QSBMs: Lightweight Quantized Simulated Bifurcation Ising Machines

Tingting Zhang, *Member, IEEE,* and Jie Han, *Senior Member, IEEE*

***Abstract*—Ising machines have shown great potential as efficient domain-specific accelerators for solving combinatorial optimization problems (COPs). Derived from quantum mechanics, simulated bifurcation (SB) achieves massive parallelism in updating the spin states in an Ising machine. Although SB speeds up the search for a solution compared to traditional simulated annealing, it requires more hardware resources since continuous variables are used for the positions of oscillators to obtain discrete spin states. In this article, lightweight quantized SB Ising machines (QSBMs) are developed to achieve a better tradeoff between search performance and hardware efficiency. In various quantization schemes, ternary and multiple-value quantized SB (qSB) algorithms discretize the position variables for the multiply-and-accumulate (MAC) operations in SB. The ternary qSB with dynamic threshold settings converts the MAC into addition, while uniform and logarithmic quantization schemes improve precision in the number representation when solving large-scale COPs. Three hardware-efficient QSBMs are subsequently designed with a fully connected topology. Synthesized on a Xilinx Virtex UltraScale+ field-programmable gate array (FPGA), the costly multiplication is implemented by using simple logic operators. Fully connected 2048-spin QSBMs use up to 50.8% fewer lookup tables and up to 82.5% fewer flip-flops than conventional FPGA-based SB machines. The QSBMs are superior in both long and short searches, respectively reaching 99 .1% and 98.5% of the best known solution in** $1.46\ ms$ **and** $0.73\ ms$ **on solving 2000-spin Ising problems.**

***Index Terms*—Simulated Bifurcation, Quantization, Ising Model, Ising Machine, Combinatorial Optimization.**

I. Introduction

As Dennard's scaling is approaching the end, the performance improvement of general-purpose processors is slowing down. This leads to the development of domain-specific hardware accelerators based on novel computing paradigms. Combinatorial optimization problems (COPs) exist in cell placement, wire routing, and logic minimization in very large-scale integrated designs [1]; however, they are computationally non-deterministic polynomial time-hard to solve. As a potential solution, the Ising model describes the ferromagnetism in a set of spins, where spins naturally orient to let the Ising system converge to the lowest energy state. The Ising model-based computing architecture, referred to as the Ising machine, shows promise for efficiently solving COPs in the post-Moore era [2].

Existing Ising machines are broadly categorized into two classes. One utilizes physical systems to implement spins through quantum phenomena (e.g., superconducting flux qubits) [3] or classical analog platforms (e.g., optical/electronic oscillators) [4]–[6]. The other relies on heuristic algorithms to numerically simulate the spin dynamics [7]–[9]. The latter supports dense spin-to-spin interactions with a high precision and offers high reliability, which however is challenging for the implementation of the former class. There are two classical types of heuristic algorithm-based Ising machines [10], [11]. The first type is the annealing Ising machine, which simulates the thermal annealing process [8], [12]–[17]. Despite the incorporation of techniques such as parallel trials [12], [13], parallel tempering [14], and quantum fluctuation [17], most annealing Ising machines update interconnected spins sequentially in a stochastic manner. The adoption of a two-layer structure in stochastic cellular automata annealing enables the simultaneous updates of all spin states [8], [18]; however, this approach imposes additional memory overhead and extra constraints in order to ensure that two replicas of spins maintain identical states. The probabilistic bit (p-bit) based annealing Ising machine models spins as p-bits and numerically emulates their behavior [19]. The second type is the dynamic Ising machine, which describes the dynamics of oscillator networks. Representative examples include simulated bifurcation (SB) Ising machines inspired by Kerr-nonlinear parametric oscillator networks [9], [20]–[23], emulated coherent Ising machines inspired by degenerate optical parametric oscillators [24], and emulated oscillator-based Ising machines inspired by electronic nonlinear oscillator networks [25]–[27]. Both p-bit annealing Ising machines and dynamic Ising machines make the parallel update of all spins possible, irrespective of connectivity.

Among those heuristic-based Ising machines that update spin states in parallel, the SB Ising machine distinguishes itself by eliminating the dependence on random number generators to ensure energy convergence, thus reducing hardware complexity and improving computational efficiency. However, it incurs a high hardware cost, especially for the multiply-and-accumulate (MAC) operation due to the use of continuous variables for the oscillator positions to obtain discrete spin states (-1 or $+1$). The recently developed discrete SB (dSB) binarizes the position values by using their signs for MAC [20]. It achieves a significant reduction in computational complexity, but at the cost of a longer search time than the original SB [9]. Therefore, it becomes interesting to investigate the effect of

Tingting Zhang was with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada. Current address: the Department of Electrical and Computer Engineering, McGill University, Montreal H3A 0E9, Canada (e-mail: ttzhang@ualberta.ca).

J. Han is, with the Department of Electrical and Computer Engineering, University of Alberta, Edmonton, AB T6G 1H9, Canada. (e-mail: jhan8@ualberta.ca).

quantizing the position values in SB for a better trade off between computational efficiency and search performance.

In this article, quantized SB machines (QSBMs) are designed for solving COPs quickly in a short search and accurately in a long search. Three quantized SB (qSB) algorithms are initially proposed by applying different quantization schemes for the position values as inputs to the MAC. The ternary qSB quantizes the position values into $\{0, \pm 1\}$, depending on a dynamic threshold that changes with time. Multiple-value qSB algorithms, based on uniform and logarithmic quantizations, are further developed to achieve a more accurate number representation for solving large-scale COPs. The performance of the qSB is evaluated by analyzing the network dynamics and solving instances of Ising problems with up to 2000 fully and sparsely connected spins. To implement the proposed qSB algorithms, efficient QSBMs are developed, which avoid the hardware-consuming multiplication operations.

Some of our preliminary results were reported in [21]. The novel contributions of this article are summarized as follows:

- The dynamics of the proposed and existing SB algorithms are analyzed by solving an instance of a two-spin Ising problem.
- A general architecture of the QSBM is developed for the proposed qSB. In particular, hardware efficient quantization and multiplication units are respectively designed for three qSB algorithms.
- Three 2048-spin QSBMs are implemented and synthesized on a Xilinx Virtex UltraScale+ field-programmable gate array (FPGA) and evaluated on benchmarks of max-cut problems.

The rest of this article is organized as follows: Section II presents preliminaries. Section III proposes three qSB algorithms. The effects of quantization used in qSB algorithms are analyzed in Section IV. The circuit designs of the QSBMs are presented in Section V. The hardware performance of the QSBMs is evaluated in Section VI. Finally, Section VII concludes the article.

II. Preliminaries

An N-spin Ising problem is to find the spin states, denoted by $\mathbf{s}$, that minimize the total energy (i.e., Hamiltonian) of the Ising model. The Hamiltonian, $H(\mathbf{s})$, is given by [28]

$$H(\mathbf{s}) = -\sum_i^N h_i s_i - \tfrac{1}{2}\sum_{i,j}^N J_{ij} s_i s_j, \tag{1}$$

where s_i (or s_j, $\in \{-1, +1\}$) is the state of the ith (or jth) spin, J_{ij} is the interaction between the ith and jth spins (so $J_{ij} = J_{ji}$ and $J_{ii} = 0$), and h_i is the external field (or bias) placed on the ith spin. Since the model can be reduced to one without the external fields by introducing an ancillary spin [**?**], we focus on the Ising model without external fields in the following discussion.

SB digitally emulates the adiabatic evolution of oscillator networks [29]. The behavior of each spin in the Ising model is therefore simulated by an oscillator. Let $\boldsymbol{x}$ and $\boldsymbol{y}$ be the oscillator position and momentum values, respectively. The Hamiltonian of an N-spin SB system, $H_{SB}(\boldsymbol{x}, \boldsymbol{y})$, can be expressed by

$$H_{SB}(\boldsymbol{x}, \boldsymbol{y}) = V(\boldsymbol{x}) + \sum_{i=1}^N \tfrac{a_0}{2} y_i^2, \tag{2}$$

where $V(\boldsymbol{x})$ is the potential energy related to oscillator position values $\boldsymbol{x}$, y_i is the momentum value of the ith oscillator, and a_0 is a manually tuned constant.

SB simulates Hamiltonian dynamics with quantum adiabatic bifurcation in a nonlinear oscillator network [9], [23], [29]. To restrain errors introduced by using continuous position variables to represent discrete spin states, two variants, called ballistic SB (bSB) and dSB were developed in [**?**]. The potential energy is computed by

$$V(\boldsymbol{x}) = \begin{cases} \frac{a_0 - a(t)}{2}\sum_i^N x_i^2 - \frac{c_0}{2}\sum_i^N\sum_j^N J_{ij} x_i x_j & \text{in bSB} \\ \frac{a_0 - a(t)}{2}\sum_i^N x_i^2 - c_0\sum_i^N\sum_j^N J_{ij} x_i sgn(x_j) & \text{in dSB} \end{cases} \tag{3}$$

where x_i is the position value of the ith oscillator, $a(t)$ is a time-dependent control parameter to guarantee the adiabatic evolution, c_0 is a manually tuned constant, and $sgn(\cdot)$ outputs "+1" if the input is a positive number and outputs "−1" if the input is a negative number.

Both bSB and dSB essentially search for an approximate solution by solving a pair of differential equations related to the positions and momenta of oscillator networks [**?**]. They follow the Hamiltonian equations of motion, given by [**?**]

$$\dot{x_i} = \frac{\partial H_{SB}}{\partial y_i} = a_0 y_i, \tag{4}$$

$$\dot{y_i} = -\frac{\partial H_{SB}}{\partial x_i} = -\{a_0 - a(t)\}x_i + c_0 P_i, \tag{5}$$

where $\dot{x_i}$ and $\dot{y_i}$ denote the derivatives of x_i and y_i with respect to time; P_i is used for updating y_i. x_i is replaced by its sign and y_i is reset to 0 when $|x_i| > 1$. The bSB and dSB differ from the expressions of P_i, given by [**?**]

$$P_i = \begin{cases} \sum_{j=1}^N J_{ij} x_j & \text{in bSB} \\ \sum_{j=1}^N J_{ij} sgn(x_j) & \text{in dSB} \end{cases}. \tag{6}$$

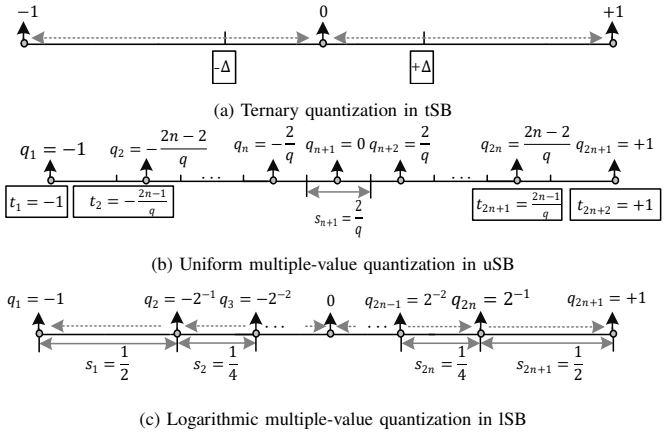
To solve an Ising problem, position values are first randomly initialized. Then, the semi-implicit Euler method is usually utilized to solve the pairs of differential equations in (4) and (5). At the end of a search, the sign of x_i indicates the state of the ith spin. Note that in what follows, we denote the Hamiltonian and potential energy of various SB systems by $H(\boldsymbol{x}, \boldsymbol{y})$ and $V(\boldsymbol{x})$, respectively, with a subscript to indicate the type of SB.

