University of Nebraska - Lincoln
DigitalCommons@University of Nebraska - Lincoln

CSE Journal Articles | Computer Science and Engineering, Department of

1993

Generating Tests for Delay Faults in Nonscan Circuits

Prathima Agrawal
AT&T Bell Laboratories

Vishwani D. Agrawal
AT&T Bell Laboratories

Sharad C. Seth
University of Nebraska - Lincoln, seth@cse.unl.edu

Follow this and additional works at: http://digitalcommons.unl.edu/csearticles

 Part of the Computer Sciences Commons

Agrawal, Prathima; Agrawal, Vishwani D.; and Seth, Sharad C., "Generating Tests for Delay Faults in Nonscan Circuits" (1993). *CSE Journal Articles*. 31.
http://digitalcommons.unl.edu/csearticles/31

This Article is brought to you for free and open access by the Computer Science and Engineering, Department of at DigitalCommons@University of Nebraska - Lincoln. It has been accepted for inclusion in CSE Journal Articles by an authorized administrator of DigitalCommons@University of Nebraska - Lincoln.

Generating Tests for Delay Faults in Nonscan Circuits

PRATHIMA AGRAWAL
VISHWANI D. AGRAWAL
AT&T Bell Laboratories
SHARAD C. SETH
University of Nebraska
at Lincoln

This new method allows any sequential-circuit test generation program to produce path delay tests for nonscan circuits. To test a given path, the authors augment the netlist model of the circuit with a logic block in which testing for a certain single stuck-at fault is equivalent to testing for a path delay fault. The test sequence for the stuck-at fault performs all the necessary delay fault test functions: initialization, path activation, and fault propagation. The authors present results on benchmarks for nonscan and scan/hold modes of testing.

MOST WORK REPORTED on delay testing is applicable only to the scan type of circuits. This restricted problem is, of course, more tractable than delay testing of general sequential circuits. But a large number of VLSI circuits are still designed without scan, and they must be tested for delay faults by means of functional test vectors supplied by the designer. The quality and completeness of such vectors remain questionable. Although researchers have reported some results on delay testing for nonscan circuits,[1] work on test generation algorithms is needed. In this article, we propose a delay test method for synchronous circuits without scan.

Even a properly designed circuit can have timing problems due to physical faults or variations in process parameters. In the literature, researchers have used two models to account for such effects: the gate delay model and the path delay model. The gate delay model holds that a delay fault exists when a gate's delay exceeds its nominal value by more than a preset threshold value. Gate delay faults require delay simulation, unless one assumes that a faulty gate has only large deviations from the nominal value (the gross-delay model).

The path delay model holds that any circuit path (between clocked storage elements) with a total delay exceeding the clock interval is faulty. For our method, we use a path delay model similar to one given by Malaiya and Narayanaswamy.[2] A path can originate at a primary input or a flip-flop and terminate at either a primary output or a flip-flop. A delay fault will cause the propagation time of a signal (either a rising or a falling transition) through the path to exceed the clock period. In general, a test for a path delay fault has three phases: initialization, path activation, and propagation. Each phase contains one or more input vectors.

The activation phase consists of two vectors that initiate a signal transition at the path's origin and propagate it to the path's end. Concepts of path testing in combinational and scan-based circuits are applicable to this phase. Two schools of thought exist, based on the way the path is activated. In the fully transitional path (FTP) approach,[3] both vectors sensitize the entire path at each gate.

In the alternative approach, which we use in our method, only the second vector fully sensitizes the path. The first vector, however, sensitizes the path through gates that must propagate

An earlier version of this article was presented at the Fifth International Conference on VLSI Design, Bangalore, India, January 1992.

0740-7475/93/0300-0020$03.00 © 1993 IEEE
doi: 10.1109/54.199801

noncontrolling values (such as 0 through an OR gate) along the path. The advantage of this approach is that it sets fewer signals than the FTP approach, and, as a result, some paths with no FTP tests can also be activated. Smith[4] and Lin and Reddy[5] specify the required signal states. Test robustness—that is, test validity for arbitrary delays—requires that certain signals remain unchanged during the two vectors of the path activation phase. No static hazard should be produced on these signals. *Static hazard* refers to a glitch, or pulse, in a steady signal. The position and width of this pulse may depend on the actual delays in the circuit. We refer to the robustness of path activation by the two activation vectors as *combinational robustness*.[6]

The initialization phase, which precedes path activation, sets the flip-flops suitably for the other phases. The final phase, propagation, sensitizes a path from the tested path's destination flip-flop to a primary output. One can make the initialization and propagation phases independent of circuit delays by slowing down the system clock. Chatterjee and d'Abreu[7] have recently reported on their use of a slow clock in delay testing, a technique Malaiya and Narayanaswamy[2] suggested earlier. The path activation phase, which involves two vectors, applies the first vector with a slow clock. The second vector, however, must use the rated clock. Consequently, even flip-flops other than the destination flip-flop can have faulty values at the end of the activation phase if several faulty paths are simultaneously activated. To make the test sequentially robust, one must impose added restrictions on flip-flop states.[6]

Delay fault test generation methods for nonscan sequential circuits have been presented by Chakraborty, Agrawal, and Bushnell,[6] and, with the FTP approach, by Devadas.[8]

Delay faults in a nonscan circuit

In a synchronous, sequential circuit, the combinational logic boundaries consist of primary inputs, primary outputs, and flip-flops. A common clock signal of a given frequency (or period) synchronizes all flip-flops. Proper operation requires that any signal changes at the combinational logic inputs that will propagate to the outputs must do so within the clock period. Our fault model consists of single paths between the inputs and outputs of the combinational logic. Of course, only sensitizable paths can be tested for delay faults.

