SOPHIE: A Scalable Recurrent Ising Machine Using Optically Addressed Phase Change Memory

Guowei Yang*, Sina Karimi*, Carlos A. Ríos Ocampo†, Ayse K. Coskun*, Ajay Joshi*
*Boston University, †University of Maryland, College Park
{guoweiy, sikarimi, acoskun, joshi}@bu.edu, riosc@umd.edu

***Abstract*—Ising problems are nondeterministic-polynomial-hard (NP-hard) problems prevalent in various domains, such as statistical physics, circuit design, and machine learning. They pose significant challenges for traditional algorithms and architectures. Researchers have recently developed nature-inspired Ising machines to tackle these optimization problems efficiently. Many optimization problems can be mapped to the Ising model, and physical laws will drive the Ising machine towards the solution. However, existing Ising machines suffer from scalability issues, i.e., performance drops when problem sizes exceed their physical capacity.**

In this paper, we propose SOPHIE, a Scalable Optical PHase-change memory (OPCM) based Ising Engine. SOPHIE integrates architectural, algorithmic, and device optimizations to address scalability challenges in Ising machines. We architect SOPHIE using 2.5D integration, where we integrate a controller chiplet, a DRAM chiplet, laser sources, and multiple OPCM chiplets. SOPHIE utilizes OPCMs to perform matrix-vector multiplications efficiently. Our symmetric tile mapping at the architecture level reduces approximately half of the OPCM array area, enhancing the scalability of SOPHIE. We use algorithmic optimizations to efficiently handle large problems that cannot fit within hardware constraints. Specifically, we adopt a symmetric local update technique and a stochastic global synchronization strategy. These two algorithmic approaches decompose large problems into isolated tiles, reduce computation requirements, and minimize communication in SOPHIE. We apply device-level optimizations to adopt the modified algorithm. These device-level optimizations include employing bi-directional OPCM arrays and dual-precision analog-to-digital converters. SOPHIE is $3\times$ faster than the state-of-the-art photonic Ising machines on small graphs and $125\times$ faster than the FPGA-based designs on large problems. SOPHIE alleviates the hardware capacity constraints, offering a scalable and efficient alternative for solving Ising problems.

***Index Terms*—optical computing, phase change memory, Ising machine, processing-in-memory**

I. Introduction

Combinatorial optimization problems pervade numerous domains, such as network routing, scheduling, and circuit design [1]–[3]. These problems are typically nondeterministic-polynomial (NP) hard. Thus, conventional von Neumann architectures often fail to provide efficient solutions as the scale and complexity of these optimization problems grow [4]. Even after applying heuristic approximation approaches, the computation time for combinatorial optimization is still long because it takes many iterations to reach the answer. For example, the max-cut problem would take up to 10 minutes for a small graph with 3000 nodes [5]. Consequently, researchers have been exploring non-conventional architectures to solve combinatorial optimization problems more efficiently.

Ising machines stand out as promising non-von Neumann architectures for solving combinatorial optimization problems. By mapping combinatorial optimization onto Ising machines, these problems can be represented as Ising models. The Ising model, originating from statistical mechanics, provides a framework for understanding the behavior of spins within a system. The Ising machine seeks to minimize the energy of the Ising model, offering high-performance and energy-efficient solutions to the combinatorial problems mapped to the Ising model.

Existing implementations of Ising machines can be categorized into machines implementing physics-based coupling and those implementing computation-based coupling. Physics-based Ising machines encode couplings into physical connections. Examples of this type of machine include D-Wave quantum annealers [6], electronic Oscillator-based Ising Machines (OIM) [7], and Bistable Resistively-coupled Ising Machines (BRIM) [8]. While these machines deliver notable speedup and energy efficiency compared to conventional architectures, they show a significant speedup drop when the problem mapped to them is larger than the hardware capacity of the machine.

Computation-based Ising machines simulate node connections through computations like matrix multiplication, offering enhanced support for graphs with dense connections. Examples of these machines include Coherent Ising Machines (CIM) [9] and Integrated Nanophotonic Recurrent Ising Sampler (INPRIS) [4]. Computation-based Ising machines also suffer from significant performance degradation for problems larger than their capacity because of glue computation and communication overhead. Additionally, computation-based Ising algorithms are memory-bound on traditional electronic devices like GPUs [10], and moving the coupling coefficients between memory and compute leads to significant time and energy overheads. However, computation-based Ising machines are not limited by physical constraints of problem size. Therefore, they are easier to scale to solve larger problems.

In this work, we introduce SOPHIE, a Scalable Optical PHase-change memory (OPCM) based Ising Engine. SOPHIE is the first Ising machine employing OPCM and uses a cross-layer approach that combines algorithm, device, and architecture-level optimizations to simultaneously achieve scalability and speedup. A key design choice in

This research was partially funded by the NSF CCF 2131127 grant.

SOPHIE is using OPCM, which has emerged as a promising processing-in-memory (PIM) device [11]. By storing matrices within an array of phase change material (PCM) cells and leveraging photonic signals for reading and writing the cells, OPCM offers substantial advantages over traditional electronic devices in terms of performance and energy efficiency for Matrix-Vector Multiplication (MVM) [11]. OPCMs feature ultra-compact footprints (< 10 µm), yet strong amplitude modulations, in a multilevel manner, which allows to codify information in the optical transmission of a photonic waveguide [12], [13]. These features are an advantage over conventional thermo-optical (TO) and electro-optical (EO) modulators, whose weaker optical modulation makes photonic MVM units, such as Mach-Zehnder interferometer (MZI) and micro-ring resonator (MRR) arrays, too large and difficult to scale [14]. Moreover, OPCMs' behavior is nonvolatile owing to the stable amorphous and crystalline states of PCMs, making them, in addition, a more energy-efficient computing device over volatile TO and EO modulators. The combination of nonvolatility and strong amplitude modulations in a multilevel manner enables PIM capability. This means matrix elements can be stored within PCM cells, and computations can be performed directly on them. PIM capability diminishes the time and energy required to transfer large coupling matrices, rendering OPCM a prime choice for computation-based Ising machines.

While OPCMs can improve the performance and efficiency of computation-based Ising machines, scalability still remains a critical concern, motivating a cross-layer algorithm, device, and architecture design. When supporting a graph with n nodes, OPCM must accommodate the $n \times n$ adjacency matrix. Since each OPCM cell occupies a 30×30 µm^2 area, and real-world graphs may contain more than tens of thousands of nodes, it is impossible to store such a large graph in OPCM. Therefore, SOPHIE breaks down the large matrix into smaller tiles and distributes them across multiple OPCM arrays. To avoid creating bottlenecks in inter-tile communication, we introduce a novel symmetric local update technique for the Ising algorithm at the algorithm level. By isolating most of the iterations within local tiles, our technique minimizes inter-tile communication with minimal impact on solution quality. Additionally, we propose stochastic global iteration to further reduce communication overhead among tiles. At the device level, we configure the OPCM array and its electrical peripherals to align with the requirements of the enhanced algorithm. We employ dual-precision analog-to-digital converters (ADCs) to improve the computation's performance and energy efficiency. We also use bi-directional OPCM arrays to support transposed MVM operations. At the architecture level, SOPHIE employs symmetric tile mapping to reduce the accelerator's area and enhance its scalability.

We implement our enhanced Ising algorithm in Python for functional simulation, which enables testing the solution quality and determining the number of iterations required for convergence. We build custom models to calculate the power, performance, and area (PPA) results of our accelerator.

Our specific contributions are summarized as follows:

- OPCM-based Ising Machine: We develop SOPHIE, a computation-based scalable Ising machine employing OPCMs. OPCMs can perform MVM efficiently, which is suited for computation-based Ising machines. The OPCM array and its electrical peripherals are customized to line up with the enhanced algorithm. These customizations include the utilization of dual-precision ADCs and the ability to perform transposed MVMs.
- Symmetric Local Update Technique: We introduce a symmetric local update technique for the existing recurrent Ising machine algorithm [15]. This method partitions the recurrent computation on a large matrix into smaller tiles, allowing most computations to be confined within symmetric tile pairs. This approach significantly reduces inter-tile communication, which is a major bottleneck of the scalability for larger graphs.
- Stochastic Global Iteration: We present a stochastic global iteration technique that further reduces the overall computation demand and inter-tile communication while maintaining acceptable solution quality. This approach speeds up the computation, especially on large graphs.
- Symmetric Tile Mapping: The coupling matrix of an Ising problem is symmetric and can be decomposed into tiles. We map a pair of symmetric tiles to one OPCM array to save approximately half of the array area. This approach makes the design more scalable, especially for large graphs.

We evaluate SOPHIE using small and large graphs from GSET and K-graphs generated by Rudy graph generator [16]. Our solution can reduce half of the OPCM area and reduce $25\% - 50\%$ of the computation demand and global synchronization overhead while maintaining acceptable solution quality. SOPHIE demonstrates $3\times$ speedup against state-of-the-art photonic Ising machines on small graphs and $125\times$ speedup against FPGA-based designs on large problems. SOPHIE alleviates the hardware capacity constraints, offering a scalable and efficient alternative for addressing Ising problems.

II. Background

In this section, we start by explaining the principles behind OPCM cells. Next, we will discuss Ising machines, outlining their types and characteristics. Finally, we present photonic recurrent Ising sampler (PRIS) algorithm [15], one of the proposed methods for probabilistically finding the ground state of an arbitrary Ising problem, which we use as a reference point.

