

# AC Test Quality: Beyond Transition Fault Coverage

Yaron Aizenbud  
Moshe Leibowitz

Paul Chang  
Dave Smith

Bernd Koenemann

Vijay Iyengar  
Barry Rosen

IBM  
Haifa Research Group  
Israel

IBM  
Test Design Automation  
Endicott

IBM  
EDA  
Poughkeepsie

IBM  
Research Division  
Yorktown Heights

## Abstract

This paper describes a simulator to determine the quality of AC tests beyond just the transition fault coverage. The notion of sizes of detected delay faults is relevant when modeling point delay defects as described in the literature for CMOS circuits. The computation described in this paper and the fault sizes computed are valid under the single fault assumption. Experimental results from the simulator are presented for real designs. Such experiments are crucial in determining test strategies that are both cost effective and achieve high test quality.

## 1. Introduction

AC testing (also referred to as delay fault testing) has now become a necessary part of the manufacturing test process for integrated circuits and higher assemblies. Uses of AC testing in some form can be found in bipolar and CMOS technologies [1, 4, 15]. Fault models such as the transition fault model can be used as a basis for testing gross delay faults. It has been reported that a significant fraction of defects result in large excess delays [1]. Clearly, a transition fault based test strategy can be quite effective for these defects. However, there are other defects that can be modeled only by delay faults with sizes smaller than the cycle time of the product being tested. The timing associated with stimuli and observation during test becomes even more important for these smaller delay faults. A delay test method that derives these times from the normal system timing is described in [15]. More recently, techniques were presented to optimize the test times using delay data at the logic gate level [8, 9, 14]. All techniques for tight test application times are driven by the same goal, i.e., to detect the smallest possible delay faults. The paths along which faults are excited and propagated depend on the patterns used. Therefore, the test pattern set used is the other factor that affects the sizes of detected delay faults. This paper describes an industrial system to determine "how well" delay faults are detected by the combination of the test application times and the test pattern set. The *Small Delay Fault Simulator* (SDFS) works with the given times and patterns, regardless of their origins.

Models have been presented for computing the sizes of detected delay faults [3, 10, 13, 16, 17]. This paper uses

a modified version of the model in [10] that is better suited to the gate delay model used in the industry. The handling of inertia has also been enhanced in this new model. Due to their increased complexity, computations of multiple intervals of detected fault sizes [17] were not incorporated in SDFS. Calculations simpler than [10] have been proposed [3, 13, 16], but these simplified calculations yield incorrect results in some situations [7].

## 2. AC Test in the Scan Environment

Since the simulator described in this paper is designed for scan designs, we first review AC testing in this environment. The basic concepts of scan design were formalized into a comprehensive approach for system design, debug and maintenance in the late '60s and early '70s (e.g., [11, 12, 21]). Since then, many different versions of scan design implementations have proliferated. What they have in common is the serial access to internal storage elements, where they differ is mainly in the clocking structure associated with the operating the storage elements.

Level sensitive scan design (LSSD) [2] is characterized by a unique clocking discipline: any two storage elements that can communicate with each other via combinational logic must be controlled from separate clock primary inputs. The LSSD storage elements are implemented as shift register latches (SRLs) consisting of two latches called L1 and L2. For scanning, the L1s are clocked by a dedicated shift A clock and the L2s are clocked by a dedicated shift B clock. The L1s generally have a separate system data port which is controlled by a separate LSSD system clock (typically referred to as C1). The LSSD rules further prohibit any combinational feedback loops, and require that storage elements are latches which can only be updated when pulsing dedicated clock primary inputs (strongly clocked design). It should be noted, that these restrictions often lead to a dichotomy between normal functional operation and test mode operation in practical LSSD networks. In many designs, the LSSD clocks are only used for testing, while other pins are used for functional clocking. The functional clock signals appear to be clock gating signals in the LSSD test mode and vice versa.

Y. Aizenbud is currently with Intel at Haifa, Israel.

Although scan design based testing is generally considered to primarily support static testing, efforts to include dynamic testing into the Scan design framework had been made from the beginning. An early description can, for example, be found in the so called 2-cycle test introduced in [18], which suggests to gate two consecutive clock pulses into the logic under test after the network has been initialized by Scan-In. The first pulse is intended to release signal transitions, whose effects are captured by the second pulse and logged out using the Scan-Out facilities. A failure to capture the proper response state is expected to indicate a dynamic defect in the network.

The original LSSD test system included delay test generation facilities [6, 20] based on the 2-cycle test concept. Faults are assumed to indefinitely delay a rising or a falling transition at logic block inputs/outputs and are treated as conditional stuck-at faults (that is, a transition of appropriate polarity is required at the fault site to inject the stuck-at fault) in fault simulation. The test generation algorithm sensitizes a path from an input (or SRL) to an output (or SRL) through a single fault site, and sets up conditions and clock pulses for producing, propagating, and capturing a transition of the appropriate polarity. To minimize the chance of masking small delay defects by delay variations in the propagating blocks, the system attempts to select two "disjoint" paths (if possible the longest and the shortest in terms of delay) through the fault site. Each test is individually timed according to the propagation delay along the sensitized path(s) and the tester accuracy limitations, to allow for characterizing any delay faults as accurately as possible. The timing for a given test is dominated by the longest sensitized path(s) in this test. This can lead to less accurate testing of faults that happen to lie on shorter sensitized paths in the same test. The conditional stuck-at fault simulation marks faults as detected regardless of the sensitized path length. Further, an approach based on event driven simulation is used to estimate the sizes of delay faults detected [19].

