

Approximate and Stochastic Ising Machines

Jie Han

(with Tingting Zhang, Qichao Tao, Bailiang Liu, Xingjian Gao,
Honglan Jiang and Siting Liu)

Department of Electrical and Computer Engineering
University of Alberta, Edmonton, AB, Canada

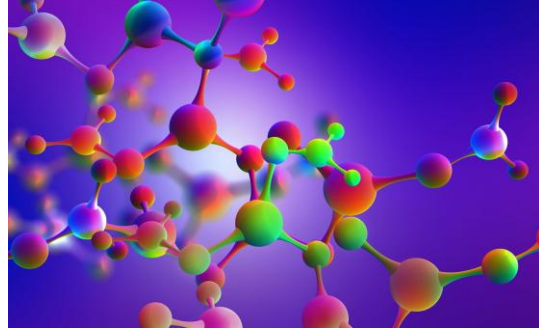
Outline

- Background and Motivation
- An Approximate Simulated Oscillator Ising Machine
 - Function Approximation
 - Circuit Design for FPGAs
- A Stochastic Simulated Bifurcation Ising Machine
 - Stochastic Simulated Bifurcation Cells
 - Stochastic Computing Circuit Design
- An Approximate Parallel Annealing Ising Machine
 - An Improved Parallel Annealing Algorithm
 - Approximate Circuit Design

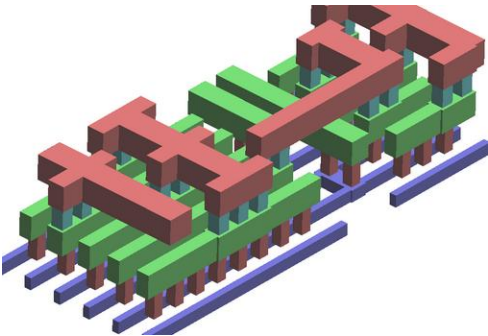
Combinatorial Optimization Problems (COPs)



Autonomous vehicle



Drug discovery

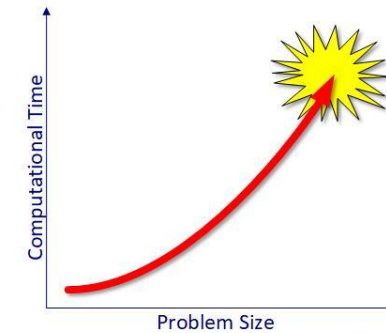


VLSI routing



Smart grid

- Solution space and time increase rapidly with the number of variables



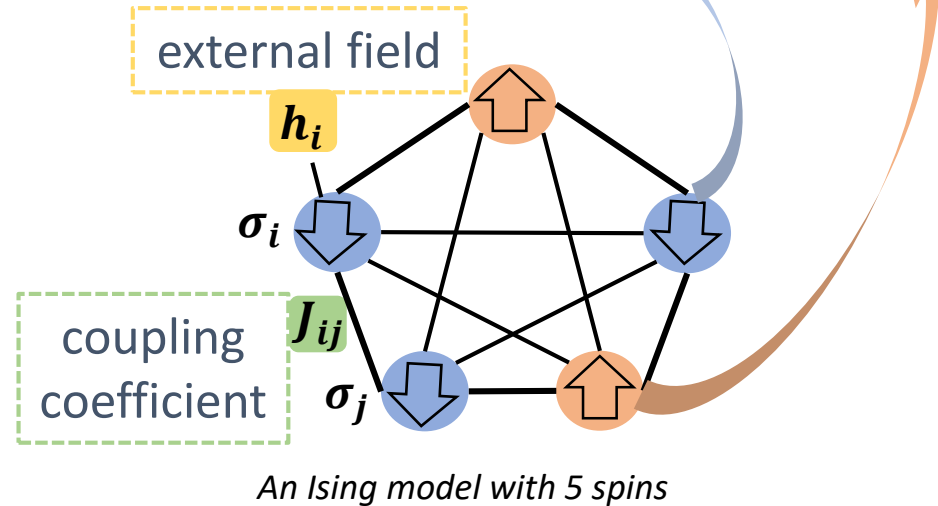
Near Optimal
Solution?

Ising machines

The Ising Model

■ Basics:

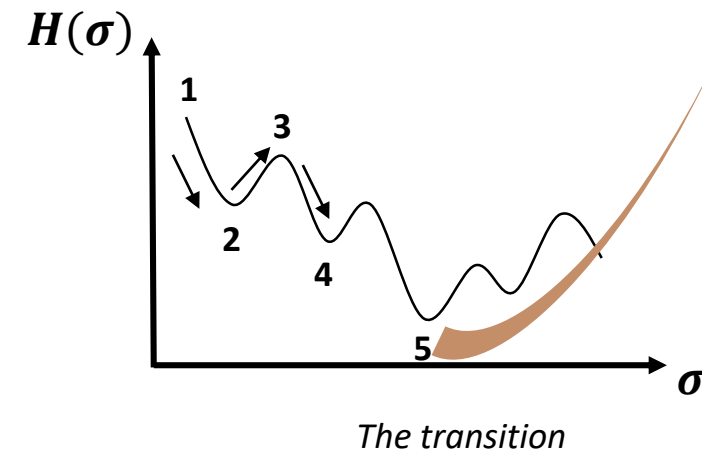
- It describes ferromagnetic interactions of magnetic spins
- Each spin takes **an upward (+1) state** or **a downward (-1) state**



■ The Hamiltonian of the Ising model

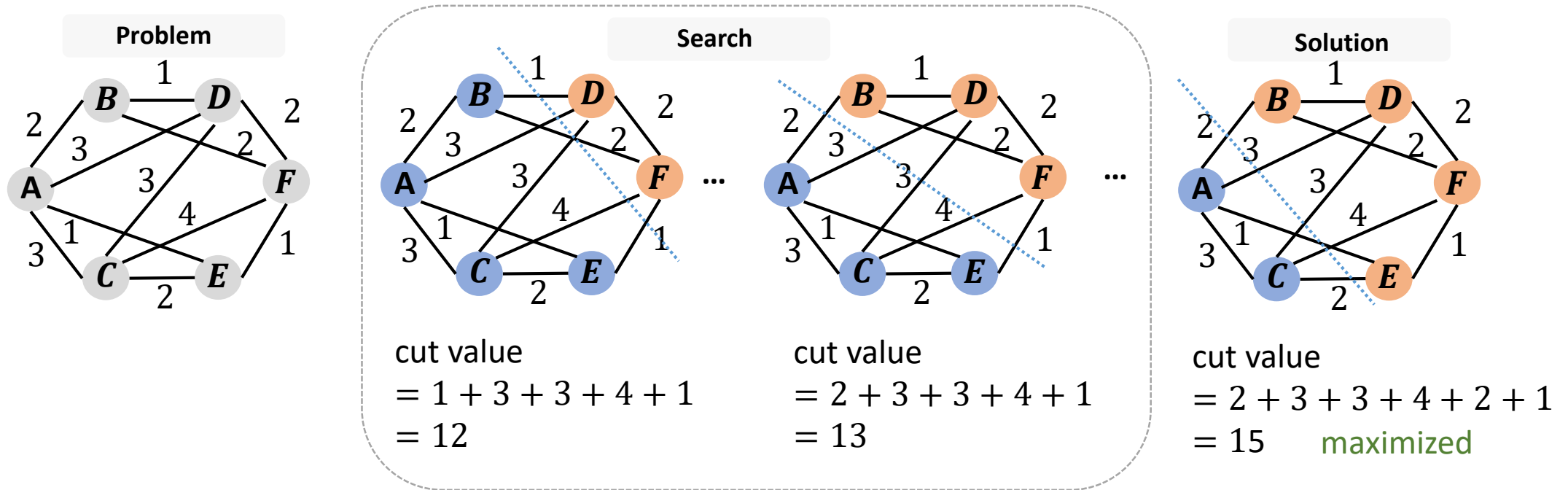
- Converges to **the lowest energy state**

$$H(\sigma) = -\frac{1}{2} \sum_{i,j} J_{ij} \sigma_i \sigma_j - \sum_i h_i \sigma_i$$



Max-cut Problems (MCPs)

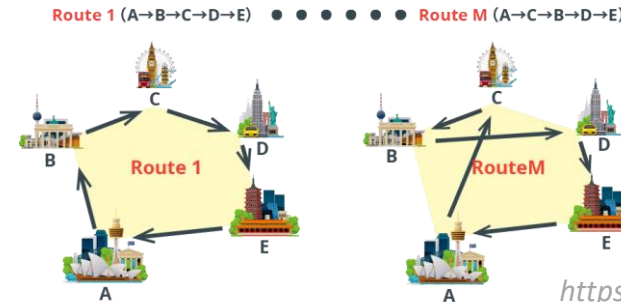
■ **Definition:** The MCP splits vertices into two groups to maximize the cut value, which is the total weight of edges between the two groups.



An example of max-cut problems

Solving a Problem using an Ising Machine

Given a COP



<https://annealing-cloud.com/en/knowledge/1.html>

1
Ising Formulation

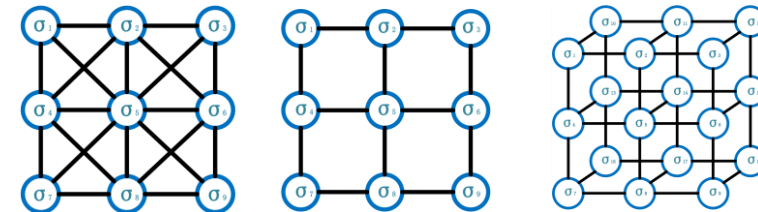
To adapt the formulation of the COP to the Ising model

$$H(\sigma) = -\frac{1}{2} \sum_{i,j} J_{ij} \sigma_i \sigma_j - \sum_i h_i \sigma_i$$

2
Embedding

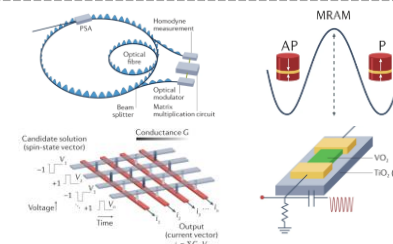
To satisfy limitations in the physical implementation of the Ising machine

Sparingly connected topology



3
Search

To search for the solution by updating the spin values in the Ising machine



CMOS digital Ising machine

challenge

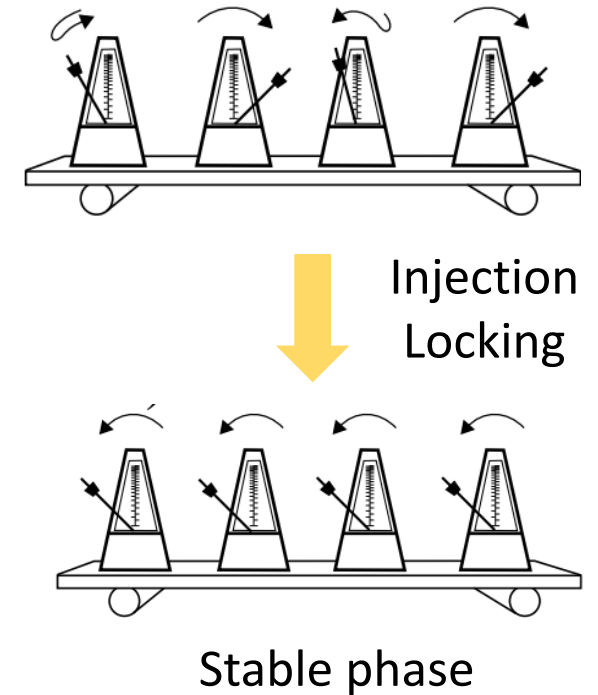
Parallel spin update with hardware efficiency

Outline

- Background and Motivation
- An Approximate Simulated Oscillator Ising Machine
 - Function Approximation
 - Circuit Design for FPGAs
- A Stochastic Simulated Bifurcation Ising Machine
 - Stochastic Simulated Bifurcation Cells
 - Stochastic Computing Circuit Design
- An Approximate Parallel Annealing Ising Machine
 - An Improved Parallel Annealing Algorithm
 - Approximate Circuit Design

