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

CSE Conference and Workshop Papers | Computer Science and Engineering, Department of

1-1-2007

Symbolic Path Sensitization Analysis and Applications

Jian Kang
University of Nebraska at Lincoln, jkang@cse.unl.edu

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

Shashank K. Mehta
Indian Institute of Technology, Kanpur, skmehta@cse.iitk.ac.in

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

Part of the Computer Sciences Commons

Kang, Jian; Seth, Sharad C.; and Mehta, Shashank K., "Symbolic Path Sensitization Analysis and Applications" (2007). *CSE Conference and Workshop Papers*. Paper 4.
http://digitalcommons.unl.edu/cseconfwork/4

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 Conference and Workshop Papers by an authorized administrator of DigitalCommons@University of Nebraska - Lincoln.

Symbolic Path Sensitization Analysis and Applications

Jian Kang and Sharad C. Seth
Department of Computer Science and Engineering
University of Nebraska - Lincoln
Lincoln, NE, USA
jkang, seth@ cse.unl.edu

Shashank K. Mehta
Computer Science and Engineering Department
Indian Institute of Technology
Kanpur -208016, India
skmehta@cse.iitk.ac.in

Abstract
A new symbolic approach models the sensitization paths to selected primary output(s) as Boolean equations, with satisfying solutions representing the set of all sources of single and multiple sensitizations in the circuit. The paper discusses two applications of this idea: model-free fault diagnosis and input sensitization analysis.

Keywords

Sensitization equation, CNF, Satisfiabilty, Fault diagnosis, Testability analysis

1. INTRODUCTION

The efforts at capturing multiple fault effects symbolically in equation form have a long history. Poage [1] in the 1960's used Boolean variables to denote the logical states of each circuit line, the variables are then built into logic equations which reflects the logical values of each line in the circuit.

In this paper, we use Boolean variables to capture the sensitization of lines in the circuit, since for testing related applications our interest is more in the presence or absence of sensitization paths to observation points than in the values of the variables. For each test pattern, we logic simulate the circuit and capture the sensitization conditions on each line as Boolean equations. We show two variations of sensitization equations and their applications in two areas: fault diagnosis and input sensitization analysis, with emphasis on the first one. We show how the two-variable equations, which give exact sensitization information, could be used in fault diagnosis of model-free faults. The single-variable equations are exact only under restricted conditions but still useful in input sensitization analysis.

The organization of the remaining part of the paper is as follows. In section 2, we introduce the sensitization equations. Section 3 develops the application of sensitization equations to model-free fault diagnosis and shows experiment results. Section 4 shows the application of input sensitization analysis and its results. Section 5 concludes the paper with a summary of future extensions.

2. SENSITIZATION EQUATIONS

2.1 Background

In Poage's work [1], faults are associated with lines in the circuit and the three mutually exclusive states of line *a* (stuck-at-1, stuck-at-0, and normal) are denoted by Boolean variables a_1, a_0 and a_n respectively. Further, the signal values at the source and destination of line *a* are distinguished as Boolean variables a_{in} and a_{out}, where the un-complemented (complemented) variable means that the line assumes value 1 (0). The equations for the value at the destination of line *a* can be written as:

$$a_{out} = a_1 + a_n \cdot a_{in} \qquad (1)$$

$$a'_{out} = a_0 + a_n \cdot a'_{in} \qquad (2)$$

Equation (1) means that value 1 on the output of line *a* may be caused by stuck-at-1 fault on this line, or this line is normal and its input value is 1. Similarly, Equation (2) means that value 0 on the output of line *a* may be caused by stuck-at-0 fault on this line, or this line is normal and its input value is 0.

As faults are assumed to be associated only with lines, other circuit elements (gates, fanouts, etc) can be modeled simply by their functionality. For example, for the AND gate A=BC, the equations will be:

$$a_{in} = b_{out} c_{out}$$

$$a'_{in} = b'_{out} + c'_{out}$$

Similarly, for a branch B of stem S, the equations will be:

$$b_{in} = s_{out}$$

$$b'_{in} = s'_{out}$$

The above model suffices to capture the circuit behavior under single or multiple faults. Later work extended Poage's approach to test generation for multiple stuck-at faults at checkpoints [2] and diagnosis of multiple stuck-at faults [3], by generating symbolic equations in the nested form for each circuit output using back substitution.

