

Robust SAT-Based Search Algorithm for Leakage Power Reduction

Fadi A. Aloul^a, Soha Hassoun^b, Karem A. Sakallah^a, David Blaauw^a

^aDepartment of Electrical Engineering and Computer Science
University of Michigan - Ann Arbor
{faloul, karem, blaauw}@eecs.umich.edu

^bDepartment of Electrical Engineering and Computer Science
Tufts University
soha@eecs.tufts.edu

Abstract. Leakage current promises to be a major contributor to power dissipation in future technologies. Bounding the maximum and minimum leakage current poses an important problem. Determining the maximum leakage ensures that the chip meets power dissipation constraints. Applying an input pattern that minimizes leakage allows extending battery life when the circuit is in stand-by mode. Finding such vectors can be expressed as a satisfiability problem. We apply in this paper an incremental SAT solver, PBS [1], to find the minimum or maximum leakage current. The solver is called as a post-process to a random-vector-generation approach. Our results indicate that using a such a generic SAT solver can improve on previously proposed random approaches [7].

1 Introduction

One of the challenges in designing integrated circuits is limiting energy and power dissipation. The concerns are many, including packaging and cooling costs, battery life in portable systems, and power supply grid design. As process geometries scale to achieve a 30% gate delay reduction per technology generation, the typical scaling of the supply voltage (V_{dd}) by 30% promises to reduce the power by 50% [4]. Such supply voltage scaling will also require scaling the threshold voltage (V_t) for MOS devices to sustain the gate delay reductions. Decreasing V_t however results in an exponential increase in the subthreshold leakage current. Thus, although the overall power is decreasing, the power dissipation due to the leakage component is increasing. It is expected that within the next 2 process generations leakage power dissipation will contribute as much as 50% of the total power dissipation for high performance designs [21]. Particularly for devices that spend a significant percentage of their operation in standby mode, such as mobile device, is leakage current a critical concern. For such devices, it is not uncommon for standby leakage power to be the dominant factor in the total battery life time.

Several circuit techniques have been proposed in recent years to minimize leakage currents. A common approach is to use a dual- V_t process where transistors are assigned either

a high or a low threshold voltage, the high V_t devices having typically 30-50% more delay but as much as 30x less leakage than the low V_t device. This approach requires one additional process step and therefore entails some additional manufacturing cost. One dual- V_t implementation uses a high V_t NMOS or PMOS transistor to shut off the supply to low V_t logic [9]. This approach results in very good reduction in standby leakage current, but requires significant layout area overhead for the large power supply shutoff transistors. It also suffers from power supply integrity issues. In multi-threshold CMOS, speed critical gates are assigned a low V_t and non-speed critical gates are assigned a high V_t , thereby reducing the leakage of these gates. This approach has the advantage that it requires no additional area overhead. However, multi-threshold CMOS typically has a significant performance penalty since the majority of the gates must be assigned high V_t in order to obtain a significant savings in the standby leakage current. Recently, multi-threshold CMOS techniques that combine device sizing and V_t assignment have been proposed [17]. Another recent leakage minimization approach is based on the observation that a stack of two OFF devices has a significantly reduced leakage compared to a single OFF device [14]. A pull-down transistor is therefore replaced by two series connect transistors. However, this technique has a significant area overhead.

One of the key observations of gate leakage is that it strongly depends on the input state of a gate [7]. Based on this observation, Halter and Najm modified the circuit's latches to force their outputs high or low during sleep mode without losing the state of the latch. By assigning a low leakage state to output nodes of the latches, significantly leakage current can be saved during standby mode. Forcing the output to a high or low state requires a pass transistor in parallel with a shut to Vdd or Gnd and results in a slight increase in the latch delay. For flip-flop based designs, an alternate methods was recently proposed that does not increase the delay of the flip-flop noticeably [21]

In order to minimize the leakage in this approach, the circuit state with the minimum leakage state must be determined. On the other hand, the circuit state with the maximum leakage is important for designers to ensure that the circuit meets the standby power constraints which in turn impacts battery-life. Several methods for finding the minimum or maximum leakage state have been proposed. In [7], a the minimal leakage state was determined with random vectors whose number was selected to achieve a specific statistical confidence and tolerance. Bobba and Hajj proposed a graph-based heuristic to estimate maximum leakage power [3]. A constraint graph is built as follows: for each gate, 2^k vertices are created. The weight of each vertex represents the leakage power when the inputs are in a particular state. Edges between the vertices represent constraints that only one of the 2^k input assignments is possible. Other edges are added that enforce the logic functionality between gates. The proposed heuristic uses a greedy linear algorithm to find a *maximum weight independent set* to maximize the weights of the vertices under the constraint that no edges between any pair of selected vertices are selected.

