LUT-based RRAM Model for Neural Accelerator Circuit Simulation

Max Uhlmann[*]
Tommaso Rizzi[*]
uhlmann,rizzi@ihp-microelectronics.com
IHP - Leibniz-Institut für innovative Mikroelektronik
Frankfurt (Oder), Germany

Jianan Wen
IHP - Leibniz-Institut für innovative Mikroelektronik
Frankfurt (Oder), Germany

Emilio Pérez-Bosch Quesada
IHP - Leibniz-Institut für innovative Mikroelektronik
Frankfurt (Oder), Germany

Bakr Al Beattie
Ruhr University Bochum
Bochum, Germany

Karlheinz Ochs
Ruhr University Bochum
Bochum, Germany

Eduardo Pérez
IHP - Leibniz-Institut für innovative Mikroelektronik, BTU Cottbus-Senftenberg
Frankfurt (Oder), Cottbus, Germany

Philip Ostrovskyy
IHP - Leibniz-Institut für innovative Mikroelektronik
Frankfurt (Oder), Germany

Corrado Carta
IHP - Leibniz-Institut für innovative Mikroelektronik, TU Berlin
Frankfurt (Oder), Berlin, Germany

Christian Wenger
IHP - Leibniz-Institut für innovative Mikroelektronik, BTU Cottbus-Senftenberg
Frankfurt (Oder), Cottbus, Germany

Gerhard Kahmen
IHP - Leibniz-Institut für innovative Mikroelektronik, BTU Cottbus-Senftenberg
Frankfurt (Oder), Cottbus, Germany

ABSTRACT

Neural hardware accelerators have been proven to be energy-efficient when used to solve tasks which can be mapped into an artificial neural network (ANN) structure. Resistive random-access memories (RRAMs) are currently under investigation together with several different memristive devices as promising technologies to build such accelerators combined together with complementary metal-oxide semiconductor (CMOS)-technologies in integrated circuits (ICs). While many research groups are actively developing sophisticated physical-based representations to better understand the underlying phenomena characterizing these devices, not much work has been dedicated to exploit the trade-off between simulation time and accuracy in the definition of low computational demanding models suitable to be used at many abstraction layers. Indeed, the design of complex mixed-signal systems as a neural hardware accelerators requires frequent interaction between the application- and the circuit-level that can be enabled only with the support of accurate and fast-simulating devices' models. In this work, we propose a solution to fill the aforementioned gap with a lookup table (LUT)-based *Verilog-A* model of IHP's 1-transistor-1-RRAM (1T1R) cell. In addition, the implementation challenges of conveying the communication between the abstract ANN simulation and the circuital analysis are tackled with a design flow for resistive neural hardware accelerators that features a custom *Python* wrapper. As a demonstration of the proposed design flow and 1T1R model, an ANN for the MNIST handwritten digit recognition task is assessed with the last layer verified in circuit simulation. The obtained recognition confidence intervals show a considerable discrepancy between the purely application-level *PyTorch* simulation and the proposed design flow which spans across the abstraction layers down to the circuital analysis.

[*]Both authors contributed equally to this research.

NANOARCH '23, December 18–20, 2023, Dresden, Germany
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 979-8-4007-0325-6/23/12…$15.00
https://doi.org/10.1145/3611315.3633273

CCS CONCEPTS

• **Hardware** → *Emerging tools and methodologies*; Memory and dense storage.

KEYWORDS

Neural Accelerator, Memristor, RRAM, Artificial Neural Network, Lookup Table

ACM Reference Format:
Max Uhlmann, Tommaso Rizzi, Jianan Wen, Emilio Pérez-Bosch Quesada, Bakr Al Beattie, Karlheinz Ochs, Eduardo Pérez, Philip Ostrovskyy, Corrado Carta, Christian Wenger, and Gerhard Kahmen. 2023. LUT-based RRAM Model for Neural Accelerator Circuit Simulation. In *18th ACM International Symposium on Nanoscale Architectures (NANOARCH '23), December 18–20, 2023, Dresden, Germany*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3611315.3633273

