ELSEVIER

Microelectronics Journal 38 (2007) 706–715

Microelectronics Journal

www.elsevier.com/locate/mejo

Using SAT-based techniques in power estimation

Assim Sagahyroon*, Fadi A. Aloul

Department of Computer Engineering, American University of Sharjah, UAE

Received 14 February 2006; received in revised form 8 April 2007; accepted 2 May 2007
Available online 27 June 2007

Abstract

Recent algorithmic advances in Boolean satisfiability (SAT), along with highly efficient solver implementations, have enabled the successful deployment of SAT technology in a wide range of applications domains, and particularly in electronic design automation (EDA). SAT is increasingly being used as the underlying model for a number of applications in EDA. This paper describes how to formulate two problems in power estimation of CMOS combinational circuits as SAT problems or 0–1 integer linear programming (ILP). In these circuits, it was proven that maximizing dissipation is equivalent to maximizing gate output activity, appropriately weighted to account for differing load capacitances. The first problem in this work deals with identifying an input vector pair that maximizes the weighted circuit activity. In the second application we attempt to find an estimate for the maximum power-up current in circuits where power cut-off or gating techniques are used to reduce leakage current. Both problems were successfully formulated as SAT problems. SAT-Based and generic Integer Linear Programming (ILP) solvers are then used to find a solution. The experimental results obtained on a large number of benchmark circuits provide promising evidence that the proposed complete approach is both viable and useful and outperforms the random approach.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: CMOS circuits; Power estimation; Optimization algorithms; Integer linear programming; Boolean satisfiability

1. Introduction

The last few 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 [1], FPGA routing [2], global routing [3], logic synthesis [4], sequential equivalence checking [5], and VLSI testing [6]. 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]. A recent survey of SAT applications in EDA can be found in [8].

SAT solvers have traditionally been used to solve *decision* problems. Given a set of Boolean variables and constraints expressed in products-of-sum form (also known as conjunctive normal form (CNF)), the goal is to identify a variable assignment that will satisfy all constraints in the problem or prove that no such assignment exists. Recently, SAT solvers have been extended to handle pseudo-Boolean (PB) constraints [3,9–13], which are simple inequalities that are equivalent to 0–1 integer linear programming (ILP) constraints. PB constraints are more expressive and can replace an exponential number of CNF constraints. Another key advantage of PB constraints is the ability to express *optimization* problems which are traditionally handled as ILP problems. Hence, SAT solvers can now handle both decision and optimization problems. Recent studies have shown that SAT solvers can compete with the best available generic ILP solvers in solving 0–1 ILP problems arising in specific applications [3,9].

The growing integration scales in VLSI and the recent surge in the deployment and initialization of portable

*Corresponding author.
E-mail address: asagahyroon@aus.edu (A. Sagahyroon).

0026-2692/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mejo.2007.05.001

electronic devices has brought power dissipation to the forefront as a major design concern. While average power dissipation is important in mobile applications where battery life is critical, maximum or peak power is related to circuit reliability. Excessive power dissipation may cause run-time errors and device destruction due to overheating, while excessive instantaneous current through the power and ground (P&G) nets may result in performance degradation due to large voltage drops along the P&G nets, and circuit failures due to electromigration. Hence, the estimation of maximum possible power in VLSI circuits is essential for determining the appropriate packaging and cooling techniques and for optimizing the power and ground routing networks [14–19].

In CMOS combinational circuits, dynamic power dissipation due to the switching current from charging and discharging capacitances can be a major contributor to the total power dissipated in the circuit. This signal switching activity is input pattern dependant. To estimate the maximum switching activity that a circuit might experience, it is necessary to search for a vector pair $\{V1, V2\}$ that tends to maximize this activity. It has been established that finding this vector pair is an NP-complete problem [14].

On the same note and due to the continuing decrease in feature size and increase in chip density, static power dissipation due to leakage current has become a major component of the overall power dissipated by a circuit. One of the techniques used to reduce the effect of leakage current is power-gating [20]. When power gating is used, designers partition the circuit into different blocks, each operating on a block-specific voltage. When not in use a functional block can be put to sleep by shutting off its power supply. This power cut-off gating technique is implemented in CMOS circuits by employing sleep-transistors that can be used to connect or disconnect the path of the block to the power supply V_{dd} or its path to ground. When these circuits blocks are waken-up from their sleep mode, a significant power-on charging current is induced [20]. An estimate of the maximum current during power-up is essential in designing reliable CMOS circuits. However, unlike the maximum switching current discussed earlier which depends on two-input vectors, the maximum wake-up current depends only on one input vector [16]. Hence, to estimate the maximum instantaneous power-up current one need to search for an input vector $V1$ that maximizes the wake-up activity of the circuit.

Even though in recent years we have seen a surge in the application of SAT techniques to assist in finding solutions to various EDA problems, very few researchers reported on the use of SAT or ILP-based techniques in power related research. In [14], Devadas et al. formulated the power dissipation of CMOS circuits as a Boolean function in term of the primary inputs. They attempted to maximize the function by solving a weighted max-satisfiability problem using exact and approximate algorithms. The technique proved to be practically applicable only to small circuits. Leakage power, which also contributes to the total power consumed in a CMOS circuit in addition to the dynamic or switching activity power, was discussed in [21]. In their work, Aloul et al. used a SAT-based approach to determine the minimum or maximum leakage state of a CMOS combinational circuit. Their results showed significant improvement over the random-based approach.

The primary objective of this paper is to demonstrate the validity of using advanced ILP solvers, SAT-based and generic-based, in addressing the two estimation problems discussed above. In this work, we first show how to model the two problems as SAT, i.e. 0–1 ILP, instances, and secondly we evaluate the performance of the latest SAT and generic ILP solvers in handling these instances. We show that generic ILP solvers are likely to achieve better results than SAT-based and random-based approaches when solving the proposed power problems.