2.2 Sensitization Equations

Our symbolic sensitization analysis originates from these early works, but with major differences. For sensitization,

1081-7735/07 $25.00 © 2007 IEEE
DOI 10.1109/ATS.2007.21

IEEE computer society

our interest is more in the presence or absence of sensitization paths to primary outputs than in absolute signal values, hence for a given test pattern we capture the *sensitization condition* on each line by Boolean variables. When these line conditions are linked together by the circuit topology, collectively the set of equations represent sensitization paths from any line to primary outputs.

We use stuck-at-v' (stuck-at-v) to represent the sensitizing (non-sensitizing) fault condition on a line, where v is its fault-free value. Then, for a test pattern, a line can be in only one of the three states: stuck-at-v', stuck-at-v, or normal, which can be encoded by two Boolean variables. Table 1 shows the encoding we use among the many possibilities with two variables.

Table 1. Boolean encoding of line states

a_f	a_n	Meaning
0	0	*a* stuck-at-v
0	1	*a* normal
1	-	*a* stuck-at-v'

This encoding was chosen so as to simplify the expression for the *sensitizing* faulty state on a line, which is represented simply by a_f. The two non-sensitizing states, stuck-at-v and normal, are represented by $a'_f a'_n$ and $a'_f a_n$ respectively.

Based on the encoding in the table, we can now define the sensitization equations for each circuit element. We use notations A_{in} and A_{out} to denote that there is sensitization on input and output of line *a* respectively. Following [1], we assume that faults are only on lines, while other circuit elements are fault-free and they simply propagate the sensitization or non-sensitization effects.

Lines: The sensitization on the output of a line can be due to either a sensitizing fault on the line, or input sensitization when the line is normal:

$$A_{out} = a_f + a'_f\, a_n A_{in}$$

which reduces, through Boolean simplification, to:

$$A_{out} = a_f + a_n A_{in} \qquad (3)$$

The complement of equation (3), which corresponds to non-sensitization of the line, can similarly be reduced to:

$$A'_{out} = a'_f\, a'_n + a'_f\, A'_{in} \qquad (4)$$

Note that any one of equations (3) and (4) is sufficient to model the sensitization condition on a line because the other equation can be derived from it by Boolean algebra. For simplicity, we use equation (3) in our modeling.

In case a line can be determined, *a priori,* to be fault-free, equation (3) reduces to:

$$A_{out} = A_{in}$$

Basic Gates: For basic gates, the propagation equations are gate and signal specific. We illustrate this for the AND gate A=AND(B,C).

If the error-free input values are BC=11, then the fault on either line B or C will propagate to A. Hence:

$$A_{in} = B_{out} + C_{out}$$

Similarly, for input values BC=01, the output will change only if B is sensitized but not C, therefore:

$$A_{in} = B_{out} C'_{out}$$

The equations for other types of gates and input values can be analyzed similarly.

Fanout Branches: The sensitization equation for a fanout branch, say B, in terms of stem S, can be written by regarding the fanout as a buffer that preserves the sensitization. Thus:

$$B_{in} = S_{out}$$

Primary Inputs: When the source of the primary input is reached, there is no possibility of sensitization arriving at the source, hence:

$$A_{in} = 0$$

Primary Outputs: The sensitization of primary outputs represents an external constraint on the solution that is determined by the application.

For the fault-diagnosis problem, a primary output is sensitized if and only if it is determined to be a failing output by the tester. Therefore, a failing primary output *z* imposes the constraint that Z_{out} should be true. Similarly, a non-failing primary output *y* imposes the constraint that Y'_{out} should be true. The logical product $\pi(T)$ of these literals, then, represents the overall constraints imposed by the primary outputs for pattern T.

The output constraints for input sensitization analysis are deferred to the discussion of the application in Section 4.

Example 1 (Diagnosis): Consider a very simple example circuit composed of just a fanout stem with two branches, as shown in Figure 1. The values of each line are marked in the figure. For this case, we observe that output B is faulty, while C is fault-free.

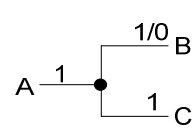

Figure 1. Simple fanout branch

The sensitization equations are shown below:

$$\pi(T) = B_{out} \cdot C'_{out}$$
$$B_{out} = b_f + b_n B_{in} \quad \text{and} \quad B_{in} = A_{out}$$
$$C_{out} = c_f + c_n C_{in} \quad \text{and} \quad C_{in} = A_{out}$$
$$A_{out} = a_f + a_n A_{in} \quad \text{and} \quad A_{in} = 0$$

Using Boolean expansion, the equation for $\pi(T)$ is equal to:

$$\pi(T) = B_{out} \cdot C'_{out} = b_f + a_f c'_f \, c'_n$$

This gives two solutions (B stuck-at-v'), and (A stuck-at-v', C stuck-at-v), which are easily verified to be all the solutions.

