

## Tutorial: Single-input-change (SIC) vectors for delay testing

K. Han

**Abstract:** Path-delay fault (PDF's) cause logic circuits to malfunction at the desired clock rate. In this paper, I review a classification of the PDF's and present a simulation-based method using single-input change (SIC) patterns to derive tests for singly-testable (ST) (PDF's). A sixteen-valued algebra is used for rising and falling PDF's from all inputs.

### 1. Introduction

Delay fault testing allows testing for delay faults. A delay fault occurs in a circuit when one or more paths in the circuit fail to propagate a signal within the time interval specified by the clock period. Detection of delay faults requires two-pattern tests. An initialization vector is applied and the circuit is allowed to stabilize. Then, the test vector is applied and the circuit outputs are sampled at clock speed. The response is then compared to that of the fault-free circuit to determine the presence or the absence of a delay fault. Hence, correct operation of a circuit at the intended speed can only be guaranteed if there is no delay fault in the circuit.

With the continuously increase in the operating speed of VLSI circuits, delay fault testing is likely to become industrially accepted in the near future [1]. With delay fault testing (i.e. application of two-pattern tests to ensure temporal correctness of the design), at-speed testing (i.e. test at the intended operating speed of the circuit) is also becoming an essential part of the verification process of today's VLSI circuits since it allows to optimize the test time and provides the means to test for delay faults.

PDF's are classified, according to their testability, as singly-testable (ST) or non-ST [2]. An ST PDF affects the circuit speed, and has at least one test that is either robust [3] or validatable non-robust [4] or non-robust. Robust and validatable non-robust tests *guarantee* detection of the target fault in the presence of other PDFs. A non-robust test guarantees detection in the case of a single PDF, but may become *invalid* when there are multiple PDFs. A non-ST PDF has neither a robust nor a

validatable non-robust nor a non-robust test. These PDF's either do *not* affect the speed, or may affect it only if many non-ST PDF's *simultaneously* exist [2, 5]. Therefore, delay tests with 100 % ST PDF coverage guarantee the speed of a circuit under a single PDF assumption.

The present algorithm allows an extremely efficient simulation of single-input change patterns. The present research begins from two results given in a recent paper. A sixteen-valued algebra is used for both rising and falling PDF's that are concurrently simulated. With this algebra, Boolean operations directly give the input-output relationships for gates. Using machine-word parallelism in all stages of computation, the circuit is simulated by assigning random values to its inputs. Transitions are then propagated from each input while all others remain steady. Finally, detected PDF's are recorded. In one pass, both transitions are simulated from all inputs concurrently.

### 2. Path Delay Fault Model

The path delay fault model was first proposed by Smith [6]. In path delay fault model, any path with a total delay exceeding the system clock interval is said to have a path delay fault. This models distributed defects that affect an entire path. For each physical path  $P$ , connecting a primary input to a primary output of the circuit, there are two corresponding delay paths. The rising path (falling path) is the path traversed by a transition that is initiated as a rising (falling) transition at the input of path  $P$  and changes the direction of transition whenever it passes through an inverting gate. There are definitions that are frequently used in path delay fault testing [3].

**Definition 2.1:** Let  $G$  be a gate on path  $P$  in a logic circuit, and let  $T$  be an input to gate  $G$ ;  $T$  is called an off-path sensitizing input if  $r$  is not on path  $P$ .

**Definition 2.2:** A two-pattern test  $\langle V1, V2 \rangle$  is

called a robust test for a delay fault on path P, if the test detects that fault independently of all other delays in the circuit [7].

**Definition 2.3:** A two-pattern test  $\langle V1, V2 \rangle$  is called a non-robust test for a delay fault on path P if it detects the fault under the assumption that no other path in the circuit involving the off-path inputs of gates on P has a delay fault [2].

Gharaybeh et al. [2] classify path delay faults into three categories: singly-testable (ST), multiply-testable (MT), and singly-testable dependent (ST-dependent).

**Singly-Testable:** A path-delay fault is singly-testable (ST) iff there exists a delay test that guarantees its detection when it is the only path-delay fault in the circuit. This test is either robust or validatable non-robust or non-robust.

**Singly-Testable Dependent:** A path-delay fault is singly-testable dependent (ST dependent) iff it cannot propagate a late transition given the absence of certain ST faults. The ST dependent path delay fault set is a subset of the robust dependent (RD) set [8], but may include some functional sensitizable faults.

