Maximum Circuit Activity Estimation Using Pseudo-Boolean Satisfiability

Hratch Mangassarian, *Student Member, IEEE,* Andreas Veneris, *Senior Member, IEEE,* and Farid N. Najm, *Fellow, IEEE*

***Abstract*—With lower supply voltages, increased integration densities and higher operating frequencies, power grid verification has become a crucial step in the VLSI design cycle. The accurate estimation of maximum instantaneous power dissipation aims at finding the worst-case scenario where excessive simultaneous switching could impose extreme current demands on the power grid. This problem is highly input-pattern dependent and is proven to be NP-hard. In this work, we capitalize on the compelling advancements in satisfiability solvers to propose a pseudo-Boolean satisfiability-based framework that reports the input patterns maximizing circuit activity, and consequently peak dynamic power, in combinational and sequential circuits. The proposed framework is enhanced to handle unit gate delays and output glitches. In order to disallow unrealistic input transitions, we show how to integrate input constraints in the formulation. Finally, a number of optimization techniques, such as the use of gate switching equivalence classes, are described to improve the scalability of the proposed method. An extensive suite of experiments on ISCAS85 and ISCAS89 circuits confirms the robustness of the approach compared to simulation-based techniques and encourages further research for low-power solutions using Boolean satisfiability.**

***Index Terms*—Maximum Circuit Activity, Peak Dynamic Power, SAT, Pseudo-Boolean Satisfiability.**

I. Introduction

Lower supply voltages, increased integration densities, and higher operating frequencies, among other factors, are producing devices that are more sensitive to power dissipation and reliability problems [1, 2]. Excessive power dissipation can lead to overheating, electromigration and a reduced chip lifetime [3]. Also, large instantaneous power consumption causes voltage drop and ground bounce, resulting in circuit delays and soft errors [4]. Therefore, accurate power estimation during the design phase is crucial to avoid a time-consuming redesign process and in the worst-case an extremely costly tape-out failure [3]. As a result, reliability analysis has steadily become a critical part of the design process of digital circuits.

In CMOS circuits, power dissipation depends on the extent of circuit switching activity, which is input pattern dependent. The maximum circuit activity estimation problem aims at finding the input patterns which cause peak instantaneous dynamic power: A worst-case scenario where excessive simultaneous gate switching imposes extreme current demands on the power grid [1], leading to unwanted voltage drops. This problem is NP-complete for combinational circuits, and PSPACE-complete for sequential circuits [5].

Existing methods for maximum activity estimation can be classified into the two general categories of simulation-based and non-simulative approaches [6]. The former rely on extensive circuit simulations under representative input vectors. On the other hand, non-simulative approaches use characteristics of the circuit and stochastic properties of input vectors to perform power estimation without explicit circuit simulation [6].

In this work, we leverage the advancements and ongoing research in satisfiability-based solvers [19, 20, 22] to tackle this problem using a symbolic approach. Boolean satisfiability (SAT) solvers and their extensions, such as quantified Boolean formula satisfiability (QBF) solvers and pseudo-Boolean satisfiability (PBS) solvers, have become attractive tools for solving theoretically intractable problems in VLSI CAD, in areas such as testing [21], verification [25] and physical design [27]. Furthermore, any improvement to the state-of-the-art in satisfiability solving translates into an immediate benefit to all satisfiability-based solutions.

A pseudo-Boolean satisfiability-based framework is presented for generating tight lower bounds on maximum weighted circuit activity within a clock-cycle [17]. The described framework is applicable to both combinational and sequential circuits, and to both zero and unit gate delay models. Glitches are accounted for in the unit gate delay formulation. As a formal method, our technique may be less scalable than simulation-based frameworks. For this reason, it is intended as a complementary, rather than an alternative, approach to simulations, that can discover "hidden" activity corner-cases. Additionally, the proposed method is not applicable in the presence of variability or uncertainty in gate delays.

In order to disallow unrealistic input transitions and invalid initial states, we show how to integrate input constraints in the problem formulation. Several optimization techniques are presented to improve the scalability of the proposed method. In particular, switching equivalence classes are used to group gates that are most likely to switch in tandem, thus reducing the symbolic problem size.

An extensive suite of experiments on ISCAS85 and ISCAS89 circuits confirms the effectiveness of SAT in a low-

H. Mangassarian is with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada, M5S 3G4 (hratch@eecg.toronto.edu).

A. Veneris is with the Department of Electrical and Computer Engineering and with the Department of Computer Science, University of Toronto, Toronto, ON, Canada, M5S 3G4 (veneris@eecg.toronto.edu).

F. N. Najm is with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON, Canada, M5S 3G4 (f.najm@utoronto.ca).

power analysis application. Coupled with the modeling flexibility offered by SAT and its extensions, this encourages further research in the use of satisfiability-based tools as platforms to solve other low-power problems.

The paper is organized as follows. Section II summarizes previous work. Section III presents background on SAT and PBS. Section IV briefly discusses assumptions and preliminaries. Section V gives the pseudo-Boolean satisfiability formulations for the maximum activity problem in combinational and sequential circuits. Section VI extends the framework to handle glitches due to gate delays. Section VII describes how to apply input constraints to disallow unrealistic input transitions. Section VIII discusses optimizations and heuristics. Section IX shows experimental results and Section X concludes the paper.

II. Previous Work

A number of techniques have been proposed in the literature to estimate the maximum peak power dissipation [9-15] or the maximum instantaneous current [4, 7, 8] of a CMOS circuit. In [4, 7], a loose upper bound on the maximum instantaneous current is generated in linear time by propagating signal uncertainties. This bound is subsequently tightened using a branch-and-bound algorithm that considers spatial signal correlations. Extending the characterization of signal correlations from [4, 7] and the authors in [8] exploit mutually exclusive gate switching to generate tighter upper bounds. However, for larger circuits, the gap between the generated upper bounds and lower bounds obtained using simulations can remain considerable.

In [9], the authors present an Automatic Test Pattern Generation (ATPG) based greedy algorithm that attempts to maximize fanout-weighted gate flips. They also provide a statistical quality measure for the generated lower bounds on maximum circuit activity. In [10], the method is extended to cover sequential circuits as well as glitches. A continuous optimization method is set forward in [11], which treats the Boolean input space as a real-valued vector space and makes use of a gradient based heuristic to estimate the maximum power.

The authors in [12] use genetic algorithms to compare the effect of different delay models on peak power. They conclude that peak power estimated using a zero-delay model is inaccurate, whereas peak power estimated using a unit-delay model is reasonably accurate. In [13], various genetic spot-optimization heuristics are employed to avoid local maxima during the search for maximum single-cycle activities using a variable delay model. [13] reports significant improvements over simulations. Both [12] and [13] are able to handle larger circuits and more general delay models than symbolic approaches such as ours. However, unlike symbolic techniques, they are unable to prove the optimality of their results.

The work presented in [14] considers n independent and identically distributed samples of power-per-cycle. Drawing on the theory of asymptotic extreme order statistics, they model the largest of the n sample values using a Weibull distribution. They then perform a maximum likelihood estimation of the largest power value. The approach of [6] is an extension of [14], which handles more corner cases and uses different statistical distributions. For example, instead of a Weibull-maxima model, [6] uses the so-called Beta-exceedances model. Both of these methods are considered simulative approaches, and are therefore highly input-pattern dependent and lack the exhaustive property of symbolic techniques. However, they can handle any delay model and they scale better than symbolic techniques.

The approach that is closest to this work is given in [15], where the power dissipation of a circuit is modeled as a multi-output Boolean function in terms of the primary inputs. A disjoint cover enumeration as well as a branch-and-bound algorithm are used to maximize the number of weighted gate transitions. An approximation strategy for upper bounding maximum power is also proposed. However, the described techniques can become computationally expensive. Furthermore, sequential circuits are not covered.

The work in [16], published after our original paper [17], proposes the use of temporal and spatial windows in order to split the original maximum activity estimation problem into smaller, more manageable subproblems. They present a high-level algorithm for using *symbolic simulation* to create a symbolic network, which is then translated to a pseudo-Boolean optimization problem. They do not describe the construction of this symbolic network in detail. The addition of spatial and temporal restrictions on the optimization problem in [16] is orthogonal to our work, and provides a viable method to scale activity estimation techniques, including the approach described in this paper. It should be noted that our work [17] was the first to propose such a PBO-based approach to activity estimation.

III. Background

A. Boolean Satisfiability

A *propositional logic formula* Φ can be constructed over a set of Boolean variables using Boolean *connectives* such as $\bar{\ }$ (negation), $\wedge$ (conjunction) and $\vee$ (disjunction). Φ is said to be satisfiable or SAT if it has a *satisfying assignment*: a *truth assignment* to each of its variables that causes it to evaluate to 1. Otherwise, Φ is said to be unsatisfiable or UNSAT. The problem of Boolean satisfiability consists of determining whether Φ is SAT. In modern SAT solvers, the logic formula Φ is given in *Conjunctive Normal Form* (CNF) as a conjunction of *clauses* where each clause is a disjunction of *literals*. A literal is an instance of a variable or its negation. In order for a formula to be SAT, at least one literal in each clause must evaluate to 1. For example, the CNF formula given in (1) is SAT since $\{x_1 = \mathtt{1}, x_2 = \mathtt{0}, x_3 = \mathtt{1}\}$ is a satisfying assignment.

$$\Phi = (x_1 \vee x_2) \wedge (x_1 \vee \bar{x}_2 \vee \bar{x}_3) \wedge (x_3) \tag{1}$$

A logic circuit can be converted to a CNF formula in linear time under reasonable assumptions [21], such that there is a one-to-one correspondence between the variables of the generated CNF formula and the gates of the corresponding circuit, and such that satisfying variable assignments in the CNF formula correspond to valid gate output values in the circuit.

As such, a circuit and its corresponding SAT formulation are often referred to interchangeably in this paper.

Modern SAT solvers implement Conflict-Driven Clause Learning (CDCL) [30]. They are able to solve large industrial SAT problems with millions of variables and clauses. During the search, SAT solvers prune parts of the search-space that do not contain satisfying assignments by analyzing their decisions and learning *conflict clauses*. For example, consider the CNF formula in (1) and suppose that the solver has made the unsatisfiable variable assignments $\{x_1 = \mathtt{0}, x_2 = \mathtt{1}, x_3 = \mathtt{1}\}$. A *conflict* is generated and analyzed by the solver, which realizes that $\{x_1 = \mathtt{0}\}$ cannot be extended to a satisfying assignment, and therefore adds the conflict clause (x_1) to Φ in order to force $\{x_1 = \mathtt{1}\}$.

B. Pseudo-Boolean Satisfiability

A *pseudo-Boolean constraint* over Boolean variables $x_0, x_1, \ldots, x_{n-1}$ is an inequality of the form:

$$\sum_{i=0}^{n-1} c_i \cdot l_i \geq c_n \tag{2}$$

where $c_i \in \mathbb{Z}$ and l_i is a literal corresponding to x_i, i.e. $l_i = x_i$ or $l_i = \bar{x}_i$. Note that a CNF clause is a special case of a pseudo-Boolean constraint with $c_i = 0$ or 1, and $c_n = 1$. A pseudo-Boolean constraint becomes *satisfied* if (2) holds.

