DESIGN, AUTOMATION & TEST IN EUROPE
09 – 13 March 2020 · ALPEXPO · Grenoble · France
The European Event for Electronic System Design & Test

Introduction to Dynamic Stochastic Computing

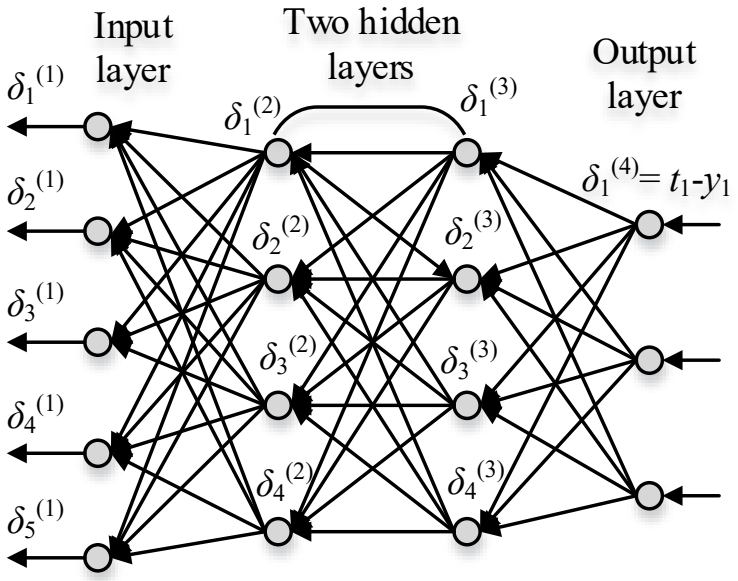
Siting Liu and Jie Han
Department of Electrical and Computer Engineering
University of Alberta
Edmonton, AB, Canada

https://ieeexplore.ieee.org/document/9165977

Outline

Motivation

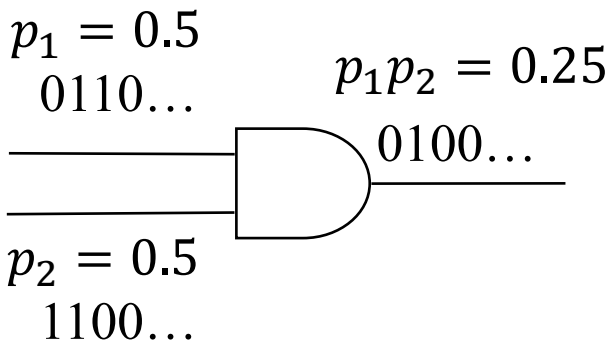
- With the rapid development in machine learning, a large computational load is imposed on a computing system as the learning models become more and more complex. However, the demand for machine learning-based mobile application is increasing, which requires low-cost and energy-efficient devices.



Signal flow of training in a simple deep learning model

Conventional computing → Stochastic computing (SC)

- Low power, small area



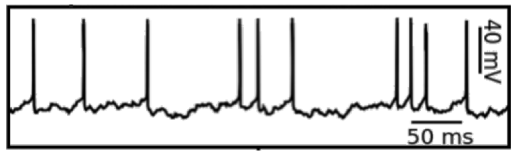
A unipolar stochastic multiplier.

- Biologically plausible

A stochastic sequence:

0101001000…

A spike train:

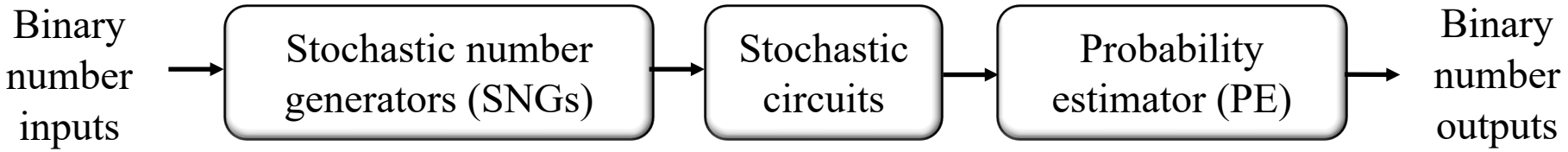


[Hayes 2015]

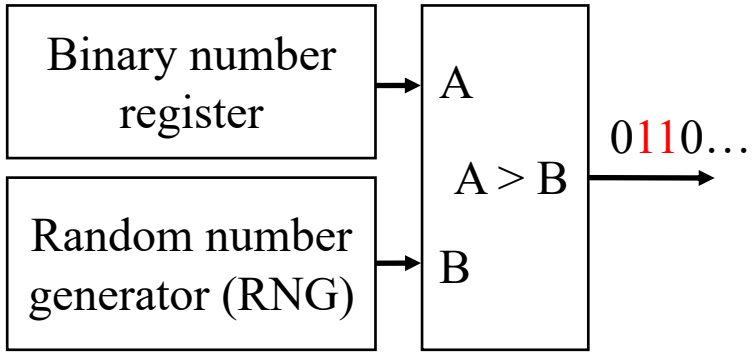
Time-consuming, high energy & hardware cost.

Background: stochastic computing

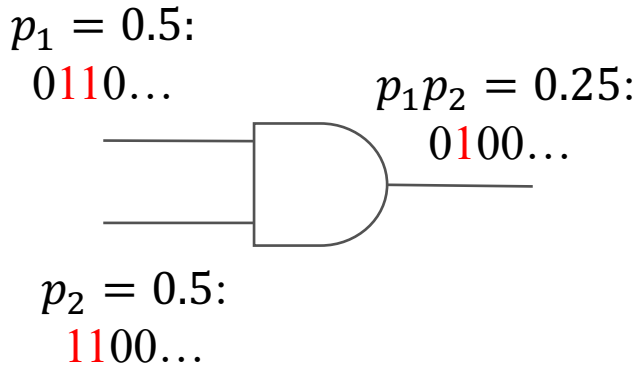
- In stochastic computing (SC), information is encoded and processed by random binary bit streams.



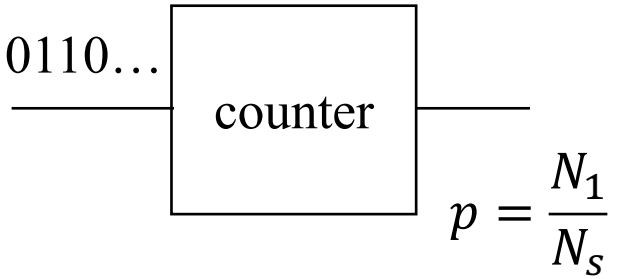
A stochastic computing system.



