IEEE Journal on Exploratory Solid-State Computational Devices and Circuits

Received 30 January 2023; revised 20 February 2023; accepted 10 March 2023.
Date of publication 14 March 2023; date of current version 29 May 2023.

Digital Object Identifier 10.1109/JXCDC.2023.3256981

A Full-Stack View of Probabilistic Computing With p-Bits: Devices, Architectures, and Algorithms

SHUVRO CHOWDHURY[1], ANDREA GRIMALDI[2] (Graduate Student Member, IEEE), NAVID ANJUM AADIT[1], SHAILA NIAZI[1], MASOUD MOHSENI[3], SHUN KANAI[4,5], HIDEO OHNO[4,5] (Life Fellow, IEEE), SHUNSUKE FUKAMI[4,5] (Member, IEEE), LUKE THEOGARAJAN[1], GIOVANNI FINOCCHIO[2] (Senior Member, IEEE), SUPRIYO DATTA[6] (Life Fellow, IEEE), and KEREM Y. CAMSARI[1] (Senior Member, IEEE)

[1]Department of Electrical and Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA 93106 USA
[2]Department of Mathematical and Computer Sciences, Physical Sciences and Earth Sciences, University of Messina, 98166 Messina, Italy
[3]Google Quantum AI, Venice, CA 90291 USA
[4]Research Institute of Electrical Communication, Tohoku University, Sendai 980-8577, Japan
[5]Center for Science and Innovation in Spintronics, Tohoku University, Sendai 980-8577, Japan
[6]Elmore Family School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA

CORRESPONDING AUTHOR: S. CHOWDHURY (schowdhury@ucsb.edu)

This work was supported in part by the U.S. National Science Foundation under Grant CCF 2106260, in part by the Office of Naval Research Young Investigator Grant, and in part by the Semiconductor Research Corporation. The work of Shun Kanai was supported by the Japan Science and Technology Agency (JST)-PRESTO under Grant JPMJPR21B2. The work of Shunsuke Fukami was supported in part by JST-CREST under Grant JPMJCR19K3 and in part by MEXT Initiative to Establish Next-generation Novel Integrated Circuits Centers (X-NICS) under Grant JPJ011438. The work of Andrea Grimaldi and Giovanni Finocchio was supported in part by "The Italian Factory of Micromagnetic Modeling and Spintronics" funded by the Italian Ministry of University and Research (MUR) under Project PRIN 2020LWPKH7, and in part by the Petaspin Association (www.petaspin.com).

ABSTRACT The transistor celebrated its 75th birthday in 2022. The continued scaling of the transistor defined by Moore's law continues, albeit at a slower pace. Meanwhile, computing demands and energy consumption required by modern artificial intelligence (AI) algorithms have skyrocketed. As an alternative to scaling transistors for general-purpose computing, the integration of transistors with unconventional technologies has emerged as a promising path for domain-specific computing. In this article, we provide a full-stack review of probabilistic computing with p-bits as a representative example of the energy-efficient and domain-specific computing movement. We argue that p-bits could be used to build energy-efficient probabilistic systems, tailored for probabilistic algorithms and applications. From hardware, architecture, and algorithmic perspectives, we outline the main applications of probabilistic computers ranging from probabilistic machine learning (ML) and AI to combinatorial optimization and quantum simulation. Combining emerging nanodevices with the existing CMOS ecosystem will lead to probabilistic computers with orders of magnitude improvements in energy efficiency and probabilistic sampling, potentially unlocking previously unexplored regimes for powerful probabilistic algorithms.

INDEX TERMS Artificial intelligence (AI), combinatorial optimization, domain-specific hardware, machine learning (ML), p-bits, p-computers, quantum simulation, sampling, spintronics, stochastic magnetic tunnel junctions (sMTJs).

I. INTRODUCTION

THE slowing down of the Moore era of electronics has coincided with the recent revolution in machine-learning (ML) and artificial intelligence (AI) algorithms. In the absence of steady transistor scaling and energy improvements, training and maintaining large-scale ML models in data centers have become a significant energy concern [1]. The widespread implementation of AI, particularly in industries such as autonomous vehicles [2], is an indication that the energy crisis caused by large-scale ML models is not just a data center problem, but a global concern.

Efforts of extending the Moore era of electronics by improving conventional transistor technology continue vigorously. Examples of this approach include

3-D heterogeneous integration, 2-D materials for transistors and interconnects [3], new transistor physics via negative capacitance [4], [5], or entirely new approaches using spintronic and magnetoelectric phenomena to build energy-efficient switches [6], [7].

A complementary approach to extending Moore's law is to *augment* the existing CMOS ecosystem with emerging, nonsilicon nanotechnologies [8], [9]. One way to achieve this goal is through heterogeneous CMOS + X architectures, where X stands for a CMOS-compatible nanotechnology. For example, X can be magnetic, ferroelectric, memristive, or photonic systems. We also discuss an example of this complementary approach, the combination of CMOS with magnetic memory technology, purposefully modified to build probabilistic computers.

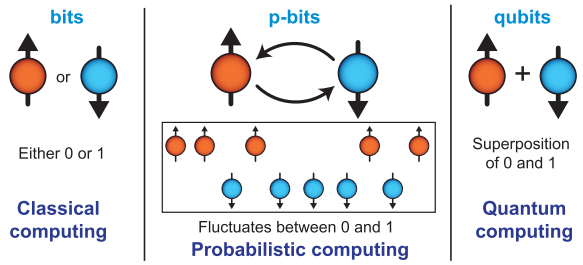

FIGURE 1. Bit, p-bit, and qubit. Each column shows a schematic representation of the basic computational units of classical computing (left), probabilistic computing (middle), and quantum computing (right). These are, respectively, the bit, the p-bit, and the qubit.

II. FULL-STACK VIEW AND ORGANIZATION

Research on probabilistic computing with p-bits originated at the device and physics level, first with stable nanomagnets [10], followed by low-barrier nanomagnets [11], [12]. In [12], the p-bit was formally defined as a binary stochastic neuron realized in hardware. In both approaches with stable and unstable nanomagnets, the basic idea is to exploit the natural mapping between the intrinsically noisy physics of nanomagnets to the mathematics of general probabilistic algorithms [e.g., Monte Carlo, Markov Chain Monte Carlo (MCMC)]. Such a notion of *natural computing* where physics is matched to computation was clearly laid out by Feynman [13] in his celebrated *Simulating Physics with Computers* talk. Subsequent work on p-bits defined it as an abstraction between bits and qubits (see Fig. 1) with the possibility of different physical implementations. In addition to searching for energy-efficient realizations of single devices, p-bit research has extended to finding efficient architectures (through massive parallelization, sparsification [14], and pipelining [15]) along with the identification of promising application domains. This full-stack research program covering hardware, architecture, algorithms, and applications is similar to the related field of quantum computation where a large degree of interdisciplinary expertise is required to move the field forward (see the related reviews [16], [17]). The purpose of this article is to serve as a consolidated summary of recent developments with new results in hardware, architectures, and algorithms. We provide concrete and previously unpublished examples of ML and AI, combinatorial optimization, and quantum simulation with p-bits (see Fig. 2).

III. FUNDAMENTALS OF p-COMPUTING

A large family of problems (see Fig. 2) can be encoded to coupled p-bits evolving according to the following equations [12]:

$$m_i = \text{sign}\left[\tanh(\beta I_i) - r_{[-1,+1]}\right] \tag{1}$$

$$I_i = \sum_j W_{ij} m_j + h_i \tag{2}$$

where m_i is defined as a bipolar variable ($m_i \in \{-1, +1\}$), r is a uniform random number drawn from the interval $[-1, 1]$, $[W]$ is the coupling matrix between the p-bits, β is the inverse temperature, and $\{h\}$ is the bias vector. In physical implementations, it is often more convenient to represent p-bits as binary variables, $s_i \in \{0, 1\}$. A straightforward conversion of (1) and (2) is possible using the standard transformation, $m \rightarrow 2s - 1$ [18].

As stated, (1) and (2) do not place any restrictions on $[W]$, which may be a symmetric or asymmetric matrix. If an updated order of p-bits is specified, these equations take the coupled p-bit system to a well-defined steady-state distribution defined by the eigenvector (with eigenvalue $+1$) of the corresponding Markov matrix [12]. Indeed, in the case of Bayesian (belief) networks defined by a directed graph, updating the p-bits from parent nodes to child nodes takes the system to a steady-state distribution corresponding to that obtained from Bayes' theorem [19].

If the $[W]$ matrix is symmetric, one can define an energy, E, whose negated partial derivative with respect to p-bit m_i gives rise to (2)

$$E(m_1, m_2, \ldots,) = -\left(\sum_{i<j} W_{ij} m_i m_j + \sum_i h_i m_i\right). \tag{3}$$

In this case, the steady-state distribution of the network is described by [20]

$$p_i = \frac{1}{Z} \exp\left(-\beta E_i\right) \tag{4}$$

also known as the Boltzmann law. As such, iterating a network of p-bits described by (1) and (2) eventually approximates the Boltzmann distribution which can be useful for probabilistic sampling and optimization. The approximate sampling avoids the intractable problem of exactly calculating Z. Remarkably, for such undirected networks, the steady-state distribution is invariant with respect to the update order of p-bits, as long as connected p-bits are not updated at the same time (more on this later). This feature is highly reminiscent of natural systems where asynchronous dynamics make parallel updates highly unlikely and the update order does not change the equilibrium distribution. Indeed, this gives the hardware implementation of asynchronous networks of p-bits massive parallelism and flexibility in design.

