

Test Generation for Primitive Path Delay Faults in Combinational Circuits*

Ramesh C. Tekumalla Prem R. Menon
Department of Electrical and Computer Engineering
University of Massachusetts
Amherst, MA 01003 USA

Abstract

This paper presents a method of identifying primitive path-delay faults in combinational circuits, and deriving robust tests for all robustly testable primitive faults. It uses the concept of sensitizing cubes to reduce the search space. This approach helps identify faults that cannot be part of any primitive fault, and avoids attempting test generation for them. Sensitization conditions determined for primitive fault identification are also used in test generation, reducing test generation effort. Experimental results on some of the ISCAS'85 and MCNC'91 benchmark circuits indicate that they contain a fair number of primitive multiple path delay faults which must be tested.

1 Introduction

Delay fault testing is the phase that succeeds the functional and logic testing phases. In this phase, the circuit is assumed to be functionally correct and free from logic faults. The propagation delays along leads and gates in a circuit determine its timing behavior. The accuracy of timing behavior of a circuit is determined by delay fault testing. There are two kinds of fault models used for this purpose. One is the gate delay fault model and other is the path delay fault model[1]. Since the clock period is determined by the delay of the longest sensitizable path, the path delay fault model has been widely used. In this model, the delay of the path is assumed to increase. Testing path delay faults makes testing gate delay faults unnecessary because path delays include gate delays also.

There has been considerable amount of work in delay testing [2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12] of combinational circuits. The focus of the work has been on obtaining robust tests for single path delay faults. (A robust test detects a fault independent of the delays in the rest of the circuit.) It has been observed that a large fraction of faults in most

circuits do not have robust tests. Some of these may be redundant and need not be tested because they do not affect circuit timing. Thus, an important problem in delay testing is "What are the faults that may affect circuit timing and must be tested?"

A necessary and sufficient set of faults, called primitive faults, that must be tested to guarantee timing correctness of the circuit was identified in [7]. A method for identifying primitive faults in two-level circuits was also proposed in [7]. Its extension to multi-level circuits using the Equivalent Normal Form (ENF) [13], is not practical for large circuits. More recently, a method of identifying primitive faults in combinational circuits and generating tests for them was reported in [15]. Results were presented only for identifying primitive faults of size 2. Another method of identifying faults based on signal stabilization times has been published in [14], but test generation was not discussed.

The goal of this paper is to develop a method for identifying all primitive faults in combinational circuits, and deriving tests for all robustly testable primitive faults. If a primitive fault is not robustly testable, non-robust tests for such faults are obtained. A distinguishing feature of our method is the use of sensitizing cubes for achieving both the above goals. The sensitizing cubes of a path specify all mandatory assignments for testing it, and reduce the search space for test generation.

The rest of the paper is organized as follows. The concept of sensitizing cubes of paths in multi-level combinational circuits, and a method of deriving them are discussed in Section 2. Section 3 shows how this concept can be used in deriving tests for single and multiple faults. Experimental results are presented in Section 4, followed by the summary and conclusions in Section 5.

2 Sensitizability of paths

In this section, we introduce the concept of sensitizing cubes of a combinational circuit, and discuss some of their properties. A method of deriving sensitizing cubes is also presented.

*This research was supported in part by the NSF grant MIP-9320886.

We consider the circuit feeding each output of the circuit separately. A rising (falling) transition delay fault on a path π will be represented by $\uparrow (\downarrow)\pi$, where the arrow denotes the direction of signal transition at the destination.

Definition 1: A value assignment to a subset of inputs of a circuit that produces a 0 or 1 at the output is called a *cube*. A cube that produces 0(1) is referred to as a 0(1)-cube. A cube may also be represented as a set (or product) of input literals. A value assignment to all inputs corresponds to a vertex, also called an *input vector* or simply a *vector*.

Definition 2: A *multipath* π consists of a set of single paths $\{\pi_1, \pi_2, \dots, \pi_n\}$ to the same destination. Every on-path input of every gate G on $\pi_i \in \pi$ is an on-path input of π . Every other input of G is a side-input of π . A multiple path delay fault $\uparrow(\downarrow)\pi$ is a situation in which every single path in π has a rising(falling) transition delay fault.

Definition 3: A multipath π is *static sensitized* to 1(0) by a vector v if it sets the destination to 1(0) and sets all side inputs to their respective non-controlling values.

