Exciting Stuck-Open faults in CMOS Circuits Using ILP Techniques

Fadi Aloul and Assim Sagahyroon
Department of Computer Engineering
American University of Sharjah - UAE
{faloul, asagahyroon}@aus.edu

Bashar Al Rawi
Department of Computer Engineering
American University in Dubai - UAE
Bashar.Alrawi@mymail.aud.edu

Abstract

To excite a stuck-open fault in a CMOS combinational circuit, it is only necessary that the output of the gate containing the fault takes on opposite values during the application of two successive input vectors to the primary inputs of the circuit. In this paper we formulate the excitation problem as an ILP instance and use two advanced ILP solvers, one is Boolean satisfiability (SAT)-based and the other is generic, to search for a pair of input vectors that maximizes the simultaneous excitation of as many stuck-open faults in the circuit as possible. The proposed approach was tested using benchmarks from the ISCAS 89 suite of circuits. Experimental results indicate that fault-excitation, within a reasonable CPU time limit, is possible in most cases.

1. Introduction

To review the problem of detecting FET stuck-open faults in CMOS gates, consider the circuit diagram of the 2-input NOR gate shown in Figure 1. Assume the existence of an open circuit on the *N2* transistor. Input condition $a = 0$ and $b = 1$ should pull the output node to logic 0 value; but due to the presence of the open fault the output node will not be pulled to ground level. Hence the gate output will assume the value of the previous logic state for a short time. That is, the presence of the stuck-open fault introduces a sequential behavior and that is why sometimes stuck-open faults are referred to as memory faults [25]. To generalize, consider the block diagram of a general CMOS gate shown in Figure 2. A CMOS gate consists of two complementary networks of *p*-channel FETs (PFETs) and *n*-channel FETs (NFETs). The output of a CMOS gate is a 0 (1) if a path through the conducting NFETs (PFETs) is established from the output to *Vss* (*Vdd*) and no conducting path are established in the PFET (NFET) network. To detect a single PFET (NFET) transistor stuck-open fault, an initializing input vector *T1* is used to set the output to 0 (1). This vector is then followed by a second vector *T2* that is intended to establish a conducting path in the PFET (NFET) network. When *T2* is applied and the PFET (NFET) network was presumed to be fault-free then the circuit is output is pulled to logic 1 value (logic 0 for NFET networks). Otherwise the circuit will maintain a faulty gate output value which the value established by the initializing vector *T1*.

The vector pair <*T1*, *T2*> is called a two-pattern test. Therefore, a stuck-open fault in a CMOS gate realizing a function *f* is detectable if there exists and input vector *T2* for which the output becomes floating and shows the previous logic value that has been established by the application of an initializing vector *T1*.

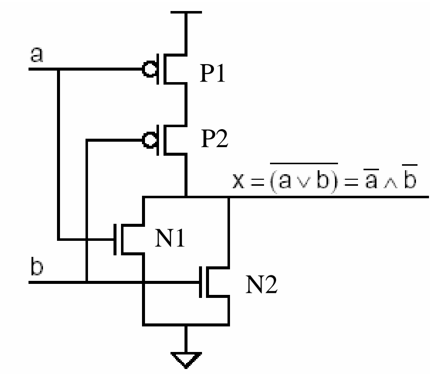


Figure 1. CMOS NOR gate

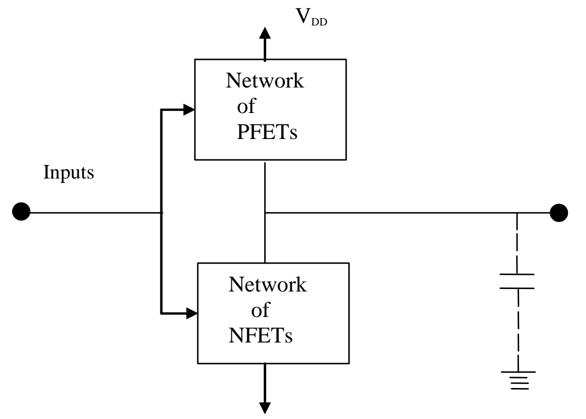


Figure 2. General CMOS gate

If the CMOS gate is embedded in a very large combinational circuit then the detection problem is aggravated due to the fact that the test pair must excite the fault and must be able of propagating the effect of the fault to an observable output.