A pseudo-Boolean formula Ψ is a conjunction of pseudo-Boolean constraints. The problem of pseudo-Boolean satisfiability (PBS) questions the existence of a truth assignment to $x_0, x_1, \ldots, x_{n-1}$ satisfying all the pseudo-Boolean constraints in Ψ. A pseudo-Boolean optimization (PBO) problem tries to find a satisfiable assignment to a PBS problem Ψ that also minimizes a given *objective function*:

$$\mathcal{F}(\mathbf{x}) = \sum_{i=0}^{n-1} d_i \cdot l_i \tag{3}$$

where $\mathbf{x} = \langle x_0, \ldots, x_{n-1} \rangle$ and $d_i \in \mathbb{Z}$.

For example, given Ψ and $\mathcal{F}$ as shown in (4) below, both $\{x_1 = 1, x_2 = 0, x_3 = 1\}$ and $\{x_1 = 1, x_2 = 0, x_3 = 0\}$ are satisfying assignments. However, the former minimizes $\mathcal{F}$.

$$\begin{aligned} \Psi &= (2x_1 - 3x_2 \geq 1) \wedge (x_1 + x_2 + \bar{x}_3 \geq 1) \\ \mathcal{F} &= \bar{x}_3 - x_1 + 2\bar{x}_2 \end{aligned} \tag{4}$$

The classical approach for solving combinatorial optimization problems, including PBO, has historically been branch-and-bound [31]. In general, these algorithms are able to prune the search tree by using estimates on the value of the optimization function. [31] gives an overview of branch-and-bound techniques for PBO. Motivated by recent advances in SAT solvers, the most effective SAT techniques, including clause learning, lazy data structures and conflict-driven branching heuristics, have been extended to PBO [32]. In this work, we use the PBO solver MINISAT+ [22] which translates pseudo-Boolean constraints to SAT and runs a state-of-the-art SAT solver [20] on the produced SAT instance. The latter approach is particularly suited to problems consisting of mostly SAT clauses and relatively few pseudo-Boolean constraints [22], which is the case in this work. Furthermore, any advancements in SAT solving directly enhances such a strategy.

In MINISAT+, the objective function is minimized using a linear search. MINISAT+ first runs the SAT solver without considering $\mathcal{F}(\mathbf{x})$ in order to get an initial SAT solution $\mathbf{x_0}$, with $\mathcal{F}(\mathbf{x_0}) = k$, where k is the corresponding initial value of the objective function. The new pseudo-Boolean constraint $\mathcal{F}(\mathbf{x}) \leq k - 1$ is subsequently added to the original problem. The SAT solver is then run on the updated CNF formula and this process is repeated until the problem becomes UNSAT. The solution corresponding to the last k before the problem becomes UNSAT is the optimal solution minimizing the objective function.

IV. Assumptions and Preliminaries

Flip-flop-controlled *synchronous* digital circuits are considered. Primary inputs and flip-flop (DFF) outputs can only switch at the beginning of the clock-cycle. This assumption is considered valid in related previous work as well.

The dynamic power dissipation of a CMOS circuit during a clock-cycle can be approximated as follows:

$$P = \frac{1}{2} V_{dd}^2 \sum_{i=1}^{m} C_i \cdot f_i \tag{5}$$

where m is the number of circuit gates, C_i is the capacitive load on gate g_i and f_i is the output *transition count* of g_i during a clock-cycle. Under the assumption that the clock period is sufficiently small, it is sound to interpret (5) as the instantaneous dynamic power during that clock-cycle [9-15]. In the remainder of this work, the terms *circuit activity* and *switched capacitance* refer to the summation in (5) and are used interchangeably.

The following notation is used throughout this paper. T represents a combinational or sequential circuit and $\mathcal{G}(T)$ denotes the set of gates in T excluding primary inputs and states. Symbol m denotes the number of gates in $\mathcal{G}(T)$. Symbols x and s are the Boolean vectors respectively denoting the primary inputs and state elements (DFFs) of a sequential circuit T. Variables x_i and s_i denote the ith primary input and state element of T. Variable g_i refers to ith gate in T and can assume all basic gate types, such as AND, OR, XOR, NOT and BUFFER.

Circuit unrolling consists of replicating the combinational component of a sequential design and connecting the next-state of each time-frame to the current-state of the following time-frame. As will be described later, this process allows the PBO solver to reason on the operation of a sequential circuit. A superscripted variable (e.g. g^j) denotes the copy of that variable in the jth copy of the unrolled circuit T. s^0 denotes the initial-state of T. $\textsc{fanouts}(g_i)$ ($\textsc{fanins}(g_i)$) denotes the set of fanouts (fanins) of g_i. Finally, in all the examples, it is assumed that $C_i = |\textsc{fanouts}(g_i)|$ for internal gates and $C_i = 1$ for primary output gates.

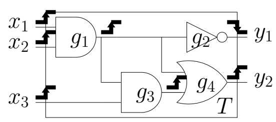
V. Zero-Delay Maximum Activity Computation using PBO

A. Maximum Activity for Combinational Circuits

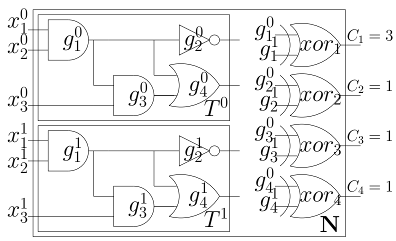
Under a zero-delay model, each gate transition count f_i is a Boolean variable because g_i can flip at most once per clock-cycle. Accordingly, the summation in (5) can be rewritten as:

$$\sum_{i=1}^{m} C_i \cdot \big(g_i(x^0) \oplus g_i(x^1)\big) \tag{6}$$

where x^0 and x^1 are consecutively applied primary input vectors and $g_i(x)$ denotes the steady-state value of gate g_i given primary input vector x.



(a) Zero-delay gate switching



(b) Zero-delay PBO formulation

Fig. 1. Zero-delay PBO formulation for combinational circuits

One needs to find the pair of consecutive primary input vectors $\langle x^{0^*}, x^{1^*}\rangle$ maximizing (6). This problem is formulated as a PBO problem as follows. A new circuit $\mathbf{N}$ is constructed which contains two replicas of the original circuit T, named T^0 and T^1. The primary input vector x^0 (x^1) is applied to T^0 (T^1). Next, every pair of corresponding gates, g_i^0 in T^0 and g_i^1 in T^1, is fed to a new `XOR` gate, called xor_i, in $\mathbf{N}$. Clearly, the output of each xor_i yields $g_i(x^0) \oplus g_i(x^1)$. Fig. 1(b) shows the construction of $\mathbf{N}$ for the circuit given in Fig. 1(a).

Let $\mathrm{CNF}(\mathbf{N})$ denote the translation of $\mathbf{N}$ into CNF clauses (which are also pseudo-Boolean constraints). Clearly, the solution of the following PBO problem maximizes the value of (6):

$$\begin{aligned} \Psi &= \mathrm{CNF}(\mathbf{N}) \\ \mathcal{F} &= -\sum_{i=1}^{m} C_i \cdot xor_i \end{aligned} \tag{7}$$

Note that only the target function $\mathcal{F}$ is not already given as a set of clauses. The pseudo-Boolean formula Ψ, which is simply the CNF of $\mathbf{N}$, markedly suits the choice of the non-native (SAT-based) PBO solver MiniSat+ [22], since the latter is only left to translate $\mathcal{F}$ into CNF.

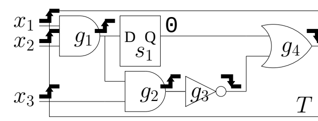
Example 1 *Consider the original circuit T and the corresponding construction $\mathbf{N}$ shown in Fig. 1. Disregarding primary input flips, an optimal solution to the associated PBO problem is $\langle x^{0^*}, x^{1^*}\rangle = \langle\langle 0,0,0\rangle, \langle 1,1,1\rangle\rangle$, which amounts to a total switched capacitance of 6 units by flipping all four gate outputs as shown in Fig. 1(a).*

B. Maximum Activity for Sequential Circuits

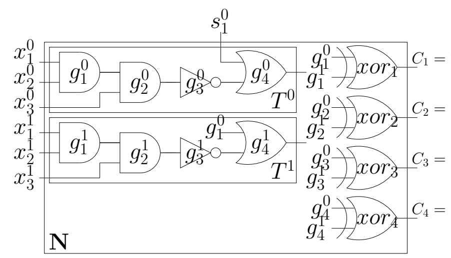
Let $g_i(s^0, x)$ denote the steady-state value of gate g_i given initial-state s^0 and primary input vector x. For a sequential circuit, the transition count f_i depends on both primary input transitions and the initial state. Therefore, estimating the peak power per cycle for sequential circuits is equivalent to finding a triplet $\langle s^{0^*}, x^{0^*}, x^{1^*}\rangle$ consisting of an initial state s^0 and consecutive primary input vectors, x^0 and x^1, that maximizes the following summation:

$$\sum_{i=1}^{m} C_i \cdot \Big(g_i\big(s^0, x^0\big) \oplus g_i\big(s^1, x^1\big)\Big) \tag{8}$$

where s^1 denotes the next-state of the circuit.



(a) Zero-delay switching



(b) Zero-delay PBO formulation

Fig. 2. Zero-delay PBO formulation for sequential circuits

Finding this triplet is formulated as a PBO problem as follows. First, `DFF` inputs (outputs) are transformed into circuit pseudo-outputs (pseudo-inputs). A new circuit $\mathbf{N}$ is constructed which contains two replicas of this *full-scanned* circuit. Moreover, the pseudo-outputs of the first time-frame T^0 are connected to the corresponding pseudo-inputs of the second time-frame T^1. This two time-frame iterative logic array expansion of the original sequential circuit is referred to as *circuit unrolling*. Next, similarly to the combinational case, every pair of corresponding gates, g_i^0 in T^0 and g_i^1 in T^1, is fed to a new `XOR` gate. Clearly, the output of each xor_i yields $g_i\big(s^0, x^0\big) \oplus g_i\big(s^1, x^1\big)$. Fig. 2(b) shows the construction of $\mathbf{N}$ for the sequential circuit given in Fig. 2(a). Note that in this example, $s_1^1 = g_1^0$.

The resulting PBO problem can be expressed by the set of equations in (7), using the above description of the circuit $\mathbf{N}$.

Example 2 *Consider the circuit T and the corresponding $\mathbf{N}$ shown in Fig. 2. Not counting flips at `DFF` outputs (s_1) or primary inputs, an optimal solution to the PBO problem for sequential circuits given in this section is $\langle s^{0^*}, x^{0^*}, x^{1^*}\rangle = \langle\langle 0\rangle, \langle 0,0,0\rangle, \langle 1,1,1\rangle\rangle$, which amounts to a total switched capacitance of 5 units as shown in Fig. 2(a). However, this solution might be suboptimal if gate delays are considered. Section VI describes how delay is integrated into the PBO problem.*

The given problem formulation allows for any initial state and primary input transitions to be returned in the optimal solution. In Section VII, we describe how to add constraints to the PBS problem to disallow certain initial states, as well as illegal or unlikely combinations of primary inputs.

