A logic chip delay-test method based on system timing

by F. Motika
N. N. Tendolkar
C. C. Beh
W. R. Heller
C. E. Radke
P. J. Nigh

In this paper we present a novel approach to delay-testing of VLSI logic chips based on the level-sensitive scan design (LSSD) methodology. The objective of the delay test is to reduce significantly the failures of multi-chip modules at system integration test while minimizing the complexity and cost of subassembly testing. Because system timing data are used to derive test specifications, the delay defects that are most likely to cause a system path failure are detected a high percentage of the time. With the implementation of the delay test in the wafer production line, the system final-test failure rate of multi-chip modules used in IBM mainframe machines has dropped significantly.

Introduction

Large computer systems require high-speed circuits and at least moderately high levels of circuit integration on-chip. Both of these requirements have led to the use of tighter chip ground rules for wires, contacts, and devices. Processing defects in these features must be even more scrupulously guarded against than in the past. A key reason for this increased requirement is that there are in general more defects in high-performance circuits, such as non-dc-detectable partial opens and shorts, which are more likely to occur and cause delay defects.

With narrower interconnection lines, smaller line separations, and smaller contacts, a greater likelihood exists, for example, that a small defect will lead to an unacceptably high resistance between lines or contacts, and subsequently to an unacceptably slow circuit transition. The possibility that defects could cause permanent stuck-at-0 or stuck-at-1 dc circuit failures had earlier led to the extensive development of dc stuck-at-fault testing methodology and equipment. As is subsequently indicated, the increased sensitivity of faster systems to timing defects has forced system manufacturers to pay attention to failures which result in circuit transitions slower than specifications permit, in addition to those situations where circuits are classified as "dc good" or "dc bad." CMOS chips with relatively low-power output devices have joined those with emitter-coupled logic (ECL) at the heart of high-speed systems. As a result, a number of recently published papers have dealt with the problems of chip delay-testing [1–3].

Whatever the source of delay variation in logic paths from package inputs to package outputs, one can only hope to measure *path* faults. As with any logic testing

problem, in practice one must be concerned with the structure of the logic, and with the testing methodology. Early work on LSSD [4, 5] demonstrated that separating combinational logic with scan latches makes both dc and delay-testing more manageable.

Path faults now are those which arise if signal propagation time through the path is greater than the interval between two appropriately chosen clock times [6]. Early work on "delay faults" focused on device delays which could, for example, arise from the natural spread induced by variations in process, temperature, and voltage expected from production components and packages. No special distinction was made for those extreme device or circuit delays arising, for example, from almost open lines or contacts induced by point defects (see, however, [7]). Much of the earlier work [8–10] dealt with deterministic test-pattern generation specifically aimed at delay faults on particular device inputs or outputs. Restrictions were noted on the path types sensitizable for some test-generation procedures [11, 12]. A summary of delay-testing up to 1984 has been given in [13]. Delay-testing of high-speed gate array chips, using deterministic patterns in the context of scan design techniques, has been reported by several computer manufacturers [14, 15].

In dc testing, the use of random patterns was first discussed long ago [16, 17]. The concept of weighted random testing was discovered and applied [18]. In recent work, tester-generated pseudorandom test patterns have been used for dc testing on logic with LSSD constraints [19–21]. The extension of these ideas to delay-testing has come about as a natural succession, along with the development of built-in package pseudorandom test generation [22]. A new requirement is the definition of a "transition fault" [23, 24], which may arise from a slow-to-rise or a slow-to-fall circuit defect. Thus, a dc fault on a logic gate input or output is a limiting (very long time) case of a transition fault. A test at the package inputs for such a fault must create the appropriate transition at the point of the fault and also propagate the effect of the fault to a timed, measurable point. One can ask about the distribution of lengths of the test-pattern sequences required to detect faults, as well as the total number required to give some defined coverage. Waicukauski et al. [23] show that the number of random patterns required for a given coverage of transition faults will be roughly twice that of the corresponding dc coverage. A more precise estimate has been developed which also permits estimation of test length for paths where detection probability can be estimated from the topology [25].

The effects of weighting random patterns have been dealt with by Wunderlich [26, 27]. Given a circuit where detection probabilities for various faults can be computed or estimated, one can optimize the random search and either increase coverage for given pattern sequence lengths or reduce length for high coverage. In this work, the test-case logic circuits published by Brglez et al. [28] are used.

This paper describes an approach to delay-testing which is especially suited to guard against the effects of extreme delay faults in packaging components. The presence of random defects, e.g., almost open lines or contacts, can give rise to circuits with very slow-to-rise or very slow-to-fall switching transitions. The circuit delays due to such defects in general lie far outside the limits set by typical temperature, process, and voltage variations in devices, or by variations in net length and, hence, device loading. An earlier paper by one of the authors [29] pointed out that large digital systems are far more sensitive to large defects than they are to the often compensating and always smaller variations normally encountered. Simplification of delay-testing is achieved, as we describe it, by running synchronized successive dc patterns at speeds dictated by the system timing of particular logic path types on a chip or, alternatively, by a simple but appropriately accurate path strobing, one test cycle at a time.

Former methods required a tester that could independently manipulate each I/O pin, clock, and scan ring, thus facilitating, in principle, measurement of any path on the chip. In practice, such an approach is not efficient in situations where most serious system timing failures are due to large timing delays relative to a circuit transition time. The kind of testing discussed here sets up pulses at all input pins at the same time, turns on all latch clocks at an appropriate common time, and also strobes all comparable outputs or latches simultaneously. This method is called the *part number path type* (PNPT) delay test. Good coverage is obtained for timing faults with long delays.

Several sections of the paper deal with adjusting the speed with which an individual logic chip design is tested to values determined by the timing of its logic paths of the various types encountered under LSSD rules in system design. In turn, this method affords a scheme for calculating the likelihood of detecting timing defects of various magnitudes (durations) [29, 30]. Degradation of this coverage due to the limits of precision of tester measurements is also presented.

Finally, conclusions on the success of the method are stated and some possible extensions are discussed.

General description of the method

The method of logic chip delay-testing described in this paper is based on the LSSD methodology [4] and system timing concept widely used in IBM high-performance mainframe systems [31]. This design methodology, signal

propagation timing information, and the test patterns applied with specific clocking sequences are the key elements necessary for the implementation of this delay test method [32].

The structure of the logic of a typical LSSD chip is shown in **Figure 1**. For test purposes it can be visualized as blocks of independent combinational logic partitions separated by shift-register latches (SRLs). Each SRL usually consists of a pair of master/slave latches (L1 and L2) concatenated into one or more SRL strings, as shown in **Figure 2**. During test the primary inputs (PIs) and the primary outputs (POs) of a chip are directly controlled and observed by the tester; internal test points, also referred to as pseudo-PIs and -POs, are accessed via the SRLs by applying appropriate clock-signal sequences. The functions of individual clocks are described as follows:

- A clock – controlling the scan port of the L1 latch.
- B clock – controlling the scan/data port of the L2 latch.
- C1 clock – controlling the data port of the L1 latch.
- C2 clock – controlling the scan/data port of the L2 latch.

The L2 latch has only one port, which serves as the scan and data port. For the double-latch design, the B and C2 clocks are usually supplied by different chip PIs. However, on-chip these clocks are connected (forming an OR) before feeding the L2 latch. In Figure 2 this is denoted by B/C2.

The loading of a test vector into the SRL string is performed by serially applying the bit pattern to the shift-register input (SRI) and alternately toggling the A and B clocks until all of the SRLs have been loaded. Similarly, the interrogation of the contents of the SRLs is performed in the same serial shift fashion while observing the shift-register output (SRO). These two sequences are referred to respectively as "Load SRLs" and "Unload SRLs" operations. In some cases the two sequences can be combined into a single load/unload shift operation to improve test efficiency. Also, for logic structures with multiple SRL strings, this operation is usually performed simultaneously for all strings.

In addition to accepting data from the previous L2 latch in the SRL string, the L1 latch also captures system data or data from the combinational logic when the C1 clock is pulsed. Conversely, the L2 latch receives data from the L1 latch of the same SRL when the C2 or B clock is pulsed and launches data to the subsequent combinational logic partition. During normal system operation, output signals from the combinational logic are first clocked into the L1 latch and then shifted into the L2 latch. The L2 latch in turn feeds the next stage of combinational logic.