Johnson et al. [8] experiment with greedy heuristics and an exact branch and bound search to find maximum and minimum leakage bounds. They propose a leakage observatory metric that reflects how a particular circuit input affects the state and thus magnitude of leakage current for all the circuit components. This metric can be calculated once before assigning any input variables, or it can be re-calculated repeatedly after each partial assignment. Their experiments show that the dynamic re-calculation of observatory metric after partial

assignments improved the quality of the results found by the greedy heuristics to the point that it matched those found by the exact branch and bound algorithm.

Recently, SAT has been shown to be very successful in various applications, such as formal verification [2], FPGA routing [13], and timing analysis [16]. In this paper we propose using a SAT solver, PBS [1], to find the minimum or maximum leakage current. Finding the input vector that causes such a current corresponds to solving a SAT instance [8]. PBS was chosen because: (a) it allows incremental exploration of the solution space, (b) it handles both CNF and pseudo-Boolean constraints that can express gate leakage restrictions, and (c) it implements the latest enhancements in SAT. The obtained leakage vectors can be used by designers to determine if the circuit meets the required leakage specifications or to assign the circuit to a low leakage state during standby mode.

We begin the paper with an overview of Boolean Satisfiability. We then describe how to model the leakage problem to solve it via PBS. We conclude with an example and experimental results.

2 Boolean Satisfiability

The satisfiability problem involves finding an assignment to a set of binary variables that satisfies a given set of constraints. In general, these constraints are expressed in *conjunctive normal form* (CNF). A CNF formula ϕ on n binary variables x_1, \dots, x_n consists of the conjunction (AND) of m clauses $\omega_1, \dots, \omega_m$ each of which consists of the disjunction (OR) of k literals. A literal l is an occurrence of a Boolean variable or its complement. We will refer to a CNF formula as a clause database (DB).

Most current SAT solvers [1, 10, 12, 19, 20] are based on the original Davis-Putnam backtrack search algorithm [5]. The algorithm performs a search process that traverses the space of 2^n variable assignments until a satisfying assignment is found (the formula is satisfiable), or all combinations have been exhausted (the formula is unsatisfiable). Originally, all variables are unassigned. The algorithm begins by choosing a decision assignment to an unassigned variable. A decision tree is maintained to keep track of variable assignments. After each decision, the algorithm determines the implications of the assignment on other variables. This is obtained by forcing the assignment of the variable representing an unassigned literal in an unresolved clause, whose all other literals are assigned to 0, to satisfy the clause. This is referred to as the *unit clause* rule. If no conflict is detected, the algorithm makes a new decision on a new unassigned variable. Otherwise, the backtracking process unassigns one or more recently assigned variables and the search continues in another area of the search space.

As an example, a CNF instance $f = (a + b)(\bar{b} + c)$ consists of 3 variables, 2 clauses, and 4 literals. The assignment $\{a = 0, b = 1, c = 0\}$ leads to a conflict, whereas the assignment $\{a = 0, b = 1, c = 1\}$ satisfies f .

Several powerful methods have been proposed to expedite the backtrack search algorithm. One of the best methods is known as the conflict analysis procedure [10] and has been implemented in almost all SAT solvers, such as GRASP [10], Chaff [12], PBS [1], SATIRE [19], and SATO [20]. Whenever a conflict is detected, the procedure identifies the causes of the conflict and augments the clause DB with additional clauses, known as conflict-induced clauses, to avoid regenerating the same conflict in future parts of the search process. In essence, the procedure performs a form of learning from the encountered conflicts. Significant

speedups have been achieved with the addition of conflict-induced clauses, as they tend to effectively prune the search space.

