29.2 Snap-SAT: A One-Shot Energy-Performance-Aware All-Digital Compute-in-Memory Solver for Large-Scale Hard Boolean Satisfiability Problems

Shanshan Xie, Mengtian Yang, S. Andrew Lanham, Yipeng Wang, Meizhi Wang, Sirish Oruganti, Jaydeep P. Kulkarni

University of Texas, Austin, TX

Boolean satisfiability (SAT) is a non-deterministic polynomial time (NP)-complete problem with many practical and industrial data-intensive applications [1]. Examples (Fig. 29.2.1) include anti-aircraft mission planning in defense, gene prediction in vaccine development, network routing in the data center, automatic test pattern generation in electronic design automation (EDA), and model checking in software. The objective of a SAT solver is to identify the values of n Boolean variables x_i that satisfy all clauses in a conjunctive normal form (CNF) [5]. However, the time required to determine the satisfiability of a SAT problem increases exponentially with respect to the variable size, which is energy and resource-consuming. A prior software SAT solver [3] requires frequent data transfer and memory access due to the CPU computations, solution-search, and repetitive variable updates, increasing the computational latency and energy cost. Another approach to designing a SAT solver is to leverage a continuous-time dynamical system using analog circuitry [5]. However, such dedicated analog arithmetic components incur a large area and energy overhead as they cannot be reused during non-SAT applications. Moreover, the analog SAT computations necessitate frequent SRAM read/write access which increase hardware implementation costs. Therefore, there is a critical need for advancing energy and area-efficient hardware SAT solver designs.

Since the SAT variables are initially stored in the memory arrays, a compute-in-memory (CIM) approach is naturally suitable for solving SAT problems as it can utilize local write-backs in memory arrays for variable updates. In contrast to the previous SAT solvers, we present an all-digital CIM SAT solver that accelerates iterative computations utilizing the existing static random-access memory (SRAM) arrays to significantly minimize off-chip memory accesses and the corresponding energy costs. The key attributes of the Snap-SAT approach are: 1) massively parallel in-memory local computations in a one-shot manner and support for local variable update to minimize data movement; 2) reliable computations under process and temperature variations, which can be seamlessly scaled to advanced CMOS technologies due to the all-digital design approach; 3) SRAM bitcells that can be reused for regular mode operation in non-SAT applications to reduce the area overhead and hardware implementation complexity; 4) scalability to large-scale hard SAT problems with different variable/clause sizes and user-defined solver algorithms; 5) no data converter circuit and intra-data movement are needed for the SAT computations; 6) 65nm CMOS prototype measurements demonstrating 7.5-to-151× (12-to-181×) speedup and $7*10^4$× ($1.3*10^4$×) energy improvement over a software SAT solver using a Xeon W-2195 CPU (Snapdragon 845 ARM) processor.

Figure 29.2.2 shows the mapping between the SRAM array and the k-SAT CNF $F(x)$. An M clause CNF is expressed as $F(x)=\wedge C_m$. Each clause C_m is a disjunction (OR) between k literals (e.g., $C_0=x_0\ OR\ x_1\_bar\ OR\ x_5$) and a literal is a Boolean variable x_i or its negation x_i_bar. Each clause is mapped to a column, and the configurations of each variable are stored in two 6T SRAM bitcells. The first bitcell ($P_{m,i}$) stores the information regarding whether x_i or x_i_bar is present in the clause m. The second bitcell ($D_{m,i}$) holds the literal data associated with variable i inside clause m, which could be either x_i or x_i_bar (e.g., $D=x_i$ means variable i in the clause is in true form and $D=x_i\_bar$ means variable i is in its complemented form). If the variable is absent in clause m, the literal data $D_{m,i}$ is set to the variable itself (x_i) by default. Consequently, with this mapping, a variable update can be achieved locally by flipping the entire data row, as each data row represents a single variable. An example is illustrated in Fig. 29.2.2, where $C_1=x_1\ OR\ x_3\_bar\ OR\ x_4\_bar$. In this case, $P_{1,3}$ is 1, and $D_{1,3}$ is x_3_bar. On the other hand, for variable x_5, $P_{1,5}$ is 0 and $D_{1,5}$ is a Don't Care condition because neither x_5 nor x_5_bar is in clause 1. Additionally, each of the two 6T SRAM bitcells is paired with a 3-Transistor (3T) NAND gate. The output of a NAND column (CL) is the result of OR operations in clause C_m, where CL=1 indicates that C_m is False (unsatisfied) and CL=0 interprets that C_m is True (satisfied).

