# Efficient Path Delay Testing Using Scan Justification

Kyung-Hoi Huh, Yong-Seok Kang, and Sungho Kang

**Delay testing has become an area of focus in the field of digital circuits as the speed and density of circuits have greatly improved. This paper proposes a new scan flip-flop and test algorithm to overcome some of the problems in delay testing. In the proposed test algorithm, the second test pattern is generated by scan justification, and the first test pattern is processed by functional justification. In the conventional functional justification, it is hard to generate the proper second test pattern because it uses a combinational circuit for the pattern. The proposed scan justification has the advantage of easily generating the second test pattern by direct justification from the scan. To implement our scheme, we devised a new scan in which the slave latch is bypassed by an additional latch to allow the slave to hold its state while a new pattern is scanned in. Experimental results on ISCAS'89 benchmark circuits show that the number of testable paths can be increased by about 45% over the conventional functional justification.**

Manuscript received April 26, 2002; revised Feb. 6, 2003.

Kyung-Hoi Huh (Phone: +82 31 680 0253, e-mail: khhuh@dopey.yonsei.ac.kr) is with the ILJIN Technology Center, Pyeongtaek, Gyeonggi-do, Korea.

Yong-Seok Kang (e-mail: einfluke@LG-Elite.com) is with the LG Electronics, Seoul, Korea.

Sungho Kang (e-mail: shkang@yonsei.ac.kr) is with Yonsei University, Seoul, Korea.

## I. Introduction

As the complexity and speed of digital circuits increases, the importance of testing circuits more efficiently also increases. Moreover, as the quality of chips increases, it becomes more important to check the functionality of the circuits as well as the signal transition delays. The purpose of testing delay faults [1] is to make sure that manufactured circuits operate correctly at a functional clock rate. Path delay faults commonly used in delay testing model defects as cumulative propagation delays along circuit paths that exceed a system clock limit [2], [3]. However, testing every path in a circuit is usually impractical because the number of paths may increase exponentially with the number of gates in the circuit. Therefore, it is important to select the path for delay fault tests, and this subject has been extensively researched [4]-[7].

A test for a path delay fault consists of a vector pair that launches a transition at the start of a path and sets up appropriate sensitizing conditions along the path. The time it takes for the transition to reach the destination of the path is the delay of the path for that transition. Extensive work has been done in path delay fault testing and designs for testability of combinational circuits [3], [8]-[15]. Tests for path delay faults are classified as hazard free robust tests, robust tests, and nonrobust tests according to their constraints [8].

To observe the propagation delay of a path passing through a fault site in a combinational circuit, two phase clocks are applied to the input latches and the output latches [2], [16]-[18]. In the first step, an initialization vector is loaded into the input latches. After the initialization vector has become stable, a transition propagation vector is loaded into the input latches by the input clock transition. It is harder to generate test patterns for delay faults in sequential circuits than in combinational circuits, because the primary output of sequential circuits is controlled not only by the primary input

but also by the initial states, which are difficult to control. Generally, there are two methods for delay fault testing using standard scan flip-flops: one is the scan shifting method [1], [19], and the other is the functional justification method [20]-[22]. With the scan shifting method, it is often impossible to generate test patterns for many testable paths. To avoid the problem of the scan shifting method and achieve more accuracy, the functional justification method is preferred. In the functional justification method, a set of the first test patterns is loaded in scan flip-flops and a set of the second test patterns is determined by the functionality of the circuit, which means that the second test patterns are the feedback values following the first test patterns through the circuit. However, a problem remains: the two-vector pair has to meet the logic values for delay fault testing throughout the whole time frame.

A scan design with enhanced scan flip-flops [9] solved this problem by making all vector pairs applicable. This scan uses two flip-flops to store two-vector pairs each line, so that sequential circuits behave as combinational circuits. Although delay testability is improved, the area of this scan is increased by the additional flip-flop, and the area overhead of the enhanced scan design is too high for this design to be used in practical sequential circuits. Other techniques for efficient delay fault testing include a slow-fast testing method and a method using simple clock control circuits [23]-[29]. As the slow-fast delay testing method, [23] and [24], uses a slow clock at the first time frame, the test application time is also slow. There are also several alternatives to the enhanced scan [25]-[28]. These methods control clocks without increasing path delays in a circuit. They use a simple clock control circuit to produce single bit transitions on state variables and a parity check circuit to observe state variable flip-flops. The area overhead of these methods is comparable to the enhanced scan, but a performance penalty is incurred. Moreover, these clock control methods are complicated and hard to apply to large sequential circuits.

