IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 1, 2022, accepted April 12, 2022, date of publication April 18, 2022, date of current version April 21, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3167709

A Software-Circuit-Device Co-Optimization Framework for Neuromorphic Inference Circuits

PAUL QUIBUYEN, TOM JIAO, AND HIU YUNG WONG, (Senior Member, IEEE)
Electrical Engineering Department, San José State University, San Jose, CA 95192, USA

Corresponding author: Hiu Yung Wong (hiuyung.wong@sjsu.edu)

This work was supported in part by the San José State University College of Engineering Small Group Project Team Fund.

ABSTRACT Neuromorphic circuits, which usually use analog computation for vector-matrix multiplication (VMM) in neural networks (NN), are promising machine learning accelerators with much lower latency and power consumption than digital ones. Analog computation is expected to have a more efficient design space than digital computation since the signals are not digitized. Therefore, it is very suitable for Internet-of-Thing (IoT) applications that require ultra-low power consumption at a low cost. For IoT applications, sometimes it is also desirable to eliminate the digital circuits (such as adders, registers, shifters, multiplexers, and Analog-to-Digital Converters) between the VMM arrays to further reduce the power consumption. However, the optimization of a purely analog circuit is more difficult and requires full SPICE circuit simulations. In this paper, we present a software-circuit-device co-optimization framework using a python wrapper for automatic full circuit SPICE simulation and analysis for neuromorphic circuits. This framework allows users to experiment with how the NN design (software) affects the performance of the hardware neuromorphic circuits. It takes Verilog-A or SPICE models from calibrations or PDK in various technologies and emerging memories (such as ReRAM) without further calibration (unlike using behavior models). We show that the simulation time is reasonable even with hundreds of thousands of synapses under limited computation resources. Using ReRAM and a 45nm generic technology as an example, the effects of feedback network and OpAmp design, software ML architecture, and input data accuracy on the inference accuracy are studied.

INDEX TERMS Device-technology co-optimization (DTCO), emerging memory, neural network, neuromorphic computation, ReRAM, SPICE simulation.

I. INTRODUCTION

Machinelearning (ML), particularly the neural network (NN), has revolutionized almost every aspect of our daily life. There are two phases in an ML process, namely machine training and data inference [1]. In the machine training phase, a machine is trained by, usually, a large amount of data. In the data inference phase, the trained machine is used to infer the properties of the data (e.g. determines the number value of a hand-written digit image [2]). Both phases involve the movement of a large amount of data, resulting in an increase in latency and energy consumption [3]. This creates an almost impassable barrier to the further scaling of the traditional von Neumann computing architecture, in which the memory and computing units are separated by a higher and higher ''memory wall'' [4].

The associate editor coordinating the review of this manuscript and approving it for publication was Mostafa Rahimi Azghadi.

To circumvent the memory wall, Compute-in-Memory (CiM) has been proposed and attracted a lot of attention in the last decade [5]–[7]. By performing the computation in the memory element, most of the data movement is obviated and power consumption can be reduced substantially. Among various CiM ideas, using emerging memories, such as Resistive Random-Access-Memory (ReRAM) [8], [9], Ferroelectric Random-Access-Memory (FeRAM) [10], Spin Transfer Torque Random-Access-Memory (STT-RAM) [11], Phase Change Memory (PCM) [12], Electro-Chemical Random-Access-Memory (EC-RAM) [13], etc., is the most promising one to be used for the task for NN. This is because 1) they are non-volatile, 2) they naturally form an array for a single constant time step analog computation to replace the Vector-Matrix-Multiplication (VMM), which is the most power- and time-consuming operation in NN [6], [14], [15], and 3) they have very small form factors (e.g. ReRAM and PCM are simple cross-point memories

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

which can be formed using the Back-End-of-Line (BEoL) in a Complementary Metal-Oxide-Semiconductor (CMOS) process [16]). In this neuromorphic computing architecture, the emerging memories mimic the synapses in a biological neural network.

Due to its non-volatility and low energy consumption compared to the traditional architecture, the NN accelerator using emerging memory is expected to be used in Internet-of-Things (IoT) [17], [18], which is usually powered by an irreplaceable battery or energy harvested from the environment. Very often, the NN used in IoT is trained offline using large servers and the weights are uploaded to the IoT only for inference. Therefore, in this paper, only the inference properties of the emerging memory-based NN are studied.

Since emerging memory-based NN are mostly analog in nature, its optimization is not trivial. The accuracy, power consumption, area, and speed depend on the NN (number of layers and nodes), the Digital-to-Analog (DAC) converter accuracy, emerging memory electrical characteristics, amplifier properties, and the rectification unit performance. For IoT applications, to minimize the cost while satisfying the accuracy and power consumption requirements, a co-simulation framework for software-circuit-device optimization is necessary.

A. RELATED WORKS

There have been various studies of the neuromorphic circuits using simulations but most of them do not use SPICE simulations and cannot use the technology Process Design Kit (PDK) directly. In [6], a MATLAB framework was built to study the temperature, loading resistance, and input voltage range effect on the performance, however, without considering their interactions. It only studies the precision of circuit behavior instead of the accuracy of the final ML outputs. Since neuromorphic circuits are fault-tolerant, the findings might be over pessimistic. In [19], a system-level simulator is built intended to be used with behavioral models for large system simulation. However, calibration against the SPICE model is required for behavior models. Despite its lower speed, SPICE simulation provides more insight into circuit design and avoids the use of behavior models and additional calibrations. In [14], SPICE is used but only for studying the behavior of loading resistance and has no interaction with ML algorithm design.

Some work has been done to develop system-level simulation frameworks to quantify the performance trade-offs between area, read/write energy, read/write latency, and leakage power in implementing ReRAM arrays [20]–[24]. However, they do not cover other performance metrics when they are applied in compute in-memory architectures and neuromorphic circuits, such as classification accuracy, and computational speed.

Lammie *et al.* [23] developed a simulation framework that allows the sparse weights and neurons using L1 regularization and dropout during training. Also, the training is quantization-aware, which takes into account the limited resolution of the ReRAM weights. After training, the neuromorphic circuit is generated in SPICE or other RRAM-based DL simulation frameworks to obtain the test accuracy. The framework allows the simulation of various device nonidealities such as device-to-device variability, finite conductance states, stuck R_{ON}, R_{OFF}, etc., as well as functional models of the circuit blocks.

A Pytorch-based analog ANN simulation framework was developed with hardware-aware training and a CUDA-capable (GPU accelerated) C++ simulator [26]. The circuits such as DAC, ADCs, etc., are replaced with functional models and are not capable of transistor-level accuracy.

A simulation framework of differential-architecture crossbar arrays is developed in [27] to simulate spiking recurrent neural networks with PCM.