III. Quantized Schemes for Simulated Bifurcation

Depending on the sign bit, dSB binarizes x_j to $\{-1, 1\}$ when computing P_i, as in (6). Thus, the MAC is simplified to addition and subtraction, so dSB achieves a significant reduction in hardware. However, it results in an unstable solution quality and relatively slow convergence in energy [21]. In this section, we improve the SB with different quantization methods for the position values used as inputs to the MAC for a better trade-off between solution quality and computational complexity.

A. Ternary Simulated Bifurcation

The ternary SB (tSB) utilizes three-valued position variables (such as $x_j \in \{-1, 0, 1\}$, where $j \in [1, N]$) to compute $\boldsymbol{P}$. Similar to dSB, the computation of $\boldsymbol{P}$ avoids using multipliers. Moreover, $\boldsymbol{x}$ is compressed as a result of a high probability of quantizing position values to zeros.



(a) Ternary quantization in tSB

(b) Uniform multiple-value quantization in uSB

(c) Logarithmic multiple-value quantization in lSB

Fig. 1: Quantization schemes in SB algorithms.

Let $\hat{\boldsymbol{x}}$ be the quantized position values. As shown in Fig. 1(a), a dynamic threshold Δ determines the mapping from $\boldsymbol{x}$ to $\hat{\boldsymbol{x}}$. In this way, P_i is computed as

$$P_i = \sum_{j=1}^{N} J_{ij} tri(x_j), \tag{7}$$

where

$$tri(x_j) = \begin{cases} 0 & |x_j| \le \Delta \\ sgn(x_j) & others \end{cases}, \tag{8}$$

where an empirically optimized value for Δ, when using the ternary quantization method, is $\Delta^* = \frac{0.7\|\boldsymbol{x}\|_{l1}}{N}$ to minimize the Euclidean distance between $\hat{\boldsymbol{x}}$ and $\boldsymbol{x}$ [30].

To compute Δ^*, the L1 norm of $\boldsymbol{x}$ needs to be evaluated and it requires the accumulation of N elements in $\boldsymbol{x}$ in each time step, thus incurring a relatively high cost. The elements in $\boldsymbol{x}$ evolve from random values around 0 to +1 or −1. Therefore, Δ^* increases with time from an extremely small positive value to around 0.7. We then use a less computationally expensive function related to time $\Delta(t)$ to simulate the function of Δ^*, as

$$\Delta(t) = 0.7 \times \frac{t}{T_s}, \tag{9}$$

where t and T_s denote the current time step and the total number of time steps, respectively.

The tSB computes the potential energy as

$$V_{tSB}(\boldsymbol{x}) = \frac{a_0 - a(t)}{2} \sum_i^N x_i^2 - c_0 \sum_i^N \sum_j^N J_{ij} x_i tri(x_j). \tag{10}$$

The Hamiltonian of the tSB system ($H_{tSB}(\boldsymbol{x}, \boldsymbol{y})$) is given by replacing $V(\boldsymbol{x})$ in (2) with $V_{tSB}(\boldsymbol{x})$. The derivative of y_i with time is given by replacing P_i in (5) with (7).

B. Uniformly Quantized Simulated Bifurcation

Multiple-value quantization is further considered to improve computational accuracy. It is relatively costly to use a dynamic threshold to identify different quantization levels. Thus, a fixed threshold is used in the multiple-value qSB. Assume that the position values are quantized to q $(= 2n + 1)$ levels. The gap between two adjacent levels is called the quantization step size. Let q_l be the quantized value of the lth level of the quantization intervals from t_l to t_{l+1}. Its step size is denoted by s_l $(= t_{l+1} - t_l)$.

As shown in Fig. 1(b), uniform quantization uses the same step size s_l $(= \frac{2}{q})$ in different quantization levels. To simplify the computation, the position variables in the 1st and $(2n + 1)$th quantization levels, i.e., when $\frac{2n-1}{q} < |x_j| \le 1$, are quantized to $sgn(x_j)$. Except for these two levels, for the lth level, q_l is given by the middle value in $[t_l, t_{l+1})$, as $q_l = -1 + \frac{2l-1}{q}$ to maintain the accuracy. In this way, x_j is quantized to $\{0, \pm\frac{2}{q}, ..., \pm\frac{2n-2}{q}, \pm 1\}$. Thus, assuming $p_{x_j} = round(\frac{q|x_j|}{2})$, where $round()$ changes a value to its nearest integer, the q-level uniform qSB (uSB) computes P_i as

$$P_i = \sum_{j=1}^{N} J_{ij} u(x_j), \tag{11}$$

where

$$u(x_j) = \begin{cases} sgn(x_j) \cdot \frac{2p_{x_j}}{q} & p_{x_j} <= n-1 \\ sgn(x_j) & others \end{cases}. \tag{12}$$

Therefore, the uSB computes the potential energy as

$$V_{uSB}(\boldsymbol{x}) = \frac{a_0 - a(t)}{2} \sum_i^N x_i^2 - c_0 \sum_i^N \sum_j^N J_{ij} x_i u(x_j), \tag{13}$$

and its Hamiltonian ($H_{uSB}(\boldsymbol{x}, \boldsymbol{y})$) is given by replacing $V(\boldsymbol{x})$ in (2) with $V_{uSB}(\boldsymbol{x})$. Similarly, the derivative of y_i with time is obtained by replacing P_i in (5) with (11).

C. Logarithmic Quantized Simulated Bifurcation

The bifurcation that occurs at the beginning of a search plays an important role in SB. Thus, computation with a relatively high precision is preferred for the initially small position values. When the magnitudes of the position values become larger with time, it will be harder to change their sign bits, which makes the system vulnerable to local minima. Therefore, less accurate computation for positions with relatively large absolute values may help the system search for a better solution.

Logarithmic quantization [31] is then applicable in the multiple-value qSB. Different from uniform quantization, logarithmic quantization results in an exponential difference in s_l and s_{l+1}. The two quantized values are uniformly distributed in the base 2 logarithmic domain. In this way, as presented in Fig. 1(c), the q-level logarithmic qSB (lSB) quantizes x_j into $\{0, \pm 2^0, \pm 2^{-1}, ..., \pm 2^{-(n-1)}\}$, so the multiplications in $\boldsymbol{P}$ are realized directly through shift operations over a wide numerical representation range for x_j. Assume $|x_j| = 2^{\widetilde{x_j}}$, then $\widetilde{x}_j = \lceil log_2 |x_j| \rceil$. In the q-level lSB, P_i is computed by

$$P_i = \sum_{j=1}^{N} J_{ij} lg(x_j), \tag{14}$$

where

$$lg(x_j) = sgn(x_j) \cdot 2^{\widetilde{x_j}}. \tag{15}$$

The expression of P_i is further simplified as

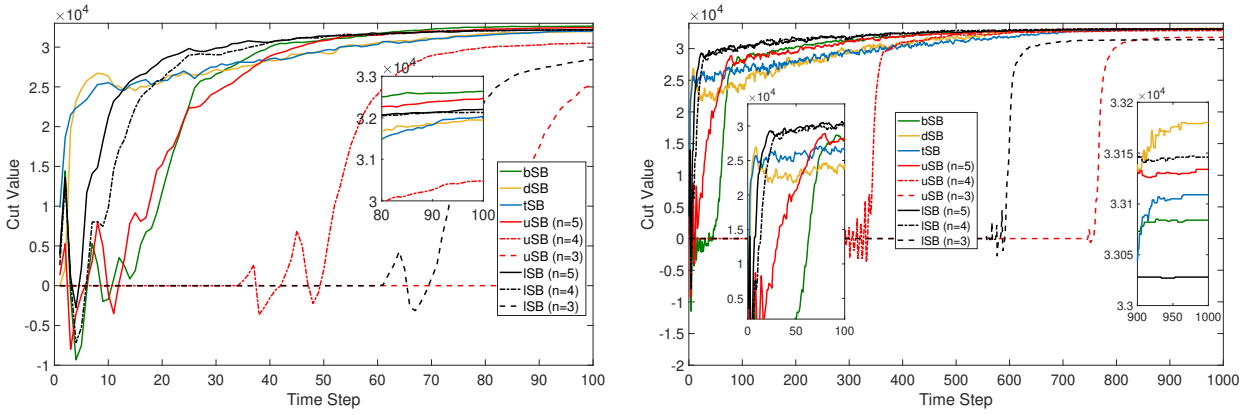
$$P_i = \sum_{j=1}^{N} sgn(x_j) s(J_{ij}, \widetilde{x}_j), \tag{16}$$

where $s()$ right shifts the binary representation of J_{ij} by $|\widetilde{x}_j|$ bits, when $\widetilde{x_j} > -n$; otherwise, $s()$ outputs 0.

The lSB computes the potential energy as

$$V_{lSB}(\boldsymbol{x}) = \frac{a_0 - a(t)}{2} \sum_i^N x_i^2 - c_0 \sum_i^N \sum_{j=1}^N sgn(x_j) s(J_{ij}, \widetilde{x}_j) x_i, \tag{17}$$

and similarly, its Hamiltonian ($H_{lSB}(\boldsymbol{x}, \boldsymbol{y})$) is given by replacing $V(\boldsymbol{x})$ in (2) with $V_{lSB}(\boldsymbol{x})$. $\dot{y_i}$ is expressed by replacing P_i in (5) with (16).



(a) $T_s = 100$ (b) $T_s = 1000$

Fig. 2: Time evolutions of the cut values in a single trial on solving a 2000-spin fully connected Ising problem, the K_{2000} benchmark, by using SB algorithms. A fast convergence in Hamiltonian (indicated by the quickly increasing cut values) is preferred for accelerating the search. The fluctuations in Hamiltonian (indicated by the instability of cut values) are beneficial for jumping out of the local minima in a long search.