**Multiply-Testable:** A path-delay fault is multiply-testable (MT) iff it is neither ST nor ST-dependent. A test is possible only when several paths have simultaneous delay faults.

Figure 1 compares this classification of path-delay faults with others.

| Lam et al (IEEE-TCAD'95)                | non-RD          |                     | RD                      |                           |
|---|-----------------|---------------------|-------------------------|---------------------------|
| Cheng and Chen (IEEE-TCAD'96)           | robust testable | non-robust testable | functional sensitizable | redundant                 |
| Gharaybeh et al. (JETTA'97)             | singly-testable |                     | multiply testable       | singly-testable dependent |
| Sivaraman and Strojwas (VLSI Design'97) | primitive SPDF  |                     | primitive MPDF          | primitive-dependent       |

Figure 1: Path delay fault classification

### 3. Concurrent Simulation

Figures 2 (a) and (b) show simulation of a circuit with the SIC patterns  $\langle 01, 11 \rangle$  and  $\langle 11, 01 \rangle$ , respectively, which differ only in the polarity of the transition at a. The value s1 represents the steady logic "1" value at b. Using Definition of singly-testable, both upward transition a124x and a134x are ST in Figure 2 (a). However, only

downward transition a134x is ST in Figure 2 (b). While conventional methods separately simulate the tests shown in Figures 2 (a) and (b), both transition polarities are simulated polarities concurrently. A sixteen-valued algebra is used with signal values: {n0, f0, r0, b0, no, fo, ro, bo, ne, fe, re, be, n1, f1, r1, b1}. Each value is an ordered pair, denoted by (pds, ts). Here, pds is a subset of {n, f, r, b}, i.e., {none, falling, rising, both}, and ts is a subset of {0, o, e, 1}, i.e., {0, odd, even, 1}.

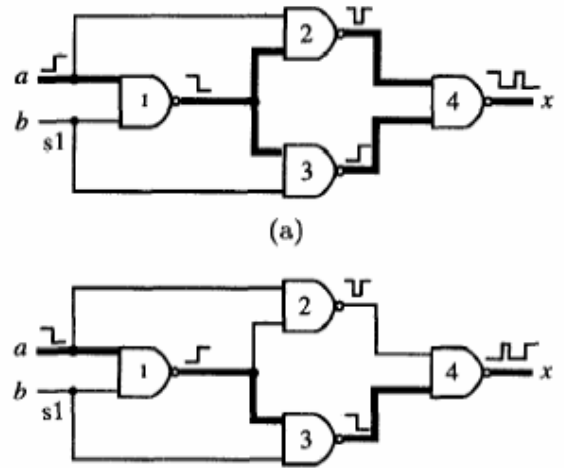


Figure 2: Simulation with (a)  $\langle 01, 11 \rangle$ , (b)  $\langle 11, 01 \rangle$ .

**Definition 3.1 Path detectability status:** A node has pds = n if no sub-path from any input to that node is statically sensitized with either the rising or the falling SIC patterns. If there is static sensitization with the falling SIC pattern only, then pds = f. Conversely, pds = r indicates sensitization with the rising SIC pattern only. If static sensitization is with both the falling and rising SIC patterns, then pds = b.

**Definition 3.2 Transition status:** A node has ts=0 (odd) if the polarity of the transition at the given node, from the initial to the final states, is opposite to that of the SIC pattern. Similarly, ts = e (even) if the transition is of the same polarity. If the node value is "0" regardless of the SIC polarity, then ts = "0". Similarly, ts = "1" if the value is "1".

Table I is the two-input NAND gate propagation table. For example, if the NAND gate inputs are fbe, n1g, then the output is bo. Propagation tables for all

other gates can be derived from Table I by modeling them as NAND gate circuits. Consider the example of Fig. 3. The top values alone show concurrent simulation using the 16-valued algebra for the delay tests of Fig. 2(a) and (b). The 16-valued algebra represents the rising and falling waveforms at a in Fig. 2(a) and (b) with be. Similarly, n1 represents the steady logic “1” value at b. The bottom values in Fig. 3 correspond to rising and falling SIC vector pairs at b, i.e., h10; 11i and h11; 10i. Fig. 3 shows concurrent simulation of four SIC vectors.