Finally, in [28], a comprehensive summary of various simulation frameworks of CiM is provided. The simulation frameworks are classified based on e.g. the programming languages, the capability of training simulations, the inclusion of periphery circuit simulation, the types of device supported, etc.

B. SUMMARY OF THIS WORK

In this paper, we developed and realized a software-circuit-device co-optimization framework for neuromorphic inference circuits using purely SPICE simulation based on our previous works [29]–[31]. This framework automatically constructs the neuromorphic circuit based on software ML. It can take the Verilog or SPICE model for its 4 major components, namely the DAC, neuromorphic memory, current comparator, and rectification unit. It performs full analog simulations. To minimize the energy consumption and the area for IoT applications, no select transistors are included and no digital circuits are used in the design (although they can be included if needed). This allows us to study the close interaction between the neuromorphic device and various parts of the circuits, which is only possible with a full analog circuit simulation. Since the periphery circuits are an important part of the simulation, using SPICE models from the PDK makes it more accurate and easier to perform co-optimization.

The paper is organized as the following. Section II explains the framework and its capabilities with an example to show the interplay of the resistors in the feedback network of the current comparator. Section III discusses the effect of the DAC and input voltage range on inference accuracy. Section IV discusses the OpAmp design consideration based on inference accuracy requirement and its interaction with the feedback network. Section V demonstrates the software-circuit co-optimization, followed by conclusions.

II. THE FRAMEWORK

A typical neuromorphic circuit for NN is shown in Fig. 1. As mentioned earlier, to minimize the area and power consumption for IoT applications, only pure analog neuromorphic circuits are studied. Therefore, it does not have

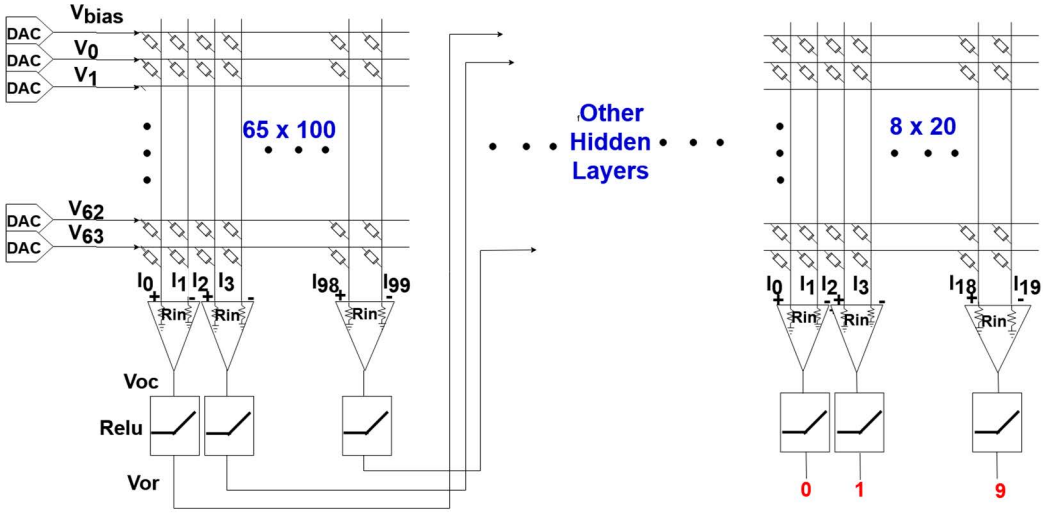

FIGURE 1. A typical neuromorphic circuit with the four critical components (DAC, ReRAM, Current Comparator, and Rectification Unit) implemented in this study is shown. The details of the cross-bar array can be found in Fig. 3. Each array corresponds to the VMM between two layers in a NN. The left-most one is the VMM from the input layer to the first hidden layer. The rightmost one is from the last hidden layer to the output layer. This circuit corresponds to the [50,20,8] NN for the UCI handwritten test set. Note that 2 arrays (for the VMM from 50 to 20 and 20 to 8) are not shown for clarity.

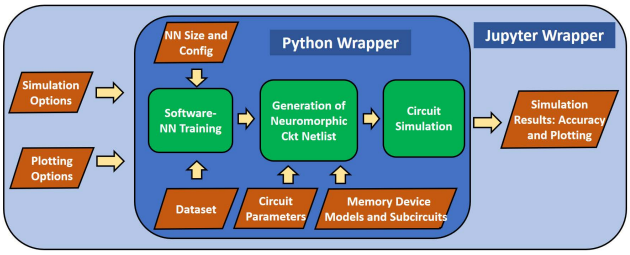

FIGURE 2. The Software-Circuit-Device co-optimization framework in this study.

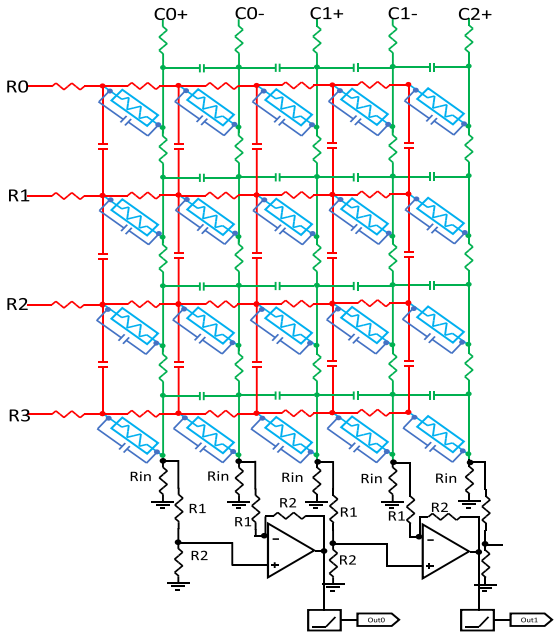

FIGURE 3. Crossbar array schematic including the parasitic resistance and capacitance.

inter-layer Analog-to-Digital Converters (ADC), multiplexers, registers, shifters, and adders. It has 4 major components, namely the DAC, emerging memory (ReRAM is shown as an example), current comparator, and rectification unit [32]. The weight of the NN is encoded as the conductance of the ReRAM. Since the conductance is always positive, to encode negative weights, two ReRAM is used to encode one weight and a current subtractor is used to convert the difference of the currents to reflect the true values. When the weight is positive, it is applied to the left string and the right string element is set to the maximum gap size to achieve minimal conductance (corresponds to $\sim$300kΩ) and vice versa when the weight is negative.