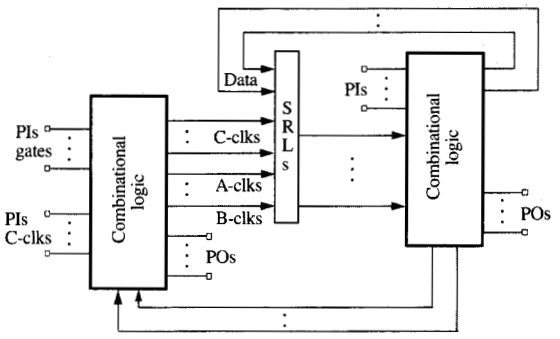

Figure 1

LSSD structure and clocking network.

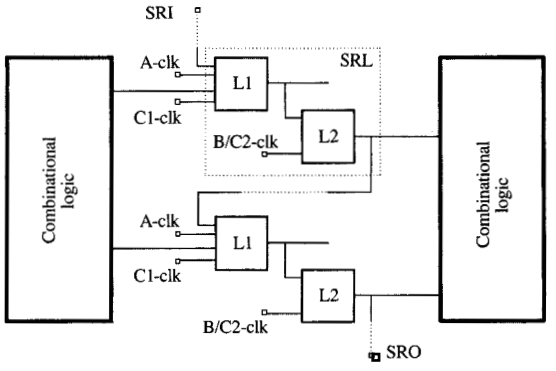

Figure 2

LSSD double-latch design.

For a typical LSSD double-latch design, the basic steps involved in delay-testing are derived from the following dc test sequence:

1. Load SRLs.
2. Stimulate PIs and compare POs.
3. Stimulate PIs and apply C1 clock.
4. Apply C2 clock and compare POs.
5. Unload SRLs.

This dc sequence, with minor variations in conjunction with timing specifications, forms the foundation for delay

Table 1 Maximum allowable delay in ns for ten test-case System/3090 logic chip designs for the four path types.

Design	*L2L1*	*L2PO*	*PIL1*	*PIPO*
1	2	6	5	6
2	6	5	5	3
3	11	15	7	9
4	12	9	9	7
5	16	15	17	16
6	—	—	—	8
7	10	10	7	6
8	13	10	5	8
9	—	—	—	15
10	—	5	6	4

testing. Such sequences may be repeated multiple times with different test vectors. At each step, a specific bit pattern is scanned into the SRLs and applied to the PIs of the chip. Delay-testing requires a pair of successive bit patterns involving a change in at least one of the bits. The test may consist of a sequence of changing PI and pseudo-PI values at a given time and capturing the PO or pseudo-PO values after a specified time interval. If any observed value does not match the expected value, the test is terminated, because the chip contains a defect. Otherwise, the test continues with the application of another test pattern, until all test patterns are applied and no defect is found.

The simplified description given above relates to a typical double-latch master/slave logic structure and system clocking design; in practice, however, there are many variations in the implementation of LSSD logic. Although these design variations may imply changes to test sequences and timing parameters, the basic delay-test concept still applies.

- *Delay-test criteria*

Consider the logical chip structure of Figure 1. There are four distinct *path types* of interest within this structure: PI to PO, PI to L1, L2 to PO, and L2 to L1. To delay-test a given chip design, one first calculates the delays associated with each path, and then determines the *longest* or *maximum* path for each path type. The following four delay values are then used for the delay test:

D_1 = MAX {of all PI-to-PO delay paths},
D_2 = MAX {of all PI-to-L1 delay paths},
D_3 = MAX {of all L2-to-PO delay paths},
D_4 = MAX {of all L2-to-L1 delay paths}.

The delay value of a path is calculated by adding the circuit-switching speeds of gates and the wire delay values along the path. A timing analysis program [31] normally used during system logic design is used to calculate the delays. These delay values are referred to as the *delay-test specifications* of the individual design. The additional path type between L2 and L1 forming the SRL scan string should be delay-tested using load/unload technology timing requirements. An example of these specifications for ten test-case System/3090 logic chip designs is shown in **Table 1**.

For delay-testing of a given design, the delay specifications determine the times at which PIs are switched, latch clocks are pulsed, and POs are compared to expected values.

System sensitivity considerations

- *Delay defect detection coverage*

Suppose a defect-free path has a delay value of p ns (i.e., the signal takes p ns to get from the beginning to the end of the path). If a manufacturing defect causing an additional circuit delay in this path occurs, we say that the delay defect is of size d ns if the path delay is $p + d$ ns. The delay defect *detection coverage* $P(d)$ is the fraction of the delay defects of a given size d that are detected by the delay test. The coverage depends on the following factors:

- Path length (delay value) distribution for the chip design.
- Test criterion and method used in a given test.
- Test-pattern coverage.
- Tester tolerance.
- Product deviation.

For a defect d of given size, we first calculate the probability that a chip path with the defect of size d will fail the test criterion $P(d)$. This calculation uses the chip path length distribution and the test criterion; the formulas used are very similar to the ones used for the system sensitivity calculation [29]. The set of ten typical chip designs shown in Table 1 was chosen to study the delay defect detection coverage for the PNPT delay test. By using a timing analysis program, the path delay value distribution for each chip design is obtained. The tester setting for each chip path is determined by the path group (e.g., PI-to-PO) to which it belongs, as discussed earlier.

The coverages for PNPT delay test for a perfect tester (0-ns tolerance) and for a tester with 3-ns overall timing tolerance are shown in **Figure 3**. The tester tolerance of 3 ns implies that the tester clock or strobe can occur up to t ns later than the desired optimum, where t is uniformly distributed between 0 and 3 ns. We assume that the test-pattern coverage for transition faults is 100%. The test-pattern coverage is 100% if, for every possible location of a defect, the set of patterns contains at least one pattern which, when applied, creates a transition propagated from some PI or pseudo-PI to a PO or pseudo-PO along

the path containing the defect. An approximation of the fault-detection coverage $P(d)$ when the test-pattern coverage is $x\%$ can be obtained by multiplying the calculated coverage by $x/100$.

The curves in Figure 3 show that the coverage increases monotonically with increasing defect size. Comparing the coverages for 0-ns and 3-ns-accuracy testers, we find that the benefit of going to a tester with higher accuracy decreases rapidly as the defect size increases beyond 6 ns. As expected, the effectiveness for a 0-ns-tolerance tester is higher than that for a 3-ns tester.

We later use the detection coverage $P(d)$ to determine the system test module fallout due to delay defects.

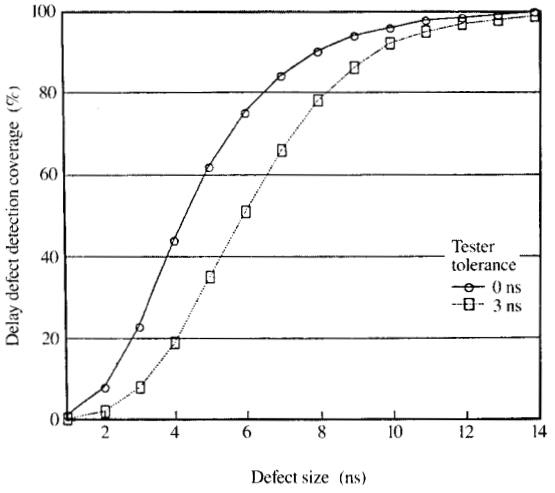

Figure 3

Part number path type chip delay test.

• *System test module fallout reduction*

Chips containing delay defects can cause multi-chip modules to fail system integration test. The operation of testing modules in the system at cycle time is called system test. Effective chip delay test can significantly reduce module fallout at system test. The ratio of the total number of modules that fail system test to the total number of modules tested is known as the *system test module fallout.* The higher the effectiveness of the chip-level test, the lower the system test module fallout. In this section we consider the effect of chip delay test on system test module fallout, while restricting ourselves to system test failures that are due to delay defects; when test coverage is used, reference is made to transitional-fault coverage.