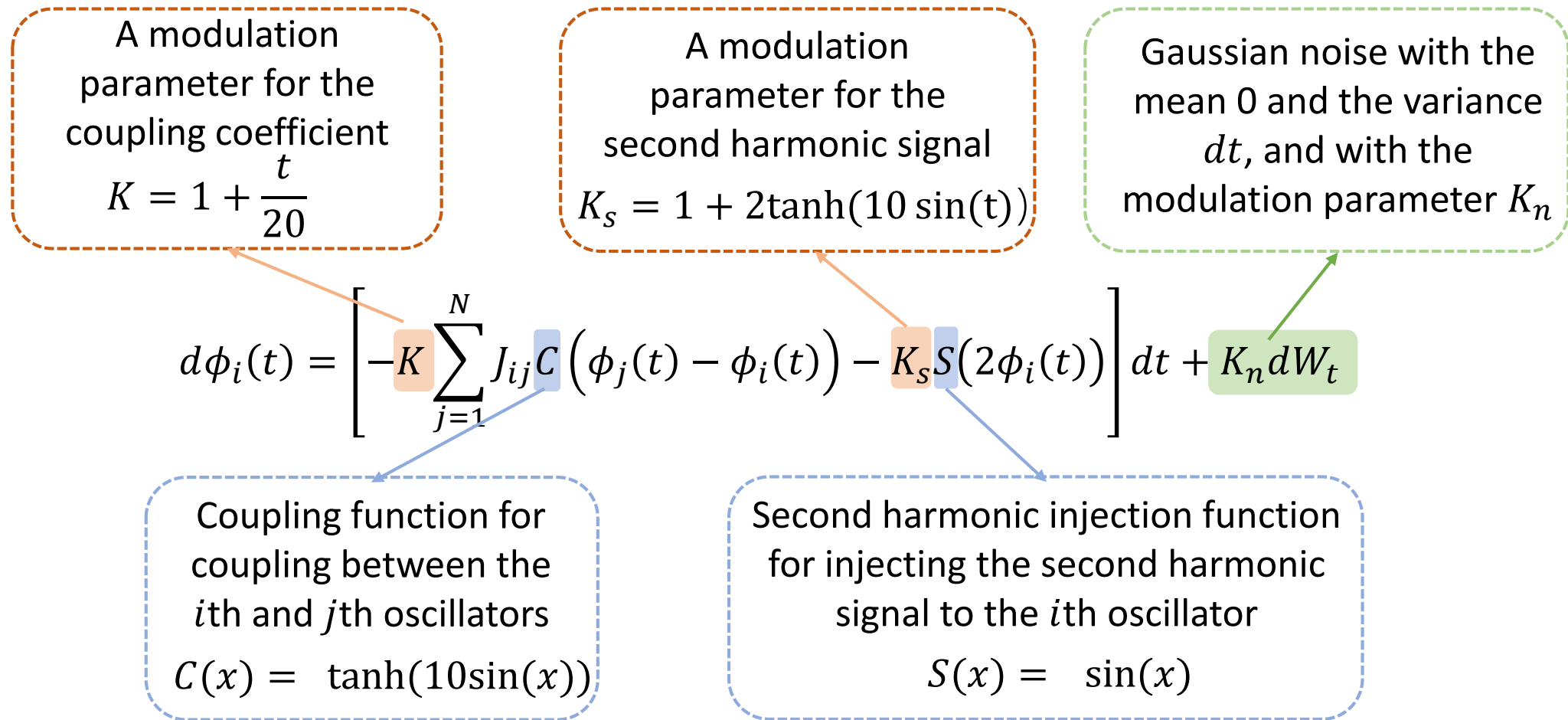
Oscillator-based Ising Machines

- **General Idea:** To implement the Ising model by an oscillator network and **encode spins into oscillator phases**.
- **Basics:** Oscillator's phase becomes **stably locked** under second harmonic injection locking (SHIL).
- **Mechanisms:** Coupled oscillator network dynamics **settle naturally to local minima** of a continuous version of the associated Ising Hamiltonian.
- **Challenge:** sparsely connected topology; limited precision of coupling coefficients due to the use of resistive or capacitive coupling.



Emulation of Oscillator-based Ising Machines

- **General Idea:** To build an oscillator network and **encode spins into oscillator phases**.
- Derivative of oscillator phases with time



Function Approximation

■ Coupling Function:

$$C(x) = \tanh(10\sin(x))$$

$$C(x) \approx \begin{cases} 0, & x=0 \text{ or } 1 \\ 1, & 0 < x < 1 \\ -1, & 1 < x < 2 \end{cases}$$

■ Second Harmonic Injection Function:

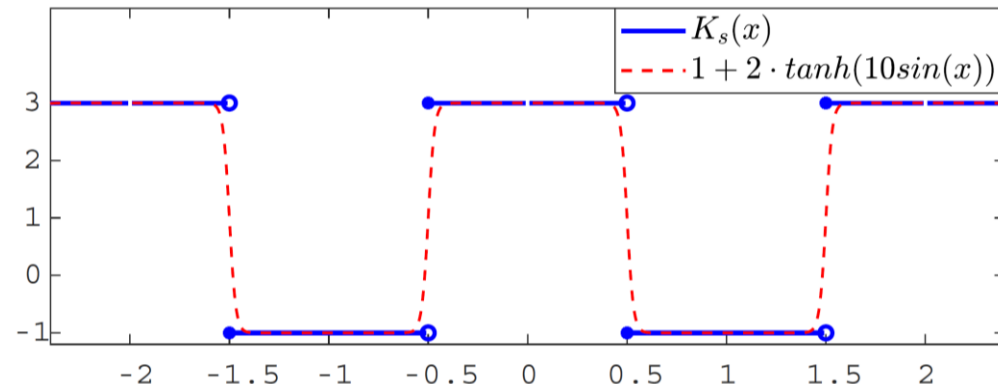
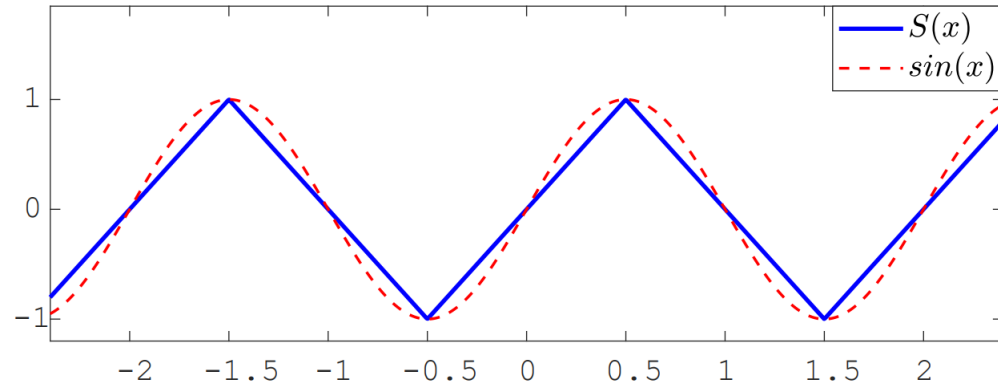
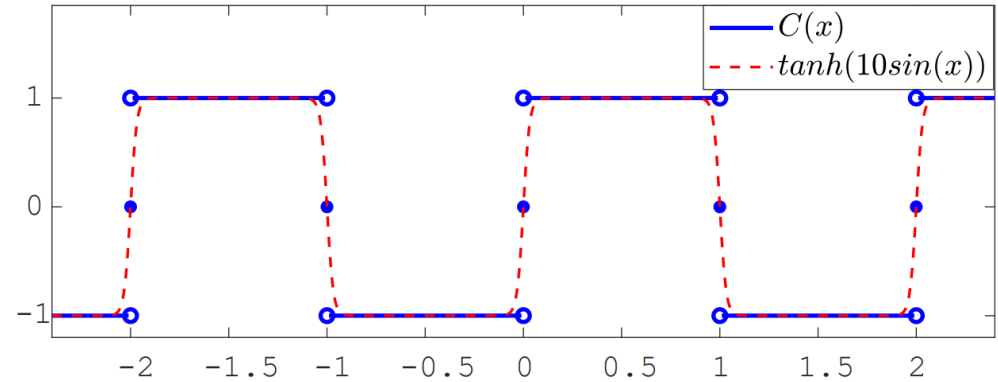
$$S(x) = \sin(x)$$

$$S(x) \approx \begin{cases} 2 - 2x, & 0.5 \leq x < 1.5 \\ 2x - 4, & -0.5 \leq x < 0.5 \end{cases}$$

■ K_s generation function:

$$K_s(x) = 1 + 2\tanh(10\sin(x))$$

$$K_s(x) \approx \begin{cases} -1, & -1.5 \leq x < 0.5 \\ 3, & 0.5 \leq x < 1.5 \end{cases}$$



Design Idea

■ Issues:

- Using a random number generator for each spin in a 1024-spin FPGA-based annealer increases hardware costs by at least tenfold.
- $\tanh(\cdot)$, $\sin(\cdot)$ are hardware consuming for the implementation by digital circuits.
- The period of 2π is inefficient for implementation.

■ Solutions:

- **Omitting the Gaussian noise**: It decreases the solution quality by a mere 0.8% for solving max-cut problems.
- **Using approximate linear piece-wise functions** for ease of implementation.
- Using **a period of 2**.

An FPGA-Oriented Model

■ The derivative of the i th oscillator phase with time :

$$d\phi_i(t) = \left[-\boxed{K} \sum_{j=1}^N J_{ij} \boxed{C}(\phi_j(t) - \phi_i(t)) - \boxed{K_s} \boxed{S}(2\phi_i(t)) \right] dt$$

Diagram illustrating the components of the derivative equation:

- $K = 1 + \frac{t}{20}$ (Prop. K_s generation)
- C (Prop. coupling function)
- S (Prop. Second Harmonic Injection Function)

■ The update of the i th oscillator phase:

1. Compute the new oscillator phase
2. Limit the oscillator phases to be within $[-2, 2]$

$$\phi'_i(t+1) = \phi_i(t) + d\phi_i(t)$$

$$\phi_i(t+1) = \boxed{B}(\phi'_i(t+1))$$

Prop. Evolution Bound Function

$$B(x) = \begin{cases} x, & 0 \leq x < 2 \\ x - 4, & 2 \leq x < 4 \end{cases}$$

The Proposed Algorithm

■ Inputs and Parameters:

- J : a matrix for the coupling coefficients;
- ΔT : the time step size for each iteration;
- T_{stop} : the simulation stop time.

■ Output:

- Φ : a matrix for the coupling coefficients.

■ The flow for updating the oscillator phase

- Calculate **the phase difference**
- Compute **the local field**
- Compute **the value of SHIL**
- Compute **the effect** of the local field and SHIL **on the oscillator phase**
- **Update** the oscillator phase

Algorithm 1 Oscillator-based Ising Machine Simulation Algorithm

Input: J , T_{stop} , and ΔT

Output: Φ

- 1: Initialize oscillator phases
- 2: **for** $q = 1$ to $T_{stop}/\Delta T$ **do**
- 3: $K \leftarrow 1 + q\Delta T/20$
- 4: $K_s \leftarrow 1 + 2 \times M(q\Delta T)$
- 5: **for** each oscillator, i , **do**
- 6: $pd_i \leftarrow \Phi_{q-1,i} \cdot ones(1, N) - \Phi_{q-1}$
- 7: $lf_i \leftarrow -K(J_i \cdot C(pd_i))$
- 8: $ss_i \leftarrow -K_s S(2\Phi_{q-1,i})$
- 9: $\Delta\Phi_{q,i} \leftarrow lf_i + ss_i$
- 10: **end for**
- 11: $\Phi_q \leftarrow B(\Phi_{q-1} + \Delta T \cdot \Delta\Phi_q)$
- 12: **end for**

The total number of iterations

The current time

Circuit Design

Injection Signal Generator for SHIL

$$ss_i \Leftarrow -K_s S(2\Phi_{q-1,i})$$

Local Field Calculator

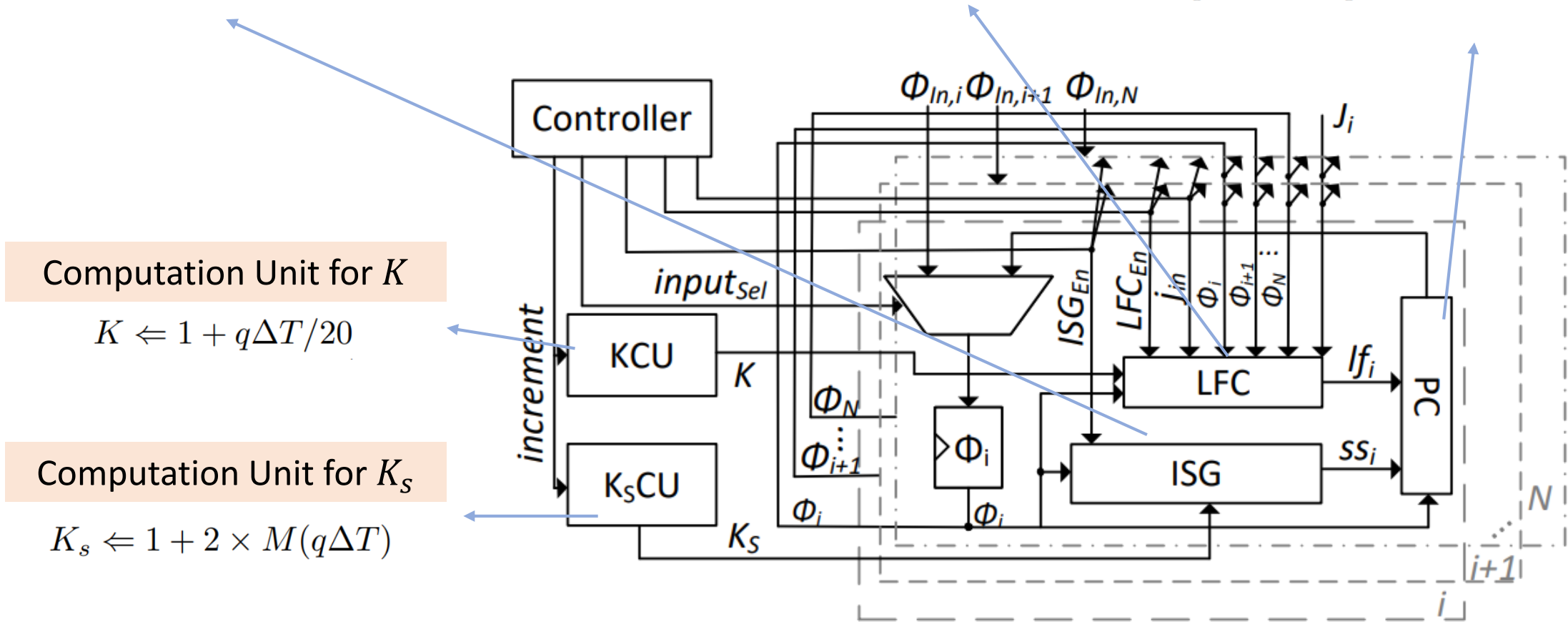
$$pd_i \Leftarrow \Phi_{q-1,i} \cdot ones(1, N) - \Phi_{q-1}$$

