KLIMA: K-Local In-Memory Accelerator for Combinatorial Optimization

Dmitri Strukov

UC Santa Barbara

Korea-U.S. Forum on Nanotechnology (Gyeonggi-do, Korea)

July 3rd, 2025

DARPA QuICC Collaborators:

UCSB: **T. Bhattacharya**, G. Hutchinson, D. Kwon
HPE (lead): R. Beausoleil (PI), F. Bohm, M. Mohseni, G. Pedretti, T. Van Vaerenbergh
FZ Jülich (Germany): D. Dobrynin, A. Heitmann, M. Hizzani, J.P. Strachan
1QBit (Canada): I. Rozada, E. Valiante, X. Zhang

Acknowledgements:
DARPA's Quantum-Inspired Classical Computing (QuICC) program

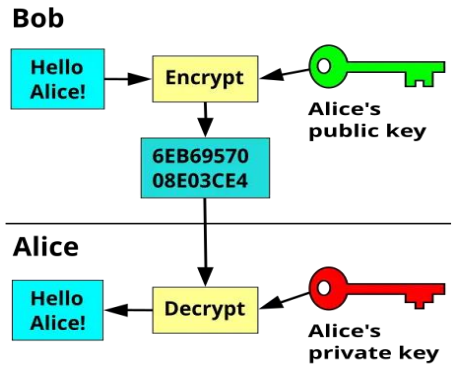
Combinatorial Optimization

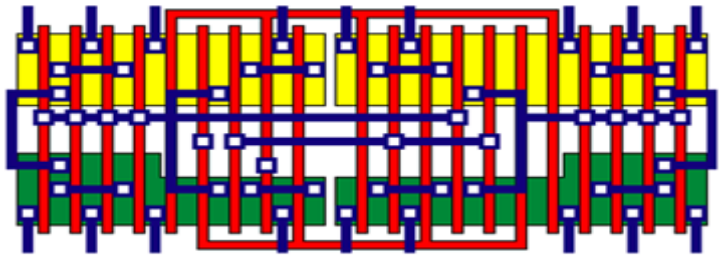
Combinatorial optimization: Find assignment of discrete variables (**x**) corresponding to optimal value, e.g., minimum, of a cost function

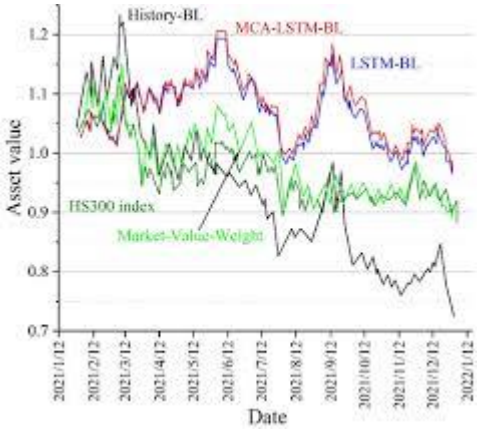
$$H(\mathbf{x}) = \sum_{i_1} J_{i_1}^{(1)} x_{i_1} + \cdots + \sum_{i_1 < \cdots < i_k} J_{i_1 \ldots i_k}^{(k)} x_{i_1} \ldots x_{i_k}$$

Many applications in industrial and scientific scenarios:

- **ML model robustness verification**
- **Test Pattern Generation**
- **Cryptography**
- **Logistics**
- **Finance**
- **Network/ Circuit Routing**
- **AI / Mission Planning**
- **Gene prediction**
- **Material design**







Major Types of Combinatorial Optimization

- Quadratic Unconstrained Binary Optimization (QUBO): Find binary vector $\mathbf{x} \in \{0,1\}$ that minimizes quadratic cost function $Q(\mathbf{x})$

$$Q(\mathbf{x}) = 1 - x_1 - x_2 - x_3 + x_1x_2 + x_1x_3 + x_2x_3 + x_1x_5$$

- Polynomial Unconstrained Binary Optimization (PUBO): Find binary vector $\mathbf{x} \in \{0,1\}$ that minimizes (multilinear) polynomial cost function $P(\mathbf{x})$

$$P(\mathbf{x}) = 1 - x_1 - x_2 - x_3 + x_1x_2 + x_1x_3 + x_2x_3 + x_1x_5 - x_1x_2x_3 - x_1x_4x_5$$

- *K*-SAT: Find assignment of Boolean variables v to satisfy CNF-type Boolean expression. Equivalent to PUBO via $v = (1 - x), \bar{v} = x, \wedge \rightarrow +, \vee \rightarrow \times$

$$\text{3-SAT} = (v_1 \vee v_2 \vee v_3) \wedge (\overline{v_1} \vee v_4 \vee \overline{v_5}) \rightarrow P(\mathbf{x}) = (1 - x_1)(1 - x_2)(1 - x_3) + x_1(1 - x_4)x_5$$

PUBO and K-SAT can be converted to equivalent QUBO by order reduction methods

Motivation for High Order Solvers

Many practical problems are described by **higher-order** cost functions....

Contemporary SOTA approaches are either QUBO solvers or dedicated 3-SAT solvers

Conversion to QUBO results in **heavy overhead**, i.e., slower convergence due to larger configuration space due auxiliary variables and new shallow minima in the energy landscape.

→ Need to solve higher-order problems using native formulation

Combinatorial Optimization Algorithms

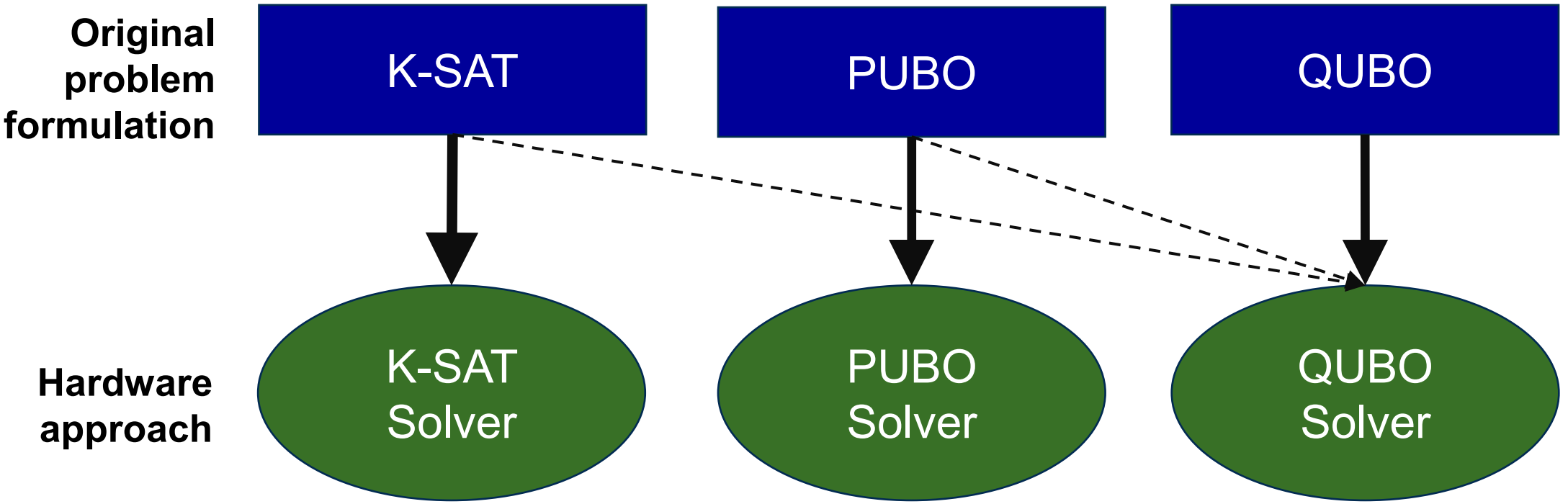
- Many Algorithms:
  - Complete algorithms, local search, ant colony, genetic,…
  - Time to global minima scales exponentially with # variables for hard problem

- Heuristic algorithms:

  - Ising Machines (and closely related Hopfield Neural Networks and Boltzmann Machines) implements gradient-descent-like heuristic algorithms
  - Efficient local search algorithms, e.g. WalkSAT, that solve SAT problems in native form and rely on gradient-descent-like heuristics