If $q(d)$ is the percentage of modules which would fail system test due to defects of size d assuming no chip delay test, and $q'(d)$ is the percentage of modules which would fail system test due to defects of size d assuming that chips which have been delay-tested are used, then

$$q'(d) = q(d)[1 - P(d)].$$

Here $P(d)$ is the delay defect detection coverage as defined earlier, with a value between 0 and 1. The percentage reduction in the module fallout attributed to delay test is

$$100 \frac{\sum_d [q(d) - q'(d)]}{\sum_d q(d)},$$

showing that the effect of chip delay testing on system test module fallout depends upon the frequency of module failures due to a given size defect $[q(d)]$ and delay defect detection coverage $P(d)$. The $q(d)$ can be determined by observing system test module fallout due to delay defects when non-delay-tested chips are used. Another method of determining $q(d)$ is to use delay defect frequency distribution, system path length

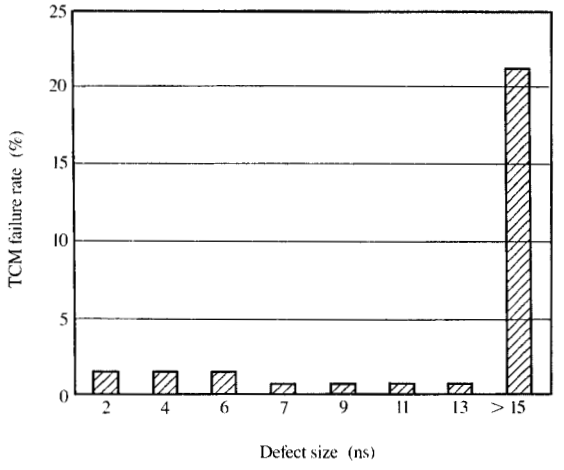

Figure 4

TCM fallout at system test due to delay defects of a given size.

distribution, and the analytical method given in [30]. In **Figure 4** we show $q(d)$ derived from system test data for an early production of modules. The total module fallout rate, without chip delay test, is 30%.

We analyze the effect of PNPT delay test on the system test module fallout for these modules. The delay defect

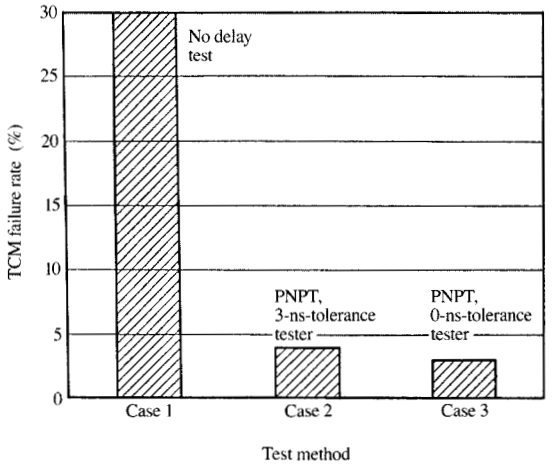

Figure 5

TCM fallout for no delay test and part number path type test.

detection coverage $P(d)$ for these modules is shown in Figure 3. Using $P(d)$ and $q(d)$, we obtain $q'(d)$ for PNPT delay test and compute the percentage reduction in module fallout attributed to PNPT delay test using the formulas given above. The calculated system test module fallout values for three different cases are shown in **Figure 5.** In Case 1, chips that undergo no chip-level delay test are mounted on modules. In Case 2, chips that pass the PNPT on a 3-ns-tolerance tester are mounted on modules. In Case 3, chips that pass the PNPT on a 0-ns-tolerance tester are mounted on modules. For each case, we show the percentage of the total modules tested that would fail the system test because of a delay defect.

In this example, assuming 100% transitional-fault test-pattern coverage, the percentage reduction in module fallout attributed to PNPT delay test with a perfect tester is 90%; with a 3-ns-tolerance tester, it is 87%. In practice, the test-pattern coverage is less than 100% but close to it, so the reduction in module fallout would be slightly less than the above numbers but close to them.

We conclude that with the use of PNPT delay-testing for logic chips, a significant reduction can be achieved in the system test module fallout.

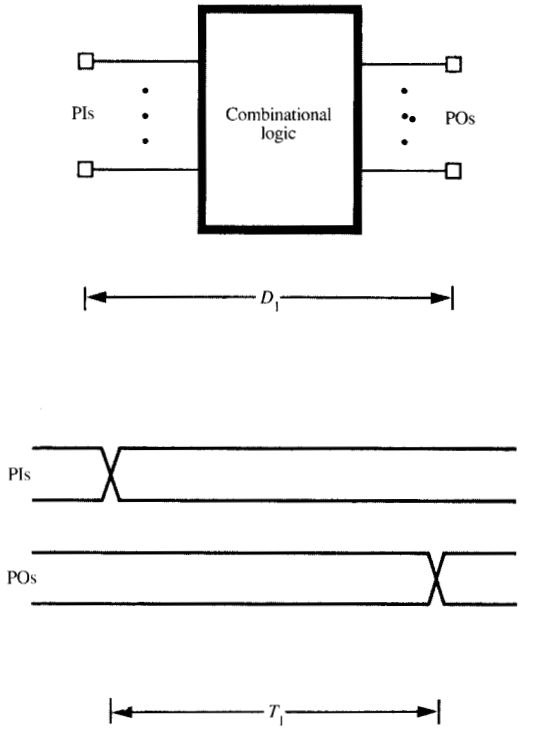

Figure 6

PI-to-PO path test.

Path types and test sequences

In this section we describe the type of test sequences required to test for delay defects in each path type. Although our examples treat each sequence and path type individually, the actual test sequences usually combine multiple path delay tests within the same sequence. Furthermore, some path types may require multiple test sequences or more than one application of the same sequence with different timing conditions in order to optimize the delay test effectiveness.

The path types we consider are those shown in Figure 2, representing a typical double-latch-design logic structure without embedded RAMs. In this type of design, the leading edge of the clock pulse launches the data from the input of the latch, while the trailing edge captures these data in the latch. The on-chip path delay values are calculated via a timing analysis tool which is discussed later. This structure can be subdivided into the following five path types: PI-to-PO, L1-to-L2, L2-to-PO, L2-to-L1, and PI-to-L1.

• *Testing PI-to-PO paths*

As illustrated in **Figure 6**, D_1 represents the longest PI-to-PO path timing for a particular chip design. T_1 represents the strobe timing of the POs with respect to the PIs and is equivalent to $T_1 = D_1 + T_{os}$, where T_{os} (tester off-set time) is the additional delay time due to tester loading, tester uncertainty, etc. The basic sequence for applying the test patterns is the following:

1. Load SRLs.
2. Stimulate PIs at T_0.
3. Strobe POs at T_1.
4. Compare POs to expected values.

We note here that all PIs are assumed to be stimulated simultaneously, and all POs are to be observed T_1 time later. This, of course, is not the most effective path delay test when one considers chip boundary conditions resulting in partial system paths. A simple enhancement to the test would be to use maximum path delays associated with each individual PI and/or PO and then to program the tester with per-pin timing. This enhancement could be applied to all PI or PO path tests.

• *Testing L2-to-PO paths*

The L2-to-PO test is similar to the PI-to-PO test above, with the exception that the inputs to the combinational logic originate at the shift-register L2 latch rather than from the PIs. In **Figure 7**, D_2 represents the longest L2-to-PO path timing on the same chip design. T_2 represents the strobe timing of the POs with respect to the L2 output transitions and is equivalent to $T_2 = D_2 + T_{os}$. The basic sequence for applying the test patterns is the following:

1. Load SRLs (inhibit B clock on last shift).
2. Pulse C2/B clock at T_0 (move data from L1 to L2).
3. Strobe POs at T_2.
4. Compare POs to expected values.

