A High-Performance Stochastic Simulated Bifurcation Ising Machine

Tingting Zhang
University of Alberta
Edmonton, Canada
ttzhang@ualberta.ca

Hongqiao Zhang
ShanghaiTech University
Shanghai, China
zhanghq2@shanghaitech.edu.cn

Zhengkun Yu
ShanghaiTech University
Shanghai, China
yuzhk2023@shanghaitech.edu.cn

Siting Liu
ShanghaiTech University
Shanghai, China
liust@shanghaitech.edu.cn

Jie Han
University of Alberta
Edmonton, Canada
jhan8@ualberta.ca

Abstract

Ising model-based computers, or Ising machines, have recently emerged as high-performance solvers for combinatorial optimization problems (COPs). A simulated bifurcation (SB) Ising machine searches for the solution by solving pairs of differential equations related to the oscillator positions and momenta. It benefits from massive parallelism but suffers from high energy. As an unconventional computing paradigm, dynamic stochastic computing implements accumulation-based operations efficiently. By exploiting the advantages in algorithm and hardware codesign, this article proposes a high-performance stochastic SB machine (SSBM) with efficient hardware. To this end, we develop a stochastic SB (sSB) algorithm such that the multiply-and-accumulate (MAC) operation is converted to multiplexing and addition while the numerical integration is implemented by using signed stochastic integrators (SSIs). Specifically, the sSB stochastically ternarizes position values used for the MAC operation. Two types of SB cells are constructed. A stochastic computing SB cell contains two SSIs with a high area efficiency, while a binary-stochastic computing SB cell contains one binary integrator and one SSI with a reduced delay. Based on sSB, an SSBM is then built by using the proposed SB cells as the basic building block. The designs and syntheses of two SSBMs with 2000 fully connected spins require at least 10.62% smaller area than the state-of-the-art designs. It shows the potential of stochastic computing for SB to efficiently solve COPs.

CCS Concepts

• **Hardware** → **Emerging architectures**; • **Theory of computation** → *Models of computation.*

Keywords

simulated bifurcation, Ising model, stochastic computing

DAC '24, June 23–27, 2024, San Francisco, CA, USA

ACM ISBN 979-8-4007-0601-1/24/06.
https://doi.org/10.1145/3649329.3655927

ACM Reference Format:
Tingting Zhang, Hongqiao Zhang, Zhengkun Yu, Siting Liu, and Jie Han. 2024. A High-Performance Stochastic Simulated Bifurcation Ising Machine . In *61st ACM/IEEE Design Automation Conference (DAC '24), June 23–27, 2024, San Francisco, CA, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3649329.3655927

1 Introduction

Combinatorial optimization is essential in social and industrial applications, such as financial portfolio management and chip design [1]. Ising model-based solvers, or Ising machines, have emerged as an efficient domain-specific architecture for solving COPs with a polynomial time. The Ising model describes the ferromagnetism among a set of magnetic spins in statistical physics [2]. Ising machines can broadly be classified into two categories. One describes the spin dynamics based on a physical model, implemented by, for example, using superconducting circuits [3], pulse lasers [4], or oscillators [5]. Another class utilizes various heuristic algorithms, such as simulated annealing (SA) [6] and simulated bifurcation (SB) [7], to numerically emulate spin dynamics. The latter is advantageous over the former due to the ability to solve Ising problems with dense spin-to-spin interactions with a high precision. However, it faces with the drawback of a relatively high hardware cost.

Conventional SA emulates heating and cooling processes in metals. The performance bottleneck in the sequential update of spin states requires developing variants of SA for fast energy convergence. For instance, the stochastic cellular automata (SCA) annealing [8] leverages a two-layer spin structure for parallel spin update. However, the use of spin replicas leads to low scalability. SB emulates the adiabatic evolution of a classical nonlinear Hamiltonian system of a network of oscillators [7, 9]. It accelerates the search by the simultaneous update of spin states. Nevertheless, SB uses continuous oscillator positions to obtain discrete spin states, so it requires complex hardware for computation. Hence, it is imperative to develop a hardware-friendly architecture for an SB machine.

As an unconventional paradigm, stochastic computing encodes a number by an independent bit stream of 0's and 1's, which enables performing arithmetic operations by using simple logic gates [10]. However, a long sequence is often required for a high accuracy, thus leading to a high latency and energy consumption. With a high efficiency, dynamic stochastic computing uses each bit in a stochastic sequence, referred to as a dynamic stochastic sequence (DSS), to encode a variable signal [11]. It is well suited for iterative

accumulation-based computations. In particular, it can solve differential equations by using numerical integration with a significantly reduced circuit complexity [11].

This paper presents the design of a first SB Ising machine using stochastic computing, referred to as a stochastic SB machine (SSBM). The SSBM is implemented by using signed stochastic integrators (SSIs) for numerical integration and stochastic ternarization to simplify multipliers in the multiply-and-accumulate (MAC) operation. The contributions of this work lie in the following novelties in the algorithm and architecture design:

- A stochastic SB (sSB) algorithm ternarizes the values of oscillator positions so that the MAC operation is converted to multiplexing and addition operations;
- A stochastic computing SB cell (SC-SBC) is built by using two SSIs for area efficiency, and a binary-stochastic computing SB cell (BSC-SBC) is constructed by using one binary integrator and one SSI for delay reduction;
- To implement the sSB, a prototype SSBM is designed by using the SC-SBC or BSC-SBC as a basic building block for solving COPs such as the max-cut problem (MCP).