A. OPCM Background

An OPCM cell consists of a part of an optical waveguide with cladding-embedded $Ge_2Sb_2Te_5$ (GST), as shown in Figure 1. The OPCM cell, also known as the GST cell, can switch reversibly between the non-volatile fully amorphous and fully crystalline states using heat stimuli. Interestingly, intermediate states can also be achieved by controlling the ratio between

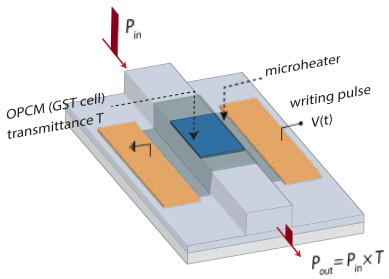

Fig. 1. Electrically programmed and optically addressed phase change memory, also illustrating the in-memory processing concept: multiplication between the input pulse amplitude and the OPCM transmittance.

amorphous and crystalline domains, thus achieving a semi-continuous modulation [17]. Given that the crystalline state is significantly more absorptive than the amorphous, each phase state in the material translates into a unique transmittance level of the waveguide, which is also non-volatile. Multi-bit data can thus be encoded in the transmittance of each OPCM cell, whose precise and reliable writing becomes crucial.

Writing on the GST cell, i.e., changing the phase state of the cell, is done with a transition-triggering heat stimulus either using electrical or optical pulses. Electrical switching employs a variety of waveguide-embedded microheaters to induce the reversible phase transition via Joule heating [18]. Previous work has shown switching energies as low as 5.55 nJ and 860.71 nJ to amorphize and crystallize, respectively, using graphene microheaters [19]. Optical switching, on the other hand, uses the self-heating of the GST due to the power it absorbs from pump optical pulses within the waveguide. This method consumes less energy at the GST cell than electrical switching: 460 pJ for amorphization and 140 pJ for crystallization [17]. However, this method is challenging to scale for large architectures due to the difficult on-chip routing of pump pulses, which requires energy-hungry and complex switching networks to reach each individual cell [20]. An alternative solution is to use a dedicated grating for out-of-chip coupling into each GST cell. However, this solution requires a larger footprint to accommodate the extra grating couplers ($\approx 20\times20$ μm^2 to 50×50 μm^2 each) and complicated experimental setups [12]. Even though electrical switching consumes more power than optical switching, features like scalability, compatibility with microelectronics, and form factor make electrical switching a better solution.

State-of-the-art experiments have impressively reached up to 64 deterministic states within a single GST cell, enabling us to store up to 6 bits per cell [21]. The multilevel response of GST cells can, in turn, be exploited to perform in-memory analog computation [22]. Scalar-scalar multiplication could be implemented by mapping a multiplicand to the GST transmittance and multiplier to the amplitude of the input pulse, as shown in Figure 1. The free propagation of the pulse through the GST cell renders the product of multiplicand and multiplier at the output of the waveguide. When many GST cells are cascaded in a cross-bar-like array, MVM can be achieved, which is precisely the MVM approach we adopt here [12].

B. Ising Machines

Basics: The Ising model describes the Hamiltonian of a system of a set of spins that are coupled together. Each spin can take a value of -1 or $+1$ ($\sigma_{1\leq i\leq N} \in \{-1, +1\}$). The interaction between spins is described by matrix K, which is a $N \times N$ symmetric matrix. In the absence of an external magnetic field, the resulting Hamiltonian is:

$$H = -\frac{1}{2} \sum_{1\leq i,j\leq N} \sigma_i K_{ij} \sigma_j \tag{1}$$

The Ising problem consists of finding the ground state of the spins coupled by matrix K of a quadratic Hamiltonian H (Equation 1). Previous work has shown that many combinatorial problems can be reduced to an Ising problem [23]. Hence, any solution for finding the ground state of an arbitrary Ising model, which is an NP-hard problem, is also applicable in other optimization problems as long as that can be reduced to the Ising model.

There are no known algorithms that can find the exact ground state of an arbitrary Ising model in polynomial time. Therefore, researchers have employed heuristic and meta-heuristic methods to approximate a solution near the ground state [24]. Moreover, a physical system governed by the same Hamiltonian inherently moves towards lower-energy states. To find a solution to the Ising model, one can implement the Ising machines either through physical systems or heuristic algorithms. Remarkably, Ising machines exhibit significant computational efficiency (measured in terms of time to solution and energy consumption), up to six orders of magnitude higher than conventional systems when addressing problems of comparable scale [5], [8]. By leveraging the mapping of combinatorial problems to Ising problems and employing Ising machines to solve these Ising problems, we can achieve significantly faster and more efficient solutions to combinatorial problems.

Solving max-cut problems using Ising Machine: An excellent example of combinatorial problems, readily mappable to Ising problems, is the max-cut problem. Figure 2 shows a max-cut problem solved by mapping the problem to an Ising problem. Given an undirected graph, the max-cut problem is to partition the nodes of the graph into two subsets so as to maximize the total weight of the edges between the two subsets [5]. Each node of the graph can be mapped to an Ising spin, and each edge between two nodes represents the coupling between the two spins. The state of the spin, either $+1$ or -1, represents the subset where the node belongs. According to the Ising model, the state of the spins will move towards the ground state, which minimizes the energy of the Ising model. The ground state of the Ising model is exactly the solution to the max-cut problem. For example, Figure 2(a) shows an undirected graph with five nodes initialized with random states. Here, nodes with the color black have spins with a value of -1, and nodes with the color white have spins with a value of $+1$. Suppose the Ising machine eventually reaches the ground state (either by using a physical system or

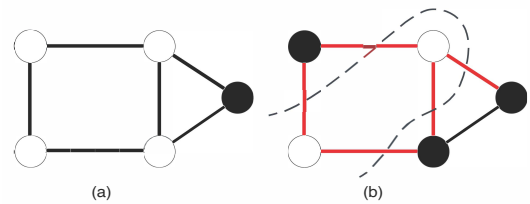

Fig. 2. Solving a max-cut problem using the Ising model. (a) Each node is mapped to a spin. Spins are coupled according to the graph's adjacency matrix. Nodes in black indicate spins with a value -1, and nodes in white indicate spins with a value $+1$. (b) After the Ising model reaches its ground state, the spins configuration represents the solution to the max-cut problem. Each node with the same spin value will be placed in the same subset. Edges connecting these two subsets are the solution to the max-cut problem.

heuristic algorithm) shown in Figure 2(b). Here, the solution to max-cut could be interpreted by putting spins with the same value in the same subset, e.g., white nodes ($+1$ spins) in one subset and black nodes (-1 spins) in another. The total weight of the edges connecting those two subsets is then maximized. Many optimization problems can be converted to the max-cut problem and then solved by Ising machines [25].

Types of Ising machines: Ising machines can be categorized into two classes: physics-based Ising machines capable of implementing and sampling an Ising model, and computation-based Ising machines capable of implementing heuristic and meta-heuristic methods.

Physics-based Ising machines implement a physical system of spins and offer an interface for programming the coupling properties of the Ising model. Physical systems governed by such Hamiltonian (as in Equation 1) naturally move toward low-energy states. One of the well-known Ising machines is quantum annealers, such as D-Wave [26]. The underlying theory is inspired by adiabatic quantum computation. However, many challenges come with quantum annealing Ising machines. First, it requires cryogenic operating conditions, which results in excessive power consumption (16 kW for D-Wave's 2000 qubit system [26]). Second, due to physical coupling constraints, the number of nodes supported by this device is very limited, especially for dense graphs. Finally, reaching the ground state is not guaranteed as a result of relaxing adiabatic conditions in the adiabatic quantum annealing computation. Another implementation of physical-based Ising machines is the Bistable Resistively-coupled Ising Machine (BRIM) [8]. In this design, each spin is represented as the polarity of a capacitor. The coupling between each node is represented as the conductance between each capacitor. Strong coupling means high conductance, which enables connected nodes to equilibrate more easily, while weak coupling means low conductance, allowing less interaction between the two capacitors. The sign of the coupling is supported by connecting the capacitors' same or the opposite polarity. BRIM shows significant improvement in terms of performance, energy, and area compared to previous designs. However, BRIM would have significant performance degradation for problems beyond its hardware capacity. Although some work has been done to address this problem using multiple chips to create a larger Ising machine [27], it would still require storing all the spins on the chips.

Computation-based Ising machines implement the nodes and coupling between them by computation such as matrix multiplication. These Ising machines use heuristic and meta-heuristic algorithms to achieve an approximate answer as fast as possible. Photonic accelerators are one candidate for implementing these heuristic algorithms. Although the computational speedup provided by the photonic architectures is still polynomial, they operate on much faster clock rates, offering orders-of-magnitude speedup compared to conventional architectures. One of the examples of such architectures is INPRIS [4]. INPRIS is the photonic implementation of the Photonic Recurrent Ising Sampler (PRIS) [15], which models arbitrary Ising-type Hamiltonian in Equation 1. INPRIS employs an array of tunable Mach-Zehnder interferometers (MZIs) to perform MVM. INPRIS is able to perform each iteration in an order of magnitude faster than digital electronics implementation. Compared to OPCM, MZI occupies a larger area [12], [28]. Therefore, we chose to use OPCM in SOPHIE.