The energy functional defined by (3) is often the starting point of discussions in the related field of Ising machines [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43] with different implementations (see [44] for a comprehensive review). In the case of p-bits, however, we view (1) and (2) more fundamental than (3) because the former can also be

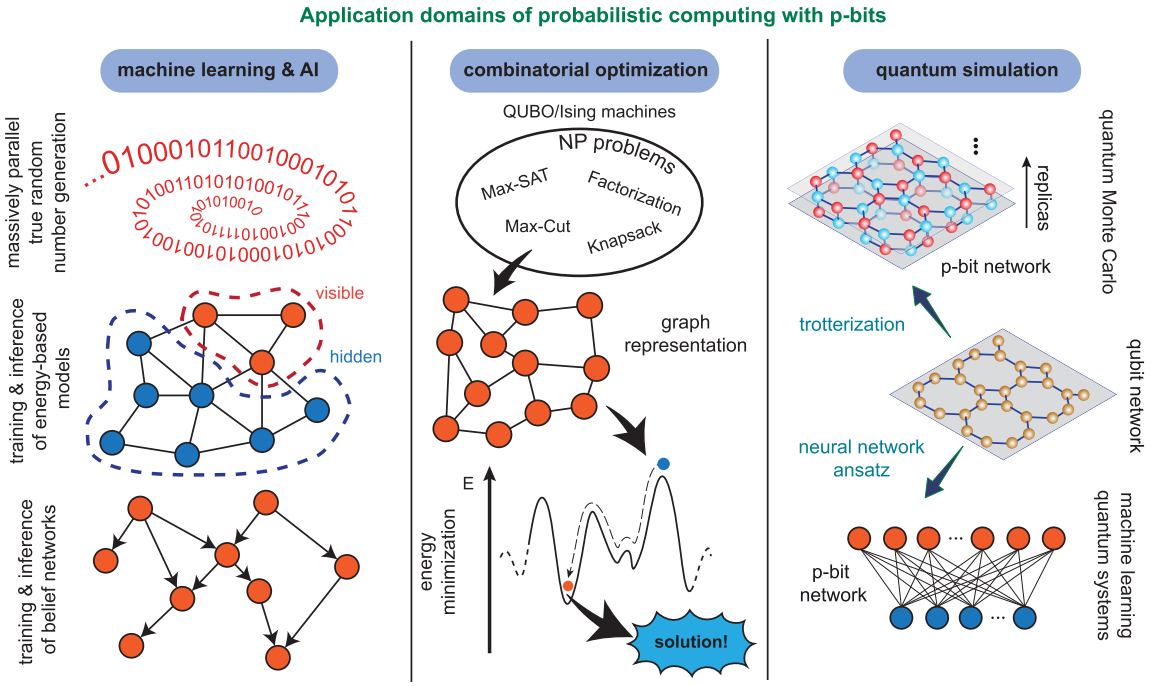

FIGURE 2. Applications of probabilistic computing. Potential applications of p-bits are illustrated. The list broadly includes problems in combinatorial optimization, probabilistic ML, and quantum simulation.

used to approximate hard inference on *directed* networks, while the latter always relies on undirected networks. Compared to undirected networks using Ising machines, work on directed neural networks for Bayesian inference has been relatively scarce, although there are exciting developments [19], [45], [46], [47], [48], [49], [50].

Finally, the form of (3) restricts the type of interactions between p-bits to a linear one since the energy is quadratic. Even though higher-order interactions (k-local) between p-bits are possible [18] (also discussed in the context of Ising machines [51], [52]), such higher-order interactions can always be constructed by combining a standard probabilistic gate set at the cost of extra p-bits. In our view, in the case of electronic implementation with scalable p-bits, trading an increased number of p-bits for simplified interconnect complexity is almost always favorable. That being said, algorithmic advantages and the better representative capabilities of higher-order interactions are actively being explored [51], [53].

IV. HARDWARE: PHYSICAL IMPLEMENTATION OF p-BITS

A. p-BITS

The p-bit defined in (1) describes a tunable and discrete random number generator. Its physical implementation includes a broad range of options from noisy materials to analog and digital CMOS (see Fig. 3). The digital CMOS implementations of p-bits often consist of a pseudorandom number generator (PRNG) (r), a lookup table for the activation function (tanh), and a threshold to generate a one-bit output. Digital input with a specified fixed point precision (e.g., ten bits with one sign, six integers, and three fractional) provides tunability through the activation function. Digital p-bits have been very useful in prototyping probabilistic computers up to tens of thousands of p-bits [14], [54], [55]. They also serve a useful purpose to illustrate *why* analog or mixed-signal implementations of p-bits with nanodevices are necessary. Even using some of the most advanced field programmable gate arrays (FPGAs), the footprint of a digital p-bit is very large: Synthesizing such digital p-bits with PRNGs of varying quality of randomness results in tens of thousands of individual transistors. In single FPGAs that do not use time-division multiplexing of p-bits or off-chip memory, only about 10 000–20 000 p-bits with 100 000 weights (sparse graphs with degree 5–10) fit, even within high-end devices [14].

On the other hand, using nanodevices such as CMOS-compatible stochastic magnetic tunnel junctions (sMTJs), millions of p-bits can be accommodated in single cores due to the scalability achieved by the magnetoresistive random access memory (MRAM) technology, exceeding 1-Gb MRAM chips [56], [57]. However, before the stable MTJs can be controllably made stochastic, challenges at the material and device level must be addressed [58], [59] with careful magnet designs [60], [61], [62]. Different flavors of magnetic p-bits exist [63], [64], [65], [66], for a recent review (see [67]). Unlike synchronous or trial-based stochasticity (see [68]) that requires continuous resetting, the temporal noise of low-barrier nanomagnets makes them ideally suited to build autonomous, physics-inspired probabilistic computers, providing a constant stream of tunably random bits [69]. Following earlier theoretical predictions [70], [71], [72], recent breakthroughs in low-barrier magnets have shown great promise, using stochastic MTJs with in-plane anisotropy where fluctuations can be of the order of nanoseconds [73], [74], [75]. Such near-zero barrier nanomagnets should be more tolerant to device variations because when the energy-barrier Δ is low, the usual exponential dependence of fluctuations is much less pronounced. These stochastic MTJs may be used in electrical circuits with a few

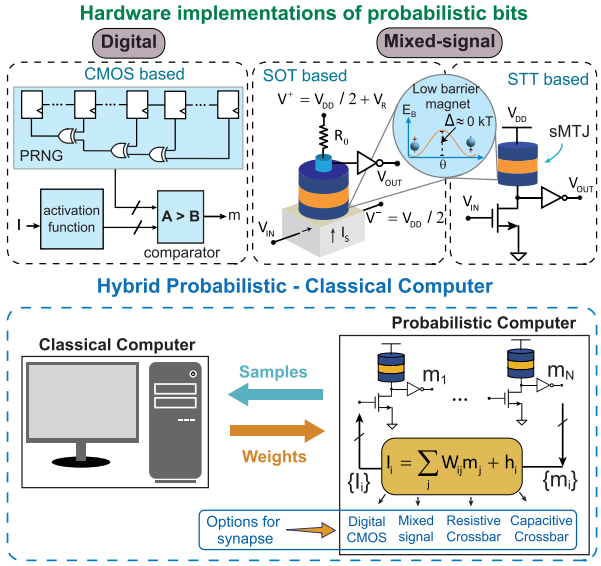

FIGURE 3. Different hardware options for building a probabilistic computer. Top: Various magnetic implementations of a p-bit. These include both digital (CMOS) and mixed-signal implementations (based on, e.g., sMTJ with low-barrier magnets). Bottom: Hybrid of classical and probabilistic computing schemes is shown where the classical computer generates weights and programs the probabilistic computer. The probabilistic computer then generates samples accordingly with high throughput and sends them back to the classical computer for further processing. Like the building blocks of p-bits, the synapse of the probabilistic computer can be designed in several ways, including digital, analog, and a mix of both techniques.

additional transistors (see Fig. 3) to build hardware p-bits. Two flavors of stochastic MTJ-based p-bits were proposed in [12] (spin orbit torque (SOT)-based) and in [70] (spin transfer torque (STT)-based). Both these p-bits have now been experimentally demonstrated in [18], [76], and [77] (STT) and in [78] (SOT). While many other implementations of p-bits are possible, from molecular nanomagnets [79] to diffusive memristors [80], resistive random access memory (RRAM) [81], perovskite nickelates [82], and others, two additional advantages of the MRAM-based p-bits are the proven manufacturability (up to billion-bit densities) and the amplification of room-temperature noise. Even with the thermal energy of kT in the environment, magnetic switching causes large resistance fluctuations in MTJs, creating hundreds of millivolts of change in resistive dividers [70]. Typical noise on resistors (or memristors) is limited by the $(kT/C)^{1/2}$ limit which is far lower (millivolts) even at extremely low capacitances (C). This feature of stochastic MTJs ensures that they do not require explicit amplifiers [83] at each p-bit, which can become prohibitively expensive in terms of area and power consumption. Estimates of sMTJs-based p-bits suggest that they can create a random bit using 2 fJ per operation [18]. Recently, a CMOS-compatible single-photon avalanche diode-based implementation of p-bits showed similar, amplifier-free operation [84] and the search for the most scalable, energy-efficient hardware p-bit using alternative phenomena continues.

