Efficient Analog Circuits for Boolean Satisfiability

Xunzhao Yin, *Student Member, IEEE*, Behnam Sedighi, Melinda Varga, Mária Ercsey-Ravasz, Zoltán Toroczkai, and Xiaobo Sharon Hu, *Fellow, IEEE*

***Abstract*— Efficient solutions to nonpolynomial (NP)-complete problems would significantly benefit both science and industry. However, such problems are intractable on digital computers based on the von Neumann architecture, thus creating the need for alternative solutions to tackle such problems. Recently, a deterministic, continuous-time dynamical system (CTDS) was proposed [1] to solve a representative NP-complete problem, Boolean Satisfiability (SAT). This solver shows polynomial analog time-complexity on even the hardest benchmark k-SAT ($k \geq 3$) formulas, but at an energy cost through exponentially driven auxiliary variables. This paper presents a novel analog hardware SAT solver, `AC-SAT`, implementing the CTDS via incorporating novel, analog circuit design ideas. `AC-SAT` is intended to be used as a coprocessor and is programmable for handling different problem specifications. It is especially effective for solving hard k-SAT problem instances that are challenging for algorithms running on digital machines. Furthermore, with its modular design, `AC-SAT` can readily be extended to solve larger size problems, while the size of the circuit grows linearly with the product of the number of variables and the number of clauses. The circuit is designed and simulated based on a 32-nm CMOS technology. Simulation Program with Integrated Circuit Emphasis (SPICE) simulation results show speedup factors of $\sim 10^4$ on even the hardest 3-SAT problems, when compared with a state-of-the-art SAT solver on digital computers. As an example, for hard problems with $N = 50$ variables and $M = 212$ clauses, solutions are found within from a few nanoseconds to a few hundred nanoseconds.**

***Index Terms*— Boolean satisfiability solver, hardware for constrained optimization, mix analog digital integrated circuits.**

I. Introduction

With Moore's law coming to end [2], exploring novel computational paradigms (e.g., quantum computing and neuromorphic computing) is more imperative than ever. While quantum computing is a promising venue, it is far from being brought to practical reality, with many challenges still to be faced, both in physics and engineering. Neuromorphic computing systems, e.g., cellular neural networks (CNNs) [3]–[5] and IBM's TrueNorth [6], have been shown to be promising alternatives for solving a range of problems in, i.e., sensory processing (vision and pattern recognition) and robotics. Analog mixed-signal information processing systems such as CNNs can offer extremely power/energy efficient solutions to some problems that are costly to solve by digital computers [7]. Such systems have received increasing attention in recent years (see [8]–[10]), including parallel analog implementations (see [11]).

In analog computing [12], the algorithm (representing the "software") is a dynamical system often expressed in the form of differential equations running in continuous time over real numbers, and its physical implementation (the "hardware") is any physical system, such as an analog circuit, whose behavior is described by the corresponding dynamical system. The equations of the dynamical system are designed such that the solutions to problems appear as attractors for the dynamics and that the output of the computation is the set of convergent states to those attractors [13]. Although it has been shown that systems of ordinary differential equations can simulate any turing machine [3], [14], [15], and hence, they are computationally universal, and they have not yet gained widespread popularity due to the fact that designing such systems is problem specific and usually difficult. However, if an efficient analog engine can be designed to solve nonpolynomial (NP)-complete problems, then according to the Cook–Levin theorem [16], it would help solve efficiently *all* problems in the NP class, as well as benefit a very large number of applications, both in science and engineering.

In this paper, we consider designing analog circuits for solving a representative NP-complete problem, the Boolean Satisfiability (SAT) problem. SAT is quintessential to many electronic design automation problems, and is also at the heart of many decision, scheduling, error-correction, and security applications. Here, we focus on k-SAT, for which is well known to be NP-complete for $k \geq 3$ [16]. The currently best known deterministic sequential discrete algorithm that exploits some properties of the search space has a worst case complexity of $\mathcal{O}(1.473^N)$ [17]. Other algorithms are based on heuristics, and while they may perform well on some SAT formula classes, there are always other formulas on which they take exponentially long times or get stuck indefinitely. Some of the better known SAT solvers include Zchaff [18], MiniSat [19], RSat [20], WalkSAT [21], focused record-to-record travel [22], and Focused Metropolis Search [23].

Manuscript received January 11, 2017; revised July 28, 2017; accepted August 28, 2017. Date of publication October 6, 2017; date of current version December 27, 2017. This work was supported in part by the National Science Foundation under Grants CCF-1644368 and 1640081, and by the Nanoelectronics Research Corporation, a wholly-owned subsidiary of the Semiconductor Research Corporation, through Extremely Energy Efficient Collective Electronics, an SRC-NRI Nanoelectronics Research Initiative under Research Task ID 2698.004. The work of M. Ercsey-Ravasz was supported in part by the European Commission Horizon 2020 Program under Grant 668863-SyBil-AA and in part by the Romanian CNCS-UEFISCDI under Research Grant PN-III-P2-2.1-BG-2016-0252. *(Corresponding author: Xunzhao Yin.)*

X. Yin and X. S. Hu are with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN 46556 USA (e-mail: xyin1@nd.edu; shu@nd.edu).

B. Sedighi is with Qualcomm Inc., San Diego, CA 92121 USA (e-mail: behnam.sedighi@gmail.com).

M. Varga is with the Biochemistry and Molecular Biology Program, The College of Wooster, Wooster, OH 44691 USA (e-mail: mvarga@wooster.edu).

Z. Toroczkai is with the Department of Physics and the Interdisciplinary Center for Network Science and Applications, University of Notre Dame, Notre Dame, IN 46556 USA (e-mail: mvarga@nd.edu; toro@nd.edu).

M. Ercsey-Ravasz is with the Faculty of Physics, Hungarian Physics Institute, Babes-Bolyai University, 400084 Cluj-Napoca, Romania (e-mail: ercsey.ravasz@phys.ubbcluj.ro).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2017.2754192

They typically consist of decision, deduction, conflict analysis, and other functions [24] that employ the capability of digital computers to assign values to literals, conduct Boolean constraint propagation (BCP), and backtrack conflicts [25], [26].

A number of hardware-based SAT solvers have been proposed in the past. FPGAs-based solutions have been investigated to accelerate the BCP part found in all "chaff-like" modern SAT solvers [27]–[29]. Speedups of anywhere between 3X and 38X have been reported when comparing these FPGA-based solvers over MiniSat [19], a well-known, high-performance software solver. A custom digital integrated-circuit (IC)-based SAT solver, which implements a variant of general responsibility assignment software patterns and accelerates traversal of the implication graph and conflict clause generation, has been introduced in [30] and [31]. A speed up of $\sim 10^3$ over MiniSat was reported based on simulation with extrapolation. Performance of these hardware-based approaches still has a lot of room for improvement since the algorithms that these hardware accelerators are based on are designed for digital computers and, thus, can typically expect to achieve limited speedup.

Recently, an analog SAT solver circuit was introduced in [32] using the theoretical proposal from [33] based on the CNN architecture. However, the theory in [33] has exponential analog-time complexity and, thus, is much less efficient than the SAT solver from [1], which forms the basis for this paper. Furthermore, the circuit from [32] seems to have been implemented only for a 4×4 problem size, and no hardware simulation and comparison results were reported.

Mostafa *et al.* [11] propose a distributed mixed (analog and digital) algorithm that is implementable on VLSI devices. It is based on a heuristic method combined with stochastic search, drawing on the natural incommensurability of analog oscillators. Assuming $\mathrm{P} \neq \mathrm{NP}$, in order to have efficient, polynomially scaling solution times, one would require exponentially many computing elements, i.e., exponentially scaling hardware resources. However, the method in [1] trades time-cost for *energy-cost*, which in practical terms is preferable to massive amounts of hardware resources. It is quite possible that from an engineering point of view the ideal approach combines both types of tradeoffs: time versus energy and time versus hardware (distributed). The heuristic stochastic search in [11] is effectively a simulated annealing method that implies high exponential runtimes for worst case formulas. In contrast, the analog approach in [1] is fully deterministic and extracts maximum information about the solution, embedded implicitly within the system of clauses, and can solve efficiently the hardest benchmark SAT problems—at an energetic cost [1].

Here, we propose a novel analog *hardware* SAT solver, referred to as AC-SAT.[1] AC-SAT is based on the deterministic continuous-time dynamical system (CTDS) in the form of coupled ordinary differential equations presented in [1]. As mentioned above, this system finds SAT solutions in analog polynomial time, however, at the expense of auxiliary variables that can grow exponentially, when needed (see [1], [34] for details). Though this CTDS is an incomplete solver, it does minimize the number of unsatisfied clauses when there are no solutions, and thus it is also a MaxSAT solver. The overall design of AC-SAT is *programmable and modular*, and thus, it can readily solve any SAT problem of size equal or less than what is imposed by the hardware limitations, and can also be easily extended to solve larger problems. Moreover, to avoid resource-costly implementations of the complex differential equations in CTDS, we introduce a number of novel analog circuit implementation ideas which lead to much smaller amount of hardware than straightforward implementations, while preserving the critical deterministic behavioral properties of CTDS equations.

We have validated our design through Simulation Program with Integrated Circuit Emphasis (SPICE) simulations. Our simulations show that AC-SAT can significantly outperform (over tens of thousands times faster than) MiniSat, with the latter running on the latest, high-performance digital processors. For hard SAT problems with 50 variables and over 200 clauses, compared with the projected performance of a possible custom hardware implementation based on a recent FPGA solver [29], AC-SAT offers more than $\sim$600X speedup. Monte Carlo simulations further demonstrate that AC-SAT is robust against device variations.

In the rest of this paper, we first review the basic CTDS theory and some of its variants in Section II. Section III introduces the overall AC-SAT design. In Section III-D, we present two alternative designs for a specific component in AC-SAT. Section IV first discusses simulation-based validation of AC-SAT, compares the different component designs, and then summarizes performance results for AC-SAT with respect to a software implementation of the CTDS SAT solver and MiniSat. Finally, we conclude this paper in Section V.

