12.2 p-Circuits: Neither Digital nor Analog

Ming-Che Li[1], Archisman Ghosh[1], Risi Jaiswal[1], Lakshmi Anirudh Ghantasala[1,2], Behtash Behin-Aein[2], Shreyas Sen[1], Supriyo Datta[1]

[1]Purdue University, West Lafayette, IN
[2]Ludwig Computing, Mill Valley, CA

2025 IEEE International Solid-State Circuits Conference (ISSCC) | 979-8-3315-4101-9/25/$31.00 ©2025 IEEE | DOI: 10.1109/ISSCC49661.2025.10904553

Custom integrated circuits aim to solve important problems with ultra-high efficiency, making use of analog and digital circuits, with well-known trade-offs. This work is about a new paradigm which is neither analog nor digital, we call it a p-circuit [See for example, 1-12]. In terms of inputs and outputs, our p-circuits look-like digital circuits (Fig. 12.2.1), thus requiring no ADC's or DAC's. However, the output is not a Boolean function of the inputs. It is a random binary variable whose probability P(B=1) of being *1* is given by an analog function A of the inputs which takes on values continuously between *0* and *1*.

Building p-circuits: Figure 12.2.2 shows how a p-circuit can be built using building blocks each of which takes four inputs, combines them to compute an analog quantity A(b_1, b_2, b_3, b_4) and generates a binary output B with P(B=1) = A as shown in Fig. 12.2.1. This single output is replicated four-ways as shown to facilitate the creation of a two-dimensional array through tiling. The circuit operates by sequentially updating the output of each of the *N* building blocks (or *p-gates*) based on its current inputs. Each of the *N* p-bits is updated ***it*** (denoting iterations) times. What problems can we solve with this two-dimensional array? For starters, we can solve Quadratic Unconstrained Binary Optimization (QUBO) problems described by cost functions E of the form $E = -\Sigma_i b_i h_i - 0.5\ \Sigma_{i,j} (b_i W_{ij} b_j)$ where the indices (i,j) run over the sites of the two-dimensional lattice, and the weight matrix W_{ij} is non-zero only if i and j are nearest neighbors. Optimization requires us to find configurations *{b}* that minimize the cost function *E*.

A statistical approach to this problem is to generate samples from a probability distribution function (PDF) *P ~ exp(-E({b}))* so that low *E* configurations appear with high probability. This can be done if we generate new samples from the existing sample by modifying a single p-bit b_k out of the collection *{b}* such that $P(b_k=1) = [1+\exp(-\varepsilon_k)]$^(-1) where $\varepsilon_k = E(b_k=0) - E(b_k=1) = h_k + \Sigma_j (W_{kj} b_j)$. The algorithm is implemented by repeatedly performing a ***core operation***, consisting of: (1) looking at ***n*** binary inputs (where ***n*** depends on the number of non-zero elements of W_{kj} for a given *k*), and (2) generating a random binary output following the probability given above. Several ASIC implementations [13-25] use similar concepts in commercial process. We will now describe an ASIC implementation using a commercial 65nm process, which solves a class of QUBO problems [1].

ASIC design: Figure 12.2.3 shows the system architecture: The ASIC, comprising a 1,440 p-bit computer, employs a 72 p-gate array as its primary computing units for stochastic computing. Additionally, it incorporates two types of memory: weight memory and p-bit memory. These memories serve the purpose of providing weight values W_{kj} from the cost function E that define the specific QUBO problem. The iterative process involves 1,440 p-bits, which are updated over 20 cycles. During each cycle, 72 p-bits are updated through 72 p-gates. Figure 12.2.3 (bottom-left) shows the implementation of a p-gate, which takes up to 7 p-bits as inputs along with their corresponding weights. Following the equation $\varepsilon_k = h_k + \Sigma_j (W_{kj} b_j)$, the probability is calculated using an exponential LUT and compared with a random number generated by an Xoshiro128+ PRNG, as depicted in Fig. 12.2.3 (bottom-right). After each cycle, an updated p-bit is written back into the p-bit memory. Each iteration updates all p-bits to generate a new sample from the PDF, *P ~ exp(-E({b}))*.

