An Analog SAT Solver based on a Deterministic Dynamical System (Invited Paper)

Xunzhao Yin
University of Notre Dame
Notre Dame, Indiana
xyin1@nd.edu

Zoltán Toroczkai
University of Notre Dame
Notre Dame, Indiana
toro@nd.edu

Xiaobo Sharon Hu
University of Notre Dame
Notre Dame, Indiana
shu@nd.edu

ABSTRACT

Boolean Satisfiability (SAT), the first problem proven to be NP-complete, is intractable on digital computers based on the von Neumann architecture. An efficient SAT solver can benefit many applications such as artificial intelligence, circuit design, and functional verification. Recently, a SAT solver approach based on a deterministic, continuous-time dynamical system (CTDS) was introduced [14]. This approach shows polynomial analog time-complexity on even the hardest k-SAT ($k \geq 3$) problem instances, but at an energy cost dependent on exponentially growing auxiliary variables. This paper reports 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 co-processor and is programmable for handling different problem specifications. Furthermore, with its modular design, AC-SAT can be readily extended to solve larger size problems. SPICE simulation results show that AC-SAT can indeed solve the SAT problems, and it has 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.

ACM Reference Format:
Xunzhao Yin, Zoltán Toroczkai, and Xiaobo Sharon Hu. 2017. An Analog SAT Solver based on a Deterministic Dynamical System (Invited Paper). In *Proceedings of ACM ICCAD conference, Irvine, CA USA, November 2017 (ICCAD'17),* 6 pages.
https://doi.org/10.475/123_4

1 INTRODUCTION

With Moore's Law coming to end [37], 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) [6, 7, 21] and IBM's TrueNorth [23], have been shown to be promising alternatives for solving a range of problems in, say, sensory processing (vision, pattern recognition) and robotics. Analog mix-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 [32]. Such systems have received increasing attention in recent years (e.g., [8, 22, 34]), including parallel analog implementations [26].

In analog computing [20], 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 the output of the computation is the set of convergent states to those attractors [3]. Although it has been shown that systems of ordinary differential equations can simulate any Turing machine [4, 6, 31], and hence they are computationally universal, 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 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 our previous paper [38] and 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 it is well known that for $k \geq 3$, k-SAT is NP-complete [16]. The currently best known deterministic, sequential discrete algorithm that exploits some properties of the search space has a worst case complexity of $O(1.473^N)$ [5]. 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 [25], MiniSat [13], RSat [28], WalkSAT [30], Focused Record-to-Record Travel (FRRT) [12] and Focused Metropolis Search (FMS) [29]. They typically consist of decision, deduction, conflict analysis and other functions [11] that employ the capability of digital computers to assign values to literals, conduct Boolean constraint propagation (BCP) and backtrack conflicts [33, 39].

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 [9, 10, 36].

ICCAD'17, November 2017, Irvine, CA USA

ACM ISBN 123-4567-24-567/08/06…$15.00
https://doi.org/10.475/123_4

Speedups of anywhere between 3X and 38X have been reported when comparing these FPGA based solvers over MiniSat [13], 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 (GRASP) and accelerates traversal of the implication graph and conflict clause generation, has been introduced in [17, 18]. A speed up of $\sim 10^3$ over MiniSat was reported based on simulation together with extrapolation. Performance of these hardware based approaches still have 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 [2] using the theoretical proposal from [24] based on the CNN architecture. However, the theory in [24] has exponential analog-time complexity, thus is much less efficient than the solver from [14], which forms the basis for this paper. Furthermore, the circuit from [2] seems to have been implemented only for a 4×4 problem size, and no hardware simulation and comparison results were reported.

[26] proposes 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 P≠NP, in order to have efficient, polynomially scaling solution times, one would require exponentially many computing elements, that is, exponentially scaling hardware resources. However, the method in [14] 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 vs energy and time vs hardware (distributed). The heuristic stochastic search in [26] is effectively a simulated annealing method, which implies high exponential runtimes for worst case formulas. In contrast, the analog approach in [14] 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 [14].

In [38] and this paper 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 [14]. 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 [14], [15] 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*, 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 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 a recent FPGA solver [36], AC-SAT offers more than ~600X speedup.