Because the original LSSD delay test method attempts to sensitize specific paths for each fault, and uses individual timing for each test, it was found to suffer from generating too many tests and timing sets to be cost efficient. Consequently, this delay test system was not used in production testing. Only static LSSD testing was performed for many years.

The LSSD delay test activities were revived in the first half of the '80s, when a particular delay defect caused larger than normal chip quality problems in a mainframe assembly. The chip test scheme adopted to detect these defects [15] was designed to be compatible with the limited number of available timing sets on existing test equipment. Stuck-at fault tests were converted into 2-cycle tests with pattern independent timings that were derived from the system timing during normal operation. These 2-cycle tests, despite being derived from stuck-at fault tests without an explicit delay fault model, proved to be economical and quite effective in detecting a large number of delay defects, and have become widely used in LSSD based production testing.

The practical success of the delay tests derived from stuck-at fault tests, and the need to improve quality even

further, have led to the pursuit of a new LSSD test system for explicit delay fault test generation, which builds on the experiences gained thus far. The new system is similar to the original LSSD delay test system in scope, with three major differences:

- The new system uses the transition fault model [22], which does not attempt to sensitize specific paths and allows for fault propagation on glitches.
- The new system allows for SRL initialization by skewed load (that is, the Scan-In is stopped on an A-clock which potentially leaves different values in the L1 and L2 of an SRL), while the original system always obtained the initialization from the system data port of the L1s. The test based on skewed load, when possible, is easier to generate since the test generator does not have to consider two time images of the combinational logic.
- The new system does not time individual tests, but groups of tests with common sequences of dynamic events (that is clock pulses, etc.). This simplifies the test hardware required.

These differences are intended to reduce the number of generated tests and unique timing sets compared to the original system. Timing that varies by pattern and by fault (as well as by pin) would provide better tests in principle (and has worked well on small benchmarks [14]), but the experience with the original LSSD delay test method suggests that highly variable timing has excessive data handling costs in practice.

Like the original system, the new system operates the product under test in the LSSD mode. That means, only the designated LSSD clocks are used and all timings must be derived relative to these LSSD clocks. The test sequences have the following general structure:

1. Scan-In.
2. If needed, pulse LSSD system clocks to initialize SRLs from system data ports.
3. Initialize primary inputs.
4. Release transitions by changing stimuli to primary inputs and/or pulsing LSSD clocks (release clocks).
5. If needed, change gating primary inputs, and/or gating SRLs by pulsing LSSD clocks (gating clocks).
6. Capture propagated transition effects by strobing primary outputs and/or pulsing LSSD clocks (capture clocks).
7. Scan-Out.

Steps 1, 2, 3, and 7 are performed statically, while steps 4, 5, and 6 are the actual dynamic events performed at-speed. Step 5 is generally an artifact of the difference between functional and LSSD clocking. Functional clock signals intended for (pseudo-) edge-triggered clocking often gate two LSSD clocks (at least two LSSD clocks are required) such that only one of the two LSSD clocks is gated through at any time. A delay test that uses one of the LSSD clocks as release clock and the other as capture clock can only be performed by changing the gating signal (i.e. the functional clock signal) between release and capture. In many cases, however, this gating is not

required, and the test sequences are essentially 2-cycle tests as described above.

Each unique dynamic event sequence produced during test generation for a group of tests is timed such that on the tester no violations of minimum pulse width, set-up time, and hold time constraints occur on the involved LSSD clock signals and on the tester stimulus-change and output-strobe commands.

### 3. Simulation Models

In this section we describe the models used in SDFS for logic simulation and for signal timing calculations. Both the logical and timing calculations have to be performed in two contexts - for the faultless circuit and in the presence of the single delay fault being simulated.

#### Logic Model

A two-level hierarchical circuit model is used. At the highest level, the logic is represented as an interconnection of N-blocks, which correspond to entities in the technology dependent library used by the designers. The N-blocks, in general, are multiple input, multiple output entities that have both physical and logical characteristics. Each N-block can be expanded as an interconnection of primitive logic gates (e.g., AND, OR, NOT, tristate buffer, latch primitive). These primitive entities in the expansion are called E-blocks. This expansion not only represents the normal logical operation of the N-block, but also has placeholders for single stuck-at faults and single transition faults that correspond to possible defects in the N-block. Logic simulation, in both the faultless and faulty case, can be performed using these E-block expansions for the circuit. The computation in SDFS is done using single output entities called segnets which are simply derived by extracting the E-block expansion for each output of each N-block. SDFS can handle two types of segnets. Latch segnets contain memory elements and have no combinational paths between their inputs and output. Combinational segnets have no memory elements in them and so all paths between the inputs and the segnet output go through combinational logic. Combinational segnets are classified as *standard* if their function is AND, NAND, OR, NOR, or NOT. They are otherwise classified as *nonstandard* segnets. The calculations for standard segnets take advantage of the knowledge of input controlling values to compute more precise waveforms than in the case of nonstandard segnets. The simulation process in SDFS always starts at primary inputs and latch segnets, and stops at primary outputs and latch segnets.

Figure 8 shows a circuit as an interconnection of N-blocks that will be used to illustrate the computations in SDFS. The two latches in this example are controlled by the primary clock inputs C1 and C2.

#### Delay Model

Delays are modeled at the N-block level, since these correspond to physical entities in the technology library. Three types of delays are considered in SDFS. They are block I/O delays, source-to-sink delays and block time constraints.

