MERRC: A Memristor-Enabled Reconfigurable Low-Power Reservoir Computing Architecture at the Edge

Fabiha Nowshin, Yi Huang, Graduate Student Member, IEEE, Md. Rubel Sarkar, Qiangfei Xia, Fellow, IEEE, and Yang Yi, Senior Member, IEEE

Abstract— The massive growth in the Internet of Things (IoT) has led to an increase in demand for devices receiving and transmitting data to and from the cloud during operations. Edge computing has been developed in an attempt to bring computations in proximity to the devices to overcome latency and cost overhead. IoT applications heavily reliant on machine learning (ML) tasks such as image or voice recognition can benefit from edge devices that facilitate real-time operations without costly data transmission back and forth from memory. In this work, we develop MERRC, a memristor-enabled reconfigurable architecture that incorporates processing-in-memory and reservoir computing to carry out ML tasks of image classification at the edge. This design uses a novel masking circuit to allow for image segmentation, which is combined with a delay-based reservoir to form a recurrent neural network. We further implement the final stage of classification using a fabricated memristor crossbar. Our hardware measurement results on the MNIST dataset of the delay-based reservoir with memristor crossbar arrays provide a recognition accuracy of 98%. On a more complex image classification dataset of CIFAR-10, MERRC shows a high accuracy of 88%, further highlighting the edge computing capabilities of the architecture.

Index Terms— Edge device, memristor, image recognition, reservoir computing, processing-in-memory, analog integrated circuit design.

I. Introduction

THE advent of machine learning (ML) has paved the way for many applications in the modern world such as speech recognition, natural language processing and developing software for computer vision [1]. While traditional ML methods faced limitations when analyzing natural data in its original form, deep learning, a subset of machine learning, is independent of prior data processing and possesses the capacity to automatically extract features from extensive datasets [2], [3]. Multiple processing layers are used in deep learning where the preceding layer extracts some features before sending the data to the next layer [3]. Deep learning, often referred to as a universal learning approach, has achieved notable success in speech recognition and image classification. Convolutional Neural Network (CNN), a branch of deep learning, has shown extremely high, state-of-the-art accuracies on ImageNet task, while a 3.57% error rate in the ResNet-152 task, outperforming human accuracy [4], [5], [6], [7], [8]. With the rapid increase in the Internet of Things (IoT), there has been a massive growth in the data on edge devices such as smartphones, IoT sensors and notebook computers, to bring computations closer to the devices, a paradigm known as edge computing [9]. As depicted in Fig. 1, when transitioning from cloud computing to edge computing, these edge devices dependent on ML tasks benefit from leveraging deep learning algorithms on applications like image and speech recognition, overcoming the costly data transmission to and from the cloud.

The implementation of deep learning algorithms in application-specific integrated circuit (ASIC) chips is crucial to edge devices [10], [11]. When deployed on edge devices, deep learning becomes exceptionally advantageous, especially in domains like natural language processing, computer vision, and the IoT [12], [13]. However, their efficient deployment at the edge is becoming increasingly difficult due to inefficiencies in the conventional von Neumann architectures [14]. Executing computations on traditional von Neumann computer systems entails a significant volume of data moving between physically distinct memory and processing components [15]. This data transfer incurs not only time and energy costs but also creates a fundamental bottleneck in terms of performance. Hence the conventional von Neumann computing architecture presents significant issues related to both physical space utilization and power consumption when dealing with the substantial computational demands and vast data volumes inherent in AI tasks. One promising alternative is processing-in-memory (PIM) architectures, which offer a compelling solution to the constraints of von Neumann architectures [16]. In PIM systems, processing tasks are executed directly in or near memory, significantly reducing the need for data to traverse between memory and processing units. Emerging embedded nonvolatile memory technologies (eNVMs) like memristors are more suited to performing PIM operations owing to their high speed, low power consumption and a small area on chip,

Manuscript received 23 March 2023; revised 20 September 2023; accepted 21 October 2023. This work was supported in part by the U.S. National Science Foundation (NSF) under Grant CCF-1750450, Grant ECCS-1731928, Grant ECCS-2128594, Grant ECCS-2314813, Grant CCF-1937487, and CCF-2133475. This article was recommended by Associate Editor X. Liu. (Corresponding author: Yang Yi.)

Fabiha Nowshin, Md. Rubel Sarkar, and Yang Yi are with the Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA 24061 USA (e-mail: fabiha27@vt.edu; rubel@vt.edu; yangyi8@vt.edu).

Yi Huang and Qiangfei Xia are with the Department of Electrical and Computer Engineering, University of Massachusetts Amherst, Amherst, MA 01003 USA (e-mail: yihuang@umass.edu; qxia@umass.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSI.2023.3329337.

Digital Object Identifier 10.1109/TCSI.2023.3329337

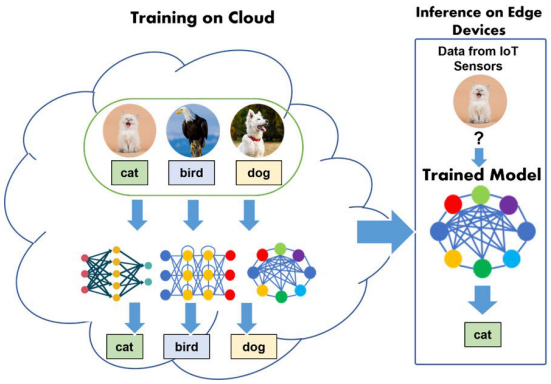


Fig. 1. IoT data on edge devices.

all critical for performing deep learning algorithms at the edge [17].

Arising from the multiple layers and training complexity, implementing deep learning algorithms on ASIC is especially challenging on edge devices due to their constraints on area and power consumption [11]. Furthermore, the noise and imperfections in circuit components add to the inaccuracies in the model. A more hardware-friendly approach is reservoir computing, a type of recurrent neural network (RNN) that can resolve some of the issues of training complexity by only training the output layer. The spatiotemporal information processing capabilities of reservoir computing have allowed it to solve complex tasks in the areas of time-series prediction, classification and segmentation, de-noising, and channel equalization and control [18]. However, the large number of neurons inside the reservoir becomes challenging to implement in the hardware [19]. A delayed dynamical reservoir, which is a subset of reservoir computing, presents a promising choice for hardware implementations. In this approach, the reservoir's intricate dynamics can be replaced with a single nonlinear node followed by a delayed loop, effectively generating high-dimensional patterns. Utilizing such a simplified structure is beneficial when executing in hardware, bypassing the tedious connections in a traditional reservoir.

In this work, we address latency and power consumption issues, and take a hardware-friendly approach to develop a memristor-enabled reconfigurable low-power reservoir computing architecture (MERRC) at the edge. We primarily focus on the development of the architecture, including the input preprocessing through a mask followed by the reservoir nonlinear node and the feedback loop before integrating with the reconfigurable memristor crossbar to train the output layer. The MERRC architecture showcases reconfigurability through its utilization of reservoir computing as well. The reservoir, by adjusting its delay factor and parameters, can be tailored to different applications without altering the fundamental structure of the architecture. The reconfigurability of MERRC becomes evident when considering the varying requirements of different applications. By fine-tuning the memristor weights and adjusting the reservoir parameters, it can effectively adapt to different datasets and tasks. The key contributions of our work are summarized below:

- We combine a delayed dynamical feedback reservoir system (DeDFRS) known for its proficiency in spatiotemporal information processing and hardware simplicity with a reconfigurable memristor crossbar tailored for PIM. This facilitates training in the output layer, effectively bringing computation nearer to the data source and providing support for edge devices.
- For the input preprocessing, we design the first analog masking circuit to obtain variation and add sequentialization in response to the input to fully utilize the available dimensionality in the DeDFRS.
- We further integrate a fabricated memristor crossbar array in the PIM architecture to carry out the output layer training of the DeDFRS. Each memristor cell can be reprogrammed to the specific weight values that is determined by the output training, making the architecture reconfigurable.
- We demonstrate our circuits and architecture on the MERRC board with fabricated memristor crossbar arrays and carry out evaluations on the MNIST dataset, achieving an accuracy of 98%.
- Further simulation analysis on the more complex dataset of CIFAR-10 shows a very high accuracy of 88%, indicating the capability of the design for a more thorough image recognition task, making it suitable for edge devices.

II. Edge Computing on ASIC

Data from IoT endpoints often demand real-time processing in order to reduce the data footprint in AI/ML applications. Field Programmable Gate Arrays (FPGAs) and Graphical Processing Units (GPUs) are suitable hardware devices that can be used for training and inference in such applications. Unlike FPGAs and GPUs that can be readily configured by the designer at a much lower cost, ASIC-based hardware is customizable for a specific application and cannot be used for general purposes [20]. While FPGAs and GPUs offer cost-effective reconfigurability, they are burdened by high power consumption and occupy significant space. To enable real-time data processing, it becomes crucial to move computations closer to the data source, a strategy known as edge computing. In contrast to FPGAs and GPUs, ASICs are better suited for edge computing due to their advantages of low power consumption, increased security, compact size and high speed in processing real-time data. The Edge TPU, Google's ASIC for edge computing uses the Cloud IoT and Google's Cloud TPU to enable ML applications at the edge using AI algorithms in hardware with low power and area [11], [21]. The ShiDianNao is another example of an ASIC-based solution for edge computing that specializes on image classification tasks, and when used for CNN-based deep learning tasks it provides a 30x faster speed than NVIDIA K20M GPU [11], [22]. In this section, we highlight the concepts of PIM and reservoir computing to explain their functionality and usage in edge devices.