II. Background

Solving a k-SAT problem is to find an assignment to N Boolean variables $x_i \in \{0, 1\}$, $i = 1, \ldots, N$, such that they satisfy a given propositional formula F. F in conjunctive normal form (CNF) is expressed as the conjunction of M clauses C_m, $m = 1, \ldots, M$, i.e., $F = \bigwedge_{m=1}^{M} C_m$, where each clause is formed by the disjunction of k literals (which are variables or their complements). An example of a clause for 3-SAT would be $C_5 = (x_3 \vee \overline{x}_{19} \vee x_{53})$. Following [1], an analog variable s_i, which can take any real value in the range $s_i \in [-1, 1]$, is associated with the Boolean variable x_i such that $s_i = -1$ corresponds to x_i being FALSE ($x_i = 0$) and $s_i = 1$ to x_i being TRUE ($x_i = 1$). The formula $F = \bigwedge_{m=1}^{M} C_m$ can be encoded via the $M \times N$ matrix $\mathbf{C} = \{c_{m,i}\}$ with $c_{m,i} = 1$ when x_i appears in clause C_m, $c_{m,i} = -1$ when its complement $\overline{x}_i$ (negation of x_i) appears in C_m and $c_{m,i} = 0$ when neither appears in C_m. To every clause C_m, we associate an analog function $K_m(\mathbf{s}) \in [0, 1]$ given by

$$K_m(\mathbf{s}) = 2^{-k} \prod_{i=1}^{N} (1 - c_{m,i} s_i). \tag{1}$$

[1]We refer to AC-SAT as an Analog Circuit SAT solver since its main processing engine is analog. However, the entire system is a mixed-signal one as a digital verification component is also included in the hardware system.

It is easy to see that clause C_m is satisfied, iff $K_m = 0$. Defining a "potential energy" function

$$V(\mathbf{s}, \mathbf{a}) = \sum_{m=1}^{M} a_m K_m^2 \tag{2}$$

where $a_m > 0$ are auxiliary variables, one can see that all the clauses are satisfied iff $V = 0$. Thus the SAT problem can be reformulated as search in $\mathbf{s}$ for the global minima of V (since the condition $V \geq 0$ always applies). If the auxiliary variables a_m are kept as constants, then for most hard problems any hill-descending deterministic algorithm [which evolves the variables $s_i(t)$] would eventually become stuck in local minima of V and not find solutions. To avoid this, the auxiliary variables are endowed with a time-dependence coupled to the analog clause functions K_m. Ercsey-Ravasz and Toroczkai [1] proposed

$$\dot{s}_i = \frac{ds_i}{dt} = -\frac{\partial}{\partial s_i} V(\mathbf{s}, \mathbf{a}), \quad i = 1, \ldots, N \tag{3}$$

$$\dot{a}_m = \frac{da_m}{dt} = a_m K_m \quad m = 1, \ldots, M \tag{4}$$

in which (3) describes a gradient descent on V, and (4) is an exponential growth driven by the level of non-SAT in K_m (which also guarantees that $a_m(t) > 0$, at all times). (3) can be rewritten as

$$\frac{ds_i}{dt} = \sum_{m=1}^{M} a_m D_{m,i} \tag{5}$$

where

$$D_{m,i} = -\frac{\partial}{\partial s_i} K_m^2 = 2K_m c_{m,i} \prod_{\substack{j=1 \\ j \neq i}}^{N} (1 - c_{m,j} s_j). \tag{6}$$

For the auxiliary variables a_m, the formal solution to (4) is

$$a_m(t) = a_m(0) e^{\int_0^t d\tau K_m(\mathbf{s}(\tau))} \tag{7}$$

and thus the expression (2) of V is dominated by those K_m terms that have been unsatisfied for the longest time during the dynamics, resulting in an analog version of a focused search-type [23] dynamics. Also note that systems (3), (4) are not unique; however, it is simple from a theoretical point of view, and incorporates the necessary ingredients for solving arbitrary SAT problems, due to the exponentially accelerated auxiliary variables. For details on the performance of the algorithm, the reader is referred to [1].

It is important to observe that while the scaling of the analog time t to find solutions is polynomial, in hardware implementations, the a_m variables represent voltages or currents and thus the energetic resources needed to find solutions may become exponential for hard formulas which is, of course necessary, assuming P≠NP. However, the a_m variables do not need to grow exponentially all the time and unlimitedly, as in (4) and for that reason form (4) is not ideal for physical implementations. The challenge is then finding other variants that still significantly outperform digital algorithms, yet they are feasible in terms of physical implementations and costs. Note that such systems as ours essentially convert time costs into energy costs.

Here we introduce another form with the help of time-delays, which, however, still keeps the focused nature of the search dynamics but allows the a_m's to *decrease* as well when the corresponding clauses are (nearly) satisfied

$$\frac{da_m}{dt} = a_m \{K_m(\mathbf{s}(t)) - [1 - \dot{\delta}_m(t)] K_m(\mathbf{s}(t - \delta_m(t)))\}, \quad \forall m \tag{8}$$

with

$$a_m(0) > 0, \text{ and } \delta_m(0) = 0, \forall m \tag{9}$$

where the delay functions $\delta_m(t) \in [0, t]$ determine the history window of $K_m(\mathbf{s}(t))$ trajectory that has impact on the variation of a_m. The formal solution to (8) is

$$a_m(t) = a_m(0) e^{\int_{t-\delta_m(t)}^{t} d\tau K_m(s(\tau))}. \tag{10}$$

Clearly, the case $\delta_m(t) = t$ corresponds to (4), while $\delta_m(t) = 0$ recovers the case of constant a_m values that correspond to the naive energy minimization case. One approach to choosing $\delta_m(t)$ is setting it to a small value initially and doubling it every time the dynamics is stuck or hits an upper threshold (set, e.g., by a maximum allowed voltage value). This typically only requires a few iterations. Other delay functions are being investigated. It is important to note that the decrease of satisfied clause's associated a_m due to this time-delayed form relatively reduces the clause's weight in (5), thus increases other clauses' weights in the focused search space, enhancing the driving capability of unsatisfied clauses.

III. System Design

In this section, we present AC-SAT, our proposed analog SAT solver circuit based on the CTDS theory in Section II. Though it is possible to implement the CTDS equations digitally, the hardware would be much more costly in terms of area, power, and performance. Thus, we opt for an analog implementation that also bears affinity with the operations in the CTDS. Our circuit design aims to keep the hardware solver configurable and modular while keeping the circuit simple and power efficient. These considerations require careful design of the overall architecture and some modifications to the algorithm itself, which will be elaborated later in this paper.

Fig. 1 shows the high-level block diagram of AC-SAT. It consists of three main components: signal dynamics circuit (SDC) which implements the dynamics of variable signals s_is in (5), auxiliary variable circuit (AVC) which implements the dynamics of auxiliary variables a_ms in (4), and digital verification circuit (DVC) which checks whether all the clauses have been satisfied and outputs the satisfied assignments of variables. The AVC contains $\mathscr{M}$ identical elements, each of which receives the relevant s_is signals from the SDC as inputs, and generates a_m ($m \in [1, M]$) (where $M \leq \mathscr{M}$) variables as outputs. The SDC, containing $\mathscr{N}$ identical elements, in turn receives a_ms as feedback from the AVC and evolves the s_i ($i \in [1, N]$) signals (with $N \leq \mathscr{N}$), accordingly. The SDC outputs the analog values of s_is to the AVC and the

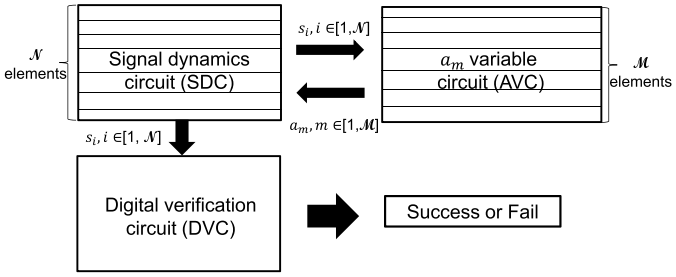

Fig. 1. High-level block diagram of AC-SAT. The SDC contains $\mathcal{N}$ elements while the AVC contains $\mathcal{M}$ elements. It can solve k-SAT problem instances with up to $\mathcal{N}$ variables and $\mathcal{M}$ clauses.

digital version of s_is to the DVC. Based on the digital values of s_i, the DVC determines whether a solution to the given SAT problem has been found at that time.

The given block diagram can solve any k-SAT problem with N or up to $\mathcal{N}$ Boolean variables and M or up to $\mathcal{M}$ clauses. However, a naive and direct implementation of the dynamical equations (4), (5), and (6) would incur large hardware costs. Instead, here we present implementations of the SDC and AVC circuits that are much more resource-efficient than the direct approach. Below, we elaborate the design of the three circuit components using the 3-SAT problem (i.e., three nonzero $c_{m,j}$s for each clause) as an example. AC-SAT for any k-SAT problem can be designed following the same principle.

A. Signal Dynamics Circuit

The SDC contains an array of analog elements that realize the dynamics specified by (5) and (6). Though it is possible to implement the multiplications and voltage controlled current source (VCCS) in (5) and (6) straightforwardly based on operational amplifiers, such implementations can be rather costly. We introduce several novel circuit design ideas to implement the dynamics in (5) and (6). We will show that the accuracy of the circuit is sufficient for the type of dynamical systems being considered here.

Given a 3-SAT problem with N variables, the SDC enables an array of N analog elements, referred to as s_i element, for evaluating the s_i $(i = 1, \ldots, N)$ signals. Fig. 2(a) shows the conceptual design of the s_i element that realizes (5). The s_i element contains a capacitor C connected to the M branch blocks (where M is the total number of clauses in the 3-SAT problem), an analog inverter, an inverted Schmitt trigger, and a digital inverter. The voltage across capacitor C, i.e., V_i, and the output of the analog inverter $\overline{V_i}$ represent the analog value of signal s_i and $-s_i$, respectively. Signal $s_i \in [-1, 1]$ (respectively, $-s_i \in [1, -1]$) is mapped to $V_i \in [GND, V_{DD}]$ (respectively, $\overline{V_i} \in [V_{DD}, GND]$). The inverted Schmitt trigger and the digital inverter output the digital versions of $-s_i$ and s_i, denoted by $\overline{Q_{s_i}}$ and Q_{s_i} (i.e., taking on values of either GND or V_{DD}).