It was proven that the problem of finding a two-pattern test <*T1*, *T2*> is NP-hard [20]. Several methods have been proposed to generate this type of test [8, 11, 14, 9]. However none of these methods have attempted to investigate the use Boolean satisfiability (SAT) techniques to search for the test vector pair. In [3], SAT was used to compute test vectors for stuck-at faults but stuck-open faults were not considered.

In this paper we present a SAT based approach to search for two vectors that would simultaneously excite stuck-open faults in as *many* CMOS gates in combinational circuits as possible. Note that in this work we are only concerned with finding a pair that would excite many faults. Propagating the fault effect to an external output and assessing the fault coverage in the circuit is left for future work. It has been observed that the likelihood that a number of faults in the circuit are typically easy to detect is high. This makes the application of initial input pair either random or user supplied to a fault simulation system very desirable. One can argue that, applying the excitation vector pair as a user-supplied test pair to the fault simulation system would definitely improve the probability of detecting more faults when compared to the approach of applying random vectors to the fault simulator.

This paper is organized as follows. Section 2 provides a discussion of Boolean satisfiability. Sections 3 and 4 explain the formulation of the problem within its context. In Section 5 we present and discuss the experimental results. The paper is concluded in Section 6.

2. Boolean Satisfiability

Recent years have seen a remarkable growth in the use of Boolean Satisfiability (SAT) models and algorithms for solving various problems in Electronic Design Automation (EDA). This is mainly due to the fact that SAT algorithms have seen tremendous improvements in the last few years, allowing larger problem instances to be solved in different applications domains. Such applications include formal verification [4], FPGA routing [19], global routing [1], logic synthesis [17], and sequential equivalence checking [5]. SAT has also been extended to a variety of applications in Artificial Intelligence including other well-known NP-complete problems such as graph colorability, vertex cover, and Hamiltonian path [7].

In SAT, given a formula *f*, the objective is to identify an assignment to a set of Boolean variables that will satisfy a set of constraints. If an assignment is found, it is known as a *satisfying* assignment, and the formula is called *satisfiable*. Otherwise if an assignment doesn't exist, the formula is called *unsatisfiable*. The constraints are typically expressed in conjunctive normal form (CNF). In CNF, the formula consists of the conjunction (AND) of m clauses $\omega_1, \ldots, \omega_m$ each of which consists of the disjunction (OR) of k literals. A literal l is an occurrence of a Boolean variable or its complement. Hence, in order to satisfy a formula, each of its clauses must have at least one literal evaluated to true.

As an example, the CNF instance

$$f(a, b, c) = (a + \bar{b}) \cdot (a + b + c) \quad (1)$$

Table 1. CNF formulas representing simple gates.

Gate Type	Gate Equation	CNF Expression
NOT gate symbol	$z = NOT(x)$	$(x + z) \cdot (\bar{x} + \bar{z})$
NOR gate symbol	$z = NOR(x, y)$	$(\bar{x} + \bar{z}) \cdot (\bar{y} + \bar{z}) \cdot (x + y + z)$
NAND gate symbol	$z = NAND(x, y)$	$(x + z) \cdot (y + z) \cdot (\bar{x} + \bar{y} + \bar{z})$
AND gate symbol	$z = AND(x, y)$	$(x + \bar{z}) \cdot (y + \bar{z}) \cdot (\bar{x} + \bar{y} + z)$
OR gate symbol	$z = OR(x, y)$	$(\bar{x} + z) \cdot (\bar{y} + z) \cdot (x + y + \bar{z})$
XOR gate symbol	$z = XOR(x, y)$	$(\bar{x} + y + z) \cdot (x + \bar{y} + z) \cdot (\bar{x} + \bar{y} + \bar{z}) \cdot (x + y + \bar{z})$

consists of 3 variables, 2 clauses, and 5 literals. The assignment $\{a = 0, b = 1, c = 0\}$ leads to a conflict, whereas the assignment $\{a = 0, b = 0, c = 1\}$ satisfies *f*.

Despite the problem being NP-Complete, there have been dramatic improvements in SAT solver technology over the past decade. This has lead to the development of several powerful SAT solvers, e.g. zchaff and miniSAT, that are capable of handling problems consisting of thousands of variables and millions of constraints [2, 12, 15, 18, 27]. These solvers claim competitive results in runtime efficiency and robustness.