A. Memristor-Based Processing-in-Memory

Processing-in-memory (PIM) is a non-von Neumann computing paradigm that was developed in an attempt to alleviate the issue of the Memory Wall [16], [17] By carrying out

computations near or in memory, PIM has the potential to achieve massive parallelism especially in deep learning operations. The computation process of deep learning involves large amounts of vector-matrix multiplications (VMMs) in the different layers that involve storing and fetching weights to and from the memory hierarchy. Utilizing PIM for VMM operations can significantly improve performance in terms of power and latency, saving ample amount of time in transmitting and receiving data [23].

In recent years, PIM has received immense attention in the area of deep learning owing to its ability to accelerate ML operations by carrying out computations at the source of the data. There have been several notable works on PIM-based deep learning accelerators using SRAM, DRAM and emerging eNVMs as the core memory element [24], [25], [26], [27], [28], [29]. However, the traditional memory technologies of DRAM and SRAM rely on the charge storage phenomenon and as technology scales, there is a high possibility of unreliability arising from lost charges [30]. Memristor, a particular emerging eNVM, is a promising candidate for PIM operations and is considered suitable for analog edge computing for processing data generated from IoT devices [31]. With their nonvolatile memory property, compatibility to be integrated with CMOS technology, ability to mimic biological synapses, and their analog conductance modulation properties, memristors have successfully carried out ML tasks in hardware [30], [32], [33], [34]. Memristors have a compact size in the scale of nanometers and can be used to form dense crossbar structures. Their fast-switching speed and capacity to store data and perform computations on them makes memristors a convenient replacement for their SRAM and DRAM counterparts and hence suitable to be used in edge devices.

B. Reservoir Computing

The two major types of deep neural networks (DNNs) are feedforward neural networks (FNNs) that possess the ability to process static input data and recurrent neural networks (RNNs) that can process both temporal and spatial input data. In contrast to FNNs, RNNs exhibit dynamic behavior due to recurrent connections in the hidden layer, allowing data to persist within the network for a certain duration [35]. While RNNs closely mimic the biological nervous system, they entail complex, resource-intensive, and time-consuming training procedures. As a response to this challenge, reservoir computing emerged as a potential solution by significantly simplifying training, focusing solely on the output layer [19]. By employing small training datasets and linear optimization, reservoir computing demands fewer computational resources.

Generally, reservoir computing has three interconnected layers of neurons, the input layer, the physical reservoir and the output layer, where the activations of the network units are given by $x(t) = (x_1(t), \ldots, x_N(t))$, $r(t) = (r_1(t), \ldots, r_N(t))$ and $y(t) = (y_1(t), \ldots, y_N(t))$ respectively for each time step t. The activations between the internal units are carried out using Equation (1) with $\alpha = (\alpha_1, \ldots, \alpha_M)$ as the activation function of the internal units inside the reservoir,

$$r(t+1) = \alpha W^{in}\{x(t+1) + W r(t) + W^{ir} y(t)\} \quad (1)$$

$$y(t+1) = \beta W^{out}\{x(t+1), r(t+1), y(t)\} \quad (2)$$

where W^{in} is the weight matrix between the input and the internal units and W is the internal weight matrix inside the reservoir. The weight matrix from the output to the internal reservoir is W^{ir} and the weight matrix from the reservoir to the output is W^{out} in Equation (2). In order to prevent the issue of vanishing gradient in the training procedure, the connections between the input and the reservoir are randomly initialized and the connections in the output layer are trained via a regularized linear least-square optimization technique, drastically streamlining the training process [36], [37].

Echo State Networks (ESNs) and Liquid State Machines (LSMs) are two subsets of reservoir computing that follow the fundamental idea that the connections from both the input to the reservoir as well as the internal connections inside the reservoir are fixed and random. While both the networks follow the procedure of training the output layer, the key difference lies in the structure of ESN and LSM, where ESN is a rate-based approximation and LSM is based on biologically inspired spiking neural network (SNN) [38]. Implementing ESN and LSM using ASICs proves to be considerably resource-intensive due to the extensive connections within the reservoir. In contrast, the delayed dynamical feedback reservoir system (DeDFRS), an innovation derived from reservoir computing, significantly reduces the resource requirements and time consumption by substituting the entire reservoir with a delay node. Owing to its simplicity and nonlinear nature, various implementations of DeDFRS have emerged in the fields of electronics, optoelectronics, and optics [39], [40], [41], [42]. By introducing the dynamic and feedback properties the DeDFRS can replicate the operating procedure of the typical reservoir, making it a favorable choice for hardware implementations, especially well-suited for edge devices.

III. MERRC Architecture

In this paper, we introduce a reconfigurable memristor-based processing-in-memory architecture with reservoir computing, MERRC, designed for edge devices. Our primary aim is to bring computations closer to the source of the data to overcome the issues of latency and power consumption. With the use of the DeDFRS, this architecture benefits from the spatial temporal information processing capability of RNNs while having a simplified hardware structure, significantly reducing the number of resources and reducing the area consumption. As depicted in Fig. 2, since the DeDFRS has the advantages of overcoming the training complexity by only training the output layer, we incorporate a memristor-based PIM architecture to allow for classification. Our memris-tor crossbars are reconfigurable, implying that they can be reprogrammed to required values based on the training of edge devices, providing the architecture of the flexibility lacking in general ASIC-based implementations. Furthermore, the MERRC architecture demonstrates its versatility through reservoir computing. By adjusting delay factors and parameters, the reservoir can be customized for various applications without altering the core structure. This adaptability shines

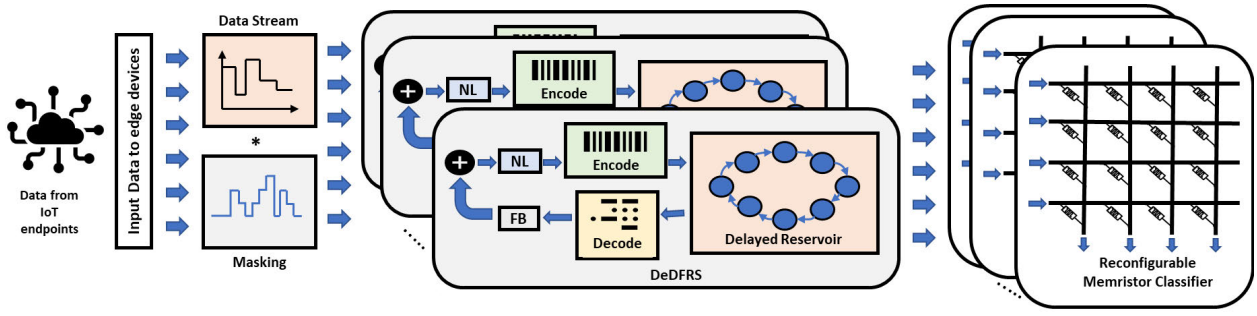


Fig. 2. Overview of the memristor-enabled reconfigurable architecture using reservoir computing (MERRC).

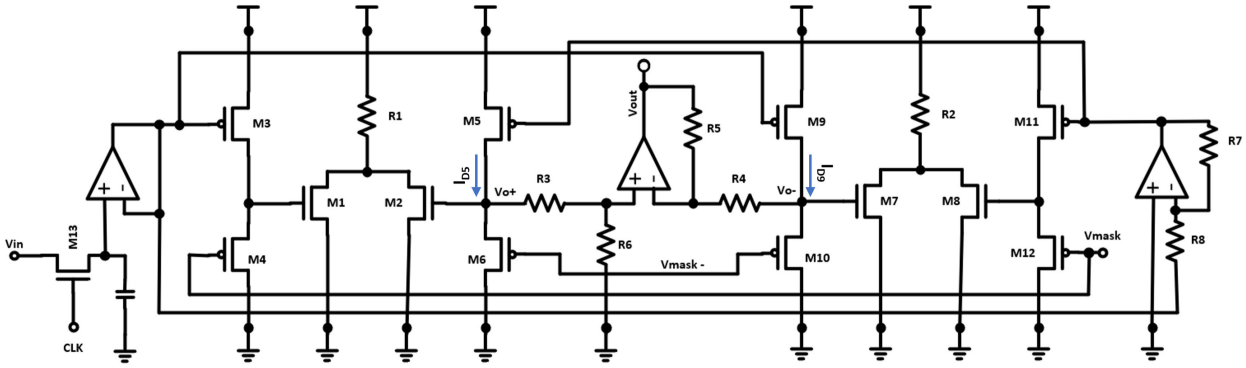


Fig. 3. Implementation of masking.

when addressing diverse application requirements. MERRC can effectively adapt to a range of datasets and tasks by reprogramming the memristor weights and the reservoir parameters. This section covers the details of our implementation, focusing on the circuit designs of MERRC, including the novel masking design and the design of the DeDFRS and the integration of our fabricated memristor crossbar arrays.

A. Masking in DeDFRS

As discussed in the preceding section, reservoir computing significantly reduces the training complexity by only training the output layer of the neural network while possessing the spatiotemporal information processing capability. Time-delayed reservoir systems further simplifies this procedure by undergoing time-multiplexing of a nonlinear neuron onto a delayed feedback loop forming the RNN [43], [44], [45]. The high-dimensional states for computation are provided by the virtual nodes specified along the time delay. In the time-delayed reservoir, one nonlinear node is responsible for driving all the virtual nodes in the delay line, and since the states of every node are generated from the same transformation, scaling factors cannot be introduced in the virtual nodes for variability in the network. However, a masking technique is used at the time-delayed reservoir's input to acquire the variety in the virtual nodes' responses to an input signal. In the past, this procedure has been carried out by randomly introducing a series of values, each allocated to a virtual reservoir node [46], [47], [48], [49].

