Using SAT-Based Techniques in Test Vectors Generation

Fadi A. Aloul and Assim Sagahyroon
Department of Computer Science & Engineering, American University of Sharjah, Sharjah, UAE
Email: {faloul, asagahyroon}@aus.edu

***Abstract* - The growing size and complexity of VLSI circuits have made quality and reliability requirements increasingly stringent. The work presented in this paper investigates the application of Boolean Satisfiability (SAT)-based techniques to address two distinct VLSI testing activities, namely, test vector generation to excite stuck-open faults in CMOS circuits, and test vector generation for dynamic burn-in testing. The presence of a stuck-open fault renders an otherwise combinational logic gate sequential, therefore causing a malfunction of the integrated circuit. On the other hand, burn-in screening has been an integral part of semiconductors manufacturing to assure that reliability goals are achieved. The purpose of this type of testing is to apply to the device under test a set of input patterns which maximizes the circuits nodal activity, and by so doing causing an increase in its power dissipation that leads to device failures like electromigration and hot-carrier degradation at an early stage of the device operation.**

The search for input or test patterns to either excite a stuck-open fault, or to maximize the activity in the circuit is an NP-complete problem. In this work, we discuss the applicability of SAT methodologies in tackling these two testing problems. We experiment with SAT and Integer Linear Programming (ILP) solvers to compute solution sets for these two testing activities.

***Keywords* - Stuck-Open, Burn-in Testing, Integer Linear Programming, Boolean Satisfiability.**

I. INTRODUCTION

Advances in VLSI technology has lead to the design of complex digital systems with millions of components in a single chip. This in turn has intensified the complexity of testing such chips to verify their correct functionality and assess their long term reliability. Testing enters into the life cycle of a VLSI device in several places and for the most part, test activities are interwoven with the design of the chip. The work presented here discusses the application of Boolean Satisfiability (SAT) and Integer Linear Programming (ILP)-based techniques in two test-related activities. At first we tackle the problem of detecting stuck-open faults in CMOS circuits, followed by a look at the costly test vector generation problem for dynamic burn-in testing.

Test generation for stuck-open faults is more expensive than for single-stuck at faults because of the fact that to detect a stuck-open you need to search for a vector pair and not a single input vector. Burn-in screening has been an integral part of semiconductor manufacturing to guarantee that targeted reliability goals are achieved. However, burn-in is a major contributor to both Integrated Circuits (ICs) test cost and turn around time.

Recent strides in SAT have made it an attractive platform for solving various digital VLSI design problems. A vast body of research in the area of Boolean satisfiability (models, algorithm and solvers) with extremely encouraging results has been produced in the last few years. Even though traditionally SAT solvers have been used to solve *decision*-based problems [3, 13, 18, 21, 33], recently, these solvers have been extended to tackle Pseudo-Boolean (PB) constraints which are linear inequalities with integer coefficients [1, 7, 10, 11, 31, 32]. As a result, researchers can now use PB constraints to express *optimization* problems that are traditionally handled as ILP problems. Furthermore, PB constraints are more expressive and can be used to replace possibly a very large number of the traditional SAT input *conjunctive normal form* (CNF) constraints. We were additionally motivated by the successful application of these techniques in the electronic design automation domain, such as formal verification [6], FPGA routing [22], global routing [1], logic synthesis [20], power leakage [2], and power optimization [29].

In this paper, we show how to formulate the proposed two test-related problems as SAT instances and explore the possible advantages and limitations of using SAT techniques to solve the problems.

The paper is organized as follows, in Section 2, we present background information about the two proposed problems. In Sections 3 and 4, we discuss the formulation of the two problems using SAT techniques. In Section 5, we present the experimental results and the paper is concluded in Section 6.

II. BACKGROUND

In the following sub-sections, we briefly discuss the stuck-open fault and the burn-in testing requirements.

A. Stuck-Open Faults

Stuck-open faults are peculiar to CMOS technology; they have the adverse effect of making a combinational logic gate exhibits sequential-like behavior. To briefly review the effect of these types of faults consider the 2-input CMOS NAND Gate shown in Figure 1. Assume the existence of an open circuit condition on transistor *P1*. Now, if input vector $A = 0$,

doi:10.4304/jait.1.4.153-162

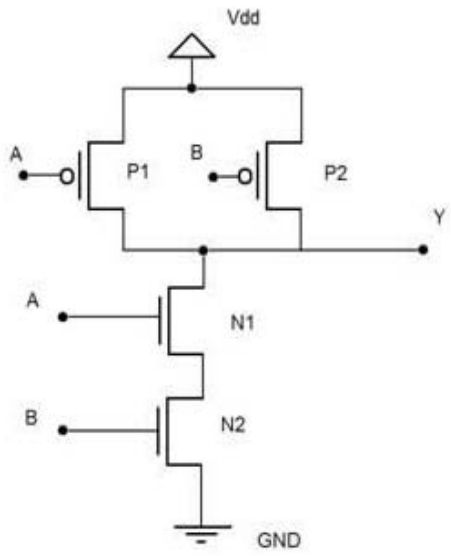


Figure 1. CMOS NAND Gate

$B = 1$ is applied, in a fault-free circuit, transistor *P1* will close thus pulling-up the output node *Y* to logic 1; however, the presence of the stuck-open fault will prevent this and the gate output (Y) will retain its previous logic state at the output node for a short duration thus leading to a sequential behavior. This makes test generation for stuck open faults much more expensive than for single stuck-at faults since a vector pair <*T1*, *T2*> is required to detect a given fault. For example, to detect the a stuck-open fault in the network of the PFET transistors in a CMOS gate, the first vector *T1* should set the gate output to logic 0. The following test vector *T2* should be selected such that it provides a path between the output and the power supply (Vdd) through the faulty transistor. For the NAND Gate in Figure 1, *T1* would be $A = 1$ and $B = 1$; followed by *T2* where $A = 0$ and $B = 1$.

It was proven that the problem of finding a two-pattern test <*T1*, *T2*> is NP-hard [23]. Several methods have been proposed to generate this type of test [8, 9, 12, 17]. However none of these methods have attempted to investigate the use of SAT techniques to search for the test vector pair. In [5], SAT was used to compute test vectors for stuck-at faults but stuck-open faults were not considered.

B. IC Burn-In Testing

Burn-in testing of Integrated Circuits (ICs) is a form of electronic testing performed under elevated temperature coupled with other stress conditions. There are four main types of burn-in tests employed in industry. They are static, dynamic, monitored and test-in burn-in. An overview of these tests is presented in [24].