Figure 29.2.3 shows the circuit details of the Snap-SAT design. The SRAM controller, 6T SRAM bitcells, precharge unit, and read/write driver are the same as the baseline SRAM array design. In addition to the baseline array circuits, NAND gates, local update, parallel counter, and a Snap-SAT controller are added to the memory array to support the SAT computations. $P_{m,i}$ and $D_{m,i}$ are stored in the 6T SRAM bitcells, and the storage nodes of the two bitcells are connected to two transistor gates in the NAND unit. The middle input of the NAND is connected to the compute control signal (CP), which enables/disables the SAT computation. The input order (P, CP, D) is designed to ensure that the control signal with the highest priority ($P_{m,i}$) controls the first transistor gate to prevent discharge current from the variables that are not present in the clause. In the computation mode, CP<n-1:0> are asserted, and all the clause computations are achieved in parallel in a single cycle on the compute line (CL<M-1:0>), shared along the column. This demonstrates the massive parallel computation capability of the Snap-SAT design directly on the memory bitcells. Fig. 29.2.3 illustrates a computation flow example for a 3-SAT problem. Initially, CL is precharged to V_{DD} by turning on the CL_PC signal. The compute signals (CP<n-1:0>) enable the NAND gates during the clause computation. After CPs are ON, CL<9> is discharged to zero, assuming only Clause 10 is satisfied after the first round of computation. Next, after the results appear on the CLs, the counter-enable signal (Counter_EN) in the controller is asserted to latch the computation results on CLs and to start the parallel counter. Because only 1 clause is fulfilled, the counter output (unSAT) is $M-1$, where M is the total number of clauses. After receiving the counter result, the controller randomly selects one of the variables in a random unfulfilled clause, and the random selection is achieved using a linear feedback shift register. Note that the variable and clause selection steps vary according to different algorithms. After the variable is selected, a local write-after-read operation is performed, and the data from the read driver is inverted to flip the selected variable (e.g., x_0), and written back using the local write driver, thus eliminating the data movement, compared with a conventional SAT solver [3].

Another critical aspect is the flexibility to reconfigure various algorithms. As shown in Fig. 29.2.4, MiniSAT (MS) [3] is a better-known complete algorithm for software SAT solvers, but from a hardware perspective, WalkSAT (WS) and Schoning's (SC) algorithms are more compatible with the CIM approach because of their iterative search nature. Hence, they are employed as the primary Snap-SAT algorithms. For benchmarking, random k-SAT formulas are generated at a fixed clause-to-variable ratio (CTV) since the CTV ratio determines the hardness of a SAT problem. The CTV ratio is set to 4.3 for a 3-SAT problem, targeting a hard problem regime. For an n-variable problem, there are $2^kC(n,k)$ unique clauses that can appear in a CNF. M clauses are sampled uniformly by randomly choosing k variables out of n without repetition and converting variables to literals by either negating them or leaving them un-negated with equal probability. The computation evolution using WS in the Snap-SAT design shows a 10.5× speedup over the Xeon W-2195 CPU software solver on a hard 3-SAT problem.

Solution time and energy consumption of the Snap-SAT design are evaluated in Fig. 29.2.5, where the flowchart shows that the solution time is quantified as the time interval from the start of the computation until all clauses are satisfied, while the benchmark loading and parameter setup phases are excluded for both test-chip and software processors. Using WalkSAT, this work shows a 12× (7.5×) speedup compared to the solution time of the ARM (CPU) processor using the state-of-the-art algorithm, MiniSAT. Furthermore, compared to the ARM (CPU) solver, the design reduces the energy consumption by $1.3*10^4$× ($7*10^4$×), showing promising energy and speed improvements over the software-based approaches using well-known algorithms. A detailed design comparison of prior SAT solvers in CMOS processes [4-6] is presented in Fig. 29.2.6. The test-chip summary, experiment setup, die micrograph, and area/power breakdown are shown in Fig. 29.2.7. In summary, the Snap-SAT design achieves at least an order of magnitude energy/speed improvement over the software approaches, and has been extensively tested on different hard k-SAT problems with different variable sizes, clause sizes, and CTV ratios. This demonstrates a promising, fast, reliable, reconfigurable, and scalable compute-in-memory design for solving and accelerating large-scale hard SAT problems, suggesting its potential for solving time-critical SAT problems in real-life applications (e.g., defense, vaccine development, etc.).

