NeuroSim: A Circuit-Level Macro Model for Benchmarking Neuro-Inspired Architectures in Online Learning

Pai-Yu Chen, *Student Member, IEEE*, Xiaochen Peng, and Shimeng Yu, *Member, IEEE*

***Abstract*—Neuro-inspired architectures based on synaptic memory arrays have been proposed for on-chip acceleration of weighted sum and weight update in machine/deep learning algorithms. In this paper, we developed NeuroSim, a circuit-level macro model that estimates the area, latency, dynamic energy, and leakage power to facilitate the design space exploration of neuro-inspired architectures with mainstream and emerging device technologies. NeuroSim provides flexible interface and a wide variety of design options at the circuit and device level. Therefore, NeuroSim can be used by neural networks (NNs) as a supporting tool to provide circuit-level performance evaluation. With NeuroSim, an integrated framework can be built with hierarchical organization from the device level (synaptic device properties) to the circuit level (array architectures) and then to the algorithm level (NN topology), enabling instruction-accurate evaluation on the learning accuracy as well as the circuit-level performance metrics at the run-time of online learning. Using multilayer perceptron as a case-study algorithm, we investigated the impact of the "analog" emerging nonvolatile memory (eNVM)'s "nonideal" device properties and benchmarked the tradeoffs between SRAM, digital, and analog eNVM-based architectures for online learning and offline classification.**

***Index Terms*—Emerging nonvolatile memory (eNVM), machine learning, neural network (NN), neuromorphic computing, offline classification, online learning, synaptic devices.**

I. Introduction

NEURO-INSPIRED computing has attracted a lot of interest as the traditional Boolean computing based on CMOS technology is reaching its physical limits [1]. Recent advances in neuro-inspired learning algorithms have achieved tremendous success in speech and image recognition, implemented with conventional CPUs/GPUs and/or FPGAs that are based on the sequential von Neumann architecture, which involves separation of the computing unit and memory between a data bus path. Due to the requirement of high bandwidth and power consumption for data communication via this data bus, traditional von Neumann architecture is inadequate for fast training and/or classification, such as real-time image and speech training and/or recognition. This challenge is recognized as the "von Neumann bottleneck" that degrades the overall efficiency and performance of the system [2]. In contrast, a biological brain (e.g., mammalian brain) enables parallel processing of a massive amount of information in a small area with high efficiency and low power consumption. Therefore, the ultimate goal of neuro-inspired computing is to develop artificial neural network (NN) that emulates highly efficient processing of biological information to bridge this efficiency gap between the network and real brain. The most straightforward approach is to simulate NN in software by using a von Neumann-based computer. However, this approach is limited due to significantly higher power consumption to mimic a biological brain with $\sim 10^{13}$ synapses, which can only be feasible with a very powerful supercomputer [3]. Another approach is to build artificial NN (ANN) on digital hardware, which is an ASIC implementation of the software NN. Several custom CMOS ASIC neuromorphic hardware accelerators have been developed in the recent years, such as BrainScaleS [4], Neurogrid [5], TrueNorth [6], Eyeriss [7], and DNPU [8]. These accelerators could operate more efficiently in terms of speed and power consumption than the software-based one. However, it is impractical to implement an ANN on-chip with similar complexity as a biological brain, because the current CMOS technology is not adequate to provide sufficient on-chip memory resources. Typically, the weight information of a single synapse is stored using 6- or 8-transistor SRAM cells, which is not area-efficient (with cell size 100–200 F^2, F is the lithography feature size). Therefore, it is attractive to explore more compact synaptic devices using emerging nonvolatile memory (eNVM) devices (with cell size 4–12 F^2).

In general, eNVM devices could form the "digital" synapses if only the ON and OFF binary states are exploited and multiple binary cells could be grouped to represent multilevel weights in a binary format. On the other hand, eNVM could also form the "analog" synapses by exploiting the multilevel conductance states [9]. However, these analog eNVM devices may not be as reliable as traditional CMOS digital circuits or digital eNVM devices, potentially suffering from nonideal device properties, such as limited precision, finite conductance ON/OFF ratio, weight update nonlinearity, variation and noise, etc. Prior works [10]–[12] have studied the impact of several nonideal eNVM synaptic device properties on the learning accuracy by incorporating the device behavioral model directly to the algorithm's code, therefore they could not address the impact on the circuit-level performance (e.g., area, latency, dynamic energy, and leakage power). On the other hand, the

Manuscript received March 10, 2017; revised July 13, 2017 and September 24, 2017; accepted December 20, 2017. Date of publication January 4, 2018; date of current version November 20, 2018. This work was supported by NSF under Grant NSF-CCF-1552687 and Grant NSF-CCF-1740225. This paper was recommended by Associate Editor D. Chen. *(Corresponding author: Pai-Yu Chen.)*

The authors are with the School of Electrical, Computer and Energy Engineering, Arizona State University, Tempe, AZ 85287 USA (e-mail: pchen72@asu.edu; shimengy@asu.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCAD.2018.2789723

reported architectural simulator platforms (e.g., PRIME [13], Harmonica [14], and ISSAC [15]) have demonstrated powerful capability and flexibility at the system-level design, but they have limited considerations at the aforementioned nonideal device properties (they only considered the weight precision and/or variation). MNSIM [16] is a circuit-level macro model of neuro-inspired architecture, but the accuracy in this model is the output error of weighted sum (matrix-vector multiplication), which is just one step of the algorithms thus it lacks the run-time learning accuracy of the entire algorithms. In such context, it is crucial to develop a circuit-level macro model that can be integrated with the learning algorithm (or NN) to form a simulation platform that is hierarchically organized from the device level, circuit level up to the algorithm level, where each level covers a wide variety of design options.

In this paper, we developed NeuroSim for the neuro-inspired architecture design at the circuit level. NeuroSim can support various mainstream and emerging memory technologies, and NeuroSim can also support various machine learning NNs to form a complete simulation framework that is instruction-accurate. Compared to a full SPICE simulation, NeuroSim leverages the circuit/device model parameters on a small but realistic neuro-inspired architecture, enabling fast early stage design exploration. The inputs to the simulator include memory types, nonideal device parameters, transistor technology nodes, network topology and array size, training dataset and traces, etc. The outputs of the simulator include the circuit-level performance metrics, such as area, latency, dynamic energy and leakage power consumption and algorithm-level learning accuracy in run-time. As a case study, NeuroSim is used to support a 2-layer multilayer perceptron (MLP) NN to benchmark different neuro-inspired architecture design options in the online learning and offline classification. The main contributions of this paper can be summarized as follows.

1) Developed NeuroSim with a wide variety of flexible design options to estimate the circuit-level performance of neuro-inspired architectures to facilitate the design space exploration.
2) Calibrated the circuit-level performance metrics based on cell characteristics at logic gate level, and performed validation of these metrics compared to layout and SPICE simulation.
3) Demonstrated the use of NeuroSim in the MLP simulator to be capable of evaluating both the learning accuracy and circuit-level performance metrics at the run-time of the algorithm.
4) Investigated the design tradeoffs with SRAM, digital, and analog eNVM-based neuro-inspired architectures considering the nonideal device properties in the online learning and offline classification.

In the discussions, we will illustrate the usage of NeuroSim with different scenarios, as well as pointing out the limitations of NeuroSim. At the current stage, the target users of NeuroSim are device engineers, as it is most suitable for them to quickly benchmark system-level performance with synaptic device properties in basic NN algorithms. The source code of NeuroSim v1.0 is publicly available at [17]. In this version, only analog eNVM architectures are included.

The rest of this paper is organized as follows. Section II introduces the background of synaptic devices and their array architectures. Section III describes the NeuroSim architecture and its customization, usage, and limitations. Section IV discusses the performance modeling and validation of NeuroSim.

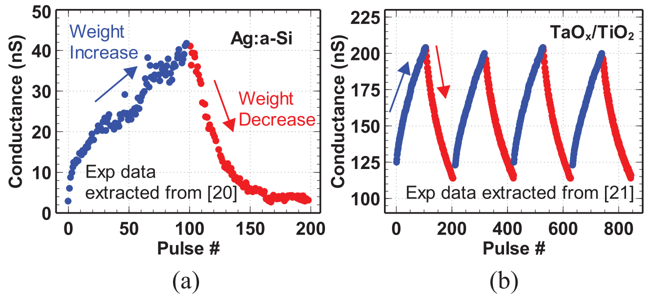

Fig. 1. Reported measured experimental data of weight update in (a) Ag:a-Si and (b) TaO_x/TiO_2-based synaptic devices.

Section V demonstrates a 2-layer MLP NN with the support of NeuroSim to benchmark SRAM, digital, and analog eNVM architectures in online learning and offline classification. Section VI summarizes this paper.

II. Background on Synaptic Devices and Array Architectures

A. Synaptic Devices

In the neuromorphic system, a synapse serves as a communication path for one neuron to pass the signal to another neuron, and its strength (e.g., weight) can be gradually modified during the learning. For a complex neuromorphic system, the number of synapses is far more than the number of neurons. To prevent a tremendous hardware cost with CMOS-based synapses, the current progress in nanotechnology is paving the way toward implementation of synapses using low-cost and ultra-high density memory array [18]. In fact, due to its maturity, the floating-gate memory technology has been successfully implemented on a single chip as synapses for the neuromorphic computing [19]. To achieve even higher integration density, faster speed, and lower programming voltage, compact synaptic devices based on eNVM are proposed for the neuromorphic systems, including resistive random access memory [20]–[25] and phase change memory [11], [26], [27], etc. These eNVM devices are two-terminal, and they can represent the weight with their analog multilevel conductance states. In this paper, we simply refer to these devices as "synaptic devices". Modification of the weight (weight update) thus relies on the transition between different conductance states in the synaptic devices, which is typically triggered by electrical inputs. The detailed physical mechanism of this conductance transition can be different for different types of synaptic devices. Generally, the conductance of synaptic devices can be increased and decreased with positive and negative programming voltage pulses, which are referred to as weight increase and decrease, respectively. An ideal synaptic device behavior assumes a linear update of the weight with identical programming voltage pulses. As shown in Fig. 1, however, the realistic devices reported in literature do not follow such ideal trajectory, exhibiting "nonideal" properties, such as nonlinear and noisy weight increase/decrease, limited precision, and finite ON/OFF ratio. For example, Ag:a-Si devices [20] show a nonlinear and noisy weight increase/decrease; though TaO_x/TiO_2 devices [21] exhibit a more linear and smooth weight increase/decrease, the ON/OFF ratio is very limited (~2). Such nonideal behaviors commonly exist in today's synaptic devices [20]–[23], possibly due to the inherent drift and diffusion dynamics of the ions/vacancies in these materials.

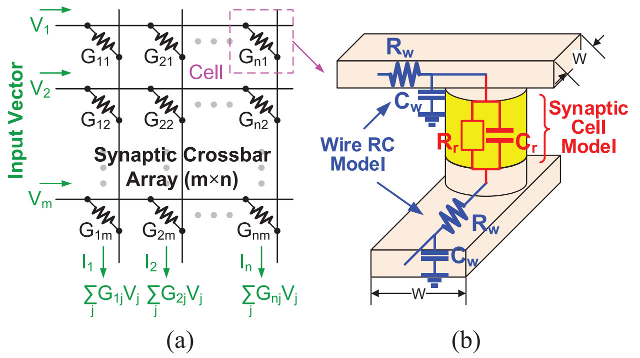
Input Vector
V_1
V_2
V_m
G_{11} G_{21} G_{n1}
G_{12} G_{22} G_{n2}
G_{1m} G_{2m} G_{nm}
Cell
Synaptic Crossbar Array (m×n)
I_1 I_2 I_n
$\sum_j G_{1j}V_j$ $\sum_j G_{2j}V_j$ $\sum_j G_{nj}V_j$
(a)
R_w
C_w
W
Wire RC Model
R_r
C_r
Synaptic Cell Model
(b)

Fig. 2. (a) Weighted sum operation in an eNVM-based synaptic crossbar array structure. (b) Equivalent RC model of a synaptic device.