To overcome the difficulties of the conventional delay test methods, we propose a new test algorithm in which the second test pattern is generated by scan justification, and the first test pattern is processed by functional justification. With the conventional functional justification, it is hard to generate the proper second test pattern because it uses a combinational circuit for the pattern, but the proposed scan justification has the advantage of easily generating the second test pattern because of direct justification from the scan. To implement this scheme, we devised a new scan, in which the slave latch is bypassed by an additional latch to allow the slave to hold its state while a new pattern is scanned-in.

## II. Test Method Using Scan Justification

### 1. Test Method

To clarify the concept of the scan justification, we define several terms.

- $P1$ ($P2$): the set of the first (second) time frame test patterns. This consists of $Pp1$ ($Pp2$) and $Ps1$ ($Ps2$)
- $Pp1$ ($Pp2$): the set of primary input patterns of the first (second) time frame
- $Ps1$ ($Ps2$): the set of scan input patterns of the first (second) time frame
- $Rp1$ ($Rp2$): the set of primary output response patterns corresponding to $P1$ ($P2$)
- $Rs1$ ($Rs2$): the set of scan output response patterns for $P1$ ($P2$)

Functional justification controls the scan flip-flops to store the first test patterns that have more X-values than the second test patterns. To overcome this inefficiency, the scan flip-flops can be used to store $Ps2$. On the other hand, $Ps1$ is generated through the function of the combinational logic of a sequential circuit, so this method is defined as a scan justification method. According to this method, the second test patterns have to be stored in scan flip-flops before the first test patterns are ready at the start of the circuit. Because it is impossible to propagate the first test patterns through the same scan paths that are holding the second test patterns, an additional latch is required. In addition, for handling the scan justification method, an effective way of generating $Ps1$ is required. In this paper, we introduce a method that controls the primary inputs and scan flip-flops of the circuit.

Figure 1 shows the scheme of the scan justification method; the gray-filled arrows reveal the essential data flow, and the white-filled arrow can have any logic value. At the beginning, $Ps2$ is loaded in a latch by scan shifting, and then $Ps0$ is also shifted into another latch. To make the first test pattern by functional justification, $Pp0$ is ready on the primary input, and in the meantime, $Ps0$ is also ready on the scan output. During one period of the slow clock, $Pp0$ and $Ps0$ are applied, and then $Ps1$, which is used for the first test pattern, reaches an input latch and is simultaneously transparent to the output of the scan. To initialize the circuit during one more consecutive period of the slow clock, $Pp1$ and $Ps1$ are applied. As a result, the circuit is initialized, and there only remains the work of putting the second test patterns in the inputs of the circuit. Because $Ps2$ is already stored in the latch, we get the final responses, $Rp2$ and $Rs2$, which are the outputs for the second test patterns.

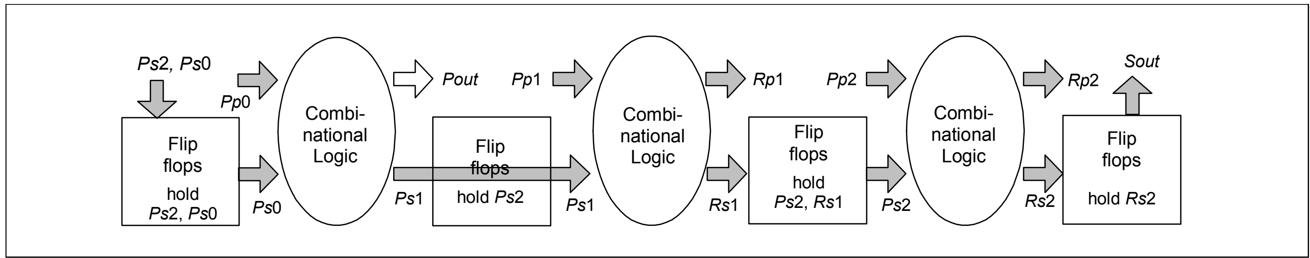


Fig. 1. The scheme of the scan justification method.

