SACHI: A Stationarity-Aware, All-Digital, Near-Memory, Ising Architecture

Siddhartha Raman Sundara Raman, Lizy K. John, Jaydeep P. Kulkarni
The University of Texas at Austin
s.siddhartharaman@utexas.edu

2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA) | 979-8-3503-9313-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/HPCA57654.2024.00061

***Abstract*—Recently there have been efforts to solve difficult computation problems harnessing or drawing inspiration from nature. A prominent example is the use of Ising machines for solving NP-complete problems [1], [23]. Ising machines have evolved from quantum/optical annealers and oscillator-based designs [1] to the recent CMOS-based Von-Neumann [36]/in-memory designs [35]. While prior works have demonstrated the power of Ising machines to solve complex real-world problems, the state-of-the-art Ising accelerators are dedicated accelerators that are useful only for a class of problems, involve complex data converter circuits (ADCs/DACs), are unreliable compared to the rest of the CMOS SoC due to the use of process-variation sensitive/specific embedded memory technologies.**

In this paper, we present an all-digital Ising architecture realized using repurposing of L1 cache of a CPU. It relies on processing in-memory technology implemented in SRAM. SACHI solves the reliability problems of prior works such as BRIM, eliminates the need for ADCs/DACs, and provides Ising compute acceleration with minor hardware overhead over a CPU pipeline. The novelty of the proposed approach consists of (i) tightly coupled interfacing of the accelerator to the CPU, (ii) reuse/repurposing of existing hardware to provide acceleration, (iii) ability to achieve higher parallelism than earlier Ising designs due to reuse-aware compute, and (iv) improved performance/energy for a wide variety of large-sized high precision real-life optimization problems using novel compute/mapping strategies. In comparison to BRIM, the proposed all-digital Ising accelerator achieves (i) 36x, 160x, 286x, 300x better performance, (ii) 72x, 79x, 80x, and 75x improved energy, (iii) reuse of 4x, 32x, 200x, and 4000x is observed for asset allocation, molecular dynamics, image segmentation, and traveling salesman respectively.

I. Introduction

Recently, there has been a surge in research focused on solving combinatorial optimization problems (COPs) by drawing inspiration from nature's principles or harnessing natural phenomena. A significant illustration of such an approach is the utilization of Ising machines [3] [7], which hold great promise in tackling NP-complete optimization problems. Ising machines [1], [23] have emerged as a promising frontier, offering innovative approaches that leverage the principles of statistical mechanics to represent and solve these optimization problems efficiently. These are well-suited for optimization problems [11], particularly those involving minimizing energy in physical systems, such as max-cut, asset allocation, graph partitioning, traveling salesman, etc. These differ from classical machines, as classical machines are versatile and can handle a wide range of computational tasks, from scientific simulations/data analysis to general-purpose computing. Ising machines use concepts of spins and interaction coefficients

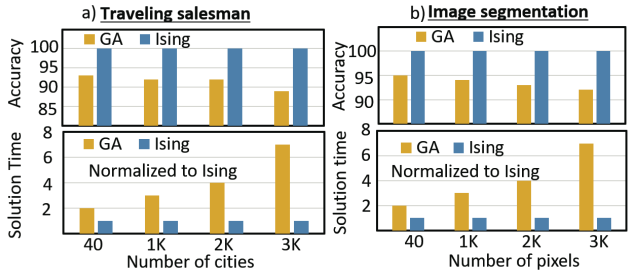

Fig. 1. **Ising machines provide higher accuracy and lower solution time. (i) Top - Solution accuracy under iso-performance condition, and (ii) Bottom - execution time normalized to Ising under iso-accuracy, comparing Genetic Algorithm (GA) and Ising for a) traveling salesman, and b) image segmentation problems.**

(IC) to represent and solve optimization problems efficiently [16]. They encode variables and constants as spins and ICs, respectively. They utilize the Hamiltonian energy function as a heuristic to find optimal solutions to COPs. For instance, Ising machines can achieve more accurate solutions in less time compared to other heuristic-based optimization algorithms, like genetic algorithms. Fig.1 illustrates the solution accuracy of genetic algorithms (GA) [14] and Ising machines for traveling salesman and image segmentation problems in the top two figures. Ising machines are seen to provide greater than 99% accuracy whereas genetic algorithms achieve less than 95% accuracy. In iso-accuracy scenarios, the solution time for Ising is 2x-6x smaller than GA.

From an architectural standpoint, the distinctions in the representation of spins and ICs give rise to different Ising machine implementations, which have been realized using physical [23] or iterative models [35]. The traditional realization of Ising machines captures the dynamics of the physical Ising model by using qubits [9] [8], coupled oscillators [34] [33], and optical annealers [20] [12]. An excellent summary of three and a half generations of Ising machines is provided in [23]; hence we avoid a detailed description here. The major challenge of these approaches is the need for cryogenic operating temperature in qubits, increased power requirement in coupled oscillators, and high area requirement in optical annealers. Another approach involves the usage of CMOS-based Von-Neumann-like iterative Ising machines [36], which perform iterative updates to the spins to achieve the approximate ground state solution. However, the issue is that in a real-life application consisting of many variables, extensive energy is

2378-203X/24/$31.00 ©2024 IEEE
DOI 10.1109/HPCA57654.2024.00061

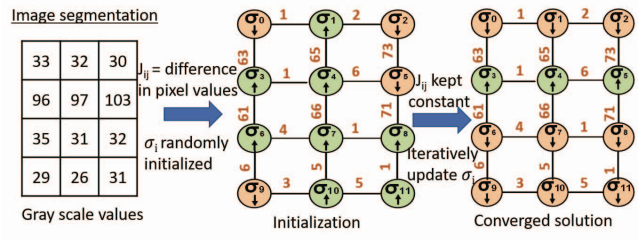

Fig. 2. **COP workload mapping to Ising Machine - Image segmentation example for 4*3 image with the edges showing interaction coefficient as a difference between pixel values. Spin of +1 indicated by green) and -1 by orange. Spins are randomly initialized and Ising machine converges to a segmented image.**

spent on the data movement of variables. In order to reduce the data movement costs, computing in/near memory (CIM/CNM) based Ising machines are being investigated. The advantage of CIM designs is that modern high-density, inexpensive on-chip memory is repurposed to map large-sized COP onto them for performing efficient in-memory compute.

The prevailing challenges with the existing state-of-the-art accelerators, whether in the physical approach (BRIM) [1] or the iterative approach (Ising-CIM) [35], are multi-faceted. These are dedicated domain-specific accelerators optimized for only a specific subset of COPs, and do not support different resolutions for efficient compute. Furthermore, their reliance on analog data converters or blocks render them susceptible to process variations, causing reliability issues. In addition, the lack of reuse in them leads to increased data movement, further exacerbating energy/performance concerns.

The major contributions of the paper are:

- Architecting an all-digital Ising machine that repurposes the L1 cache hardware for in-memory compute, and presenting an accelerator that is tightly coupled to the CPU pipeline, thereby minimizing extra hardware
- A reuse-aware computing strategy along with multiple data-stationary PIM designs that leverage the compute strategy to perform less redundant compute, achieving high parallelism and energy efficiency
- A tuple mapping strategy is proposed to abstract the incoming graph structure, that makes SACHI scalable to any graph used to represent large-size real-life COPs
- A mixed encoding scheme to enable SACHI to be reconfigurable to any precision upto 32-bit for in/near-memory compute, without the usage of DACs/ADCs
- Evaluation of SACHI using several real-world complex COPs indicate that SACHI offers 160x/36x/286x/300x better performance and 79x/72x/80x/75x better energy over BRIM for molecular dynamics/asset allocation/image segmentation/traveling salesman problems. Furthermore, SACHI offers a speedup of 90x and energy improvement of 75x over Ising-CIM.

II. BACKGROUND

A. Mapping COP onto Ising Model

The Ising model [28] [4], originating from statistical mechanics, is used to study the alignment of spin orientations

Property	BRIM	Ising-CIM	SACHI
Dedicated accelerator	Yes	Yes	No, **repurpose** L1 cache
Ising machine	Physical	Iterative	Iterative
Architecture	Coupled oscillator	In-memory	Near-memory
ADC/DAC	Yes	No	No
Problem size/ Ising graph	1000 nodes/ all graphs	Any size/ King's Graph	**Scalable** to any size/ all graphs
Max. Compute resolution	Signed 4-bit	Unsigned 2-bit	**Reconfigurable**, upto signed 32-bit
Reuse	No - 1 compute for every bit fetched from memory		High reuse using **reuse-aware compute**
Memory array modifications	No	Yes	No

Fig. 3. **SACHI in comparison to to state of the art Ising architectures BRIM and Ising-CIM - SACHI is a repurposable, scalable, reconfigurable near-memory architecture with no modifications to the memory array, no ADCs/DACs, achieving better reuse/energy as compared to prior Ising accelerators like BRIM [1], Ising-CIM [35]**

(either up-spin or down-spin) in a magnetic material under the presence of external perturbations. Individual spins interact with one another and flip their orientations, so that the collection of spins in a magnetic material reach a minimum ensemble energy state (called the 'ground state') [35]. The spins and ICs represent the variables and the relationship between these variables, respectively, in a COP. The minimum ensemble energy state represents the optimal solution to COP. For image segmentation [13], IC identifies the edge value between 2 neighboring pixels (spins) by finding the difference between them, with the mapping onto Ising model (see Fig.2). For traveling salesman [6], IC represents the distance between the 2 cities (spins). For asset allocation, which investigates the feasibility of splitting a net worth of USD X Million valued across N assets (spins) among people, IC is the value of each asset allocated. The optimal solution to COPs for iterative Ising model is obtained by minimizing Hamiltonian energy [10], a function of pair-wise coupling among those spins, given as:

$$H = -\sum_{ij} J_{ij}\sigma_i\sigma_j - \sum_i h_i\sigma_i \tag{1}$$

where J_{ij} represents the ICs, σ_i represents the target spin (for which the update is being performed), σ_j represents the neighboring spins of σ_i, h_i represents the external field, with i, j representing a pair of nodes in a graph. The minimization of H carried out by a divide and conquer update of each spin, based on its interaction with its neighbors [25] results in:

$$H_\sigma = \sum -J_{ij} * \sigma_j - h_i \tag{2}$$

The **spin update** is carried out based on the sign of H_σ:

$$\sigma = \begin{cases} -1, & \text{if } H_\sigma > 0 \\ +1, & H_\sigma < 0 \\ +1/-1 & H_\sigma = 0 \end{cases} \tag{3}$$

Optimization Problem (COP)	Typical problem size	Graph connectivity	Resolution for 1K spins (R)	Size of R-bit COP	Size of 8-bit COP
Image segmentation	1000-1M pixels	Densely connected	6-bit	Fits in L1 cache	Exceeds L1 size
Asset allocation	100-1000 assets	Sparsely connected	7-bit	Fits in L1 cache	Exceeds L1 size
Molecular dynamics	100K-1M atoms	King's (8-neighbor)	4-bit	Fits in L1 cache	Fits in L1 cache
Traveling salesman	10-30K cities	Fully connected	5-bit	Exceeds L1 size	Exceeds L1 size

Fig. 4. **- Real-life COPs have a wide range of problem sizes with varied graph connectivity, varied minimum resolution (4-7 bits) to achieve 90% accuracy under iso-performance condition. Using a fixed 8-bit IC for all COPs results in additional data movement cost because 8-bit COP overflows when placed in 64KB L1 cache while lower resolution (4-7bit) compute fits inside the L1 cache, motivating a reconfigurable, scalable compute architecture.**

The local spin update might result in H being trapped in a local minimum. Simulated annealing is then performed to achieve the global minima by probabilistic spin-flips.