VI. Modeling Delay

Different input signal arrival times might cause a gate to flip several times during one clock-cycle. In fact, glitches due to gate propagation delays can dominate the maximum instantaneous power in some cases [10,12]. On the other hand, empirical results in [12] show that a *unit gate delay* model yields reasonably accurate power estimates. This section discusses the integration of unit gate delay into the problem formulation. It is also explained how this can be extended to arbitrary (but fixed) delay using a preprocessing step.

What follows is applicable to sequential circuits with no combinational loops (sequential loops are obviously allowed), in order to avoid unstable and metastable signals. In other terms, the full-scanned version of the sequential circuit is a Directed Acyclic Graph (DAG). For each DAG node $n_i \in \{x, s, \mathcal{G}(T)\}$, we define its *max-level* $L(n_i)$ and *min-level* $l(n_i)$ as follows:

Definition 1

$$L(n_i) = \begin{cases} \max\limits_{\{n_j \in \text{FANINS}(n_i)\}} L(n_j) + 1 & \textit{if } n_i \in \mathcal{G}(T) \\ 0 & \textit{if } n_i \in \{x, s\} \end{cases}$$

Definition 2

$$l(n_i) = \begin{cases} \min\limits_{\{n_j \in \text{FANINS}(n_i)\}} l(n_j) + 1 & \textit{if } n_i \in \mathcal{G}(T) \\ 0 & \textit{if } n_i \in \{x, s\} \end{cases}$$

$L(g_i)$ and $l(g_i)$ essentially denote the lengths of, respectively, the longest and shortest simple paths to gate g_i, in terms of number of gates, starting from a primary input in x or a pseudo-input in s. Let $\mathcal{L} = \max_{g_i \in \mathcal{G}} L(g_i)$ designate the largest max-level in the circuit. Under a unit-delay model, time t is a discrete variable, meaningful in $\{0, \ldots, \mathcal{L}\}$. Moreover, the signal arrival time at the output of gate g_i following a certain path from a primary input or a `DFF` is equal to the length of the traveled path to gate g_i.

Let $\mathcal{G}_t$ describe the set of all gates whose max-levels and min-levels bound t inclusively.

Definition 3 $\mathcal{G}_t = \big\{ g_i \in \mathcal{G} | l(g_i) \leq t \leq L(g_i) \big\}$

If some gate g_i does not belong to $\mathcal{G}_t$, then either $l(g_i) > t$ or $L(g_i) < t$. The former implies that the shortest signal arrival time from an input or a pseudo-input to a fanin of g_i takes at least t time-steps. So g_i can only flip strictly after time-step t. Similarly, the latter implies that g_i can only flip strictly before time-step t. Therefore, any gate that could potentially flip at time-step t belongs to $\mathcal{G}_t$.

Consider a circuit whose gate logic values have stabilized given initial state s^0 and primary input vector x^0. In a unit-delay framework where each gate requires one time-step to switch, this is equivalent to applying s^0 and x^0 at $t = -1$. The primary input vector x^1 is applied at the start of a new clock-cycle at $t = 0$, and we let $g_i^t(s^0, x^0, x^1)$ denote the value of gate g_i at time-step t. Note that the output value of g_i depends on both x^0 and x^1 because if $t < l(g_i)$, $g_i^t(s^0, x^0, x^1) = g_i(s^0, x^0)$, which is defined in Section V-B. Accordingly, the total switched capacitance can be given by:

$$\sum_{t=1}^{\mathcal{L}} \sum_{g_i \in \mathcal{G}_t} C_i \cdot \left(g_i^{t-1}(s^0, x^0, x^1) \oplus g_i^t(s^0, x^0, x^1) \right) \tag{9}$$

The inner summation in (9) adds the capacitances of the gates whose outputs flip at time t. This summation only checks gates in $\mathcal{G}_t$ and disregards all other gates, because only the gates in $\mathcal{G}_t$ can potentially flip at time-step t. The outer summation adds the total switched capacitances across time-steps $t = 1$ to $\mathcal{L}$.

To maximize (9), we will again construct a new circuit $\mathbf{N}$ that will be used by the PBO solver. In order to do so, we need to create an `XOR` gate for each term in the summation of (9), representing each potential glitch. As such, we need to store the value of each gate at only time-steps when its output value may potentially flip. The remainder of this section describes and proves the correctness of a circuit construction $\mathbf{N}$ that "remembers" all gate flips in T.

This construction is illustrated with the use of an example. Consider the sequential circuit T shown in Fig. 2(a). First, `DFF` inputs (outputs) are transformed into circuit pseudo-outputs (pseudo-inputs). The min-level and max-level of each node can be calculated in linear time by visiting nodes in topological order starting from primary inputs and pseudo-inputs. As a result, the sets $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_{\mathcal{L}}$ can be generated. For the circuit in Fig. 2(a), these sets are as follows:

$$\mathcal{G}_1 = \{g_1, g_2, g_4\}, \mathcal{G}_2 = \{g_2, g_3, g_4\},$$
$$\mathcal{G}_3 = \{g_3, g_4\}, \mathcal{G}_4 = \{g_4\}.$$

For each time-step t, for $0 \leq t \leq \mathcal{L}$, we associate a *time-circuit* T^t containing the following *time-gates*:

$$\mathcal{G}(T^t) = \begin{cases} \{g_i^t | g_i \in \mathcal{G}_t\} & \text{if } t \geq 1 \\ \{g_i^0 | g_i \in \mathcal{G}(T)\} & \text{if } t = 0, \end{cases} \tag{10}$$

as shown in Fig. 3. At the base case, T^0 contains all the gates in T, whereas T^t, for $t \geq 1$, contains every gate that can potentially switch at time t. The new circuit $\mathbf{N}$ in Fig. 3 accommodates all these time-circuits $T^0, T^1, \ldots, T^{\mathcal{L}}$.

Now we describe the gate interconnections in $\mathbf{N}$. The gates of T^0 are interconnected identically to the original full-scanned circuit, given pseudo-input vector s^0 and primary input vector x^0, as shown in Fig. 3. For a time-gate g_i^t in time-circuit T^t, with $t \geq 1$, there are three cases for connecting it to its new inputs, depending on each fanin of the corresponding gate g_i in the original circuit T:

1) If the fanin gate was originally another internal gate, the given time-gate must be connected to the most recent corresponding time-gate *strictly before the current time-step*: No two time-gates in the same time-circuit can be connected because they can only change simultaneously.

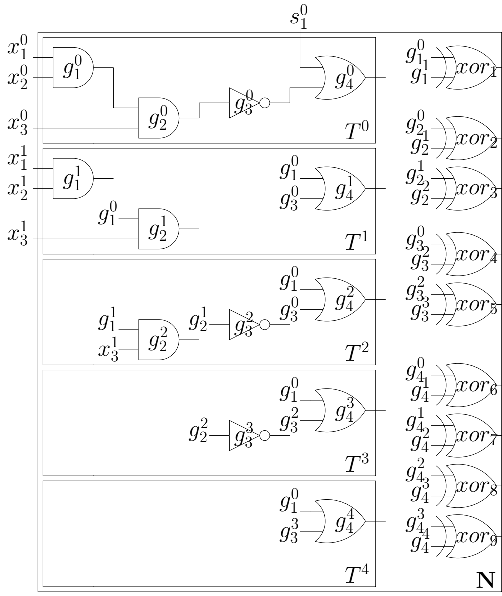


Fig. 3. Unit-delay sequential PBO formulation

2) If the fanin gate was originally a primary input, the given time-gate must be connected to the corresponding new primary input in x^1.
3) If the fanin gate was originally a DFF output, the given time-gate must be connected to the corresponding pseudo-output in T^0.

Formally, consider a gate $g_i \in \mathcal{G}(T)$, such that $\text{FANIN}(g_i) = \{g_\alpha, x_\beta, s_\gamma\}$, where $g_\alpha \in \mathcal{G}(T)$, x_β is a primary input and s_γ is a DFF output. Each of these fanins corresponds to one of the three different cases above. In the new circuit $\mathbf{N}$, for each time-step $t \geq 1$ where g_i^t exists, it will be connected to the following fanins:

$$\text{FANIN}(g_i^t) = \left\{ g_\alpha^{\max\{j | g_\alpha^j \in \mathcal{G}(T^j), j<t\}}, x_\beta^1, \text{FANIN}(s_\gamma)^0 \right\}$$

Here, $\text{FANIN}(s_\gamma)$ denotes the only fanin of s_γ, which is its corresponding pseudo-output, or next-state. $\text{FANIN}(s_\gamma)^0$ denotes the time-gate corresponding to this pseudo-output in T^0 of $\mathbf{N}$.

In what follows, we use the notation $g_i@t$ to denote the value of gate g_i in the original circuit T at time-step t.

Lemma 1 *Given any s^0, x^0, x^1, the time-gate g_i^t in time-circuit T^t of $\mathbf{N}$ holds the value of $g_i@t$ in T, $\forall i$ and $\forall t \geq 0$.*

Proof: The proof uses strong induction on the time-step variable t.

- *Base case:* At time-step $t = 0$, every gate g_i in T assumes its steady-state value given initial state s^0 and primary input vector x^0. Furthermore, time-circuit T^0 in $\mathbf{N}$ is a replica of the original full-scanned circuit of T, with primary inputs set to x^0 and initial state s^0. As such, given any s^0, x^0, x^1, the time-gate g_i^0 in time-circuit T^0 of $\mathbf{N}$ holds the value of $g_i@0$ in T, $\forall i$.
- *Inductive hypothesis:* Given any s^0, x^0, x^1, the time-gate g_i^j in time-circuit T^j of $\mathbf{N}$ holds the value of $g_i@j$ in T, $\forall i$ and $\forall j[0 \leq j < t]$.
- *Inductive step:* Consider a hypothetical gate g_i in T whose fanins cover all three cases for interconnecting time-gates in $\mathbf{N}$. In other terms, let $\text{FANIN}(g_i) = \{g_\alpha, x_\beta, s_\gamma\}$, where $g_\alpha \in \mathcal{G}(T), x_\beta$ is a primary input and s_γ is a DFF output. As discussed, the corresponding time-gate g_i^t in T^t of $\mathbf{N}$ will have $\text{FANIN}(g_i^t) = \left\{ g_\alpha^{\max\{j | g_\alpha^j \in \mathcal{G}(T^j), j<t\}}, x_\beta^1, \text{FANIN}(s_\gamma)^0 \right\}$. We must prove that g_i^t in T^t of $\mathbf{N}$ holds the value of gate $g_i@t$ in T. In order to do so, we must show that the three types of fanins in $\text{FANIN}(g_i^t)$ respectively hold the values of $g_\alpha@t-1$, $x_\beta@t-1$ and $s_\gamma@t-1$.
  1) Consider $g_\alpha^{\max\{j | g_\alpha^j \in \mathcal{G}(T^j), j<t\}}$. By the inductive hypothesis, time-gate $g_\alpha^{\max\{j | g_\alpha^j \in \mathcal{G}(T^j), j<t\}}$ holds the value of gate $g_\alpha @ \max\{j | g_\alpha^j \in \mathcal{G}(T^j), j < t\}$. Furthermore, by definition of $\mathcal{G}(T^j)$ in (10) and $\mathcal{G}_t$, this is the last time-step before t during which gate g_α can flip in the original circuit T. As such, time-gate $g_\alpha^{\max\{j | g_\alpha^j \in \mathcal{G}(T^j), j<t\}}$ holds the value of gate $g_\alpha@t-1$ in T.