To see why the s_i element in Fig. 2(a) can be used to evaluate (5), let us denote the current from each of the branch block as $I_{m,i}$. Then, we have

$$C\frac{dV_i}{dt} = \sum_{m=1}^{M} I_{m,i}. \tag{11}$$

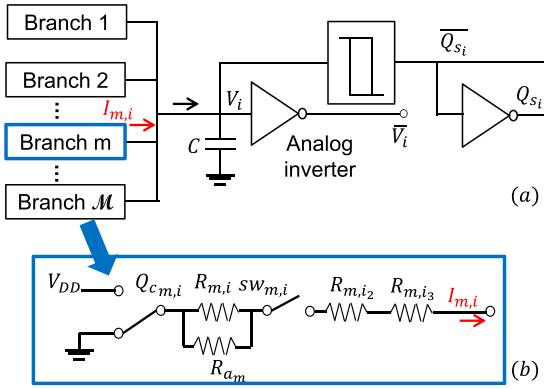

Fig. 2. (a) Design of one array element in the SDC. (b) Detailed conceptual design of the branch block.

Comparing (5) with (11), we see that the s_i element in Fig. 2(a) precisely realizes (5) if we have

$$I_{m,i} = Ca_m D_{m,i}. \tag{12}$$

In order to design a branch block to satisfy (12), we first make some observations related to the $D_{m,i}$ quantities. In a 3-SAT problem, there are only three nonzero $c_{m,j}$s in each C_m clause. Let us denote them as $c_{m,i}$, c_{m,i_2}, and c_{m,i_3}. Then $D_{m,i}$ in (5) can be shown to have the following form:

$$\begin{aligned} &D_{m,i} \\ &= 2^{-2} \times c_{m,i}(1 - c_{m,i}s_i)(1 - c_{m,i_2}s_{i_2})^2(1 - c_{m,i_3}s_{i_3})^2 \\ &= \begin{cases} 2^{-2}(+1 - s_i)(1 - c_{m,i_2}s_{i_2})^2(1 - c_{m,i_3}s_{i_3})^2 & \text{if } c_{m,i} = 1 \\ 0 & \text{if } c_{m,i} = 0 \\ 2^{-2}(-1 - s_i)(1 - c_{m,i_2}s_{i_2})^2(1 - c_{m,i_3}s_{i_3})^2 & \text{if } c_{m,i} = -1. \end{cases} \end{aligned} \tag{13}$$

Referring to (13), one can readily see that $D_{m,i}$ has the following properties.

1) If any of s_i, s_{i_2}, and s_{i_3} is satisfied, i.e., reaches 1 or -1 (indicating x_i is either TRUE or FALSE), $D_{m,i}$ becomes zero. According to (5), a zero $D_{m,i}$ means that clause C_m has no impact on the variation of s_i. On the other hand, when none of the three variables is satisfied, but one of them gets closer to being satisfied, the magnitude of $D_{m,i}$ reduces, again.
2) The sign of $D_{m,i}$ is the same as that of $c_{m,i}$ since $(1-s_i)$ cannot be negative. If $c_{m,i} = 1$ (respectively, -1), the contribution of $D_{m,i}$ to ds_i/dt is positive (respectively, negative), i.e., it tries to push s_i toward $+1$ (respectively, -1).

Based on the above observations, let us examine the conceptual design of the branch block in Fig. 2(b). Specifically, the branch block contains two switches and four tunable resistive elements. (The resistive elements here are used to simplify the drawing, and the details about their design will be described later.) Here, $R_{m,i}$ represents the resistive element associated with s_i, while R_{m,i_2} and R_{m,i_3} represent the resistive elements associated with the other two signals s_{i_2} and s_{i_3} in (13). R_{a_m} represents the resistive element associated with auxiliary variable a_m. One switch is controlled by signal $sw_{m,i}$, which is left open if $c_{m,i} = 0$ (indicating that

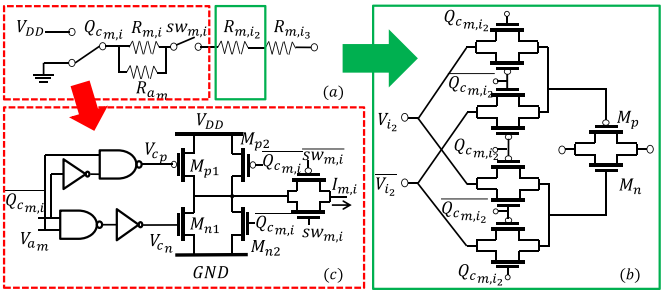

Fig. 3. Detailed design of the branch block in Fig. 2(b). (a) Conceptual design of the branch block. (b) Circuit implementation for R_{m,i_2} and R_{m,i_3} [elements in the green box in Fig. 3(a)]. (c) Circuit implementation for the switch as well as R_{a_m} and $R_{m,i}$ [elements in the red box in Fig. 3(a)].

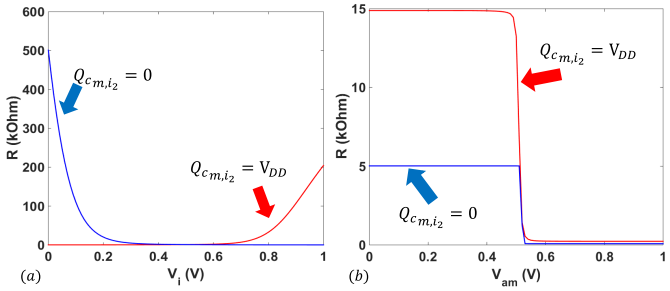

Fig. 4. SPICE simulation results depicting (a) value of resistor R_{m,i_2} or R_{m,i_3} in Fig. 3(a) as a function of V_i and (b) value of resistor $R_{m,i}||R_{a_m}$ in Fig. 3(a) as a function of V_{a_m}.

x_i does not appear in clause C_m), and is closed otherwise. The other switch is controlled by $Q_{c_{m,i}}$, the digital version of $c_{m,i}$. If $c_{m,i} = 1$ (respectively, -1), indicating that x_i (respectively, $\overline{x_i}$) is present in the clause, the switch controlled by $Q_{c_{m,i}}$ connects to V_{DD} (respectively, *GND*). It can be readily seen that

$$I_{m,i} = \begin{cases} (V_{DD} - V_i)/(R_{m,i}||R_{a_m} + R_{m,i_2} + R_{m,i_3}) & \text{if } c_{m,i} = 1 \\ 0 & \text{if } c_{m,i} = 0 \\ (GND - V_i)/(R_{m,i}||R_{a_m} + R_{m,i_2} + R_{m,i_3}) & \text{if } c_{m,i} = -1. \end{cases} \quad (14)$$

If the values of R_{a_m}, $R_{m,i}$, R_{m,i_2}, and R_{m,i_3} are chosen properly, the $I_{m,i}$ value derived from the branch block would have the same properties as identified for $D_{m,i}$ above.

The actual realization of the four resistive elements in Fig. 2(b) is given in Fig. 3. The implementation of R_{m,i_2} and that of R_{m,i_3} are the same and the one for R_{m,i_2} is shown in Fig. 3(b). Consider the R_{m,i_2} block. The two terminals of the transmission gate formed by transistor M_p and M_n correspond to the terminals of R_{m,i_2}. The gate terminals of M_p and M_n are connected to V_{i_2} and $\overline{V_{i_2}}$ via four additional transmission gates controlled by $Q_{c_{m,i_2}}$ and $\overline{Q_{c_{m,i_2}}}$. It can be derived that this realization of R_{m,i_2} exhibits the desired properties outlined above for $D_{m,i}$ in (13). The SPICE simulation results depicting the relationship between the resistance value and V_i are given in Fig. 4(a). For example, assuming that c_{m,i_2} is 1, i.e., $Q_{c_{m,i_2}} = V_{DD}$ [corresponding to the red line in Fig. 4(a)], the gates of M_n and M_p are connected to $\overline{V_{i_2}}$ and V_{i_2}, respectively. If variable s_{i_2} is satisfied, i.e., V_{i_2} is close to V_{DD} [where $V_{DD} = 1$ *V* in Fig. 4(a)] and $\overline{V_{i_2}}$ is close to *GND*, then both M_n and M_p are OFF, R_{m,i_2} has a very large value (around 200 kΩ) and $I_{m,i}$ is close to zero. This means that clause C_m has no impact on the variation of s_i which is exactly the desired behavior. On the other hand, if s_{i_2} is not satisfied, as it gets closer to its target (i.e., $+1$), the magnitude of $I_{m,i}$ reduces because R_{m,i_2} increases as can be seen by the increase in the resistance value as V_i gets close to V_{DD}. The blue line in Fig. 4(a) corresponds to the case where $c_{m,i_2} = -1$ and its behavior can be explained in the same way as above.

The circuit block for implementing the switch controlled by $Q_{c_{m,i}}$ and the two resistive elements $R_{m,i}$ and R_{a_m} is shown in Fig. 3(c). The gates of transistor M_{p1} and M_{p2} (respectively, M_{n1} and M_{n2}) control the connection to V_{DD} (respectively, *GND*). One of the gates connects directly to $\overline{Q_{c_{m,i}}}$ (representing the negated $c_{m,i}$ signal), while the other is controlled by

$$V_{c_p} = \overline{V_{a_m} Q_{c_{m,i}}} \quad \text{respectively,} \quad V_{c_n} = V_{a_m} \overline{Q_{c_{m,i}}}. \quad (15)$$

Note that though V_{a_m} in (15) (as input to the two NAND gates) seems to be treated as a digital signal, it actually remains as an analog signal while the NAND gates and inverters operate in the linear $V_{in} - V_{out}$ region to produce analog outputs as desired. The SPICE simulation results in Fig. 4(b) indicate that the block realizes the switch function due to $c_{m,i}$ as well as $(+1 - s_i)$ and $(-1 - s_i)$ in (13), and incorporates the a_m term in (5). Consider the case that $c_{m,i} = -1$, i.e., $Q_{c_{m,i}} = 0$. Then $V_{c_p} = V_{DD}$ and $V_{c_n} = V_{a_m}$, which means that the block in Fig. 3(c) is connected to *GND* and the current flowing through the block is dependent on the voltage representing a_m, V_{a_m}. The blue line in Fig. 4(b) shows the equivalent resistance value of the block versus V_{a_m}. As V_{a_m} gets larger, M_{n1} exhibits smaller resistance, resulting in larger impact of a_m on the current flowing through the block. Note that initially V_{a_m} is very small, and thus the right two transistors M_{p2} and M_{n2} that are of smaller sizes than M_{p1} and M_{n1} are employed to ensure proper current flow as well as serve as a current boost. The red line in Fig. 4(b) corresponds to the case where $c_{m,i} = 1$, i.e., $Q_{c_{m,i}} = V_{DD}$.

