CiM-BNN:Computing-in-MRAM Architecture for Stochastic Computing Based Bayesian Neural Network

Gu, Huiyi; Jia, Xiaotao; Liu, Yuhao; Yang, Jianlei; Wang, Xueyan; Zhang, Youguang; Cotofana, Sorin Dan; Zhao, Weisheng

DOI
10.1109/TETC.2023.3317136

Publication date
2024

Document Version
Final published version

Published in
IEEE Transactions on Emerging Topics in Computing

CiM-BNN:Computing-in-MRAM Architecture for Stochastic Computing Based Bayesian Neural Network

Huiyi Gu, *Student Member, IEEE*, Xiaotao Jia, *Member, IEEE*, Yuhao Liu, *Student Member, IEEE*, Jianlei Yang, *Member, IEEE*, Xueyan Wang, *Member, IEEE*, Youguang Zhang, *Member, IEEE*, Sorin Dan Cotofana, *Fellow, IEEE*, and Weisheng Zhao, *Fellow, IEEE*

(Invited Paper)

***Abstract*—Bayesian neural network (BNN) has gradually attracted researchers' attention with its uncertainty representation and high robustness. However, high computational complexity, large number of sampling operations, and the von-Neumann architecture make a great limitation for the further deployment of BNN on edge devices. In this article, a new computing-in-MRAM BNN architecture (CiM-BNN) is proposed for stochastic computing (SC)-based BNN to alleviate these problems. In SC domain, neural network parameters are represented in bitstream format. In order to leverage the characteristics of bitstreams, CiM-BNN redesigns the computing-in-memory architecture without complex peripheral circuit requirements and MRAM state flipping. Additionally, real-time Gaussian random number generators are designed using MRAM's stochastic property to further improve energy efficiency. Cadence Virtuoso is used to evaluate the proposed architecture. Simulation results show that energy consumption is reduced more than 93.6% with slight accuracy decrease compared to FPGA implementation with von-Neumann architecture in SC domain.**

***Index Terms*—Bayesian neural network, computing-in-memory, energy efficiency, STT-MRAM, stochastic computing.**

I. Introduction

Bayesian neural network (BNN) has the support of fundamental probability theory [1] that makes up for the shortcomings of traditional DNNs in mathematical theory. DNNs use a large volume of weight data that is difficult to store in on-chip memory of embedded designs [2]. Small sample learning ability of BNN makes it possible to be deployed on edge devices. And BNN is ideal for specific scenarios such as medical, security, auto-driving due to its uncertainty representation [3], high robustness [4] and strong interpretability [5]. In BNN, network parameters are modeled as probability distribution, and T concrete neural networks are instantiated through sampling operations. High computational complexity and calculation pressure, especially in sampling and feed-forward propagation, exacerbate the speed disparity between processor and memory. Compared with DNNs, BNN exists a more prominent "memory-wall" problem with traditional von-Neumann architecture.

Approximate computing can approximately (inexactly) process data to save power and achieve high performance while maintaining an acceptable result [6]. Stochastic Computing (SC) is a non-conventional approximate computation method based on probabilities [7]. SC reduces calculation complexity, hardware costs and power consumption [8] by simplifying complex data computation into bit operations. The work of [9] proposes a StocBNN that introduces BNN into SC domain to simplify data representation and network computing. However, StocBNN is still based on the traditional von-Neumann architecture, so it cannot solve the "memory wall" problem in BNN.

Computing-in-memory is considered one of the main technical routes to realize intelligent computing. It realizes the fusion of storage elements and compute elements to alleviate "memory wall" problem [10]. Some emerging non-volatile memories (eNVMs) with low power consumption are welcomed such as RRAM, PCM, FeFET and MRAM [11], [12]. Non-volatile computing-in-memory can be achieved by using small-capacity

Manuscript received 20 October 2022; revised 28 July 2023; accepted 11 September 2023. Date of publication 25 September 2023; date of current version 6 December 2024. This work was supported in part by the National Natural Science Foundation of China under Grants 62006011, U20A20204, 62072019, and 62004011, and in part by the 111 Talent Program under Grant B16001. *(Corresponding author: Xiaotao Jia.)*

Huiyi Gu, Yuhao Liu, and Youguang Zhang are with the School of Electronic and Information Engineering, Beihang University, Beijing 100191, China (e-mail: guhuiyi@buaa.edu.cn; liuyho@foxmail.com; zyg@buaa.edu.cn).

Xiaotao Jia is with the School of Integrated Circuit Science and Engineering, Beihang University, Beijing 100191, China, and also with the Zhongfa Aviation Institute, Beihang University, Hangzhou, Zhejiang 311115, China (e-mail: jiaxt@buaa.edu.cn).

Xueyan Wang and Weisheng Zhao are with the School of Integrated Circuit Science and Engineering, Beihang University, Beijing 100191, China (e-mail: wangxueyan@buaa.edu.cn; weisheng.zhao@buaa.edu.cn).

Jianlei Yang is with the School of Computer Science and Engineering, State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, China (e-mail: jianlei@buaa.edu.cn).

Sorin Dan Cotofana is with the Computer Engineering Laboratory, Delft University of Technology, 2628 CD Delft, Netherlands (e-mail: S.D.Cotofana@tudelft.nl).

Digital Object Identifier 10.1109/TETC.2023.3317136

memristor crossbar arrays combined with peripheral readout circuits made from discrete components [13]. PCM has been used to implement mixed-precision in-memory computing in [14]. RRAM is leveraged in [15] to develop a computing-in-memory accelerator for more efficient real-time inference.

It is noteworthy that MRAM is often used for in-memory computing because of its several desirable attributes, such as non-volatility, decoupled sensing and buffering, high density and near-zero leakage [16]. While RRAM and other eNVMs can realize multi-bit data computing, they may need more complex peripheral circuits and have higher read latency. MRAM only has two states, namely AP and P, which enable direct bitstream storage and lower latency. Using MRAM to design a computing-in-memory architecture in SC domain based on the work [9] can solve the current BNN application limitations mentioned above. Furthermore, the stochastic switching property of MRAM could also be utilized as Gaussian random number generators. Considering these factors, this paper proposes a computing-in-MRAM BNN architecture in SC domain.

The main contributions of this work are summarized as follows:

- A new computing-in-MRAM BNN architecture (CiM-BNN) is proposed in SC domain. BNN inference in this architecture is carried out in situ which is helpful to deal with the "memory wall" problem.
- The core circuits are all made up of spintronic devices to achieve corresponding functions. Gaussian random number generator (GRNG) is implemented based on the stochastic switching behavior of MRAM. Computing-in-memory arrays are built upon the binary characteristic of MRAM. The system energy consumption is significantly reduced due to its almost-all-spin characteristic.
- Hardware implementations with low circuit complexity are explored. The operations in BNN are finally simplified from Dot-product operation to Read operation in computing-in-MRAM arrays without complex peripheral circuits. Inference stage does not involve the state flipping of MTJs, which improves circuit stability.

The remainder of this paper is organized as follows. Section II introduces the current research status of BNN inference acceleration. Section III briefly describes the relevant prior knowledge. Section IV explains essential computing-in-MRAM cells and calculation principles in detail. The overall CiM-BNN architecture is described in Section V. Section VI evaluates experimental results in Cadence Virtuoso. The summary can be found in Section VII.

