An efficient synchronous-updating memristor-based Ising solver for combinatorial optimization

Mingrui Jiang[1], Keyi Shan[1], Xia Sheng[2], Cat Graves[2], John Paul Strachan[3], Can Li[1]
[1]Department of EEE, The University of Hong Kong, Hong Kong SAR, China; [2]Hewlett Packard Labs, Milpitas, CA, United States; [3]Peter Grünberg Institut (PGI-14) and RWTH Aachen University, Germany; Email: canl@hku.hk

***Abstract*—**Despite showing significant potential in solving combinatorial optimization problems, existing memristor-based solvers update node states asynchronously by performing matrix multiplication column-by-column, leaving the massive parallelism of the crossbar not fully exploited. In this work, we propose and *experimentally* demonstrate solving the optimization problems with a synchronous-updating memristor-based Ising solver, which is realized by a binary neural network-inspired updating algorithm and a physics-inspired annealing method. The newly proposed method saves more than 5× time and 35× energy consumption compared to the state-of-the-art mem-HNN for finding the optimal solution to a 60-node Max-cut problem.

I. Introduction

Combinatorial optimization problems (COPs) aim at minimizing a cost or energy function over a set of high-dimensional variables, which find great importance in a large variety of industrial applications such as supply chain management, flight scheduling, circuit layout design, etc. Such problems are highly time-consuming for an exact solution with conventional hardware, as most fall into non-deterministic polynomial-time (NP)-hard problems. On the other hand, such problems can be solved efficiently with heuristic approaches by mapping them to Ising models implemented on various platforms, including superconductor q-bits [1], coherent light [2, 3], CMOS oscillators [4], and nano-oscillators [5].

In addition to physics-inspired Ising machines, a neuromorphic algorithm called Hopfield neural network (HNN) implemented on memristor crossbar arrays has shown great potential both in the time and energy consumption towards solutions compared to Ising machines. One challenge for this method is that the system can get stuck in local minima. So, to get the global minima (the best solution), many techniques have been explored, including using stochastic neurons for implementing simulated annealing (SA) [6-8], using Mott memristors as chaos sources [9], changing self-feedbacks for chaotic simulated annealing (CSA) [8, 10], or harnessing intrinsic noises by hysteretic thresholds [11]. However, *to the best of our knowledge*, all these demonstrations update only one node at each iteration (i.e., asynchronous), by performing the matrix multiplication via one or several crossbar columns at once (see **Fig. 1**). This is because of the restrictions of the discrete-time HNN (DHNN). Therefore, they did not fully utilize the parallelism of the memristor crossbar array.

To enable the synchronous updating and unleash the massive parallelism of the memristor crossbar array (**Fig. 2**), we learn from the training process of a binary neural network and *for the first time,* apply it in a memristor-based Ising solver. Besides, a physics-inspired annealing method is applied to get rid of local minima. The software-hardware co-optimized memristor-based Ising solver (MIS) is experimentally implemented in an integrated memristor crossbars platform, demonstrating more than 5× speed-up and 35× energy-saving in solving middle-size COPs and more speed-up and energy-saving for larger-sized COPs.

II. Parallel Updating Algorithm

The Ising machines aim to find a spin configuration that gives a minimum Ising Hamiltonian, which is similar to the process that finds a lower-cost function in the training of a neural network. The Ising Hamiltonian is defined as

$$H_{\text{Ising}} = -\sum_{i<j} J_{ij}\sigma_i\sigma_j = -\frac{1}{2}\boldsymbol{\sigma}^T\boldsymbol{J}\boldsymbol{\sigma},$$

where $\boldsymbol{\sigma} = \{\sigma_1, \sigma_2, \ldots, \sigma_i, \ldots, \sigma_N\}$ encodes the spin configuration with each component $\sigma_i \in \{-1, +1\}$. $\boldsymbol{J}$ is a $N \times N$ symmetric spin coupling matrix which is defined by the combinatorial optimization problem and can be represented in crossbars. The final $\boldsymbol{\sigma}$ configuration that leads to the minimum Ising Hamiltonian is the solution to the problem.