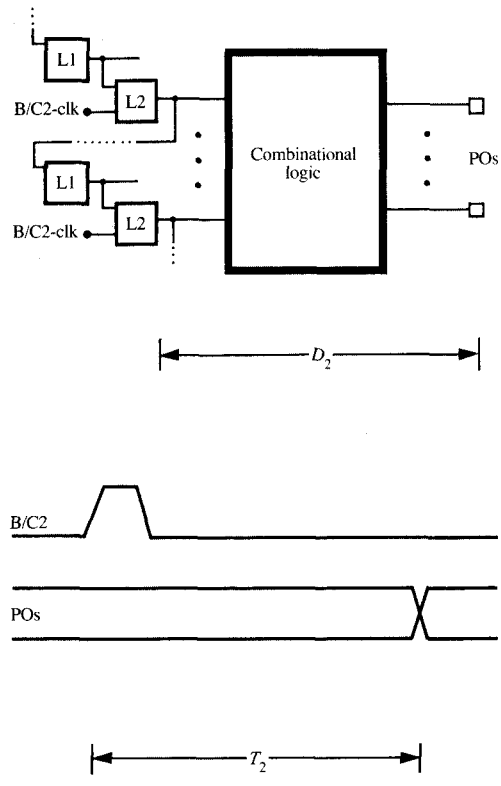

Figure 7
L2-to-PO path test.

• *Testing PI-to-L1 paths*

Figure 8 illustrates how the delay test is applied to a PI-to-L1 path. At the beginning of a tester cycle, all PIs are stimulated simultaneously. After a specific time, say T_x, the C1 clock is pulsed and returned to the off value at T_3. If there is a defect which makes any path delay longer than T_3, an error is latched in one of the L1s. D_3 in this diagram represents the longest PI-to-L1 path on a chip, and $T_3 = D_3 + T_{os}$. The basic sequence for applying the test patterns is the following:

1. Stimulate PIs at T_0.
2. Pulse C1 clock at T_x (where $T_x = T_3 - CPW$).
3. Unload SRLs.

• *Testing L1-to-L2 paths*

The L1-to-L2 scan-path delay test is normally incorporated into the load/unload SRL operation, as shown in **Figure 9**. Appropriate scan-clock pulse widths T_{sc} and pulse separation are used to ensure shifting functionality at specified rates and satisfy L1-to-L2 timing T_{sc} requirements.

• *Testing L2-to-L1 paths*

The L2-to-L1 path discussed here is the system path between SRLs, as shown in **Figure 10**, and not the scan path. When setting up tests for this path type, one must consider two additional conditions. The first potentially conflicting condition is due to the product minimum clock pulse width and pulse separation requirements, tester accuracy limitations, and the longest L2-to-L1 path delay T_c. Depending on the actual parameters, one might not be able to set up for an optimum T_4 delay test, but rather introduce an artificial slack and thereby mask some delay faults smaller than the above-mentioned slack. T_4 represents the time between the leading edge of the C2 clock and the trailing edge of the C1 clock. The second condition results as a consequence of the LSSD master/slave design for the L1 and L2 latches. One must ensure that the last L2 clock associated with the loading of the SRL is performed correctly with respect to the timing of the next L1 clock.

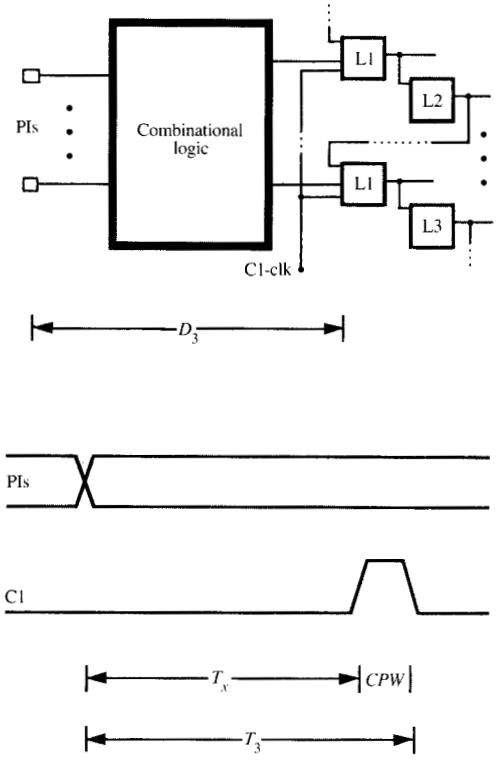

Figure 8

PI-to-L1 path test.

While an ideal clock sequence for this test might consist of a C2 pulse followed by a C1 pulse, the actual sequence must include the loading of the SRLs. This is accomplished by inhibiting the last B clock during the SRL load, then applying the C2 and C1 clocks in the same tester cycle. In this way the data originate at the L2 latches by clocking the C2, propagate through the combinational logic, and are then captured in the L1 latches when the C1 clock is pulsed. The sequence also makes use of the ORed relationship between the C2 and B clocks in controlling the L2 latches. The basic sequence for applying the test patterns is the following:

1. Load SRLs (inhibit B clock on last shift).
2. Pulse C2/B clock at T_0 (move data from L1 to L2).
3. Pulse C1 clock at $T_4 - CPW$.
4. Unload SRLs.

• *Testing embedded RAMs*

In logic structures with embedded RAMs, the output from combinational logic circuits may feed the RAM, and likewise the RAM outputs may feed other combinational logic circuits. In a general case, the inputs and outputs of a RAM may not be directly controllable or observable at the primary input and output pins. Therefore, in addition to the five path types considered in the earlier sections, paths from logic to the RAM and from the RAM to logic must be considered. Writing of data into and reading of data out of the RAM are controlled by the RAM write and read control clocks (the read clock is optional). Typically, embedded RAMs are used in the write, read, or read-after-write modes, and the delay tests are treated accordingly.

Write path test

When the RAM is used in the write mode, it is used to store data only. This is very similar to the use of L1 latches in the LSSD double-latch design. Hence, the RAM itself can be considered as the receiving

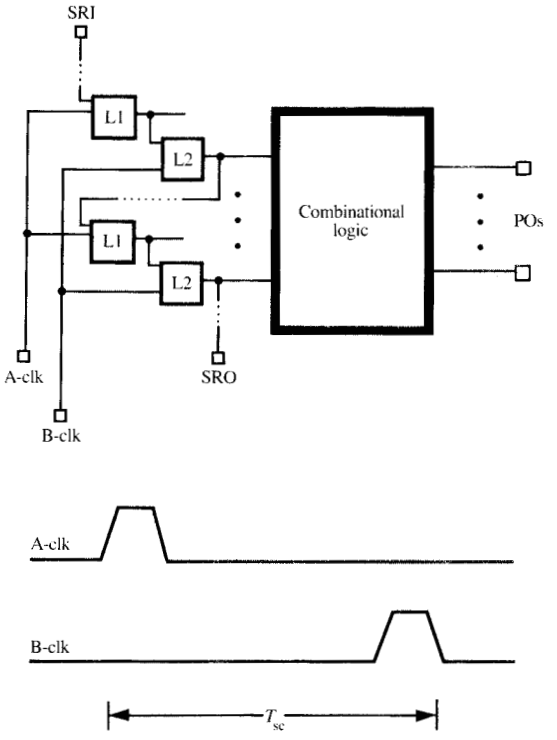

Figure 9

L1-to-L2 path test.

observation point or pseudo-PO controlled by the write clock (WC), as shown in **Figure 11**. Since there are two possible originating points (PIs and L2s), there are two basic path types, PI-to-embedded-RAM and L2-to-embedded-RAM. Testing of delay defects within the write path can be accomplished in the same manner as for PI-to-L1 and L2-to-L1 latches. However, in this case the write clock is used instead of the system C1 clock. Care must be taken to ensure that data are written into the correct address. This can be achieved by applying patterns for writing a particular address within a specified timing. In other words, one should make sure that the switching of PIs and the pulsing of C1 and WC occur in the same tester cycle, while maintaining the write path delay (T_w) timing relationship.

Read path test

When the RAM is used in the read mode, its behavior is similar to that of a read-only storage (ROS) function, which can be treated as part of the combinational network between a set of PIs or L2s and some POs and L1s. Data flow from the source through the RAM to the sink. Therefore, testing a RAM in the read mode is similar to testing logic-only LSSD parts. The basic path delay test described earlier should be able to detect delay defects in the read path. **Figure 12** illustrates the concept of delay test applied to the read path of an embedded RAM, where T_r represents the read path delay.