C. PRIS Algorithm

SOPHIE is based on a modified version of the PRIS algorithm [15], which is a fast and efficient solution to the Ising problems using MZI networks. So here we provide an overview of the PRIS algorithm. The PRIS algorithm first performs eigenvalue dropout, a preprocessing step to improve the solution quality. Assume the Ising coupling matrix K is a symmetric real-valued matrix of size $N \times N$. Matrix K can be eigenvalue decomposed as

$$K = UDU^* \tag{2}$$

where U is a unitary matrix, U^* is its conjugate transpose, and D is a real-valued diagonal matrix. The transformation matrix C can be calculated using

$$C = USq_\alpha(D)U^* \tag{3}$$

where

$$\begin{aligned} Sq_\alpha(D) &= 2Re(\sqrt{D + \alpha\Delta}) \\ \alpha \in [0,1] \quad \Delta_{ii} &= \sum_{j \neq i} |K_{ij}| \end{aligned} \tag{4}$$

C is also a symmetric real-valued matrix. The parameter α needs to be adjusted to improve the solution quality and performance of the algorithm.

Once we obtain the transformation matrix C after eigenvalue dropout, we can start the recurrent computations. Suppose $S^{(t)}$ is a vector of spins at time step t with the size N ($S^{(t)} \in \{0,1\}^N$), the transformation from time step t to $t+1$ can be described as

$$\begin{aligned} X^{(t)} &\sim \mathcal{N}(CS^{(t)}|\phi), \\ S^{(t+1)} &= Th_\theta(X^{(t)}) \end{aligned} \tag{5}$$

where $\mathcal{N}(\mu|\phi)$ is a normal distribution with mean μ and standard deviation ϕ.

$Th_\theta(X^{(t)})$ is a non-linear thresholding function

$$Th_{\theta_i}(S_i) = \begin{cases} 0, & \text{if } X_i < \theta_i, \\ 1, & \text{otherwise.} \end{cases} \tag{6}$$

where the threshold is

$$\theta_i = \sum_j C_{ij}/2 \tag{7}$$

When the algorithm runs recurrently for a sufficiently large number of iterations, the system will move towards low-energy states, which represent a good solution to the Ising problem [15]. Although PRIS is very efficient in solving the Ising problem, once the problem becomes larger than the hardware's capacity, it will suffer from the same performance degradation as the other designs. So, we modify the PRIS algorithm and propose a new architecture that can handle larger Ising problems much more efficiently than previously proposed solutions.

III. Architecture

In this section, we first describe our modified algorithm tailored to the OPCM arrays, which enhances the scalability of the Ising machine. Then, we discuss SOPHIE's system architecture, micro-architecture, and dataflow to solve the Ising problem.

A. Proposed Modification to the PRIS Algorithm

The essential part of PRIS is the recurrent MVM described by Equation 5. Although photonic devices can perform MVM very efficiently, the matrix C obtained by Equation 3 must fit entirely into the photonic device. This is not practical for large Ising problems, as photonic devices occupy a large footprint. If we apply the standard tiling technique on MVM, the communication and glue computation will become a significant bottleneck [27]. However, suppose we can make inter-tile communication less frequent or reduce the amount of data communicated. In that case, we can reduce this overhead and make the algorithm more scalable to large problem sizes. We propose the following modifications (Algorithm 1) to the PRIS algorithm to make it possible.

1) Symmetric Local Update: As shown in Figure 3, the standard tiling technique on MVM decomposes the input vector, output vector, and the matrix C into multiple tiles. Each input vector tile will be multiplied with a matrix tile to generate a partial sum tile, and this is called the local computation. The partial sums (before thresholding) of the tiles from the same columns of the matrix C will be accumulated into an output vector tile, effectively performing the glue computation and global communication. The local computations can be significantly accelerated because each OPCM array can compute one tile of MVM efficiently in one cycle, and many OPCM arrays can work in parallel. However, the global synchronization, i.e., the glue computation and the global communication between different tiles, cannot be parallelized in this way [27]. Therefore, they will become the bottleneck for large MVM computations according to Amdahl's Law.

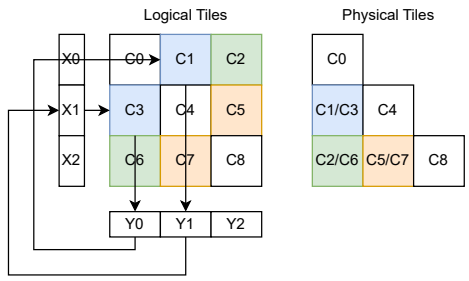

Fig. 3. Symmetric local update and tile mapping. The recurrent computations on a pair of symmetric tiles can execute locally, assuming other tiles remain constant. A pair of symmetric tiles are transposes of each other, so they will share the same OPCM array.

Algorithm 1: Modified PRIS Algorithm

Data: Tiles of matrix C, initial state for all spins in a batch
Result: Final state of the spins in a batch

```
Generate scheduling information and initialize device
for global_iter do
    Load the list of selected tile pairs
    foreach selected pair of symmetric tiles do
     // Spatial unrolling
        OPCMProgramming()
        BufferInit()
        foreach job in the batch do
            for local_iter do
                SymmetricLocalUpdate()
    GlobalSync()
```

Unfortunately, the PRIS algorithm executes recurrent MVM very frequently and performs global synchronization for every iteration, exacerbating this overhead.

To make the PRIS algorithm scalable to large Ising problems, we need to reduce the overhead of global synchronization. A simple intuition is to run more local computations before performing a global synchronization. In the PRIS algorithm, the output of MVM will become the input for the next iteration, making it a recurrent computation. However, if we apply the standard tiling technique to the recurrent MVM, the local computation on each tile is no longer recurrent – the output of one tile will affect the input of another tile rather than itself (except for diagonal tiles), meaning that it cannot perform the next local computation without a global synchronization. Therefore, we need to find a closed loop in the computation of local tiles so that they can execute recurrent computations on themselves without needing global synchronization.

We start by focusing on a pair of symmetric tiles and identifying the closed loop in the computation with those two tiles. Take the pair of $C3$ and $C1$ in Figure 3 as an example: the computation starts from the input vector tile $X1$ and the matrix tile $C3$, whose MVM result will affect the output vector tile $Y0$ in the first iteration of PRIS. $Y0$ will become the input vector tile $X0$ in the next iteration, and will be multiplied with $C1$ and will contribute to $Y1$. Then, in the third iteration, the updated $Y1$ will become

the input vector tile $X1$, and the computation will repeat as above. We can identify a closed loop in two consecutive iterations, where only a pair of symmetric tiles ($C3$ and $C1$) are involved, as well as their associated input and output tiles. An exception is the diagonal tiles, whose symmetric tiles are themselves, so the computation on themselves is naturally a closed loop. In PRIS, the matrix tiles stay constant throughout the computation, and the input and output vector tiles change frequently. If we assume that some updates in the input and output vector tiles can be skipped, the local computation can be executed independently without needing global synchronizations. For example, if we treat all other tiles as constants, the computations within $C3$ and $C1$ will only depend on, and will only affect, their local copies of $X0$, $X1$, $Y0$, and $Y1$. With this assumption, we can run many local MVMs on a pair of symmetric tiles without performing a global synchronization. This technique significantly reduces the global synchronization overhead and makes the algorithm more suitable for large graphs.

We name this closed-loop computation the local iteration: a pair of symmetric tiles (or a diagonal tile) performs recurrent MVM independent of other tiles, assuming all other tiles stay constant. This way, we can avoid a large portion of global synchronization overhead. This algorithm requires some additional buffers in the hardware, e.g., every matrix tile needs to store the partial sum of the column, which is called the offset vector. Besides, they need separate buffers for their individual copy of the input vector, output vector, local partial sum vector, and matrix tiles. These buffers are relatively small compared to the matrix tiles, and SRAM can easily implement them.

2) Stochastic Global Iteration: As mentioned above, every pair of symmetric tiles assumes the local MVM results of other tiles stay constant during local iterations. However, they still need to exchange the updated partial sums and input vectors (spin states) periodically with each other. Otherwise, the error will accumulate and become unacceptable. After a specified number of local iterations, each tile will update its local input vector with the average of the tiles from the same column and update its offset vector with the sum of the local MVM results of all other tiles in the same column. We call this global synchronization, and the local iterations on all tiles plus global synchronization are called global iterations.

Given that the PRIS algorithm is a stochastic process, we propose to further apply the following optimizations to reduce both the computation and the communication requirements:

Stochastic Tile Computation: During the global iteration, we do not execute the local iterations on all the tiles; instead, we randomly pick only part of the tiles. We make sure to select both tiles in a symmetric pair or the diagonal tiles so that the chosen tiles can still perform symmetric local updates. Only the randomly selected tiles will be mapped to the hardware, and they will send the updated input vector and the change of the local MVM results to other blocks. This method significantly reduces the total amount of computation and communication. Experiments show that we can reduce the computation and communication operations associated with up to 50% of the tiles while maintaining good solution quality.