IV. Effects of Quantization on Simulated Bifurcation

A. Time Evolution

The max-cut problem (MCP) is to split vertices in an undirected weighted graph into two groups to maximize the cut value, which is the total weight of edges between two groups. Fig. 2 shows the time evolutions of the cut values when solving K_{2000} benchmark using tSB, uSB, and lSB with $T_s = 100$ and 1000 time steps (i.e., iterations). The time evolution of the cut values shows energy convergence of the Ising model. Since tSB using different Δ (Δ^* and $\Delta(t)$) shows a similar pattern in energy convergence, the tSB with Δ^* is considered as an example. Note that the parameter settings for all datasets are as follows: $a_0 = 1$, $a(t)$ increases from 0 to 1, and $c_0 = \frac{\sqrt{N-1}}{2\sqrt{\sum_{i,j} J_{ij}}}$.

The tSB shows a similar tendency in energy convergence as dSB due to their similar mechanism. At the beginning of the search, the cut value obtained by using tSB increases more quickly than using dSB when $T_s = 1000$. Near the end of the search, the tSB has stronger fluctuations than dSB when $T_s = 100$, which indicates that the tSB is more likely to find a better solution. The uSB with $n = 3, 4$ and the lSB with $n = 3$ lead to a relatively slow convergence at the beginning due to the lack of ability to recognize the small position values (quantized to zeros) when computing $\boldsymbol{P}$. Benefiting from the wide range of numerical representation, the lSB with $n \geq 4$ increases the cut value quickly at the beginning, whereas the uSB requires $n \geq 5$ for a quick convergence. The uSB with $n = 5$ performs similarly to bSB. Although bSB sharply increases the cut value at the beginning, it cannot continuously decrease the energy with the time step, especially for a large number of iterations. However, in the uSB, the randomness introduced by quantization makes the system transverse more local minima. Note that the energy still shows a decreasing tendency even near the end of the search. The lSB with $n \geq 4$ shows a better convergence, compared with other SB algorithms. It decreases the energy in a relatively short time at the beginning of a search and maintains beneficial slight fluctuations at the end.

B. Solution Quality of using tSB, uSB and lSB

TABLE I: The Values of P_g and S_g for the Max-cut Problems on 2000-node Fully and Sparsely Connected Graphs

Values of P_g and S_g with T_s		Gset					K_{2000}				
		bSB	**dSB**	**tSB1**	**uSB5**	**lSB4**	**bSB**	**dSB**	**tSB1**	**uSB5**	**lSB4**
$T_s = 100$	$P_{99\%}$	29%	2%	5%	21%	10%	0	0	3%	12%	6%
	$S_{99\%}$	1344	22k	8978	1953	4370	-	-	15k	3602	7442
$T_s = 1000$	$P_{99.5\%}$	85%	72%	77%	85%	84%	75%	0	55%	83%	70%
	$S_{99.5\%}$	2427	3617	3133	2427	2512	3321	-	5767	2598	3824
$T_s = 10000$	$P_{99.9\%}$	0	58%	78%	59%	61%	0	0	15%	4%	12%
	$S_{99.9\%}$	-	53k	30k	51k	48k	-	-	283k	1128k	360k

We consider the metrics of probability-to-target (P_g) and step-to-target (S_g) for the evaluation of SB algorithms [32]. P_g is computed by dividing the number of trials that are able to obtain the target solution by the total number of trials, where g indicates the quality of the target solution. For example, $P_{99.9\%}$ gives the probability of obtained solutions reaching 99.9% of the best-known value. S_g estimates the number of time steps required to find the target solution with a probability of 99%, given by $S_g = T_s \cdot \frac{log 0.01}{log(1-P_g)}$, where T_s is the total number of time steps.

Table I compares qSB, including tSB1 (tSB with (9) as the threshold for ternarization), uSB5 (uSB with $n = 5$), and lSB4 (lSB with $n = 4$), with the bSB and dSB in terms of P_g and S_g on MCPs with 2000-node graphs, K_{2000} and five Gset datasets from the Gset benchmark [33] in 10^3 trials. For the Gset benchmarks, bSB can quickly find a good solution, and dSB is better suited for reaching a high solution quality by a long search. Compared with dSB, the proposed qSB algorithms perform better for both long and short searches in most cases. Moreover, they also perform similarly as bSB for a short search; however, for sparsely connected problems, bSB excels at achieving high-quality results within a very short time. Due to the massive node connectivity in the K_{2000}, SB requires a larger number of time steps to obtain a good solution. For $T_s = 10000$, it is difficult for bSB and dSB to reach 99.9% of the best-known cut value, whereas the proposed qSB can find a better solution due to their ability to jump out of the local minima. Note that for

TABLE II: Summary of Simulated Bifurcation Algorithms

SB Algorithms	Complexity of P				Solution Search	
	Compression		Operation		Time	Structure
bSB [9]	N/A		MUL	H	SS	SM
dSB [9]	N/A		SIN	L	LS	MM
tSB	DYN	H	SIN	L	SS, LS	SM, MM
uSB	STA	M	MUL	M	SS, LS	SM, MM
lSB	STA	L	SHIF	L	SS, LS	SM, MM

Compression: $\boldsymbol{x}$ is dynamically (DYN) or statically (STA) compressed. Operation: $J_{ij}x_j$ in $\boldsymbol{P}$ is implemented by multiplication (MUL), sign conversion (SIN), or shift (SHIF) operations. "H", "M", and "L" indicate that the compression degree or the complexity of the operation is high, moderate, or low. Time: the algorithm is suitable for a short search (SS) or a long search (LS). Structure: the algorithm is suitable for a single SB machine (SM) or multiple SB machines (MM) for choosing the best solution.

$T_s = 10000$, the average solution quality of dSB is superior to that of bSB. It indicates that the proposed qSB can obtain 99.9% of the best-known cut value with fewer time steps by up to an order of magnitude.

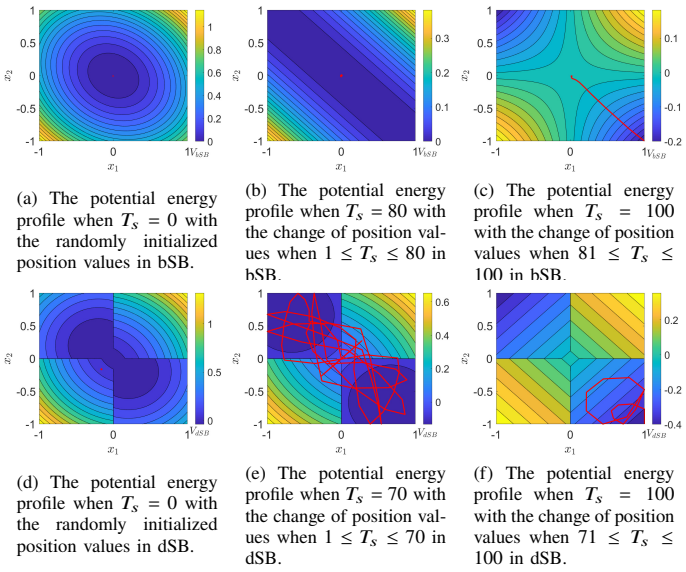
Table II summarizes the characteristics of different SB algorithms. The proposed qSB can compress the data by quantizing a group of position values to zeros. The position values are highly compressed in the tSB, but the dynamic threshold will incur additional overhead in the implementation. The tSB, dSB, and lSB algorithms significantly simplify the multiplication in $\boldsymbol{P}$. Compared with bSB, the design of the multipliers can be customized in uSB for the quantized position values to improve hardware efficiency. Moreover, the proposed qSB can obtain a good solution in a short search, and also jump out of the local minima for energy convergence in a long search.

C. Dynamics Analysis

To analyze the dynamics of various SB systems, we consider solving a two-spin Ising problem with coupling coefficients $J_{12} = J_{21} = -1$. We first analyze the dynamics of dSB and bSB systems and then show the dynamics of tSB1, uSB5, and lSB4 with 100 time steps.

Figs. 3(a)-(c) and (d)-(f) present the potential energy profile when using bSB and dSB systems, respectively. Note that the evolution is divided into initial, intermediate and final stages. This is determined by the bifurcation time point, i.e., $T_s = 80$ and $T_s = 70$, respectively, for bSB and dSB. When $T_s = 0$, the SB system stays at the initial stage; when T_s is smaller than the bifurcation time point, the SB system stays at the intermediate stage; and when T_s is greater than the bifurcation time point, the SB system stays at the final stage.

The position values are randomly initialized as zeros at the beginning, which are around the origin (the center of the 2D potential energy profile), as shown in Figs. 3(a) and (d). Then, at the intermediate stage for bSB presented in Fig. 3(b), indicated by the dark blue area, the potential energy becomes lower when one position value is negative and another one is positive. Moreover, the position values, indicated by the red line that is hard to see, still stay around the origin. At the end, indicated by Fig. 3(c), two local minima of the potential energy appear when one position value is around -1 and the other position value is around $+1$. The position values quickly jump from the origin to the boundaries $+1$ or -1, resulting in a fast energy convergence. For dSB, the position values frequently change



(a) The potential energy profile when $T_s = 0$ with the randomly initialized position values in bSB.

(b) The potential energy profile when $T_s = 80$ with the change of position values when $1 \le T_s \le 80$ in bSB.

(c) The potential energy profile when $T_s = 100$ with the change of position values when $81 \le T_s \le 100$ in bSB.

(d) The potential energy profile when $T_s = 0$ with the randomly initialized position values in dSB.

(e) The potential energy profile when $T_s = 70$ with the change of position values when $1 \le T_s \le 70$ in dSB.

(f) The potential energy profile when $T_s = 100$ with the change of position values when $71 \le T_s \le 100$ in dSB.

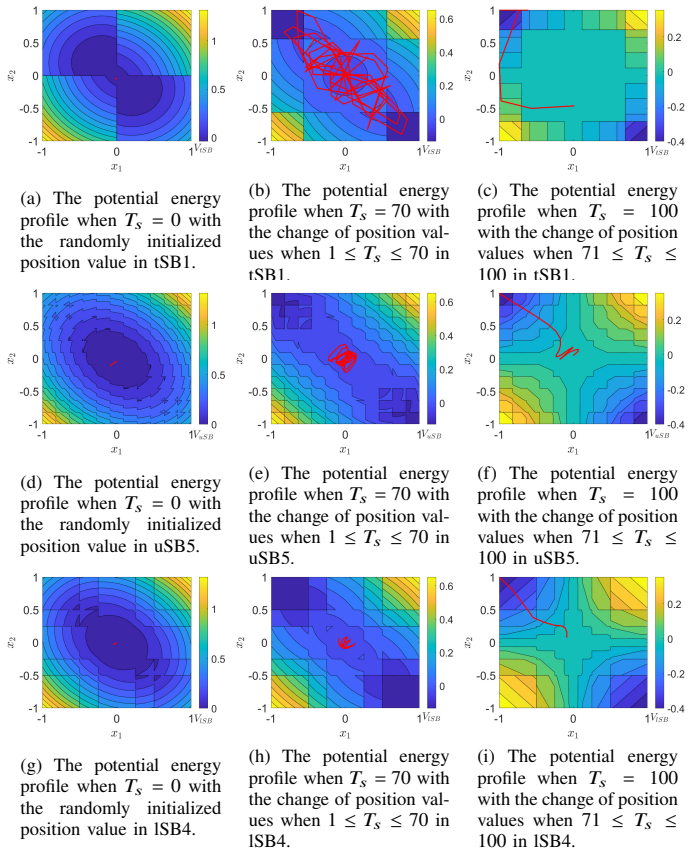
Fig. 3: The 2D potential energy profile for solving a two-spin Ising problem by using dSB and bSB systems. The change of position values during each stage with time is indicated by the red line.

between two local minima at the intermediate stage, as shown in Fig. 3(e). The position values exhibit oscillatory evolution towards one of two local minima at the final stage, as shown in Fig. 3(f).