Essentially, Burn-in is a production process that removes weak or low reliability ICs using high temperature and voltage stress conditions for time typically in the order of 4 to 168 hours. Burn-in is expensive and may take between 5% to 40% of product costs [14]. In dynamic burn-in testing, the design of test patterns that are able to cause the switching activity of the nodes preferably in a uniform manner in all parts of the circuit is still an open research problem. Targeting weak nodes in a circuit in order to expose their early failures is also critical for successful burn-in testing.

Hunag and others [15] discussed a methodology to generate weighted random patterns which can maximally excite a circuit during burn-in testing. Their approach is based on a probability model for switching transitions of gates and a procedure to obtaining the signal transition probability distribution of the primary inputs of the circuit. It then generates weighted random patterns according to the obtained signal probability distribution. In [28], genetic algorithms are used to generate a sequence of test vectors that seek to continuously maximize the switching activity and hence the heat dissipation in a circuit. The use of Automatic Test Pattern Generation (ATPG) during burn-in is addressed by Benso and others in [4]. The goal of their proposed ATPG is to generate test patterns that are able to force transitions into each node of a full-scan circuit to guarantee a uniform distribution of the stress during the dynamic burn-in test. Their algorithm attempts to equalize the transitions forced into the circuit in order to avoid over stressing part of the device and possible damaging it. Alternatively, other researchers explored the shortening of the burn-in test period by applying high voltage stress tests techniques [26]. The authors used the Weibull statistical analysis to model the infant mortality failure distribution. Their results indicated that, the use of these statistical analysis combined with high voltage stress testing can significantly reduce the required burn-in time.

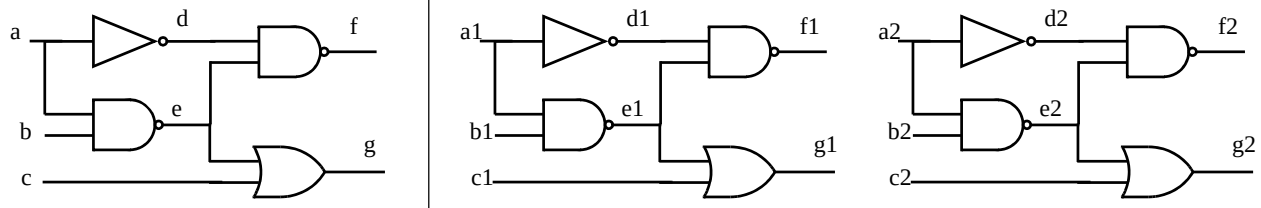
III. Stuck-Open Faults Two Test Vectors Generation Viewed as a SAT Problem

The idea here is to formulate the two vectors search as a SAT problem, and then use SAT solvers to identify two vectors <*T1*, *T2*> that would excite the maximum number of stuck-open faults in the circuit under test.

The optimization problem consists of the following set of constraints:

1. A set of clauses representing the circuit's logical behavior after the application of input vector *V1*.
2. A set of clauses representing the circuit's logical behavior after the application of input vector *V2*. Note that the set of constraints in (1) and (2) are identical but the variables are renamed differently.
3. A set of clauses representing XOR gates between the outputs of gates in (1) and (2). The number of XOR gates equals the number of gates in the original circuit. An XOR gate output of logic 1 indicates that a transition (0 to 1 or 1 to 0) has occurred at the output of the gate in the original circuit upon the successive

(a)



(b)

Circuit A Consistency Function

$(a_1 + d_1) \cdot (\overline{a_1} + \overline{d_1})$

$(a_1 + e_1) \cdot (b_1 + e_1) \cdot (\overline{a_1} + \overline{b_1} + \overline{e_1})$

$(d_1 + f_1) \cdot (e_1 + f_1) \cdot (\overline{d_1} + \overline{e_1} + \overline{f_1})$

$(\overline{c_1} + g_1) \cdot (\overline{e_1} + g_1) \cdot (c_1 + e_1 + \overline{g_1})$

Circuit B Consistency Function

$(a_2 + d_2) \cdot (\overline{a_2} + \overline{d_2})$

$(a_2 + e_2) \cdot (b_2 + e_2) \cdot (\overline{a_2} + \overline{b_2} + \overline{e_2})$

$(d_2 + f_2) \cdot (e_2 + f_2) \cdot (\overline{d_2} + \overline{e_2} + \overline{f_2})$

$(\overline{c_2} + g_2) \cdot (\overline{e_2} + g_2) \cdot (c_2 + e_2 + \overline{g_2})$

XORs on Gate Outputs

$(\overline{d_1} + d_2 + d) \cdot (d_1 + \overline{d_2} + d)$

$(\overline{d_1} + \overline{d_2} + \bar{d}) \cdot (d_1 + d_2 + \bar{d})$

$(\overline{e_1} + e_2 + e) \cdot (e_1 + \overline{e_2} + e)$

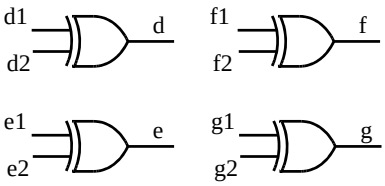
$(\overline{e_1} + \overline{e_2} + \bar{e}) \cdot (e_1 + e_2 + \bar{e})$

$(\overline{f_1} + f_2 + f) \cdot (f_1 + \overline{f_2} + f)$

$(\overline{f_1} + \overline{f_2} + \dot{f}) \cdot (f_1 + f_2 + \dot{f})$

$(\overline{g_1} + g_2 + g) \cdot (g_1 + \overline{g_2} + g)$

$(\overline{g_1} + \overline{g_2} + \bar{g}) \cdot (g_1 + g_2 + \bar{g})$



Transition Objective Function

$Maximize(d + e + f + g)$

Solution:
Max Transitions = 4
$\{a_1, b_1, c_1\} = \{1, 1, 0\}$
$\{a_2, b_2, c_2\} = \{0, 1, 1\}$

(c)

XORs on PIs

$(\overline{a_1} + a_2 + a) \cdot (a_1 + \overline{a_2} + a)$

$(\overline{a_1} + \overline{a_2} + \bar{a}) \cdot (a_1 + a_2 + \bar{a})$

$(\overline{b_1} + b_2 + b) \cdot (b_1 + \overline{b_2} + b)$

$(\overline{b_1} + \overline{b_2} + \bar{b}) \cdot (b_1 + b_2 + \bar{b})$

$(\overline{c_1} + c_2 + c) \cdot (c_1 + \overline{c_2} + c)$

$(\overline{c_1} + \overline{c_2} + \bar{c}) \cdot (c_1 + c_2 + \bar{c})$

Force at most one PI to be different between both input vectors

$a + b + c = 1$

Solution:
Max Transitions = 4
$\{a_1, b_1, c_1\} = \{0, 1, 0\}$
$\{a_2, b_2, c_2\} = \{1, 1, 0\}$

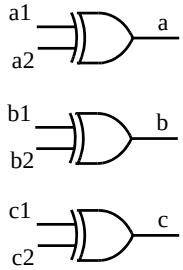


Fig. 2. An illustrative example showing how to determine the two vectors that will excite the maximum number of stuck-opens in a circuit. (a) Original circuit (b) Constraints needed to compute the input pair (c) Additional constraints when computing the input pair with the hamming distance condition imposed.

application of the vector *V1* followed by vector *V2*.