An SNG.



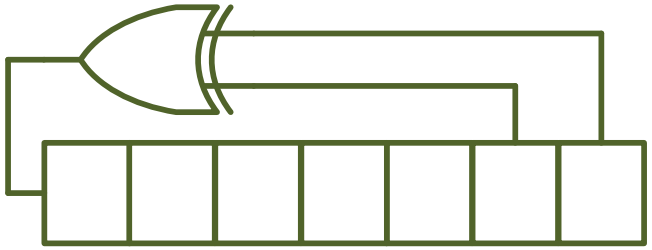
A unipolar stochastic multiplier.



A probability estimator.

Limitations of Stochastic Computing

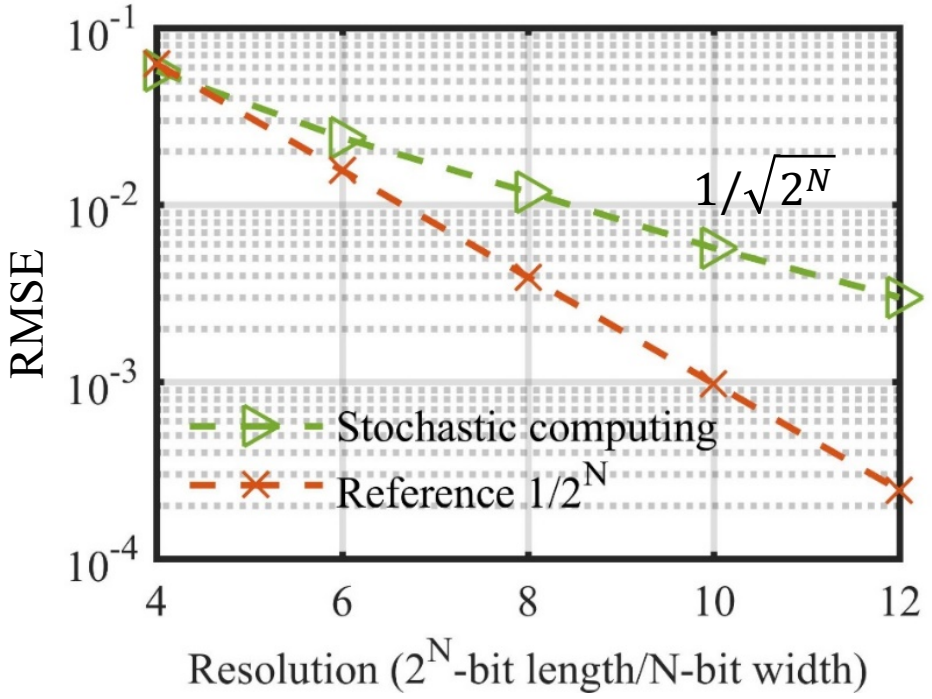
- Low accuracy
  - Random fluctuations



Linear feedback shift register (LFSR) as a random number generator (RNG).

- Potential solution: dynamic stochastic computing (DSC)
  - An extreme case: 1-bit SC.
  - Reduce sequence length without harming the accuracy

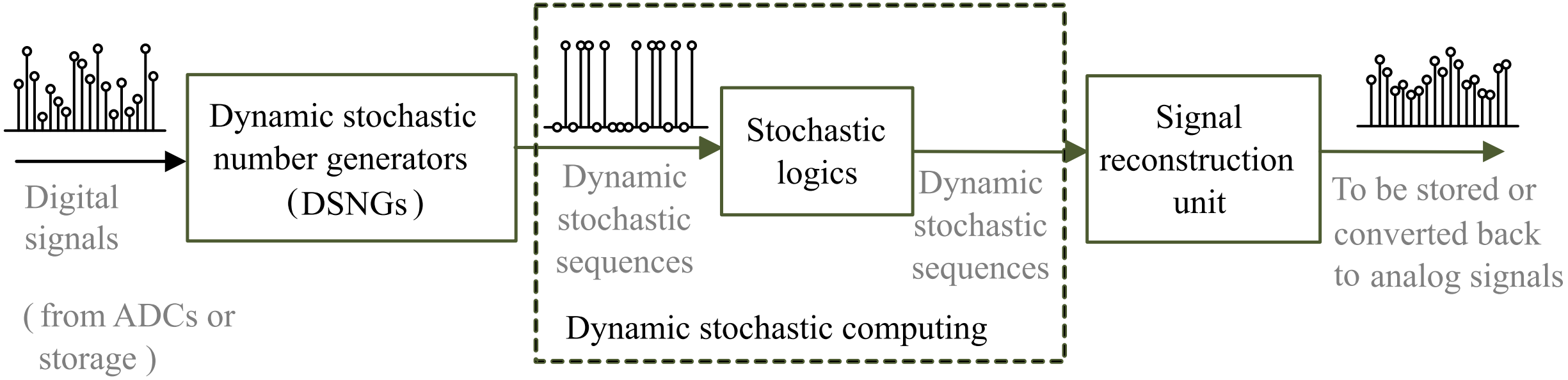
- High latency



Accuracy of a unipolar stochastic multiplier using pseudorandom sequences.

Dynamic Stochastic Computing Systems

- In a dynamic stochastic computing (DSC) system, the stochastic sequence can encode a consistently varying signal instead of a static number as in conventional stochastic computing systems.



A dynamic stochastic computing (DSC) system.

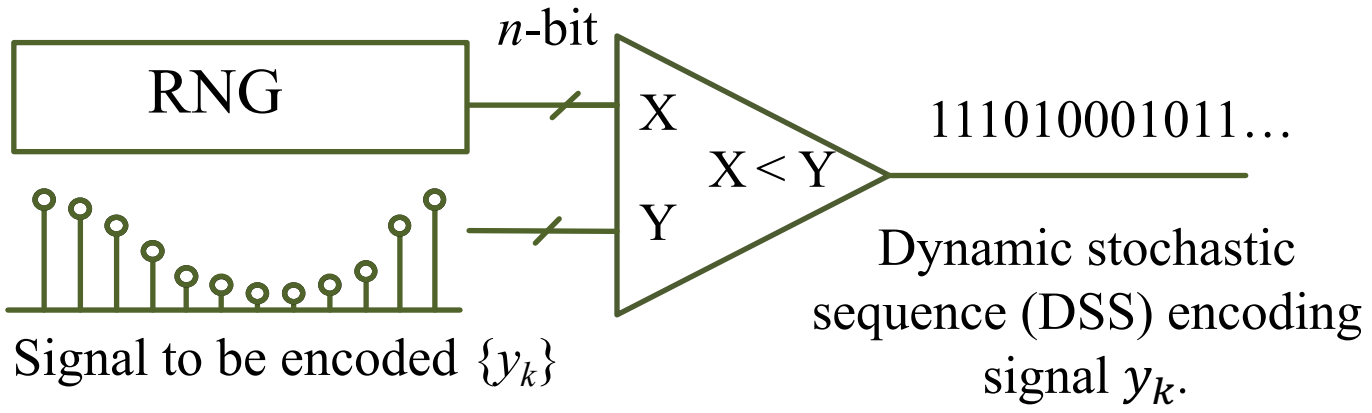
Generation of Dynamic Stochastic Sequence

- 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 .