B. Prior Generations of Ising Machines

Ising machines are realized using physical or iterative models. Physical machines use various technologies, such as quantum annealers [19], optical annealers [20], ring oscillator-based coupled oscillators [34] [32], resistive-coupled oscillators [1] [23] for computation. Quantum annealers encode spins with qubits and ICs with qubit-qubit coupling, but they require cryogenic operation, leading to cooling costs. Optical annealers utilizing light polarization/phase for encoding spins/ICs face challenges with increased area and susceptibility to noise. The ring oscillator approach utilizes multiple inverter stages to encode spins. These are again exclusively used for solving COPs and face scalability challenges due to the area overhead caused by using multiple inverter stages to represent a single spin state. In the resistive coupled oscillator approach (BRIM), spins are stored in capacitors, and resistances are programmed according to ICs (disadvantages in Sec.III). We would like to mention that the philosophy of physical Ising machine is completely different from that of iterative Ising machines. The main reason for comparing with BRIM is that it is the only prior work in this domain familiar to architects.

Iterative machines, like Hitachi's Ising machine [36] and digital annealers in CIM-Spin [25], use dedicated logic for accelerating Ising compute. However, these approaches suffer from drawbacks, including high data movement (because of Von-Neumann architecture), limited scalability, dedicated acceleration, reduced energy efficiency, and acceleration specific to only certain set of COPs. To address data movement issues, Ising-CIM [35] utilizes embedded DRAM [27] [22] [17] to perform computations in memory (disadvantages in Sec.III).

III. DESIGN GOALS AND MOTIVATION

SACHI is motivated by the weaknesses of prior Ising machines along with an overview of how SACHI overcomes these issues. Earlier architectures need data converters like DACs/ADCs and need specific technologies and/or programming steps coupled to the devices. For instance, ZIV diodes

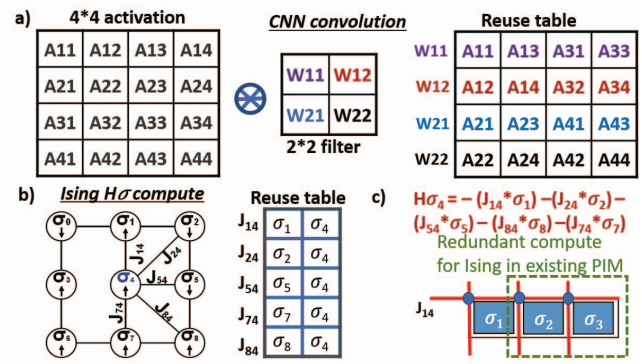

Fig. 5. **a) CNNs offer reuse, as the same weight is shared across multiple activations. b) However, Ising compute for H_σ natively offers no reuse as each J_{ij} is uniquely mapped between σ_i and σ_j. c) Memory array mapping for existing PIM design (Ising-CIM) [35] shows that although the last 2 columns are computed, they are redundant. This is because J_{14} does not interact with σ_1 and σ_3 implying that $J_{14}*\sigma_1$ and $J_{14}*\sigma_3$ are redundant. This redundant compute results in energy degradation. The cause for this degradation is primarily because of the unnecessary discharge of bit cells associated with the redundant compute.**

and programmable resistances are required in BRIM, analog charge-sharing with a modified embedded DRAM memory array is required in Ising-CIM. SACHI creates an all-digital architecture with standard components (no specific devices/technology requirements), and can be easily integrated into CMOS SoC. Fig.3 summarizes the main features of state-of-the-art Ising machines compared to SACHI.

*1) **Repurposability**:* Domain-specific dedicated accelerators like BRIM/Ising-CIM require frequent CPU-accelerator interaction causing performance/energy overhead. BRIM/Ising-CIM are dedicated because of the difficulty in integrating (i) coupled oscillators made of ZIV diodes and (ii) modified embedded DRAM array into the CPU pipeline.

Instead of a dedicated accelerator, SACHI repurposes the L1 cache when needed, reusing SRAM for in-memory compute with minimal CPU-friendly digital logic.

*2) **Scalability**:* Ising-CIM is explicitly designed for King's graph, without any restriction on the size of COP (detailed in Sec.IV.B). BRIM cannot be scaled for solving large-sized COPs but makes no assumption about the underlying graph. BRIM's scalability is hindered due to the following factors: (i) The number of programmable switches/diodes needed for node interactions scales as $O(n^2)$, where n is the number of nodes. (ii) Ensuring that the capacitance to encode spins does not discharge is crucial to prevent inadvertent spin-flips. Discharge is likely to happen with large problem size.

SACHI addresses the need to scale to large real-life COPs with diverse connectivity (Fig.4), with its unique tuple mapping, tuple-rep property, and storage-array-based updates.

*3) **Reconfigurability**:* The compute precision/resolution (R) of Ising-CIM/BRIM is restricted to 2-bit/4-bit. Ising-CIM's restriction arises due to the data mapping that can support only upto 2-bits. BRIM's limitations arise from (i) Challenges in obtaining accurate resistances for higher IC values and the requirement of a configurable DAC for multi-bit R. (ii) Obtaining an 8-bit design involves representing 256 values in

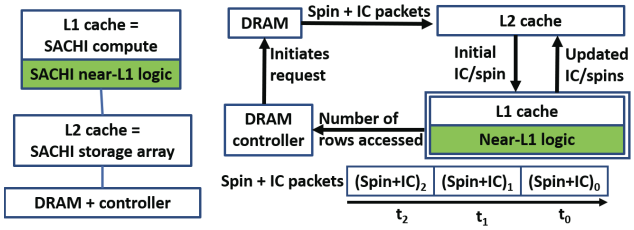

Fig. 6. **Overview of SACHI- SACHI's compute array is mapped onto L1 cache, storage array onto L2 cache with minimal near-L1 logic (0.3% of CPU area), enabling easy integration into CPU pipeline. DRAM controller prefetches requests based on the number of uncomputed rows in compute array to minimize data movement cost between DRAM and storage array.**

1V range, requiring an infeasibly small voltage resolution for DAC of 1V/256=3mV

SACHI addresses the need to support higher R using mixed-encoding scheme that reduces dot-product into XNOR, enabling high-precision PIM compute without DAC/ADCs. However, a larger R requires more memory space. For 1K spins, achieving 90% accuracy requires a minimum R of 4-7 bits, depending on the COP as shown in Fig.4. SACHI's PIM/near-memory compute makes no assumptions about R and can be reconfigured for any R, without accuracy loss.

*4) **Reuse**:* Reuse for a variable is defined as the number of required computes performed for a variable that is mapped onto (i) a row of compute array in PIM-designs like Ising-CIM, SACHI (ii) a row of coupled oscillator nodes/array in BRIM. BRIM/Ising-CIM offer no reuse.

BRIM has no reuse because every IC mapped onto the coupled oscillator array using ZIV diodes is used in only 1 required compute. Reuse in PIMs is dependent on the algorithm and efficient mapping. For example, CNNs inherently exhibit reuse >1 due to shared weights across activations in a layer, making them suitable for a weight-stationary approach (Fig. 5a). In contrast, the Ising model for computing H_σ (eqn.2) does not naturally offer reuse since each pair of spins and ICs in the graph has a unique mapping. Fig. 5b) shows that each J_{ij} is uniquely mapped between σ_i and σ_j, making reuse equal to 1. Fig.5c) explains the increased energy requirement due to redundant compute. With σ_1, σ_2, σ_3 stored in compute array, and J_{14} mapped onto a row of compute array, the only required output is $J_{14}*\sigma_1$. However, there are 2 additional redundant computes ($J_{14}*\sigma_2$, $J_{14}*\sigma_3$). The reason for redundant compute is because J_{14} does not interact with σ_2 and σ_3. These redundant computes further result in energy overhead due to the unnecessary discharge of bitcells associated with redundant compute. Mathematically, an IC tuple (IC[S] = <IC1, IC2..ICn> ; n>1) needs to be formed for each spin, with reuse equal to the number of elements in the formed IC tuple. A design satisfies "IC criterion", if reuse is >1. Existing PIM design (Ising-CIM) does not satisfy the IC criterion, implying no parallelism across a row of the compute array row.

SACHI uses a reuse-aware data-stationary compute strategy to improve reuse across all elements in a row of compute array, improving parallelism and energy efficiency.

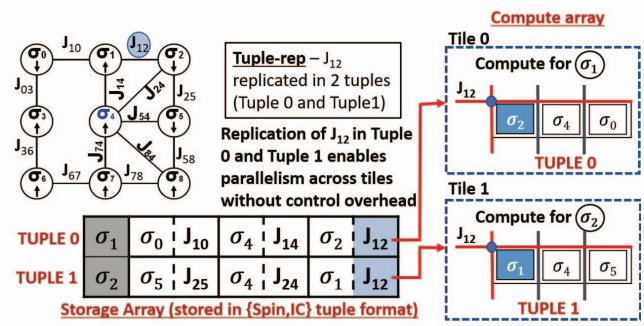

Fig. 7. **Illustration of tuple mapping strategy - a) Storage array organized as tuples, with each consisting of the connected spins and ICs for a given spin. Ising compute operates at the abstraction level of these tuples, thereby not requiring visibility of the incoming graph helping achieve scalability to any graph type. b) Tuple-rep ensures 1:1 mapping between a storage array tuple and a compute tile row, enabling the partitioning of the graph into sub-graphs, with each sub-graph computed independently in different tiles without any control overhead**

IV. THE SACHI ARCHITECTURE

We describe the SACHI architecture, which combines elements of re-purposability, scalability, reconfigurability, and reuse-aware data-stationary near-memory compute, to realize a high-performance, energy-efficient Ising machine.