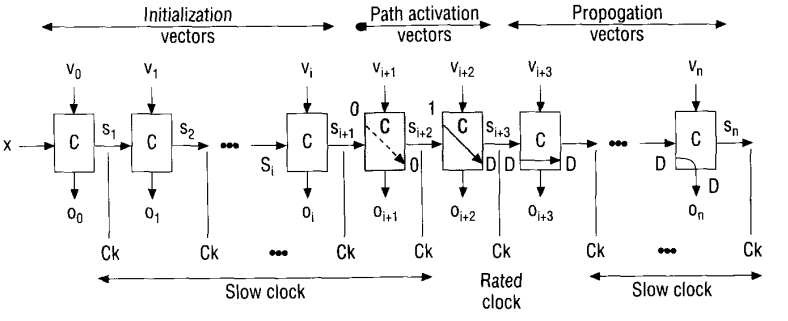


Figure 1. *Testing of a path delay fault in a nonscan circuit.*

For each path, we have two potential faults—one based on the time that a rising transition takes to propagate through the path and the other on that of a falling transition. In general, a path originating and ending at flip-flops requires additional steps (and greater effort) in test generation than one that starts at a primary input or terminates at a primary output. Hence, without loss of generality, we will restrict the discussion to paths between flip-flops.

In this path delay fault model, testing requires an initialization of the circuit to a state in which a transition can be propagated through the path and the resulting transition captured in the destination flip-flop. If the delay fault is present in the circuit—that is, if the path delay exceeds the clock period—the state of the destination latch will be incorrect. The test then makes this state observable at a primary output.

Figure 1 illustrates the major phases and assumptions of the testing procedure. An array of duplicated combinational blocks (C) represents the circuit's operation on application of the test sequence (v_0 through v_n). The primary inputs are applied from the top, and the primary outputs appear beneath the combinational logic blocks. Present-state inputs are shown entering from the left, and next-state signals appear on the right. (Figure 1 does not show flip-flops on the state lines.) We apply the clock (Ck) that transfers the next-state signals to the present-state input of the next block just before we apply the primary inputs. Thus, applying the state inputs s_i and the primary inputs v_i to C produces the primary outputs o_i and the next-state signals s_{i+1}. By activating Ck, we latch the signals s_{i+1} into the flip-flops.

Initialization sequence. The vectors v_0 through v_i form the initialization sequence. The state inputs prior to the application of v_0 are *don't care* (x). Some circuits have a *clear* input, which sets all flip-flops into the 0 state. In such cases, v_0 may apply the *clear* signal. For a circuit without a *clear* signal, $v_0, v_1, \ldots, v_i$ will bring the circuit to a known state. At the end of the initialization sequence, all flip-flops are set in states required by the path activation vectors. To ensure that the circuit initializes irrespective of

Table 1. Signal values for path activation.

Name of signal value	v_{i+1}	v_{i+2}	Hazard requirement
U0	x	0	None
U1	x	1	None
S0	0	0	No static hazard
S1	1	1	No static hazard
R	0	1	No dynamic hazard
F	1	0	No dynamic hazard
XX	x	x	None

any delay faults, we run the clock at a slower rate.

Path activation vectors. The two consecutive inputs to the combinational logic, (v_{i+1}, s_{i+1}) and (v_{i+2}, s_{i+2}), set specific input states for the gates on the path under test. These vectors are the path activation vectors. We specify the combination of signal values during the application of these two vectors as shown in Table 1. The signals U0 and U1 specify the state only for v_{i+2}. The signals S0 and S1 specify steady value for both vectors without a static hazard. R and F are hazard-free transitions. XX indicates the values for both vectors are *don't care*.

The off-path signals follow those given by Lin and Reddy.[5] The outputs of gates on the path assume rising and falling values. As the transition propagates along the path, it can have dynamic hazards.

The off-path signals feeding gates on the path assume values among U0, U1, S0, and S1. During the second vector, v_{i+2}, the off-path signals must sensitize the path. However, during the first vector, v_{i+1}, the off-path signals remain in the *don't-care* state if the path signal applies the controlling value (0 for AND and NAND, 1 for OR and NOR). These conditions are the same as given by Lin and Reddy, but they differ from those required for FTP testing. In our case, v_{i+2} fully sensitizes the entire path (solid arrow in shaded C block in Figure 1), but unlike FTP testing, v_{i+1} may not sensitize the entire path (dashed arrow in shaded C block).