Stochastic Spin Update: With the symmetric local update technique, each tile will have a separate copy of the spin states stored in their input/output buffers. During global synchronization, we need to compute the average of the input vectors of all tiles in the same column and broadcast it to the entire column of tiles. Rather than computing the average from all copies, we randomly pick the spin states (input vector) from one tile and broadcast them to the entire column. This technique reduces the communication and computation overhead for the average computation. The stochastic spin update technique will not significantly affect the solution quality because it approximates averaging all spin copies if the number of global iterations is sufficiently large. The effect of this technique can also be seen as increasing the noise ϕ in PRIS.

Applicability to other Ising Machines: The optimizations discussed above are tailored for OPCM devices, but they may also apply to other Ising machines depending on their device or hardware characteristics. Given that the optimizations we propose involve handling part of the tiles' computation and communication, they are more suitable for computation-based Ising machines than physics-based ones. Below, we briefly discuss the applicability of our two optimizations to other Ising machines.

1) The Symmetric Local Update technique maps a pair of symmetric tiles into one OPCM array. This method saves approximately half of the OPCM area and enables iterative local computations within the pair of tiles so that global communication is greatly reduced. This technique also applies to software and other computation-based Ising machines to reduce global communications in local computations. Other Ising machines may benefit from the data locality and decreased communication requirements. However, in order to reduce area (memory capacity) with this method, the device must be able to access both the stored matrix and its transpose efficiently. For example, specially designed ReRAM devices [29] may have the potential to benefit from this ability.
2) The Stochastic Global Iteration technique effectively reduces the number of tiles to compute and the amount of global communication in each global iteration. Software and computation-based Ising machines could potentially benefit from the decrease in the computation and communication requirements.

B. System Architecture

As depicted in Figure 4, SOPHIE comprises a host processor and one or more accelerators. The accelerator is designed as a 2.5D-integrated system with different chiplets integrated onto the interposer. These chiplets include the DRAM chiplet, the controller chiplet, the laser sources, and multiple OPCM chiplets that house the processing elements (PEs).

The host processor interacts with the accelerator through a conventional electrical bus. The controller chiplet schedules the operations of other chiplets, manages communication between the host, DRAM chiplets, and OPCM chiplets, and

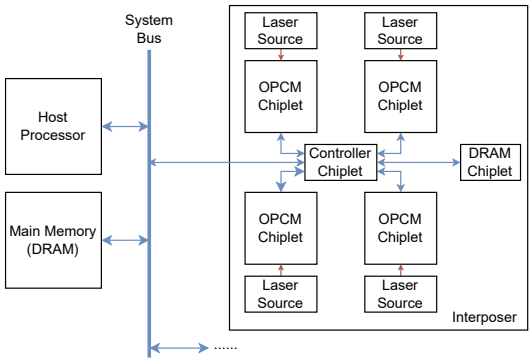

Fig. 4. Architecture of SOPHIE. It consists of a 2.5D integrated accelerator that interfaces with the host processor and main memory through the system bus. Communication between the chiplets on the interposer uses electric links.

carries out glue computations. The scheduling information is generated offline, so the controller only needs simple state machines to execute them. Given that we overlap communication with computation and OPCM programming, which are the dominant operations, the controller chiplet is not on the critical path. The DRAM chiplet contains DDR4 memory and stores all the coupling matrix (matrix C in Section II-C) tiles assigned to its interposer, along with their corresponding buffer contents. The OPCM chiplets, on the other hand, store the coupling matrix tiles in OPCM arrays, execute local MVM computations, and relay the results to the controller during global synchronization, as described in Section III-A. Electrical links connect the chiplets on the interposer. In the case of a multi-interposer system, the global synchronization between different interposers is through the system bus.

C. Micro-architecture

Each OPCM chiplet contains multiple PEs, as shown in Figure 5, and each PE will execute the local iterations of a pair of symmetric tiles, as described in Section III-A1. A PE comprises an OPCM crossbar array and associated SRAM buffers, control logic, and programming circuitry. The OPCM array [12] is a PIM device that can perform MVM with the stored matrices efficiently. It features a waveguide crossbar and phase change material (e.g., GST) next to the cross points. Each GST cell's transmittance encodes one element of a matrix tile. We employ two sets of E-O and O-E converters and Directional Couplers (DCs) to enable bi-directional operations. For instance, in Figure 5, the left E-O converters modulate the amplitude of multiple lasers, each directed to a row waveguide. Specially designed DCs evenly split the input laser to all the GST cells in the same row, effectively broadcasting the input across an entire row. The GST cells perform multiplication by attenuating the laser intensity. DCs then merge the attenuated laser from the same column to the same output port, executing an accumulation operation. The output of each column is the vector dot product between the input vector and a column of the matrix, and the output of all columns is the MVM result between the input vector and the matrix.

Given the input vector $S = \{s_1, s_2, ..., s_n\}^\top$ and the matrix C, sending the laser from the left and reading the output from the bottom computes the output vector as follows:

$$Y_1 = C^\top S \tag{8}$$

Alternatively, if the laser is sent from the bottom and the result is read from the left, the output will be:

$$Y_2 = CS \tag{9}$$

This approach allows us to multiply a vector with a matrix and its transpose without re-programming the OPCM array. As Section III-A1 mentions, we need to perform MVM with a pair of symmetric matrix tiles that are transposes of each other. With the transpose ability of the PE, the two symmetric tiles can be stored in one OPCM array, saving approximately half of the OPCM area. Additionally, to support both positive and negative values in the matrix, we use two arrays to represent the positive and negative parts, respectively. The partial MVM results of the positive and negative arrays are subtracted in the analog electronic domain [30].

The E-O converters have only 1-bit precision because the spins are 1-bit binary variables. The O-E converters consist of photodetectors (PD), noise generators (NG), and dual-precision ADCs, shown in Figure 5. The photodetector converts the optical intensity into the analog electric domain, and the noise generator adds analog noise to the analog signal. The total amount of noise, including the inherent noise of the signal and the noise produced by the noise generator, is represented by the noise standard deviation ϕ in the PRIS algorithm (Section II-C, Equation 5). Algorithm parameter ϕ is agnostic to hardware's noise, and we adjust the noise generator to ensure the total noise's standard deviation equals ϕ. During most of the local iterations, the dual-precision ADC operates in 1-bit mode, acting as a thresholding unit to produce the local spin states. The threshold of the 1-bit ADC is also adjustable, described by Equation 7. During global synchronization, we need to update the offset vector, which is the sum of local MVM results from other tiles, requiring multi-bit local sum results. Therefore, the ADC operates in an 8-bit mode during the last local iteration before the global synchronization, spending more cycles to generate an 8-bit local sum. This dual-precision design reduces the time and energy spent on most of the local iterations while still supporting the global synchronization of the proposed algorithm.

D. Mapping and Scheduling

As described in Section III-A1, we need to process a pair of symmetric tiles of the matrix C in the same PE. The PE's transpose ability makes it possible to store a pair of symmetric tiles (which have the same contents but are transposed versions of each other) using only one OPCM array and perform MVM with them. Therefore, we map a pair of symmetric logical tiles onto one physical OPCM array. Although they share the same OPCM array, they still need separate SRAM buffers for their input/output, offset, and local partial sum vectors

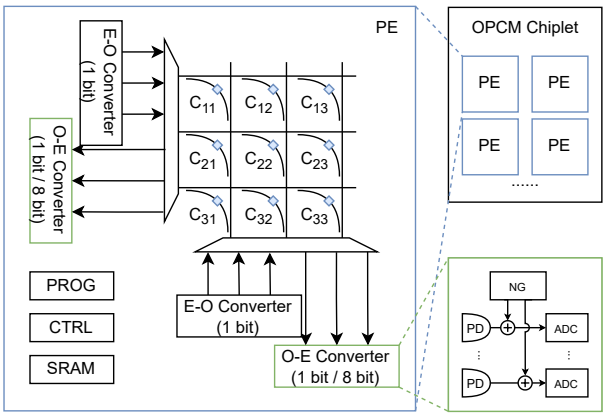

Fig. 5. OPCM chiplet and PE architecture.

(Section III-A1). Those buffers associated with the pair of tiles are also mapped to the same PE's SRAM buffers. Each PE executes the local iterations on a pair of symmetric tiles in a time-duplexing fashion. Except for the diagonal tiles, whose symmetric tiles are themselves, they will be individually mapped onto a single PE. This technique reduces almost half of the OPCM area, making the design more scalable for large graphs.

However, the communication pattern becomes complicated when two logical tiles at symmetric positions are mapped to one physical OPCM array. When accessing a logical tile, we need to keep track of the physical tile it maps to and schedule the communication between PEs and DRAM accordingly. For example, we need to gather/scatter data from a row of logical tiles; however, those logical tiles may not reside in the same row of physical tiles. Moreover, the Stochastic Global Iteration technique randomly selects only a portion of tiles to compute, making the communication pattern more complex. To simplify the accelerator's control logic, we statically generate all the mapping and scheduling information before the computation starts and send it to the controller chiplet. The randomness in the stochastic global iterations (described in Section III-A2) is also determined offline beforehand. For example, we randomly generate a list of logical tiles to be selected in each global iteration before initiating the computation and then send this list to the controller. The controller only needs simple SRAM arrays and state machines to execute the scheduling plan generated by the host, and this execution overhead is negligible.