The Hamiltonian can be minimized following a gradient descent rule, but the challenge is that the spin values must be either +1 or -1. The solution is to assign a real number x_i to represent the intermediate spin values, update it based on the accumulated gradient after each iteration with constrain rules, and then convert that into the binarized spin values σ_i after the iterations. The most computation expensive step during the operation, the calculation of the gradient, can be performed efficiently in the memristor crossbar, following equation:

$$\nabla H_{\text{Ising}} = -\boldsymbol{J}\boldsymbol{\sigma} = -\boldsymbol{J}\text{sign}(\boldsymbol{x}) \qquad (1)$$

The process finds similarities with the training of a BNN, where real numbered weights are being updated based on the accumulated gradient with some constraining rules, and binarized afterward [12]. Accordingly, techniques to improve the gradient descent can also be applied to our method. In this work, we apply momentum with a value of 0.99, to speed up the convergence and lower the energy consumption. The algorithm is described in more detail in **Fig. 3**.

Another challenge is that the gradient descent-based algorithm can cause the system to be trapped in local minima, preventing the optimal solution. Inspired by quantum adiabatic annealing [1], we propose a new annealing method that is designed for memristor Ising solver, which follows the principle of gradual non-convexity (**Fig. 4**). The system

Hamiltonian is defined as $H_{\text{system}}(t) = \lambda(t)H_{\text{initial}} + H_{\text{Ising}}$, where H_{initial} is the initial Hamiltonian with easy access to its ground state. In this work, we use $H_{\text{initial}} = \frac{1}{2}\sum_i x_i^2$, whose ground state is located when all spin values $x_i = 0$. $\lambda(t)$ is a time-dependent coefficient, which starts with a sufficiently large value so that the H_{initial} dominates and the ground state can be easily found. Then, $\lambda(t)$ will gradually decrease to zero and lead the system to the ground state of H_{Ising}.

III. Integrated Memristor-based Solver for Max-Cut Problems

The proposed algorithm is experimentally implemented in our integrated memristor crossbar system. **Fig. 5** shows the experimental setup of the memristor Ising solver. The selection transistors in the 64x64 1T1R array and the peripheral circuits (amplifiers, sample and hold, ADC, etc.) are tape-out at TSMC's 180 nm technology node. 50 nm×50 nm Ta/TaO$_x$/Pt memristor devices are integrated with our in-house back-end process. Our memristor device can stably maintain multilevel states (16 states shown in **Fig. 6**). An off-chip measurement circuit is developed in a printed circuit board (PCB) to communicate with the memristor chip via a microcontroller. In this work, spin couplings are stored in the memristor crossbar for computing gradient in the analog domain, while spin updates are performed in the digital domain.

To evaluate the performance of our proposed method, we choose a typical NP-hard problem, "Max-cut". The problem aims to divide all vertices (nodes) in a graph $G(V, E)$ into two subsets that can cut the most edges connecting vertices. The graph of a 64-node Max-cut problem with edge density of 0.5 (defined as $2E/(V(V-1))$) is shown in **Fig. 7**. The connection matrix of the problem is then converted to conductance matrix and programmed into our memristor crossbar. **Fig. 8** and **Fig. 9** show the experimental readout conductance matrix and the corresponding conductance distribution, respectively, which shows that the problem can be programmed accurately. Then, the gradient can be computed in a physical crossbar with **Eq. 1**. **Fig. 10** shows the computing error distribution with random inputs. One sees that the gradients can be calculated precisely in the analog domain.

IV. Benchmarks

Once the problem is programmed into the memristor array, the Max-cut problem can be solved iteratively with the above algorithm without the need to re-program the array. **Fig. 11** shows the *experimental* energy evolving process of our method (MIS) as compared to the naïve discrete-time HNN (DHNN) [13] and the DHNN with simulated annealing (SA) [7, 8, 11]. One sees that the success probabilities of reaching the optimal solution (energy of -188) with our MIS, SA, and DHNN are 0.88, 0.10, and 0.01, respectively. The final solutions for this experiment are shown in **Fig. 12**. The success probability and the average final energy during iterations are compared in **Fig. 13**, which clearly demonstrates that our method can find better solutions than others. The successful and efficient convergence is based on a fine-tuned step size (**Fig. 14**), which is similar to the choice of learning rate during the training of a BNN.