The application of v_{i+2} precedes the application of the rated clock period to the flip-flops. If the signal arriving at the destination flip-flop is a rising transition, the correct value latched in this flip-flop will be 1. Whenever the path delay exceeds the rated clock period, the latched value will be 0. Thus, we denote the destination flip-flop's state by D, which has the same meaning as in the D-algorithm.[9] That is, D denotes a 1 in the fault-free circuit and a 0 in the faulty circuit. Similarly, for a falling transition arriving at the destination flip-flop, the state is $\bar{D}$.

Certain inputs to gates along the path under test have the values S0 and S1. The signals on these lines must be free from static hazard during path activation. To satisfy this requirement, all steady values (S0 and S1) must be implied by identical signal assignments for both path activation vectors at the combinational logic inputs. We will discuss methods to generate such vectors later.

Propagation vectors. In Figure 1, all vectors except v_{i+2} are applied at a slow clock rate. Thus, we can treat all other time frames as fault-free. The propagation vectors, $v_{i+3}, \ldots$, propagate the destination flip-flop's state to an observable output. In the time frame i+2, however, delays can also affect the states of other flip-flops. Thus, any flip-flops whose states are changed by the application of v_{i+2} can potentially assume values that depend on delays. Even flip-flops whose states do not change can have incorrect values due to delays and static hazards. An ideal test, therefore, requires propagation vectors to observe the D or $\bar{D}$ state of the destination flip-flop when all other flip-flops are uninitialized (set to x).[6]

Tests so derived will work irrespective of any delay or timing situations that occur in the circuit. This is a pessimistic approach that we can easily implement. However, we make a simplifying, single-delay-fault assumption: While testing a path, we assume all other paths have significantly smaller delays, and we assume all flip-flops other than the destination are in correct states at the start of the propagation phase. This assumption makes the tests sequentially nonrobust.[6]

Our single-delay-fault assumption is similar in spirit to the single-stuck-at-fault assumption. One may argue that due to the correlation of delays along shared paths, our assumption is not justified. For example, the paths terminating at two flip-flops may contain several common gates with large delays, in which case both flip-flops can have incorrect states. However, similar conditions can and do exist among stuck-at faults in fabricated chips, and the viability of the single-fault model must rest on the unlikelihood of multiple-fault masking.

We can pose the problem of delay fault masking as follows: When we generate the propagation sequence to differentiate between true and faulty states at unit Hamming distance, what are the chances that this sequence will fail to differentiate faulty states at a greater Hamming distance from the true state? Moreover, one should remember that the purpose of the test is detection, not diagnosis.

Further analysis has shown that this optimistic assumption may, in fact, be quite realistic in certain multipath activation situations.[10]

Path search algorithm

We generate paths from sources to destinations one at a time, using a depth-first search algorithm.[11] The source of a path can be either a primary input or a flip-flop output; its destination can be either a primary output or a flip-flop input. The path generation algorithm searches consecutively for all paths from one source before moving on to the next source. To generate a new path, the algorithm first tries an extension of the last generated path. If such an extension is possible, the search continues until reaching a destination. If no extension is possible, the algorithm backtracks to the last unused fan-out and resumes the search from there. Should the backtrack reach a source and not find an unused fan-out, the search for paths from the next source starts.

By preprocessing the circuit structure and storing appropriate information at each node, we can adapt the basic algorithm to generate paths in decreasing order of path lengths (or delays). We can also adapt it to generate a sample of paths by randomizing the next choice in the forward search procedure.

In our implementation of the algorithm, we used a preprocessing step to count the number of paths from any gate to all destinations. With this information, the algorithm sorts the fan-out list of each gate output in decreasing path count order. We obtained excellent results for the sequential-circuit benchmarks, on a Silicon Graphics (SGI 340) computer.[11] For example, it took 322 CPU seconds to generate 244,854 paths in circuit s9234.

Test generation model

Given a path and a transition, we create a modified circuit in which a test for a specified single stuck-at fault will detect the delay fault. We define this stuck-at fault as follows:

1. The stuck-at fault must be activated only when the two path activation vectors have been applied to the combinational logic. Since the circuit is sequential, initialization vectors, if necessary, precede the path activation vectors.
2. Once the stuck-at fault is activated, its effect in the form of D or $\bar{D}$ is injected into the path's destination flip-flop. This happens at the second path activation vector (v_{i+2} in Figure 1). Prior to its activation, the stuck-at fault must not interfere with the circuit's normal operation.
3. After the fault effect is stored in the destination flip-flop, the stuck-at fault must allow the fault-free function of the circuit during the propagation phase.