Intelligent decision heuristics and random restarts [6], also played an important role in enhancing the SAT solvers performance. Chaff [12] proposed an effective decision heuristic, known as VSIDS, and implemented several other enhancements, including random restarts, which lead to dramatic performance gains on many CNF instances.

Recently, a new SAT solver, known as SATIRE [19], introduced two new enhancements, namely incremental satisfiability and handling non-CNF constraints. The former enhancement allows sets of related problems to be solved incrementally without the need to solve each problem separately. The latter enhancement, involves expressing complex constraints whose encoding in CNF is impractical. In particular, it handles Pseudo-Boolean (PB) expressions which are expressions of the form $\sum c_i l_i \leq n$, where c_i and n are constant real values and l_i are literals of Boolean decision variables, i.e. x_j or \bar{x}_j . This attracted our attention, since the ability to reason over constraints which include real-valued components is a key feature when looking at the application of SAT solvers to leakage detection.

In this paper, we will use PBS [1], a new SAT solver that handles both CNF and PB constraints. It combines the state-of-the-art search techniques implemented in Chaff [12] and SATIRE [19]. Unlike previously proposed stochastic local search solvers [18], PBS is complete (i.e. can prove both satisfiability and unsatisfiability) and has been shown to achieve several order-of-magnitude speedups when compared to SATIRE [19].

3 Bounding Leakage Using SAT Algorithms

In this section, we present our SAT-based approach for identifying the possible power leakage in a circuit. A SAT problem is created for each circuit with an objective function to minimize or maximize the possible leakage. Each problem consists of two groups of constraints: (1) a large set of CNF clauses modeling the circuit’s logical behavior (2) objective constraint which specifies the amount of desired leakage.

3.1 Representing Circuits in CNF

Circuits are 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 :

$$\phi = \bigcup_{x \in Q} \phi_x \quad (1)$$

where Q denotes all gate outputs and primary inputs in the circuit. Figure 1, shows generalized CNF formulas for the NAND, NOR, and INVERTER gates.

3.2 Representing Leakage Constraints

After representing the circuit’s logical behavior, we need to define an objective for the desired leakage. The objective function of the leakage estimate problem will be represented in PB, where Boolean decisions on the primary inputs of the circuit will determine a possible

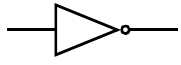
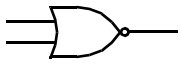
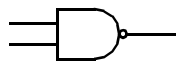
Gate Type	Gate Formula
 $z = NOT(x_1)$	$(x_1 + z) \cdot (\bar{x}_1 + \bar{z})$
 $z = NOR(x_1, \dots, x_j)$	$\left[\prod_{i=1}^j (\bar{x}_i + \bar{z}) \right] \cdot \left(\sum_{i=1}^j x_i + z \right)$
 $z = NAND(x_1, \dots, x_j)$	$\left[\prod_{i=1}^j (x_i + z) \right] \cdot \left(\sum_{i=1}^j \bar{x}_i + \bar{z} \right)$

Fig. 1. CNF formulas representing simple gates.

leakage value for the circuit. In other words, this can be viewed as a constraint representing the predicate, “There exists a leakage $< k$ ”, where k is a real value. Such constraints are typically hard to express in CNF.

Given a *gate* with n inputs and leakage values c associated with each of the 2^n possible input combinations, a PB is generated as follows:

$$\sum_{i=1}^{2^n} c_i q_i \leq k \quad (2)$$

where q and k represent the input combination and maximum possible desired leakage, respectively. Consequently, a single PB constraint can be expressed for all gates in the circuit.

As an example, consider a 2-input NAND gate. Using the leakage power values listed in Table 1, the objective function will be expressed as:

$$10(\bar{a}\bar{b}) + 173(\bar{a}b) + 304(a\bar{b}) + 544(ab) \leq k \quad (3)$$

However, PBS’s input format allows a single literal to be associated with each coefficient entry, i.e. the term (ab) needs to be replaced by a single literal term. Hence, for a 2-input gate, four new variables are declared, each of which represents a possible input combination. Replacing (ab) by S can be easily expressed in CNF using the following set of clauses:

$$(\bar{a} + \bar{b} + S) \cdot (a + \bar{S}) \cdot (b + \bar{S}) \quad (4)$$

The following set of clauses, in addition to the objective PB constraint and the CNF clauses representing the circuit’s logical behavior compose the required input for PBS.