ASIC measurement results: Figure 12.2.4 shows measured results from our IC. Figure 12.2.4 (top) shows the solution to the problem described in [1]. The convergence index reaches a threshold value after some iterations. Figure 12.2.4 (bottom-left) shows the measured power. The p-bit computer consumes 328uW active power at 10MHz at 0.5V core voltage, with leakage power contributing to an additional 57.42uW. Notably, approximately 70% of this power is consumed by the p-gate array.

Energy per operation: Figure 12.2.5 (left) compares the energy cost of the core operation discussed earlier, estimated for each of four options, namely, (1) CPU, (2) 125MHz FPGA, (3) 10MHz ASIC and (4) clockless circuit with s-MTJ's. We evaluated the energy based on solving a QUBO problem like the one described earlier except that the lattice is 3D and non-rectangular [1] for which each p-bit looks at 5 inputs rather than 4. The number of p-bits *N* = 1,440, while the problem required ***it*** = 25,000 iterations. We have also used the same architecture to solve other QUBO problems featuring different values of (*N, **it***) requiring different amounts of *total energy*, but the *energy per operation* is characteristic of the hardware used to implement it. Figure 12.2.5 (right) shows the energy and times for several other implementations reported in the literature, which we discuss below in the section "How we compare."

The costliest implementation (Fig. 12.2.5 left) is on a CPU for which we estimate ~uJ per operation, while our ASIC implementation requires ~pJ per operation, which is six orders of magnitude smaller. Note that these energy estimates should be applicable to any algorithm that can be implemented by repeatedly performing the same core operation requiring a p-gate with 5 binary inputs, allowing us to evaluate the analog function simply using an LUT with 2^5 = *32* entries. But if each p-gate were to look at many more inputs, the core operation may consume more energy. But how versatile is our core operation and the p-gate implementing it? We believe it can be used way beyond the QUBO problem described above as illustrated by the following example from a common generative model.

Future directions: Figure 12.2.6 shows a series of transformations each of which has a form similar to what we discussed, namely $c_k = F(h_k + \Sigma_j (W_{kj} b_j))$ turning a set {*b*} into a set {*c*} . Given a specific non-linear function *F*, training algorithms have been developed that can find appropriate [*W*], {*h*} for each transformation such that a random input image is transformed into a recognizable image. However, we cannot implement the standard algorithms with the p-gates we discussed since, (1) the non-linear function F is usually deterministic while our p-gates are probabilistic, and (2) F operates on a large number of continuous variables while our p-gate operates on a small number of binary variables.

We need to change these training conditions so that the resulting [*W*], {*h*} can be implemented using p-gates for which Fig. 12.2.6 provides a proof-of-concept, hopefully a stepping stone to more complex generative models like diffusion models or even large language models. Compared to the analog (or multi-bit digital) gates commonly used in implementing deep neural networks (DNN's), the advantage of p-gates is that they work with binary inputs. But these binary quantities do not just ***approximate*** *the analog information, they* ***embed*** *it statistically.* One might think that we would have to average many samples to get acceptable results, but the results in Fig. 12.2.6 were obtained ***with just one sample.*** We believe that the true power of p-circuits lies in providing a natural platform for such probabilistic applications and algorithms.

How do we compare? Ising computing is of course not new to this community [13-28], and the prior results in Fig. 12.2.5 (right) show energy costs ranging from pJ to µJ across various designs. Note, however, that time-to-solution (TTS) and energy-to-solution depend on the design choices, annealing method, and the target Ising problem. We also note that much previous literature [17, 19, 22-24] has applied in-memory computing techniques, significantly reducing the energy cost associated with repeatedly loading varying weights compared to classic digital designs, like our ASIC. We propose that the ***energy cost per operation per spin*** for a given number of inputs would be a fair metric for comparing different Ising solvers.