In the rest of the paper, we first review the basic CTDS theory in Section 2. Sec. 3 introduces the overall AC-SAT design at a high level. Sec. 4 first shows simulation-based validation results from functionality perspective, and then compares AC-SAT with software implementations of the CTDS SAT solver and MiniSat solver. Finally, Sec. 5 concludes the paper.

2 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 [14], 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

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

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, it is clear 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. Ref [14] proposed

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

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

[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.

in which (3) describes a gradient descent on V and (4) is an exponential growth driven by the level of non-satisfiability 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 which have been unsatisfied for the longest time during the dynamics, resulting in an analog version of a focused search-type [29] dynamics. Also note that system (3) - (4) is 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 see [14].

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 are feasible in terms of physical implementations and costs. Such systems as ours essentially convert time costs into energy costs.

3 SOLVER DESIGN OVERVIEW

In this section, we present an high level overview of AC-SAT, our proposed analog SAT solver circuit based on the CTDS theory in Sec. 2. Our circuit design aims to make the hardware solver configurable and modular while keeping the circuit simple and power efficient.

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_i's in (5), auxiliary variable circuit (AVC) which implements the dynamics of auxiliary variables a_m's 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_i's 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_m's 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_i's to the AVC and the digital version of s_i's to the DVC. Based on the digital values of s_i's, the DVC determines whether a solution to the SAT problem is found at that time.

Below, we briefly demonstrate the conceptual design of the three circuit components of AC-SAT using the 3-SAT problem (i.e., three non-zero $c_{m,j}$'s for each clause) as an example. AC-SAT for any k-SAT problem can be designed following the same principle. Due to the paper space, parts of implementation details of the three components are omitted, and can be found in [38].

3.1 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. Several novel circuit design ideas have been introduced to implement the dynamics in (5) and (6). We present the basic design of the SDC array element along with some signal notations. More detailed implementations can be found in [38].

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 shows the conceptual design of the s_i element that realizes (5). The s_i element contains a capacitor, C, connected to the $\mathscr{M}$ Branch blocks achieving $D_{m,i}$ in (6) (where $\mathscr{M}$ is the maximum number of clauses in a 3-SAT problem that can be fitted into the circuit), 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 $s_i \in [-1, 1]$ (resp., $-s_i \in [1, -1]$) is mapped to $V_i \in [GND, V_{DD}]$ (resp., $\overline{V_i} \in [V_{DD}, GND]$). 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}).

3.2 Auxiliary Variable Circuits

As pointed out in Section 2, the auxiliary variables, a_m's as defined in (4), are used to help avoid the gradient descent search being stuck in non-solution attractors. The a_m signal follows an exponential growth driven by the level of non-satisfiability in clause C_m, and we developed AVC to realize the exponential growth.

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. These a_m elements correspond to the clauses of a given

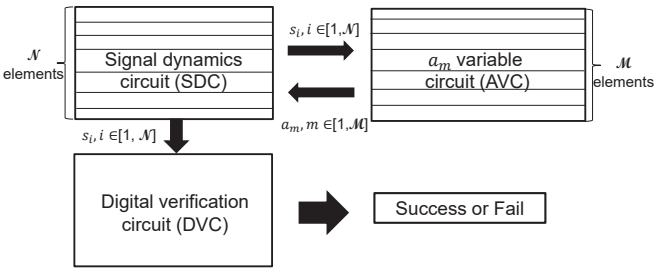


Figure 1: High-level block diagram of AC-SAT. The SDC contains $\mathscr{N}$ elements while the AVC contains $\mathscr{M}$ elements. It can solve k-SAT problem instances with up to $\mathscr{N}$ variables and $\mathscr{M}$ clauses.

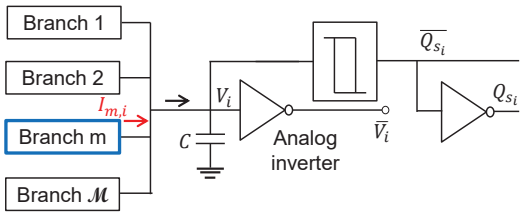


Figure 2: Design of one array element in the SDC.