B. SYNAPSE

The second central part of the p-computer architecture is the synapse, denoted by (2). Much like the hardware p-bit, there are several different implementations of synapses ranging from digital CMOS, analog/mixed-signal CMOS, as well as resistive [85] or capacitors crossbars [86], [87]. The synaptic equation looks like the traditional matrix–vector product (MVP) commonly used in ML models today, however, there is a crucial difference, thanks to the discrete p-bit output (0 or 1), the MVP operation is simply an addition over the active neighbors of a given p-bit. This makes the synaptic operation simpler than continuous multiplication and significantly simplifies digital synapses. In analog implementations, the use of in-memory computing techniques through charge accumulation could be useful with the added simplification of digital outputs of p-bits [88], [89].

It is important to note how the p-bit and the neuron for eventually integrated p-bit applications can be mixed and matched, as an example of creatively combining these pieces (see the FPGA-stochastic MTJ combination reported in [77]). The best combination of scalable p-bits and synapses may lead to energy-efficient and large-scale p-computers. At this time, various possibilities exist with different technological maturity.

V. ARCHITECTURE CONSIDERATIONS

A. GIBBS SAMPLING WITH p-BITS

The dynamical evolution of (1) and (2) relies on an *iterated* updating scheme where each p-bit is updated one after the other based on a predefined (or random) update order. This iterative scheme is called Gibbs sampling [90], [91]. Virtually all applications discussed in Fig. 2 benefit from accelerating Gibbs sampling, attesting to its generality.

In a standard implementation of Gibbs sampling in a synchronous system, p-bits will be updated one by one at every clock cycle as shown in Fig. 4(a). It is crucial to ensure that the effective input each p-bit receives through (2) is computed *before* the p-bit updates. As such, T_{clk} has to be longer than the time it takes to compute (2). In this setting, a graph with N p-bits will require N clock cycles (NT_{clk}) to perform a complete sweep, where T_{clk} is the clock period. This requirement makes Gibbs sampling a fundamentally serial and slow process.

A much more effective approach is possible by the following observation: even though updates between *connected* p-bits need to be sequential, if two p-bits are not directly connected, updating one of them does not directly change the input of the other through (2). Such p-bits can be updated in parallel without any approximation. Indeed, one motivation of designing *restricted* Boltzmann machines (RBMs; see [92]) over unrestricted BMs is to exploit this parallelism: RBMs consist of separate layers (bipartite) that can be updated in parallel. However, this idea can be taken further. If the underlying graph is *sparse*, it is often easy to split it into disconnected chunks by coloring the graph using a few colors. Even though finding the *minimum* number of colors is an NP-hard problem [93], heuristic coloring algorithms (such as Dsatur [94]) with polynomial complexity can color the graph very quickly, without necessarily finding a minimum. In this

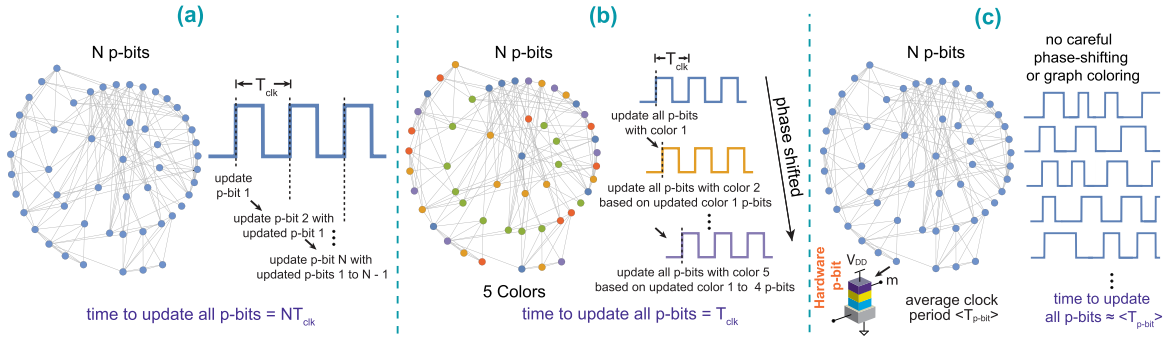

FIGURE 4. Architectures of p-computer. (a) Synchronous Gibbs: all p-bits are updated sequentially. N p-bits need N clock cycles (NT_{clk}) to perform a complete sweep, T_{clk} being the clock period. (b) Pseudo-asynchronous Gibbs: a sparse network can be colored into a few disjoint blocks where connected p-bits are assigned a different color. Phase-shifted clocks update the color blocks one after the other. N p-bits need ≈ one clock cycle T_{clk} to perform a complete sweep, reducing O(N) complexity of a sweep to O(1), where we assume the number of colors $c \ll N$. (c) Truly asynchronous Gibbs: a hardware p-bit (e.g., a stochastic MTJ-based p-bit) provides an asynchronous and random clock with period $\langle T_{p\text{-bit}} \rangle$. N p-bits need approximately one clock to perform a complete sweep, as long as synapse time is less than the clock on average. No graph coloring or engineered phase shifting is required.

context, obtaining the minimum coloring is not critical, and sparse graphs typically require a few colors.

Such an approach was taken on sparse graphs (with no regular structure) to design a massively parallel implementation of Gibbs sampling in [14] [see Fig. 4(b)]. Connected p-bits are assigned a different color, and unconnected p-bits are assigned the same color. Equally phase-shifted and same-frequency clocks update the p-bits in each color block one by one. In this approach, a graph with N p-bits requires only one clock cycle (T_{clk}) to perform a complete sweep, reducing O(N) complexity for a full sweep to O(1), assuming the number of colors is much less than N. Therefore, the key advantage of this approach is that the p-computer becomes faster with larger graphs since probabilistic "flips per second," a key metric measured by tensor processing unit (TPU) and GPU implementations [95], [96] linearly increases with the number of p-bits. It is important to note that these TPU and GPU implementations also solve Ising problems in sparse graphs, however, their graph degrees are usually restricted to 4 or 6, unlike more irregular and higher degree graphs implemented in [14].

We term this graph-colored architecture the pseudo-asynchronous Gibbs because while it is technically synchronized to out-of-phase clocks, it embodies elements of the truly asynchronous architecture we discuss next. While graph coloring algorithmically increases sampling rates by a factor of N, it still requires a careful design of out-of-phase clocks. A much more radical approach is to design a *truly asynchronous* Gibbs sampler as shown in Fig. 4(c). Here, the idea is to have hardware building blocks with naturally asynchronous dynamics, such as an sMTJ-based p-bit. In such a p-bit, there exists a natural "clock," $\langle T_{p\text{-bit}} \rangle$, defined by the average lifetime of a Poisson process [97]. As long as $\langle T_{p\text{-bit}} \rangle$ is not faster than the average synapse time ($t_{synapse}$) to calculate (2), the network still updates N spins in a single $\langle T_{p\text{-bit}} \rangle$ timescale. This is because the probability of simultaneous updates is extremely low in a Poisson process and further reduced in highly sparse graphs.

In fact, preliminary experiments implementing such truly asynchronous p-bits with ring-oscillator activated clocks show that despite making occasional parallel updates, the asynchronous p-computer performs similarly compared to the pseudo-asynchronous system where incorrect updates are avoided with careful phase-shifting [98]. The main appeal of the truly asynchronous Gibbs sampling is the lack of any graph coloring and phase shift engineering while retaining the same massive parallelism as N p-bits requires approximately a single $\langle T_{p\text{-bit}} \rangle$ to complete a sweep. Given that the FPGA-based p-computers already provide about a 10× improvement in sampling throughput to optimized TPU and GPUs [14], such asynchronous systems are promising in terms of scalability. Stochastic MTJ-based p-bits should be able to reach high densities on a single chip. Around 20 W of projected power consumption can be reached considering 20-μW p-bit/synapse combinations at 1M p-bit density [54], [60], [99]. The ultimate scalability of magnetic p-bits is a significant advantage over alternative approaches based on electronic or photonic devices.

B. SPARSIFICATION

Both the pseudo-asynchronous and the truly asynchronous parallelisms require *sparse* graphs to work well. The first problem is the number of colors: if the graph is dense, it requires a lot of colors, making the architecture very similar to the standard serial Gibbs sampling.

The second problem with a dense graph is the synapse time $t_{synapse}$. If many p-bits have a lot of neighbors, the synapse unit needs to compute a large sum before the next update. If the time between two consecutive updates is $\langle T_{p\text{-bit}} \rangle$, it requires $t_{synapse} \ll \langle T_{p\text{-bit}} \rangle$ to avoid information loss and reach the correct steady-state distribution [54], [101].

However, if the graph is sparse, each p-bit has fewer connections, and the updates can be faster without any dropped messages. Any graph can be sparsified using the technique proposed in [14], similar in spirit to the minor-graph embedding (MGE) approach pioneered by D-Wave [102], even though the objective here is to not find an embedding but to sparsify an existing graph. The key idea is to split p-bits into different copies, using ferromagnetic COPY gates. These p-bits distribute the original connections among them, resulting in identical copies with fewer connections. An important point is that the ground state of the original graph remains unchanged [14], so the method does not involve approximations, unlike other sparsification techniques, for example, based on low-rank approximations [103].