B. Crossbar Array Architecture

A set of synapses that fully connects between two layers of neurons can be viewed as a weight matrix. The most compact and simplest array structure for synaptic devices to form this weight matrix is the crossbar array structure, where each synaptic device is located at each cross point. The crossbar array structure can achieve a high integration density of 4 F^2/cell. As shown in Fig. 2(a), if the input vector is encoded by read voltage signals, the weighted sum operation (matrix-vector multiplication) can be performed in a parallel fashion with synaptic crossbar array [28], [29]. The weighted sum result in terms of the output currents are then obtained at the end of each column. Ideally, it can be expressed in a matrix form

$$\begin{pmatrix} I_1 \\ I_2 \\ \vdots \\ I_n \end{pmatrix} = \begin{pmatrix} G_{11} & G_{12} & \cdots & G_{1m} \\ G_{21} & G_{22} & \cdots & G_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ G_{n1} & G_{n2} & \cdots & G_{nm} \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ \vdots \\ V_m \end{pmatrix} \tag{1}$$

where each G element in the weight matrix is the conductance of the synaptic devices. Fig. 2(b) shows the equivalent RC model of a single cell in the crossbar array structure, which can be duplicated to form the whole array. The wire parasitics (R_w and C_w) not only bring extra latency and energy consumption in the array, but also causes IR drop (reduction of access voltage) along the weighted sum path. Aggressive downscaling of the wire width (W) will make the IR drop more severe. Hence, the weighted sum current in (1) may not be accurately obtained. Fortunately, the sneak path problem [30] of the unselected cells in the array for conventional memory application does not exist in the weighted sum operation, if all the cells are participating in the computation. For weight update operation, though a fully parallel write scheme has been proposed [21], programming all the cells in the array may consume too much peak power that the peripheral circuits can provide. Therefore, row-by-row write scheme has to be used. As there is no isolation between cells, it is necessary to apply some voltage (smaller than the programming voltage) at all the unselected rows and columns to prevent the write disturbance on unselected cells during weight update [31]. In this scheme, a lot of energy is consumed to charge up all unselected rows and columns for every single operation. Therefore, the crossbar array architecture may not be energy-efficient in weight update.

C. Pseudo-Crossbar Array Architecture

The write disturbance problem in crossbar array architecture is a concern in both traditional memory and neuromorphic applications. In principle, a two-terminal selector device should be added to prevent the interference between cells [32], however, the development of selector device is not mature at this stage. A common design solution is to add a cell selection transistor in series with the eNVM device, forming the one-transistor one-resistor (1T1R) array architecture, as shown in Fig. 3(a). The word line (WL) controls the gate of the transistor, which can be viewed as a switch for the cell. The source line (SL) connects to the source of the transistor. The eNVM cell's top electrode connects to the bit line (BL), while its bottom electrode connects to the drain of the transistor through a contact via. In such case, the cell area of 1T1R array is then determined by the transistor size, which is typically >6 F^2 depending on the maximum current required to be delivered into the eNVM cell. Larger current needs larger transistor gate width/length (W/L). However, conventional 1T1R array is not able to perform the weighted sum operation that follows (1). In this case, we have to modify the conventional 1T1R array by rotating the BLs by 90°, which is known as the pseudo-crossbar array architecture [33], as shown in Fig. 3(b). In weighted sum operation, all the transistors will be transparent when all WLs are turned on. Thus, the input vector voltages are provided to the BLs, and the weighted sum currents are read out through SLs in parallel. It should be noted that the IR drop problem still exists in the pseudo-crossbar array, and the wire RC model in Fig. 2(b) can also be applied here to study the IR drop problem.

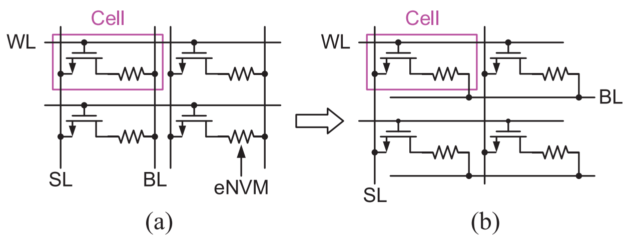

Fig. 3. Transformation from (a) conventional 1T1R array to (b) pseudo-crossbar array by 90° rotation of BL to enable weighted sum operation.

The voltage bias schemes for weight update are shown in Fig. 4. As the weight increase and decrease need different programming voltage polarities, the weight update process requires two steps with different voltage bias schemes. In weight update, the selected cells will be on the same row, and programming pulses or biases (if no update) are provided from the SL, allowing the selected cells to be tuned on in parallel. To perform weight update for the entire array, a row-by-row operation is still necessary. Ideally, the entire row should be selected at a time to ensure the maximum parallelism. However, only part of a row may be selected at a time if the write circuitry cannot afford such a large write current. With only the selected WL activated, the unselected cells at all other rows can be free from the write disturbance, meanwhile achieving significant reduction on the energy consumption in biasing these unselected rows. In principle, similar row-by-row write scheme could be applied to the crossbar array with selectors.

III. NeuroSim Architecture

A. Overview

NeuroSim is a circuit-level macro model developed in C++ that can be used to estimate the area, latency, dynamic energy, and leakage power of neuromorphic hardware accelerators with SRAM and eNVM-based architectures to facilitate the

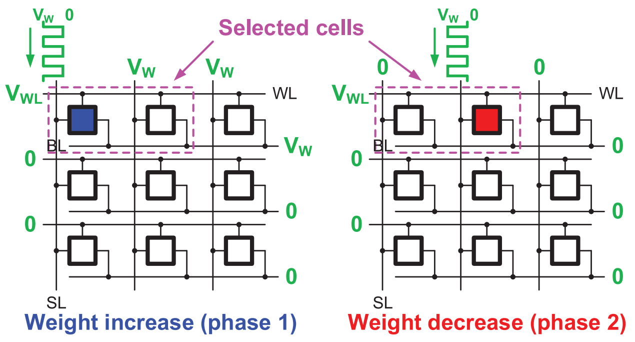

Fig. 4. Voltage bias schemes in the write operation of pseudo-crossbar array. Two separate phases for weight increase and decrease are required. In this example, the left cell of the selected cells will be updated in phase 1, while the right one will be updated in phase 2.

design space exploration. The framework of NeuroSim follows the principles of CACTI [34] for SRAM cache and NVSim [35] for NVM. These simulators focus on the design for traditional memory application, and do not support the memory design for neuromorphic computing. In contrast, NeuroSim is dedicated to support neuro-inspired architectures. The hierarchy of the simulator consists of different levels of abstraction from the memory cell parameters and transistor technology parameters, to the gate-level subcircuit modules and then to the array architecture including the peripheral circuits. Fig. 5(a) shows an overview of the high-level architecture with neuromorphic hardware accelerator to implement NNs. NNs generally require multiple (or deep) layers for better learning performance, where each layer contains the synaptic core and neuron periphery. A synaptic core is specifically designed for weighted sum and weight update. It takes the digital input vector and gives out the weighted sum result in the digital format. Thus, the digital communication is used between synaptic cores while the analog computation will just be done within the core only. The synaptic core further consists of the synaptic array and array periphery. The synaptic array [such as Figs. 2(a) or 3] is the core unit of weighted sum computation and the array periphery helps transform the results to be the digital format if necessary. Currently, NeuroSim supports SRAM, digital, and analog eNVM-based synaptic cores, as shown in Fig. 5(b)–(d). On the other hand, the neuron periphery is responsible for nonlinear activation function and communication from one synaptic core to another. Currently, NeuroSim can implement nonlinear activation function using an SRAM/eNVM array-based look-up table, while it also supports the low-precision activation function, such as thresholding with step function. As the circuit implementation of neuron periphery is more flexible and can vary between different NNs, we will only show an example one of low-precision activation function in the case study of Section V in this paper.

B. Synaptic Cores

In this section, we introduce the detailed architecture of three types of synaptic cores: 1) SRAM; 2) digital; and 3) analog eNVM. In NeuroSim, the synaptic cores are considered to be at one hierarchy level higher than the subcircuit modules, as they consist of both memory array and peripheral circuits that are closely jointed to form a standalone weighted sum computation unit. Important parameters at this level include synaptic array types and sizes, operating modes of peripheral circuits, and the number of synapses that can be accessed in parallel during weighted sum and weight update, etc.

1) SRAM Synaptic Core: The circuit block diagram of SRAM synaptic core is shown in Fig. 5(b). As SRAM cells can only store binary bits, we group multiple SRAM cells along the row as one synapse to represent a higher weight precision. The weighted sum and weight update operation in the SRAM-based synaptic core are essentially row-by-row-based, which is similar to the read and write operation in a conventional SRAM memory.

In the weighted sum operation, the input vector is encoded using multiple clock cycles to represent its precision. For each row, an input vector bit of 1 means the row will be selected for read, otherwise the row will be skipped. To select a row, the WL is activated through the WL decoder. To access all the cells on the selected row, the BLs are precharged by the precharger and the write driver in weighted sum and weight update, respectively. After the memory data are read by the sense amplifier (S/A), the adder and register are used to accumulate the partial weighted sum in a row-by-row fashion. To make sure the overflow will not occur during the accumulation, the adder and register need to have a longer bit-width than the weight precision of a synapse. The adder and shift register pair at the bottom performs shift and add of the weighted sum result at each input vector bit cycle to get the final weighted sum. The bit-width of the adder and shift register needs to be further extended depending on the precision of input vector. If the values in the input vector are only 1 bit, then the adder and shift register pair is not required. For the write operation, new weights will be provided from the input of the write driver. All the cells on the same row can be updated at the same time, thus the weight update operation is also row-by-row-based.

2) Digital eNVM Synaptic Core: The circuit block diagram of digital eNVM synaptic core is shown in Fig. 5(c). By replacing the SRAM core memory with eNVM without much modification on the whole digital circuit architecture, we potentially get smaller synaptic core area. The way the digital eNVM synaptic core works is very similar to the SRAM one, thus it can just use the traditional 1T1R array as the synaptic array. As mentioned in Section I, we have to group multiple binary 1T1R cells along the row as one synapse to represent a higher weight precision.

The weighted sum operation in digital eNVM synaptic core is also row-by-row-based. After the memory data are read out by the voltage S/A, adder and register will perform accumulation on the partial weighted sum through row-by-row. One key difference compared to the SRAM synaptic core is the use of multiplexer (Mux). As the cell size in 1T1R array is much smaller, it will not be area-efficient to put all the read periphery circuits underneath the array. Therefore, it is necessary to use a Mux to share the read periphery circuits among synaptic array columns. However, this inevitably increases the latency of weighted sum as time multiplexing is needed because of the sharing. For the weight update, the column decoder can select a group of synapses at a time depending on the design, and the programming voltages will be provided from the decoder driver. Two phases are required to program the cells to be ON and OFF because they need different WL voltages.