The common implementation challenge for high-order solvers is efficient hardware for computing cost function derivatives or Boolean function gains

Our Approach: Solve Problems in Native High-Order Form



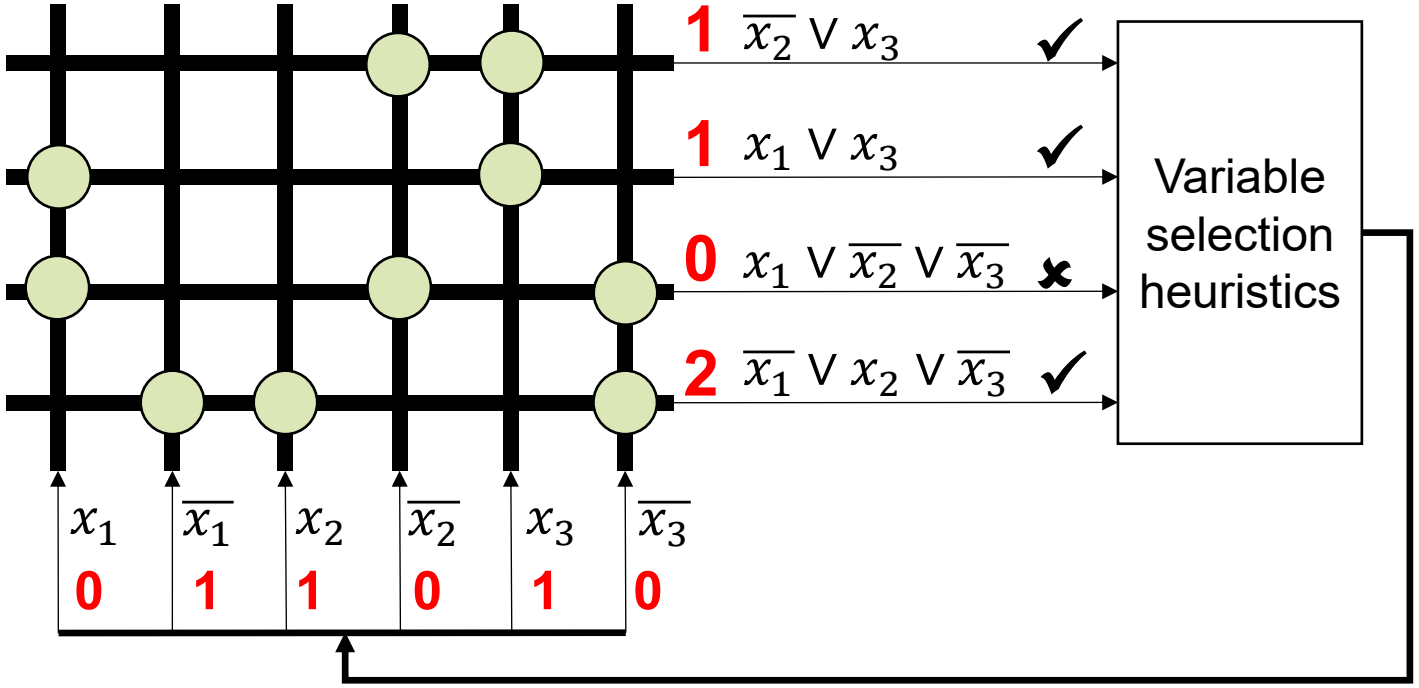
The goal is to develop arbitrary-order (any K or order) solver that preserves high-order interactions and solve problem in the native (SAT or PUBO or PUSO) space

→ KLIMA: K-Local In-Memory Accelerator

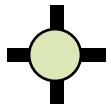
SAT Solver: Parallel Clause Evaluation with In-Memory Computing

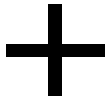
- **Main Idea:** Parallel clause evaluation with in-memory computing on dense crossbar memory circuit
- **Example:** $3\text{SAT} = (\overline{x_1} \vee x_2 \vee \overline{x_3}) \wedge (x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_1 \vee x_3) \wedge (\overline{x_2} \vee x_3)$

clause



- #clauses × #literals crossbar circuit
- Each clause of a problem is mapped to single row
- Zero sensed current indicates unsatisfied clause
- "1" unit sensed current indicates weakly satisfied clause


= high conductance


= low conductance

S. Park et al. ASP-DAC'21 29-34 (2021); G. Pedretti et al. Zeroth and high-order logic with content addressable memories. IEDM'23 (2023)

SAT Solver: Parallel Gain (Gradient) In-Memory Computation

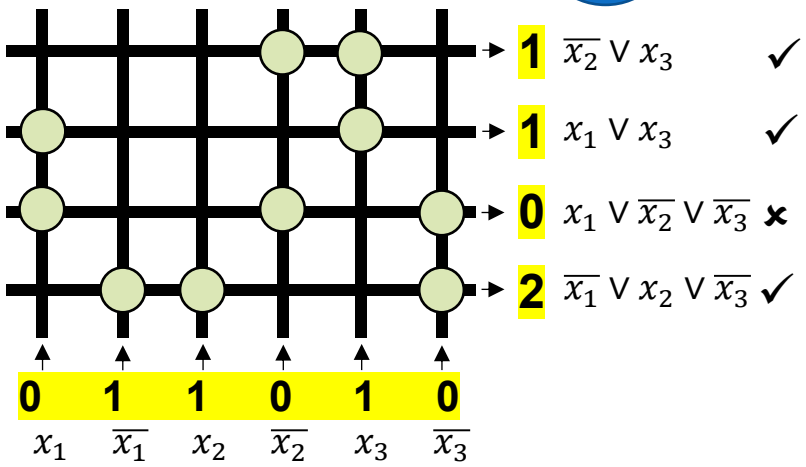
- **Gain computation**

Cost function decrease w.r.t. variable state change = Make Value – Break Value

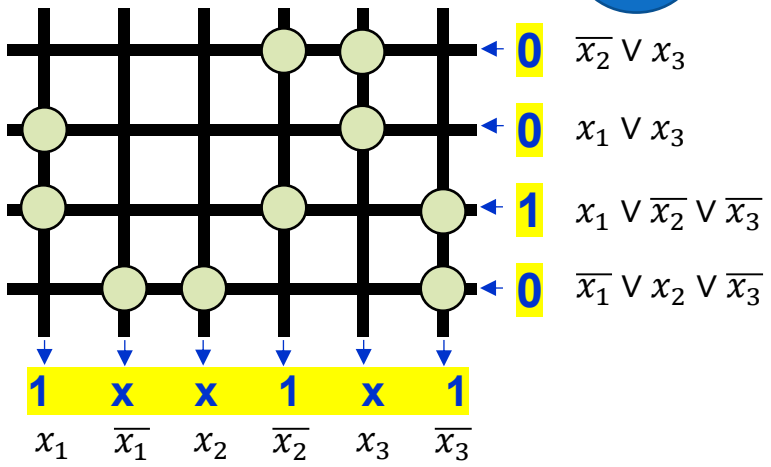
(# new clauses satisfied → Make Value; # previously satisfied clauses becoming unsatisfied → Break Value)

Key idea: Take full advantage of analog outputs and reverse signal flow to compute make and break values

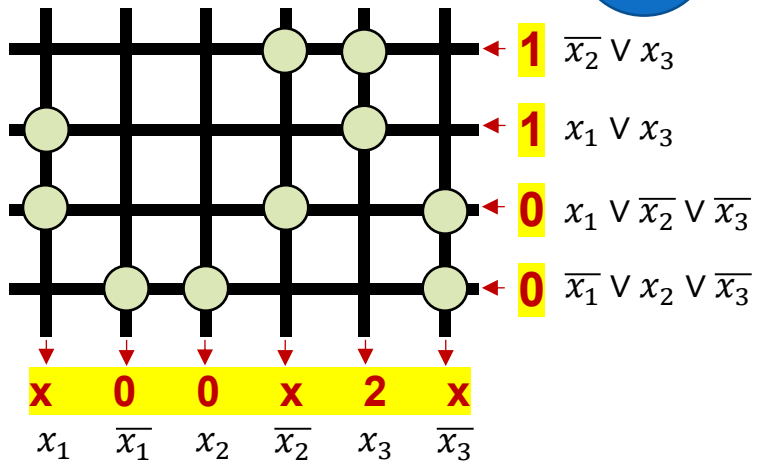
- **Clause evaluation** 1



- **Make computation** 2a



- **Break computation** 2b



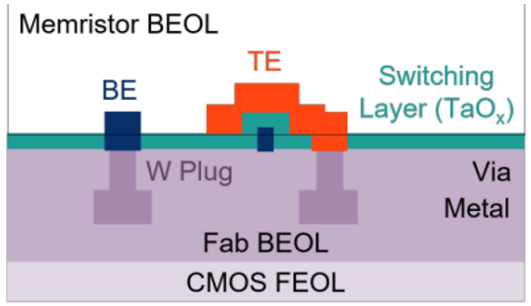
Energy decrease w.r.t. x_3 = 1 (Make Value) – 2 (Break Value) = -1