For optimal utilization of the newly established state space within the reservoir, it is essential to introduce diversity in the input sequence. To achieve uniform connectivity between every virtual node and the processed input signal, the masking procedure is replicated for each temporal delay in the reservoir. In earlier implementations of masking, waveform generators were employed to time-multiplex input data in the fields of optoelectronics and optics [50], [51], [52]. While there have been some works on implementing the masking procedure solely on software as part of the input preprocessing technique [43], there have been other works on generating a digital binary mask and a six-level digital mask [53], [54]. There is a lack of proper implementation of masking for reservoir computing in the analog domain as well as the ability to time-multiplex the input data with the masking signal. Therefore, in this work, we design a novel masking circuit, depicted in Fig. 3, that combines both the input and the masked signal for the nonlinear node in the reservoir.

To imprint coupling weights from the input to the reservoir, W_i^{in}, onto the input data stream, a piecewise function, the masking function, can be used with a certain period τ and in intervals of ϕ. Equations (3) and (4) defines such a masking function and the corresponding output resulting from the imprinting of the input,

$$V_{mask} = W_i^{in} \quad \text{for} \quad (i-1)\phi \le t \le i\phi \tag{3}$$

$$V_{out}(t) = \sum_{k=1}^{N} V_{in}^{k}(t) \cdot V_{mask}^{k}(t). \tag{4}$$

To implement Equation (4), we use our design from Fig. 3 where V_{in} is the input data stream which is initially sampled using the clock signal CLK and the output signal V_{in^+} is

applied to the gates of the transistors M_3 and M_9. Using the opamp inversion technique, the sampled input signal V_{in} in converted to V_{in-} to drive the transistors M_5 and M_{11}. The masking function V_{mask}, responsible for the variations in the input data has its positive node applied at the gates of the transistors M_4 and M_{12}, and the negative node applied at the gates of transistors M_6 and M_{10}.

The design from Fig. 3 utilizes the operational procedure of a differential amplifier with the current mirror structure. The current variation due to the voltage of V_{mask} is carried on to the branches of V_{o+} and V_{o-} through the mirror connection. Through the concept of transconductance g_m and gain in differential amplifier, Equation (5) can be formulated as,

$$A_v = \frac{V_{o+} - V_{o-}}{V_{in+} - V_{in-}} = -g_{m5,9} \quad R_{3,4} \parallel r_{o5,9} \parallel r_{o6,10}. \tag{5}$$

Using the relationship between the transconductance and the drain current I_D, we can rewrite the Equation (5) as follows,

$$V_{o+} - V_{o-} = \frac{r}{r} \frac{2K'(\frac{W}{L})_5 I_{D5}}{\overline{W}} 2V_{in}(R_3 \parallel r_{o5} \parallel r_{o6}) \tag{6}$$

$$V_{o+} - V_{o-} = K'^2(\frac{W}{L})_5(V_m - V_{tp})2V_{in} R_3 \parallel r_{o5} \parallel r_{o6} \tag{7}$$

where I_{D5} was replaced by the drain current in Equation (7) from the transistors M_6 and M_{10} which are operating in the saturation regime. The final output V_{out} can be calculated using Equation (8) and replacing $V_{o+} - V_{o-}$ from Equation (7) given by,

$$V_{out} = \frac{R_{5,6}}{R_{3,4}} \cdot (V_{o+} - V_{o-}). \tag{8}$$

The ratio of the resistors $R_{5,6}$ and $R_{3,4}$ is used to determine the amplification range of the output signal, which is the multiplied input and mask signal for each portion of the data stream in the intervals of ϕ.

B. Delayed Dynamical Feedback Reservoir

The complete structure of the developed DeDFRS core is depicted in Fig. 4, consisting of 5 major modules of nonlinearity, analog to spike encoder (ASE), neuron core, decoder core and feedback. The details of the circuits and architecture are discussed in this section.

1) Nonlinearity: In order to translate the incoming inputs nonlinearly onto outputs, we incorporate a type of activation function known as the Mackey-Glass function. Mathematical models of biological systems are often defined using delayed differential equations, where one equation can represent countless number of ordinary differential equations [55], [56]. A modified version of the delayed differential equation is the Mackey-Glass function shown in Equation (9) that includes the embedded delay property of dynamical systems, which is crucial when implementing the DeDFRS [57].

$$x_t = \frac{\beta u_\tau}{1 + u_\tau^n} - \alpha \cdot u \tag{9}$$

From Equation (9) it is evident that the current input depends on the present and the historical input data where u_τ is the input at the time of $(t - \tau)$. This delayed property allows the Mackey-Glass function to have a variable gradient which is essential in RNNs to avoid the issue of vanishing gradients. The scaling parameters of β and α as well as the nonlinear exponent n further assist in the tuning and shaping of the nonlinear function as well as increasing its strength to find the optimal hyperparameters [43] In the past there have been several analog electronic implementations of this function to control chaos in the analog delay line, owing to the simplistic implementation and hyperparameter tuning of the circuit [43], [58], [59].

To implement this function, we optimize and modify our circuitry from [60] to fit the Mackey-Glass equation using the circuit from Fig. 4. Without the use of any external supply, this circuit effectively tracks incoming input when the signal falls within the lower range, resulting in a positive gradient. As the signal surpasses the threshold of the nmos transistors, the output gradually discharges, yielding a negative gradient. In contrast to the previous implementation detailed in [60], this circuitry offers a broader range of output values, negating the necessity for additional transistor components and buffers. As a result, it significantly reduces both the circuit's footprint and latency.

2) Analog to Spike Encoder: Converting the incoming analog signals to spiking signals has the potential to offer higher power and energy efficiency, arising from the binary spiking nature of signals which can significantly simplify the circuitry [61]. Spiking signals emulate the neurons in the biological systems more closely, firing each time a defined threshold is exceeded [62]. Therefore, when implementing the DeDFRS, injecting spiking signals is a more viable option when adding the delay constant rather than applying the delay factor to a fully analog signal which will suffer from noise and low robustness.

Electronic neuron models can imitate the functionality of biological neurons, and to date there have been several implementations including the Hodgkin-Huxley, Izhikevich and Leaky-Integrate-and-Fire (LIF) models [63], [64], [65]. Compared to their counterparts, LIF neuron models possess a simplistic hardware implementation while incorporating the leakage factor similar to biological neurons offering efficiency in terms of power and latency [66]. Furthermore, the LIF neurons also offer a good approximation of the complex models of Hodgkin-Huxley and the Izhikevich neurons, deeming them a more suitable option. In this work, we incorporate these LIF neurons for the transformation of the analog signals however, the main working mechanism of the LIF neurons is based on a capacitor sensing technique which relies on the magnitude of the incoming current. We use a transconductance-based amplifier in our ASE block from Fig. 4 to convert the nonlinear outputs into current values. The diode-connected current mirror structure facilitates the elimination of the load variation affecting the output currents. The transistors (M_4, M_5), (M_6, M_7) and (M_8, M_9) are matched pairs and the linear scaled values of I_1 and I_2 are applied to the LIF neurons.

Once the currents enter the LIF neuron, the voltage of V_{leak} ensures a small leakage from the input while the capacitor is allowed to charge to a voltage potential of 0.8V specified at the gate of M_4 at the LIF neuron. This instigates the firing of

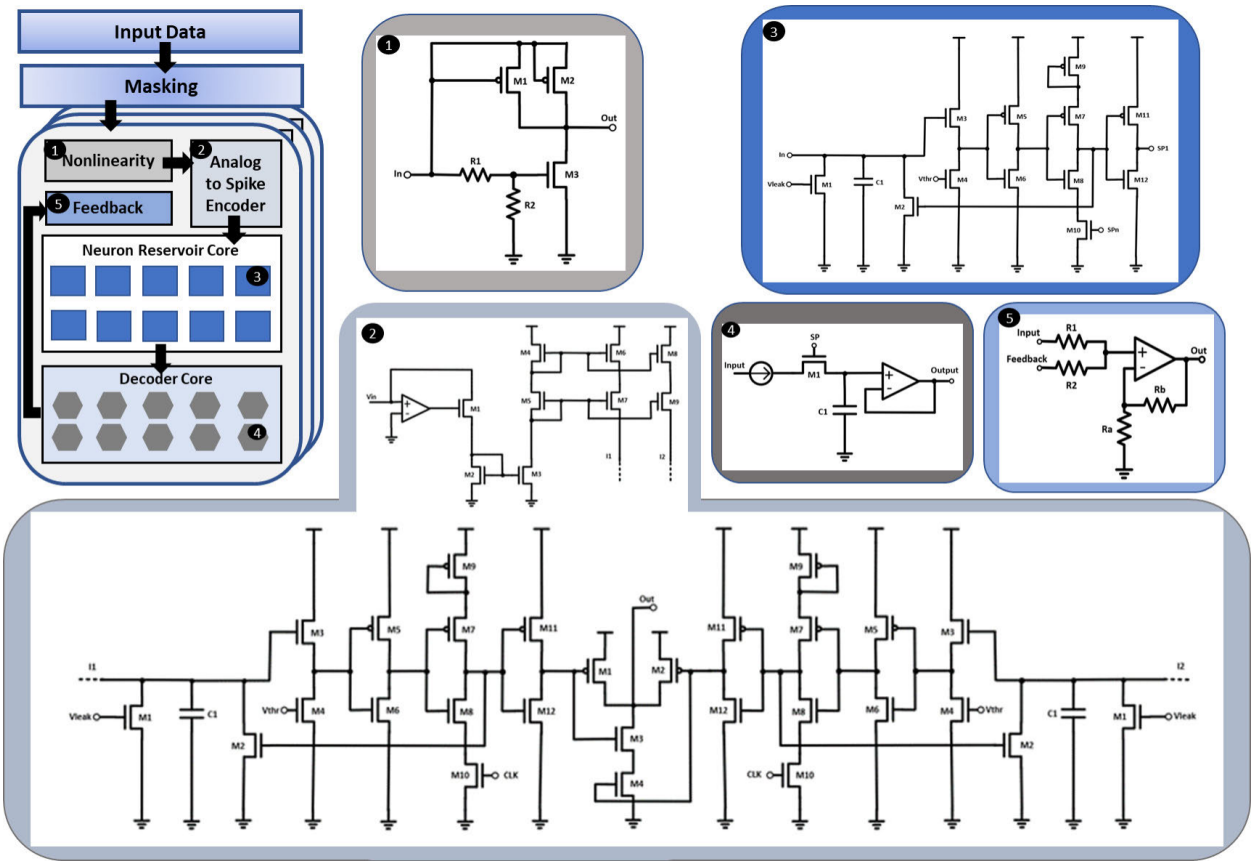


Fig. 4. Individual modules of the DeDFRS core.

the spike where the time is given by Equation (10).

$$t = C \int_{0}^{V_{thr}} \frac{1}{\frac{(W/L)_7}{(W/L)_5} I_1 - I_{leak}} dV \tag{10}$$

The spike signal is strengthened by the cascaded buffers in the neuron which also resets the charged capacitor with the transistor M_2 until the incoming voltage. A clock signal of 10MHz controls spike settings of LIF neurons influencing the window of the firing time. The output signal generated from this ASE is a combination of two spike trains where t_1 is the spike time from the first neuron and t_2 is the spike time of the second neuron.