A. Repurposable architecture

SACHI repurposes modern-day CPU SoC components to accelerate Ising Hamiltonian computation (H_σ) with minimal area overhead. The components used include DRAM, L2 cache, CPU, and a repurposed L1 cache, utilizing 8T SRAM bitcells (Fig.6). The storage array is mapped onto L2 cache, while the compute array is mapped onto L1 cache, with additional near L1 peripheral logic occupying only 0.3% of AMD's Zen3 CPU area. The data flow in SACHI is as follows: For spins and ICs that fit on-chip, DRAM is accessed once at the beginning to fill the storage arrays, and no further re-access is needed. The storage array (L2 cache) acts as a buffer for storing the initial Ising graph transferred from DRAM, while the compute array (L1 cache) performs H_σ computations using the spins/ICs from the storage array and writes the updated spins back onto the storage array. In cases where variables and ICs do not fit on-chip, multiple rounds of computations are required to optimize large graphs, necessitating rewriting of the compute and storage arrays. To optimize this long latency operation, we use a prefetching approach that anticipates potential accesses. CIM accesses have structured and predictable address patterns, unlike regular memory access. We access the rows in the compute array top-to-bottom in successive cycles, and a counter in the DRAM controller tracks the number of remaining rows to be accessed. When the count reaches a threshold (meeting DRAM-to-storage + storage-to-compute array data movement latency), a prefetch request is initiated to ensure the timely arrival of DRAM-requested data.

B. Scalability using tuple mapping

*1) **Scalability to all graphs**:* SACHI operates on the incoming graph at the abstraction level of tuples, enabling H_σ

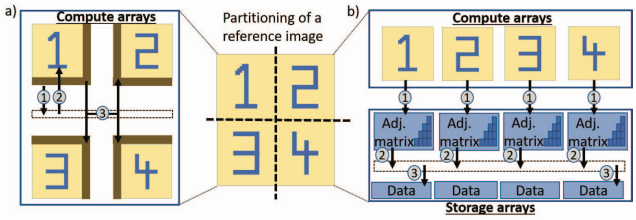

Fig. 8. **Scalability to any graph size - a) Ising-CIM approach: duplicate edge cells (dark brown) onto adjacent CIM arrays. 1 indicates updated spin value computed in the array. 2 indicates writeback of the same updated spin value for non-edge cells. 3 indicates the broadcast of updated spin value in the case of edge cells (duplication) to adjacent CIM arrays. This approach minimizes interaction between compute arrays to only edge cells, but uses King's Graph properties to accomplish this and is performance-inefficient b) SACHI approach: store the adjacency matrix and buffer the updated spins of relevant tuples by reading the adjacency matrix. 1 indicates updated spin value computed in the array. 2 indicates read of the adjacency matrix. 3 indicates update of relevant tuples in storage array. This is graph-type agnostic and performance-efficient**

compute to be incognizant of the graph connectivity. This unique feature enables SACHI to perform Ising computations for all graphs, irrespective of their connectivity. Fig.7a) illustrates our tuple mapping strategy. Each row in the storage array is a tuple for a particular spin, consisting of the neighboring spin states, the connecting ICs, and the external magnetic field (Fig.7a) Furthermore, the same IC/spin is present in more than one row, called the **"tuple-rep"** property (Fig.7b). For instance, J_{12} is an entry present in the tuples of both σ_1 and σ_2. This enables the compute for $H_{\sigma 1}$ and $H_{\sigma 2}$ to be independent of each other by ensuring 1:1 mapping between a tuple in the storage array, and a row in the compute array, enabling partitioning of large graphs into subgraphs. If not for tuple-rep (J_{12} was present only in σ_1's tuple), H_σ compute for σ_2 introduces an interdependency of rereading the storage array for obtaining J_{12} from the tuple corresponding to σ_1, causing performance bottlenecks with control overhead.

*2) **Scalability to any graph size**:* To efficiently solve large COPs, reducing inter-CPU core interactions is crucial. For PIM designs, this involves minimizing interactions between sub-arrays of compute array, while being graph-type agnostic, and extending the same philosophy to reduce inter-core interactions. Firstly, unlike deep-neural networks, where the output of one layer feeds as input to another, requiring data movement, Ising model does not incur any "layerwise" data movement. The only required minimal data movement is for spin-updates. Therefore, the algorithm inherently requires fewer inter-array interactions. In **Ising-CIM**, when graphs are partitioned, spins on the edge of the partition are duplicated across 2 adjacent CIM arrays. Non-edge cells perform local spin updates in eDRAM (2 in Fig.8a) based on the computed updated spin value (1 in Fig.8a). Edge cells undergo a local read-modify-write(update) based on the broadcasted updated spin value from adjacent CIM arrays (3 in Fig.8a). Although only edge cells necessitate interaction between adjacent arrays, this approach has several drawbacks. Execution time is dependent on a)cycles per iteration (CPI) and b)number of iterations (IT). With respect to CPI, the local update in Ising-CIM hinders

Mixed encoding (enc.) scheme			Mixed encoding enables multiplication via in-memory XNOR compute					
1-bit spin (S) enc.	Reconfigurable R-bit IC enc. (R=9,3 shown)		S XNOR IC (In-memory XNOR)		(S XNOR IC) + 1 (near-memory addition)		Spin(S)*IC result for Hσ compute	
	R = 9	R = 3	R = 9	R = 3	R = 9	R = 3	R = 9	R = 3
0	9'h087	3'h3	9'h178	3'h4	9'h179	3'h5	9'h179	3'h5
0	9'h179	3'h5	9'h086	3'h2	9'h087	3'h3	9'h087	3'h3
1	9'h087	3'h3	9'h087	3'h3	9'h088	3'h4	9'h087	3'h3
1	9'h179	3'h5	9'h179	3'h5	9'h07A	3'h6	9'h179	3'h5

Fig. 9. **SACHI's reconfigurability achieved using mixed encoding scheme, with -1/+1 spins stored as 0/1, ICs encoded in 2's complement form to enable in-memory XNOR for dot product between J_{ij} and σ_j without DAC/ADC (unlike BRIM). (S XNOR IC)/((S XNOR IC)+1) is computed to enable multi-bit signed IC dot product (unlike Ising-CIM). 9-bit J_{ij}= 135 (9'h087), -135 (9'h179) and 3-bit J_{ij}= 3 (3'h3), -3 (3'h5) product with σ_j=1 (0),-1 (1) is shown**

Ising compute performance, making each compute a 2-cycle operation (1 each for compute and update), because of read-write conflict. SACHI overcomes this, making compute 1-cycle operation, as the data movement from compute-storage array is overlapped with useful compute in compute array. Furthermore, there is no read-write conflict, as the write happens to a separate array. Therefore, Ising-CIM has 2x CPI compared to SACHI. With respect to IT, local update leads to performance gain only in large-sized COPs. For instance, in localized King's Graph, there is a minor performance gain of 0.1x, as opposed to 1.8x in a complete graph for 1M spin configuration. The gain is with respect to performing storage array based update, similar to SACHI. Therefore, in Ising-CIM, it is beneficial to retain the original values and reap benefits from improved CPI. In SACHI, this happens naturally for COPs that require re-write to compute array. A portion of the storage array stores the adjacency matrix, and is read (2 in Fig.8b) to identify the relevant tuples containing the incoming spin, allowing updates to the storage array (3 in Fig.8b) only for the relevant tuples. This ensures that when the compute array is re-written, few tuples already have the updated spin values Therefore, SACHI provides the ideal middle-ground mimic-ing local update behavior for fast convergence, when necessary, while providing improved performance due to 1-cycle compute+update operation.

*3) **Scalability to any graph size**:* To efficiently solve large COPs, reducing inter-CPU core interactions is crucial. For PIM designs, this involves minimizing interactions between sub-arrays of compute array, while being graph-type agnostic, and extending the same philosophy to reduce inter-core interactions. Firstly, unlike deep-neural networks, where the output of one layer feeds as input to another, requiring data movement, Ising model does not incur any "layerwise" data movement. The only required minimal data movement is for spin-updates. Therefore, the algorithm inherently requires fewer inter-array interactions. In **Ising-CIM**, when graphs are partitioned, spins on the edge of the partition are duplicated across 2 adjacent CIM arrays. Non-edge cells perform local spin updates using the local write drivers in eDRAM (2 in Fig.8a) based on the