$$lf_i \Leftarrow -K(\mathbf{J}_i \cdot C(pd_i))$$

Phase Calculator

$$\Delta\Phi_{q,i} \Leftarrow lf_i + ss_i$$

$$\Phi_q \Leftarrow B(\Phi_{q-1} + \Delta T \cdot \Delta\Phi_q)$$



Computation Unit for K

$$K \Leftarrow 1 + q\Delta T/20$$

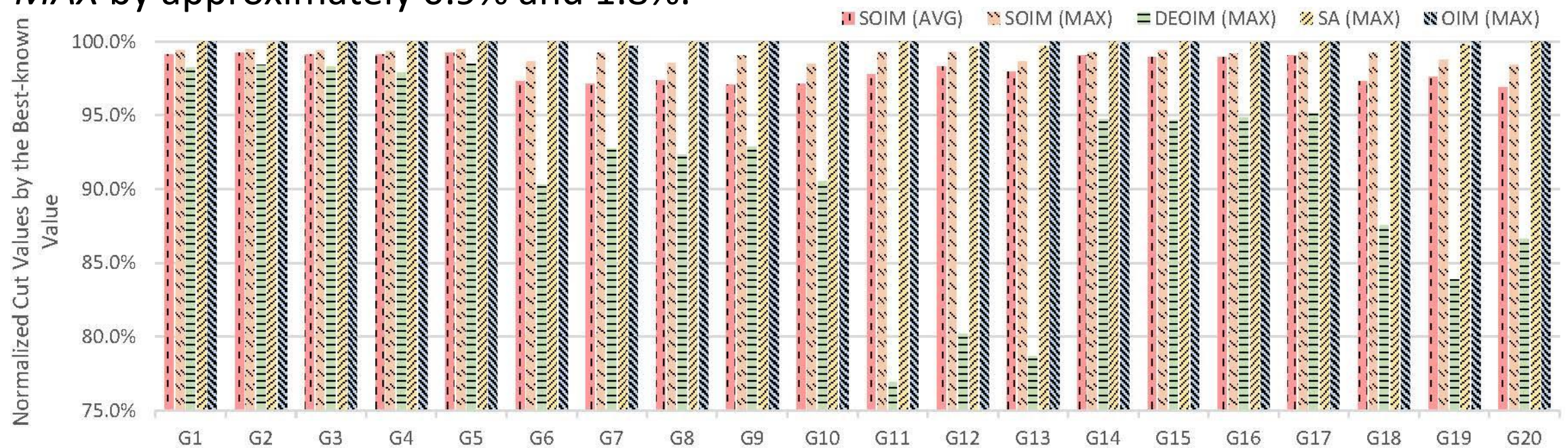
Computation Unit for K_s

$$K_s \Leftarrow 1 + 2 \times M(q\Delta T)$$

The top-level circuit architecture of a simulated oscillator-based Ising machine

Solution Quality

- The proposed SOIM algorithm returns **average values of 99.1% in MAX and 98.2% in AVG**.
- Compared to DEOIM ^[1] : an average 8% increase in MAX.
- Compared to the SA ^[2] and the OIM simulation algorithm ^[4] : a marginal degradation in MAX by approximately 0.9% and 1.8%.



The performance for solving max-cut problems by using different Ising model-based algorithms

[1] T. Wang and J. Roychowdhury, "OIM: Oscillator-based Ising machines for solving combinatorial optimisation problems," in UCNC. Springer, 2019, pp. 232–256

[2] T. G. Myklebust, "Solving maximum cut problems by simulated annealing," arXiv preprint arXiv:1505.03068, 2015.

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

Hardware Performance

- FPGA: a Zynq UltraScale+ ZCU104FPGA board with an XCZU7EVFFVC1156-2-E processor.
- Compared with: FPGA Annealer (FA) ^[5] and simulated bifurcation machine (SBM) ^[6].
- **A fully connected topology**: easy to embed a given COP to the Ising machine.
- **5-bit precision** for the coupling coefficient: a stronger ability in solving complex COPs.

Comparison of FPGA Implementations for Different Ising Machines

Design	Size	Coefficient Precision	Topology	LUTs	FFs	DSPs
SOIM	50	5 bits	Complete ¹	5,706	3,456	50
SOIM	100	5 bits	Complete ¹	11,622	6,928	100
SOIM	500	5 bits	Complete ¹	58,989	34,631	500
FA [5]	1,024	3 level	Sparse ²	468,183	-	-
SBM [6]	2,048	1 bit	Complete ¹	260,953	281,274	104

¹ For the complete topology, all the spins are connected to each other.

² For the sparse topology, not all the spins are coupled together.

[5] C. Yoshimura, M. Hayashi, T. Okuyama, and M. Yamaoka, "FPGA-based annealing processor for Ising model," in CANDAR. IEEE, 2016, pp. 436–442.

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

Outline

- Background and Motivation
- An Approximate Simulated Oscillator Ising Machine
 - Function Approximation
 - Circuit Design for FPGAs
- A Stochastic Simulated Bifurcation Ising Machine
 - Stochastic Simulated Bifurcation Cells
 - Stochastic Computing Circuit Design
- An Approximate Parallel Annealing Ising Machine
 - An Improved Parallel Annealing Algorithm
 - Approximate Circuit Design

Simulated Bifurcation (SB)

- **Basics:** Computationally solve a pair of differential equations related to the momenta and positions of oscillators; **enables massive processing parallelism.**

$$H(\sigma) = -\frac{1}{2} \sum_{i,j} J_{ij} \sigma_i \sigma_j - \sum_i h_i \sigma_i$$

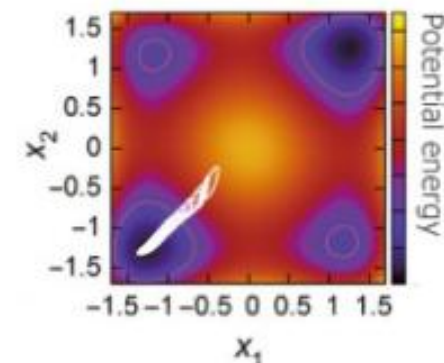
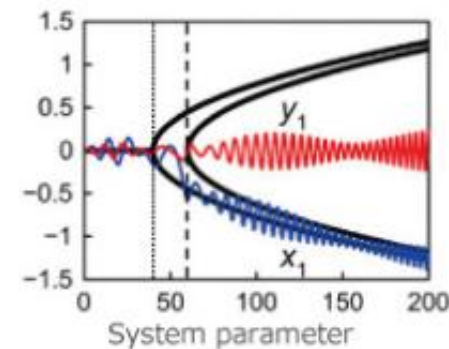
$$\dot{x}_i = \frac{\partial H_{SB}}{\partial y_i} = a_0 y_i,$$

$$\dot{y}_i = \frac{\partial H_{SB}}{\partial x_i} = -\{a_0 - a(t)\}x_i + c_0 p_i + \eta(t)h_i$$

x_i is replaced with its sign and y_i is initialized to 0 if $|x_i| > 1$.

x_i and y_i : the position and the momentum of the i th oscillator (seen as the i th spin);
 $\dot{x}_i$ and $\dot{y}_i$ denote time derivatives;
 $a(t)$ and $\eta(t)$ are control parameters; a_0 and c_0 are positive constants.

- **Bad news:** Solving differential equations is not easy, especially when the matrices are **large (i.e., compute-intensive)**.



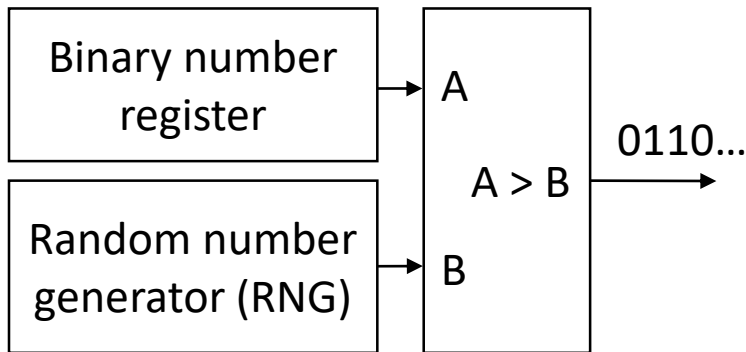
Hamiltonian convergence of Bifurcation Ising machines.

Stochastic Computing (SC)

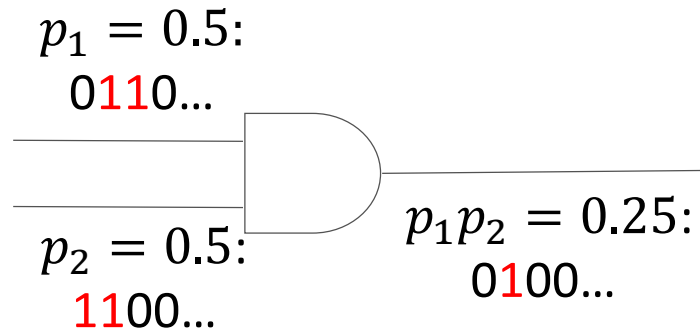
- In SC, values are represented and processed as random bit streams of 0s and 1s; simple logic gates/counters can perform arithmetic operations.



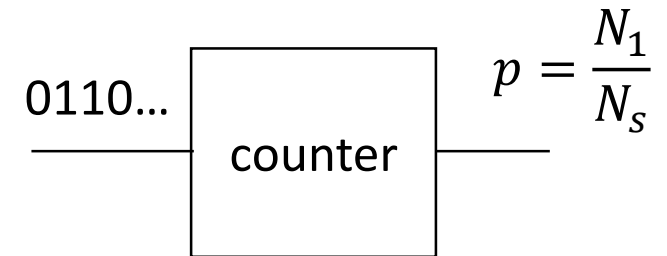
A stochastic computing system.



An SNG.



A unipolar stochastic multiplier.



A probability estimator.

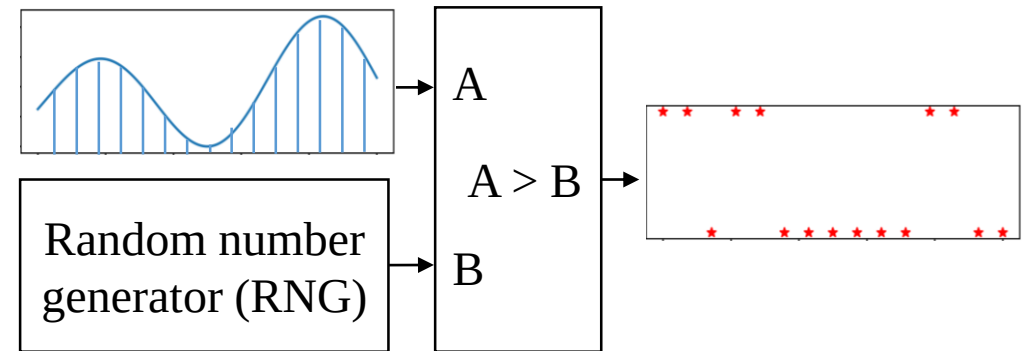
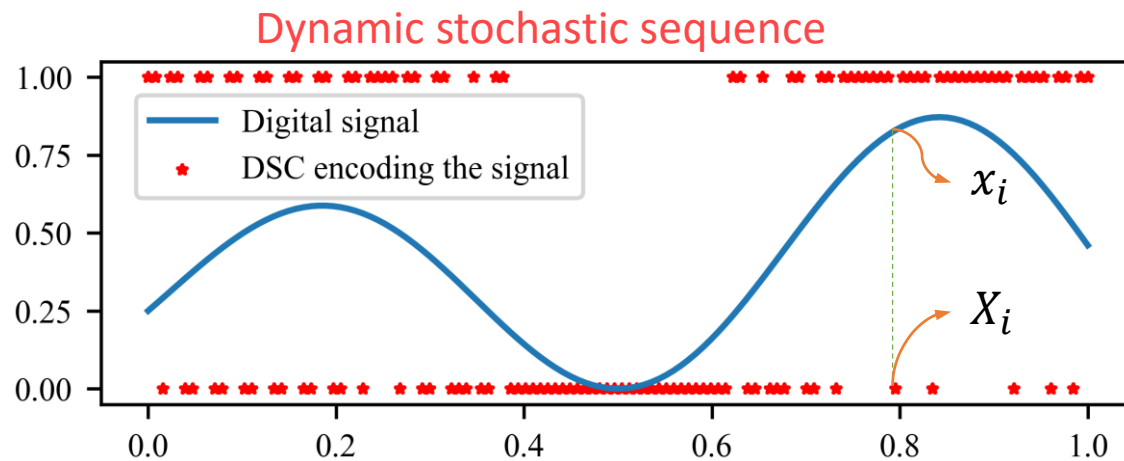
N_1 : the number of 1s.

N_s : the number of all bits.

Dynamic Stochastic Computing (DSC)

- **Good news:** In DSC, signals are sampled as random bit streams of 0s and 1s; each bit encodes a (changing) value or probability of the signal.

Specifically, we use **dynamic stochastic sequences (DSS's)** in DSC.



Dynamic Stochastic Computing (Cont'd)

- **Good news:** In DSC, signals are sampled as random bit streams of 0s and 1s; each bit encodes a (changing) value or probability of the signal.

Ordinary differential equations (ODEs)

In our previous work (DAC'17), DSC was used to solve ODEs.