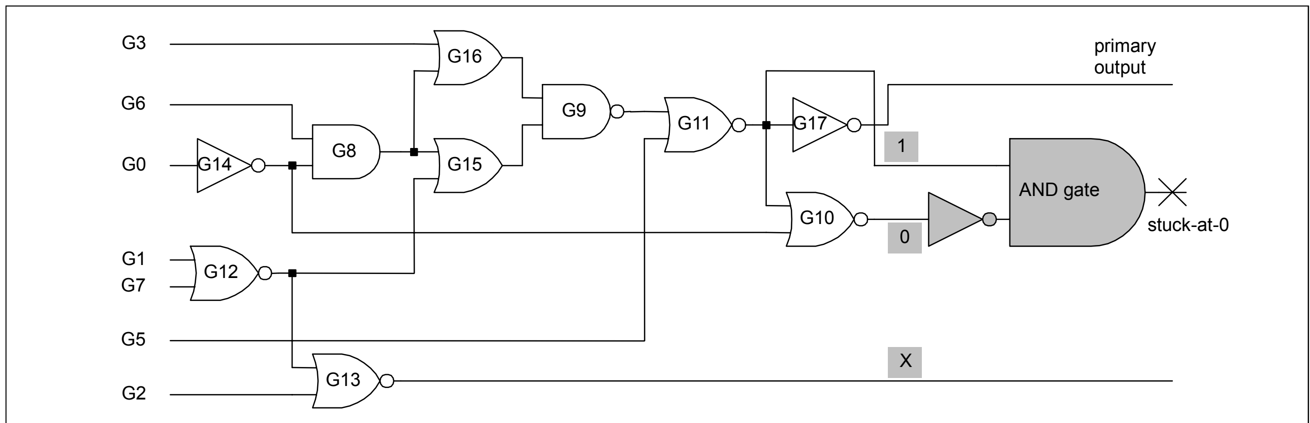


Fig. 2. Modified combinational logic of S27 benchmark circuit.

### 2. Method to Find *Ps*1 Using an ATPG

To apply the scan justification method to large sequential circuits, we propose a method using an automatic test pattern generator (ATPG) to find *Ps*1. An ATPG tool generates the test patterns for a given stuck-at fault. By inserting INVERTER in each feedback output that needs logic-0 in the first test patterns, all the feedback values of the combinational logic block will be modified to logic-1s or X-values. Every output line with logic-1 except the primary outputs is combined with an AND gate, and then a stuck-at-0 fault is inserted in the output of the AND gate. Using the ATPG for this circuit, the *Pp*0 and *Ps*0 can be obtained.

Figure 2 illustrates this method for S27, an ISCAS'89 sequential benchmark circuit. If the required first test patterns are '10X'(G11, G10, G13) for instance, the output of gate G11 is directly connected to the input of an AND gate. Between gate G10 and the AND gate, INVERTER is inserted to set the first test pattern's value to logic-1. However, as it doesn't matter what logic value the output of gate G13 has, gate G13 does not require any modification of the net. In Fig. 2, the gray-filled gates are the ones that are added in the circuit for the given example. After the stuck-at-0 fault is inserted in the output of the AND gate, the ATPG is used, and as a result, *Pp*0 and *Ps*0, which can generate *Ps*1, are obtained.

The fault coverage of this experiment is significant for the performance of the scan justification method because it shows directly how many paths can be detected. If the ATPG tool stops finding patterns for the stuck-at-0 fault, the test of the corresponding path is aborted.

## III. New Scan Flip-Flop Design

To meet the requirements of the test procedure, we had to implement a new scan flip-flop. This scan flip-flop had to be able to pass *Ps*1 to the inputs of the next time frame while holding the second test pattern. Because of this constraint, the hardware overhead would have to be born by scan flip-flops, but if the scan justification and the functional justification method could be used together, the tradeoff between the test time and the fault coverage could be easily decided.

Figure 3 shows the implementation in a CMOS of the proposed scan flip-flop structure that contains three latches (*Latch* 1*, Latch* 2*, Latch* 3) and two multiplexers. The gray-filled areas are additional elements that are not found in the standard scan flip-flop. In physical circuits, INVERTERs are inserted in the line 'Data in,' 'Scan in,' and the output of the scan. With these new elements, the total area overhead for a scan flip-flop is increased by about 50 percent over the

standard scan flip-flop. This increased area overhead could be a serious problem; but if the fault coverage of the delay testing is increased almost to that of the enhanced scan method, this overhead can be quite reasonable. As the area overhead of the enhanced scan flip-flop is about 70 percent of the standard scan flip-flop, this new scan flip-flop can also be accepted. The *test_opt* is the signal that can select which test modes are to be adapted, so this has to be added to external pins.

To store a certain logic value, some kind of latch is needed. The main idea in implementing this scan is to use *Latch* 3 as a bypass for *Latch* 2 (slave). In other words, *Latch* 3 is used to keep the logic value while *Latch* 2 is used for the slave. To control the data flow, several switches are also required for proper operation as shown in Fig. 3. Lastly, the gray-filled multiplexer is implemented to get the value stored in *Latch* 3.