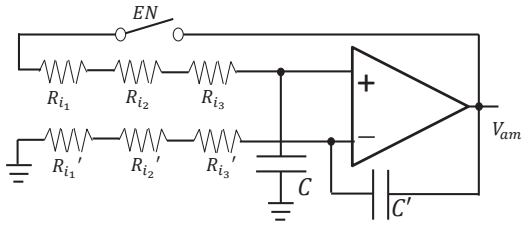


Figure 3: Conceptual design of one element in the AVC. Actual realization of the resistive element can be found in [38].

problem. Fig. 3 illustrates the conceptual design of the a_m element, based on a non-inverting 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_{i_1}, R_{i_2} and R_{i_3} are associated with the three signals in a_m's clause while R'_{i_1}, R'_{i_2} and R'_{i_3} are identical to R_{i_1}, R_{i_2} and R_{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:

$$C\frac{dV_{a_m}}{dt} = V_{a_m}/(R_{i_1} + R_{i_2} + R_{i_3}). \tag{8}$$

This equation implies the exponential growth of V_{a_m}. The R's in Fig. 3 are tunable resistive elements, and they are implemented by transmission gates that are controlled by variable signals (i.e., V_{i_1}, V_{i_2}, and V_{i_3}). These R's represent the satisfiability of corresponding variables in clause C_m. If C_m is satisfied, at least one of the R's will cut off the current path from op-amp's inverting input to ground and from non-inverting input to V_{a_m}. Detailed implementation of the AVC elements and the impact realization of V_{a_m} on the Branch blocks of SDC in Fig. 2 can be found in [38].

3.3 Digital Verification

The goal of the DVC is to determine if a solution (the set of s_i's) 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 $\mathcal{M}$ XOR gates and an array of $\mathcal{M}$ NAND gates as shown in Fig. 4. 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

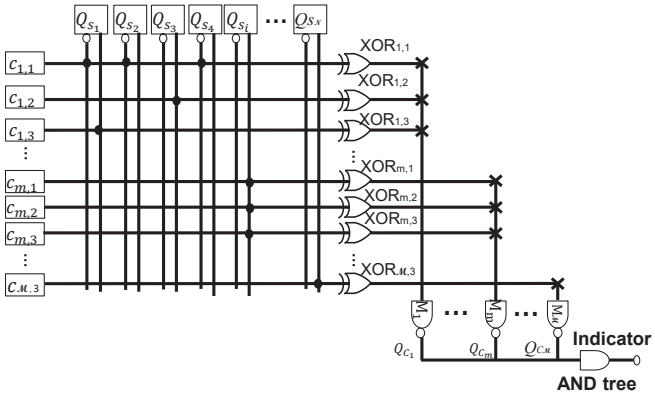


Figure 4: Schematic of the DVC.

logic "0"). The outputs of the DVC are digital 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. For unsatisfiable problems, our solver is a MaxSAT solver, and will output results with minimum number of unsatisfied clauses.

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 will be presented elsewhere.

3.4 Alternative AVC Designs

The op-amp based AVC described in Section 3.2 realizes an exponentially growing a_m variable aiming to address hard SAT problems (some SAT instances with constraint density α=$M/N \gtrsim 4.25$) 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 an alternative circuit design to implement an a_m function that has a $(1 - \epsilon_2 e^{-qt})$-type growth to a saturation value. In the remainder we will refer to this version of a_m growth as the "simpler" version.

Fig. 5 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 resistive elements. The first order differential equation that governs V_{a_m} can be written as

$$C\frac{dV_{a_m}}{dt} = (V_{DD} - V_{a_m})/(R_{i_1} + R_{i_2} + R_{i_3}). \tag{9}$$

R_{i_1}, R_{i_2}, R_{i_3} are same as the resistive elements in Fig. 3, realized by transmission gates controlled by V_{i_1}, V_{i_2} and V_{i_3}, depicted in [38]. 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 cut off the current path from V_{DD} to the capacitor. This circuit

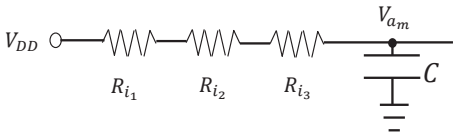


Figure 5: Conceptual design of the AVC element realizing the $(1 - \epsilon_2 e^{-qt})$-type growth. Implementation of the resistive elements can be found in [38].

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. 5 does not realize the exponential growth specified in (4), it can indeed get captured into non-solution attractors indefinitely for some very hard formulas. However, we 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 (solvable), and the dynamics would only rarely get stuck.