1. N-block I/O delays are block internal delays which represent the propagation time of a transition from a given input pin of the N-block to a given output pin.
2. Source-to-sink delays are block external delays which describe the propagation time of a transition along a given net, from a N-block output pin to another N-block input pin.
3. N-block time constraints (for latches only) specify the time relations between incoming signals. Included are setup and hold times between data-clock pairs, clock minimum pulse width for each clock, and the minimum separation time between clock signals.

These N-block delays can be easily represented in the segnet model also, since segnets merely consider each output of the N block separately.

Each delay is actually *two* numbers, representing best and worst (i.e., fastest and slowest) values for nominal process conditions. For example, a delay may have best and worst case values of 4 and 5 nsecs. Having a range allows one to represent the delay variations due to factors like logic values on other inputs, which may determine the path taken through a complex N-block. In principle, statistical information about variations in process conditions could be used to adjust the best case delays downward and the worst case delays upward from their nominal values. SDFS does not use a statistical model.

The delays for the N-block in the example in Figure 8 are given in Figure 9. Equal best and worst case delays are given in the example for the sake of simplicity. Source-to-sink delays are ignored for the sake of simplicity.

#### Fault Model

The inputs to SDFS are the hierarchical model (logic and delays) of the circuit, the input pattern set, and the test application timings for the patterns being simulated. The delay fault model used is essentially as described in [10]. The actual advantages and disadvantages of various models are too complex to be discussed here [7, 10]. Single delay faults (slow-to-rise and slow-to-fall) are considered at inputs and outputs of segnets. In this model, a single delay fault is said to be detected if the corresponding transition fault (of the same polarity at the same site) is detected. A *detection threshold*  $\epsilon$  is computed for the detected fault, such that the delay fault is guaranteed to be detected for all sizes greater than  $\epsilon$ . A larger than acceptable detection threshold implies that either the test application times need to be tightened or the test pattern set needs to be improved.

### 4. Simplified Waveforms and Tester Timing

Information about stimulus applied by the tester is represented by simplified waveforms as suggested in [10]. Waveforms for nets not carrying clocks are represented using four quantities (*IV*, *FV*, *EA*, *LS*) that are defined below.

- *IV*(*s*) – *Initial Value*  
Initial logic value at net *s* before transitions are launched.

- $FV(s)$  – Final Value  
Final logic value at net  $s$  after transitions have been launched, and all transients have stabilized.
- $EA(s)$  – Earliest Arrival  
Earliest possible arrival time of a transition at net  $s$  due to any transition that has been launched. Signal  $s$  has the initial value  $IV(s)$  at all times  $t < EA(s)$ .
- $LS(s)$  – Latest Stabilization  
Latest possible stabilization time of the final value at  $s$  after all transients have settled. Signal  $s$  has the final value  $FV(s)$  at all times  $t > LS(s)$ .

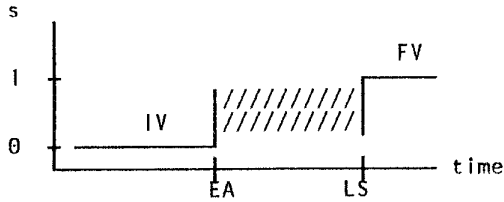


Figure 1. Simplified waveform representation

Figure 1 illustrates these two determinate logic values and two times associated with a net. This simplified representation ignores all waveform details during the interval of time  $EA(s) \leq t \leq LS(s)$ . A net may have  $FV(s) = IV(s)$  or  $FV(s) = \overline{IV}(s)$ . But the second case does not imply steadiness. A waveform may briefly take other values during the interval of uncertainty. Steadiness is represented by  $EA(s) > LS(s)$ . Of course some nets hold steady values. Such a net has the same initial and final values, and it also has this value at all times  $t$ . Steadiness is represented by  $EA(s) = +\infty$  (no disturbance ever arrives) and  $LS(s) = -\infty$  (stabilization has already occurred).

Clock signals require two transitions to be described as shown in Figure 2, leading to two sets of early and late times. One set specifies the clock turn on times ( $CONea$  and  $CONls$ ) and the other specifies the clock turn off times ( $COFes$  and  $COFls$ ).

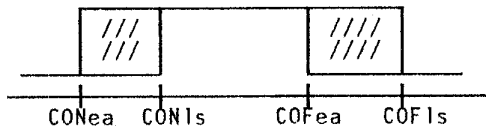


Figure 2. Clock transition times

The test application specification includes the signal transitions being applied on primary inputs, the timing associated with these transitions, and the times when primary outputs are observed. We assume that the simplified waveforms for clock and nonclock primary inputs can be extracted from the test application specification. We also extract the relevant observation times for the primary outputs.

## 5. Computations of Faultless Waveforms

Computing the waveforms requires both logic value calculations and timing calculations. SDFS uses a companion logic simulator to evaluate the initial and final logic values. We will first consider propagation of simplified waveforms through combinational logic. First, consider the standard segnet case.