3) Reservoir Core: The spike train originating from the ASE comprises two spikes during each clock cycle, and these are introduced into the reservoir core. This core is composed of a cluster of LIF neurons, interconnected sequentially as depicted in Fig. 2 within the MERRC architecture. These neurons work in the similar manner discussed in the preceding section, however, the incoming input current for capacitor charging is applied from the nonlinearly transformed input. The spike train generated from the former neuron is applied as the clock of the latter neuron, ensuing the subsequent spiking times in the neurons that follow. This creates a delay factor of τ inside the DeDFRS where the discrete reservoir state can be given by:

$$r_i(j) = x(j\tau - (N - i)\theta), \tag{11}$$

where for each time step j, x is the nonlinearly transformed input for each node from i to N, where N is the number of neurons in the reservoir core and θ is the interval between the neurons defined by τ/N.

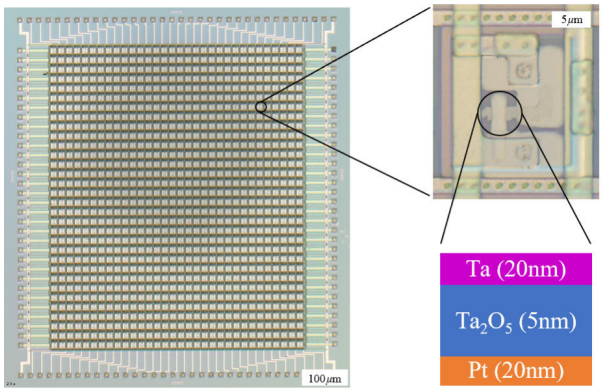


Fig. 5. Cross-section image of the memristor crossbar.

4) Decoder: The output from each neuron in the reservoir core is applied to the decoder core consisting of N number of modules demonstrated in Fig. 4. The incoming input is applied at the drain of the transistor M_1, while the spiking signal from each output neuron controls the charging of the capacitor in the decoder unit. The output signal from the decoder is strengthened by the buffer unit and the final output from the DeDFRS is given by Equation (12) where ρ is the scaling factor of the input.

$$y(t) = mg\{x(t), x(t - \tau) + \rho x(t)\} \tag{12}$$

5) Feedback: The DeDFRS requires a feedback module to incorporate the past and current input before injecting the input

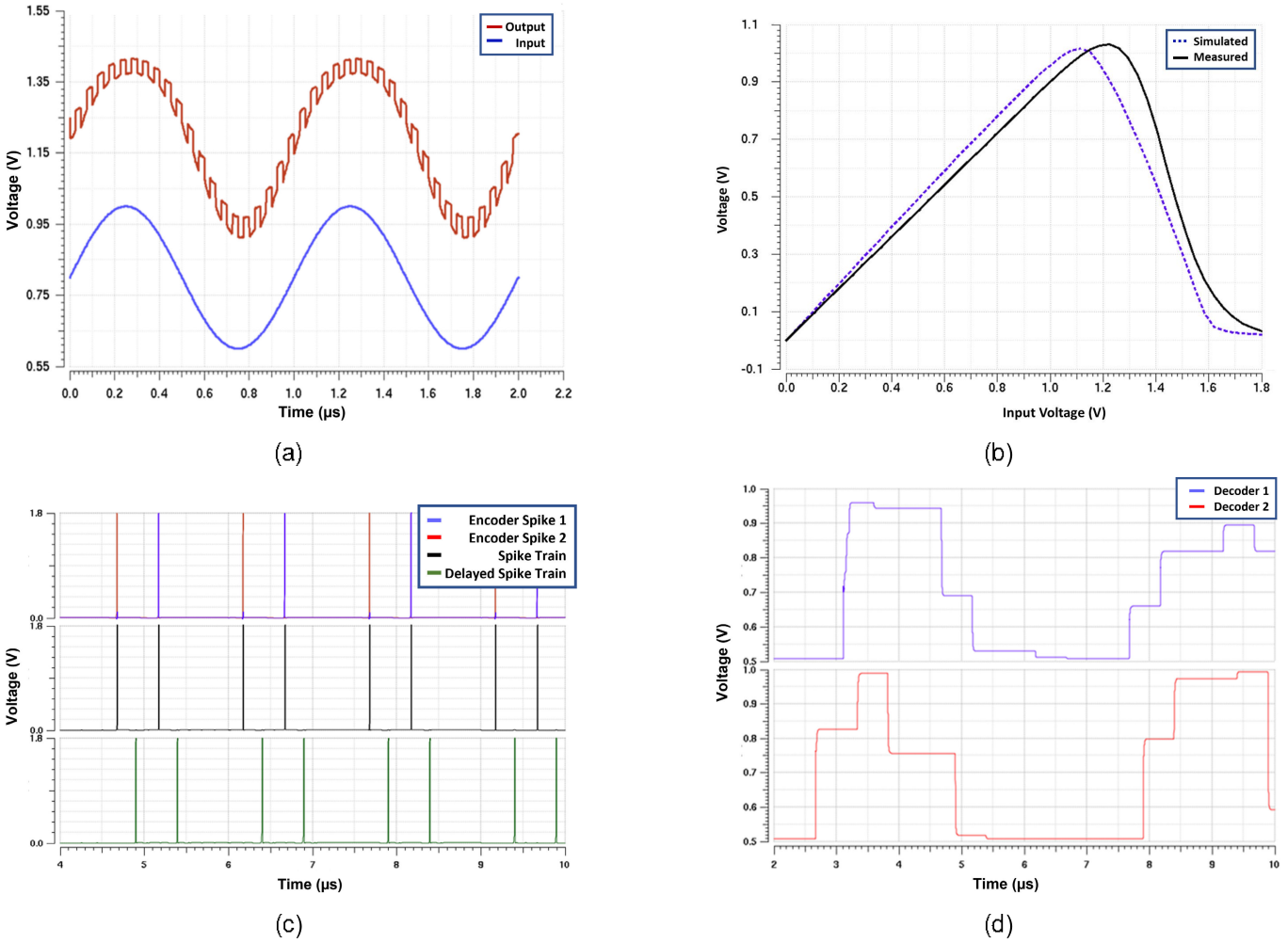


Fig. 6. Plot of the DeDFRS modules. (a) Masking circuit output. (b) Mackey-Glass function output. (c) Simulation of the ASE and delay modules in the DeDFRS. (d) Decoder output simulation results.

signal to the complete MERRC architecture. This feedback is implemented with a summing amplifier module shown in Fig. 4. By controlling the ratios of the resistors in the summing amplifier of R_1 and R_2, a smaller factor of the feedback signal from the previous loop in the DeDFRS is combined with an amplified incoming masked input to be applied to the loop in the MERRC architecture. This helps to improve the performance and accuracy of the DeDFRS architecture by incorporating the temporal context of the input signal.

C. Processing-In-Memory Using Memristor Crossbar Arrays

To further enhance power efficiency and boost overall system throughput, we implemented a one-layer perceptron classifier using 1T1R (one transistor one memristor) crossbar arrays in parallel with analog VMMs. The outputs from the DeDFRS layer, serving as inputs for the perceptron, are encoded by varying the amplitude of input voltages to the memristor arrays. The perceptron's weights are translated into the conductance of two memristors arranged as differential pairs, where voltages of identical amplitude but opposite polarities are applied. These weights undergo initial offline training and are subsequently programmed onto the on-chip memristors. The input voltages are applied to the rows of the memristor arrays at the same time while the output currents are collected at the columns as the results of the parallel VMMs which is shown in Equation (13)

$$I_j = \sum_{i=0}^{X^n} (G_{ij}^{-} + G_{ij}^{+}) V_i \tag{13}$$

where G_{ij}^{+} and G_{ij}^{-} is the conductance of the differential memristors, respectively, V_i is the input voltage of the i-th row, and I_j is the output current of the j-th column.