This paper is organized as follows. Section 2 includes background information on SAT and ILP. Section 3 describes the estimation problems and their formulation using 0–1 ILP. Experimental results are presented in Section 4 and the paper is concluded in Section 5.

2. Boolean SAT and integer linear programming

Boolean SAT is often used as the underlying model in the field of computer aided designs of integrated circuits. A number of SAT solvers have been proposed and implemented [22–25]. These solvers employ powerful algorithms that are sufficiently efficient to deal with large-scale SAT problems that typically arise in the EDA domain. Most of these algorithms claim competitive results in runtime efficiency and robustness.

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 does not exist, the formula is called *unsatisfiable*. In general, the SAT problem is defined as follows:

Definition 1. Given a Boolean function $f(x_1, \ldots, x_n)$ on n binary variables $x_1, \ldots, x_n$, the SAT problem is concerned with finding an assignment to the variables $\{x_1, \ldots, x_n\}$ that makes the function equal to a constant 1, or proving that the function is equal to constant 0.

The constraints are typically expressed in CNF. In CNF, the formula consists of the conjunction (AND) of m clauses $\omega_1, \ldots, \omega_{\mathrm{m}}$ 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, a CNF instance $f(a,b,c) = (a + \bar{b}) \cdot (a + b + c)$ 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 [36], there have been dramatic improvements in SAT solver technology over the past decade. This has lead to the development of several powerful SAT solvers that are capable of solving problems consisting of thousands of variables and millions of constraints in a few seconds [22–26].

Recently, SAT solvers [3,9–13] have been extended to handle PB constraints which are linear inequalities with integer coefficients that can be expressed in the normalized form [3,35] of

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n \geqslant b, \tag{1}$$

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" [3]. Furthermore, PB extends SAT solvers to handle optimization problems as opposed to only decision problems. Subject to a given set of CNF and PB constraints, one can request the minimization (or maximization) of an objective function which consists of a linear combination of the problem's variables. Note that each CNF constraint can be viewed as a PB constraint. For example the CNF constraint $(a + \bar{b})$ can be viewed as the PB constraint $a + \bar{b} \geqslant 1$. PB constraints represent 0–1 ILP inequalities.

In this paper, we are interested in using advanced ILP solvers to estimate (i) the maximum power dissipation in CMOS combinational circuits due to switching activity and (ii) the maximum power-up current when power-gating techniques are utilized.

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:1

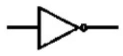
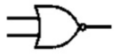
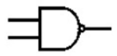
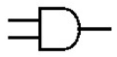
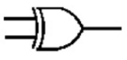
$$\varphi \bigcup_{x \in Q} \varphi_x \tag{2}$$

where Q denotes all gate outputs and primary inputs in the circuit. Table 1 shows generalized CNF formulae for various gates. For example, a NOR gate with inputs x and y and output z is represented using the following set of clauses $(\bar{x} + \bar{z}) \cdot (\bar{y} + \bar{z})(x + y + z)$. 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 description and formulation

The proposed methodology for estimating both, the maximum switching activity and the maximum power-up current operates at or assumes a gate level description of the circuits. In the following section, we elaborate on the estimation problem by providing the needed theoretical framework. Additionally, we describe how to express the estimation problems as SAT instances.

Table 1
CNF formulas representing simple gates

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

3.1. Estimation of weighted maximum switching activity

Power dissipation in CMOS combinational circuits arises from the following sources [17,27]: dynamic power dissipation due to switching current from charging and discharging the parasitic capacitances, dynamic power dissipation due to short-circuit current when both n-channel and p-channel transistors are momentarily on at the same time and static power dissipation due to leakage current and subthreshold current.

The first source, namely dynamic power dissipation, is due to the signal switching activity that takes place during charging and discharging of the capacitors [27]. The signal switching activity depends on the input patterns applied to the circuit. Therefore to determine the maximum switching activity that a circuit can experience, it is necessary to search for a two-vector input sequence $\{V1, V2\}$ that produces the maximum power dissipation [14,28,29]. The exact solution to this power maximization problem is NP-complete [14]. A possible scenario is to exhaustively simulate all possible patterns; however, this is practical for circuits with a small number of primary inputs. For large circuits, several heuristic-based approaches were reported in the literature. In [30], an upper bound of maximum transition or switching density of individual nodes of combinational circuits was computed via propagation of uncertainty waveforms. Test generation-based approaches have also been reported [18]. A test generation strategy is devised for finding test patterns that would produce the maximum node switching corresponding to the maximum power dissipation; the capacitive load of a gate was approximated using its fanout; nodes with large fanout are then assigned transitions which are justified backwards until the primary inputs are reached. In [29], the switching activity maximization problem is shown to be equivalent to a fault-testing problem on a transformed circuit. A maximum weighted activity is achieved by test vectors covering a selected set of faults of the transformed circuit. In [19], a statistical approach based on the asymptotic theory of extreme order statistics was presented. The method is based on the theory of extreme order statistics applied to the probabilistic distributions of the cycle-by-cycle power consumption, the maximum likelihood estimation, and the Monte-Carlo simulation.

As explained earlier, dynamic power can be a major source of energy dissipation in CMOS circuits. The relationship between the logical behavior of a CMOS combinational circuit and the energy that the circuit dissipates is described using the following equation [14]:

$$E = 0.5CV_{\mathrm{dd}}^2 S_{\mathrm{G}}, \tag{3}$$

E is the energy dissipated by the CMOS gate, C is the output capacitance for the gate and S_{G} is the total number of gate output transitions, V_{dd} is the voltage of the power source and also the assumed voltage swing of the node.