Secondly, if all p-bits along with the weights can fit on a chip then the entire circuit can operate autonomously without clocking and this can reduce the energy cost significantly. Indeed our SPICE simulations of clockless circuits using experimentally benchmarked models for stochastic magnetic tunnel junction's (s-MTJ's) suggest 20µW x 50ps ~ *fJ* per operation [1, 9]. Such s-MTJ's have only been demonstrated in laboratories but similar numbers should be achievable with other standard devices that can be taped out. The difficulty, however, is to scale up this clockless operation to address large problems.

Finally, we note that this paper is not about a particularly energy-efficient implementation of Ising computing, of which there are many already. Indeed we have not yet implemented several standard techniques like compute-in-memory or the ones noted in Fig. 12.2.5 (right). It is well-known that specialization can reduce energy and delay, the challenge is to make it broadly applicable. Our primary message here is that a very broad variety of problems can be addressed through repeated application of the core building block or p-gate shown in Fig. 12.2.1 and hence the need to benchmark its energy and delay **per elementary operation** for different implementations. As we have discussed, p-gates arise naturally in Ising computing but are relatively unknown in the context of feed-forward DNN's which are the staple of modern artificial intelligence (AI) [29,30]. *We hope this paper will encourage the use of p-gates beyond the narrow confines of Ising computing.*

We end by noting an interesting similarity between p-circuits and quantum circuits that involve qubits coherently coupled to other qubits. A system of *N* qubits has an associated wavefunction with 2^N components whose squared magnitude gives the PDF. Quantum circuits multiply the wavefunction by a unitary matrix *U*, while p-circuits multiply the PDF by a stochastic matrix *W*. Both *U and W* are huge matrices of size 2^Nx 2^N, too large for direct computation. But p-circuits can be used to generate samples efficiently that follow the correct PDF generated by *W*. However, they are not very effective if the problems involve a complex unitary matrix *U*, like the quantum Fourier transform (QFT) in Shor's algorithm, which can be sampled efficiently only with quantum computers.

Acknowledgement:
We thank Drs. Shuvro Chowdhury and Kerem Camsari for sharing details of their work in [1] on a quantum Monte Carlo problem whose ASIC implementation is described in Fig. 12.2.3 of this paper. This work was partially supported by ONR-MURI Grant No. N000142312708, OptNet: Optimization with p-Bit Networks.
Competing interests: SD has a financial interest in Ludwig Computing.

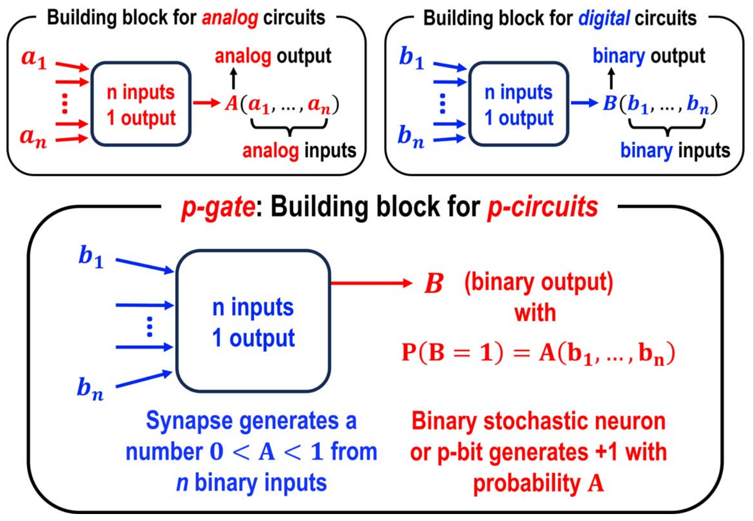

Building block for analog circuits
a_1
a_n
n inputs
1 output
analog output
$A(a_1, \ldots, a_n)$
analog inputs
Building block for digital circuits
b_1
b_n
n inputs
1 output
binary output
$B(b_1, \ldots, b_n)$
binary inputs
p-gate: Building block for p-circuits
b_1
b_n
n inputs
1 output
B (binary output) with
$P(B = 1) = A(b_1, \ldots, b_n)$
Synapse generates a number $0 < A < 1$ from n binary inputs
Binary stochastic neuron or p-bit generates +1 with probability A