E. Dataflow

Algorithm 1 provides a simplified overview of the modified algorithm's execution. Given a large Ising problem, SOPHIE decomposes it into smaller tiles, schedules and executes the tiles on the physical PEs sequentially for many iterations, and then gathers the results. First, the host CPU will preprocess the input graph of the Ising problem to obtain the transform matrix C, as described in Section II-C. Based on the configurations of tile size, hardware capacity, and the number of jobs to run in a batch, the host generates tiling and scheduling information and the initial states of each tile, then transfers all the data to the DRAMs of the appropriate accelerators. The controller then loads the first set of blocks scheduled to run and transfers them to the SRAM of the designated OPCM chiplets. Then, the coupling matrix tiles are programmed onto the OPCM arrays. Each tile's associated buffer content (described in Section III-A1) is also transferred to the SRAM of the OPCM chiplet. The programming and communication can overlap to enhance performance.

Following OPCM programming and buffer initialization, the PEs in the OPCM chiplet initiate the local iterations for the first job in the batch. Data are read from SRAM buffers, converted to optical domains, and sent to the OPCM array. Each optical output of an MVM operation is converted to electric analog signals. The signals will be applied to analog noise, converted to 1-bit digital, and finally stored back in the SRAM buffer as the input to the next iteration. The recurrent local computations will repeat for a specified number of local iterations for every job in the batch. Except for the last local iteration of each job, the ADC will operate in 8-bit mode to obtain the local MVM result of the current tile. The chiplet then performs global synchronization by sending the buffer contents back to the controller and receiving them for the next scheduled tile. This process repeats for a series of local iterations for other sets of tiles. The controller can overlap the global synchronization with the OPCM re-programming and the local iterations of the next set of tiles. Once the required number of global iterations is reached, the controller sends the result from DRAM back to the host's main memory.

IV. Evaluation

In this section, we first evaluate our modified algorithm's solution quality and performance, and then discuss the performance, power, and area of SOPHIE. Finally, we compare SOPHIE's performance with other works.

A. Evaluation Methodology

We develop a functional simulator in Python to test the solution quality of our algorithm, find the optimal parameter setting, and obtain the number of iterations required for a solution. The functional simulator also counts the total number of each type of operation, and these numbers serve as the input for power and performance estimation.

We build in-house tools to compute the power, performance, and area of SOPHIE. At a high level, SOPHIE is assumed to comprise one or more accelerators. Each accelerator houses an interposer embedded with electrical links [31]. The interposer hosts a controller chiplet, a DRAM chiplet, a laser source chiplet, and 4 OPCM chiplets. Each OPCM chiplet, occupying an area of 486 mm^2, contains 64 PEs. The size and number of OPCM arrays are chosen to maximize the energy, delay, and area product (EDAP), which will be discussed in Section IV-C. Each PE incorporates 64×128 OPCM cells (positive and negative parts) to store a 64×64 matrix tile. The area of the OPCM array is calculated assuming each OPCM cell is 30×30 μm^2 [12], with MRRs having a diameter of 20 μm.

We assume the accelerator operates at 5 GHz. Photonic circuits can operate at extremely high frequencies. The existing implementation of OPCM is demonstrated at 18 GHz [12], delivering extremely high MVM performance. However, designing efficient and reliable peripheral electronic circuits to operate at such high frequencies is not feasible. Therefore, we set the accelerator to run at 5 GHz, a frequency at which we can design electronic circuits in GF22FDX CMOS technology [32]. The power consumption of the accelerator includes laser power and electric power. The laser power is determined by the optical loss of the photonic circuits. We compute the optical loss of the OPCM array [12] assuming the losses of the GST cell, waveguide crossing, and DC are 0.6 dB, 0.0028 dB, and 0.01 dB, respectively [12], and the combined quantum efficiency of the laser and photodetector is 10%. Laser power is calculated backward based on the loss of the photonic circuits and the energy required at the photodetector. Under the configuration selected in Section IV-C, the laser source consumes 469 mW per wavelength. The electric programming energy of each GST cell is assumed to be the average of amorphizing and crystallizing (5.55 nJ and 860.71 nJ, respectively [19]), and it takes 400 ns to program the entire array [19]. The E-O conversion costs 1 pJ/bit [12], and the O-E conversion at 5 GS/s consumes 29 mW [33]. We synthesize the CMOS arithmetic units for glue computation individually with GF22FDX technology node using Cadence tools to get the area and energy consumption of each unit, which is then used to get the energy consumption of each operation. We utilize the memory compiler to generate an SRAM array for the specified technology node. We assume that the SRAM operates at 1 GHz, and the SRAM accesses are interleaved across multiple SRAM arrays to keep pace with the 5 GHz accelerator frequency. With the optimal configuration chosen in Section IV-C, the SRAM total capacity is 7.6 MB, occupying an area of 11.5 mm^2 and consuming 540 mW power. The control logic is simple because it only needs to execute the pre-generated scheduling information. We synthesize the control logic using TSMC40 technology node and scale the power and area numbers to 22 nm node. Specifically, the control logic consumes 26 mW power and occupies 11,536 μm^2 area. DRAM accesses are assumed to be 20 pJ/bit [34]. We assume the DRAM latency to be 40 ns if within the same interposer, or 80 ns for cross-interposer access [35]. The system comprises a 16-lane CXL bus, providing a total bandwidth of 64 GB/s. The above numbers are used in combination with the hardware configuration and operation counts to compute the system's PPA.

Table I presents a summary of the graphs used as benchmarks for our system. As mentioned in Section II-B, we can map a real-world combinatorial problem into the max-cut problem of a graph, and solve it with Ising machines. We evaluate our system by solving the max-cut problem of graphs from GSET and K-graphs [16]. GSET and K-graphs have been chosen as benchmarks because they are utilized in prior Ising machine algorithms and implementations. These graphs are extensively examined in algorithmic literature, and they either have straightforward ground truth solutions or have well-recognized best-known solutions [5]. As a result, they are frequently employed in prior Ising machine implementations, and we select them to facilitate an equitable comparison with other works. GSET comprises graphs generated by the Rudy graph generator [16]. Specifically, we choose G1 and G22 to facilitate a fair comparison with other Ising machines that also evaluated these graphs. K-graphs are complete graphs with random edge weights. We generate both small K-graphs (K100) and large K-graphs (K16384 and K32768) using Rudy [16] to evaluate the scalability and performance of SOPHIE.

TABLE I
BENCHMARK GRAPHS

Graph	# Nodes	Description
G1	800	From GSET dataset
G22	2000	
K100	100	Randomly generated complete graphs
K16384	16384	
K32768	32768	

B. Evaluation of the Modified Algorithm

1) Effect of Algorithm Parameters: The two parameters in the original PRIS algorithm, namely the noise standard deviation ϕ and the eigenvalue dropout factor α, have a significant impact on the solution quality and the number of iterations required to converge to a solution. Our improved algorithm effectively adds more noise to PRIS due to the stochastic global iterations. Here, we will discuss the effects of these two parameters and how they affect hardware implementation.

Figure 6 shows the solution quality of our modified algorithm on G1 and G22 with various ϕ and α, under the settings of tile size 64, 10 local iterations per global iteration, running 500 global iterations, picking all tiles during the global iteration, and stochastic spin update applied. Each data point is an average of 10 runs.

Effect of algorithm parameters: The behavior of ϕ and α in the modified algorithm closely resembles that in the original PRIS algorithm. Together, these two parameters affect the solution quality for a given graph. However, there's a slight difference: The optimal noise ϕ for the modified algorithm is smaller than that for the original algorithm. This is because our modified algorithm involves a stochastic process, which effectively introduces more noise to the computation. The eigenvalue-dropout factor α has similar effects in both versions, with the optimal setting around zero for G1 and G22. It is used as a design knob to fine-tune the algorithm and improve the solution quality [15]. Note that the optimal settings depend on the graph order and graph density [4], so the optimal α might not be zero for other graphs.

Optimal setting for modified algorithm: For the modified algorithm, both graphs achieve the best solution quality at $\alpha = 0$. The optimal setting that leads to the best solution quality is $\phi = 0.2, \alpha = 0$ for G1, and $\phi = 0.1, \alpha = 0$ for G22. With the optimal setting, the highest average cut values are within 5% of the best-known results from previous works [5]. The

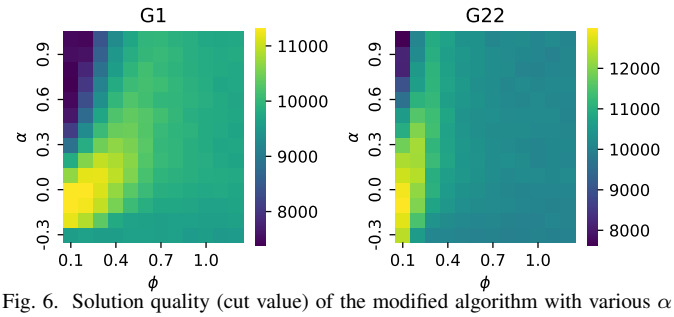

Fig. 6. Solution quality (cut value) of the modified algorithm with various α and ϕ settings for G1 and G22.