- Definition: dynamic stochastic sequence (DSS):

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

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



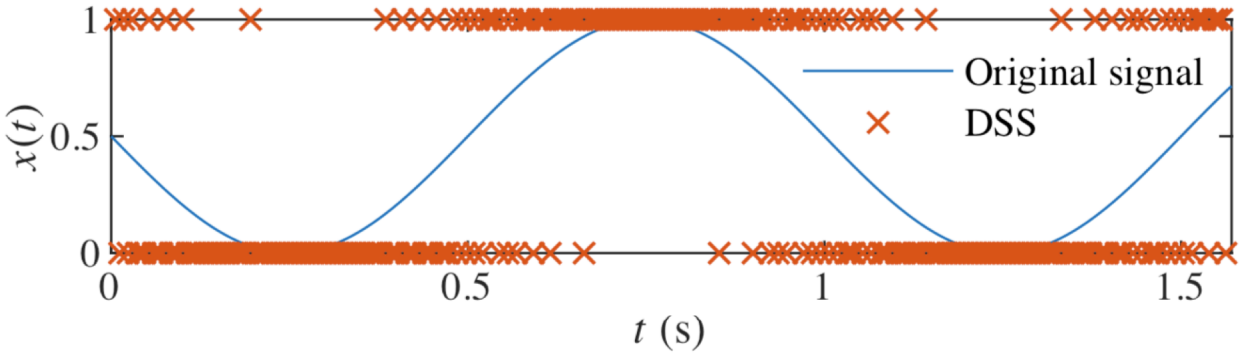
- A dynamic SNG (DSNG) can be realized by comparing each sampling point with a uniform random number generated by the RNG.

An example of a DSS

- Dynamic stochastic sequence (DSS):

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

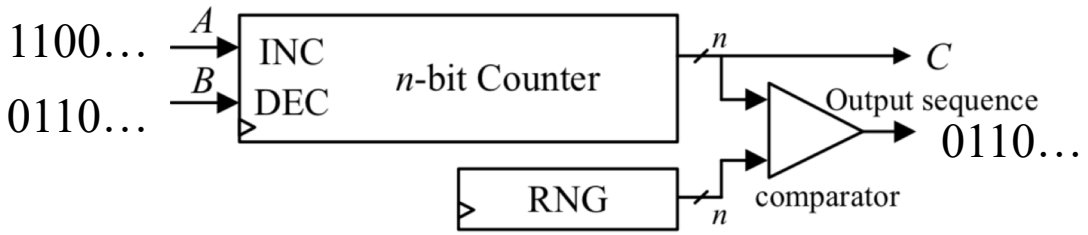
$$\mathbb{E}[A_k] = y_k.$$



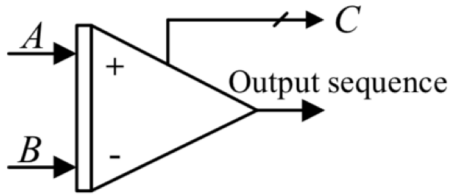
A DSS encoding a 1-Hz sine wave. The DSS is decimated for a clear view.

A higher frequency of '1' is observed in the DSS when the signal is close to 1 and vice versa.

Stochastic Integrator



A stochastic integrator.



A symbol.

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 $\{P_i\}$.

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

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

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

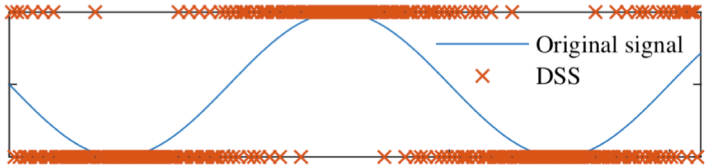
$$P_k = P_0 + \frac{1}{2^n}\sum_{i=0}^{k-1}(a_i - b_i)\,. \quad (2)$$

Taking the expectation of (2)

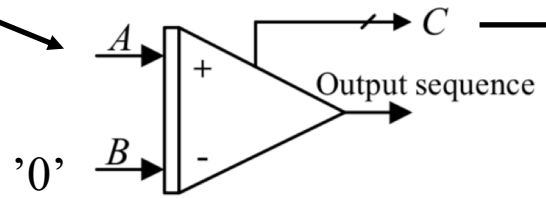
$$\mathbb{E}[P_k] = P_0 + \frac{1}{2^n}\sum_{i=0}^{k-1}(\mathbb{E}[a_i] - \mathbb{E}[b_i]). \quad (3)$$

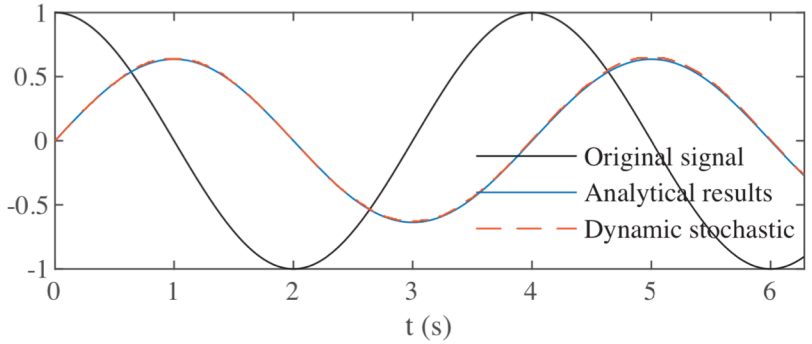
Stochastic integrator for iterative accumulations

❑ Numerical integration by performing a Riemann sum



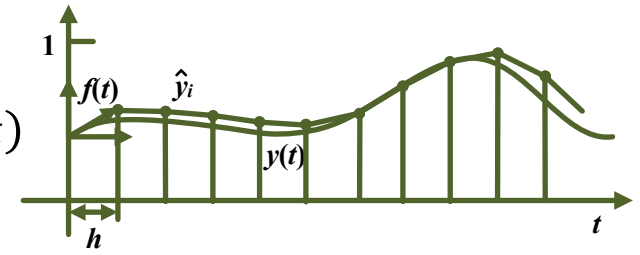
*A segment of the signal is shown with decimation for a clear view.