3) Analog eNVM Synaptic Core: In NeuroSim, the analog eNVM-based synaptic core supports two types of the synaptic array architecture: 1) the crossbar and 2) the pseudo-crossbar array architectures, as described earlier in Sections II-B and II-C. In this paper, we will only discuss the pseudo-crossbar array architecture. As shown in Fig. 5(d),

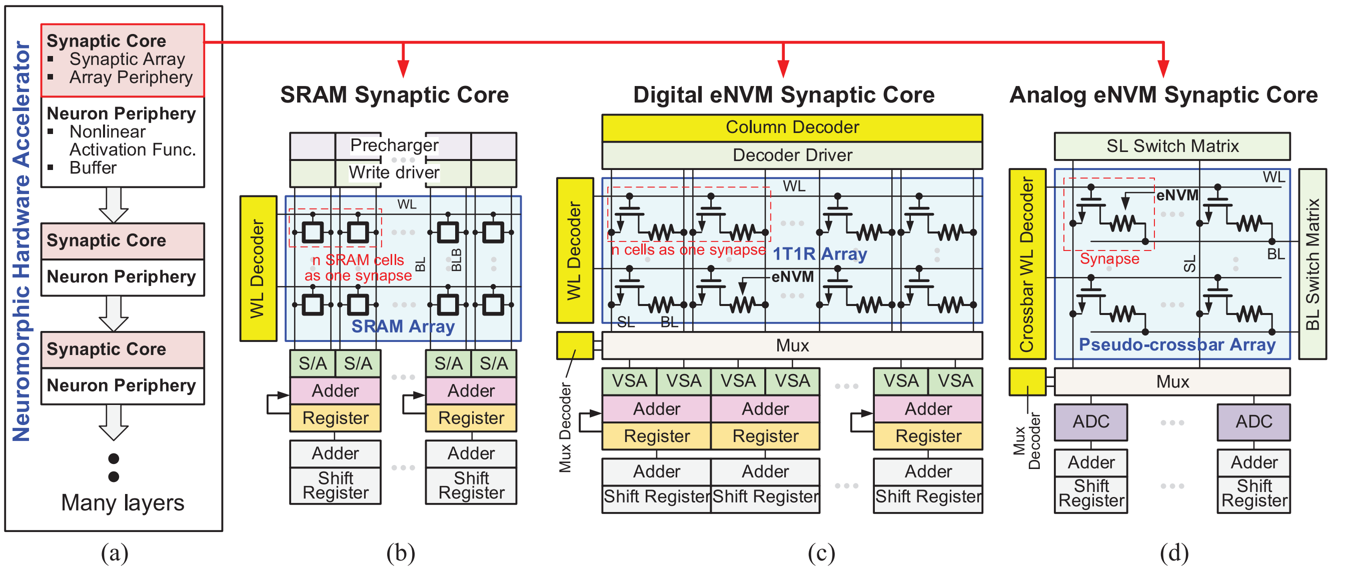

Fig. 5. (a) Overview of high-level architecture with neuromorphic hardware accelerator. (b) Circuit block diagram of SRAM synaptic core. (c) Circuit block diagram of digital eNVM synaptic core. (d) Circuit block diagram of analog eNVM synaptic core with the pseudo-crossbar array.

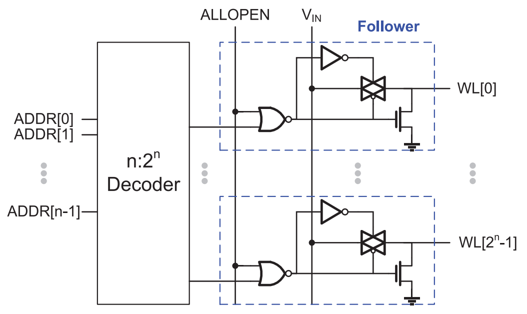

Fig. 6. Circuit diagram of the crossbar WL decoder. Follower circuit is attached to every row of the decoder to enable activation of all WLs when ALLOPEN = 1.

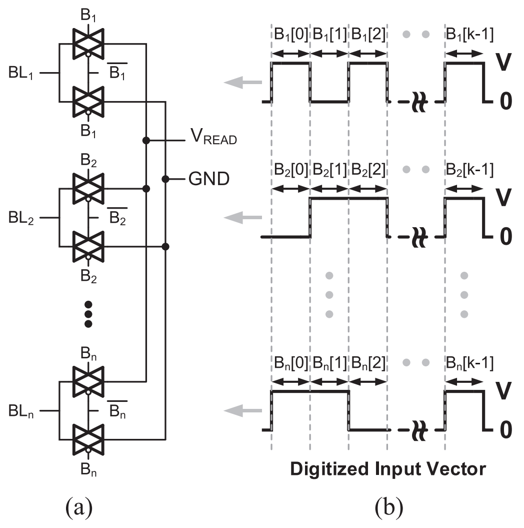

Fig. 7. (a) Transmission gates of the BL switch matrix in the weighted sum operation. A vector of control signals (B_1–B_n) from the registers (not shown here) decide the BLs to be connected to either a voltage source or ground. (b) Control signals in a bit stream to represent the precision of the input vector.

the peripheral circuits in analog eNVM synaptic core are quite different than the one in the SRAM and digital eNVM synaptic core. First, the WL decoder is modified to be "crossbar WL decoder," which has an additional feature to activate all the WLs for making all the transistors transparent for weighted sum. Inspired from [16], the crossbar WL decoder is constructed by attaching the follower circuits to every output row of the traditional decoder, as shown in Fig. 6. If ALLOPEN = 1, the crossbar WL decoder will activate all the WLs no matter what input address is given, otherwise it will function as a traditional WL decoder.

Second, switch matrices are used for fully parallel voltage input to the array rows or columns. Fig. 7(a) shows the BL switch matrix for example. It consists of transmission gates that are connected to all the BLs, with control signals (B_1–B_n) of the transmission gates stored in the registers (not shown here). In the weighted sum operation, the input vector signal is loaded to B_1–B_n, which decide the BLs to be connected to either the read voltage or ground. In this way, the read voltage that is applied at the input of transmission gates can pass to the BLs and the weighted sums are read out through SLs in parallel. If the input vector is not 1 bit, it should be encoded using multiple clock cycles. The reason why we do not use analog voltage to represent the input vector precision is the I–V nonlinearity of eNVM cell, which will cause the weighted sum distortion or inaccuracy [29]. As shown in Fig. 7(b), B_1–B_n are a vector of bit streams. Similarly, the adder and shift register pair in Fig. 5(b) will perform shift and add on the weighted sum results of all bit cycles to obtain the final weighted result.

To convert these analog weighted sum currents to digital outputs, we use the read circuit [36] to employ the principle of the integrate-and-fire neuron model. The read circuit integrates the weighted sum current on the finite capacitance of the array column. Once the voltage charges up above a certain threshold, the read circuit fires an output pulse and the capacitance is discharged back. The precision required for this analog-to-digital conversion (ADC) determines the pulse width in each bit of the input vector. As the cell size in 1T1R array is much smaller compared to the ADC size, multiple synaptic array columns may share one ADC through Mux to improve the area efficiency, which is similar to the case in digital eNVM synaptic core. However, this inevitably increases the latency of weighted sum as time multiplexing is needed because of the sharing.

Analog eNVM synaptic cores can also use multiple multibit cells to represent a higher precision, but it needs an extra Mux and an extra adder and shift register pair at the bottom of synaptic core to perform the summing. On the other hand, the weight update operation in analog eNVM synaptic core is performed row-by-row with the write pulses coming from the SL switch matrix. As mentioned in Section II-C, the weight update operation requires two phases because there are two voltage polarities for weight increase and decrease.

C. Transistor and Memory Cell Models

At the device level, NeuroSim is featured with various design options in transistor technologies and memory cells. The transistors can be configured to be high-performance or low-standby-power type with different technology nodes from 130 nm down to 7 nm, where FinFET is used at 14 nm and beyond. The transistor models are calibrated based on predictive technology model (PTM) [37]. Compared to industry transistor models, PTM is available to the public and it has a wide range of technology nodes, which is suitable for the design space exploration at the early design stage. Important parameters in transistor models include device W/L, the operating voltage (V_{DD}), threshold voltage (V_{TH}), gate and parasitic capacitance, and nMOS/pMOS saturation/off current density across different temperatures, etc. In particular, V_{TH} is extracted at the gate voltage (V_{GS}), where the drain current density (J_{DS}) is 300 nA/μm under $V_{DS} = V_{DD}$. In bulk MOSFET, the total gate capacitance is the sum of ideal, fringe, and overlap gate capacitance, while the total drain capacitance includes the capacitance in the diffusion region from junction to bottom, channel and the other three sidewalls. Based on these parameters, the area and intrinsic RC model of standard logic gates (INV, NAND, NOR, transmission gates, etc.) can be calculated analytically thus the circuit-level performance metrics of each subcircuit module can be estimated.

The design of SRAM and eNVM cells in NeuroSim is also flexible. We use conventional 6T SRAM (extendable to 8T SRAM), where all transistors' W/L can be adjusted. The transistor technology defined for other digital circuits also applies to SRAM's transistors. On the other hand, eNVM cells in NeuroSim have basic device parameters, such as max/min conductance, read/write voltage and pulse width, the number of conductance states (weight precision), *I–V* nonlinearity degree, etc. These parameters play an important role in the array-level performance, and will further affect the peripheral circuit design in the synaptic core.

D. Customization

In NeuroSim, synaptic cores are like standard templates. For a more complex design, it may be necessary to insert one or more hierarchical layers that uses the synaptic cores or other circuit modules as building blocks. If the users could not find a circuit module that serves their needs, they can build up a new module with the assistance of NeuroSim's infrastructure, which is relatively easier than writing a new module from scratch as NeuroSim has a clear device-to-circuit hierarchy for doing so. At the device level, the customization is even easier because the users can just modify the device parameters to fit different device characteristics. However, it should be noted that NeuroSim only has basic eNVM parameters as mentioned in Section III-C. It does not capture detailed device properties, such as weight update nonlinearity and variations, as it is currently not designed to provide weighted sum results or new weight matrix pattern as feedback thus the basic device parameters are sufficient for the circuit-level performance evaluation.

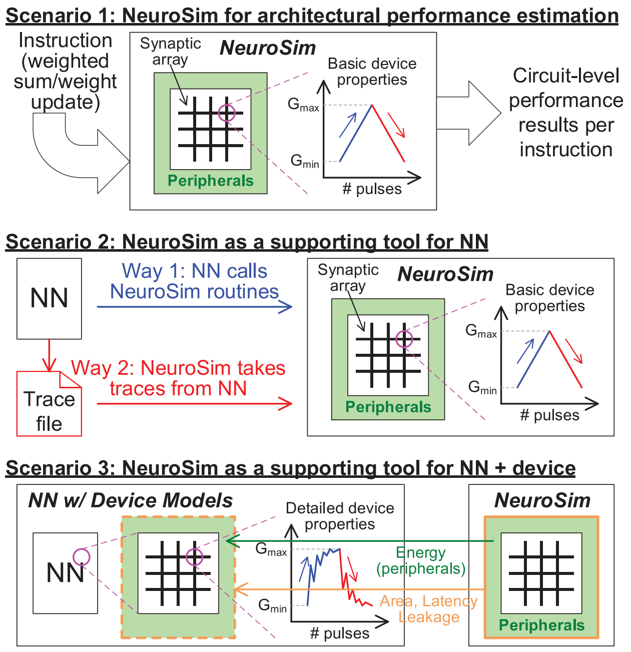
Scenario 1: NeuroSim for architectural performance estimation
Instruction (weighted sum/weight update)
Synaptic array
NeuroSim
Basic device properties
Gmax
Gmin
# pulses
Peripherals
Circuit-level performance results per instruction
Scenario 2: NeuroSim as a supporting tool for NN
NN
Way 1: NN calls NeuroSim routines
Way 2: NeuroSim takes traces from NN
Trace file
Synaptic array
NeuroSim
Basic device properties
Gmax
Gmin
# pulses
Peripherals
Scenario 3: NeuroSim as a supporting tool for NN + device
NN w/ Device Models
NN
Detailed device properties
Gmax
Gmin
# pulses
Energy (peripherals)
Area, Latency Leakage
NeuroSim
Peripherals

Fig. 8. Different usage scenarios for NeuroSim.