The SDC also converts the analog signals V_is to digital signals Q_{s_i}, via an inverted Schmitt trigger. The inverted Schmitt trigger circuit is shown in Fig. 5(a). The digital signals are then sent to the DVC to check if a solution has been found. The inverted Schmitt trigger circuit exhibits hysteresis in its transfer curve as seen from the simulation result in Fig. 5(b) and, hence, can perform analog-digital conversion with minimal noise impact. Putting all the above discussions together, one can conclude that the SDC correctly implements the system dynamics defined by (3).

B. Auxiliary Variable Circuits

As pointed out in Section II, the auxiliary variables, a_ms as defined in (4), are used to help avoid the gradient descent search being stuck in nonsolution attractors. The a_m signal follows an exponential growth driven by the level of non-SAT in clause C_m. A direct way to implement an exponential

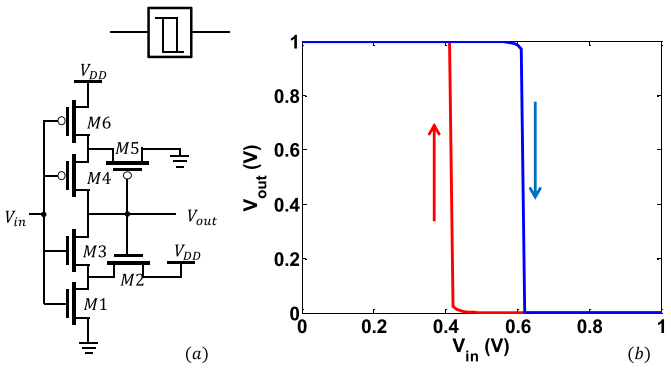

Fig. 5. Schmitt-trigger inverter and its transfer characteristic.

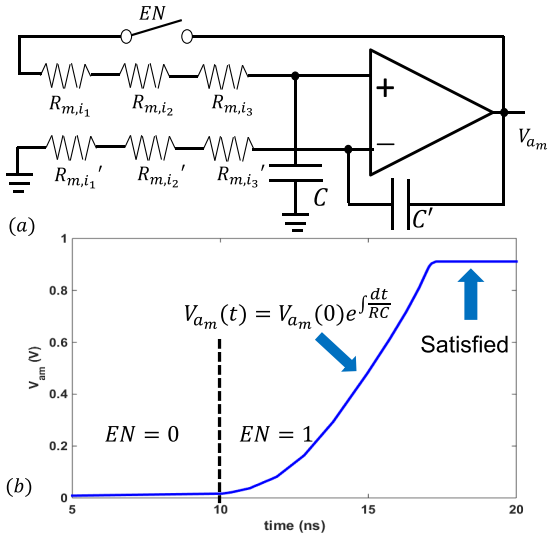

Fig. 6. (a) Conceptual design of one element in the AVC. Actual realization of the resistive element is similar to the circuit in Fig. 3(b). (b) SPICE simulation result depicting the waveform of V_{a_m} versus time. $V_{a_m}(0)$ represents the initial value of V_{a_m}.

function is through an operational amplifier (op-amp), which we present below. Note that we have realized the analog version of equation (4) in a resource-efficient manner, similar to the implementation in DVC, to avoid costly multiplications and VCCS implementations.

The AVC contains an array of $\mathscr{M}$ a_m elements where $\mathscr{M}$ is the maximum number of clauses in a given problem that the AVC can handle. Fig. 6 illustrates the conceptual design of the a_m element, similar to a noninverting integrator. Here, the value of a_m (for clause C_m) is represented by the voltage at the output of the op-amp, i.e., V_{a_m}. Resistive elements R_{m,i_1}, R_{m,i_2}, and R_{m,i_3} are associated with the three signals in a_m's clause while R'_{m,i_1}, R'_{m,i_2}, and R'_{m,i_3} are identical to R_{m,i_1}, R_{m,i_2}, and R_{m,i_3}, respectively. The two capacitors, C and C', have identical values as well. Together with the resistive elements, they control the speed of V_{a_m} growth. The switch controlled by EN is realized by a transmission gate to control the start of the a_m element. The first order differential equation of V_{a_m} can be written as

$$C\frac{dV_{a_m}}{dt} = V_{a_m}/(R_{m,i_1} + R_{m,i_2} + R_{m,i_3}). \tag{16}$$

The Rs in Fig. 6(a) are tunable resistive elements implemented by transmission gates which have similar circuit topology to that shown in the green box in Fig. 3. For example, for R_{m,i_1} and R'_{m,i_1}, the transmission gates (M_p and M_n) are controlled, via four other transmission gates, by analog signals V_{i_1} and V'_{i_1}, representing x_is presence in clause C_m. The other two pairs of Rs are designed in the same way. If any of the three variables in clause C_m is satisfied, the corresponding V_i turns off the respective transmission gate and cut off the current paths from op-amp's inverting input to ground and from noninverting input to V_{a_m}.

The circuit in Fig. 6(a) exactly realizes the exponential growth specified in (4) up to an upper bound on V_{a_m}, i.e., the op-amp's supply voltage. Fig. 6(b) plots an example V_{a_m} value growth with time before and after associated signals s_is get satisfied. After EN is set to 1 (i.e., the switch is closed), V_{a_m} starts to grow exponentially, following the differential equation in (16). According to Fig. 4(b), as V_{a_m} increases, the resistant value of $R_{m,i}||R_{a_m}$ drops down, leading to a larger current $I_{m,i}$ in the corresponding branch block in Fig. 2(b), which is consistent with (12). This current, together with other currents that are associated with V_i in Fig. 2(a), contributes to the variation of V_i which is specified in (11). There are two cases that may stop the evolution of V_{a_m}, which are as follows.

1) As stated above, if any one of the three analog signals in clause C_m is satisfied, the current paths in Fig. 6(a) is cut off, and V_{a_m} stops at a certain voltage. This indicates that V_{a_m} has finished its utility as an auxiliary variable to drive the corresponding clause to the satisfied state.
2) If V_{a_m} reaches its upper bound before any of the three variables in the corresponding clause is satisfied, the circuit stops evolving since the V_{a_m} value is unable to drive this yet unsatisfied clause any more. This impacts the effectiveness of avoiding being stuck in a nonsolution attractor during the gradient descent search process.

The upper bound on V_{a_m} imposes a physical limitation on the hardware realization of the CTDS theory.[2]

Although the AVC design given in Fig. 6 realizes the exponential growth, it requires $\mathscr{M}$ op-amps, resulting in a large amount of area and power consumption. There exist other ways to achieve exponential signal growth, e.g., circuits with positive feedback often have exponential growth in certain ranges. Besides the exponential growth implementation, we will introduce alternative AVC designs in the next section.

C. Digital Verification and Interface Circuits

The goal of the DVC is to determine if a solution (the set of s_is) to the given problem has been found within a user specified time bound. The DVC is implemented readily through the use of an array of $3M$ XOR gates and an array

[2]As discussed in Section II, (3) and (4) are not unique, and the effect of the maximum voltage limitation depends on the equations themselves. For example, Section II introduces an alternative, delay-based formulation for a_m in (8), which allows a_m to decrease when the corresponding clause is satisfied. This delay-based formulation of a_m postpones reaching the a_m upper bound. The implementation of (8) for the op-amp-based approach is currently under development.

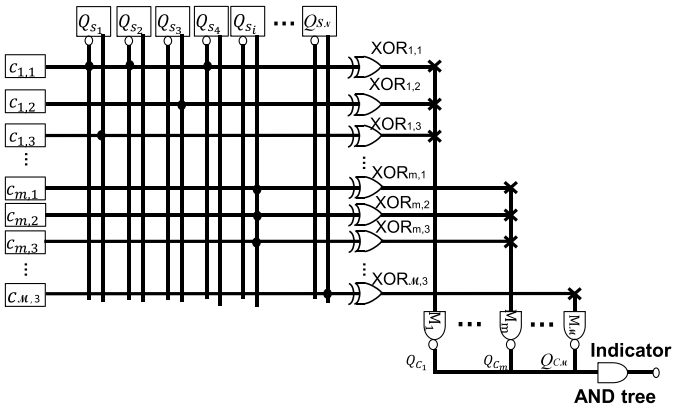

Fig. 7. Schematic of the DVC.

of M NAND gates as shown in Fig. 7. The input to the DVC is the digital representation of s_i's and $-s_i$'s, i.e., Q_{s_i} and $\overline{Q_{s_i}}$, from the SDC. Each NAND gate corresponds to a clause and its inputs correspond to the literals present in the clause. Note that in the DVC, we only include those $c_{m,i}$s whose values are $+1$ (represented by logic signal "1") and -1 (represented by logic "0"). The outputs of the DVC are analog values Q_{C_m}, for clauses C_m, and indicator, which is set to 1 if the circuit finds a solution, otherwise it remains at 0. The DVC is an asynchronous circuit, and the output of the DVC constantly records whether a solution is found or not. By setting a time bound T, the DVC regards the problems whose solutions are found within T as satisfiable problems, the rest are considered either unsatisfiable or unsatisfiable within the alloted time. Note that for problem instances where no solutions are found in the given time bound, our approach does not provide a formal proof of unsatisfiability (as our algorithm is an incomplete algorithm). However, our solver is a MaxSAT solver, because it does not use any assumptions about the solvability of the formula and minimizes the number of unsatisfied clauses within the allotted resources or time. Theoretical analysis of the performance of the solver as a MaxSAT solver is out of the scope of this paper and will be presented elsewhere.