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

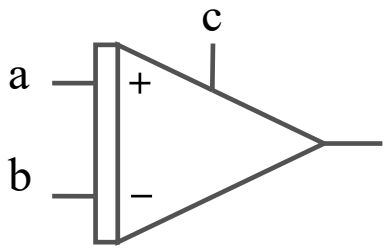
Euler Method: $\frac{dy(t)}{dt} = f(t)$



$$\hat{y}_k = y_0 + h\sum_{i=0}^{k-1} f(hi) \approx y(hk).$$

A stochastic integrator provides an unbiased estimate to the Euler solution with a step size of $1/2^n$.

Stochastic Integrator:



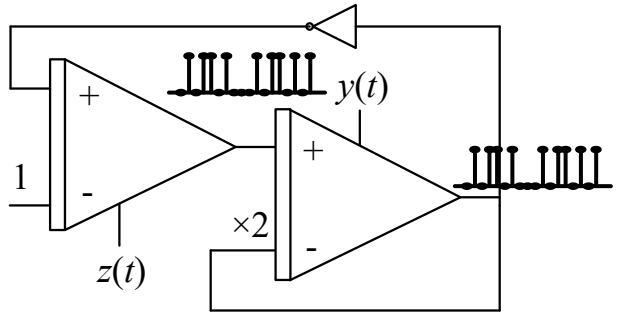
$\mathbb{E}[c_k] = c_0 + \frac{1}{2^N}\sum_{i=0}^{k-1}(\mathbb{E}[a_i] - \mathbb{E}[b_i])$.

Let $c_0 = y_0, \mathbb{E}[a_i] - \mathbb{E}[b_i] = f(hi)$, then

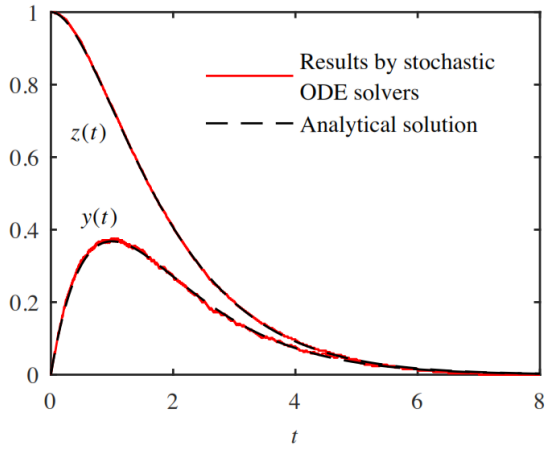
$\mathbb{E}[c_k] = \hat{y}_k \approx y(\frac{k}{2^n})$, with $h = \frac{1}{2^n}$.

Hardware ODE solver using DSC

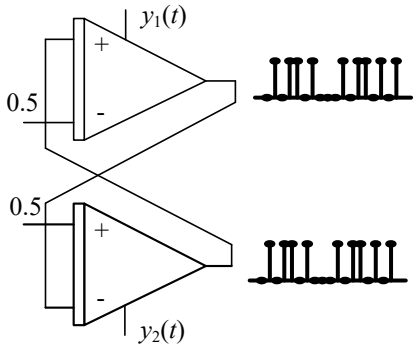
- Numerical solution of ordinary differential equations (ODEs) using the Euler method.
- A second-order ODE