Definition 4: A fault $\uparrow(\downarrow)\pi$ is *primitive* if 1) π is static sensitizable for a 1(0) at its destination and 2) there exists no multipath $\psi \subset \pi$ that is static sensitizable for a 1(0) at its destination [7].

Definition 5: A cube q is a *sensitizing* cube of a multipath π if (1) it sets every side-input to the non-controlling value when the on-path input is non-controlling, (2) no-side input has a controlling value when the on-path input is controlling and (3) there exists no $q' \subset q$ that satisfies these conditions. A sensitizing cube that produces 1(0) at the output is called a sensitizing 1(0)-cube.

The set of sensitizing cubes of a multipath π that produce 1(0) at its destination will be denoted $S_{1(0)}(\pi)$. We shall now discuss some properties of sensitizing cubes, explicitly considering only cubes in $S_1(\pi)$. Similar results hold for the sensitizing cubes in $S_0(\pi)$.

Lemma 1: If a cube $q \in S_1(\pi)$ contains a literal l_i , then there exists a multipath π that contains a single path from x_i , where x_i is the input corresponding to the literal l_i .

The proofs of all lemmas and theorems can be found in [16].

Lemma 2: No vector in $q \in S_1(\pi)$ will static sensitize $\lambda \subset \pi$.

Lemma 3: Any vector in $q \in S_1(\pi)$ will static sensitize a multipath $\psi \supseteq \pi$.

Lemmas 2 and 3 show a method of limiting the search for identifying static sensitizable multipaths. For the fault on a multipath to be primitive, it should not contain any static sensitizable subset of multipaths.

Lemma 4: Consider a pair of sensitizing cubes $q_1 \in S_1(\pi)$ and $q_2 \in S_1(\psi)$. If $q_1 \subseteq q_2$, then $\pi \supseteq \psi$.

Lemma 4 shows that the sensitizability conditions of π can be derived from the sensitizing cubes in $S_1(\psi)$. Thus, sensitizing cubes in $S_1(\pi)$ do not provide any useful

information for identifying multipaths that may contribute to primitive faults.

Lemma 5: If $S_1(\pi)$ is empty, then π is not static sensitizable for a 1 at its destination.

Before discussing the proposed method for obtaining sensitizing cubes, we shall discuss the ENF [13] of multi-level circuits. The ENF is a two-level expression obtained from the multi-level circuit by writing the expression for each gate output in a two-level form. Boolean simplification rules are not used to reduce the expression. The output of a gate is a set of cubes with the gate number tagged to literals in the cubes. Literals corresponding to the same primary input but with different tags are treated as different primary inputs as we proceed from the primary inputs to the output. The expression at the output of a gate is obtained by manipulating the expression on its inputs depending on the function realized by the gate. The final ENF expression contains product terms that may have a literal and its complement but with different tags. Example 1 shows the ENF of the circuit in Fig. 1.

Each literal in the ENF corresponds to a path from a primary input to output. A literal is complemented (uncomplemented) if the corresponding path in the original multi-level circuit has an odd (even) inversion parity. Also, an input combination that makes all the terms containing a literal l_π equal to 1 and all other terms equal to 0 is a static sensitizing condition for the path π [8]. Similarly, it can be shown that a necessary assignment of inputs that makes a product term containing l_{π_i} equal to 1 is a sensitizing cube of a multipath π , where $\pi_i \in \pi$.

No efficient algorithm has been suggested for obtaining the ENF of a given circuit. The major factor that prohibits such a task is the space requirement for storing the cubes as we go along from the input to the output. The other deterring factor is the number of variables that have to be kept track of. The final ENF expression may contain as many distinct literals (with different tags) as the number of single paths in the circuit. We shall see in Lemma 6 that some of the cubes in the ENF do not correspond to any sensitizing cubes and hence do not identify any primitive faults. We use the following approach for obtaining all the necessary sensitizing cubes for the circuit.

We obtain the sensitizing 1-cubes of a multi-level circuit from a two-level sum-of-products expression, called the collapsed form of the circuit. The collapsed form is obtained by expanding the (usually) factored expression representing the circuit on a gate by gate basis. All the rules of boolean algebra (including $a + ab = a$) are used to reduce the expression. The inputs to a gate before collapsing the gate are sets of cubes and the output of the gate after collapse is a set of cubes obtained by using the logic implemented by the gate. For example, if there is a two input NAND gate with the sets of cubes f_1 and f_2 on its inputs, the output

of the gate after collapse is $\overline{f_1} + \overline{f_2}$. The sensitizing 0-cubes of the circuit are obtained from the collapsed version of the complemented (the destination gate of the original multi-level circuit is replaced by a complementary gate, e.g NAND by AND, OR by NOR etc.) circuit. Let us call the set of cubes in the collapsed form of original (complemented) circuit as C^1 (C^0).