Acknowledgement:
This research is supported in parts by NSF CAREER Award, Micron Foundation faculty grant and AMD Chair Endowment. Testchip fabrication is supported by TSMC University Shuttle Program.

References:
[1] J. Marques-Silva, "Practical Applications of Boolean Satisfiability," *Workshop on Discrete Event Systems*, pp. 74-80, 2008.
[2] A. Montanari et al., "Clusters of Solutions and Replica Symmetry Breaking in Random k-Satisfiability," *Jour. Statistical Mechanics*, P04004, 2008.
[3] N. Eén et al., "An Extensible SAT-Solver," *Conf. on Theory and Applications of Satisfiability Testing*, pp. 502-518, 2003
[4] Y. Su et al., "FlexSpin: A Scalable CMOS Ising Machine with 256 Flexible Spin Processing Elements for Solving Complex Combinatorial Optimization Problems," *ISSCC*, pp. 272-273, 2022.
[5] M. Chang et al., "An Analog Clock-free Compute Fabric base on Continuous-Time Dynamical System for Solving Combinatorial Optimization Problems," *IEEE CICC*, 2022.
[6] H. Mostafa et al., "An Event-Based Architecture for Solving Constraint Satisfaction Problems," *Nature Communications*, pp. 1-10, 2015.

 978-1-6654-9016-0/23/$31.00

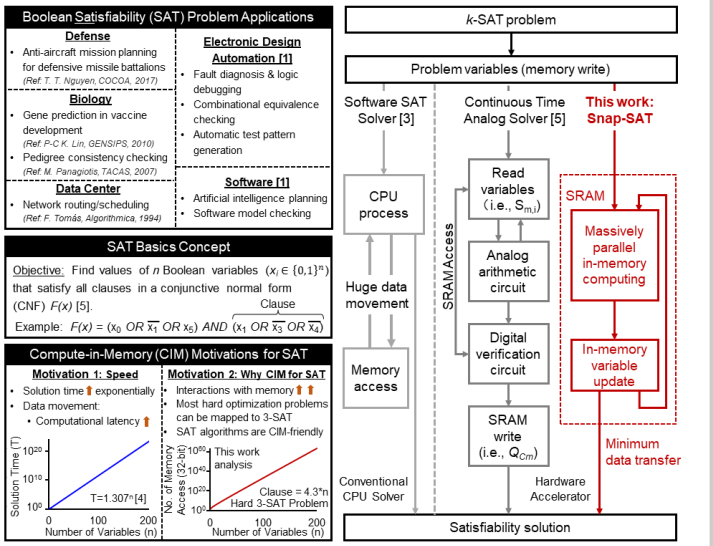


Figure 29.2.1: Boolean satisfiability (SAT) problem applications and basic concepts along with Snap-SAT design motivations and high-level comparisons with prior software and analog SAT solver.

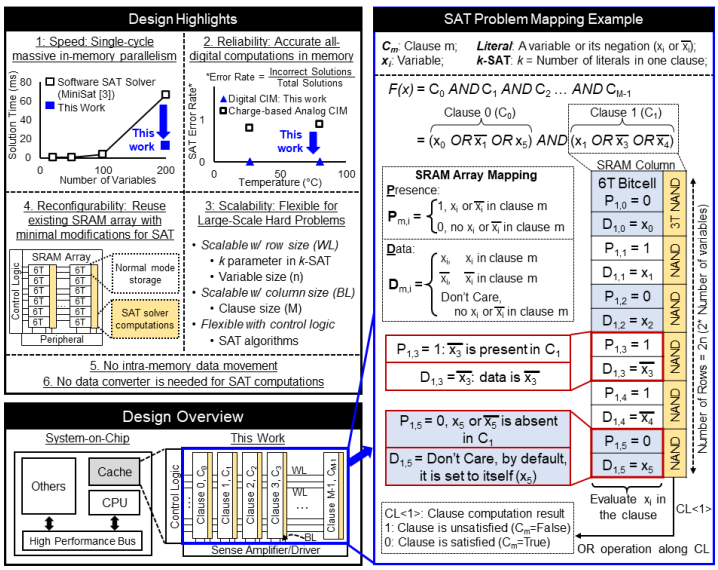


Figure 29.2.2: Snap-SAT design highlights, design overview, and mapping between the conjunctive normal form (CNF) *F(x)* and 6T SRAM column.