E. Usage of NeuroSim

As a circuit-level macro model, NeuroSim does not incorporate the learning algorithms, and it estimates the circuit-level performance of a synaptic core by taking either the data patterns of the input vector and weight matrix from the algorithm, or the average parameters of these patterns to have a good approximation of performance evaluation. For the latter one, for example, we can assume the activity of the input vector is 0.5 (50% 1 and 0 in the vector). At the device level, it may assume an average conductance of the synaptic devices and an average number of pulses for the weight update operation in analog eNVM synaptic core. To illustrate how NeuroSim works, we have considered three different usage scenarios as shown in Fig. 8, and they are described below.

1) NeuroSim for Architectural Performance Estimation: In this scenario, NeuroSim alone is used to estimate the circuit-level performance metrics of neuro-inspired architectures. As mentioned earlier, a synaptic core in NeuroSim takes weighted sum or weight update instruction with specified data pattern or average parameters to calculate the circuit-level performance per instruction, and it will quickly show the performance breakdown results from the synaptic core to its subcomponents. Thus, using NeuroSim alone is very handy for quick circuit-level performance benchmark without the need to run a full SPICE simulation. The prototype of NeuroSim has been used in this scenario in our several earlier works, such as [33], [38], and [39].

2) NeuroSim As Supporting Tool for NN: In this scenario, NeuroSim is used as a supporting module to provide circuit-level performance estimation for NN simulators, which is helpful for NN researchers to explore the design space of NN architectures at early design stage. This scenario can be done in two ways. The first way is that the NN simulator calls NeuroSim routines at every weighted sum and weight update operation, which makes the performance evaluation instruction-accurate. However, this approach may not be applicable if the NN simulator is not compatible with C++ NeuroSim interface. The second way is that NeuroSim takes the trace of data patterns that are recorded during the run-time of the NN simulator, which is essentially a trace-based simulation. This approach is much simpler than the first one and has no limitation on the platform, but it is much less efficient and requiring more simulation time to fetch the data from a trace file as it can be very large in size.

3) NeuroSim As Supporting Tool for NN+Device: This scenario is very similar to the second one, except the difference that part of the circuit-level performance estimation that NeuroSim provides may only be on the array peripherals, because the NN simulator has already incorporated a more complex synaptic array and device behavioral model. In this example, the NN simulator can estimate the energy consumption of the synaptic array more precisely and efficiently with its device model, thus NeuroSim is only responsible for the energy consumption on the array peripherals. For other performance metrics, NeuroSim still provide the estimation based on the whole architecture because they are more at the scope of architecture or circuit level. As mentioned in Section I, several works [10]–[12] published by the device community have demonstrated such an NN+device framework for evaluation of learning accuracy with various synaptic array and device characteristics, but they cannot address the circuit-level performance. Thus, we believe that NeuroSim can be a good supporting tool to fill up the gap between the algorithm and device for these frameworks as well as enabling the co-optimization of circuit and device for the device engineers.

F. Limitations of NeuroSim

Despite that NeuroSim features a wide variety of design options for the usage/support of circuit-level performance benchmark in neuro-inspired architectures, there are still several aspects that NeuroSim has not incorporated yet, leaving the room for future improvement. These include the following.

1) The ability to automatically map NN to several partitions of synaptic core and neuron periphery.
2) The interconnection, routing and network topology of synaptic cores at the architecture level.
3) The overhead of off-chip memory access.
4) A complete set of modules in support of machine learning NNs, such as convolutional NN (CNN) or recurrent NN (RNN).
5) The ability to adapt other NN types, such as spiking NN (SNN).

For 1), currently the users have to manually instantiate the synaptic cores by providing the synaptic array sizes that equal to the weight matrix sizes of the algorithm, thus only custom design is supported. The automatic mapping of the weight matrix sizes to arbitrary synaptic array sizes is to be developed for reconfigurable design. For 2), the overhead of latency and energy due to interconnection or routing between synaptic cores may become noticeable as the synaptic array size scales up. This will be the issue to solve after 1) is done. For 3), the overhead of off-chip memory access cannot be ignored if only part of the weights are stored on-chip. NeuroSim may have to be integrated with some third-party C++ DRAM modules (e.g., DRAMSim2 [40]) to take this overhead into account. For 4), currently NeuroSim partially supports CNN but more modules are still under development. For example, it has the module for the convolutional kernel and average pooling but no maximum pooling or batch normalization. On the other hand, RNN requires a different type of synaptic core that can achieve recurrent connections, which is not included in NeuroSim yet. Therefore, the users may have to bring their own design to NeuroSim if there is no existing module available there. For 5), the synaptic core and other subcircuit modules in NeuroSim are designed to support the key operations in machine learning NNs in a synchronous fashion. Event-driven asynchronous SNN works in a different way that the key operations rely on the timing between spikes to encode information, which NeuroSim cannot implement with its current form. Considering the limitations listed above are more at the algorithm and architecture level, at the current stage we would like to position NeuroSim as a circuit-level macro model that is most suitable for the device engineers to quickly benchmark various synaptic devices and neuro-inspired architectures with a basic NN algorithm.

IV. Performance Estimation Models

As a circuit-level estimation tool, NeuroSim is beneficial in exploring the design space of neuro-inspired architectures at early design stage. Typical circuit-level performance metrics include the area, latency, dynamic energy, and leakage power. Compared to the time-consuming SPICE simulation, NeuroSim provides fast estimation of the performance metrics using analytical models or predefined values provided by the user with reasonable accuracy. In this section, we introduce the performance estimation models in NeuroSim.

A. Model Setup

Fig. 9 shows the basic execution flow of subcircuit module functions to obtain the performance results of subcircuit modules. Before performance estimation, the subcircuit module has to be constructed and initialized. In the initialization step, functionality of the subcircuit module is outlined, such as the module interface, operating modes, and logic gates with sizing information (transistor W/L). In general, we predefine the transistor W/L for the logic gates in subcircuit modules according to the drivability that are needed. Specifically, we design the transistor W/L for the transmission gates that drives the array, such as the ones in the decoder driver, switch matrix and Mux of the eNVM-based synaptic array. We consider the worst case where the synaptic array has all its eNVM at the lowest resistance, and calculate the maximum effective resistance of the transmission gates (R_{TG}) under a coefficient of IR drop tolerance (IR_DROP_TOL):

$$R_{\text{TG}} = R_{\text{WORST\_ROW/COL}} \times \text{IR\_DROP\_TOL} \quad (2)$$

where $R_{\text{WORST\_ROW/COL}}$ is the total resistance of all eNVM cells in parallel in a row or column depending on either the transmission gate connects to the array row or column. By setting up a small IR_DROP_TOL (0.1 by default), we can make sure the input voltage can be delivered into the array without noticeable degradation in most cases.

At the architecture level, the flow is similar to the one for subcircuit modules. We show the execution flow of a synaptic core as a basic example of architecture in Fig. 10. In the initialization step of synaptic core, initialization of all subcircuit modules that belong to this synaptic will be performed. The same organization is also applied for the rest of performance estimation functions. In this way, a well-defined nested hierarchy from subcircuits to architectures can be constructed, enabling bottom-up level-by-level performance estimation.

B. Area Estimation

Once the transistors' sizing in each logic gate is known, the logic gates' layout height or width can then be calculated given the other dimension fixed. For example, if the driver

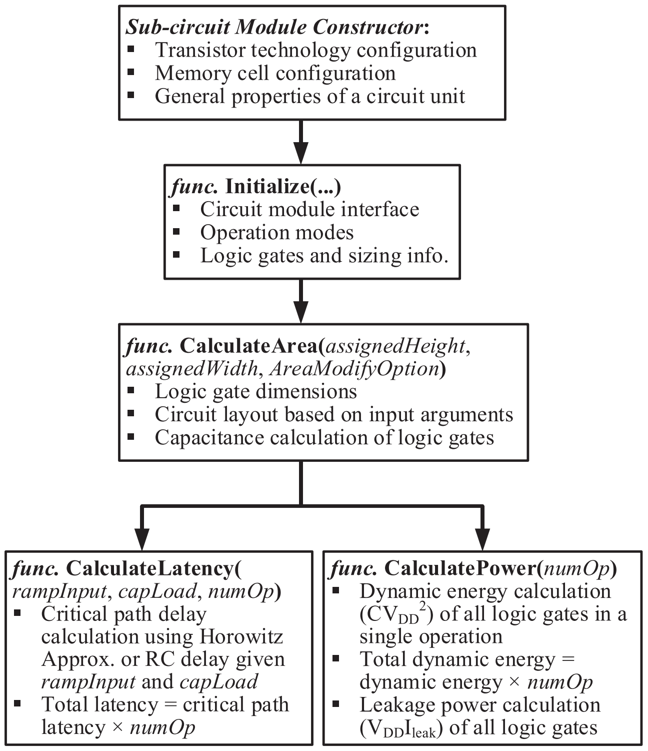
Sub-circuit Module Constructor:
▪ Transistor technology configuration
▪ Memory cell configuration
▪ General properties of a circuit unit
func. **Initialize(...)**
▪ Circuit module interface
▪ Operation modes
▪ Logic gates and sizing info.
func. **CalculateArea(***assignedHeight*, *assignedWidth*, *AreaModifyOption***)**
▪ Logic gate dimensions
▪ Circuit layout based on input arguments
▪ Capacitance calculation of logic gates
func. **CalculateLatency(***rampInput*, *capLoad*, *numOp***)**
▪ Critical path delay calculation using Horowitz Approx. or RC delay given *rampInput* and *capLoad*
▪ Total latency = critical path latency × *numOp*
func. **CalculatePower(***numOp***)**
▪ Dynamic energy calculation (CV_{DD}^2) of all logic gates in a single operation
▪ Total dynamic energy = dynamic energy × *numOp*
▪ Leakage power calculation ($V_{DD}I_{leak}$) of all logic gates

Fig. 9. Software execution flow of subcircuit module functions to estimate the performance metrics.