The following lemmas show the relationship between the collapsed form and the ENF.

Lemma 6: Consider a cube q in the ENF that has a literal and its complement but with different tags. Then, q is not a sensitizing cube for any multipath.

Lemma 7: Every cube in C^1 contains at least one cube in the ENF and the cubes in the ENF that are not contained in any cube in C^1 contain only false vertices.

Lemmas 6 and 7 show that the cubes of the collapsed form expression are sufficient for identifying all primitive faults in the circuit.

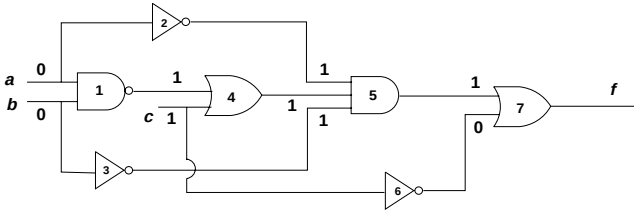


Figure 1: Sensitizing cubes for multi-level circuits

Example 1: Consider the circuit in Fig. 1. The ENF expression for the circuit is $\overline{a_{1457}} \overline{a_{257}} \overline{b_{357}} + \overline{a_{257}} \overline{b_{1457}} \overline{b_{357}} + \overline{a_{257}} \overline{b_{357}} c_{457} + \overline{c_{67}}$. The sensitizing cubes $C^1 = \overline{a} \overline{b} + \overline{c}$. It can be seen that all information in the ENF is contained in the sensitizing cubes and the terms in the collapsed are lost only because of the simplifications $f_1 \cdot f_1 = f_1$ and the cube containment $f_1 + f_1 \cdot f_2 = f_1$. $\square$

The sensitizing cubes of a multipath are similar to functional sensitizing cubes [4]. However, the sensitizing cubes differ from functional sensitizing cubes in that the latter have no restrictions on side inputs when the on-path inputs are controlling. Therefore, a sensitizing cube for a multipath is also a functional sensitizing cube of every single path in the multipath. It follows from Lemmas 6 and 7 that the cubes of the collapsed form constitute all the sensitizing cubes in the on-set of the multi-level circuit. Similarly, the sensitizing cubes in the off-set are C^0 .

Example 2: Consider the circuit in Fig. 1. The $S_1(\pi)$ for $\pi = [\{a, 1, 4, 5, 7\}, \{b, 1, 4, 5, 7\}]$ is $00x$, i.e. $\overline{a} \cdot \overline{b}$. It can be seen that no proper subset of π in the above multipath has a sensitizing 1-cube. The cube $00x$ is a functional sensitizing cube for a 1 at the destination of every single path in π . $\square$

3 Primitive fault identification and test generation in combinational circuits

We shall now use the sensitizing cubes discussed in the preceding section for primitive fault identification and test generation. The process of primitive fault identification can be made simpler if we can identify static sensitizable multipaths such that no proper subset of such multipaths are static sensitizable. Lemma 4 shows conditions under which some static sensitizable multipaths can be discarded. The following theorems identify primitive faults for rising and falling transitions using cubes in C^1 and C^0 .

Theorem 1: Consider a set of sensitizing cubes $S = \{q_1, q_2, \dots, q_n\}$. The appropriate fault on a multipath π with a static sensitizing vector v , is *primitive* iff (1) v is a common vertex for each of the cubes in S (2) no cube $q_i \in S$ has an essential vertex and (3) there exists no vertex that is common to only a subset of cubes in S .

Theorem 2: An essential vertex v in a sensitizing cube q is a static sensitizing vector for a multipath π and the appropriate path delay fault on it is *primitive*.

Theorems 1 and 2 show that only essential vertices and common vertices in sensitizing cubes that do not have essential vertices need be considered for identifying primitive faults.

The primitive fault identification algorithm is shown in Fig. 2. After identifying the sets of sensitizing cubes C^1 and C^0 , it uses Theorems 1 and 2 to identify vectors and the corresponding static sensitizable multipaths. The static sensitizable multipaths are identified by tracing paths in the multilevel circuit, after applying the vectors identified by Theorems 1 and 2. The appropriate faults on the multipaths thus identified are primitive.