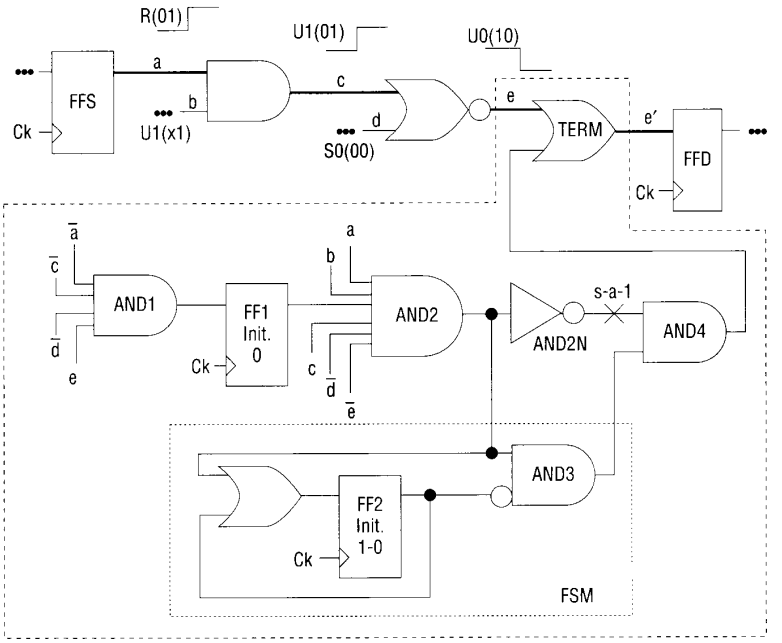


Figure 2. *Test generation model for a falling transition at destination.*

Figure 2 shows how we implement the above functions. To test the path *ace* (shown in bold lines), which lies between two flip-flops, FFS and FFD, we insert a modeling block as shown within the dashed lines. The signal requirements for gates along the path during the activation vectors v_{i+1} and v_{i+2} are captured by the gates AND1 and AND2. The output of AND1 feeds flip-flop FF1, whose output feeds AND2. The output of AND2 is inverted to generate signal AND2N, on which we introduce a stuck-at-1 fault. The flip-flop FF1 is initialized to 0 state and is controlled by the circuit's system clock, Ck. Clearly, the fault AND2N s-a-1 will be activated only when the path is activated by two consecutive vectors.

At that time the state of the signal AND2N will be $\bar{D}$. The AND gate AND4 and the OR gate TERM insert this $\bar{D}$ state into the destination flip-flop FFD under the control of a finite-state machine (FSM, within dotted lines in Figure 2). The type of gate we use for TERM depends on the type of transition arriving at the destination *e*. The gate should sensitize the path between AND4 and FFD during the second path activation vector v_{i+2}. Since *e* has a falling transition, we

use an OR gate for TERM.

During initialization, the path can be activated one or more times. Because of the slow clock, however, the fault effect must not enter the destination flip-flop FFD. Holding flip-flop FF2 in a 1 state during initialization ensures this. At the end of the initialization phase, the test generator clears FF2 to a 0 state. This procedure guarantees that the AND4 gate's output, as forced by the FSM, remains 0 before and after fault activation to provide continuity through the gate TERM. The procedure also ensures that the circuit function remains normal.

In the path activation phase, the FSM produces a 1 output to inject a $\bar{D}$ into FFD only when the path is activated and AND2 turns to 1. Subsequently, the FSM settles into a 1 state with a 0 output and remains in that state throughout the propagation phase. Figure 3 presents the FSM state diagram. As shown in Figure 2, the FSM is implemented with the single flip-flop FF2, which is clocked by Ck.

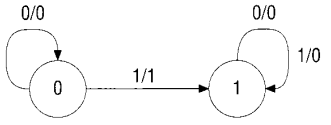


Figure 3. *FSM state diagram.*

When a rising transition arrives at the destination flip-flop FFD, we insert the AND gate TERM in the path to inject a D in the flip-flop. As explained earlier, TERM must sensitize the path for the fault effect to enter FFD during the second path activation vector v_{i+2}. This construction is shown in Figure 4.

Hazard elimination for robust tests. We can incorporate the test generation model just described into any sequential-circuit test generator program for stuck-at faults by modifying the netlist according to the path under consideration. However, robust test generation may require special effort.

The presence of hazards on steady signals S0 and S1 can invalidate a delay test. In general, test generator programs do not attempt to relate the signal values that occur in two different time frames. The vectors are not guaranteed to produce hazard-free signals. Therefore, S0 and S1 signals require special consideration.

Notice that the steady-signal requirement must be imposed only on the two path activation vectors v_{i+1} and v_{i+2}. We feed all steady signals to an AND gate to generate a steady-signal function, SS. We feed the S1 signals directly and the S0 signals in their complemented form to the AND gate. The test generator must then ensure that 1) both v_{i+1} and v_{i+2} are contained in the same prime-implicant cube of the function SS, and 2) no static hazard is produced by the application of these two vectors at the signal SS. Requirement 1 appears to be enough for hazard-free tests, but the following example shows that is not true.