The transistor and the interconnection between the cells were fabricated in a commercial foundry using 2–µm technology. The memristor devices were integrated on top of the CMOS substrates at the university cleanroom. We cleaned the exposed metal vias on the CMOS substrates with argon plasma to remove the native oxide. Three different metal layers– 7-nm Ag, 3-nm Ti, and 50-nm Pd–were deposited sequentially by DC sputtering under an 8.8×10^{-7}–torr background vacuum, lifted off in acetone with a sonication process for 10s at room temperature, and then annealed at 300-°C for 40 minutes in N_2 atmosphere. The bottom electrode which consists of 20-nm Pt with a 2 to 2.5-nm Ti adhesion layer was then patterned and evaporated onto the metal pads, followed by liftoff in acetone. A 5-nm Ta_2O_5 blank layer was sputtered in 90-W

radio frequency power with 20 standard cubic centimeters per minute (sccm) Ar flow (background vacuum in the sputterer was better than 3.3×10^{-7} torr). The top electrode which is 20-nm Ta and 20-nm Pt was patterned by photolithography, deposited by DC sputtering, and then lifted off in acetone at room temperature [67]. The crossbar structure and the cross-section image of our memristor are demonstrated in Fig. 5.

IV. Measurement Results Analysis

We developed a prototype of the DeDRFS in GlobalFoundries (GF) standard 180nm CMOS technology to verify and demonstrate the functionality of each of the modules. The design occupies a total area of 0.0126mm^2 and consumes 10.8mW of power.

A. DeDFRS Performance Evaluation

In order to evaluate the prototype, we initially created our masking circuit from Fig. 3. To demonstrate the functionality of masking, a sine wave of 1MHz is applied to the input terminals of the masking circuit with a square wave representing a binary mask with an amplitude of 250mV at the mask terminals of Fig 3. Equation (7) can be simplified and modified to Equation (14) as

$$V_{out} = \alpha \cdot V_{in} \cdot V_{mask} \tag{14}$$

where α is the scaling factor to amplify the output signal from the circuit. The results from Fig. 6(a) demonstrate the amplified output imprinted with the mask signal that was applied indicative of the variability in this network. Fig. 6(b) indicates the output of the Mackey-Glass nonlinearity function, simulated for inputs in the range of 0 to 1.8V. Based on these simulation results, the function can spread out the input data nonlinearly before injecting it into the DeDFRS reservoir core. The system performance is improved by this novel Mackey-Glass nonlinear circuit.

The resulting signal from the nonlinearly transformed masked input is applied to the AES with its output demonstrated in Fig. 6(c). The input spikes generated from the two LIF neurons are spaced 0.5µs apart. With the combination of these two spike trains, the input signal injected into the neuron is portrayed as the spike train with two spikes per clock cycle. The delayed spike train with a factor of $\tau = 0.25\mu s$ can be observed from the plot in Fig. 6(c). The delay factor τ for each spike train in each neuron in the reservoir core can be calibrated for the DeDFRS by changing the value of the input current. The system's capacity for calibration enhances its ability to function effectively over a broad range, thereby leading to an improvement in overall system performance. The output signal from the delayed module is sent as spikes and we demonstrate the output from the spike to the analog signal converter in Fig. 6(d). Based on the spike arrival time, the decoder charges and discharges producing the analog output. The plot from Fig. 6(d) depicts the variation in the output signal based on the timing of the spikes. As evident from our plot, the timing between the spikes determines the analog output range of the decoder. This output can then be applied to the final layer of the reservoir for classification purposes.

TABLE I
Power and Area of the MERRC Architecture

	Mask	Mackey-Glass	ASE	Neuron	Decoder	Memristor Bitcell
Power (mW)	2.59	1.21×10^{-7}	0.262	2.023×10^{-8}	0.199	-
Area (μm)	22.44 ×9	22.44 ×6.53	89.6× 83.8	22.44× 23	22.44× 40.1	4×4

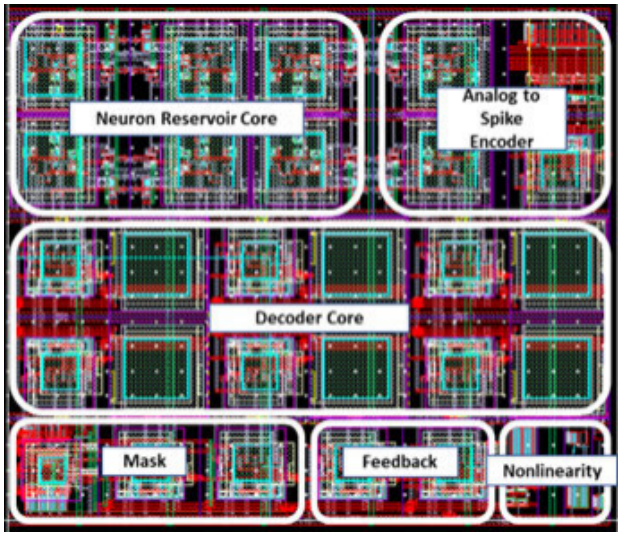


Fig. 7. Layout of the DeDFRS.

We evaluate the power and area of the design using post-layout simulations on GF-'s standard 180nm CMOS technology with a supply voltage of 1.8V as demonstrated in Table I. The major components on the DeDFRS consisting of the ASE, the neuron and the decoder consumes 0.5mW of power while the masking circuit consumes 2.59mW of power. Each neuron in this design consumes a significantly low power of merely 2.023×10^{-8}mW. As these neurons dictate the reservoir's size, specifically the size of the node, increasing the reservoir leads to reduced overall power consumption. Therefore, on the post-layout simulations with 6 neurons in the reservoir neuron core followed by 6 decoders in the decoder core as well as our control circuity blocks, the total power consumed by this design is 10.8mW.

With the use of proper layout techniques, planning and using multi-finger transistors, our post layout simulation showed insignificant drops in voltage. From the post-layout measurements, the area occupied by each module is shown in Table I and the complete layout consisting of the mask and the DeDFRS is demonstrated in Fig. 7. With each neuron and decoder occupying 22.4µm × 23µm and 22.4µm × 40.1µm respectively, the complete design encompasses an area of 0.0126mm^2 with 6 neurons and 6 decoders each. The fabricated memristor cell with the 1T1R structure has a junction area of 4µm × 4µm each. Our reconfigurable memristors in the crossbar allow us to utilize the size of the classification layer based on the input datasets.

B. MERRC Board Implementation

To substantiate the functionality and efficiency of our MERRC circuits and architecture, we experimented utilizing

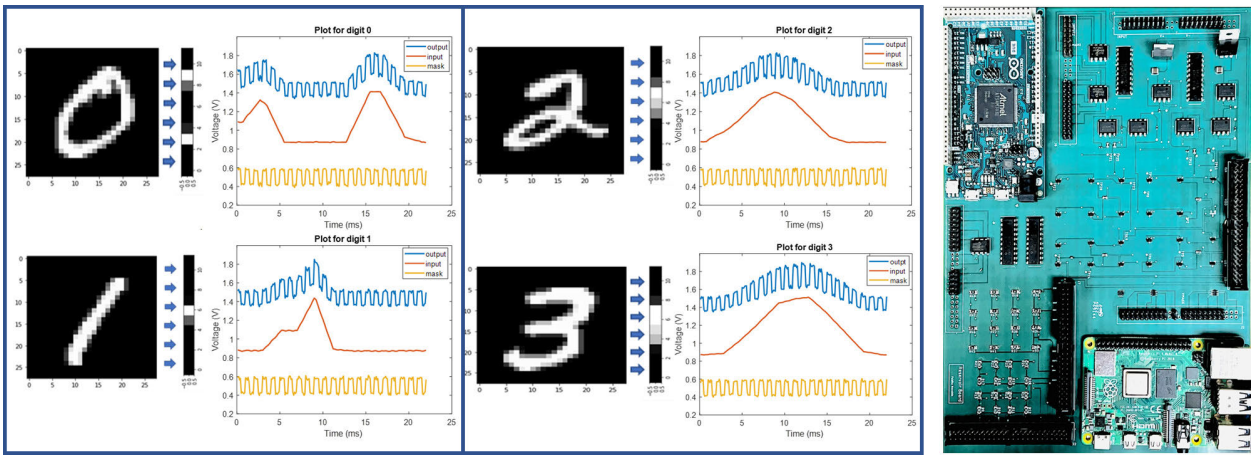


Fig. 8. Measurement data of the MERRC Board.

Fig. 9. The memristor crossbar array measurement setup.

the MNIST dataset. The experiment was facilitated using a hybrid setup comprising a Printed Circuit Board (PCB), an 8GB RAM Raspberry Pi 4 Computer Module B and a memristor crossbar. Leveraging the detailed circuitry conceptualized in Cadence Virtuoso, we transitioned the design onto a PCB board to realistically emulate the MERRC architecture. The design of the masking circuits was implemented on the PCB by using the Eagle software. The equations and functionality of the circuitry was translated into a layout that could be realized on the PCB. This involved adapting the circuitry's behavior into components that are feasible to construct using PCB fabrication techniques.

The masking circuit shown in Fig. 3 implemented on 180nm technology was replicated on the PCB board using the same circuit topology and transistors following the similar width to length ratio and scale. Similarly, the reservoir core that was initially designed using Cadence Virtuoso at the 180 nm CMOS transistor level was translated to the Raspberry Pi through the use of the same circuit equations and functional logic. This essentially means that the core's behavior was captured in equations that could be executed on the Raspberry Pi's computing platform. Since reservoir computing only trains the output layer, our reconfigurable memristor crossbar embodies the output layer, the only layer subjected to training in the architecture. These memristor crossbars offer remarkable flexibility, capable of being reprogrammed to accommodate varying values based on edge device training, a feature not commonly found in standard ASIC-based implementations.