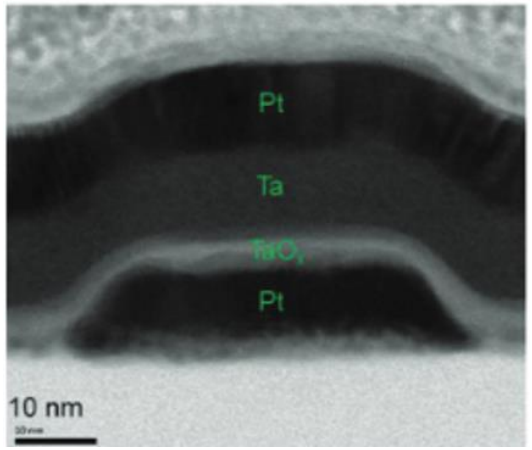
Massively parallel (in-memory) computation of gains, irrespective of K

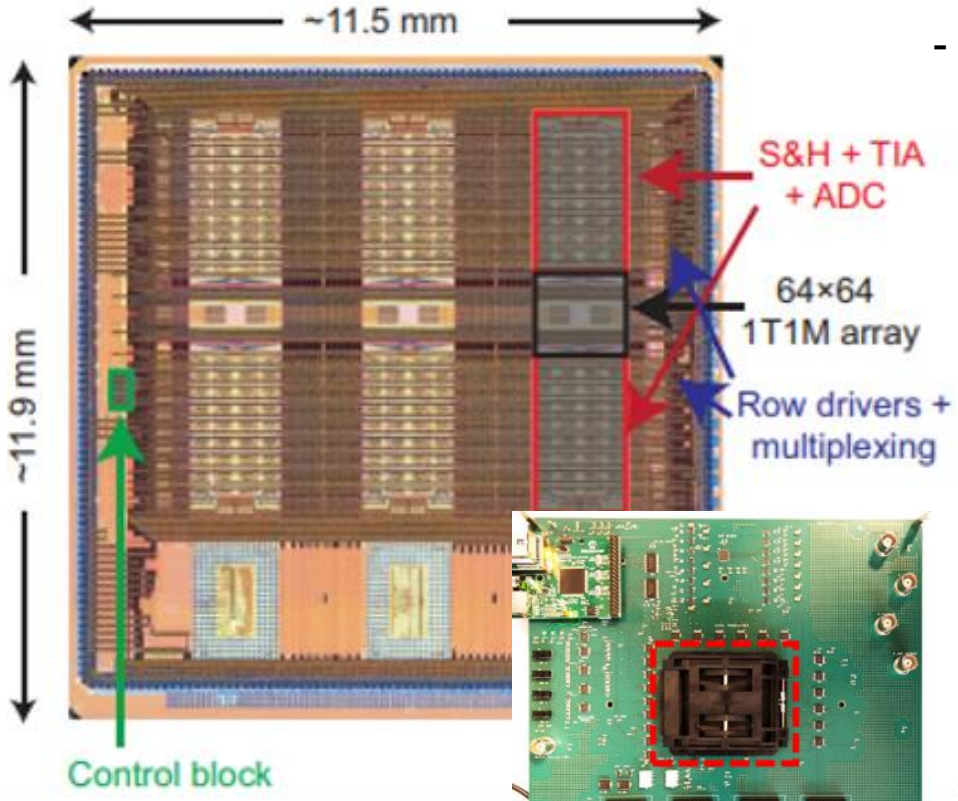
T. Bhattacharya et al., Computing High-Degree Polynomial Gradients in Memory, Nature Comm 2025