Those results (Fig. 13 & Fig. 14) also show a high agreement between the experiments and the simulations based on our experimentally validated crossbar model [14], which provides a powerful tool to forecast the performance of our method in scaled problems. **Fig. 15a**, **Fig. 15b** show success probability against problem sizes with different numbers of iterations, by implementing our method and SA respectively. Time-to-solution (TTS, defined as the time needed to reach the 99% success probability) of our method is 74× faster than SA when the problem size N=100 and the ratio further grows with larger problems **(Fig. 16)**. Naïve DHNN is not compared here because the local-searching algorithm has a tiny chance of success. For problems with various densities, our method shows great overall performance due to its all-to-all connection property (**Fig. 17**).

Another benefit of deploying this method in the crossbar is that the conductance variation of memristor devices can be used as a perfect noise source so that the system will not get stuck in the ground state of H_{initial}. **Fig. 18** shows that the intrinsic noise from the memristor crossbar is large enough to head-start the convergence but not too large to degrade the performance.

Finally, the area and energy consumption is estimated with consideration of a peripheral circuit design at a 16 nm technology node (detailed breakdown shown in **Fig. 19**). The results show that our system offers at least 5× improvement in the time-to-solution and 35× in energy efficiency compared to the recent reported best-performed solver (**Fig. 20**) primarily due to its better parallelism. Those numbers are based on the 60-node Max-cut problems and can be further improved for larger problems.

V. Conclusion

To sum up, we developed a synchronous-updating algorithm with a physics-inspired annealing method to solve the combinatorial optimization problems in memristor crossbars. This method is different from previously reported memristor-based approaches in the way that all nodes can be updated in parallel to make full use of the memristor's parallelism. The idea is experimentally validated in our integrated memristor platform, and our analysis shows the proposed memristor-based Ising solver can improve both the speed, energy, and final accuracy by joint benefits from the new algorithm and hardware.

Acknowledgment

This work was supported in part by the Early Career Scheme from the Research Grant Council of Hong Kong SAR under Grant 27210321; in part by the NSFC Excellent Young Scientist Fund (Hong Kong and Macau) under Grant 62122005; in part by the Mainland-Hong Kong Joint Funding Scheme (MHKJFS) under Project MHP/066/20; and in part by the ACCESS—AI Chip Center for Emerging Smart Systems, sponsored by InnoHK funding, Hong Kong SAR.

References

[1]. S. Boixo, et al. *Nat. Phys.* (2014) **[2]**. T. Inagaki, et al. *Science* (2016) **[3]**. P.L. Mcmahon, et al. *Science* (2016) **[4]**. W. Moy, et al. *Nat. Electron.* (2022) **[5]**. S. Dutta, et al. *Nat. Electron.* (2021) **[6]**. J. H. Shin, et al. *IEDM* (2018) **[7]**. M. C. Hong, et al. *IEDM* (2021) **[8]**. M.R. Mahmoodi, et al. *Nat. Commun.* (2019) **[9]**. S. Kumar, et al. *Nature* (2020) **[10]**. K. Yang, et al. *Sci. Adv.* (2020) **[11]**. F. Cai, et al. *Nat. Electron.* (2020) **[12]**. M. Courbariaux, et al. *NeurIPS* (2015) **[13]**. J.J. Hopfield *Proc. Natl. Acad. Sci. U.S.A.* (1984) **[14]**. R. Mao, et al. *IEEE Trans. Circuits Syst. II Express Briefs* (2022) **[15]**. J. Liu, et al. *ISSCC* (2022).