II. Related Work

The current work related to BNN inference acceleration can be roughly divided into two categories. The first category focuses on software and hardware optimizations within the existing von-Neumann architecture [9], [17], [18], [19], [20], [21]. Jia et al. [17] optimize BNN feed-forward process by data reuse strategy, and realize an energy consumption reduction of 73% and a 4× speedup. FPGA-based design in [18] can intelligently skip redundant computations of dropout masks and zero-corresponding computations during Bayesian CNN inferences. 3-D BayesCNNs hardware acceleration architecture is designed and an automatic framework is provided in [19]. Cai et al. [20] explore the design space for massive amount of Gaussian variable sampling tasks in BNNs and propose two high performance Gaussian (pseudo) RNG. Awano et al. [21] replace costly GRNGs with Bernoulli RNG and improve 57.5% energy efficiency compared to [20]. StocBNN in [9] implements BNN deployment in SC domain. These works indeed achieve performance improvements in BNN. However, they are based on traditional architectures and most of them only focus on either the sampling or the feed-forward process, resulting in limited effectiveness.

The second category explores the computing-in-memory implementation of eNVMs in BNN [22], [23], [24], [25], [26]. Yang et al. [22] make the first exploration of a BNN hardware accelerator enabled by SOT-MRAM and demonstrate 24× reduction in energy consumption. Lin et al. [23] utilize RRAM inherent stochasticity to develop a typical risk-sensitive reinforcement learning task with a four-layer BNN. Works in [24], [25] achieve high quality uncertainty quantification and high-precision BNN calculation with RRAM. Ahmed et al. [26] design a binary dropout-based BNN with an end-to-end approach and implement the architecture with MRAM. These studies demonstrate the potential of eNVMs for various machine learning applications, highlights the importance of exploring new computing paradigms, but they still face certain challenges. These architectures are all conducted in digital domain, the devices require high-precision digitization capabilities. Furthermore, complex peripheral circuits such as complex analog-to-digital converters (ADCs) and digital-to-analog converters (DACs) may also be necessary.

Considering these factors, this paper presents a computing-in-MRAM BNN implementation in SC domain. The data representation, computation and output reading in CiM-BNN are different from existing works.

III. Preliminaries

This work essentially integrates and innovates the advantages of Bayesian neural network, approximate computing and non-volatile computing-in-memory. The purpose is to find an appropriate computing diagram and architecture that could alleviate the limitations of current BNN. By optimizing hardware architecture, data flow design, circuit energy consumption to explore BNN diversified deployment. Before the description of overall architecture, some preliminaries are discussed in this section.

A. Bayesian Neural Network

Different from standard neural networks, the network parameters of BNN are modeled as the probability distribution instead of fixed parameters. Mathematical foundation support of BNN is Bayesian theory which makes neural networks interpretable [27]. In Bayesian theory, posterior probability is derived from prior probability distribution and network input

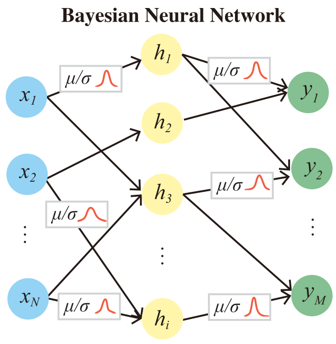


Fig. 1. BNN diagrammatic sketch. Weight is represented by Gaussian probability distribution and output also conforms to Gaussian probability distribution after Dot-product operation.

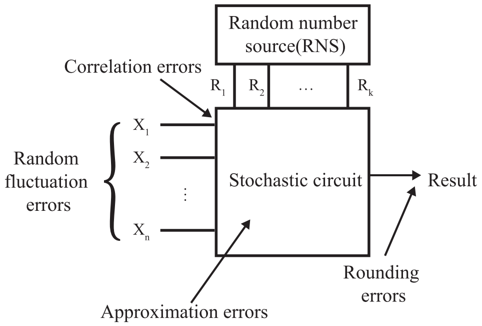


Fig. 2. Error sources in general SC circuits [29].

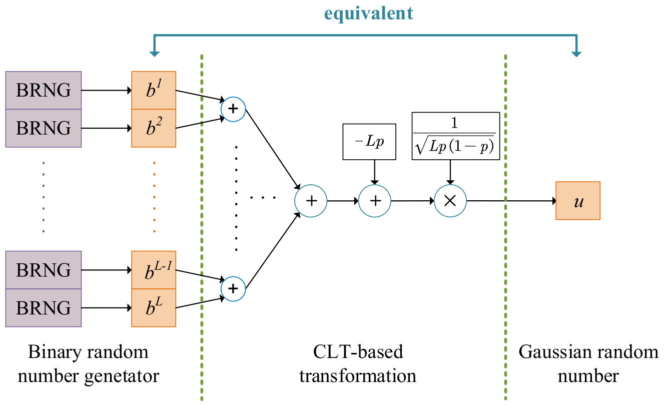


Fig. 3. Equivalence relation between 0-1 sequence and Gaussian random number [9].

data. Bayesian theory can be written as (1):

$$P(W|D) = \frac{P(D|W)P(W)}{P(D)}, \tag{1}$$

Here, W means weight parameters, characterized by mean (μ) and standard variation (σ), with Gaussian distribution. D means input node data of neural network. $P(W)$ is defined as the prior probability distribution. It is artificially estimated data and represents the probability of network parameters before any data is entered. $P(D|W)/P(D)$ is the likelihood, it can be regarded as an adjustment factor to make the estimated probability distribution closer to the true one. $P(W|D)$ is the posterior probability distribution. By assuming prior probability distribution and learning posterior probability, BNN can continuously train network parameters. Variational inference method is generally applied in network training [28], making posterior probability density estimation tractable and achieving faster learning speed. After Dot-product operation, BNN output also conforms to probability distribution characterized by mean and standard variation as shown in Fig. 1. Standard variation is the uncertainty measure of network. BNN's uncertainty quantification ability greatly improves network robustness.

For the inference stage of well-trained BNN, weight distribution is instantiated into T specific neural networks through sampling operations. Input data (such as images) would be fed into all T neural networks and generate T outputs. Final inference result is determined by those outputs. [20] makes it straightforward for the first time that BNN needs to perform "weight sampling and feed-forward propagation" many times in the inference stage. According to the accuracy requirements of tasks, the value of T can be different, generally about 100. The increasing demand for memory access and computationally intensive operations lead BNN to face more serious "memory wall" problem than DNN.

B. Approximate Computing

Approximate computing is considered one of the few paradigms that could reduce power consumption by orders of magnitude. It sacrifices the pursuit of accurate results to obtain better performance [30]. Therefore, approximate computing has strong application prospects in error-tolerant scenarios such as signal processing, machine learning, data mining, etc. [31]. It has been applied at different levels, from devices to systems, from hardware to algorithms.

As a non-conventional type of approximate computing, SC is first proposed by von Neumann in 1950s [32]. It simplifies computing requirements and trades off computational errors for low-cost hardware, low power consumption and other benefits [33]. As shown in Fig. 2, SC is not an accurate calculation method. It has several error sources which are peculiar to SC, including rounding errors, approximation errors, random fluctuations errors and correlation errors [29].

In SC, data exists in the form of bitstream. Under different coding modes, data will be converted into different forms of 0-1 sequence, named stochastic numbers (SNs). By counting 1's in bitstream, we can know the corresponding digital value of SNs. For instance, in unipolar representation, one SN can represent a digital value in [0,1] (i.e. $P(x=1)$). 01101001 contains four 1 s in an eight-bit stream, so the value is $\frac{1}{2}$. In unipolar, the arithmetic operation (multiplication) is replaced by logical operation (AND). Compared with arithmetic operation, logic operation is bit-wised. Each memory cell only need to store one bit data (0 or 1). As the length of the bitstream increases, the precision of the value improves, but it will consume more runtime.