### Standard Segnet Calculation

The  $EA$  and  $LS$  times for the output waveform are computed in two stages. The algorithms to calculate  $PreEA$  and  $PreLS$  for standard segnets are as in [10]. Consider a standard segnet  $G$  with output  $y$ , where  $QCI$  ( $QCF$ ) is the set of all input nets to the segnet with controlling initial (final) values. Similarly,  $QNI$  ( $QNF$ ) is the set of all input nets to the segnet with noncontrolling initial (final) values. Preliminary estimates of  $EA(y)$  and  $LS(y)$  are given by

$$PreEA(y) = \begin{cases} \max_{s \in QCI} \{EA(s) + D_s\} & \text{if } QCI \neq \phi \\ \min_{s \in QNI} \{EA(s) + D_s\} & \text{if } QCI = \phi \end{cases}$$

$$PreLS(y) = \begin{cases} \min_{s \in QCF} \{LS(s) + D_s\} & \text{if } QCF \neq \phi \\ \max_{s \in QNF} \{LS(s) + D_s\} & \text{if } QCF = \phi \end{cases}$$

where  $D_s$  is a placeholder for the appropriate delay number, selected as follows. For  $PreEA$  calculations, best case delays are used. For  $PreLS$  computations, worst case delays are used. The initial and final values at a segnet's input (output) determine the type of transition at that input (output). The proper delay number, from a segnet's input to its output, is the one that matches the I/O transition types. For example, having a transition from 0 to 1 at a segnet's input and a transition from 1 to 0 at the output implies that  $D_s$  should be a rise-to-fall delay number.

The  $EA$  and  $LS$  estimations for the standard and nonstandard segnets are improved by detecting certain cases with steady-signal values on the output  $y$ :

$$EA(y) = \begin{cases} PreEA(y) & \text{if } PreEA(y) \leq PreLS(y) \\ +\infty & \text{if } PreEA(y) > PreLS(y) \end{cases}$$

$$LS(y) = \begin{cases} PreLS(y) & \text{if } PreEA(y) \leq PreLS(y) \\ -\infty & \text{if } PreEA(y) > PreLS(y) \end{cases}$$

### Nonstandard Segnet Calculation

Propagating  $EA$  and  $LS$  values for a nonstandard segnet does not have the benefit of knowing the controlling and noncontrolling values. In this case, we conservatively allow the possibility of any input change to affect the output. Consider a nonstandard segnet  $G$  with output  $y$ , where  $Q$  is the set of all input nets to the segnet. Preliminary estimates of  $EA(y)$  and  $LS(y)$  are given by:

$$PreEA(y) = \min_{s \in Q} \{EA(s) + D_s\}$$

$$PreLS(y) = \max_{s \in Q} \{LS(s) + D_s\}$$

As in the case of standard segnets, the delay  $D_s$  to use depends on the polarity of transitions and whether  $PreEA$  and  $PreLS$  is being computed. The refinement to  $EA$  and  $LS$  is exactly the same as in the standard case.

The calculation of the faultless waveforms starts with the primary inputs and outputs of release latches. The simplified waveforms for primary inputs are available from the test application specification. The waveforms at release latch outputs are calculated using the clocking information as shown below.

For each release latch, the waveform at its clock port needs to be calculated. In the most general case, this can be done for each pattern by propagating the timings for the two transitions associated with the clock pulse (i.e.,  $CONea$  and  $CONIs$ ,  $COFea$  and  $COFIs$ ) from the clock primary input to the latch clock port. The calculations are identical to those described for the corresponding numbers  $EA$  and  $LS$  for a single transition earlier. The  $EA$  and  $LS$  times at the release latch outputs can then be calculated from the clock pulse timings as illustrated by the example below. This assumes that the transitions at the outputs of the release latches are triggered by their clocks. This assumption is valid for the sequences analyzed by SDFS. Consider a latch with an active high clock whose output is undergoing a 0 to 1 transition.

$EA$  at latch output =  $CONea$  + best case clock rise to output rise delay in the latch  
 $LS$  at latch output =  $CONIs$  + worst case clock rise to output rise delay in the latch

The faultless waveform calculation will be illustrated using the running example in Figure 8. The initial contents of both latches is 0. The simplified waveforms at the primary inputs are given below in Figure 3. As explained earlier these are extracted from the tester timing specification and the pattern applied. For example,  $CONea = CONIs = 1$  and  $COFea = COFIs = 51$  for clock C1.

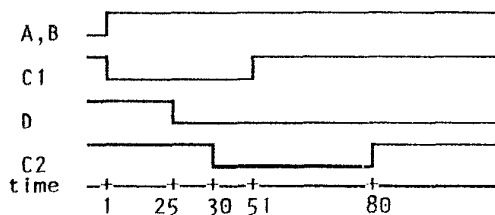


Figure 3. Input waveforms

The clock pulses are propagated to the release and capture latches using the calculations described earlier for each of the two transitions in the pulses. The clock waveforms at nets CR and CC in Figure 8 are given below in Figure 4.

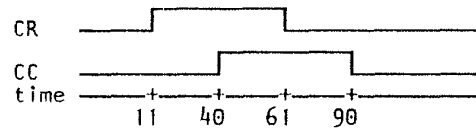


Figure 4. Clock pulse propagation

The output of the release latch has a 0 to 1 transition triggered by the clock CR. The early time  $EA$  for this transition can be computed to be 21 by adding the best case clock to output rise delay (10) for this latch to the  $CONea$  of the waveform at CR (11). The late time  $LS$  for this transition computed using the  $CONIs$  at CR also turns out to be 21. The 0 to 1 transition at F has  $EA = LS = 41$ . The output G of the other AND gate has the same initial and final value, namely 0. The simplified waveform computed for G has  $EA = 41$  and  $LS = 55$ , indicating that the value of G is unknown from time 41 to 55. The waveforms computed for nets E, F, G, H are given below in Figure 5. The arrows on some transitions should be ignored for now and will be explained in the next section. The faultless machine will capture the 1 on the data input H in the capture latch at the end of the clock pulse CC.