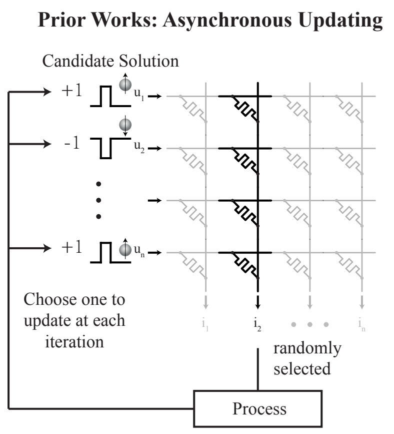
Prior Works: Asynchronous Updating
Candidate Solution
+1
-1
+1
Choose one to update at each iteration
randomly selected
Process

Fig. 1: The diagram of the working principle for existing asynchronous updating mem-HNN solver, which updates one or several nodes at each iteration.

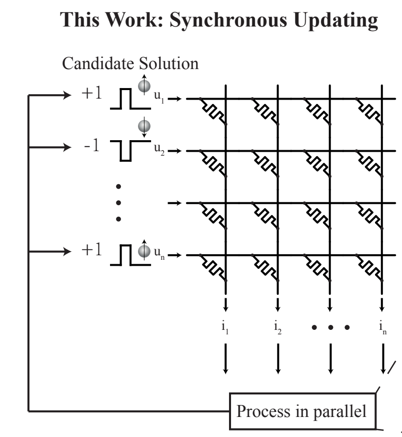

Fig. 2: The diagram of the proposed synchronous updating memristor-based Ising solver, which updates all nodes at each iteration.

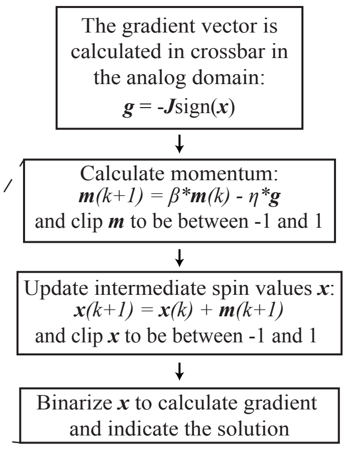
The gradient vector is calculated in crossbar in the analog domain: $\boldsymbol{g} = -\boldsymbol{J}\mathrm{sign}(\boldsymbol{x})$
Calculate momentum: $\boldsymbol{m}(k+1) = \beta * \boldsymbol{m}(k) - \eta * \boldsymbol{g}$ and clip $\boldsymbol{m}$ to be between -1 and 1
Update intermediate spin values $\boldsymbol{x}$: $\boldsymbol{x}(k+1) = \boldsymbol{x}(k) + \boldsymbol{m}(k+1)$ and clip $\boldsymbol{x}$ to be between -1 and 1
Binarize $\boldsymbol{x}$ to calculate gradient and indicate the solution

Fig. 3: The diagram of the the synchronous updating algorithm.

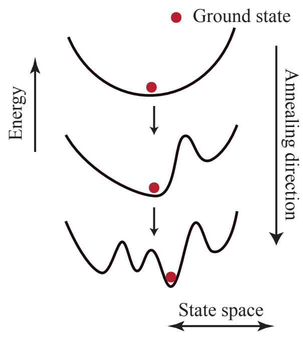

Fig. 4: The working principle of proposed gradually non-convexity annealing that inspired from quantum adiabatic annealing.