It is easy to see that all three components, SDC, AVC, and DVC, are modular and programmable. By modular, we mean that the basic elements in each circuit can be repeated for different problem sizes (i.e., the number of variables N and the number of clauses M). By programmable, we mean that any k-SAT problem instance can be solved by the same SDC, AVC, and DVC implementation as long as the problem size is less than or equal to the hardware specification.

Below, we briefly describe the I/O interface between the CPU and AC-SAT. AC-SAT is used as a coprocessor, similarly to other reconfigurable coprocessors (such as dynamically reconfigurable FPGAs). To facilitate configuration, AC-SAT can be augmented with an on-chip reconfiguration memory as well as a simple controller. Based on the problem description (given in the CNF), CPU writes to memory the configuration information. The controller then uses the memory contents to set the respective switches in the SDC, AVC, and DVC components.

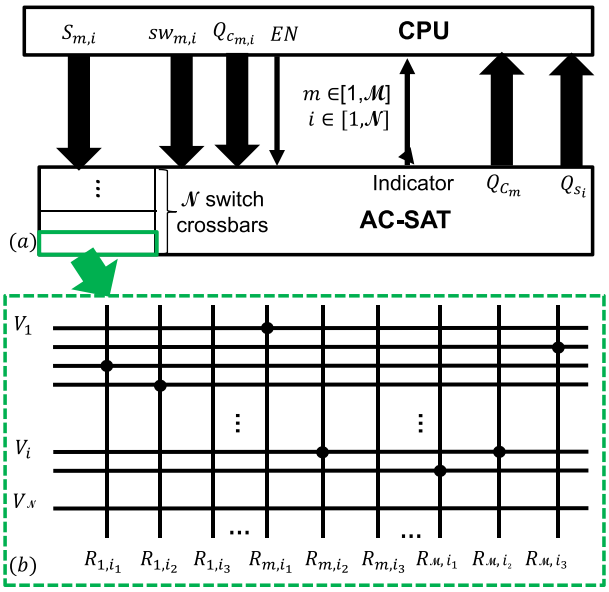

Fig. 8. (a) Inputs and outputs of AC-SAT. (b) Structure of a switch crossbar.

The main configuration information sent from CPU to AC-SAT includes the following: EN, $S_{m,i}$, $sw_{m,i}$, and $Q_{c_{m,i}}$. EN activates AC-SAT. $S_{m,i}$ describes the appearance of variable signals in the corresponding clauses, e.g., $S_{m,i}=1$ (or 0) means that variable s_i is (or is not) in the mth clause. $sw_{m,i}$ is used to deactivate unused branch blocks in the SDC, e.g., $sw_{m,i}=0$ would deactivate the mth branch block for variable s_i. $Q_{c_{m,i}}$ is the digital version of $c_{m,i}$ (see Section III-A for its definition). AC-SAT receives the inputs from CPU and delivers $sw_{m,i}$ and $Q_{c_{m,i}}$ to the memory associated with the SDC component. Since $S_{m,i}$ cannot be used directly by the resistive elements in the DVC and AVC that require the internal variable signals V_is [see Figs. 3 and 6(a)], we use $\mathcal{N}$ switch crossbars [Fig. 8(b)] to accomplish the mapping from the variable signals V_i to each resistive element in the SDC and AVC based on $S_{m,i}$ (i.e., $S_{m,i}$ is used to set the state of the corresponding switch). For output, AC-SAT indicates Q_{s_i} and Q_{C_m} and indicator from the DVC to CPU. Q_{s_i} indicates the values of variable signals, and Q_{C_m} indicates the states of all clauses (satisfied or not). All the inputs and outputs are digital signals. Once AC-SAT finds a solution, signal indicator acts as an interrupt to CPU and CPU reads the AC-SAT outputs, i.e., Q_{C_m} and Q_{s_i}. On the other hand, if the circuit runs out of time and no solution is found, AC-SAT outputs the results with minimum unsatisfiable clauses, but with indicator = 0.

D. Alternative AVC Designs

The op-amp-based AVC described in Section III-B realizes an exponentially growing a_m variable aiming to address hard SAT problems (some SAT instances with constraint density $\alpha=M/N \gtrsim 4.25$) within its physical limitation. However, for application type SAT problems, i.e., which are not specially designed to be very hard, exponential growth for a_m is not always necessary. Below we describe two alternative circuit designs to implement an a_m function that has a

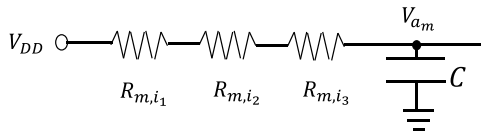

Fig. 9. Conceptual design of the AVC element realizing the $(1-\epsilon_2\, e^{-qt})$-type growth. Implementation of the resistive elements is similar to those in Fig. 3.

$(1-\epsilon_2 e^{-qt})$-type growth to a saturation value. In the remainder, we will refer to this version of a_m growth as the "simpler version."

Fig. 9 depicts the conceptual design of the a_m element realizing the simpler a_m growth, where capacitor C is charged to V_{DD} through three tunable resistors. The first order differential equation that governs V_{a_m} can be written as

$$C\frac{dV_{a_m}}{dt} = (V_{DD} - V_{a_m})/(R_{m,i_1} + R_{m,i_2} + R_{m,i_3}). \quad (17)$$

R_{m,i_1}, R_{m,i_2}, and R_{m,i_3} are the same as the resistors in Fig. 6, realized by transmission gates controlled by V_{i_1}, V_{i_2}, and V_{i_3}, similar to that in Fig. 3. If any of the three variables s_i in clause C_m is satisfied, the corresponding V_i turns off the respective transmission gate and cuts off the current path from V_{DD} to the capacitor. This circuit guarantees the continuous growth of a_m since V_{a_m} is charged by V_{DD} till it reaches its upper bound V_{DD} or any of the three variables in the clause is satisfied.

It is important to note that as the circuit in Fig. 9 does not realize the exponential growth specified in (4), it can indeed get captured into nonsolution attractors indefinitely for some very hard formulas. However, we have found that even for many hard problems, it works more efficiently than the op-amp-based a_m element (with the same threshold value) in finding solutions for smaller size problems (as long as they are solvable), and the dynamics would only rarely get stuck. We will discuss this aspect more in the evaluation section via simulation results.

Similar to the op-amp-based AVC, in the simpler AVC design in Fig. 9, some V_{a_m}s may reach V_{DD} before the CTDS converges to a solution. One way to alleviate this physical limitation is to increase the range of V_{a_m}. However, such an approach has its limitations in practical circuits (e.g., the limited voltage supply allowed). This, in fact, is a fundamental limitation due to the NP hardness of 3-SAT. Nonetheless, it is possible to improve the V_{a_m} driving capability in the CTDS and increase the size of the hard problems that can be solved with the same physical range of V_{a_m}. Below, we discuss an alternative implementation of the simpler a_m element to demonstrate that it is worthwhile to investigate different implementations of the AVC.

Recall that the delay function $\delta_m(t)$ in (8) is to assist a_m to keep relevant information from a limited range of the trajectory's past history instead of the entire history. We consider combining the simpler a_m element with this time-delayed form, and choose $\delta_m(t) = \delta$ (meaning that we are integrating over a fixed time window of length δ). The corresponding a_m element is shown in Fig. 10. Capacitor C is charged to V_{DD} through three tunable resistive elements and

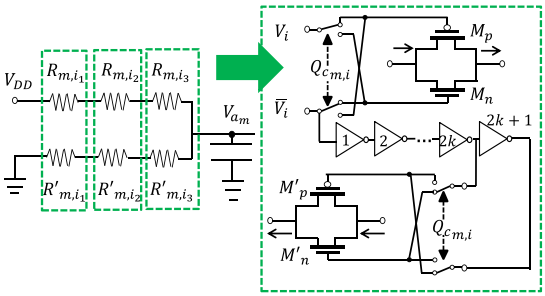

Fig. 10. Circuit realization of the time-delayed simpler a_m growth.

discharged to *GND* through the other three resistive elements. The first order differential equation of V_{a_m} can be written as

$$C\frac{dV_{a_m}}{dt} = \frac{V_{DD} - V_{a_m}}{R_{m,i_1} + R_{m,i_2} + R_{m,i_3}} + \frac{-V_{a_m}}{R'_{m,i_1} + R'_{m,i_2} + R'_{m,i_3}}. \quad (18)$$

The six resistive elements are implemented by transmission gates similar to those for R_{m,i_2} in Fig. 3. Specifically, $R'_{m,i_1}, R'_{m,i_2}, and\, R'_{m,i_3}$ are controlled by $\mathbf{s}$ values (representing all s_i values) earlier values, i.e., $\mathbf{s}(t-\delta_m)$. A chain of an odd number of analog inverters is used to realize the delay δ as shown at the right of Fig. 10. If signals $\mathbf{s}(t)$ reach their targets at time t, the path from V_{DD} to capacitor C is cut off, while the discharge path is still conducting current. V_{a_m} keeps decreasing until the discharge path is cut off after δ. Hence, the circuit properly implements the time-delayed simpler a_m growth function.

IV. Evaluation

In this section, we present our evaluation study of AC-SAT. We first describe the basic functional validation and then discuss the robustness of AC-SAT against device variations. We finally compare the performance of AC-SAT with a state-of-the-art digital solver.

A. Functional Validation

We have built our proposed analog SAT solver, AC-SAT, at the transistor level in HSPICE based on the predictive technology model 32-nm CMOS model [35]. All the circuit components use $V_{DD} = 1V$. To achieve sufficient driving capability, the minimum transistor size is set to $W = 1\mu m$ and $L = 40nm$ while actual transistor sizes are selected according to their specific roles. For logic gates, the transistor sizes are chosen to ensure equal pull-up and pull-down strength. For the branch block in Fig. 3, the relative W/L values of transistor M_{p1} and M_{n1} (i.e., the size of R_{a_m}) with respect to the W/L values of the other transistors (i.e., the sizes of $R_{m,i}$, R_{m,i_2}, and R_{m,i_3}) determine the contribution of a_m to $I_{m,i}$. Thus, tradeoff between the size of R_{a_m} and the impact of a_m should be carefully considered. In our implementation, since $R_{m,i}$ (dependent on the size of M_{p2} and M_{n2}) is mainly used for proper current flow at the beginning, it should not dominate the current flow as R_{a_m} starts affecting $I_{m,i}$. Hence, we chose the transistor sizes such that they result in the ratio of R_{a_m} to $R_{m,i}$, R_{m,i_2},

and R_{m,i_3} being 64, 4, and 4, respectively. The sizes of the transistors in other circuits are determined in a similar fashion. Note that the transistor sizes shown above are just a lower bound for the technology model that we are using. The absolute values of the transistor sizes are not critical (the equations of the solver are adimensional), and other transistor sizes should also work as long as their relative sizes are close to the ones that we have shown.