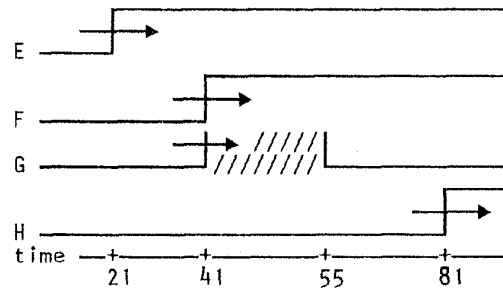


Figure 5. Internal waveforms

The conservatism in our nonstandard segnet calculations can be seen by considering a slightly altered circuit example. Replace the two AND gates and one OR gate in Figure 8 by a nonstandard gate with inputs A, E and D and output H as illustrated in Figure 10. The delays for the nonstandard block are also specified in Figure 10. The function computed by this segnet is the same of that computed by the 3 standard gates (i.e.,  $H = A.E + E.D$ ). We will use the same waveforms as before at nets A, E and D and recompute the waveform at H using the nonstandard segnet calculations.  $EA$  for the 0 to 1 transition at H can be computed to be 61 by shifting the waveforms at A and E by the best case rise-rise delay of 60 and taking the minimum of the early values.  $LS$  for the waveform at H is computed to be 81 by taking the maximum stabilization time of the waveforms at A and E shifted by the worst case rise-rise delays. Since no fall-rise delay is given for the input D, the waveform at D does not affect the calculations for H in this case. The uncertainty interval for the waveform at H is much bigger in the nonstandard case and this is a direct consequence of the

loss of information about input controlling values that was used in the standard case.

## 6. Faulty Machine Calculations

The effects of each delay fault are propagated to each net in the cone of influence of the fault using one logic value (*FPV*) and two reference times (*FRTa*) and (*FRTb*) and a reference fault size ( $\rho$ ) as in [10]. The *FPV* at a net is the faulty machine value that a transition fault simulator would propagate from the fault site to that net. These quantities are used to assert that in the presence of the delay fault of size  $\delta$  the value of the net is *FPV* during the interval of time from *FRTa* to *FRTb* +  $\delta$ , provided that  $\delta > \rho$ . For the fault to be detected an incorrect value must be observed at least at one observation point. The observation time (*OBT*) at a primary output is specified in the test application timing on the tester. The effective observation time *OBT* at a capture latch has to be computed by adding the worst case hold time to the *COFls* time for its clock input. This calculation guarantees that the fault effect has to last long enough to satisfy the hold time requirements for the latch.

The faulty machine waveform calculation starts at the fault site. At the fault site, the fault effect results in the initial value persisting for an extra period of time equal to the size of the fault:

$$FRTa = -\infty ; FRTb = EA(\text{faultsite}) ; \rho = 0.$$

Any net outside the cone of influence of the fault has *FPV* equal to its faultless final value *FV*, which stabilizes after time *LS*. Hence, for such a net

$$FRTa = LS ; FRTb = +\infty ; \rho = 0.$$

Within the cone of influence of the fault, *SDFS* uses the natural event-driven algorithm [10] to calculate the logic value *FPV* and the numbers *FRTa*, *FRTb* and  $\rho$  at any segnet affected by the fault. Any difference between these four values and the default four values

$$FPV = FV ; FRTa = LS ; FRTb = +\infty ; \rho = 0$$

constitutes an event. In particular, *SDFS* may propagate an event to a net when the corresponding transition fault does not propagate.

### Standard Segnet Calculation

Consider a standard segnet *G* with output *y*, where *QC* is the set of all input nets to the segnet with controlling fault propagation values and *QN* is the set of all input nets to the segnet with noncontrolling fault propagation values. Then *FRTb*, *FRTa* and  $\rho$  are calculated as follows:

if  $QC \neq \phi$  then:

$$FRTb(y) = \max_{s \in QC} \{FRTb(s) + D_s\}$$

$$FRTa(y) = FRTa(s_0) + D_{s_0}$$

$$\rho(y) = \max\{\{FRTa(y) - FRTb(y)\}, \rho(s_0)\}$$

$$(s_0 = s \text{ in } QC \text{ which maximizes } FRTb(s) + D_s)$$

otherwise,  $QC = \phi$  :

$$FRTb(y) = \min_{s \in QN} \{FRTb(s) + D_s\}$$

$$FRTa(y) = \max_{s \in QN} \{FRTa(s) + D_s\}$$

$$\rho(y) = \max_{s \in QNF} \{\{FRTa(y) - FRTb(y)\}, \rho(s)\}$$

end

### Nonstandard Segnet Calculation

The calculations for nonstandard segnets are conservative because of lack of information on input controlling values. All the inputs are required to be at their *FPV* for the output to attain its *FPV*. Also, since all the inputs signals are assumed to be relevant, the output  $\rho$  value is bound by the worst input  $\rho$  value. Consider a nonstandard segnet *G* with output *y*, where *Q* is the set of all input nets to the segnet. The evaluation of *FRTb*, *FRTa* and  $\rho$  are given by:

$$FRTb(y) = \min_{s \in Q} \{FRTb(s) + D_s\}$$

$$FRTa(y) = \max_{s \in Q} \{FRTa(s) + D_s\}$$

$$\rho(y) = \max_{s \in Q} \{\{FRTa(y) - FRTb(y)\}, \rho(s)\}$$

### Detection Threshold

A detectable delay fault causes one or more observation points to have the value  $FPV = \overline{FV}$  at the observation time (*OBT*). *SDFS* computes a detection threshold  $\epsilon$  for each fault such that the fault is guaranteed to be detected for all sizes greater than  $\epsilon$ . Consider a fault *f* that is detected at the observation point *y*. The observation time *OBT*(*y*) and the quantities describing the fault effect at each detection point *y* (i.e., *FRTa*, *FRTb*,  $\rho$ ) are used to determine the detection threshold  $\epsilon$ .