To a first degree of approximation, the capacitance C is assumed to be directly proportional to the fanout f of the gate [14,29]. In this work, the effect of interconnect capacitance is ignored. It is clear from the equation above and the assumptions made so far, that the product $(f \cdot S_{\mathrm{G}})$ is directly proportional to the power consumed. Therefore, to maximize dynamic power dissipation, we need to search for an input vector pair $\{V1, V2\}$, that tends to maximize the *weighted* sum of the gates output transitions. The weights are determined by the capacitances or the fanouts of each gate in the circuit. The weighted switching activity (W) of the circuit can be approximated using the following equation:

$$W = \sum_{\mathrm{allgates}} f_i(g_i(V1) \oplus g_i(V2)), \tag{4}$$

where f_i is the fanout of gate g_i, and $g_i(V1)$ is the output of the gate when vector input $V1$ is applied while $g_i(V2)$ is its output when the vector $V2$ is applied at the primary inputs. Here, the summation of "XOR equal to 1" is the number of switching nodes when the input vector $V1$ is followed by input vector $V2$. Note that a zero-delay model is assumed for all the gates in the circuit.

To estimate the maximum power dissipated, it is necessary to identify two-vector input sequence $\{V1, V2\}$ that produces weighted maximum switching activity in the circuit under consideration. In this work, we assume that after the first vector $V1$ is applied, the circuit is allowed to stabilize before the application of the second vector $V2$.

A 0–1 ILP problem is created with following four groups of constraints:

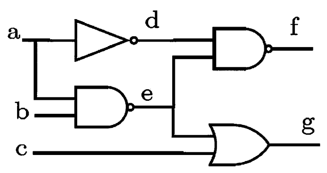
1. A set of CNF constraints representing the circuit's logical behavior after the application of input vector $V1$.
2. A set of CNF constraints representing the circuit's logical behavior after the application of input vector $V2$. Note that the set of constraints in (1) and (2) are identical but the variables are renamed differently.
3. A set of CNF constraints 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. A PB objective constraint which specifies the weights of the XOR outputs. Weights are computed based on the capacitance of the gate which is assumed to be proportional to the fanout of the gate.

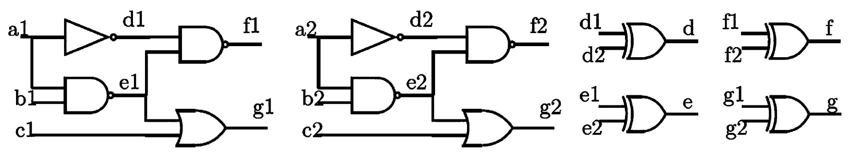
Constraints (1) and (2) represent the circuit's logical behavior following the application of the two vectors respectively. The constraints are represented as explained in Section 2. Constraint (3) compares the output of the same gate for the two vectors. If a transition or a change in the output has 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 Section 2. 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 weighted summation of gate transitions* $>k$" where k is an integer value. In formulating the problem, integer coefficients are used to represent the fanout (capacitance) of each gate.

3.2. An illustrative example

In this subsection we use the circuit shown in Fig. 1 to provide the reader with an example that clearly illustrates the various steps of the proposed approach.

The circuit shown in the example has three primary inputs a, b and c. To generate consistency functions for two circuit instances (Circuit A and Circuit B in the above example), the variables were renamed as a_1, a_2, b_1, b_2, c_1, c_2, d_1, d_2, etc. The CNF clauses representing the circuits' consistency functions are generated. CNF clauses representing the XOR gates between the outputs of gates in circuits A and B are expressed as shown in the output conditions. The objective function consists of the sum of all XOR outputs. Each output is associated with an integer coefficient that is equal to the fanout of the gate. In the given example d, f, and g have a fanout of 1 and e has a fanout of 2. The optimization instance is passed to the ILP solver which returns the assignment: $\{a_1, b_1, c_1, a_2, b_2, c_2\} = \{1, 1, 0, 0, 1, 1\}$. The assignment yields the maximum





Circuit A Consistency Function

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

Circuit B Consistency Function

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

Transition Objective Function

Maximize(d + 2e + f + g)

XORs on Gate Outputs

$$(\overline{d_1}+d_2+d)\cdot(d_1+\overline{d_2}+d)\cdot$$
$$(\overline{d_1}+\overline{d_2}+\bar{d})\cdot(d_1+d_2+\bar{d})\cdot$$
$$(\overline{e_1}+e_2+e)\cdot(e_1+\overline{e_2}+e)\cdot$$
$$(\overline{e_1}+\overline{e_2}+\bar{e})\cdot(e_1+e_2+\bar{e})\cdot$$
$$(\overline{f_1}+f_2+f)\cdot(f_1+\overline{f_2}+f)\cdot$$
$$(\overline{f_1}+\overline{f_2}+\dot{f})\cdot(f_1+f_2+\dot{f})\cdot$$
$$(\overline{g_1}+g_2+g)\cdot(g_1+\overline{g_2}+g)\cdot$$
$$(\overline{g_1}+\overline{g_2}+\bar{g})\cdot(g_1+g_2+\bar{g})$$

Solution:

Max Transitions = 5

$\{a_1, b_1, c_1\} = \{1, 1, 0\}$

$\{a_2, b_2, c_2\} = \{0, 1, 1\}$

Fig. 1. An illustrative example showing how to determine the weighted number of possible transitions in a circuit. (a) Original circuit. (b) Constraints needed to compute the input pair.

possible switching activity that the given circuit can experience.

3.3. Maximum power-up current estimation

As technology scales, leakage power is becoming a growing problem for battery-operated devices and will grow exponentially as power supplies and threshold voltages scale down in future processes.