Table 1. Leakage power for 2-input NAND, 2-input NOR, and an Inverter.

Inputs		Leakage Power (pA)		Input	Leakage Power (pA)
A	B	NAND	NOR	A	NOT
0	0	10	308	0	159
0	1	173	540	1	271
1	0	304	168		
1	1	544	112		

3.3 Identifying Minimum/Maximum Leakage

Initially, the objective leakage is unknown for a circuit, unless it is specified by a user. Therefore, a valid primary input assignment is derived by running the SAT solver through the CNF clauses representing the circuit’s logical behavior only. Based on the leakage values in Table 1, the total leakage h for the circuit is computed using the valid primary input assignment and consequent assignments of the internal gate inputs. The total leakage is decremented by 1 to identify the minimum possible leakage in the circuit. This value denotes the new objective value. A PB constraint expressing the new objective leakage $h - 1$ is added to the problem. The problem is tested using the SAT solver. If the problem is satisfiable, the new total leakage k is computed using the new input assignment and a new PB constraint is added with the new leakage objective $k - 1$. Otherwise, the problem is unsatisfiable and the circuit is proved to have a minimum possible leakage of h . Equivalently, in order to identify the maximum possible leakage, a similar approach is used, but the objective leakage limit is incremented by 1.

The performance of the proposed approach is further improved due to the incremental satisfiability feature in PBS. Ideally, one would need to solve the problem independently from scratch every time the objective leakage is modified. However, the incremental approach provides the ability to modify previously solved problems by the addition or removal of constraints, thereby reusing decision sequences and retaining information learned by the solver from run to run.

We should note that other ways can be used to update the new leakage objective, such as using *binary search*. However, when compared with the monotonic tightening of the leakage constraint, the latter was faster, since most of the learning (i.e. conflict-induced clauses) is relinquished with binary search.

3.4 Example

Let’s consider the circuit shown in Figure 2. The circuit consists of two 2-input NAND gates and an Inverter. There are four possible input combinations to the circuit. Clearly, the minimum leakage is obtained by setting $\{a = 0, b = 0\}$. In order to obtain the minimum leakage using PBS, the CNF clauses representing the circuit’s consistency function are generated. The objective function is initialized to the maximum possible leakage of all gates, which corresponds to $544 + 544 + 271 = 1359$ and all multi-literal terms are replaced by new variables. Suppose, PBS identifies the satisfiable assignment

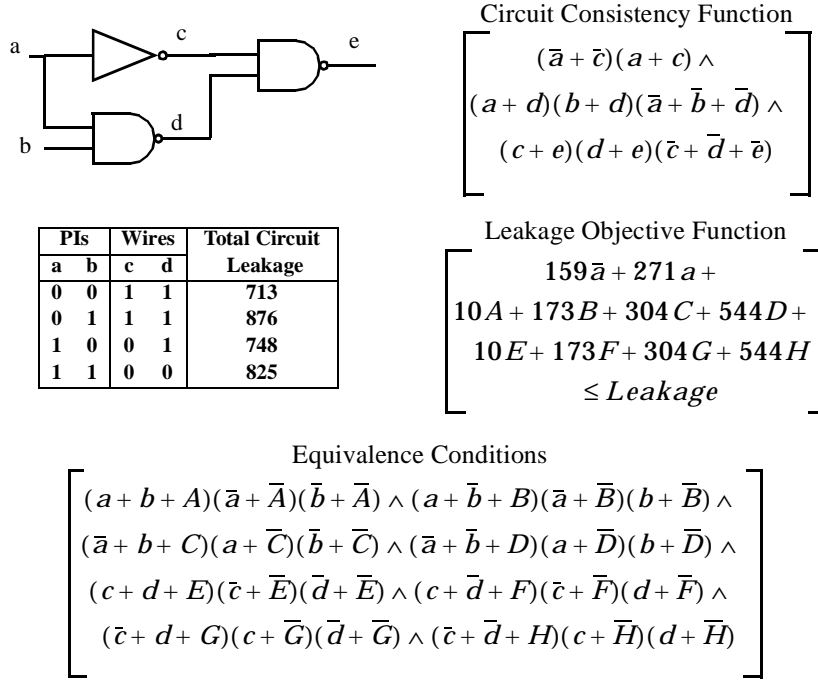


Fig. 2. Illustrative example for identifying the minimum leakage.