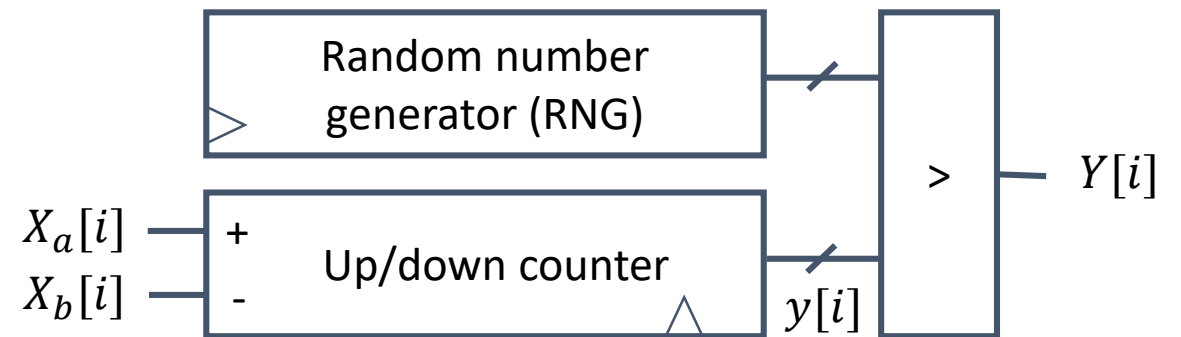
$$\frac{dy(t)}{dt} = f(t)$$

Euler method $\hat{y}_i \approx \sum f_i$



Instead $\hat{y}_i \approx \sum F_i$ F_i : DSS encoding $f(t)$

A Sequential Stochastic Integrator (SSI):



$$y[i] = y(t)|_{t=hi} \approx \sum [x_a(t) - x_a(t)]$$

Integration in Simulated Bifurcation

- **Differential Equations** ^[7] :

$$\begin{aligned} \dot{x}_{i,t} &= f(\mathbf{y}_t)_i = a_0 y_{i,t} \\ \dot{y}_{i,t} &= g(\mathbf{x}_t)_i = -\{a_0 - a(t)\}x_{i,t} + c_0 J X_{i,t} \end{aligned}$$

- **Semi-implicit Euler Integration** ^[7] :

$$\begin{aligned} y_{i,t+1} &= y_{i,t} + \eta g(\mathbf{x}_t)_i \\ x_{i,t+1} &= x_{i,t} + \eta f(\mathbf{y}_{t+1})_i \end{aligned}$$

Iterative Accumulation



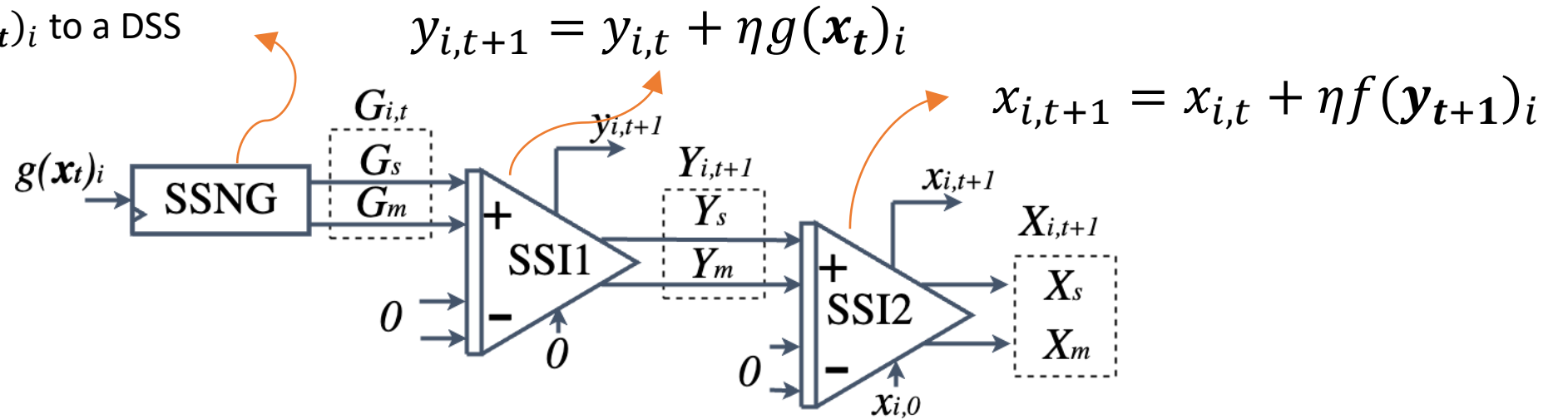
$$y_{i,0} = 0, a_0 = 1, f(\mathbf{y}_t)_i = y_{i,t}$$

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

A Stochastic Computing SB Cell

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

Convert a binary $g(x_t)_i$ to a DSS

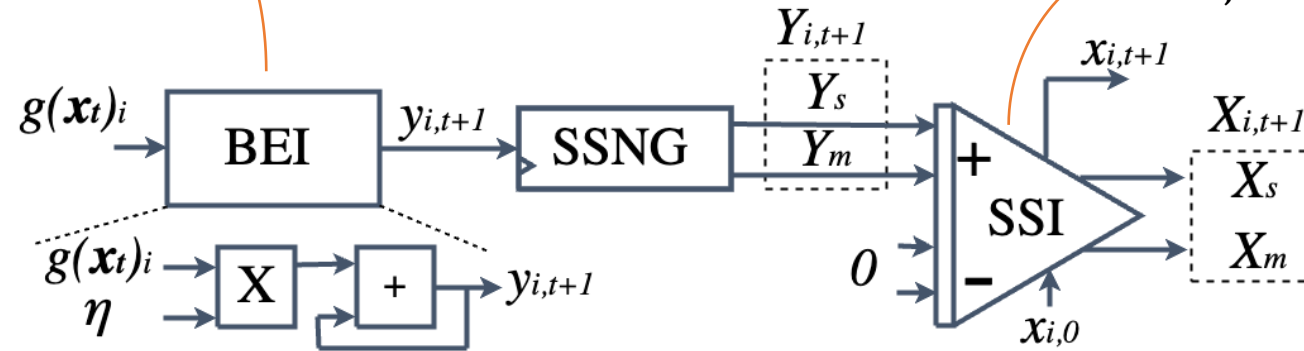


The Stochastic Computing SB Cell (SC-SBC):
Aimed for higher area efficiency.
(SSI: Sequential Stochastic Integrator)

A Binary-Stochastic Computing SB Cell

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

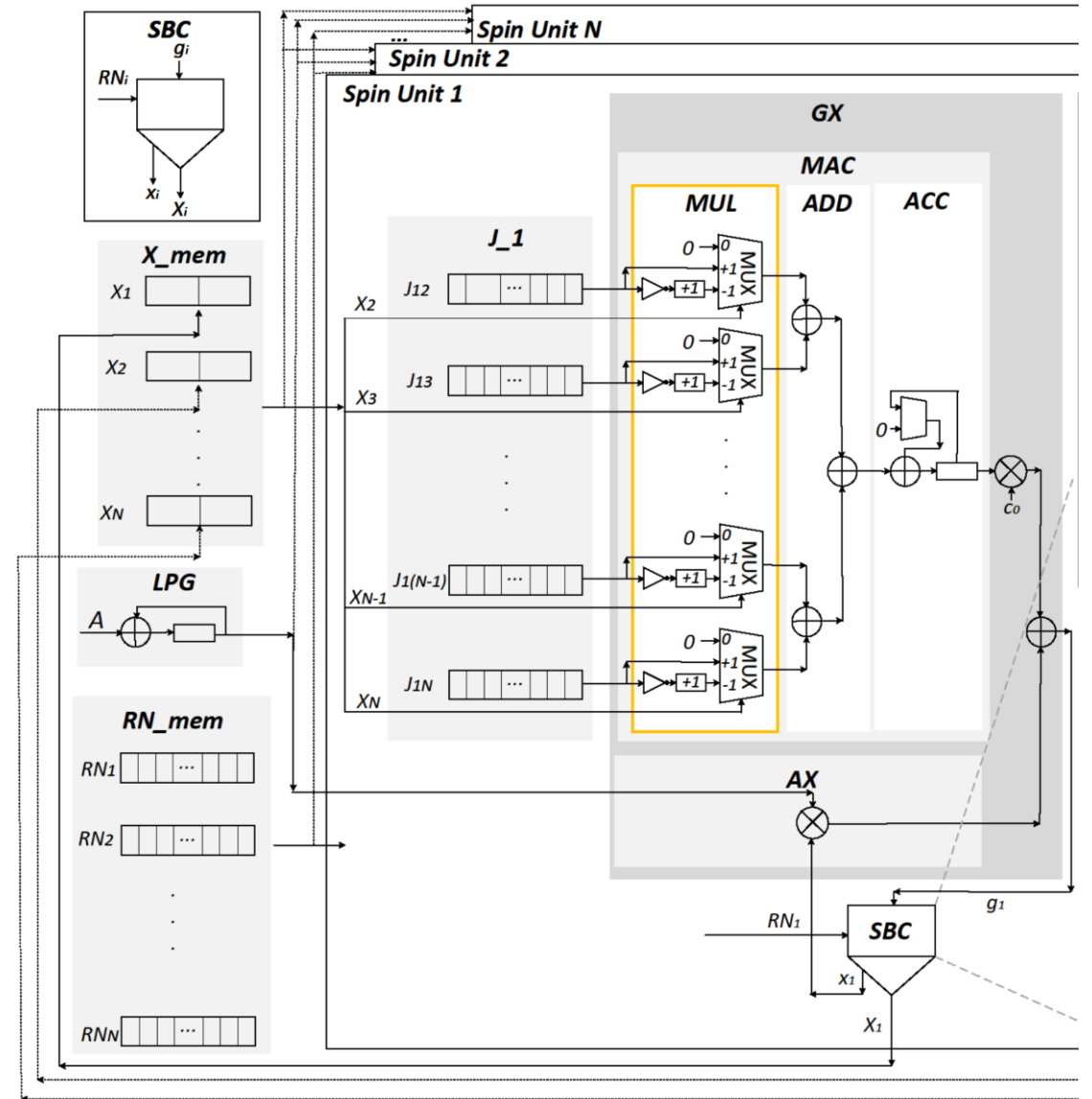
$$y_{i,t+1} = y_{i,t} + \eta g(\mathbf{x}_t)_i \quad \text{and} \quad x_{i,t+1} = x_{i,t} + \eta f(\mathbf{y}_{t+1})_i$$



The Binary-Stochastic Computing SB Cell (BSC-SBC):
 Aimed for higher speed.
 (BEI: binary Euler integrator)

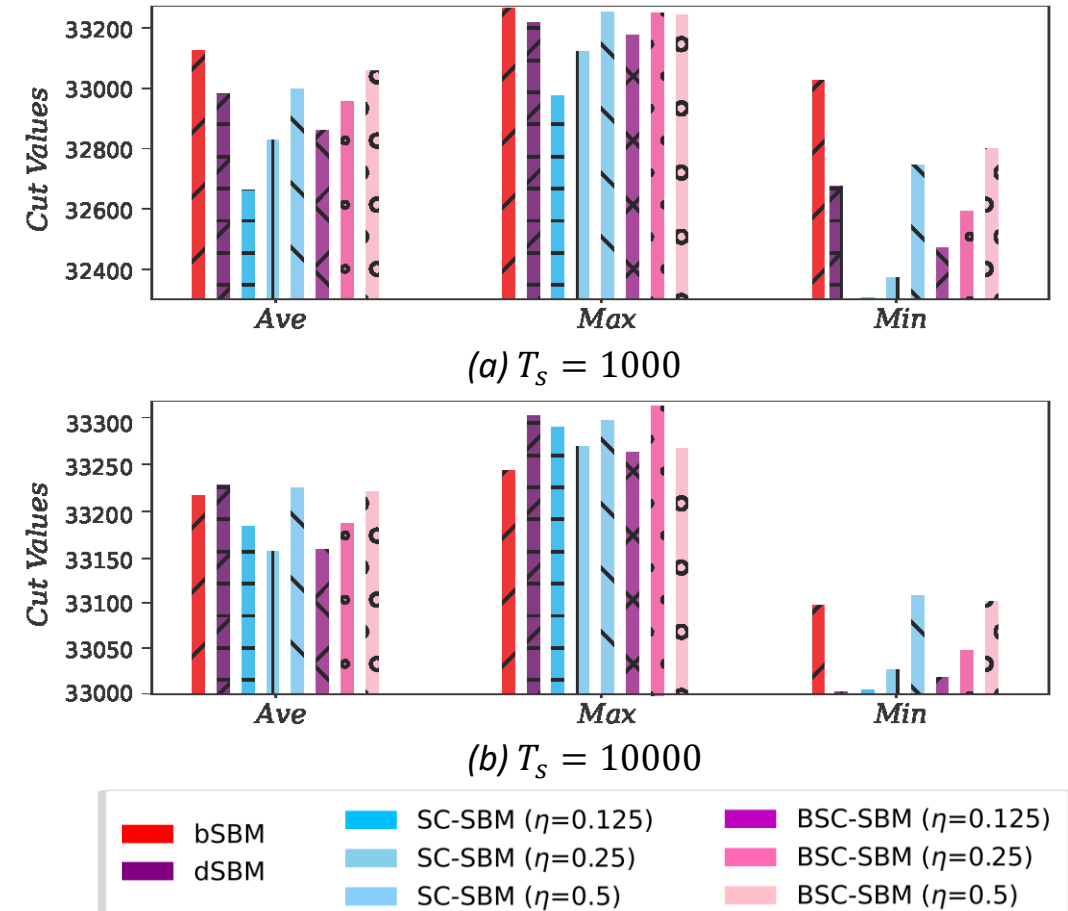
Stochastic Simulated Bifurcation Machine (SSBM)

- **A linearly increasing parameter generator (LPG):** to implement $-\{a_0 - a(t)\}$ as a linearly increasing parameter by a pre-calculated value A ;
- **An X_mem :** to store the position values generated from the SBC (either an SC-SBC or a BSC-SBC).;
- **A RN_mem :** to stores N n -bit random numbers for SBCs in the N spin units;
- **N spin units:** to update the position and momentum values by using SBCs.



Performance Evaluation

- The proposed SSBM: higher *Ave* and *Min* values are obtained with $\eta = 0.5$ than with $\eta = 0.125, 0.25$.
- Evaluated by *Ave* and *Min*, when $\eta = 0.5$, the BSC-SBM performs better than the SC-SBM when $T_s = 1000$; the SC-SBM performs better than the BSC-SBM when $T_s = 10000$.
- It shows the advantages of BSC-SBM in a short search, and SC-SBM in a long search.



* bSBM: ballistic simulated bifurcation machine;
dSBM: discrete simulated bifurcation machine.