Fig. 2 shows the framework. It consists of a jupyter and python (version 3.7.3) wrapper. The jupyter wrapper is the graphical user interface of the framework which allows the user to view the plots and results of the simulations. It also serves as a user-friendly interface for setting the simulation and plotting options. The python wrapper receives these settings and initializes the training of a software-based Neural Network. The weights of the NN are then automatically mapped into resistances of the memory devices in the crossbar array (Fig. 1 and Fig. 3). In this study, ReRAM is used and its Verilog-A model is developed based on [33]. Temperature dependence of leakage current is added by calibrating to the experiment in [6]. Moreover, time integration methodology in the Verilog-A code is improved over the original code so that program and erase are independent of the initial bias before voltage sweeping [29]

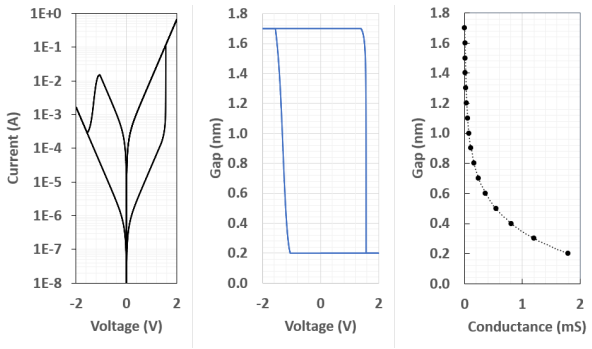

FIGURE 4. Electrical characteristics of the ReRAM used in this study. Left: Current-Voltage characteristics; Middle: Gap size as a function of voltage; Right: Gap size relationship to the conductance.

(a limitation in the original model). The framework is also compatible with other types of memory devices as long as the SPICE or Verilog-A model and the equation relating the conductance to the parameter of the device are provided. In the case of the ReRAM, the programmed parameter is the gap size between the filament and the top electrode. Fig. 4 shows the electrical characteristic of the ReRAM and its relationship to the gap size. Cadence Spectre is used for the circuit simulation [34] but other EDA software can be integrated using the same setup. A generic 45nm technology available in Cadence is used to design the peripheral circuits.

Once the weights have been mapped into the parameters of the memory devices, the python wrapper proceeds to generate the netlist of the neuromorphic circuit from the model files and subcircuit templates of the memory devices and other circuit blocks such as the DACs, current subtractor, and the rectification unit as shown in Fig. 1 to Fig. 3. The input test data is also scaled and converted into binary representation to be fed into the DACs.

In order to assess the accuracy of the neuromorphic circuit, multiple simulations are performed using the co-optimization framework. Each of these simulations corresponds to a data point in the test dataset. Multiprocessing is enabled in the framework allowing up to 30 parallel Spectre simulations.

As an initial illustration, using the framework, a software-based NN of size [50,20,8] (i.e. 3 hidden layers each with 50, 20, and 8 nodes, respectively) was trained with 1617 images from the UCI dataset of handwritten digits [35]. Each image is an 8×8 matrix. Therefore, the input layer has 64 nodes and the output layer has 10 nodes for the digits from 0 to 9. Fig. 1 illustrates the corresponding circuits with 8966 ReRAM (note that this includes the bias rows and each ReRAM array corresponds to the VMM from one layer to another layer and therefore there are 4 ReRAM arrays). The python wrapper generates the netlist of the neuromorphic circuit using Verilog-A models of ReRAM weights, DAC, Op-Amp, and ReLU circuit blocks. Here only Verilog-A models are used. The effect of Op-Amp design using SPICE models will be discussed in the following sections. The open-loop gain, A_{OL}, of the op-amps is set to 80dB. The resistances of the feedback network in the current subtractor (Fig. 2), R_{in}, R_1, and R_2 were set to 100Ω, 1kΩ, and 100kΩ respectively. The neuromorphic circuit and the software-NN were then both tested on the remaining 180 images. The accuracy of the neuromorphic circuit and the software-NN are both 96.67%. The MLPClassifier in Scikit-Learn (version 0.20.3) is used with default settings (e.g. adam solver is used with the regularization parameter of 0.0001, shuffle=True, and batch_size=auto, which is 200 in this study) [36].

The output of the current subtractor in Fig. 2 is given by the equation:

$$V_{sub} = \frac{(I_{col+} - I_{col-})\frac{R_{in}R_2}{R_1}}{1 + \frac{R_2+R_1}{R_1 A_{OL}}} \tag{1}$$

here, I_{col+} and I_{col-} are the currents flowing through the positive and negative columns (Fig. 1). Ideally, with a very large open-loop gain, the output is given by:

$$V_{sub(ideal)} = (I_{col+} - I_{col-})\frac{R_{in}R_2}{R_1} \tag{2}$$

Therefore, the current subtractor also acts as a current-to-voltage converter with a ratio of $\frac{R_{in}R_2}{R_1}$ and, thus, $1 + \frac{R_2+R_1}{R_1 A_{OL}}$ is the non-ideality factor. Based on the non-ideal equation (Eq. (1)), it is expected that for a fixed R_2, as R_1 increases, the non-ideality factor and gain error will decrease. However, if R_1 is too large, the closed-loop gain would be too small to amplify the currents to useable voltage levels and causes errors in the computation. Therefore, to keep R_1 large enough for high close-loop gain while reducing the gain error, R_{in} can be increased as long as R_{in} is still much less than $R_2 + R_1$ and the resistances in the crossbar. Therefore, this is a non-trivial optimization problem. Fig. 5 shows that the trade-off between R_{in} and R_1 is not trivial (for $R_2 = 100$k Ω). Based on the area requirement, resistance accuracy, and inference accuracy requirement, one may choose different R_{in} and R_1 pairs for the application. For example, $R_{in}/R_1 = 10\Omega/100\Omega$ gives the highest accuracy of 97.22%. But one will choose $R_{in}/R_1 = 1\Omega/0.1\Omega$ (96.67% inference accuracy) to get the smallest layout area but might have the worse process variation.

The framework also enables the plotting of the voltage, current, resistance, power, and gap size of the ReRAMs for visualization and debugging. Fig. 6 shows the potential distribution in various layers (layer 0 to layer 3) of an example (hand-written digit "3") and the input potential. It can be seen that the potential is very uniform across the rows because of the small potential drop across the horizontal lines. The difference in the currents flowing through the ReRAMs in adjacent columns represents a multiplication operation on the input voltage and the conductance of the ReRAM. The total current flowing in a column is the sum of all these products and represents the accumulation operation. These form the basis of the Multiply-and-Accumulate operation needed in ANNs. In Fig. 7, the total column current in the last layer is maximum in column 3. The neuromorphic circuit correctly predicts the label (3) of the input shown in Fig. 6. Note that the current plot has half the width of other plots because

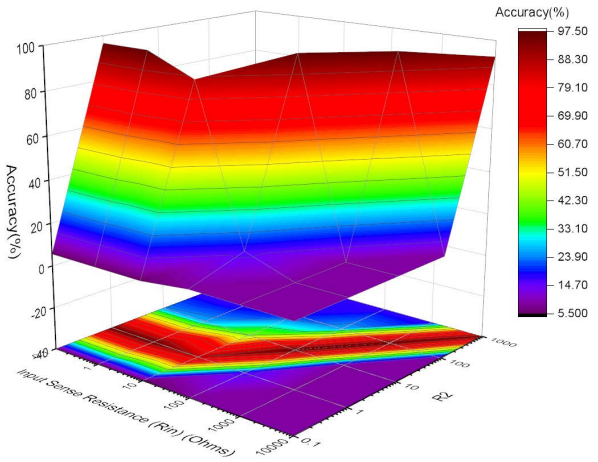

FIGURE 5. Training accuracy as a function of R_{in} and R_1 in the current comparator in Fig. 3. This is for the [50,20,8] NN for the UCI hand-written test set with $R_2 = 100k\Omega$.