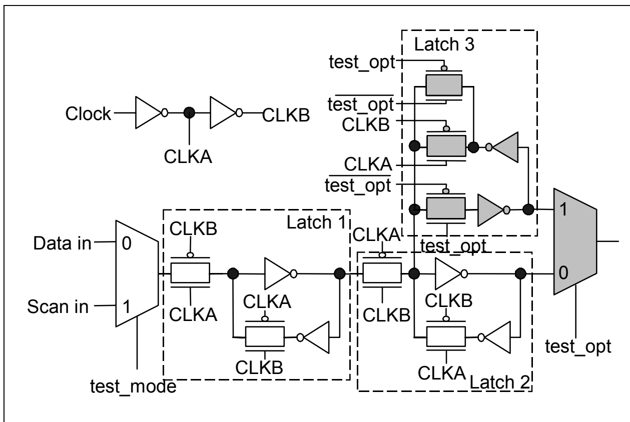


Fig. 3. New scan flip-flop.

In *normal operational mode* (the test control signals, *test_mode* and *test_opt*, are low), this scan flip-flop operates as a normal flip-flop by a system clock. Moreover, there are two scan shifting modes, which are as follows. The first one (*test_mode* is high, but *test_opt* is low) shifts patterns using *Latch* 2 as a slave, while *Latch* 3 is used as a slave in the second one (*test_mode* and *test_opt* are high).

The functional justification method can also be achieved by using this scan flip-flop. When *test_opt* is set at a constant low, this new scan flip-flop operates just like the standard scan flip-flop except there is a small delay time corresponding to the additional multiplexer. In *clocking mode* (*test_mode* is low, but *test_opt* is high), *test_opt* plays the role of the normal clock in the second time frame in the scan justification method. This will be covered in detail in section IV. Table 1 shows four types of modes corresponding to each of the control signals, *test_mode*, *test_opt*.

Table 1. Types of operation modes.

| *test_mode* | *test_opt* | Operation mode |
|---|---|---|
| 0 | 0 | Normal operation mode |
| 0 | 1 | Clocking mode |
| 1 | 0 | L2-scan shifting mode |
| 1 | 1 | L3-scan shifting mode |

## IV. Test Algorithm

For testing delay faults in sequential circuits, we propose one strategy for the functional justification and another for the scan justification method. In the functional justification method, while *test_opt* remains low, the first test patterns are shifted into *Latch* 2 of each scan flip-flop by applying *L2-scan shifting mode*. Then one period of a slow clock is generated on the clock line, and one period of a normal clock is continuously applied to the circuit in *normal operation mode*. As a result, the response patterns are stored in scan flip-flops and all of them are observable through the scan output.

On the other hand, in the scan justification method, *Ps*2 is shifted into *Latch* 3 in *L3-scan shifting mode*, and *Ps*0 is also shifted into *Latch* 2 in *L2-scan shifting mode*. Then *Pp*0 and *Ps*0 are applied by one period of a slow clock in *normal operation mode*. *Ps*1, a set of feed back values for *Pp*0 and *Ps*0, is stored in the scan flip-flops, and *Pp*1 is ready for the primary inputs, so the circuit may be initialized by another slow clock. Because *Ps*2 is being stored in *Latch* 3, *test_opt* must be set high. However, if both *test_opt* and the clock signal are high, *Latch* 3 will not be able to hold *Ps*2 during the last test sequence. Therefore, *test_opt* plays the role of the normal clock while the real clock remains low, so the time when *test_opt* remains high has to be the same as the normal clock period. *Latch* 1 then goes to capture mode so that the response pattern stored in *Latch* 1 will not be affected by an unexpected value. For this reason, the clock must go to high when *test_opt* goes down.

Figure 4 shows the complete test sequence. After the test sequence is completed, *Ps*2 is stored in scan flip-flops, so we can confirm whether the signal transition occurs by observing the scan output in *L2-scan shifting mode*.

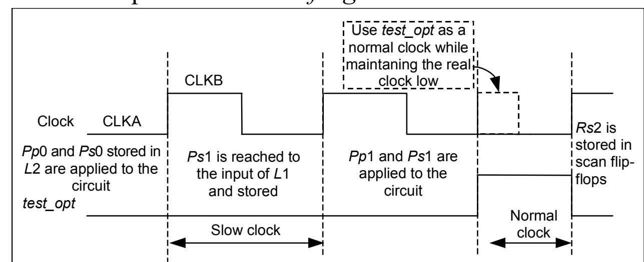


Fig. 4. Timing diagram of testing.