The work in [9] proved that Gaussian random numbers could be represented by 0-1 sequence and participate in neural network computing directly without central limit theorem (CLT)-based transformation. Fig. 3 shows the equivalence relation between a series of binary random numbers and a Gaussian random number. It indicates that data in digital domain can be directly generated with Binary random number generators (BRNGs) that could randomly generate 1 (or 0) with probability p (or $1-p$). Based on this proof, BRNGs can be directly used to generate corresponding bitstream data that conforms to Gaussian distribution. Meanwhile, the work has proved that bitstream could participate in the feed-forward propagation as a whole with updated equivalent mean μ' and standard deviation σ'. Its transformation process is shown in (2):

$$\begin{aligned} y_i &= \sum_{j=1}^{N} w_{ij}x_j = \sum_{j=1}^{N} \left(u_{ij}\sigma_{ij} + \mu_{ij}\right)x_j \\ &= \sum_{j=1}^{N} \left(h_{ij}\sigma'_{ij} + \mu'_{ij}\right)x_j = \sum_{j=1}^{N} h_{ij}\sigma'_{ij}x_j + \sum_{j=1}^{N} \mu'_{ij}x_j \quad (2) \end{aligned}$$

Here, $h_{ij} = \frac{\sum_{k=1}^{L} b_{ij}^{k}}{L}$, is the digital value of SNs in unipolar encoding. $\sigma'_{ij} = \sqrt{\frac{L}{p(1-p)}}\sigma_{ij}$ and $\mu'_{ij} = \mu_{ij} - \sqrt{\frac{Lp}{1-p}}\sigma_{ij}$. This equation forms the basis of BNN in SC domain and provides a solid foundation for the in-memory computing application with spintronic devices. Next, we try to design suitable computing-in-memory arrays for $\mu'_{ij}x_j$ and $h_{ij}\sigma'_{ij}x_j$.

C. Magnetic Tunnel Junction

MRAM has better process deviation and lower write voltage than other eNVMs [34]. It is treated as one of the most suitable candidate for on-chip cache applications and has some exploratory applications in approximate computing [35]. MRAM uses different collective magnetization states of ferromagnetic layers to store binary data (0 or 1). The core component is Magnetic Tunnel Junction with Perpendicular Magnetic Anisotropy (PMA-MTJ). MTJ consists of three layers: two ferromagnetic layers (CoFeB) and an intermediate oxidation layer (MgO). One of the ferromagnetic layers is fixed layer in which the magnetization direction is fixed. The other one is free layer in which the magnetization direction can be changed according to the injected current.

MTJ has two states: parallel (low resistance) and anti-parallel (high resistance). When the magnetization directions of two ferromagnetic layers are consistent, MTJ presents a parallel state (P) and the resistance value is represented by R_P. When the magnetization directions are opposite, MTJ presents an antiparallel state (AP) and the resistance is represented by R_{AP}. Data is written by switching the magnetization direction of free layer. TMR ratio is used to describe the resistance difference characteristics of MTJ in two states. It is a critical attribute of MTJ and directly affects MTJ's actual features.

There are two primary device types of MTJ, magnetic field driven MTJ and current driven MTJ. Compared with magnetic field driven MTJ, current driven MTJ can achieve higher density and faster write speed [16], so it is currently the mainstream device. Spin Transfer Torque-based MTJ (STT-MTJ) is current driven MRAM. It uses injected current to change the magnetism direction of free layer. Fig. 4 shows a typical structure of STT-MTJ. The spin-polarized current is created by passing a current through a thin magnetic layer. If the spin-polarized current ($I_{P\rightarrow AP}$) flows from free layer to reference layer, the MTJ state will be switched from P state to AP. On the contrary, if $I_{AP\rightarrow P}$ is formed, the MTJ state will be switched from AP state to P. Once spin-polarized current is greater than its critical reverse current I_{C0}, magnetization direction is affected by the intensity of current and pulse duration with a switching probability [36]. By applying different programming voltages with suitable duration, MTJ can be flipped with a fixed probability as shown in Fig. 5. Based on these, MTJs can be used as the fundamental devices of computing-in-memory arrays and simplified SC domain GRNG.

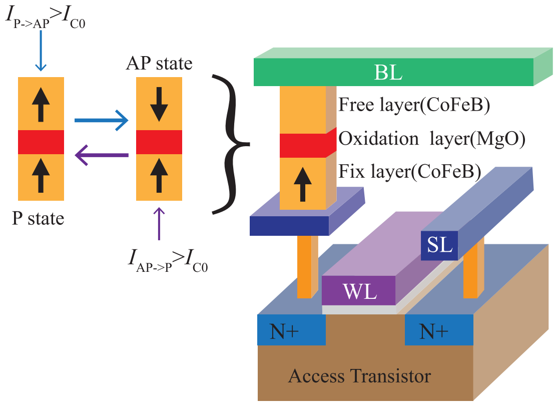


Fig. 4. Typical Spin Transfer Torque-based Magnetic Tunnel Junction (STT-MTJ) structure. Current is injected perpendicular to the MTJ and passes through the thin magnetic layer to form a spin-polarized current. Spin-polarized current enables MTJ to switch between AP and P states.

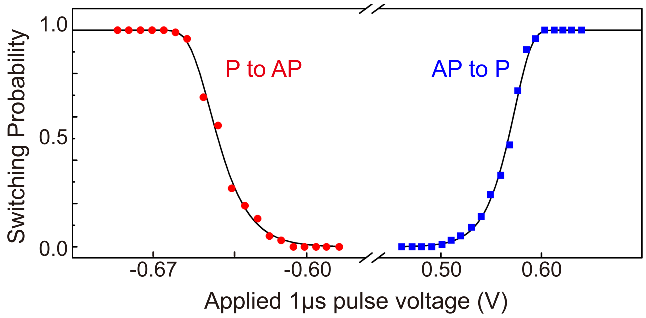


Fig. 5. Experimental measurements of the switching probability with different programming voltages [36].

IV. Calculation Principle of CiM-BNN

As can be seen from (2), BNN calculation can be divided into two parts. One part is $\mu'_{ij}x_j$, and the other part is $h_{ij}\sigma'_{ij}x_j$. The two parts involve different numbers of parameters, so different computing-in-MRAM cells (CiM-cells) are designed. STT-MRAM is used to store the well-trained BNN parameters. CiM-cell of mean stores the value of μ' and CiM-cell of standard deviation stores σ'. CiM-cell of mean is slightly different

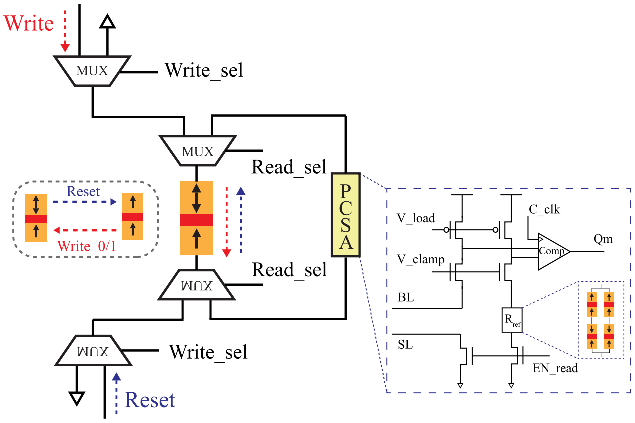
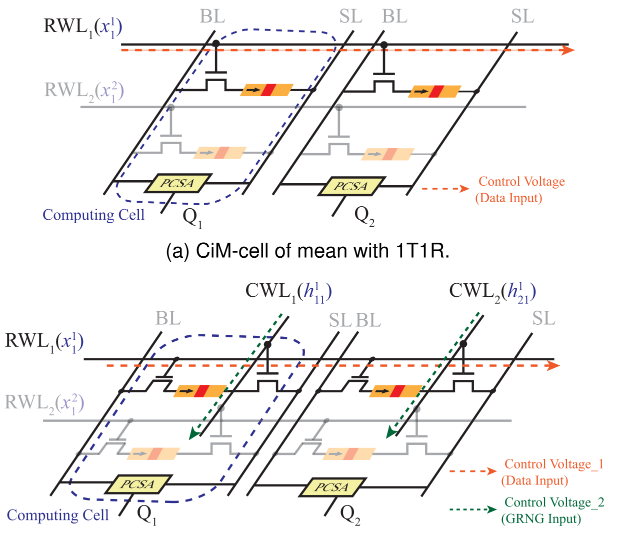


Fig. 6. Real-time GRNG based on MTJ. Reset operation puts MTJ to parallel state. Write operation changes MTJ state. PCSA reads out the result. R_{ref} is set as $(R_P + R_{AP})/2$ in this work.