As shown in Fig. 4(a), the potential energy profile of using tSB at the initial stage is similar to that of using dSB as presented in Fig. 3(d). At the intermediate stage in Fig. 4(b), the areas darker than the surrounding regions indicate the presence of three local minima in the potential energy, with one located at the origin and the other two at the upper left and lower right corners. The position values change surrounding these three local minima, as shown in Fig. 4(b). As presented in Fig. 4(c), at the final stage, there are two local minima that show one of the two position values being negative and the other being positive. When the position values are both positive or negative, the potential energy is higher, as seen in the upper right and lower left corners. Note that due to the quantization of the small position values to 0, the potential energy near the origin is the same. The position values after bifurcation oscillates to one of the two local minima. As shown in Figs. 4(d)-(i), multiple-value qSB systems, including the uSB and lSB systems, behave like the bSB system (as shown in Fig. 3(a)-(c)). Due to the quantization, the potential energy profile exhibits a staircase pattern. The position values evolve from the origin, as shown as the red dots that are hard to see in Figs. 4(d) and (g), then circulate around the origin, as shown in Figs. 4(e) and (h), and finally approaching one of the two local minima, as shown in Figs. 4(f) and (i).

V. Designs of Quantized Simulated Bifurcation Machines

This section presents a general architecture for the QSBM based on the tSB1, uSB5 and lSB4 algorithms. The implementations for ternary QSBM (TSBM), uniform QSBM (USBM),



(a) The potential energy profile when $T_s = 0$ with the randomly initialized position value in tSB1.

(b) The potential energy profile when $T_s = 70$ with the change of position values when $1 \le T_s \le 70$ in tSB1.

(c) The potential energy profile when $T_s = 100$ with the change of position values when $71 \le T_s \le 100$ in tSB1.

(d) The potential energy profile when $T_s = 0$ with the randomly initialized position value in uSB5.

(e) The potential energy profile when $T_s = 70$ with the change of position values when $1 \le T_s \le 70$ in uSB5.

(f) The potential energy profile when $T_s = 100$ with the change of position values when $71 \le T_s \le 100$ in uSB5.

(g) The potential energy profile when $T_s = 0$ with the randomly initialized position value in lSB4.

(h) The potential energy profile when $T_s = 70$ with the change of position values when $1 \le T_s \le 70$ in lSB4.

(i) The potential energy profile when $T_s = 100$ with the change of position values when $71 \le T_s \le 100$ in lSB4.

Fig. 4: The 2D potential energy profile for solving a two-spin Ising problem by using qSB systems.

and logarithmic QSBM (LSBM) share a similar structure, but differ in the quantization and multiplication units depending on the quantization schemes.

A. The Overall Architecture

Fig. 5 shows the general architecture of an N-spin QSBM. Note that N spins are grouped into P_b sets, where each set has N_b ($= \frac{N}{P_b}$) spins. There are P_b spin units in the QSBM, which are processed in parallel. Each spin unit is responsible for updating the position and momentum values for N_b oscillators. The QSBM consists of a system controller, an evolution controller, a memory block for the quantized position values (QX), and P_b spin units. These units are introduced as follows:

- Evolution controller: It computes the time-changing parameter $-(a_0 - a(t))$ used for computing $\dot{y}_i$ in (5) for all spin units. Additionally, for the TSBM, it also generates the threshold Δ in (9) for ternarization. Since these two parameters change linearly with the time step, an accumulator is used to compute $a(t)$, which increases from 0 to 1. The value in the register increases by A in each time step. Then, an adder is used for generating a (that is equivalent to $-(a_0 - a(t))$) in the linear parameter generator (LPG). A multiplier is used to multiply $a(t)$ by TH (that is approximately equivalent to 0.7) to generate Δ in the threshold generator (TG).
- QX: It uses a double-buffer structure, as in [34]. It broadcasts N quantized position values from one of the two buffers to all the spin units for computing $\boldsymbol{P}$ and writes the new quantized position values from each spin unit to another buffer.
- Spin unit: Each spin unit consists of a memory for the coupling coefficients for N_b spins ($\boldsymbol{JB}$), a memory for N_b time-evolved position values ($\boldsymbol{X}$), a memory for N_b time-evolved momentum values ($\boldsymbol{Y}$), a matrix-vector multiplication unit (MM), a momentum update unit (MUU), a position update unit (PUU), a boundary unit (BU), and a quantization unit (QUA). Note that for the ith spin unit, $\boldsymbol{JB}$, $\boldsymbol{X}$ and $\boldsymbol{Y}$ are, respectively, denoted with the corresponding subscripts.

B. The Spin Unit Design

1) The Quantization Unit (QUA)

We assume the threshold, position, and momentum values are represented by p bits with one sign bit and $p-1$ fractional bits. In what follows, the QUA designs in the TSBM, USBM, and LSBM are introduced.

For the tSB1, the positions are ternarized to -1, $+1$, or 0 depending on the threshold Δ as in (8). As shown in Fig. 6, it is implemented by using a comparator, CMP. When quantizing the jth oscillator position value x_j, the CMP outputs a two-bit signal qx_j to indicate the ternarized position value, which is "10", "01", or "00" when $x_j < -\Delta$, $x_j > \Delta$, or $|x_j| \le \Delta$, respectively.

The uSB5 quantizes the position value x_j into one in $\{0, \pm\frac{2}{11}, \pm\frac{4}{11}, \pm\frac{6}{11}, \pm\frac{8}{11}, \pm 1\}$. The uniform quantization is implemented in the following three steps: Step (1) to obtain the absolute position value of x_j, denoted by $\hat{x}_j$ using $p-1$ fractional bits; Step (2) to determine the magnitude of the quantized value; and Step (3) to identify whether or not the value of x_j is negative. Following these steps, x_j is encoded as a four-bit signal, $qx_j[3:0]$. The less significant three bits in $qx_j[3:0]$, i,e., $qx_j[2:0]$, are used to encode $\hat{x}_j$ for Step (2) to indicate quantizing $\hat{x}_j$ to 0, $\frac{2}{11}$, $\frac{4}{11}$, $\frac{6}{11}$, $\frac{8}{11}$, or 1. $qx_j[3]$ is used to detect whether x_j is a negative value for Step (3).

As shown in Table III, to save hardware, we first approximate the binary representation of $\frac{k}{11}$ (where $1 \le k \le 9$, $k \in \mathbb{N}$). It is rounded to the nearest value r_k that can be represented by four fractional bits, as

$$r_k = \frac{1}{16}\lfloor \frac{16k}{11} + \frac{1}{2} \rfloor \approx \frac{k}{11}, \tag{18}$$

where we first scale the value of $\frac{k}{11}$ by 16 to map the fractional part of $\frac{k}{11}$ to four fractional bits; then, we add $\frac{1}{2}$ to ensure that the result is rounded to the nearest integer by using the floor operation; finally, the integer is divided by 16 to obtain the approximate representation r_k for $\frac{k}{11}$.

Based on this approximation, the conditions and operations are approximated by using those inside the brackets in Table III.

Then, the Karnaugh map (K-map) in Table IV is used to simplify expressions for $qx_j[2:0]$. Note that the minterms for $qx_j[2]$, $qx_j[1]$, and $qx_j[0]$ are circled by using the cyan dash-dotted line, the red solid line, and the blue dash line, respectively.

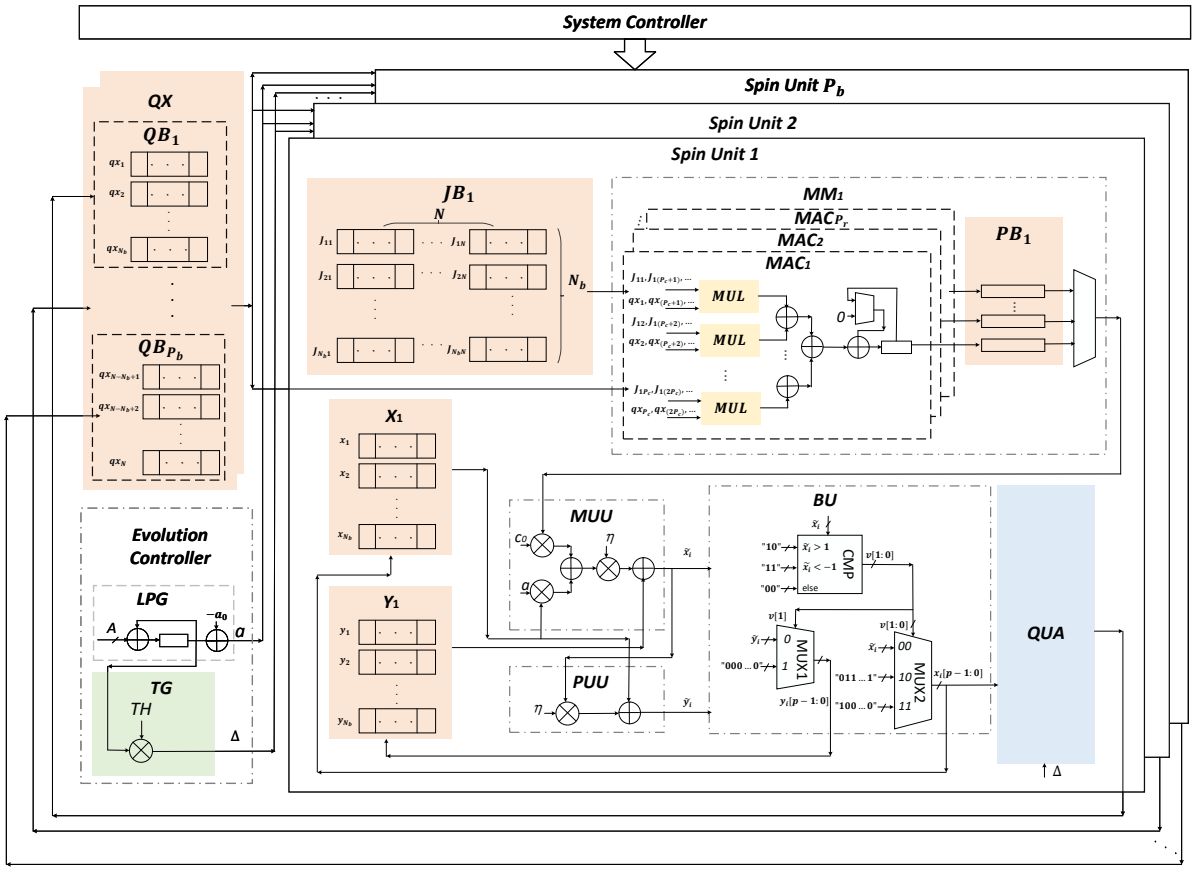


Fig. 5: A general architecture of an N-spin QSBM.