To demonstrate that AC-SAT indeed behaves as specified by the CTDS dynamics in (3) and (4), we examine the waveforms of signals s_i and a_m. Fig. 11 shows three sets of s_i and a_m waveforms from a 3-SAT problem instance having 50 variables and 212 clauses: Fig. 11(a) for the op-amp-based a_m implementation [realizing the $(\epsilon_1\ e^{qt})$-type a_m growth], Fig. 11(b) for the simpler a_m implementation (realizing the $(1-\epsilon_2\ e^{-qt})$-type a_m growth), and Fig. 11(c) for the time-delayed simpler a_m implementation. For all three designs, AC-SAT successfully finds a solution after a certain time as indicated by the vertical dashed lines. Note that AC-SAT determines whether a solution is found via the DVC. As can be seen from the s_i trajectories, the s_i signals stabilize (i.e., converge) after a solution is found. Comparing the a_m trajectories in the three different designs, one can see that the a_ms grow most rapidly in the op-amp-based design due to the exponential growth function while some of the a_ms (the ones corresponding to the satisfied clauses) in the time-delayed implementation decrease after they reach their peak magnitude, just as predicted by (8).

B. Scaling Considerations

Besides functionality, the impact of interconnect parasitics on the circuit is another important consideration toward practical and modular designs of the solver. As the circuit size increases [i.e., O($\mathscr{M}\ \mathscr{N}$)], for each variable array element in the SDC, the total parasitic capacitance from the branch blocks (Fig. 2) increases linearly with the number of branch blocks $\mathscr{M}$, namely the maximum number of clauses that the solver can handle. Given a problem instance, if variable x is involved in y clauses ($y < \mathscr{M}$), then y branch blocks associated with x are active, while all other branch blocks are turned off. However, all the branch blocks contribute parasitic capacitance to the dynamical evolution of the variable x. To investigate the impact of parasitic capacitance, we have conducted a number of simulations of the solver circuit with various number of branch blocks (i.e., $\mathscr{M}$) in the SDC, i.e., 100, 500, 1000, 5000, and 10000 branch blocks for each variable array element. We used the circuits to solve various problem instances with 10, 20, and 30 variables, and evaluated the time to find a solution. Simulation results shown in Fig. 12 demonstrate that as the solver circuit becomes larger, the solver still functions correctly, but takes longer time to find solutions due to larger parasitic capacitance.

Another issue due to interconnect scaling is the capacitance value associated with the AVC elements. As the parasitic capacitance associated with variable signals increases with the number of branch blocks, the dynamic evolution of the variable signals becomes slower due to the RC charging rule. As a consequence, the AVC element, whose internal capacitance (i.e., contributed by the two capacitors in the AVC element) is much smaller than the parasitic capacitance associated with the variable signals, will charge V_{a_m} quickly, and reach the V_{a_m} upper bound before the variable signals find the solution. This trend could make the AVC element less

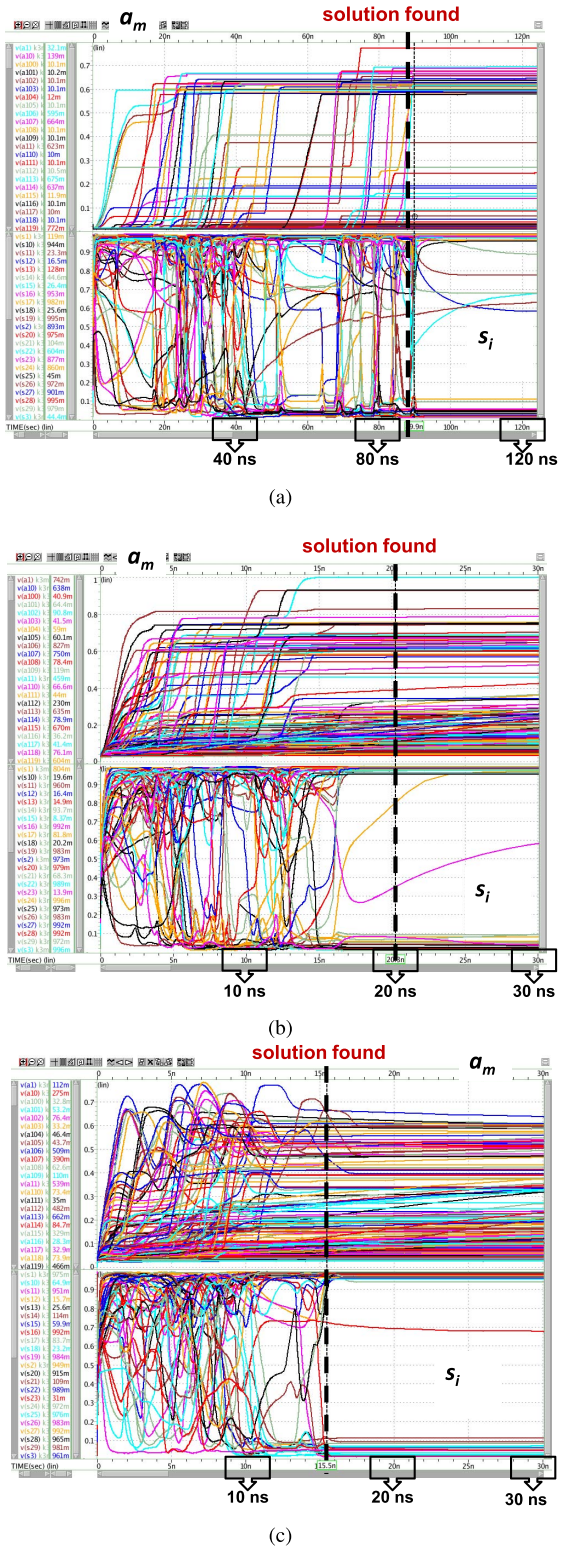

Fig. 11. Waveforms of signals representing $s_i(t)$ and $a_m(t)$ for a 3-SAT problem with 212 clauses and 50 signal variables. The problems/formulas have a constraint density α=M/N=4.25, and are considered to be hard problems. (a) With $\epsilon_1 e^{qt} a_m$ growth. (b) With $1-\epsilon_2 e^{-qt} a_m$ growth. (c) With time delayed a_m growth.

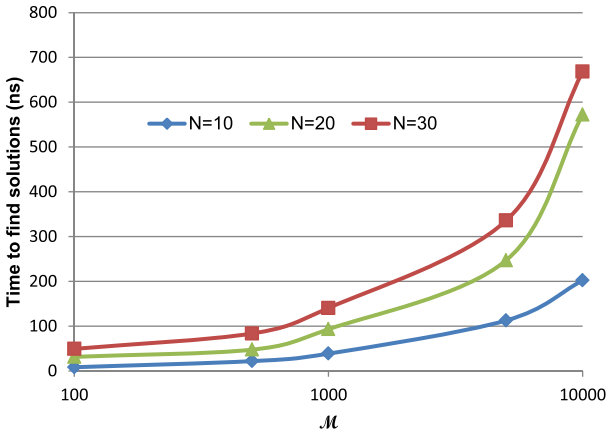

Fig. 12. SPICE simulation run times to find a solution with various AC-SAT circuit sizes. N is the number of variables in a problem instance. The number of clauses in each problem instance is $4.25 \times N$ (corresponding to hard problems).

effective and thus lead to the solver not able to find a solution. Therefore, it is critical to increase the values of the capacitors in the AVC elements as the circuit size increases. A basic approach is to choose the capacitance value such that the V_{a_m}s RC constant is comparable or smaller than the variable signal RC constant. As the parasitic capacitance of the SDC increases linearly with the number of branch blocks, the capacitance in the AVC element should also scale proportionally.

C. Device Variation Study

After validating that AC-SAT indeed can solve SAT problems correctly, we further investigate the robustness of AC-SAT against device variations. Typical analog circuits can be rather sensitive to device variations if not designed well. However, AC-SAT has two unique advantages in this aspect. First, the circuit itself does not rely on device matching. Second, the CTDS theory has been shown in theory to be robust against noise [36]. To demonstrate the robustness of our proposed AC-SAT system, we have conducted Monte Carlo simulations with respect to transistor size variations for randomly chosen 3-SAT problems. Specifically, we let the transistor widths follow a Gaussian distribution with standard deviation $(\Delta W/W)$ of $0.05\mu m/\sqrt{W \times L}$ for all transistor widths, which is an acceptable variance distribution for the 32-nm technology node [37]. In other words, the solver circuit is simulated with the Monte Carlo method considering 5% transistor width variations. For each problem, 100 Monte Carlo runs were performed. Fig. 13 shows the waveforms of one $a_m(t)$ signal and one $s_i(t)$ signal plus the output of DVC for one problem instance for 100 Monte Carlo simulations. As can be seen from the signal trajectories, the signals evolve consistently in the Monte Carlo simulations, and the results demonstrate the robustness of the circuit. Moreover, since analog circuits generally use mature technology nodes (e.g., 180 and 90 nm), we in fact validated our design in a relatively aggressive way. The circuit is expected to perform much better under mature technologies, whose variations would be much smaller than 5%.