Read-after-write test

When the RAM is used in the read-after-write mode, the contents of some RAM cells are changed and the data are read out of the RAM. To test this mode of operation, signals originating at the PIs or L2s and the write clock are applied at the appropriate time, and then the data are clocked into L1 and/or measured at POs at the specified time. The concept of gross delay testing for the read-after-write mode of an embedded RAM is shown in Figure 12, where T_{raw} denotes the read-after-write path delay and T_w represents the write path delay.

Test patterns

A classical delay test methodology [9] is built upon a set of specifically defined faults, i.e., slow-to-rise and slow-to-fall transition-delay faults [9, 22]. These faults behave somewhat like temporary stuck-at-0 and stuck-at-1 faults [23]. As in stuck-fault test generation, path-sensitization techniques are used to derive test vectors. However, in contrast to stuck-fault test generation, a set of two sequential patterns is created to detect faults along the path. The first pattern is used to sensitize a path and to set up target nets to desired logic states (1 or 0). The second one switches these nets to their complementary values, such that transition faults can be observed at POs

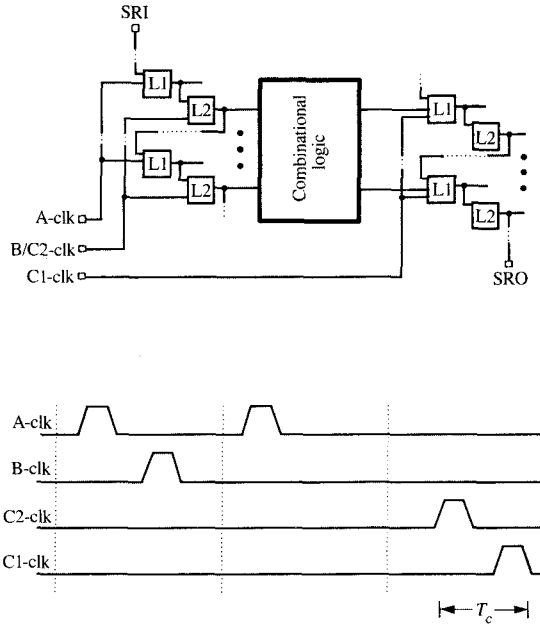

Figure 10

L2-to-L1 path test.

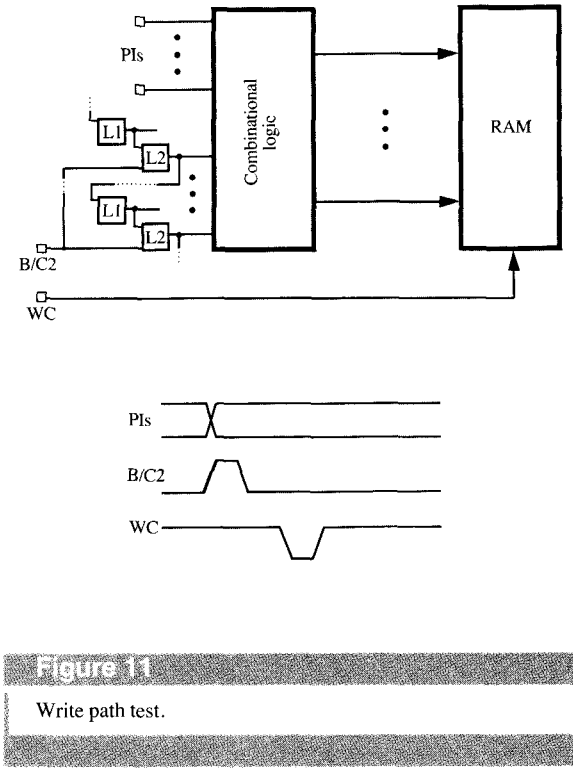

Figure 11

Write path test.

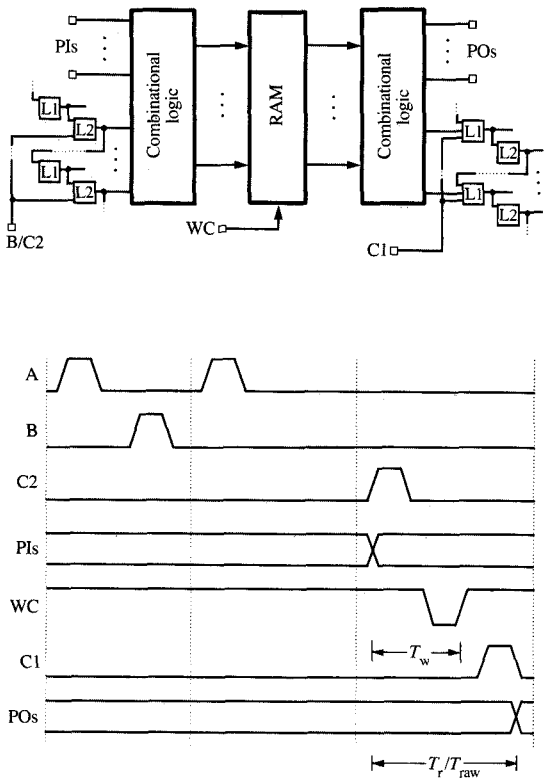

Figure 12

Read-after-write path test.

or internal test-point SRLs. In [23], they are called, respectively, initialization and transition propagation patterns. Usually these test-pattern sets are derived or verified from an appropriate delay model. It is conceivable that each may assume a different timing. While this approach is very thorough and exact, it may not necessarily be the most cost-effective one. Support for this test methodology requires not only an advanced tester but also a sophisticated software system. In addition, the test time may be unacceptably long.

As stated in the Introduction, this paper advocates a more realistic approach. Timing data and test patterns are independently created. They are merged by a postprocessor and then applied to the target tester. With this approach, one set of fixed timing for each I/O is used throughout the entire test. Therefore, "on-the-fly" timing is not required. Furthermore, this approach is not limited to a specific test-pattern generator. Patterns derived from various sources, such as deterministic algorithms, hardware random-pattern generators, or functional patterns, are compatible with it. At present, at least two types of test patterns have been successfully applied to this test methodology. The first one uses stuck-fault test patterns generated from a deterministic PODEM [33] -like pattern generator. The second type uses hardware-generated patterns based on a weighted random-pattern algorithm.

- *Stuck-fault test pattern*

Since IBM large systems deal mostly with LSSD parts, the previously mentioned stuck-fault test patterns are typical LSSD test patterns. When these patterns are applied with relaxed timing (on a single-cycle basis), the procedure is called a dc test. However, when these same patterns are applied with tightly controlled timing, the procedure becomes a delay test. A typical dc stuck-at-fault LSSD test sequence is listed below:

1. Load SRLs.
2. Stimulate PIs.
3. Measure POs.
4. Pulse C1 clocks.
5. Pulse C2 clocks.
6. Measure POs.
7. Unload SRLs.

As indicated previously, the slow-to-rise transition fault behaves like a temporary stuck-at-0 fault; the slow-to-fall transition fault behaves like a temporary stuck-at-1 fault. Therefore, when the same test sequence is executed under specific timing constraints, it becomes an effective way to detect slow transitions. A modified tester sequence for achieving delay test is listed in the following sequence:

1. Load SRLs (holding the last B clock).
2. Stimulate PIs simultaneously.
   - Pulse B clocks.
   - Pulse C1 clocks
     (timed relative to the PIs for the PI–L1 path test);
     (timed relative to the B for the L2–L1 path test).
   - Measure POs
     (timed relative to the PIs for the PI–PO path test);
     (timed relative to the B for the L2–PO path test).
3. Pulse C2 clocks.
   - Measure POs
     (timed relative to the C2 for the L2–PO path test).
4. Unload SRLs.

Note that delay test of the four possible paths is achieved by combining the first two patterns of the original tester loop. The modification of these patterns, i.e., holding the last B clock and applying it timed with C1 and measure POs, is called a transition shift in [23]. Studies have shown that transition-fault test coverage of these

modified test patterns is about 85%. At first, a special simulator developed for CMOS open faults [34] was used for this study. CMOS open faults behave somewhat like transition faults. The detection of these open faults also requires a set of two patterns to charge and discharge specific nets.