4. An objective constraint which consists of the sum of all XOR outputs.

Constraints (1) and (2) represent the circuit's logical behavior following the application of the two vectors respectively. The circuit's logical behavior is represented as a CNF formula by simply conjuncting the CNF expressions for the gate outputs in the circuit. An example of CNF expressions for simple gates is given in Table I.

Constraint (3) compares the output of the same gate for the two vectors. If a transition or a change in the output has occurred the XOR gate will produce an output of 1, else, the XOR gate output will be 0. Here also, the XOR constraint is expressed using the principles explained above. A new variable is declared for each XOR gate's output to indicate whether a transition occurred in the original circuit. Finally, the goal of the objective function in constraint (4) is to identify the two input vectors that would maximize the number of transitions in the circuit. This is expressed as a PB constraint consisting of the sum of the XOR gate's outputs. In other

TABLE I. CNF EXPRESSIONS FOR SIMPLE GATES.

Gate	CNF Expression
$z = NOT(x)$	$(x + z)(\bar{x} + \bar{z})$
$z = AND(x, y)$	$(x + \bar{z}) \cdot (y + \bar{z}) \cdot (\bar{x} + \bar{y} + z)$
$z = OR(x, y)$	$(\bar{x} + z) \cdot (\bar{y} + z) \cdot (x + y + \bar{z})$
$z = NAND(x, y)$	$(x + z) \cdot (y + z) \cdot (\bar{x} + \bar{y} + \bar{z})$
$z = XOR(x, y)$	$(\bar{x} + y + z) \cdot (x + \bar{y} + z) \cdot (\bar{x} + \bar{y} + \bar{z}) \cdot (x + y + \bar{z})$

words, this can be viewed as a constraint representing the predicate, "*there exist two input vectors that can cause a summation of gate transitions > k*" where *k* is an integer value. In the context of this work, *k* is selected to be equal or less than the total number of gates in the circuit.

Figure 2 (a, b) is an illustration of the proposed approach applied to a simple combinational circuit. The solver returned two vectors that force *all* gates to assume two different values upon the application of each vector.

A. Generating Robust Input Vector Pairs <V1, V2>

As discussed earlier, detection of a stuck-open fault in a combinational CMOS circuit requires a two-pattern test consisting of an *initialization* vector followed by a *test* vector. The second vector (i.e. the test vector) may differ in multiple bits from the initializing vector. In the presence of arbitrary delays in the circuit, all these bits may not change simultaneously, and therefore a different vector may appear temporarily during the transition from the initializing vector to the test vector. In cases like these, the desired initialization might change and the two-pattern test are said to be invalidated. To ensure robust testability the two vectors must be at only a unit Hamming distance apart, i.e. the two vectors should differ in one bit position only not multiple bit positions [9, 27].

To satisfy this requirement in terms of the hamming distance, extra constraints are added to guarantee that the two input vectors differ by a single PI value only. The constraints include:

1. A Set of clauses representing XOR gates between the primary inputs of gates in (1) and (2). The number of XOR gates equals the number of primary inputs in the original circuit. An XOR gate output of logic 1 indicates that the two vectors have different values for the same primary input.
2. A PB constraint is added to ensure that the sum of all PI-XOR gates is equal to 1.

Figure 2 (c) shows an example where the solver was successful in finding a pair that is only a unit hamming distance apart.

IV. TEST VECTORS GENERATION FOR DYNAMIC BURN-IN VIEWED AS A SAT-PROBLEM

To continuously maintain a nodal activity in the circuit it is critical to find a set or a sequence of vectors that when applied to the primary inputs of the circuit will tend to cause a switching activity in most of the gate outputs if not all of them. Stress uniformity requires that an ideal sequence is a sequence that tends to flip all the nodes. On the other hand, the ability to apply a set of vectors that tend to maximize the activity of a particularly suspected weak node(s) is also desirable.

The primary objective here is to identify a sequence of vectors $\{V_1, V_2, \ldots, V_n\}$ such that when applied to the circuit inputs, it will continuously tend to cause maximal transitional activities in all of the nodes in the circuit and therefore maximizing its heat dissipation exposing weak nodes. In this paper, the problem we try to address is the computation of such a vector set.

The idea here is to create a SAT instance for each circuit representation, with the objective function being the maximization of the nodal activity. Each circuit copy is represented as a CNF formula by simply conjuncting the CNF expressions for the gate outputs in the circuit. CNF expressions of simple gates were described in Section 3.

The sequence of steps followed to formulate the problem and develop the constraints is explained below:

1. Create a set of CNF constraints representing the circuit's logical behavior after the application of an input vector V_1 (*Circuit A*).
2. Create a set of CNF constraints representing the circuit's logical behavior after the application of input vector V_2. Note that, the set of constraints in (i) and (ii) are identical but the variables are renamed differently (*Circuit B*).
3. Create a set of CNF constraints representing the circuit's logical behavior after the application of input vector V_3, following vector V_2 (*Circuit C*).
4. .....
5. Create a set of CNF constraints representing the circuit's logical behavior after the application of input vector V_n, following vector V_{n-1} (*Circuit n*).

Next, a set of CNF constraints representing XOR gating scenarios between the outputs of gates in the different circuits is generated (for example, *Circuit A* with *Circuit B*, next *Circuit B* with *Circuit C*, etc). An XOR gate output of logic 1 (0) indicates that a toggle has (not) occurred upon the successive application of the two vectors. Finally, a PB constraint with the objective of maximizing the total transition activity is specified.