This paper is organized as follows. Section 2 introduces the basics of SB and dynamic stochastic computing. Section 3 presents the sSB algorithm and SB cell designs using SSIs. The circuit design of the SSBM is discussed in Section 4. Section 5 reports the experimental results. Section 6 concludes this paper.

2 Preliminaries

2.1 The Ising Model and Simulated Bifurcation

The Hamiltonian (H) of an N-spin Ising model is given by [1]

$$H = -\frac{1}{2}\sum_{i,j}^{N} J_{ij}s_i s_j - \sum_{i}^{N} h_i s_i, \tag{1}$$

where s_i (or s_j) is the state of the ith (or jth) spin with the value of +1 or −1; J_{ij} describes the interaction between s_i and s_j; h_i is the external field on s_i. Since the model can be reduced to one without the external fields by introducing an ancillary spin [12], we focus on the Ising model without external fields in the following discussion.

An SB algorithm numerically models the adiabatic evolution of oscillator networks. It essentially solves pairs of differential equations related to the positions and momenta of oscillators. The Hamiltonian equations of the system are given by [12]

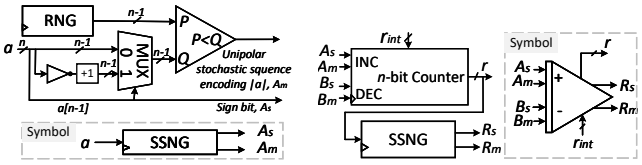
$$\dot{x_{i,t}} = f(\boldsymbol{y_t})_i = a_0 y_{i,t}, \tag{2}$$

$$\dot{y_{i,t}} = g(\boldsymbol{x_t})_i = -(a_0 - a(t))x_{i,t} + c_0 P_{i,t}, \tag{3}$$

where $\boldsymbol{x_t}(=\{x_{1,t},...,x_{N,t}\})$ and $\boldsymbol{y_t}(=\{y_{1,t},...,y_{N,t}\})$ are position and momentum vectors at the tth time step, respectively; $\dot{x_{i,t}}$ and $\dot{y_{i,t}}$ are the derivatives of the ith oscillator position ($x_{i,t}$) and the ith oscillator momentum ($y_{i,t}$) with respect to the tth time step, respectively; a_0 and c_0 are constants; $a(t)$ is a linearly-increasing variable. $x_{i,t}$ is replaced by its sign ($sgn(x_{i,t})$) and $y_{i,t}$ is reset to 0 when $|x_{i,t}| > 1$. $P_{i,t}$ is computed by using the interactions among spins ($\boldsymbol{J}$) and position variables ($\boldsymbol{x_t}$). Two variant SBs, ballistic SB (bSB) and discrete SB (dSB), differ in the expressions of $P_{i,t}$, given by [12]

$$P_{i,t} = \begin{cases} \sum_{j=1}^{N} J_{ij}x_{j,t} & \text{in bSB} \\ \sum_{j=1}^{N} J_{ij}sgn(x_{j,t}) & \text{in dSB} \end{cases}. \tag{4}$$

When t reaches the predefined number of iterations for update, the sign of $x_{i,t}$ indicates the state of the ith spin, i.e., s_i in (1). The states of spins provide the solution found by the SB.



(a) The circuit diagram of an SSNG. (b) The circuit diagram of an SSI.

Figure 1: Elements in sign-magnitude stochastic computing [13].

2.2 Dynamic Stochastic Computing and Stochastic Integrators

In dynamic stochastic computing, changing signals are encoded by DSS's consisting of '0's and '1's. The probability of each bit being '1' is equivalent to the (changing) probability encoded in a DSS [11]. Various schemes are used to encode values within different number ranges, including unipolar, bipolar, and sign-magnitude representations. The latter two can be used to encode signed values between $[-1, 1]$. However, the bipolar representation faces a larger accuracy loss for numbers near zero due to its larger variance [13]. Since SB deals with signed position and momentum values with magnitudes around zero at the beginning of a search, the two-bit sign-magnitude representation is used in this work, one bit for the sign and the other for the magnitude [14].

Figure 1(a) shows the circuit diagram of a signed stochastic number generator (SSNG). The input is an n-bit varying signal within $[-1, 1]$, denoted as a. The sign bit for a, i.e., its most significant bit, $a[n-1]$, is output as A_s. The random number generator (RNG) generates uniformly distributed numbers within $[0, 1]$. The RNG and the comparator are then used to generate a magnitude bit, denoted by A_m, and its expectation satisfies $E[A_m] = |a|$. If the magnitude of a is larger than the random number, a '1' is generated; otherwise, a '0' is generated. Thus, the sign-magnitude DSS of a, denoted by A ($=\{A_sA_m\}$), encodes the value of either 0, −1, or +1 with $E[\hat{A}] = a$, where $\hat{A}$ is the encoded value of the DSS A.