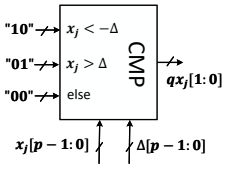


Fig. 6: The circuit diagram of QUA for tSB1.

TABLE III: Encoding x_j in the QUA for uSB5

Conditions	**Operations**	$qx_j[2:0]$
$\hat{x}_j < \frac{1}{11}$ (ap. $\hat{x}_j \leq \frac{1}{16}$)	Quantized to 0	"100"
$\frac{1}{11} \leq \hat{x}_j < \frac{3}{11}$ (ap. $\frac{1}{16} < \hat{x}_j < \frac{4}{16}$)	Quantized to $\frac{2}{11}$ (ap. $\frac{3}{16}$)	"000"
$\frac{3}{11} \leq \hat{x}_j < \frac{5}{11}$ (ap. $\frac{4}{16} \leq \hat{x}_j \leq \frac{7}{16}$)	Quantized to $\frac{4}{11}$ (ap. $\frac{6}{16}$)	"001"
$\frac{5}{11} \leq \hat{x}_j < \frac{7}{11}$ (ap. $\frac{7}{16} < \hat{x}_j \leq \frac{10}{16}$)	Quantized to $\frac{6}{11}$ (ap. $\frac{9}{16}$)	"010"
$\frac{7}{11} \leq \hat{x}_j < \frac{9}{11}$ (ap. $\frac{10}{16} < \hat{x}_j \leq \frac{13}{16}$)	Quantized to $\frac{8}{11}$ (ap. $\frac{12}{16}$)	"011"
$\hat{x}_j \geq \frac{9}{11}$ (ap. $\hat{x}_j > \frac{13}{16}$)	Quantized to +1	"111"
Conditions	**Operations**	$qx_j[3]$
$x_j[p-1] = 1$	Multiplied with -1	"1"
Else	Do nothing	"0"

ap.=" approximated by".

TABLE IV: Karnaugh map for $qx_j[2:0]$ for uSB5

$\hat{x}_j[p-4]\hat{x}_j[p-5]$ \ $\hat{x}_j[p-2]\hat{x}_j[p-3]$	00	01	11	10
00	100	001	011	010
01	100	001	011	010
11	000	001	111	011
10	000	001	111	010

The simplified expressions of $qx_j[2]$, $qx_j[1]$, and $qx_j[0]$ are given by

$$\begin{aligned} qx_j[2] &= \overline{\hat{x}_j[p-2]} \cdot \overline{\hat{x}_j[p-3]} \cdot \overline{\hat{x}_j[p-4]} \\ &\quad +\hat{x}_j[p-2] \cdot \hat{x}_j[p-3] \cdot \hat{x}_j[p-4] \\ &= \overline{\hat{x}_j[p-2] + \hat{x}_j[p-3] + \hat{x}_j[p-4]} \\ &\quad +\hat{x}_j[p-2] \cdot \hat{x}_j[p-3] \cdot \hat{x}_j[p-4], \end{aligned} \tag{19}$$

$$qx_j[1] = \hat{x}_j[p-2], \tag{20}$$

$$\begin{aligned} qx_j[0] &= \hat{x}_j[p-2] \cdot \hat{x}_j[p-4] \cdot \hat{x}_j[p-5] \\ &\quad +\hat{x}_j[p-3], \end{aligned} \tag{21}$$

where "$\cdot$" indicates a bit-wise AND operation and "+" indicates a bit-wise OR operation.

As in Table III, $qx_j[3] = 1$ if and only if $x_j[p-1] = 1$. Thus, the expression for $qx_j[3]$ is straightforward to obtain, as

$$qx_j[3] = x_j[p-1]. \tag{22}$$

Fig. 7 gives the circuit diagram of the QUA for uSB5. The S1 unit implements Step (1) with the output $\hat{x}_j[p-2,0]$. According to (19)-(21), the S2 unit is used to generate the encoded signal $qx_j[2:0]$ by AND, OR and NOT operations for Step (2). The S3 implements Step (3).

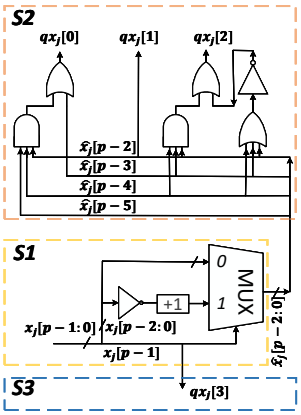


Fig. 7: The circuit diagram of the QUA for uSB5.

The lSB4 algorithm quantizes the position values into $\pm\frac{1}{8}$, $\pm\frac{1}{4}$, $\pm\frac{1}{2}$, ±1, or 0. Similar to uniform quantization, logarithmic quantization is implemented in the following three steps: Step (1) to obtain the absolute position value of x_j, as $\hat{x}_j[p-2:0]$ with $p-1$ fractional bits; Step (2) to determine how many bits to be right shifted; and Step (3) to identify whether the value of x_j is negative and whether to quantize x_j to 0. Similar to uSB5, the position values are encoded as $qx_j[3:0]$. As shown in Table V, $qx_j[1:0]$ is used to encode $\hat{x}_j$ to indicate right shifting 0, 1, 2, or 3 bits for Step (2). Four conditions can be identified by using the first four fractional bits of $\hat{x}_j$. Three cases are considered and identified for $qx_j[3:2]$, as shown in Table V. When $\hat{x}_j < \frac{1}{16}$ (not considered for $qx_j[1:0]$), $qx_j[3:2]$ is encoded to "10" to indicate x_j being quantized to 0. When $\hat{x}_j \geq \frac{1}{16}$, if x_j is a negative value, $qx_j[3:2]$ is encoded to "01"; otherwise, it is encoded to "00".

TABLE V: Encoding x_j in the QUA for lSB4

Conditions	**Operations**	$qx_j[1:0]$
$\frac{1}{16} \leq \hat{x}_j < \frac{1}{8}$	Right shift 3 bits	"11"
$\frac{1}{8} \leq \hat{x}_j < \frac{1}{4}$	Right shift 2 bits	"10"
$\frac{1}{4} \leq \hat{x}_j < \frac{1}{2}$	Right shift 1 bits	"01"
$\hat{x}_j \geq \frac{1}{2}$	Do nothing	"00"
Conditions	**Operations**	$qx_j[3:2]$
$\hat{x}_j < \frac{1}{16}$	Quantized to 0	"10"
$\hat{x}_j \geq \frac{1}{16}$ and $x_j[p-1]=1$	Multiplied with -1	"01"
Else	Do nothing	"00"

Table VI gives the Karnaugh map for $qx_j[1:0]$. Note that when $\hat{x}_j < \frac{1}{16}$, i.e., $\hat{x}_j[p-2:p-5]$ = "0000", $qx_j[1:0]$ could be any value in this encoding stage. Note that the minterms for $qx_j[1]$ and $qx_j[0]$ are circled by using the red solid line and the blue dash line, respectively. Then $qx_j[1]$ and $qx_j[0]$ are

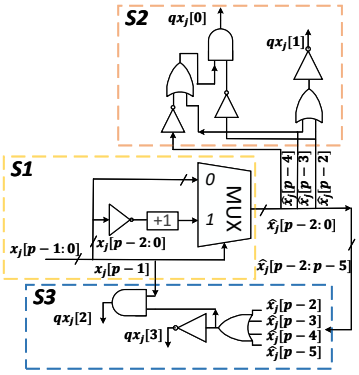


Fig. 8: The circuit diagram of the QUA for lSB4.

expressed as

$$qx_j[1] = \overline{\hat{x}_j[p-2]} \cdot \overline{\hat{x}_j[p-3]} = \overline{\hat{x}_j[p-2] + \hat{x}_j[p-3]}, \tag{23}$$

$$qx_j[0] = \overline{\hat{x}_j[p-2]} \cdot \hat{x}_j[p-3] + \overline{\hat{x}_j[p-2]} \cdot \overline{\hat{x}_j[p-4]} \tag{24}$$
$$= \overline{\hat{x}_j[p-2]} \cdot (\hat{x}_j[p-3] + \overline{\hat{x}_j[p-4]}).$$

TABLE VI: Karnaugh map for $qx_j[1:0]$ for lSB4

$\hat{x}_j[p-4]\hat{x}_j[p-5]$ \ $\hat{x}_j[p-2]\hat{x}_j[p-3]$	00	01	11	10
00	XX	01	00	00
01	11	01	00	00
11	10	01	00	00
10	10	01	00	00

Notes: "XX" indicates a don't-care condition.

In the encoding stage in Step (3), $qx_j[3]$ ="1" when $\hat{x}_j[p-2:p-5]$ ="0000", and $qx_j[2]$ ="1" when $\hat{x}_j[p-2:p-5] \neq$ "0000" and $x_j[p-1]$ = "1". Thus, $qx_j[3]$ and $qx_j[2]$ are expressed as

$$qx_j[3] = \overline{\hat{x}_j[p-2] + \hat{x}_j[p-3] + \hat{x}_j[p-4] + \hat{x}_j[p-5]}, \tag{25}$$

$$qx_j[2] = x_j[p-1] \cdot (\hat{x}_j[p-2] + \hat{x}_j[p-3] + \hat{x}_j[p-4] + \hat{x}_j[p-5]). \tag{26}$$

Similarly, the S1 unit in Fig. 8 is used to obtain $\hat{x}_j$ at Step (1). The implementations for encoding x_j to $qx_j[1:0]$ and $qx_j[3:2]$ can be found in the S2 and S3 units, respectively. Three inverters, two OR gates, and an AND gate are used to generate $qx_j[1:0]$. A four-input OR gate, an inverter, and an AND gate are used to generate $qx_j[3:2]$.