An example illustrating the above steps is shown in Figure 3. In the given example, CNF expressions representing three consistency functions for three circuit instances (*A*, *B*, and *C*) are generated. For each instance the variables were renamed differently $(a_1, a_2, a_3, b_1, b_2, b_3, \ldots)$. Similarly CNF clauses representing the XORing between gate outputs

Circuit A Consistency Function

$(\overline{a_1} + \overline{d_1}) \cdot (\overline{b_1} + \overline{d_1}) \cdot (a_1 + b_1 + d_1)$
$(c_1 + e_1) \cdot (\overline{c_1} + \overline{e_1})$
$(\overline{d_1} + f_1) \cdot (\overline{e_1} + f_1) \cdot (d_1 + e_1 + \overline{f_1})$
$(b_1 + \overline{g_1}) \cdot (e_1 + \overline{g_1}) \cdot (\overline{b_1} + \overline{e_1} + g_1)$

Circuit B Consistency Function

$(\overline{a_2} + \overline{d_2}) \cdot (\overline{b_2} + \overline{d_2}) \cdot (a_2 + b_2 + d_2)$
$(c_2 + e_2) \cdot (\overline{c_2} + \overline{e_2})$
$(\overline{d_2} + f_2) \cdot (\overline{e_2} + f_2) \cdot (d_2 + e_2 + \overline{f_2})$
$(b_2 + \overline{g_2}) \cdot (e_2 + \overline{g_2}) \cdot (\overline{b_2} + \overline{e_2} + g_2)$

Circuit C Consistency Function

$(\overline{a_3} + \overline{d_3}) \cdot (\overline{b_3} + \overline{d_3}) \cdot (a_3 + b_3 + d_3)$
$(c_3 + e_3) \cdot (\overline{c_3} + \overline{e_3})$
$(\overline{d_3} + f_3) \cdot (\overline{e_3} + f_3) \cdot (d_3 + e_3 + \overline{f_3})$
$(b_3 + \overline{g_3}) \cdot (e_3 + \overline{g_3}) \cdot (\overline{b_3} + \overline{e_3} + g_3)$

XOR Output Conditions

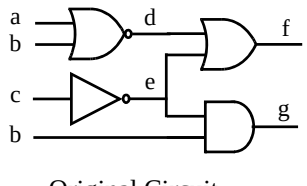
$(\overline{d_1} + d_2 + D1) \cdot (d_1 + \overline{d_2} + D1)$
$(\overline{d_1} + \overline{d_2} + \overline{D1}) \cdot (d_1 + d_2 + \overline{D1})$
$(\overline{e_1} + e_2 + E1) \cdot (e_1 + \overline{e_2} + E1)$
$(\overline{e_1} + \overline{e_2} + \overline{E1}) \cdot (e_1 + e_2 + \overline{E1})$
$(\overline{f_1} + f_2 + F1) \cdot (f_1 + \overline{f_2} + F1)$
$(\overline{f_1} + \overline{f_2} + \overline{F1}) \cdot (f_1 + f_2 + \overline{F1})$
$(\overline{g_1} + g_2 + G1) \cdot (g_1 + \overline{g_2} + G1)$
$(\overline{g_1} + \overline{g_2} + \overline{G1}) \cdot (g_1 + g_2 + \overline{G1})$
$(\overline{d_2} + d_3 + D2) \cdot (d_2 + \overline{d_3} + D2)$
$(\overline{d_2} + \overline{d_3} + \overline{D2}) \cdot (d_2 + d_3 + \overline{D2})$
$(\overline{e_2} + e_3 + E2) \cdot (e_2 + \overline{e_3} + E2)$
$(\overline{e_2} + \overline{e_3} + \overline{E2}) \cdot (e_2 + e_3 + \overline{E2})$
$(\overline{f_2} + f_3 + F2) \cdot (f_2 + \overline{f_3} + F2)$
$(\overline{f_2} + \overline{f_3} + \overline{F2}) \cdot (f_2 + f_3 + \overline{F2})$
$(\overline{g_2} + g_3 + G2) \cdot (g_2 + \overline{g_3} + G2)$
$(\overline{g_2} + \overline{g_3} + \overline{G2}) \cdot (g_2 + g_3 + \overline{G2})$

Transition Objective Function

$$Maximize \begin{pmatrix} D1 + E1 + F1 + \\ D2 + E2 + F2 + \\ G1 + G2 \end{pmatrix}$$

Solution:
Max Transitions = 6

$\{a_1, b_1, c_1\} = \{1, 1, 1\}$
$\{a_2, b_2, c_2\} = \{1, 1, 0\}$
$\{a_3, b_3, c_3\} = \{1, 1, 1\}$



Original Circuit

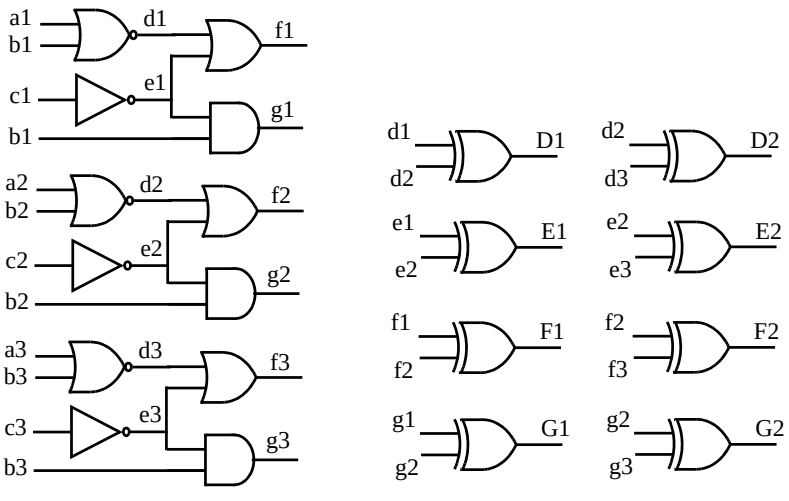


Fig. 3. An illustrative example showing how to determine the sequence of vectors that will maximize the switching activity among all nodes in the given circuit.

in the three circuits are also generated. Finally an objective function with the primary goal of maximizing the transition in the circuit upon the application of three different vectors $\{V_1, V_2, V_3\}$ is specified. In the given example, the solver returned a 3-vectors sequence that was capable of producing a total of 6 transitions in the circuit.

V. Experimental Results

A. Stuck-Open Results

Table II summarizes the results obtained using the SAT-based ILP solver PBS 4.0 [1, 25] and the commercial ILP solver CPLEX 7.0 [16]. The PBS experiments were conducted on a Pentium-IV 2.8 Ghz workstation running Linux with 500 MB of RAM. The CPLEX experiments were conducted on a SunBlade 1000 workstation with 2MB cache running SunOS 5.9. We used the default settings for PBS and CPLEX. We used the MCNC [19] benchmark circuits. Each benchmark was sensitized using "sis" [30] into a circuit consisting of 2-input NAND, NOR and NOT gates. The runtime was set to a limit of 1,000 seconds. Note that both solvers perform a complete search, i.e. if an optimal solution is found, no other test pair exists that can excite a larger number of stuck-open faults.

