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 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, 31], recently, these solvers have been extended to tackle Pseudo-Boolean (PB) constraints which are linear inequalities with integer coefficients [1, 7, 10, 11, 29, 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 \( A = 0 \),
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 $<T_1, T_2>$ 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 $V_1$.
2. A set of clauses representing the circuit’s logical behavior after the application of input vector $V_2$. 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
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.

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

Constraints (1) and (2) represent the circuit’s logical behavior followed by vector V1 and V2.

Application of the vector V1 followed by vector V2.

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

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
A. Generating Robust Input Vector Pairs \(<V_1, V_2>\)
Upon the application of each vector.

Bits from the initializing vector. In the presence of arbitrary test vector. In cases like these, the desired initialization might necessarily, and therefore a different vector may appear temporarily, and extra constraints are added to guarantee that the two inputs differ by a single PI value only. The constraints to ensure robust testability the two vectors must be at only a unit change and the two-pattern test are said to be invalidated. To maximize 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

<table>
<thead>
<tr>
<th>Gate</th>
<th>CNF Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>(z = \text{NOT}(x))</td>
<td>((x + z)(\bar{x} + z))</td>
</tr>
<tr>
<td>(z = \text{AND}(x, y))</td>
<td>((x + z)(y + z)(\bar{x} + \bar{y} + z))</td>
</tr>
<tr>
<td>(z = \text{OR}(x, y))</td>
<td>((x + z)(y + z)(\bar{x} + y + \bar{z}))</td>
</tr>
<tr>
<td>(z = \text{NAND}(x, y))</td>
<td>((x + z)(y + z)(\bar{x} + y + \bar{z}))</td>
</tr>
<tr>
<td>(z = \text{XOR}(x, y))</td>
<td>((\bar{x} + y + z)(\bar{x} + \bar{y} + z))</td>
</tr>
</tbody>
</table>

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 \(<V_1, V_2>\)
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.

<table>
<thead>
<tr>
<th>Gate</th>
<th>CNF Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>(z = \text{NOT}(x))</td>
<td>((x + z)(\bar{x} + z))</td>
</tr>
<tr>
<td>(z = \text{AND}(x, y))</td>
<td>((x + z)(y + z)(\bar{x} + \bar{y} + z))</td>
</tr>
<tr>
<td>(z = \text{OR}(x, y))</td>
<td>((x + z)(\bar{x} + y + \bar{z}))</td>
</tr>
<tr>
<td>(z = \text{NAND}(x, y))</td>
<td>((x + z)(y + z)(\bar{x} + y + \bar{z}))</td>
</tr>
<tr>
<td>(z = \text{XOR}(x, y))</td>
<td>((\bar{x} + y + z)(\bar{x} + \bar{y} + z))</td>
</tr>
</tbody>
</table>
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.

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.
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 10000 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 represent 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 an Intel Xeon 3 Gzh 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. \[ \text{Percentage} = \frac{\text{Value}}{\text{upper bound}} \times 100 \]. 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
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 PBS and the generic ILP solver CPLEX. 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.