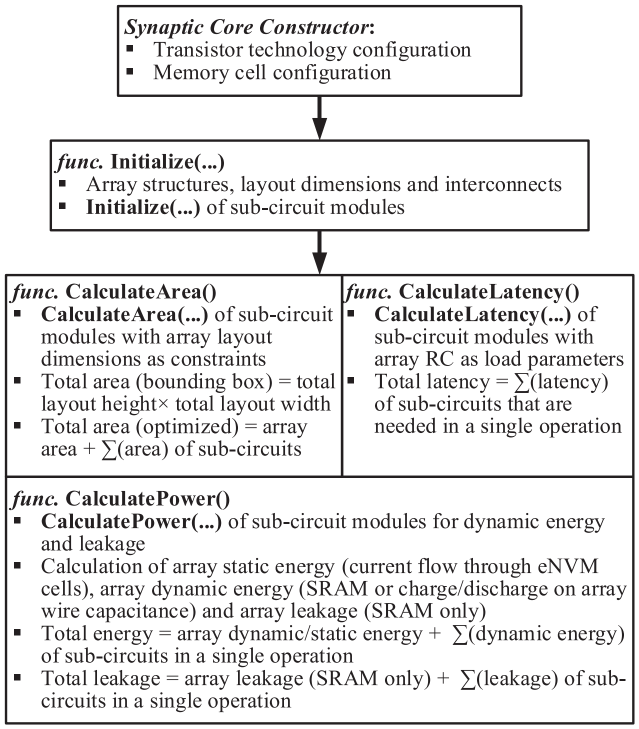
Synaptic Core Constructor:
▪ Transistor technology configuration
▪ Memory cell configuration
func. **Initialize(...)**
▪ Array structures, layout dimensions and interconnects
▪ **Initialize(...)** of sub-circuit modules
func. **CalculateArea()**
▪ **CalculateArea(...)** of sub-circuit modules with array layout dimensions as constraints
▪ Total area (bounding box) = total layout height× total layout width
▪ Total area (optimized) = array area + ∑(area) of sub-circuits
func. **CalculateLatency()**
▪ **CalculateLatency(...)** of sub-circuit modules with array RC as load parameters
▪ Total latency = ∑(latency) of sub-circuits that are needed in a single operation
func. **CalculatePower()**
▪ **CalculatePower(...)** of sub-circuit modules for dynamic energy and leakage
▪ Calculation of array static energy (current flow through eNVM cells), array dynamic energy (SRAM or charge/discharge on array wire capacitance) and array leakage (SRAM only)
▪ Total energy = array dynamic/static energy + ∑(dynamic energy) of sub-circuits in a single operation
▪ Total leakage = array leakage (SRAM only) + ∑(leakage) of sub-circuits in a single operation

Fig. 10. Software execution flow of performance estimation functions at the architecture level (a synaptic core for example).

has a large transistor W/L and its layout height is constrained by the array row pitch, then NeuroSim will try to find the minimum layout width of the driver that can accommodate this W/L by folding. In FinFET, the area estimation model becomes a little bit different because the number of fins needs to be an integer.

In general, we use the same cell layout height for most of the logic gates in subcircuit modules, and calculate its cell layout width depending on its transistor W/L. For the synaptic array, the layout dimensions of a memory cell can be predefined by the user. If the transistor size of the 1T1R or pseudo-crossbar array is estimated [using the same method as (2)] to be larger than the predefined memory cell size, NeuroSim will report an error and request a larger predefined size. After the synaptic array dimensions are determined, NeuroSim will estimate the layout dimensions of subcircuit modules. There are three input arguments as shown in Fig. 9. The first two arguments, assignedHeight and assignedWidth, are the constraints on the layout height and width, respectively. If one of them is provided, the other dimension can then be estimated to obtain the total area. If neither of them is provided, the logic gates will be placed in the most straightforward way for total area estimation. The third input argument, AreaModifyOption, can be specified for special adjustment of area, which has the following options.

1) *NONE:* This option is the regular one, indicating no further adjustment after the area estimation.
2) *MAGIC:* In this option, it is assumed that the layout of subcircuit module can be "magically" folded into any shape while conserving its total area, guaranteeing no waste of area. This option is designed for quick and simple estimation, but it will give the most optimistic result.
3) *OVERRIDE:* In this option, the estimated layout dimensions will be just overridden by the input arguments *assignedHeight* and *assignedWidth* for the total area. This is designed for the user-defined layout dimensions, or for the cases where both dimensions are constrained.

In the subcircuit module's area calculation function, the capacitance at logic gate level is estimated at the last step (Fig. 9) because the total drain capacitance is dependent on the layout structure of logic gate. For example, logic gates with different number of folding have different area and sidewall length of diffusion region. At the architecture level, NeuroSim provides two different total area estimations, as shown in Fig. 10. The first one is to estimate the bounding box area that encloses the entire layout of architecture. The other one is to directly sum up the area of array and subcircuit modules, which may be optimistic but it actually reflects the real case, where the layout is always optimized to save chip cost. For the area results in this paper, we use the latter one (the optimized one).

C. Latency Estimation

Once the capacitances at logic gate level are all known, the latency and dynamic energy consumption can then be estimated based on RC analyses. We follow the same methods of estimation in CACTI [34] and NVSim [35]. For digital logic gates, the latency is defined as the time required for the output voltage to reach the switching voltage threshold after the input voltage reaches it. We use Horowitz equation to calculate the latency in digital logic gates

$$\text{Latency} = \tau_{\mathrm{f}}\sqrt{\ln(\mathrm{v_s})^2 + \frac{2}{\text{rampInput}\times\tau_{\mathrm{f}}}\beta(1-v_s)} \qquad (3)$$

where v_s is the normalized switching voltage threshold (typically 0.5). rampInput is the input voltage ramp rate, and 1/rampInput represents the rise time of the input voltage signal. $\beta = 1/(g_m R)$ is the reciprocal of the normalized input transconductance g_m times the output resistance R. $\tau_f = \mathrm{RC}$ is the total RC time constant at the output node (assuming a step input). After the latency estimation, the output ramp rate of digital logic gates rampOutput will also be evaluated

$$\text{rampOutput} = (1-\text{v}_\text{s})/\text{Latency} \tag{4}$$

which can be provided as the rampInput to the next stage of the digital logic gate. For transmission gates used to pass analog voltage signals, we use 2.3RC (0–90% voltage rise time) instead of (3) to estimate the latency. The latency estimation at all levels always considers the worst-case scenario. Generally, there are three input arguments to the latency calculation function of a subcircuit module, as shown in Fig. 9. rampInput determines the voltage ramp rate to the input of the subcircuit module. capLoad is the load capacitance at the output node of subcircuit module. rampInput and capLoad are required for the critical path latency calculation. The third argument, numOp, is the number of repeated operations, which is designed for the convenience of the higher levels that may need multiple times of access to a single subcircuit module. The total latency of a subcircuit module can then be regarded as the critical path latency multiplied by numOp.

At the architecture level, the total latency can be calculated as the sum of latency of the subcircuit modules, as shown in Fig. 10. For the weighted sum operation of an eNVM synaptic core, the array RC is considered as the load parameters for the subcircuit modules that drives the array. For the weight update operation of an eNVM synaptic core, the latency of device weight tuning is included in the latency calculation of switch matrices.

D. Power Estimation

In the power estimation function of subcircuit modules, the dynamic energy consumption (expressed as $\text{CV}_{\text{DD}}{}^2$) and leakage power are calculated, as shown in Fig. 9. Since all the capacitances at logic gate level are known, the dynamic energy consumption of subcircuit module can then be calculated by summing up the $\text{CV}_{\text{DD}}{}^2$ at all nodes. Similarly, if numOp is given, the total dynamic energy consumption in a number of operations can be obtained.

In eNVM synaptic array, the energy consumption is mainly static energy consumption (i.e., the current flow through eNVM cells), as shown in Fig. 10. The energy consumption on the selected analog eNVM cell at weight increase/decrease phase can be simply written as

$$E_{\text{cell}} = \text{GV}_W^2\text{NT}_{\text{PULSE}}. \tag{5}$$

In (5), G is the conductance of a cell. V_W is the write voltage for weight increase/decrease. N is the number of applied write pulses and T_{PULSE} is the pulse width. Besides the eNVM cell, the dynamic energy consumption on the array wire capacitance as well as SRAM cells (for SRAM architecture) will also be calculated. Then, the total energy consumption for a synaptic core can be estimated as the sum of the dynamic/static energy consumption of array and the dynamic energy consumption of subcircuit modules.

On the other hand, leakage power represents the power consumption due to subthreshold leakage current (I_{leak}) in the transistor channel when the transistor is turned off, where the simplest form of expressing it is $V_{\text{DD}}I_{\text{leak}}$. For a simple logic like INV, I_{leak} is just the average of nMOS and pMOS off current. For a NAND or NOR logic that has more than one input, we estimate the leakage current based on the average case. In this way, an additional predefined ratio will be applied to the

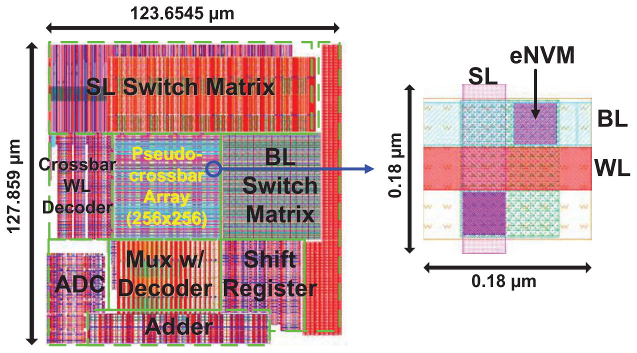

Fig. 11. Example layout of the analog eNVM synaptic core (256 × 256 array size) at FreePDK 45 nm.

leakage power calculation result. For example, the leakage of a NAND3 can be expressed as

$$\text{Leakage}_{\text{NAND3}} = V_{\text{DD}}I_{\text{off,PMOS}} \times 3 \times \text{AR\_LEAK}_{\text{NAND3}} \tag{6}$$

where $\text{AR\_LEAK}_{\text{NAND3}}$ represents the average ratio for leakage current in a NAND3 logic. In the synaptic array, the total leakage power will be simply the sum of leakage of SRAM cells (for SRAM architecture) and all subcircuit modules, as shown in Fig. 10. eNVM cells do not need power to maintain their data thus they do not have leakage.

E. Validation

NeuroSim offers a wide variety of design options for benchmarking neuro-inspired architectures. Being the essential bases for the entire simulation framework, the parameters in subcircuit modules, memory cell and transistor models should be accurate enough to support the validity of NeuroSim. In such context, we have performed SPICE and layout-level calibration of subcircuit modules to validate the analytical models. As mentioned in Section III-C, the transistor model parameters are calibrated based on PTM. The area estimation, including logic gates and subcircuits, is based on generic design rules. As shown in Fig. 11, we have calibrated the area estimation of an analog eNVM synaptic core with an array size of 256×256 at 45 nm technology node by comparing to its layout using FreePDK45 process design kit [41]. As is shown in the layout, the peripheral circuits (i.e., switch matrix) take substantial area due to the requirement of relaxing W/L for transmission gate for minimizing the IR drop to maintain good accuracy in the analog computation in the synaptic array. The entire layout area is measured to be 15 810 μm^2, with a cell size of 0.0324 μm^2 (4 F × 4 F), while the area estimation by NeuroSim (optimized) is 15 454 μm^2, achieving an error rate of −2.5%.