In Table II, the circuit name is provided in the first column, the number of primary inputs and the number of gates are given in columns two and three, respectively. The *Time* column indicates the runtime (in seconds) for each solver. The *MaxExcited* columns represents the total number of gates for which the solver succeeded in exciting their stuck-open faults. The*%-excited* is the percentage of gates excited in proportion to the total number of gates in the circuit.

We run the solvers once without the hamming distance constraint and then with the constraint imposed. From the results it is clear that, when the constraint is removed both PBS and CPLEX are able to excite a higher percentage of faults, however, as explained earlier, the test pairs can be invalidated. Overall the performance of the SAT-based ILP solver, PBS, is better than the generic ILP solver, CPLEX, when we consider both, run time and percentage of faults excited. Finally, note that in some instances the solvers returned close to 75% excitation (with the hamming distance constraint).

B. Burn-In Test Results

For the sake of brevity, a subset consisting of sixteen circuits with varying sizes from the MCNC suite of benchmarks [19] is selected to test the proposed approach. In all cases the SAT-based 0-1 ILP MiniSAT+ [11] solver was used. The experiments were run on a Intel Xeon 3 Ghz station running Linux and equipped with 4 GB of RAM. Utilizing different sets of constraints, the following scenarios were assessed:

1. A search for *n* vectors with all nodes having similar weights.
2. Same as in the above step, but adding constraints ensuring that each node will flip at least once.
3. Modification of the objective function to allow the search to target a particular weak node and find vectors that continuously cause activity at this node - in the given illustration example (Figure 3), if node *d*, for example, is selected, this can be achieved by modifying the objective function to be maximize $(D_1 + D_2)$.

Results for the above scenarios are listed in Tables III, IV, V respectively. Columns 1, 2, and 3 of the tables list the name of circuit, number of primary inputs and the total number of gates in the circuit, respectively. We incrementally increased the number of consecutively generated vectors from 2 up to 7 vectors. A time-out limit of 1,000 seconds is set for all the experiments.

In Table III (all nodes have equal preference), as expected, the time needed to search for the vectors increases with *n*, where *n* is the number of vectors. The *Value* column is the best objective value (number of transitions) returned by the solver. The *Percent* (%) column shows the percentage of the actual activity attained when the vectors are applied relative to the theoretical upper bound where we assume all nodes in the circuit will toggle, i.e. $Percentage = (Value/\text{upper bound}) \times 100$. The *upper bound* is computed by multiplying the number of gates in the circuit by $(n-1)$.

From Table III, the search, in few cases was able to find a reasonably high objective function in a short amount of time. In the case of circuits *i2*, *i4*, and *i5* which are relatively large circuits, the search computed a sequence that had an above 90% value function. Interestingly, for some smaller circuits, the search failed to find a useful sequence. For example, circuit *alu2* with only ten inputs, the search either timed-out or returned a low value. In other cases, such as *count*, it was clear that the best possible sequence is only within a 71% of the maximum possible switching value, hence since the search is complete, there is no need to look for any sequence that will reveal a higher percentage.

In Table IV (constraints are added to ensure that each node toggles at least once), when a short sequence is requested, we notice that a number of instances where *unsatisfiable*, i.e. no possible sequence exists, however, as *n* increases several instances became *satisfiable*. The added constraints should not affect the overall complexity of the problem, since the number of variables, *v*, is still the same, i.e. the problem complexity is 2^v. However, in general, adding constraints to a satisfiable problem, reduces the possible number of solutions, making it difficult for a SAT solver to find a solution in a reasonable time. In the same way, adding constraints to an unsatisfiable problem, eliminates parts of the search space, making it easier for a SAT solver to complete the search in less time. This is clearly seen in Table 4, where satisfiable problems became harder to solve and unsatisfiable problems became easier to solve. Some circuits continued to be unsatisfiable regardless of *n*. It is important to note that in some of these cases the topology of the circuit has an impact on the toggling activity achieved. For example, in Table IV, it might not be possible to find a sequence that maximizes the activity beyond what the search has found, simply because it is not possible and a sequence does not exist.

The results of Table V are generated by randomly selecting a node (assuming it is a weak node that needs to be

TABLE II. RESULTS FOR THE STUCK-OPEN EXPERIMENT USING THE SAT-BASED 0-1 ILP SOLVER PBS AND THE GENERIC ILP SOLVER CPLEX.

Circuit Name	#PI	#Gates	Without Hamming Distance Constraint						With Hamming Distance Constraint					
			PBS			CPLEX			PBS			CPLEX		
			Time	MaxExcited	%-Excited	Time	MaxExcited	%-Excited	Time	MaxExcited	%-Excited	Time	MaxExcited	%-Excited
pm1	16	67	0.03	55	82.1	0.86	55	82.1	0.03	20	29.9	4.46	20	29.9
pcle	19	71	0.69	48	67.6	0.66	48	67.6	0.02	45	63.4	1.03	45	63.4
x2	10	73	0.01	54	74.0	0.95	54	74.0	0.02	33	45.2	1.98	33	45.2
parity	16	75	732	50	66.7	68.66	50	66.7	0.04	12	16.0	1000	12	16.0
cu	14	78	0.2	52	66.7	1.49	52	66.7	0.03	25	32.1	3.44	25	32.1
cc	21	79	0.24	64	81.0	1.11	64	81.0	0.05	46	58.2	4.5	46	58.2
cm150a	21	79	0.01	77	97.5	0.08	77	97.5	0.02	60	75.9	2.86	60	75.9
pcler8	27	104	7.18	74	71.2	1.27	74	71.2	0.05	54	51.9	3.03	54	51.9
mux	21	106	0.09	98	92.5	1.29	98	92.5	0.09	64	60.4	35.95	64	60.4
i3	132	132	0.01	132	100.0	0.04	132	100.0	0.88	6	4.5	1000	6	4.5
frg1	28	143	0.02	140	97.9	1.24	140	97.9	0.09	53	37.1	1000	53	37.1
b9	41	147	0.27	130	88.4	2.96	130	88.4	0.07	57	38.8	1000	57	38.8
f51m	8	150	0.1	115	76.7	6.04	115	76.7	0.11	66	44.0	14.62	66	44.0
comp	32	178	1000	118	66.3	1000	123	69.1	0.26	28	15.7	1000	28	15.7
lal	26	179	4.09	157	87.7	9.78	157	87.7	0.08	67	37.4	673	67	37.4
c8	28	211	62.51	169	80.1	13.36	169	80.1	0.17	66	31.3	1000	66	31.3
my_adder	33	225	1000	154	68.4	1000	161	71.6	0.31	85	37.8	1000	84	37.3
i2	201	242	0.02	238	98.3	2.4	238	98.3	5.71	17	7.0	1000	16	6.6
9symml	9	252	59.04	148	58.7	259	148	58.7	1.64	66	26.2	279.3	66	26.2
C432	36	282	94.17	251	89.0	12.87	251	89.0	53.5	151	53.5	1000	141	50.0
i4	192	308	0.01	308	100.0	0.11	308	100.0	2.46	17	5.5	1000	14	4.5
i5	133	445	0.04	445	100.0	0.18	445	100.0	1.86	222	49.9	1000	70	15.7
alu2	10	462	51.9	238	51.5	1000	224	48.5	3.07	169	36.6	1000	156	33.8
term1	34	525	1000	390	74.3	489	409	77.9	2.22	175	33.3	1000	164	31.2
C1355	41	552	1000	277	50.2	1000	275	49.8	113	107	19.4	1000	68	12.3
C499	41	567	1000	312	55.0	1000	307	54.1	120	99	17.5	1000	74	13.1
apex6	135	803	1000	511	63.6	1000	557	69.4	23.35	175	21.8	1000	151	18.8
alu4	14	878	1000	430	49.0	1000	392	44.6	17.16	329	37.5	1000	236	26.9
too_large	38	1071	1000	707	66.0	1000	726	67.8	30.14	273	25.5	1000	201	18.8
vda	17	1417	519	400	28.2	1000	321	22.7	46.42	235	16.6	1000	226	15.9