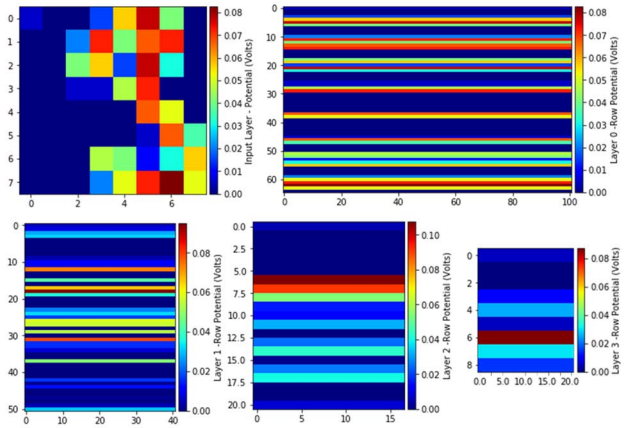

FIGURE 6. Top left: Example of an image (hand-written digit "3") encoded with potential. Others: Potential maps of the 4 ReRAM arrays in the [50,20,8] NN.

the difference in the currents of two adjacent branches is displayed. Fig. 8 and Fig. 9 plot the ReRAM resistance maps and power consumption maps. The total power consumption is only 69μW.

By enabling parasitic simulation in the simulation options of the Jupyter wrapper, parasitic components can be added to the netlist as well as the post-layout extractions of the subcircuits. This is useful for measuring the speed of the neuromorphic circuit with transient simulations.

The ReRAM may be formed between the silicide poly and M1 (poly/M1) or M1 and M2 (M1/M2). This framework allows users to incorporate the parasitic resistance and capacitance of the wires in the simulation (Fig. 3) to study its transient response.

Table 1 show the extracted parasitic capacitance and resistance when ReRAM is formed at the cross-points of poly/M1 or M1/M2. Minimum spacing and line width of the 45nm technology are used.

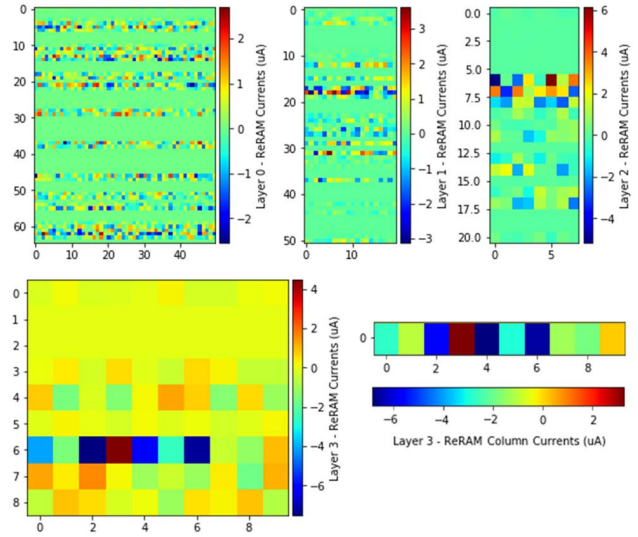

FIGURE 7. ReRAM current maps of various arrays in the [50,20,8] NN. Note that they display the current difference in each ReRAM pair. The bottom shows the output layer total $I_{col+} - I_{col-}$ for each column. "3" has the largest current and thus the NN recognizes the digit "3" correctly.

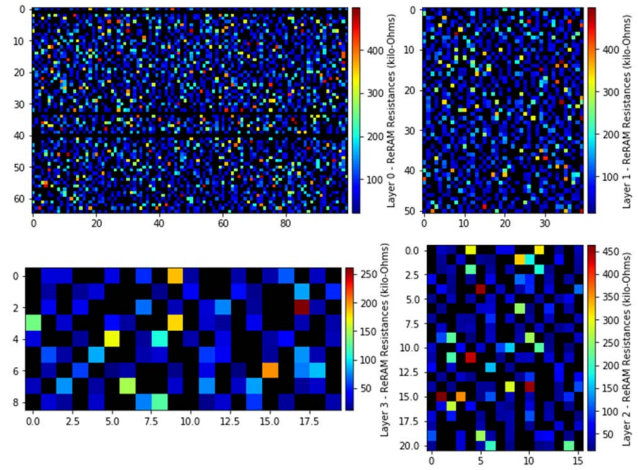

FIGURE 8. Resistance maps of various arrays in the [50,20,8] NN.

TABLE 1. Extracted parasitic capacitance and resistance of the generic 45nm technology.

Top /Bottom layers	Bottom Layer Cap(fF)	Bottom Layer Res (mΩ)	Top Layer Cap (fF)	Top Layer Res (mΩ)
M1/M2	0.008	404.8	0.0075	234.1
Poly/M1	0.035	103333.3	0.0019	128.8

III. THE EFFECT OF DAC AND VOLTAGE RANGE

In reality, the input data to the neuromorphic circuit in an IoT application can be analog or digital. Here it is assumed that the input signal has been digitized (e.g. the digitized camera image). Depending on the accuracy and power requirements, it can be digitized to different numbers of binary digits. Therefore, a DAC is needed to convert the digital signal to the analog voltage for the neuromorphic circuit (Fig. 1). The UCI images are used in this study. The pixel values in the UCI data

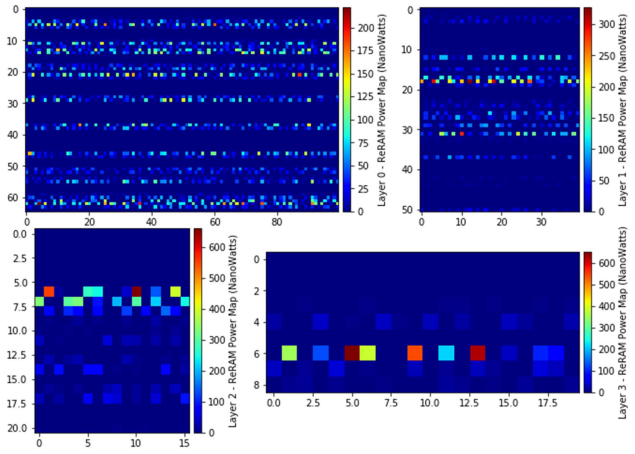

FIGURE 9. Power dissipation maps of various arrays in the [50,20,8] NN with total power dissipation = 69μW.