4 EVALUATION

In this section, we present our validation of AC-SAT. We describe the basic functional validation and then compare the performance of AC-SAT with a state-of-the-art digital solver..

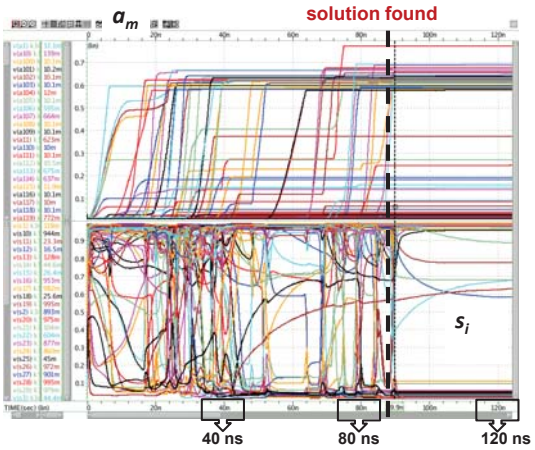
4.1 Functional Validation

We have built our proposed analog SAT solver, AC-SAT, at the transistor level in HSPICE based on the PTM 32nm CMOS model [1]. All the circuit components use $V_{DD} = 1V$. The transistor sizes can all be found in [38].

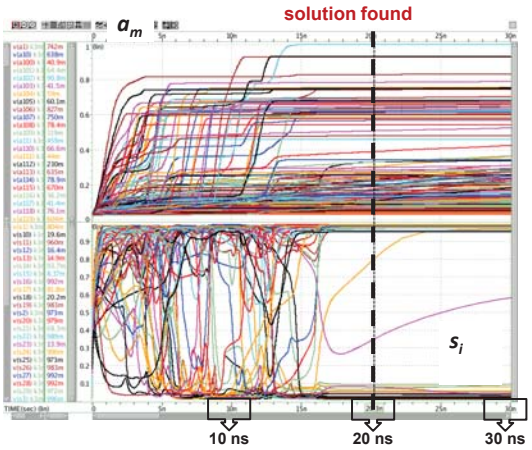
To demonstrate that AC-SAT indeed behaves as specified by the CTDS dynamics in (3) and (4), we examine the waveforms of signals s_i and a_m. Fig. 6 shows the two sets of s_i and a_m waveforms from a 3-SAT problem instance having 50 variables and 212 clauses: Fig. 6(a) for the op-amp based a_m implementation (realizing the $(\epsilon_1 e^{qt})$-type a_m growth), and Fig. 6(b) for the simpler a_m implementation (realizing the $(1 - \epsilon_2 e^{-qt})$-type a_m growth). For both designs, AC-SAT successfully finds a solution after a certain time as indicated by the vertical dashed lines. As can be seen from the s_i trajectories, the s_i signals stabilize (i.e., converge) after a solution is found, and one can see that the a_m's grow most rapidly in the op-amp based design due to the exponential growth function.

4.2 Performance Comparisons

To further investigate the effectiveness of AC-SAT, we compare the simpler a_m based AC-SAT design with (i) a software program that solves the system (3)-(4) using an adaptive Runge-Kutta, fifth-order Cash-Karp method and (ii) the software MiniSat solver [13]. 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 1 summarizes the average time needed to find solutions for each problem size. The AC-SAT column reports the analog/physical



(a) With $\epsilon_1 e^{qt} a_m$ growth.



(b) With $1 - \epsilon_2 e^{-qt} a_m$ growth.

Figure 6: 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.

times taken by AC-SAT. 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.) As seen from the data in Table 1, AC-SAT demonstrates average speedup factors of $\sim 10^5$ to $\sim 10^6$ and $\sim 10^4$ over software CTDS and MiniSat, respectively. Moreover, AC-SAT is also very competitive compared with existing hardware based approaches which we have illustrated in details in [38].

5 CONCLUSIONS

We presented a proof-of-principle analog system, AC-SAT, based on the CTDS in [14] 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 co-processor. SPICE simulation results show that our AC-SAT can indeed solve SAT problems efficiently, and can tolerate well device variations.

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 (see [38]) is 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.