stressed) in each circuit. The optimization objective was modified to maximize the switching activity of this particular node. We run a search for a 7-vector sequence and the results clearly show that in each and every instance the solver was capable of generating a sequence that succeeded in toggling the node the maximum number of possible times with 7 vectors which is 6 toggles. Furthermore, the time it took the search to find the vector sequence was almost insignificant.

Table VI shows the results of comparing the performance of the SAT-based 0-1 ILP solver MiniSAT+ to the generic commercial ILP solver, CPLEX [16] when solving the proposed problem. Obtained results show the superiority of the SAT-based solver over CPLEX in most instances. Given the black-box nature of CPLEX, it was hard to justify its lower performance on the tested instances.

VI. CONCLUSIONS

Recent years have seen a substantial increase in the use of Boolean Satisfiability (SAT)-based techniques and tools in successfully contributing to the solution of electronic design automation problems. The main contribution of the work discussed in this paper is to explore the possible advantages and the likely shortcomings of using SAT and ILP techniques to solve the test generation problem for two distinct cases. Specifically:

- We formulated both presented problems as SAT 0-1 ILP instances, and
- Tested the formulation with advanced SAT and ILP techniques using circuits that vary in size and complexity while making use of advanced solvers to search for test vectors.

For stuck-open faults, the proposed methodology provided promising results. The number of excited faults has ex-

TABLE III. RESULTS FOR THE BURN-IN EXPERIMENT USING THE MCNC BENCHMARK SET. A SAMPLE OF 16 INSTANCES ARE SHOWN. ALL NODES HAVE EQUIVALENT WEIGHTS. TIME REPRESENTS THE TIME NEEDED IN SECONDS BY MINISAT+ TO SOLVE THE INSTANCES. VALUE IS THE BEST OBJECTIVE VALUE FOUND BY MINISAT+. % IS THE PERCENTAGE OF THE ACTIVITY REPORTED BY THE SOLVER (VALUE) RELATIVE TO THE MAXIMUM POSSIBLE UPPER BOUND.

Name	#	#	2 Vectors			3 Vectors			4 Vectors			5 Vectors			6 Vectors			7 Vectors		
	PI	Gates	Time	Value	%	Time	Value	%	Time	Value	%	Time	Value	%	Time	Value	%	Time	Value	%
unreg	36	145	0.22	113	78	1.88	226	78	4.8	339	78	16.89	452	78	21.86	565	78	33.92	678	78
count	35	161	0.23	114	71	1.68	228	71	4.78	342	71	12.22	456	71	26.41	570	71	43.78	684	71
lal	26	179	0.2	157	88	1.49	314	88	9.94	471	88	14.08	628	88	76.12	785	88	187.4	942	88
i2	201	242	0.05	238	98	0.1	476	98	0.51	714	98	1.26	952	98	2.13	1190	98	3.43	1428	98
cht	47	249	0.12	236	95	0.79	472	95	1.53	708	95	3.34	944	95	7.45	1180	95	13.56	1416	95
C432	36	282	0.35	251	89	3.71	502	89	11.42	753	89	49.6	1004	89	89.8	1255	89	168	1506	89
i4	192	308	0.06	308	100	0.09	616	100	0.11	924	100	0.23	1232	100	0.69	1540	100	2.38	1848	100
ex2	85	351	6.1	242	69	41.8	484	69	233	726	69	758	968	69	>1K	1103	63	>1K	951	45
i5	133	445	0.05	445	100	0.22	890	100	0.36	1335	100	0.28	1780	100	>1K	1134	51	>1K	1348	50
alu2	10	462	49.4	238	52	>1K	451	49	>1K	649	47	>1K	890	48	>1K	841	36	>1K	938	34
term1	34	525	93.6	409	78	>1K	798	76	>1K	1169	74	>1K	892	42	>1K	1119	43	>1K	1384	44
x4	94	635	202	501	79	>1K	985	78	>1K	1448	76	>1K	1186	47	>1K	1487	47	>1K	1658	44
i6	138	764	4.27	559	73	28.8	1118	73	>1K	1077	47	>1K	1542	50	>1K	1670	44	>1K	2054	45
i7	199	1011	15.5	673	67	37.7	1346	67	>1K	1393	46	>1K	1597	39	>1K	2052	41	>1K	2610	43
i9	88	1218	38	778	64	>1K	1254	51	>1K	1731	47	>1K	2053	42	>1K	2789	46	>1K	2906	40
vda	17	1417	192	400	28	>1K	638	23	>1K	900	21	>1K	1151	20	>1K	1418	20	>1K	1691	20

TABLE IV. RESULTS FOR THE BURN-IN EXPERIMENT USING THE MCNC BENCHMARK SET. A SAMPLE OF 16 INSTANCES ARE SHOWN. ALL NODES HAVE EQUIVALENT WEIGHTS. AN EXTRA CONDITION IS ADDED THAT ENSURES THAT EACH NODE SWITCHES AT LEAST ONCE. TIME REPRESENTS THE TIME NEEDED IN SECONDS BY MINISAT+ TO SOLVE THE INSTANCES. VALUE IS THE BEST OBJECTIVE VALUE FOUND BY MINISAT+. % IS THE PERCENTAGE OF THE ACTIVITY REPORTED BY THE SOLVER (VALUE) RELATIVE TO THE MAXIMUM POSSIBLE UPPER BOUND.