For latency, dynamic energy, and leakage power consumption, we pick the representative modules for validation, such as the decoder, adder, Mux, and switch matrix. As shown in Fig. 12, we calibrated the analytical equations in these performance estimation models at different synaptic array sizes from 8 × 8 to 256 × 256 with SPICE simulation based on PTM at 22, 32, and 45 nm technology node. In Fig. 12, the latency and dynamic energy of the decoder is more like a staircase function with respect to the array size. This is because the decoder has two stages and every two address bits will be predecoded, thus the decoder structure will have less changes from 2^{N-1} to 2^N address bits where N is an even number. On the other hand, the latency of Mux and switch matrix does not increase with larger array size, because all the signal paths are independent and parallel. In Fig. 12, the leakage

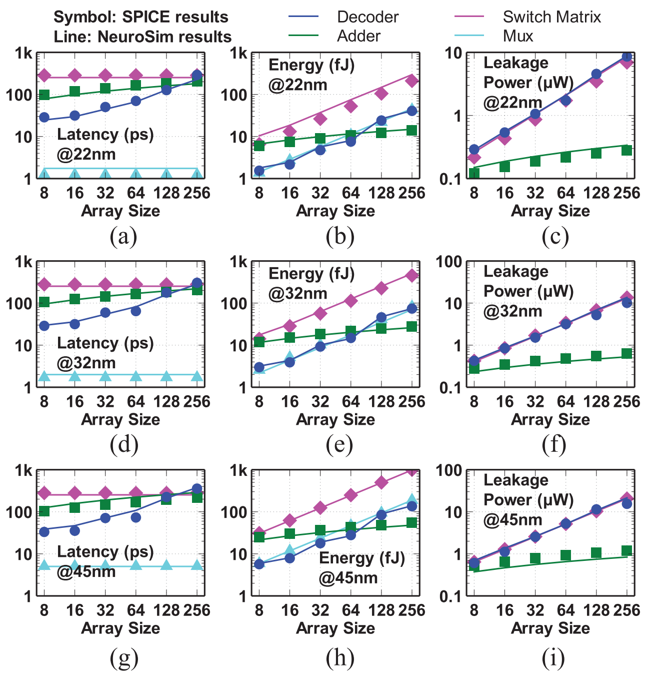

Fig. 12. Validation on main circuit modules (decoder, switch matrix, adder, mux) with different synaptic array sizes: (a) latency at 22 nm, (b) energy at 22 nm, (c) leakage power at 22 nm, (d) latency at 32 nm, (e) energy at 32 nm, (f) leakage power at 32 nm, (g) latency at 45 nm, (h) energy at 45 nm, and (i) leakage power at 45 nm technology node.

power of Mux is not shown, because it only has transmission gates where the transistor's subthreshold leakage current does not exist with a floating drain and the gate leakage current can be negligible. In Fig. 12, the average absolute error rates of the subcircuit modules at these technology nodes are ~14.86%, ~10.51%, and ~13.96% for the latency, dynamic energy, and leakage power, respectively. The validation results are reasonably accurate considering these performance metrics are modeled by simplified analytical equations as described in Sections IV-C and IV-D, which we believe is sufficient for a quick estimation of the circuit-level performance at early design stage.

V. Case Study by Using NeuroSim

In this section, we demonstrate the third usage scenario of NeuroSim in Section III-E using a 2-layer MLP NN with MNIST handwritten digits [42] as the training and testing dataset to implement online learning and offline classification. The network topology is 400(input layer)-100(hidden layer)-10(output layer). Four hundred neurons of input layer correspond to 20 × 20 MNIST image (edge cropped), and 10 neurons of output layer correspond to 10 classes of digits. Such simple 2-layer MLP can achieve 96%–97% in the software baseline. In online learning, the MLP simulator emulates hardware to train the network with images randomly picked from the training dataset (60k images) and classify the testing dataset (10k images). In offline classification, the network is pretrained by software, and the MLP simulator only emulates hardware to classify the testing dataset.

A. Adapt MLP Network to Hardware

For the hardware implementation, the MNIST input images are converted to black and white (1-bit) data to reduce the complexity of input encoding, as shown in Fig. 13(a). For

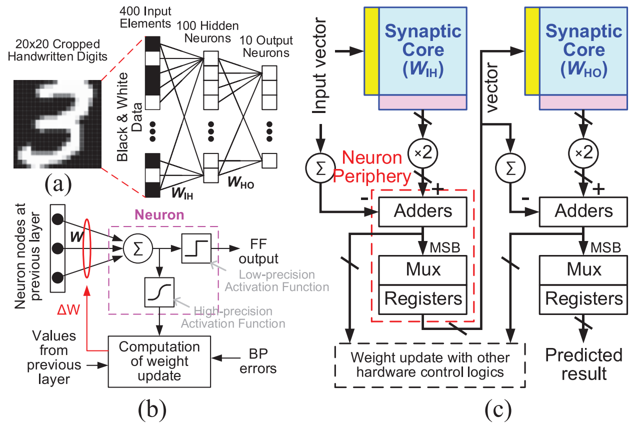

Fig. 13. (a) 2-layer MLP NN. (b) Schematic of a neuron node. (c) Circuit block diagram for hardware implementation of the 2-layer MLP NN.

design simplicity, the neuron node is modularized to take the weighted sum of 1-bit input data and truncate it to 1-bit output value through a low-precision activation function (Heaviside step function, e.g., a simple comparator circuit) for the input of next neuron node, as shown in Fig. 13(b). In this way, offline classification, which is purely feed forward (FF), can be realized in 1-bit, as demonstrated in [43]. However, the computation on the back propagation (BP) of weight update generally needs higher precision to update the small errors. Fig. 13(c) shows the circuit block diagram for hardware implementation of the 2-layer MLP network. The weighted sum operation is performed using the synaptic cores. However, the weights used in a regular synaptic array can only represent positive values ($W_H = 0 \sim 1$), while the weights in algorithm can be either positive or negative values ($W_A = -1 \sim 1$). The algorithm's weighted sum is then written as

$$W_A V = (2W_H - J)V = 2W_H V - \mathrm{JV} \tag{7}$$

where V is the input vector and J is the matrix of all ones that has the same dimension as W_A and W_H. In (7), $W_H V$ is the weighted sum output from the synaptic core. Therefore, we squeeze W_A from (−1~1) to the range of W_H(0~1): i.e., −1 is mapped to 0, 0 is mapped to 0.5, and 1 is mapped to 1. To reconstruct $W_A V$, we have to perform a two-step read from the array: first, we read out $W_H V$, and then multiply $W_H V$ by 2 using a 1-bit left-shift, and then subtract JV (basically the sum of vector) from $W_H V$ through the adder at the periphery. The MSB (sign bit in 2's complement notation) of the adder output will be the 1-bit output of the low-precision activation function. It should be noted that we only consider the main subcircuit modules for the neuron periphery at current stage of this paper, and the hardware for BP error calculation as well as the detailed control logics will be included in future work.

B. NeuroSim As Supporting Module for MLP Simulator

The MLP simulator is shown in Fig. 14. It has a hierarchical organization from the algorithm level down to the device level with consideration of synaptic array and realistic device properties in detail, and it can be regarded as a standalone functional simulator that is able to evaluate the learning accuracy and the circuit-level performance for the synaptic array only during learning. To form a complete framework, NeuroSim is needed to provide circuit-level performance estimation.

At the run-time of NN, the MLP simulator iteratively performs FF and BP, which contains a series of weighted sum and

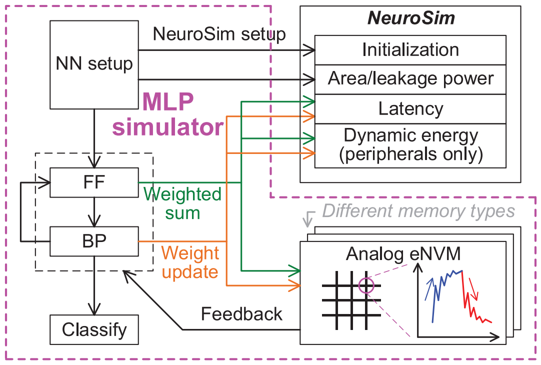

Fig. 14. NeuroSim as a supporting module to the MLP simulator. At the run-time of NN, the weighted sum and weight update instructions will be given to both the synaptic array/device model and NeuroSim for evaluation of computation error and circuit-level performances, respectively.

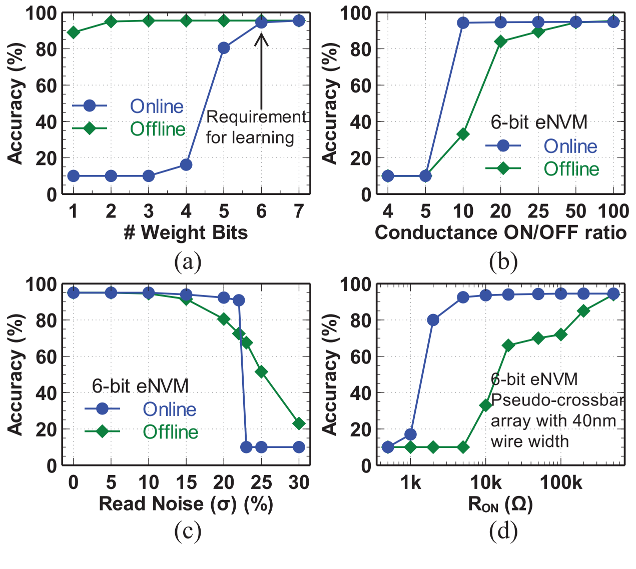

Fig. 15. Impact of (a) weight precision, (b) conductance ON/OFF ratio, (c) read noise, and (d) R_{ON} in online learning and offline classification.

weight update operations, respectively. Whenever a weighted sum or weight update instruction is given, the instruction will be passed to the synaptic array and device behavioral model for calculation of computation error, as well as passed to NeuroSim for evaluation of circuit-level performances. As mentioned in the third usage scenario in Section III-E, NeuroSim can be just responsible for the dynamic energy calculation of the array peripherals because the MLP simulator can better handle that of the synaptic array by itself.

C. Impact of Synaptic Device Properties on Accuracy

To quantify the impact of the aforementioned nonideal device properties in Section II-A, we performed sensitivity analyses in online learning and offline classification. Fig. 15(a) shows the requirement of weight precision. Because the memory resources are limited on-chip, we have to truncate the synapse weights into finite precisions. The result suggests that 6-bit weight is required for online learning, while 2-bit weight is needed for offline classification (at least for MNIST dataset) and 1-bit weight introduces slight degradation. Fig. 15(b) shows the learning accuracy with different conductance ON/OFF ratios. Limited ON/OFF ratio < 50 will degrade the accuracy of offline classification. The network may adapt itself to this limited ON/OFF ratio during learning thus the online learning can tolerate more (ON/OFF

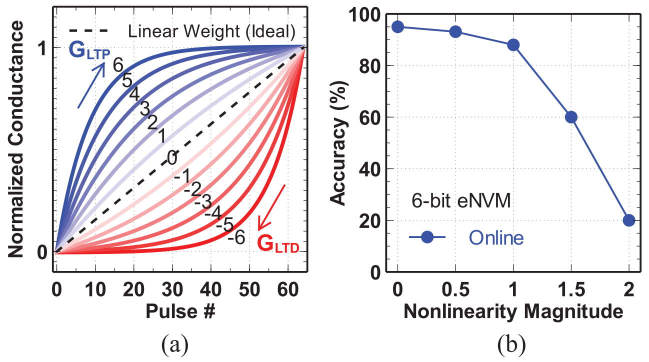

Fig. 16. (a) Analog eNVM device behavioral model of the nonlinear weight update. The nonlinearity is labeled from +6 to -6. (b) Impact of nonlinearity in online learning.

ratio > 10 is needed). For the read noise, the accuracy loss will become noticeable when σ is beyond 15%–25%, as shown in Fig. 15(c). It is also found in Fig. 15(b)–(d) that although the online learning can tolerate more on these nonideal properties, the accuracy drop in online learning is usually much sharper once the nonideal effect is beyond the tolerance range, which is probably because the network will deviate more from its correct form with both erroneous FF and BP results.

At array level, we also study the IR drop problem based on the pseudo-crossbar array architecture at a wire width of 40 nm. With smaller eNVM ON-state resistance (R_{ON}, which corresponds to the max conductance state), not only will the wire resistance get closer to R_{ON}, but also the access transistor needs to be sized larger, which makes the array wire longer and further worsen the IR drop problem. The results in Fig. 15(d) suggests that R_{ON} should be higher than 500 kΩ to prevent accuracy drop in the offline classification, while online learning can also tolerate more IR drop as long as $R_{\mathrm{ON}} > 10$ kΩ possibly because of the ability for the network to adapt itself to this spatial effect.