  2,3) Consider x_β^1 and $\text{FANIN}(s_\gamma)^0$. Since x_β (s_γ) is set to x_β^1 ($s_\gamma^1 = \text{FANIN}(s_\gamma)^0$) in T at all time-steps $t \geq 0$, clearly x_β^1 ($\text{FANIN}(s_\gamma)^0$) in $\mathbf{N}$ is equal to $x_\beta@j$ ($s_\gamma@j$) in T, $\forall j[0 \leq j < t]$, and in particular for $j = t-1$.

  Since g_i^t in T^t of $\mathbf{N}$ performs the same logic function as g_i in T and its fanins hold the values of the fanins of g_i at time $t-1$, it follows that g_i^t holds the value of $g_i@t$.

■

Lemma 1 shows that the values of the time-gates g_i^t in $\mathbf{N}$ are consistent with the definition of $g_i^t(s^0, x^0, x^1)$ used in (9). As such, the final step is to add an XOR gate for every pair of g_i^t and $g_i^{t'}$, where there exists no $g_i^{t''}$ with $t < t'' < t'$ in $\mathbf{N}$. By construction, this is equivalent to adding an XOR gate between every g_i^t and $g_i^{\max\{j | g^j \in \mathcal{G}(T^j), j<t\}}$. This is shown in Fig. 3. The weighted sum of these XOR gates yields:

$$\sum_{t=1}^{\mathcal{L}} \sum_{g_i \in \mathcal{G}_t} C_i \cdot \left(g_i^{\max\{j | g^j \in \mathcal{G}(T^j), j<t\}}(s^0, x^0, x^1) \oplus g_i^t(s^0, x^0, x^1) \right) \tag{11}$$

Moreover, since time-step $\max\{j | g^j \in \mathcal{G}(T^j), j < t\}$ is by definition the last time-step before t in which gate g_i could have flipped, it follows that the value of $g_i^{\max\{j | g^j \in \mathcal{G}(T^j), j<t\}}(s^0, x^0, x^1)$ is equal to that of $g_i^{t-1}(s^0, x^0, x^1)$. Replacing this in (11) yields (9), which is to be maximized by the PBO solver.

Example 3 *Consider the circuit T in Fig. 4 and the corresponding $\mathbf{N}$ in Fig. 3. Using a unit-delay model and not counting flips at DFF outputs or primary inputs, an optimal solution to the described PBO problem is $\langle s^{0*}, x^{0*}, x^{1*} \rangle = \langle \langle 0 \rangle, \langle 1, 1, 0 \rangle, \langle 0, 0, 1 \rangle \rangle$, which amounts to a total switched capacitance of 6 units as shown in Fig. 4.*

The circuit $\mathbf{N}$ shown in Fig. 3 generates the activity produced by the triplet $\langle s^{0*}, x^{0*}, x^{1*} \rangle = \langle \langle 0 \rangle, \langle 1, 1, 0 \rangle, \langle 0, 0, 1 \rangle \rangle$ as follows. Recall that T^t ($\forall t \geq 1$) does not contain gates

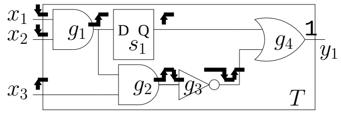


Fig. 4. Unit-delay gate switching in a sequential circuit

that cannot flip at time-step t, by construction. For instance, there is no time-gate corresponding to the inverter g_3 in T^1 because g_3 cannot flip at time-step 1. Furthermore, as shown in Lemma 1, the interconnections in $\mathbf{N}$ are made such that each time-gate g_i^t holds the value of $g_i@t$ in T.

- *In T^0, $g_1^0 = 1, g_2^0 = 0, g_3^0 = 1, g_4^0 = 1$. These values represent the initial state of the circuit (at time-step 0), before x^{1*} starts propagating inside.*
- *In T^1, $g_1^1 = 0, g_2^1 = 1, g_4^1 = 1$. Therefore, $xor_1 = 1, xor_2 = 1, xor_6 = 0$, yielding two gate flips and a total switched capacitance of 3 so far, since $C_1 = 2$.*
- *In T^2, $g_2^2 = 0, g_3^2 = 0, g_4^2 = 1$. Therefore, $xor_3 = 1, xor_4 = 1, xor_7 = 0$, yielding a total switched capacitance of 5 so far.*
- *In T^3, $g_3^3 = 1, g_4^3 = 1$. Therefore, $xor_5 = 1, xor_8 = 0$, yielding a total switched capacitance of 6 so far.*
- *Finally, in T^4, we have $g_4^4 = 1$. Therefore, $xor_9 = 0$ and the total switched capacitance is 6.*

Fig. 4 illustrates all gate switches on the original sequential circuit T.

The procedure outlined in this section can be extended to a more general delay model, where each gate has an arbitrary but fixed delay. This is done as follows. A linear time preprocessing step is described in [10], which generates, for each gate, the sequence of time instants at which it might flip. For each gate g_α, let $t^i_{g_\alpha}$ and $t^f_{g_\alpha}$ respectively denote the first and last time instants at which g_α might flip. A circuit-level time sequence that includes *all* possible gate flipping time instants can be subsequently created. In order to apply the methodology described in this section to an arbitrary delay model, for each gate g_α, $l(g_\alpha)$ and $L(g_\alpha)$ should be respectively set to the *indices* of $t^i_{g_\alpha}$ and $t^f_{g_\alpha}$ in the sorted circuit-level time sequence. Note that this generalization is not applicable if the delay of each gate is variable or is given by an interval of possible values.

It should be noted that using a general delay model would significantly increase the size of the circuit $\mathbf{N}$ because the number of time instants at which the inputs of a given gate can switch scales exponentially with the topological level of the gate (*i.e.*, its min-level) [16]. This is in contrast to unit-delay, where the number of time instants at which the inputs of a given gate can switch scales linearly with its min-level. The authors of [16] propose a viable approach to allow such techniques to handle general gate delays as well as larger circuits, by splitting the problem into smaller subproblems that are easier to solve, using a sequence of spatial and temporal windows. Furthermore, advances in PBO solvers will increase the applicability of our method.

VII. Input constraints

Digital designs might operate under certain assumptions, where some input patterns are considered illegal or unlikely to occur. Not taking these assumptions into account can produce unrealistic maximum activity estimates which can result in an over-conservative design. In this section, we outline how to add input or state constraints to the problem to exclude unwanted input combinations or sequences of input combinations. For instance, the following is an example of such a constraint. Given the initial state $s^0 = \langle 0, 0, X, X \rangle$, the input sequence $\langle x^0, x^1 \rangle = \langle \langle X, 1, 0 \rangle, \langle 1, 0, X \rangle \rangle$ is illegal. Here X denotes a don't-care. This constraint can be translated to the following SAT clause:

$$(s_1^0 \vee s_2^0 \vee \bar{x}_2^0 \vee x_3^0 \vee \bar{x}_1^1 \vee x_2^1)$$

During the PBO search, any assignment of the triplet $\langle s^0, x^0, x^1 \rangle$ that violates the given condition will produce a conflict due to this clause, forcing the solver to backtrack from unwanted parts of the search-tree.

Similarly, sets of unreachable initial-state cubes, possibly including don't-cares, can be ruled out from the search. For instance, the illegal initial-state cubes $s^0 = \langle 1, X, 0 \rangle$ and $s^0 = \langle 1, 1, 1 \rangle$ can be ruled out with the two SAT clauses: $(\bar{s}_1^0 \vee s_3^0) \wedge (\bar{s}_1^0 \vee \bar{s}_2^0 \vee \bar{s}_3^0)$. The interested reader can refer to [34] for a discussion on sequential reachability, which is outside the scope of this work.

Furthermore, it is also possible to rule out input sequences that are deemed as *unlikely*. Here we consider a type of constraint limiting the number of bit flips in the primary inputs to at most d flips. In other terms, the Hamming distance between x^0 and x^1 is constrained to be less than or equal to d:

$$\textstyle\sum_i x_i^0 \oplus x_i^1 \leq d.$$

In this case, a naive encoding into clauses that does not use auxiliary variables will blow up in memory, especially if d is large. In order to express this type of constraints using clauses, first an `XOR` gate $a_i = x_i^0 \oplus x_i^1$ is added in $\mathbf{N}$ for every primary input bit x_i in T. Next, we construct a bitonic sorter [29] using `AND` and `OR` gates, which takes in the Boolean variables $a_1, a_2, \ldots$ and generates the corresponding sorted output sequence in decreasing order, denoted as $b_1, b_2, \ldots$. Finally, the unit clause $(\bar{b}_{d+1})$ is added to the problem, forcing $b_{d+1} = \mathtt{0}$. Since the b_i's are sorted in decreasing order, this will automatically force $b_{d+2} = \mathtt{0}, b_{d+3} = \mathtt{0}, \ldots$. Consequently at most d bits in the output of the bitonic sorter $(b_1, \ldots, b_d)$ can be set to `1` by the solver. Hence, by construction, at most d bits in $a_1, a_2, \ldots$ can be set to `1`, which means that at most d primary inputs can flip simultaneously. This construction requires $O(|x| \log^2 |x|)$ clauses, where $|x|$ denotes the number of primary inputs [22].

VIII. Optimizations and Heuristics

In this section, optimization techniques are presented for improving the PBO formulation.

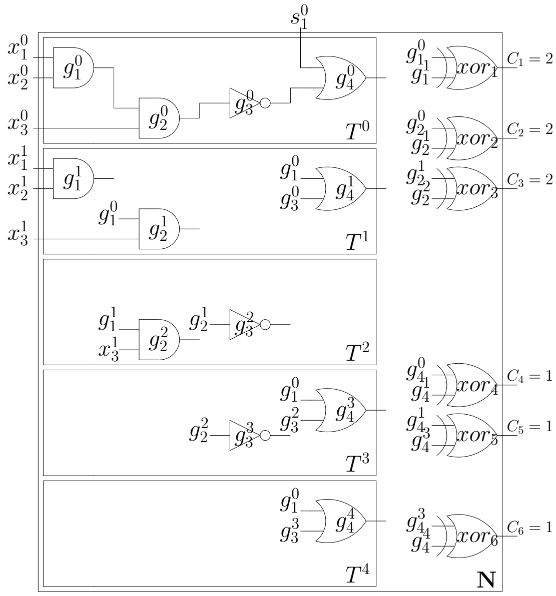


Fig. 5. Optimized PBO formulation, unit-delay model

A. Reduction of $\mathcal{G}_t$

The definition of $\mathcal{G}_t$ given in Definition 3 can be tightened. In fact, irrespective of the triplet $\langle s^0, x^0, x^1 \rangle$, it is sometimes known in advance that a certain gate g_i can never flip at time-step t even though $g_i \in \mathcal{G}_t$. This can happen if $l(g_i) \leq t \leq L(g_i)$, but there exists no path p of length *exactly* t ($|p| = t$) from a primary input or `DFF` output to the output of g_i. For example, in the circuit of Fig. 2(a), although $l(g_4) = 1$ and $L(g_4) = 4$, g_4 can never flip at time-step 2. Hence, in Fig. 3, the time-gate g_4^2 is redundant because its output will always be the same as that of g_4^1. The following is a tighter definition of $\mathcal{G}_t$.