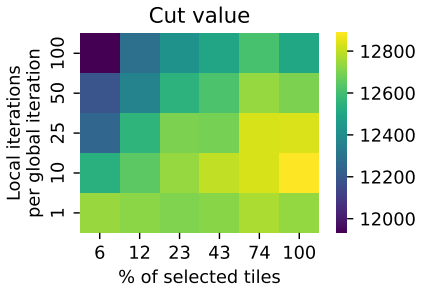

Fig. 7. Impact of stochastic tile computation on the solution quality for G22. Applying more local iterations per global iteration decreases the solution quality. Selecting fewer tiles to compute in every global iteration also deteriorates the solution quality. However, the impact is relatively small.

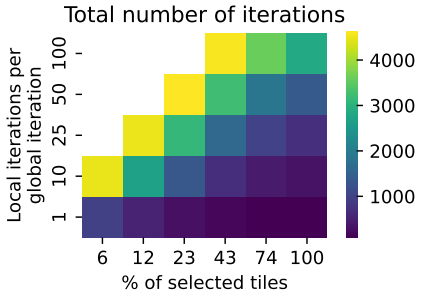

Fig. 8. Total number of iterations required to reach 95% of the known-best solution for G22. Blank cells indicate they fail to converge to the target solution quality within 5000 iterations. As we aggressively apply the symmetric local update and stochastic tile computation techniques (toward the upper-left corner) to reduce the computation and synchronization overhead, we need more number of iterations.

optimal settings of ϕ and α depend on both the graph order and graph density [4]. Therefore, we can generate a lookup table of ϕ and α for common (graph order, graph density) pairs before any computations.

Hardware implementation effects: As mentioned in Section III-C, parameter ϕ in the modified algorithm denotes the total amount of noise, including the noise from OPCM itself and the noise added from noise generators. We adjust the noise generator so that the total amount of noise meets the optimal settings, making the ϕ parameter agnostic to hardware implementations. However, if this algorithm is applied to other devices with significant noise, the optimal ϕ setting might not be achieved even if we don't use the noise generator. In this case, the system's solution quality might be affected.

2) Effect of Stochastic Tile Computation: As mentioned in Section III-A2, we apply the stochastic tile computation by randomly picking only part of the tiles in each global iteration. This technique reduces overall computation and communication demand but could also hinder the quality of the solution. Figure 7 shows the relation between the solution quality of G22 with various settings of the number of local iterations per global iteration and the percentage of tiles selected in each global iteration. Every experiment runs for a total of 5000 local iterations, and all other parameters are the optimal settings obtained from the above discussions. Each data point is the average of 10 runs. Increasing the number of local iterations per global iteration, and the percentage of the selected tiles, both positively affect the solution quality. However, the impact on solution quality is small – the worst setting is still within 10% of the known best solution. This is because a total of 5000 local iterations are so large for this graph size that even the most aggressive setting will converge. We observed similar behavior in other graphs as well. The results indicate that we can perform 10 local iterations per global iteration and select only about $50\% - 75\%$ of the tiles to compute during the global iterations while having a negligible impact on the solution quality. This optimal setting means that the stochastic global iteration technique can reduce up to 50% of the total computation (and the associated communication) with an acceptable impact on the solution quality.

While the stochastic global iteration technique significantly reduces the overall computation and global synchronization overhead, the system might spend more iterations to converge to a solution. Figure 8 shows the total number of (local) iterations required to reach 95% of best-known solution with various percentages of tiles selected in each global iteration for G22. All other parameters are the optimal settings obtained in the previous discussion. Each data point is the average of 100 runs. The blank cells in the grid indicate they failed to reach the target solution quality within a total of 5000 iterations. As shown in the figure, picking fewer tiles to compute in each global iteration makes the algorithm spend more iterations to converge. Skipping more global synchronization (i.e., running more local iterations per global iteration) also leads to an increased total number of iterations. However, the number of iterations does not necessarily indicate the run time because local iterations are much faster than global synchronizations. Therefore, we need timing analysis to decide the optimal settings.

C. Power, Performance, and Area Results

The tile size and batch size settings affect the power, performance, and area of SOPHIE. Given the total number of OPCM cells, changing the size of each tile (OPCM array) affects the required laser power and also the total area of the photonic devices. Increasing the number of jobs per batch will amortize the global synchronization and OPCM programming cost but will require more SRAM buffers. Figure 9 shows the EDAP per job for running a graph of order 32768 for 500 global iterations, 10 local iteration per global iteration, and with one accelerator in the system. From the figure, a tile size

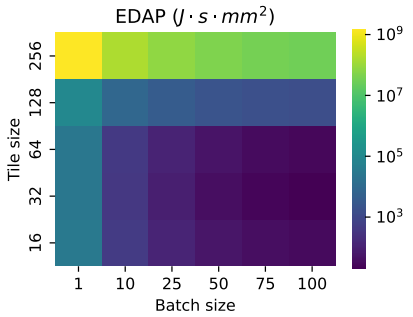

Fig. 9. Energy, delay, and area product (EDAP) of one accelerator running K32768. Batch size of 100 and tile size of 64 yields the best EDAP.

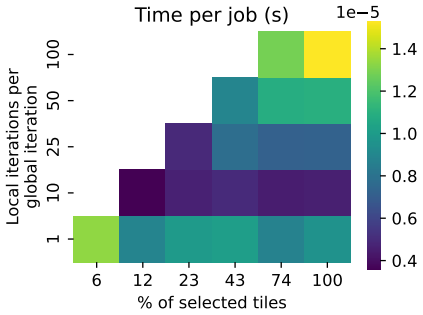

Fig. 10. Run time per job to reach the solution for G22. (Within 5% of best-known solution). Blank cells indicate they fail to reach the target solution quality within 5000 iterations.

of 64 and batch size of 100 achieves the lowest EDAP number, and we will use this setting for the remaining analyses.

We evaluate the performance of SOPHIE using G22. Figure 10 shows the run time per job to reach 95% of the known best solution, with various settings of local iterations per global iteration and the percentage of tiles selected in each global iteration. To reduce simulation time, we use G22 (2000 nodes) as the benchmark. We focus on the scalability for large graphs that do not fit in the hardware, so in this experiment, we limit the total OPCM capacity to 512×512 coupling coefficients, such that programming overhead is accounted for. The figure shows that as the number of local iterations per global iteration increases, the run time first decreases and then increases. This is because the modified algorithm reduces global synchronization overhead and the overall computation per global iteration; however, it might need more global iterations to converge to a solution. Therefore, there's a trade-off between the time/energy cost per global iteration and the number of global iterations required. According to Figure 10 and Figure 7, 10 local iterations per global iteration and selecting 74% of the tiles during the global iteration yields the best performance and acceptable solution quality.

D. Comparison with Other Works

Small Graph: We first compare the performance of SOPHIE against previous works using small graphs K100, G1, and G22, which can fit entirely into the hardware, shown in Table II. We benchmark our work against INPRIS [4], PRIS [15], CIM [9], BRIM [8], BLS [5], and D-Wave [36]. Previous papers have reported the results of other architectures. We simulate our system with the settings in Section IV-A and using 4 accelerators. Given that the graphs can fit into the accelerator entirely, we need to program the OPCM array just once, and this programming cost is included in the simulation. The run time is amortized for each job in a batch.

From Table II, we can see that our design is at least $3\times$ faster than INPRIS, $161\times$ faster than PRIS, and is $7419\times - 25000\times$ faster than CIM. It is also $1.25\times$ faster than BRIM, and is orders of magnitude faster than the CPU and the D-Wave quantum annealer. This is mainly because the OPCMs can compute MVM at very high speed, and the modified algorithm reduces the global synchronization overhead. Please note these results are using 4 SOPHIE accelerators so that the largest graph G22 can fit in the hardware. The SOPHIE results include the amortized initial programming time, while results from other works do not include the initial programming/setup time. The solution quality of SOPHIE under this configuration is slightly lower than other works (error $< 5\%$ vs. $< 1\%$), which is due to the proposed modification to the PRIS algorithm. This is due to the trade-off between solution quality and speed, as discussed in Section IV-B and IV-C.

Large Graph: We then evaluate our design with K16384 and K32768 graphs, which can not fit entirely into OPCM arrays. Shown in Table III, we compare run time of our design with SB [37] and mBRIM$_{3D}$ [27] (concurrent mode). Both mBRIM$_{3D}$ and SB require the hardware capacity to be larger than the problem size and use multiple accelerators or chips to increase the total hardware capacity. Although it works for K16384, it might be difficult to solve problems with much larger sizes. The overall hardware capacity of such designs is constrained not just by the physical area but also by integration and communication technologies, such as 3D integration and through-silicon vias. Consequently, accommodating an extremely large graph within the hardware will pose a significant challenge. For OPCM-based Ising machines, it is even harder to support large graphs because the footprint of photonic devices is much larger than that of electronic devices. Therefore, our system is designed to work on the decomposed subproblems sequentially in a time-duplexing manner without fitting the entire problem into hardware at once. Fortunately, the overhead caused by the time-duplexing is minimized through the architecture and algorithm optimizations.