The experimentation procedure included the creation of the transition-fault model, generation of PODEM-like test patterns based on the stuck-fault model, resimulation of these patterns on the transition-fault model, and calculation of test coverage. The results of the eight test designs are listed in **Table 2**.

Later, another experiment was performed using the weighted random-pattern test system transition-fault simulator developed by IBM East Fishkill. Ten different test designs were used; their average transition-fault test coverage of the PODEM-like patterns was found to be 87.22%. In both cases, stuck-fault test coverage of these parts is above 99.5%.

Table 2 Transition-fault coverage for the test of eight designs.

Design	*#TSF*	*#UTF*	*#PAT*	*%TC*
1	1343	291	65	80.5
2	2910	421	402	86.2
3	4244	502	169	88.2
4	2958	328	84	87.8
5	1892	302	88	83.1
6	3934	587	208	85.0
7	2510	508	800	80.0
8	3737	376	281	86.3

#TSF = number of transition faults.
#UTF = number of untested transition faults.
#PAT = number of patterns.
%TC = transition-fault test coverage (average = 84.86%).

- *Weighted random patterns*

The same test concept has been implemented with the weighted random-pattern (WRP) test system [19–21]. WRP delay tests consist of multiple test sequences similar to the one described above. These sequences are structured to allow delay tests of all possible paths for the specific design being tested. The WRP test sequences can be used multiple times with different timing setups to optimize testing of individual or subgroups of path types.

Experimentation has shown that the transition-fault test coverage of WRP on the same ten test parts mentioned above is about 98.91%. This increased coverage results from the additional patterns (20 times) typically generated by the WRP system. The above results are achieved on patterns generated for stuck-fault testing only. The WRP test system is also capable of generating patterns optimized specifically for transition-fault testing, potentially resulting in even higher test coverage.

- *RAM test patterns*

The dc stuck-fault test patterns generated for LSSD are capable of testing the surrounding logic and a fraction of the RAM addresses. Functional patterns are required to fully test the actual embedded RAM. For example, in one set of functional patterns the RAM is first initialized by writing a specific data pattern (column bar). The initialization patterns exercise the write mode for all addresses. This is followed by reading out the content of the first address (read path test of this address). Next, a complementary data pattern is written into this address and then read out (read-after-write test). This test sequence repeats for all the addresses of the RAM. Given appropriate addressing sequences (such as a Grey code) and data patterns (such as checkerboard), functional patterns suitable for effective delay tests can be performed.

Test-system considerations

New test-system requirements exist for the above delay-testing scheme which did not exist when only dc stuck-at-fault testing was performed. However, because these requirements have been kept to a minimum, the test-system cost and complexity are not excessive. In this section the test-system requirements are briefly discussed.

As previously mentioned, adequate test-system timing accuracy is necessary to implement the described approach. Delay defect detectability can be directly related to this accuracy, as we described earlier. The edge timings of the tester are programmed according to the following formula:

$$\textit{Programmed delay} = \textit{nominal path delay} + \textit{path delay tolerance} + \textit{tester tolerance}.$$

The tester tolerance is assumed to be the worst-case edge placement value. The path delay tolerance and the tester tolerance should include an acceptable amount of short-term product and tester drift, respectively. Any tester calibration needed to achieve this tolerance should have minimum impact on throughput.

The causes of inaccuracy are well known [35, 36], but the tester tolerance also includes tester interface effects [2, 37]. The following factors, which were previously ignored, must now be considered:

- Mutual inductance between pins in the probe.
- Distance from the pin electronic cards to the chip under test.
- Impedance matching of the tester and the interface transmission lines.
- Transmission-line loading effects.
- Reliable, low-resistance probe-to-chip connections.

Assuming that the accuracy of the tester is adequate, repeatability becomes the dominant parameter. Edge-placement repeatability is critical, since any drift over time may cause fault-free products to fail or faulty products to pass. In order to diagnose device or tester problems, it is important to be able to repeat the results of devices tested earlier. Although a tester may have an accuracy specification of 1.5 ns, for example, the actual drift over time may be much lower (e.g., 200 ps). For such a test system the accuracy is bounded only by its calibration technique.

Although there is increased emphasis on tester accuracy and the test environment, the tester requirements for this delay-testing scheme are not as rigorous as other high-speed testing methods. Since this method relies only upon path delay testing, there is no tester requirement for high-speed vector application rates. The pin requirement is only that groups of pins (e.g., all inputs) must be switched simultaneously; thus, there is no per-pin requirement. Also, no cycle-to-cycle timing changes are required. These test-system functions, which are among the most expensive, are not needed to implement the presented delay-testing scheme.

Complexity of delay test

We wish to make the following observations regarding the complexity of delay test. Comparing PNPT delay test to the test that requires a very accurate tester and measurement of delay of each path, we find that PNPT is much simpler in terms of tester programming and can be used on a tester that has accuracy requirements much lower than those for the latter. PNPT uses one timing criterion per path type, whereas accurate measurement of every path would require the tester to use timing specification of each path on a chip. Thus PNPT avoids the problem of generating and manipulating thousands of timing specifications per chip design, and the tester programming and hardware complexity that would be required to test each path on a chip to its unique timing specification. If the system test module fallout is mostly due to relatively large defects (e.g., 25–30% of a system cycle), PNPT delay test is a very cost-effective way of reducing system test module fallout.

Delay testing can also act as a gross timing model verification tool and as a product monitor. Early in the product life, measurements can be taken on actual circuit designs using the described method to confirm the accuracy of the timing analysis data from which the product was designed. During the lifetime of the product, changes in the amount of delay-testing fallout can help diagnose an increase in delay defects. Discovering these problems during chip test can reduce the time required to diagnose potential product quality problems.

Methodology enhancements

Several delay-test methodology enhancements exist that can further improve the coverage and effectiveness of delay tests. Although improved tester timing accuracy benefits all forms of delay test, test data generation, tester architecture, and test throughput should be considered in the implementation of the delay-test methodology. Some possible enhancements are the following:

Enhanced delay-test pattern generation As shown previously, delay-test coverage of dc test patterns is around 85%. This is based on the fact that the same test patterns give a dc test coverage of 99% or higher. However, with additional specifically generated delay-test patterns, the test coverage can be increased above 97% [34].

Test by individual I/O pin timing Within a particular path type, there are long paths and short paths of various timings. A single strobe for all paths within a group certainly will leave shorter paths exposed. This is even more critical for a small delay defect in a shorter path. Logically, one could test the product according to individual I/O timing, thereby further improving PI-to-latch and latch-to-PO test effectiveness.

Test by per-pattern and per-pin timing This is the classical test methodology, which optimizes timing for each individual test pattern and I/O. However, it requires a far more sophisticated tester and software support system than the present approach, which is done mostly with the existing test data generation system. Timing data and dc test patterns from two different sources are merged and used for manufacturing.

A simpler approach is to generate patterns oriented toward sensitizing the longest path. In current deterministic pattern generators, the shortest paths are chosen to be sensitized. Given a test directed toward a fault on a long path and using the described method, the delay-test effectiveness can be further improved.

Concluding remarks

We have considered the problem of delay-testing logic chips for reducing the failures of multi-chip modules due to delay defects. The method we propose (PNPT delay test) makes a trade-off between the complexity of testing and the effectiveness of the test. Rather than attempting to detect every delay defect regardless of the size of the defect and the system path in which it occurs, the proposed test method is devised to detect those defects that have a high probability of causing a system path to fail. The complexity of testing is reduced by grouping all chip paths of the same type into path-type groups and specifying the maximum allowable delay for each path-

type group. Timing analysis programs are used to calculate the maximum allowable delay for each path group on each chip design. The delay test determines whether each path on a chip meets the defined criterion for the group to which it belongs.

We have also presented a model for computing the coverage and projecting the system test module fallout reduction. Our evaluation shows that using PNPT delay testing, a significant reduction can be achieved in the number of modules that fail system test due to delay defects. The success of PNPT delay test is due mainly to two complementary elements:

- Very effective delay fault coverage for "large" delay defects.
- System module fallout largely due to these "large" delay defects.