TABLE I  
TWO-INPUT NAND GATE PROPAGATION TABLE

| OUT | Input # 2 |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|-----|-----------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|     | n0        | f0 | r0 | b0 | no | fo | ro | bo | ne | fe | re | be | n1 | f1 | r1 | b1 |    |
| n0  | n1        | n1 | n1 | n1 | n1 | n1 | n1 | n1 | n1 | n1 | n1 | n1 | n1 | n1 | n1 | n1 |    |
| f0  | n1        | n1 | n1 | n1 | f1 | f1 | f1 | f1 | n1 | n1 | n1 | n1 | f1 | f1 | f1 | f1 |    |
| r0  | n1        | n1 | n1 | n1 | n1 | n1 | n1 | n1 | r1 | r1 | r1 | r1 | r1 | r1 | r1 | r1 |    |
| b0  | n1        | n1 | n1 | n1 | f1 | f1 | f1 | f1 | r1 | r1 | r1 | r1 | b1 | b1 | b1 | b1 |    |
| I   | no        | n1 | f1 | n1 | f1 | ne | fe | ne | fe | n1 | f1 | n1 | f1 | ne | fe | ne | fe |
| n   | fo        | n1 | f1 | n1 | f1 | fe | fe | fe | n1 | f1 | n1 | f1 | fe | fe | fe | fe | fe |
| p   | ro        | n1 | f1 | n1 | f1 | ne | ne | ne | r1 | b1 | r1 | b1 | re | be | re | be |    |
| u   | bo        | n1 | f1 | n1 | f1 | fe | fe | fe | r1 | b1 | r1 | b1 | be | be | be | be |    |
| t   | ne        | n1 | n1 | r1 | r1 | n1 | n1 | r1 | r1 | no | no | ro | ro | no | no | ro | ro |
| fe  | n1        | n1 | r1 | r1 | f1 | f1 | b1 | b1 | no | no | ro | ro | fo | fo | bo | bo |    |
| #   | re        | n1 | n1 | r1 | r1 | n1 | n1 | r1 | r1 | ro | ro | ro | ro | ro | ro | ro | ro |
| 1   | be        | n1 | n1 | r1 | r1 | f1 | f1 | b1 | b1 | ro | ro | ro | bo | bo | bo | bo |    |
|     | n1        | n1 | f1 | r1 | b1 | ne | fe | re | be | no | fo | ro | bo | n0 | f0 | r0 | b0 |
|     | f1        | n1 | f1 | r1 | b1 | fe | fe | be | be | no | fo | ro | bo | f0 | f0 | b0 | b0 |
|     | r1        | n1 | f1 | r1 | b1 | ne | fe | re | be | ro | bo | ro | bo | r0 | b0 | r0 | b0 |
|     | b1        | n1 | f1 | r1 | b1 | fe | fe | be | be | ro | bo | ro | bo | b0 | b0 | b0 | b0 |

TABLE I  
Two-Input NAND Gate Propagation Table

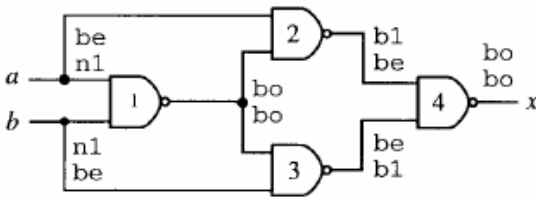


Figure 3: Concurrent simulation for input a and b.

One random vector is generated such that each input value is either n0 or n1. Rising and falling SIC vectors is concurrently simulated by assigning the value be to each input while the other inputs remain at their n0 or n1 values. For primary inputs, primary outputs, and gate outputs, computer memory to store the propagation value for each input change (see Fig. 3) is allocated. These value are then propagated concurrently toward primary

outputs.

The local sensitization of a gate to sub-paths from each of its inputs for rising and falling SIC vectors is denoted by Sr and Sf , respectively. Any of these values is “1” if the two fault-detection conditions are locally satisfied: a) a transition is launched at the gate input under consideration and b) the corresponding side inputs have non-controlling values under v2. A PDF upward P (downward P) is detected if all Sr (Sf ) flags along the path are “1.” To store the detection status of PDF’s, an array of bits —only one bit per PDF are allocated. Initially, all bits are “0.” A bit is changed to “1” if the corresponding PDF is detected. Path-enumeration method is used to calculate the array index corresponding to a given path. The calculation of the array index for each path is done in time linear in the size of the circuit [9]. The number of indexes needed is equal to the number of PDF’s simulated.