As shown in Table III, for K16384, our design with only one accelerator is $32\times$ faster than SB with 8 FPGAs. However, our single accelerator design is $35\times$ slower than mBRIM$_{3D}$ with 4 chips. This is mainly because of the overhead in global synchronization and re-programming of the OPCM arrays. By incorporating more accelerators into our design, we can reduce the overall run time. With 4 accelerators, our design is $125\times$ faster than the 8-FPGA SB implementation but is still $8.8\times$ slower than mBRIM$_{3D}$. We can further improve the performance of the system by adding more accelerators. For K32768, both SB and mBRIM$_{3D}$ need more instances to accommodate the problem. However, SOPHIE can support such a large graph without adding more accelerators. For SOPHIE, the run time for K32768 is about $3\times$ of the runtime for K16384

TABLE II
PERFORMANCE (SOLUTION QUALITY) FOR SMALL GRAPHS

Architecture	Type	K100	G1	G22
SOPHIE	Photonic	0.31 µs ($T_{90\%}$)	0.096 µs (4.1%[a])	0.2 µs (3.9%[a])
INPRIS [4]	Photonic	$1-10$ µs ($T_{90\%}$)	-	-
PRIS [15]	FPGA	50 µs$-$1 ms ($T_{90\%}$)	-	-
CIM [9]	Photonic	2.3 ms ($T_{90\%}$)	-	5 ms (0.8%[b])
BRIM [8]	Electric	-	-	0.25 µs (0.3%[b])
BLS [5]	CPU	-	13 s (0.1%[a])	560 s (0.1%[a])
D-Wave [36]	Quantum	5×10^{18} s ($T_{90\%}$)	-	-

$T_{90\%}$: 90% probability to reach ground state.
[a] Average error relative to the best-known solution.
[b] Best-case error relative to the best-known solution.

TABLE III
PERFORMANCE COMPARISON FOR LARGE GRAPHS

Architecture	Type	# accelerators	K16384	K32768
SOPHIE	Photonic	1	38.25 µs	129.0 µs
		2	20.40 µs	68.80 µs
		4	9.69 µs	32.34 µs
SB [37]	FPGA	8	1.21 ms	-
mBRIM_{3D} [27]	Electric	4	1.1 µs	-

without the need for additional accelerators. The performance improvement of SOPHIE by using more accelerators may appear super linear, but this is not accurate. The issue arises because the total number of tiles to compute is not evenly divisible by the total number of PEs available. Consequently, the hardware remains underutilized. This mismatch leads to slight variations in hardware utilization rates across different configurations, resulting in slightly anomalous data points.

V. RELATED WORK

In SOPHIE, we tackle the scalability challenges of Ising machines by employing symmetric local updates and stochastic global iterations. These methods lead to decreased computational and communication requirements, enabling efficient handling of larger-scale problems. There have been many prior works addressing the scalability of the Ising machines.

One solution would be to divide the main problem into many sub-problems and solve them. D-Wave tool [38] proposes an algorithm that can be applied to any Ising machine to solve problems larger than the hardware capacity. Their proposed algorithm is a divide-and-conquer solution for larger problems. However, due to the dependency of the sub-problems on each other, one of the main drawbacks of this solution is the number of reprogramming required for the Ising machine. For many Ising machines, the time required for reprogramming can be much larger compared to the computation time, so the reprogramming would become a significant bottleneck. As an example, D-wave takes 11.7 ms to program and only 240 µs to perform the rest of the steps [39].

Sharma et al. [27] designed a multi-chip architecture that can increase the capacity of a physics-based Ising machine. Their design employs multiple chips of the BRIM Ising machine that communicate together in order to increase the total capacity of the physical Ising machine. They offer multiple operation modes that can accelerate solving the Ising problem. Their experimental design consists of 4 BRIM chips, each with a capacity of 8192 nodes, resulting in an Ising machine with a total capacity of 16384 nodes. Their design presents a rapid and energy-efficient approach to addressing Ising problems. However, a notable limitation of their design is the necessity to store all spin configurations on chips to solve the Ising problem. For instance, in the case of the K32768 graph, assuming each chip can hold 8192 nodes [27], it would require 16 chips to tackle the problem adequately.

Simulated bifurcation (SB) [40] is another heuristic algorithm derived from a quantum bifurcation machine. SB supports all-to-all coupled spins and offers parallelism, which can be exploited using parallel processors. Tatsumura et al. [37] propose a multi-chip architecture using FPGAs that can perform SB in parallel to solve large problems. Compared to SOPHIE, the 8-FPGA implementation of SB is $47\times$ slower than the 4-accelerator version of this work.

Our proposed architecture, referred to as SOPHIE, effectively addresses the scalability challenges encountered in Ising machines, surpassing previous approaches. SOPHIE alleviates the global synchronization cost in divide-and-conquer solutions through the implementation of symmetric local updates, which isolate pairs of tiles and reduce global communication. Additionally, we confront the glue computation and communication bottlenecks associated with global synchronization by employing stochastic global iteration, reducing the computation and communication requirement. Our design incorporates OPCM tiles to enhance computational throughput, further improving the performance.

VI. CONCLUSION

Ising machines stand out as promising non-von Neumann architectures that are tailored for solving combinatorial optimization problems. Previous implementations of Ising machines required the hardware capacity to be larger than the problem size; otherwise, their performance would have degraded significantly. We propose SOPHIE, which uses OPCM as a computation-based Ising machine. While designing SOPHIE, we perform algorithm-level, device-level, and architecture-level optimizations. Specifically, our modified algorithm incorporates a symmetric local update technique and a stochastic global synchronization strategy, which reduce the overall computation demand and global synchronization overhead. We apply device-level optimizations to support the modified algorithm, including employing bi-directional OPCM arrays and dual-precision ADCs. Our symmetric tile mapping method at the architecture level reduces approximately half of the OPCM array area, enhancing the scalability of the system.

SOPHIE reduces approximately half of the OPCM area and reduces $25\% - 50\%$ of the computation demand and global synchronization overhead while maintaining acceptable solution quality. SOPHIE achieves $3\times$ speedup against state-of-the-art photonic Ising machines on small graphs and $125\times$ speedup against FPGA-based designs on large problems. With the above techniques, SOPHIE alleviates the hardware capacity constraints of Ising machines, offering a scalable and efficient alternative for addressing Ising problems.

REFERENCES

[1] S. Kanamaru, D. Oku, M. Tawada, S. Tanaka, M. Hayashi, M. Yamaoka, M. Yanagisawa, and N. Togawa, "Efficient Ising Model Mapping to Solving Slot Placement Problem," in *2019 IEEE International Conference on Consumer Electronics (ICCE)*. Las Vegas, NV, USA: IEEE, Jan. 2019, pp. 1–6.

[2] C. Johnson, D. H. Allen, J. Brown, S. Vanderwiel, R. Hoover, H. Achilles, C.-Y. Cher, G. A. May, H. Franke, J. Xenedis, and C. Basso, "A wire-speed power^{tm} processor: 2.3GHz 45nm SOI with 16 cores and 64 threads," in *2010 IEEE International Solid-State Circuits Conference - (ISSCC)*. San Francisco, CA, USA: IEEE, Feb. 2010, pp. 104–105.

[3] D. P. Landau and K. Binder, *A guide to Monte Carlo simulations in statistical physics*, 4th ed. Cambridge: Cambridge University Press, 2015.

[4] M. Prabhu, C. Roques-Carmes, Y. Shen, N. Harris, L. Jing, J. Carolan, R. Hamerly, T. Baehr-Jones, M. Hochberg, V. Čeperić, J. D. Joannopoulos, D. R. Englund, and M. Soljačić, "Accelerating recurrent Ising machines in photonic integrated circuits," *Optica*, vol. 7, no. 5, pp. 551–558, May 2020.

[5] U. Benlic and J.-K. Hao, "Breakout Local Search for the Max-Cut Problem," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 3, pp. 1162–1173, Mar. 2013.

[6] M. W. Johnson, M. H. S. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk, E. M. Chapple, C. Enderud, J. P. Hilton, K. Karimi, E. Ladizinsky, N. Ladizinsky, T. Oh, I. Perminov, C. Rich, M. C. Thom, E. Tolkacheva, C. J. S. Truncik, S. Uchaikin, J. Wang, B. Wilson, and G. Rose, "Quantum annealing with manufactured spins," *Nature*, vol. 473, no. 7346, pp. 194–198, 2011.

[7] T. Wang and J. Roychowdhury, "OIM: Oscillator-Based Ising Machines for Solving Combinatorial Optimisation Problems," in *Unconventional Computation and Natural Computation*, I. McQuillan and S. Seki, Eds. Cham: Springer International Publishing, 2019, vol. 11493, pp. 232–256, series Title: Lecture Notes in Computer Science.

[8] 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)*. Seoul, Korea (South): IEEE, Feb. 2021, pp. 749–760.

[9] T. Inagaki, Y. Haribara, K. Igarashi, T. Sonobe, S. Tamate, T. Honjo, A. Marandi, P. L. McMahon, T. Umeki, K. Enbutsu, O. Tadanaga, H. Takenouchi, K. Aihara, K.-i. Kawarabayashi, K. Inoue, S. Utsunomiya, and H. Takesue, "A coherent Ising machine for 2000-node optimization problems," *Science*, vol. 354, no. 6312, pp. 603–606, Nov. 2016.

[10] K. Tatsumura, A. R. Dixon, and H. Goto, "FPGA-Based Simulated Bifurcation Machine," in *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, 2019, pp. 59–66.

[11] G. Yang, C. Demirkiran, Z. E. Kizilates, C. A. R. Ocampo, A. K. Coskun, and A. Joshi, "Processing-in-Memory Using Optically-Addressed Phase Change Memory," in *2023 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, Aug. 2023, pp. 1–6.