Definition 4 $\mathcal{G}_t = \big\{ g_i \in \mathcal{G}(T) | \exists$ *a path* p *from a primary input or state to* g_i *with* $|p| = t \big\}$

In other terms, $\mathcal{G}_t$ is the set of all gates reachable in exactly t steps from a primary input or a `DFF` output. The sets $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_{\mathcal{L}}$ can be generated using a variation of breadth-first traversal of the original circuit, starting from primary inputs and pseudo-inputs and memorizing the set of newly reached gates at each time-step. In Fig. 5, $\mathbf{N}$ is optimized to use Definition 4 for $\mathcal{G}_t$.

B. Sequences of `BUFFER`s and/or `NOT`s

Suppose gate g_i is a `BUFFER` or a `NOT`. If the input of g_i flips, then the output of g_i flips. Therefore, for every sequence of `BUFFER`s and/or `NOT`s, it is sufficient to put only one `XOR` at the input of the first `BUFFER`/`NOT` and to add the load capacitances of the other gates to the weight of this `XOR`. In Fig. 5, this optimization is used to reduce the number of `XOR`s. For large circuits with significant numbers of `NOT`s and `BUFFER`s, this can significantly reduce the size of the $\mathbf{N}$ as well as $\mathcal{F}$, and therefore the number of pseudo-Boolean constraints.

C. Non-zero Initial Activity using Simulations

As described in Subsection III-B, the PBO solver gradually tightens the upper bound on the objective function, and therefore the lower bound on maximum circuit activity. This is done until either the maximum is proved or the solver times-out. However, instead of starting from an activity of 0, it is possible to first run random simulations for R seconds, record the generated maximum activity M and then force the solver to start from an activity of at least $\alpha \cdot M$, for some user-specified $\alpha \in [0, 1]$, using an appropriate pseudo-Boolean constraint. If α is close to 1, this has the advantage of guiding the solver into parts of the search-space that might potentially yield higher circuit activities, and saves it the time of finding possibly many suboptimal solutions in other parts of the search-space. However, this will make the initial PBS problem harder. Therefore finding the first solution that yields a circuit activity greater than $\alpha \cdot M$ may take a longer time. Moreover, the PBO solver may have a harder time learning from its mistakes.

D. Gate Switching Equivalence Classes

Subsection VIII-B considers a special case of several gates always switching together for sequences of `BUFFER` and/or `NOT` gates. In fact, this may happen in more general cases. Also, due to structural correlations, some gates are more *likely* to switch in tandem. We utilize such gate switching correlations by building equivalence classes for simultaneously switching gates as follows.

We run random simulations for R seconds. Each gate or time-gate is associated with its *switching signature*, which is a sequence of `0`s and `1`s indicating switching times based on these simulations. For each gate, a `1` indicates a switch, while a `0` indicates that no switch occurred for a given input vector. In the case of non-zero gate delays, the switching of each time-gate is recorded (this is equivalent to recording all glitches of a gate during the simulation of the original circuit).

Next, these gates are sorted in their signatures' lexicographical order (*i.e.*, in increasing order of the binary numbers given by these signatures. For example, $001010 < 010001$). Successive gates (or time-gates) with equal signatures are grouped into the same equivalence class. These gates (or time-gates) are assumed to be likely to switch simultaneously, although they are not guaranteed to do so.

In the construction of $\mathbf{N}$, the "switch detecting" `XOR` gates described in this work are only added for one representative gate (or time-gate) in each equivalence class. Furthermore, all the output capacitances of the gates in an equivalence class are now added to the weight of the `XOR` of the representative in the objective function. Less added `XOR`s leads to a smaller $\mathcal{F}$, which results in a smaller PBO to SAT translation in the solver [22], improving the scalability of the technique.

The downside is that the resulting circuit activities returned by the PBO solver can now contain a small error, due to the fact that gates belonging to the same equivalence class might not always switch together. In order to avoid returning "false positive" (*i.e.*, unrealizable) switching activities due to this approximation, we always simulate the resulting solutions

Algorithm 1 Gate Switching Equivalence Classes

```
procedure SolveUsingEquivalenceClasses
    Run random simulations for R seconds to obtain the switching signature of each (time-)gate.
    Sort signatures in lexicographical order and group gates with equal signatures into same equivalence class.
    Pick a representative (time-)gate for each equivalence class.
    In the construction of N, add "switch detecting" XORs only for these representatives.
    Add the output capacitances of every gate in the same equivalence class to the weight of that XOR.
    Run the PBO solver.
    Simulate each returned solution to get the real switching activity.
end procedure
```

(input sequences) returned by the solver and record their real switching activities. Here, we can generate a trade-off based on the original simulation time R to set up the equivalence classes. The longer R, the more accurate the results of the PBO solver will be, but the smaller the size of gate switching equivalence classes and therefore the bigger the problem size. With shorter simulation times, the formulation will be smaller, however there will be more noise in the solution returned by the PBO solver. It should be emphasized that this approximation can miss input transitions that cause more switching in the original circuit and therefore cannot be used to prove that the maximum circuit activity has been found.

IX. Experimental Results

The zero-delay and unit-delay formulations of the proposed PBO-based approach for circuit activity estimation are implemented in C++. The resulting PBO problem is solved using the MiniSat+ [22] engine. The optimizations given in Subsections VIII-A and VIII-B are integrated into the formulation by default, and experiments are run with and without the heuristics described in Subsections VIII-C and VIII-D. All experiments are conducted on a Pentium IV 2.8 GHz Linux platform with 2 GB of memory.

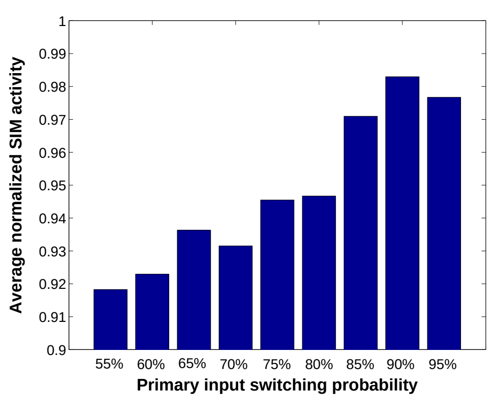


Fig. 6. Normalized SIM activity versus p given 100 seconds

Our approach is compared to parallel-pattern random simulations, referred to as SIM, with 32-bit words (32 simultaneous vector simulations). In SIM, let p denote the (user-specified) probability that a given primary input flips. In other terms, $\forall i$, let $Pr(x_i^0 \neq x_i^1) = p$. We have experimented with several values of p ranging from 55% to 95%, using a time-out of 100 seconds and a representative set of thirty instances from ISCAS85 and ISCAS89 circuits, using both zero and unit-delay models. Fig. 6 shows the results in the form of average normalized SIM activities for each input switching probability p. More precisely, for each instance and each primary input switching probability p, we compute the ratio of the generated activity to the maximum activity for that instance among all switching probabilities. It can be seen that $p = 90\%$ yields the highest average normalized activity (0.983), whereas 95% and 85% respectively yield ratios of 0.977 and 0.971. Lower p's yield inferior activities, with $p = 55\%$ resulting in the lowest average ratio of 0.918. Therefore, the switching probability of each primary input in SIM in the remainder of our experiments is set to 90%. This value is also consistent with [9].

For sequential circuits, SIM continuously picks a new, arbitrary, initial state s^0, and applies x^0, x^1 from s^0. This guarantees a fair comparison with the PBO-based approach, which can also explore arbitrary initial states. At the end of this section, we provide experimental results using input constraints for both of these methods. The time-out is set to $10\,000$ seconds, and the generated sequence of increasing switching activities along with their corresponding run-times is recorded for each experiment. Depending on the size of the circuit, roughly a million to forty million vectors are simulated in $10\,000$ seconds for SIM.

On the other hand, three sets of PBO experiments are performed. The first is the original formulation for combinational or sequential circuits, with zero or unit-delay. This includes the $\mathcal{G}_t$ reduction technique described in Subsection VIII-A, as well as the optimization for `BUFFER`/`NOT` sequences given in Subsection VIII-B. The second set of experiments adds to this the heuristic given in Subsection VIII-C. Here $R = 5$ seconds of random simulations are run to find an initial maximum activity M before our method is applied, and the PBO solver is forced to start from an activity of at least $\alpha \cdot M$ with $\alpha = 0.9$. This value is chosen as a compromise between an α which yields an original problem that is too difficult to solve (e.g. $\alpha \geq 1$) and one that has little effect on the solver. The third set of experiments adds the switching equivalence classes heuristic described in Subsection VIII-D to the original PBO formulation. Here $R = 2$ seconds of random simulations are performed to obtain the switching signatures of each gate or time-gate.