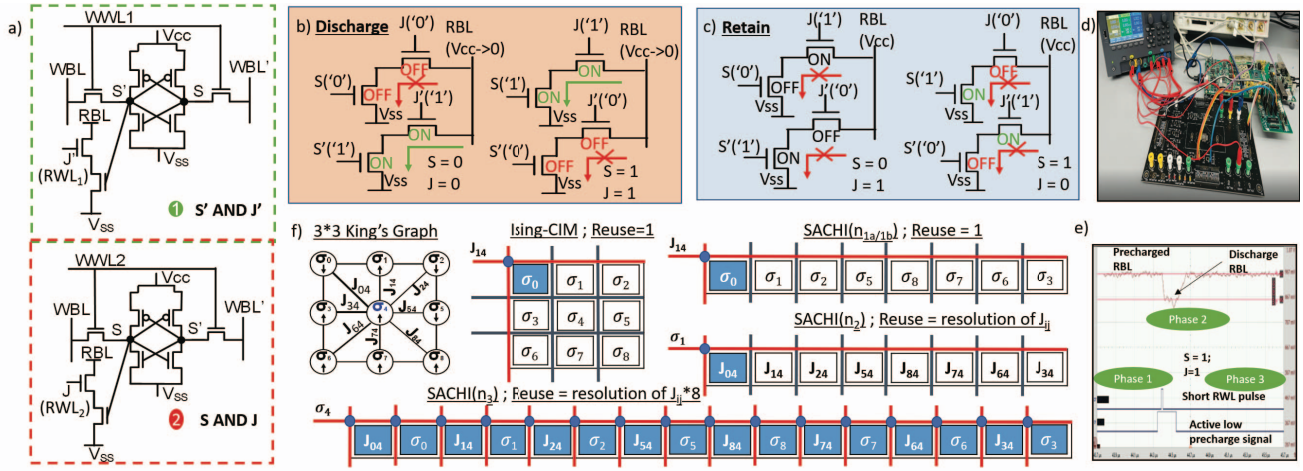

Fig. 10. **SACHI's use of in-memory XNOR - a) 2 8T SRAM bitcells in the same column storing complimentary values of spin/IC with the bitcells computing S'&J', S&J b) RBL discharge c) RBL retain for XNOR compute with no memory array modifications/requirement of DAC and ADC d) In-memory XNOR test-structure in a silicon prototype e) Oscilloscope capture confirming bitline discharge for S=1 XNOR J=1 f) SACHI reuse in different stationarities illustrated with King's graph - SACHI(n_{1a}) has reuse=1, as in Ising-CIM as J_{14} is used only for multiplying with σ_0. SACHI(n_2), SACHI(n_3) employ IC, mixed stationary design respectively to improve reuse to J_{ij} resolution, (J_{ij} resolution*8) respectively**

computed updated spin value (1 in Fig.8a). Edge cells undergo a local read-modify-write based on the broadcasted updated spin value from adjacent CIM arrays (3 in Fig.8a). Although only edge cells necessitate interaction between adjacent arrays, this approach has several drawbacks. Firstly, the local update for non-edge cells causes the loss of the original spin value before completing an iteration. While this approach is scalable to large-sized King's Graph due to King's Graph's localized interaction nature, wherein the original spin value is not used again in the same iteration, it cannot be extended to other complex graphs with non-local interaction where the original spin value is reused sometime later. Secondly, the read-modify-write operation hinders Ising compute performance, making each compute a 2-cycle operation (1 each for compute and update). **SACHI** enables scalability by repurposing the storage array for writing the computed updated spin values (1 in Fig.8b), ensuring that the original spin value remains intact in the compute array, without requiring interaction between compute arrays. A portion of the storage array stores the adjacency matrix for connectivity information. While H_σ compute is incognizant of graph connectivity, the update needs to be aware of the connectivity. This matrix is read (2 in Fig.8b) to identify the relevant tuples containing the incoming spin, allowing updates to the storage array (3 in Fig.8b) only for the relevant tuples. Furthermore, data movement latency between storage and compute arrays is hidden by performing useful compute in the compute arrays.

C. Reconfigurability using mixed encoding scheme

*1) **Mixed-encoding**:* SACHI uses a mixed-encoding scheme, wherein +1/-1 spins are encoded as 1/0 and ICs are represented using 2's complement form to enable PIM dot-product for signed multi-bit ICs without DAC/ADC. The dot product (Fig.9) between J_{ij} and σ_j for eqn.2 is simplified as an XNOR operation, leveraging the binary nature of spins, enabling PIM without any array modifications:

$$J_{ij} * \sigma_j = \begin{cases} J_{ij}\text{XNOR}\sigma_j, & \sigma_j > 0 \\ J_{ij}\text{XNOR}\sigma_j + 1, & \sigma_j < 0 \end{cases} \quad (4)$$

*2) **In/near-L1 compute**:* The XNOR operation is performed by activating multiple rows simultaneously of the compute array. To accomplish this, L1 cache, typically made of an 8T SRAM bitcell (unchanged from what is used in modern-day CPUs for L1 caches [30]) with decoupled read and write ports (RWL, RBL, WWL, WBL - read/write word/bit lines) is employed. [26] (Fig. 10a). This bitcell operates in two modes: (i) regular read/write mode and (ii) Ising compute mode. In the regular mode, data is written/read using WBL/RBL by enabling WWL/RWL, respectively. RWL is repurposed for computation during Ising compute mode. Logical AND between input (J) and stored value (S) is achieved by driving RWL_2 with J. For XNOR (S AND J) OR (S' AND J') operation, S' is stored in a different bitcell in the same column, and RWL_1 is driven with J' (Fig.10a). RBL discharges when either (S AND J) or (S' AND J') is high, indicating an XNOR value of 1 (Fig. 10b). RBL retains its precharged value when both (S AND J) and (S' AND J') are low, indicating an XNOR value of 0 (Fig. 10c). The oscilloscope capture for a prototype in TSMC 65nm technology process shows RBL discharge for a single column of SRAM array (size 100*100) (Fig. 10d,e). The waveform for a selected bitcell storing '1' is summarized as: Phases 1/3 - Precharge: An active low precharge signal charges the RBL to 1V. Phase 2 - Compute: A short RWL pulse indicates the incoming value ('1'). Since both the storage node and the incoming RWL are '1', RBL discharges, resulting in XNOR value of 0 (Fig. 10b illustrates the discharge path). Dot product accumulation is performed using full adders situated near memory. This enables (i) parallel execution of PIM

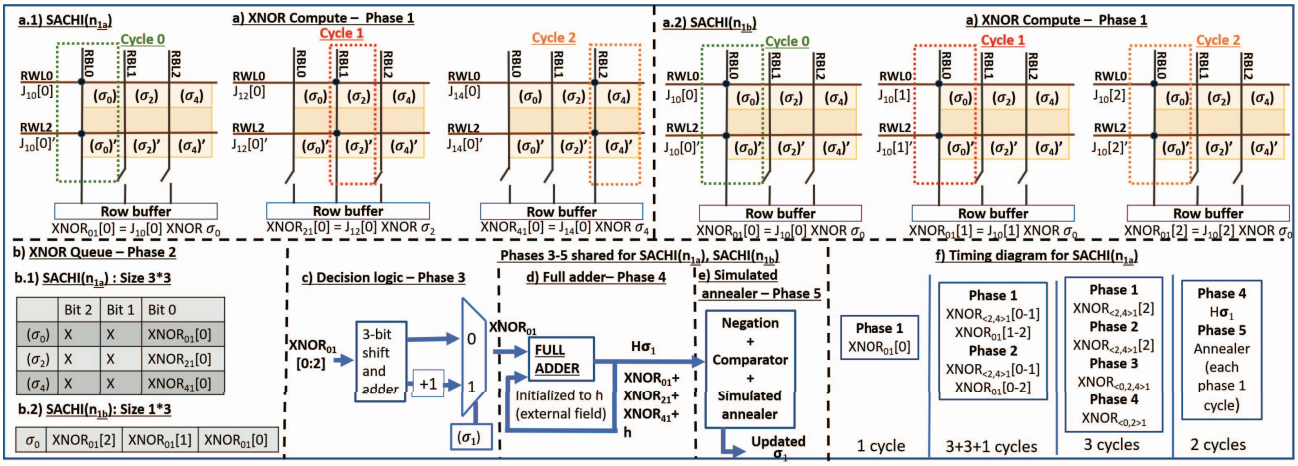

Fig. 11. **a.1) SACHI(n_{1a}) a.2) SACHI(n_{1b}) for graph in Fig.2 with 3-bit J_{ij}. SACHI(n_{1a}) computes i^{th}bit of all ICs before proceeding to the $(i+1)^{th}$ bit (reuse=1). SACHI(n_{1b}) computes all bits of an IC before proceeding to the next IC (reuse=1). Only 1 column is highlighted to indicate reuse of 1. b) Bitwise XNOR requires XNOR queue to queue the different bits computed b.1) SACHI(n_{1a}) requires more number of entries in the queue than b.2) SACHI(n_{1b}) because of Phase 3 preferring SACHI(n_{1b}) order of compute c) Phase 3 performs shift and add among all bits of XNOR compute d) Phase 4 adds the partial XNOR products and e) Phase 5 for simulated annealing. Phases 3-5 are the same for SACHI(n_{1a}) and SACHI(n_{1b}) f) Timing diagram for SACHI(n_{1a}) showing the overlap between phases 1-4 for H_σ compute**

XNOR and accumulation, and (ii) enhanced accuracy due to the reduced susceptibility to process variations in the digital full adder. Accurate in/near-L1 design without assumptions about R makes SACHI support any R-bit compute.

D. Reuse-aware data stationary compute

We propose near-memory architectures that progressively achieve higher reuse: (i) spin stationary (SACHI (n_{1a})), (SACHI(n_{1b})), (ii) IC stationary (SACHI(n_2)), (iii) Mixed stationary (SACHI(n_3)). (Fig.10f) illustrates an overview of the three methods with King's graph.

1) ***SACHI(n_{1a}) = Spin stationary design****:* This design involves storing spins (σ) onto the compute array and mapping ICs onto RWL in a bit-serial manner. This approach, illustrated in Fig. 11 allows sharing a J_{ij} bit across a row of bitcells in the compute array, which is organized as tiles and filled in order (successive spins in the same tile) without interleaving.

The compute is as follows: (i) In **phase 1**, the sharing of J_{ij} across a row of spins leads to redundant XNOR computes. The read-out of redundant dot products is blocked by disabling the bit select of each column (Fig. 11a.1), giving a throughput of 1 XNOR compute every cycle. As you will see, the hardware requirements are influenced by the order of XNOR compute. In SACHI(n_{1a}), XNOR of r^{th} bit of all the neighboring spins is performed before computing XNOR of $(r+1)^{th}$ bit (Fig. 11a.1). This compute order necessitates a storage buffer (called XNOR queue) to store XNOR of individual bits in **phase 2**(Fig. 11b.1). The minimum size of the queue equals (number of neighboring spins * (resolution of J_{ij}+1). In **phase 3** (Fig. 11c), the computed XNOR values generate partial products by shifting and adding individual bits. Additionally, a decision is made based on σ_j to choose between XNOR and (XNOR + 1) after XNOR'ing all the bits of J_{ij} with σ_j. In **phase 4**, the full adder, which is initialized to the external magnetic field, accumulates the partial dot products for each spin (Fig. 11d). In **phase 5** (Fig. 11e), the computed sum is negated to obtain H_σ and is passed through annealer.