Recently, SAT solvers [1, 6, 10, 24, 26] have been extended to handle pseudo-Boolean (PB) constraints which are linear inequalities with integer coefficients that can be expressed in the normalized form [1] of:

$$a_1x_1 + a_2x_2 + \ldots + a_nx_n \geq b \quad (2)$$

where $a_i, b \in Z$ and x_i are Boolean variables. PB constraints can, in some cases, replace an exponential number of CNF constraints. They have been found to be very efficient in expressing "counting constraints" [1]. Another key advantage of PB constraints is the ability to express *optimization* problems (as opposed to only *decision* problems) which are traditionally handled as integer linear programming (ILP) problems. Hence, SAT solvers can now handle both decision and optimization problems.

Note that circuits can be easily represented as a CNF formula by conjuncting the CNF formulas for each gate output. A gate output can be expressed using a set of clauses which specify the valid input-output combinations for the given

gate. Hence, a CNF formula φ for a circuit is defined as the union of set of clauses φ_x for each gate with output x:

$$\varphi = \bigcup_{x \in Q} \varphi_x \quad (3)$$

where Q denotes all gate outputs and primary inputs in the circuit. Table 1 shows generalized CNF formulas for various gates. For example, a NOR gate with inputs x and y and output z is represented using the following set of clauses $(\bar{x} + \bar{z}) \cdot (\bar{y} + \bar{z}) \cdot (x + y + z)$. If x is assigned the value 1, the first clause will imply $z = 0$, since this is the only possible assignment that will satisfy the first clause. Similarly if x and y are assigned the value 0, z will be implied to 1 since this is the only assignment that will satisfy the third clause.

3. Problem Formulation and Implementation

In this paper, we are interested in using SAT solvers to identify two input vectors that can excite the maximum number of stuck-open faults in a circuit. 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 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 constraints are represented as explained in the background section of this paper. 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 constraint is expressed using the principles explained in the background section. 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 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.

Figure 3 (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.

4. Generating Robust Input Vector Pairs

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 when compared with 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 [9, 22].

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:

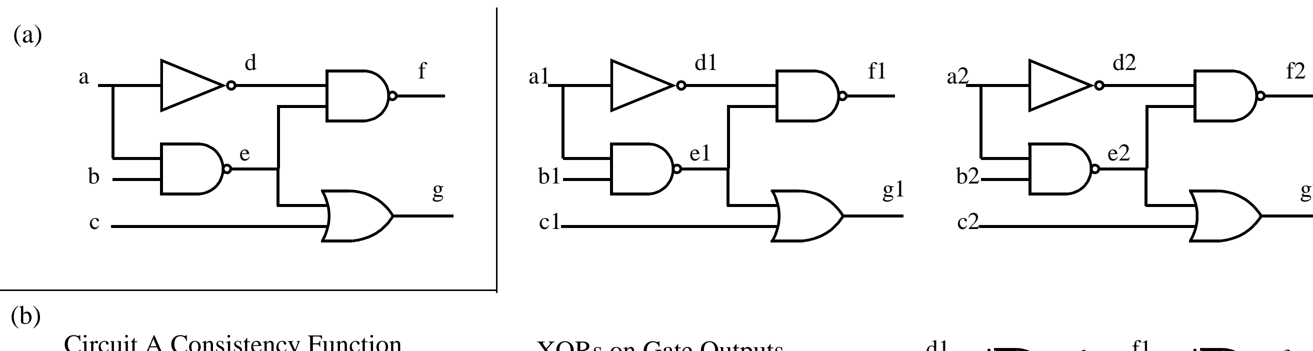
1. A Set of clauses representing XOR gates between the primary inputs of gates in (1) and (2). The number of XOR gates equals the number of primary inputs in the original circuit. An XOR gate output of logic 1 indicates that the two vectors have different values for the same primary input.
2. A PB constraint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

Figure 3 (c) shows an example where the solver was successful in finding a pair that is only a unit hamming distance apart.