TABLE II  
NAND-GATE SENSITIZATION TABLE

| Sr,Sf | Output |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|-------|--------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|       | n0     | f0 | r0 | b0 | no | fo | ro | bo | ne | fe | re | be | n1 | f1 | r1 | b1 |    |
| n0    | xx     | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | 00 | xx | xx | xx |    |
| f0    | xx     | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | 00 | 01 | xx | xx |    |
| r0    | xx     | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | 00 | xx | 10 | xx |    |
| b0    | xx     | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | 00 | 01 | 10 | 11 |    |
| no    | xx     | xx | xx | xx | xx | xx | xx | xx | xx | 00 | 00 | xx | xx | 00 | 00 | xx | xx |
| I     | fo     | xx | xx | xx | xx | xx | xx | xx | xx | 01 | xx | xx | 00 | 00 | xx | xx |    |
| n     | ro     | xx | xx | xx | xx | xx | xx | xx | 00 | 00 | 10 | 10 | 00 | 00 | 10 | 10 |    |
| p     | bo     | xx | xx | xx | xx | xx | xx | xx | 01 | xx | 11 | 00 | 00 | 10 | 10 | 10 |    |
| u     | ne     | xx | xx | xx | xx | 00 | xx | 00 | xx | xx | xx | xx | 00 | xx | 00 | xx |    |
| t     | fe     | xx | xx | xx | xx | 00 | 01 | 00 | 01 | xx | xx | xx | 00 | 01 | 00 | 01 |    |
| #     | re     | xx | xx | xx | xx | xx | xx | 10 | xx | xx | xx | xx | 00 | xx | 00 | xx |    |
| 1     | be     | xx | xx | xx | xx | xx | xx | 10 | 11 | xx | xx | xx | 00 | 01 | 00 | 01 |    |
|       | n1     | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
|       | f1     | xx | 01 | xx | 01 | 00 | 00 | 00 | 00 | xx | 01 | xx | 01 | 00 | 00 | 00 | 00 |
|       | r1     | xx | xx | 10 | 10 | xx | xx | 10 | 10 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 |
|       | b1     | xx | xx | xx | 11 | xx | xx | 10 | 10 | xx | 01 | xx | 01 | 00 | 00 | 00 | 00 |

TABLE II  
NAND Gate Sensitization Table

Table II gives the NAND gate local sensitization (Sr; Sf) for given input and output values. The two situations described above may be examined in Table II. With (input, output) = (be, bo), the sensitization is 11, indicating that the sub-path with be is statically sensitized for both rising and falling SIC vectors. For (n1, bo), however, the sensitization is 00, indicating that the sub-path with n1 is statically sensitized for neither rising nor falling SIC vectors. For some input values, there

are values that the output never attains. Table II indicates this with xx. For example, if one input is n0, then the output is always n1. Local sensitization tables for all other gates can be derived from Table II by modeling them with NAND gates.

| Value | n0   | f0   | r0   | b0   | no   | fo   | ro   | bo   |
|-------|------|------|------|------|------|------|------|------|
| Code  | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 |
| Value | ne   | fe   | re   | be   | n1   | f1   | r1   | b1   |
| Code  | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |

TABLE III  
Coding for The 16-Valued Algebra

Table III shows an encoding for the 16-valued algebra. Four bits, X3X2X1X0, encode each value. With this scheme, we derive the NAND gate propagation table (see Table I) using Boolean operations. Given a NAND gate with inputs a and b and output x, bits X3X2X1X0 are related to A3A2A1A0 and B3B2B1B0 as follows:

$$\begin{aligned} X_3 &= \overline{A_3 \cdot B_3} & X_2 &= \overline{A_2 \cdot B_2} \\ X_1 &= A_1 \cdot B_3 + A_3 \cdot B_1 & X_0 &= A_0 \cdot B_2 + A_2 \cdot B_0 \end{aligned}$$