HPE's "SuperT" Memristor Prototype System

- **In-house memristors BEOL-integrated with 180nm TSMC CMOS circuits**

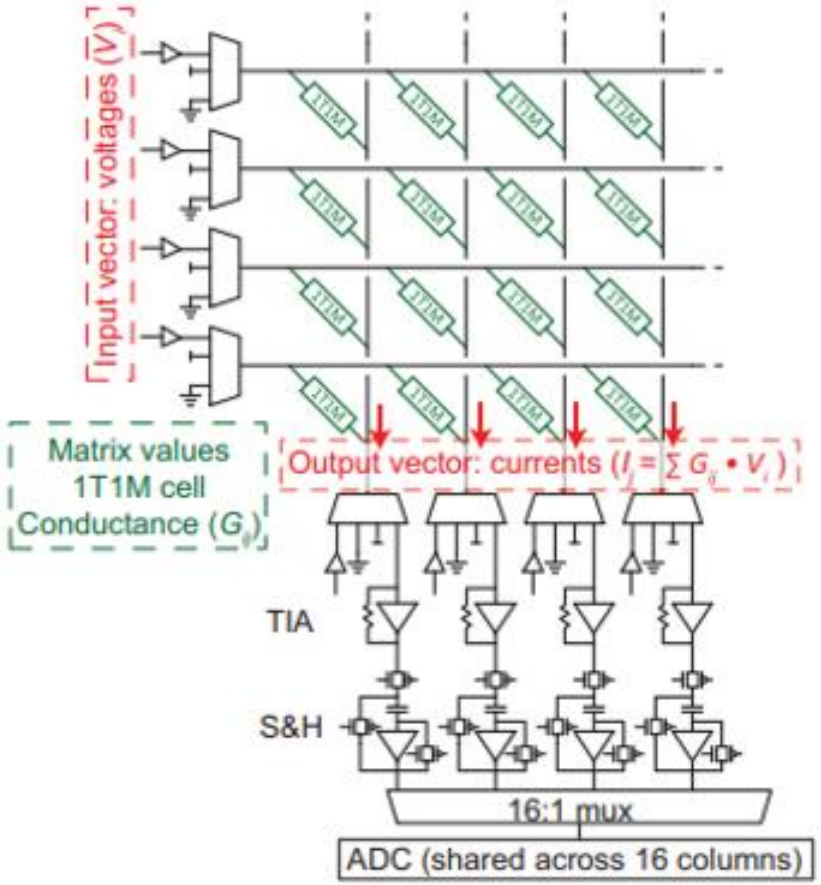




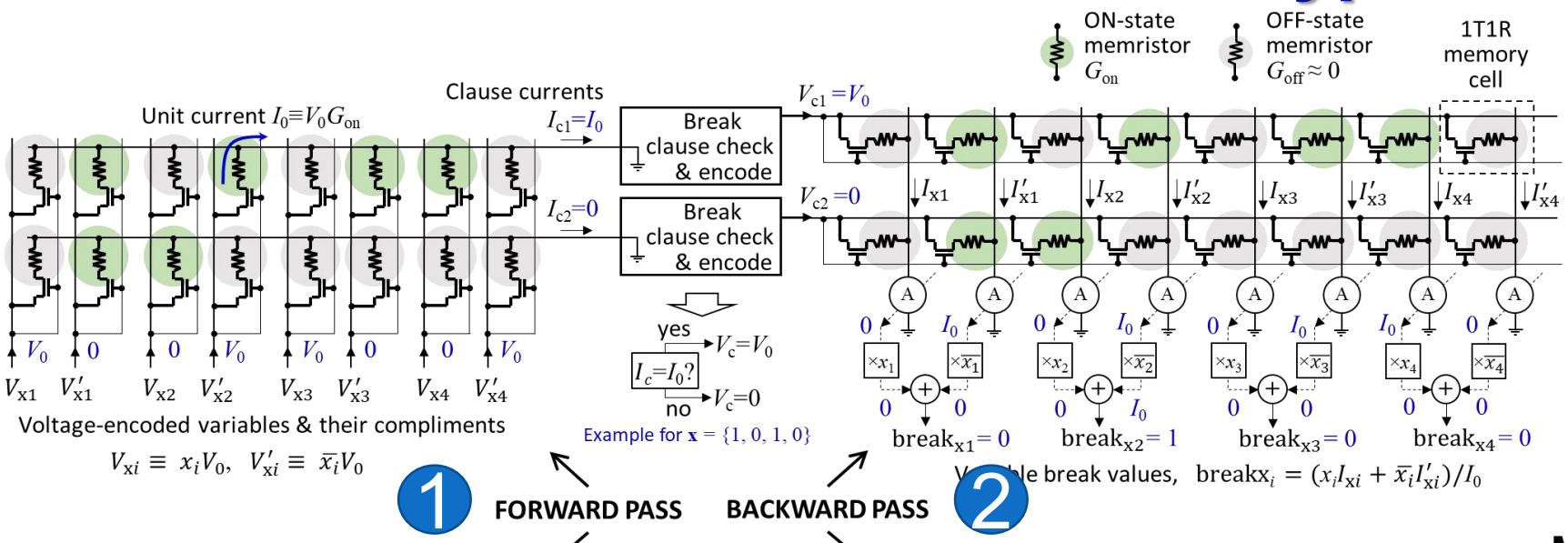


- TaOx memristor device technology
  - scalable to 25 nm, < 1ns and <fJ write time & energy
  - 10 year retention and >10^5 switching endurance
  - up to 8 bit effective bit precision

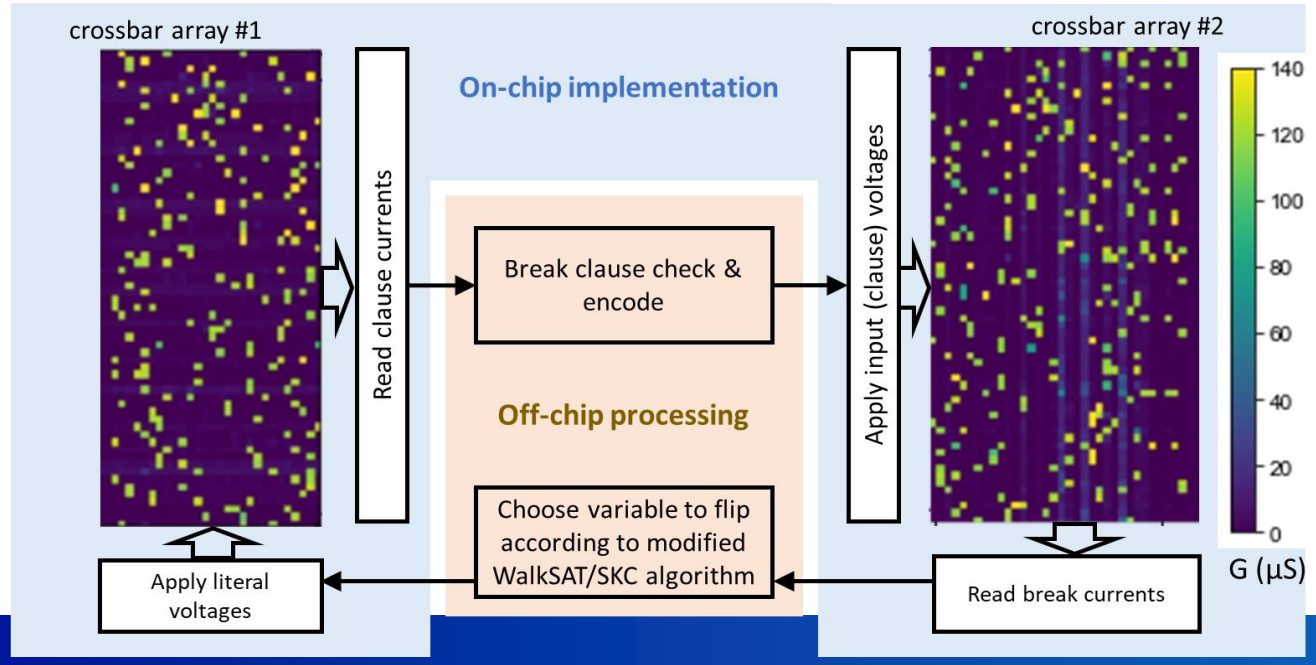
- **Crossbar arrays & CMOS periphery**
  - 2x 64x64 memristor 1T1M crossbar arrays
  - Integrated DACs/TIAs/ADCs
  - >5-bit mixed-signal VMM computation



ReRAM-Based Accelerator Prototype for K-SAT Problems

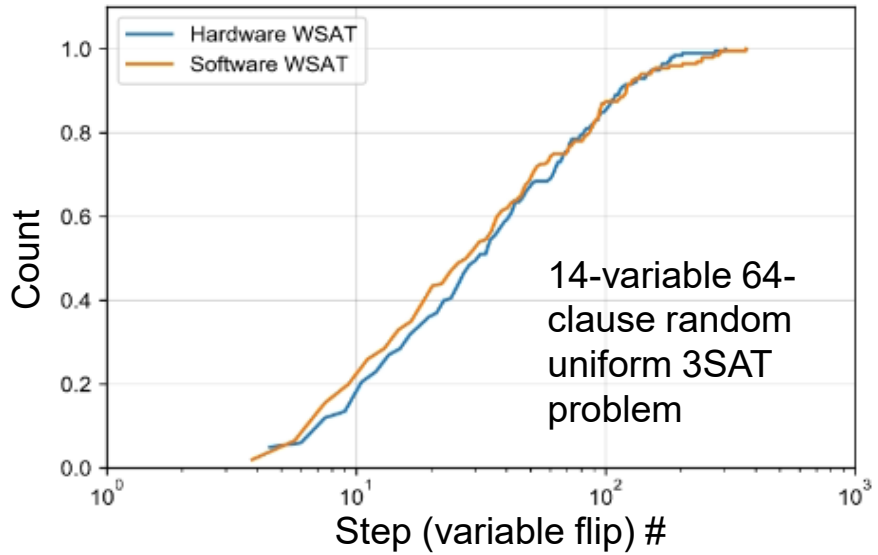


1 FORWARD PASS BACKWARD PASS 2



- WalkSAT/SKC algorithm
- In-memory computing crossbar operation are implemented experimentally, while the remaining functions are emulated on PC

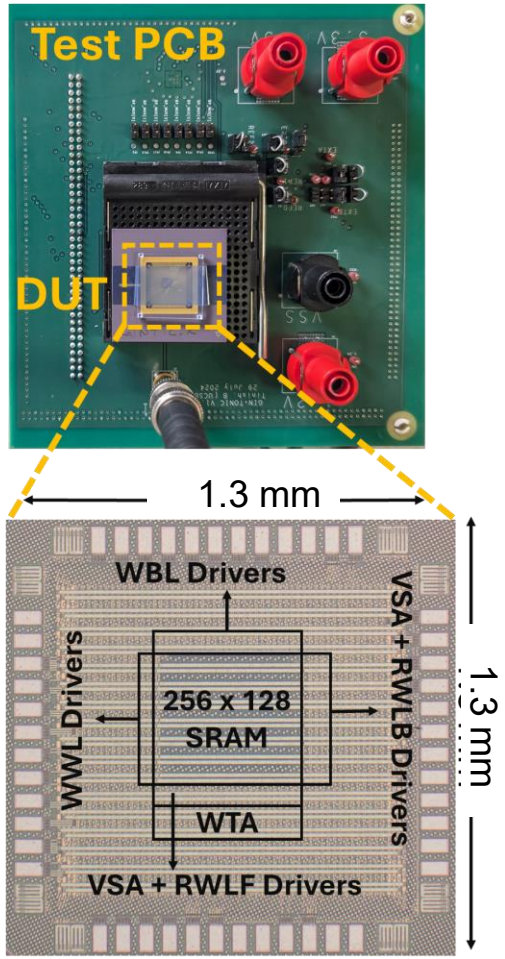
- **Main results: Run-length distribution**