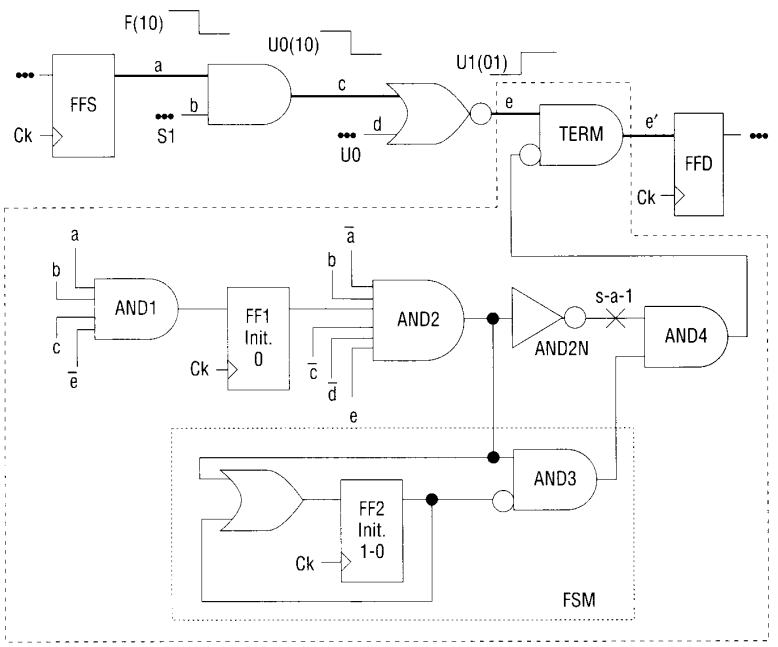


Figure 4. *Test generation model for a rising transition at destination.*

Example. Suppose the path shown in bold in Figure 5 is to be activated for a falling transition at its input. The figure shows the AND1, AND2, and AND gates for the example path. The ON set of a signal is the set of all primary input vectors that set the signal to 1. Similarly, the vectors in the OFF set imply a 0 on the signal. One can easily verify that the only cube of SS that has non-null intersections with the ON sets of both AND1 and AND2 gates is x=1, y=1, and b=1. However, that does not provide a robust test for the fault considered because signal D (and hence SS) can have a static hazard due to the transition on input a.

The example points out the need to restrict the candidate cubes of SS to those that do not produce any static hazard at SS. A practical procedure would be to first generate the ON set of SS by means of a Podem-like line justification procedure.[12] By definition, the ON set contains all the cubes (with 0, 1, or X specified in each bit position) that set SS to 1. If 1's and 0's in a cube are identified

with S1's and S0's, respectively, the resulting value at the line SS will be S1. Thus, we get a potential path activation vector pair (v_{i+1}, v_{i+2}), which we could supply to the sequential test generator. If the test generator does not succeed in augmenting this vector pair with the necessary initialization and propagation vectors, it backtracks to generate the next cube in the ON set of SS.

The procedure just outlined does not represent a complete robust-test generation algorithm because a cube generated by the line justification procedure may be overspecified, imposing unnecessary constraints on the initialization and propagation phases. To overcome this problem, we can use a simulation-based approach to generate all prime cubes that contain the cube generated by line justification. The maximal-compaction method suggested by Pomeranz and Reddy[13] is adaptable for this purpose.

To adapt the test generation procedure to the regular-clock testing mode, we do not use a slow clock, and we can activate the path under test several times during the test sequence. We model this situation by eliminating the FSM from the circuits of Figures 2 and 4. Thus, the output of AND2N, with an s-a-1 fault, feeds directly into an input of TERM.

Implementation and results

The implementation of our path delay test generation method combines the path search algorithm and circuit modification discussed in the preceding two sections and a sequential-circuit test generator. We used Steed,[12] a test generation algorithm that generates ON sets and OFF sets for each combinational logic output. It then generates a test vector for the given single stuck-at fault, using a combinational test generator. Starting in the given initial state, Steed generates an initialization sequence to precede the test vector. If the test vector has produced the fault effect at the destination flip-flop of the path under test, Steed adds a propagation sequence to propagate the fault effect to a primary output. Although we could use any sequential-circuit test generator, Steed appears to be ideal because it assumes the circuit to be fault-free in the initialization and propagation phases. This assumption is valid in our case due to the slowing of the clock.

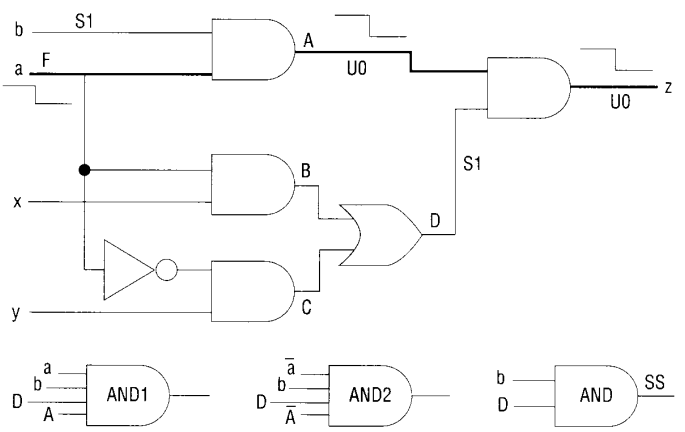


Figure 5. *Circuit in robust-test example.*

We developed a prototype system that generates delay tests for two faults (with rising and falling transitions, respectively) on each path. The system consists of three programs: a path generator, a stuck-at-fault model builder, and the Steed test generator. The path generator places all paths in the form of signal arrays in a file. The model builder reads a path, builds two models for the rising and falling transition faults, specifies the single stuck-at fault for each transition, and calls Steed twice. We use a modified version of Steed, which generates tests only for specified faults. We run Steed without fault simulation because, as explained earlier, the fault-free time frame assumption is valid for delay faults. The test generator produces vector sequences for any detectable faults. Then, the model builder prepares models for the next path in the file and calls the test generator. The three programs run in a Unix shell environment. We did not implement the robustness condition in the prototype system.