One of the techniques used in leakage power management is to partition the design into different blocks, where each block operates on a specific voltage. This technique provides designers with the ability to switch off the block supply when the functional block does not have any logic activity to perform. Switching the supply off will minimize leakage current and the power dissipation caused by it. This power cut-off or power gating technique is implemented using a sleep transistor. For example, in CMOS circuits, a PMOS sleep transistor with a high threshold voltage can be used to switch-on or switch-off the V_{dd} supply of a block. Similarly, an NMOS transistor can be used to connect or disconnect the block path to ground. When these circuit blocks are woken-up from their sleep mode, a significant power-on charging current comparable to that of a normal switching current is induced [20]. This current, if excessive, produces surges that may cause L$\mathrm{d}i/\mathrm{d}t$ and IR voltage drops and electromigration. This has a negative impact on circuit reliability and performance. Therefore an estimate of the maximum current during power-up is essential in designing reliable and high performance CMOS combinational circuits. Unlike the maximum switching current which depends on *two* input vectors [17], maximum wake-up current depends only on *one* input vector. Assume that we are using a PMOS sleep transistor and all internal nodes are fully discharged during sleep mode. Therefore, power-up current will be proportional to the total charge that needs to be recovered after wake-up [16]. This power up current is given by

$$P = \sum_{\text{allgates}} (\mathrm{Val}(g)\cdot C(g)\cdot V_{\mathrm{dd}}). \tag{5}$$

In (5), $C(g)$ represents the load capacitance of the gate g, $\mathrm{Val}(g)$ is the logic value of the gate output and V_{dd} is the supply voltage. Assuming that all gates have the same input capacitance and ignoring wire capacitance and assuming $C(g)$ is proportional to the fanout of the gate, the above equation can be simplified to [16]

$$P = \sum_{\text{allgates}} (\text{Val}(g) \cdot \text{Fanout}(g)). \tag{6}$$

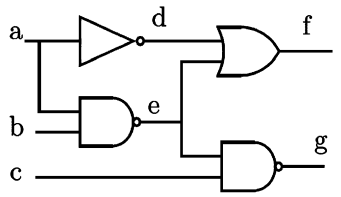
P can be used as a measure of the power up current. Hence, the estimation problem reduces to finding the input vector that maximizes P. A gate with a logical output value of 1 implies that there is a charge of $(C(g) \cdot V_{\text{dd}})$ stored in the load capacitance. A possible solution is to exhaustively simulate all possible input vectors, however, this is impractical for circuits with large number of inputs. In [16,20], algorithms based on automatic test pattern generation (ATPG) techniques were used to find a vector that maximizes P. The results reported were encouraging but further investigation of this problem is still warranted. In this paper, we formulate the estimation problem as a SAT (i.e. 0–1 ILP) instance. We experiment with powerful SAT-based and generic ILP solvers in finding an estimate for the power P. The presented approach is *complete* and identifies the input vector that guarantees the maximum possible value for P.

To estimate the maximum instantaneous power, one needs to search for an input vector that maximizes the weighted wake-up activity of P. The weight of each gate is determined by its fanout as indicated in Eq. (6).

The power estimation problem is formulated as an 0–1 ILP problem as follows: (i) a set of CNF constraints representing the logical behavior of the circuit; and (ii) a PB objective function representing the power-up for each gate.

3.4. An illustrative example

An example is shown in Fig. 2 that clearly illustrates the various steps of the proposed approach. The circuit shown in the example has four gates and three primary inputs. A total of seven variables are needed in the problem. CNF constraints representing the circuit's logical behavior are generated. The objective function consists of the sum of the output of all four gates. Each output is associated with an integer coefficient that is equal to the fanout of the gate.



Circuit Logic (CNF)

$(a + d) \cdot (\bar{a} + \bar{d})$

$(a + e) \cdot (b + e) \cdot (\bar{a} + \bar{b} + \bar{e})$

$(\bar{d} + f) \cdot (\bar{e} + f) \cdot (d + e + \bar{f})$

$(c + g) \cdot (e + g) \cdot (\bar{c} + \bar{e} + \bar{g})$

Objective Function (PB)

$Maximize(d + 2e + f + g)$

Solution:

Max power-up $= 5$

$\{a, b, c\} = \{0, 1, 0\}$

Fig. 2. An illustrative example showing how to determine the maximum power-up in a circuit.

In the given example d, f, and g have a fanout of 1 and e has a fanout of 2. The optimization instance is passed to advanced 0–1 ILP solvers which return the assignment: $\{a, b, c\} = \{0, 1, 0\}$. The assignment yields the input vector that generates a maximum power-up of 5. The goal here is to use the advanced features in 0–1 ILP solvers (e.g. intelligent decision heuristics and learning) to speedup the search for finding the required input vector.

4. Experimental results

In this section, we will report and discuss the experimental results for both problems. The results were obtained using the SAT-based ILP solvers PBS 4.0 [3], Galena [9], and MiniSAT+ [11], in addition to the commercial ILP solver CPLEX 7.0 [31]. PBS, Galena, and MiniSAT+ are considered to be three of the best SAT-based 0–1 ILP solvers and have won several categories in the annual SAT/PB competition. PBS implements advanced CNF learning techniques [23,24]. Galena uses advanced cutting-plane PB learning and post reduction of PB constraints to cardinality constraints. MiniSAT+ uses efficient methods of converting PB constraints to CNF constraints as shown in [32]. CPLEX is a commercial tool that is considered one of the best available generic ILP solvers. It has been used by many researchers to evaluate the performance of SAT-based 0–1 ILP solvers.

All experiments were conducted on an Intel Xeon 3.2 Ghz workstation running Linux with 4 GB of RAM. We used the default settings for PBS, Galena, MiniSAT+, and CPLEX. We used the MCNC [33] benchmark circuits. Each benchmark was sensitized using "sis" [34] into a circuit consisting of 2-input NAND, NOR and inverter gates. The runtime was set to a limit of 1000 s. Note that all solvers perform a complete search, i.e. if an optimal solution is found, no other input vector can return a better solution.