Selecting a chip delay-testing method for a given product involves making a trade-off between the complexity (cost) and the effectiveness (benefit) of testing. Compared to testing every path on every chip, each to its own delay criterion, the PNPT delay-test method is easy to implement and yet effective in detecting the delay defects to which the system is most sensitive. For certain applications, where a higher-performance tester is not available and cannot be justified, the PNPT delay-test method is a cost-effective alternative. The described delay test does not, however, as stated earlier, test for product delay variations due to process shifts, but is intended to test for delay-type defects in the chip semiconductor fabrication process.

Advances in technology and packaging resulting in fast circuit switching speeds and relatively small system cycle times will make subnanosecond defects a significant contributor to future multi-chip module fallout. For this reason, the delay test that may be adequate for today's product may not be so for those of the future. Delay-testing for future products continues to be a challenge, but several cost-effective approaches look encouraging.

Acknowledgments

The work described in this paper is the result of development and implementation done by members of the Timing Test Data Generation and System Test groups in DSD Poughkeepsie and Kingston, and by the members of the Test Analysis & Engineering and Test Methodology groups in GTD East Fishkill. The authors are grateful to C. H. Carnell for his key contribution to the implementation of this methodology in manufacturing test, to J. J. Ellison for his help in establishing the timing data transfer link between the system designer and chip manufacturing final test, to A. D. Shore for the physical diagnostics pinpointing the ac defects, and to J. A. Waicukauski for the extension of the concept to the weighted random-pattern methodology. The authors would also like to thank E. B. Eichelberger, E. Lindbloom, D. H. Byers, P. J. Salvatori, D. Wu, A. Elias, G. L. Steinbrueck, L. Parker, G. Bell, D. Bossen, M. Y. Hsiao, E. Sharp, M. Reiter, and J. Zeigler for their technical contributions, discussions, and managerial support.

References

1. K. Kishida, F. Shirotori, Y. Ikemoto, S. Yishiyama, and T. Hayashi, "A Delay Test System for High Speed Logic LSIs," *Proceedings of the 23rd Design Automation Conference*, Las Vegas, June 1986, pp. 786–790.
2. M. R. Barber, "Subnanosecond Timing Measurements on MOS Devices Using Modern VLSI Test Systems," *Proceedings of the 1983 IEEE International Test Conference*, 1983, pp. 170–180.
3. Eun Sei Park, M. R. Mercer, and T. W. Williams, "Statistical Delay Fault Coverage and Defect Level for Delay Faults," *Proceedings of the 1988 IEEE International Test Conference*, pp. 492–499.
4. E. B. Eichelberger and T. W. Williams, "A Logic Design Structure for LSI Testability," *Proceedings of the 14th Design Automation Conference*, New Orleans, June 20–22, 1977, pp. 462–468.
5. E. B. Eichelberger, E. I. Muehldorf, R. G. Walther, and T. W. Williams, "Method of Propagation Delay Testing a Level Sensitive Array Logic System," U.S. Patent 4,063,080, December 1977.
6. G. L. Smith, "Model for Delay Faults Based upon Paths," *Proceedings of the 1985 IEEE International Test Conference*, Philadelphia, 1985, pp. 342–349.
7. Donald S. Cleverley, "The Role of Testing in Achieving Zero Defects," *Proceedings of the 1982 IEEE International Test Conference*, November 1982, pp. 248–253.
8. M. A. Breuer, "The Effects of Races, Delays, and Delay Faults on Test Generation," *IEEE Trans. Computers* **C-23,** 1078–1092 (October 1974).
9. E. P. Hsieh, R. A. Rasmussen, L. J. Vidunas, and W. T. Davis, "Delay Test Generation," *Proceedings of the 14th Design Automation Conference*, New Orleans, June 20–22, 1977, pp. 486–491.
10. T. M. Storey and J. W. Barry, "Delay Test Simulation," *Proceedings of the 14th Design Automation Conference*, New Orleans, June 20–22, 1977, pp. 492–494.
11. J. D. Lesser and J. J. Shedletsky, "An Experimental Delay Test Generator for LSI Logic," *IEEE Trans. Computers* **C-29,** 235–248 (1980).
12. C. C. Liaw, S. Y. H. Su, and Y. K. Malaiya, "Test Generation for Delay Faults Using Stuck-at Fault Test Set," *Proceedings of the 1980 IEEE International Test Conference*, Philadelphia, November 1980, pp. 167–175.
13. Y. K. Malaiya and R. Narayanaswamy, "Modeling and Testing for Timing Faults in Synchronous Sequential Circuits," *IEEE Design & Test of Computers* **1,** 62–74 (November 1984).
14. T. Hayashi, K. Hatayama, K. Sato, and T. Natabe, "A Delay Test Generator for LSI Logic," *Proceedings of the 14th International Conference on Fault Tolerant Computing*, June 1984, pp. 146–149.
15. T. Shimono, K. Oozeki, M. Takahashi, M. Kawai, and S. Funatsu, "An AC/DC Test Generation System for Gate Array LSIs," *Proceedings of the 1985 IEEE International Test Conference*, Philadelphia, 1985, pp. 329–333.
16. M. A. Breuer, "A Random and an Algorithmic Technique for Fault Detection Test Generation for Sequential Circuits," *IEEE Trans. Computers* (short notes) (Special Issue on Fault Tolerant Computing) **C-20,** 1364–1370 (1971).
17. P. Agrawal and V. D. Agrawal, "Probabilistic Analysis of Random Test Generation Method for Irredundant

Combinatorial Logic Networks," *IEEE Trans. Computers* **C-24,** 691–695 (1975).
18. H. D. Schnurmann, E. Lindbloom, and R. G. Carpenter, "The Weighted Random Test-Pattern Generator," *IEEE Trans. Computers* **C-24,** 695–700 (1975).
19. F. Motika, J. A. Waicukauski, E. Lindbloom, and E. B. Eichelberger, "An LSSD Pseudo Random Pattern Test System," *Proceedings of the 1983 IEEE International Test Conference*, Philadelphia, November 1983, pp. 283–288.
20. J. A. Waicukauski and F. Motika, "Testing VLSI Chips with Weighted Random Patterns," *Proceedings of the 1989 International Symposium on VLSI Technology, Systems and Applications*, Taipei, Taiwan, May 1989, pp. 149–154.
21. J. A. Waicukauski, E. Lindbloom, E. B. Eichelberger, and O. P. Forlenza, "A Method for Generating Weighted Random Test Patterns," *IBM J. Res. Develop.* **33,** 149–161 (March 1989).
22. Z. Barzilai and B. K. Rosen, "Comparison of AC Self Testing Procedures," *Proceedings of the 1983 IEEE International Test Conference*, Philadelphia, November 1983, pp. 89–94.
23. J. A. Waicukauski, E. Lindbloom, B. K. Rosen, and V. S. Iyengar, "Transition Fault Simulation by Parallel Pattern Single Fault Propagation," *Proceedings of the 1986 IEEE International Test Conference*, Philadelphia, 1986, pp. 543–549.
24. Y. Levendel and P. R. Menon, "Transition Faults in Combinational Circuits: Input Transition Test Generation and Fault Simulation," *Digest of Papers, Fault-Tolerant Computing Systems 16*, Vienna, Austria, 1986, pp. 278–283.
25. J. Savir and W. H. McAnney, "Random Pattern Testability of Delay Faults," *Proceedings of the 1986 IEEE International Test Conference*, Philadelphia, 1986, pp. 263–273.
26. H.-J. Wunderlich, "On Computing Optimized Input Probabilities for Random Tests," *Proceedings of the 24th IEEE/ACM Design Automation Conference*, Miami Beach, 1987, pp. 392–398.
27. H.-J. Wunderlich, "Self-Test Using Unequiprobable Random Patterns," *Digest of Papers, Fault-Tolerant Computing Systems 17*, Ann Arbor, 1987, pp. 258–263.
28. F. Brglez, P. Powell, and R. Hum, "Accelerated ATPG and Fault Grading by Testability Analysis," *Proceedings of the International Symposium on Circuits and Systems*, Kyoto, Japan, 1985, pp. 695–698.
29. N. N. Tendolkar, "Analysis of Timing Failures due to Random AC Defects in VLSI Modules," *Proceedings of the 22nd Design Automation Conference*, Las Vegas, June 1985, pp. 709–714.
30. N. N. Tendolkar, "A Logic Chip Delay Test Method for Reducing Timing Failures in Multichip Modules," *Proceedings of the 1987 International Symposium on VLSI Technology, Systems and Applications*, Taipei, Taiwan, May 1987, pp. 218–222.
31. Robert B. Hitchcock, Sr., Gordon L. Smith, and David D. Cheng, "Timing Analysis of Computer Hardware," *IBM J. Res. Develop.* **26,** 100–105 (January 1982).
32. C. Beh, F. Motika, C. Radke, C. Cornell, N. Tendolkar, W. Heller, P. Salvatori, R. Hitchcock, and E. Sharp, "Improved Delay Testing for High Speed Logic," U.S. Patent File S/N-07/062,310, June 15, 1987 (IBM Invention Disclosure No. FI9-87-017).
33. P. Goel, "An Implicit Enumeration Algorithm to Generate Tests for Combinational Logic Circuits," *IEEE Trans. Computers* **C-30,** 215–222 (1981).
34. D. Leet, P. Shearon, and D. France, "A CMOS LSSD Test Generation System," *IBM J. Res. Develop.* **28,** 625–635 (September 1984).
35. S. Sugamori, K. Takeuchi, H. Maruyama, and S. Kamata, "High-Fidelity Device Tester Interface," *Proceedings of the 1983 IEEE International Test Conference*, Philadelphia, 1983, pp. 371–378.
36. B. G. West, "Attainable Accuracy of Autocalibrating VLSI Test Systems, *Proceedings of the 1983 IEEE International Test Conference*, Philadelphia, 1983, pp. 193–199.
37. S. Sugamori, H. Maruyama, and T. Sudo, "Analysis and Definition of Overall Timing Accuracy in VLSI Test System," *Proceedings of the 1981 IEEE International Test Conference*, 1981, pp. 143–153.