Similarly, the NAND-gate sensitization table is derived. Given X3X2X1X0 and A3A2A1A0, sensitization flags Sr and Sf for the gate input a are as follows:

$$\begin{aligned} S_r(a) &= A_1 \cdot X_3 + \overline{A_3} \cdot X_1 \\ S_f(a) &= A_0 \cdot X_2 + \overline{A_2} \cdot X_0. \end{aligned}$$

The propagation and sensitization tables represent Boolean operations. Therefore, these as lookup tables are not stored. This facilitates machine-word parallelism since 32 Boolean operations are executed in parallel as a word operation (assuming a 32-bit word length). Path tracing also uses machine word parallelism by performing AND operations on sr and sf words along the paths in a depth-first manner as described earlier. Using parallel-vector concurrent simulation, 32\*2n SIC vectors (n is the number of inputs) are simulated in one pass. Therefore, an exhaustive simulation of SIC vectors needs only n2 to the n power / 64n equal to 2 to the n-6 power passes.

## 4. Experimental Results

| Circuit | Inputs | Total PDFs | Our method       |           | Schulz <i>et al.</i> |           |
|---------|--------|------------|------------------|-----------|----------------------|-----------|
|         |        |            | Detected ST PDFs | CPU (sec) | Detected ST PDFs     | CPU (sec) |
| c432    | 36     | 167,852    | 13,193           | 19        | 9,151                | 291       |
| c499    | 41     | 18,880     | 10,816           | 27        | 120,216              | 625       |
| c880    | 60     | 17,284     | 10,275           | 36        | 5,720                | 414       |
| c1355   | 41     | 8,346,432  | 735,234          | 312       | 326,551              | 1,677     |
| c1908   | 33     | 1,458,112  | 105,725          | 105       | 38,310               | 886       |
| c2670   | 233    | 1,359,908  | 76,280           | 427       | 50,071               | 2,200     |
| c3540   | 50     | 53,206,632 | 493,498          | 514       | 355,135              | 5,898     |
| c5315   | 178    | 2,682,610  | 252,010          | 731       | 144,697              | 4,913     |
| c7552   | 207    | 1,452,988  | 178,193          | 1,135     | 154,968              | 7,744     |

TABLE IV  
Comparison with Non-concurrent Simulation

| Circuit | Inputs | Total PDFs | ST PDFs |       |       | No. of Passes | CPU (sec) |     |
|---------|--------|------------|---------|-------|-------|---------------|-----------|-----|
|         |        |            | Rise    | Fall  | Total |               | SIM       | DET |
| s27     | 7      | 56         | 26      | 24    | 50    | 2             | 0         | 9   |
| s208.1  | 18     | 284        | 142     | 142   | 284   | 31            | 0         | 7   |
| s298    | 17     | 462        | 185     | 179   | 364   | 5             | 0         | 42  |
| s382    | 24     | 800        | 368     | 366   | 734   | 51            | 1         | 113 |
| s386    | 13     | 414        | 207     | 207   | 414   | 61            | 1         | 10  |
| s400    | 24     | 896        | 368     | 385   | 753   | 51            | 1         | 178 |
| s420.1  | 34     | 948        | 474     | 474   | 948   | 46,471        | 1,511     | 47  |
| s510    | 25     | 738        | 369     | 369   | 738   | 26            | 1         | 15  |
| s526    | 24     | 820        | 364     | 356   | 720   | 515           | 12        | 68  |
| s526n   | 24     | 816        | 362     | 356   | 718   | 515           | 13        | 71  |
| s820    | 23     | 984        | 492     | 492   | 984   | 202           | 7         | 169 |
| s953    | 45     | 2,312      | 1,156   | 1,156 | 2,312 | 886           | 77        | 147 |
| s1488   | 14     | 1,924      | 957     | 959   | 1,916 | 97            | 5         | 151 |
| s1494   | 14     | 1,952      | 961     | 966   | 1,927 | 97            | 5         | 155 |