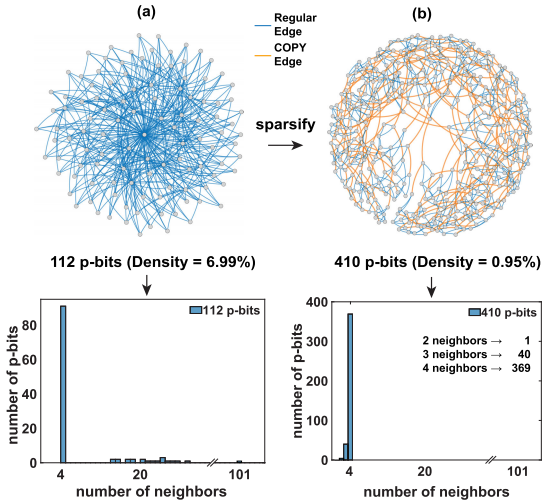

FIGURE 5. (a) Original graph of a 3SAT instance uf20-01.cnf [100] having 112 p-bits and a graph density of 6.99%. (Graph density, $\rho = 2|E|/(|V|^2 - |V|)$, where $|E|$ is the number of edges and $|V|$ is the number of vertices in the graph.) Some of the p-bits have many local neighbors up to 101 neighbors as shown in the histogram that slows down the synapse and the p-bits need to update slowly. (b) Sparsified graph of the same instance having 410 p-bits. COPY gates are inserted between each pair of copies of the same p-bits (COPY edges are highlighted in orange). The graph has a density of 0.95% and the maximum number of neighbors is limited to 4. The synapse operations are now faster and hence the p-bits can be updated faster. Even though the example shown here starts from a low-density graph, the sparsification algorithm we give is general and applicable to any graph.

Fig. 5(a) shows an example of this process where the original graph of a satisfiability (3SAT) instance has been sparsified as shown in Fig. 5(b). Irrespective of the input graph size, a sparsified graph has fewer connections locally and thus the neurons hardly ever need to be slowed down. One disadvantage of this technique is the increased number of p-bits; however, the reduced synapse complexity and the possibility of massive parallelization outweigh the costs incurred by additional p-bits, which we consider to be cheap in scaled, nanodevice-based implementations.

VI. ALGORITHMS AND APPLICATIONS

A. COMBINATORIAL OPTIMIZATION VIA INVERTIBLE LOGIC

When using the Ising model to solve an optimization problem, the first step is to provide a mapping between the Ising model and the problem to be solved. Early work on quantum annealing stimulated by D-Wave's quantum annealers generated a significant amount of useful research in this area [104], some of which are being adopted by quantum-inspired classical annealers. There are usually many different ways to find a mapping, for example, some strategies may employ more nodes than others to encode the same instance, while others might result in graphs with topology unsuited to the computational architecture of choice. In this context, the invertible logic approach introduced in [12] stands out for its flexibility and sparse encodings.

The process of mapping an instance into an Ising model can be broadly summarized into three steps, as illustrated by Fig. 6. In Fig. 6, the steps of the invertible logic encoding of three combinatorial optimization problems, maximum satisfiability (left column), number partitioning (middle column), and knapsack (right column), are shown. First, each problem is formalized into a tight mathematical formulation (top row). Next, the problem is mapped into an invertible Boolean logic circuit (central row), meaning that each logic gate can be operated using any terminals as input–output nodes (similar to those discussed in the context of quantum annealing [105], [106] and memcomputing [107], [108]). Finally, the probabilistic circuit is algorithmically encoded into an Ising model (bottom row). Each logic gate has several Ising encodings that map the energy landscape of its logic operator. After the Boolean logic formulation of a problem, this step can be automated in standardized synthesis tools. The overall approach results in relatively sparse circuits, as illustrated in the bottom row of Fig. 6, where all three problems show similarly sparse matrices $[W]$, with bias vectors $\{h\}$ shown under.

The key advantage of this approach, compared to heuristic and dense formulations of [104], is due to the generality of Boolean logic, quite similar to how present-day digital VLSI circuits are constructed in sparse, hardware-aware networks using billions of transistors. As such, much of the existing ecosystem of high-level synthesis can be directly used to find invertible logic-based encodings for general optimization problems.

B. MACHINE LEARNING: ENERGY-BASED MODELS

Energy-efficient ML with BMs is a promising application for probabilistic computers with a recent experimental demonstration in [76]. Mainstream ML algorithms are designed and chosen with CPU implementation in mind and hence some models are heavily preferred over others even though they are often less powerful. For example, the use of RBMs over the more powerful unrestricted or deep BMs is motivated by the former's efficient software implementation in synchronous systems. However, by exploiting the technique of sparsity and massively parallel architecture described earlier, fast Gibbs sampling with deep Boltzmann machines (DBMs) can dramatically improve state-of-the-art ML applications like visual object recognition and generation, speech recognition, autonomous driving, and many more [109]. Here, we present an example where a sparse DBM is trained with MNIST handwritten digits (see Fig. 7). We randomly distribute the visible and hidden units on the sparse DBM with massively parallel pseudo-asynchronous architecture that yields multiple hidden layers as shown in Fig. 7(c).

Contrasting with earlier unconventional computing approaches where the MNIST dataset is reduced to much smaller sizes [110], [111], we show how the full MNIST dataset (60 000 images and no downsampling) can be trained using p-computers in FPGAs. We use 1200 mini-batches having 50 images in each batch to train the network using the contrastive divergence (CD) algorithm. The process of learning is accomplished using a hybrid probabilistic and classical computer setup. The classical computer computes the gradients and generates new weights, while the p-computer generates samples according to those weights [see Fig. 7(b)]. During the positive phase of sampling, the p-computer operates in its clamped condition under the direct

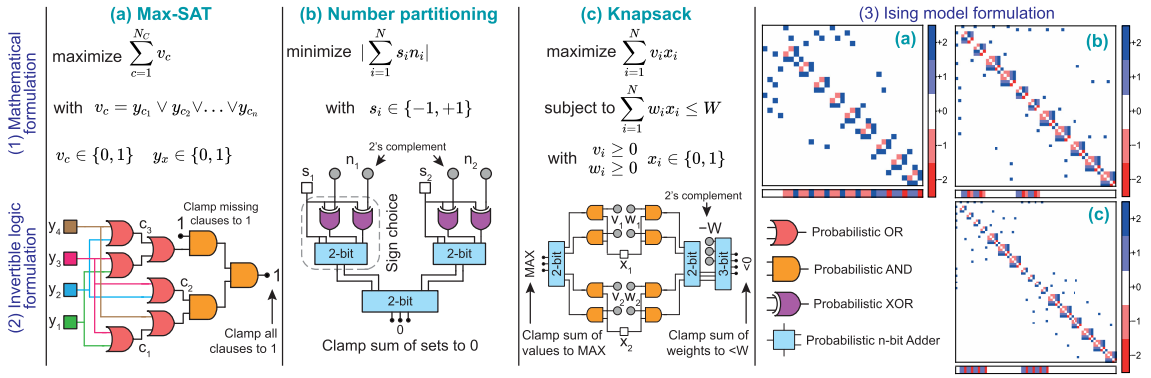

FIGURE 6. **Invertible logic encoding. The encoding process of three optimization problems—(a) maximum satisfiability problem, (b) number partitioning, and (c) knapsack problem—is streamlined and visually summarized into three steps: (1) problem first has to be condensed into a concise mathematical formulation; then, (2) invertible Boolean circuit that topologically maps the problem is conceived; finally, (3) invertible Boolean circuit is converted into an Ising model using probabilistic and/or/not gates [14].**

influence of the training samples. In the negative phase, the p-computer is allowed to run freely without any environmental input. After training, the deep network not only can classify images, but also generate images. For any given label, the network can create a new sample (not present in the training set) [see Fig. 7(d)]. This is an important feature of energy-based models and is commonly demonstrated with diffusion models [112].

C. QUANTUM SIMULATION

One primary motivation for building quantum computers is to simulate large quantum many-body systems and understand the exotic physics offered by them [113]. Two major challenges with quantum computers are the necessity of using cryogenic operating temperatures and the vulnerability to noise, rendering quantum computers impractical, especially considering practical overheads [114]. Simulating these systems with classical computers is often extremely time-consuming and mostly limited to small systems. One potential application of p-bits is to provide a room-temperature solution to boost the simulation speed and potentially enable the simulation of large-scale quantum systems. Significant progress has been made toward this end in recent years.