The time to compute H_σ for an R-bit IC with N neighbors for a specific spin is in O(R*N). Notably, N cycles are required to fill one column in the XNOR queue, repeated (R-1) times to fill (R-1) columns, and 1 cycle is needed to obtain the first R-bit value with shift and add. This results in ((R-1)*N+1) cycles before phase 3 becomes active (Fig.11f). During this time, phases 3-5 are idle, referred to as "idle time". The reuse is 1, as every J_{ij} bit fetched from the storage array is used in 1 XNOR compute, not satisfying the IC criterion.

2) ***SACHI(n_{1b}) = Optimized SACHI(n_{1a})****:* SACHI(n_{1a}) can be improved in terms of (i) effective resource utilization by minimizing the "idle time" (ii) reducing the overhead of XNOR queue and (iii) better interleaving of tiles.

To tackle the first two issues, we modify the order of XNOR computes to achieve a directed throughput, allowing earlier activation of phases 3-5. Additionally, we store adjacent rows of the storage array in adjacent tiles to improve interleaving.

The compute involves mapping RWL onto successive bits of a specific J_{ij} in consecutive cycles to perform XNOR during **phase 1**. This differs from SACHI(n_{1a}), where this mapping is delayed by N cycles. In other words, all bits of a particular J_{ij} are XNOR-computed before moving on to the next J_{ij}, e.g. $XNOR_{01}[0\text{-}2]$ is followed by $XNOR_{21}[0\text{-}2]$. This order is evident as the bitline select of column 0 is turned ON three times (Fig. 11a.2), reducing the XNOR queue size to 1 row with R columns (Fig. 11b.2) in **phase 2**. **Phases 3-5** are same as SACHI(n_{1a}). Additionally, phase 3 is initiated earlier than SACHI(n_{1a}) since J_{ij} bits are computed earlier to shift and add.

The compute time for H_σ is in O(R*N). However, the idle time is reduced from (R-1)*(N) to R cycles SACHI(n_{1b}) does not satisfy the IC criterion, as reuse is 1 (as every J_{ij} bit fetched is used only in 1 compute).

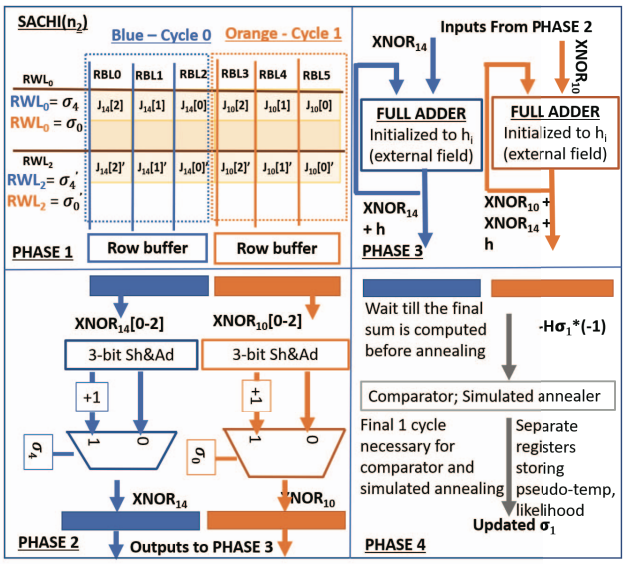

Fig. 12. **SACHI(n_2): IC stationary design performing XNOR across all J_{ij} bits in a single cycle in Phase 1, improving maximum reuse to the resolution of J_{ij} (3 columns highlighted for 3-bit J_{ij}), eliminating XNOR queue in Phase 2. The decision logic, full adder, and simulated annealing logic are shifted by a cycle compared to SACHI(n_1)**

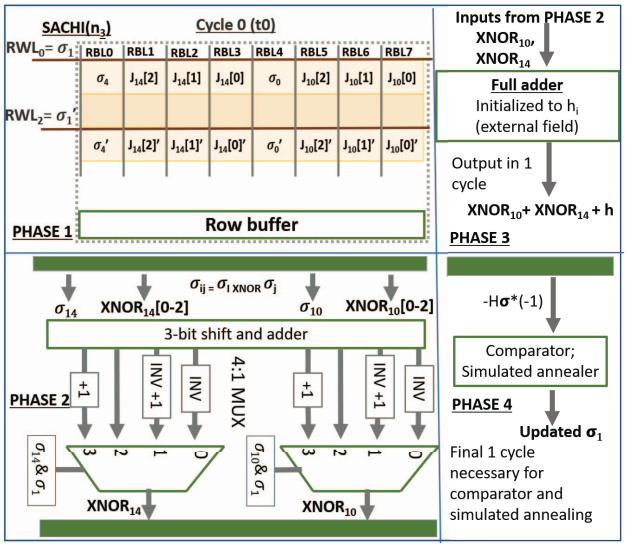

Fig. 13. **SACHI(n_3): Mixed stationary design making use of reuse-aware compute to perform XNOR across all J_{ij} bits and neighbors of target spin in a single cycle in Phase 1, improving maximum reuse to (neighbors)*(J_{ij} resolution) (entire memory array highlighted). The decision logic and adder in Phase 3 are modified to support high throughput**

*3) **SACHI(n_2) = Interaction Coefficient stationary design**:* SACHI($n_{1a/b}$) suffer from (i) low SRAM throughput (ii) low reuse/performance (iii) XNOR queue overhead. To address these issues, SACHI(n_2) computes multiple XNOR operations in parallel by storing ICs (J_{ij}) onto compute array and mapping spins onto RWL. For every i^{th} row's RWL mapped to a spin, [i+num spins]th row's RWL is mapped to the inverted spin value for XNOR compute.

The compute illustrated in Fig. 12 proceeds as follows: In **phase 1**, multiple bitline selects are turned ON to read the computed data from multiple columns so that all bits of a particular XNOR partial product between J_{ij} and S_j are obtained in 1 cycle. Thus, the throughput of the SRAM array is improved from 1 in SACHI($n_{1a/b}$) to R in SACHI(n_2), thereby eliminating the XNOR queue. In **phase 2**, the decision between XNOR and XNOR+1 is taken based on the target spin. In **phase 3**, full adder accumulates the partial dot products.

H_σ compute time is reduced from O(N*R) in SACHI(n_1) to O(N) in SACHI(n_2) and the reuse is equal to resolution (R) of SACHI($n_{1a/b}$), as a single spin is shared by multiple bit-cells, thus satisfying the IC criterion.

*4) **SACHI(n_3) = Mixed stationary design**:* Although SACHI(n_2) offers improved reuse and performance, the unique mapping between 2 spins for a particular IC and the way H_σ is computed using eqn.2 limits the reuse to R.

To further increase reuse, we propose a reuse-aware mixed stationary strategy, based on a few observations. (i) The spins are always binary-valued. (ii) The dot product between J_{ij} and σ_j is resolved as the dot product between J_{ij} and σ_i, if σ_j and σ_i are the same. If the spins are the same, the reuse-aware equation resolves to eqn.2, and inversion of XNOR output if the spins differ. The binary nature of spins makes equality checking of σ_j and σ_i an XNOR operation, which can be computed in parallel with XNOR of J_{ij} and σ_i. This can be summarized as:

$$J_{ij}*\sigma_j = \begin{cases} J_{ij}\text{XNOR}\sigma_i, & \sigma_i > 0 \;\&\; \sigma_i\text{XNOR}\sigma_j = 1 \\ J_{ij}\text{XNOR}\sigma_i + 1, & \sigma_i < 0 \;\&\; \sigma_i\text{XNOR}\sigma_j = 1 \\ J_{ij}\text{XOR}\sigma_i, & \sigma_i > 0 \;\&\; \sigma_i\text{XNOR}\sigma_j = 0 \\ J_{ij}\text{XOR}\sigma_i + 1, & \sigma_i < 0 \;\&\; \sigma_i\text{XNOR}\sigma_j = 0 \end{cases} \quad (5)$$

σ_i is mapped onto RWL; J_{ij} and σ_j are stored in the compute array, making it a mixed stationary design. σ_i is shared across a complete row with no requirement of bitline select, reading all columns, decreasing the column circuitry's complexity. Consider a 4-spin network with a target spin (σ_1) connected to spins $\sigma_{2\text{-}4}$. ICs are stored in a row within the compute array, and the required computes are $J_{12}*\sigma_2$, $J_{13}*\sigma_3$, and $J_{14}*\sigma_4$. From a memory design perspective, if we store J_{12}, J_{13}, and J_{14} in a row, we can perform $J_{12}*\sigma_i$, $J_{13}*\sigma_i$, $J_{14}*\sigma_i$, as all elements in a row share the same WL (mapped to σ_i). In the reuse-aware architecture, we map σ_i to be σ_1, enabling us to compute all dot product operations concurrently. However, it's crucial to verify whether $\sigma_1(\sigma_i)$ is the same as σ_2, σ_3, σ_4 to get the correct H_σ value.

The compute as illustrated in Fig. 13 proceeds as follows: In **phase 1**, the throughput of the SRAM array is as high as (N*R), as all XNOR computations of σ_i are done in parallel. This high throughput enables compute without the usage of XNOR queue. **Phase 2** performs shift and add logic for computing the XNOR dot products followed by the decision logic to select between the 4 options in eq.5 to compute the reuse aware equation of H_σ. In **phase 3**, adder performs the

Instruction	Primary Opcode (PO)	Secondary opcode(SO)	Usage
FIST (Repurposed x86)	0xDB	0x00	DRAM write
FIST(Repurposed x86)	0xDB	0x01	DRAM to storage array
FIST(Re-purposed x86)	0xDB	0x10	Storage to compute array
New XNORM DEST,[SRC1],[SRC2],BIT	0x30	-	In-memory XNOR

Fig. 14. **Software support - FIST in x86 with different secondary opcodes to indicate movement from DRAM to storage/compute array. New XNORM instruction with SRC1 = the storage array address that is mapped onto RWL, SRC2 = the compute array address, BIT = J_{ij} resolution and DEST = register for storing XNOR**

accumulation of partial dot products at once, to support the high throughput.

The time to compute H_σ is independent of R and N, implying O(1) compute and the target spin is used across the entire row, satisfying IC criterion with reuse of N*R.

E. Software support

The overhead in terms of additional compiler support is minimal for SACHI, as CPU assembly instructions can be repurposed to support SACHI because of the repurposable nature of SACHI. For instance, FIST (integer store), in x86 64-bit ISA, has a primary opcode (PO) of 0xDB without using a secondary opcode (SO). SO is used to indicate SACHI requests. A secondary opcode of 0x00 refers to DRAM write, 0x01 for DRAM to storage array write, 0x10 for transfer from storage to compute array. XNORM instruction (XNORM DEST,[SRC1],[SRC2], BIT), where SRC1= the address mapped onto RWL, SRC2= address of compute array, BIT = J_{ij} resolution, is added to perform PIM XNOR, followed by near-memory reuse-aware compute (Fig.14).