$$\epsilon(y) = \max\{OBT(y) - FRTb(y), \rho(y)\}$$

For a fault that is detected at multiple observation points or by multiple patterns the minimum detection threshold is reported.

Consider the slow-to-rise delay fault at the output *E* of the release latch in Figure 8. The faulty machine calculations are summarized in the Figure 6 below, where the input waveforms are as in Figure 3.

net	FPV	FRTa	FRTb	rho
A	1	1	+inf	0
D	0	25	+inf	0
E	0	-inf	21	0
F	0	-inf	41	0
G	0	55	+inf	0
H	0	-inf	81	0

Figure 6. Faulty machine results

The effect of the fault is indicated in Figure 5 by marking with arrows the transition being delayed by the fault. The effect of the fault on net G is to cause the leading edge of the 0-1-0 glitch to be delayed. A detailed simulation of the faultless waveform at G would show a glitch between 41 and 55. The simplified fault effect calculations done in SDFS do not show this detail, but correctly assert that this signal has the fault propagation value 0 for all times after 55. The fault effect does propagate through the other AND gate and reaches the capture latch. The effective observation time for this latch is  $90 + 5 = 95$ . Therefore the detection threshold computed is 14.

Fault effect propagation through nonstandard blocks will be illustrated using the circuit in Figure 10.  $FRTa$  at H is determined by  $FRTa$  of input D and is computed to be 85.  $FRTb$  at H is determined by  $FRTb$  of input E and is computed to be 81.  $\rho$  at H is 4. In this case, these conservative fault effect calculations do not result in a larger threshold and  $\epsilon$  computed is 14.

As mentioned earlier, SDFS does not calculate multiple ranges of detected fault sizes. Consider the faulty waveform shown in Figure 7, where the transitions delayed by the fault are marked by arrows. The detection threshold computed by SDFS (marked in the figure) corresponds to a fault large enough to delay edge C until after  $OBT$ . In this example, the fault is also detected if it is large enough to delay edge E until after  $OBT$  but still small enough to leave edge D before  $OBT$ . A test is at least as good as SDFS says it is. The test may be somewhat better, but much more elaborate and expensive calculations would be needed to determine the ranges of detected fault sizes.

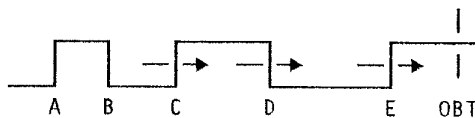


Figure 7. A glitching waveform delayed by a fault

## 7. Experimental Results

In this section we will present data generated from some typical runs on CMOS designs. The characteristics of these designs are given in Figure 11. This table gives the numbers of primary inputs, primary outputs, and latches in each design. The N-blocks are counted in two categories - standard and nonstandard. The number of expanded primitive gates (E-blocks) is also listed. Typical

delays for a three input NAND gate in these designs range from 0.5 nsecs to 1.0 nsecs. Only those delay faults that are located on the boundaries of the N-blocks are considered for analysis by SDFS. The runs reported here exclude delay faults on signals that gate or carry clock signals.

The first set of runs were performed by first filtering the test patterns through a transition fault simulator. SDFS is invoked for a (fault, pattern) pair only if the transition fault simulator determines the fault to be detected by this particular pattern. Faults are not dropped when the corresponding transition faults are detected since subsequent patterns may detect these faults with smaller detection thresholds. A fault is dropped only if a detection threshold of zero is achieved. The runs could be made more efficient by determining a lower bound for the detection threshold at each fault site for the given set of test sequences and timings and then dropping the fault when that lower bound is achieved. The table in Figure 12 summarizes the results of the first set of simulation runs. This table lists the number of delay faults that were filtered down for simulation by SDFS. The pattern count includes any pattern that was simulated by SDFS against one or more delay faults detected by it. The average of the detection thresholds over all the simulated faults is given. The CPU time taken (on IBM 3090-600) is split into two categories - time taken for the SDFS calculations and the total time taken by SDFS and the transition fault simulator that is filtering the (fault, pattern) pairs. The largest design Z took too much time to simulate if the faults were left in until they were detected with zero threshold. Upper bounds on the detection thresholds achieved were computed by performing the simulation run for this design with faults being dropped after being detected by ten patterns. Dropping a fault early will give conservative results since later detections of the fault with smaller thresholds are not simulated. The amount of conservatism depends on the design, on how many detections are simulated before dropping, and on the order in which patterns are simulated. For example, dropping the faults after the tenth detection resulted in an average detection threshold of 16.6 nsecs for the design X. This should be compared with the value 16.2 nsecs computed without early dropping of faults. The detection profile showing the percentages of faults detected with different threshold ranges is given in Figure 13. This profile gives a more detailed summary of how well the delay faults are detected and also helps in determining ways of improving the coverage.