1) SIMULATING QUANTUM SYSTEMS WITH TROTTERIZATION

One approach is to build a p-computer enabling the scalable simulation of sign-problem-free quantum systems by accelerating standard Quantum Monte Carlo (QMC) techniques [115]. The basic idea is to replace the qubits in the original lattice with hardware p-bits and replicate the new lattice according to the Suzuki–Trotter transformation [116]. Recently, the convergence time of a 2-D square-octagonal qubit lattice initially prepared in a topologically obstructed state was compared among a CPU, a physical quantum annealer [117], and a p-computer (both digital and analog) [118]. For this particular problem, it was shown that an FPGA-based p-computer emulator can be around 1000 times faster than an optimized C++ (CPU) program. Based on SPICE simulations of a small p-computer, we project that significant further acceleration should be possible with a truly asynchronous implementation. Probabilistic computers

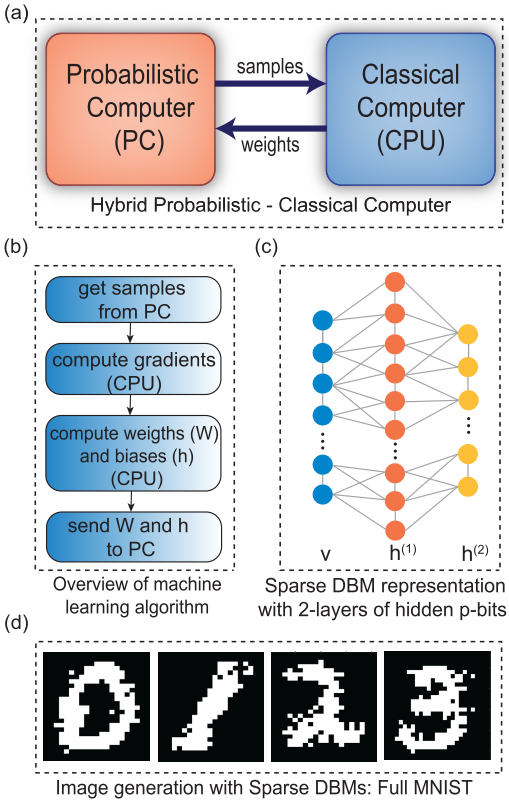

FIGURE 7. **Generative neural networks with p-bits. (a) Hybrid computing scheme with a probabilistic computer and a classical computer is demonstrated where the probabilistic computer generates samples according to the weights given by the CPU with a sampling speed of around 100 flips/ns. (b) Overview of the learning procedure for the hybrid setup. Receiving the samples from the probabilistic computer, the CPU computes the gradient, updates the weights and biases, and sends them back to the probabilistic computer until converged. (c) Sparse DBM is utilized here as a hardware-aware graph that can be represented with multiple hidden layers of p-bits. Both interlayer and intralayer connections are allowed between visible and hidden units. (d) Images shown here are generated with a sparse DBM of 4264 p-bits after training the network with full MNIST. The label p-bits are clamped to a specific image and the network evolves to that image by annealing the system from $\beta = 0$ to $\beta = 5$ with a step size of 0.125.**

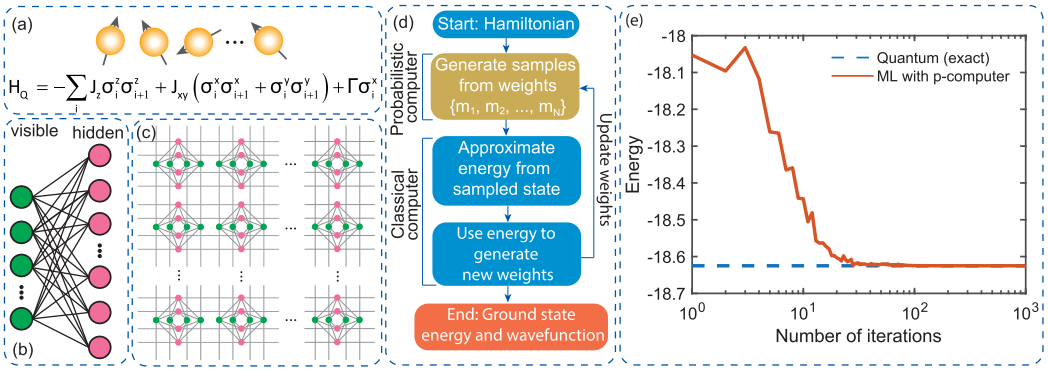

FIGURE 8. ML quantum systems with p-bits. (a) Heisenberg Hamiltonian with a transverse field ($\Gamma = +1$) is applied to an FM coupled ($J_Z = +1$ and $J_{xy} = +0.5$) linear chain of 12 qubits with periodic boundary. (b) To obtain the ground state of this quantum system, an RBM is employed with 12 visible and 48 hidden nodes, where all nodes in the visible layer are connected to all nodes in the hidden layer. (c) This ML model is then embedded into a hardware-amenable sparse p-bit network arranged in a chimera graph using MGE. We use a coupling strength of 1.0 among the replicated visible and hidden nodes in the embedded p-bit network. (d) Overview of the ML algorithm and the division of workload between the probabilistic and classical computers in a hybrid setting is shown. (e) FPGA emulation of this probabilistic computer performs variational ML in tandem with a classical computer, converging to the quantum (exact) result as shown.

can be used for quantum Hamiltonians beyond the usual Transverse Field Ising Model, such as the antiferromagnetic Heisenberg Hamiltonian [119] and even for the emulation of gate-based quantum computers [120]. However, for generic Hamiltonians (e.g., random circuit sampling), the number of samples required in naive implementations seems to grow exponentially [120] due to the notorious *sign-problem* [121]. However, clever basis transformations [122] might mitigate or cure the sign problem [123] in the future.

2) MACHINE-LEARNING QUANTUM MANY-BODY SYSTEMS

With the great success of ML and AI algorithms, training stochastic neural networks (such as BMs) to approximately solve the quantum many-body problem starting from a variational guess has generated great excitement [124], [125], [126] and is considered to be a fruitful combination of quantum physics and ML [127]. These algorithms are typically implemented in high-level software programs, allowing users to choose from various network models and sizes according to their needs. However, as with classical ML, the difficulty of training strongly hinders the use of deeper and more general models. With scaled p-computers using millions of magnetic p-bits, massively parallel and energy-efficient *hardware* implementations of the more general unrestricted/deep BMs may become feasible, paving the way to simulate practical quantum systems.

To demonstrate one such example of this approach, we show how p-bits laid out in sparse, hardware-aware graphs can be used for ML quantum systems (see Fig. 8). The objective of this problem is to find the ground state of a many-body quantum system, in this case, a 1-D FM Heisenberg Hamiltonian with an external transverse field. We start with an RBM, which is one of the simplest neural network models, and use its functional form as the variational guess for the ground state probabilities (the wave function is obtained by taking the square root of probabilities according to the Born rule). A combination of probabilistic sampling and weight updates gradually adjusts the variational guess such that the final guess points to the ground state of the quantum Hamiltonian. Emulating this variational ML approach with p-bits requires a few more steps. An RBM network contains all-to-all connections between the visible and hidden layers which are not conducive for scalable p-computers because of the large fan-out demanded by the all-to-all connectivity. An alternative is to map the RBM onto a sparse graph through MGE [102]. Using a hybrid setup with fast sampling in a probabilistic computer coupled with a classical computer, the iterative process of sampling and weight updating can then be performed. The key advantage of having a massively parallel and fast sampler is the selection of higher-quality states of the wave function to update the variational guess. Fig. 8 shows an example simulation of how a p-computer learns the ground state of a 1-D FM Heisenberg model. The scaling of p-computers using magnetic p-bits may allow much larger implementations of quantum systems in the future.

D. OUTLOOK: ALGORITHMS AND APPLICATIONS BEYOND

Despite the large range of applications we discussed in the context of p-bits, much of the sampling algorithms have been either standard MCMC or generic simulated annealing-based approaches. Future possibilities involve more sophisticated sampling and annealing algorithms such as parallel tempering (PT) (see [128], [129] for some initial investigations). Further improvements to hardware implementation include *adaptive* versions of PT [130] as well as sophisticated nonequilibrium Monte Carlo (NMC) algorithms [131]. Ideas involving *overclocking* p-bits such that they violate the $t_{\text{synapse}} \ll \langle T_{\text{p-bit}} \rangle$ requirement for further improvement [14] or sharing synaptic operations between p-bits [82] could also be useful. A combination of these ideas with *algorithm-architecture-device* codesigns may lead to orders of magnitude improvement in sampling speeds and quality. In this context, as a sampling throughput metric, increasing flips/ns is an important goal. In addition, solution quality, the possibility of cluster updates or algorithmic techniques also need to be considered carefully. Given the plethora of approaches from multiple communities, we also hope that model problems, benchmarking studies comparing different Ising machines, probabilistic

accelerators, physical annealers, and dynamical solvers will be performed in the near future by all practitioners, including ourselves.

We believe that the codesign of algorithms, architectures, and devices for probabilistic computing may not only help mitigate the looming energy crisis of ML and AI, but also lead to systems that may unlock previously inaccessible regimes using powerful probabilistic (randomized) algorithms [132]. Just as the emergence of powerful GPUs made the well-known backpropagation algorithm flourish, probabilistic computers could lead us to previously unknown territory of energy-based AI models, combinatorial optimization, and quantum simulation. This research program requires a concerted effort and interdisciplinary expertise from all across the stack and ties into the larger vision of unconventional computing forming in the community [133].

ACKNOWLEDGMENT

Supriyo Datta has a financial interest in Ludwig Computing.

REFERENCES

[1] D. Patterson et al., "Carbon emissions and large neural network training," 2021, *arXiv:2104.10350*.

[2] S. Sudhakar, V. Sze, and S. Karaman, "Data centers on wheels: Emissions from computing onboard autonomous vehicles," *IEEE Micro*, vol. 43, no. 1, pp. 29–39, Jan. 2023.

[3] R. Chau, "Process and packaging innovations for Moore's law continuation and beyond," in *IEDM Tech. Dig.*, Dec. 2019, p. 1.

[4] J. C. Wong and S. Salahuddin, "Negative capacitance transistors," *Proc. IEEE*, vol. 107, no. 1, pp. 49–62, Jan. 2019.

[5] M. A. Alam, M. Si, and P. D. Ye, "A critical review of recent progress on negative capacitance field-effect transistors," *Appl. Phys. Lett.*, vol. 114, no. 9, Mar. 2019, Art. no. 090401.