Due to budget constraints and current limitations in utilizing GF fabrication services, we have opted to implement the MERRC architecture in a PCB format. Our objective is to uphold its functionality and validate the proof of concept within the confines of our resources at this time. We adopted PCB prototyping as a faster yet robust alternative for validating our technology, thus providing valuable insights into its potential real-world performance. While implementing a 180 nm CMOS transistor-level design on a PCB board involves translating complex transistor-level behavior since PCBs operate on a different scale and technology, the circuit components were approximated and adapted to match PCB fabrication possibilities. Our overall objective was to achieve successful emulation of the CMOS design's behavior, functions, and tasks within the constraints and capabilities of PCB fabrication

This approach effectively bridged the gap between theoretical and real-world simulations, allowing for a thorough examination of the MERRC architecture and masking circuits within a simulated chip environment. Therefore, combining the PCB board consisting of the masking circuitry, the DeDFRS in the Raspberry Pi and the memristor crossbars in the output layer, we have a complete MERRC architecture. and accomplished a successful trial run of our design using the MNIST dataset. This step not only demonstrated the viability and functionality of our design but also established a robust platform for further exploration and fine-tuning of the technology in a practical context.

Furthermore, in our experiments, we processed the images from the MNIST dataset using an "area interpolation" technique. This method, followed by an unrolling procedure, effectively condensed the original 28×28 pixel representation of each digit to a compact 1×12 format, facilitating smoother processing and analysis. The "interpolation by area" method is based on calculating the average pixel value of each area within the image and utilizing this value to determine the new pixel value in the resized image. This process facilitates the preservation of crucial features while ensuring that the resulting image maintains a high level of precision and accuracy.

In Fig. 8, we demonstrate the image pixel rescaling process, wherein each image is channeled through the DAC module on the MERRC board, integrating it into the architecture. Incorporating a binary mask in the form of a square wave, we infuse randomness and variability in the data and imprint this signal on the input wave consisting of the pixels of each digit from the MNIST dataset. The MERRC board successfully applied the mask onto the input signal, as shown in the plots from Fig. 8. We used the Raspberry Pi module to implement the Reservoir Core, as detailed in Algorithm 1. This allows us to efficiently process the MNIST dataset and validate the design of the MERRC architecture.

Algorithm 1 Reservoir Core

Initialize: Parameters of the Reservoir including the number of nodes N and weight matrix dimensions W
Initialize: Array for the virtual nodes N and reservoir history matrix R
Initialize: Nonlinearity function of Mackey-Glass mg with β and n
Data: Input of masked image data U[n × m]
for i = 1 to m do
  for j = 1 to N do
    delay = r[j − 1 : 1] ← add delay from previous
    a = mg(delay + U[i], β, n)
    Vn = U · (Vn + a) ← calculate the virtual node
  end for
  r = [Vn · Win + r_{prev} · W_{res}]
  y = W_{out} · r + Win · U + bias
  r_{prev} = r ← set the current term to the previous term
  h1 = relu(W_{h1} · y + bias)
  h2 = relu(W_{h2} · h1 + bias)
  out = relu(h2)
end for

C. Hybrid Memristor and CMOS Platform for MERRC

The experimental evaluation of the PIM perceptron is conducted with a 32×32 1T1R array. The 1T1R chip is landed on a probe station under a customized probe card, which is electrically connected to a set of customized PCBs with high-speed cables. The PCBs are designed with onboard analog-to-digital converters (ADCs), digital-to-analog converters (DACs), and trans-impedance amplifiers (TIA) to facilitate the reading and programming of memristor arrays. The PCBs communicate with a peripheral computer through a microcontroller unit (MCU) on the motherboard. In the programming phase, the set or reset voltages are applied to the rows or columns of the 1T1R array by DACs on the PCBs. Multiple set/reset voltages are used to tune the conductance of memristors to the target conductance range representing the weights of the output layer trained offline. In the inference phase, the input voltages (0-0.2V) are applied to the rows of the 1T1R array by DACs and output currents from the columns are converted to voltages by TIAs. The output voltages are then converted to digital signals by ADCs and sent to the peripheral computer for further processing using the MCU. The measurement setup

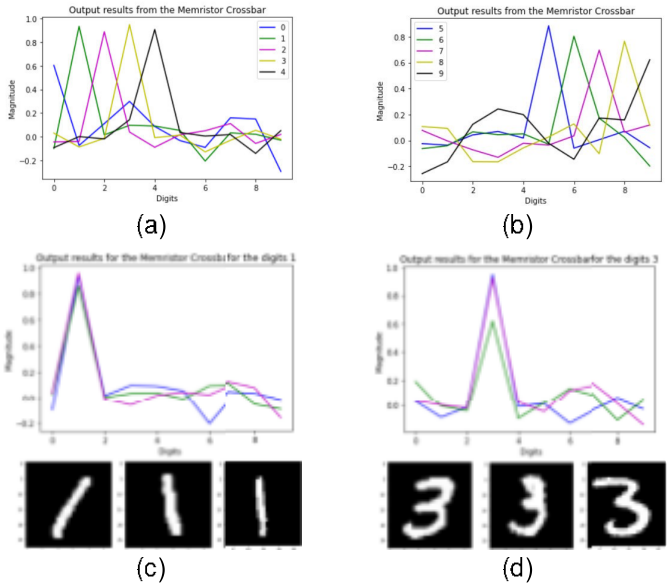


Fig. 10. Measurement results from the memristor crossbar. (a) Outputs from digits 0 to 4. (b) Outputs from digits 5 to 9. (c) Output from 3 noisy images of digit 1. (d) Output from 3 noisy images of digit 3.

of the memristor arrays with the MCU and PCB units similar to [65] is depicted in Fig. 9.

We use the weights from the final classification layer from the MERRC architecture to map onto the crossbar array. For 10 output digits, we program the 28 × 10 crossbar array with the weights of the classification layer. With the memristor crossbar emulating the fully connected layers of the final training portion of the MERRC architecture, the output currents from the crossbar are converted to voltages using ADCs in the PCBs and transferred to the peripheral computer via the MCU unit. We plot the output voltages after passing through the final activation layer of ReLU unit as depicted in Fig. 10.

The memristor output layer is not immune to the effects of noise and variation. Any perturbations introduced in the stored weights or conductance states of memristors can impact the accuracy of the final classification outcomes. Moreover, noise can accumulate during readout, potentially leading to errors in the classification process. When programming our memristors to the specific values, we experienced weight fluctuations and inaccuracies as well as deviations from the exact conductance value. However, despite experiencing these noise and variation effects, our architecture was able to successfully classify the MNIST images. Based on the simulation results it is evident that for each digit inputted, we obtain the final output indicated by the highest value for the MNIST digits from 0 to 9 shown in Fig. 10(a) and (b).

Furthermore, the system is also robust against noise where with the use of multiple noisy images of the same digit yields the correct output. We demonstrate the outputs for the digits 1 in Fig. 10(c) and 3 in Fig. 10(d) for three different noisy inputs each to indicate the robustness of the system. We further compare our MERRC architecture encompassing all its integrated components, including the masking circuit, reservoir core, and the memristor crossbar with the state-of-the-art memristor-based architectures shown in Table II. This comprehensive evaluation allows us to assess the overall

TABLE II
COMPARISON OF PERFORMANCES WITH MEMRISTOR-BASED ARCHITECTURES

	[68]	[69]	[70]	[71]	[72]	[73]	This work
Network	RNN-RC	MLP	RNN-LSTM	SNN	CNN	CNN	RNN-RC
Data Type	Time Sequence	Pattern	Pattern	Spiking Sequence	Image	Image	Time Sequence
Power	$0.05mW$	$2.18mW$	$312.40mW$	N/A	$7.44mW$	N/A	$10.8mW$
Accuracy	99.6%	96.5%	96.1%	89.1%	96.2%	92.79%	98%
Layer	3	4	3	4	5	4	3
Application	Temporal Signal Processing	Pattern Recognition	Pattern Recognition	MNIST	MNIST	MNIST	MNIST

TABLE III
COMPARISON OF MEMRISTOR-BASED DESIGNS ON THE CIFAR-10 DATASET

	[74]	[75]	[76]	[77]	This work
Year	2020	2021	2022	2021	2023
Accuracy	84%	88%	85%	87%	88%

power, performance, and efficiency of the architecture in practical scenarios, considering the collective contribution of all components. With a power consumption of 10.8mW, our design can be comparable to other state-of-the-art works. With the MNIST dataset using 3 layers, our MERRC architecture can achieve the highest accuracy of 98% among other designs.

D. Evaluation on the CIFAR-10 Dataset

The MERRC architecture has exhibited remarkable efficacy in processing and categorizing small-scale image datasets. However, it posed an interesting challenge when it came to simulation, especially for larger datasets like CIFAR-10. To bridge this gap between the architecture's design and practical simulations, we replicated our hardware parameters onto a software environment following Algorithm 1. In this software simulation, we utilized the computational power of a 12-GB NVIDIA Tesla K80 GPU with 13G RAM. While the MERRC architecture itself is rooted in hardware design, we harnessed the capabilities of this powerful GPU to simulate the behavior of the architecture and assess its performance with the CIFAR-10 dataset. Through a process of transla-tion, we mimicked the hardware parameters and behavior of the MERRC architecture on the software platform. This step allowed us to execute the architecture's algorithms and processing logic in a simulated manner, thus enabling us to assess its performance without physically fabricating the entire hardware setup. To introduce the necessary randomness and nonlinearity that are characteristic of real-world image datasets, we applied a randomized mask to the simulation. This mask, combined with the MG function, enhanced the architecture's responses to input data, making the simulation more representative of actual operational scenarios.