(a) CiM-cell of mean with 1T1R.

(b) CiM-cell of standard deviation with 2T1R.

Fig. 7. Basic memory cells of CiM-BNN. Data is calculated row by row, only one RWL is activated at the same time,e.g.when x_1^1 is activated, no other rows have any input data.

from CiM-cell of standard deviation, so they will be described separately.

In CiM-BNN, input data x and random sampling parameter h are treated as transistors' control signals to control the on-off of the transistors. h is generated by real-time GRNG based on MTJ for resolving the circuit complexity. Array calculation results are read out by pre-charge sense amplifier (PCSA), so the multiplication in BNN is finally simplified to read operation in MRAM arrays. Multiplexers (MUXs) and counters are used to realize data accumulation within SC method. All digital data are converted into bitstreams in advance and all calculations are performed in SC domain. BNN inference is achieved without data movement with proposed computing architecture.

A. Real-Time GRNG Based on MTJ

GRNG is used in weight sampling stage of BNN. Weight needs to be randomly sampled multiple times to ensure the inference accuracy. In other design schemes, GRNG usually occupies extensive hardware resources due to its complexity. Different ways of improving GRNG performance are proposed in [37], [38], [39].

In this paper, real-time GRNG is designed to generate h and selection signal of MUXs. Referring to the stochastic number generator circuit [40], we use MTJ to design real-time GRNG. By performing reset, write and read three operations, MTJ state can be continuously switched to generate random numbers. Fig. 6 shows the detailed structure and random numbers generation process. When $write\_sel$ is set to be high level, $read\_sel$ is set to be low. MTJ would be reset to parallel state first through reset current and then written to different states through write current. When $read\_sel$ is set to be high level, $write\_sel$ is set to be low, and PCSA is employed for reading out MTJ's current state. The resistance value of reference cell is set between R_P and R_{AP}, normally $(R_P + R_{AP})/2$ or $(R_P * R_{AP})^{1/2}$. C_clk controls the working states of PCSA. The internal property of MTJ reduces the power consumption of random numbers generation. By selecting appropriate pulse voltage and duration of injection current, the switching probability of MTJ can be set to 50%. It should be note that, based on (2), even if the transition probability of the MTJ cannot be guaranteed to be exactly 50%, or if it cannot be ensured that each output y_i corresponds to the same transition probability, it will not affect the results. After measuring the value of p for different MTJs in advance, the correct calculation can be achieved by adjusting the values of μ'_i and σ'_i.

B. CiM-Cell of Mean

As shown in Fig. 7(a), traditional 1T1R structure is chosen as basic CiM-cell to realize the multiplication of x and μ'. Equivalent mean μ' are stored in MTJ arrays. Input signal x is applied on the Wordline that plays the role of row driver (RWL in Fig. 7). PCSA shown in Fig. 6 is also employed between Bitline (BL) and Sourceline (SL) for identifying the final state of calculation results.

First, set the initial state of MTJs to parallel state (P), then write μ' into MTJs by injecting appropriate current. Each MTJ stores one bit, and μ' in bitstream form is stored vertically. x is a voltage control signal. When $x = 1$, the corresponding RWL will be activated. When $x = 0$, the transistor remains off and no current passes through. It should be noted that only one RWL can be activated at a time, but all columns are calculated together. Then the equivalent resistance value R_{eq} of CiM-cells can be obtained. PCSA will get the result of $\mu'^k_{ij} x^k_j$ by comparing R_{eq} between BL and SL with reference resistance R_{ref}. If R_{eq} $>R_{ref}$, PCSA outputs 0, If R_{eq} $<R_{ref}$, PCSA outputs 1. Truth table of the circuit is shown in Table I. Here, we define the anti-parallel state of MTJ as "0" and the parallel state as "1".

C. CiM-Cell of Standard Deviation

CiM-cell needs to be modified slightly for realizing $h_{ij}\sigma'_{ij}x_j$. 2T1R structure shown in Fig. 7(b) is the basic CiM-cell of

TABLE I
TRUTHTABLE OF $\mu'_{ij}x_j$ IN EQN. (2)

x(Input)	μ'(STT state)	R_{eq}	Q_m(Output)
0 (0)	0 (AP)	∞	0
0 (0)	1 (P)	∞	0
1 (VDD)	0 (AP)	R_{AP}	0
1 (VDD)	1 (P)	R_P	1

TABLE II
TRUTHTABLE OF $h_{ij}\sigma'_{ij}x_j$ IN EQN. (2)

x(Input 1)	h(Input 2)	σ'(STT state)	R_{eq}	Q_m(Output)
0 (0)	0 (0)	0 (AP)	∞	0
0 (0)	0 (0)	1 (P)	∞	0
0 (0)	1 (VDD)	0 (AP)	∞	0
0 (0)	1 (VDD)	1 (P)	∞	0
1 (VDD)	0 (0)	0 (AP)	∞	0
1 (VDD)	0 (0)	1 (P)	∞	0
1 (VDD)	1 (VDD)	0 (AP)	R_{AP}	0
1 (VDD)	1 (VDD)	1 (P)	R_P	1

standard deviation. Input signal x is still applied on RWL and random number h is applied on the Wordline that plays the role of column driver (CWL in Fig. 7(b)).

Similarly, set the initial state of MTJs to parallel state (P), then write equivalent standard deviation σ' into MTJs. By changing the control voltage according to the value of x and h independently, three different equivalent resistances are formed between BL and SL. PCSA displays the final result. Truth table of $h_{ij}\sigma'_{ij}x_j$ is shown in Table II. In SC domain, multiplying three numbers requires two AND gates, while in CiM-cell, no logical computing unit is required. Designing computing-in-MRAM architecture can save large numbers of AND gates and simplify circuit complexity.

V. ARCHITECTURE OF CIM-BNN

A. Overview of Main Architecture

The overall structure of CiM-BNN is shown in Fig. 8. The system includes computing-in-MRAM array (CiM-array) of mean, CiM-array of standard deviation, PCSA arrays, GRNGs, MUXs, counters and peripheral circuits. The well-trained network parameters (represented by μ' and σ') are written into CiM-arrays in advance. The network performs real-time calculation with continuous data input. PCSA outputs of corresponding columns in two CiM-arrays are transmitted into MUXs for addition. Counters accumulates 1's in bitstream and get corresponding digital domain data as the inference results.

Here, we suppose that input neural count is M, output neural count is N and bitstream length is L. The dimension of CiM-array of mean and CiM-array of standard deviation are both $[M \times L, N]$.

B. Computing-in-MRAM Arrays

CiM-arrays are built according to the two CiM-cells in Section IV. Each array have $(M \times L)$ rows, corresponding to input neuron nodes in bitstream format, and N columns, corresponding to output neuron nodes. Weight parameters in bitstream format are stored vertically in the array and each MTJ stores one bit as shown in Fig. 9.