The stochastic integrator is an essential sequential circuit in dynamic stochastic computing. As shown in Fig. 1(b), an SSI using the sign-magnitude representation consists of a counter and an SSNG. The counter with n-bit width counts from -2^{n-1} to $2^{n-1}-1$. Consider two discrete-time digital signals a and b within $[-1, 1]$. Their DSS's A ($=\{A_sA_m\}$) and B ($=\{B_sB_m\}$) determine the increment and decrement of the counter. Let the values of the input DSS's A and B at the tth clock cycle be $\hat{A}_t$ and $\hat{B}_t$, and the integers stored in the counter at the tth and $(t+1)$th clock cycle be $\hat{r}_t$ and $\hat{r}_{t+1}$, respectively. The counter updates its value by following [13]

$$\hat{r}_{t+1} = \hat{r}_t + \hat{A}_t - \hat{B}_t. \tag{5}$$

The value encoded by the n-bit signed integer $\hat{r}$, denoted by r, is a signed value between −1 and 1, satisfying $r = \frac{\hat{r}}{2^{n-1}}$. Finally, it is converted to a sign-magnitude DSS, denoted as R ($=\{R_sR_m\}$). Hence, let the initial value for r be r_{int}, the SSI performs the computation of r for integrating $a - b$ with the step size $\frac{1}{2^{n-1}}$.

3 Stochastic Simulated Bifurcation (sSB)

3.1 The Algorithm

Compared to bSB, dSB binarizes position values to $\{-1, 1\}$ when computing $P_{i,t}$, as in (4). In this way, the MAC operation is simplified

to addition and subtraction. However, the binarization introduces large numerical approximations in representing position values since the magnitudes are around zero early in a search. It results in an instability of the obtained solution quality and slow energy convergence [12]. Moreover, the computation in SB is deterministic, thus reducing its ability to jump out of local minima. Therefore, we introduce zero as an additional quantized position value and utilize a random threshold to determine the quantized value.

As a result, the sSB ternarizes position variables to $\{-1, 0, 1\}$ when computing $P_{i,t}$. It does not only avoid the massive use of multipliers but also reduces power due to a high probability of quantizing position values to zero. To solve an N-spin Ising problem, sSB introduces a vector ***rand*** with N random numbers. Each position value is compared with a random number to determine the ternarized value. Thus, $P_{i,t}$ in (3) is computed as

$$P_{i,t} = \sum_{j=1}^{N} J_{ij} R(x_{j,t}), \tag{6}$$

where

$$R(x_{j,t}) = \begin{cases} +1 & x_{j,t} > \text{rand}_j \\ -1 & x_{j,t} < -\text{rand}_j \\ 0 & others \end{cases}, \tag{7}$$

where rand_j $(1 \le j \le N)$ is a random number within $[0, 1]$ and $-1 \le x_{j,t} \le 1$.

3.2 Formulations of Stochastic SB Cell Designs

The semi-implicit Euler integration method has shown its efficiency for SB to find a high-quality and stable solution [12]. In this way, the momentum values are first updated and then the position values are updated by using the new momentum values, as follows

$$y_{i,t+1} = y_{i,t} + \eta g(\boldsymbol{x_t})_i, \tag{8}$$

$$x_{i,t+1} = x_{i,t} + \eta f(\boldsymbol{y_{t+1}})_i, \tag{9}$$

where η is the time step size for the integrator, $g(\boldsymbol{x_t})_i$ and $f(\boldsymbol{y_{t+1}})_i$ are given in (2) and (3).

In SB, the momentum values are usually initialized to zero. Thus, $y_{i,0} = 0$, and $y_{i,t+1}$ is obtained by iterative computation as

$$y_{i,t+1} = \eta \sum_{k=0}^{t} g(\boldsymbol{x_k})_i. \tag{10}$$

Moreover, a_0 in (2) is usually set to 1 and the position values are usually initialized to random numbers. Thus, $f(\boldsymbol{y_{t+1}}) = y_{i,t+1}$. Then, based on (9) and (10), the ith oscillator position value at the $(t+1)$th time step $(x_{i,t+1})$ can be computed as

$$x_{i,t+1} = x_{i,0} + \eta^2 \sum_{j=0}^{t} \sum_{k=0}^{j} g(\boldsymbol{x_k})_i. \tag{11}$$

3.2.1 The Stochastic Computing SB Cell (SC-SBC) We propose an SC-SBC to solve the pairs of differential equations. As shown in Fig. 2(a), an SC-SBC consists of two SSIs and one SSNG: one of the two SSIs (named SSI1) is used for updating the momentum value and the other (named SSI2) is used for updating the position value.

To reduce the errors introduced by the DSS encoding, $g(\boldsymbol{x_t})_i$ in (3) is first computed as the input of the SSNG and then converted to a DSS $G_{i,t}$ with the encoded value of $\hat{G}_{i,t}$. Let the n-bit signed binary numbers stored in the counters in SSI1 and SSI2 at the tth (or $(t+1)$th) time step be $\hat{y}_{i,t}$ (or $\hat{y}_{i,t+1}$) and $\hat{x}_{i,t}$ (or $\hat{x}_{i,t+1}$), respectively. The SC-SBC updates the values in the two counters at the $(t+1)$th time step by

$$\hat{y}_{i,t+1} = \hat{y}_{i,t} + \hat{G}_{i,t}, \tag{12}$$

$$\hat{x}_{i,t+1} = \hat{x}_{i,t} + \hat{Y}_{i,t+1}, \tag{13}$$

where $\hat{Y}_{i,t+1}$ is the encoded value of the DSS $Y_{i,t+1}$ for $y_{i,t+1}$ $(= \frac{\hat{y}_{i,t+1}}{2^{n-1}})$.