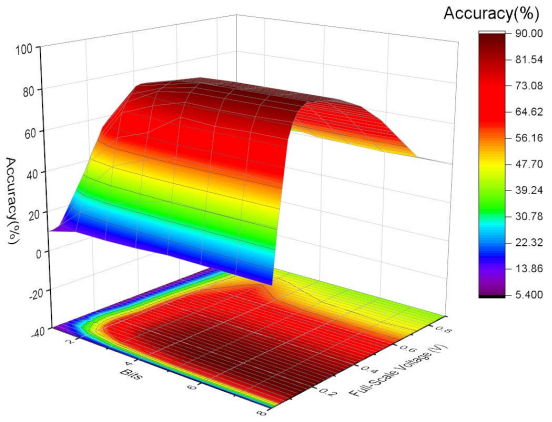

FIGURE 10. NN inference accuracy vs DAC resolution and scaling voltage for a [8,8,8] neuromorphic circuit. Top: 3D view. Bottom: Top View.

set are in the range of $p = 0$ to 16 and can be represented by 5 bits. To emulate images digitized with a different number of bits, M, the following equation is used to transform the pixel values, p, in the testing data set.

$$V_i = V_s Int\left(p\frac{2^M - 1}{2^N - 1}\right)/(2^N - 1) \tag{3}$$

where V_i is the input voltage to the neuromorphic circuits (e.g. $i = 0$ to 63 in Fig. 1), $N = 5$, and V_s is the scaling voltage. $Int()$ is a round-off-to-integer function. Note that the training process still uses the original UCI data (i.e. 5 bits).

We then study how M and V_s affect the inference accuracy. Two NN are tested, namely [50,20,8] and [8,8,8]. Fig. 10 and Fig. 11 show the surface plots of the prediction accuracy of these two NN as a function of M and V_s.

The accuracy of the [50,20,8] and [8,8,8] software neural networks are 96.67% and 90% respectively. As expected, for a reasonable V_s, as the DAC resolution is increased, the neuromorphic circuit accuracy increases until it reaches the software-neural network accuracy as the limit, in general.

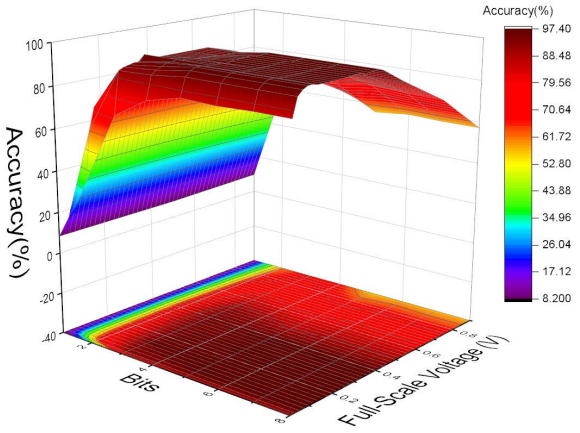

FIGURE 11. NN inference accuracy vs DAC resolution and scaling voltage for a [50,20,8] neuromorphic circuit. Top: 3D view. Bottom: Top View.

However, note that for the [50,20,8] NN, the hardware accuracy when $V_s = 0.1V$ and $M \geq 5$ is 97.22% and is *higher* than the software accuracy. This shows the non-trivialness in neuromorphic circuit optimization.

It can also be seen that the larger NN is more robust than the smaller NN. Firstly, the smaller NN [8,8,8] only has high accuracy (within 10% of the peak accuracy) from $V_s = 0.1V$ to $V_s = 0.4V$ while the larger one [50,20,8] has high accuracy from $V_s = 0.04V$ to $V_s = 0.48V$. Moreover, there is a wide range of V_s (also $0.04V$ to $0.48V$) in which the accuracy is high even with $M = 3$ for the large NN but this happens only for $V_s = 0.16V$ in the small NN.

IV. EFFECT OF THE OPAMP DESIGN

As shown in Fig. 1, the OpAmp plays an important role in the neuromorphic circuit. It is an essential part of the current comparator. It also acts as the buffer for the rectification unit (see [30]). Therefore, it is important to study its effect on inference accuracy.

Table 2 shows the effect of the OpAmp open-loop gain (A_{OL}) on the inference accuracy compared to the software one. R_2 is set to 100kΩ and $R_1/R_{in} = 10$. It is found that an open-loop gain of 80dB is required to attain software accuracy. When R_{in} is small (e.g. 10Ω), as discussed earlier, there is a requirement of high A_{OL} so that the gain error in Eq. (1) can be reduced. For example, at $A_{OL} = 60$dB and $R_{in} = 10\Omega$, the inference accuracy is reduced substantially by 47%. Therefore, it might be desirable to use $R_1/R_{in} = 50/500$ by using unsilicided poly resistance to reduce the design requirement of the OpAmp.

A two-stage amplifier using folded cascode followed by a common source amplifier with a layout area of about $3\mu m^2$ is designed using the generic 45nm PDK. Fig. 12 shows the schematic with $A_{OL} = 68$dB. Fig. 13 shows it is stable with the feedback. It is found that the stability increases R_2/R_1 increases. This is expected because the feedback factor is