The analysis by SDFS can be used to compare different test methodologies. For example, the tester cost versus capability trade-off may suggest reducing the pattern set by keeping only those patterns that detect transition faults or stuck-at faults undetected by previous patterns. This reduced pattern set would have the same transition and stuck-at fault coverage but may have significantly poorer coverage considering the sizes of the detected delay faults. The degradation in coverage can be computed by determining the detection thresholds for the reduced pattern set (without dropping the faults as before). We did not generate this reduced pattern set for analysis. However, upper bounds on the detection thresholds achieved by the reduced pattern set can be easily

computed by SDFS using the original full pattern set. The faults are dropped after the first detection in this simulation run to compute the bounds. Results of such a simulation run for each of the examples are summarized in Figure 14. These indicate that the degradation of coverage is not significant for these examples and the corresponding pattern sets.

### 8. Conclusions

We have described a simulator that can quantify the quality of detection for delay faults for real designs. This tool is useful in grading the test timings and the pattern sets used in AC testing of designs. Such an analysis is crucial to compare different test strategies and to design a cost effective AC test scheme for digital ICs.

### References

- [1] O.Bula, J. Moser, J. Trisko, M.Weissman and F. Woytowich, "Gross Delay Defect Evaluation for a CMOS Logic Design System Product," *IBM Journal of Research and Development*, (March/May 1990), Vol. 34, No.2/3, pp. 325-338.
- [2] E.B. Eichelberger and T.W. Williams, "A Logic Design Structure for LSI Testability," *Proc. 14th ACM-IEEE Design Automation Conf.* (June 1977), pp. 462-468.
- [3] F. Fink, K. Fuchs, and M.H. Schulz, "An Efficient Parallel Pattern Gate Delay Fault Simulator ...," *Proc. European Test Conf.*, (April 1991), pp. 171-180.
- [4] T. Hayashi, K. Hatayama, K.Sato, and T. Natabe, "A Delay Test Generator for Logic LSI," *Proc. IEEE International Symp. on Fault-Tolerant Computing*, (June 1984), pp. 146-149.
- [5] R. B. Hitchcock, Sr., G. L. Smith, and D. D. Cheng, "Timing analysis of computer hardware," *IBM Journal of Research and Development*, (January 1982), Vol. 26, No. 1, pp. 100-105.
- [6] E.P.Hsieh, et al, "Delay Test Generation," *Proceedings Design Automation Conf.*, pp. 486-491, 1977.
- [7] V.S. Iyengar, S.M. Reddy and B.K. Rosen, "Delay Fault Testing," *IBM Research Report RC 16488*, (January 1991).
- [8] V.S. Iyengar and G. Vijayan, "Test Application Timing - The Unexplored Issue in AC Test," *Proc. International Test Conf.*, (October 1991), pp. 840-847.
- [9] V.S. Iyengar and G. Vijayan, "Optimized Test Application Timing for AC Test," IBM Research Report RC 15692, April 17, 1990; To appear in *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*.
- [10] V.S. Iyengar, B.K. Rosen, and J.A. Waicukauski, "On Computing the Sizes of Detected Delay Faults," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 9, Num. 3, (March 1990), pp. 299-312.
- [11] Kobayashi et al., "Flip-Flop Circuit with FLT Capability", *Proc. HECEO Conf.*, p. 962, 1968.
- [12] K.Maling and E.L.Allen, "A Computer Organization and Programming System for Automated Maintenance", *IEEE Trans. Electr. Comp.*, pp. 887-895, 1963.
- [13] W.-W. Mao and M.D. Ciletti, "A Simplified Six-Waveform Type Method for Delay Fault Testing," *Proc. 26th Design Automation Conf.*, pp. 730-733, 1989.
- [14] W.-W. Mao and M.D. Ciletti, "A Variable Observation Time Method for Testing Delay Faults," *Proc. 27th Design Automation Conf.*, pp. 728-731, June 1990.
- [15] F. Motika, N.N. Tendolkar, C.C. Beh, W.R. Heller, C.E. Radke, and P.J. Nigh, "A Logic Chip Delay Test Method Based on System Timing," *IBM Journal of Research and Development*, Vol. 34, Num. 2/3, (March/May 1990), pp. 299-313.
- [16] A.K. Pramanick and S.M. Reddy, "On the Detection of Delay Faults," *Proc. International Test Conf.* (September 1988), pp. 845-856.
- [17] A.K. Pramanick and S.M. Reddy, "On the Computation of the Ranges of Detected Delay Fault sizes," *Proc. International Conf. on CAD* (November 1989), pp. 334-338.
- [18] R.J.Preiss, "Fault Test Generation", in *Design Automation of Digital Systems* edited by M.A.Breuer, p. 393, 1972.
- [19] T.J.Snethen and T.M.Storey, "Index of Slack as Delay Fault Measurement Tool," *IBM Tech. Disclosure Bulletin*, Vol. 17 (No.12), May 1975.
- [20] T.M.Storey and J.W.Barry, "Delay Test Simulation," *Proceedings Design Automation Conf.*, pp. 492-494, 1977.
- [21] A.Toth and C.Holt, "Automated Database-Driven Testing", *Computer*, pp. 13-19, January 1974.
- [22] J.A.Waicukauski et al., "Transition Fault Simulation by Parallel Pattern Single Fault Propagation", *Proc. IEEE International Test Conf.* (October 1986), pp. 542-549.