Acknowledgement: The authors acknowledge useful discussions with S. Datta, A. Raychowdhury, M. Niemier and G. Cauwenberghs. This project was supported in part by the National Science Foundation under grant numbers CCF-1644368 and 1640081, and the Nanoelectronics Research Corporation (NERC), a wholly-owned subsidiary of the Semiconductor Research Corporation (SRC), through Extremely Energy Efficient Collective Electronics (EXCEL), an SRC-NRI Nanoelectronics Research Initiative under Research Task ID 2698.004 (XSH,ZT). MER was funded by a European Commission Horizon 2020 Program Grant No. 668863-SyBil-AA and a Romanian CNCS-UEFISCDI Research Grant No. PN-III-P2-2.1-BG-2016-0252.

REFERENCES

[1] [n. d.]. Predictive Technology Model (PTM). http://ptm.asu.edu/. ([n. d.]).
[2] D.A. Basford, J.M. Smith, R.J. Connor, B.J. MacLennan, and Holleman J. 2016. The Impact of Analog Computational Error on an Analog Boolean Satisfiability Solver. In *ISCAS*.
[3] A Ben-Hur, HT Siegelmann, and S Fishman. 2002. A theory of complexity for continuous time systems. *Journal of Complexity* 18 (2002), 51–86. https://doi.org/{10.1006/jcom.2001.0581}
[4] M.S. Branicky. 1994. Analog computation with continuous ODEs. *Workshop on Physics and Computation, Dallas TX USA* (1994), 265–274.
[5] T. Brueggemann and W. Kern. 2004. An improved local search algorithm for 3-SAT. *Theoretical Computer Science* 329, 1-3 (2004), 303–313.
[6] L.O. Chua, T Roska, and PL Venetianer. 1993. The CNN is universal as the Turing machine. *TCAS I* 40, 4 (APR 1993), 289–291. https://doi.org/{10.1109/81.224308}
[7] LO Chua and L Yang. 1988. Cellular Neural Networks - Theory. *TCAS I* 35, 10 (OCT 1988), 1257–1272. https://doi.org/{10.1109/31.7600}
[8] Ramiz Daniel, Jacob R Rubens, Rahul Sarpeshkar, and Timothy K Lu. 2013. Synthetic analog computation in living cells. *Nature* 497, 7451 (2013), 619–623.
[9] John D Davis, Zhangxi Tan, Fang Yu, and Lintao Zhang. 2008. Designing an efficient hardware implication accelerator for SAT solving. In *SAT 2008*. Springer, 48–62.
[10] John D Davis, Zhangxi Tan, Fang Yu, and Lintao Zhang. 2008. A practical reconfigurable hardware accelerator for Boolean satisfiability solvers. In *Proceedings of the 45th annual Design Automation Conference*. ACM, 780–785.
[11] Martin Davis, George Logemann, and Donald Loveland. 1962. A machine program for theorem-proving. *Commun. ACM* 5, 7 (1962), 394–397.