In order to speed up the SAT and ILP solvers by eliminating a large number of input vectors and thus reducing the search space, an initial objective goal was identified by generating 10K random primary input vectors and identifying the (i) maximum weighted switching activity achieved and (ii) maximum power-up current estimate using these randomly generated vectors. The random approach helps eliminate significant parts of the search space as shown in Tables 2 and 3 [15]. The random approach runtime did not exceed one minute in all cases.

4.1. Weighted maximum switching activity results

Table 2 lists the experimental results for PBS, Galena, MiniSAT+, and CPLEX. The first four columns describe the circuit; #PI is the number of primary inputs, #Gates represents the total number of gates in the circuit; column MaxPos gives the theoretical upper bound or the maximum weighted switching activity that can be attained only if all

Table 2
Experimental results for the weighted maximum switching activity experiment using the SAT-based 0–1 ILP solvers PBS, Galena, MiniSAT+, and the generic ILP solver CPLEX

Circuit name	#P1	#Gates	MaxPos	Random	MiniSAT+			PBS4			Galena			CPLEX		
					Time	Value	%-Max	Time	Value	%-Max	Time	Value	%-Max	Time	Value	%-Max
pcle	19	71	92	68	0.32	69	75.0	0.42	69	75.0	0.11	69	75.0	0.48	69	75.0
parity	16	75	89	60	1.38	61	68.5	304.8	61	68.5	29.22	61	68.5	1.1	61	67.4
cc	21	79	104	85	0.3	88	84.6	0.09	88	84.6	0.14	88	84.6	0.59	88	84.6
cm150a	21	79	89	76	0.1	84	94.4	0.02	84	94.4	0.04	84	94.4	0.4	84	94.4
pcler8	27	104	140	90	0.22	110	78.6	1.47	110	78.6	0.62	110	78.6	0.81	110	78.6
mux	21	106	130	101	5.25	115	88.5	0.25	115	88.5	0.31	115	88.5	1.36	115	88.5
cordic	23	124	152	100	3.33	113	74.3	>1K	112	73.7	233	113	74.3	40.3	113	74.3
i3	132	132	132	76	0.09	132	100	0.02	132	100	0.02	132	100	0.03	132	100
frg1	28	143	167	119	1.43	164	98.2	0.01	164	98.2	0.29	164	98.2	0.98	164	98.2
sc1	19	143	198	149	1.29	179	90.4	0.23	179	90.4	0.31	179	90.4	1.64	179	90.4
mnreg	36	145	163	110	0.97	131	80.4	>1K	127	77.9	112	131	80.4	1.34	131	80.4
b9	41	147	183	130	0.21	164	89.6	0.18	164	89.6	0.45	164	89.6	1.45	164	89.6
count	35	161	221	133	0.84	173	78.3	>1K	171	77.4	>1K	171	77.4	1.4	173	78.3
lal	26	179	247	185	0.69	223	90.3	0.9	223	90.3	0.92	223	90.3	2.13	223	90.3
c8	28	211	289	199	9.85	235	81.3	679	235	81.3	173	235	81.3	18.1	235	81.3
i2	201	242	255	115	3.1	251	98.4	0.2	251	98.4	0.18	251	98.4	2.12	251	98.4
cht	47	249	328	229	1.55	295	89.9	>1K	286	87.2	696	295	89.9	5.8	295	89.9
C432	36	282	436	266	103	358	82.1	>1K	358	82.1	>1K	347	79.6	15	358	82.1
apex7	49	295	414	255	22.4	325	78.5	>1K	325	78.5	621	325	78.5	131	325	78.5
ttt2	24	303	425	281	41.4	336	79.1	29.2	336	79.1	66.9	336	79.1	55.6	336	79.1
l4	192	308	308	165	0.33	308	100	0.02	308	100	0.03	308	100	0.05	308	100
example2	85	351	493	279	32.3	353	71.6	>1K	342	69.4	>1K	280	56.8	18.2	353	71.6
l5	133	445	511	288	14.4	511	100	0.67	511	100	>1K	288	56.4	0.11	511	100
term1	34	525	739	485	>1K	485	65.6	>1K	577	78.1	>1K	559	75.6	474	608	82.3
vda	17	1417	2419	665	>1 K	665	27.5	439	689	28.5	490	689	28.5	365	689	28.5
frg2	142	1701	2564	1472	>1K	1472	57.4	>1K	1658	64.7	>1K	1657	64.6	459	1979	77.2
pair	172	1955	2906	1415	>1K	1415	48.7	>1K	1844	63.5	>1K	1415	48.7	>1K	2019	69.5
C6288	32	2400	4271	2202	>1K	2226	52.1	>1K	2311	54.1	>1K	2548	59.7	>1K	2229	47.5
t481	16	4767	7176	3877	>1K	4624	64.4	>1K	4889	68.1	>1K	3955	55.1	>1K	5267	73.4

the gates switched simultaneously. The Random column represents the maximum weighted switching activity obtained using the random vector generation approach [15]. The Time column indicates the runtime (in seconds) for each solver. The column labeled Value represents the maximum weighted activity value obtained using each solver. The %-Max column gives the percentage of the activity reported by the solver (Value) relative to the maximum upper bound (MaxPos). Several observations are in order:

- In all but three cases, CPLEX was successful in computing the optimal weighted switching activity.
- PBS, Galena, and MiniSAT+ were able to compute the optimal weighted switching value in all but 12, 9, and 6 cases, respectively. But for circuits where the solvers timed-out, the solver did return the best weighted switching value seen so far. This can be viewed as a lower bound of the possible optimal weighted switching value. Giving any of the solvers extra time would have probably helped tighten the gap or even solve the problem by finding the optimal value.
- The ILP solvers are fast for most small- and medium-size circuits but as circuits become larger, the ILP solvers become slower. Perhaps, larger circuits can be partitioned to speed up the search process.
- The random approach was unable of identifying the optimal weighted switching activity value in any of the presented instances.
- The proposed approach was able to improve on top of the random approach by a factor of 20%on average and in some cases by 55%. For example, PBS was able to improve on the value obtained by the random approach by a factor of 55% for the *i2* circuit. Detecting such a difference in the weighted switching activity estimation can be very useful (see Fig. 3).
- ILP solvers can thus be used to verify and perhaps improve the optimal weighted switching activity value identified by the random approach.
- In some cases, the optimal weighted switching activity value returned by the two solvers was very close to the theoretical value (MaxPos). In scenarios like these, the architectural aspects of the circuit can be modified or careful attention has to be paid to the cooling techniques used.

Table 3
Experimental results for the maximum power-up current estimation experiment using the SAT-based 0–1 ILP solvers PBS, Galena, and MiniSAT+, and the generic ILP solver CPLEX

Circuit name	#P1	#Gates	MaxPos	Random	MiniSAT+			PBS4			Galena			CPLEX		
					Time	Value	%-Max	Time	Value	%-Max	Time	Value	%-Max	Time	Value	%-Max
pcler8	27	104	140	80	9.09	80	57.1	0.21	80	57.1	0.21	80	57.1	0.01	80	57.1
mux	21	106	130	87	0.09	89	68.5	0.02	89	68.5	0.01	89	68.5	0.01	89	68.5
cordic	23	124	152	96	11.2	98	64.5	6.58	98	64.5	2.03	98	64.5	0.01	98	64.5
frg1	28	143	167	106	1.89	111	66.5	9.63	111	66.5	2.07	111	66.5	0.01	111	66.5
sct	19	143	198	128	0.37	129	65.2	0.04	129	65.2	0.01	129	65.2	0.01	129	65.2
b9	41	147	183	120	3.01	123	67.2	059	123	67.2	0.34	123	67.2	0.01	123	67.2
lal	26	179	247	159	1.34	162	65.6	0.29	162	65.6	0.08	162	65.6	0.01	162	65.6
c8	28	211	289	185	5.55	192	66.4	4.93	192	66.4	0.55	192	66.4	0.01	192	66.4
9sym	9	252	356	229	0.2	230	64.6	0.1	230	64.6	0.02	230	64.6	0.01	230	64.3
C422	36	282	436	263	2.9	274	62.8	32.2	274	62.8	2.82	274	62.8	0.01	274	62.8
m2	24	303	425	267	317	267	62.8	2	267	62.8	2.93	267	62.8	0.02	267	62.8
C880	60	442	616	352	>1K	352	57.1	>1K	355	57.6	>1K	360	58.4	0.09	367	59.6
i5	133	445	511	304	51.1	348	68.1	>1K	332	65.0	>1K	345	67.5	0.04	348	68.1
alu2	10	462	738	425	1.13	426	57.7	0.25	426	57.7	0.09	426	57.7	0.12	426	57.6
term1	34	525	739	477	>1K	477	64.5	82.3	485	65.6	158	485	65.6	0.03	485	65.6
C1355	41	552	894	578	>1K	578	64.7	>1K	579	64.8	>1K	579	64.8	1.19	588	65.8
C499	41	567	891	526	>1K	526	59.0	>1K	526	59.0	>1K	536	60.2	0.19	544	61.1
C1908	33	771	1219	737	>1K	737	60.5	>1K	745	61.1	>1K	742	60.9	0.25	754	61.9
alu4	14	878	1419	802	45.7	802	56.5	6.21	802	56.5	2.27	802	56.5	3.37	802	56.4
C2670	155	1087	1628	963	>1K	963	59.2	>1K	989	60.7	>1K	973	59.8	1.41	1014	52.3
vda	17	1417	2419	1570	>1K	1572	65.0	2.29	1572	65.0	0.22	1572	65.0	0.17	1572	65.0
C3540	50	1526	2538	1456	>1K	1456	57.4	>1K	1486	58.6	>1K	1496	58.9	20.5	1529	60.2
pair	172	1955	2906	1628	>1K	1628	56.0	>1K	1701	58.5	>1K	1683	57.9	0.45	1787	61.5
C6288	32	2400	4271	2833	>1K	2833	66.3	>1K	2833	66.3	>1K	2833	66.3	3.71	2833	66.3
C5315	178	2513	3879	2205	>1K	2205	56.8	>1K	2248	58.0	>1K	2232	57.5	22.1	2368	61.0
l10	257	3366	5454	2936	>1K	2936	53.8	>1K	3002	55.0	>1K	2996	54.9	5.84	3120	57.2
C7552	206	3381	5547	3156	>1K	3156	56.9	>1K	3229	58.2	>1K	3227	58.2	16.6	3306	59.6
i8	133	3764	6504	3818	>1K	3818	58.7	>1K	3818	58.7	>1K	3818	58.7	5.61	3882	59.7
t481	16	4767	7176	4785	>1K	4952	69.0	442	4952	69.0	7.08	4952	69.0	1.11	4952	69.0