[6] S. Manipatruni et al., "Scalable energy-efficient magnetoelectric spin–orbit logic," *Nature*, vol. 565, no. 7737, pp. 35–42, Dec. 2018.

[7] P. Debashis et al., "Low-voltage and high-speed switching of a magnetoelectric element for energy efficient compute," in *IEDM Tech. Dig.*, Dec. 2022, pp. 34–36.

[8] A. Chen, "Emerging research device roadmap and perspectives," in *Proc. IEEE Int. Conf. IC Design Technol.*, May 2014, pp. 1–4.

[9] G. Finocchio, M. Di Ventra, K. Y. Camsari, K. Everschor-Sitte, P. K. Amiri, and Z. Zeng, "The promise of spintronics for unconventional computing," *J. Magn. Magn. Mater.*, vol. 521, Mar. 2021, Art. no. 167506.

[10] B. Behin-Aein, V. Diep, and S. Datta, "A building block for hardware belief networks," *Sci. Rep.*, vol. 6, no. 1, pp. 1–10, Jul. 2016.

[11] B. Sutton, K. Y. Camsari, B. Behin-Aein, and S. Datta, "Intrinsic optimization using stochastic nanomagnets," *Sci. Rep.*, vol. 7, p. 44370, Mar. 2017.

[12] K. Y. Camsari, R. Faria, B. M. Sutton, and S. Datta, "Stochastic p-bits for invertible logic," *Phys. Rev. X*, vol. 7, no. 3, 2017, Art. no. 031014.

[13] R. P. Feynman, "Simulating physics with computers," *Int. J. Theor. Phys.*, vol. 21, nos. 6–7, pp. 467–488, 1982.

[14] N. A. Aadit et al., "Massively parallel probabilistic computing with sparse Ising machines," *Nature Electron.*, pp. 1–9, 2022.

[15] J. Kaiser, R. Jaiswal, B. Behin-Aein, and S. Datta, "Benchmarking a probabilistic coprocessor," 2021, *arXiv:2109.14801*.

[16] S. Misra et al., "Probabilistic neural computing with stochastic devices," *Adv. Mater.*, Nov. 2022, Art. no. 2204569.

[17] P. J. Coles, "Thermodynamic AI and the fluctuation frontier," 2023, *arXiv:2302.06584*.

[18] W. A. Borders, A. Z. Pervaiz, S. Fukami, K. Y. Camsari, H. Ohno, and S. Datta, "Integer factorization using stochastic magnetic tunnel junctions," *Nature*, vol. 573, no. 7774, pp. 390–393, Sep. 2019.

[19] R. Faria, J. Kaiser, K. Y. Camsari, and S. Datta, "Hardware design for autonomous Bayesian networks," *Frontiers Comput. Neurosci.*, vol. 15, p. 14, Mar. 2021.

[20] E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*. Hoboken, NJ, USA: Wiley, 1989.

[21] A. Houshang, M. Zahedinejad, S. Muralidhar, J. Checinski, A. A. Awad, and J. Åkerman, "A spin Hall Ising machine," 2020, *arXiv:2006.02236*.

[22] Y. Su, J. Mu, H. Kim, and B. Kim, "A scalable CMOS Ising computer featuring sparse and reconfigurable spin interconnects for solving combinatorial optimization problems," *IEEE J. Solid-State Circuits*, vol. 57, no. 3, pp. 858–868, Mar. 2022.

[23] S. Bhanja, D. K. Karunaratne, R. Panchumarthy, S. Rajaram, and S. Sarkar, "Non-Boolean computing with nanomagnets for computer vision applications," *Nature Nanotechnol.*, vol. 11, no. 2, pp. 177–183, Feb. 2016.

[24] P. Debashis, R. Faria, K. Y. Camsari, J. Appenzeller, S. Datta, and Z. Chen, "Experimental demonstration of nanomagnet networks as hardware for Ising computing," in *IEDM Tech. Dig.*, Dec. 2016, pp. 3–34.

[25] P. L. McMahon et al., "A fully programmable 100-spin coherent Ising machine with all-to-all connections," *Science*, vol. 354, pp. 614–617, Nov. 2016.

[26] S. Dutta et al., "An Ising Hamiltonian solver based on coupled stochastic phase-transition nano-oscillators," *Nature Electron.*, vol. 4, no. 7, pp. 502–512, Jul. 2021.

[27] J. Chou, S. Bramhavar, S. Ghosh, and W. Herzog, "Analog coupled oscillator based weighted Ising machine," *Sci. Rep.*, vol. 9, no. 1, pp. 1–10, Oct. 2019.

[28] M. Yamaoka et al., "A 20 k-spin Ising chip to solve combinatorial optimization problems with CMOS annealing," *IEEE J. Solid-State Circuits*, vol. 51, no. 1, pp. 303–309, Dec. 2015.

[29] T. Wang and J. Roychowdhury, "OIM: Oscillator-based Ising machines for solving combinatorial optimisation problems," in *Proc. Int. Conf. Unconventional Comput. Natural Comput.* Cham, Switzerland: Springer, 2019, pp. 232–256.

[30] Y. Shim, A. Jaiswal, and K. Roy, "Ising computation based combinatorial optimization using spin-Hall effect (SHE) induced stochastic magnetization reversal," *J. Appl. Phys.*, vol. 121, no. 19, 2017, Art. no. 193902.

[31] T. Inagaki et al., "A coherent Ising machine for 2000-node optimization problems," *Science*, vol. 354, no. 6312, pp. 603–606, Nov. 2016.

[32] M. Baity-Jesi et al., "Janus II: A new generation application-driven computer for spin-system simulations," *Comput. Phys. Commun.*, vol. 185, no. 2, pp. 550–559, Feb. 2014.

[33] M. Yamaoka, C. Yoshimura, M. Hayashi, T. Okuyama, H. Aoki, and H. Mizuno, "24.3 20 k-spin Ising chip for combinational optimization problem with CMOS annealing," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2015, pp. 1–3.

[34] N. G. Berloff et al., "Realizing the classical XY Hamiltonian in polariton simulators," *Nature Mater.*, vol. 16, no. 11, pp. 1120–1126, Nov. 2017.

[35] T. Takemoto, M. Hayashi, C. Yoshimura, and M. Yamaoka, "A 2×30 k-spin multichip scalable annealing processor based on a processing-in-memory approach for solving large-scale combinatorial optimization problems," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2019, pp. 52–54.

[36] H. Goto, K. Tatsumura, and A. R. Dixon, "Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems," *Sci. Adv.*, vol. 5, no. 4, Apr. 2019.

[37] M. Aramon, G. Rosenberg, E. Valiante, T. Miyazawa, H. Tamura, and H. Katzgrabeer, "Physics-inspired optimization for quadratic unconstrained problems using a digital annealer," *Frontiers Phys.*, vol. 7, p. 48, Apr. 2019.

[38] A. Mallick, M. K. Bashar, D. S. Truesdell, B. H. Calhoun, S. Joshi, and N. Shukla, "Using synchronized oscillators to compute the maximum independent set," *Nature Commun.*, vol. 11, no. 1, pp. 1–7, Sep. 2020.

[39] K. Yamamoto et al., "7.3 STATICA: A 512-spin 0.25 M-weight full-digital annealing processor with a near-memory all-spin-updates-at-once architecture for combinatorial optimization with complete spin-spin interactions," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 138–140.

[40] S. Patel, P. Canoza, and S. Salahuddin, "Logically synthesized and hardware-accelerated restricted Boltzmann machines for combinatorial optimization and integer factorization," *Nature Electron.*, vol. 5, no. 2, pp. 92–101, Feb. 2022.

[41] R. Afoakwa, Y. Zhang, U. K. R. Vengalam, Z. Ignjatovic, and M. Huang, "A CMOS-compatible Ising machine with bistable nodes," 2020, *arXiv:2007.06665*.

[42] A. Lu et al., "Scalable in-memory clustered annealer with temporal noise of FinFET for the travelling salesman problem," in *IEDM Tech. Dig.*, Dec. 2022, pp. 5–22.

[43] W. Moy, I. Ahmed, P.-W. Chiu, J. Moy, S. S. Sapatnekar, and C. H. Kim, "A 1,968-node coupled ring oscillator circuit for combinatorial optimization problem solving," *Nature Electron.*, vol. 5, no. 5, pp. 310–317, May 2022.

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

[45] M. Marsman, G. Maris, T. Bechger, and C. Glas, "Bayesian inference for low-rank Ising networks," *Sci. Rep.*, vol. 5, no. 1, Mar. 2015.

[46] M. T. McCray, M. A. Abeed, and S. Bandyopadhyay, "Electrically programmable probabilistic bit anti-correlator on a nanomagnetic platform," *Sci. Rep.*, vol. 10, no. 1, pp. 1–11, Jul. 2020.

[47] P. Debashis, V. Ostwal, R. Faria, S. Datta, J. Appenzeller, and Z. Chen, "Hardware implementation of Bayesian network building blocks with stochastic spintronic devices," *Sci. Rep.*, vol. 10, no. 1, p. 16002, Sep. 2020.

[48] S. Nasrin, J. Drobitch, P. Shukla, T. Tulabandhula, S. Bandyopadhyay, and A. R. Trivedi, "Bayesian reasoning machine on a magneto-tunneling junction network," *Nanotechnology*, vol. 31, no. 48, Nov. 2020, Art. no. 484001.