<table>
<thead>
<tr>
<th>Circuit Name</th>
<th>#PI</th>
<th>#Gates</th>
<th>Without Hamming Distance Constraint</th>
<th>With Hamming Distance Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>PBS</td>
<td>CPLEX</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Time</td>
<td>MaxExcited</td>
</tr>
<tr>
<td>pm1</td>
<td>16</td>
<td>67</td>
<td>0.03</td>
<td>55</td>
</tr>
<tr>
<td>pcle</td>
<td>19</td>
<td>71</td>
<td>0.69</td>
<td>48</td>
</tr>
<tr>
<td>x2</td>
<td>10</td>
<td>73</td>
<td>0.01</td>
<td>54</td>
</tr>
<tr>
<td>parity</td>
<td>16</td>
<td>75</td>
<td>0.23</td>
<td>52</td>
</tr>
<tr>
<td>cu</td>
<td>14</td>
<td>78</td>
<td>0.25</td>
<td>52</td>
</tr>
<tr>
<td>cc</td>
<td>21</td>
<td>79</td>
<td>0.24</td>
<td>64</td>
</tr>
<tr>
<td>cm150a</td>
<td>21</td>
<td>79</td>
<td>0.01</td>
<td>77</td>
</tr>
<tr>
<td>pcier8</td>
<td>27</td>
<td>104</td>
<td>0.72</td>
<td>54</td>
</tr>
<tr>
<td>mux</td>
<td>33</td>
<td>152</td>
<td>0.09</td>
<td>54</td>
</tr>
<tr>
<td>j3</td>
<td>132</td>
<td>132</td>
<td>0.00</td>
<td>74</td>
</tr>
<tr>
<td>frg1</td>
<td>28</td>
<td>143</td>
<td>0.02</td>
<td>140</td>
</tr>
<tr>
<td>f51m</td>
<td>8</td>
<td>150</td>
<td>0.1</td>
<td>115</td>
</tr>
<tr>
<td>comp</td>
<td>32</td>
<td>178</td>
<td>0.00</td>
<td>109</td>
</tr>
<tr>
<td>hal</td>
<td>26</td>
<td>179</td>
<td>0.01</td>
<td>179</td>
</tr>
<tr>
<td>c8</td>
<td>28</td>
<td>211</td>
<td>32.81</td>
<td>169</td>
</tr>
<tr>
<td>my_adder</td>
<td>33</td>
<td>255</td>
<td>0.00</td>
<td>154</td>
</tr>
<tr>
<td>j2</td>
<td>92</td>
<td>201</td>
<td>0.02</td>
<td>238</td>
</tr>
<tr>
<td>syml</td>
<td>9</td>
<td>252</td>
<td>59.04</td>
<td>148</td>
</tr>
<tr>
<td>C432</td>
<td>36</td>
<td>282</td>
<td>94.17</td>
<td>251</td>
</tr>
<tr>
<td>j4</td>
<td>92</td>
<td>308</td>
<td>0.01</td>
<td>308</td>
</tr>
<tr>
<td>j5</td>
<td>133</td>
<td>445</td>
<td>0.04</td>
<td>445</td>
</tr>
<tr>
<td>alu2</td>
<td>10</td>
<td>462</td>
<td>51.9</td>
<td>238</td>
</tr>
<tr>
<td>term1</td>
<td>34</td>
<td>525</td>
<td>000</td>
<td>390</td>
</tr>
<tr>
<td>C1355</td>
<td>41</td>
<td>552</td>
<td>000</td>
<td>277</td>
</tr>
<tr>
<td>C499</td>
<td>41</td>
<td>567</td>
<td>000</td>
<td>312</td>
</tr>
<tr>
<td>apex6</td>
<td>135</td>
<td>803</td>
<td>000</td>
<td>511</td>
</tr>
<tr>
<td>alu4</td>
<td>41</td>
<td>878</td>
<td>000</td>
<td>430</td>
</tr>
<tr>
<td>too_large</td>
<td>38</td>
<td>1071</td>
<td>000</td>
<td>707</td>
</tr>
<tr>
<td>vda</td>
<td>17</td>
<td>1417</td>
<td>519</td>
<td>400</td>
</tr>
</tbody>
</table>

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-

© 2010 ACADEMY PUBLISHER
cceeded 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

### TABLE VI. Results for the Burn-In Experiment, Comparing the Performance of MiniSAT+ and CPLEX. All nodes have equivalent weights.