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

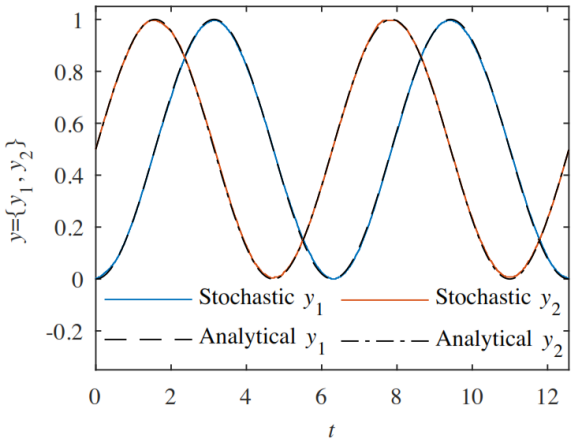


- A system of ODEs



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

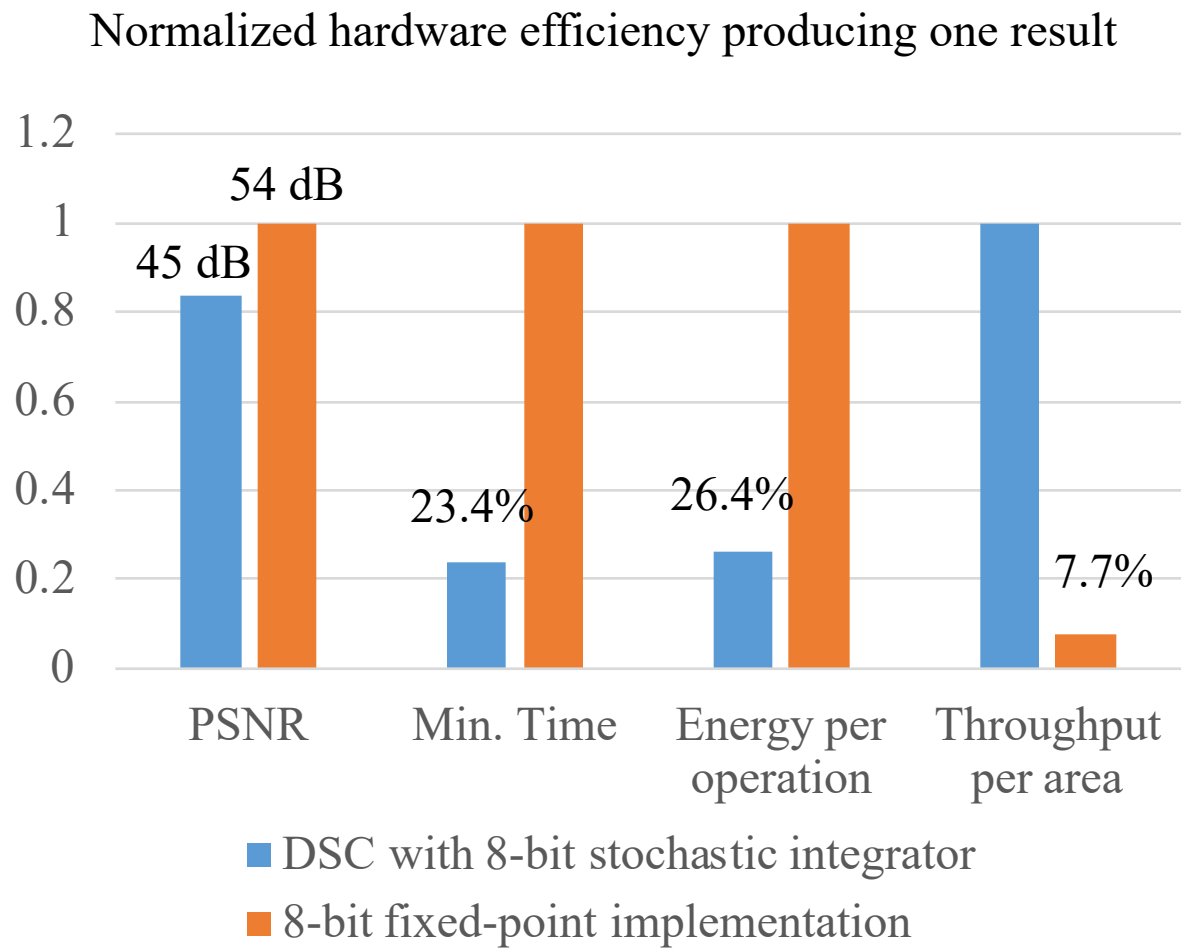
$$\frac{dy_2(t)}{dt} = 0.5 - y_1(t)$$



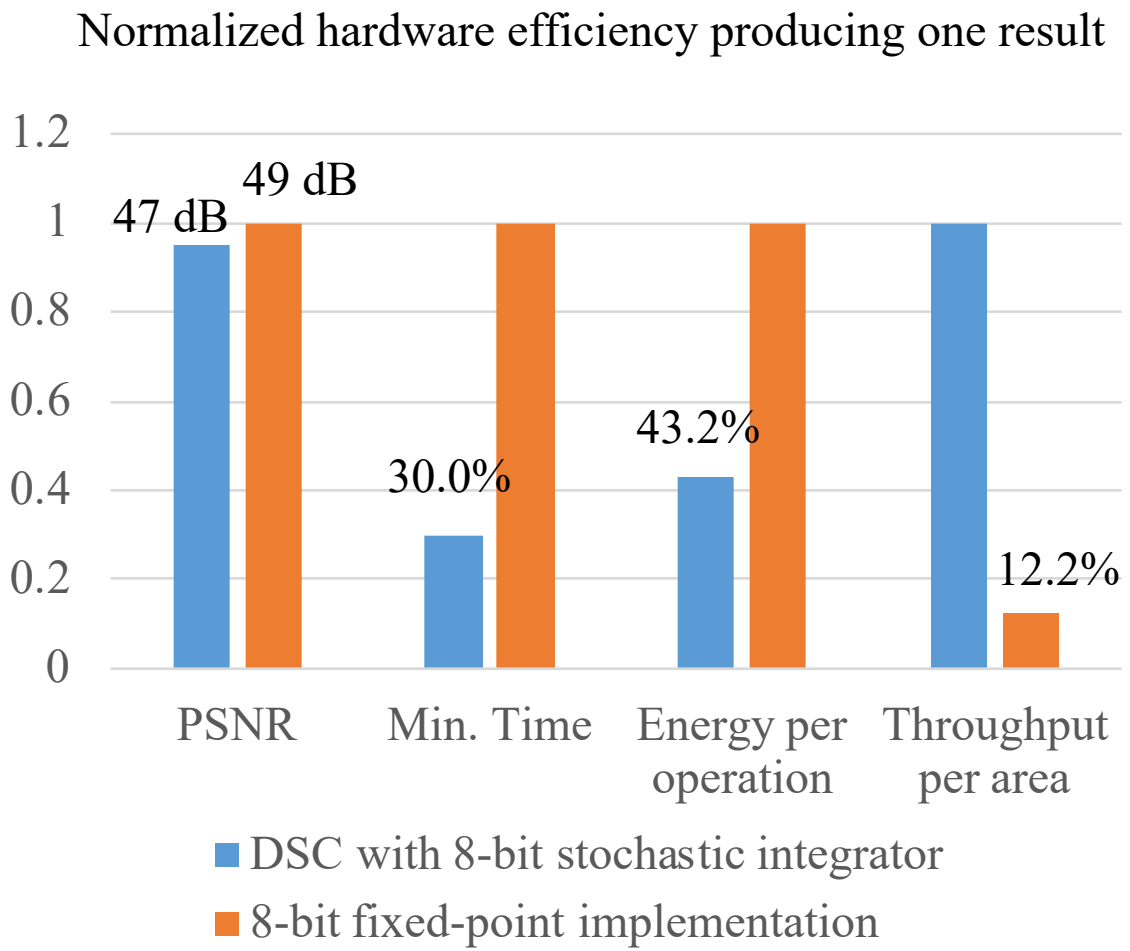
Hardware solution produced by stochastic ODE solver vs. analytical solution.

Hardware efficiency of DSC-based ODE solver

❑ A second-order ODE



❑ A system of ODEs

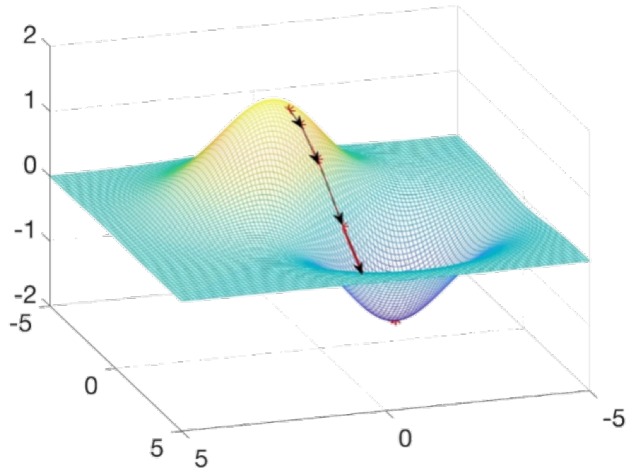


Stochastic Integrator for Gradient Descent

❑ Gradient descent

As a basic optimization algorithm, gradient descent (GD) has widely been used in machine learning to optimize the weights of a model by minimizing a loss function.