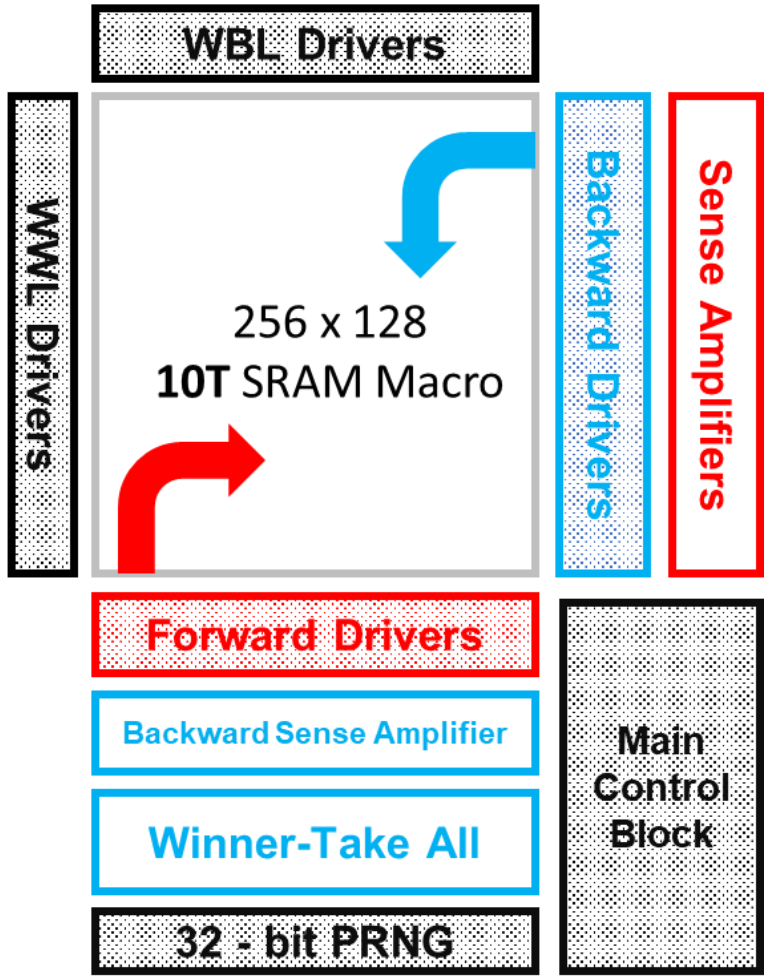
T. Bhattacharya et al, Nature Comm 2024

SRAM-Based Accelerator Prototype for K-SAT Problems

- **Setup & Chip Micrograph**



- **Top level diagram**



(shaded boxes = digital, rest analog)

- **Custom** 256x128 10T bi-directional SRAM array with simultaneous clause evaluation and make/break/gradient computation
- **Analog input Winner-Take-All** (WTA) circuit for selecting variable with minimum break-value, resulting in **ADC-free operations**
- **Two operation modes:**
  - Integrated to run WalkSAT/SKC on any SAT instance in real time
  - Hybrid to run arbitrary heuristic using off-chip ADCs to digitize gradient information and host PC to synchronize control sequence

T. Bhattacharya et al, VLSISymp'25

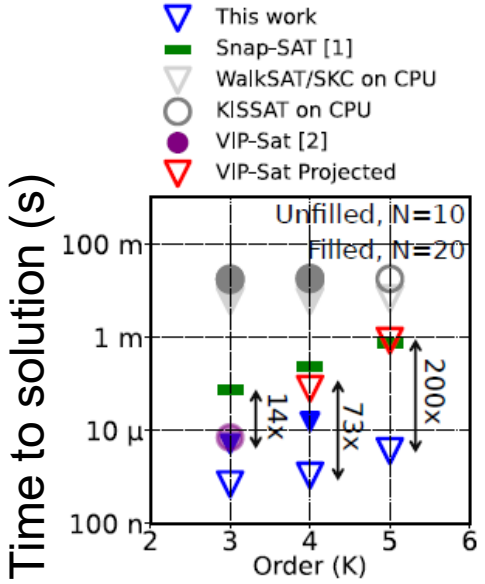
Experimental Results on SATLIB Problems

Specifications	Stochastic SAT [1]	VIP-Sat [2]	Sci Rep 2024 [3]	Snap-SAT [4]	KLIMA (This work)
Technology	65nm	65nm	65 nm	65nm	55nm
Type	Analog	Digital	Digital	Digital	Mixed
Architecture	Stochastic CT	Near-Memory	Ring Oscillator	SRAM In/Near-Memory	SRAM In-Memory
Flips / iteration	Multiple	Multiple	Multiple	Single	Single
Chip Area (mm^2)	0.37	1.115	1.8	0.93	0.544
Maximum order	3	3	2	2 - 128	64
# Variables (N)	20	50	20	128	64
# Clauses (M)	91	218	91	1024	256
Solvability	100%‡	100%‡ , 98%ǁ	100%‡	72%◊	100%‡, 98%ǁ
Solution Time	6.6 us‡	7 us‡, 18.7 usǁ	15.7 ms‡	70 us‡, 710 us◊	5.12 us‡, 45 usǁ
Solution Energy	11 nJ‡	20.8 nJǁ	0.15 mJ‡	1098 nJ◊	59 nJ‡, 518 nJǁ

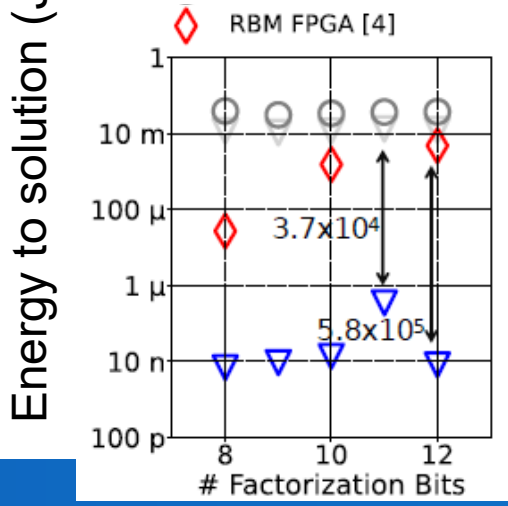
‡N=20, ǁN=50, ◊N=60, where N is # of variables.

T. Bhattacharya et al, VLSISymp'25 & HotChips'25

Random uniform



Semiprime factoring



Quadratic Ising Machine Implementation for QUBO

❑ **Typical system of ODEs and ...**

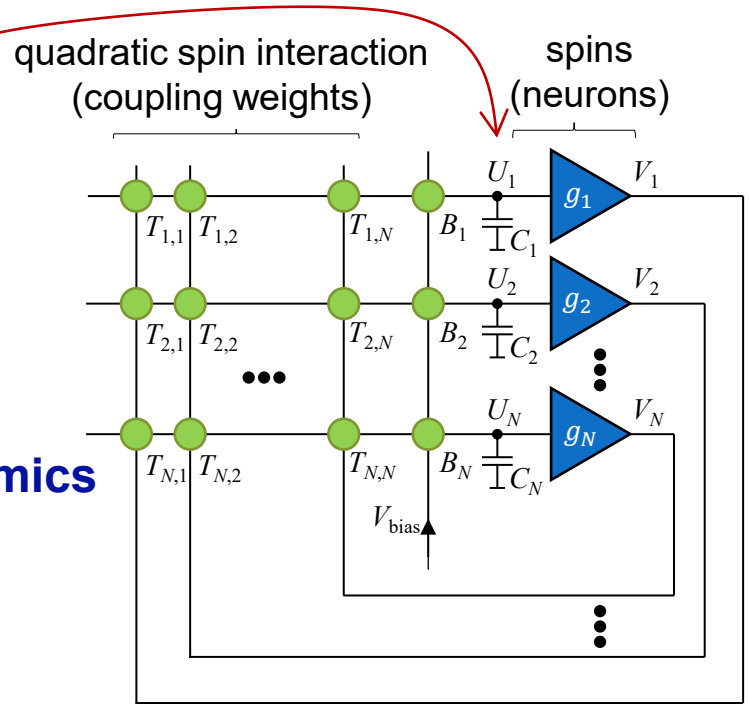
$$C_i \frac{\mathrm{d}U_i}{\mathrm{d}t} = \sum_j^N T_{ij} V_j + V_{\mathrm{bias}} B_i - U_i G_i,$$

where $U_i = g_i^{-1}(V_i)$ and $G_i = \sum_j^N T_{ij} + B_i$

$$\propto \frac{\partial E}{\partial V_i}$$

❑ **... energy (cost) function governing dynamics**

$$E = -\frac{1}{2}\sum_i^N \sum_j^N T_{ij} V_i V_j + V_{\mathrm{bias}} B_i V_i$$



- Network seeks minima of quadratic E by following gradient descent-like dynamics
- **Pre-activation is a partial derivative of energy function** with respect to mapped variable

Coupling weights are the most critical component, dot product is the most common operation

Our approach: electronic (mixed-signal, in-memory-computing) implementation with

- **Dense** coupling weights: eFlash, metal-oxide memristors or SRAM for binary weights
- CMOS for the remaining functions (less numerous spins etc.)