2) The Matrix-vector Multiplication Unit (MM)

We adopt the parallelism strategy in [34] for implementing the MM for hardware efficiency. Let $\boldsymbol{QX}$ be a vector of ternary, uniformly and logarithmic quantized position values. Fig. 9 shows the parallelization in the matrix-vector multiplication $\boldsymbol{P} = \boldsymbol{J} \cdot \boldsymbol{QX}$. For an N-spin QSSM, the elements of the coupling coefficient matrix $\boldsymbol{J}$ with N rows and N columns, are split into P_b sub-matrices $\boldsymbol{JB}$. The ith $\boldsymbol{JB}$, denoted by $\boldsymbol{JB_i}$, has the

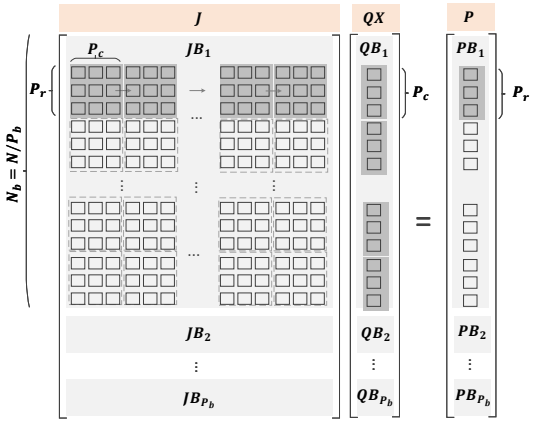


Fig. 9: Parallelization in the MAC operation $\boldsymbol{P} = \boldsymbol{J} \cdot \boldsymbol{QX}$ [34].

$(iN_b + 1)$th to $(iN_b + N_b)$th rows of $\boldsymbol{J}$. To process $\boldsymbol{JB_i}$, two other parameters are introduced for parallelism, one for the row parallelism, P_r, and the other for the column parallelism, P_c. $\boldsymbol{JB_i}$ with $N_b \times N$ values is divided into several $P_r \times P_c$-sized blocks. For each $\boldsymbol{JB_i}$, a $P_r \times P_c$ coupling coefficient matrix is multiplied with a quantized position vector of P_c values in $\boldsymbol{QX}$ in each cycle to obtain the P_r MAC results, where each multiplication is processed in parallel. To get the result of the $P_r \times N$ coupling coefficient matrix multiplied with a quantized position vector of N values, which is a vector of P_r values in $\boldsymbol{P}$, $\frac{N}{P_c}$ cycles are required. We need to accumulate $\frac{N}{P_c}$ MAC results from $\frac{N}{P_c}$ MAC operations between $\frac{N}{P_c} \times P_r \times P_c$ coupling coefficient matrices and $\frac{N}{P_c}$ quantized position vectors of P_c values. These operations are repeated by $\frac{N_b}{P_r}$ times in each spin unit for computing N_b $\boldsymbol{PB}$ values.

The MM in the ith spin unit, namely MM_i, implements $\boldsymbol{JB_i} \cdot \boldsymbol{QX} = \boldsymbol{PB_i}$. Given the MM in spin unit 1 (MM_1) in Fig. 5 as an example, the MM_1 consists of P_r MAC units, a memory block for the product PB_1, and a multiplexer. Each MAC has P_c multiplier units (MULs), an addition unit, and an accumulator. The addition unit adds the P_c multiplication results. The addition results are accumulated for $\frac{N}{P_c}$ times to obtain the product of a row of $\boldsymbol{J}$ and $\boldsymbol{QX}$. The products from the P_r MAC units are stored in the memory block PB_1. Each value in PB_1 is selected concurrently for the MUU unit. The MUL design depends on the specific quantization scheme. It takes the encoded quantized position value for the jth oscillator, qx_j, and the coupling coefficient J_{ij} as inputs to compute P_{ij}. Note that J_{ij} and P_{ij} are represented by q bits with one sign bit and $q-1$ fractional bits. qx_j has two bits for tSB1, whereas it has 4 bits for uSB5 and lSB4.

For tSB1, taking the $qx_j[1:0]$ signal as the selection signal, the multiplication of $J_{ij}tri(x_j)$ in (7) is implemented by using a multiplexer, MUX, as shown in Fig. 10. It outputs the product, denoted by P_{ij}, as 0 when qx_j ="00", as $-J_{ij}$ when qx_j ="10", and as J_{ij} when qx_j ="01".

As shown in Table III, the position values are approximately quantized to $\frac{3}{16}$, $\frac{6}{16}$, $\frac{9}{16}$, or $\frac{12}{16}$. This value can be formulated as $\frac{3k}{16}$, where $1 \le k \le 4$, $k \in \mathbb{N}$. To reduce hardware, as shown in the table in Fig. 11, we first represent approximate quantized

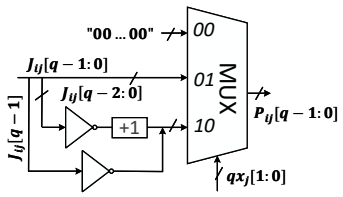


Fig. 10: The circuit diagram of the MUL for tSB1.

values $\frac{3k}{16}$ as the addition of two different powers of 2, as

$$\frac{3k}{16} = 2^a + 2^b, \tag{27}$$

where

$$a = \begin{cases} -3 & k = 1,2 \\ -1 & others \end{cases}, \quad b = \begin{cases} -4 & k = 1,3 \\ -2 & others \end{cases}. \tag{28}$$

Considering that multiplication with a power of 2 can be implemented by simple shifting, the multiplication between J_{ij} and one of $\frac{3}{16}$, $\frac{6}{16}$, $\frac{9}{16}$, or $\frac{12}{16}$ is implemented by adding two shifted J_{ij} values, as

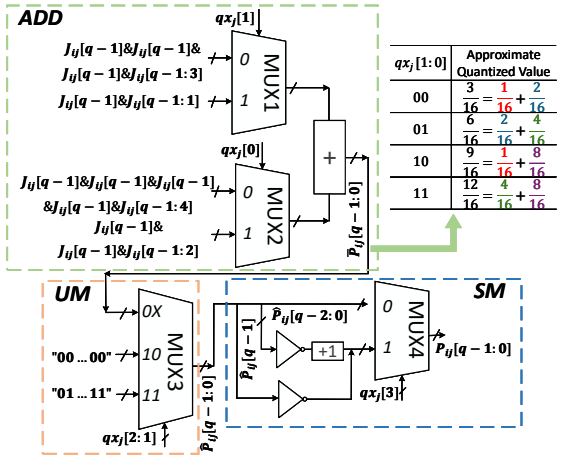
$$J_{ij} \cdot \frac{3k}{16} = 2^a J_{ij} + 2^b J_{ij}. \tag{29}$$

Moreover, when $k = 1, 2$, $qx_j[1]$ = "0", otherwise $qx_j[1]$ = "1"; when $k = 1, 3$, $qx_j[0]$ = "0", otherwise $qx_j[0]$ = "1". Thus, we use two multiplexers, MUX1 and MUX2, given $qx_j[1]$ and $qx_j[0]$ as the selection signals, respectively, to choose two input operands for the adder. Thus when $qx_j[1]$ = "1", $\frac{J_{ij}}{2}$, in the form of "$J_{ij}[q-1]\&J_{ij}[q-1:1]$", is chosen, otherwise $\frac{J_{ij}}{8}$, in the form of "$J_{ij}[q-1]\&J_{ij}[q-1]\&J_{ij}[q-1]\&J_{ij}[q-1:3]$", is output from MUX1. When $qx_j[0]$ = "0", $\frac{J_{ij}}{16}$, which is represented by "$J_{ij}[q-1]$ & $J_{ij}[q-1]\&J_{ij}[q-1]\&J_{ij}[q-1]\&J_{ij}[q-1:4]$", is chosen, otherwise $\frac{J_{ij}}{4}$, which is represented by "$J_{ij}[q-1]\&J_{ij}[q-1]\&J_{ij}[q-1:2]$", is output from MUX2. Note that & is a concatenation operator. In this way, we have two q-bit outputs from MUX1 and MUX2 in the addition unit (ADD). Then, a q-bit adder is used to add them. Subsequently, given the addition result denoted by $\bar{P}_{ij}$, MUX3 chooses one from $\{0, 1, \bar{P}_{ij}\}$ as the output $\hat{P}_{ij}$ in the unsigned multiplication unit (UM). Note that due to the limited numerical range provided by p bits with one sign bit and $p-1$ fractional bits, the binary representation of +1 is approximated by its nearest value "$01\cdots11$" to meet the precision limitation. Finally, the MUX4 is used to generate the product $\hat{P}_{ij}$ by determining whether to multiply $\hat{P}_{ij}$ with -1 according to $qx_j[3]$ in the signed multiplication unit (SM).

Fig. 12 gives the circuit design for implementing $J_{ij}lg(x_j)$ in (14). It is composed of two multiplexers, MUX1 and MUX2, respectively, given $qx_j[1:0]$ and $qx_j[3:2]$ as the selection signals. MUX1 chooses one from $\{J_{ij}, \frac{J_{ij}}{2}, \frac{J_{ij}}{4}, \frac{J_{ij}}{8}\}$ as the output $\hat{P}_{ij}$ by $qx_j[1:0]$ in the UM. Subsequently, the MUX2 outputs P_{ij}, which is selected from $\{0, \hat{P}_{ij}, -\hat{P}_{ij}\}$ by $qx_j[3:2]$ in the SM.

3) The Momentum Update Unit (MUU) and the Position Update Unit (PUU)

As shown in Fig. 5, the MUU is implemented by using three multipliers and two adders. For updating the ith oscillator



$qx_j[1:0]$	Approximate Quantized Value
00	$\frac{3}{16} = \frac{1}{16} + \frac{2}{16}$
01	$\frac{6}{16} = \frac{2}{16} + \frac{4}{16}$
10	$\frac{9}{16} = \frac{1}{16} + \frac{8}{16}$
11	$\frac{12}{16} = \frac{4}{16} + \frac{8}{16}$

Fig. 11: The circuit diagram of the MUL for uSB5.

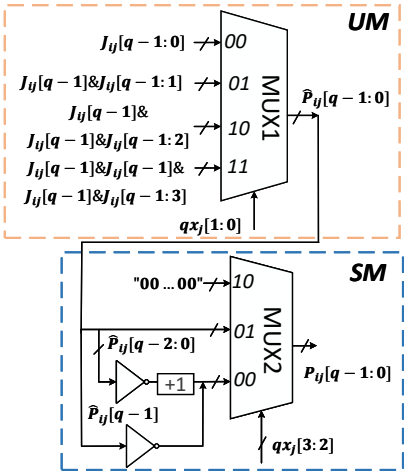


Fig. 12: The circuit diagram of the MUL for lSB4.

momentum value, the corresponding P_i is first selected from PB_1 and then multiplied with the manually determined c_0. The product is added with the ith oscillator momentum value multiplied with the linearly increasing value a to compute the derivative of y_i with respect to time. Then, the derivative is multiplied with a time step for integration η and added with the current y_i value to compute the new y_i value ($\tilde{y}_i$).