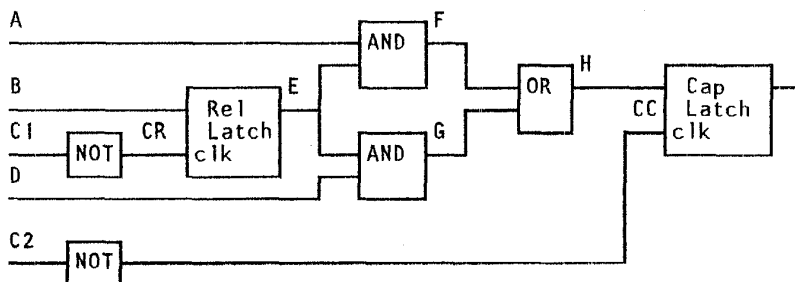


Figure 8. Example sequential circuit

N-Block	delay type	value
AND	input rise to output rise	20
AND	input fall to output fall	30
OR	input rise to output rise	40
OR	input fall to output fall	30
NOT	input rise to output fall	10
NOT	input fall to output rise	10
LATCH	clock turn on to output rise	10
LATCH	hold time required	5

Figure 9. Block delays for the example

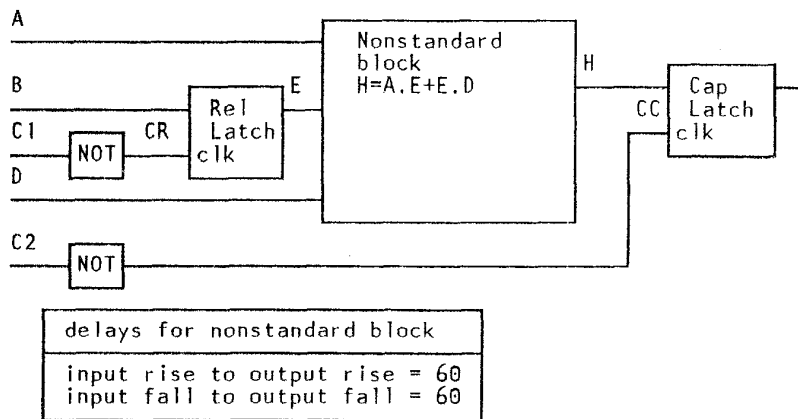


Figure 10. Modified sequential circuit with nonstandard block

Design	Primary Inputs	Primary Outputs	Latches	N-Blocks		E-Blocks
				Standard	Nonstandard	
X	59	15	116	199	481	3182
Y	147	115	333	912	1654	11343
Z	246	185	2090	5558	5942	58303

Figure 11. Design characteristics

Design	Faults Simulated	Patterns Simulated	Average Detection Threshold nsecs	SDFS run time min:sec	Total run time min:sec
X	427	17393	16.2	6:12	12:47
Y	3135	40360	50.4	217:19	267:49
Z**	18214	82306	18.8	585:04	809:51

Figure 12. Summary of results. \*(Faults not dropped in designs X and Y. Faults dropped after 10 detections in design Z.)

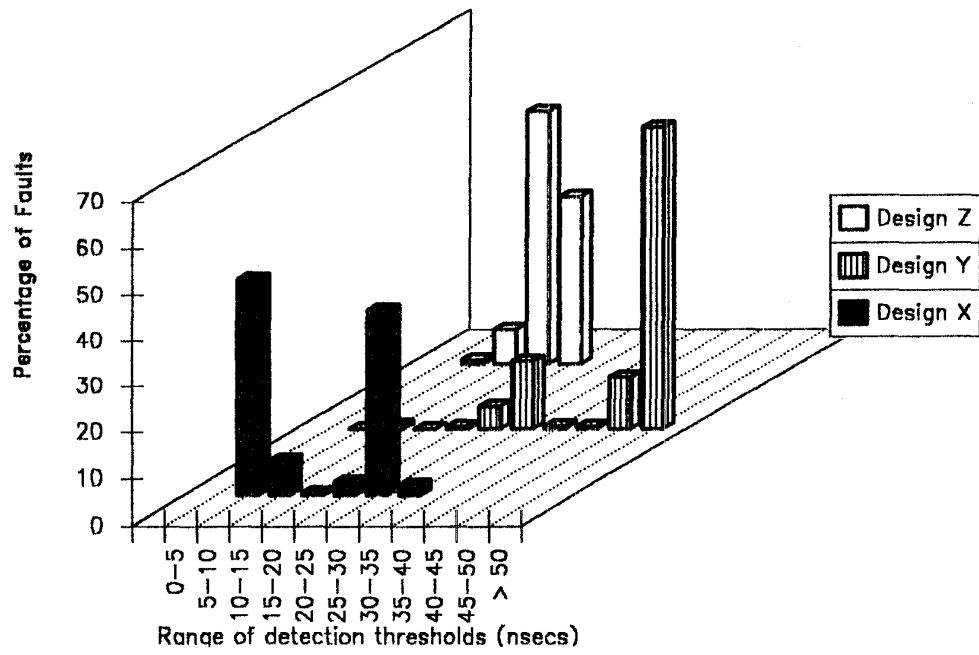


Figure 13. Profile of detection thresholds achieved in experiment summarized in Figure 12.

Design	Average Detection Threshold	SDFS run time min:sec	Total run time min:sec
X	16.8	0:05	6:25
Y	50.6	31:49	78:08
Z	19.3	45:38	242:28

Figure 14. Results of experiment where faults are dropped after their first detection.