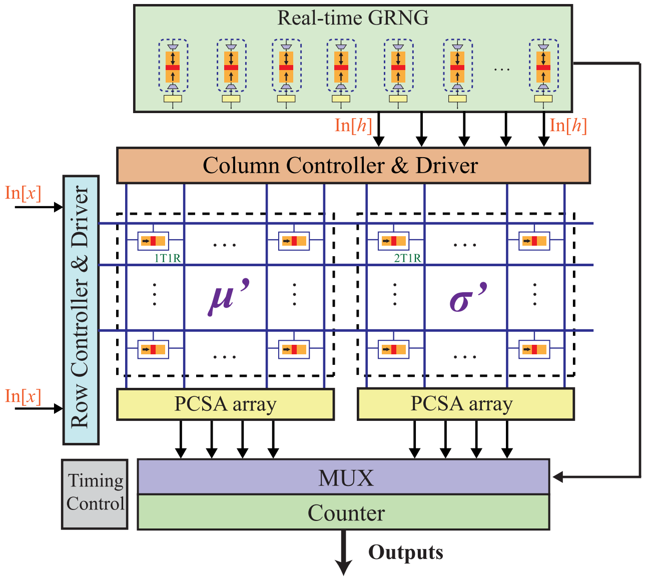
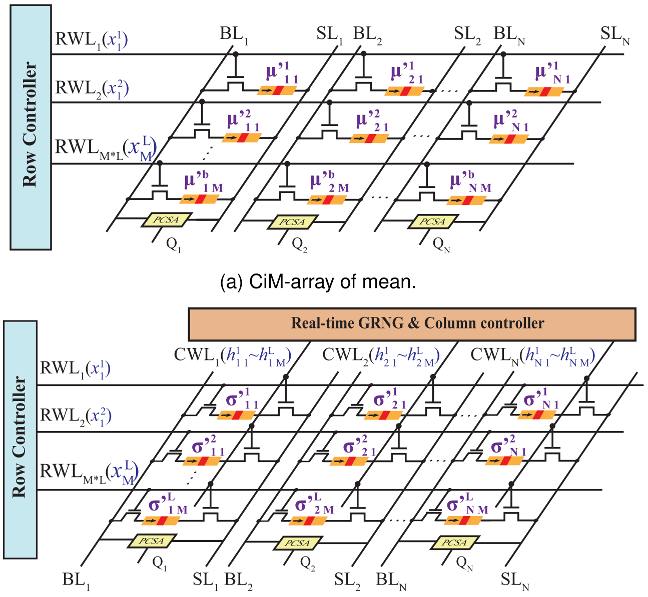


Fig. 8. Overview architecture of CiM-BNN. Random numbers generated by real-time GRNG are fed into CiM-array of standard deviation as sampling parameters and into MUXs as selection signals.



(a) CiM-array of mean.

(b) CiM-array of standard deviation.

Fig. 9. Computing-in-MRAM arrays with M input and N output (Bitlength = L). The dimension of both array are $[M \times L, N]$.

In Fig. 9(a), μ' is pre-stored in CiM-array of mean. This array is utilized to perform the calculation of $\mu'_{ij}x_j$. First, set the initial voltage of all RWLs to 0v, then change input voltage of RWLs row by row according to the value of x. PCSAs can obtain the multiplication result of x and μ' in the array in real time. For example, when the first bit of x_1 (x_1^1) is applied on the RWL$_1$, PCSAs will simultaneously sense the first-bit result of all output neuron nodes $Q_1^1 Q_2^1 \cdots Q_N^1$.

For CiM-array of standard deviation shown in Fig. 9(b), the computing mechanism is similar to that in CiM-array of mean. The array is utilized to perform the calculation of $h_{ij}\sigma'_{ij}x_j$. All RWLs and CWLs keep off first, then the corresponding row (x_j^k) are activated. Real-time GRNGs generate different random numbers ($h_{1j}^k h_{2j}^k \cdots h_{Nj}^k$) and all CWLs are activated simultaneously according to corresponding random numbers. Calculation results are obtained by PCSAs. BNN requires multiple sampling, so CiM-array of standard deviation will be calculated T times.

C. Data input/output

The operation of the entire computing-in-memory circuit is controlled by timing control logic. Equivalent weight parameter μ' and σ' need to be pre-processed by a shifter and converter from digital domain to SC domain. Then control logic writes equivalent weight parameters to CiM-arrays at one time. Input data x is converted into bitstream as voltage signals to control different RWLs and random sampling parameter h is real-time generated by GRNGs to control different CWLs.

The result of the CiM-arrays are read out by PCSAs. A proper bias voltage V_clamp is applied in NMOS transistor and V_load is applied in PMOS transistor to prevent read disturbance. V_load and V_clamp are set to about 0.8 V. PCSA is triggered at the rising edge of C_clk. The resistance difference is converted into voltage difference which could be sensed by a dynamic latched voltage comparator with clock enabled.

VI. Experimental Results

A series of experiments are designed to prove the feasibility and inherent advantages of CiM-BNN.

A. Functional Verification

Functional verification is the first step of the experiments to ensure that the circuit can operate multiplication and addition correctly. In two cases of given input (specific bitstream data) and random real data input (image input data), multiple function verification experiments are carried out, respectively. For each CiM-array of mean and CiM-array of standard deviation, two 8*1 small arrays are taken to perform functional verification. The state of MTJs in the first array are all parallel state ("1") and in the second array are all anti-parallel state ("0"). Q_1 corresponds to PCSA outputs of the first array, and Q_2 corresponds to the second one.

It can be summarized from Fig. 10(a) and (c) that when the input data are given manually, Q_1 and Q_2 show the corresponding results meeting the conditions in Tables I and II, respectively. And when the input data is random real data and h is generated by STT-MRAM (with 50% switching probability), Q_1 and Q_2 shown in Fig. 10(b) and (d) also meet the situation of Tables I and II.

TABLE III
Inference Accuracy Comparison

Model	Dataset	DigtBNN	StocBNN		CiM-BNN	
			64	128	64	128
4-FC	MNIST	98.29%	97.2%	97.51%	97.16%	97.19%
	Fashion-MNIST	90.02%	87.84%	88.13%	87.78%	88%
LeNet-5	MNIST	99.25%	99%	99.15%	89.98%	99.1%
	Fashion-MNIST	99.36%	99.28%	99.34%	99.27%	99.31%

Fig. 10(e) shows the calculation results of MUX with given input, where η represent the multiplication results of $\mu'_{ij}x_j$ and γ is equivalent to $h_{ij}\sigma'_{ij}x_j$. When the selection signal of MUX is high level, MUX outputs γ, and when the selection signal is low level, it outputs η. Maintain 50% of the selection signal at high level, MUX can output the calculation result meeting (3).

$$\frac{y_i}{2} = \frac{\sum_{j=1}^{N} h_{ij}\sigma'_{ij}x_j + \sum_{j=1}^{N} \mu'_{ij}x_j}{2}. \tag{3}$$

Fig. 10(f) shows functional verification of MUX under random real data input. The selection signal is also generated by STT-MRAM with 50% probability and the results meet the expectations.