V. EXPERIMENTAL METHODOLOGY

1) ***Configuration****:* SACHI configuration incorporating (i) a compute array consisting of 16 tiles, each tile (size 10KB) capable of storing 100 spins and 8-bit ICs, (ii) a storage array (size 160KB) with 2 read ports (iii) digital peripheral logic sufficient to support compute array throughput is used for the experiments. The power/performance (PP) estimation takes into account data movement from the storage-compute array, in-memory compute and digital logic.

2) ***Benchmarks****:* Experiments use the following benchmarks, with the size of variables from 500-1M. We use the Ising formulation in [11] to model these benchmarks. In all these benchmarks, we formulate the problem as $H_\sigma = -\Sigma J_{ij}*\sigma_j$ or $H = -\Sigma J_{ij}*\sigma_i*\sigma_j$. This enables compute in all Ising machines, that can support dot-product of J_{ij} and σ_j. Therefore, no benchmark-dependent configuration of Ising machines is required, as long as the Ising machines are generic enough to compute the underlying graphs. These machines compute H_σ for all workloads, assuming a certain type of graph and initialized spins/ICs associated with the different workloads.

a) Asset allocation: Given m assets with $80M value, divide the assets (J_{ij} represents value) equally between 2 people. This is provided as an example of number partitioning in [11]. This problem can be formulated as checking if $H_\sigma = \Sigma J_{ij}*\sigma_j$ is zero. Here, σ_j=+1/-1 helps distinguish between the 2 people, J_{ij} signifies the value of each asset. This is a sparsely connected graph of number of spins equal to m.

b) Image segmentation: Given a densely connected image of m*n pixels, this benchmark identifies the max cut that splits the image into foreground and background (spin +1/-1), with J_{ij} (edge weight) indicating the difference in pixel values between neighbors. H_σ formulation is the same as that of formulation given for max cut in [1] [11]

c) Traveling salesman: Given a network of connected cities (spins), shown as a complete graph, this problem finds a route with total distance lesser than W between 2 given cities. The decision version of traveling salesman mentioned in [11] is used. This problem checks if $H = \Sigma J_{ij} * \sigma_i * \sigma_j < W$ between the 2 cities. J_{ij} is the distance, σ_i & σ_j = 1, implies a route between the two cities. This is achieved by solving $H_\sigma = -\Sigma J_{ij} * \sigma_j$ and checking if the associated H is lesser than W.

d) Molecular dynamics: Given a set of atoms in a molecule connected as King's graph, this identifies the atomic spin states in the lowest energy configuration.This is to show readers that J_{ij} in ferro-magnetism in eqn.1 is the force of attraction between 2 neighboring atoms [3].

3) ***SRAM/Data movement Power/Perf (PP) estimation****:* SRAM data array with peripheral circuits are designed using Cadence Virtuoso and the device parameters are modeled using FreePDK 45nm technology [24]. In-memory XNOR compute energy is measured when RWL is turned ON, and RBL is discharged. To measure RWL energy (pJ/bit), RWL under-driven approach [18] [21] with RWL capacitance of 50fF is used. To calculate the discharge energy of RBL (pJ/bit), 35fF RBL capacitance [18] is assumed for the SRAM array of 100 rows/columns (to match the size of a compute tile) for SRAM operating voltage of 1V. The compute latency of the SRAM array is found to be 2ns. The data movement energy for mapping RWL onto compute array is 1pJ/bit, assuming that the data movement energy is $\sim$800x the addition energy [15], with 100ns latency for storage to compute array movement. Furthermore, for applications whose storage requirement is greater than compute array size, (i) energy costs include data movement from DRAM to the storage array, SRAM write, and DRAM controller logic for prefetch (ii) performance cost includes SRAM write latency.

4) ***Digital logic PP estimation****:* Synthesis of digital circuits using Synopsis RTL Design Compiler is used to quantify power/energy having a cycle time of 5ns for 45nm technology node and operating voltage of 1V. We have assumed a clock cycle time of 5ns solely because of the slower standard-cell logic gates in 45nm technology node used for simulations. 45nm PDK was chosen because it is open-sourced. With performance scaling, as (technology node) as mentioned in [2], the cycle time of SACHI can match modern CPUs.

5) ***BRIM/Ising-CIM Comparison****:* To understand trade-offs between proposed and existing designs, factors include **(a)** storing input variables and ICs onto DRAM (necessary for BRIM/Ising-CIM/SACHI), **(b)** loading variables onto storage/compute arrays from DRAM (necessary for Ising-CIM/SACHI), data movement from DRAM to coupled oscil-

lator (necessary for BRIM) and **(c)**Ising compute. Across all designs, loading (both into DRAM(a) and from DRAM(b)) involves data movement at 64B per cycle, and the number of cycles depends on COP size. For example, a COP with 100 spins King's Graph, 8-bit J_{ij}, requires ∼13 cycles for storage onto DRAM, with a fixed loading energy cost of 1pJ/bit. The technology used is 45nm, cycle time of 5ns, and digital logic synthesis in 45nm to ensure a fair comparison across designs. **BRIM** evaluation considers Ising compute using a coupled oscillator, and associated digital logic/DAC. H compute in coupled oscillator+DAC takes 4-13 cycles per iteration. In the best case (used for comparison with SACHI), 1 cycle is needed for each of memory array read, DAC, and compute using coupling oscillator topology and annealing control. However, considering additional cycles for precharge, write into the memory array, along with a sequential DAC, a total of 13 cycles are needed for H compute. Typically, physical Ising machines can compute multiple spins in parallel. However, in BRIM, this is limited by 2 factors: i) The presence of storage capacitor introduces delays in capturing voltage changes, restricting spin initially at '0' to undergo fast transition from '0' to '1'. This is especially when input voltage transitions from its neighboring spins are abrupt, which is most likely the case. ii) Leakage through unconnected paths (spins that are unconnected to each other-similar to an unconnected cell in the case of DRAM), from passive capacitor is especially important when the node voltage is close to the trip-point of the ZIV diode. The power for coupled oscillator logic is 250mW for 2000 spins (100 neighbors per spin) and is proportional to the number of spins and neighbors. For a single 8-bit DAC, the power is ∼0.004mW, and there are 16 banks (1 DAC per bank) with associated digital logic consisting of 16:1 8-bit multiplexers (to map DAC outputs onto coupling units)/(16*8) flops per bank (storing the output of DAC). In contrast, SACHI has no DAC/associated digital logic, saving on performance and power. **Ising-CIM** compute involves XNOR compute in eDRAM, and annealing control. XNOR compute requires 3 cycles each for computing the updated spin values and performing the update, scaling with the number of spins and neighbors due to lack of parallelism. XNOR in eDRAM requires 1.2x power compared to 8T SRAM due to increased operating voltage. Annealing power is the same for all designs.

VI. RESULTS

*1) **Comparison with BRIM**:* BRIM is compared with SACHI using all benchmarks, assuming 1K spins and 4-bit interaction coefficients in (Fig.15).

a) Performance: SACHI(n_3) performs ∼36x better, for asset allocation as shown in Fig.15b. For large COPs like traveling salesman with high graph connectivity, SACHI(n_3) performs ∼300x better including the loading effect, as SACHI(n_3) offers high parallelism across neighbors per node, while BRIM does not. In benchmarks with lesser graph connectivity and loading overhead, like image segmentation and molecular

a)

Problem	Design's applicability			Reuse	
	BRIM	Ising-CIM	SACHI	SACHI vs BRIM	SACHI vs Ising-CIM
Asset allocation	Yes	No	Yes	4x	N/A
Image segmentation	Yes	No	Yes	200x	N/A
Traveling salesman	Yes	No	Yes	4000x	N/A
Molecular Dynamics	Yes	Yes	Yes	32x	64x

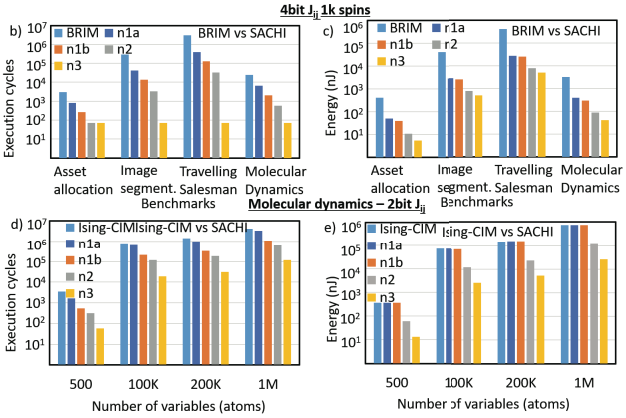

Fig. 15. **SACHI comparison with BRIM and Ising-CIM for reuse and designs applicability to different COPs. SACHI comparison with BRIM - b) Number of cycles c) total energy to solve COP including loading SACHI comparison with Ising-CIM - e) Number of cycles f) total energy to solve COP including loading**

dynamics, SACHI(n_3) performs ∼286x, ∼160x better as the opportunity for parallelism across neighbors is less.

b) Energy: SACHI(n_3) improves energy by ∼72x for asset allocation, ∼80x for image segmentation, and ∼79x for molecular dynamics as shown in Fig.15c. The need for greater data transfer for traveling salesman results in a slight reduction in energy efficiency (although still has 75x improvement). This is because the Ising graph's fully connected nature necessitates retrieval of more spins/ICs from DRAM/L2 into L1 cache. This increased data transfer incurs additional power consumption leading to a greater overall energy demand. The increased data movement requirement in traveling salesman leads to slightly degraded energy, but still ∼75x energy improvement (Fig.15c).

The major reason for SACHI's superior performance is its increased reuse. In BRIM, reuse is 1, as one IC fetched from memory is used in 1 H compute in the coupling units in BRIM. The reuse in SACHI(n_3) is ∼200x for image segmentation, ∼4000x for travelling salesman, ∼32x for molecular dynamics (tabulated in Fig.15a).