To convert the integers in the counters into stochastic numbers in the sign-magnitude representation, both sides of (12) and (13) are normalized by $\frac{1}{2^{n-1}}$. Then, we have

$$E[\hat{Y}_{i,t+1}] = E[\hat{Y}_{i,t}] + \tfrac{1}{2^{n-1}} E[\hat{G}_{i,t}], \tag{14}$$

$$E[\hat{X}_{i,t+1}] = E[\hat{X}_{i,t}] + \tfrac{1}{2^{n-1}} E[\hat{Y}_{i,t+1}], \tag{15}$$

where $\hat{X}_{i,t+1}$ (or $\hat{X}_{i,t}$) is the encoded value of the DSS $X_{i,t+1}$ (or $X_{i,t}$) for $x_{i,t+1}$ (or $x_{i,t}$); $E[\hat{Y}_{i,t+1}] = y_{i,t+1} = \frac{\hat{y}_{i,t+1}}{2^{n-1}}$, $E[\hat{Y}_{i,t}] = y_{i,t} = \frac{\hat{y}_{i,t}}{2^{n-1}}$, $E[\hat{X}_{i,t+1}] = x_{i,t+1} = \frac{\hat{x}_{i,t+1}}{2^{n-1}}$, and $E[\hat{X}_{i,t}] = x_{i,t} = \frac{\hat{x}_{i,t}}{2^{n-1}}$.

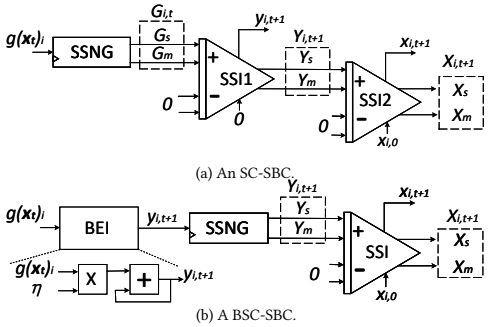
Given $E[\hat{G}_{i,t}] = g(\boldsymbol{x_t})_i$ and the initially encoded value by the integer stored in the counter in SSI1, $y_{i,0} = 0$ (thus $E[\hat{Y}_{i,0}] = 0$), based on (14), $E[\hat{Y}_{i,t+1}]$ is obtained as

$$E[\hat{Y}_{i,t+1}] = \eta \sum_{k=0}^{t} \tfrac{1}{2^{n-1}} g(\boldsymbol{x_k})_i. \tag{16}$$

Let $x_{i,0}$ be the initially encoded value by the integer stored in the counter in SSI2. Based on (15) and (16), $E[\hat{X}_{i,t+1}]$ is obtained as

$$E[\hat{X}_{i,t+1}] = x_{i,0} + \tfrac{1}{2^{n-1}} \sum_{j=0}^{t} \sum_{k=0}^{j} \tfrac{1}{2^{n-1}} g(\boldsymbol{x_k})_i, \tag{17}$$

which is equivalent to (11) when $\eta = \frac{1}{2^{n-1}}$.



(a) An SC-SBC.

(b) A BSC-SBC.

Figure 2: Circuit block diagrams of the proposed SB cells.

3.2.2 The Binary-Stochastic Computing SB Cell (BSC-SBC) The derivative in (3) leverages the interactions between spins to help determine the evolution of position and momentum values. To improve the computational accuracy, we propose a BSC-SBC. As shown in Fig. 2(b), it consists of a binary Euler integrator (BEI) to update the momentum value, an SSNG to convert the binary momentum value to its DSS, and an SSI to update the position value. In the BEI, the derivative value is first multiplied with the step size. Then, the product is added to the value updated in the previous time step for implementing a Euler integration.

Given $g(\boldsymbol{x_t})$ and η as inputs, BEI performs the computation in (8) to obtain $y_{i,t+1}$. $y_{i,t+1}$ can be formulated as (10). As the output of the SSNG, $Y_{i,t+1}$ satisfies $E[\hat{Y}_{i,t+1}] = y_{i,t+1}$ and then it is sent to SSI. Similar as the analysis in Section 3.2.1, the output of SSI satisfies

$$E[\hat{X}_{i,t+1}] = x_{i,0} + \frac{\eta \sum_{j=1}^{t+1} E[\hat{Y}_{i,j}]}{2^{n-1}} = x_{i,0} + \frac{\eta \sum_{j=0}^{t} \sum_{k=0}^{j} g(\boldsymbol{x_k})_i}{2^{n-1}} \tag{18}$$

which is equivalent to (11) when $\eta = \frac{1}{2^{n-1}}$.

4 A Stochastic Simulated Bifurcation Machine

4.1 The Overall Architecture

To implement the sSB, the position values are first initialized and the time-changing parameter in (3) is calculated. After obtaining the ternarized position values, the $P_{i,t}$ value in (6) is computed for updating the momentum value as per (3). Subsequently, the position value is updated using the new momentum value by (2). After running a predefined number of iterations for the update, the sign bit in the position value provides the solution for the COP. The hardware design of an SSBM is presented in Fig. 3 for the sSB, by using either SC-SBCs or BSC-SBCs as basic building blocks to implement the integration.