Figure 12.2.1: Building blocks for analog (top left), digital (top right), and p-circuits (bottom). P(B=1) denotes probability of B being 1 rather than 0.

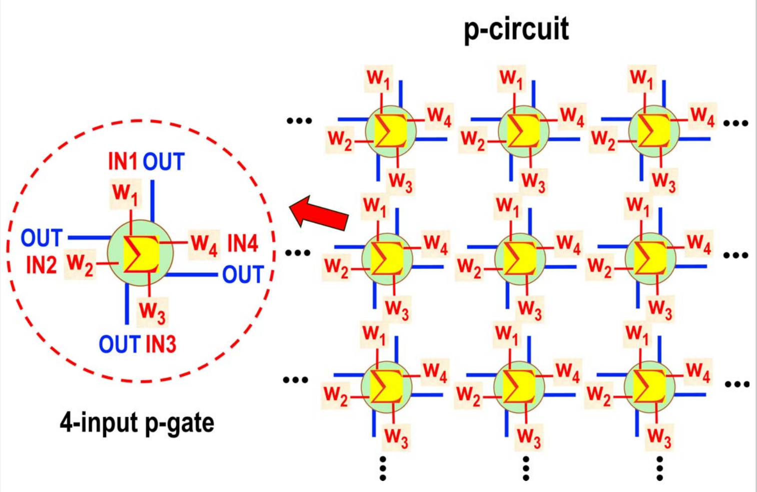


Figure 12.2.2: Building a circuit by tiling together 4-input p-gates. If each p-gate has more inputs then a more elaborate layout is required.

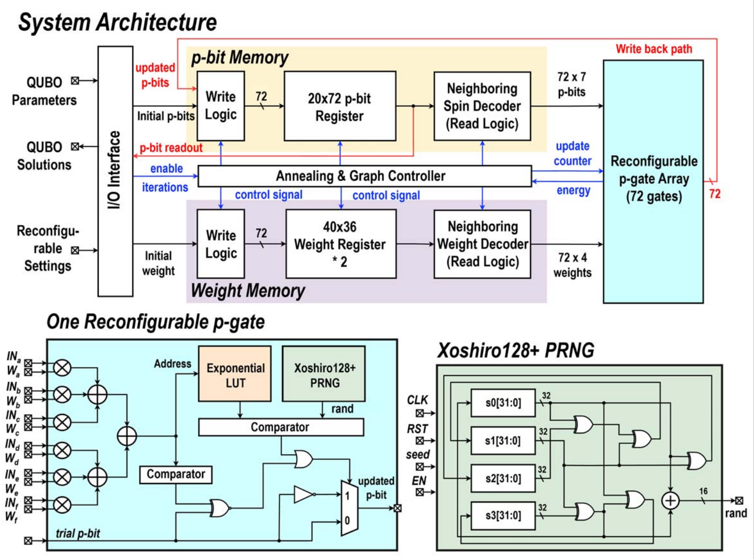


Figure 12.2.3: System level architecture of the prototype p-bit ASIC. An example p-circuits architecture (bottom left). Xoshiro128+ PRNG (bottom right).