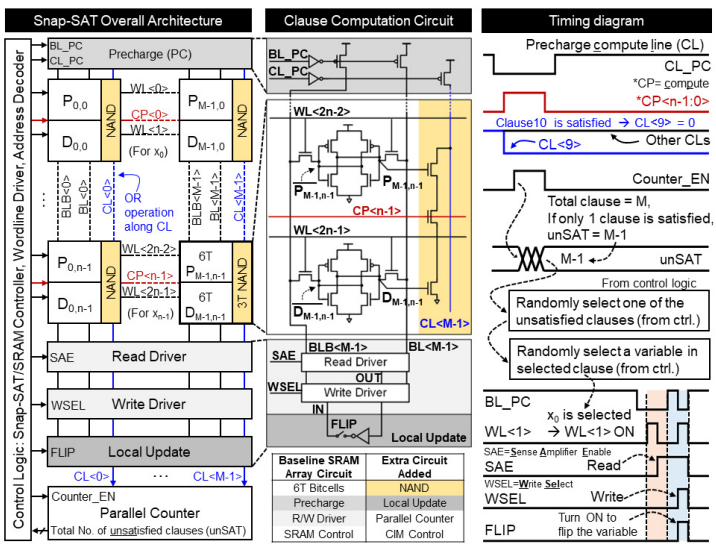


Figure 29.2.3: Overall Snap-SAT architecture with circuit diagram of clause computation column, and an example timing diagram from precharge, compute, to local update operations when only one clause is satisfied.

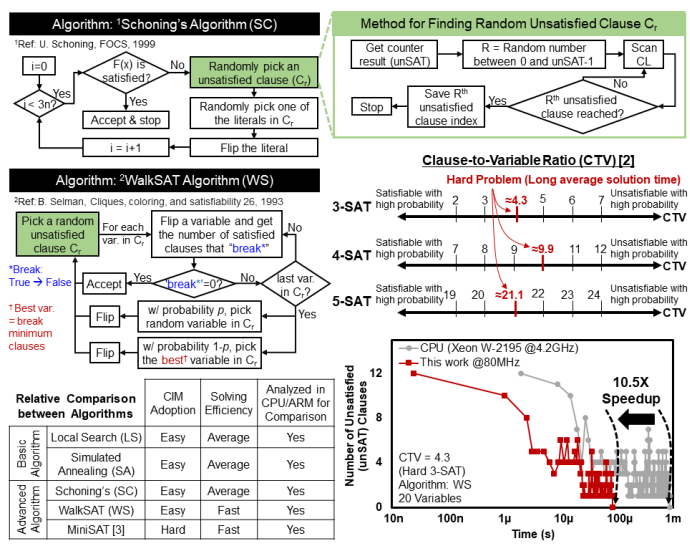
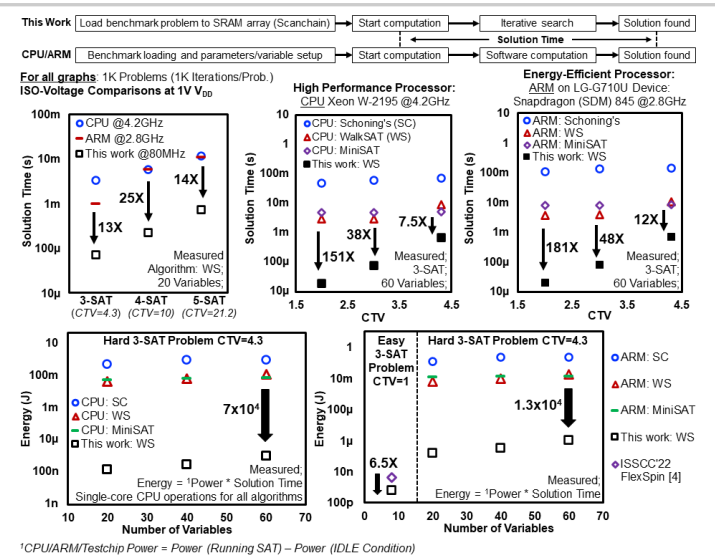


Figure 29.2.4: Schoning's (SC) and WalkSAT (WS) algorithm flowchart; well-known SAT algorithms relative comparison; clause-to-variable ratio for 3-SAT to 5-SAT; and computation evolution graph of Snap-SAT and Xeon W-2195 CPU.