In Fig. 3, the N-spin SSBM consists of a linearly increasing parameter generator (LPG), N spin units and two memory blocks (an X_mem and a RN_mem). The LPG computes $-(a_0 - a(t))$ in (3) as a linearly increasing parameter by a pre-calculated value A for the determined iteration for the update. Then, the spin unit computes the derivative value in (3) and updates the position and momentum values by using either an SC-SBC or a BSC-SBC. The X_mem stores the ternarized position values generated from the SBC. The DSS of the position value obtained by comparing the position value with a random number takes one value from +1, −1, and 0. Its generation mechanism is similar to that of the ternarized position value in (7). Therefore, the DSS's of position values generated by the SB cell are used as $R(x_{j,t})$ in (6). The RN_mem stores N n-bit random numbers for SBCs in the N spin units. To improve the accuracy of stochastic computing, the random numbers are generated by using low-discrepancy Sobol sequences [13]. Moreover, to reduce the variance of the estimated position value, the same random number is used in the SSNGs of an SBC [13]. In this way, the RNi is used in the spin unit i ($1 \le i \le N$).

4.2 The Spin Unit Design

Given spin unit 1 as an example, the J_1 module stores $N-1$ interactions (J_{1j}, where $1 \le j \le N$ and $j \ne 1$) between the 1st spin and the other $N-1$ spins (Note that $J_{11} = 0$). The GX module generates the derivative value in (3), i.e., g_1 for spin unit 1. The AX calculates the first term in (3). The P_1 in (6) is calculated in the MAC and then multiplied by c_0 to implement the second term in (3). Since sSB ternarizes the position values for the MAC as in (6), the multiplications of interaction values and ternarized position values are implemented by using multiplexing operations in MUL. The MUL and ADD modules can be reused to save hardware at a cost of delay; an accumulator (ACC) is required to obtain the P_1 value.

4.2.1 The Circuit Design of the SC-SBC Taking g_1 and RN_1 as the inputs, in the SC-SBC, the binary value g_1 is first converted to its DSS G_1 (G_sG_m). Since the counter in SSI1 updates the value only depending on one increment signal, it is implemented by using n-bit adders and a register, where $n = \lceil |log(\eta)| \rceil + 1$. Since G_1 can only be 0, +1 and −1, to adapt to the number representation in the n-bit adders, $n-1$ copies of G_s and one G_m, i.e., $(G_sG_s...G_m)_2$, are used as one input of the adder. The updated momentum from the output of the adder is then converted to its DSS Y_1 (Y_sY_m) by the SSNG. Given $(Y_sY_s...Y_m)_2$ as the input, SSI2 updates the position value. Required by the update rule, the position value during the search is constrained within $[-1, 1]$; otherwise, the position value is replaced by its sign and the momentum value is reset to 0. To simplify the implementation, the carry out bit C_{out} from the n-bit adder in SSI2 is used to detect if an overflow occurs. If $C_{out} = 1$, the momentum value will be reset to 0 and the position value will retain its previous value; otherwise, their values will be updated depending on the outputs of the adders.

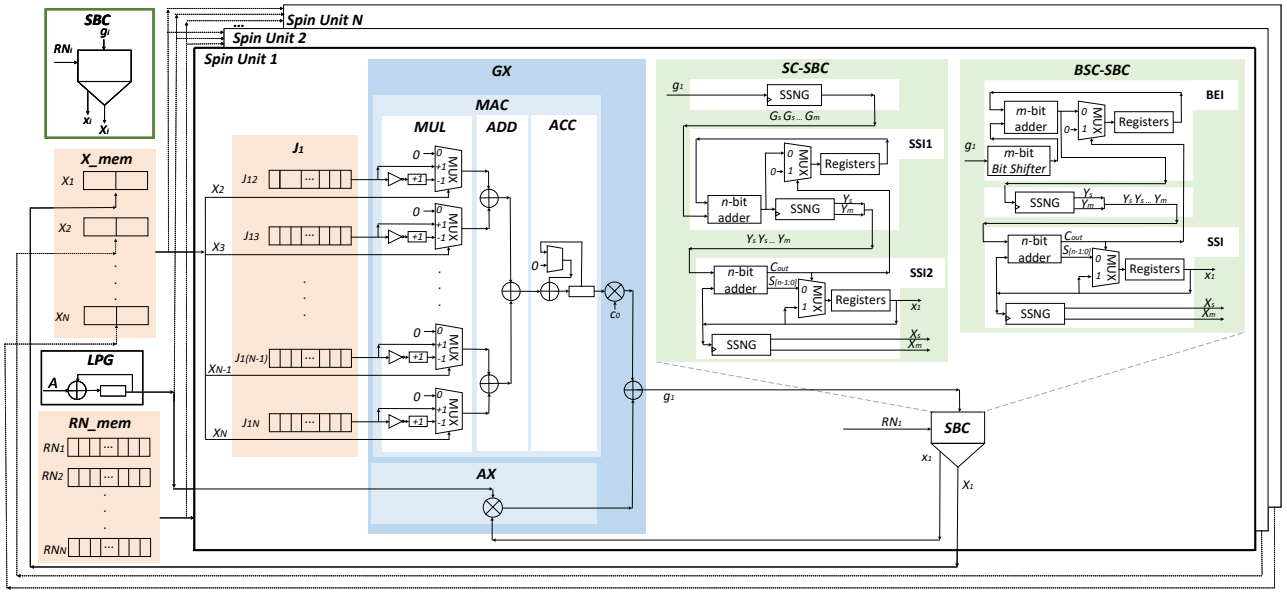


Figure 3: A general architecture of an N-spin SSBM. LPG: a linearly increasing parameter generator; GX: a momentum derivative (as in (3)) calculator; MAC: a multiply-accumulate unit (as in (6)); MUL: an element-wise multiplication unit; ADD: an addition unit; ACC: an accumulator; AX: a calculator for the first term in (3); SBC: a simulated bifurcation cell; J_1: an interaction memory unit for the 1st spin; X_mem: a memory unit for DSS's of position values; and RN_mem: a memory unit for random numbers.