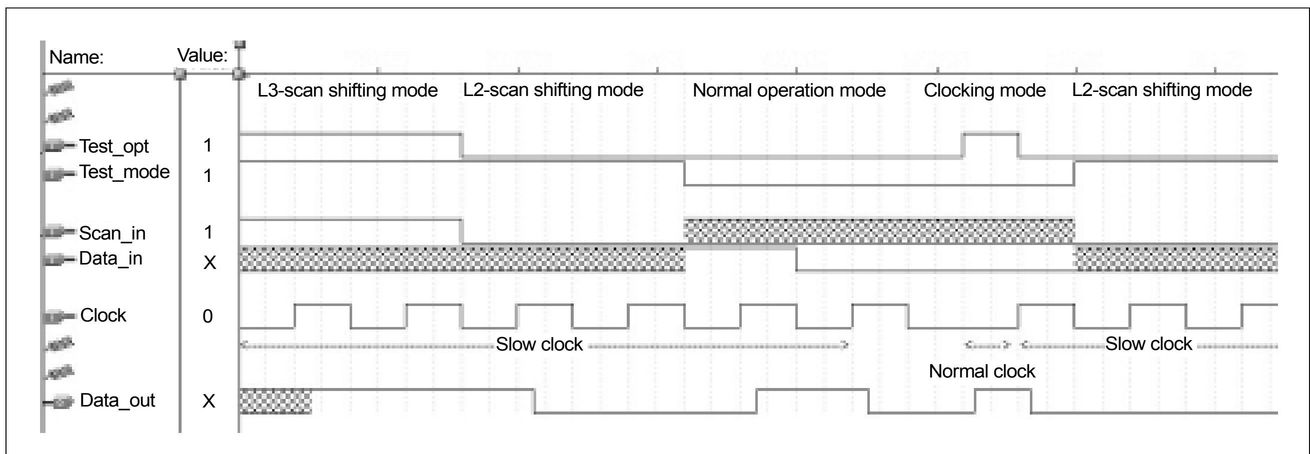


Fig. 5. Example of simulation.

Figure 5 shows an example of the timing simulation of the complete test sequence. In this figure, *clocking mode* is especially significant. In the first sequence, the high logic value is shifted into *Latch* 3 in *L3-scan shifting mode*, and that value is held until the *test_opt* signal goes to high again. In *normal operation mode*, *Data_out* follows the signal *Data_in*, but the output *Data_out* goes to high while *Data_in* still remains low, and then the operation mode changes from *normal operation mode* to *clocking mode*. This is because the high logic value stored in *Latch* 3 is connected to *Data_out* by setting *test_opt* high. Therefore, when the new scan flip-flops are applied to a sequential circuit for delay testing, the second test patterns are reliably generated to the circuit in this way.

To test delay faults efficiently, our proposed new test algorithm uses both the functional justification method and the scan justification method. With the new test algorithm, the testable paths can be increased. This is shown in Fig. 6. At first, to achieve the scan justification, all possible delay tests are generated for the combinational part. Among the obtained first test patterns, which patterns can be generated by the functionality of the circuit is investigated by the scan justification method (explained in section II). During this investigation, *P*0 can also be obtained. By applying three test patterns, *P*0, *P*1, and *P*2, to the circuit, the scan justification sequence is completed.

The functional justification sequence is optional, which means that if the fault coverage of the scan justification does not reach the requirement of the users, this sequence can be added. The functional justification sequence starts with generating the tests using the functional justification method. Among the testable paths of the functional justification method, if there are paths that cannot be tested in the scan justification sequence, they can be tested by applying corresponding test patterns to the circuit. Because this new test algorithm consists of two sequences, it has the advantage of easily determining the tradeoff between the test time and the fault coverage. When engineers want to test circuits fast, they can use only the functional justification method. On the other hand, when they want to test circuits precisely, they can achieve their requirements through all the test sequences.

```
/* ---------- Scan justification ---------- */ {
   Generate delay tests for the combinational logic (P1, P2);
   for (i = 0; i < Num_tests; i++) {
   /* Num_tests: number of delay tests for the combinational logic */
      Modify the combinational logic corresponding to the each
                                   first test patterns(Ps1(i));
      ATPG for the inserted stuck-at-0 fault;
      if (ATPG is succeeded) {
         Memorize the input patterns(a part of P0) and the tested
                                   path (a part of TP);
         Num_success++;
   }}
   for (i = 0; i < Num_success; i++)
      P0, P1, P2 are applied to the circuit;
}
```

(a)

```
/* -------- Functional justification -------- */ {
   Generate delay tests using functional justification;
   Compare TP with the tested paths;
   if (there exist the paths which can be only detected by
                                   functional justification){
      for (i = 0; i < Num_paths; i++) {
      /* Num_paths: the number of paths which can be only detected
                                   by functional justification */
      Apply the test corresponding to the each path (Path(i))
                                   to the circuit;
}}}
```

(b)

Fig. 6. (a) Test algorithm–Scan justification, (b) Test algorithm–Functional justification.

## V. Experimental Results

Tables 2, 3, and 4 show some of the results of the path delay

fault test for the ISCAS'89 sequential benchmark circuits. The two numbers in parentheses indicate testable paths per total paths. For circuits that have less than 5000 paths, all possible paths are considered, and for circuits that have more than 5000 paths, 5000 random paths are considered.