$\{a = 1, b = 0, c = 0, d = 1, e = 1\}$., which corresponds to the total leakage of 748. A new PB constraint, with an objective leakage of 747, is added to further reduce the leakage. A satisfiable assignment is identified with values $\{a = 0, b = 0, c = 1, d = 1, e = 0\}$. This corresponds to a leakage of 713. Again, a new PB constraint is added with an objective leakage of 712. PBS fails to identify a satisfiable solution. Hence, the minimum leakage is identified at 713.

4 Experimental Results

In order to evaluate the performance of the approach, we measured the maximum and minimum leakage for the MCNC benchmarks [11]. Each benchmark was sensitized using *sis* [15] to a circuit consisting of only 2-input NAND, NOR and Inverter gates*. All experiments were conducted on a AMD 1.2Ghz, equipped with 512 MBytes of RAM, and running Linux. We used PBS [1] as our SAT solver with the latest enhancements, such as conflict diagnosis and random restarts, enabled. PBS implements two decision heuristics, *FIXED* and *VSIDS* [12]. In practise, both heuristics are competitive depending on the problem's structure. Therefore, we tested each circuit using both heuristics. Our table of results report the best of both runs. The runtime limit for all experiments was set to 5000 seconds.

* Table 1 was used for the gate leakage values.

Table 2. Experimental Results on the MCNC Circuits.

Circuit	# Gates	# PI	Minimum Leakage				Maximum Leakage				Max/Min Difference	
			Random	PBS			Random	PBS			% Actual	
			Leak	Leak	% Imp	Time	Leak	Leak	% Imp	Time		
x2	73	10	11937	11937	0.00	0.15	22718	22718	0.00	0.14	47.46	10781
cm152a	24	11	5536	5536	0.00	0.09	7140	7140	0.00	0.04	22.46	1604
cm151a	39	12	5679	5679	0.00	0.09	11070	11070	0.00	0.11	48.70	5391
cm162a	54	14	8172	8172	0.00	0.18	13611	13611	0.00	0.1	39.96	5439
cu	78	14	11859	11859	0.00	0.3	25293	25293	0.00	0.08	53.11	13434
cm163a	51	16	6820	6820	0.00	0.46	13276	13490	1.61	0.01	49.44	6670
cmb	62	16	8480	8480	0.00	0.58	16021	16484	2.89	0.65	48.56	8004
pm1	67	16	9444	9444	0.00	0.23	19606	19606	0.00	0.33	51.83	10162
parity	75	16	12915	12653	2.03	0.04	14608	15431	5.33	0.01	18.00	2778
tcon	41	17	7112	7112	0.00	0.04	12331	12566	1.91	0.01	43.40	5454
pcle	71	19	13511	13338	1.28	0.01	18117	18381	1.46	3.67	27.44	5043
sct	143	19	20368	19743	3.07	0.76	48958	49262	0.62	1.4	59.92	29519
cc	79	21	11820	11186	5.36	4.4	24512	24987	1.90	0.2	55.23	13801
cm150a	79	21	12855	12373	3.75	26	22687	23527	3.70	10.4	47.41	11154
mux	106	21	20135	19873	1.30	4.57	31586	31884	0.93	5.4	37.67	12011
cordic	124	23	24159	23363	3.29	8.5	30087	31687	5.32	29	26.27	8324
lal	179	26	24852	23273	6.35	3.1	57324	59213	3.30	14	60.70	35940
pcler8	104	27	15698	15303	2.52	3.39	22646	23445	3.53	682	34.73	8142
frg1	143	28	27162	24579	9.51	49.9	40383	41657	3.15	4401	41.00	17078
comp	178	32	31430	28848	8.22	526	41012	46791	14.1	2311	38.35	17943
b9	147	41	24190	21202	12.4	222	42456	46028*	8.41	5000	53.94	24826
i3	132	132	30572	28429	7.01	593	30724	36992	20.4	478	23.15	8563
ttt2	303	24	59429	57302	3.58	83.5	96223	98775	2.65	98.5	41.99	41473
C1355	552	41	143865	126413*	12.1	5000	172857	177057*	2.43	5000	28.60	50644
i6	764	138	147182	139552*	5.18	5000	240500	249871*	3.90	5000	44.15	110319
alu4	878	14	195600	195600	0.00	40	222422	222422	0.00	55	12.06	26822
x3	1174	135	246910	231567*	6.21	5000	339315	348420*	2.68	5000	33.54	116853
vda	1417	17	442553	442497	0.01	2	483430	483531	0.02	65	8.49	41034
C6288	2400	32	739329	720514*	2.54	5000	801441	816729*	1.91	5000	11.78	96215