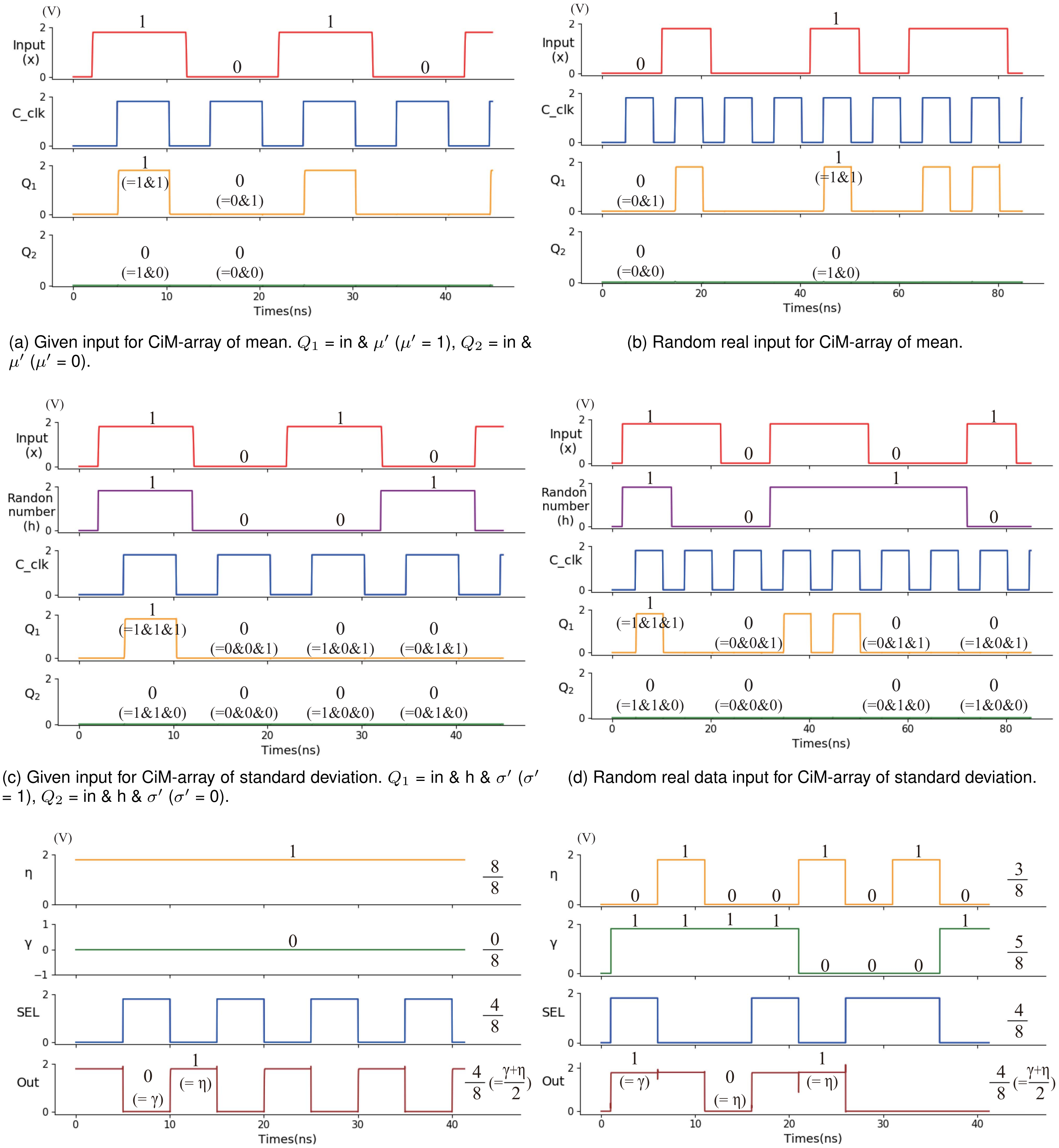
B. Inference Accuracy

We select MNIST [41] and Fashion-MNIST [42] to evaluate the classification accuracy of CiM-BNN. The BNN is pre-trained using Edward [43]. A 784-200-200-10 configuration with fully connected layers (named as 4-FC) and LeNet-5 [44] are adopted. Considering the issue of compounding errors over multiple layers [45], we implement the first layer with SC method and the remaining layers are implemented in the digital domain. ReLU is taken as the activation function. The concrete neural network count T is set as 100 and bitstream length is selected as 64 and 128. Three methods are implemented to evaluate the inference accuracy as well as the energy efficiency: (1) Conventional binary radixbase computing domain BNN (DigtBNN), (2) Stochastic computing domain BNN (StocBNN) and (3) CiM-BNN. CiM-BNN is implemented based on the architecture shown in Fig. 8. All three methods are realized using Python, and the experimental results are demonstrated in Table III.

CiM-BNN and StocBNN have basically the same computational paradigm on software implementation, but they are slightly different in MUX selection signals. Hardware circuit is redesigned in CiM-BNN, and selection signal of MUXs are shared with the same GRNG to reduce hardware cost further. In StocBNN, selection signals are generated by different GRNGs. It can be seen that CiM-BNN maintains the inference accuracy well when compared with StocBNN. For 4-FC and LeNet-5, when the bitstream length is 128, there is only a slight decrease with about 0.33% and 0.05% in MNIST and 0.15% and 0.03% in Fashion-MNIST, respectively.

C. Hardware Performance

In order to demonstrate the energy efficiency of CiM-BNN, two comparisons are performed. CiM-BNN is implemented with



(a) Given input for CiM-array of mean. Q_1 = in & μ' (μ' = 1), Q_2 = in & μ' (μ' = 0).

(b) Random real input for CiM-array of mean.

(c) Given input for CiM-array of standard deviation. Q_1 = in & h & σ' (σ' = 1), Q_2 = in & h & σ' (σ' = 0).

(d) Random real data input for CiM-array of standard deviation.

(e) Given input for MUX. When SEL is high level ("1"), out is γ, otherwise, out is η.

(f) Random real data input for MUX.

Fig. 10. Functional verification of circuit calculation. (a)~(f) represent the final results of different modules under the condition of given input and random real data input. Q_1 is the output of array 1, in which MTJs are stored with parallel state ("1"). Q_2 is the output of array 2, in which MTJs are stored with anti-parallel state ("0").

45 nm CMOS and 40 nm MTJ technologies and evaluated by Virtuoso. Fig. 11 illustrates CiM-BNN energy consumption of different components at different bitlengths. The consumption of CiM-arrays in blue is the sum of CiM-array of mean and CiM-array of standard deviation. For CiM-BNN, the demand for Gaussian random numbers and the amount of multiplication and addition are directly proportional to the bitlength, resulting in a linear increase in energy consumption as the bitlength increases. The overall consumption of CiM-BNN with 128 b is 1.342 μJ/image, which is twice that of 64-bit and four times that of 32-bit. Energy consumption of CiM-arrays, GRNG, and other components also conform to this linear relationship.

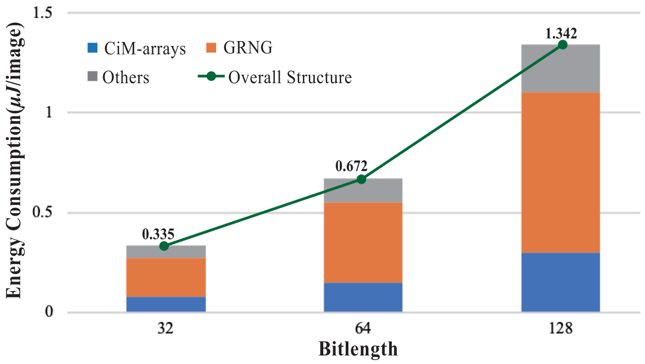


Fig. 11. Energy consumption with different bitlengths. The consumption of CiM-arrays is the sum of $\mu'_{ij}x_j$ and $h_{ij}\sigma'_{ij}x_j$. Overall structure contains the energy consumption of CiM-arrasys, GRNG and the other remaining components.

TABLE IV
ENERGY COMPARISON WITH RELATED WORK

	Platform	Parameter	Energy (μJ/image)
DigtBNN [17]	FPGA	8-bit FixP	52.608
StocBNN [9]	FPGA	bitlength=128	20.862
A.Malhotra [25]	ASIC+RRAM	4-bit FixP	9.383
W2W-PIM [24]	ASIC+RRAM	8-bit FixP	2.790*
BayBNN [26]	ASIC+MRAM	1-bit FixP	2.000
CiM-BNN	ASIC+MRAM	bitlength=128	1.342

* The energy is obtained by comparing and converting the energy consumption between [20] and [24].