To get a better comparison between the different designs, we performed Monte Carlo simulations on both the

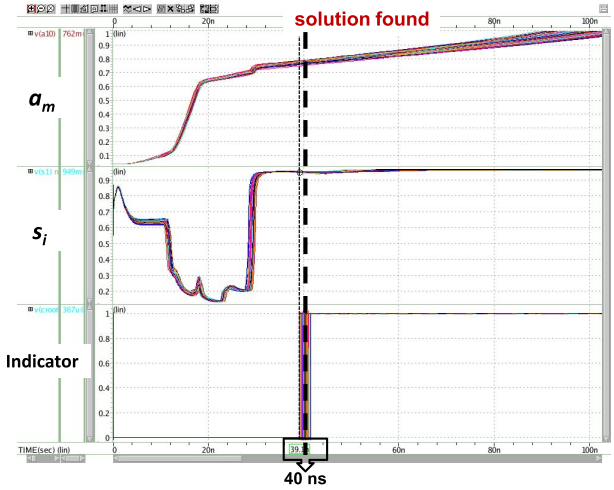

Fig. 13. Waveforms of signals representing a single $a_m(t)$ and $s_i(t)$ for a 3-SAT problem instance (op-amp-based design) with 42 clauses and 10 signal variables under Monte Carlo simulations. The third row represents the output of DVC, which turns to V_{DD} once the solution is found.

op-amp-based and the simpler a_m-based AC-SAT. The two designs are used to solve 2000 randomly generated, hard ($\alpha = 4.25$) 3-SAT problems containing 1000 instances of a small problem size ($N = 10$) and 1000 instances of a larger problem size ($N = 50$). It has been verified that all these 2000 instances are solvable. With the same fixed supply voltage and initial conditions, AC-SAT with the op-amp-based a_m design solves 91.1.% of the $N = 10$ instances and 58.2% of the $N = 50$ instances, respectively, while AC-SAT with the simpler a_m design solves 86.9% of the $N = 10$ instances and 46.5% of the $N = 50$ instances, respectively. (Note that AC-SAT did not solve all the problems because of the physical voltage limit we imposed.)[3] These results indicate that, as expected, within the same physical constraints, AC-SAT based on the exponential growth a_m is more effective than AC-SAT with the $(1 - \epsilon_2\ e^{-qt})$-type a_m growth. Note that an exponential growth a_m circuit implemented with an op-amp does consume larger area and energy, while the simpler a_m circuit trades off area and energy with solver capability.

D. Performance Comparisons

To further investigate the effectiveness of AC-SAT, we compare the simpler a_m-based AC-SAT design with: 1) a software program that solves the systems (3), (4) using an adaptive Runge-Kutta, fifth-order Cash-Karp method and 2) the software MiniSat solver [19]. The software programs are running on the same digital computer. We randomly generated 5000 hard ($\alpha = 4.25$) 3-SAT problems that contain 1000 instances for each problem size of N = 10, 20, 30, 40, 50. The same initial conditions are applied whenever appropriate. Table I summarizes the average time needed to find solutions for each problem size. The AC-SAT column reports the analog/physical times taken by AC-SAT.

[3]This is not a limitation of the CTDS theory, but rather the supply voltage bound set in our design. In fact, our software implementation of CTDS is able to solve all the problems. Relaxing the voltage bound will help solve more problem instances, which is left for future work.

TABLE I
PERFORMANCE COMPARISON OF AC-SAT, SOFTWARE CTDS, AND MINISAT

SAT solver		AC-SAT	CTDS	MiniSat
Platform		ASIC 32nm CMOS	Intel Core i7-4700 @2.4 GHz	Intel Core i7-4700 @ 2.4GHz
Average time for each size N (s)	N=10	4×10^{-9}	4.40×10^{-4}	2.3×10^{-4}
	N=20	7×10^{-9}	3.91×10^{-3}	2.4×10^{-4}
	N=30	10^{-8}	1.62×10^{-2}	2.8×10^{-4}
	N=40	1.2×10^{-8}	5.22×10^{-2}	3.1×10^{-4}
	N=50	1.4×10^{-8}	1.13×10^{-1}	3.7×10^{-4}

The CTDS and MiniSat columns report the CPU times of the two software implementations, respectively. (To be fair, only the times taken by the solved problems for all three methods are included.) Observe that the times in the CTDS column increase nearly exponentially as the problem size increases. This is natural, since the numerical integration happens on a digital Turing machine, and in order to ensure the pre-set accuracy of computing the chaotic trajectory the Runge-Kutta algorithm has to do a very large number of window-refining discretization steps. As seen from the data in Table I, AC-SAT demonstrates average speedup factors of $\sim10^5$ to $\sim10^6$ and $\sim10^4$ over software CTDS and MiniSat, respectively.

AC-SAT is also very competitive compared with existing hardware-based approaches. For example, a recent work [29] reported a CPU+FPGA-based MiniSat solver achieving $\sim$4X performance improvement over CPU-based MiniSat. Since ASIC implementations typically achieves a maximum of 10X performance improvement over their FPGA counterparts [38], compared with a projected ASIC version of the FPGA design in [29], AC-SAT would still result in $\sim$600X or higher speedup. We do not directly compare with the custom digital IC in [30] since our simulation-based system cannot solve the large size problems considered in [30]. (Note that the total solving times reported in [30] are extrapolated instead of directly obtained from simulation.) It is reported in [30] that an average speedup of $\sim10^3$X over CPU-based MiniSat is obtained. As contrast, AC-SAT achieves $\sim10^4$X speedup over CPU-based MiniSat. T

Readers may be concerned with the complexity of the analog hardware design as well as other issues such as noise. It is important to note that the analog solver core is modular and consists of arrays with the same topology. Furthermore, the CTDS theory has been shown to be robust against noise [36]. AC-SAT is programmable, which means that different problem instances can be programmed or mapped to the AC-SAT circuit. AC-SAT is also modular, implying that: 1) it can be more easily extended to construct a larger solver and 2) multiple AC-SAT components can be used to solve the same problem instance by providing different initial conditions, hence allowing larger space to be searched simultaneously.

The current implementation of AC-SAT, however, does have some limitations. In particular, while the modular structure allows possible expansion to solve problems with larger numbers of variables and clauses, it can only address problems with clauses that have no more than the given number of k literals ($k = 3$ here). One way to solve such problems is to use the host processor to convert k-SAT problems (where $k > 3$) to 3-SAT problems (which can be done in polynomial time [16]). How to directly tackle such challenges in hardware is left for future work.

V. CONCLUSION

We presented a proof-of-principle analog system, AC-SAT, based on the CTDS in [1] to solve 3-SAT problems. The design can be readily extended to general k-SAT problems. AC-SAT is modular, programmable and can be used as a SAT solver coprocessor. In this implementation the circuit size grows polynomially [$O(N^2)$] as the problem size increases. Three different design alternatives were proposed and verified for implementing the auxiliary variable dynamics required by the CTDS. Detailed SPICE simulation results show that AC-SAT can indeed solve SAT problems efficiently and can tolerate well device variations. Compared with other SAT solvers, AC-SAT can achieve $\sim10^4\times$ speedup over MiniSat running on a state-of-the-art digital processor, and can offer over 600× speedup over projected digital ASIC implementation of MiniSat.

Regarding the practical use of a hardware solver, we note that there are instances in the SAT contests that take a very long time (e.g., days or even months) to solve. The reason for the long (and exponentially growing) running time is due not only to the size of the problems, but also to their hardness. It has been demonstrated that when the constraint density (M/N) of a problem instance is between 4 and 5 (for 3-SAT), the problem can be very hard and take exponentially growing time for current software solvers to find a solution. Our work, together with its theoretical basis, however, provides a means to trade time for energy in order to speed up computations. With the circuit-friendly theory and proof-of-principle hardware implementation, we can solve hard SAT problems much faster than with software solvers on digital machines, however, at the expense of other resources such as energy (voltage and or/current values). Such tradeoffs are desirable for certain time-sensitive problems.

The CTDS equations (especially the dynamics for the auxiliary variables) and their analog implementations are not unique. It is quite possible that better forms and implementations exist. The fact that our proof-of-principle circuit implementations significantly outperform state-of-the-art solvers on digital computers are an indication that analog hardware SAT solvers have a great potential as application-specific processors for discrete optimization. As future work, we will further investigate alternative implementations of the auxiliary variable dynamics as well as methods to handle problem instances that do not fit on a given hardware implementation, e.g., through problem decomposition. Moreover, we will explore other methods that can, in principle, solve SAT problems even more efficiently, e.g., by combining clause learning (handled by a digital processor) with our analog solver.

ACKNOWLEDGMENT

The authors would like to thank S. Datta, A. Raychowdhury, M. Niemier, and G. Cauwenberghs for their useful discussions and anonymous reviewers for helpful comments.

REFERENCES

[1] M. Ercsey-Ravasz and Z. Toroczkai, "Optimization hardness as transient chaos in an analog approach to constraint satisfaction," *Nature Phys.*, vol. 7, no. 12, pp. 966–970, Dec. 2011.

[2] M. M. Waldrop, "More than Moore," *Nature*, vol. 530, pp. 144–147, Feb. 2016.

[3] L. O. Chua, T. Roska, and P. L. Venetianer, "The cnn is universal as the Turing machine," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 40, no. 4, pp. 289–291, Apr. 1993.

[4] L. O. Chua and L. Yang, "Cellular neural networks: Theory," *IEEE Trans. Circuits Syst.*, vol. CSI-35, no. 10, pp. 1257–1272, Oct. 1988.

[5] Q. Lou, I. Palit, A. Horvath, X. S. Hu, M. Niemier, and J. Nahas, "TFET-based operational transconductance amplifier design for CNN systems," in *Proc. 25th Ed. Great Lakes Symp. (VLSI)*, 2015, pp. 277–282.

[6] P. A. Merolla *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 2014.

[7] H. Siegelmann, *Neural Networks and Analog Computation: Beyond the Turing Limit*. New York, NY, USA: Springer-Verlag, 2012.

[8] R. Daniel, J. R. Rubens, R. Sarpeshkar, and T. K. Lu, "Synthetic analog computation in living cells," *Nature*, vol. 497, no. 7451, pp. 619–623, 2013.

[9] J. Lu, S. Young, I. Arel, and J. Holleman, "A 1 TOPS/W analog deep machine-learning engine with floating-gate storage in 0.13 μm CMOS," *IEEE J. Solid-State Circuits*, vol. 50, no. 1, pp. 270–281, Jan. 2015.

[10] R. S. Amant *et al.*, "General-purpose code acceleration with limited-precision analog computation," *ACM SIGARCH Comput. Archit. News*, vol. 42, no. 3, pp. 505–516, 2014.

[11] H. Mostafa and L. K. Müller, and G. Indiveri, "An event-based architecture for solving constraint satisfaction problems," *Nature Commun.*, vol. 6, Dec. 2015, Art. no. 8941.