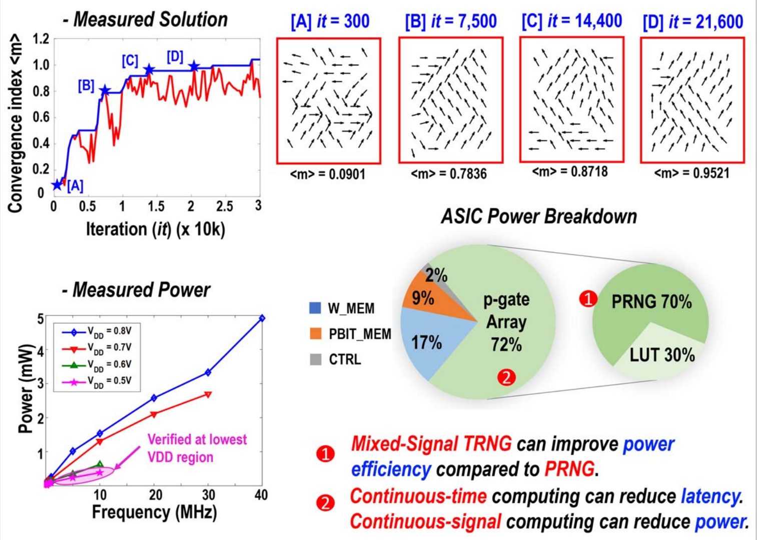


Figure 12.2.4: Measurement results of an example p-bit ASIC fabricated with commercial process. Possibility of improvements.

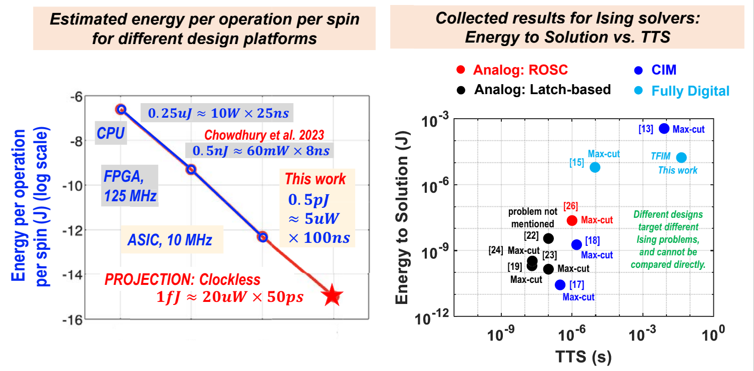


Figure 12.2.5: Energy per operation per spin of the core operation, estimated for each of four options as discussed in text (left). Energy to solution and TTS plot for recent state-of-the-art Ising solvers (right).

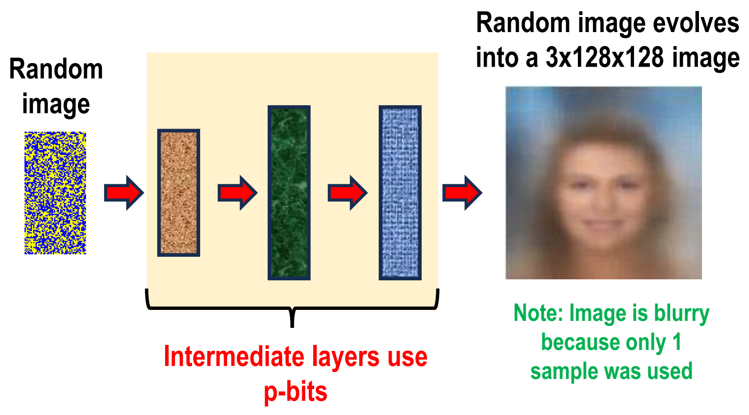
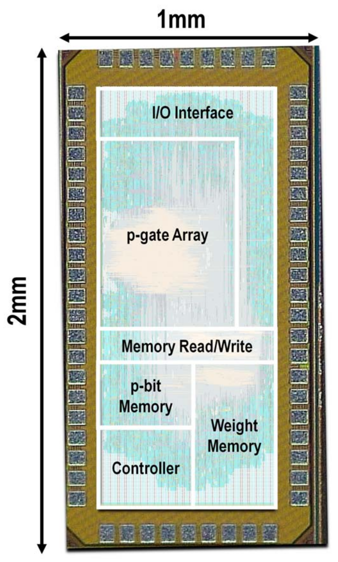

Random image
Random image evolves into a 3x128x128 image
Intermediate layers use p-bits
Note: Image is blurry because only 1 sample was used

Figure 12.2.6: A sequence of p-bit layers trained to turn a random initial image into a recognizable image.