The robust path delay fault test has the advantage that the fault being tested can be detected regardless of the delay defects of other paths. Although the robust path delay fault test is a test of high quality, it has fewer testable paths than the nonrobust path delay fault test. If it is impossible to generate a robust test, a nonrobust test has to be considered. Although the quality of this test set is lower, there are more testable paths. Moreover, the nonrobust test can detect delay defects if all off-path inputs reach their final values prior to the on-path transition. Table 4 shows some of results of this test.

The off-paths of the nonrobust test have more X-values than those of the robust test in the second time frame; therefore, the fault coverage shown in the tables improves from Table 2 to Table 4. These results show that the proposed algorithm can detect more path delay faults than previous scan test methods. For example, if S15850 is designed with the conventional scan flip-flops, 1868 paths can be tested robustly using the functional justification method. On the other hand, 4049 paths can be tested using the new test algorithm. Compared to the enhanced scan method, the delay fault coverage of the new test algorithm is almost the same. Thus, the proposed scan and test algorithm can be efficiently used for high delay fault coverage with a small area overhead.

Table 2. Comparison results for hazard free robust tests.

| Benchmark Circuits | Functional justification (%) (# of testable paths / # of total paths) | Enhanced scan (%) (# of testable paths / # of total paths) | New test algorithm (%) (# of testable paths / # of total paths) |
|---|---|---|---|
| S420 | 5.69 (42 / 738) | 100.00 (738 / 738) | 55.69 (411 / 738) |
| S526 | 15.24 (125 / 820) | 82.93 (680 / 820) | 79.88 (655 / 820) |
| S713 | 56.66 (2833 / 5000) | 73.64 (3682 / 5000) | 73.64 (3682 / 5000) |
| S838 | 3.07 (62 / 2018) | 100.00 (2018 / 2018) | 59.22 (1195 / 2018) |
| S953 | 32.00 (740 / 2312) | 99.13 (2292 / 2312) | 84.13 (1945 / 2312) |
| S1423 | 38.60 (1930 / 5000) | 93.52 (4676 / 5000) | 89.26 (4463 / 5000) |
| S1494 | 25.08 (1254 / 5000) | 99.08 (4954 / 5000) | 89.48 (4474 / 5000) |
| S9234 | 44.02 (2201 / 5000) | 78.76 (3938 / 5000) | 77.58 (3879 / 5000) |
| S13207 | 10.44 (522 / 5000) | 22.84 (1142 / 5000) | 18.04 (902 / 5000) |
| S15850 | 33.86 (1693 / 5000) | 79.76 (3988 / 5000) | 75.98 (3799 / 5000) |

Table 3. Comparison results for robust tests.

| Benchmark Circuits | Functional justification (%) (# of testable paths / # of total paths) | Enhanced scan (%) (# of testable paths / # of total paths) | New test algorithm (%) (# of testable paths / # of total paths) |
|---|---|---|---|
| S420 | 28.18 (208 / 738) | 100.00 (738 / 738) | 80.49 (330 / 738) |
| S526 | 17.20 (141 / 820) | 84.63 (694 / 820) | 84.63 (694 / 820) |
| S713 | 61.32 (3066 / 5000) | 77.78 (3889 / 5000) | 77.78 (3889 / 5000) |
| S838 | 31.47 (635 / 2018) | 100.00 (2018 / 2018) | 81.91 (1653 / 2018) |
| S953 | 41.22 (953 / 2312) | 99.57 (2302 / 2312) | 95.50 (2208 / 2312) |
| S1423 | 41.80 (2090 / 5000) | 95.86 (4793 / 5000) | 94.10 (4705 / 5000) |
| S1494 | 41.12 (2056 / 5000) | 99.52 (4976 / 5000) | 96.84 (4842 / 5000) |
| S9234 | 47.08 (2354 / 5000) | 81.84 (4092 / 5000) | 81.16 (4058 / 5000) |
| S13207 | 12.70 (635 / 5000) | 27.48 (1374 / 5000) | 21.42 (1071 / 5000) |
| S15850 | 37.36 (1868 / 5000) | 82.72 (4136 / 5000) | 80.98 (4049 / 5000) |

Table 4. Comparison results for nonrobust tests.