Table 2 (next page) lists results obtained with the prototype system on most of the sequential-circuit benchmarks. The prototype effectively generates delay tests, but its implementation is not particularly efficient. Although all the system programs use somewhat similar data structures, they repeatedly access data from the disk. We have not included disk I/O time in the measured run times given in Table 2. However, our shell-based implementation degrades response time drastically for large circuits. Another difficulty is the test generator's slow response due to page faults, especially for large circuits.

The total number of faults is twice that of the total number of paths. Our prototype assumed circuits to have a global *clear* input to initially set all flip-flops into the 0 state. This initial state was used by the test generator. The reported CPU times, on the Sun Sparc2 workstation, do not include path generation time, which was relatively small. Also not included is the time required by Steed's ON set and OFF set generation, which our prototype repeated for each fault. In two circuits, s1196 and s1238, which contained larger numbers of paths, we

Table 2. Path delay test generation in nonscan mode.

Circuit name	No. of faults	Coverage (%)	Test Gen. CPU seconds Total	Test Gen. CPU seconds Per fault	Av. vectors per fault
s27	56	37.5	2.2	0.039	3
s208	290	16.6	18.8	0.065	8
s298	462	20.8	150.4	0.326	21
s344	710	25.9	1,155.9	1.628	5
s349	730	25.2	1,168.1	1.600	5
s382	800	1.3	34,962.6	43.703	33
s400	896	1.1	42,278.0	47.185	33
s420	738	6.4	334.0	0.453	9
s444	1,070	5.6	14,209.6	13.280	55
s510	738	23.4	113.6	0.154	24
s526	820	3.8	42,521.2	51.855	62
s526n	816	3.8	42,564.4	52.162	62
s820	984	37.4	2,952.0	2.917	9
s832	1,012	36.4	3,162.6	3.125	9
s953	2,266	40.2	1,287.7	0.568	11
s1196	4,000*	51.0	48,355.3	12.090	3
s1238	4,000*	41.5	28,050.5	7.013	3
s1488	1,924	37.5	901.3	0.469	15
s1494	1,952	37.0	928.8	0.476	15

*Partial set of faults

analyzed only 4,000 faults each. Although not always so, the test generation time generally increases with the number of vectors. However, the average number of vectors per fault is not a function of circuit size but may depend on the circuit's sequential complexity.

Our prototype implementation seriously limited the capability to process very large circuits. For example, for the circuit s9234, which has very few sensitizable paths, Steed required nearly 41 seconds per fault. The first 1,500 path delay faults in the list produced no test. For s5378, one 13-gate path was tested by just two vectors for the rising transition and by three vectors for the falling transition. However, the test generation time was 150 seconds per fault. We processed a very small fraction of paths for these two circuits (thus, they are not listed in Table 2).

In Table 2, coverage ranges from a low of 1.1% in s400 to a high of 51.0% in s1196. As mentioned earlier, our prototype does not include hazard elimination for robust tests. For some faults, hazard-free tests may not exist. Thus, the coverage of robust tests will be even lower. A low path coverage, however, may not be a serious concern if we use delay tests with stuck-at fault tests. This is because our delay tests can cover all *sensitizable* paths, resulting in better testing in situations requiring speed sorting of VLSI devices.

Scan circuit testing. We can easily apply our test generation method to circuits with scan design. There are two methods of delay testing of scan hardware. Since all flip-flops are controllable and observable through the scan register, we can test a path delay fault with just two vectors, v_1 and v_2. However, we also must specify the corresponding flip-flop states s_1 and s_2 for each vector.

The first method uses the normal scan register for scan-in and scan-out of flip-flop states.[14] In this case, s_2 must be generated either by the combinational circuit when v_1 is applied with the flip-flops initialized to state s_1, or by a single-bit shift of the scan register. To generate tests, we first implement the scan register. The test generation model circuit will contain only the gates AND1, AND2, and the flip-flop FF1, initialized to 0. We make the output of AND2 a new primary output on which a stuck-at-0 fault is tested.

The test generator then uses the scan register to set the circuit to any desired state s_1. It also controls the *mode control* input. Thus, it generates the vectors with either the functional mode or the scan mode, whichever is convenient. No propagation sequence is generated because the stuck-at fault is now on a primary output. We can obtain robust tests by imposing the steady-signal condition discussed earlier.

In the second method, a modified scan design includes a hold latch between each flip-flop and the combinational logic.[2] The hold latch operates in two modes: transparent and hold. All hold latches are controlled by a common *hold clock* primary input, which keeps them in the transparent mode during normal operation. Again, a path delay test consists of two vectors, v_1 and v_2, and their corresponding states, s_1 and s_2. First we scan in s_1 and load it into the hold latches by changing the hold clock from transparent to hold mode. Then we apply v_1 to the primary inputs. Next we scan s_2 into the scan shift register. We apply v_2 to the primary inputs while the hold clock changes all hold latches to transparent mode. We then set the circuit in normal mode and, one clock period later, apply Ck to capture the fault effect in the destination flip-flop. A scan-out follows.