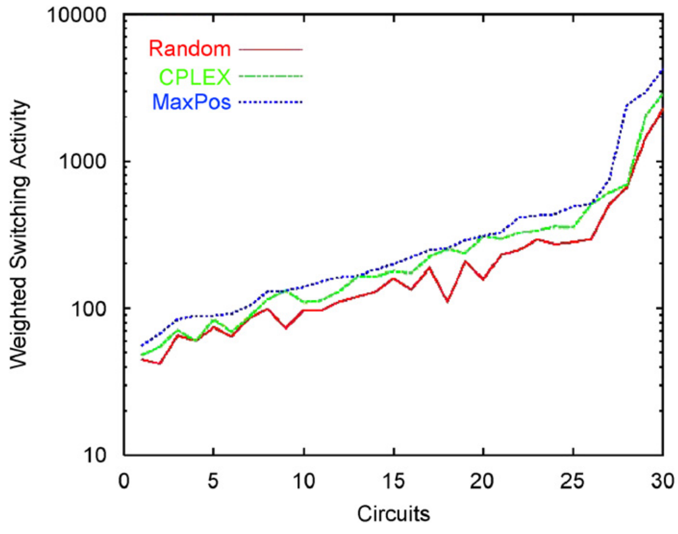


Fig. 3. Maximum values of the weighted switching activity using the random vector generation approach and CPLEX solver. The theoretical upper bound of the weighted switching activity value (MaxPos) is also shown.

- It was also clear in other cases that the maximum possible weighted switching activity value is much smaller than the theoretical value. For example, while the maximum theoretical value for the *vda* circuit was 2419, three solvers are able to prove that the maximum possible weighted switching value is only 689.
- Finally, for the presented application, generic ILP solvers, e.g. CPLEX, perform much better than SAT-based 0–1 ILP solvers, e.g. PBS, Galena, and MiniSAT+.

4.2. Maximum power-up current estimation results

Table 3 summarizes the results for the maximum power-up current estimation problem. The circuit name is provided in the first column. Columns two and three list the number of primary inputs and the number of gates in each circuit. Column four of the table (MaxPos) is the theoretical upper maximum for P (it assumes that all the gates in the circuit will contribute to the summation in Eq. (6)). The Random column represents the best estimate found using random vector generation. The Time column indicates the runtime (in seconds) for each solver. The Value column represents the maximum weighted activity value obtained using each solver. The %-Max column gives the percentage of the activity reported by the solver (Value) relative to the maximum upper bound (MaxPos). While

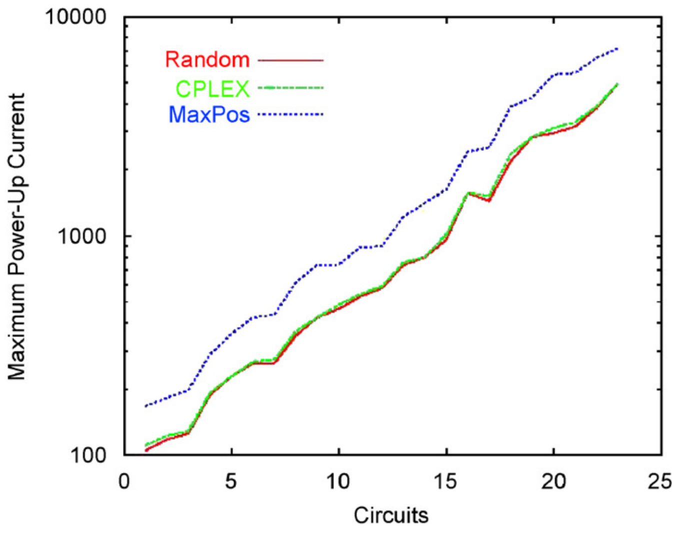


Fig. 4. Maximum values of the power-up current using the random vector generation approach and CPLEX solver. The theoretical upper bound of the power-up current value (MaxPos) is also shown.

PBS, Galena, and MiniSAT+ timed out on some instances, CPLEX was successful in solving all presented instances. Nevertheless, all four search engines succeeded in improving the results obtained using the random approach by returning higher estimates or proving that the estimate returned by the Random approach is optimal (see Fig. 4). In cases where the solver completes the search (times-out), the estimate provides an upper (lower) bound of the maximum wake-up switching activity possible in the circuit.

5. Conclusions

Recently, Boolean satisfiability (SAT) techniques have been drawing wide research interest and has been shown to be successful in various applications in the Electronic Design Automation domain. In this paper we presented a complete 0–1 ILP-based solution to two power estimation problems that have a direct impact on circuit reliability and performance.

We formulated the problems within an Integer Linear Programming (ILP) context and experimented with advanced SAT-based 0–1 ILP solvers, e.g. PBS, Galena, and MiniSAT+, and generic ILP solvers, e.g. CPLEX, to demonstrate that an optimal solution can be found in most cases and in a reasonable amount of time. Results indicate that by using these solvers we can improve on random-based approaches. Furthermore, generic ILP solvers tend to outperform SAT-based 0–1 ILP solvers on the proposed problems. The proposed approach is complete and is guaranteed to find the optimal answer given enough time and memory resources. In many cases, the optimal value obtained by the ILP solvers was significantly smaller than the maximum theoretical value which represents a pessimistic upper bound. Future work will include the extension and investigation of the viability of this approach when applied to sequential circuits.