*2) **Comparison with Ising-CIM**:* Ising-CIM is limited to King's Graph with 2-bit unsigned ICs. Hence we restrict our comparison to 2-bit molecular dynamics COP.

a) Performance: SACHI(n_3) performs ∼70x/∼80x better for 500/1M atoms (including loading) (Fig.15d). This is because of the parallelism across neighbors and J_{ij} resolution in SACHI(n_3), while Ising-CIM does not have such parallelism.

b) Energy: SACHI(n_{1a})/SACHI(n_{1b}) have the same energy because the increased performance in SACHI(n_{1b}) is compensated by decreased power in SACHI(n_{1a}). The energy improvement in SACHI(n_3) for 500/1M spins is ∼40x/∼75x respectively (Fig.15e) because of the increased voltage requirement for eDRAM and reduced performance in Ising-CIM.

The maximum reuse is 1 in Ising-CIM, as every bit of IC is used only in 1 H_σ compute performed in the eDRAM compute array. This leads to SACHI(n_3) providing ∼16x more reuse in molecular dynamics (Fig.15a tabulates the reuse factors).

*3) **Comparison with Other Optimized Solvers**:* SACHI's performance in comparison to solutions implemented using Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and dedicated Optimized Solvers is of interest. GA/PSO is implemented on an Intel i5-8265U CPU running at 3.9GHz. For the GA implementation, GAlib [31] is used. The solution quality is measured by comparing GA's accuracy to SACHI's 100% accuracy. GA's accuracy is lower, as shown in (Fig.16a). This could be attributed to GA's global-only search for selecting the best candidates in each generation. In contrast, Ising/PSO performs updates based on neighbors, resulting in faster convergence (Fig.16b). In PSO, the selection criterion considers personal best (pbest) and global best (gbest) for all candidates, where pbest is compared against gbest at the end of each iteration to update the fitness. The comparison with dedicated optimized solvers/algorithms (OPTSolv) for benchmarks such as Concorde solver for traveling salesman, Ford-Fulkerson network flow for image segmentation, LAMMPS [29] for molecular dynamics, is also presented in Fig.16. SACHI outperforms these solvers by 27-34x because of parallelism across N and R from reuse-aware near-memory compute.

*4) **Scalability**:* SACHI is highly scalable for graphs with diverse connectivity, accommodating various sizes without limitations. Fig.17 illustrates its performance as CPI (clock cycles per Hamiltonian iteration) for spin counts ranging from 500 to 1M, showcasing the impact of compute array overflow in each scenario. For instance, in the case of (a) 500 spins: Spins fit inside the compute array for all SACHI designs, (b) 200K spins: Spins fit inside the compute array for all SACHI designs except SACHI(n_3), (c) 300K spins: Spins do not fit inside the compute array for SACHI(n_2) and SACHI(n_3), (d) 1M spins: Spins do not fit inside the compute array for all SACHI designs. A few noteworthy points are as follows: (i) SACHI(n_3) demonstrates the highest performance across all workloads due to the parallel compute of all spin neighbors and IC bits. (ii) SACHI(n_2) and SACHI(n_3) share the same CPI (as there is 1 neighbor/spin), except for 200K spins in asset allocation where SACHI(n_2) outperforms SACHI(n_3) due to SRAM write latency, as SACHI(n_3) has a filled compute array and needs rewriting for the next round of compute. (iii) SACHI(n_{1a}) shows moderate performance due to inefficient load balancing among compute tiles with adjacent spins in the same tile. SACHI(n_{1b}) resolves this issue by re-organizing neighboring spins across different tiles. (iv) Traveling salesman benchmark has the highest CPI, primarily due to the high connectivity of the underlying complete graph, affecting SACHI(n_1) and SACHI(n_2) designs, whose performance depends on the number of neighbors per spin. (v) For many COPs, 1M spins are large enough, but for image segmentation, 2M pixels (HD video) and 8M pixels (UHD video) cases were studied. They take 10^9 and $2*10^{10}$ CPI, respectively.

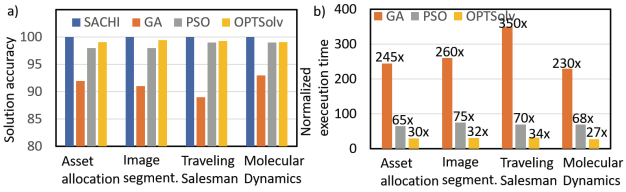

Fig. 16. **Comparison with GA/PSO/optimized solvers (OPTSolv) - Solution accuracy, normalized execution time wrt SACHI for asset allocation, image segmentation, traveling salesman and molecular dynamics**

*5) **Reconfigurability**:* Reconfigurability is shown by using SACHI for different J_{ij} resolutions, configured based on COP. Fig.18a-d shows the sensitivity of CPI for 1M spins and J_{ij} resolution from 2 to 8 bits. SACHI(n_2) and SACHI(n_3) show no change in CPI, as the performance is independent of J_{ij} resolution. SACHI(n_{1a}), SACHI(n_{1b}) show performance improvement with reduced resolution, as there are fewer in-memory XNOR computes.

*6) **Time to solution/Solution quality**:* The time to solution depends on the number of iterations needed to converge at a solution, CPI, and the clock period. The solution convergence is reached when the Hamiltonian energy (H) remains unchanged, and simulated annealing cannot further reduce the energy. Fig. 19a) illustrates the H change with iteration number for 1M assets in the asset allocation COP. Simulated annealing's contribution to the overall execution time is less than 1%, enhancing accuracy by 0.8%. It is implemented by probabilistically flipping based on the Metropolis acceptance criterion [35], comparing likelihood against a predefined value within the annealer block (Sec.4). The number of iterations across SACHI designs is the same, as they all arrive at the same H at the end of each iteration. The execution/annealing time is the highest for traveling salesman COP because of the high degree of graph connectivity. Asset allocation requires the fewest iterations, followed by molecular dynamics, owing to limited graph connectivity (Fig.19b). To assess the accuracy loss due to resolution reduction, the change in the number of iterations needed to converge at a solution with J_{ij} resolution is observed. An 8-bit J_{ij} resolution strikes the best balance between reduced memory footprint requirement and loss in

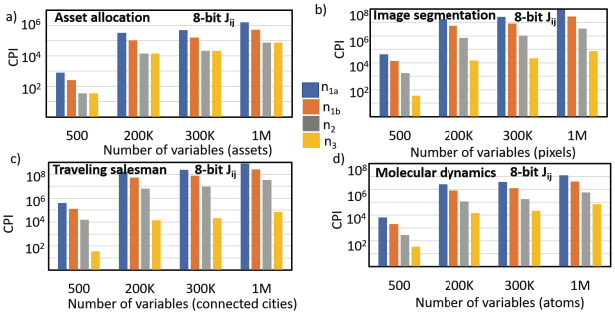

Fig. 17. **Scalability - Cycles Per Hamiltonian iteration (I) with increasing variable size for a) Asset allocation b) Image segmentation c) Traveling salesman and d) Molecular dynamics COPs showing SACHI's scalability to different large variable count COPs**

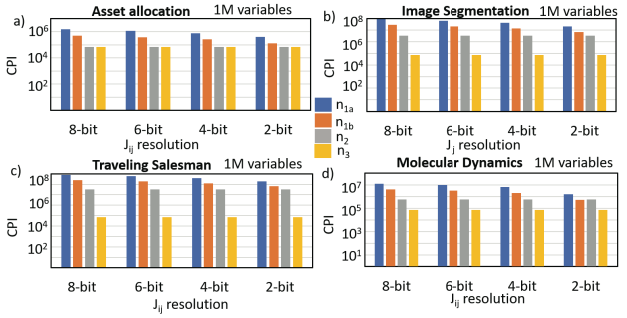

Fig. 18. **Reconfigurability - Cycles Per Iteration (CPI, i.e. Hamiltonian iterations) with decreasing IC resolution: a) Asset allocation b) Image segmentation c) Traveling salesman d) Molecular dynamics showing the feasibility of SACHI being reconfigurable to different IC resolution**

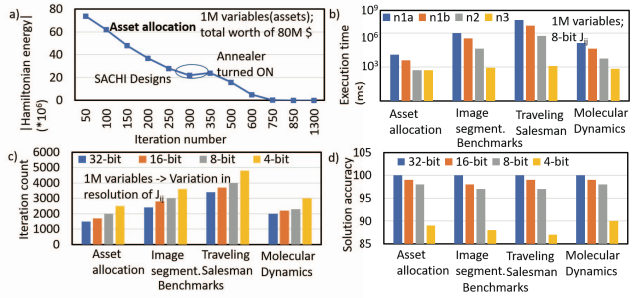

Fig. 19. **Time to solution/solution accuracy - a) Hamiltonian energy (H) reduction with increasing iteration count for Asset allocation, using simulated annealing (SA) to move out of local minima. b) Solution time improvement from SACHI(n_1) to SACHI(n_3) architecture due to reuse aware compute c) Increased iteration count with lower resolution for IC due to frequent requirement of SA steps d) Solution accuracy degradation with lower IC resolution under iso-performance condition**

accuracy. Reducing the resolution below 8-bit (e.g., 4-bit) leads to a sharp increase in the number of iterations (Fig.19c). For 32-bit resolution, the number of iterations is the lowest, as accurate compute decreases the chances of being trapped in locally optimal solutions and reduces the need for simulated annealing iterations.Fig.19d) shows the solution accuracy for lower J_{ij} resolutions once the 32-bit J_{ij} has converged at the solution. While 4-bit reduces accuracy below 90%, 8-bit retains accuracy with a smaller memory footprint than 32-bit.

VII. Discussion

*1) **Impact on conventional workloads**:* SACHI does not detrimentally affect other workloads. The L1 datapath is split as (a) fill into L1 from L2 and (b) read from L1. The fill datapath does not get impacted at all. The read from L1 cache is not affected because (i) we do not make modifications to the memory array for PIM (L1 caches in modern processors are already 8T) (ii) although there is an additional 2:1 multiplexer delay in the L1 controller/periphery to choose between normal and compute mode of L1 cache, that can be easily retimed and absorbed by synthesis tools. Furthermore, the additional near-memory peripheral logic does not intervene in normal operation as this is a separate datapath.