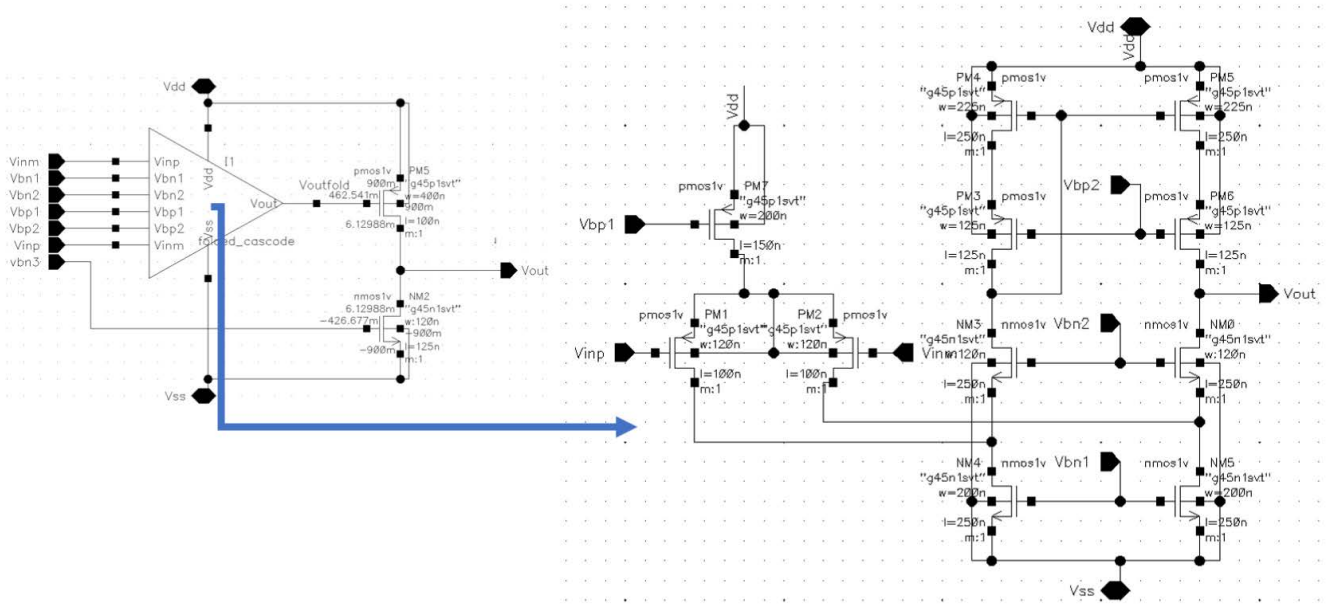

FIGURE 12. A two-stage amplifier with A_{OL} = 68dB.

TABLE 2. Effect of OpAMP open loop gain on inference accuracy as function of R_{IN}/R_1. The First number is the accuracy compared to software and the second number is A_{OL}.

R_{in} / R_1 / A_{OL}	10Ω/100Ω	50Ω/500Ω	100Ω/1000Ω
60dB/66dB	-47%, 60dB	-2%, 66dB	-2%, 60dB
80dB	+2%,	0%	0%

R_1/R_2. Therefore, the system is more stable when R_1/R_2 is smaller.

V. SOFTWARE CIRCUIT CO-OPTIMIZATION

An optimal NN is not necessarily the one that gives the highest accuracy. Particularly, in IoT applications, the NN size, power consumption, and circuit areas need to be considered, to minimize the power consumption and cost. This cannot be studied by software ML alone.

As an example, we consider the application of 1-hidden layer NN for UCI and MNIST [37] handwritten datasets. MNIST is a larger database of handwritten digits (70000 images) and the neural networks are trained and tested with 60000 and 10000 images, respectively. Moreover, each MNIST has 28 × 28 pixels with pixel values between 0 and 255.

We study the change of inference accuracy as a function of the number of nodes in the hidden layer. The node number changes from 100 to 13. For the 100-node case, the MNIST circuit uses about 160,000 ReRAM. With 30 CPU cores, the inference simulations are completed within about 70 hours.

Fig. 14 shows that by using software ML, one cannot predict the trend and actual performance precisely when it

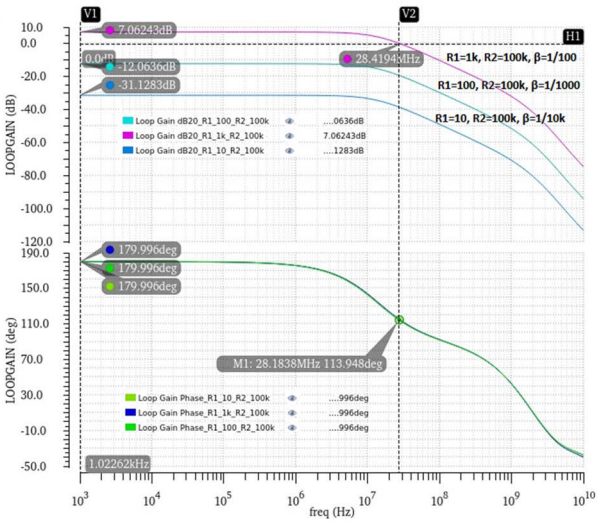

FIGURE 13. Loop gain properties of the amplifier in Fig. 12 with the feedback circuit in Fig. 3.

is applied to the neuromorphic circuit. For example, as the number of nodes decreases in the UCI case, the software ML predicts that the accuracy will drop rapidly when the number of nodes is reduced from 100 to 50. However, the actual hardware accuracy does not change. This results in a larger design space than that predicted by the software.

On the other hand, the MNIST study shows that even with 13 nodes, the software ML can still achieve >90% accuracy (equivalent to almost a 10X reduction in the array size and number of the current comparator). However, the accuracy is not acceptable (only 75%) when it is implemented in the neuromorphic circuit. Therefore, it is very important to

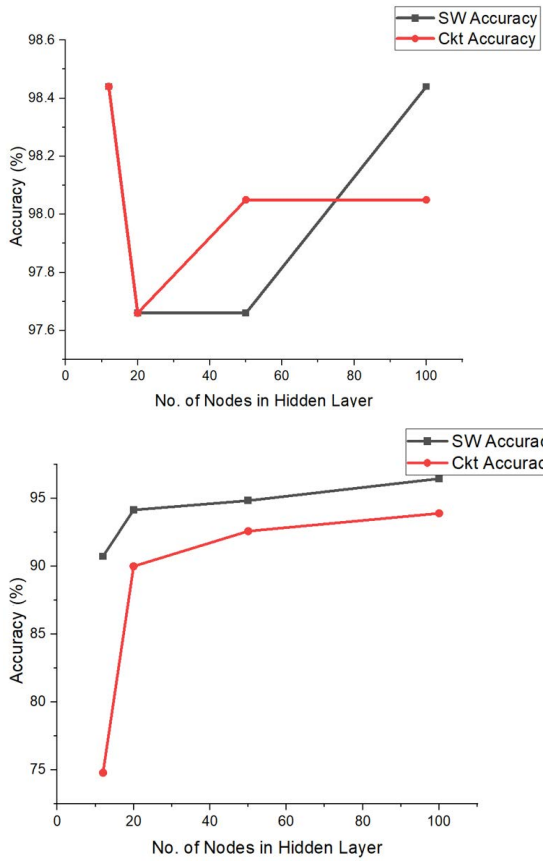

FIGURE 14. Inference accuracy vs hidden layer size in software ML and neuromorphic hardware circuit. Top: UCI dataset. Bottom: MNIST dataset.

co-optimize the software architecture and the neuromorphic circuit.

The UCI inference accuracy increases again when the NN only has 13 hidden nodes. This shows that NN is too complicated to understand (known as a black box) and often gives unexpected results. Therefore, it is very important to perform a full SPICE circuit simulation to find the optimal setup.

VI. CONCLUSION

A Software-Circuit-Device co-optimization framework for neural network inference is developed and presented. This framework allows users to perform software machine learning and co-optimize with the corresponding neuromorphic circuit through SPICE simulations. It takes Verilog-A or SPICE models from the PDK without the need for additional calibration to behavior models.