Hardware Evaluation

- The dense connectivity between spins leads to an increase in area and power.
- The spins in SC-SBM and BSC-SBM require 1.5X and 1.3X more normalized power per spin than [8], respectively, due to the 3.9X larger connectivity.
- The proposed SC-SBM and BSC-SBM utilize at least 10.62% smaller normalized area than [8].

	D-wave [3]	JSSC'15 [14]	JSSC'21 [8]	ISSCC'21 [15]	CICC'21 [16]	JSSC'22 [17]	Prop. SC-SBM	Prop. BSC-SBM
Computing Method	Quantum Annealing	CMOS Annealing	SCA Annealing	Metropolis Annealing	Simulated Annealing	Simulated Annealing	Simulated Bifurcation	Simulated Bifurcation
Technology	Superconductor	65nm CMOS	65nm CMOS	65nm CMOS	65nm CMOS	65nm CMOS	40nm CMOS	40nm CMOS
# Spins	2k	20k	512	16k	252	480	2k	2k
Topology	Chimera	Lattice	Complete	King	King	King	Complete	Complete
# Spin Interactions	5	5	511	8	8	8	1999	1999
Coefficient Bit-Width	N/A	2	5	5	4	4	2	2
Spin Type	Qubit	SRAM	SRAM	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	0.74 mW	0.64 mW
Area per Spin (Normalized Area)	N/A	289 μm^2 (6.86 $\times$)	12207 μm^2 (1.13 $\times$)	552 μm^2 (3.28 $\times$)	1671 μm^2 (12.41 $\times$)	832 μm^2 (6.17 $\times$)	6370 μm^2 (1 $\times$)	6453 μm^2 (1.01 $\times$)
Frequency	N/A	100 MHz	320 MHz	100 MHz	64 MHz	200 MHz	250 MHz	250 MHz
# Spin Update Cycles	N/A	N/A	512	22	N/A	1	20	20

[3] M. W. Johnson et al., "Quantum annealing with manufactured spins," *Nature*, vol.473, no. 7346, pp. 194–198, 2011.

[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*, 2021.

[14] 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.

[15] T. Takemoto et al., "4.6 a 144kb annealing system composed of 9X16kb annealing processor chips with scalable chip-to-chip connections for large-scale combinatorial optimization problems," in *ISSCC*, 2021, pp. 64–66.

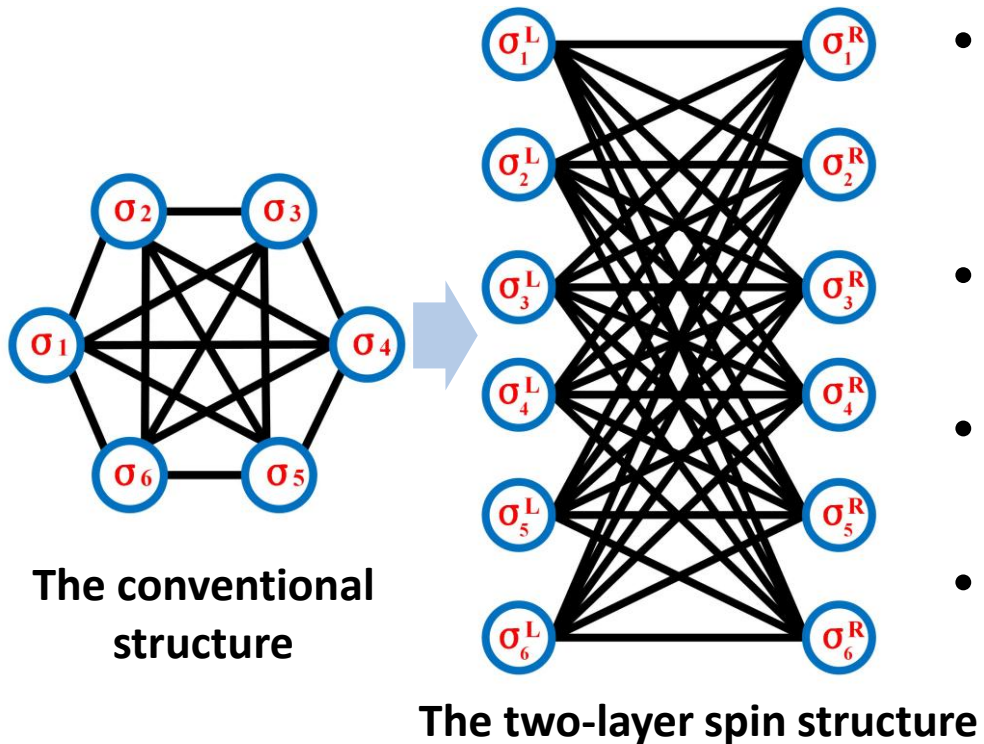
[16] Y. Su et al., "A 252 spins scalable CMOS Ising chip featuring sparse and reconfigurable spin interconnects for combinatorial optimization problems," in *CICC. IEEE*, 2021, pp. 1–2.

[17] 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.

Outline

- Background and Motivation
- An Approximate Simulated Oscillator Ising Machine
 - Function Approximation
 - Circuit Design for FPGAs
- A Stochastic Simulated Bifurcation Ising Machine
 - Stochastic Simulated Bifurcation Cells
 - Stochastic Computing Circuit Design
- An Approximate Parallel Annealing Ising Machine
 - An Improved Parallel Annealing Algorithm
 - Approximate Circuit Design

Parallel Annealing



- Each spin is connected with all the other spins in the other layer but has no interaction with the spins in the same layer.
- The interactions between σ_i^R and σ_i^L are called self-interactions (ω_i).
- All spins in the right and left layers are updated simultaneously per iteration.
- The Hamiltonian (H_P) :

$$H_P = - \sum_{i,j} J_{ij} \sigma_i^L \sigma_j^R - \frac{1}{2} \sum_i h_i (\sigma_i^L + \sigma_i^R) + \omega_i \sum_i (1 - \sigma_i^L \sigma_i^R)$$

Is it efficient for solving constrained combinatorial optimization problems?

Temperature Function

- For conventional parallel annealing algorithm:

$$T_s = \frac{1}{\beta_0 \ln(1 + s)}$$

T_s approaches a value that makes P_i to be low but not zero.

Spins can flip at the end of annealing.

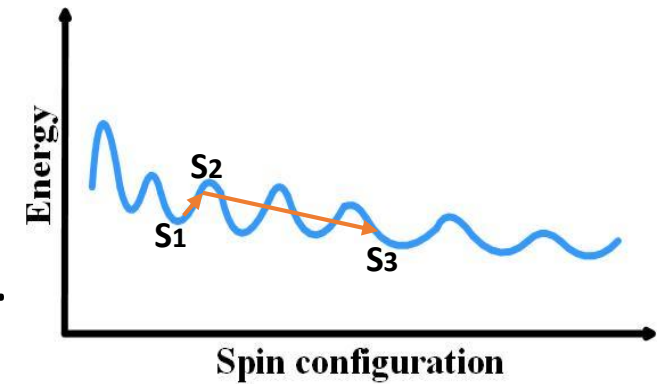
- For the improved parallel annealing algorithm:

$$T_s = T_{init} r^{s-1}$$

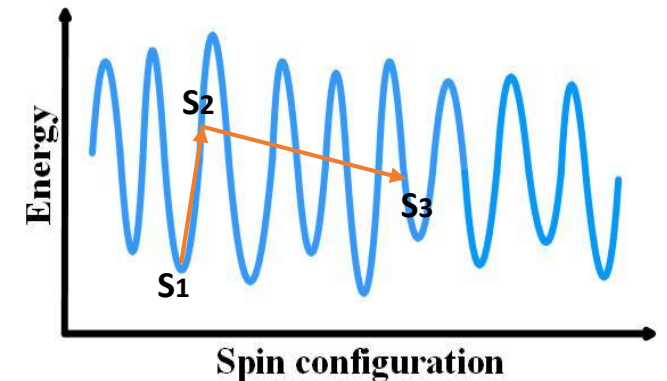
T_s approaches a value that makes P_i to be zero.

Spins cannot flip at the end of annealing.

Max-cut Problems

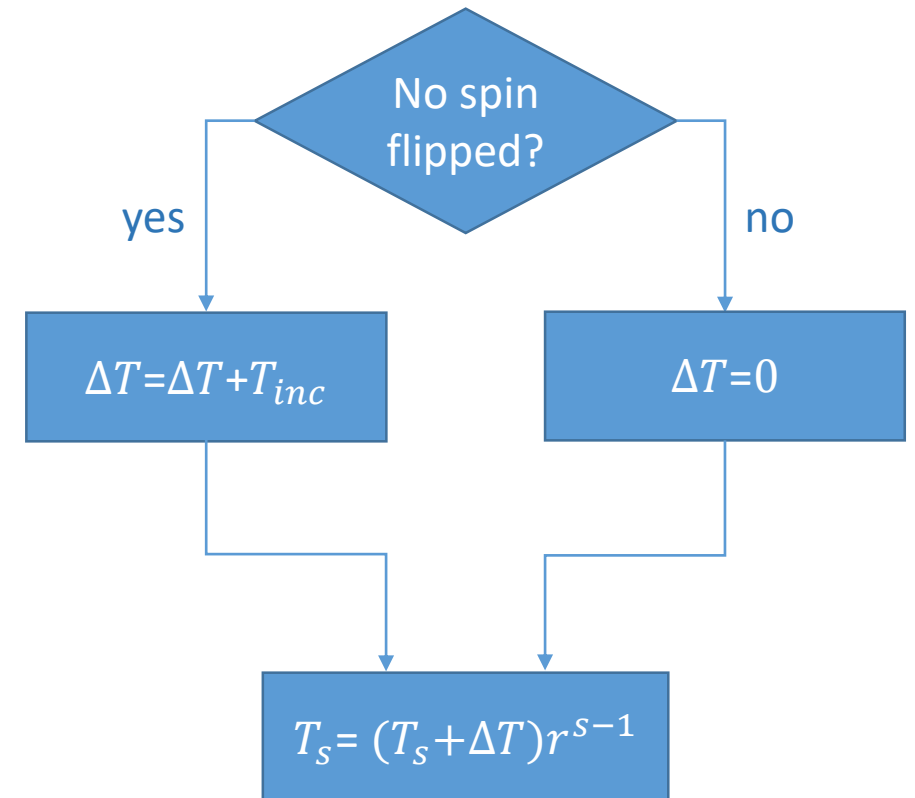


Travelling Salesman Problems

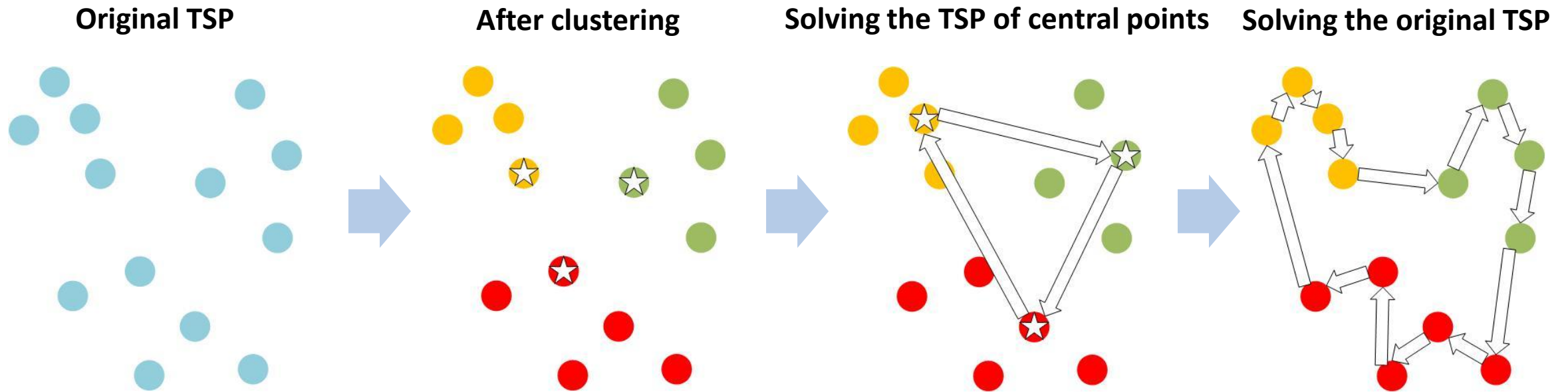


Dynamic Offset

- The Ising model is stuck in a local minimum and hard to escape when T_s is low.
- A dynamic offset ΔT is introduced.
 - T_s increases by ΔT when there is no flip among the spins.
 - ΔT resets to zero when the configuration of spins changes.
- The Ising model can escape from a local minimum quickly and have a higher probability to find a better solution.



