Using SAT Techniques in Dynamic Burn-in Vector Generation

Fadi A. Aloul[1] and Assim Sagahyroon[2]

Department of Computer Science & Engineering, American University of Sharjah, UAE

[1]faloul@aus.edu

[2]asagahyroon@aus.edu

Abstract - **Dynamic burn-in testing is an integral component of any test plan that seeks to produce reliable integrated circuits. Despite its importance in ensuring the reliability of semiconductors, burn-in has been a major contributor to overall test cost and turnaround time. In this work we discuss the application of advanced Boolean satisfiability (SAT) techniques to generate a set of vectors or input stimuli that increases the nodal activity in the circuit and hence the elevation of its temperature. The vectors are designed to uniformly stress all parts of the circuit. Additionally, we present a SAT-based methodology where weak nodes can selectively be targeted for high switching activity in an effort to detect potential failures. Finally, SAT-based solvers are compared against generic Integer Linear Programming (ILP) solvers when handling the vector generation problem.**

Keywords - Testing, Power, Vector Generation, Integer Linear Programming, Boolean Satisfiability.

I. Introduction

All integrated circuits that pass production tests are not identical. When put to actual use, some will fail very quickly while others will function for a long time. Burn-in ensures reliability of tested devices by testing, either continuously or periodically, over a long period and causing the bad devices to actually fail. According to correlation studies, the occurrence of potential failures can be accelerated at elevated temperatures [1]. 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 [8]. In dynamic burn-in testing, the design of test patterns 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 [9] 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 [18], 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 [15]. 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.

In this work, we formulate the test patterns generation problem for dynamic burn-in as a Boolean satisfiability (SAT) problem with the two primary objectives: one being the generation of a sequence that uniformly stress the device under test and secondly, the ability to target and stress weak nodes in the circuit in order to expose the early failure of these nodes.

The rest of the paper is organized as follows: in Section 2 we discuss the motivation behind this work and describe how the problem is formulated. Discussion of the results is presented in Section 3, and we conclude the paper in Section 4.

II. Motivation and Problem Formulation

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 Boolean satisfiability (SAT) solvers have been used to solve *decision*-based problems [13], recently, these solvers have been extended to tackle Pseudo-Boolean (PB) constraints which are linear inequalities with integer coefficients [2, 6, 7, 20]. As a result, researchers can now use PB constraints to express *optimization* problems that are traditionally handled as integer linear programming (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 [5], FPGA routing [14], global routing [2], logic synthesis [12], power leakage [3], and power optimization [19].

In the following section we detail the formulation of the test search problem as a SAT instance and in subsequent sections, we will assess the possibility of solving it using advanced SAT-based algorithms, and secondly using generic-based ILP solvers. A distinct advantage is the fact that SAT-based and ILP-based searches are complete, that is, the entire search space is examined, and either the problem is proven satisfiable, i.e. a solution does exist, or unsatisfiable, i.e. the problem has no solution.

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. An example of CNF expressions for simple gates is given below:

- $z = NOT(x)$: CNF is $(x+z)(\bar{x}+\bar{z})$
- $z = AND(x,y)$: CNF is $(x+\bar{z})\cdot(y+\bar{z})\cdot(\bar{x}+\bar{y}+z)$
- $z = OR(x,y)$: CNF is $(\bar{x}+z)\cdot(\bar{y}+z)\cdot(x+y+\bar{z})$
- $z = NAND(x,y)$: CNF is $(x+z)\cdot(y+z)\cdot(\bar{x}+\bar{y}+\bar{z})$
- $z = XOR(x,y)$: CNF is $\begin{array}{l}(\bar{x}+y+z)\cdot(x+\bar{y}+z)\cdot\\(\bar{x}+\bar{y}+\bar{z})\cdot(x+y+\bar{z})\end{array}$

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

i. Create a set of CNF constraints representing the circuit's logical behavior after the application of an input vector V_1 (*Circuit A*).
ii. 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*).
iii. 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*).
iv. ......
v. 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 1. 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 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.

III. Experimental Results and Discussion

For the sake of brevity, a subset consisting of sixteen circuits with varying sizes from the MCNC suite of benchmarks [11] is selected to test the proposed approach. In all cases the SAT-based 0-1 ILP MiniSAT+ [7] 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:

i. A search for *n* vectors with all nodes having similar weights.
ii. Same as in (i) above, but adding constraints ensuring that each node will flip at least once.
iii. 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 1), 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 I, II and III respectively. Columns I, II and III of the tables list the name of circuit, number of primary inputs and the total number of gates in the circuit. 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 I (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 $(n-1)$.

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

$$(\overline{d_1}+d_2+D1)\cdot(d_1+\overline{d_2}+D1)$$
$$(\overline{d_1}+\overline{d_2}+\overline{D1})\cdot(d_1+d_2+\overline{D1})$$
$$(\overline{e_1}+e_2+E1)\cdot(e_1+\overline{e_2}+E1)$$
$$(\overline{e_1}+\overline{e_2}+\overline{E1})\cdot(e_1+e_2+\overline{E1})$$
$$(\overline{f_1}+f_2+F1)\cdot(f_1+\overline{f_2}+F1)$$
$$(\overline{f_1}+\overline{f_2}+\overline{F1})\cdot(f_1+f_2+\overline{F1})$$
$$(\overline{g_1}+g_2+G1)\cdot(g_1+\overline{g_2}+G1)$$
$$(\overline{g_1}+\overline{g_2}+\overline{G1})\cdot(g_1+g_2+\overline{G1})$$