How to design higher-order Ising machines to solve PUBO natively?
→ requires efficient hardware for computing higher order derivatives

High-Order Ising Machines

- **PUBO energy function example**

monomial coefficient

unit-coefficient monomial

monomial

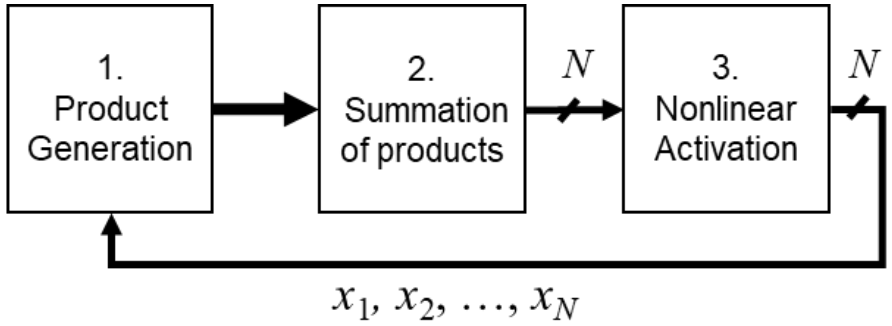
$$E = a_1 x_1 x_2 + a_2 x_1 x_3 + a_3 x_1 x_2 x_3 + a_4 x_1 x_2 x_4$$

- **Energy partial derivatives assuming real-valued (continuous IMs) x:**

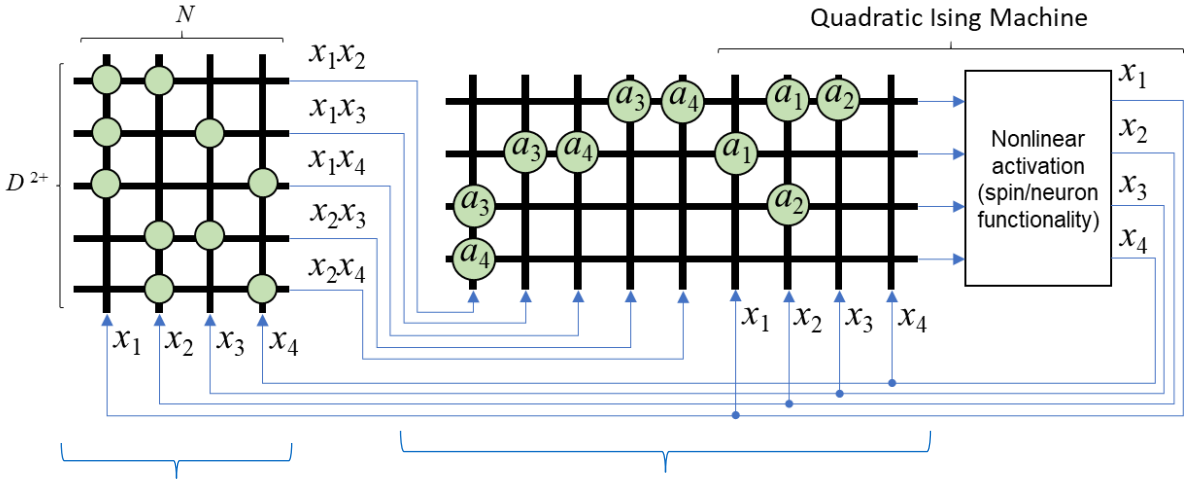
$$\frac{\partial E}{\partial x_1} = a_1 x_2 + a_2 x_3 + a_3 x_2 x_3 + a_4 x_2 x_4$$
$$\frac{\partial E}{\partial x_2} = a_1 x_1 + a_3 x_1 x_3 + a_4 x_1 x_4$$
$$\frac{\partial E}{\partial x_3} = a_2 x_1 + a_3 x_1 x_2$$
$$\frac{\partial E}{\partial x_4} = a_4 x_1 x_2$$

derivative monomial

- **High-level implementation**



- **Derivative-type high-order Ising machine**



(derivative monomials) product generation

summation of (derivative product) products

T. Bhattacharya et al., "Unified framework for efficient high-order Ising machine hardware implementations" (2024)

Product Generation with Binary Variables

- **PUBO energy function example**

monomial coefficient

unit-coefficient monomial

monomial

$$E = a_1 x_1 x_2 + a_2 x_1 x_3 + a_3 x_1 x_2 x_3 + a_4 x_1 x_2 x_4$$

- **Differentials**

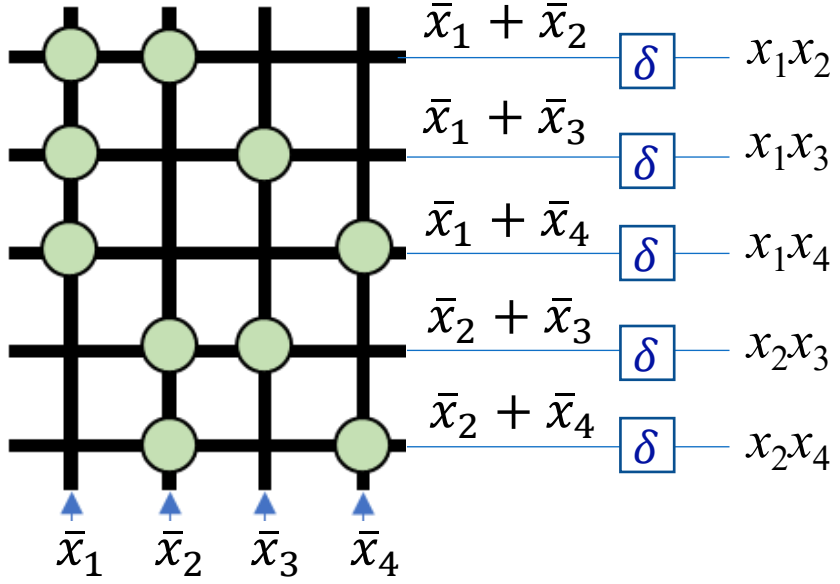
$$\Delta E_{x1} = (a_1 \delta[\bar{x}_2] + a_2 \delta[\bar{x}_3] + a_3 \delta[\bar{x}_2 + \bar{x}_3] + a_4 \delta[\bar{x}_2 + \bar{x}_4]) \times (2x_1 - 1)$$
$$\Delta E_{x2} = (a_1 \delta[\bar{x}_1] + + a_3 \delta[\bar{x}_1 + \bar{x}_3] + a_4 \delta[\bar{x}_1 + \bar{x}_4]) \times (2x_2 - 1)$$
$$\Delta E_{x3} = (a_2 \delta[\bar{x}_1] + a_3 \delta[\bar{x}_1 + \bar{x}_2]) \times (2x_3 - 1)$$
$$\Delta E_{x4} = (a_4 \delta[\bar{x}_1 + \bar{x}_2]) \times (2x_4 - 1)$$

flipping direction sign

because for binary variable product

$$x_2 x_3 = \delta[\bar{x}_2 + \bar{x}_3] \text{ where Kroneker function } \delta(y) = \begin{cases} 1, y = 0 \\ 0, \text{otherwise} \end{cases}$$