Table IV presents a comparison of CiM-BNN with some related works of different implementation platforms. Compared to StocBNN [9] that is implemented on FPGA, CiM-BNN has the same computation process but achieves a 93.6% reduction in energy consumption. When compared to the computing-in-RRAM architecture in [25] and [24], CiM-BNN reduces the energy consumption by 85.7% and 51.9%, respectively. Compared to the binary BNN implementation in BayBNN [26], which has a simpler data representation, CiM-BNN still reduces energy consumption by 32.9% due to more efficient architecture. The read latency of RRAM computing-in-memory architecture is reported to be 10 ns in [25] and 100 ns in [24]. In this work, the read latency of MRAM is only 3 ns. The CiM-arrays can be divided into multiple small arrays for parallel computation that could improve the computation latency of CiM-BNN. This approach may increase the circuit area by adding some additional circuits such as PCSA, MUX and so on, but the area of CiM-arrays won't increase. The trade-off between circuit area and computation latency could be made based on specific application scenarios.

VII. Conclusion

This paper designs the computing-in-MRAM architecture for BNN in SC domain. CiM-BNN combines the advantages of BNN, SC, eNVMs and in-memory computing. It resolves the disadvantages of high computational complexity and high hardware resource consumption of BNN, as well as the "memory wall" of traditional von-Neumann architecture. On the premise that Gaussian random numbers can be represented by bitstream and bitstream can participate in feed-forward propagation, this paper utilizes binary feature of MTJ to realize high efficiency in-memory computing. Cadence Virtuoso is used to simulate the proposed architecture. Compared with FPGA implementation, CiM-BNN maintains inference accuracy well and reduces the overall energy consumption by 93.6%. In the future, we will try to improve the utilization efficiency of circuits and design more computing-in-memory architectures suitable for complex Bayesian neural networks.

References

[1] Z. Ghahramani, "Probabilistic machine learning and artificial intelligence," *Nature*, vol. 521, no. 7553, pp. 452–459, 2015.

[2] T. Yuan, W. Liu, J. Han, and F. Lombardi, "High performance CNN accelerators based on hardware and algorithm co-optimization," *IEEE Trans. Circuits Syst. I, Regular Papers*, vol. 68, no. 1, pp. 250–263, Jan. 2021.

[3] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision?," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5574–5584.

[4] C. R. N. Tassi, "Bayesian convolutional neural network: Robustly quantify uncertainty for misclassifications detection," in *Proc. Mediterranean Conf. Pattern Recognit. Artif. Intell.*, 2019, pp. 118–132.

[5] H. Park, A. Haghani, and X. Zhang, "Interpretation of Bayesian neural networks for predicting the duration of detected incidents," *J. Intell. Transp. Syst.*, vol. 20, no. 4, pp. 385–400, 2016.

[6] W. Liu, F. Lombardi, and M. Shulte, "A retrospective and prospective view of approximate computing," in *Proc. IEEE*, vol. 108, no. 3, pp. 394–399, Mar. 2020.

[7] H. Ichihara, S. Ishii, D. Sunamori, T. Iwagaki, and T. Inoue, "Compact and accurate stochastic circuits with shared random number sources," in *Proc. IEEE 32nd Int. Conf. Comput. Des.*, 2014, pp. 361–366.

[8] A. Mondal and A. Srivastava, "Power optimizations in MTJ-based neural networks through stochastic computing," in *Proc. IEEE Int. Symp. Low Power Electron. Des.*, 2017, pp. 1–6.

[9] X. Jia et al., "An energy-efficient Bayesian neural network implementation using stochastic computing method," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published, doi: 10.1109/TNNLS.2023.3265533.

[10] A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory devices and applications for in-memory computing," *Nature Nanotechnol.*, vol. 15, no. 7, pp. 529–544, 2020.

[11] S. Yu, "Neuro-inspired computing with emerging nonvolatile memorys," in *Proc. IEEE*, vol. 106, no. 2, pp. 260–285, Feb. 2018.

[12] K. Zhang et al., "High on/off ratio spintronic multi-level memory unit for deep neural network," *Adv. Sci.*, vol. 9, no. 13, 2022, Art. no. 2103357.

[13] W.-H. Chen et al., "CMOS-integrated memristive non-volatile computing-in-memory for ai edge processors," *Nature Electron.*, vol. 2, no. 9, pp. 420–428, 2019.

[14] M. Le Gallo et al., "Mixed-precision in-memory computing," *Nature Electron.*, vol. 1, no. 4, pp. 246–253, 2018.

[15] S. Yu, W. Shim, X. Peng, and Y. Luo, "RRAM for compute-in-memory: From inference to training," *IEEE Trans. Circuits Syst. I, Regular Papers*, vol. 68, no. 7, pp. 2753–2765, Jul. 2021.

[16] Z. Guo et al., "Spintronics for energy- efficient computing: An overview and outlook," in *Proc. IEEE*, vol. 109, no. 8, pp. 1398–1417, Aug. 2021.

[17] X. Jia, J. Yang, R. Liu, X. Wang, S. D. Cotofana, and W. Zhao, "Efficient computation reduction in Bayesian neural networks through feature decomposition and memorization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 4, pp. 1703–1712, Apr. 2021.

[18] Q. Wan and X. Fu, "Fast-BCNN: Massive neuron skipping in Bayesian convolutional neural networks," in *Proc. IEEE/ACM 53rd Annu. Int. Symp. Microarchitecture*, 2020, pp. 229–240.

[19] H. Fan et al., "FPGA-based acceleration for Bayesian convolutional neural networks," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 41, no. 12, pp. 5343–5356, Dec. 2022.

[20] R. Cai et al., "VIBNN: Hardware acceleration of Bayesian neural networks," in *Proc. Int. Conf. ACM Architectural Support Program. Lang. Operating Syst.*, 2018, pp. 476–488.

[21] H. Awano and M. Hashimoto, “B2N2: Resource efficient Bayesian neural network accelerator using Bernoulli sampler on FPGA,” *Integration*, vol. 89, pp. 1–8, 2023.

[22] K. Yang, A. Malhotra, S. Lu, and A. Sengupta, “All-spin Bayesian neural networks,” *IEEE Trans. Electron Devices*, vol. 67, no. 3, pp. 1340–1347, Mar. 2020.

[23] Y. Lin et al., “Uncertainty quantification via a memristor Bayesian deep neural network for risk-sensitive reinforcement learning,” *Nature Mach. Intell.*, vol. 5, pp. 714–723, 2023.

[24] X. Li et al., “Enabling high-quality uncertainty quantification in a pim designed for Bayesian neural network,” in *Proc. IEEE Int. Symp. High-Perform. Comput. Archit.*, 2022, pp. 1043–1055.

[25] A. Malhotra, S. Lu, K. Yang, and A. Sengupta, “Exploiting oxide based resistive RAM variability for Bayesian neural network hardware design,” *IEEE Trans. Nanotechnol.*, vol. 19, pp. 328–331, 2020.

[26] S. T. Ahmed, K. Danouchi, C. Münch, G. Prenat, L. Anghel, and M. B. Tahoori, “SpinDrop: Dropout-based Bayesian binary neural networks with spintronic implementation,” *IEEE Trans. Emerg. Sel. Topics Circuits Syst.*, vol. 13, no. 1, pp. 150–164, Mar. 2023.

[27] L. V. Jospin, H. Laga, F. Boussaid, W. Buntine, and M. Bennamoun, “Hands-on Bayesian neural networks–a tutorial for deep learning users,” *IEEE Comput. Intell. Mag.*, vol. 17, no. 2, pp. 29–48, May 2022.

[28] A. Graves, “Practical variational inference for neural networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 2348–2356.

[29] W. J. Gross and V. C. Gaudet, *Stochastic Computing: Techniques and Applications*. Berlin, Germany: Springer, 2019.

[30] G. Rodrigues, F. Lima Kastensmidt, and A. Bosio, “Survey on approximate computing and its intrinsic fault tolerance,” *Electronics*, vol. 9, no. 4, 2020, Art. no. 557.