4.2.2 The Circuit Design of the BSC-SBC The BSC-SBC shares a similar mechanism with the SC-BSC, but it updates the momentum value using BEI. In BEI, the incremental value of the momentum, i.e., g_1 multiplied with the time step η, is implemented by right shifting g_1 with $n-1$ bits. An m-bit adder performs the addition of this value to the current momentum (where m depends on the computation precision in GX). The addition result is then converted to a DSS using an SSNG and sent to the SSI.

5 Evaluation

5.1 Solving Max-cut Problems (MCPs)

The solution quality is evaluated on an MCP, the widely used K_{2000} benchmark [8, 15], which is an Ising problem with 2000 fully connected spins. The average (*Ave*), maximum (*Max*), and minimum (*Min*) of the cut values are obtained by different SB machines with the time step (T_s) of 1000 and 10000 from 100 trials, as shown in Fig. 4. A larger *Ave*, *Max* and *Min* indicate a higher general performance, a higher likelihood to jump out of the local optima, and a higher stability, respectively. The manually tunable time step sizes (η in (8) and (9)) of 0.125, 0.25, and 0.5 are considered, thus the bit-width of the adder in the SSI (n) can be 4, 3, and 2, respectively. Note that Fig. 4 presents the highest values in the three metrics obtained by the bSBM and dSBM using different time step sizes.

Compared with dSBM, the bSBM attains higher *Ave*, *Max*, and *Min* when $T_s = 1000$ because the energy converges in a short search in the bSBM. However, with the increase of T_s, it becomes vulnerable for bSBM to be stuck in the local minima. The dSBM outperforms the bSBM in finding a better solution, indicated by higher *Ave* and *Max*, when $T_s = 10000$. Due to the computational accuracy loss introduced by the use of binarized position values, the dSBM cannot achieve a fast convergence but it has a higher probability of jumping out of the local minima in a long search.

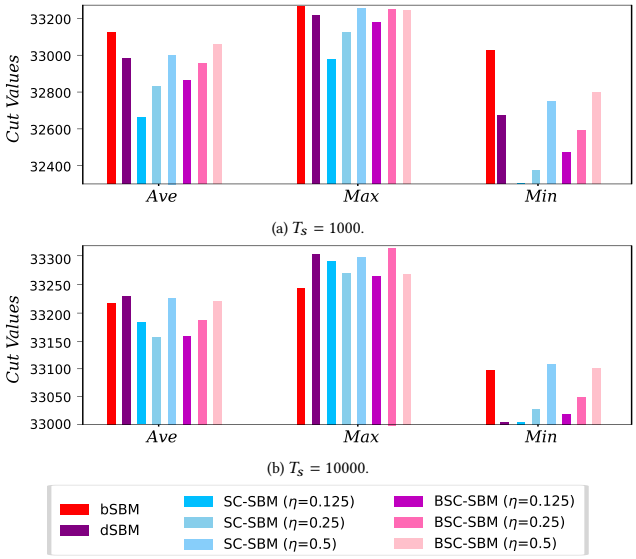


Figure 4: Solution quality (in the statistics of cut values) of using the bSB machine (bSBM), the dSB machine (dSBM), the sSB machine built by using SC-SBCs (SC-SBM), and the sSB machine built by using BSC-SBCs (BSC-SBM) to solve the K_{2000} benchmark. K_{2000}: 2000 nodes, 1999000 edges, a complete graph, best-known cut value: 33337.

Table 1: The Values of P_g and S_g for the K_{2000} Benchmark

Vaules of P_g and S_g with T_s		SB Machines			
		bSBM	dSBM	SC-SBM	BSC-SBM
T_S = 1000	$P_{99.5\%}$	38%	4%	6%	22%
	$S_{99.5\%}$	7633	112811	74426	18534
T_S = 10000	$P_{99.8\%}$	0	6%	4%	2%
	$S_{99.8\%}$	-	744265	1128110	2279481

$P_g = \frac{Trg}{Tr}$, where Tr is the total number of trials and Tr_g is the number of trials that an SB machine obtains the target solution with the target quality g [15].
$S_g = T_s \frac{log0.01}{log(1-P_g)}$, where T_s is the total number of time steps [15].