References

[1] A. Biere, B. A. Cimatti, E. Clarke, M. Fujita, Y. Zhu, Symbolic model checking using SAT procedures instead of BDDs, in: Proceedings of the Design Automation Conference (DAC), 1999, pp. 317–320.
[2] G. Nam, F.A. Aloul, K.A. Sakallah, R. Rutenbar, A comparative study of two boolean formulations of FPGA detailed routing constraints, in: Proceedings of the International Symposium on Physical Design (ISPD), 2001, pp. 222–227.
[3] F.A. Aloul, A. Ramani, I.L. Markov, K.A. Sakallah, generic ILP versus specialized 0–1 ILP: an update, in: Proceedings of the International Conference on Computer-Aided Design (ICCAD), 2002, pp. 450–457.
[4] S. Memik, F. Fallah, Accelerated boolean satisfiability-based scheduling of control/data flow graphs for high-level synthesis, in: Proceedings of the International Conference on Computer Design (ICCD), 2002.
[5] P. Bjesse, K. Claessen, SAT-based verification without state space traversal, in: Proceedings of Formal Methods in Computer-Aided Design (FMCAD), 2000, pp. 372–389.
[6] Z. Zeng, K. Talupuru, M. Ciesielski, "Functional test generation based on word-level SAT, J. System Archit. 51 (2005) 488–511.
[7] N. Creignou, S. Kanna, M. Sudan, Complexity Classifications of Boolean Constraint Satisfaction Problems, SIAM Press, 2001.
[8] J. Marques-Silva, K. Sakallah, Boolean satisfiability in electronic design automation, in: Proceedings of the Design Automation Conference (DAC), 2000, pp. 675–680.
[9] D. Chai, A. Kuehlmann, A fast pseudo-boolean constraint solver, in: Proceedings of the Design Automation Conference (DAC), 2003, pp. 830–835.
[10] H. Dixon, M. Ginsberg, Inference methods for a pseudo-boolean satisfiability solver, in: Proceedings of the National Conference on Artificial Intelligence (AAAI), 2002, pp. 635–640.
[11] N. Een, N. Sorensson, An extensible SAT-solver, in: Proceedings of the International Conference on Theory and Applications of Satisfiability Testing (SAT), 2003, pp. 502–508.
[12] H. Sheini, K. Sakallah, Pueblo: a modern pseudo-boolean SAT solver, DATE 2 (2005) 684–685.
[13] J. Whittemore, J. Kim, K. Sakallah, SATIRE: a new incremental satisfiability engine, in: Proceedings of Design Automation Conference (DAC), 2001, pp. 542–545.
[14] S. Devadas, K. Keutzer, J. White, Estimation of power dissipation in CMOS combinational circuits using Boolean function manipulation, IEEE Trans. on CAD 11 (3) (1992) 373–383.
[15] J. Halter, F. Najm, A gate-level leakage power reduction method for ultra-low-power CMOS circuits, in: Proceedings of the IEEE 1997 Custom Integrated Circuits Conference, 1997, pp. 475–478.
[16] F. Lei, L. Hei, K. Saluja, Estimation of maximum power-up current, ASPDAC, 2002, pp. 51–58.
[17] M. Pedram, Power minimization in IC design: principles and applications, ACM Trans. Design Automat. Electron. Systems 1 (1) (1996) 3–56.
[18] C. Wang, K. Roy, Maximum power estimation for CMOS circuits using deterministic and statistic approaches, in: Proceedings of the 9th Inernational Conference on VLSI Design, 1995, pp. 364–369.
[19] Q. Wu, Q. Qiu, M. Pedram, Estimation of peak power dissipation in VLSI circuits using the limiting distributions of extreme order statistics, IEEE Trans. Computer-Aided Des. Integr. Circ. Systems 20 (8) (2001) 942–956.
[20] F. Lei, L. He, Maximum Current Estimation Considering Power Gating, ISPD, 2001.
[21] F. Aloul, S. Hassoun, K. Sakallah, D. Blaauw, Robust SAT-based search algorithm for leakage power reduction, in: Proceedings of the International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS), 2002, pp. 167–177.

[22] E. Goldberg, Y. Novikov, BerkMin: a fast and robust SAT-solver, in: Proceedings of the Design Automation and Test Conference in Europe (DATE), 2002, pp. 142–149.
[23] J. Marques-Silva, K. Sakallah, GRASP: a search algorithm for propositional satisfiability, IEEE Trans. Comput. 48 (5) (1999) 506–521.
[24] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, S. Malik, Chaff: engineering an efficient SAT solver, in: Proceedings of the Design Automation Conference (DAC), 2001, pp. 530–535.
[25] H. Zhang, SATO: an efficient propositional prover, in: Proceedings of the International Conference on Automated Deduction, 1997, pp. 272–275.
[26] R.J. Bayardo Jr., R.C. Schrag, Using CSP look-back techniques to solve real world SAT instances, in: Proceedings of the 14th National Conference on Artificial Intelligence, 1997, pp. 203–208.
[27] K. Roy, S. Prasad, Low-Power CMOS VLSI Circuit Design, Wiley, 2000.
[28] M.S. Hsiao, Peak power estimation using genetic spot optimization for large VLSI circuits, in: Proceedings of the European Design and Test Conference, 1999, pp. 175–179.
[29] S. Manich, J. Figueras, Maximizing the weighted switching activity in combinational CMOS circuits under variable delay model, in: Proceedings of the European Design and Test Conference, 1997, pp. 597–602.
[30] F. Najm, M. Zhang, Extreme delay sensitivity and the worst-case switching activity in VLSI circuits, in: DAC, 1995, pp. 623–627.
[31] ILOG CPLEX, 〈http://www.ilog.com/products/cplex〉.
[32] N. Een, N. Sorensson, Translating pseudo-boolean constraints into SAT, J. Satisfiab Boolean Model Comput. 2 (2006) 1–26.
[33] MCNC Benchmarks, 〈http://www.cbl.ncsu.edu/CBL_Docs/Bench.htm〉.
[34] E. Sentovich, et al., SIS: A System for Sequential Circuit Synthesis, University of California, Berkeley, 1992 (UCB/ERL M92/41).
[35] P. Barth, A Davis–Putnam based enumeration algorithm for linear pseudo-boolean optimization, Technical Report MPI-I-95-2-003, Max-Planck-Institut Für Informatik, 1995.
[36] S. Cook, The complexity of theorem proving procedures, in: Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, 1971, pp. 151–158.