Cubes containing essential vertices are identified and processed first. Cubes without essential vertices are processed in a separate phase. An essential vertex in a cube is identified by assigning values to the inputs so that only the product representing that cube evaluates to 1. Starting with the values specified in the cube, additional input values are assigned so that all other terms evaluate to 0. The paths sensitized by this vector are determined by tracing the paths going backwards towards primary inputs. All the primitive single faults are identified during this phase.

Theorem 1 is applied to cubes without essential vertices to identify primitive multiple path faults. Let S be the set of sensitizing cubes without essential vertices. The vertices common to proper subsets of S are determined. A common vertex of a set of cubes is determined by making input assignments consistent with all cubes in the set. Let v_i be the common vertex of a set of cubes $S_i \subset S$. The multiple fault static sensitized by v_i is primitive if no $S_j \supset S_i$ has a common vertex.

Example 3: Consider the circuit in Fig. 3. The set

Procedure_Identify_Primitive_faults()

```

{
  enumerate the paths in the circuit;
  obtain the sensitizing cubes for each output logic;
  for each sensitizing cube  $q_i$  {
    obtain an essential vertex in  $q_i$ ;
    if such a vertex exists, identify the multipath  $\pi$  for which
      it is a static sensitizing vector;
    the appropriate transition on  $\pi$  is primitive;
    else retain the sensitizing cube for processing at a later
      stage;
  }
  if there exist sensitizing cubes  $S = \{q_1, \dots, q_n\}$  without
    essential vertices
  for  $i = 1, 2, \dots, n$  {
     $q_{v1} = q_i$ ;
    for  $j = 2, \dots, n$  {
      for  $k = j, \dots, n$  {
        obtain the non-null intersection,  $q_{v1} = q_{v1} \cap q_k$ ;
        identify a vertex in  $q_{v1}$  that is not contained in any
          sensitizing cube that is not in  $S$ ;
        identify a multipath  $\pi$  static sensitizable by  $q_{v1}$ ;
        the appropriate fault on  $\pi$  is primitive;
         $q_{v1} = q_i$ ;
      }
    }
  }
}

```

Figure 2: Primitive Fault Identification

of sensitizing cubes C^1 for the circuit are $10x$, $1x0$, $x10$ and $01x$. In the first phase, the essential vertices 101 in $10x$ and 011 in $01x$ are obtained. These two vertices static sensitize the multipaths $\{(b, 1, 3, 5, f)\}$ and $\{(a, 2, 4, 5, f)\}$ respectively. The cubes $1x0$ and $x10$ are left for processing in phase two because they do not have essential vertices. In phase two, the intersection of the above two cubes 110 is obtained, which is a static sensitizing vector for the multipath $\{(c, 1, 3, 5, f), (c, 2, 4, 5, f)\}$. It can be verified that all static sensitizable multipaths that contribute to primitive faults, are identified by the above three vertices. The primitive faults identified are $\uparrow\{b, 1, 3, 5, f\}$, $\uparrow\{a, 2, 4, 5, f\}$ and $\uparrow\{(c, 1, 3, 5, f), (c, 2, 4, 5, f)\}$. The primitive faults for the falling transitions can be identified using the cubes in C^0 . $\square$

The test generation procedure is given in Fig. 4. The static sensitizing vectors obtained during primitive fault identification are used as final vectors in tests for primitive faults. The initial vector is obtained by complementing the values at the start of the target multipath. If all remaining inputs are kept constant in the vector pair, all side inputs to

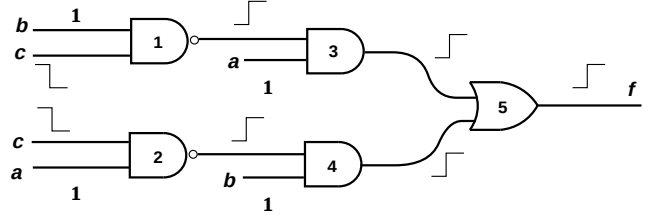


Figure 3: Circuit with a primitive multiple fault

Procedure_Test_Generation()

```

{
  Procedure_Identify_Primitive_Faults();
  for each primitive fault  $\uparrow(\downarrow)\pi$  {
    obtain a vector pair  $\langle V_1, V_2 \rangle$  such that  $V_2$  is a static
      sensitizing vector that produces 1(0) at the
      destination of  $\pi$  and  $V_1$  differs from  $V_2$  only on the
      values on literals at the start of  $\pi$ ;
    apply the vector pair and check for robustness
      conditions;
    if robustness conditions are not satisfied, the vector
      pair is a non-robust test;
  }
}

```

Figure 4: Test generation for primitive faults

the multipath will have a stable non-controlling value or a final non-controlling value. The vector pair is a robust test if all gates with a final controlling value on on-path inputs have stable controlling values on side inputs. Since the final vector in the test is a static sensitizing vector, the vector pair is a non-robust test for the target fault if robustness conditions are not satisfied.