Tables I and II show the experimental results for ten ISCAS85 and twenty ISCAS89 benchmarks, respectively. We describe these two tables simultaneously because they have the same structure. The first and second rows respectively show the circuit names and the corresponding numbers of gates. The maximum circuit activities in Tables I and II are in *units of switched capacitance*, where $C_i = |\textsc{fanouts}(g_i)|$ for internal gates and $C_i = 1$ for primary output gates. For each experiment, the generated maximum activity values are

TABLE I
MAXIMUM ACTIVITIES PER CYCLE OBTAINED BY PBO AND SIM FOR COMBINATIONAL CIRCUITS

			T	**c432**	**c499**	**c880**	**c1355**	**c1908**	**c2670**	**c3540**	**c5315**	**c6288**	**c7552**
			$\lvert\mathcal{G}(T)\rvert$	164	555	381	549	404	709	965	1579	3398	2325
zero delay	**PBO**		$100s$	***193**	441	470	447	430	**753**	**1053**	1395	3023	2064
			$1000s$	***193**	**493**	**482**	**480**	**445**	754	**1054**	**1611**	3191	2282
			$10\,000s$	***193**	**493**	***482**	**480**	456	773	**1058**	**1689**	**3678**	2544
		+VIII-C	$100s$	***193**	462	**476**	**462**	**433**	744	979	**1578**	3078	
		$R=5s$	$1000s$	***193**	471	481	471	441	**764**	1008	1593	**3497**	2236
		$\alpha=0.9$	$10\,000s$	***193**	485	***482**	479	**459**	**775**	1032	1638	3497	**2620**
		+VIII-D	$100s$	**193**	478	474	457	427	727	973	1528	**3449**	2135
		$R=2s$	$1000s$	**193**	482	**482**	473	443	739	1041	1606	3449	**2399**
			$10\,000s$	**193**	485	**482**	**480**	**459**	773	1053	1654	3449	2484
	SIM		$100s$	176	421	414	424	389	657	938	1470	2980	**2179**
			$1000s$	178	421	437	426	389	659	938	1470	3107	2198
			$10\,000s$	178	444	437	426	389	671	950	1476	3195	2222
unit delay	**PBO**		$100s$	838	2172	1330	**2647**	2133	2082	2633	7140	64720	6198
			$1000s$	1006	2713	2508	2770	2176	2467	5096	7140	64720	6198
			$10\,000s$	***1041**	2779	2743	2781	2720	2779	6670	8034	64720	10477
		+VIII-C	$100s$	879	**2494**	1712		**2691**	2466				
		$R=5s$	$1000s$	941	2741	**3103**	**2974**	**2720**	2605				
		$\alpha=0.9$	$10\,000s$	***1041**	2900	**3196**	**2987**	**3329**	2804	**6813**	**8706**	101921	12744
		+VIII-D	$100s$	**902**	2277	**3056**	2487	2510	2503	3809	3895	82055	6195
		$R=2s$	$1000s$	**1032**	**2928**	3056	2562	2524	**2725**	4776	7178	96622	6195
			$10\,000s$	**1041**	**2928**	3056	2909	2896	**2886**	6332	8583	**147010**	**13517**
	SIM		$100s$	880	2292	2366	2234	2500	**2508**	**6540**	**8100**	**118364**	**11699**
			$1000s$	949	2299	2366	2348	2509	2605	**6570**	**8199**	**129672**	**11811**
			$10\,000s$	964	2316	2366	2450	2509	2606	6596	8216	139341	11900

recorded after 100, 1000 and 10 000 seconds. For each circuit and delay model, activities are compared between the four sets of experiments, namely, PBO, PBO+VIII-C with $R = 5$ seconds and $\alpha = 0.9$, PBO+VIII-D with $R = 2$ seconds and finally SIM. The highest activity after each time-period is highlighted in bold. An empty table cell indicates that no bound is found up to that time. Finally, a "$*$" in front of an activity value indicates that the PBO solver *proved* that the generated activity is in fact the absolute maximum. This occurs if the incremental PBS formula becomes `UNSAT`, signaling that no higher circuit activity can be found.

For instance, using a unit-delay model, in circuit c432, both PBO and PBO+VIII-C prove the maximality of 1041 by 10 000 seconds, whereas the maximum activity generated by SIM is (964). Note that for PBO+VIII-D, the activities found by the solver are not recorded directly since they are not guaranteed to be exact due to uncertainties in switching equivalence classes, as described in Subsection VIII-D. Instead, the corresponding input sequences returned by the solver are simulated on the circuit and the resulting activities are recorded. As a result, even when PBO+VIII-D "proves" the maximality of a switching activity, this is not shown using a "$*$" in Tables I and II.

Overall, by the 10 000 second time-out, PBO, PBO+VIII-C and PBO+VIII-D respectively yield 6%, 7% and 7% higher activities than SIM on average. Our approaches yield an average 11% improvement over simulations using a zero-delay model, and 3% improvement using a unit-delay model. All these averages are pushed down by some of the negative results of the PBO-based approaches for the largest circuits, such as s38584. At the 100 and 1000 second marks, SIM slightly outperforms the PBO-based approaches. However, a longer time-out benefits our methods. There exists previous work [9] that provides a statistical quality measure for maximum activities generated using simulations. On the other hand, attempting to estimate a reasonable PBO time-out is a very difficult endeavor because of the nature of SAT and PBO solvers, in the sense that it is difficult to predict their performance. The assumption

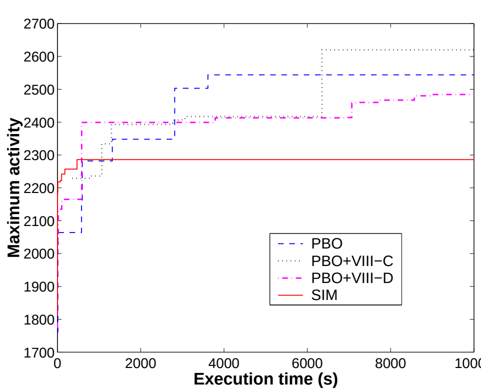


Fig. 7. Maximum activity versus execution time for c7552, zero-delay model

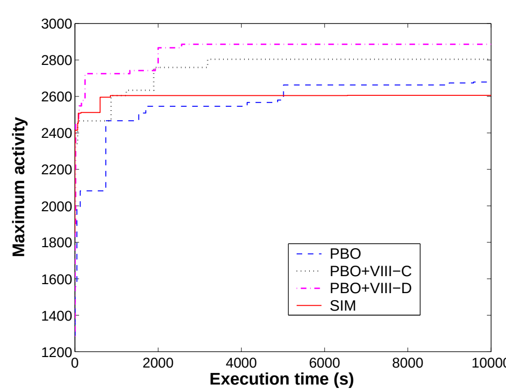


Fig. 8. Maximum activity versus execution time for c2670, unit-delay model

TABLE II
MAXIMUM ACTIVITIES PER CYCLE OBTAINED BY PBO AND SIM FOR SEQUENTIAL CIRCUITS

			T	s208	s298	s344	s382	s386	s444	s526	s820	s832	s953
			$\lvert\mathcal{G}(T)\rvert$	113	148	191	201	172	224	236	432	300	298
zero delay	**PBO**		$100s$	***76**	***139**	***199**	***194**	***203**	***201**	***233**	***364**	***365**	***274**
			$1000s$	***76**	***139**	***199**	***194**	***203**	***201**	***233**	***364**	***365**	***274**
			$10\,000s$	***76**	***139**	***199**	***194**	***203**	***201**	***233**	***364**	***365**	***274**
		+VIII-C	$100s$	***76**	***139**	***199**	***194**	***203**	***201**	***233**	***364**	***365**	***274**
		$R = 5s$	$1000s$	***76**	***139**	***199**	***194**	***203**	***201**	***233**	***364**	***365**	***274**
		$\alpha = 0.9$	$10\,000s$	***76**	***139**	***199**	***194**	***203**	***201**	***233**	***364**	***365**	***274**
		+VIII-D	$100s$	**76**	**139**	**199**	**194**	**203**	**201**	**233**	**364**	**365**	**274**
		$R = 2s$	$1000s$	**76**	**139**	**199**	**194**	**203**	**201**	**233**	**364**	**365**	**274**
			$10\,000s$	**76**	**139**	**199**	**194**	**203**	**201**	**233**	**364**	**365**	**274**
	SIM		$100s$	**76**	**139**	195	188	**203**	198	224	360	361	265
			$1000s$	**76**	**139**	196	190	**203**	198	231	360	361	265
			$10\,000s$	**76**	**139**	196	190	**203**	198	231	360	361	269
unit delay	**PBO**		$100s$	***118**	***195**	***439**	***265**	***267**	***321**	***303**	***465**	***475**	556
			$1000s$	***118**	***195**	***439**	***265**	***267**	***321**	***303**	***465**	***475**	***570**
			$10\,000s$	***118**	***195**	***439**	***265**	***267**	***321**	***303**	***465**	***475**	***570**
		+VIII-C	$100s$	***118**	***195**	***439**	***265**	***267**	***321**	***303**	***465**	***475**	556
		$R = 5s$	$1000s$	***118**	***195**	***439**	***265**	***267**	***321**	***303**	***465**	***475**	***570**
		$\alpha = 0.9$	$10\,000s$	***118**	***195**	***439**	***265**	***267**	***321**	***303**	***465**	***475**	***570**
		+VIII-D	$100s$	**118**	**195**	**439**	**265**	**267**	**321**	**303**	**465**	**475**	533
		$R = 2s$	$1000s$	**118**	**195**	**439**	**265**	**267**	**321**	**303**	**465**	**475**	**570**
			$10\,000s$	**118**	**195**	**439**	**265**	**267**	**321**	**303**	**465**	**475**	**570**
	SIM		$100s$	**118**	**195**	432	262	**267**	320	**303**	463	472	**568**
			$1000s$	**118**	**195**	433	263	**267**	320	**303**	463	473	568
			$10\,000s$	**118**	**195**	433	263	**267**	320	**303**	463	**475**	**570**

			T	s713	s1238	s1423	s1488	s1494	s9234	s13207	s15850	s38417	s38584
			$\lvert\mathcal{G}(T)\rvert$	454	545	806	666	660	6054	9290	10967	25452	22158
zero delay	**PBO**		$100s$	***485**	444	710	***684**	***685**	3010	**3727**	2720	12077	11425
			$1000s$	***485**	460	726	***684**	***685**	4266	**3727**	2720	12077	11425
			$10\,000s$	***485**	***474**	757	***684**	***685**	**4533**	5181	6072	12077	11425
		+VIII-C	$100s$	***485**	432	**713**	***684**	***685**	**3161**				
		$R = 5s$	$1000s$	***485**	460	713	***684**	***685**	4074		5225	9401	10519
		$\alpha = 0.9$	$10\,000s$	***485**	***474**	***770**	***684**	***685**	4404	**5489**	6451	11852	11193
		+VIII-D	$100s$	**485**	**456**	710	**684**	**685**	3010	3671	2633	**12718**	11413
		$R = 2s$	$1000s$	**485**	**469**	**748**	**684**	**685**	**4344**	3671	**6830**	**12718**	11413
			$10\,000s$	**485**	**474**	**770**	**684**	**685**	4433	4905	**7300**	**12718**	11413
	SIM		$100s$	437	451	634	**684**	683	3085	3506	**3995**	10702	**12609**
			$1000s$	439	451	641	**684**	**685**	3085	3562	4089	10929	**12734**
			$10\,000s$	439	451	641	**684**	**685**	3085	3572	4089	11422	**12869**
unit delay	**PBO**		$100s$	667	747	1104	**1450**	1430	3155	4708	3906	**21879**	15522
			$1000s$	1306	844	1483	***1450**	***1450**	3155	4708	3906	**21879**	15522
			$10\,000s$	1696	870	**3848**	***1450**	***1450**	5922	**10779**	3906	**21879**	15522
		+VIII-C	$100s$	847			1440	**1449**		4708	3855		13742
		$R = 5s$	$1000s$	1187		**2484**	***1450**	***1450**		4708	3855	20109	14310
		$\alpha = 0.9$	$10\,000s$	1577	845	3596	***1450**	***1450**	5843	7546	3855	20109	14310
		+VIII-D	$100s$	1425	849	1116	1429	1433	2873	4719	3767	21306	15253
		$R = 2s$	$1000s$	1441	849	1512	**1450**	**1450**	5374	4719	3767	21306	15253
			$10\,000s$	1671	854	3012	**1450**	**1450**	5374	8484	3767	21306	15253
	SIM		$100s$	**1683**	**890**	**1631**	**1450**	1450	**5774**	**7892**	**7324**	20120	**19113**
			$1000s$	**1725**	**897**	1840	**1450**	1450	**5878**	**7959**	**7371**	20293	**19286**
			$10\,000s$	**1731**	**913**	1986	**1450**	1450	**5946**	8638	**8042**	21829	**20509**

is that these power estimation runs can be performed by the engineer overnight and therefore can have reasonably long time-outs, which benefits our method.