The optimization result is an accumulation of multiple steps of the gradients ($\nabla L(\mathbf{w}_n)$).



Gradient descent searching for local minimum.

Let $L(\mathbf{w})$ be a multivariate differentiable loss function, where $\mathbf{w}$ is a vector of weights that are to be trained or optimized. GD computes the local minimum of the loss function by an iterative optimization:

$$\mathbf{w}_{n+1} = \mathbf{w}_n - \eta \nabla L(\mathbf{w}_n), \qquad (1)$$

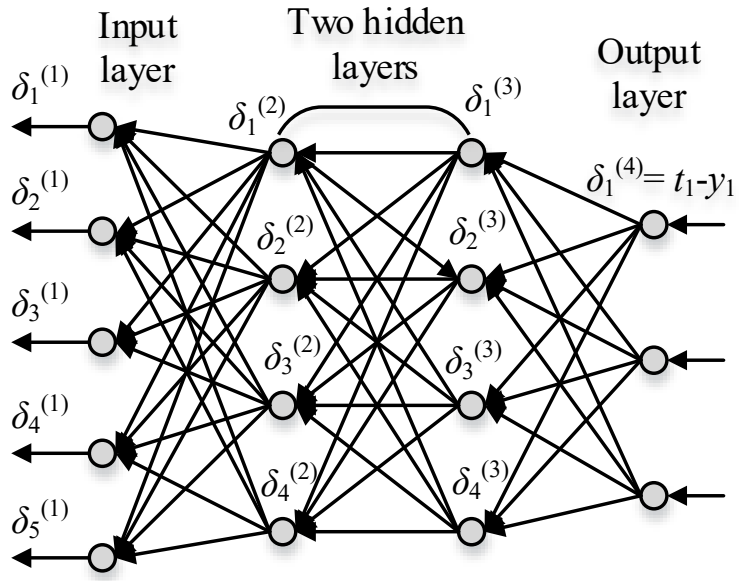
where $\nabla L(\mathbf{w}_n)$ is the gradient, and η is the step size or learning rate, which determines how fast the model learns. If using a constant η, (1) is accumulated for $i = 0, 1, 2, \ldots, N-1$

$$\mathbf{w}_N = \mathbf{w}_0 - \eta \sum_{n=0}^{N-1} \nabla L(\mathbf{w}_n). \qquad (2)$$

Training of a Neural Network

❑ Forward- and Backward-propagation

1. During the forward-propagation, the output of neuron i in layer k, $y_i^{(k)}$, is recorded.



Signal flow of training (backward phase) in a simple deep learning model

2. During the backward-propagation, the local gradients, $\delta_j^{(k)}$ are computed.

3. The gradients for each neuron are then obtained by $\Delta w_{i,j}^{(k-1)} = \eta \delta_j^{(k)} y_i^{(k-1)}$, where η is the step size.

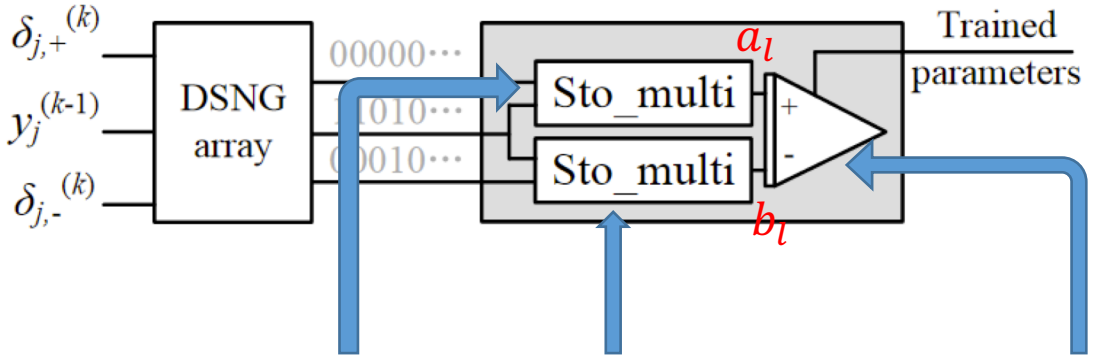
4. $\delta_j^{(k)}$ is further decomposed into $\delta_{j,+}^{(k)}$ and $\delta_{j,-}^{(k)}$, and $\Delta w_{i,j}^{(k-1)} = \eta \left(\delta_{j,+}^{(k)} - \delta_{j,-}^{(k)} \right) y_j^{(k-1)}$, in order to work with the stochastic integrator.

5. Sum-over $\Delta w_{i,j}^{(k-1)}$ for each training sample provides the trained parameter of $w_{i,j}^{(k-1)}$.

*$\delta_{j,+}^{(k)}$: local gradient contributed by the target

*$\delta_{j,-}^{(k)}$: local gradient contributed by the NN output

DSC-based gradient descent circuit design



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

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 lth sample in the training data.