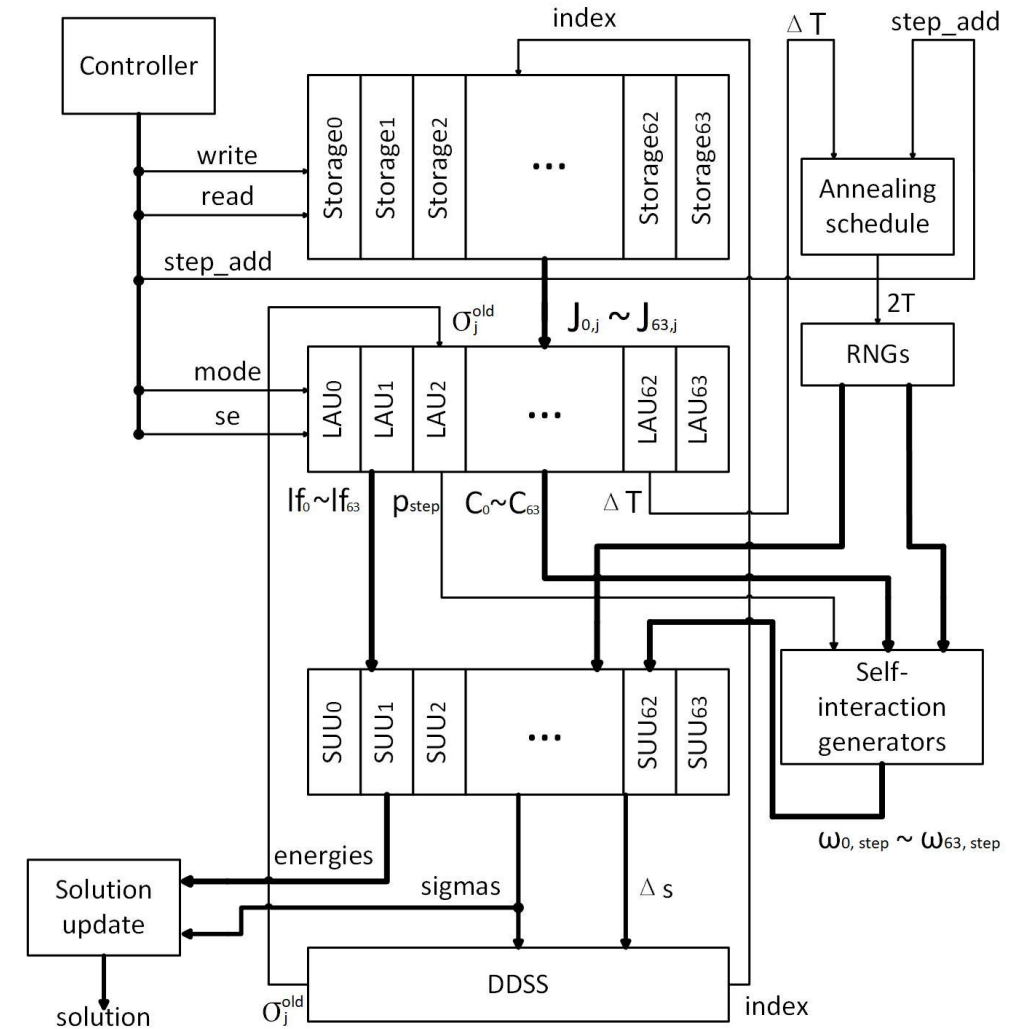
A Clustering Approach



Overall Architecture

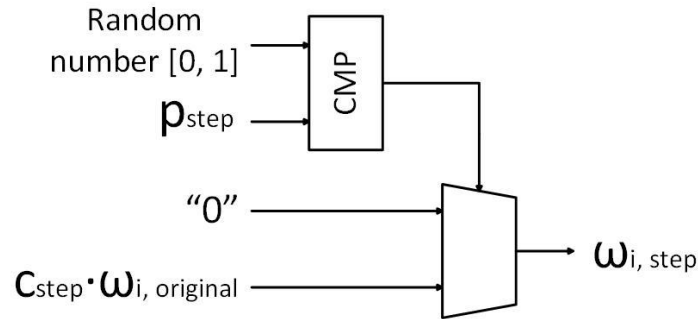
- SUUs update states of all spins according to the initial local fields.
- DDSS outputs the flipped spins to LAUs and corresponding index to storage blocks.
- LAUs calculate new local fields by dropout rate (p_{step}), momentum scaling factor (c_i), and dynamic offset (ΔT) and output to SUUs for the next update cycle.
- Self-interaction generator updates ($\omega_{i,step}$), RNGs generate random numbers between $-2T$ and output to SUUs.
- Solution update unit compares the new energy with the former lowest energy.

The Architecture of the IPA Machine (adapted from JSSC'20)

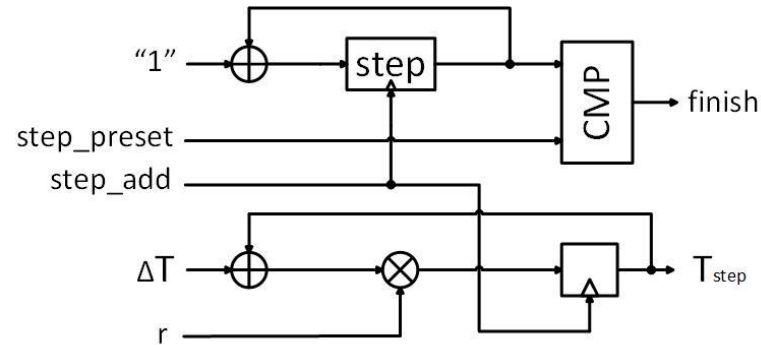


Annealing Circuit Design

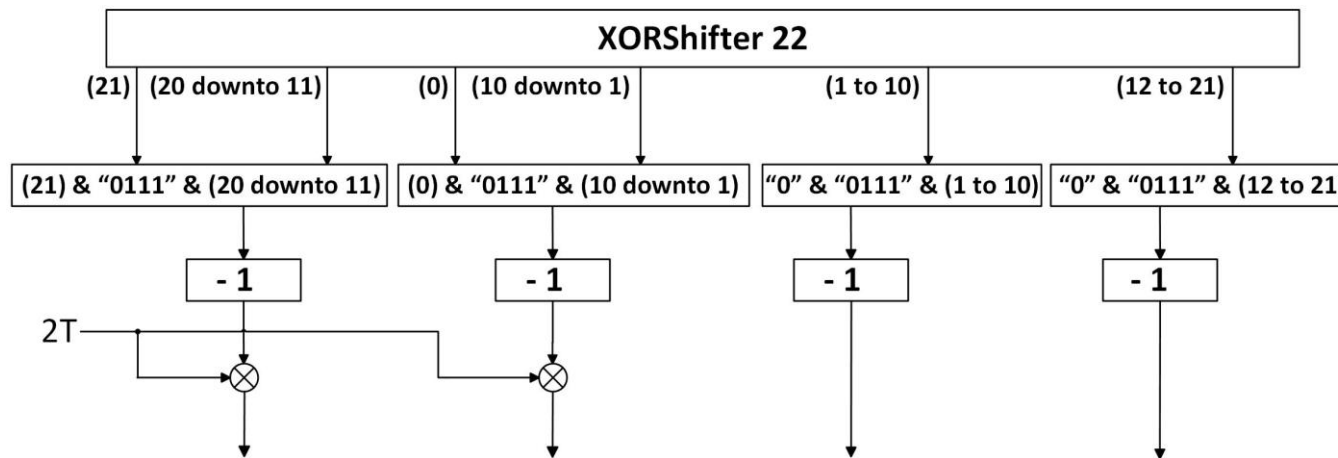
Self-interaction generating unit (SIGU)



Annealing schedule unit (ASU)



Random number generator (RNG)



- The self-interaction ($\omega_{i,step}$) is 0 when $p_{step} > rand$, the rand number is between 0 and 1.
- The T_{step} is calculated by:

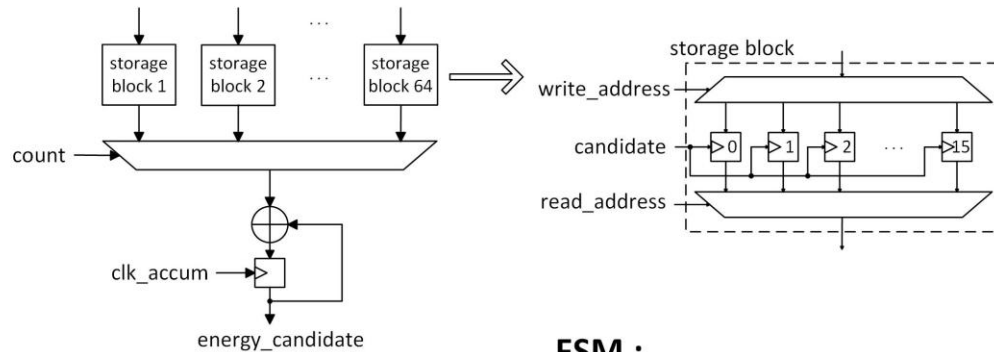
$$T_{step} = (T_{step} + \Delta T) r^{step-1}.$$

- The finish becomes 1 when step reaches step_preset.
- Random numbers are generated by a 22-bit XORshifter.

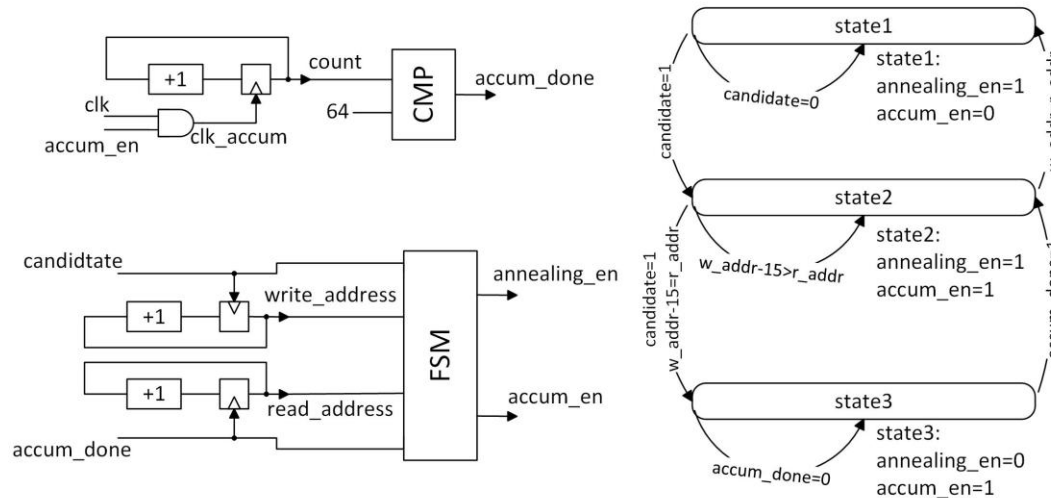
Solution Update Unit

The Energy Calculator

Registers array for energy:

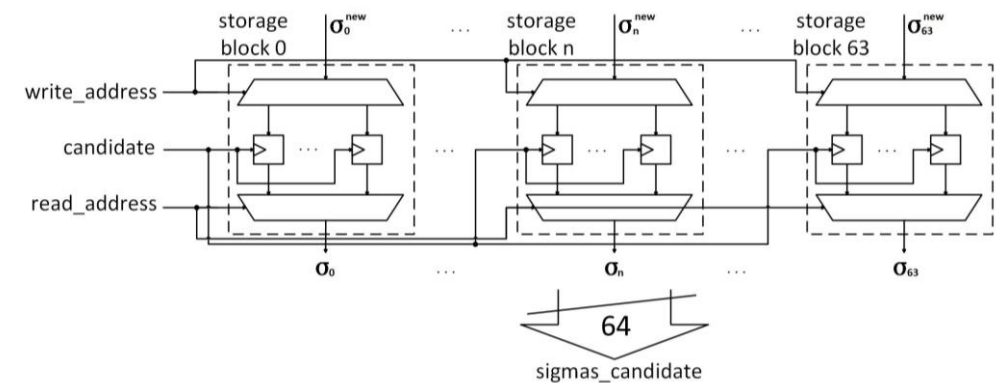


FSM :



- Data that waiting for accumulation is stored in registers arrays.
- The accum_done becomes 1 when 64 values in registers are accumulated.
- Writing process and annealing process need stop when there is no space in the register array.
- The energy_candidate and sigma_candidate are sampled when accum_done becomes 1.

Registers array for sigma:



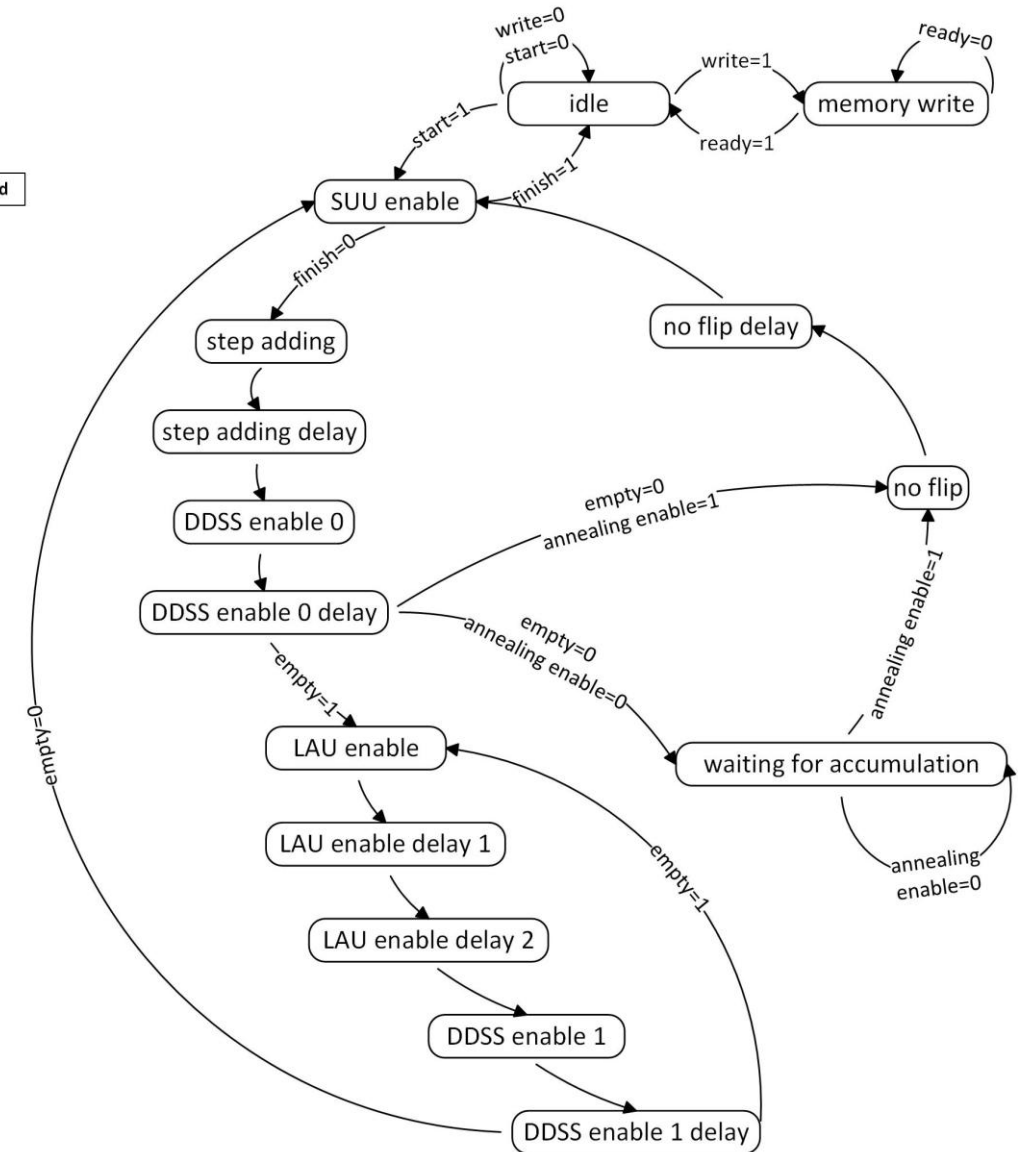
Control Unit

The Instruction

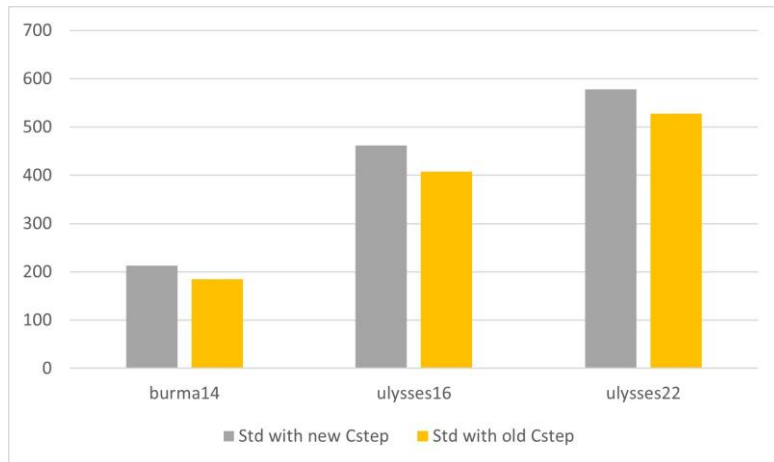
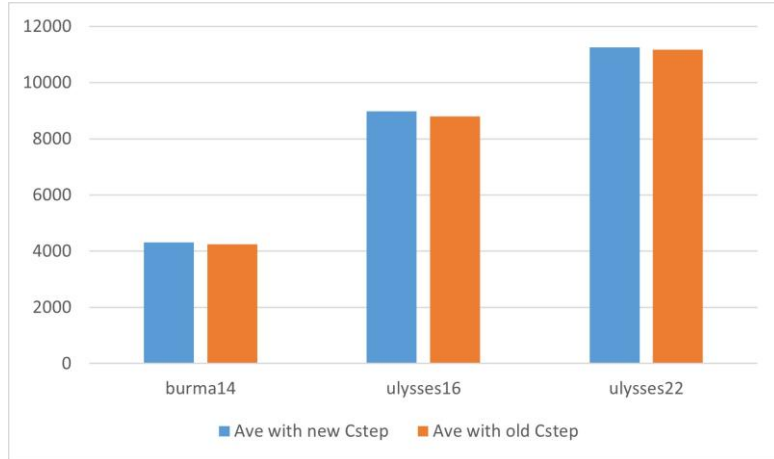
11	10	9	8	7	6	5	4	3	2	1	0
mode	rstdynamic	noflip	C_update	se1	se0	write	read	DDSS_en	LAU_en	SUU_en	step_add

state	instruction
idle	000000000000
Memory write	000000100000
SUU enable	000000000010
Step adding	000001000000
Step adding delay	000001000000
DDSS enable0	000010001000
DDSS enable0 delay	000010001000
LAU enable	110000010000
LAU enable delay1	110000010000
LAU enable delay2	110000010100
DDSS enable1	000000001000
DDSS enable1 delay	000000001000
Waiting accumulation	000000000000
No flip	000011000000
No flip delay	001011000000

State Transition



Approximate Momentum Scaling Factor



The average and standard deviation of travel distance after applying a linear momentum scaling factor.

- The conventional momentum scaling factor (old c_{step}) is:

$$c_{step} = \sqrt{\frac{step}{step_{max}}}.$$

It requires a multiplier to calculate $c_{step} \cdot \omega_i$.

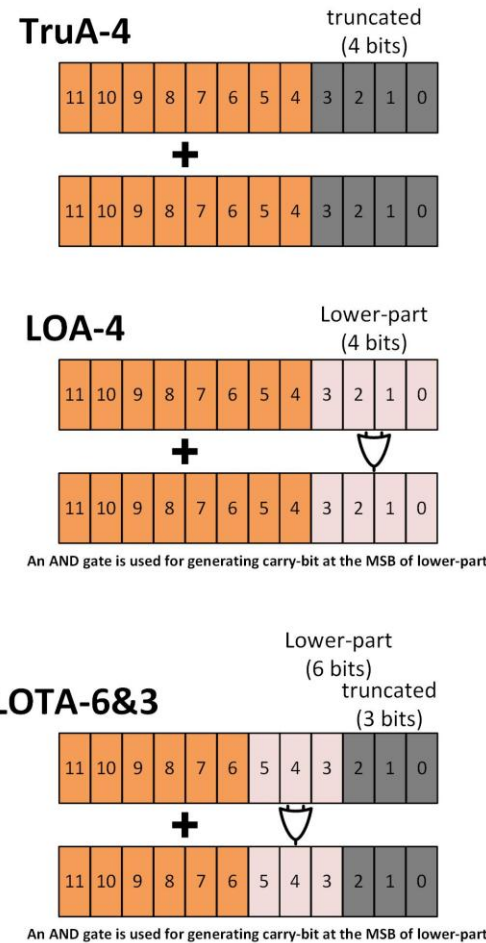
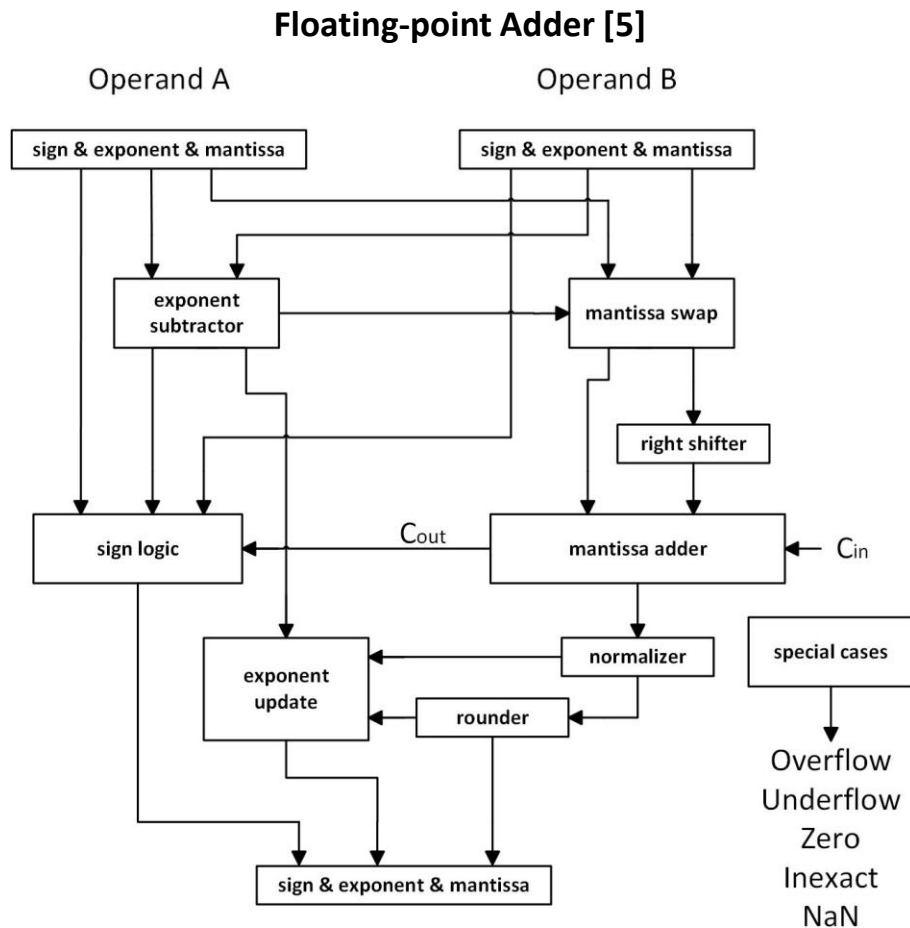
- The **approximate momentum scaling factor** (new c_{step}) is:

$$c_{step} = \frac{step}{step_{max}}.$$

It only requires an adder to calculate $c_{step} \cdot \omega_i$, as $(c_{step} \cdot \omega_i - c_{step-1} \cdot \omega_i)$ is a constant. And the adders in LAUs are reused for $c_{step} \cdot \omega_i$ calculations.

- The **approximation results in a decline in solution quality**: the largest increase on average is 1.9% for ulysses16, and the largest increase in the standard deviation is 15.2% for burma14.

Approximate Floating-point Adder



- Truncated adders (TruA), lower-part-OR adders (LOA), and **lower-part-OR truncated adders (LOTA)** are considered to improve the hardware efficiency.
- The **mantissa adder** in a half-precision floating-point adder (12-bit adder, 10-bit mantissa, 1-bit for hidden 1 and 1-bit for overflow detection) is replaced by the **approximate adders**.

Approximated Floating-point Adder

	Ave	Max	Min	Std	VR (%)	Area (mm ²)	Power (mW)	Delay (ns)
No Approximate	2.67	3.08	2.39	0.17	0	6.300	41.289	3.97
TruA-3	2.69	3.14	2.39	0.18	0	6.275	41.035	3.90
TruA-4	2.74	3.58	2.39	0.24	0	6.266	40.931	3.89
LOA-4	2.71	3.06	2.39	0.18	0	6.273	41.263	3.84
LOA-5	2.77	3.17	2.39	0.21	0	6.266	41.223	3.82
LOA-6	3.08	3.82	2.65	0.26	4	6.259	41.080	3.79
LOTA-4&3	2.71	3.17	2.39	0.20	0	6.269	41.130	3.84
LOTA-5&3	2.72	3.06	2.39	0.21	0	6.262	41.108	3.82

- Simulated results are obtained by synthesizing circuits using the Synopsys Design Compiler.
- The CMOS 28 nm technology is applied with a supply voltage of 1.0 V , temperature of 25°C, and clock frequency of 200 MHz.

Conclusion

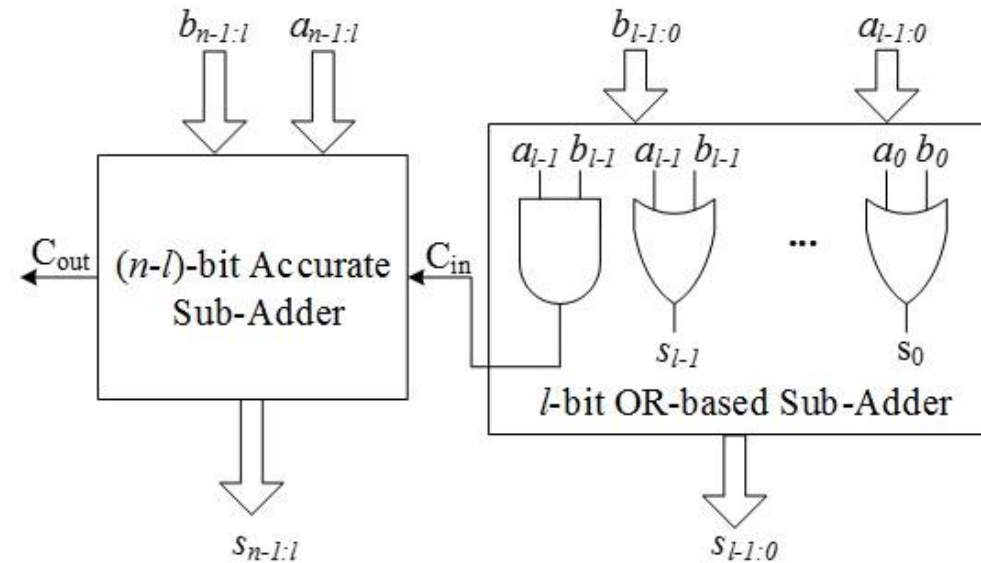
- ▶ Approximate and stochastic computing inherently introduce errors, which can impact the convergence and solution quality of Ising machines.
- ▶ The introduced noise can enhance the system's ability to escape from local minima, such as using approximate functions in the emulated oscillator Ising machine. In the annealing Ising machine using approximate adders, it may cause a loss in solution quality.
- ▶ Different types of Ising machines exhibit varying degrees of error resilience.
- ▶ Ongoing work includes the optimization of approximate circuits and advanced design of stochastic circuits for Ising machines.

Acknowledgment (and Advertisement)

We thank the Natural Sciences and Engineering Research Council (NSERC) of Canada, Alberta Innovates, and the University of Alberta for financial supports.

We are looking for motivated graduate students, including M.Sc. and Ph.D. students, to join our group. If you or your friends/colleagues are interested, please contact jhan8@ualberta.ca for further information.

Lower-part OR Adders (LOAs)



The n -bit lower-part-OR adder (LOA).