2.3. Simplified Equations

The above sensitization equations require two Boolean variables per line (a_f and a_n). Here, we investigate the possibility of using just one variable per line to represent the sensitization conditions. Such a formulation will result in smaller problem size. We analyze the information loss associated with the resulting equations, and its impact on the accuracy of sensitization analysis.

In the simplification, the fault-excitation state, stuck-at-v' of line *a* is explicitly represented by the Boolean variable a_f . Its complement, a'_f , must then ambiguously represent the stuck-at-v and normal states. The sensitization conditions at the output of line *a* are defined as:

$$A_{out} = a_f + A_{in} \qquad (5)$$

$$A'_{out} = a'_f \, A'_{in} \qquad (6)$$

Equation (5) correctly captures the sensitization of the output of line *a* if the absence of a line variable in a term is assumed to represent the normal line condition. Equation (6) correctly captures the non-sensitization at the output when a'_f represents the normal state and there is no incoming sensitization. However, it over-constrains the condition when the line is stuck-at-v because the non-sensitization of the line output is *independent* of the sensitization condition on the line input: in this case the correct solution for non-sensitization is just *a* stuck-at-v and not (*a* stuck-at-v and A'_{in}). Further, Equation (6) misses another valid solution, (*a* stuck-at-v and A_{in}).

Example 2 (Diagnosis): For the circuit in Example 1, Boolean expansion of the equations using single variable gives:

$$\pi(T) = B_{out} \cdot C'_{out} = b_f$$

which corresponds to single fault (B stuck-at-v). The double fault, (A stuck-at-v', C stuck-at-v), which has a masking relationship, is missing.

Note that, if only a single source for sensitization is assumed to occur, there is no over-constraining and both the equations exactly represent the sensitization conditions at the line output. Hence, this formulation will give correct results for sensitizations resulting from a single source but will miss some multi-source sensitizations that have a masking relationship. When the sources of sensitization are restricted to the primary inputs the stuck-at-v and the normal states are easily seen to be equivalent. Hence we have the following results:

Lemma 1: The simplified equations are exact under the assumption that the sensitization originates from a single line source in a combinational logic circuit.

Lemma 2: The simplified equations exactly capture all single and multi-source sensitizations restricted to the primary inputs in a combinational logic circuit.

Actually, Lemma 2 can be shown to be true, generally, for sensitization sources restricted to any *cut* of the circuit.

Example 3 (Input Sensitization): We analyze the sensitization conditions for the output of the two-input one-bit multiplexer circuit shown in Figure 1 in terms of its primary inputs.

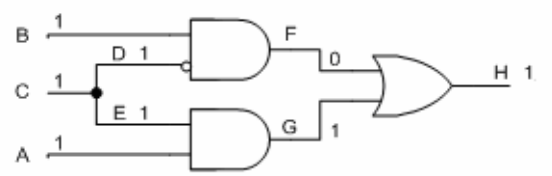

Figure 1. One-bit multiplexer

The sensitization equations are summarized below:

$$\pi(T) = H_{out} \qquad G_{out} = A_{out} + E_{out} \qquad A_{out} = a_f$$
$$E_{out} = C_{out} \qquad H_{out} = F'_{out} \cdot G_{out} \qquad B_{out} = b_f$$
$$D_{out} = C_{out} \qquad F_{out} = D_{out} \cdot B'_{out} \qquad C_{out} = c_f$$

Note that the equations for intermediate lines are in the reduced form because the sources of sensitization are assumed to be only at the primary inputs. After back-substitution and expansion,

$$\pi(T) = a_f + b_f c_f$$

which gives the two irredundant solutions possible for this problem: (A stuck-at-v'), and (B stuck-at-v', C stuck-at-v'), which means, the sensitization of line *a* alone or sensitization of lines *b* and *c* could be detected.