To generate delay tests, we make all flip-flop outputs (present states) primary inputs and all flip-flop inputs (next states) primary outputs. The modeled circuit again contains AND1, AND2, and

FF1 initialized to 0. We also make the output of AND2 a primary output, on which we introduce a stuck-at-0 fault. For robust tests, the steady-signal condition must be imposed.

Table 3 summarizes test generation results obtained for the scan/hold method. As we expected, the coverages are higher than those for the nonscan circuits given in Table 2. We generated only two vectors for each testable path. CPU time for test generation is also more manageable. However, the two largest circuits again posed problems for the test generator. It analyzed only 1,474 faults in s5378 and 92 faults in s9234. These faults were not randomly selected but were taken from the beginning of the path file. Thus, the coverage shown for these two circuits may not be correct.

For the scan/hold method, we implemented the hazard elimination part of our algorithm. In Table 3, the coverage labeled "Robust" is the percentage of path faults for which robust tests were found. The "Total" coverage additionally includes the paths for which only nonrobust tests were found. The total coverage is comparable to results reported by other researchers.[15]

Table 3 shows that for several circuits we found robust tests for most sensitizable paths, pointing to the usefulness of a robust-test generator that can select the proper test when several possible tests exist. However, this result holds true only for scan/hold circuits. We would expect a lower robust coverage for nonscan and normal scan circuits, which we plan to investigate in the future.

OUR TEST GENERATION METHOD allows any sequential-circuit test generator to produce path delay tests for nonscan circuits. In addition, the method's generality permits easy application to the simpler cases of scan and scan/hold circuits. Our hazard avoidance procedure, when fully implemented, will allow generation of robust tests.

To simplify the problem of path delay test generation, we used a single-faulty-path model. Although the single-fault model is popular in stuck-at-fault testing, its use in delay testing needs further analysis and experience.

A direct application of our technique is path delay fault simulation. For each critical path, we add the modeling block with a stuck-at fault. Then a concurrent fault simulation with any given vector sequence determines the path delay fault coverage. Another application involves currently popular static path analyzers, which generate superfluous data on failing paths whose delay cannot cause an incorrect output. We call these *sequentially false paths*. Application of our test generation method to dynamic timing analysis eliminates such paths from consideration.[16] D&T

Acknowledgments

We thank D. Bhattacharya for the example in Figure 5. Sharad C. Seth acknowledges consulting support from AT&T Bell Laboratories.

References

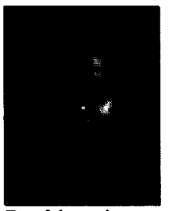
1. I. Deol, C. Mallipeddi, and T. Ramakrishnan, "Amdahl Chip Delay Test System," *Proc. Int'l Conf. Computer Design*, IEEE Computer Society Press, Los Alamitos, Calif., 1991, pp. 200-205.
2. Y.K. Malaiya and R. Narayanaswamy, "Modeling and Testing for Timing Faults in Synchronous Sequential Circuits," *IEEE Design & Test of Computers*, Vol. 1, No. 4, Nov. 1984, pp. 62-74.
3. C.T. Glover and M.R. Mercer, "A Method of Delay Fault Test Generation," *Proc. 25th Design Automation Conf.*, IEEE CS Press, 1988, pp. 90-95.

Table 3. *Path delay test generation in scan/hold mode.*

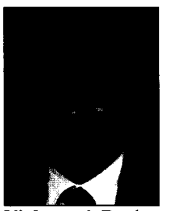
Circuit name	No. of faults	Coverage (%)		Test gen. CPU seconds	
		Robust	Total	Total	Per fault
s27	56	89.3	89.3	0.14	0.0024
s208	290	100.0	100.0	1.15	0.0040
s298	462	74.7	76.2	2.07	0.0045
s344	710	90.6	90.6	4.45	0.0063
s349	730	88.1	88.1	4.35	0.0060
s382	800	88.0	88.0	4.53	0.0057
s400	896	80.7	80.7	5.31	0.0059
s420	738	100.0	100.0	13.87	0.0188
s444	1,070	67.6	67.6	6.15	0.0058
s510	738	99.3	100.0	3.80	0.0050
s526	820	85.5	86.3	4.52	0.0055
s526n	816	85.7	86.5	3.91	0.0048
s820	984	99.3	100.0	4.90	0.0050
s832	1,012	97.7	98.4	5.54	0.0083
s953	2,266	99.6	100.0	14.44	0.0060
s1196	4,000*	56.0	59.9	27.01	0.0068
s1238	4,000*	45.5	45.8	27.21	0.0670
s1488	1,924	99.1	99.6	9.37	0.0049
s1494	1,952	98.2	98.7	4.90	0.0025
s5378	1,474*	39.8	67.8	5,190.10	3.5210
s9234	92*	48.9	69.6	3.04	0.0330

