timing are Boolean untrue and cannot be sensitized functionally. However, the timing tool does not perform logic simulation and is unaware of false paths. Hence, the timing for these paths is met during timing closure. This also results in a large amount of timing pessimism, and as demonstrated in Table II, manufactured chips operate significantly faster than their specified frequency.

TABLE III

CRITICAL PATH IDENTIFICATION FOR CHIP SCR.

Clock domain	Frequency (Mhz)	Node criticalities	Number of paths	Average path length (gates)
1	250	0.0003–1.0	36	34
2	166	0.003–0.905	144	36
3	156	0.0001–1.0	8	47
4	312	0.0001–1.0	56	25
5	125	0.01–1.0	91	37

TABLE IV

CRITICAL PATH IDENTIFICATION FOR CHIP MCR.

Clock domain	Frequency (Mhz)	Node criticalities	Number of paths	Average path length (gates)
1	125	0.05567–1.0	48	60
2	281	0.0006–1.0	57	35
3	250	0.0005–0.5	80	30

B. Variation-aware critical paths

Next, we present results on the integrated methodology for ASICs SCR and MCR from Table I. An in-house statistical timing tool called Einstat enhanced with our methods for criticality computation and path tracing was used for all the experiments.

In Table III, we present results on criticality computation and critical path identification for the 5 asynchronous clock domains in Chip SCR. In Columns 1 and 2, we list the domains and their clock frequencies. For each domain, 400 nodes having the highest criticalities were chosen for critical path tracing. In Column 3, we present the range in criticalities for these 400 nodes for each domain. Note that for Domains 1, 3, 4 and 5, we were able to identify some nodes having a criticality of 1.0; these nodes lie on the critical path across the entire process space. Hence, it is important to test the paths passing through these nodes. In Columns 4 and 5, for each domain, we show the number of critical paths identified and the average length in gates of these critical paths. Note that Domain 4 having the highest clock frequency of 312 MHz, has the shortest average critical path length. The converse is not true, however, for the slowest clock domain, Domain 5. Path length measured in number of gates does not always correspond to delay. Hence, methods proposed to test the topologically-longest path through each gate may not always be testing critical paths.

In Table IV, we present results for the 3 clock domains in Chip MCR. As for the clock domains in Chip SCR, the 400 nodes having the highest criticality in each domain were chosen. We were again able to identify some nodes having a 1.0 criticality. The average critical path length for Domain 1 (125 MHz) is exactly twice the average critical path length for Domain 3 (250 MHz). From the 1.0 node criticalities in Column 3 of Tables III and IV, we see that our method is indeed identifying the most critical nodes in the designs.

Finally, in Figures 9 and 10, we compare the critical paths identified using the new methodology with critical paths obtained from single-corner deterministic timing. In Figure 9, we examine the critical paths for Domain 1 of Chip SCR. We compare 108 worst-case, least-slack paths from deterministic timing with 108 statistically-critical paths identified using the integrated methodology. The 108 paths in each list were sorted in order of increasing slack and pair-wise datapoints for the two paths in the same position in each list were plotted against their respective slacks. The Y-axis in Figure 9 shows the 3-σ slack of the paths identified from statistical timing. The X-axis shows the slack of the paths from deterministic timing.

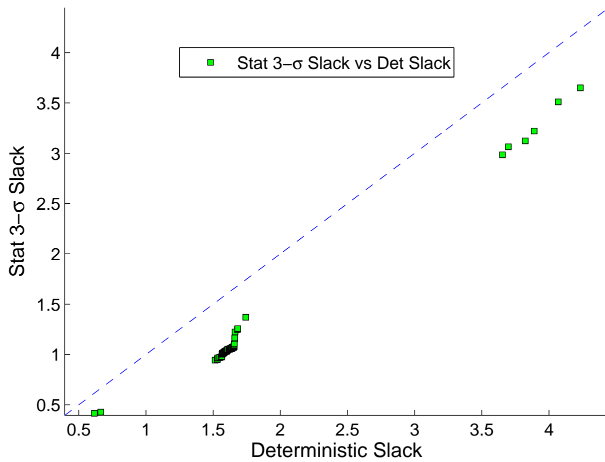

Fig. 9. Statistically-critical paths vs. deterministic paths for Domain 1 of Chip SCR.

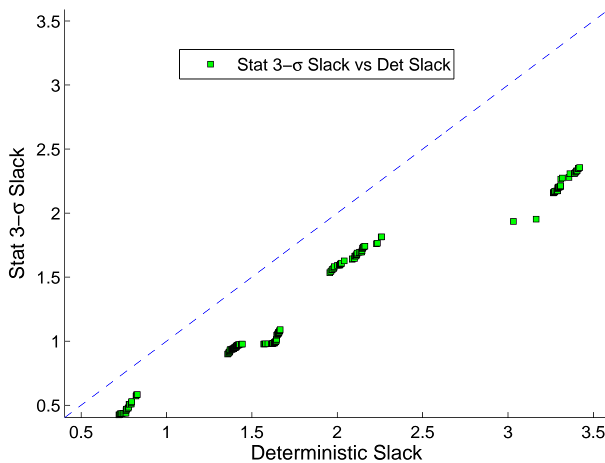

Fig. 10. Statistically-critical paths vs. deterministic paths for Domain 5 of Chip SCR.