<table>
<thead>
<tr>
<th>Name</th>
<th>PI</th>
<th>Gates</th>
<th>2 Vectors</th>
<th>7 Vectors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>MiniSAT+</td>
<td>CPLEX</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Time</td>
<td>Value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Time</td>
<td>Value</td>
</tr>
<tr>
<td>unreg</td>
<td>36</td>
<td>145</td>
<td>0.22</td>
<td>113</td>
</tr>
<tr>
<td>count</td>
<td>35</td>
<td>161</td>
<td>0.23</td>
<td>114</td>
</tr>
<tr>
<td>lal</td>
<td>26</td>
<td>179</td>
<td>0.2</td>
<td>157</td>
</tr>
<tr>
<td>i2</td>
<td>201</td>
<td>242</td>
<td>0.05</td>
<td>238</td>
</tr>
<tr>
<td>cht</td>
<td>47</td>
<td>249</td>
<td>0.12</td>
<td>236</td>
</tr>
<tr>
<td>C432</td>
<td>36</td>
<td>282</td>
<td>0.35</td>
<td>251</td>
</tr>
<tr>
<td>i4</td>
<td>192</td>
<td>308</td>
<td>0.06</td>
<td>308</td>
</tr>
<tr>
<td>ex2</td>
<td>85</td>
<td>351</td>
<td>6.1</td>
<td>242</td>
</tr>
<tr>
<td>i5</td>
<td>133</td>
<td>445</td>
<td>0.05</td>
<td>445</td>
</tr>
<tr>
<td>alu2</td>
<td>10</td>
<td>462</td>
<td>49.4</td>
<td>238</td>
</tr>
<tr>
<td>term1</td>
<td>34</td>
<td>525</td>
<td>93.6</td>
<td>409</td>
</tr>
<tr>
<td>x4</td>
<td>94</td>
<td>635</td>
<td>202</td>
<td>501</td>
</tr>
<tr>
<td>i6</td>
<td>138</td>
<td>764</td>
<td>4.27</td>
<td>559</td>
</tr>
<tr>
<td>i7</td>
<td>199</td>
<td>1011</td>
<td>15.5</td>
<td>673</td>
</tr>
<tr>
<td>i9</td>
<td>88</td>
<td>1218</td>
<td>38</td>
<td>778</td>
</tr>
<tr>
<td>vda</td>
<td>17</td>
<td>1417</td>
<td>192</td>
<td>400</td>
</tr>
<tr>
<td>total</td>
<td></td>
<td></td>
<td>602</td>
<td>5662</td>
</tr>
</tbody>
</table>

### TABLE V. Results for the Burn-In Experiment Using MiniSAT+ when a weak node is selected and targeted for activity.

<table>
<thead>
<tr>
<th>Name</th>
<th>PI</th>
<th>Gates</th>
<th>2 Vectors</th>
<th>7 Vectors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Time</td>
<td>Value</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Time</td>
<td>Value</td>
</tr>
<tr>
<td>unreg</td>
<td>36</td>
<td>145</td>
<td>0.02</td>
<td>6</td>
</tr>
<tr>
<td>count</td>
<td>35</td>
<td>161</td>
<td>0.05</td>
<td>6</td>
</tr>
<tr>
<td>lal</td>
<td>26</td>
<td>179</td>
<td>0.04</td>
<td>6</td>
</tr>
<tr>
<td>i2</td>
<td>201</td>
<td>242</td>
<td>0.09</td>
<td>6</td>
</tr>
<tr>
<td>cht</td>
<td>47</td>
<td>249</td>
<td>0.08</td>
<td>6</td>
</tr>
<tr>
<td>C432</td>
<td>36</td>
<td>282</td>
<td>0.09</td>
<td>6</td>
</tr>
<tr>
<td>i4</td>
<td>192</td>
<td>308</td>
<td>0.09</td>
<td>6</td>
</tr>
<tr>
<td>ex2</td>
<td>85</td>
<td>351</td>
<td>0.11</td>
<td>6</td>
</tr>
<tr>
<td>i5</td>
<td>133</td>
<td>445</td>
<td>0.17</td>
<td>6</td>
</tr>
<tr>
<td>alu2</td>
<td>10</td>
<td>462</td>
<td>0.11</td>
<td>6</td>
</tr>
<tr>
<td>term1</td>
<td>34</td>
<td>525</td>
<td>0.17</td>
<td>6</td>
</tr>
<tr>
<td>x4</td>
<td>94</td>
<td>635</td>
<td>0.16</td>
<td>6</td>
</tr>
<tr>
<td>i6</td>
<td>138</td>
<td>764</td>
<td>0.24</td>
<td>6</td>
</tr>
<tr>
<td>i7</td>
<td>199</td>
<td>1011</td>
<td>0.33</td>
<td>6</td>
</tr>
<tr>
<td>i9</td>
<td>88</td>
<td>1218</td>
<td>0.35</td>
<td>6</td>
</tr>
<tr>
<td>vda</td>
<td>17</td>
<td>1417</td>
<td>0.4</td>
<td>6</td>
</tr>
</tbody>
</table>

---


---

© 2010 ACADEMY PUBLISHER