Example 4: Consider the circuit in Fig. 3. The vector 110 static sensitizes the multipath $\{(c, 1, 3, 5, f), (c, 2, 4, 5, f)\}$ to 1. Since the variable at the start of the multipath is c , the initial vector can be obtained by complementing the value on c in the final vector. It can be seen that $\langle 111, 110 \rangle$ is a robust test for the multiple fault $\uparrow\{(c, 1, 3, 5, f), (c, 2, 4, 5, f)\}$. $\square$

4 Experimental Results

A test generator for primitive path delay faults using the methods discussed in the preceding sections has been implemented in the C language. Primitive fault identification and test generation were performed on combinational versions of several of the ISCAS'89 and MCNC'91 benchmark circuits, using the following procedure: First, the single paths in the circuit were enumerated by a depth-first traversal of the circuit starting from each circuit output. SIS [17]

Circuit	# Gates	# Faults	# Single Primitive Faults		# Multiple Primitive Faults		Test Time Sec
			Total	Robust	Total	Robust	
s208	104	284	257	256	6	1	11.31
s298	119	502	343	312	32	5	16.04
s344	160	710	632	613	130	3	86.81
s349	161	730	633	613	133	4	82.01
s382	158	800	666	636	22	9	48.66
s386	159	414	400	394	34	1	35.61
s400	162	896	685	638	33	14	34.23
s420	218	948	826	826	14	1	140.16
s444	181	1070	718	656	39	14	94.42
s510	211	738	706	658	64	2	123.31
s526	193	820	686	663	81	7	89.46
s641	379	3444	2048	1872	274	54	412.61
s713	393	43624	4291	2693	305	0	1050.74
s820	289	984	936	887	103	6	438.16
s832	287	1012	928	828	205	10	534.77
s838	446	3428	2932	2931	30	1	492.28
s1488	590	2762	1973	1759	291	54	1521.19
s1494	472	2064	1773	1681	223	25	1562.80
s1196	529	6194	3010	2783	1347	58	1572.98
bbsse	101	358	330	321	35	2	14.59
cse	146	914	725	696	112	18	64.49
dk16	244	898	623	559	106	19	144.39
keyb	2238	1192	874	783	164	16	187.82
planet	646	3924	2773	2267	688	61	921.67
sse	117	410	350	311	33	0	22.06
scf	916	7036	5076	3074	410	27	1683.13
s1	446	2556	1824	1491	735	44	682.52
styr	477	1964	1373	1238	432	32	1701.68
tma	140	506	476	444	63	4	34.54

Table 1: Primitive Fault Coverages for ISCAS'89 and MCNC'91 benchmark circuits

was then used to obtain the collapsed form expressions for each output and its complement. The product terms of these expression are the sensitizing cubes used for primitive fault identification and test generation. The robust tests for multipaths are obtained as explained in the procedure. A multipath can have many sensitizing cubes. As explained in the procedure and since all the cubes are considered, a robust test will be found if one exists.

Experimental results for combinational versions of the benchmarks are shown in Table 1. The circuit and the number of gates are shown in columns 1 and 2 respectively. Column 3 gives the total number of single faults in the circuit. Columns 4 and 5 show the number of primitive single faults and the number of robustly testable single faults, respectively. Columns 6 and 7 show the number of primitive multiple faults and the number of robustly testable

primitive multiple faults, respectively. Non-robust tests are derived for all the robustly untestable primitive faults. Total run times on a SUN SPARC 20 workstation for primitive fault identification and test generation, are shown in Column 8.

5 Conclusions

This paper has presented a test generation algorithm for primitive path-delay faults in combinational circuits. Experimental results obtained with an implementation of the algorithm show good delay-fault coverage and reasonable run times. One of the main features of the algorithm is the use of sensitizing cubes to determine mandatory assignments for path-delay tests. This helps reduce the search space for primitive fault identification and test generation.