Each datapoint in Figure 9 therefore represents two slack values of the same path, one obtained from deterministic timing, and the other identified by the integrated methodology. All the datapoints in Figure 9 lie below the 45-degree diagonal. Hence the 3-σ slack of the paths identified from statistical timing is less than the slack of the paths obtained from single-corner deterministic timing. Figure 9 therefore demonstrates that the paths identified using the integrated methodology are indeed more critical than those obtained from deterministic timing.

In Figure 10, we examine critical paths for Domain 5 of Chip SCR. We compare 180 worst-case, least-slack paths from deterministic timing with 180 statistically-critical paths identified using the integrated methodology. Again, the 3-σ slack of the paths identified from statistical timing is less than the slack of the paths obtained from single-corner deterministic timing. Hence, the paths for Domain 5 of Chip SCR identified using the integrated methodology are indeed more critical than those from deterministic timing.

IX. Conclusions

We have described a novel variation-aware methodology for performance verification of contract manufactured ASICs. This methodology is targeted at uncovering performance violations in defect-free ASICs arising from the aggregation of very small delay changes caused by process variation. We have described parameterized statistical timing that accounts for timing relationships in the circuit across the entire process and environmental parameter space. Moreover, we have shown how circuit nodes that are critical across the process space can be identified. We have presented a novel method to statistically trace the critical paths passing through these nodes. In our experiments, the critical paths identified by the integrated methodology are significantly more critical than those obtained from deterministic timing. Finally, the proposed methodology has been integrated with a low-cost at-speed structural test architecture that is used to verify the performance of the identified critical paths.

Acknowledgements

We thank Gary Grise, Pam Gillis, Jose Martinez and Frank Woytowich for help with the integrated performance verification methodology.

References

[1] P. S. Zuchowski, P. A. Habitz, J. D. Hayes, and J. H. Oppold, "Process and environmental variation impacts on ASIC timing," *Proc. Int. Conf. Computer Aided Design*, pp. 336–342, 2004.

[2] V. Iyengar, T. Yokota, K. Yamada, T. Anemikos, R. Bassett, M. Degregorio, R. Farmer, G. Grise, M. Johnson, D. Milton, M. Taylor, and F. Woytowich, "At-speed structural test for high-performance ASICs," *Proc. Int. Test Conf.*, pp. 2.4:1–10, 2006.

[3] N. Tendolkar, D. Belete, A. Razdan, H. Reyes, B. Schwarz, and M. Sullivan, "Test methodology for Freescale's high performance e600 core based on PowerPC instruction set architecture," *Proc. Int. Test Conf.*, pp. 1–9, 2005.

[4] L.-C. Wang, J.-J. Liou, and K.-T. Cheng, "Critical path selection for delay fault testing based upon a statistical timing model," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, pp. 1550–1565, Nov. 2004.

[5] K.-T. Cheng, S. Dey, M. Rodgers, and K. Roy, "Test challenges for deep sub-micron technologies," *Proc. Design Automation Conf.*, pp. 142–149, 2000.

[6] A. Crouch, J. Potter, and J. Doege, "AC scan path selection for physical debugging," *IEEE Design & Test of Computers*, vol. 20, pp. 34–40, Sept.-Oct. 2003.

[7] W. Qiu, J. Wang, D. Walker, D. Reddy, X. Lu, Z. Li, W. Shi, and H. Balachandran, "K longest paths per gate (KLPG) test generation for scan-based sequential circuits," *Proc. Int. Test Conf.*, pp. 223–231, 2004.

[8] W. Qiu, X. Lu, J. Wang, Z. Li, D. Walker, and W. Shi, "A statistical fault coverage metric for realistic path delay faults," *Proc. VLSI Test Symp.*, pp. 37–42, 2004.

[9] J.-J. Liou, A. Krstic, L.-C. Wang, and K.-T. Cheng, "False-path-aware statistical timing analysis and efficient path selection for delay testing and timing validation," *Proc. Design Automation Conf.*, pp. 566–569, 2002.

[10] B. Bailey, A. Metayer, B. Svrcek, N. Tendolkar, E. Wolf, E. Fiene, M. Alexander, R. Woltenberg, and R. Raina, "Test methodology for Motorola's high-performance e500 core based on PowerPC instruction set architecture," *Proc. Int. Test Conf.*, pp. 574–583, 2002.

[11] M. Sharma and J. Patel, "Finding a small set of longest testable paths that cover every gate," *Proc. Int. Test Conf.*, pp. 974–982, 2002.

[12] X. Lu, Z. Li, W. Qiu, D. Walker, and W. Shi, "Longest path selection for delay test under process variation," *Proc. Asia South Pacific Design Automation Conf.*, pp. 98–103, 2004.

[13] C. Visweswariah, K. Ravindran, K. Kalafala, S. Walker, S. Narayan, D. Beece, J. Piaget, N. Venkateswaran, and J. Hemmett, "First-order incremental block-based statistical timing analysis," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, pp. 2170–2180, Oct. 2006.

[14] J. Xiong, V. Zolotov, N. Venkateswaran, and C. Visweswariah, "Criticality computation in parameterized statistical timing," *Proc. Design Automation Conf.*, pp. 63–68, 2006.

[15] H. Chang, V. Zolotov, C. Visweswariah, and S. Narayan, "Parameterized block-based statistical timing analysis with non-Gaussian and nonlinear parameters," *Proc. Design Automation Conf.*, pp. 71–76, 2005.

[16] Cadence Design Systems, *http://www.cadence.com/products/digital_ic*.