| Benchmark Circuits | Functional justification (%) (# of testable paths / # of total paths) | Enhanced scan (%) (# of testable paths / # of total paths) | New test algorithm (%) (# of testable paths / # of total paths) |
|---|---|---|---|
| S420 | 47.43 (350 / 738) | 100.00 (738 / 738) | 100.00 (738 / 738) |
| S526 | 36.59 (300 / 820) | 87.80 (720 / 820) | 87.80 (720 / 820) |
| S713 | 83.32 (4166 / 5000) | 86.02 (4301 / 5000) | 86.02 (4301 / 5000) |
| S838 | 56.59 (1142 / 2018) | 100.00 (2018 / 2018) | 100.00 (2018 / 2018) |
| S953 | 66.61 (1540 / 2312) | 100.00 (2312 / 2312) | 100.00 (2312 / 2312) |
| S1423 | 64.00 (3200 / 5000) | 97.10 (4855 / 5000) | 97.14 (4857 / 5000) |
| S1494 | 83.84 (4192 / 5000) | 99.78 (4989 / 5000) | 99.78 (4989 / 5000) |
| S9234 | 64.94 (3247 / 5000) | 91.72 (4586 / 5000) | 91.70 (4585 / 5000) |
| S13207 | 19.54 (977 / 5000) | 38.52 (1926 / 5000) | 38.52 (1926 / 5000) |
| S15850 | 58.30 (2915 / 5000) | 91.48 (4574 / 5000) | 91.48 (4574 / 5000) |

## VI. Conclusion

Delay testing has become an area of focus in the field of digital circuits as the speed and the density of circuits have greatly increased. The fact that the internal inputs and outputs of a combinational logic block are controllable and observable only through a scan path makes delay testing very difficult. In this paper, we proposed a new scan flip-flop and a new test algorithm to overcome the difficulty of delay testing. With the new test algorithm, the second test pattern is performed by scan justification, and the first test pattern is performed by functional justification. In this way, test patterns are easily generated. For applying this idea efficiently, we developed a new scan flip-flop, which has the advantage of being able to use both functional justification and scan justification. To store each pattern respectively in the scan flip-flop, two latches are laid parallel to a multiplexer, so additional delays occur only in the multiplexer, which is an improvement over the standard scan. Experimental results on ISCAS'89 benchmark circuits show that the number of testable paths can be increased by about 45% over the conventional functional justification.

## References


[1] V.S. Iyengar, B.K. Rosen, and I. Spillinger, "Delay Test Generation. II. Algebra and Algorithms," *Proc. of IEEE Int'l Test Conf.*, 1988, pp. 867-876.

[2] G.L. Smith, "Model for Delay Faults Based upon Paths," *Proc. of IEEE Int'l Test Conf.*, 1985, pp. 342-349.

[3] S. Reddy, C. Lin, and S. Patel, "An Automatic Test Pattern Generator for the Detection of Path Delay Faults," *Proc. of IEEE Int'l Conf. on Computer-Aided Design*, 1987, pp. 284-287.

[4] W. Li, S. Reddy, and S. Sahni, "On Path Selection in Combinational Logic Circuits," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 1989, pp. 26-39.

[5] A. Majhi, V. Agrawal, J. Jacob, and L. Patnaik, "Line Coverage of Path Delay Faults," *IEEE Trans. on VLSI*, 2000, pp. 610-613.

[6] A. Murakami, S. Kajihara, T. Sasao, R. Pomeranz, and S. Reddy, "Selection of Potentially Testable Path Delay Faults for Test Generation," *Proc. of IEEE Int'l Test Conf.*, 2000, pp. 376-384.

[7] M. Sharma, J. Patel, "Finding a Small Set of Longest Testable Paths that Cover Every Gate," *Proc. of IEEE Int'l Test Conf.*, 2002, pp. 974-982.

[8] C. Lin, S. Reddy, "On Delay Fault Testing in Logic Circuits," *IEEE Trans. on Computer-Aided Design*, 1987, pp. 694-703.

[9] S. Dasgupta, R. Walther, and T. Williams, "An Enhancement to LSSD and Some Applications of LSSD in Reliability, Availability and Serviceability," *Proc. of IEEE Int'l Symp. on Fault-Tolerant Computing*, 1981, pp. 32-34.

[10] A. Saldanha, R. Brayton, and A. Sangiovanni-Vicentelli, "Equivalence of Robust Delay-fault and Single Stuck-fault Test Generation," *Proc. of IEEE Design Automation Conf.*, 1992, pp. 173-176.

[11] K.-T Cheng, H. Chen, "Delay Testing for Non-robust Untestable Circuits," *Proc. of IEEE Int'l Test Conf.*, 1993, pp. 954-961.

[12] W. Lam, A. Saldanha, R. Brayton, and A. Sangiovanni-Vicentelli, "Delay Fault Coverage and Performance Tradeoffs," *Proc. of IEEE/ACM Design Automation Conf.*, 1993, pp. 446-452.

[13] W. Ke, P. Menon, "Synthesis of Delay-Verifiable Combinational Circuits," *IEEE Trans. on Computers*, 1995, pp. 213-222.