Technology	65nm LP CMOS
Packaging	Chip-on-Board
Design Highlights	1. First ASIC for transverse field Ising model 2. Reconfigurable architecture 3. 10^3~10^6 energy improvement over previous CPU and FPGA work
# p-bits	1,440 (40x36)
# p-gate	72
p-gate Size	2,990um^2
Power/p-bit	5uW @ 0.5V, 10MHz
Core Size	0.95mm^2
Supply	Core: 0.5-0.8V Analog IO: 0.8-1V Digital IO: 2.5V

Figure 12.2.7: IC Micrograph and specification.

References:
[1] S. Chowdhury et al., "Accelerated quantum Monte Carlo with probabilistic computers." *Communications Physics* 6.1 (2023): 85.
[2] Bunaiyan, Saleh, Supriyo Datta, and Kerem Y. Camsari. "Heisenberg machines with programmable spin circuits." *Physical Review Applied* 22.1 (2024): 014014.
[3] S. Niazi et al., "Training deep Boltzmann networks with sparse Ising machines." *Nature Electronics* (2024): 1-10.
[4] N. S. Singh et al., "CMOS plus stochastic nanomagnets enabling heterogeneous computers for probabilistic inference and learning." *Nature Communications* 15.1 (2024): 2685.
[5] W. Whitehead et al., "CMOS-compatible Ising and Potts annealing using single-photon avalanche diodes." *Nature Electronics* 6.12 (2023): 1009-1019.
[6] J. Kaiser et al., "Hardware-aware in situ learning based on stochastic magnetic tunnel junctions." *Physical Review Applied* 17.1 (2022): 014016.
[7] Hassan, Orchi, Supriyo Datta, and Kerem Y. Camsari. "Quantitative evaluation of hardware binary stochastic neurons." *Physical Review Applied* 15.6 (2021): 064046.
[8] R. Faria et al., "Hardware design for autonomous bayesian networks." *Frontiers in computational neuroscience* 15 (2021): 584797.
[9] R.F. Sutton et al., "Autonomous Probabilistic Coprocessing With Petaflips per Second," *IEEE Access*, vol. 8, pp. 157238-157252, 2020.
[10] W.A. Borders et al., "Integer factorization using stochastic magnetic tunnel junctions." *Nature* 573.7774 (2019): 390-393.
[11] K.Y. Camsari et al., "Stochastic p-bits for invertible logic." *Physical Review X* 7.3 (2017): 031014.
[12] Behin-Aein, Behtash, Vinh Diep, and Supriyo Datta. "A building block for hardware belief networks." *Scientific reports* 6.1 (2016): 29893.
[13] M. Yamaoka et al., "20k-spin Ising chip for combinational optimization problem with CMOS annealing," *2015 IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, pages 1-3, 2015.
[14] T. Takemoto et al., "A 2 ×30k-spin multichip scalable annealing processor based on a processing-in-memory approach for solving large-scale combinatorial optimization problems," *2019 IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, pages 52-54, 2019.
[15] K. Yamamoto et al., "Statica: A 512-spin 0.25m-weight full-digital annealing processor with a near-memory all-spin-updates-at-once architecture for combinatorial optimization with complete spin-spin interactions," *2020 IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, pages 138-140, 2020.
[16] T. Takemoto et al., "A 144kb annealing system composed of 9×16kb annealing processor chips with scalable chip-to-chip connections for large-scale combinatorial optimization problems," *2021 IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, volume 64, pages 64-66, 2021.
[17] Y. Su, H. Kim and B. Kim, "CIM-Spin: A 0.5-to-1.2V Scalable Annealing Processor Using Digital Compute-In-Memory Spin Operators and Register-Based Spins for Combinatorial Optimization Problems," *2020 IEEE International Solid-State Circuits Conference - (ISSCC)*, San Francisco, CA, USA, 2020, pp. 480-482.