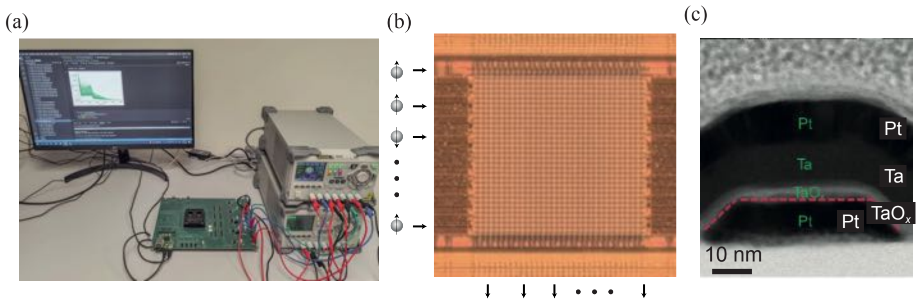

Fig. 5: (a) The integrated memristor-based Ising solver runing combinatorial optimization problems. The computer talks to the memristor chip through a microcontroller on a PCB board. (c) The photo of a 64×64 1T1R array, which is the core of the memristor-based Ising solver. (c) The cross-sectional photo of the Ta/TaO$_x$/Pt memrsitor device.

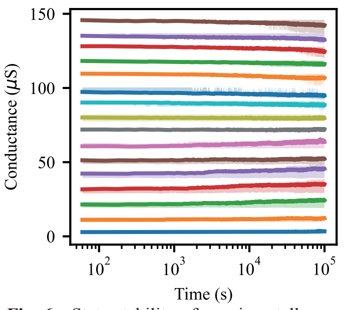

Fig. 6: State stability of expeimentally programmed 16 conductance states. The solid lines show the mean over 256 devices that are programmed to the same target conductance and the shaded area indicates the interquartile range.

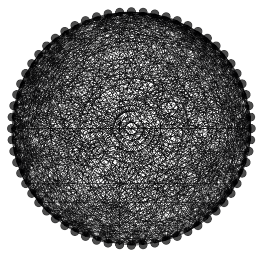

Fig. 7: The graph of the dense Max-cut problem to solve.

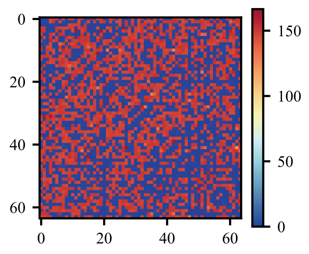

Fig. 8: The readout conductance matrix after experimentally programming the Max-cut problem into the integrated crossbar. 0s representing no connection are mapped to low conductance state (target 0 µs). 1s representing connections are mapped to high conductance state near 150 µs.

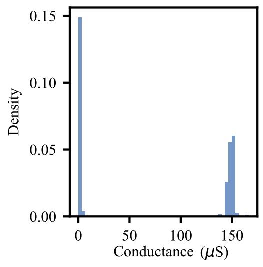

Fig. 9: The histogram of the conductance distribution.

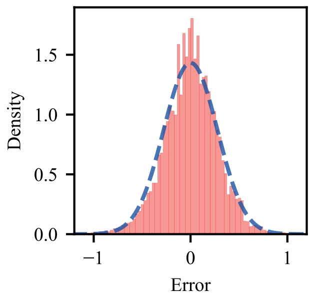

Fig. 10: Nomalized error distribution by applying 1000 random input. The standard deviation is 0.28 while the range of the output is -64 ~ 64.

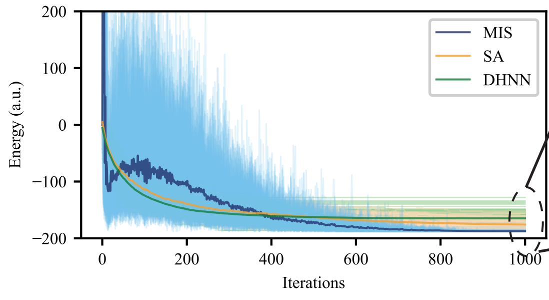

Fig. 11: The *experimental* energy evolving process with the proposed memristor-based Ising solver (**MIS**) in this work, discrete-time Hopfiled neural network (**DHNN**) and DHNN with simulated annealing (**SA**). The light color represents 100 different trials and dark color represents the mean. Step size is 0.01 in this experiment and λ changes linearly from 10 to 0 for MIS.