- **Product implementation**

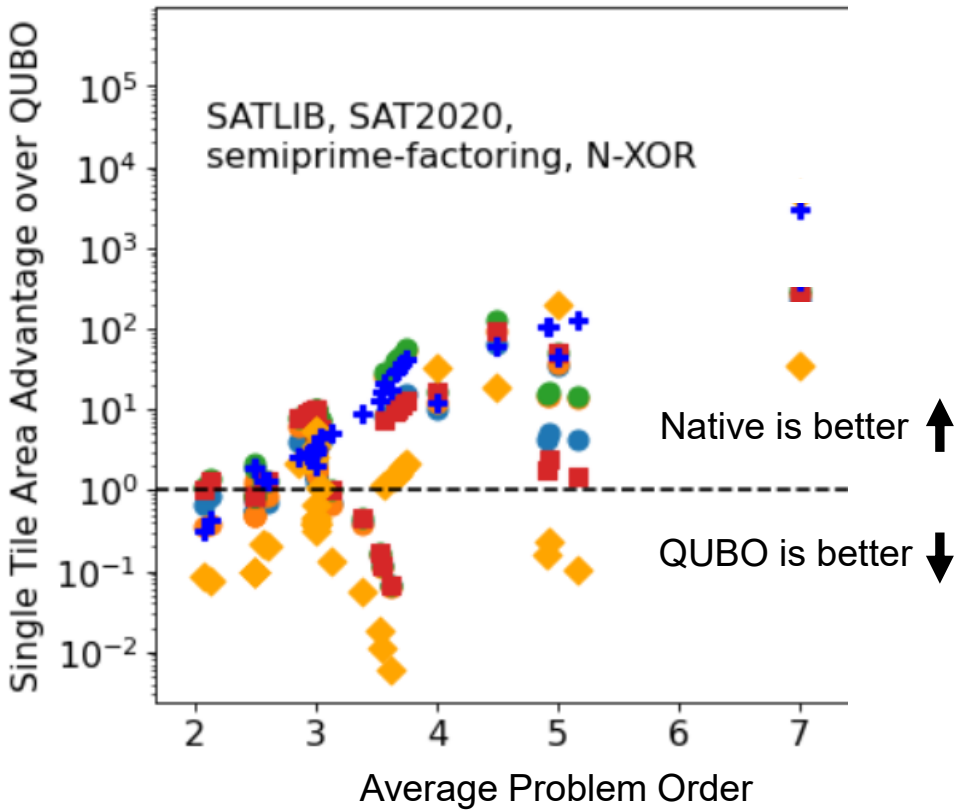


- Circuit (crossbar) complexity is independent of the problem order *K*

T. Bhattacharya et al., "Unified framework for efficient high-order Ising machine hardware implementations" (2024)

Area Advantage Modeling for Benchmark Problems

- **Density advantage over QUBO implementation**



	problem encoding	design type	operation type	required weights
●	PUBO	derivative	discrete	analog
●	PUBO	monomial	discrete	binary
●	PUBO	hybrid	discrete	analog
■	shifted PUBO	hybrid	discrete/continuous	analog
◆	PUSO	monomial	discrete	binary
✚	CNF	clause	discrete	binary

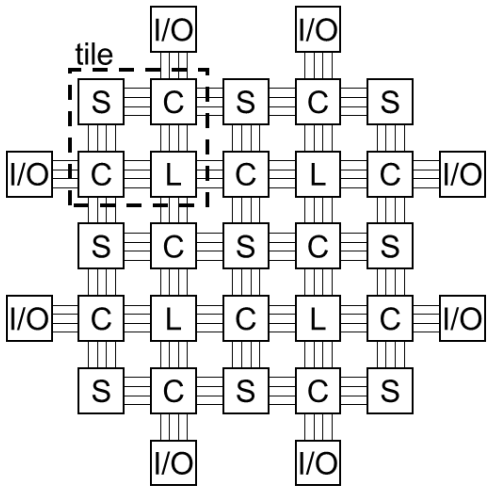
- Shifted PUBO = shift variable ranges in pre-processing step
- Hybrid and clause designs are best for most studied problems
- PUSO (spin-based objective function) design is best for XOR-dominated SATs
- Binary-weight discrete IM are viable for SRAM-based designs

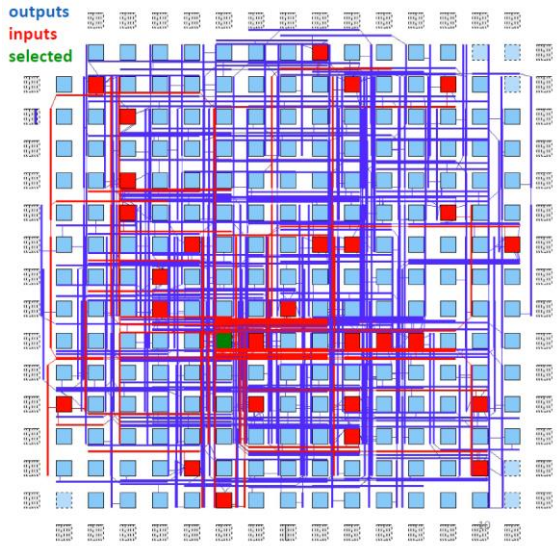
- **Hardware area is approximated by the number of crosspoint devices in xbar implementation**
- **Circuit area of proposed approaches is independent of the problem order *K***