*2) **Impact of increased L1/L2 cache size**:* In our simulations, we have assumed L1/L2 cache sizes of 10KB/160KB because SRAM bit-cell parameters are designed in Virtuoso for the same memory capacity. However, in modern CPUs, the typical L1 cache ranges from 64KB to 256KB, while L2 cache spans from 1MB to 8MB. By increasing the cache size, SACHI's performance can be further enhanced due to several factors: (i) increased parallelism across neighbors (N) and resolution (R) can be achieved (ii) higher resolution for ICs improves solution time and accuracy. (iii) the larger L1 cache can better accommodate large-sized COPs such as the traveling salesman problem. Experiments show that for the traveling salesman problem with 1M spins, a 64KB/1MB cache offers 5x/8x better performance/energy compared to the 10KB/160KB configuration. These improvements lead to 16x performance/20x energy improvement for the 256KB/8MB cache with 1M spins. It is noteworthy that performance does not degrade for any benchmark. Though increasing cache size leads to a slight rise in power due to larger RBL/RWL capacitance in larger arrays, this is outweighed by the higher performance, resulting in overall energy gain.

*3) **Additional software details**:* We assume that the cache operates in a single mode at a time, either compute/normal mode. The mode switch can be achieved by programming a special-purpose register. Additionally, ongoing work includes (i) developing a CUDA-like library/API to program SACHI as part of a complete program, (ii) extending the library to support Ising formulation of COPs, similar to the OpenQASM framework [5], and (iii) enabling the interruption of SACHI, storing contexts, ASIDs, and minimal payload in TLBs to facilitate rapid page translation during a context switch between modes (but these are outside the scope of this paper).

VIII. Conclusion

We presented SACHI, an iterative-compute based all-digital Ising architecture that employs reuse-aware stationarity schemes for Ising spins and interaction coefficients. It combines elements of re-purposability, scalability, reconfigurability, and reuse-aware data-stationary near-memory compute to achieve 160x/90x better performance, and improved energy of 79x/75x with better reuse of 32x/16x for molecular dynamics COP over BRIM/Ising-CIM, and 27x-34x over dedicated optimized solvers for different COPs.

IX. Acknowledgements

Authors would like to thank the Department of Electrical and Computer Engineering, University of Texas at Austin for their support. Furthermore, special thanks to Kavya for shaping/discussing the work during the ideation phase, and proofreading the article to further provide key suggestions.

References

[1] R. Afoakwa, Y. Zhang, U. K. R. Vengalam, Z. Ignjatovic, and M. Huang, "Brim: Bistable resistively-coupled ising machine," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2021, pp. 749–760.

[2] A. Biswas and A. P. Chandrakasan, "Conv-sram: An energy-efficient sram with in-memory dot-product computation for low-power convolutional neural networks," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 1, pp. 217–230, 2019.

[3] S. G. Brush, "History of the lenz-ising model," *Reviews of modern physics*, vol. 39, no. 4, p. 883, 1967.

[4] B. A. Cipra, "An introduction to the ising model," *The American Mathematical Monthly*, vol. 94, no. 10, pp. 937–959, 1987.

[5] A. W. Cross, L. S. Bishop, J. A. Smolin, and J. M. Gambetta, "Open quantum assembly language," *arXiv preprint arXiv:1707.03429*, 2017.

[6] M. M. Flood, "The traveling-salesman problem," *Operations research*, vol. 4, no. 1, pp. 61–75, 1956.

[7] R. J. Glauber, "Time-dependent statistics of the ising model," *Journal of mathematical physics*, vol. 4, no. 2, pp. 294–307, 1963.

[8] R. Hamerly, T. Inagaki, P. L. McMahon, D. Venturelli, A. Marandi, T. Onodera, E. Ng, C. Langrock, K. Inaba, T. Honjo *et al.*, "Experimental investigation of performance differences between coherent ising machines and a quantum annealer," *Science advances*, vol. 5, no. 5, p. eaau0823, 2019.

[9] M. W. Johnson, M. H. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk *et al.*, "Quantum annealing with manufactured spins," *Nature*, vol. 473, no. 7346, pp. 194–198, 2011.

[10] K. P. Kalinin, A. Amo, J. Bloch, and N. G. Berloff, "Polaritonic xy-ising machine," *Nanophotonics*, vol. 9, no. 13, pp. 4127–4138, 2020.

[11] A. Lucas, "Ising formulations of many NP problems," *Frontiers in Physics*, vol. 2, 2014. [Online]. Available: https://doi.org/10.3389%2Ffphy.2014.00005

[12] A. Marandi, Z. Wang, K. Takata, R. L. Byer, and Y. Yamamoto, "Network of time-multiplexed optical parametric oscillators as a coherent ising machine," *Nature Photonics*, vol. 8, no. 12, pp. 937–942, 2014.

[13] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 7, pp. 3523–3542, 2021.

[14] S. Mirjalili and S. Mirjalili, "Genetic algorithm," *Evolutionary Algorithms and Neural Networks: Theory and Applications*, pp. 43–55, 2019.

[15] O. Mutlu, S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun, "A modern primer on processing in memory," *arXiv preprint arXiv:2012.03112*, 2020.

[16] G. F. Newell and E. W. Montroll, "On the theory of the ising model of ferromagnetism," *Reviews of Modern Physics*, vol. 25, no. 2, p. 353, 1953.

[17] S. S. T. Nibhanupudi, S. R. Sundara Raman, M. Cassé, L. Hutin, and J. P. Kulkarni, "Ultra-low-voltage utbb-soi-based, pseudo-static storage circuits for cryogenic cmos applications," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 7, no. 2, pp. 201–208, 2021.

[18] S. T. Nibhanupudi, S. R. S. Raman, and J. P. Kulkarni, "Phase transition material-assisted low-power sram design," *IEEE Transactions on Electron Devices*, vol. 68, no. 5, pp. 2281–2288, 2021.

[19] S. Okada, M. Ohzeki, M. Terabe, and S. Taguchi, "Improving solutions by embedding larger subproblems in a d-wave quantum annealer," *Scientific reports*, vol. 9, no. 1, pp. 1–10, 2019.

[20] D. Pierangeli, G. Marcucci, and C. Conti, "Large-scale photonic ising machine by spatial light modulation," *Physical review letters*, vol. 122, no. 21, p. 213902, 2019.

[21] S. R. S. Raman, S. S. T. Nibhanupudi, A. K. Saha, S. Gupta, and J. P. Kulkarni, "Threshold selector and capacitive coupled assist techniques for write voltage reduction in metal–ferroelectric–metal field-effect transistor," *IEEE Transactions on Electron Devices*, vol. 68, no. 12, pp. 6132–6138, 2021.

[22] S. R. S. Raman, S. Xie, and J. P.Kulkarni, "Compute-in-edram with backend integrated indium gallium zinc oxide transistors," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2021, pp. 1–5.

[23] A. Sharma, R. Afoakwa, Z. Ignjatovic, and M. Huang, "Increasing ising machine capacity with multi-chip architectures," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, 2022, pp. 508–521.

[24] J. E. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. R. Davis, P. D. Franzon, M. Bucher, S. Basavarajaiah, J. Oh *et al.*, "Freepdk: An open-source variation-aware design kit," in *2007 IEEE international conference on Microelectronic Systems Education (MSE'07)*. IEEE, 2007, pp. 173–174.

[25] Y. Su, H. Kim, and B. Kim, "31.2 cim-spin: A 0.5-to-1.2v scalable annealing processor using digital compute-in-memory spin operators and register-based spins for combinatorial optimization problems," in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, 2020, pp. 480–482.

[26] S. R. Sundara Raman, S. S. T. Nibhanupudi, and J. P. Kulkarni, "Enabling in-memory computations in non-volatile sram designs," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 12, no. 2, pp. 557–568, 2022.

[27] S. R. Sundara Raman, S. Xie, and J. P. Kulkarni, "Igzo cim: Enabling in-memory computations using multilevel capacitorless indium–gallium–zinc–oxide-based embedded dram technology," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 8, no. 1, pp. 35–43, 2022.

[28] K. Tanahashi, S. Takayanagi, T. Motohashi, and S. Tanaka, "Application of ising machines and a software development for ising machines," *Journal of the Physical Society of Japan*, vol. 88, no. 6, p. 061010, 2019.

[29] A. P. Thompson, H. M. Aktulga, R. Berger, D. S. Bolintineanu, W. M. Brown, P. S. Crozier, P. J. in 't Veld, A. Kohlmeyer, S. G. Moore, T. D. Nguyen, R. Shan, M. J. Stevens, J. Tranchida, C. Trott, and S. J. Plimpton, "LAMMPS - a flexible simulation tool for particle-based materials modeling at the atomic, meso, and continuum scales," *Comp. Phys. Comm.*, vol. 271, p. 108171, 2022.

[30] S. Vangal, S. Paul, S. Hsu, A. Agarwal, S. Kumar, R. Krishnamurthy, H. Krishnamurthy, J. Tschanz, V. De, and C. H. Kim, "Wide-range many-core soc design in scaled cmos: Challenges and opportunities," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 5, pp. 843–856, 2021.

[31] M. Wall, "Galib: A c++ library of genetic algorithm components," *Mechanical Engineering Department, Massachusetts Institute of Technology*, vol. 87, p. 54, 1996.

[32] T. Wang and J. Roychowdhury, "Oscillator-based ising machine," *arXiv preprint arXiv:1709.08102*, 2017.

[33] T. Wang and J. Roychowdhury, "Oim: Oscillator-based ising machines for solving combinatorial optimisation problems," in *Unconventional Computation and Natural Computation: 18th International Conference, UCNC 2019, Tokyo, Japan, June 3–7, 2019, Proceedings 18*. Springer, 2019, pp. 232–256.

[34] T. Wang, L. Wu, and J. Roychowdhury, "New computational results and hardware prototypes for oscillator-based ising machines," in *Proceedings of the 56th Annual Design Automation Conference 2019*, 2019, pp. 1–2.

[35] S. Xie, S. R. S. Raman, C. Ni, M. Wang, M. Yang, and J. P. Kulkarni, "Ising-cim: A reconfigurable and scalable compute within memory analog ising accelerator for solving combinatorial optimization problems," *IEEE Journal of Solid-State Circuits*, pp. 1–13, 2022.

[36] M. Yamaoka, C. Yoshimura, M. Hayashi, T. Okuyama, H. Aoki, and H. Mizuno, "A 20k-spin ising chip to solve combinatorial optimization problems with cmos annealing," *IEEE Journal of Solid-State Circuits*, vol. 51, no. 1, pp. 303–309, 2016.