Table 1: Performance 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}

[12] Gunter Dueck. 1993. New Optimization Heuristics. *J. Comput. Phys.* 104, 1 (1993), 86–92.
[13] Niklas Eén and Niklas Sörensson. 2003. An extensible SAT-solver. In *Theory and applications of satisfiability testing*. Springer, 502–518.
[14] Maria Ercsey-Ravasz and Zoltan Toroczkai. 2011. Optimization hardness as transient chaos in an analog approach to constraint satisfaction. *Nature Physics* 7, 12 (DEC 2011), 966–970. https://doi.org/{10.1038/NPHYS2105}
[15] M. Ercsey-Ravasz and Z. Toroczkai. 2012. The chaos within Sudoku. *Scientific Reports* 2 (OCT 11 2012), 725.
[16] Michael R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)* (first edition ed.). W. H. Freeman & Co Ltd.
[17] Kanupriya Gulati and Sunil P Khatri. 2010. Accelerating Boolean Satisfiability on a Custom IC. In *Hardware Acceleration of EDA Algorithms*. Springer, 33–61.
[18] Kanupriya Gulati, Mandar Waghmode, SP Khatr, and Weiping Shi. 2008. Efficient, scalable hardware engine for Boolean satisfiability and unsatisfiable core extraction. *Computers & Digital Techniques, IET* 2, 3 (2008), 214–229.
[19] Ian Kuon and Jonathan Rose. 2007. Measuring the gap between FPGAs and ASICs. *IEEE Transactions on computer-aided design of integrated circuits and systems* 26, 2 (2007), 203–215.
[20] Shih-Chii Liu, Jörg Kramer, Giacomo Indiveri, Tobias Delbrück, and Rodney Douglas. 2002. *Analog VLSI - Circuits and Principles*. MIT Press.
[21] Qiuwen Lou, Indranil Palit, Andras Horvath, X Sharon Hu, Michael Niemier, and Joseph Nahas. 2015. TFET-based Operational Transconductance Amplifier Design for CNN Systems. In *Proceedings of the 25th edition on Great Lakes Symposium on VLSI*. ACM, 277–282.
[22] Junjie Lu, Steven Young, Itamar Arel, and Jeremy Holleman. 2015. A 1 TOPS/W analog deep machine-learning engine with floating-gate storage in 0.13 μm CMOS. *IEEE Journal of Solid-State Circuits* 50, 1 (2015), 270–281.
[23] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. 2014. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 345, 6197 (2014), 668–673.
[24] Botond Molnár and Mária Ercsey-Ravasz. 2013. Asymmetric continuous-time neural networks without local traps for solving constraint satisfaction problems. *PloS one* 8, 9 (2013), e73400.
[25] Matthew W Moskewicz, Conor F Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. 2001. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th annual Design Automation Conference*. ACM, 530–535.
[26] Hesham Mostafa, Lorenz K. Müller, and Giacomo Indiveri. 2015. An event-based architecture for solving constraint satisfaction problems. *Nature Communications* 6 (December 2015), 8941. https://doi.org/10.1038/ncomms9941
[27] Gholamreza Nikandish, Behnam Sedighi, and Mehrdad Sharif Bakhtiar. 2007. Performance comparison of switched-capacitor and switched-current pipeline ADCs. In *2007 IEEE International Symposium on Circuits and Systems*. IEEE, 2252–2255.
[28] Knot Pipatsrisawat and Adnan Darwiche. 2007. Rsat 2.0: Sat solver description. *SAT competition* 7 (2007).
[29] A Seitz, M Alava, and P Orponen. 2005. Focused local search for random 3-satisfiability. *J. of Statistical Mechanics: Theory and Experiment* (2005), P06006.
[30] Bart Selman, Henry A. Kautz, and Bram Cohen. 1996. Local Search Strategies for Satisfiability Testing. In *DIMACS Series*. 521–532.
[31] HT Siegelmann. 1995. Computation beyond the Turing limit. *Science* 268 (1995), 545–548. https://doi.org/{10.1126/science.268.5210.545}
[32] Hava Siegelmann. 2012. *Neural networks and analog computation: beyond the Turing limit*. Springer Science & Business Media.
[33] João P Marques Silva and Karem A Sakallah. 1997. GRASPâĂŤa new search algorithm for satisfiability. In *Proceedings of the 1996 IEEE/ACM international conference on Computer-aided design*. IEEE Computer Society, 220–227.
[34] Renée St Amant, Amir Yazdanbakhsh, Jongse Park, Bradley Thwaites, Hadi Esmaeilzadeh, Arjang Hassibi, Luis Ceze, and Doug Burger. 2014. General-purpose code acceleration with limited-precision analog computation. *ACM SIGARCH Computer Architecture News* 42, 3 (2014), 505–516.
[35] R Sumi, B Molnár, and M Ercsey-Ravasz. 2014. Robust optimization with transiently chaotic dynamical systems. *EPL (Europhysics Letters)* 106, 4 (2014), 40002.
[36] Jason Thong and Nicola Nicolici. 2013. FPGA acceleration of enhanced boolean constraint propagation for SAT solvers. In *Proceedings of the International Conference on Computer-Aided Design*. IEEE Press, 234–241.
[37] M. M. Waldrop. 2016. More than Moore. *Nature* 530 (February 2016), 144–147.
[38] Xunzhao Yin, Behnam Sedighi, Melinda Varga, Maria Ercsey-Ravasz, Zoltan Toroczkai, and Xiaobo Sharon Hu. 2016. Efficient Analog Circuits for Boolean Satisfiability. *arXiv preprint arXiv:1606.07467 (accepted by IEEE Transactions on VLSI)* (2016).
[39] Lintao Zhang, Conor F Madigan, Matthew H Moskewicz, and Sharad Malik. 2001. Efficient conflict driven learning in a boolean satisfiability solver. In *Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design*. IEEE Press, 279–285.