In order to generate an initial objective goal, we generated 10K random PI vectors and identified the best leakage value among all vectors. The random approach eliminates significant part of the search space and assists in speeding up PBS.

Table 2 lists the maximum and minimum leakage results for the MCNC benchmarks. (Random) represents the best leakage value obtained using the random vector generation approach [7]. (PBS-Leak) represents the final leakage value obtained using PBS. The PBS runtime (in seconds) and the %-improvement (%-Imp) on top of the random leakage value are also reported. The random approach runtime did not exceed a few minutes in most cases. A “*” in the (PBS) leakage column indicates that PBS didn’t complete the search process because it exceeded either the allowed runtime or memory limits. In such a case, the best leakage value measured is shown. All leakage values are reported in units of pA. The table also shows the percentage and actual difference between the minimum and maximum leakage found by PBS. Several observations are in order:

- in almost all reported cases, PBS was able to identify the best possible leakage value.
- in several cases the random approach was unable of identifying the optimal leakage value, especially for large circuits.
- the proposed approach was able to improve on top of the random approach by a factor of 20% in some cases. For example, PBS was able to improve on the value obtained by the random approach by a factor of 12% for the *b9* circuit. Detecting such a difference

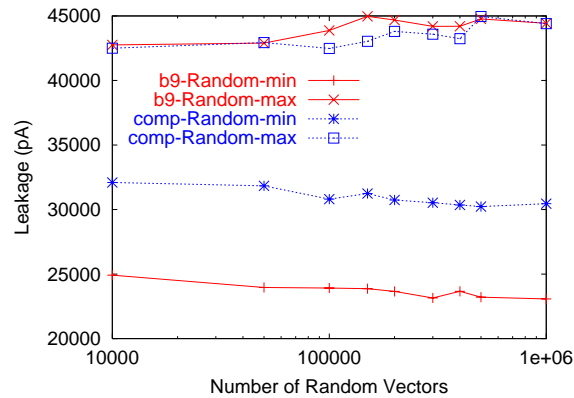


Fig. 3. Minimum and maximum leakage values obtained using random vectors.

in leakage power can be very useful.

- the difference between the minimum and maximum possible leakage can be large, especially for smaller circuits. Identifying techniques to reduce the circuit leakage, as the proposed technique, is crucial.
- PBS is fast for small circuits but as the circuit size grows PBS gets slower. Perhaps, larger circuits can be partitioned to speed up the search process.
- PBS can be viewed as a checker for the random approach.

Figure 3 shows the minimum/maximum leakage values obtained using the random vector generation approach for the *b9* and *comp* circuits. In both cases, the random approach, after generating 1M random vectors, was unable of identifying any maximal leakage values greater than 45K pA, whereas PBS was successful in measuring a maximal leakage value of 46K pA for both circuits. PBS was clearly able to improve on the minimum leakage bound as well.

5 Conclusions

In this paper, we have presented a new approach for determining the minimum or maximum leakage state for CMOS combinational circuits. The maximum leakage state can be used by designers to verify that the circuit meets the required leakage constraints. On the other hand, the minimum leakage state can be useful to reduce the leakage current during stand-by mode. The proposed method searches for the extreme circuit state using a short random search followed by a SAT-based formulation to tighten the leakage bound. The SAT-based problem formulation used both CNF and pseudo-Boolean constraints. To solve the SAT problem, PBS [1] was used since it allows incremental exploration of the solution space. The proposed methods was implemented and tested on a extensive set of circuits. The results show that in most case, the proposed methods can determine the minimum / maximum leakage state. Also, the SAT-based approach was able to obtain significant improvement over the random approach.