2.4. Solving Sensitization Equations Using SAT

As seen in Example 1, the set of sensitization equations for a test pattern *T,* given the circuit and primary-output constraints, can be reduced through back-substitution to a single equation $\pi(T)$. One way to solve $\pi(T)$ is to express it in the sum-of-products (SOP) form as we did in the three examples above. Then, each term in the SOP corresponds to one diagnosis solution. Expansion to the SOP form, however, can result in exponential explosion in

the size of the equation. The explosion problem can be avoided by converting the nested expression to a system of equations representing a gate-level structure. The latter can be transformed to its equivalent conjunctive normal form (CNF) expression that is linearly proportional in size to the gate-level circuit [4]. A SAT solver can then be employed for solving the CNF expression. This is an attractive strategy because powerful SAT solvers [5-7] are now available to solve complex problems.

3. FAULT DIAGNOSIS

In examples 1 and 2, we have shown how two-variable sensitization equations could be used for fault diagnosis by solving the equations in the SOP form. In this section, we discuss additional constraints when solving with a SAT solver.

As getting all the solutions is costly and effective diagnosis is possible by identifying solutions bounded by a small fault multiplicity, we adopt the technique in [8] to constrain the cardinality of the solutions.

The addition of fault-multiplicity constraints involves two steps. First, an additional equation is added for each line and each test pattern to denote that a line is faulty. Let Boolean variable F_a denote that line a is faulty. Then, according to the encoding used in Section 2:

$$F_a = a_f + a'_f\, a'_n = a_f + a'_n \qquad (7)$$

The second step constrains that there are exactly k suspect lines that can exhibit a faulty behavior (stuck-at-v or stuck-at-v'). A CNF for this constraint can be derived from F_a's using the counting technique described in [8]. Initially, k is set to 1 for SAT solving and then iteratively incremented if a solution cannot be found at this multiplicity. After each step, the CNF is modified to include the negation of already found solutions to avoid generating redundant solutions.

The overall procedure to find solutions of the diagnosis problem is as follows:

Step 1. Repeat for each failing pattern T and observed response R(T):

- Logic simulate T to determine the signal values.
- From fault-free output values and R(T), generate sensitization equation for every circuit line.
- Add fault-multiplicity constraints.
- Convert both the equations and constraints to CNF clauses.
- Add to the already-generated CNF clauses.

Step 2. Find solutions for the CNF clauses using a SAT solver.

Experimental Results

In order to overcome the limitations of the single stuck-at fault model in detecting un-modeled faults, we use Huisman's approach [9] which distinguishes the location of a defect from its logical behavior and assumes that, even though a defect behaves as a set of stuck-at faults on some set of nets during the application of any given test pattern, it need not behave as the *same set of stuck-at faults on every test pattern.* This *model-free* diagnosis is general enough to be able to represent a variety of fault models, including single or multiple stuck-at, bridging, cross-talk, and stuck-open faults.

The proposed technique is applied to model-free fault diagnosis on big ISCAS85 and scan version of ISCAS89 benchmarks. To minimize the effect of redundant faults on diagnosis, each circuit is first synthesized using the *rugged* script in SIS [10], and mapped to basic gates supported by the fault simulator [11]. We used MiniSAT [5] as the SAT-solver, and implemented the other tools using Perl. All experiments are run on PC with Intel D 2.66G Hz processor and 1G memory.

The tester data for our simulated experiments under the model-free assumption were generated using the following steps:

Step 1. Randomly select m lines in the circuit for fault injection, where m is the fault multiplicity.

Step 2. Randomly generate a test pattern and simulate it on the circuit. This gives fault-free response.

Step 3. Tie the selected lines to randomly selected binary values, and re-simulate the faulty circuit for the test pattern. This gives faulty response.

Step 4. If there are no failing outputs under this pattern and injected value(s), repeat steps 2 and 3.

Step 5. Repeat steps 2-4 until n failing test patterns are obtained.

Fault diagnosis is then carried out using obtained failing test patterns. As a comparison, we also implemented the diagnosis technique in [8], which inserts a multiplexer at each line to model both fault-free and faulty behaviors. We didn't consider sophisticated speed-up heuristics in this comparison as most speed-up techniques in [8], such as using structural dominator, should apply to both. However, we included the forced assignment heuristics [8] for multiplexer-based approach since it is unique for that approach.