[31] W. Liu and F. Lombardi, *Approximate Computing*. Berlin, Germany: Springer, 2022.

[32] A. Alaghi and J. P. Hayes, “Survey of stochastic computing,” *ACM Trans. Embedded Comput.*, vol. 12, no. 2s, 2013, Art. no. 92.

[33] T.-H. Chen and J. P. Hayes, “Analyzing and controlling accuracy in stochastic circuits,” in *Proc. IEEE 32nd Int. Conf. Comput. Des.*, 2014, pp. 367–373.

[34] J. Doevenspeck et al., “SOT-MRAM based analog in-memory computing for DNN inference,” in *Proc. IEEE Symp. VLSI Technol.*, 2020, pp. 1–2.

[35] Y. Wang, K. Zhang, B. Wu, D. Zhang, H. Cai, and W. Zhao, “Magnetic random-access memory-based approximate computing: An overview,” *IEEE Nanotechnol. Mag.*, vol. 16, no. 1, pp. 25–32, Feb. 2022.

[36] K. Cao et al., “In-memory direct processing based on nanoscale perpendicular magnetic tunnel junctions,” *Nanoscale*, vol. 10, no. 45, pp. 21 225–21 230, 2018.

[37] Y. Hirayama, T. Asai, M. Motomura, and S. Takamaeda-Yamazaki, “A resource-efficient weight sampling method for Bayesian neural network accelerators,” in *Proc. IEEE Int. Symp. Comput. Netw.*, 2019, pp. 137–142.

[38] H. Awano and M. Hashimoto, “BYNQNet: Bayesian neural network with quadratic activations for sampling-free uncertainty estimation on FPGA,” in *Proc. Des. Automat. Test Eur. Conf. Exhib.*, 2020, pp. 1402–1407.

[39] R. Cai, A. Ren, L. Wangy, M. Pedramy, and Y. Wang, “Hardware acceleration of Bayesian neural networks using RAM based linear feedback Gaussian random number generators,” in *Proc. IEEE Int. Conf. Comput. Des.*, 2017, pp. 289–296.

[40] X. Jia, J. Yang, P. Dai, R. Liu, Y. Chen, and W. Zhao, “SPINBIS: Spintronics based Bayesian inference system with stochastic computing,” *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 39, no. 4, pp. 789–802, Apr. 2020.

[41] C. C. Yann LeCun and C. J. Burges, “MNIST handwritten digit database,” 2010. [Online]. Available: http://yann.lecun.com/exdb/mnist

[42] H. Xiao, K. Rasul, and R. Vollgraf, “Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms,” 2017, *arXiv: 1708.07747*.

[43] D. Tran, M. D. Hoffman, R. A. Saurous, E. Brevdo, K. Murphy, and D. M. Blei, “Deep probabilistic programming,” in *Proc. Int. Conf. Learn. Representations*, 2017, pp. 1–18.

[44] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” in *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[45] S. R. Faraji, M. H. Najafi, B. Li, D. J. Lilja, and K. Bazargan, “Energy-efficient convolutional neural networks with deterministic bit-stream processing,” in *Proc. IEEE/ACM Proc. Des. Automat. Test Eurpoe*, 2019, pp. 1757–1762.

Huiyi Gu (Student Member, IEEE) received the BS degree in electronic information engineering from Beihang University, Beijing, China, in 2017. She is currently working toward the doctor degree with the School of Electronic and Information Engineering, Beihang University, Beijing, China. Her research interests include Bayesian deep learning and computing-in-memory architecture.

Xiaotao Jia (Member, IEEE) received the BS degree in mathematics from Beijing Jiao Tong University, Beijing, China, in 2011, and the PhD degree in computer science and technology from Tsinghua University, Beijing, China, in 2016. He is currently an associate professor with the School of Integrated Circuit Science and Engineering, Beihang University, Beijing, China. From 2016 to 2019, he was a postdoctoral researcher with the School of Integrated Circuit Science and Engineering, Beihang University. His current research interests include spintronic circuits, stochastic computing, Bayesian deep learning, and EDA.

Yuhao Liu (Student Member, IEEE) received the BS degree in electronic information engineering from Beihang University, Beijing, China, in 2020. He is currently working toward the master's degree with the School of Electronic and Information Engineer, Beihang University, Beijing, China. His research interests include analog circuit design and neuron computing.

Jianlei Yang (Member, IEEE) received the BS degree in microelectronics from Xidian University, Xi’an, China, in 2009, and the PhD degree in computer science and technology from Tsinghua University, Beijing, China, in 2014. He is currently an associate professor with the School of Computer Science and Engineering, Beihang University, Beijing, China. From 2014 to 2016, he was a postdoctoral researcher with the Department of ECE, University of Pittsburgh, Pennsylvania, USA. His current research interests include computer architectures and neuromorphic computing systems. He was the recipient of the First/Second place on ACM TAU Power Grid Simulation Contest in 2011/2012. He was a recipient of IEEE ICCD Best Paper Award in 2013, ACM GLSVLSI Best Paper Nomination in 2015, IEEE ICESS Best Paper Award in 2017, ACM SIGKDD Best Student Paper Award in 2020.

Xueyan Wang (Member, IEEE) received the BS degree in computer science from Shandong University, Jinan, China, in 2013, and the PhD degree in computer science and technology from Tsinghua University, Beijing, China, in 2018. From 2015 to 2016, she was a visiting scholar with the University of Maryland, College Park, MD, USA. She is currently a postdoctoral researcher with the School of Integrated Circuit Science and Engineering in Beihang University, Beijing, China. Her current research interests include highly efficient processing-in-memory (PIM) architectures and hardware security.

Youguang Zhang (Member, IEEE) received the MS degree in mathematics from Peking University, Beijing, China, in 1987, and the PhD degree in communication and electronic systems from Beihang University, Beijing, in 1990. He is currently a professor with the School of Electronic and Information Engineering, Beihang University. His research interests include microelectronics and wireless communication. In particular, he recently focuses on the circuit and system codesign for the emerging memory and computing systems.

Sorin Dan Cotofana (Fellow, IEEE) received the MSc degree in computer science from the "Politehnica" University of Bucharest, Romania, in 1984, and the PhD degree in electrical engineering from the Delft University of Technology, Delft, The Netherlands. He is currently with the Faculty of Electrical Engineering, Mathematics and Computer Science, the Computer Engineering Laboratory, Delft University of Technology, The Netherlands. He has coauthored more than 250 papers in peer-reviewed international journal and conferences, and received 12 best paper awards in international conferences. His current research interests include the following: 1) the design and implementation of dependable/reliable systems out of unpredictable/unreliable components; 2) aging assessment/prediction and lifetime reliability aware resource management; and 3) unconventional computation paradigms and computation with emerging nanodevices. He is currently the editor in chief of *IEEE Transactions on Nanotechnology*, associate editor for *IEEE Transactions on Computers*, and *IEEE Circuits and Systems Society (CASS)* distinguished lecturer and Board of Governors member.

Weisheng Zhao (Fellow, IEEE) received the PhD degree in physics from the University of Paris Sud, Paris, France, in 2007. He is currently the professor with the School of Microelectronics, Beihang University, Beijing, China. In 2009, he joined the French National Research Center(CNRS), as a tenured research scientist. Since 2014, he has been a distinguished professor with Beihang University, Beijing, China. He has published more than 200 scientific articles in leading journals and conferences, such as *Nature Electronics, Nature Communications, Advanced Materials, IEEE Transactions*, ISCA and DAC. His current research interests include the hybrid integration of nano-devices with CMOS circuit and new nonvolatile memory (40-nm technology node and below) like MRAM circuit and architecture design. He is currently the editor-in-chief for *IEEE Transactions on Circuits and Systems I: Regular Paper*.