[12] S.-C. Liu, J. Kramer, G. Indiveri, T. Delbrück, and R. Douglas, *Analog VLSI—Circuits and Principles*. Cambridge, MA, USA: MIT Press, Nov. 2002.

[13] A. Ben-Hur, H. Siegelmann, and S. Fishman, "A theory of complexity for continuous time systems," *J. Complex.*, vol. 18, pp. 51–86, Mar. 2002.

[14] M. S. Branicky, "Analog computation with continuous ODEs," in *Proc. Workshop Phys. Comput.*, Dallas, TX, USA, 1994, pp. 265–274.

[15] H. Siegelmann, "Computation beyond the Turing limit," *Science*, vol. 268, no. 5210, pp. 545–548, 1995.

[16] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Series of Books in the Mathematical Sciences), 1st ed. New York, NY, USA: W. H. Freeman, Jan. 1979.

[17] T. Brueggemann and W. Kern, "An improved deterministic local search algorithm for 3-SAT," *Theor. Comput. Sci.*, vol. 329, nos. 1–3, pp. 303–313, 2004.

[18] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an efficient sat solver," in *Proc. 38th Annu. Design Autom. Conf. (ACM)*, 2001, pp. 530–535.

[19] N. Eén and N. Sörensson, "An extensible sat-solver," in *Theory and Applications of Satisfiability Testing*. New York, NY, USA: Springer-Verlag, 2003, pp. 502–518.

[20] P. Knot and A. Darwiche, "Rsat 2.0: Sat solver description." Dept. Comput. Sci., Univ. California Los Angeles (UCLA), Autom. Reason. Group, Los Angeles, CA, USA, Tech. Rep. D-153, 2007.

[21] B. Selman, H. A. Kautz, and B. Cohen, "Local search strategies for satisfiability testing," in *Proc. DIMACS Ser.*, 1996, pp. 521–532.

[22] G. Dueck, "New optimization heuristics: The great deluge algorithm and the record-to-record travel," *J. Comput. Phys.*, vol. 104, no. 1, pp. 86–92, 1993.

[23] S. Seitz, M. Alava, and P. Orponen, "Focused local search for random 3-satisfiability," *J. Stat. Mech., Theory Exp.*, vol. 2005, p. P06006, Jun. 2005.

[24] M. Davis, G. Logemann, and D. Loveland, "A machine program for theorem-proving," *Commun. ACM*, vol. 5, no. 7, pp. 394–397, 1962.

[25] J. P. M. Silva and K. A. Sakallah, "GRASP—A new search algorithm for satisfiability," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 1996, pp. 220–227.

[26] L. Zhang, C. F. Madigan, M. H. Moskewicz, and S. Malik, "Efficient conflict driven learning in a Boolean satisfiability solver," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2001, pp. 279–285.

[27] J. D. Davis, Z. Tan, F. Yu, and L. Zhang, "Designing an efficient hardware implication accelerator for SAT solving," in *Proc. SAT*, 2008, pp. 48–62.

[28] J. D. Davis, Z. Tan, F. Yu, and L. Zhang, "A practical reconfigurable hardware accelerator for Boolean satisfiability solvers," in *Proc. 45th Annu. Design Autom. Conf.*, 2008, pp. 780–785.

[29] J. Thong and N. Nicolici, "FPGA acceleration of enhanced Boolean constraint propagation for sat solvers," in *Proc. IEEE Int. Conf. Comput.-Aided Design*, Nov. 2013, pp. 234–241.

[30] K. Gulati and S. P. Khatri, "Accelerating Boolean satisfiability on a custom ic," in *Hardware Acceleration of EDA Algorithms*. New York, NY, USA: Springer-Verlag, 2010, pp. 33–61.

[31] K. Gulati, M. Waghmode, S. P. Khatri, and W. Shi, "Efficient, scalable hardware engine for Boolean satisfiability and unsatisfiable core extraction," *IET Comput. Digit. Techn.*, vol. 2, no. 3, pp. 214–229, May 2008.

[32] D. A. Basford, J. M. Smith, R. J. Connor, B. J. MacLennan, and J. Holleman, "The impact of analog computational error on an analog Boolean satisfiability solver," in *Proc. ISCAS*, May 2016, pp. 2503–2506.

[33] B. Molnár and M. Ercsey-Ravasz, "Asymmetric continuous-time neural networks without local traps for solving constraint satisfaction problems," *PLoS ONE*, vol. 8, no. 9, p. e73400, 2013.

[34] M. Ercsey-Ravasz and Z. Toroczkai, "The chaos within Sudoku," *Sci. Rep.*, vol. 2, p. 725, Oct. 2012.

[35] *Predictive Technology Model (PTM)*. Accessed: Jun. 1, 2011. [Online]. Available: http://ptm.asu.edu/

[36] R. Sumi, B. Molnár, and M. Ercsey-Ravasz, "Robust optimization with transiently chaotic dynamical systems," *Europhys. Lett.*, vol. 106, no. 4, p. 40002, 2014.

[37] G. Nikandish, B. Sedighi, and M. S. Bakhtiar, "Performance comparison of switched-capacitor and switched-current pipeline ADCs," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2007, pp. 2252–2255.

[38] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 2, pp. 203–215, Feb. 2007.

Xunzhao Yin (S'16) received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 2013. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, USA.

He has been a Research Assistant with the University of Notre Dame since 2013. He is a member of the Center for Low Energy System Technology, where he is involved in the novel circuits and systems based on beyong-CMOS technologies. His current research interests include hardware security, low-power circuit design, and novel computing paradigms with both CMOS and emerging technologies.

Behnam Sedighi received the Ph.D. degree in electrical engineering from the Sharif University of Technology, Tehran, Iran, in 2008.

From 2009 to 2011, he was a Senior Circuit Designer with IHP Microelectronics, Frankfurt, Germany, where he was involved in analog/mixed-signal integrated circuits for optical communications. He then joined the Center of Energy Efficient Telecommunications and NICTA, University of Melbourne, Melbourne, VIC, Australia, and subsequently, the Center for Low-Energy Systems Technology, University of Notre Dame, IN, USA, as a Research Associate. He is currently with Qualcomm Inc, San Diego, CA, USA. His current research interests include broadband ICs, communication circuits and systems, data converters, and nanoelectronics.

Melinda Varga received the B.S. and M.S. degrees from Babes-Bolyai University, Cluj-Napoca, Romania, and the Ph.D. degree in physics from the University of Notre Dame, Notre Dame, IN, USA.

She is currently a Post-Doctoral Researcher with The College of Wooster, Wooster, OH, USA. She has authored or co-authored papers, which cover a wide range of topics. Her current research interests include nonlinear dynamical systems and chaos theory, complex networks, brain neuronal systems, and cellular biology.

Dr. Varga received the Research Performance Scholarship during her M.S. studies from Babes-Bolyai University And the Physics Graduate Research Dissertation Award from the University of Notre Dame in 2017. She was one of the finalists of the GSNP Student Speaker Award at the APS March Meeting in 2016.

Mária Ercsey-Ravasz received the B.Sc. and M.Sc. degrees in physics from Babes-Bolyai University, Cluj-Napoca, Romania, and the Ph.D. joint degree in physics from Babes-Bolyai University and in information technology (infobionics) from Peter Pazmany Catholic University, Budapest, Hungary in 2008.

She was a Post-Doctoral Researcher with the iCeNSA, University of Notre Dame, Notre Dame, IN, USA. She is currently a Researcher with Babes-Bolyai University and the Romanian Institute of Science and Technology, Bucharest, Romania. She has authored or co-authored more than 30 ISI articles. Her current research interests include network science with applications in different domains, such as neuroscience, analog computing, optimization problems, and nonlinear dynamics.

Dr. Ercsey-Ravasz was a recipient of Marie Curie Fellowship, the UNESCO-L'Oreal National Fellowship "For Women in Science" in 2013, and the Constantin Miculescu Award of the Romanian Academy of Sciences in 2015.

Zoltán Toroczkai received the Ph.D. degree in theoretical physics from the Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, in 1997.

He was a Post-Doctoral Researcher with the Condensed Matter Physics Group, University of Maryland, College Park, MD, USA. He was a Director Funded Fellow at the Los Alamos National Laboratory (LANL), Los Alamos, NM, USA, where he was also a regular Research Staff Member with the Complex Systems Group. From 2004 to 2005, he was the Deputy Director with the Center for Nonlinear Studies, LANL. In 2006, he joined the Department of Physics, University of Notre Dame, Notre Dame, IN, USA, where he is currently a Professor and a Concurrent Professor with the Department of Computer Science and Engineering. He has authored and co-authored over 90 papers in peer-reviewed publications. His current research interests include statistical physics, nonlinear dynamical systems, and mathematical physics with topics including fluid flows, reaction kinetics, interface growth, population dynamics, epidemics, agent-based systems and game theory, complex networks, foundations of computing, and brain neuronal systems.

Dr. Toroczkai has been on the editorial board at the *European Journal of Physics B, Scientific Reports, Chaos* and an Associate Editor of *Network Science*. He was an elected APS Fellow in 2012 upon nomination by GSNP "For his contributions to the understanding of the statistical physics of complex systems, and in particular for his discoveries pertaining to the structure and dynamics of complex networks."

Xiaobo Sharon Hu (S'85–M'89–SM'02–F'16) received the B.S. degree from Tianjin University, Tianjin, China, the M.S. degree from the Polytechnic Institute of New York, Brooklyn, NY, USA, and the Ph.D. degree from Purdue University, West Lafayette, IN, USA.

She is currently a Professor with the Department of Computer Science and Engineering, University of Notre Dame, Notre Dame, IN, USA. She has authored or co-authored more than 250 papers. Her current research interests include real-time embedded systems, low-power system design, and computing with emerging technologies.

Dr. Hu was a recipient of the NSF CAREER Award in 1997, and the Best Paper Award from the Design Automation Conference in 2001, and the IEEE Symposium on Nanoscale Architectures, in 2009. She is the Program Chair of the 2016 Design Automation Conference (DAC) and the TPC Co-Chair of 2014 and 2015 DAC. She was an Associate Editor of the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION, the *ACM Transactions on Design Automation of Electronic Systems*, and the *ACM Transactions on Embedded Computing*.