For the proposed SSBM (either SC-SBM or BSC-SBM), higher *Ave* and *Min* values are obtained with $\eta = 0.5$ than with $\eta = 0.125, 0.25$. Moreover, evaluated by *Ave* and *Min*, the BSC-SBM with $\eta = 0.5$ performs better than the SC-SBM with $\eta = 0.5$ when $T_s = 1000$, but the SC-SBM with $\eta = 0.5$ obtains a higher solution quality when $T_s = 10000$. It shows the advantages of BSC-SBM in a short search, and SC-SBM in a long search. It also indicates that although BSC-SBM provides more accurate computation than SC-SBM, it is more likely to be stuck in a local minimum, thus easier to discontinue the energy convergence process in a long search. However, the randomness introduced by the use of multiple DSS's in the SC-SBM allows the system to traverse more energy states. The SC-SBM and BSC-SBM with $\eta = 0.5$ yield a more stable solution quality than the dSBM, indicated by the larger *Min* values. When $T_s = 1000$, compared with using dSBM, using the BSC-SBM with $\eta = 0.5$ results in larger *Ave*, *Max* and *Min*. Moreover, when $T_s = 10000$, the SC-SBM with $\eta = 0.5$ consistently outperforms the bSBM.

The results in the metrics of probability-to-target (P_g) and step-to-target (S_g) are reported in Table 1. For $T_s = 1000$, the bSBM can quickly find a good solution, whereas the dSBM struggles to improve the cut values in a short search. The SSBMs can achieve a higher $P_{99.5\%}$ value than dSBM. For $T_s = 10000$, compared to dSBM and SSBMs, it is difficult for bSBM to reach 99.8% of the best-known cut value due to the lack of ability to jump out of the local minima. It shows that SSBMs find a better solution than dSBM in a short search and have a lower probability of being stuck at the local minima than bSBM in a long search.

5.2 Circuit Measurements

We first compare the proposed SC-SBM and BSC-SBM with the designs based on existing bSB and dSB algorithms. We consider 2-bit coefficient precision and the SSI in the SSBMs uses a 4-bit adder ($n = 4$). Note that all the spins perform parallel computation in the MAC unit. A pipeline strategy can be adopted depending on different hardware limitations when building a large-scale Ising machine. For a fair comparison, the bSBM and dSBM are implemented using the same experimental setup. For a 50-spin system, we use a clock frequency of 200 MHz because the bSBM has a longer critical path. Although not shown, due to space limitation, our experiments indicate that compared to the BSC-SBM, the SC-SBM requires a smaller area due to the use of two SSIs. However, it incurs a longer delay. Compared to the bSBM, the SSBM saves up to 74% in both area and power with a 1.2× speedup. Moreover, both SSBMs use a similar area as the dSBM that uses binarized position values for MAC. Notably, due to a high probability of quantizing position values to zeros in SSBMs, a reduction of at least 44% in power is achieved with a 1.19× speedup.

Table 2: The Circuit Measurements of Ising Machines

	D-wave[3]	JSSC'15[16]	JSSC'21[8]	ISSCC'21[17]	CICC'21[18]	JSSC'22[19]	ISSCC'23[20]	SC-SBM	BSC-SBM
Computing Method	Quantum Annealing	CMOS Annealing	SCA Annealing	Metropolis Annealing	Simulated Annealing	Simulated Annealing	Metamorphic Annealing	Simulated Bifurcation	Simulated Bifurcation
Technology	Superconductor	65nm CMOS	65nm CMOS	65nm CMOS	65nm CMOS	65nm CMOS	40nm CMOS	40nm CMOS	40nm CMOS
# Spins	$2k$	$20k$	512	$16k$	252	480	512	$2k$	$2k$
Topology	Chimera	Lattice	Complete	King	King	King	Complete	Complete	Complete
# Interaction per Spin	5	5	511	8	8	8	511	1999	1999
Coefficient Bit-Width	N/A	2	5	5	4	4	8	2	2
Spin Type	Qubit	SRAM	SRAM	Register	Register	Register	Register	Register	Register
Power per Spin	12.2 W	2.83 μW	1.27 mW	N/A	1.33 μW	0.18 μW	44-95 mW	0.74 mW	0.64 mW
Area per Spin (Normalized Area)	N/A	289 μm^2 (6.86 ×)	12207 μm^2 (1.13 ×)	552 μm^2 (3.28 ×)	1671 μm^2 (12.41 ×)	832 μm^2 (6.17 ×)	N/A	6370 μm^2 (1 ×)	6453 μm^2 (1.01 ×)
Frequency	N/A	100 MHz	320 MHz	100 MHz	64 MHz	200 MHz	134-336 MHz	250 MHz	250 MHz
# Spin Update Cycles	N/A	N/A	512	22	N/A	1	N/A	20	20

N/A: not reported. #: the number. Normalized area= (Area per Spin)/[(# Interaction per Spin) · (Coefficient Bit-Width) · (feature size)^2]. Simulation results for SC-SBM and BSC-SBM are obtained by using the Synopsys Design Compiler. A CMOS 40 nm technology is applied with a supply voltage of 1.0 V and a temperature of 25°C.

Table 2 compares the 2000-spin SC-SBM and BSC-SBM with the state-of-the-art works. Most of the Ising machines in the literature are implemented with a sparsely connected spin structure with different topologies. For a wider application scope, the proposed Ising machines uses a fully connected spin structure as the design in [8]. The dense connectivity between spins leads to an increase in area and power. If the power is normalized by the (squared) feature size, compared with the Ising machine in [8], the spins in SC-SBM and BSC-SBM require 1.5× and 1.3× more power per spin, respectively, due to the 3.9× larger connectivity. However, if the area is normalized for a spin by the coefficient bit-width, the number of spin interactions, and (squared) feature size, it is found that the proposed SC-SBM and BSC-SBM utilize at least 10.62% smaller area.