[49] K.-E. Harabi et al., "A memristor-based Bayesian machine," *Nature Electron.*, vol. 6, pp. 1–12, Dec. 2022.

[50] M. Golam Morshed, S. Ganguly, and A. W. Ghosh, "A deep dive into the computational fidelity of high variability low energy barrier magnet technology for accelerating optimization and Bayesian problems," 2023, *arXiv:2302.08074*.

[51] C. Bybee, D. Kleyko, D. E. Nikonov, A. Khosrowshahi, B. A. Olshausen, and F. T. Sommer, "Efficient optimization with higher-order Ising machines," 2022, *arXiv:2212.03426*.

[52] M. K. Bashar and N. Shukla, "Constructing dynamical systems to model higher order Ising spin interactions and their application in solving combinatorial optimization problems," 2022, *arXiv:2211.05365*.

[53] N. Onizawa and T. Hanyu, "High convergence rates of CMOS invertible logic circuits based on many-body Hamiltonians," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2021, pp. 1–5.

[54] B. Sutton, R. Faria, L. A. Ghantasala, R. Jaiswal, K. Y. Camsari, and S. Datta, "Autonomous probabilistic coprocessing with petaflips per second," *IEEE Access*, vol. 8, pp. 157238–157252, 2020.

[55] J. Kaiser and S. Datta, "Probabilistic computing with p-bits," *Appl. Phys. Lett.*, vol. 119, no. 15, Oct. 2021, Art. no. 150503.

[56] S. Aggarwal et al., "Demonstration of a reliable 1 Gb standalone spin-transfer torque MRAM for industrial applications," in *IEDM Tech. Dig.*, Dec. 2019, pp. 1–2.

[57] K. Lee et al., "1 Gbit high density embedded STT-MRAM in 28 nm FDSOI technology," in *IEDM Tech. Dig.*, Dec. 2019, p. 2.

[58] J. L. Drobitch and S. Bandyopadhyay, "Reliability and scalability of p-bits implemented with low energy barrier nanomagnets," *IEEE Magn. Lett.*, vol. 10, pp. 1–4, 2019.

[59] R. Rahman and S. Bandyopadhyay, "The strong sensitivity of the characteristics of binary stochastic neurons employing low barrier nanomagnets to small geometrical variations," 2021, *arXiv:2108.04319*.

[60] O. Hassan, S. Datta, and K. Y. Camsari, "Quantitative evaluation of hardware binary stochastic neurons," *Phys. Rev. Appl.*, vol. 15, no. 6, Jun. 2021, Art. no. 064046.

[61] K. Y. Camsari, M. M. Torunbalci, W. A. Borders, H. Ohno, and S. Fukami, "Double-free-layer magnetic tunnel junctions for probabilistic bits," *Phys. Rev. Appl.*, vol. 15, no. 4, Apr. 2021, Art. no. 044049.

[62] R. Rahman and S. Bandyopadhyay, "Robustness of binary stochastic neurons implemented with low barrier nanomagnets made of dilute magnetic semiconductors," *IEEE Magn. Lett.*, vol. 13, pp. 1–4, 2022.

[63] Y. Lv, R. P. Bloom, and J.-P. Wang, "Experimental demonstration of probabilistic spin logic by magnetic tunnel junctions," *IEEE Magn. Lett.*, vol. 10, pp. 1–5, 2019.

[64] K. Y. Camsari et al., "From charge to spin and spin to charge: Stochastic magnets for probabilistic switching," *Proc. IEEE*, vol. 108, no. 8, pp. 1322–1337, Aug. 2020.

[65] X. Chen, J. Zhang, and J. Xiao, "Magnetic-tunnel-junction-based true random-number generator with enhanced generation rate," *Phys. Rev. Appl.*, vol. 18, no. 2, Aug. 2022, L021002.

[66] L. Rehm et al., "Stochastic magnetic actuated random transducer devices based on perpendicular magnetic tunnel junctions," 2022, *arXiv:2209.01480*.

[67] B. R. Zink, Y. Lv, and J.-P. Wang, "Review of magnetic tunnel junctions for stochastic computing," *IEEE J. Exp. Solid-State Comput. Devices Circuits*, vol. 8, no. 2, pp. 173–184, Dec. 2022.

[68] A. Fukushima et al., "Spin dice: A scalable truly random number generator based on spintronics," *Appl. Phys. Exp.*, vol. 7, no. 8, 2014, Art. no. 083001.

[69] Y. Shao et al., "Implementation of artificial neural networks using magnetoresistive random-access memory-based stochastic computing units," *IEEE Magn. Lett.*, vol. 12, pp. 1–5, 2021.

[70] K. Y. Camsari, S. Salahuddin, and S. Datta, "Implementing p-bits with embedded MTJ," *IEEE Electron Device Lett.*, vol. 38, no. 12, pp. 1767–1770, Dec. 2017.

[71] J. Kaiser, A. Rustagi, K. Y. Camsari, J. Z. Sun, S. Datta, and P. Upadhyaya, "Subnanosecond fluctuations in low-barrier nanomagnets," *Phys. Rev. Appl.*, vol. 12, no. 5, Nov. 2019, Art. no. 054056.

[72] O. Hassan, R. Faria, K. Y. Camsari, J. Z. Sun, and S. Datta, "Low-barrier magnet design for efficient hardware binary stochastic neurons," *IEEE Magn. Lett.*, vol. 10, pp. 1–5, 2019.

[73] K. Hayakawa et al., "Nanosecond random telegraph noise in in-plane magnetic tunnel junctions," *Phys. Rev. Lett.*, vol. 126, no. 11, Mar. 2021, Art. no. 117202.

[74] C. Safranski, J. Kaiser, P. Trouilloud, P. Hashemi, G. Hu, and J. Z. Sun, "Demonstration of nanosecond operation in stochastic magnetic tunnel junctions," *Nano Lett.*, vol. 21, no. 5, pp. 2040–2045, Feb. 2021.

[75] S. Kanai, K. Hayakawa, H. Ohno, and S. Fukami, "Theory of relaxation time of stochastic nanomagnets," *Phys. Rev. B, Condens. Matter*, vol. 103, no. 9, Mar. 2021, Art. no. 094423, doi: 10.1103/PhysRevB.103.094423.

[76] J. Kaiser, W. A. Borders, K. Y. Camsari, S. Fukami, H. Ohno, and S. Datta, "Hardware-aware in situ learning based on stochastic magnetic tunnel junctions," *Phys. Rev. Appl.*, vol. 17, no. 1, Jan. 2022, Art. no. 014016.

[77] A. Grimaldi et al., "Experimental evaluation of simulated quantum annealing with MTJ-augmented p-bits," in *IEDM Tech. Dig.*, Dec. 2022, pp. 22–24.

[78] J. Yin et al., "Scalable Ising computer based on ultra-fast field-free spin orbit torque stochastic device with extreme 1-bit quantization," in *IEDM Tech. Dig.*, Dec. 2022, pp. 31–36.

[79] G. M. Gutiérrez-Finol, S. Giménez-Santamarina, Z. Hu, L. E. Rosaleny, S. Cardona-Serra, and A. Gaita-Ariño, "Lanthanide molecular nanomagnets as probabilistic bits," 2023, *arXiv:2301.08182*.

[80] K. S. Woo, J. Kim, J. Han, W. Kim, Y. H. Jang, and C. S. Hwang, "Probabilistic computing using $Cu_{0.1}Te_{0.9}/HfO_2/Pt$ diffusive memristors," *Nature Commun.*, vol. 13, no. 1, p. 5762, Sep. 2022.

[81] Y. Liu et al., "Probabilistic circuit implementation based on p-bits using the intrinsic random property of RRAM and p-bit multiplexing strategy," *Micromachines*, vol. 13, no. 6, p. 924, Jun. 2022.

[82] T. J. Park et al., "Efficient probabilistic computing with stochastic perovskite nickelates," *Nano Lett.*, vol. 22, no. 21, pp. 8654–8661, Nov. 2022.

[83] S. Cheemalavagu, P. Korkmaz, K. V. Palem, B. E. Akgul, and L. N. Chakrapani, "A probabilistic CMOS switch and its realization by exploiting noise," in *Proc. IFIP Int. Conf. VLSI*, 2005, pp. 535–541.

[84] W. Whitehead, Z. Nelson, K. Y. Camsari, and L. Theogarajan, "CMOS-compatible Ising and Potts annealing using single photon avalanche diodes," 2022, *arXiv:2211.12607*.

[85] L. Xia et al., "Technological exploration of RRAM crossbar array for matrix-vector multiplication," *J. Comput. Sci. Technol.*, vol. 31, no. 1, pp. 3–19, Jan. 2016.

[86] Y. Li et al., "Capacitor-based cross-point array for analog neural network with record symmetry and linearity," in *Proc. IEEE Symp. VLSI Technol.*, Jun. 2018, pp. 25–26.

[87] O. Hassan, K. Y. Camsari, and S. Datta, "Voltage-driven building block for hardware belief networks," *IEEE Design Test*, vol. 36, no. 3, pp. 15–21, Jun. 2019.

[88] M. Kang, S. K. Gonugondla, A. Patil, and N. R. Shanbhag, "A multi-functional in-memory inference processor using a standard 6T SRAM array," *IEEE J. Solid-State Circuits*, vol. 53, no. 2, pp. 642–655, Feb. 2018.