T. Bhattacharya et al., "Unified framework for efficient high-order Ising machine hardware implementations", 2024

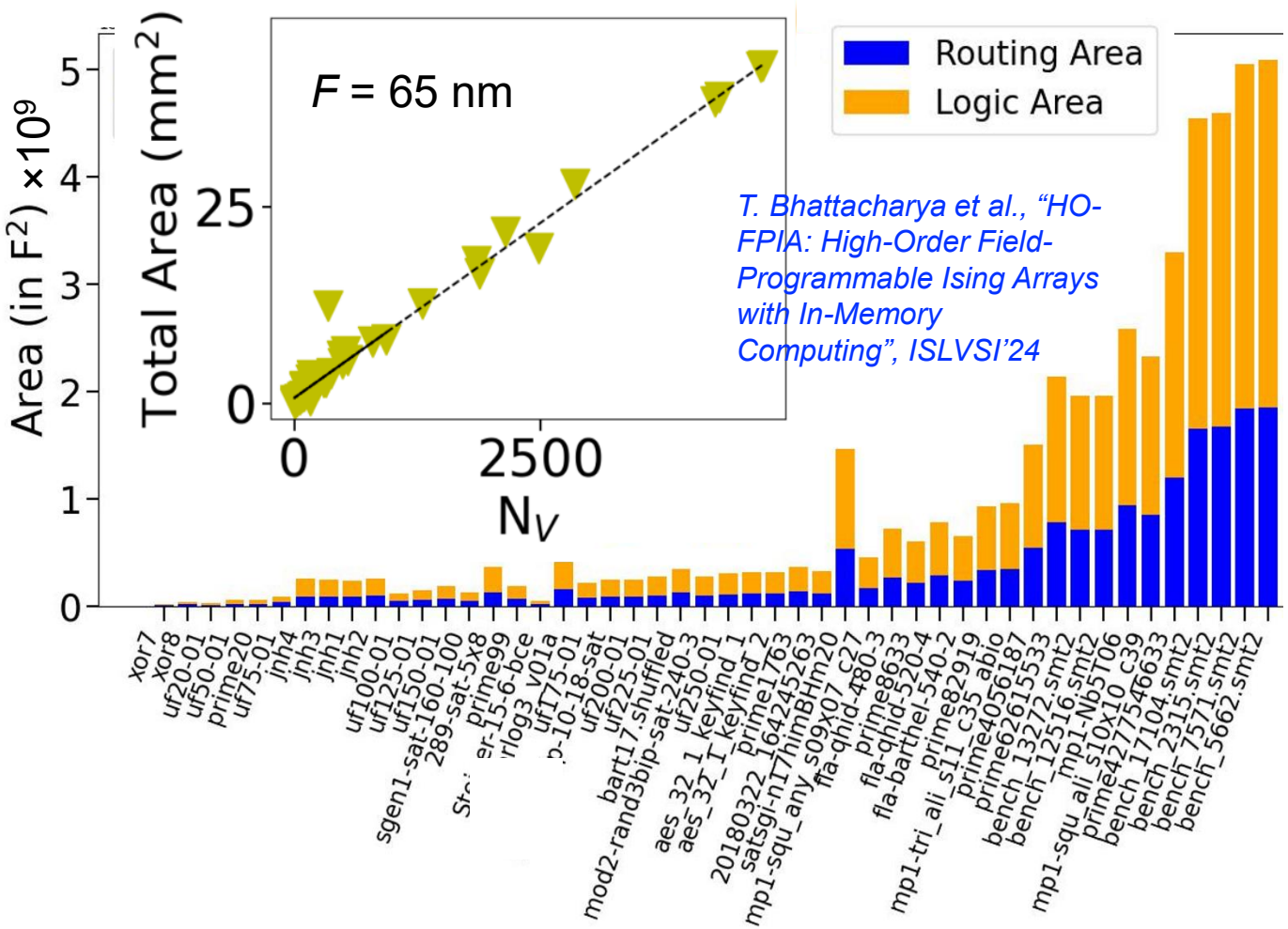
FPIA: Field Programmable Ising Arrays

- **FPIA's main features:**
  - Takes advantage of problem sparsity
  - Break a larger "logical" coupling array into many smaller ones, while ensuring inter-tile connectivity between spins
  - Classical island-type field programmable gate array (FPGA) architecture
  - Reuse routing architecture and tools developed in FPGA community





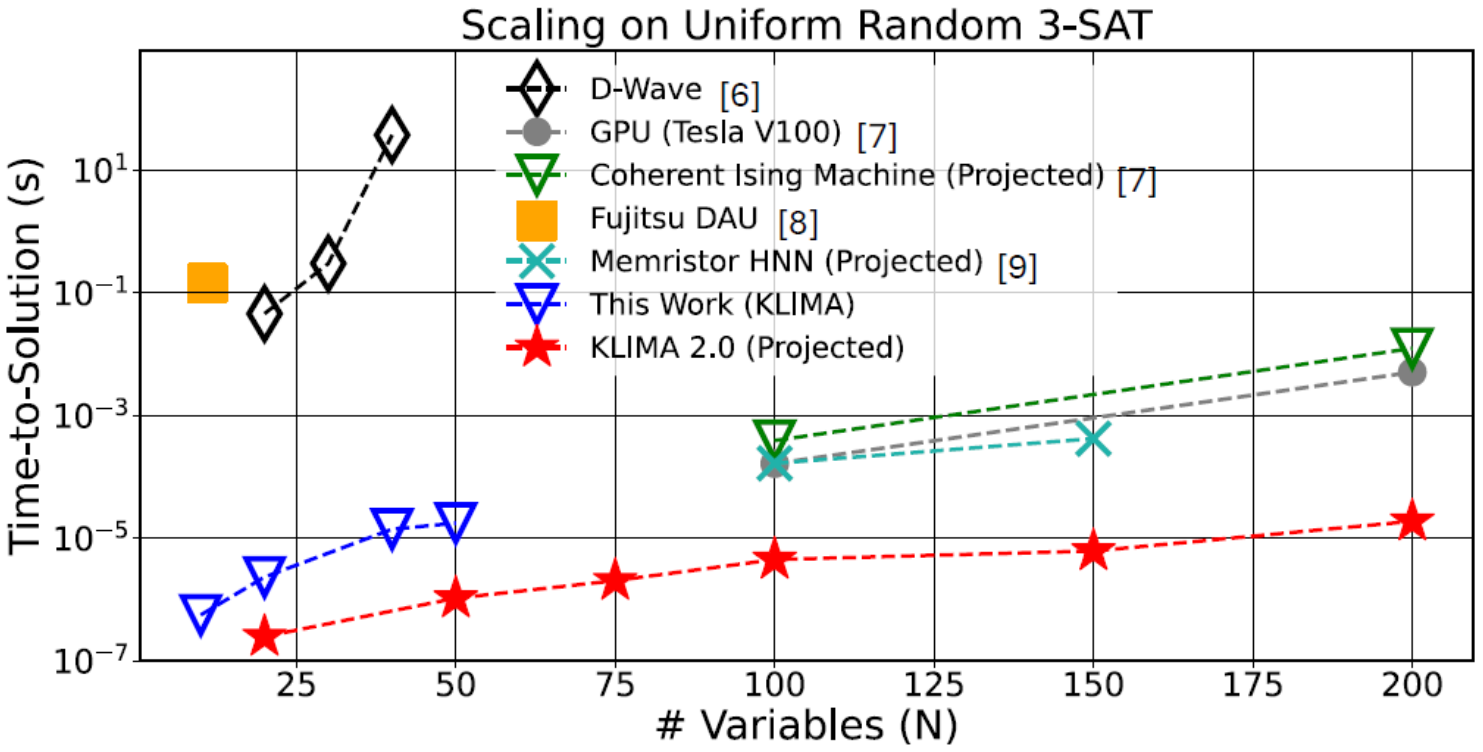
- **Area estimates for SRAM-based FPIA**



>10K (>100K) variables in ~1 cm² in 65 nm (22 nm) CMOS process for hard 3-SAT problems!

Summary and Future Work

- Experimental demo SRAM-based arbitrarily-order SAT solver outperforming SOTA solvers
- Generalized framework for building efficient arbitrary order Ising machines for solving PUBO
- Top-level architecture to implement up to 100K variable hard 3-SAT problems on a single chip
- Testing of 2nd generation SRAM-based SAT solver tile prototype



>10x expected improvement in speed and >2x energy on uniform 3SAT compared to 1st generation

T. Bhattacharya et al, HotChips'25

Selected Publications

Earlier work

- L. Gao et al. “Digital-to-analog and analog-to-digital conversion with metal oxide memristors for ultra-low power computing”, *Proc. NanoArch’ 13*, July 2013
- X. Guo et al. Modeling and experimental demonstration of a Hopfield network analog-to-digital converter with hybrid CMOS/memristor circuits”, *Frontiers in Neuroscience* 9, art. 488, 2015
- M. Mahmoodi et al. “Versatile stochastic dot product circuits based on nonvolatile memories for high performance neurocomputing and neurooptimization”, *Nature Communications* 10, art. 5113, 2019
- M. Mahmoodi et al. “An analog neuro-optimizer with adaptable annealing based on 64×64 0T1R crossbar circuit”, *Proc. IEDM’19,* pp. 14.7.1-14.7.4, Dec. 2019
- Z. Fahimi et al. “Combinatorial optimization by weight annealing in memristive Hopfield networks”, *Nature Scientific Reports* 11 (1), art. 16383, 2021

Recent (QuICC collaboration) work

- G. Pedretti et al. “Zeroth and higher-order logic with content addressable memories”, *Proc. IEDM’23*, 1-4, Dec. 2023
- M. Hizzani et al. “Memristor-based hardware and algorithms for higher-order Hopfield optimization solver outperforming quadratic Ising machines”, Proc. ISCAS’24, 2024
- D. Dobrynin et al. “Energy landscapes of combinatorial optimization in Ising machines”, *Physics Review B*, 110 045308, 2024
- T. Bhattacharya et al. “Computing high-degree polynomial gradients in memory”, *Nature Communications*, 15, art. 8211, 2024
- G. Hutchinson et al. “FPIA: Field-programmable Ising arrays with in-memory computing”, *Proc. ISPLED’14*, 2024
- T. Bhattacharya et al. “HO-FPIA: High-order field-programmable Ising arrays with in-memory computing”, *Proc. ISVLSI’14*, 2024
- G. Pedretti et al. “Solving Boolean satisfiability problems with resistive content addressable memories”, *Nature Unconventional Computing*, 2 7, 2025
- T. Bhattacharya et al. “A fully integrated mixed-signal compute-in-memory accelerator for arbitrary order Boolean satisfiability problems”, *Proc. VLSI Symposium’25*, 2025
- T. Bhattacharya et al. “KLIMA: Low-latency mixed-signal In-memory computing accelerator for solving arbitrary-order Boolean satisfiability” *Proc. HotChips’25*, 2025 (accepted)
- T. Bhattacharya et al. “Unified framework for efficient high-order Ising machine hardware implementations”, 2025 (submitted)
- G. Hutchinson et al. “CHIM: Compressed high order Ising machines”, 2025 (submitted)

Thank You!

dimastrukov@ucsb.edu