[18] Y. Su, T. T. -H. Kim and B. Kim, "FlexSpin: A Scalable CMOS Ising Machine with 256 Flexible Spin Processing Elements for Solving Complex Combinatorial Optimization Problems," *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, USA, 2022, pp. 1-3.
[19] J. Bae, W. Oh, J. Koo and B. Kim, "CTLE-Ising: A 1440-Spin Continuous-Time Latch-Based isling Machine with One-Shot Fully-Parallel Spin Updates Featuring Equalization of Spin States," *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, USA, 2023, pp. 142-144.
[20] S. Xie et al., "Snap-SAT: A One-Shot Energy-Performance-Aware All-Digital Compute-in-Memory Solver for Large-Scale Hard Boolean Satisfiability Problems," *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, USA, 2023, pp. 420-422.
[21] K. Kawamura et al., "Amorphica: 4-Replica 512 Fully Connected Spin 336MHz Metamorphic Annealer with Programmable Optimization Strategy and Compressed-Spin-Transfer Multi-Chip Extension," *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, USA, 2023, pp. 42-44.
[22] J. Bae, J. Koo, C. Shim and B. Kim, "LISA: A 576×4 All-in-One Replica-Spins Continuous-Time Latch-Based Ising Computer Using Massively-Parallel Random-Number Generations and Replica Equalizations," *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, USA, 2024, pp. 284-286.
[23] J. Bae, C. Shim and B. Kim, "e-Chimera: A Scalable SRAM-Based Ising Macro with Enhanced-Chimera Topology for Solving Combinatorial Optimization Problems Within Memory," *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, USA, 2024, pp. 286-288.
[24] J. Song et al., "A Variation-Tolerant In-eDRAM Continuous-Time Ising Machine Featuring 15-Level Coefficients and Leaked Negative-Feedback Annealing," *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, USA, 2024, pp. 490-492.
[25] Y. -C. Chu et al., "A Fully Integrated Annealing Processor for Large-Scale Autonomous Navigation Optimization," *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, USA, 2024, pp. 488-490.
[26] I. Ahmed, P. -W. Chiu and C. H. Kim, "A Probabilistic Self-Annealing Compute Fabric Based on 560 Hexagonally Coupled Ring Oscillators for Solving Combinatorial Optimization Problems," *2020 IEEE Symposium on VLSI Circuits*, Honolulu, HI, USA, 2020, pp. 1-2.
[27] S. Xie et al., "Ising-CIM: A Reconfigurable and Scalable Compute Within Memory Analog Ising Accelerator for Solving Combinatorial Optimization Problems," in *IEEE Journal of Solid-State Circuits*, vol. 57, no. 11, pp. 3453-3465, Nov. 2022.
[28] Y. Zhou et al., "A Compute-in-Memory Annealing Processor with Interaction Coefficient Reuse and Sparse Energy Computation for Solving Combinatorial Optimization Problems," *IEEE Journal of Solid-State Circuits*, vol. 59, no. 9, pp. 3094-3105, Sept. 2024.
[29] Shekhovtsov, Alexander, and Viktor Yanush. "Reintroducing straight-through estimators as principled methods for stochastic binary networks." DAGM German Conference on Pattern Recognition. Cham: Springer International Publishing, 2021.
[30] Li, Yang, et al. "Binary-Stochasticity-Enabled Highly Efficient Neuromorphic Deep Learning Achieves Better-than-Software Accuracy." Advanced Intelligent Systems 6.1 (2024): 2300399.