We extended our evaluations to the CIFAR-10 dataset to assess the architecture's capability to handle larger and more intricate image data. To accommodate this complexity, we expanded the reservoir's size to 1600 nodes. This adjustment, along with the randomized mask, contributed to achieving an impressive accuracy rate of 88% on the CIFAR-10 dataset. Comparing the MERRC architecture's performance with other memristor-based designs on the CIFAR-10 dataset, as showcased in Table III, highlighted its potential to achieve high accuracy rates. These results underscore the MERRC architecture's viability for efficiently classifying large-scale image datasets, offering a promising solution for real-world image classification challenges. The simulations conducted using the NVIDIA Tesla K80 GPU, combined with the architecture's design principles and parameters, validate its efficacy in handling larger datasets. The fusion of a reservoir with a memristor crossbar layer, alongside the ADAM optimizer, emerges as an efficient strategy to enhance the architecture's performance.

V. CONCLUSION

In this work, we introduce a memristor-enabled reconfigurable reservoir computing architecture, MERRC, particularly for edge devices. Our design consists of reconfigurable memristive synapses that can deploy a trained model onto the MERRC device and thus bring computations close to the source of data, eliminating power and latency issues. We introduce the first analog novel masking design that can add variability to the network and utilize the dimensionality in the reservoir. The DeDFRS developed can process spatiotemporal information and simplifies the training complexity by only training the output layer. Hardware simulations on the hybrid MERRC board combined with the fabricated memristor crossbar array provide the optimum accuracy of 98% with the MNIST dataset and consume 10.8mW of power and a small area of 0.0216mm^2. Simulations with a more complex image dataset of CIFAR-10 provide an accuracy of 88% for the MERRC architecture. Our findings suggest that the MERRC architecture could be an optimal choice for edge devices where memory and processing power are limited, yet efficient and accurate image recognition is crucial. Therefore, the MERRC architecture could significantly impact edge computing and real-time image processing applications where speed, accuracy, and efficiency are crucial.

REFERENCES

[1] A. I. Khan and S. Al-Habsi, "Machine learning in computer vision," Proc. Comput. Sci., vol. 167, pp. 1444–1451, Jan. 2020.
[2] J. Wäldchen and P. Mäder, "Machine learning for image based species identification," Methods Ecol. Evol., vol. 9, no. 11, pp. 2216–2225, Nov. 2018.
[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature, vol. 521, no. 7553, pp. 436–444, 2015.
[4] N. Rusk, "Deep learning," Nature Methods, vol. 13, no. 1, p. 35, 2016.
[5] M. Z. Alom et al., "A state-of-the-art survey on deep learning theory and architectures," Electronics, vol. 8, no. 3, p. 292, Mar. 2019.
[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," Commun. ACM, vol. 60, no. 6, pp. 84–90, May 2017.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, arXiv:1409.1556.

[8] C. Szegedy et al., "Going deeper with convolutions," in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR), Jun. 2015, pp. 1–9.

[9] K. Cao, Y. Liu, G. Meng, and Q. Sun, "An overview on edge computing research," IEEE Access, vol. 8, pp. 85714–85728, 2020.

[10] R. Machupalli, M. Hossain, and M. Mandal, "Review of ASIC accelerators for deep neural network," Microprocessors Microsyst., vol. 89, Mar. 2022, Art. no. 104441.

[11] S. Miryala et al., "Design and challenges of edge computing ASICs on front-end electronics," in Proc. 23rd Int. Symp. Quality Electron. Design (ISQED), Apr. 2022, pp. 19–27.

[12] J. Chen and X. Ran, "Deep learning with edge computing: A review," Proc. IEEE, vol. 107, no. 8, pp. 1655–1674, Aug. 2019.

[13] S. Deng, H. Zhao, W. Fang, J. Yin, S. Dustdar, and A. Y. Zomaya, "Edge intelligence: The confluence of edge computing and artificial intelligence," IEEE Internet Things J., vol. 7, no. 8, pp. 7457–7469, Aug. 2020.

[14] E. Garzón, A. Teman, M. Lanuzza, and L. Yavits, "AIDA: Associative in-memory deep learning accelerator," IEEE Micro, vol. 42, no. 6, pp. 67–75, Nov. 2022.

[15] E. Eleftheriou et al., "Deep learning acceleration based on in-memory computing," IBM J. Res. Develop., vol. 63, no. 6, pp. 7:1–7:16, Nov. 2019.

[16] O. Mutlu, S. Ghose, J. Gómez-Luna, and R. Ausavarungnirun, "A modern primer on processing in memory," in Emerging Computing: From Devices to Systems. Cham, Switzerland: Springer, 2022, pp. 171–243.

[17] P. Kumar, K. Zhu, X. Gao, S.-D. Wang, M. Lanza, and C. S. Thakur, "Hybrid architecture based on two-dimensional memristor crossbar array and CMOS integrated circuit for edge computing," NPJ 2D Mater. Appl., vol. 6, no. 1, p. 8, Jan. 2022.

[18] A. Canziani, A. Paszke, and E. Culurciello, "An analysis of deep neural network models for practical applications," 2016, arXiv:1605.07678.

[19] K. Nakajima and I. Fischer, Reservoir Computing. Cham, Switzerland: Springer, 2021.

[20] F. Liu, G. Tang, Y. Li, Z. Cai, X. Zhang, and T. Zhou, "A survey on edge computing systems and tools," Proc. IEEE, vol. 107, no. 8, pp. 1537–1562, Aug. 2019.

[21] Edge TPU. (2019). Google's Purpose-Built ASIC Designed to Run Inference at the Edge. [Online]. Available: https://cloud.google.com/edge-tpu/

[22] Z. Du et al., "ShiDianNao: Shifting vision processing closer to the sensor," in Proc. ACM/IEEE 42nd Annu. Int. Symp. Comput. Archit. (ISCA), Jun. 2015, pp. 92–104.

[23] X. Yang, S. Li, Q. Zheng, and Y. Chen, "Improving the robustness and efficiency of PIM-based architecture by SW/HW co-design," in Proc. 28th Asia South Pacific Design Autom. Conf. (ASP-DAC), Jan. 2023, pp. 618–623.

[24] X. Si et al., "A twin-8T SRAM computation-in-memory macro for multiple-bit CNN-based machine learning," in IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers, Feb. 2019, pp. 396–398.

[25] 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, Aug. 2019.

[26] H. Kim, T. Yoo, T. T. Kim, and B. Kim, "Colonnade: A reconfigurable SRAM-based digital bit-serial compute-in-memory macro for processing neural networks," IEEE J. Solid-State Circuits, vol. 56, no. 7, pp. 2221–2233, Jul. 2021.

[27] B. Sun et al., "MRAM co-designed processing-in-memory CNN accelerator for mobile and IoT applications," 2018, arXiv:1811.12179.

[28] P. Chi et al., "PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," ACM SIGARCH Comput. Archit. News, vol. 44, no. 3, pp. 27–39, 2016.

[29] Y. Long, T. Na, and S. Mukhopadhyay, "ReRAM-based processing-in-memory architecture for recurrent neural network acceleration," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 26, no. 12, pp. 2781–2794, Dec. 2018.

[30] N. K. Upadhyay, H. Jiang, Z. Wang, S. Asapu, Q. Xia, and J. J. Yang, "Emerging memory devices for neuromorphic computing," Adv. Mater. Technol., vol. 4, no. 4, Apr. 2019, Art. no. 1800589.

[31] H. Zhao et al., "Memristor-based signal processing for edge computing," Tsinghua Sci. Technol., vol. 27, no. 3, pp. 455–471, Jun. 2022.

[32] J. Moon et al., "Temporal data classification and forecasting using a memristor-based reservoir computing system," Nature Electron., vol. 2, no. 10, pp. 480–487, Oct. 2019.

[33] C. Li et al., "Long short-term memory networks in memristor crossbar arrays," Nature Mach. Intell., vol. 1, no. 1, pp. 49–57, Jan. 2019.

[34] Z. Wang et al., "Reinforcement learning with analogue memristor arrays," Nature Electron., vol. 2, no. 3, pp. 115–124, Mar. 2019.

[35] R. Pascanu, C. Gulcehre, K. Cho, and Y. Bengio, "How to construct deep recurrent neural networks," 2013, arXiv:1312.6026.

[36] Y. Bengio, N. Boulanger-Lewandowski, and R. Pascanu, "Advances in optimizing recurrent networks," in Proc. IEEE Int. Conf. Acoust., Speech Signal Process., May 2013, pp. 8624–8628.

[37] C. R. Vogel, Computational Methods for Inverse Problems. Philadelphia, PA, USA: SIAM, 2002.

[38] N. Soures and D. Kudithipudi, "Deep liquid state machines with neural plasticity for video activity recognition," Frontiers Neurosci., vol. 13, p. 686, Jul. 2019.

[39] K. Bai and Y. Yi, "DFR: An energy-efficient analog delay feed-back reservoir computing system for brain-inspired computing," ACM J. Emerg. Technol. Comput. Syst., vol. 14, no. 4, pp. 1–22, Oct. 2018.

[40] L. Larger et al., "Photonic information processing beyond Turing: An optoelectronic implementation of reservoir computing," Opt. Exp., vol. 20, no. 3, pp. 3241–3249, Jan. 2012.

[41] R. Martinenghi, S. Rybalko, M. Jacquot, Y. K. Chembo, and L. Larger, "Photonic nonlinear transient computing with multiple-delay wavelength dynamics," Phys. Rev. Lett., vol. 108, no. 24, Jun. 2012, Art. no. 244101.

[42] M. C. Soriano et al., "Delay-based reservoir computing: Noise effects in a combined analog and digital implementation," IEEE Trans. Neural Netw. Learn. Syst., vol. 26, no. 2, pp. 388–393, Feb. 2015.

[43] L. Appeltant et al., "Information processing using a single dynamical node as complex system," Nature Commun., vol. 2, no. 1, p. 468, Sep. 2011.

[44] L. Larger, A. Baylón-Fuentes, R. Martinenghi, V. S. Udaltsov, Y. K. Chembo, and M. Jacquot, "High-speed photonic reservoir computing using a time-delay-based architecture: Million words per second classification," Phys. Rev. X, vol. 7, no. 1, Feb. 2017, Art. no. 011015.