Table 2 compares the number of clauses in the CNF of the two techniques before and after multiplicity constraints are added. In the table, column 1 lists the circuits, columns 2 and 3 compare the number of clauses before the

Table 2. Problem size comparison

	# Original clauses		# Clauses with constraints	
Ckt	EQN	MUX	EQN	MUX
c2670	55670	86270	125081	119061
c3540	101550	144870	225040	200060
c5315	144310	211870	322143	292343
c6288	236750	342510	524609	470569
c7552	184810	270910	412171	373651
s13207	281010	424020	635349	587549
s15850	341220	506270	767068	700768
s35932	730700	1111690	1644703	1533403
s38417	1146270	1688430	2567289	2334589
s38584	1154520	1689250	2585628	2338108

multiplicity constraints are added, and columns 4 and 5 show the same comparison after adding the multiplicity constraints. EQN denotes two-variable equations and MUX denotes mux-based approach. The equation-based formulation is seen to lead to a smaller number of CNF clauses than the mux-based approach. Although both techniques use the same adder circuit [8] to constrain fault multiplicity, the need for the addition of selection lines adds to the cost of equation-based method, thus leading to an overall larger number of clauses.

Table 3 summarizes the results for single fault diagnosis averaged over ten runs for each circuit; in each run ten failing test patterns are used. Column 1 shows circuit name and column 2 the number of lines in the circuit including fanout branches. Column 3 gives the number of fault sites returned to explain the observed responses, and columns 4 and 5 give the total time needed to find all the solutions using the two approaches respectively. Both techniques give identical diagnosis solutions. Also it can be seen that although model-free diagnosis is a harder problem than model-based (such as stuck-at) diagnosis, ten patterns are able to resolve the fault to just a few lines. For speed, the two techniques take comparable time. The preprocessing for EQN and multiplexer insertion for MUX take negligible time.

Table 4 shows the results for double fault diagnosis using the two approaches with ten failing test patterns. Column 2 in the table shows the number of solutions (fault tuples) returned, column 3 shows the number of fault sites. Columns 4 and 5 show the total run time using the two approaches. Circuit c6288 took too long to solve in this case and aborted, so its data is not included. It can be seen that ten failing patterns are not enough to resolve the fault to a small number of lines for the double-fault model. Use of more failing test patterns may help, and this shows the trade-off between a more general model vs. the diagnostic resolution. The solution times for the two techniques are comparable.

Table 3. Single model-free fault diagnosis

Ckt	# Lines	# Fault Sites	Time EQN (s)	Time MUX (s)
c2670	1376	4.5	3.00	3.86
c3540	2310	5	5.98	6.60
c5315	3364	8.3	46.59	64.34
c6288	5348	3.6	97.58	128.58
c7552	4292	11.4	9.49	7.87
s13207	6827	3.7	131.00	199.60
s15850	8119	5.8	230.33	388.28
s35932	17585	3.9	67.49	61.69
s38417	26938	3.8	4239.14	4427.78
s38584	27051	3.8	4342.88	4271.64

4. INPUT SENSITIZATION ANALYSIS

As another application of sensitization analysis, we consider the problem of determining the set of all single and multiple sensitization paths from primary inputs to *any* output of a combinational circuit for a given set of test patterns. This set depends only on the function of the circuit and may be used as the basis for a function-based testability analysis. As indicated in Section 2.3, this type of analysis can be carried out exactly with the simplified equations. The computation is equivalent to fault simulation for all single and multiple faults on the primary inputs for which the equation based approach provides an elegant and efficient solution.

For input sensitization analysis, the simplified equations are enough since from Lemma 2 we know that it gives exact sensitization information for primary inputs. Furthermore, we don't need to keep track of sensitizations for lines that are not primary inputs, therefore the equations for these lines are further simplified.

The external condition that constrains the solutions in this case is the logical OR of all the non-terminals corresponding to all the primary output set to be true, because that would correspond to the condition being true when the sensitization reaches any one (or more) of the primary outputs.

Table 4. Double model-free fault diagnosis

Ckt	# Sols	# Sites	Time EQN (s)	Time MUX (s)
c2670	82.4	23.4	7.27	7.63
c3540	90.8	19.9	24.42	20.03
c5315	109.4	25.7	60.50	78.46
c7552	102.4	48.6	28.24	34.77
s13207	32	11.3	162.53	224.71
s15850	30.8	10.5	272.01	413.14
s35932	20.6	9.7	84.96	232.32
s38417	16.6	8.5	4171.91	4779.40
s38584	22	9.7	4653.75	5016.27

As in fault diagnosis, it may suffice for the testability application to determine primary-input sensitizations of only low multiplicity. Such a multiplicity constraint can be enforced by the technique used in fault diagnosis.

Experiment Results

In this section we show the preliminary results on input sensitization analysis as the basis for future work on a functional coverage metric.