**Partial set of faults*

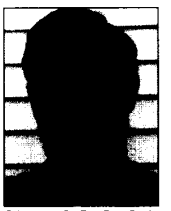
4. G.L. Smith, "Model for Delay Faults Based Upon Paths," *Proc. Int'l Test Conf.*, IEEE CS Press, 1985, pp. 342-349.
5. C.J. Lin and S.M. Reddy, "On Delay Fault Testing in Logic Circuits," *IEEE Trans. CAD*, Vol. CAD-6, No. 9, Sept. 1987, pp. 694-701.
6. T.J. Chakraborty, V.D. Agrawal, and M.L. Bushnell, "Delay Fault Models and Test Generation for Random Logic Sequential Circuits," *Proc. 29th Design Automation Conf.*, IEEE CS Press, 1992, pp. 165-172.
7. A. Chatterjee and M. A. d'Abreu, "Syndrome-Based Functional Delay Fault Location in Linear Digital Data-Flow Graphs," *Proc. Int'l Conf. Computer Design*, IEEE CS Press, 1991, pp. 212-215.
8. S. Devadas, "Delay Test Generation for Synchronous Sequential Circuits," *Proc. Int'l Test Conf.*, IEEE CS Press, 1989, pp. 144-152.
9. J.P. Roth, W.G. Bouricius, and P.R. Schneider, "Programmed Algorithms to Compute Tests and to Detect and Distinguish Between Failures in Logic Circuits," *IEEE Trans. Electronic Computers*, Vol. EC-16, No. 10, Oct. 1967, pp. 567-580.
10. S. Bose, P. Agrawal, and V.D. Agrawal, "A Path Delay Fault Simulator for Sequential Circuits," *Proc. Sixth Int'l Conf. VLSI Design*, IEEE CS Press, 1993, pp. 269-274.
11. P. Agrawal, V.D. Agrawal, and S.C. Seth, "A New Method for Generating Tests for Delay Faults in Non-Scan Circuits," *Proc. Fifth Int'l Conf. VLSI Design*, IEEE CS Press, 1992, pp. 4-11.
12. A. Ghosh, S. Devadas, and A.R. Newton, "Test Generation and Verification for Highly Sequential Circuits," *IEEE Trans. CAD*, Vol. 10, No. 5, May 1991, pp. 652-667.
13. I. Pomeranz and S.M. Reddy, "Compactest: A Method to Generate Compact Test Sets for Combinational Circuits," *Proc. Int'l Test Conf.*, IEEE CS Press, 1991, pp. 194-203.
14. V.D. Agrawal, S.K. Jain, and D.M. Singer, "Automation in Design for Testability," *Proc. Custom Integrated Circuits Conf.*, IEEE, Piscataway, N.J., 1984, pp. 159-163.
15. K.T. Cheng, S. Devadas, and K. Keutzer, "Robust Delay Fault Test Generation and Synthesis for Testability Under a Standard Scan Design Methodology," *Proc. 28th Design Automation Conf.*, IEEE CS Press, 1991, pp. 80-86.
16. P. Agrawal, V.D. Agrawal, and S.C. Seth, "DynaTAPP: Dynamic Timing Analysis With Partial Path Activation in Sequential Circuits," *Proc. EURO-DAC*, IEEE CS Press, 1992, pp. 138-141.



Prathima Agrawal heads the Networked Computing Research Department at AT&T Bell Laboratories. Her interests include computer architecture, parallel algorithms, and VLSI simulation and testing. She received her PhD from the University of Southern California. She is a fellow of the IEEE and the Institution of Electronics and Telecommunication Engineers (India). She is a member of the IEEE Computer Society.



Vishwani D. Agrawal is a distinguished member of the technical staff at AT&T Bell Laboratories and a visiting professor of electrical and computer engineering at Rutgers University. His interests include VLSI testing and neural network algorithms. He has coauthored three books. A former editor-in-chief of *IEEE Design & Test of Computers*, Agrawal presently serves as editor-in-chief of the *Journal of Electronic Testing: Theory and Applications*. He received his PhD from the University of Illinois at Urbana-Champaign. He is a fellow of the IEEE and a member of the IEEE Computer Society, the ACM, and the VLSI Society of India. He has served on the IEEE CS Board of Governors.



Sharad C. Seth is a professor of computer science and engineering at the University of Nebraska at Lincoln. He has held visiting positions at the Indian Institute of Technology, Kanpur, India, and at AT&T Bell Laboratories. His research interests include digital testing and parallel algorithms for computer-aided design. He coauthored *Test Generation for VLSI Chips* (IEEE Computer Society Press). He has served on the editorial boards of *IEEE Transactions on Computer Aided Design*, *IEEE Design & Test of Computers*, and the *Journal of Electronic Testing: Theory and Applications*. Seth received his PhD from the University of Illinois at Urbana-Champaign. He is a senior member of the IEEE, a member of the IEEE Computer Society, and a member of the ACM.

Send correspondence about this article to Vishwani D. Agrawal, AT&T Bell Laboratories, 600 Mountain Ave., Room 2C-476, Murray Hill, NJ 07974; e-mail: va@research.att.com.