[45] M. C. Soriano, J. García-Ojalvo, C. R. Mirasso, and I. Fischer, "Complex photonics: Dynamics and applications of delay-coupled semiconductors lasers," Rev. Mod. Phys., vol. 85, no. 1, pp. 421–470, Mar. 2013.

[46] F. Duport, A. Smerieri, A. Akrout, M. Haelterman, and S. Massar, "Fully analogue photonic reservoir computer," Sci. Rep., vol. 6, no. 1, p. 22381, Mar. 2016.

[47] J. Nakayama, K. Kanno, and A. Uchida, "Laser dynamical reservoir computing with consistency: An approach of a chaos mask signal," Opt. Exp., vol. 24, no. 8, pp. 8679–8692, Apr. 2016.

[48] Q. Vinckier et al., "High-performance photonic reservoir computer based on a coherently driven passive cavity," Optica, vol. 2, pp. 438–446, May 2015.

[49] Y. Kuriki, J. Nakayama, K. Takano, and A. Uchida, "Impact of input mask signals on delay-based photonic reservoir computing with semiconductor lasers," Opt. Exp., vol. 26, no. 5, pp. 5777–5788, Mar. 2018.

[50] W. Liang et al., "High spectral purity Kerr frequency comb radio frequency photonic oscillator," Nature Commun., vol. 6, no. 1, p. 7957, Aug. 2015.

[51] K. E. Callan, L. Illing, Z. Gao, D. J. Gauthier, and E. Schöll, "Broadband chaos generated by an optoelectronic oscillator," Phys. Rev. Lett., vol. 104, no. 11, Mar. 2010, Art. no. 113901.

[52] L. Larger and J. M. Dudley, "Optoelectronic chaos," Nature, vol. 465, no. 7294, pp. 41–42, May 2010.

[53] M. C. Soriano et al., "Optoelectronic reservoir computing: Tackling noise-induced performance degradation," Opt. Exp., vol. 21, no. 1, pp. 12–20, Jan. 2013.

[54] L. Appeltant, G. Van der Sande, J. Danckaert, and I. Fischer, "Constructing optimized binary masks for reservoir computing with delay systems," Sci. Rep., vol. 4, no. 1, p. 3629, Jan. 2014.

[55] L. Berezansky and E. Braverman, "A note on stability of Mackey–Glass equations with two delays," J. Math. Anal. Appl., vol. 450, no. 2, pp. 1208–1228, Jun. 2017.

[56] P. Amil, C. Cabeza, and A. C. Marti, "Exact discrete-time implementation of the Mackey–Glass delayed model," IEEE Trans. Circuits Syst. II, Exp. Briefs, vol. 62, no. 7, pp. 681–685, Jul. 2015.

[57] L. Glass and M. Mackey, "Mackey-glass equation," Scholarpedia, vol. 5, no. 3, p. 6908, 2010.

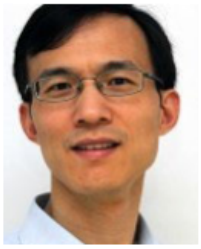
[58] M. Tateno and A. Uchida, "Nonlinear dynamics and chaos synchronization in Mackey-Glass electronic circuits with multiple time-delayed feedback," Nonlinear Theory Appl., vol. 3, no. 2, pp. 155–164, 2012.

[59] L. Junges and J. A. C. Gallas, "Intricate routes to chaos in the Mackey–Glass delayed feedback system," Phys. Lett. A, vol. 376, nos. 30–31, pp. 2109–2116, Jun. 2012.
[60] F. Nowshin, L. Liu, and Y. Yi, "Energy efficient and adaptive analog IC design for delay-based reservoir computing," in Proc. IEEE 63rd Int. Midwest Symp. Circuits Syst. (MWSCAS), Aug. 2020, pp. 592–595.
[61] W. Maass, "Networks of spiking neurons: The third generation of neural network models," Neural Netw., vol. 10, no. 9, pp. 1659–1671, Dec. 1997.
[62] S. Ghosh-Dastidar and H. Adeli, "Spiking neural networks," Int. J. Neural Syst., vol. 19, no. 4, pp. 295–308, Aug. 2009.
[63] A. L. Hodgkin and A. F. Huxley, "A quantitative description of membrane current and its application to conduction and excitation in nerve," J. Physiol., vol. 117, no. 4, pp. 500–544, Aug. 1952.
[64] E. M. Izhikevich, "Simple model of spiking neurons," IEEE Trans. Neural Netw., vol. 14, no. 6, pp. 1569–1572, Nov. 2003.
[65] R. A. Vazquez and A. Cachón, "Integrate and fire neurons and their application in pattern recognition," in Proc. 7th Int. Conf. Electr. Eng. Comput. Sci. Autom. Control, Sep. 2010, pp. 424–428.
[66] D. I. Standage and T. P. Trappenberg, "Differences in the subthreshold dynamics of leaky integrate-and-fire and Hodgkin–Huxley neuron models," in Proc. IEEE Int. Joint Conf. Neural Netw., 2005, pp. 396–399.
[67] F. Kiani, J. Yin, Z. Wang, J. J. Yang, and Q. Xia, "A fully hardware-based memristive multilayer neural network," Sci. Adv., vol. 7, no. 48, Nov. 2021, Art. no. eabj4801.
[68] Y. Zhong, J. Tang, X. Li, B. Gao, H. Qian, and H. Wu, "Dynamic memristor-based reservoir computing for high-efficiency temporal signal processing," Nature Commun., vol. 12, no. 1, p. 408, Jan. 2021.
[69] X. Liu and Z. Zeng, "Memristor crossbar architectures for implementing deep neural networks," Complex Intell. Syst., vol. 2022, pp. 1–16, Apr. 2022.
[70] K. Smagulova, O. Krestinskaya, and A. P. James, "A memristor-based long short term memory circuit," Anal. Integr. Circuits Signal Process., vol. 95, no. 3, pp. 467–472, Jun. 2018.
[71] G. Kim et al., "Self-clocking fast and variation tolerant true random number generator based on a stochastic Mott memristor," Nature Commun., vol. 12, no. 1, p. 2906, May 2021.
[72] P. Yao et al., "Fully hardware-implemented memristor convolutional neural network," Nature, vol. 577, no. 7792, pp. 641–646, Jan. 2020.
[73] J. Chen et al., "High-precision symmetric weight update of memristor by gate voltage ramping method for convolutional neural network accelerator," IEEE Electron Device Lett., vol. 41, no. 3, pp. 353–356, Mar. 2020.
[74] H. Ran et al., "Memristor-based edge computing of blaze block for image recognition," IEEE Trans. Neural Netw. Learn. Syst., vol. 33, no. 5, pp. 2121–2131, May 2022.
[75] T. V. Nguyen, J. An, and K.-S. Min, "Memristor-CMOS hybrid neuron circuit with nonideal-effect correction related to parasitic resistance for binary-memristor-crossbar neural networks," Micromachines, vol. 12, no. 7, p. 791, Jul. 2021.
[76] R. Ki and R. Sukuma, "Memristor based object detection using neu-ral network," High-Confidence Comput., vol. 2, no. 4, Dec. 2022, Art. no. 100085.
[77] X. F. Lu et al., "Exploring low power and ultrafast memristor on p-type van der Waals SnS," Nano Lett., vol. 21, no. 20, pp. 8800–8807, Oct. 2021.

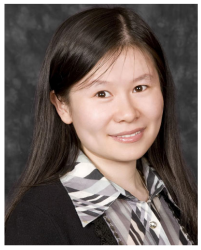
Fabiha Nowshin received the B.S. and M.S. degrees in electrical engineering from Virginia Tech, Blacksburg, VA, USA, in 2019 and 2021, respectively, where she is currently pursuing the Ph.D. degree with the Bradley Department of Electrical and Computer Engineering. Her research interests include analog-mixed signal circuit design, neuromorphic computing, emerging memory technologies, and computing-in-memory architectures.

Yi Huang (Graduate Student Member, IEEE) received the B.Eng. degree in system engineering and the M.Sc. degree in control science and engineering from the School of Artificial Intelligence and Automation, Huazhong University of Science and Technology, Wuhan, China, in 2016 and 2019, respectively. He is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Massachusetts Amherst, MA, USA. His research interests include mixed-signal circuit design and neuromorphic computing based on memristors.

Md. Rubel Sarkar received the B.Sc. degree in electrical and electronic engineering from the Ahsanullah University of Science and Technology, Dhaka, Bangladesh, in 2017. He is currently pursuing the Ph.D. degree with the Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, USA. He joined Virginia Tech in August 2022. His ongoing research interests include neuromorphic computing, hyperdimensional computing, in-memory computing, and low power VLSI systems.



Qiangfei Xia (Fellow, IEEE) received the Ph.D. degree in electrical engineering from Princeton University in 2007. He is currently a Professor in electrical and computer engineering with the University of Massachusetts (UMass) Amherst, MA, USA, and the Head of the Nanodevices and Integrated Systems Laboratory. Prior to UMass, he spent three years at Hewlett-Packard Laboratories. His research interests include beyond-CMOS devices, integrated systems, and enabling technologies with applications in machine intelligence, reconfigurable RF systems, and hardware security.



Yang Yi (Senior Member, IEEE) is currently an Associate Professor with the Bradley Department of Electrical Engineering and Computer engineering, Virginia Tech, Blacksburg, VA, USA. Her research interests include VLSI circuits and systems, computer-aided design (CAD), neuromorphic architecture for brain-inspired computing systems, and low-power circuits design with advanced nano-technologies for high-speed wireless systems.