The set of tests derived by our method detect not only all robustly testable single faults but also robustly testable multiple faults which can affect circuit timing. This set of tests would therefore be more effective in detecting timing problems in comparison with a test set for singly robust testable faults alone. The experimental results presented indicate that the number of primitive multiple faults is less than the number of single faults in all the circuits considered. Thus, the number of multiple faults to be tested is much less than the total number of multiple faults (the latter usually being much greater than the number of single faults). Still, the number of primitive multiple faults is not negligible in most cases. A measure of the effectiveness of a delay test set should include coverage of both single and multiple primitive faults.

The algorithm presented here derives robust and non-robust tests (if a robust test does not exist) for primitive faults. It is possible to modify it to obtain validatable robust tests for some of the robustly untestable faults. Further work is needed to develop a general algorithm that does not involve collapsing the circuit, to derive robust or VNR tests for all testable primitive faults. The algorithm presented here is also being extended to sequential circuits.

References

- [1] G.L. Smith, "Model for delay faults based upon paths", *Proc. Intl. Test Conf.*, 1985, pp.342-349.
- [2] C.J. Lin and S.M. Reddy, "On delay fault testing in logic circuits", *IEEE Trans. on CAD*, pp.694-703, Sept. 1987.
- [3] A. Saldanha, R.K. Brayton and A.L. Sangiovanni-Vincentelli, "Equivalence of robust delay-fault and single stuck-fault test generation", *Proc. Design Automation Conf.*, 1992, pp.173-176.
- [4] K.-T. Cheng and H.C. Chen, "Delay testing for non-robust untestable circuits", *Proc. Intl. Test Conf.*, 1993, pp.954-961.
- [5] W.K. Lam, A. Saldanha, R.K. Brayton and A.L. Sangiovanni-Vincentelli, "Delay fault coverage and performance tradeoffs" *Proc. Design Automation Conf.*, 1993, pp.446-452.
- [6] C. Lin, S.M. Reddy and S. Patil, "An automatic test pattern generator for the detection of path delay faults", *Proc. ICCAD*, 1987, pp.284-287.
- [7] W. Ke and P.R. Menon, "Synthesis of delay-verifiable combinational circuits", *IEEE Trans. on Computers.*, vol. 44, pp. 213-222, February 1995.
- [8] S. Devadas and K. Keutzer, "Synthesis of robust delay-fault-testable circuits:practice", *IEEE Trans. on CAD*, vol.11, no.3, pp.277-300, March 1992.
- [9] S. Devadas and K. Keutzer, "Validatable nonrobust delay-fault-testable circuits via logic synthesis", *IEEE Trans. on CAD*, vol.11, pp.1559-1573, December 1992.
- [10] K. Roy, K. De, J.A. Abraham and S. Lusky, "Synthesis of delay fault testable combinational logic", *Proc. ICCAD*, 1989, pp.418-421.
- [11] A. K. Pramanick and S.M. Reddy, "On the design of path delay fault testable combinational circuits", *Proc. Intl. Symposium on Fault Tolerant Computing*, 1990, pp.374-381.
- [12] I. Pomeranz and S.M. Reddy, "Design-for-Testability for path delay faults in large combinational circuits using test-points", *Proc. Design Automation Conf.*, 1994, pp.358-364.
- [13] D.B. Armstrong, "On finding a nearly minimal set of fault detection tests for combinational logic nets", *IEEE Trans. Electron. Computers*, vol. EC-15, pp.66-73, February 1966.
- [14] M. Sivaraman and A. J. Strojwas, "Primitive path delay fault identification", *Proc. 10th Intl. Conf. on VLSI Design*, January 1997, pp.95-100.
- [15] A. Krstic, K.-T. Cheng and S.T. Chakradhar, "Identification and test generation for primitive faults", *Proc. Intl. Test Conf.*, 1996, pp.423-432.
- [16] R. Tekumalla and P.R. Menon, "On test generation for primitive path delay faults in combinational circuits", *Tech. Report, TR-CSE-97-5, Dept. of ECE, Univ. of Massachusetts, Amherst*.
- [17] E.M. Sentovich, K.J. Singh et al., "SIS : A system for sequential circuit synthesis", Memo. No. UCB/ERL M92/41, Univ. of California, Berkeley, 1992.
- [18] Chih-Ang Chen and Sandip Gupta, "A satisfiability based test generator for path delay faults in combinational circuits", *Proc. Design Automation Conf.*, 1996, pp.209-214.