XOR Output Conditions

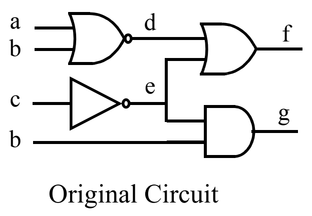
$$(\overline{d_2}+d_3+D2)\cdot(d_2+\overline{d_3}+D2)$$
$$(\overline{d_2}+\overline{d_3}+\overline{D2})\cdot(d_2+d_3+\overline{D2})$$
$$(\overline{e_2}+e_3+E2)\cdot(e_2+\overline{e_3}+E2)$$
$$(\overline{e_2}+\overline{e_3}+\overline{E2})\cdot(e_2+e_3+\overline{E2})$$
$$(\overline{f_2}+f_3+F2)\cdot(f_2+\overline{f_3}+F2)$$
$$(\overline{f_2}+\overline{f_3}+\overline{F2})\cdot(f_2+f_3+\overline{F2})$$
$$(\overline{g_2}+g_3+G2)\cdot(g_2+\overline{g_3}+G2)$$
$$(\overline{g_2}+\overline{g_3}+\overline{G2})\cdot(g_2+g_3+\overline{G2})$$

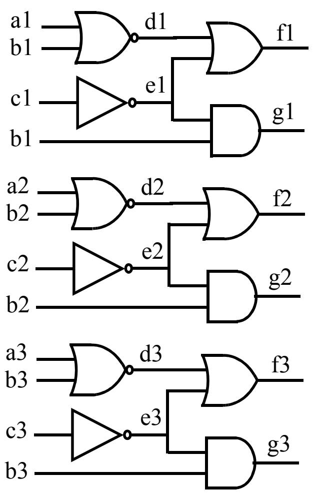
Transition Objective Function

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

Solution:
Max Transitions = 6

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





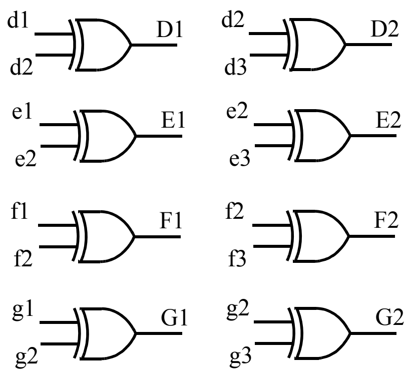


Fig. 1. An illustrative example showing how to determine the sequence of vectors in the given circuit.

From Table I, 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 II (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 this eases the search and satisfiability was achievable. The added constraints had lead to a significant increase in time. 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 II, 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 III 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 mod-

Table I. Experimental results 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 II. Experimental results 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	

ified 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 IV shows the results of comparing the performance of the SAT-based 0-1 ILP solver MiniSAT+ to the generic commercial ILP solver, CPLEX [10] 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.

IV. Conclusions

The work discussed here proposes a Boolean satisfiability (SAT)-based approach that attempts to derive a vector sequence that continuously tends to maximize the nodal activity in a circuit, and hence its heat dissipation during the burn-in test phase. Furthermore the approach attempts to uniformly force transitions in all the nodes to avoid overstressing. We also experimented with the option of intentionally targeting and maximizing activity in selected nodes; an exercise that assists in exposing the early failure of nodes. In all of the experiments, we utilized a 0-1 Integer Linear Programming (ILP) SAT-based solver, namely MiniSat+, that can handle both decision and optimization problems.

TABLE III. EXPERIMENTAL RESULTS 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

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 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.

TABLE IV. EXPERIMENTAL RESULTS 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**

REFERENCES

[1] V. D. Agrawal and S. C. Seth, "Test Generation for VLSI Circuits," *Tutorial: Test Generation for VLSI Circuits, IEEE Computer Society Press*, 1988.

[2] F. Aloul, A. Ramani, I. Markov, and K. Sakallah, "Generic ILP versus Specialized 0-1 ILP: An Update", in *Proc. of the International Conference on Computer-Aided Design* (ICCAD), 450-457, 2002.

[3] 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), 167-177, 2002.

[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*, 75-82, 2006.

[5] 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), 317-320, 1999.

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

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

[8] 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), Oct.-Dec. 1999.

[9] 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*, 567-576, 1998.

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

[11] MCNC Benchmarks, *http://www.cbl.ncsu.edu/CBL_Docs/Bench.htm*

[12] 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.

[13] 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), 530-535, 2001.

[14] G. Nam, F. A. Aloul, K. A. Sakallah, and R. Rutenbar, "A Comparative Study of Two Boolean Formulations of FPGA Detailed Routing Constraints", in *IEEE Transactions on Computers*, 53(6), 688-696, June 2004.

[15] M. Po-Leen et al, "Shortening Burn-In Test: Application of HVST and Weibull Statistical Analysis," in *IEEE Transactions on Instrumentation and Measurement*, 56(3), 2007.

[16] S. Roy, P. Chakrabarti, and P. Dasgupta, "Satisfiability Models for Maximum Transition Power," in *IEEE Trans. on VLSI*, 16(8), 941-951, 2008.

[17] S. Roy, P. Chakrabarti, and P. Dasgupta, "Bounded Delay Timing Analysis Using Boolean Satisfiability," in *Proc. of the International VLSI Design Conference,* 295-302, 2007.

[18] A. Sagahyroon, "Maximizing Heat Dissipation for Burn-In Testing," in *Proc. of the Canadian Conference on Electrical and Computer Engineering*, v. 1, 399-402, 2002.

[19] A. Sagahyroon and F. Aloul, "Using SAT-Based Techniques in Power Estimation," in *Microelectronics Journal*, vol. 38, issues 6-7, 706-715, 2007.

[20] 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, 684-685, 2005.