For the nonlinear weight update, we have built a device behavioral model. As shown in Fig. 16(a), the conductance change with number of pulses (P) is described with the following equations:

$$G_{\mathrm{LTP}} = B\left(1-e^{\left(-\frac{P}{A}\right)}\right) + G_{\min} \tag{8}$$

$$G_{\mathrm{LTD}} = -B\left(1-e^{\left(\frac{P-P_{\max}}{A}\right)}\right) + G_{\max} \tag{9}$$

$$B = (G_{\max} - G_{\min}) \Big/ \left(1 - e^{\frac{-P_{\max}}{A}}\right) \tag{10}$$

where $G_{\max}$, $G_{\min}$, and $P_{\max}$ are directly extracted from the experimental data, which represents the maximum conductance, minimum conductance, and the maximum pulse number required to switch the device between the minimum and maximum conductance states. A is the parameter that controls the nonlinear behavior of weight update, and B is simply a function of A that fits the functions within the range of $G_{\max}$, $G_{\min}$, and $P_{\max}$. A and B may be different in (8) and (9). As shown in Fig. 16(a), a set of nonlinear weight increase (blue) and weight decrease (red) behavior can be obtained by adjusting A, where each nonlinear curve is labeled with a nonlinearity value from $+6$ to -6. Fig. 16(b) shows that the impact of nonlinearity on the online learning accuracy is very critical. High accuracy can only be achieved with small nonlinearity ($<0.5\sim1$).

During weight update, synaptic devices usually show considerable variation from device to device, and even from

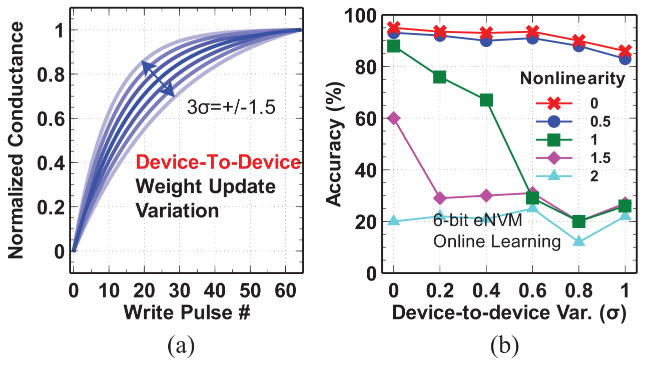

Fig. 17. (a) Illustration of the device-to-device weight update variation. (b) Online learning accuracy as a function of device-to-device weight update variation at different nonlinearity baselines.

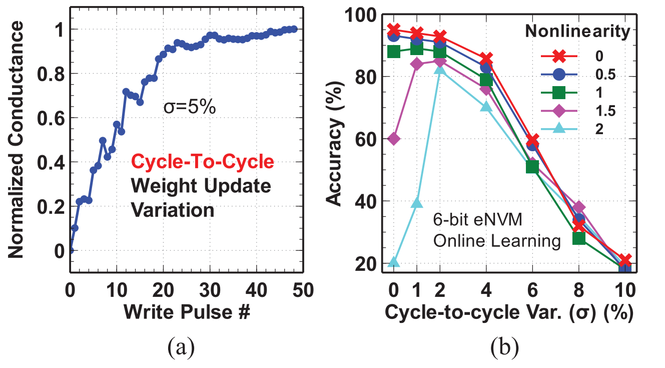

Fig. 18. (a) Illustration of the cycle-to-cycle weight update variation. (b) Online learning accuracy as a function of cycle-to-cycle weight update variation at different nonlinearity baselines.

pulse to pulse within one device. The effect of device-to-device weight update variation can be analyzed by introducing the variation into the nonlinearity baseline in Fig. 16(a) for each synaptic device, where it is defined as the nonlinearity baseline's standard deviation (σ) respect to 1 step of 6 steps in Fig. 16(a). In Fig. 17(a), an example of the device-to-device variation with $3\sigma = +/-1.5$ is illustrated. Generally, the network is resilient to the device-to-device variation [10], [11], however, the impact of device-to-device variation becomes prominent at high nonlinearity (>1), as shown in Fig. 17(b). For offline classification, there is no nonlinearity issue as the cell conductance can be iteratively programmed to the desired value [44].

On the other hand, the cycle-to-cycle weight update variation is referred to as the variation in conductance change at every programming pulse, as illustrated in Fig. 18(a). In this paper, the amount of cycle-to-cycle variation (σ) is expressed in terms of the percentage of entire weight range. As shown in Fig. 18(b), small cycle-to-cycle variation ($<2\%$) can alleviate the degradation of learning accuracy by high nonlinearity. The reason may be attributed to the random disturbance that aids convergence of the weights to an optimal weight pattern (i.e., to help the system jump out of local minima). Thus, synaptic devices with nonlinear weight update behavior may perform better than expected if they exhibit a little noisy weight update. However, too large variation ($>2\%$) overwhelms the deterministic update amount defined by BP thus is harmful to the learning accuracy.

Overall, Table I listed state-of-the-art synaptic devices in literature with the extracted realistic device parameters. The read noise and device-to-device weight update variation are

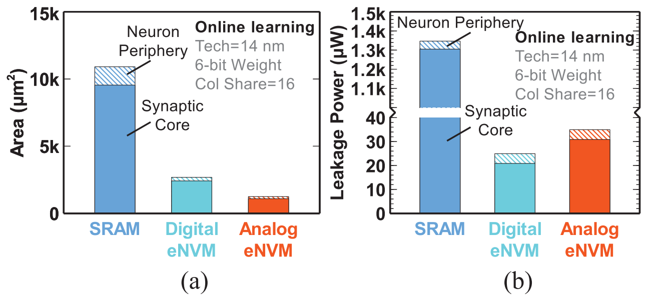

Fig. 19. (a) Area and (b) leakage power estimation of MLP architecture with SRAM, digital, and analog eNVM synaptic cores for learning.

not analyzed and included in this table because there are no sufficient experimental data and their impact on the learning accuracy is also less remarkable. However, the results suggest that today's analog eNVM devices are problematic to be used in online learning as well as offline classification. Different limiting factors shown in different devices: green color highlights good attributes, while red color highlights the major cause of learning failure. Therefore, we propose to set up the targeted eNVM specs that is able to achieve a good accuracy of 90% in online learning. As a reference, eNVM device with ideal specs is also listed, which is supposed to be as perfect as SRAM to achieve the same high accuracy of 94%–95% with 6-bit weights for online learning, because SRAM does not have the aforementioned nonideal device effects. To achieve these two desired specs, analog eNVM devices have to overcome significant challenges as suggested in Table I by more advanced device engineering. Of course, there could be circuit/architecture-level techniques to mitigate the accuracy degradation due to eNVM's nonideal effects [10]. For example, a dummy column could be added to compensate the nonzero off-state current (or limited ON/OFF ratio); nonidentical pulse programing scheme could be used to improve the weight update nonlinearity. However, these techniques come with design overhead. In this paper, we only benchmark the intrinsic device characteristics.

D. Circuit-Level Performance Benchmark

In the above section, we showed the results of learning accuracy, we will show the estimation of the area, latency, dynamic energy, and leakage power of the proposed MLP architectures for online learning and offline classification. As shown in Fig. 14, the area and leakage power can be directly obtained during the setup of NN, while the latency and energy consumption are calculated at the run-time. The benchmark is performed at 14 nm node, and we also assume the weights are 6-bit and there are 16 array columns sharing one read peripheral circuit (col share = 16), such as voltage S/A, ADC, etc., in eNVM-based synaptic cores. We estimated the total area of SRAM, digital, and analog eNVM-based architectures, as shown in Fig. 19(a). The analog eNVM core can achieve the smallest area due to smaller cell size and multiple bits per cell. In Fig. 19(b), the leakage power of these three architectures for the learning is also estimated. Unlike SRAM, the eNVM-based synaptic cores do not need power supply to maintain the data in memory cells thus their leakage power is much smaller compared to SRAM. On the other hand, the digital eNVM synaptic core has less leakage power than the analog one because a large portion of leakage power in the analog one comes from the SL/BL switch matrix.

However, SRAM has more advantages over digital and analog eNVM on the latency and dynamic energy consumption for online learning, as shown in Fig. 20. Although analog eNVM's

TABLE I
SPECS AND LEARNING ACCURACY OF REPORTED AND DESIRED ANALOG eNVMs

	Reported analog eNVMs for learning				Desired analog eNVMs for learning	
eNVM type	Ag:a-Si [20]	TaO_x/TiO_2 [21]	PCMO [22]	AlO_x/HfO_2 [23]	Targeted eNVM	Ideal eNVM
# of conductance states	97	102	50	40	64 (6 bits)	64 (6 bits)
Nonlinearity (weight increase/decrease)	2.4/-4.88	1.85/-1.79	3.68/-6.76	1.94/-0.61	1/-1	0/0
R_{ON}	26 MΩ	5 MΩ	23 MΩ	16.9 kΩ	200 kΩ	200 kΩ
ON/OFF ratio	12.5	2	6.84	4.43	50	50
Cycle-to-cycle variation (σ)	3.5%	<1%	<1%	5%	2%	0%
Accuracy for online learning	~73%	~10%	10%	~41%	~90%	~94.8%
Accuracy for offline classification	~63%	~10%	~20%	~10%	~94.5%	~94.5%

Green: good attributes, Red: major cause of learning failure

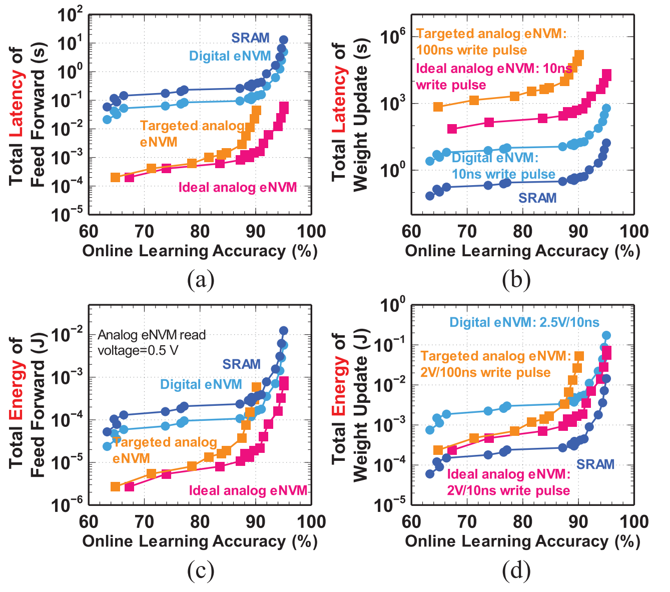

Fig. 20. Dynamical trace of (a) latency in feed forward, (b) latency in weight update, (c) energy in feed forward, and (d) energy in weight update during online learning.

parallel weighted sum operation in the FF is much faster than SRAM's and digital eNVM's row-by-row-based read, most of the latency is dominated by the eNVM's slow weight update, with two phases (two voltage polarities for weight increase and decrease) and multiple pulses per phase. Even the write pulse width of ideal eNVM is assumed to be 10 ns, the weight update latency of ideal eNVM is still ~28× slower than that of digital eNVM, and SRAM can be even faster than the digital eNVM by 36× because the write latency of a single SRAM cell (<ns) is much less than that of a single digital eNVM (10 ns) and we assume that only part of the row is selected at a time for weight update in digital eNVM. As there is only a small portion of weights being updated at each cycle, analog eNVMs with different pulse widths do not show any difference in weight update energy. In fact, most of the energy is consumed at biasing the unselected columns.

It should be noted that given the same constraint of on-chip area, SRAM has smaller capacity than eNVM if a large-scale or deep NN is implemented. Therefore, SRAM-based architecture requires more frequent off-chip memory access (i.e., from DRAM main memory) for training with large dataset. If the off-chip memory access is modeled with NeuroSim (i.e., by integration with DRAMSim2 [40], its latency/energy will be the dominant factor from the whole system point of view. In this sense, eNVM-based architecture may potentially solve this off-chip memory access bottleneck for implementing large-scale or deep NN.