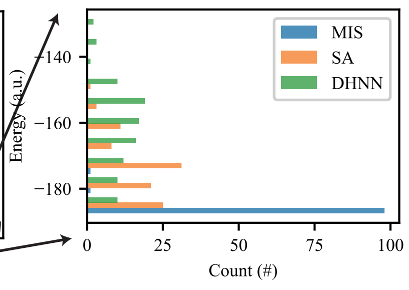

Fig. 12: The histogram of the final energy (the solution of the system) with proposed memristor-based Ising solver (**MIS**) in this work, discrete-time Hopfiled neural network (**DHNN**) and DHNN with simulated annealing (**SA**).

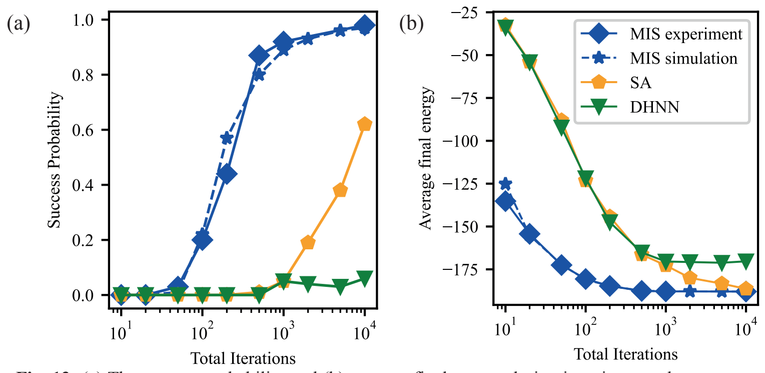

Fig. 13: (a) The success probability and (b) average final energy during iterations used per run. The blue dash line is the simulation with our experimentally validated model while the solid line is the experimental result from the physical array.

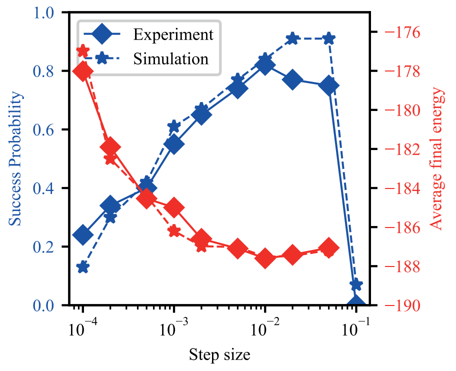

Fig. 14: The success probability and average final energy changes with the step size during the update. In a certain range, system performance improves with the increased step sizes. However, when the step size is larger than a certain value, the system fails to converge.

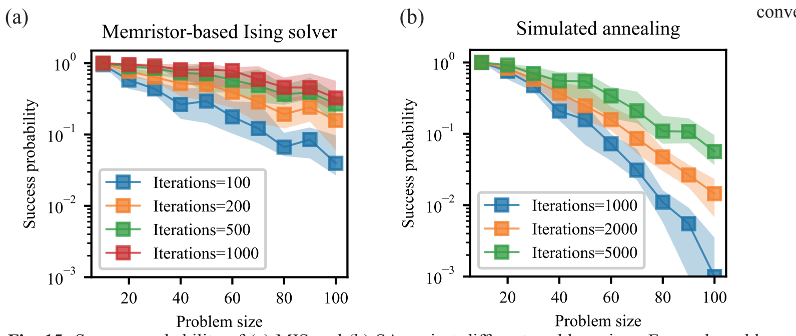

Fig. 15: Success probability of (a) MIS and (b) SA against different problem sizes. For each problem szie, 20 random generated instances are used. For each instance, 1000 trials are peformed to calculate success probability. The shaded area shows interquartile range. The square marker indicates the median.

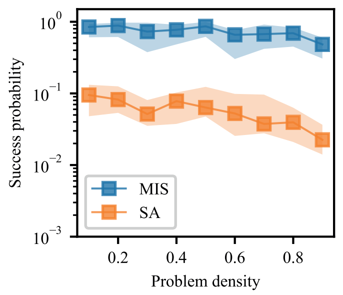

Fig. 17: Success probability vs problem density after the 1000th iteration.