It is shown that this framework is particularly useful for IoT edge inference device which has stronger requirements on power and cost and where fully analog circuits are desired. Fully analog neuromorphic circuits using ReRAM have been simulated as an example. In addition, the framework can handle MNIST data and perform inference accuracy simulation in a reasonable time even with limited computation resources. It is demonstrated that the co-design of software NN architecture, DAC, OpAmp, and its feedback network are important to optimize the neuromorphic circuit in a 45nm technology.

VII. LIMITATIONS AND FUTURE WORK

This framework requires accurate SPICE or Verilog-A models of the emerging memories. If they are not available, a Verilog-A model needs to be developed based on the behavior model, if available. Note that the frame supports stochastic simulation. For example, the initial gap size can be randomized as a parameter to the subcircuit of each ReRAM. In this demonstration, the ReRAM weights are not quantized. However, this can be done easily by digitizing the gap size for each ReRAM before calling the subcircuit.

In the current framework, modular crossbar tiles and ReRAM selectors are not used. Moreover, only fully-connected neural networks with ReLU activation functions have been tested. However, since the activation function is implemented as a subcircuit, other activation functions can be quickly added to the framework as long as the model (Verilog-A or SPICE) is available. The current version of the framework also does not support convolutional layers. There are some methods to implement the convolution operation as a vector-matrix product that is compatible with the ReRAM array such as in [38] and the framework can be modified accordingly.

In this framework, we assumed the circuit is fully analog but noise is not considered. The impact of noise due to having fully analog communication between layers should be studied to ensure the circuit works in non-ideal environments.

DATA AVAILABILITY

The framework and data that support the findings of this study are available from the corresponding author upon reasonable request. The coding of the critical parts of the framework is explained and can be found in [39], [40].

REFERENCES

[1] A. C. Müller and S. Guido, *Introduction to Machine Learning With Python*, 1st ed. Boston, MA, USA: O'Reilly Media, 2016.

[2] 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, doi: 10.1109/5.726791.

[3] E. García-Martín, C. F. Rodrigues, G. Riley, and H. Grahn, "Estimation of energy consumption in machine learning," *J. Parallel Distrib. Comput.*, vol. 134, pp. 75–88, Dec. 2019, doi: 10.1016/j.jpdc.2019.07.007.

[4] D. Ielmini and H. S. P. Wong, "In-memory computing with resistive switching devices," *Nature Electron.*, vol. 1, no. 6, pp. 333–343, 2018, doi: 10.1038/s41928-018-0092-2.

[5] H. Tsai, S. Ambrogio, P. Narayanan, R. M. Shelby, and G. W. Burr, "Recent progress in analog memory-based accelerators for deep learning," *J. Phys. D, Appl. Phys.*, vol. 51, no. 28, Jul. 2018, Art. no. 283001.

[6] M. Hu, J. P. Strachan, Z. Li, E. M. Grafals, N. Davila, C. Graves, S. Lam, N. Ge, J. J. Yang, and R. S. Williams, "Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication," in *Proc. 53nd ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Austin, TX, USA, Jun. 2016, pp. 1–6, doi: 10.1145/2897937.2898010.

[7] Y.-F. Chang, F. Zhou, B. W. Fowler, Y.-C. Chen, C.-C. Hsieh, L. Guckert, E. E. Swartzlander, and J. C. Lee, "Memcomputing (memristor + computing) in intrinsic SiO_x-based resistive switching memory: Arithmetic operations for logic applications," *IEEE Trans. Electron Devices*, vol. 64, no. 7, pp. 2977–2983, Jul. 2017, doi: 10.1109/TED.2017.2699679.