Received May 15, 1989; accepted for publication September 7, 1989

Franco Motika *IBM General Technology Division, East Fishkill facility, Route 52, Hopewell Junction, New York 12533.* Mr. Motika is currently a senior technical staff member in the area of VLSI logic test, developing the weighted random-pattern test methodology. He joined the IBM General Technology Division in 1968 in an advanced semiconductor memory characterization group and has since been involved with specials and logic test. During this time Mr. Motika's assignments have included the architecture, software and hardware development, and design and implementation of various VLSI test systems. He has received an IBM Corporate Outstanding Technical Innovation Award (1988), an Outstanding Innovation Award (1986) for the weighted random-pattern test concept, a GTD Achievement Award and a GTD Division Award (1983) for contributions to logic final test, and three Outstanding Technical Achievement Awards for test throughput and product SPQL enhancements. He has also received several Invention Achievement Awards for various patents and innovations in test. Mr. Motika received a B.S. degree in 1968 and an M.S. degree in 1969, both in electrical engineering, from the Polytechnic Institute of Brooklyn, New York. He is a member of the Computer Society of the Institute of Electrical and Electronics Engineers.

Nandakumar N. Tendolkar *IBM Data Systems Division, Poughkeepsie, New York 12602.* Dr. Tendolkar is a senior engineer in the Noise Detection and Recovery System Department. He is currently involved in the design and evaluation of large-system error detection, FRU isolation, and recovery support hardware and microcode. Dr. Tendolkar received the B. Tech. (Hons.) degree in mechanical engineering from the Indian Institute of Technology, Bombay, in 1966, the M.S. degree in operations research from Cornell University in 1968, and the Ph.D. degree in computer and information science from Syracuse University. He joined IBM in 1967 at Endicott and worked until 1972 in application of operations research to planning, participating in the design of the automated Material Distribution Center at Endicott. From 1972 to 1977, he worked at the Poughkeepsie laboratory on system performance evaluations, and from 1977 to 1980, he helped to implement the diagnostic strategy for the System/3081. Since joining his present department in 1980, Dr. Tendolkar has been involved in testing for VLSI multi-chip modules, hardware error detection, fault-isolation and recovery techniques for large computer systems, and reliability analysis of fault-tolerant systems. He has received an IBM Outstanding Innovation Award for work on 308X diagnostics. Dr. Tendolkar is a senior member of the Institute of Electrical and Electronics Engineers.

Chao C. Beh *General Technology Division, East Fishkill facility, Route 52, Hopewell Junction, New York 12533.* Mr. Beh graduated from Cheng-Kung University, Taiwan, in 1958 with a B.S. degree in electrical engineering. He received an M.S. degree in electrical engineering from Kansas State University in 1962. He is currently an advisory engineer in the area of VLSI test methodology. Mr. Beh joined the product assurance group of the IBM General Technology Division in 1969, participating in the qualification of both bipolar and FET technologies. In addition, he has also evaluated GTD's Design Automation System, with emphasis on test generation and stuck-fault modeling. In 1981 Mr. Beh transferred to the GTD East Fishkill laboratory, where he has since worked on

various tasks in the area of test methodology and application, including circuit testability analysis, stuck-fault modeling, and development of a cost-effective delay-test methodology. Mr. Beh received an IBM Outstanding Technical Achievement Award in 1985 for product SPQL enhancements.

William R. Heller *IBM Data Systems Division, Poughkeepsie, New York 12602.* Dr. Heller is currently a senior physicist in the IBM Fellow group at the Poughkeepsie laboratory, working on design algorithms and analysis of large computer systems. He has been in his present position since 1979. Before joining IBM, he received his B.S. in applied mathematics from Brown University in 1945 and, after military service in 1946 and 1947, he received his Ph.D. in theoretical solid-state physics as an AEC Predoctoral Fellow at Washington University, St. Louis, Missouri. After postdoctoral work at the University of Illinois and a year as assistant professor at Yale University, he worked from 1953 to 1959 at Shell Development Corporation, Emeryville, California, involved in studies of materials science. Joining IBM at the Thomas J. Watson Research Center, Yorktown Heights, New York, in 1959, he continued to work in materials science until 1964, when he became manager of physics in San Jose Research. In 1967 he joined the technical staff of the director at San Jose, and in 1969 came to the Components Division, East Fishkill, New York, to manage a group in engineering analysis and design techniques development. In 1978, Dr. Heller was on leave as Visiting Professor of Computer Science at the California Institute of Technology, as IBM's representative to the Silicon Structures Project developing VLSI design techniques. Dr. Heller is a Fellow of the American Physical Society, a Fellow of the Institute of Electrical and Electronics Engineers, and a member of the New York Academy of Sciences.

Charles E. Radke *IBM General Technology Division, East Fishkill facility, Route 52, Hopewell Junction, New York 12533.* Dr. Radke received a Ph.D. degree in electrical engineering from Case Institute of Technology (now Case Western Reserve University) in 1964. He joined IBM in 1964 in Endicott and has been involved in algorithm development and mathematical modeling relating mainly to design automation technology; more recently he has concentrated on semiconductor final test. In 1979 he formed a test methodology and applications group within the Semiconductor Product Development Laboratory in order to build and plan test into East Fishkill products. Dr. Radke is currently a senior engineer manager in the Semiconductor Laboratory. He manages a development group which is working on circuit design for testability and test methods to improve semiconductor product quality. Dr. Radke is a member of the Computer Society of the Institute of Electrical and Electronics Engineers, the Association for Computing Machinery, Sigma Xi, Tau Beta Pi, and Eta Kappa Nu.

Phillip J. Nigh *IBM General Technology Division, East Fishkill facility, Route 52, Hopewell Junction, New York 12533.* Mr. Nigh is a senior associate engineer at IBM East Fishkill Laboratory Test Facility. In 1983 he joined IBM and for three years had the responsibility of future product characterization and high-speed testing techniques. He received a B.S. degree from Case Western Reserve University in computer engineering in 1983 and an M.S. degree from Syracuse University in computer engineering in 1986. Mr. Nigh is currently a doctoral candidate in IBM's Resident Study Program at Carnegie Mellon University, where his research concerns the Built-In Current testing methodology. In 1985 he received an IBM Outstanding Technical Achievement Award for his work in reducing chip-related system-level fallout.