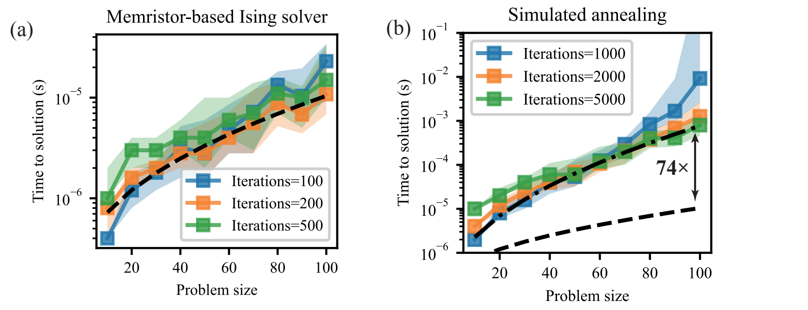

Fig. 16: Time to solution (TTS) of (a) MIS and (b) SA against different problem sizes. TTS is calculated as $T_{\text{single-run}}\times[\log(1\text{-}0.99)/\log(1\text{-}P)]$. Both methods are assumed to be implemented on memristor crossbars with a working frequency of 500Mhz. A fit curve of $TTS = ae^{b\sqrt{N}}$ is plotted to show the scaling property.

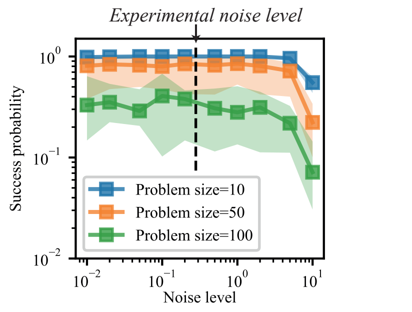

Fig. 18: Success probability vs noise levels. The dash line shows the experimental noise level.

Module	Array	Drivers	S&H	MUX	ADC	Sum
Area (μm²)	56.2	176.39	2.02	65.15	2850*4	10,619.76
Latency (ns)	<0.3	0.1	-	0.008	-	-
Energy (pJ)	0.63	0.22	0.01	0.13	1.59	2.58

Fig. 19: The area and energy consumption estimation breakdown of peripheral circuits assuming they are designed at a 16nm technology node. A state-of-the-art 8-bit ADC in [15] is used for estimation.

	This work	**mem-HNN [11]**	**PTNO [5]**	**CIM [2]**	**D-wave 2000Q [1]**	**GPU [11]**	**CPU [11]**
Representation of spins	Digital bits	-	Oscillator phases	Coherent light	Superconducting qubits	Digital bits	Digital bits
Interactions	Conductance in memristor crossbar array	-	Capacitance/ resistance	FPGA	Flux storage	Coupling matrix stored in digital memory	Coupling matrix stored in digital memory
Connectivity	All-to-all	All-to-all	All-to-all	All-to-all	Sparse	All-to-all	All-to-all
Update mechanism	Synchronous	Hybrid (10-nodes per time step)	Synchronous	Asynchronous	Synchronous	Synchronous	Asynchronous
Annealing Scheme	Gradual non-convexity annealing	Modulating intrinsic noises by hysteretic threshold	Second-harmonic injection lock	Coherent computing	Quantum annealing	Noisy mean-field annealing	Parallel tempering
Annealing time	400 ns	600 ns	-	150 μs	1ms	12.3 μs	223.6 μs
Time to Solution	1.6 μs	6.6 μs	30 us (N=100)	600 μs	10^4 s (N=55)	10 μs	223.6 μs
Power	1.29 mW	10.9 mW	2.56 mW	-	25 kW	<250 W	20 W
Energy to Solution	2.06 nJ	72 nJ	76.8 nJ	-	250 MJ	<2.5 mJ	4 mJ
Solutions per second per watt	4.85×10^8	1.38×10^7	1.3×10^7	-	4×10^{-9}	>400	250

Fig. 20: Benchmark comparison between different Ising machines and solvers in solving dense 60-node Max-cut problems.