Name	#	#	3 Vectors			4 Vectors			5 Vectors			6 Vectors			7 Vectors		
	PI	Gates	Time	Value	%	Time	Value	%	Time	Value	%	Time	Value	%	Time	Value	%
unreg	36	145	0.24	207	71	28.56	323	74	630	436	75	>1K	547	75	>1K	655	75
count	35	161	0.01	uns		0.03	uns		0.03	uns		0.06	uns		0.13	uns	
lal	26	179	0.03	uns		22.81	418	78	173.1	601	84	613.6	757	85	>1K	887	83
i2	201	242	0.19	473	98	1.05	711	98	1.76	949	98	3.73	1187	98	6.98	1425	98
cht	47	249	0.03	uns		9.49	662	89	10.79	896	90	37.5	1134	91	37.4	1368	92
C432	36	282	0.04	uns		687.4	669	79	>1K	896	79	>1K	1143	81	>1K	1390	82
i4	192	308	0.28	616	100	0.85	924	100	0.98	1232	100	2.24	1540	100	1.91	1848	100
ex2	85	351	0.04	uns		0.06	uns		0.08	uns		0.07	uns		0.5	uns	
i5	133	445	0.62	890	100	2.07	1335	100	2.17	1780	100	>1K	1193	54	>1K	1413	53
alu2	10	462	0.05	uns		0.07	uns		0.11	uns		0.31	uns		1.26	uns	
term1	34	525	0.05	uns		0.09	uns		0.1	uns		0.19	uns		0.76	uns	
x4	94	635	0.06	uns		0.07	uns		0.19	uns		>1K	1582	50	>1K	2067	54
i6	138	764	0.1	uns		0.12	uns		>1K	1838	60	>1K	2180	57	>1K	2410	53
i7	199	1011	0.07	uns		0.17	uns		0.41	uns		>1K	2563	51	>1K	2972	49
i9	88	1218	0.14	uns		0.18	uns		0.29	uns		0.36	uns		0.33	uns	
vda	17	1417	0.19	uns		0.27	uns		0.41	uns		0.96	uns		3.22	uns	

ceeded the 90% in some cases when the hamming distance constraint is ignored. The fault-excitement step discussed in this work can be integrated with and used as part of a complete test environment where fault simulation and fault propagation are implemented as well. It is worth mentioning that this approach is complete and the solvers definitely return the best vector pair reachable by the search.

For the burn-in case, experimental results indicate that in some cases the proposed approach can find a set of vectors that significantly increase the switching activity of a circuit during burn-in in a reasonable amount of time. This can contribute significantly in reducing test time cost. In the case of vector generation to target a specific node, the approach had superior results in all cases. Using random vector generation to exercise a particular node can be very expensive a fact that makes the proposed approach desirable and practical. Finally, the performance of SAT-based 0-1 ILP solvers was compared against generic ILP solvers, namely CPLEX, when solving the proposed problem and it was clear that SAT-based solvers outperform generic ILP solvers for the proposed problem.

REFERENCES

[1] F. Aloul, A. Ramani, I. Markov, and K. Sakallah, “Generic ILP versus Specialized 0-1 ILP: An Update,” in *Proc. of the Inter-*

TABLE VI. RESULTS FOR THE BURN-IN EXPERIMENT, COMPARING THE PERFORMANCE OF MINISAT+ AND CPLEX. ALL NODES HAVE EQUIVALENT WEIGHTS.

Name	# PI	# Gates	2Vectors				7 Vectors			
			MiniSAT+		CPLEX		MiniSAT+		CPLEX	
			Time	Value	Time	Value	Time	Value	Time	Value
unreg	36	145	**0.22**	113	1.39	113	**33.9**	678	>1K	659
count	35	161	**0.23**	114	1.72	114	**43.8**	684	>1K	664
lal	26	179	**0.2**	157	3.19	157	**187**	942	505	942
i2	201	242	**0.05**	238	2.02	238	**3.43**	1428	8.04	1428
cht	47	249	**0.12**	236	2.33	236	**13.5**	1416	41.7	1416
C432	36	282	**0.35**	251	7.92	251	**168**	1506	>1K	1402
i4	192	308	**0.06**	308	**0.06**	308	2.38	1848	**0.37**	1848
ex2	85	351	**6.1**	242	28.9	242	>1K	951	>1K	1170
i5	133	445	**0.05**	445	0.11	445	>1K	1348	**0.74**	2670
alu2	10	462	**49.4**	238	491.5	238	>1K	938	>1K	1117
term1	34	525	**93.6**	409	311.8	409	>1K	1384	>1K	1820
x4	94	635	**202**	501	>1K	497	>1K	1658	>1K	2660
i6	138	764	4.27	559	**3.81**	559	>1K	2054	>1K	3041
i7	199	1011	15.5	673	**7.37**	673	>1K	2610	>1K	3351
i9	88	1218	**38**	778	44.2	778	>1K	2906	>1K	3611
vda	17	1417	**192**	400	681	400	>1K	1691	>1K	1579
Total			**602**	**5662**	**>2588**	**5658**	**>9K**	**24K**	**>11K**	**29K**

TABLE V. RESULTS FOR THE BURN-IN EXPERIMENT USING MINISAT+ WHEN A WEAK NODE IS SELECTED AND TARGETED FOR ACTIVITY.

Name	# PI	# Gates	7 Vectors	
			Time	Value
unreg	36	145	0.02	6
count	35	161	0.05	6
lal	26	179	0.04	6
i2	201	242	0.09	6
cht	47	249	0.08	6
C432	36	282	0.09	6
i4	192	308	0.09	6
ex2	85	351	0.11	6
i5	133	445	0.17	6
alu2	10	462	0.11	6
term1	34	525	0.17	6
x4	94	635	0.16	6
i6	138	764	0.24	6
i7	199	1011	0.33	6
i9	88	1218	0.35	6
vda	17	1417	0.4	6

national Conference on Computer-Aided Design (ICCAD), pp. 450-457, 2002.