Critical path: $O(\log(n - l))$

Circuit area: $A_{adder} + (l \times A_{OR})$

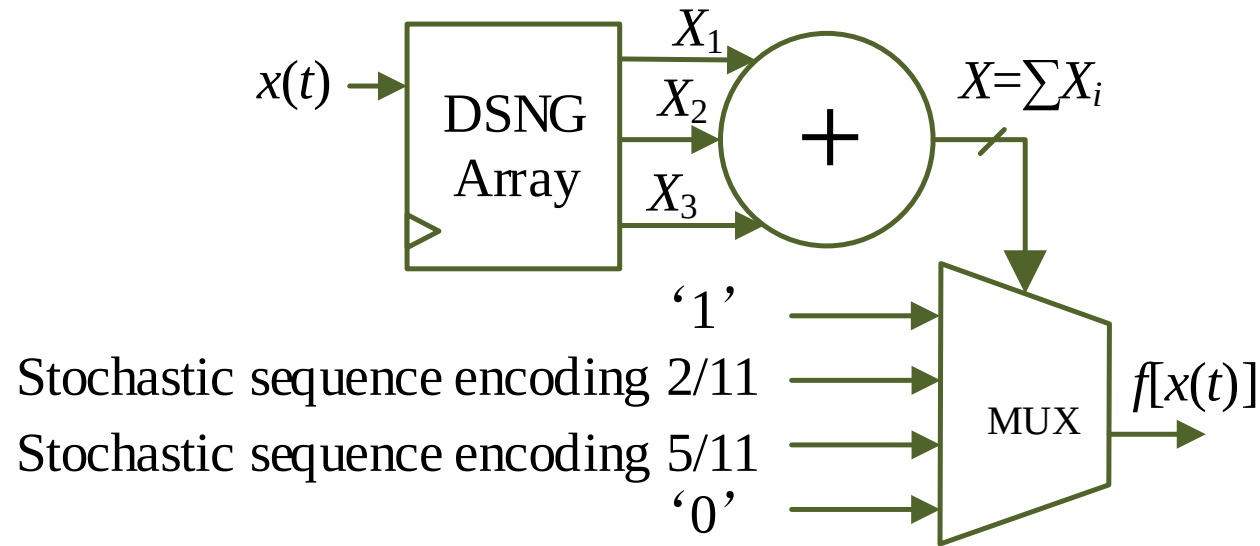
$A_{adder}: O((n - l)\log(n - l))$

A_{OR} : circuit area of the OR gate.

H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas, "Bio-Inspired Imprecise computational Blocks for Efficient VLSI Implementation of Soft-Computing Applications," IEEE Trans. Circuits Syst., 57(4):850-862, 2010.

Stochastic Circuit – Function Approximation

□ Function approximation using a multiplexing circuit



- The multiplexing circuit originally computes $f(x) = 1/11(2x^3 + 3x^2 + 6x^1)$ (Qian et al. 2011).
- If the input sequence encodes $x(t) = e^{-2t}$, the multiplexing circuit calculates the function composition $f[x(t)] = 1/11(2e^{-6t} + 3e^{-4t} + 6e^{-2t})$.

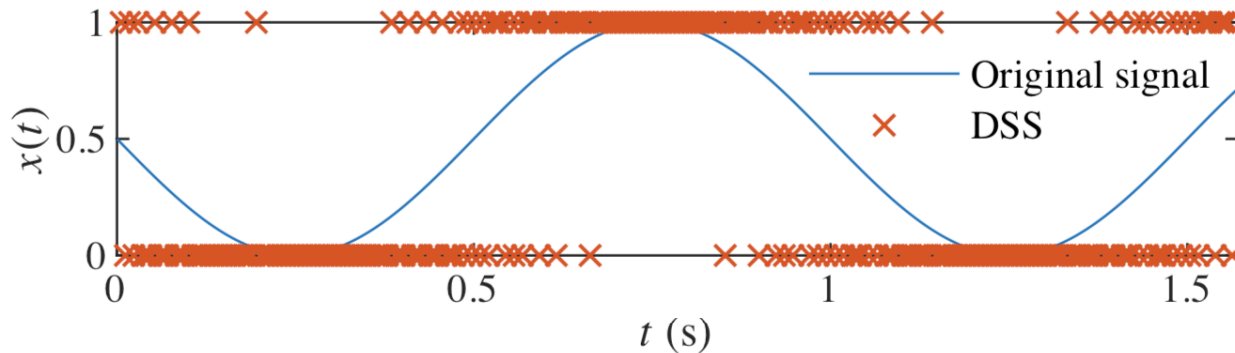
Dynamic Stochastic Computing: Definitions

❑ In dynamic stochastic computing (DSC), a random binary bit stream is used to encode a changing signal instead of a static number as in conventional stochastic computing.

❑ Definition: a dynamic stochastic sequence (DSS):

A DSS $\{A_k\}$ encoding digital signal, y_k , satisfies that the k th bit in the random binary sequence has the expectation,

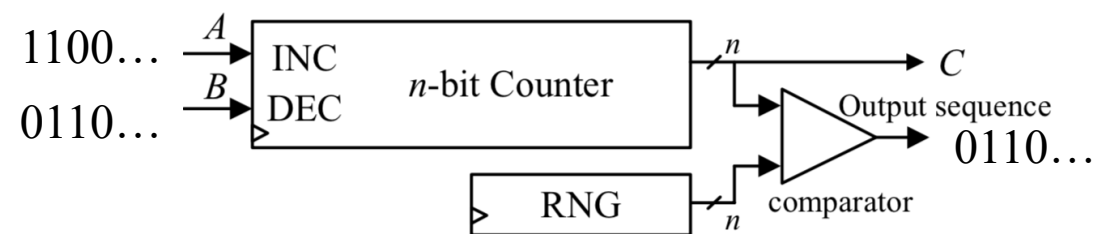
$$\mathbb{E}[A_k] = y_k.$$



A higher frequency of ‘0’ or ‘1’ is observed in the DSS when the signal is close to 0 or 1, and vice versa.

Example: A DSS encoding a 1-Hz sine wave.
(The DSS is decimated for a clear view.)

Stochastic Integrator



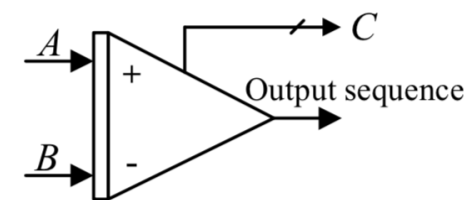
A stochastic integrator.

Function of a stochastic integrator: $c \approx \int (a - b)dt$.
 Let P_i equals to normalized C , i.e., $P_i = \frac{1}{2^n} C$.

An example of stochastic integrator ($n=8$).

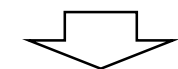
i	a_i	b_i	P_i	RN_i	c_i
0	1	0	$(0.10000000)_2$	0.75	0
1	1	1	$(0.10000001)_2$	0.20	1
2	0	1	$(0.10000001)_2$	0.19	1
3	0	0	$(0.10000000)_2$	0.62	0
.....	...	...		...	...

$\{c_i\}$ encodes signal $\{P_i\}$.



A symbol.

$$P_{i+1} = \begin{cases} P_i + 1/2^n & a_i = 1 \text{ \& } b_i = 0 \\ P_i - 1/2^n & a_i = 0 \text{ \& } b_i = 1 \\ P_i & a_i = b_i \end{cases}$$



$$P_{i+1} = P_i + \frac{1}{2^n} (a_i - b_i). \tag{1}$$

Accumulating (1) for $i = 0, 1, \dots, k - 1$

$$P_k = P_0 + \frac{1}{2^n} \sum_{i=0}^{k-1} (a_i - b_i). \tag{2}$$

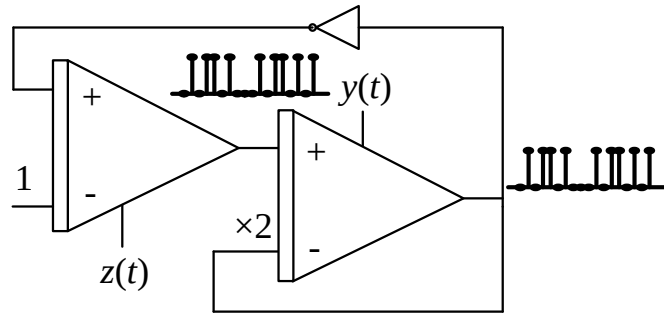
Taking the expectation of (2)

$$\mathbb{E}[P_k] = P_0 + \frac{1}{2^n} \sum_{i=0}^{k-1} (\mathbb{E}[a_i] - \mathbb{E}[b_i]). \tag{3}$$

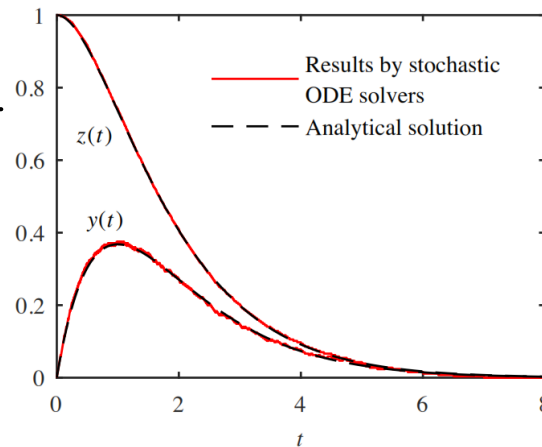
Hardware ODE Solvers using DSC

❑ Numerical solution of ordinary differential equations (ODEs) using the Euler method.

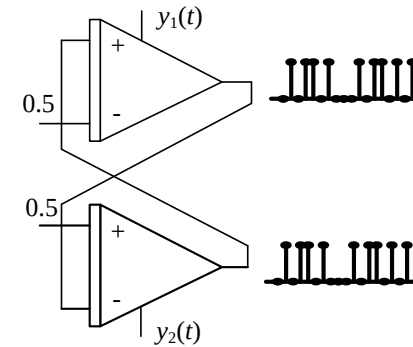
❑ A second-order ODE



$$\frac{d^2y(t)}{dt} + \frac{2dy(t)}{dt} + y(t) = 0$$

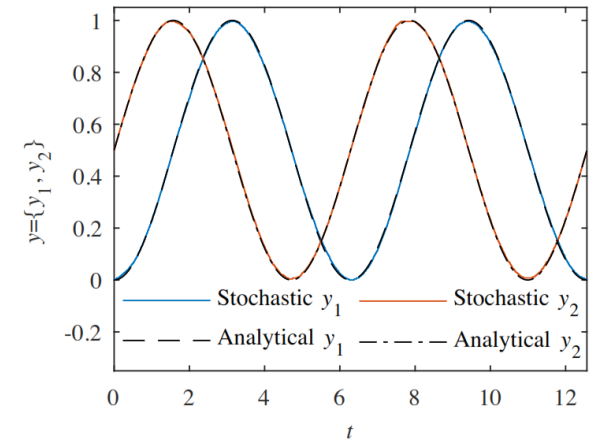


❑ A system of ODEs



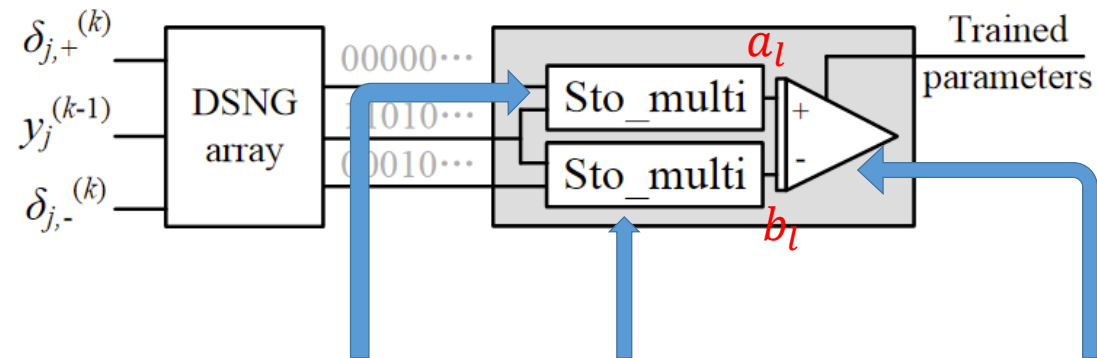
$$\frac{dy_1(t)}{dt} = y_2(t) - 0.5$$

$$\frac{dy_2(t)}{dt} = 0.5 - y_1(t)$$



Hardware solution produced by stochastic ODE solver vs. analytical solution.

DSC-based Gradient Descent Circuit Design



$$\Delta w_{i,j}^{(k-1)} = \eta \left(\delta_{j,+}^{(k)} - \delta_{j,-}^{(k)} \right) y_j^{(k-1)} = \eta \left(\delta_{j,+}^{(k)} y_j^{(k-1)} - \delta_{j,-}^{(k)} y_j^{(k-1)} \right) \quad w_{i,j}^{(k-1)} = w_{i,j}^{(k-1)}[0] + \sum \Delta w_{i,j}^{(k-1)}$$

Recall:

$$\mathbb{E}[P_L] = P_0 + \frac{1}{2^n} \sum_{l=0}^{L-1} (\mathbb{E}[a_l] - \mathbb{E}[b_l]).$$

l : the l th sample in the training data.

n : bit width of the stochastic integrator (up/down counter).

$$\mathbb{E}[P_L] = w_{i,j}^{(k-1)}[L], \text{ when } \eta = \frac{1}{2^n}, P_0 = w_{i,j}^{(k-1)}[0].$$

$$\begin{aligned} \text{Sto_multi1: } \mathbb{E}[a_l] &= \delta_{j,+}^{(k)}[l] y_j^{(k-1)}[l] \\ \text{Sto_multi2: } \mathbb{E}[b_l] &= \delta_{j,-}^{(k)}[l] y_j^{(k-1)}[l] \end{aligned} \quad (\text{DSC})$$

The proposed DSC-based gradient descent circuit provides unbiased estimate of the optimized weights.

Experiments on MNIST Dataset

- The weights of a 784-128-128-10 fully connected neural network (NN) is updated by the proposed DSC-based gradient descent circuit. The NN is used for handwritten digit recognition of MNIST dataset.