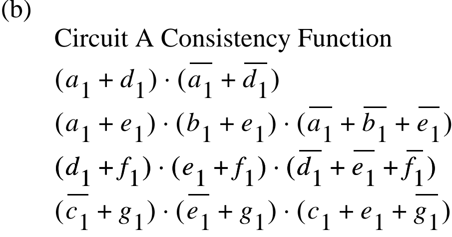
5. Experimental Results

Table I summarizes the results obtained using the SAT-based ILP solver PBS 4.0 [1, 21] and the commercial ILP solver CPLEX 7.0 [13]. 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 [16] benchmark circuits. Each benchmark was sensitized using "sis" [23] into a circuit consisting of 2-input NAND, NOR and inverter gates. The runtime was set to a limit of 1000 seconds. Note that both solvers perform a *complete* search, i.e. if an optimal so-

(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

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

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

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

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

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

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

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

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

d1, d2 → d; f1, f2 → f; e1, e2 → e; g1, g2 → 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 $= 4$
$\{a_1, b_1, c_1\} = \{0, 1, 0\}$
$\{a_2, b_2, c_2\} = \{1, 1, 0\}$

a1, a2 → a; b1, b2 → b; c1, c2 → c

Figure 3. 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.

lution is found, no other test pair exists that can excite a larger number of stuck-open faults.

In Table 2, 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 re-

sults 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).

6. Conclusion

A search strategy for a test pair that would excite stuck-open faults in CMOS combinational circuits was presented. The proposed strategy utilized advanced SAT-based and generic ILP solvers to identify two vectors that would simultaneously excite stuck-open faults in as *many* CMOS gates in combinational circuits as possible. The methodology was tested on a number of benchmark circuits and showed promising results. The presented approach is *complete* and will find the best possible input vector pair. In future work, we intend to expand the ideas presented here and develop a complete test generation system for stuck-open faults. Meanwhile, the patterns returned by the solvers can undoubtedly be used initially in conjunction with a fault simulator as user specified-inputs. This will assist in the initial detection of some faults without resorting to time consuming sensitization and propagation searches.

References

[1] 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.

[2] 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), 203-208, 1997.

[3] S. Brayton et al, "Combinational test generation using satisfiability," in *IEEE Transactions on CAD*, 15 (6), 1996.

[4] 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.

[5] P. Bjesse and K. Claessen, "SAT-based Verification Without State Space Traversal", in *Proc. of Formal Methods in Computer-Aided Design*, 372-389, 2000.

[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. Creignou, S. Kanna, and M. Sudan, "Complexity Classifications of Boolean Constraint Satisfaction Problems", *SIAM Press*, 2001.

[8] H. Ching and J. Abraham, "Transistor level test generation for physical failures in CMOS circuits," in *Proc. of the Design Automation Conference*, 1986.

[9] D. Das et al, "Universal and robust testing of stuck-open faults in Reed-Muller canonical CMOS circuits," in *International Journal of Electronics*, 90 (1), 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), 635-640, 2002.

[11] Y. Elziq, "Automatic test generation of stuck-open faults in CMOS VLSI," in *Proc. of the Design Automation Conference*, 1981.

[12] E. Goldberg and Y. Novikov, "BerkMin: A fast and robust SAT-solver," in *Proc. of the Design Automation and Test Conference in Europe* (DATE), 142-149, 2002.

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

[14] N. Jha, "Multiple Stuck-Open Fault Detection in CMOS Logic Circuits," in *IEEE Transaction on Computers*, 37 (4), 1988.

[15] J. Marques-Silva and K. Sakallah, "GRASP: A Search Algorithm for Propositional Satisfiability", in *IEEE Transactions on Computers*, (48)5, 506-521, 1999.

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

[17] 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*, 2002.

[18] 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.

[19] 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), 222-227, 2001.

[20] F. Najm and I. Hajj, "The complexity of fault detection in MOS VLSI circuits," in *IEEE Transactions on CAD*, 9 (4), 1990.

[21] PBS version 4, *http://www.eecs.umich.edu/~faloul/Tools/pbs4*

[22] S. Reddy and M. Reddy, "Testable realization on FET stuck-open faults in CMOS combinational logic circuits," in *IEEE Transactions on Computers*, 35, 1986.

[23] 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.

[24] 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.

[25] J. Soden and R. Treece, "CMOS IC stuck-open faults electrical effect and design consideration," in *Proc. of the International Test Conference*, 1989.

[26] J. Whittemore, J. Kim, and K. Sakallah, "SATIRE: A New Incremental Satisfiability Engine," in *Proc. of Design Automation Conference* (DAC), 542-545, 2001.

[27] H. Zhang, "SATO: An Efficient Propositional Prover", in *Proc. of the International Conference on Automated Deduction*, 272-275, 1997.

Table 2. Experimental results 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