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

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

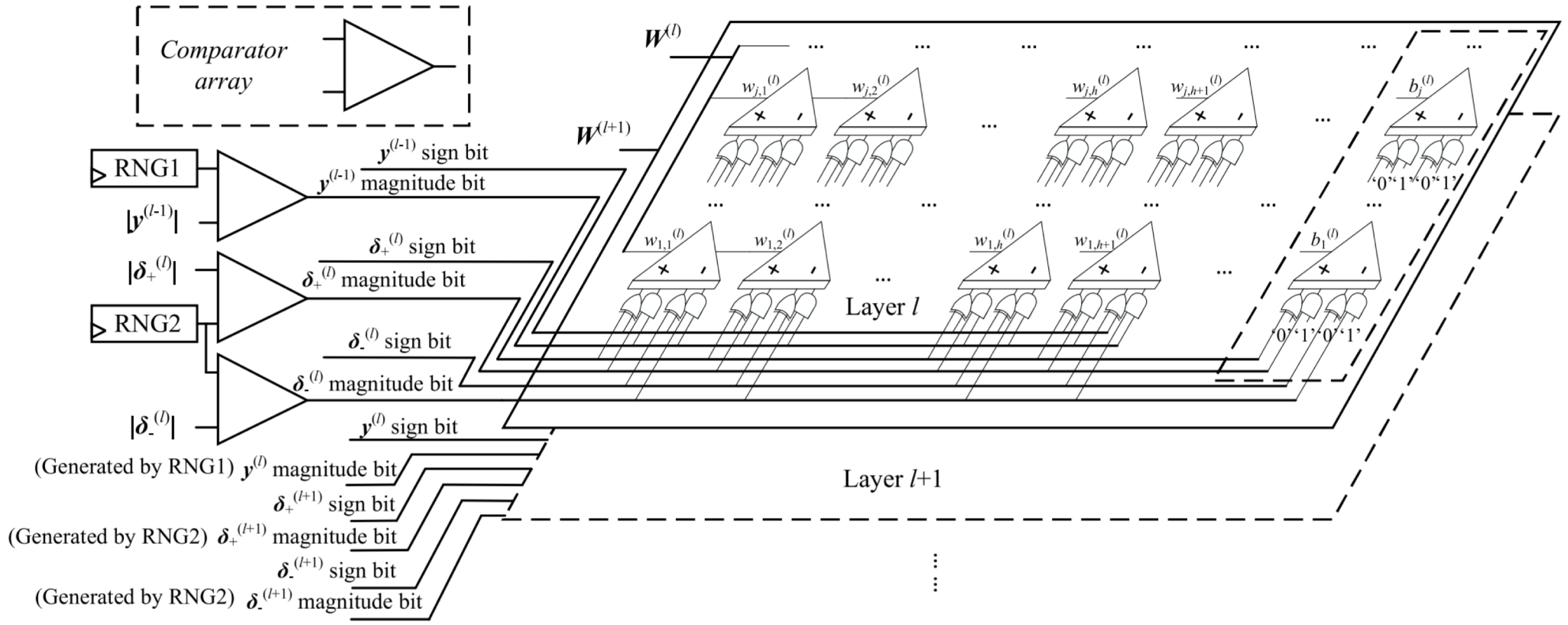
Sto_multi1: $\mathbb{E}[a_l] = \delta_{j,+}^{(k)}[l] y_j^{(k-1)}[l]$

Sto_multi2: $\mathbb{E}[b_l] = \delta_{j,-}^{(k)}[l] y_j^{(k-1)}[l]$ (DSC)

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

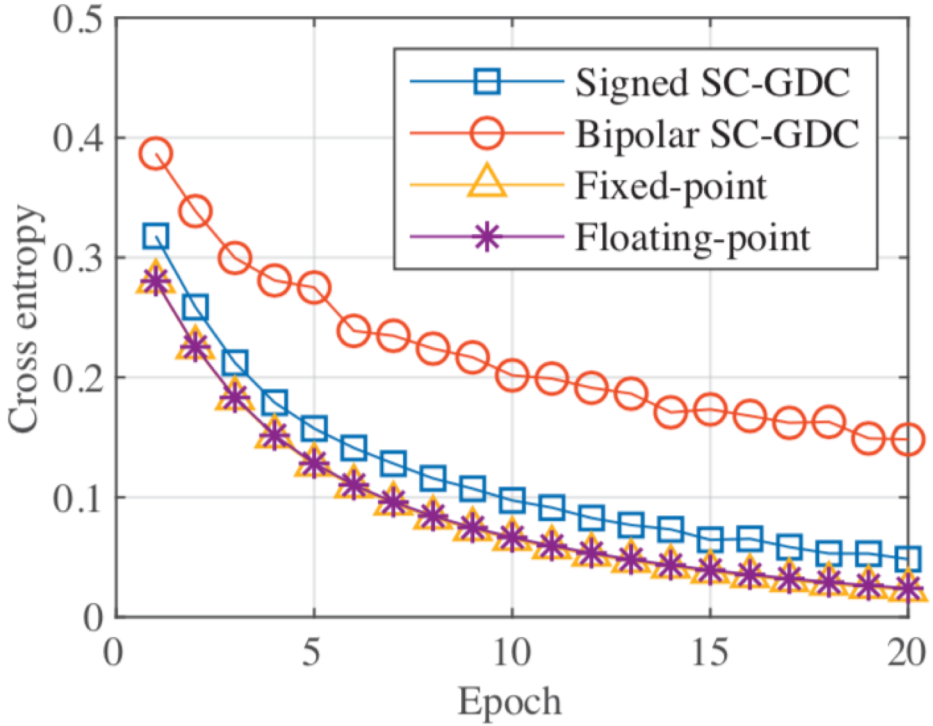
Experiments on MNIST Dataset

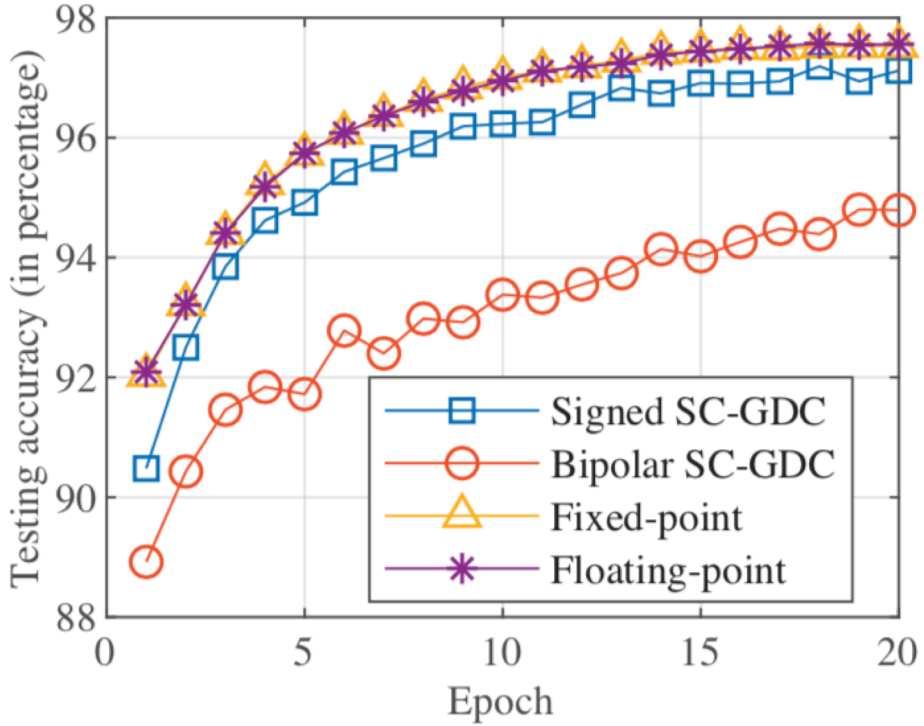
- The weights of a 784-128-128-10 fully connected neural network (NN) is updated by the proposed DSC-based gradient descent circuit. The NN is used for handwritten digit recognition of MNIST dataset.