The ith oscillator position value x_i is updated by using a multiplier and an adder before obtaining the new momentum value y_i in the PUU, as shown in Fig. 5. Then, the updated position value y_i is multiplied with the time step for integration η and added to the current position value x_i.

4) The Boundary Unit (BU)

The BU detects whether the new position value x_i ($\tilde{x}_i$) exceeds the boundary $[-1, 1]$. As shown in Fig. 5, it consists of a comparator and two multiplexers. The comparator outputs a two-bit signal $v[1:0]$, which is "10" if $\tilde{x}_i > 1$, "11" if $\tilde{x}_i < -1$, or "00" otherwise. Given $v[1]$ as the selection signal, MUX1 outputs y_i when $v[1]$ = "0" or 0 otherwise. Given $v[1:0]$ as the selection signal, MUX2 outputs x_i when $v[1:0]$ = "00", 1 when $v[1:0]$ = "10" or -1 otherwise. Note that the binary representation of 1 is approximated by "$011\cdots1$" to meet the precision requirements for the following QUA unit.

VI. Performance Evaluation

The QSBMs were designed using VHDL, synthesized with the Vivado design suite, and implemented on a Xilinx Virtex UltraScale+ FPGA, featuring 2048 fully connected spins. We set $P_r = P_c = 32$, and $P_b = 8$, $p = 8$, and $q = 8$. Note that the parameters for parallel computation, P_r, P_c, P_b, can be adjusted based on the hardware capabilities and time constraints of the specific application scenarios. The precision parameters, p and q, are determined by the required solution quality, the size of the problem, and the variations in the coupling coefficients. Other computation in the QSBM uses 16-bit precision with 4 integer bits, 11 fractional bits and one sign bit. The coupling coefficients are stored in the Block Random-Access Memories (BRAMs) and other data are stored in the registers.

Table VII presents a comparison of QSBM designs against existing application specific integrated circuits (ASICs), FPGA, optical circuits, quantum circuits, and graphics processing unit (GPU)-based hardware accelerators. The FPGA-based hardware accelerators are evaluated by their resource utilization in lookup tables (LUTs), flip flops (FFs), digital signal processors (DSPs), and memory. The quantum Ising machine implements the spin by qubits and the coherent Ising machine uses optical pulses as spins. They are susceptible to external noise and quantum decoherence, thus leading to errors in computation. The 5000-qubit Ising machine in [3] implements a sparsely-connected topology with only fifteen interactions for each qubit. The limitation in the interaction restricts the problems it can solve. Although the $100k$-spin coherent Ising machine in [36] implements a complete topology, it exhibits limited precision in representing the coupling coefficients. Moreover, it requires 56 peripheral FPGAs, a 5-km loop of optical cable, and an extensive temperature-controlled test setup. The simulated annealing machine implemented on GPUs in [37] offers advantages in terms of scalability and flexibility. However, it faces with challenges in power and memory explosion. The fully connected CMOS annealer with 512 spins implemented by 28nm CMOS technology in [38] consumes less power, but operates only at a 1 MHz frequency, thus at a cost of performance. The annealer implemented on an FPGA with 1024 fully connected spins achieves an average accuracy of 99.1% for solving Gset benchmarks in 2.38 ms, but it consumes a relatively high power [39]. The fully connected 2048-spin SB machine implemented on an FPGA in [34] achieves a high frequency. However, it only implements 1-bit coupling coefficients and requires more hardware.

The QSBM designs implement 8-bit coupling coefficients to provide a wider numerical range for representation. Table VII evaluates the performance of QSBMs by solving ten datasets from Gset benchmarks and K2000 benchmark, which are formulated as 2000-spin Ising problems. The tSB and uSB are advantageous for relatively long and short searches, respectively. The lSB shows a general advantage, which performs better than tSB for short searches and better than uSB for long searches. Therefore, $T_s = 1000$, $T_s = 500$, and $T_s = 250$ are, respectively, considered for using the TSBM, LSBM, and USBM to solve

TABLE VII: The Circuit Measurements of Ising Machines

	[35]	[36]	[37]	[34]	[38]	[39]	TSBM	USBM	LSBM
Computing Method	Quantum Annealing	Coherent Computing	Simulated Annealing	Simulated Bifurcation	Simulated Annealing	Simulated Annealing	Simulated Bifurcation	Simulated Bifurcation	Simulated Bifurcation
Platform/Technology	Superconductor	Optics	2880 CUDA cores	Arria 10 GX 1150	28nm CMOS	Virtex UltraScale+	Virtex UltraScale+	Virtex UltraScale+	Virtex UltraScale+
# Spins	5000	$100k$	800 – 20000	2048	512	1024	2048	2048	2048
Topology	Pegasus	Complete	Complete	Complete	Complete	Complete	Complete	Complete	Complete
# Interaction per Spin	15	$99k$	7999-19999	2047	511	1023	2047	2047	2047
Coefficient Bit-Width	N/A	1	2	1	4	4	8	8	8
Power per Spin	N/A	N/A	$> 0.12\ W$	N/A	$0.02\ mW$	$3.6\ mW$	$1.7\ mW$	$2.1\ mW$	$1.9\ mW$
LUT /ALM	N/A	N/A	N/A	427k	N/A	40k	210k	318k	261k
Flip Flop	N/A	N/A	N/A	281k	N/A	12k	49k	57k	53k
DSP	N/A	N/A	N/A	104	N/A	N/A	16	16	16
Memory	N/A	N/A	72GB DRAM	2MB BRAM	N/A	4MB BRAM	4MB BRAM	4MB BRAM	4MB BRAM
Frequency	N/A	N/A	N/A	$279\ MHz$	$1\ MHz$	$100\ MHz$	$200\ MHz$	$200\ MHz$	$200\ MHz$
Average Accuracy	N/A	96.3%	98.7%	N/A	N/A	99.1%	99.4% (99.1%)	98.5% (98.2%)	99.1% (98.8%)
Time	N/A	$593\ \mu s$	100-150 ms	N/A	$128\ ms$	$2.38\ ms$	$2.92\ ms$	$0.73\ ms$	$1.46\ ms$

N/A: not reported. #: the number. Average Accuracy: The average accuracy on the MCP benchmarks. LUT: Look-up table. ALM: Adaptive logic module. DSP: Digital signal processor. For the proposed SB machines, the average accuracy on the Gset benchmarks is provided, with the accuracy on the fully connected K2000 given in brackets.

Gset benchmarks. Each time step requires around 2.9 μs. Note that the Ising machine in [39] only evaluates some Gset benchmarks formulated as 800 and 1000-spin Ising problems.

Compared to USBM, TSBM utilizes 33.9% fewer LUTs and 13.5% fewer FFs; moreover, LSBM utilizes 17.9% fewer LUTs and 7.0% fewer FFs. It can be seen that the TSBM achieves an average accuracy of 99.4% for the Gset benchmark problems in 2.92 ms. The USBM shows its efficacy in a short search, obtaining an average accuracy of 98.5% in 0.73 ms, whereas the lightweight LSBM reaches a 99.1% accuracy in 1.46 ms. For solving the fully connected K2000 problem, the solution quality has a slight decrease. Compared to the 512-spin annealer in [38], which requires 128 ms to achieve a good solution quality, the proposed QSBMs show their advantage in search time. Although it seems that the proposed 2048-spin QSBMs consume more power, scaling up the 512-spin annealer is expected to result in exponentially increased power. Compared to the 1024-spin annealing machine in [39], the 2048-spin QSBMs use more resources but consume at least 41% less power per spin. Moreover, scaling up the annealing machine will dramatically increase the hardware resources for generating random numbers. Compared with the SB machine in [34], the QSBMs save up to 84% of utilized DSPs, and use 50.8% fewer LUTs, and 82.5% fewer FFs. This is due to the efficient quantization schemes and also the reduced precision for representing position values.

The proposed SB Ising machine can be scaled to solve larger problems, at the cost of additional clock cycles for completely updating all spins once. Moreover, an increased memory capacity is required to store data for the coupling coefficients, position values, and momentum values. The number of clock cycles required to process a problem of size N is approximately $\frac{N^2}{P_c \cdot P_r \cdot P_b}$, which increases quadratically with the problem size. For instance, extending the machine to handle a 4096-spin problem would approximately quadruple the time needed to complete a full spin update compared to solving a 2048-spin problem under the same architecture. Furthermore, as the problem size increases (i.e., more spins), the system requires more iterations to converge due to the larger number of possible configurations. Moreover, by customizing the multiplication operation with simple circuits, we reduce the dependence on the limited number of DSPs available on the board. This allows the proposed quantized SB Ising machine to support a higher degree of parallelism in computation with larger values of P_b, P_r, and P_c, depending on the power, time, and resource requirements of the specific application scenario. For instance, when setting $P_r = P_c = 32$ and $P_b = 16$ on TSBM, the number of LUTs and FFs will increase by 80.9% and 23.4%, respectively, while the number of DSPs will increase to 32.

VII. Conclusion

In this paper, efficient quantized simulated bifurcation Ising machines (QSBMs) are proposed for fast and low-cost combinatorial optimization. Specifically, quantized SB (qSB) algorithms use various quantization schemes to implement multiplication in MAC using simple operators. A ternary qSB algorithm dynamically ternarizes the position values for the MAC by introducing a linearly increasing threshold. Multiple-value qSB algorithms utilize uniform and logarithmic quantization to improve the precision. The qSB algorithms realize fast energy convergence and increase the probability of jumping out of the local minima. The hardware accelerators for QSBMs using ternary, uniform, and logarithmic quantization, respectively designed and implemented on FPGAs, achieve a reduction of up to 50.8% of LUTs and 82.5% of FFs, compared to conventional FPGA-based SB machines. Finally, evaluated on 2000-spin Ising problems, the QSBM is 1.63 times faster to reach on average 99.1% of the best known solutions, compared to a recent FPGA-based annealer.

Acknowledgment

The work was supported by the University of Alberta (Project Number: RES0049590), the Natural Sciences and Engineering Research Council (NSERC) of Canada (Project Numbers: RES0048688, RES0051374 and RES0054326) and Alberta Innovates (Project Number: RES0053965).

References

[1] K. Tanahashi, S. Takayanagi, T. Motohashi, and S. Tanaka, "Application of Ising machines and a software development for Ising machines," *J. Phys. Soc. Jpn.*, vol. 88, no. 6, p. 061010, 2019.