[89] N. Verma et al., "In-memory computing: Advances and prospects," *IEEE Solid State Circuits Mag.*, vol. 11, no. 3, pp. 43–55, Aug. 2019.

[90] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-6, no. 6, pp. 721–741, Nov. 1984.

[91] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. Cambridge, MA, USA: MIT Press, 2009.

[92] G. E. Hinton, *A Practical Guide to Training Restricted Boltzmann Machines*. Berlin, Germany: Springer, 2012, pp. 599–619.

[93] M. R. Garey and D. S. Johnson, *Computers and Intractability*, vol. 174, San Francisco, CA, USA: Freeman, 1979.

[94] D. Brélaz, "New methods to color the vertices of a graph," *Commun. ACM*, vol. 22, no. 4, pp. 251–256, 1979.

[95] Y. Fang et al., "Parallel tempering simulation of the three-dimensional Edwards–Anderson model with compact asynchronous multispin coding on GPU," *Comput. Phys. Commun.*, vol. 185, no. 10, pp. 2467–2478, Oct. 2014.

[96] K. Yang, Y.-F. Chen, G. Roumpos, C. Colby, and J. Anderson, "High performance Monte Carlo simulation of Ising model on TPU clusters," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, Nov. 2019, pp. 1–15.

[97] P. Debashis, R. Faria, K. Y. Camsari, S. Datta, and Z. Chen, "Correlated fluctuations in spin orbit torque coupled perpendicular nanomagnets," *Phys. Rev. B, Condens. Matter*, vol. 101, no. 9, Mar. 2020, Art. no. 094405.

[98] N. A. Aadit, A. Grimaldi, G. Finocchio, and K. Y. Camsari, "Physics-inspired Ising computing with ring oscillator activated p-bits," in *Proc. IEEE 22nd Int. Conf. Nanotechnol. (NANO)*, Jul. 2022, pp. 393–396.

[99] S. Bhatti, R. Sbiaa, A. Hirohata, H. Ohno, S. Fukami, and S. Piramanayagam, "Spintronics based random access memory: A review," *Mater. Today*, vol. 20, no. 9, pp. 530–548, 2017.

[100] H. Hoos and T. Stützle, *SATLIB: An Online Resource for Research on SAT*. Amsterdam, The Netherlands: IOS Press, Apr. 2000, pp. 283–292.

[101] A. Z. Pervaiz, L. A. Ghantasala, K. Y. Camsari, and S. Datta, "Hardware emulation of stochastic p-bits for invertible logic," *Sci. Rep.*, vol. 7, no. 1, p. 10994, Sep. 2017.

[102] V. Choi, "Minor-embedding in adiabatic quantum computation: II. Minor-universal graph design," *Quantum Inf. Process.*, vol. 10, no. 3, pp. 343–353, 2011.

[103] N. Sagan and J. Roychowdhury, "DaS: Implementing dense Ising machines using sparse resistive networks," in *Proc. 41st IEEE/ACM Int. Conf. Comput.-Aided Design*, Oct. 2022, pp. 1–9.

[104] A. Lucas, "Ising formulations of many NP problems," *Frontiers Phys.*, vol. 2, p. 5, Feb. 2014.

[105] E. Andriyash et al., "Boosting integer factoring performance via quantum annealing offsets," D-Wave Syst., Burnaby, BC, Canada, D-Wave Tech. Rep. 14-1002A-B, 2016.

[106] J. D. Biamonte, "Nonperturbative k-body to two-body commuting conversion Hamiltonians and embedding problem instances into Ising spins," *Phys. Rev. A, Gen. Phys.*, vol. 77, no. 5, May 2008, Art. no. 052331.

[107] F. L. Traversa and M. D. Ventra, "Polynomial-time solution of prime factorization and NP-complete problems with digital memcomputing machines," *Chaos, Interdiscipl. J. Nonlinear Sci.*, vol. 27, no. 2, 2017, Art. no. 023107.

[108] M. Di Ventra, *MemComputing: Fundamentals and Applications*. Oxford, U.K.: Oxford Univ. Press, 2022.

[109] R. Salakhutdinov and G. Hinton, "Deep Boltzmann machines," in *Proc. 12th Int. Conf. Artif. Intell. Statist.*, in Proceedings of Machine Learning Research, vol. 5, D. Van Dyk and M. Welling, Eds. Clearwater Beach, FL, USA: Hilton Clearwater Beach Resort, Apr. 2009, pp. 448–455. [Online]. Available: https://proceedings.mlr.press/v5/salakhutdinov09a.html

[110] S. H. Adachi and M. P. Henderson, "Application of quantum annealing to training of deep neural networks," 2015, *arXiv:1510.06356*.

[111] H. Manukian, F. L. Traversa, and M. Di Ventra, "Accelerating deep learning with memcomputing," *Neural Netw.*, vol. 110, pp. 1–7, Feb. 2019.

[112] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 2256–2265.

[113] H. Ma, M. Govoni, and G. Galli, "Quantum simulations of materials on near-term quantum computers," *npj Comput. Mater.*, vol. 6, no. 1, p. 85, Jul. 2020.

[114] R. Babbush, J. R. McClean, M. Newman, C. Gidney, S. Boixo, and H. Neven, "Focus beyond quadratic speedups for error-corrected quantum advantage," *PRX Quantum*, vol. 2, no. 1, Mar. 2021, Art. no. 010103.

[115] K. Y. Camsari, S. Chowdhury, and S. Datta, "Scalable emulation of sign-problem–free Hamiltonians with room-temperature p-bits," *Phys. Rev. Appl.*, vol. 12, no. 3, Sep. 2019, Art. no. 034061.

[116] M. Suzuki, "Relationship between d-dimensional quantal spin systems and $(d+1)$-dimensional Ising systems: Equivalence, critical exponents and systematic approximants of the partition function and spin correlations," *Prog. Theor. Phys.*, vol. 56, no. 5, pp. 1454–1469, Nov. 1976.

[117] A. D. King et al., "Scaling advantage over path-integral Monte Carlo in quantum simulation of geometrically frustrated magnets," *Nature Commun.*, vol. 12, no. 1, pp. 1–6, 2021.

[118] S. Chowdhury, K. Y. Camsari, and S. Datta, "Accelerated quantum Monte Carlo with probabilistic computers," 2022, *arXiv:2210.17526*.

[119] S. Chowdhury, S. Datta, and K. Y. Camsari, "A probabilistic approach to quantum inspired algorithms," in *IEDM Tech. Dig.*, Dec. 2019, pp. 5–37.

[120] S. Chowdhury, K. Y. Camsari, and S. Datta, "Emulating quantum interference with generalized Ising machines," 2020, *arXiv:2007.07379*.

[121] M. Troyer and U.-J. Wiese, "Computational complexity and fundamental limitations to fermionic quantum Monte Carlo simulations," *Phys. Rev. Lett.*, vol. 94, no. 17, May 2005, Art. no. 170201.

[122] D. Aharonov, X. Gao, Z. Landau, Y. Liu, and U. Vazirani, "A polynomial-time classical algorithm for noisy random circuit sampling," 2022, *arXiv:2211.03999*.

[123] D. Hangleiter, I. Roth, D. Nagaj, and J. Eisert, "Easing the Monte Carlo sign problem," *Sci. Adv.*, vol. 6, no. 33, Aug. 2020, Art. no. eabb8341.

[124] G. Carleo and M. Troyer, "Solving the quantum many-body problem with artificial neural networks," *Science*, vol. 355, pp. 602–606, Feb. 2017.

[125] Z. Cai and J. Liu, "Approximating quantum many-body wave functions using artificial neural networks," *Phys. Rev. B, Condens. Matter*, vol. 97, no. 3, Jan. 2018, Art. no. 035116, doi: 10.1103/PhysRevB.97.035116.

[126] H. Saito and M. Kato, "Machine learning technique to find quantum many-body ground states of bosons on a lattice," *J. Phys. Soc. Jpn.*, vol. 87, no. 1, Jan. 2018, Art. no. 014001, doi: 10.7566/JPSJ.87.014001.

[127] S. Das Sarma, D.-L. Deng, and L.-M. Duan, "Machine learning meets quantum physics," 2019, *arXiv:1903.03516*.

[128] N. A. Aadit, A. Grimaldi, M. Carpentieri, L. Theogarajan, G. Finocchio, and K. Y. Camsari, "Computing with invertible logic: Combinatorial optimization with probabilistic bits," in *IEDM Tech. Dig.*, Dec. 2021, pp. 3–40.

[129] A. Grimaldi et al., "Spintronics-compatible approach to solving maximum-satisfiability problems with probabilistic computing, invertible logic, and parallel tempering," *Phys. Rev. Appl.*, vol. 17, no. 2, Feb. 2022, Art. no. 024052.

[130] G. Desjardins, A. Courville, and Y. Bengio, "Adaptive parallel tempering for stochastic maximum likelihood learning of RBMs," 2010, *arXiv:1012.3476*.

[131] M. Mohseni et al., "Nonequilibrium Monte Carlo for unfreezing variables in hard combinatorial optimization," 2021, *arXiv:2111.13628*.

[132] A. Buluc et al., "Randomized algorithms for scientific computing (RASC)," 2021, *arXiv:2104.11079*.

[133] G. Finocchio et al., "Roadmap for unconventional computing with nanotechnology," 2023, *arXiv:2301.06727*.