TABLE II
BENCHMARK OF ARCHITECTURE WITH SRAM, DIGITAL, AND ANALOG eNVM-BASED SYNAPTIC CORE FOR OFFLINE CLASSIFICATION

	2-bit SRAM	2-bit digital eNVM	2-bit analog eNVM
Area	4450.8 μm^2	1071.2 μm^2	1247.3 μm^2
Latency	32.997 ms	10.39 ms	0.25 ms
Dynamic Energy	16.939 µJ	7.30 µJ	3.38 µJ
Leakage Power	475.67 µW	22.98 µW	35.29 µW

For offline classification, accuracy >93% can be achieved using either 2-bit SRAM and digital eNVM [equivalently Fig. 15(a)] or 2-bit targeted/ideal eNVM (Table I). Table II shows the circuit-level performance benchmark results of SRAM, digital, and analog eNVM-based architectures for offline classification on the entire testing dataset of 10k images. Without any training process, the analog eNVM-based architecture can be superior to the other two designs in terms of latency and energy.

VI. CONCLUSION

In this paper, we have introduced the synaptic array architectures, circuit modules, memory device/transistor models, functions, features, usage, and current limitations in NeuroSim with detailed descriptions. As a circuit-level macro model, NeuroSim alone can be a handy tool to estimate the circuit-level performance metrics of neuro-inspired architectures by taking trace of data patterns or average parameters. With clear abstractions of all hierarchical layers and well-defined interfaces of modules, NeuroSim can also be used as a supporting module to provide circuit-level performance estimation in NN learning algorithms. In the case study, we demonstrated the use of NeuroSim in a 2-layer MLP simulator to evaluate both the learning accuracy and circuit-level performance metrics at the run-time of online learning. NeuroSim provides a wide variety of design options in the circuit and device level (including nonideal properties) thus design tradeoffs can be investigated. In the case study of a small NN, it is found that SRAM still outperforms the digital and analog eNVM for online learning due to its excellent performance in weight update but with an overhead of much larger area and leakage power. eNVM may provide advantages for larger or deep NN, where the off-chip memory access will dominate due to small capacity of on-chip SRAM. Capable of parallel weighted sum operation, analog eNVM-based synaptic core is most suitable for the read-intensive applications, such as the offline classification, while the digital eNVM one may be a better choice for low standby power design. NeuroSim is an extendable platform that could be integrated with other machine learning NNs, such as CNN, which will be our on-going future work.

Acknowledgment

The authors acknowledge Z. Li, X. Sun, R. Liu, and A. Mallik for the discussions and help in this paper.

References

[1] J. Hasler and B. Marr, "Finding a roadmap to achieve large neuromorphic hardware systems," *Front. Neurosci.*, vol. 7, no. 118, pp. 1–29, 2013.

[2] J. Von Neumann, "The principles of large-scale computing machines," *Ann. History Comput.*, vol. 3, no. 3, pp. 263–273, Jul./Sep. 1981.

[3] R. Ananthanarayanan, S. K. Esser, H. D. Simon, and D. S. Modha, "The cat is out of the bag: Cortical simulations with 10^9 neurons, 10^{13} synapses," in *Proc. Conf. High Perform. Comput. Netw. Stor. Anal.*, Portland, OR, USA, 2009, pp. 1–12.

[4] S. Schmitt *et al.*, "Neuromorphic hardware in the loop: Training a deep spiking network on the brainscales wafer-scale system," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2017, pp. 2227–2234.

[5] B. V. Benjamin *et al.*, "Neurogrid: A mixed-analog–digital multichip system for large-scale neural simulations," *Proc. IEEE*, vol. 102, no. 5, pp. 699–716, May 2014.

[6] P. A. Merolla *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[7] Y.-H. Chen, T. Krishna, J. Emer, and V. Sze, "14.5 eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," in *Proc. IEEE Int. Solid-State Circuits Conf. (ISSCC)*, San Francisco, CA, USA, 2016, pp. 262–263.

[8] D. Shin, J. Lee, J. Lee, and H.-J. Yoo, "14.2 DNPU: An 8.1TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks," in *Proc. Int. Solid-State Circuits Conf. (ISSCC)*, San Francisco, CA, USA, 2017, pp. 240–241.

[9] S. Yu, Ed., *Neuro-Inspired Computing Using Resistive Synaptic Devices*. Cham, Switzerland: Springer, 2017.

[10] P.-Y. Chen *et al.*, "Mitigating effects of non-ideal synaptic device characteristics for on-chip learning," in *Proc. ACM/IEEE Int. Conf. Comput.-Aided Design (ICCAD)*, Austin, TX, USA, 2015, pp. 194–199.

[11] G. W. Burr *et al.*, "Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element," *IEEE Trans. Electron Devices*, vol. 62, no. 11, pp. 3498–3507, Nov. 2015.

[12] S. Agarwal *et al.*, "Resistive memory device requirements for a neural algorithm accelerator," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Vancouver, BC, Canada, 2016, pp. 929–938.

[13] P. Chi *et al.*, "PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," in *Proc. ACM/IEEE Int. Symp. Comput. Archit. (ISCA)*, Seoul, South Korea, 2016, pp. 27–39.

[14] X. Liu *et al.*, "Harmonica: A framework of heterogeneous computing systems with memristor-based neuromorphic computing accelerators," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 5, pp. 617–628, May 2016.

[15] A. Shafiee *et al.*, "ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *Proc. Int. Symp. Comput. Archit. (ISCA)*, Seoul, South Korea, 2016, pp. 14–26.

[16] L. Xia *et al.*, "MNSIM: Simulation platform for memristor-based neuromorphic computing system," in *Proc. ACM/IEEE Design Autom. Test Europe Conf. Exhibit. (DATE)*, Dresden, Germany, 2016, pp. 469–474.

[17] *MLP Simlator (+NeuroSim) Version 1.0*. [Online]. Available: https://github.com/neurosim/MLP_NeuroSim

[18] D. Kuzum, S. Yu, and H.-S. P. Wong, "Synaptic electronics: Materials, devices and applications," *Nanotechnology*, vol. 24, no. 38, 2013, Art. no. 382001.

[19] X. Guo *et al.*, "Temperature-insensitive analog vector-by-matrix multiplier based on 55 nm NOR flash memory cells," in *Proc. IEEE Custom Integr. Circuits Conf. (CICC)*, Austin, TX, USA, 2017, pp. 1–4.

[20] S. H. Jo *et al.*, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano Lett.*, vol. 10, no. 4, pp. 1297–1301, 2010.

[21] L. Gao *et al.*, "Fully parallel write/read in resistive synaptic array for accelerating on-chip learning," *Nanotechnology*, vol. 26, no. 45, 2015, Art. no. 455204.

[22] S. Park *et al.*, "Neuromorphic speech systems using advanced ReRAM-based synapse," in *Proc. IEEE Int. Electron Device Meeting (IEDM)*, Washington, DC, USA, 2013, pp. 625–628.

[23] J. Woo *et al.*, "Improved synaptic behavior under identical pulses using AlO_x/HfO_2 bilayer RRAM array for neuromorphic systems," *IEEE Electron Device Lett.*, vol. 37, no. 8, pp. 994–997, Aug. 2016.

[24] S. Yu *et al.*, "A low energy oxide-based electronic synaptic device for neuromorphic visual systems with tolerance to device variation," *Adv. Mater.*, vol. 25, no. 12, pp. 1774–1779, 2013.

[25] M. Prezioso *et al.*, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, no. 7550, pp. 61–64, 2015.

[26] D. Kuzum, R. G. D. Jeyasingh, B. Lee, and H.-S. P. Wong, "Nanoelectronic programmable synapses based on phase change materials for brain-inspired computing," *Nano Lett.*, vol. 12, no. 5, pp. 2179–2186, 2011.

[27] M. Suri *et al.*, "Phase change memory as synapse for ultra-dense neuromorphic systems: Application to complex visual pattern extraction," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, Washington, DC, USA, 2011, pp. 79–82.

[28] M. Hu, H. Li, Q. Wu, and G. S. Rose, "Hardware realization of BSB recall function using memristor crossbar arrays," in *Proc. Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2012, pp. 498–503.

[29] P.-Y. Chen *et al.*, "Technology-design co-optimization of resistive cross-point array for accelerating learning algorithms on chip," in *Proc. Design Autom. Test Europe Conf. Exhibit. (DATE)*, Grenoble, France, 2015, pp. 854–859.

[30] J. Liang and H.-S. P. Wong, "Cross-point memory array without cell selectors—Device characteristics and data storage pattern dependencies," *IEEE Trans. Electron Devices*, vol. 57, no. 10, pp. 2531–2538, Oct. 2010.

[31] P.-Y. Chen, L. Gao, and S. Yu, "Design of resistive synaptic array for implementing on-chip sparse learning," *IEEE Trans. Multi-Scale Comput. Syst.*, vol. 2, no. 4, pp. 257–264, Oct./Dec. 2016.

[32] S. Yu and P.-Y. Chen, "Emerging memory technologies: Recent trends and prospects," *IEEE Solid State Circuits Mag.*, vol. 8, no. 2, pp. 43–56, Jun. 2016.

[33] S. Yu *et al.*, "Scaling-up resistive synaptic arrays for neuro-inspired architecture: Challenges and prospect," in *Proc. IEEE Int. Electron Devices Meeting (IEDM)*, Washington, DC, USA, 2015, pp. 451–454.

[34] S. J. E. Wilton and N. P. Jouppi, "CACTI: An enhanced cache access and cycle time model," *IEEE J. Solid-State Circuits*, vol. 31, no. 5, pp. 677–688, May 1996.

[35] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi, "NVSim: A circuit-level performance, energy, and area model for emerging nonvolatile memory," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 31, no. 7, pp. 994–1007, Jul. 2012.

[36] D. Kadetotad *et al.*, "Parallel architecture with resistive crosspoint array for dictionary learning acceleration," *IEEE J. Emerg. Sel. Topic Circuits Syst.*, vol. 5, no. 2, pp. 194–204, Jun. 2015.

[37] *Predictive Technology Model (PTM)*. [Online]. Available: http://ptm.asu.edu/

[38] W. Qian *et al.*, "Energy-efficient adaptive computing with multifunctional memory," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 64, no. 2, pp. 191–195, Feb. 2017.

[39] P.-Y. Chen and S. Yu, "Partition SRAM and RRAM based synaptic arrays for neuro-inspired computing," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Montreal, QC, Canada, 2016, pp. 2310–2313.

[40] P. Rosenfeld, E. Cooper-Balis, and B. Jacob, "DRAMSim2: A cycle accurate memory system simulator," *IEEE Comput. Archit. Lett.*, vol. 10, no. 1, pp. 16–19, Jan./Jun. 2011.

[41] *FreePDK45*. [Online]. Available: https://www.eda.ncsu.edu/wiki/FreePDK45:Contents

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

[43] T. Tang, L. Xia, B. Li, Y. Wang, and H. Yang, "Binary convolutional neural network on RRAM," in *Proc. ACM/IEEE Asia South Pac. Design Autom. Conf. (ASP-DAC)*, 2017, pp. 782–787.

[44] L. Gao, P.-Y. Chen, and S. Yu, "Programming protocol optimization for analog weight tuning in resistive memories," *IEEE Electron Device Lett.*, vol. 36, no. 11, pp. 1157–1159, Nov. 2015.

Pai-Yu Chen, photograph and biography not available at the time of publication.

Xiaochen Peng, photograph and biography not available at the time of publication.

Shimeng Yu, photograph and biography not available at the time of publication.