Experiment Results

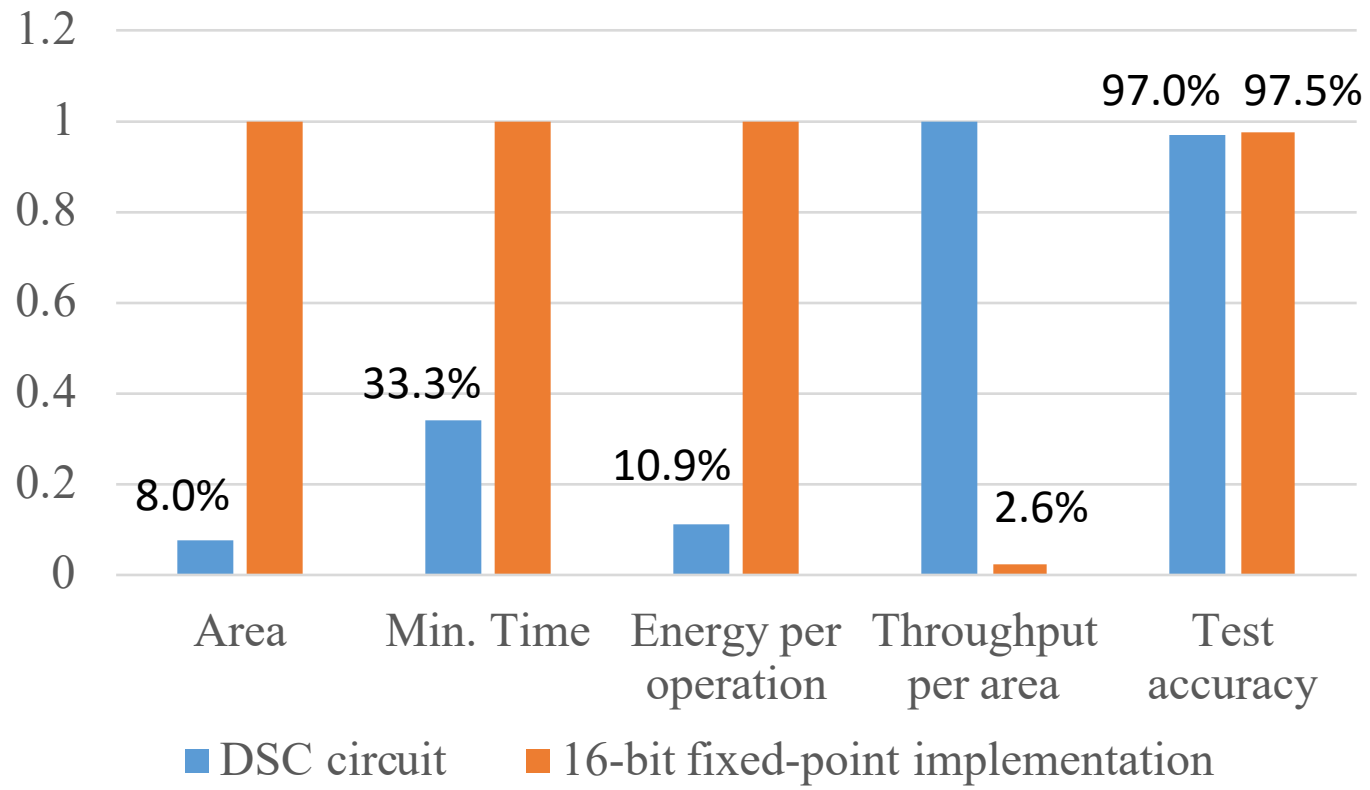
Cross entropy and test accuracy produced by the proposed dynamic stochastic circuits are shown below. The bipolar stochastic circuits, 16-bit fixed-point and floating-point implementations are also considered for comparison. The NN is trained one sample at a time using gradient descent, without any other optimization, such as batch-norm, drop-out, adaptive step size, momentum, etc.





Hardware Efficiency Assessment

Normalized hardware efficiency, DSC circuit vs. 16-bit fixed-point implementation



*Only the gradient descent (weight update) part is considered for hardware evaluation.

Conclusion

- Dynamic stochastic computing and dynamic stochastic sequences are proposed.
- Stochastic integrator is used to compute iterative accumulations with the proposed dynamic stochastic sequences.
- We show that dynamic stochastic computing can be used to solve ordinary differential equations and to implement gradient descent algorithm for efficient training of a neural network.
- Hardware evaluation and performance of the dynamic stochastic circuits are compared with fixed-point implementations. The DSC circuit takes around 30% of the runtime and on average 26.8% of the energy consumption of the fixed-point implementation, while providing significant improvement of throughput per area (8, 13, 38 times respectively), with limited accuracy degradation.

References

[1] Gaines, Brian R. "Stochastic computing systems." *Advances in information systems science*. Springer, Boston, MA, 1969. 37-172.

[2] Hayes, John P. "Introduction to stochastic computing and its challenges." *Proceedings of the 52nd DAC*. 2015.

[3] Qian, Weikang, et al. "An efficient implementation of numerical integration using logical computation on stochastic bit streams." *Proceedings of the ICCAD*. 2012.

[4] Saraf, Naman, et al. "IIR filters using stochastic arithmetic." *Proceedings of DATE*. 2014.

[5] Lipasti, Mikko and Carly Schulz. "End-to-end stochastic computing." *Proceedings of the 1st Workshop on Pioneering Processor Paradigms*. 2017.

[6] Liu, Siting and Jie, Han. "Dynamic Stochastic Computing for Digital Signal Processing Applications." *Proceedings of DATE*. 2020.

[7] Saraf, Naman, et al. "IIR filters using stochastic arithmetic." *Proceedings of DATE*. 2014.

[8] Liu, Siting, and Jie Han. "Hardware ODE solvers using stochastic circuits." *Proceedings of the 54nd DAC*. 2017.

[9] Haykin, Simon S. 2009. Neural Networks and Learning Machines. 3rd ed. New York: Prentice Hall/Pearson.

[10] Liu, Siting, et al. "Gradient descent using stochastic circuits for efficient training of learning machines." *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37.11 (2018): 2530-2541.

[11] Neugebauer, Florian, Ilia Polian, and John P. Hayes. "On the Limits of Stochastic Computing." *Proceedings of ICRC*. 2019.

[12] Zhakatayev, Aidyn, et al. "Sign-magnitude SC: Getting 10X accuracy for free in stochastic computing for deep neural networks." *Proceedings of the 55th DAC*. 2018.

Thanks for your attention!