[12] J. Feldmann, N. Youngblood, M. Karpov, H. Gehring, X. Li, M. Stappers, M. Le Gallo, X. Fu, A. Lukashchuk, A. S. Raja, J. Liu, C. D. Wright, A. Sebastian, T. J. Kippenberg, W. H. P. Pernice, and H. Bhaskaran, "Parallel convolutional processing using an integrated photonic tensor core," *Nature*, vol. 589, no. 7840, pp. 52–58, Jan. 2021.

[13] M. Wei, K. Xu, B. Tang, J. Li, Y. Yun, P. Zhang, Y. Wu, K. Bao, K. Lei, Z. Chen, H. Ma, C. Sun, R. Liu, M. Li, L. Li, and H. Lin, "Monolithic back-end-of-line integration of phase change materials into foundry-manufactured silicon photonics," *Nature Communications*, vol. 15, no. 1, p. 2786, Mar. 2024.

[14] C. Lian, C. Vagionas, T. Alexoudi, N. Pleros, N. Youngblood, and C. Ríos, "Photonic (computational) memories: Tunable nanophotonics for data storage and computing," *Nanophotonics*, vol. 11, no. 17, pp. 3823–3854, Sep. 2022.

[15] C. Roques-Carmes, Y. Shen, C. Zanoci, M. Prabhu, F. Atieh, L. Jing, T. Dubček, C. Mao, M. R. Johnson, V. Čeperić, J. D. Joannopoulos, D. Englund, and M. Soljačić, "Heuristic recurrent algorithms for photonic Ising machines," *Nature Communications*, vol. 11, no. 1, p. 249, Jan. 2020.

[16] Giovanni Rinaldi, "Rudy graph generator," https://web.stanford.edu/~yyye/yyye/Gset/, 1998.

[17] C. Ríos, M. Stegmaier, P. Hosseini, D. Wang, T. Scherer, C. D. Wright, H. Bhaskaran, and W. H. P. Pernice, "Integrated all-photonic non-volatile multi-level memory," *Nature Photonics*, vol. 9, no. 11, pp. 725–732, Nov. 2015.

[18] J. R. Erickson, N. A. Nobile, D. Vaz, G. Vinod, C. A. R. Ocampo, Y. Zhang, J. Hu, S. A. Vitale, F. Xiong, and N. Youngblood, "Comparing the thermal performance and endurance of resistive and pin silicon microheaters for phase-change photonic applications," *Optical Materials Express*, vol. 13, no. 6, pp. 1677–1688, 2023.

[19] Z. Fang, R. Chen, J. Zheng, A. I. Khan, K. M. Neilson, S. J. Geiger, D. M. Callahan, M. G. Moebius, A. Saxena, M. E. Chen, C. Rios, J. Hu, E. Pop, and A. Majumdar, "Ultra-low-energy programmable non-volatile silicon photonics based on phase-change materials with graphene heaters," *Nature Nanotechnology*, vol. 17, no. 8, pp. 842–848, Aug. 2022.

[20] W. Ma, S. Tan, K. Wang, W. Guo, Y. Liu, L. Liao, L. Zhou, J. Zhou, X. Li, L. Liang, and W. Li, "Practical two-dimensional beam steering system using an integrated tunable laser and an optical phased array," *Applied Optics*, vol. 59, no. 32, p. 9985, Nov. 2020.

[21] C. Wu, H. Yu, S. Lee, R. Peng, I. Takeuchi, and M. Li, "Programmable phase-change metasurfaces on waveguides for multimode photonic convolutional neural network," *Nature Communications*, vol. 12, no. 1, p. 96, Jan. 2021.

[22] C. Ríos, N. Youngblood, Z. Cheng, M. Le Gallo, W. H. P. Pernice, C. D. Wright, A. Sebastian, and H. Bhaskaran, "In-memory computing on a photonic platform," *Science Advances*, vol. 5, no. 2, p. eaau5759, Feb. 2019.

[23] A. Lucas, "Ising formulations of many NP problems," *Frontiers in Physics*, vol. 2, 2014.

[24] M. Gendreau and J.-Y. Potvin, *Handbook of metaheuristics*. Springer US, 2009.

[25] N. Mohseni, P. L. McMahon, and T. Byrnes, "Ising machines as hardware solvers of combinatorial optimization problems," *Nature Reviews Physics*, vol. 4, no. 6, pp. 363–379, May 2022.

[26] D.-W. Systems, "White paper: Computational power consumption and speedup," D-Wave Systems, The Quantum Computing Company, Palo Alto, Canada, Tech. Rep. 14-1005A-D, December 2017.

[27] 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*. New York New York: ACM, Jun. 2022, pp. 508–521.

[28] Y. Shen, N. C. Harris, S. Skirlo, M. Prabhu, T. Baehr-Jones, M. Hochberg, X. Sun, S. Zhao, H. Larochelle, D. Englund, and M. Soljačić, "Deep learning with coherent nanophotonic circuits," *Nature Photonics*, vol. 11, no. 7, pp. 441–446, Jul. 2017.

[29] N. Talati, S. Gupta, P. Mane, and S. Kvatinsky, "Logic Design Within Memristive Memories Using Memristor-Aided loGIC (MAGIC)," *IEEE Transactions on Nanotechnology*, vol. 15, no. 4, pp. 635–650, Jul. 2016.

[30] F. Brückerhoff-Plückelmann, J. Feldmann, H. Gehring, W. Zhou, C. D. Wright, H. Bhaskaran, and W. Pernice, "Broadband photonic tensor core with integrated ultra-low crosstalk wavelength multiplexers," *Nanophotonics*, vol. 11, no. 17, pp. 4063–4072, Sep. 2022.

[31] N. C. Harris, D. Bunandar, A. Joshi, A. Basumallik, and R. Turner, "Passage: A Wafer-Scale Programmable Photonic Communication Substrate," in *2022 IEEE Hot Chips 34 Symposium (HCS)*. Cupertino, CA, USA: IEEE, Aug. 2022, pp. 1–26.

[32] C. Auth, C. Allen, A. Blattner, D. Bergstrom, M. Brazier, M. Bost, M. Buehler, V. Chikarmane, T. Ghani, T. Glassman, R. Grover, W. Han, D. Hanken, M. Hattendorf, P. Hentges, R. Heussner, J. Hicks, D. Ingerly, P. Jain, S. Jaloviar, R. James, D. Jones, J. Jopling, S. Joshi, C. Kenyon, H. Liu, R. McFadden, B. McIntyre, J. Neirynck, C. Parker, L. Pipes, I. Post, S. Pradhan, M. Prince, S. Ramey, T. Reynolds, J. Roesler, J. Sandford, J. Seiple, P. Smith, C. Thomas, D. Towner, T. Troeger, C. Weber, P. Yashar, K. Zawadzki, and K. Mistry, "A 22nm high performance and low-power CMOS technology featuring fully-depleted tri-gate transistors, self-aligned contacts and high density MIM capacitors," in *2012 Symposium on VLSI Technology (VLSIT)*, 2022, pp. 131–132.

[33] M. Guo, J. Mao, S.-W. Sin, H. Wei, and R. P. Martins, "A 5 GS/s 29 mW Interleaved SAR ADC With 48.5 dB SNDR Using Digital-Mixing Background Timing-Skew Calibration for Direct Sampling Applications," *IEEE Access*, vol. 8, pp. 138 944–138 954, 2020.

[34] M. Horowitz, "1.1 Computing's energy problem (and what we can do about it)," *Digest of Technical Papers - IEEE International Solid-State Circuits Conference*, vol. 57, pp. 10–14, 2014.

[35] A. Cho, A. Saxena, M. Qureshi, and A. Daglis, "A Case for CXL-Centric Server Processors," May 2023.
[36] R. Hamerly, T. Inagaki, P. L. McMahon, D. Venturelli, A. Marandi, T. Onodera, E. Ng, C. Langrock, K. Inaba, T. Honjo, K. Enbutsu, T. Umeki, R. Kasahara, S. Utsunomiya, S. Kako, K.-i. Kawarabayashi, R. L. Byer, M. M. Fejer, H. Mabuchi, D. Englund, E. Rieffel, H. Takesue, and Y. Yamamoto, "Experimental investigation of performance differences between coherent Ising machines and a quantum annealer," *Science Advances*, vol. 5, no. 5, p. eaau0823, May 2019.
[37] K. Tatsumura, M. Yamasaki, and H. Goto, "Scaling out Ising machines using a multi-chip architecture for simulated bifurcation," *Nature Electronics*, vol. 4, no. 3, pp. 208–217, Mar. 2021.
[38] M. Booth, S. P. Reinhardt, and A. Roy, "Partitioning Optimization Problems for Hybrid Classical/Quantum Execution," D-Wave, Tech. Rep., Oct. 2017.
[39] D.-W. Systems, "Operation and Timing ; D-Wave System Documentation documentation," https://docs.dwavesys.com/docs/latest/c_qpu_timing.html.
[40] H. Goto, K. Endo, M. Suzuki, Y. Sakai, T. Kanao, Y. Hamakawa, R. Hidaka, M. Yamasaki, and K. Tatsumura, "High-performance combinatorial optimization based on classical mechanics," *Science Advances*, vol. 7, no. 6, p. eabe7953, Feb. 2021.