We show only irredundant subsets of inputs that can be sensitized to any primary output. For example, for a given test pattern, if an input alone can be sensitized to a primary output, then we don't further consider any super set of that input for sensitization to an output. The exact relationship of this measure to circuit testability is under investigation. An indication of the relationship is provided by the example of the multiplier circuit (c6288) in which every random pattern sensitizes all the 32 inputs, suggesting that this circuit may be easy to test with random patterns.

Table 5 shows the average results of ten runs, during each of which a random pattern is applied to the circuit and evaluated for single, double, and triple input sensitizations. Column 2 lists the number of inputs for each circuit, columns 3 to 5 show the number of single, double, and triple input sensitizations that are captured. Column 6 is the total run time for the three analyses in columns 3 to 5. It can be seen that sensitization evaluation requires only small amount of time, thus it can be used as part of an efficient high-level testability metric.

Table 5. Input sensitization analysis

Ckt	# Inputs	Single	Double	Triple	Total Time (s)
c432	36	7.2	3.2	0.9	0.02
c499	41	33.7	2.7	1.2	0.03
c880	60	37.7	2.8	1.2	0.05
c1355	41	34.2	0.6	0	0.03
c1908	33	23.1	1.8	0.6	0.03
c2670	233	138.6	3.6	0.2	0.65
c3540	50	26.3	2.1	0	0.08
c5315	178	87.4	4.2	0.5	0.65
c6288	32	32	0	0	0.12
c7552	207	85.8	36.5	47.2	1.09

5. CONCLUSION

We presented a Boolean equation-based framework to capture the sensitization conditions of a combinational circuit under the application of a test and demonstrated its application to model-free fault diagnosis and input sensitization analysis. As the constraints on the symbolic analysis are driven by initially specified constraints at primary outputs, our approach may be regarded as a symbolic version of critical path tracing. Unlike the latter, however, the equation based approach correctly handles multiple sensitizations and cancellation due to fanout reconvergence. We demonstrated a way of solving the equations using a SAT solver.

Model-free fault diagnosis could be done exactly using two-variable equations, and run time is comparable with an existing mux-based approach. The input sensitization analysis can be carried out using a simplified version of equations and can be used as the basis for high-level testability analysis. This topic is an area of our future research. Also being investigated are applications of the equation-based approach to sequential and modular logic.

ACKNOWLEDGEMENT

This work was supported by a grant from Intel Corporation. The authors would like to thank Vijay Gangaram and Yi-Shing Chang of Intel Corporation for helpful discussions.

REFERENCES

[1] J. F. Poage, "Derivation of Optimum Tests to Detect Faults in Combinational Circuits," in Symposium on Mathematical Theory of Automata, 1963, pp. 483-527.

[2] D. C. Bossen and S. J. Hong, "Cause-Effect Analysis for Multiple Fault Detection in Combinational Networks," Trans. on Computers, vol. C-20, pp. 1252-1257, 1971.

[3] M. A. Breuer, S. J. Chang, and S. Y. H. Su, "Identification of Multiple Stuck-Type Faults in Combinational Networks," Trans. on Computers, vol. C-25, pp. 44-54, 1976.

[4] T. Larrabee, "Test pattern generation using Boolean satisfiability," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 11, pp. 4-15, 1992.

[5] N. Een and N. Sorensson, "MINISAT - a SAT solver with conflict-clause minimization," in International Conference on Theory and Application of Satisfiability Testing, 2005.

[6] J. P. Marques-Silva and K. A. Sakallah, "GRASP: a search algorithm for propositional satisfiability," IEEE Trans. on Computers, vol. 48, pp. 506-521, 1999.

[7] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: engineering an efficient SAT solver," in Design Automation Conference, 2001, pp. 530-535.

[8] A. Smith, A. Veneris, M. F. Ali, and A. Viglas, "Fault diagnosis and logic debugging using Boolean satisfiability," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 24, pp. 1606-1621, 2005.

[9] L. M. Huisman, "Diagnosing arbitrary defects in logic designs using single location at a time (SLAT)," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 23, pp. 91-101, 2004.

[10] E. M. Sentovich, K. J. Singh, C. Moon, H. Savoj, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Sequential circuit design using synthesis and optimization," in International Conference on Computer Design, 1992, pp. 328-333.

[11] H. K. Lee and D. S. Ha, "HOPE: an efficient parallel fault simulator for synchronous sequential circuits," IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems, vol. 15, pp. 1048-1058, 1996.