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

[3] M. W. Johnson, M. H. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson *et al.*, "Quantum annealing with manufactured spins," *Nature*, vol. 473, no. 7346, pp. 194–198, 2011.

[4] Z. Wang, A. Marandi, K. Wen, R. L. Byer, and Y. Yamamoto, "Coherent Ising machine based on degenerate optical parametric oscillators," *Phys. Rev. A*, vol. 88, no. 6, p. 063853, 2013.

[5] J. Chou, S. Bramhavar, S. Ghosh, and W. Herzog, "Analog coupled oscillator based weighted Ising machine," *Sci. Rep.*, vol. 9, no. 1, p. 14786, 2019.

[6] B. Zhang, Z. Lin, Y. Liu, Y. Wang, D. Zhang, T. Gao, and L. Zeng, "Compact programmable true random number generator based on spin torque nano-oscillator," *IEEE Transactions on Nanotechnology*, vol. 21, pp. 648–654, 2022.

[7] R. A. Rutenbar, "Simulated annealing algorithms: An overview," *IEEE Circuits Syst. Mag.*, vol. 5, no. 1, pp. 19–26, 1989.

[8] K. Yamamoto, K. Kawamura, K. Ando, N. Mertig, T. Takemoto, M. Yamaoka *et al.*, "STATICA: A 512-spin 0.25 m-weight annealing processor with an all-spin-updates-at-once architecture for combinatorial optimization with complete spin–spin interactions," *IEEE J. Solid-State Circuits*, vol. 56, no. 1, pp. 165–178, 2020.

[9] H. Goto, K. Tatsumura, and A. R. Dixon, "Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems," *Sci. Adv.*, vol. 5, no. 4, p. eaav2372, 2019.

[10] T. Zhang, Q. Tao, B. Liu, A. Grimaldi, E. Raimondo, M. Jiménez, M. J. Avedillo, J. Nunez, B. Linares-Barranco, T. Serrano-Gotarredona *et al.*, "A review of Ising machines implemented in conventional and emerging technologies," *IEEE Trans. Nanotechnology*, vol. 23, pp. 704–717, 2024.

[11] T. Zhang, Q. Tao, B. Liu, and J. Han, "A review of simulation algorithms of classical Ising machines for combinatorial optimization," in *ISCAS*. IEEE, 2022, pp. 1877–1881.

[12] S. Tsukamoto, M. Takatsu, S. Matsubara, and H. Tamura, "An accelerator architecture for combinatorial optimization problems," *Fujitsu Sci. Tech. J*, vol. 53, no. 5, pp. 8–13, 2017.

[13] S. Matsubara, M. Takatsu, T. Miyazawa, T. Shibasaki, Y. Watanabe, K. Takemoto, and H. Tamura, "Digital annealer for high-speed solving of combinatorial optimization problems and its applications," in *ASP-DAC*. IEEE, 2020, pp. 667–672.

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

[15] Y. Kihara, M. Ito, T. Saito, M. Shiomura, S. Sakai, and J. Shirakashi, "A new computing architecture using Ising spin model implemented on FPGA for solving combinatorial optimization problems," in *IEEE NANO Conf.* IEEE, 2017, pp. 256–258.

[16] T. Miki, A. Yoshida, M. Shimada, and J. Shirakashi, "Hybridization of spin decision logics for Ising machine with logic circuits," in *IEEE NANO Conf.* IEEE, 2021, pp. 470–473.

[17] T. Okuyama, M. Hayashi, and M. Yamaoka, "An Ising computer based on simulated quantum annealing by path integral Monte Carlo method," in *ICRC*. IEEE, 2017, pp. 1–6.

[18] Q. Tao, T. Zhang, and J. Han, "An approximate parallel annealing Ising machine for solving traveling salesman problems," *IEEE Embedded Systems Letters*, vol. 15, no. 4, pp. 226–229, 2023.

[19] N. A. Aadit, M. Mohseni, and K. Y. Camsari, "Accelerating adaptive parallel tempering with FPGA-based p-bits," in *2023 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*. IEEE, 2023, pp. 1–2.

[20] H. Goto, K. Endo, M. Suzuki, Y. Sakai, T. Kanao, Y. Hamakawa, R. Hidaka, M. Yamasaki, and K. Tatsumura, "High-performance combinatorial optimization based on classical mechanics," *Sci. Adv.*, vol. 7, no. 6, p. eabe7953, 2021.

[21] T. Zhang and J. Han, "Quantized simulated bifurcation for the Ising model," in *IEEE NANO Conf.* IEEE, 2023, pp. 715–720.

[22] T. Zhang, Q. Tao, and J. Han, "Solving traveling salesman problems using Ising models with simulated bifurcation," in *ISOCC*. IEEE, 2021, pp. 288–289.

[23] T. Zhang and J. Han, "Efficient traveling salesman problem solvers using the Ising model with simulated bifurcation," in *DATE*. IEEE, 2022, pp. 548–551.

[24] E. S. Tiunov, A. E. Ulanov, and A. Lvovsky, "Annealing by simulating the coherent Ising machine," *Opt. Express*, vol. 27, no. 7, pp. 10 288–10 295, 2019.

[25] S. Sreedhara, J. Roychowdhury, J. Wabnig, and P. Srinath, "Digital emulation of oscillator Ising machines," in *DATE*. IEEE, 2023, pp. 1–2.

[26] L. Mazza, E. Raimondo, A. Grimaldi, and V. Puliafito, "Simulated oscillator-based Ising machine for two million nodes max-cut problems," in *IEEE NANO Conf.* IEEE, 2023, pp. 1037–1041.

[27] B. Liu, T. Zhang, X. Gao, and J. Han, "An efficient simulated oscillator-based Ising machine on FPGAs," in *IEEE NANO Conf.* IEEE, 2024, pp. 469–474.

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

[29] H. Goto, "Bifurcation-based adiabatic quantum computation with a nonlinear oscillator network," *Sci. Rep.*, vol. 6, no. 1, p. 21686, 2016.

[30] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," *arXiv preprint arXiv:1605.04711*, 2016.

[31] E. H. Lee, D. Miyashita, E. Chai, B. Murmann, and S. S. Wong, "Lognet: Energy-efficient neural networks using logarithmic computation," in *ICASSP*. IEEE, 2017, pp. 5900–5904.

[32] T. Kanao and H. Goto, "Simulated bifurcation assisted by thermal fluctuation," *Commun. Phys.*, vol. 5, no. 1, p. 153, 2022.

[33] C. Helmberg and F. Rendl, "A spectral bundle method for semidefinite programming," *SIAM J. Optim.*, vol. 10, no. 3, pp. 673–696, 2000.

[34] K. Tatsumura, A. R. Dixon, and H. Goto, "FPGA-based simulated bifurcation machine," in *FPL*. IEEE, 2019, pp. 59–66.

[35] A. D. King, J. Raymond, T. Lanting, R. Harris, A. Zucca *et al.*, "Quantum critical dynamics in a 5,000-qubit programmable spin glass," *Nature*, vol. 617, pp. 61–66, 2023.

[36] T. Honjo, T. Sonobe, K. Inaba, T. Inagaki, T. Ikuta, Y. Yamada, T. Kazama, K. Enbutsu, T. Umeki, R. Kasahara *et al.*, "100,000-spin coherent Ising machine," *Science advances*, vol. 7, no. 40, p. eabh0952, 2021.

[37] C. Cook, H. Zhao, T. Sato, M. Hiromoto, and S. X.-D. Tan, "GPU-based Ising computing for solving max-cut combinatorial optimization problems," *Integration*, vol. 69, pp. 335–344, 2019.

[38] R. Iimura, S. Kitamura, and T. Kawahara, "Annealing processing architecture of 28-nm CMOS chip for Ising model with 512 fully connected spins," *IEEE Trans. Circuits Syst. I: Regul. Pap.*, vol. 68, no. 12, pp. 5061–5071, 2021.

[39] Z. Huang, D. Jiang, X. Wang, and E. Yao, "An Ising model-based annealing processor with 1024 fully connected spins for combinatorial optimization problems," *IEEE Trans. Circuits Syst. II*, vol. 70, no. 8, pp. 3074–3078, 2023.

Tingting Zhang (Member, IEEE) received her B.Sc. and M.Sc. degrees from Nanjing University of Aeronautics and Astronautics (NUAA), Nanjing, China, in 2016 and 2019, respectively, and her Ph.D. degree from the University of Alberta, Alberta, Canada, in 2024. She is currently a Postdoctoral Fellow at McGill University, Quebec, Canada. Her research interests include new computing architectures, approximate computing, Ising computing, combinatorial optimization and nanoelectronic circuits and systems. She was a recipient of the Best Paper Award Candidate at the Design, Automation and Test in Europe Conference (DATE) 2022. She served as the session chair for the IEEE International Conference on Nanotechnology (IEEE-NANO) 2024 and a Technical Program Committee Member for the International Conference on Computer-Aided Design (ICCAD) 2025.

Jie Han (Senior Member, IEEE) received the B.Sc. degree in electronic engineering from Tsinghua University, Beijing, China, in 1999 and the Ph.D. degree from the Delft University of Technology, The Netherlands, in 2004. He is currently a Professor and the Director of Computer Engineering in the Department of Electrical and Computer Engineering at the University of Alberta, Edmonton, AB, Canada. His research interests include approximate computing, stochastic computing, reliability and fault tolerance, nanoelectronic circuits and systems, novel computational models for learning and biological applications. Dr. Han was a recipient of the Best Paper Awards at the International Symposium on Nanoscale Architectures (NANOARCH 2015) and the Design, Automation and Test in Europe Conference (DATE 2023), as well as several Best Paper Nominations at the 25th Great Lakes Symposium on VLSI (GLSVLSI 2015), NANOARCH 2016, the 19th International Symposium on Quality Electronic Design (ISQED 2018) and DATE 2022. He was nominated for the 2006 Christiaan Huygens Prize of Science by the Royal Dutch Academy of Science. His work was recognized by *Science*, for developing a theory of fault-tolerant nanocircuits (2005). He serves (or sersved) as an Associate Editor for the IEEE Transactions on Nanotechnology, the IEEE Transactions on Emerging Topics in Computing (TETC), the IEEE Embedded Systems Letters, the IEEE Circuits and Systems Magazine (awarded the Best Associate Editor for 2023), the IEEE Open Journal of the Computer Society, Microelectronics Reliability (Elsevier) and the Journal of Electronic Testing: Test and Application (JETTA, Springer Nature). He served as a General Chair for NANOARCH 2021, GLSVLSI 2017 and the IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT) 2013, and a Technical Program Committee Chair for NANOARCH 2022, GLSVLSI 2016, DFT 2012 and the Symposium on Stochastic & Approximate Computing for Signal Processing and Machine Learning, 2017.