[1]CPU/ARM/Testchip Power = Power (Running SAT) – Power (IDLE Condition)

Figure 29.2.5: Measured solution time using 1K 3-SAT to 5-SAT hard problems with different algorithms and for energy at V_{DD}=1V (@80MHz) with high-performance CPU processor and energy-efficient ARM processor comparisons.

		This work Snap-SAT	ISSCC'22 Flex-Spin [4]	CICC'22 [5]	Nature'15 [6]
Technology		65nm CMOS	65nm CMOS	65nm CMOS	180nm CMOS
Testchip Prototype		Yes	Yes	Yes	Yes
Compute Type		Digital	Digital	Analog	Digital & Analog
Computation Mechanism		Column-wise In-memory OR Operation	Dedicated Processing Element	Continuous-Time Dynamic	Oscillator Pulse
In-Memory Computation		Yes	Yes	No	No
Frequent SRAM Access		Not Required	Not Required	Required	Required
Hardness of Problem Tested		Hard	Easy	Hard	Hard
Maximum Number of Variables (Var)		128 (Scalable w/ row size)	8	50	50 (Test) 200 (2)
Maximum Number of Clauses		1024 (Scalable w/ column size)	8	212	218 (Test) 1260 (2)
Accuracy		100%	(Not reported)	93.5%	(Not reported)
Average Solution Time (3-SAT)	CTV=4.3 (Hard)	0.71ms (Variables=60)	(Not reported)	0.9ms (Variables=10)	7.5ms (Variables=50)
	CTV=1 (Easy)	0.41µs (Variables=8)	5µs (Variables=8)	0.15µs (Variables=10)	(Not reported)
[1]Energy (nJ) (V_{DD}=1V)	CTV=4.3 (Hard)	1098	(Not reported)	[3]21600	(Not reported)
	CTV=1 (Easy)	0.63	4.1	[3]3.6	(Not reported)

[1]Energy = Measured Average Solution Time * Measured Average Computation Power
[2]Reported without source of data (simulation or measurement)
[3]Reported peak power = 27mW (1.1V)\; Energy number is scaled to 1V

Figure 29.2.6: Snap-SAT performance comparison with prior analog and digital SAT solvers using CMOS processes.

Design Details	
Technology	65nm CMOS
Supply Voltage	0.7V~1.2V
Memory Capacity	131Kb
Chip Area	0.93mm^2
Frequency	80MHz
***k*-SAT Problem Parameters**	
k parameter (*k*-SAT)	2~128
Maximum Variable Size (n)	128 (Scalable w/ row size)
Maximum Clause Size (M)	1024 (Scalable w/ column size)
Measurement Setup	
Problem/Test Case	1,000
Iterations/Problem	1,000
Measurement Results Algorithm: WalkSAT; *k*-SAT = 3-SAT 60 Variables; CTV = 4.3 (Hard Problem)	
[1]Solution Time	713 µs
[2]Energy @V_{DD}=1V	1098 nJ
[3]Solvability	72%
[4]Accuracy	100% (All-digital design)

[1]End to end process time from start to solution found including scanchain in/out duration
[2]Energy = Measured Average Solution Time * Measured Average Computation Power
[3]Number of SAT problem that can be solved in a given time
[4]Correctness of the solved SAT problem

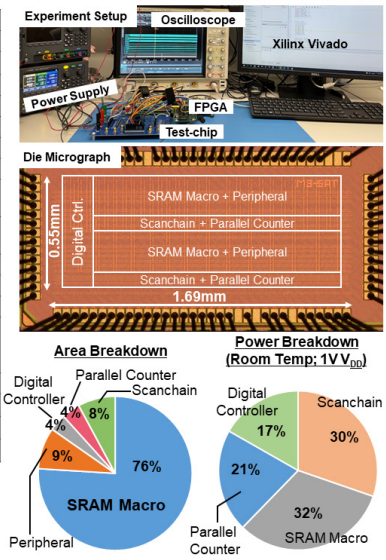


Figure 29.2.7: Test-chip summary, experiment setup, die micrograph and macro area/power breakdown.

978-1-6654-9016-0/23/$31.00