TABLE V  
Comparison with A Deterministic Method

The proposed method was implemented in the C programming language and ran it on an HP 9000/s735 workstation, a 32-bit word-length machine. Table IV shows results for the ISCAS'85 benchmarks. The first two columns give the circuit name and the number of primary inputs (PI's). The third column gives the total number of PDF's, which is twice the number of paths. The fourth and fifth columns give the number of detected ST PDF's and the total CPU time for our method. To obtain adequate fault coverage, we used 312 simulation passes, which is equivalent to simulating 312\_32\_2\_2\_PI SIC vectors. The sixth and seventh columns show the detected ST PDF's and CPU time, on a Micro-VAX, for simulation of 10 000 random vectors (approximately 312 passes) as reported by Schulz *et al.* [7]. Their simulation also includes robust PDF detection. We averaged the results of Table IV except those for c499 because Schulz *et al.* detected more ST PDF's than the total (perhaps a misprint in their paper) [7]. Our method detects more ST PDF's and is found to run faster when the CPU times are normalized. We omit

results for *c6288* because of the intractability of enumerating all PDF's and that of enumerating the detected PDF's in our method and in the method of Schulz *et al.*

Because of the SIC restriction, in general, we need more vectors than the multiple input change vectors for the same coverage. However, SIC tests have other advantages. First, they are more likely to be robust tests. Second, SIC vectors have lower power consumption and are less likely to exceed the power rating of the device under test. Our experiment, conducted to show the simulation efficiency, did not drop any vectors that did not detect new faults.

Table V shows results for combinational parts of selected ISCAS' 89 benchmarks. The first three columns give the circuit name, number of combinational inputs, and total PDF's. The simulation was stopped when the number of detected rising and falling ST PDF's (columns 4 and 5) became equal to Total (column 6), which was taken from a deterministic method [2]. That method gives the exact number of ST PDF's for a circuit. The last three columns show the number of simulation passes and the total CPU time for the simulation (SIM) and deterministic (DET) methods. Table V shows that in most circuits, the simulation method detects all ST PDF's using only a fraction of the CPU time required by the deterministic method. However, for the circuit *s420.1*, the simulation method used more CPU time. In this simulation, 8% of the CPU time was spent in bringing the ST PDF coverage to 90%. The remaining 92% of CPU time was consumed in increasing the coverage to 100%. This is due to the presence of *random vector resistant* PDF's and suggests the use of a brief simulation phase followed by a deterministic phase [2] that targets the remaining PDF's.

## 4. Conclusion

The efficiency of simulation of single-input change (SIC) vectors test is an effective method of generating tests for PDF's. The main conclusion drawn from this study is that the new 16-valued algebra concurrently simulates rising and falling PDF's. It allows an encoding that is suitable for

machine word parallelism using Boolean operations. In circuits lacking random vector resistant PDF's, the method in this paper converges rapidly to 100% coverage because SIC vectors are sufficient to detect all ST PDF's. In some cases, however, it may be advantageous to use deterministic vector generation following a brief SIC vector phase.

## References

- [1] A.K. Majhi and V.D. Agrawal, Tutorial: "Dealy Fault Models and Coverage", IEEE VLSI Design, pp. 364-369, 1998
- [2] M.A. Gharaybeh, M.L. Bushnell, and V.D. Agrawal, "Classification and test generation for path-delay faults using single stuck-fault tests", in Proc. ITC, Oct. 1995.
- [3] C.J. Lin and S.M. Reddy, "On delay fault testing in logic circuits", IEEE Trans. on CAD, vol. 6, pp. 694-703, Sept. 1987
- [4] S.M. Reddy, C.J. Lin, and S. Patil, "An automatic test pattern generator for the detection of path delay faults", in Proc. ICCAD, pp. 284-287, Nov. 1987.
- [5] W. Ke, P. R. Menon, "Synthesis of delay verifiable two-level circuits", in Proc. European D&T Conf., pp 297-301, Feb. 1994.
- [6] G.L. Smith. "Model for Delay Faults Based upon Paths", In Proc. International Test Conference on VLSI Design, pp 342-349, Sept. 1985.
- [7] M.H. Schulz, F. Fink, and K. Fuchs. "Parallel Pattern Fault Simulation of Path Delay Faults. In Proc. 26<sup>th</sup> Design Automation Conference, pp 357-363, 1989.
- [8] W.K. Lam, A. Saldhana, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. "Delay Fault Coverage, Test Set Size, and Performance Tradeoffs". IEEE Transactions on Computer Aided Design, 14(1):32-44, Jan. 1995.
- [9] S. Bose, P. Agrawal, and V. D. Agrawal, "Path delay fault simulation of sequential circuits," *IEEE Trans. VLSI Syst.*, vol. 1, pp. 453-461, Dec. 1993.