Fig. 7 and Fig. 8 show the circuit activities generated by each of the methods for c7552 with zero-delay and c2670 with unit-delay, plotted against execution time. A common observation in these figures, as well as in Tables I and II, is that SIM results tend to plateau, whereas PBO-based approaches continue producing increasing activities. However, in some cases where the size of the symbolic problem starts affecting the performance of the PBO-based approach (e.g. for benchmark s15850), even 10 000 seconds is not enough time for MINISAT+ to improve the maximum activity significantly.

There exist cases where the estimation improvement of PBO-based approaches compared to SIM is considerably large. For instance, using a unit-delay model, in circuit s1423, PBO, PBO+VIII-C and PBO+VIII-D respectively record 94%, 81% and 52% improvements over SIM. This supports the argument that our PBO-based technique, being an exhaustive symbolic approach, complements simulations by occasionally discovering "hidden" corner cases of maximum activity generating stimuli that are missed by SIM.

Benchmark c6288 with unit-delay constitutes a special case because of its disproportionately large number of levels ($\mathcal{L} = 164$), which causes $\mathbf{N}$, and subsequently the CNF of the SAT problem, to be very large. As a result, for c6288 with unit-

TABLE III
SWITCHING EQUIVALENCE CLASSES

	T	c432	c499	c880	c1355	c1908	c2670	c3540	c5315	c6288	c7552
zero	# switch XORs	129	421	302	420	312	576	785	1305	2363	1774
delay	# equivalence classes ($R = 2s$)	129	412	299	413	310	542	730	1273	2359	1679
unit	# switch XORs	1053	3565	3061	3669	4011	3199	9666	11132	143422	19428
delay	# equivalence classes ($R = 2s$)	1041	3284	2733	3316	3522	2616	8066	9437	94638	15899
	T	**s713**	**s1238**	**s1423**	**s1488**	**s1494**	**s9234**	**s13207**	**s15850**	**s38417**	**s38584**
zero	# switch XORs	168	451	526	566	571	2194	3048	3766	10345	12794
delay	# equivalence classes ($R = 2s$)	131	446	481	537	544	1720	2303	2985	9109	11254
unit	# switch XORs	3198	2497	10186	2194	2213	17241	14126	42765	61748	75973
delay	# equivalence classes ($R = 2s$)	1093	1998	2293	1871	1887	5415	4330	6758	28364	26994

delay, we explicitly tell MINISAT+ to use '-adders' (see [22]) in order to save memory, at the expense of performance.

Fig. 9, Fig. 10 and Fig. 11 plot the activities generated by PBO, PBO+VIII-C and PBO+VIII-D, respectively, against those generated by SIM, on a logarithmic scale. These numbers are compared after 100, 1000 and $10\,000$ seconds. In all cases, it can be seen that longer time-outs help PBO over SIM, given that simulation results start to plateau. In Fig. 9, after 100 and 1000 seconds, many points are still below the 45^o line. However, after $10\,000$ seconds, the PBO activities mostly beat SIM activities with some exceptions. For PBO+VIII-C, Fig. 10 and Table I only record the activities found by the PBO solver and do not show the activities generated by the first $R = 5$ seconds of simulations. As expected, the PBO solver sometimes requires more than 100 seconds to find circuit activities exceeding the maximum found by simulations after $R = 5$ seconds. The $10\,000$ second points in Fig. 10 are generally above the 45^o line. Finally, in Fig. 11, the use of equivalence classes improves the scalability of the larger problems. In particular, the activity found using PBO+VIII-D for c6288 with unit-delay is the only one that surpasses that found by SIM within $10\,000$ seconds.

Table III shows the number of switching equivalence classes found using $R = 2$ seconds of simulation (and hence the number of "switch" XOR gates in $\mathbf{N}$) for the ten largest ISCAS89 circuits and all ISCAS85 circuits. This number is compared to the original number of XOR gates in $\mathbf{N}$ without the heuristic of Section VIII-D. One can notice that the reduction in added XORs increases with the circuit size. This is expected, since given a constant simulation time, less vectors can be simulated for a large circuit compared to a small design,

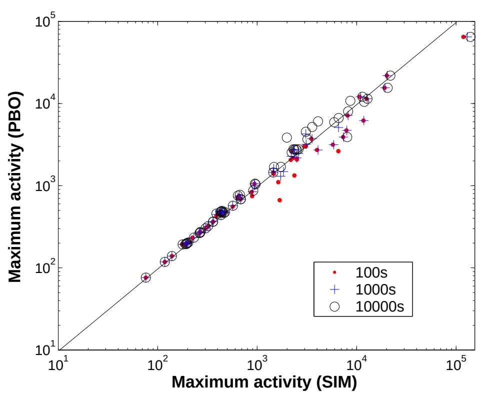


Fig. 9. SIM vs. PBO maximum activities

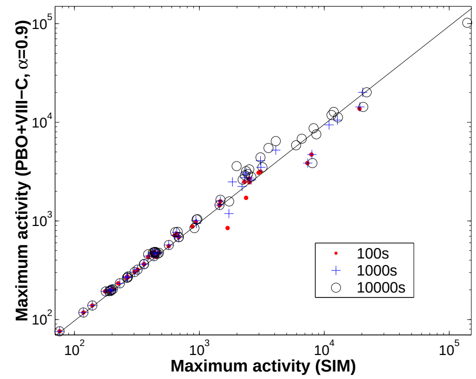


Fig. 10. SIM vs. PBO+VIII-C maximum activities

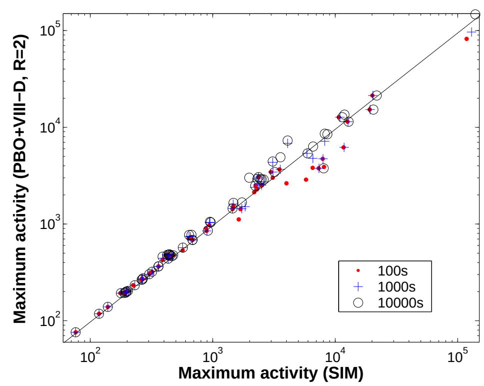


Fig. 11. SIM vs. PBO+VIII-D maximum activities

and therefore gates or time-gates will be less differentiated during simulations, resulting in larger equivalence classes. The argument for doing this is that a reduction is more useful for larger problems in order to extend scalability. The advantage of using switching equivalences classes becomes especially apparent for the c6288 benchmark with unit-delay, where due to the prohibitively large $\mathcal{F}$ in the original formulation, MINISAT+ is unable to improve on its original activity estimates using PBO (64720) and PBO+VIII-C (101921). On the other hand, using PBO+VIII-D, the found maximum activity is improved several times.

In order to observe the effects of a longer time-out, we pick ten circuits, where, using a unit-delay model, SIM either outperforms PBO or yields only slightly lower activities within $10\,000$ seconds, as shown in Tables I and II. We increase the time-out to $50\,000$ seconds, and record the generated activities in Table IV, next to the $10\,000$ second time-out results. Whereas in only 2 of these circuits (s13207 and

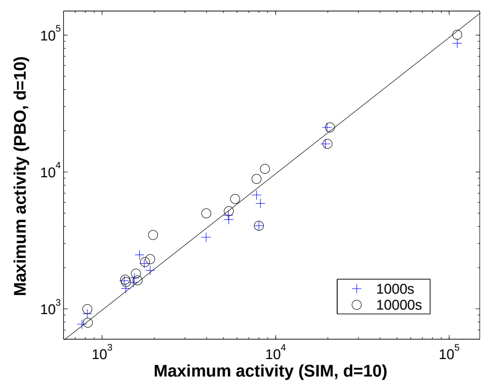


Fig. 12. SIM vs. PBO with at most $d = 10$ input flips, unit-delay model

s38584) PBO outperforms SIM within $10\,000$ seconds, this number increases to 7 within $50\,000$ seconds. Overall, PBO activities increase by 30% on average from the $10\,000$ to the $50\,000$ second mark, whereas SIM activities increase by a mere 1%. This is not surprising because MINISAT+ keeps learning clauses and focusing its search in an attempt to exhaust the problem search-space, whereas SIM continues to blindly apply pseudo-random simulations. Furthermore, PBO is able to find the maximum circuit activity for s713 within $50\,000$ seconds.

However, this indicates that the PBO activities generated within $10\,000$ seconds for the circuits in Table IV are on average *at least* 30% lower than their actual maximum activities. As such, it is useful to investigate the time required by the PBO solver to arrive at reasonably accurate maximum activity estimates. Unfortunately, there is no clear relationship between circuit size and this required time-out, due to the unpredictable behavior of PBO solvers. Furthermore, the incremental improvements in activities returned by the PBO solver do not necessarily become smaller with time. As such, it is not possible to use the sequence of returned PBO activities in a vacuum to determine a time-out. On the other hand, some PBO solvers, *e.g.* MINISAT+, regularly report a *progress* value at run-time, measuring the percentage of the search-space that has been visited or pruned so far. This could be used in determining when to stop the solver. A more robust option would be to use a statistical method such as [6, 14] as a preliminary maximum activity estimation step, which is to be confirmed by an actual input pattern returned by PBO. In this case, the PBO solver would be stopped if an activity close to the statistical estimation has been found, or if some ultimate time-out has been reached.