6 Conclusion

In this paper, a high-performance fully connected stochastic SB machine (SSBM) is designed for efficient and accurate combinatorial optimization using the Ising model. A stochastic SB (sSB) algorithm is developed to reduce the implementation complexity of the MAC by ternarizing position values. Therefore, the MAC operation can be realized by multiplexing and addition. Based on stochastic computing, two efficient SB cells are further designed by using SSIs to solve pairs of differential equations in SB. Specifically, the area-efficient SC-SBC consists of two SSIs, whereas the power-efficient BSC-SBC is built by using one binary integrator and one SSI. Experiments on a 2000-spin Ising problem show that the SSBM realizes fast energy convergence in a short search and also prevents the Ising model from being stuck at a local minimum in a long search. Moreover, the prototypes of the SSBM using the SC-SBC or BSC-SBC as a building block are respectively developed and synthesized. An improvement of at least 44% in power is achieved with a 1.19× speedup, compared to conventional SB machines. Finally, a fully connected 2000-spin system of the SSBM requires 10.62% less area than state-of-the-art designs. The performance of the SSBM will be investigated for solving larger-scale combinatorial optimization problems in future work.

Acknowledgments

The work at University of Alberta was supported by the Natural Sciences and Engineering Research Council of Canada under Grant No. RES0048688, RES0051374 and RES0054326. The work at ShanghaiTech University was supported by National Natural Science Foundation of China under Grant No. 62104127 and Shanghai Sailing Program under Grant No. 22YF1428300.

References

[1] A. Lucas, "Ising formulations of many NP problems," *Front. Phys.*, vol. 2, p. 5, 2014.
[2] N. Mohseni *et al.*, "Ising machines as hardware solvers of combinatorial optimization problems," *Nat. Rev. Phys.*, vol. 4, no. 6, pp. 363–379, 2022.
[3] M. W. Johnson *et al.*, "Quantum annealing with manufactured spins," *Nature*, vol. 473, no. 7346, pp. 194–198, 2011.
[4] Y. Yamamoto *et al.*, "Coherent Ising machines-quantum optics and neural network perspectives," *Appl. Phys. Lett.*, vol. 117, no. 16, p. 160501, 2020.
[5] T. Wang and J. Roychowdhury, "OIM: Oscillator-based Ising machines for solving combinatorial optimisation problems," in *UCNC*. Springer, 2019, pp. 232–256.
[6] S. Kirkpatrick, "Optimization by simulated annealing: Quantitative studies," *J. Stat. Phys.*, vol. 34, no. 5, pp. 975–986, 1984.
[7] H. Goto *et al.*, "Combinatorial optimization by simulating adiabatic bifurcations in nonlinear Hamiltonian systems," *Sci. Adv.*, vol. 5, no. 4, p. eaav2372, 2019.
[8] K. Yamamoto *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 JSSC*, vol. 56, no. 1, pp. 165–178, 2021.
[9] T. Zhang and J. Han, "Efficient traveling salesman problem solvers using the Ising model with simulated bifurcation," in *DATE*. IEEE, 2022, pp. 548–551.
[10] B. R. Gaines, "Stochastic computing systems," *Adv. Inf. Syst. Sci.: Volume 2*, pp. 37–172, 1969.
[11] S. Liu *et al.*, "Introduction to dynamic stochastic computing," *IEEE Circuits Syst. Mag.*, vol. 20, no. 3, pp. 19–33, 2020.
[12] H. Goto *et al.*, "High-performance combinatorial optimization based on classical mechanics," *Sci. Adv.*, vol. 7, no. 6, p. eabe7953, 2021.
[13] S. Liu *et al.*, "Gradient descent using stochastic circuits for efficient training of learning machines," *IEEE TCAD*, vol. 37, no. 11, pp. 2530–2541, 2018.
[14] A. Zhakatayev *et al.*, "Sign-magnitude SC: getting 10x accuracy for free in stochastic computing for deep neural networks," in *DAC*. ACM, 2018, pp. 1–6.
[15] T. Kanao and H. Goto, "Simulated bifurcation assisted by thermal fluctuation," *Commun. Phys.*, vol. 5, no. 1, p. 153, 2022.
[16] M. Yamaoka *et al.*, "A 20k-spin Ising chip to solve combinatorial optimization problems with CMOS annealing," *IEEE JSSC*, vol. 51, no. 1, pp. 303–309, 2015.
[17] T. Takemoto *et al.*, "4.6 a 144kb annealing system composed of 9×16kb annealing processor chips with scalable chip-to-chip connections for large-scale combinatorial optimization problems," in *ISSCC*, vol. 64. IEEE, 2021, pp. 64–66.
[18] Y. Su *et al.*, "A 252 spins scalable CMOS Ising chip featuring sparse and reconfigurable spin interconnects for combinatorial optimization problems," in *CICC*. IEEE, 2021, pp. 1–2.
[19] Y. Su *et al.*, "A scalable CMOS Ising computer featuring sparse and reconfigurable spin interconnects for solving combinatorial optimization problems," *IEEE JSSC*, vol. 57, no. 3, pp. 858–868, 2022.
[20] K. Kawamura *et al.*, "Amorphica: 4-replica 512 fully connected spin 336MHz metamorphic annealer with programmable optimization strategy and compressed-spin-transfer multi-chip extension," in *ISSCC*. IEEE, 2023, pp. 42–44.