[14] S. Devadas, K. Keutzer, "Synthesis of Robust Delay-Fault-Testable Circuits: Practice," *IEEE Trans. on Computer-Aided Design*, 1992, pp. 277-300.

[15] S. Devadas, K. Keutzer, "Validatable Non-Robust Delay-Fault-Testable Circuits via Logic Synthesis," *IEEE Trans. on Computer-Aided Design*, 1992, pp. 1559-1573.

[16] J.D. Lesser, J.J. Schedletsky, "An Experimental Delay Test Generator for LSI logic," *IEEE Trans. on Computers*, 1980, pp.

235-248.

[17] D. Bhattacharya, P. Agrawal, and V.D. Agrawal, "Test Pattern Generation for Path Delay Faults using Binary Decision Diagrams," *IEEE Trans. on Computers*, 1995, pp. 434-447.

[18] C.A. Cheng, S.K. Gupta, "Test Generation for Path Delay Faults Based on Satistiability," *Proc. of Design Automation Conf.*, 1996.

[19] H. Wittmann, M. Henftling, "Path Delay ATPG for Standard Scan Design," *Proc. of EURO-DAC*, 1995, pp. 202-207.

[20] Sungho Kang, Bill Underwood, Wai-On Law, and Haluk Konuk, "Efficient Path Delay Test Generation for Custom Designs," *ETRI J.*, vol. 23, no. 3, Sept. 2001, pp. 138-149.

[21] S. Kang, W. Law, and B. Underwood, "Path-Delay Fault Simulation for a Standard Scan Design Methodology," *Proc. of Int'l Conf. on Computer Design*, 1994, pp. 235-248.

[22] M.H. Schulz, K. Fuchs, and F. Fink, "Advanced Automatic Test Pattern Generation Techniques for Path Delay Faults," *Proc. of IEEE Int'l Symp. on Fault Tolerant Computing*, 1989, pp. 154-163.

[23] P. Agrawal, V.D. Agrawal, and S.C. Seth, "Generating Tests for Delay Faults in Nonscan Circuits," *IEEE Design and Test of Computers*, 1993, pp. 19-29.

[24] Y.-C. Hsu, S.K. Gupta, "An Automatic Test Pattern Generator for At-Speed Robust Path Delay Testing," *Proc. of IEEE Asian Test Symp.*, 1998, pp. 88-95.

[25] R.C. Tekumalla, P.R. Menon, "Delay Testing with Clock Control: An Alternative to Enhanced Scan," *Proc. of IEEE Int'l Test Conf.*, 1997, pp. 454-462.

[26] T.J. Chakraborty, V.D. Agrawal, and M.L. Bushnell, "On Variable Clock Methods for Path Delay Testing of Sequential Circuits," *IEEE Trans. on Integrated Circuits and Systems*, 1997, pp. 1237-1249.

[27] S. Majumder, V.D. Agrawal, and M.L. Bushnell, "Path Delay Testing: Variable-Clock Versus Rated-Clock," *Proc. of IEEE VLSI design*, 1998, pp. 470-475.

[28] Altaf-Ul-Amin Md., S. Ohtake, H. Fujiwara, "Design for Hierarchical Two-pattern Testability of Data Paths," *Proc. of Asian Test Symp.*, 2001, pp. 11-16.

[29] P. Agrawal, V.D. Agrawal, and S.C. Seth, "DynaTAPP: Dynamic Timing Analysis with Partial Path Activation In Sequential Circuits," *Proc. of EURO-DAC*, 1992, pp. 138-141.

**Kyung-Hoi Huh** received his BS and MS degrees in electrical and computer engineering from Yonsei University in Korea. He was an Assistant Fellow at Yonsei University, and he has been a Research Engineer at the Technology Center of ILJIN Diamond Inc. since 2001. His current research interests include VLSI design, video signal processor, VLSI testing and design for testability.

**Yong-Seok Kang** received his BS, MS, and PhD degrees in electrical and electronic engineering from Yonsei University in Korea. He was a Post Doctoral Fellow at Yonsei University, and he has been a Senior Engineer at LG Electronics in Seoul, Korea, since 2002. His current research interests include VLSI design, VLSI CAD, VLSI testing and design for testability.

**Sungho Kang** received his BS degree from Seoul National University in Korea and his MS and PhD degrees in electrical and computer engineering from the University of Texas at Austin. He was a Post Doctoral Fellow at the University of Texas at Austin, a Research Scientist at Schlumberger Laboratory for Computer Science, Schlumberger, Inc., and a Senior Staff Engineer at Semiconductor Systems Design Technology, Motorola Inc. Since 1994, he has been an Associate Professor of the Department of Electrical and Electronic Engineering at Yonsei University in Korea. His current research interests include VLSI design, VLSI CAD, VLSI testing and design for testability.