Acknowledgments

This work is funded in part by the DARPA/MARCO GigaScale Silicon Research Center and an Agere Systems/SRC Research fellowship.

References

1. F. Aloul, A. Ramani, I. Markov, K. Sakallah, "PBS: A Backtrack-Search Pseudo-Boolean Solver and Optimizer," in *Symp. on the Theory and Applications of Satisfiability Testing*, 346-353, 2002.
2. 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*, 317-320, 1999.
3. S. Bobba and I. Hajj, "Maximum Leakage Power Estimation for CMOS Circuits," in *Proc. of the IEEE Alessandro Volta Memorial Workshop on Low-Power Design*, 1999.
4. S. Borkar. "Design Challenges of Technology Scaling," *IEEE Micro*, 19(4), 23-29, 1999.
5. M. Davis, G. Logemann, and D. Loveland, "A Machine Program for Theorem Proving," in *Journal of the ACM*, (5)7, 394-397, 1962.
6. C. P. Gomes, B. Selman, and H. Kautz, "Boosting Combinatorial Search Through Randomization," in *Proc. of the National Conference on Artificial Intelligence*, 431-447, 1998.
7. J. Halter and F. Najm, "A gate-level leakage power reduction method for ultra-low-power CMOS circuits," in *Proc. of the IEEE 1997 Custom Integrated Circuits Conference*, 475-478, 1997.
8. M. Johnson, D. Somasekhar, and K. Roy, "Models and Algorithms for Bounds on Leakage in CMOS Circuits," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, (18)6, 714-725, 1999.
9. J. Kao, A. Chandrakasan, D. Anotoniadis, "Transistor Sizing Issues and Tool for Multi-Threshold CMOS Technology," in *Proc. of the Design Automation Conference*, 409-414, 1997.
10. J. Marques-Silva and K. Sakallah, "GRASP: A Search Algorithm for Propositional Satisfiability," in *IEEE Transactions on Computers*, (48)5, 506-521, 1999.
11. MCNC Benchmarks, http://www.cbl.ncsu.edu/CBL_Docs/Bench.html
12. M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an Efficient SAT Solver," in *Proc. of the Design Automation Conference*, 530-535, 2001.
13. G. Nam, F. Aloul, K. Sakallah, and R. Rutenbar, "A Comparative Study of Two Boolean Formulations of FPGA Detailed Routing Constraints," in *the Proc. of the International Symposium on Physical Design*, 222-227, 2001.
14. S. Narendra, S. Borkar, V. De, and D. Chandrakasan, "Scaling of Stack Effect and its Application for Leakage Reduction," in *Proc. of the Int'l Symp. on Low Power Electronics and Design*, 2001.
15. E. Sentovich and K. Singh and L. Lavagno and C. Moon and R. Murgai and A. Saldanha and H. Savoj and P. Stephan and R. Brayton and A. Sangiovanni-Vincentelli, "SIS: A System for Sequential Circuit Synthesis," *University of California-Berkeley, UCB/ERL M92/41*, 1992.
16. L. Silva, J. Silva, L. Silveira and K. Sakallah, "Timing Analysis Using Propositional Satisfiability," in *Proc. of the IEEE International Conference on Electronics, Circuits and Systems*, 1998.
17. S. Sirichotiyakul, T. Edwards, C. Oh, J. Zuo, A. Dharchoudhury, R. Panda, and D. Blaauw, "Stand-by Power Minimization through Simultaneous Threshold Voltage Selection and Circuit Sizing," in *Proc. of the Design Automation Conference*, 436-441, 1999.
18. J. Walsor, "Solving Linear Pseudo-Boolean Constraint Problems with Local Search," in *Proc. of the National Conference on Artificial Intelligence*, 1997.
19. J. Whitemore, J. Kim, and K. Sakallah, "SATIRE: A New Incremental Satisfiability Engine," in *Proc. of the Design Automation Conference*, 542-545, 2001.
20. H. Zhang, "SATO: An Efficient Propositional Prover," in *Proc. of the International Conference on Automated Deduction*, 155-160, 1997.
21. A. Chandrakasan, W. Bowhill, F. Fox eds., "Design of High-Performance Microprocessor Circuits," *Piscataway, NJ: IEEE Press*, 2001.