TABLE IV
PBO VERSUS SIM RESULTS WITH A $50\,000$ SECONDS TIME-OUT

	PBO		SIM	
T	$10\,000s$	$50\,000s$	$10\,000s$	$50\,000s$
c5315	8034	**8633**	**8216**	8427
c6288	64720	111346	**139341**	**142375**
c7552	10477	12569	**11900**	**12713**
s713	1696	***1829**	1731	1739
s1238	870	884	**913**	**913**
s9234	5922	**6157**	**5946**	6005
s13207	**10779**	**11740**	8638	8638
s15850	3906	**8090**	**8042**	8042
s38417	**21879**	**29282**	21829	22085
s38584	15522	**20570**	**20509**	20509

TABLE V
PBO VERSUS SIM RESULTS WITH AT MOST 10 INPUT FLIPS

	PBO		SIM	
T	$1000s$	$10\,000s$	$1000s$	$10\,000s$
c432	**921**	**996**	822	822
c499	**1410**	**1569**	1369	1375
c880	**1912**	2306	1893	1893
c1355	**1604**	**1634**	1342	1358
c1908	**2145**	**2197**	1751	1769
c2670	**1688**	**1808**	1535	1568
c3540	4486	5175	**5372**	**5372**
c5315	3337	**4985**	**3977**	3984
c6288	87107	100683	**111268**	**111268**
c7552	6790	**8895**	**7766**	7766
s713	**1554**	**1615**	1513	1602
s1238	**773**	792	766	**828**
s1423	**2478**	**3465**	1644	1966
s9234	4827	**6349**	**5337**	5842
s13207	5888	**10530**	**8174**	8678
s15850	4049	4049	**8008**	**8008**
s38417	**21154**	**21154**	19777	20556
s38584	16044	16044	**19343**	**19890**

Next, instead of attempting to characterize the reachable states of ISCAS89 circuits as input constraints, we have implemented the non-trivial Hamming distance input constraints, as described in Section VII, as a proof of the applicability and effectiveness of input constraints using PBO. Obtaining a realistic set of valid states for the ISCAS89 circuits is beyond the scope of this work. Table V shows the results of PBO versus SIM with a unit-delay model, constrained to at most $d = 10$ primary input flips, for the ISCAS benchmarks that have at least 10 primary inputs. The resulting activities are expectedly generally lower than those in Table I because of the added restrictions on input switching. Fig. 12 plots the PBO versus SIM activities with input constraints of at most $d = 10$ bit flips on a logarithmic scale. Again, our PBO-based improves with increasing time-outs. After $10\,000$ seconds, PBO generates 10% higher activities than SIM on average.

X. CONCLUSION

This work proposes a pseudo-Boolean satisfiability-based framework for finding the input sequence that maximizes single-cycle circuit activity. The method is extended to take into account multiple gate transitions during a clock-cycle. The integration of various external input constraints into the symbolic problem formulation is also described. Several optimizations are presented, such as the grouping of gates that are likely to switch in tandem into switching equivalence classes. The experimental results on ISCAS benchmarks show 7% higher activities on average compared to parallel-pattern random simulations within $10\,000$ seconds, and further improvements with longer time-outs.

The theory and results of this paper confirm the need for further research in satisfiability-based solutions for problems in low-power design, especially given the tremendous rate of advancement in satisfiability engines and their extensions.

REFERENCES

[1] P. M. Morgado, P. F. Flores and L. M. Silveira, “Generating realistic stimuli for accurate power grid analysis,” *ACM TODAES*, Vol. 14, No. 3, pp. 1-26, May 2009.

[2] M. Xakellis and F. Najm, "Statistical estimation of the switching activity in digital circuits," *IEEE DAC*, pp. 728-733, 1994.

[3] F. Najm, "A survey of power estimation techniques in VLSI circuits," *IEEE Trans. on VLSI*, Vol. 2, No. 4, pp. 446-455, Dec. 1994.

[4] H. Kriplani, F. Najm and I. Hajj, "Maximum current estimation in CMOS circuits," *IEEE DAC*, pp. 2-7, 1992.

[5] A. T. Freitas, H. C. Neto and A. L. Oliveira, "On the complexity of power estimation problems," *ILWS*, pp. 176-181, 2004.

[6] N. E. Evmorfopoulos, G. I. Stamoulis and J. N. Avaritsiotis, "A Monte Carlo approach for maximum power estimation based on extreme value theory", *IEEE trans. On CAD*, Vol. 21, No. 4, April 2002, pp 415-432.

[7] H. Kriplani, F. Najm, P. Yang and I. Hajj, "Resolving signal correlations for estimating maximum currents in CMOS combinational circuits," *IEEE DAC*, pp. 2-7, 1993.

[8] C.-T. Hsieh, J.-C. Lin and S.-C. Chang, "A vectorless estimation of maximum instantaneous current for sequential circuits," *IEEE ICCAD*, pp. 537-540, Nov. 2004.

[9] C.-Y Wang and K. Roy, "Maximum power estimation for CMOS circuits using deterministic and statistical approaches," *IEEE Trans. on VLSI*, Vol. 6, pp. 134-140, 1998.

[10] C.-Y Wang, K. Roy, "Estimation of maximum power for sequential circuits considering spurious transitions," *IEEE ICCD*, pp. 746-751, 1997.

[11] C.-Y Wang, K. Roy, "COSMOS: a continuous optimization approach for maximum power estimation of CMOS circuits," *IEEE ICCAD*, pp. 52-55, 1997.

[12] M. S. Hsiao, E. M Rudnick and J. H. Patel, "Effects of delay models on peak power estimation of VLSI sequential circuits," *IEEE ICCAD*, pp. 45-51, Nov. 1997.

[13] M. S. Hsiao, "Peak power estimation using genetic spot optimization for large VLSI circuits," *IEEE DATE*, pp. 175-179, Nov. 1999.

[14] Q. Wu , Q. Qiu and M. Pedram, "Estimation of peak power dissipation in VLSI circuits using the limiting distributions of extreme order statistics," *IEEE Trans. on CAD*, Vol. 20, No. 8, pp. 942-956, Aug. 2001.

[15] S. Devadas, K. Keutzer and J. White, "Estimation of power dissipation in CMOS combinational circuits using boolean function manipulation," *IEEE Trans. on CAD*, pp. 373-383, Mar. 1992.

[16] P. M. Morgado, P. F. Flores, J. C. Monteiro and L. M. Silveira, "Generating worst-case stimuli for accurate power grid analysis," *PATMOS*, pp. 247-257, Sep. 2008.

[17] H. Mangassarian, A. Veneris, S. Safarpour, F. N. Najm and M. S. Abadir, "Maximum circuit activity estimation using pseudo-Boolean satisfiability," *DATE*, pp. 1538-1543, Apr. 2007.

[18] J. P. Marques-Silva and K. A. Sakallah, "GRASP – A search algorithm for propositional satisfiability," *IEEE Trans. on comput.*, Vol. 48, No. 5, pp. 506-521, May 1999.

[19] M. H. Moskewicz, C. F. Madigan, Y. Zhao and L. Zhang, "Chaff: Engineering an efficient SAT solver," *IEEE DAC*, pp. 530-535, jun. 2001.

[20] N. Eén and N. Sörensson, "An extensible SAT-solver," *SAT*, pp. 502-518, 2003.

[21] T. Larrabee, "Test pattern generation using-Boolean satisfiability," *IEEE Trans. on CAD*, Vol. 11, No. 1, pp. 4-15, 1992.

[22] N. Eén and N. Sörensson, "Translating pseudo-boolean constraints into SAT," *JSAT*, Vol. 2, pp. 1-26, 2006.

[23] F. Aloul, A. Ramani, I. Markov and K. Sakallah, "PBS: A backtrack search pseudo-Boolean solver," *SAT*, 2002.

[24] H. Sheini and K. Sakallah, "Pueblo: A hybrid pseudo-boolean SAT solver," *JSAT*, Vol. 2, pp. 157-181, 2006.

[25] R. Drechsler, *Advanced Formal Verification*. Kluwer Academic Publishers, 2004.

[26] A. Smith, A. Veneris, M. F. Ali, and A. Viglas, "Fault diagnosis and logic debugging using Boolean satisfiability," *IEEE Trans. on CAD*, Vol. 24, No. 10, pp. 1606-1621, 2005.

[27] R. G. Wood and R. A. Rutenbar, "FPGA routing and routability estimation via Boolean satisfiability," *IEEE Trans. on VLSI*, Vol. 6, No. 1, pp. 222-231, Jun. 1998.

[28] M. Davis, G. Logemann and D. Loveland, "A machine program for theorem proving," *Comm. of ACM*, Vol. 5, No. 7, pp. 394-397, 1962.

[29] O. Bailleux and Y. Boufkhad, "Efficient CNF encoding of Boolean cardinality constraints," *CP*, Vol. 2833, LNCS, 2003.

[30] J. Marques-Silva, I. Lynce and S. Malik, "Conflict-Driven Clause Learning SAT Solvers", *Handbook of Satisfiability*, IOS Press, pp. 131-153, 2009.

[31] O. Roussel and V. Manquinho, "Pseudo-Boolean and Cardinality Constraints", *Handbook of Satisfiability*, IOS Press, pp. 695-733, 2009.

[32] V. Manquinho and J. Marques-Silva, "On Using Cutting Planes in Pseudo-Boolean Optimization", JSAT 2(1-4), pp. 209-219, 2006.

[33] L. Guerra E Silva, J. Marques-Silva, L. M. Silveira and K. Sakallah, "Satisfiability Models and Algorithms for Circuit Delay Computation," *ACM Trans. Des. Autom. Electron. Syst.*, Vol. 7, No. 1, pp. 137-158, Jan. 2002.

[34] R. Drechsler, *Advanced Formal Verification*. Kluwer Academic Publishers, 2004.

Hratch Mangassarian (S'01) received the B.E. degree in Computer and Communications Engineering with high distinction from the American University of Beirut in 2005, and the M.A.Sc. degree in Electrical and Computer Engineering from the University of Toronto in 2008, where he is currently working toward the Ph.D. degree. His research is on formal verification and automated design debugging of digital designs, as well as QBF and its applications to CAD.

Andreas Veneris (S'96-M'99-SM'05) received a Diploma in Computer Engineering and Informatics from the University of Patras in 1991, an M.S. degree in Computer Science from the University of Southern California, Los Angeles in 1992 and a Ph.D. degree in Computer Science from the University of Illinois at Urbana-Champaign in 1998. In 1998 he was a visiting faculty at the University of Illinois until 1999 when he joined the Department of Electrical and Computer Engineering and the Department of Computer Science at the University of Toronto where today he is a Professor. His research interests include CAD for debugging, verification, synthesis and test of digital circuits/systems, and combinatorics. He has received several teaching awards, a best paper award and two best paper nominations. He is the author of one book and he holds three patents.

He is a member of ACM, IEEE, AAAS, Technical Chamber of Greece, Professional Engineers of Ontario and The Planetary Society

Farid N. Najm (S'85-M'89-SM'96-F'03) is a Professor in the department of electrical and computer engineering (ECE) at the University of Toronto, Ontario, Canada. He received the B.E. degree in electrical engineering from the American University of Beirut in 1983 and the Ph.D. degree in ECE from the University of Illinois at Urbana-Champaign (UIUC) in 1989. From 1989 to 1992, he worked with Texas Instruments, Dallas, Texas. He then joined the ECE department at UIUC as an Assistant Professor and became Associate Professor in 1997. In 1999, he joined the ECE department at the University of Toronto, where he is now Professor and Chair.

Dr. Najm is a Fellow of the IEEE and of the Canadian Academy of Engineering (CAE). He was Associate Editor for the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS from 2001 to 2009. He has received an IEEE Transactions on CAD Best Paper Award, an NSF Research Initiation Award, an NSF CAREER Award, and was an Associate Editor for the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS from 1997 to 2002. He serves on the executive committee of the International Symposium on Low-Power Electronics and Design (ISLPED), and has served on the technical committees of various conferences, including ICCAD, DAC, CICC, ISQED, and ISLPED. In 2010, Dr. Najm authored the book *Circuit Simulation* (John Wiley and Sons, New York). His research is on CAD for VLSI, with an emphasis on circuit level issues related to power, timing, variability, and reliability.