[2] F. Aloul, S. Hassoun, K. Sakallah, and D. Blaauw, "Robust SAT-Based Search Algorithm for Leakage Power Reduction," in *Proc. of the Int'l Workshop on Power and Timing Modeling, Optimization and Simulation* (PATMOS), pp. 167-177, 2002.

[3] R. J. Bayardo Jr. and R. C. Schrag, "Using CSP Look-Back Techniques to Solve Real World SAT Instances," in *Proc. of the 14th National Conference on Artificial Intelligence* (AAAI), pp. 203-208, 1997.

[4] A. Benso, A. Bosio, S. Carlo, G. Natale, and P. Prinetto, "ATPG for Dynamic Burn-In Test in Full-Scan Circuit," in *Proc. of the IEEE 15th Asian Test Symposium*, pp. 75-82, 2006.

[5] S. Brayton and A. Sangiovanni-Vincentelli, "Combinational Test Generation Using Satisfiability," in *IEEE Transactions on CAD*, 15(9), pp. 1167- 1176, 1996.

[6] A. Biere, A. Cimatti, E. Clarke, M. Fujita, and Y. Zhu, "Symbolic Model Checking using SAT procedures instead of BDDs," in *Proc. of the Design Automation Conference* (DAC), pp. 317-320, 1999.

[7] D. Chai and A. Kuehlmann, "A Fast Pseudo-Boolean Constraint Solver," in *Proc. of the Design Automation Conference* (DAC), pp. 830-835, 2003.

[8] H. Ching and J. Abraham, "Transistor Level Test Generation for Physical Failures in CMOS Circuits," in *Proc. of the Design Automation Conference* (DAC), pp. 243-249, 1986.

[9] D. Das, S. Chakraborty, and B. Bhattacharya, "Universal and Robust Testing of Stuck-Open Faults in Reed-Muller Canonical CMOS Circuits," in *International Journal of Electronics*, 90 (1), pp. 1-11, 2003.

[10] H. Dixon and M. Ginsberg, "Inference Methods for a Pseudo-Boolean Satisfiability Solver," in *Proc. of the National Conference on Artificial Intelligence* (AAAI), pp. 635-640, 2002.

[11] N. Een and N. Sorensson, "An Extensible SAT-Solver," in *Proc. of the Int'l Conf. on Theory and Applications of Satisfiability Testing*, pp. 502-508, 2003.

[12] Y. Elziq, "Automatic Test Generation of Stuck-Open Faults in CMOS VLSI," in *Proc. of the Design Automation Conference* (DAC), pp. 347-354, 1981.

[13] E. Goldberg and Y. Novikov, "BerkMin: A Fast and Robust SAT-Solver," in *Proc. of the Design Automation and Test Conference in Europe* (DATE), pp. 142-149, 2002.

[14] C. Hawkins, J. Sequra, J. Soden, and T. Dellin, "Test and Reliability: Partners in IC Manufacturing, Part 2," in *IEEE Design and Test of Computers*, 16(4), pp. 66-73, Oct.-Dec. 1999.

[15] K. Huang, C. Lee, and J. Chen, "Maximization of Power Dissipation Under Random Excitation for Burn-In Testing," in *Proc. of the IEEE International Test Conference*, pp. 567-576, 1998.

[16] ILOG CPLEX, *http://www.ilog.com/products/cplex*

[17] N. Jha, "Multiple Stuck-Open Fault Detection in CMOS Logic

Circuits," in *IEEE Transaction on Computers*, 37(4), pp. 426-432, 1988.
[18] J. Marques-Silva and K. Sakallah, "GRASP: A Search Algorithm for Propositional Satisfiability," in *IEEE Transactions on Computers*, 48(5), pp. 506-521, 1999.
[19] MCNC Benchmarks, *http://www.cbl.ncsu.edu:16080/benchmarks/*
[20] S. Memik and F. Fallah, "Accelerated Boolean Satisfiability-Based Scheduling of Control/Data Flow Graphs for High-Level Synthesis," in *Proc. of the International Conference on Computer Design* (ICCD), 2002.
[21] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an Efficient SAT Solver", in *Proc. of the Design Automation Conference* (DAC), pp. 530-535, 2001.
[22] G. Nam, F. A. Aloul, K. A. Sakallah, and R. Rutenbar, "A Comparative Study of Two Boolean Formulations of FPGA Detailed Routing Constraints", in *Proc. of the International Symposium on Physical Design* (ISPD), pp. 222-227, 2001.
[23] F. Najm and I. Hajj, "The Complexity of Fault Detection in MOS VLSI Circuits," in *IEEE Transactions on CAD*, 9(4), pp. 995-1001, 1990.
[24] Y. Ng, Y. Hock Low, and S. Demidenko, "Improving Efficiency of IC Burn-In Testing," in *Proc. of the IEEE International Instrumentation and Measurement Technology Conference*, 2008.
[25] PBS version 4, *http://www.eecs.umich.edu/~faloul/Tools/pbs4*
[26] M. Ooi, Z. Kassim, and S. Demidenko, "Shortening Burn-In Test: Application of HVST and Weibull Statistical Analysis," in *IEEE Transactions on Instrumentation and Measurement*, 56(3), pp. 990-999, 2007.
[27] S. Reddy and M. Reddy, "Testable Realization on FET Stuck-Open Faults in CMOS Combinational Logic Circuits," in *IEEE Transactions on Computers*, 35(8), pp. 742-754, 1986.
[28] A. Sagahyroon, "Maximizing Heat Dissipation for Burn-In Testing," in *Proc. of the Canadian Conference on Electrical and Computer Engineering*, v. 1, pp. 399-402, 2002.
[29] A. Sagahyroon and F. Aloul, "Using SAT-Based Techniques in Power Estimation," in *Microelectronics Journal*, vol. 38, issues 6-7, pp. 706-715, 2007.
[30] E. Sentovich, K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli, "SIS: A System for Sequential Circuit Synthesis," *Univ of California-Berkeley, UCB/ERL M92/41*, 1992.
[31] H. Sheini and K. Sakallah, "Pueblo: A Modern Pseudo-Boolean SAT Solver," in *Proc. of the Design, Automation, and Test Conference in Europe* (DATE), vol. 2, pp. 684-685, 2005.
[32] J. Whittemore, J. Kim, and K. Sakallah, "SATIRE: A New Incremental Satisfiability Engine," in *Proc. of Design Automation Conference* (DAC), pp. 542-545, 2001.
[33] H. Zhang, "SATO: An Efficient Propositional Prover," in *Proc. of the International Conference on Automated Deduction*, pp. 272-275, 1997.