[8] S. Yu, B. Gao, Z. Fang, H. Yu, J. Kang, and H.-S.-P. Wong, "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, Mar. 2013, doi: 10.1002/adma.201203680.
[9] H.-S. P. Wong, H.-Y. Lee, S. Yu, Y.-S. Chen, Y. Wu, P.-S. Chen, B. Lee, F. T. Chen, and M.-J. Tsai, "Metal-oxide RRAM," *Proc. IEEE*, vol. 100, no. 6, pp. 1951–1970, Jun. 2012, doi: 10.1109/JPROC.2012.2190369.
[10] X. Sun, P. Wang, K. Ni, S. Datta, and S. Yu, "Exploiting hybrid precision for training and inference: A 2T-1FeFET based analog synaptic weight cell," in *IEDM Tech. Dig.*, Dec. 2018, pp. 3.1.1–3.1.4, doi: 10.1109/IEDM.2018.8614611.
[11] Z. He, S. Angizi, and D. Fan, "Exploring STT-MRAM based in-memory computing paradigm with application of image edge extraction," in *Proc. IEEE Int. Conf. Comput. Design (ICCD)*, Nov. 2017, pp. 439–446, doi: 10.1109/ICCD.2017.78.
[12] S. Kim, M. Ishii, S. Lewis, T. Perri, M. BrightSky, W. Kim, R. Jordan, G. W. Burr, N. Sosa, A. Ray, J.-P. Han, C. Miller, K. Hosokawa, and C. Lam, "NVM neuromorphic core with 64k-cell (256-by-256) phase change memory synaptic array with on-chip neuron circuits for continuous *in-situ* learning," in *IEDM Tech. Dig.*, Dec. 2015, p. 17, doi: 10.1109/IEDM.2015.7409716.
[13] J. Tang, D. Bishop, S. Kim, M. Copel, T. Gokmen, T. Todorov, S. Shin, K.-T. Lee, P. Solomon, K. Chan, W. Haensch, and J. Rozen, "ECRAM as scalable synaptic cell for high-speed, low-power neuromorphic computing," in *IEDM Tech. Dig.*, Dec. 2018, p. 13, doi: 10.1109/IEDM.2018.8614551.
[14] P. Gu, B. Li, T. Tang, S. Yu, Y. Cao, Y. Wang, and H. Yang, "Technological exploration of RRAM crossbar array for matrix-vector multiplication," in *Proc. 20th Asia South Pacific Design Autom. Conf.*, Jan. 2015, pp. 106–111, doi: 10.1109/ASPDAC.2015.7058989.
[15] M. Hu, H. Li, Q. Wu, and G. S. Rose, "Hardware realization of BSB recall function using memristor crossbar arrays," in *Proc. 49th Annu. Design Autom. Conf. (DAC)*, Jun. 2012, pp. 498–503.
[16] Y.-F. Kao, W. C. Zhuang, C.-J. Lin, and Y.-C. King, "A study of the variability in contact resistive random access memory by stochastic vacancy model," *Nanosc. Res. Lett.*, vol. 13, no. 1, p. 213, Dec. 2018, doi: 10.1186/s11671-018-2619-x.
[17] M. Ueki, K. Takeuchi, T. Yamamoto, A. Tanabe, N. Ikarashi, M. Saitoh, T. Nagumo, H. Sunamura, M. Narihiro, K. Uejima, K. Masuzaki, N. Furutake, S. Saito, Y. Yabe, A. Mitsuiki, K. Takeda, T. Hase, and Y. Hayashi, "Low-power embedded ReRAM technology for IoT applications," in *Proc. Symp. VLSI Technol. (VLSI Technol.)*, Jun. 2015, pp. T108–T109, doi: 10.1109/VLSIT.2015.7223640.
[18] A. Baumann, M. Jung, K. Huber, M. Arnold, C. Sichert, S. Schauer, and R. Brederlow, "A MCU platform with embedded FRAM achieving 350nA current consumption in real-time clock mode with full state retention and 6.5μs system wakeup time," in *Proc. Symp. VLSI Circuits*, Jun. 2013, pp. C202–C203.
[19] P.-Y. Chen, X. Peng, and S. Yu, "NeuroSim+: An integrated device-to-algorithm framework for benchmarking synaptic devices and array architectures," in *IEDM Tech. Dig.*, Dec. 2017, pp. 6.1.1–6.1.4, doi: 10.1109/IEDM.2017.8268337.
[20] C. Xu, X. Dong, N. P. Jouppi, and Y. Xie, "Design implications of memristor-based RRAM cross-point structures," in *Proc. Design, Autom. Test Eur.*, Mar. 2011, pp. 1–6.
[21] D. Niu, C. Xu, N. Muralimanohar, N. P. Jouppi, and Y. Xie, "Design trade-offs for high density cross-point resistive memory," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design (ISLPED)*, 2012, pp. 209–214.
[22] C. Xu, D. Niu, N. Muralimanohar, R. Balasubramonian, T. Zhang, S. Yu, and Y. Xie, "Overcoming the challenges of crossbar resistive memory architectures," in *Proc. IEEE 21st Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2015, pp. 476–488.
[23] C. Lammie, J. K. Eshraghian, C. Li, A. Amirsoleimani, R. Genov, W. D. Lu, and M. R. Azghadi, "Design space exploration of dense and sparse mapping schemes for RRAM architectures," 2022, *arXiv:2201.06703*.
[24] D. M. Mathew, A. L. Chinazzo, C. Weis, M. Jung, B. Giraud, P. Vivet, A. Levisse, and N. Wehn, "RRAMSpec: A design space exploration framework for high density resistive RAM," in *Embedded Computer Systems: Architectures, Modeling, and Simulation*, D. N. Pnevmatikatos, M. Pelcat, and M. Jung, Eds. Cham, Switzerland: Springer, 2019, pp. 34–47.
[25] M. J. Rasch, D. Moreda, T. Gokmen, M. Le Gallo, F. Carta, C. Goldberg, K. El Maghraoui, A. Sebastian, and V. Narayanan, "A flexible and fast PyTorch toolkit for simulating training and inference on analog crossbar arrays," in *Proc. IEEE 3rd Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Jun. 2021, pp. 1–4, doi: 10.1109/aicas51828.2021.9458494.
[26] C. Lammie, W. Xiang, B. Linares-Barranco, and M. Rahimi Azghadi, "MemTorch: An open-source simulation framework for memristive deep learning systems," 2020, *arXiv:2004.10971*.
[27] Y. Demirag, C. Frenkel, M. Payvand, and G. Indiveri, "Online training of spiking recurrent neural networks with phase-change memory synapses," 2021, *arXiv:2108.01804*.
[28] C. Lammie, W. Xiang, and M. Rahimi Azghadi, "Modeling and simulating in-memory memristive deep learning systems: An overview of current efforts," *Array*, vol. 13, Mar. 2022, Art. no. 100116, doi: 10.1016/j.array.2021.100116.
[29] H. Cao, T. Lam, H. Nguyen, A. Venkattraman, D. Parent, and H. Y. Wong, "Study of ReRAM neuromorphic circuit inference accuracy robustness using DTCO simulation framework," in *Proc. IEEE Workshop Microelectron. Electron Devices (WMED)*, Apr. 2021, pp. 1–4, doi: 10.1109/WMED49473.2021.9425210.
[30] A. Nguyen, H. Nguyen, S. Venimadhavan, A. Venkattraman, D. Parent, and H. Y. Wong, "Fully analog ReRAM neuromorphic circuit optimization using DTCO simulation framework," in *Proc. Int. Conf. Simulation Semiconductor Processes Devices (SISPAD)*, Sep. 2020, pp. 201–204, doi: 10.23919/SISPAD49475.2020.9241635.
[31] P. Quibuyen, T. Jiao, and H. Y. Wong, "2022 IEEE 4th international conference on artificial intelligence circuits and systems (AICAS)," in *Proc. IEEE 4th Int. Conf. Artif. Intell. Circuits Syst. (AICAS)*, Jan. 2022, pp. 1–5.
[32] X. Peng, R. Liu, and S. Yu, "Optimizing weight mapping and data flow for convolutional neural networks on RRAM based processing-in-memory architecture," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–5, doi: 10.1109/ISCAS.2019.8702715.
[33] Z. Jiang and H. P. Wong, *Stanford University Resistive-Switching Random Access Memory (RRAM) Verilog—A Model*. Lafayette, IN, USA: nanoHUB, 2014, doi: 10.4231/D37H1DN48.
[34] *Spectre Simulation Platform*. Accessed: Dec. 16, 2021. [Online]. Available: https://www.cadence.com/en_US/home/tools/custom-ic-analog-rf-design/circuit-simulation/spectre-simulation-platform.html
[35] D. Dua and C. Graff, *UCI Machine Learning Repository*. Irvine, CA, USA: Univ. California, 2019. [Online]. Available: http://archive.ics.uci.edu/ml
[36] *Sklearn.Neural_Network.MLPClassifier—Scikit-Learn 1.0.2 Documentation*. Accessed: Dec. 16, 2021. [Online]. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html
[37] *THE MNIST DATABASE of Handwritten Digits*. Yann LeCun, Courant Institute, NYU Corinna Cortes, Google Labs, New York Christopher J. C. Burges, Microsoft Research, Redmond. Accessed: Jan. 15, 2020. [Online]. Available: https://yann.lecun.com/exdb/mnist/
[38] S. Ko, Y. J. Soh, and J. Zhao, "Efficient implementation of multi-channel convolution in monolithic 3D ReRAM crossbar," 2020, *arXiv:2004.00243*.
[39] A. Nguyen, "Study of ReRAM neuromorphic circuit performance using DTCO simulation framework," M.S. thesis, Elect. Eng., San Jose State Univ., San Jose, CA, USA, 2021.
[40] P. Quibuyen, "Study and design of analog circuit blocks for neuromorphic computing using DTCO simulation framework," M.S. thesis, Elect. Eng., San Jose State Univ., San Jose, CA, USA, 2022.