1 INTRODUCTION

Building and deploying ANNs, and in particular deep neural networks (DNNs), in integrated circuits (ICs) for machine learning (ML) tasks has been a complex challenge for both: system and circuit designers over the past several years [2, 12]. A recent direction is the adoption of neural hardware accelerators to increase the efficiency of the most common operations as the vector-matrix multiplication (VMM). Among the different implementations for such accelerators, one possible solution to lower the power consumption and increase the operational speed consists of encoding the synaptic weights in the conductance values stored in a memory cell composed by a NMOS transistor and a RRAM in series (1T1R) [13, 16]. This allows to perform VMM operations efficiently in an analog fashion by multiplying a voltage input vector via Ohm's law with the conductive states of the RRAM devices and adding up the currents at the source contacts of the 1T1R cells by Kirchoff's law. Although promising, the RRAM technology is still in an emerging phase with active research innovating in the production steps.

To understand the underlying switching phenomena and how they impact the device's response, different component descriptions have been developed ranging from atomistic models [4], to physical-based compact representations [10], up to behavioral descriptions [9, 14]. While there is a clear trade-off between accuracy and acceptable simulation time that drives the choice of the model coherently with the specific use case, the recent progress in the simulation frameworks and the increase in the available computational resources have allowed the adoption of complex mathematical descriptions of the device even at the application level. With the tool presented in [8], for instance, the user can train the neural network while considering the full switching behaviour of the RRAM's conductance. *MemTorch* [5] is another framework developed in *Python* for the functional assessment of resistive neural hardware accelerators. It allows the user to select between different memristive devices' models, some of which include also computational demanding differential equations.

These new possibilities have been extremely beneficial at the system level, increasing the accuracy estimation since the early design phase and improving the overall performance thanks to Fault-Aware Training (FAT) and Quantization-Aware Training (QAT) techniques [3, 17]. Nevertheless, the great focus in the last years to tools at high-level of abstraction have left neglected an important part of the design flow required to develop neural hardware accelerators, namely the circuit design. Indeed, the development of mixed-signal systems still relies on professional tools such as *Design Compiler* (DS) by *Synopsys* and *Spectre* by *Cadence* for the implementation of complex logic circuits and analog components. When these established and reliable, although generally associated with a high computational cost, design tools are required, the aforementioned

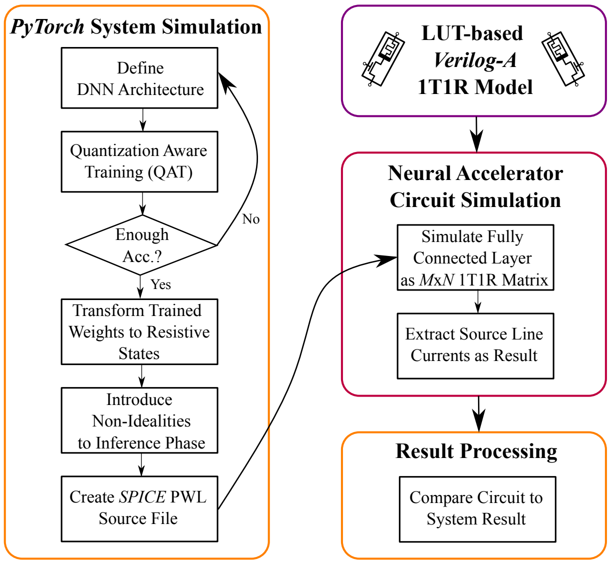
PyTorch System Simulation
Define DNN Architecture
Quantization Aware Training (QAT)
Enough Acc.?
No
Yes
Transform Trained Weights to Resistive States
Introduce Non-Idealities to Inference Phase
Create *SPICE* PWL Source File
LUT-based *Verilog-A* 1T1R Model
Neural Accelerator Circuit Simulation
Simulate Fully Connected Layer as *M*x*N* 1T1R Matrix
Extract Source Line Currents as Result
Result Processing
Compare Circuit to System Result

Figure 1: Overview of the design flow of a resistive neural hardware accelerator from system to circuit level, using the LUT-based *Verilog-A* model of a 1T1R cell.

trade-off between accuracy and simulation speed returns to be critical. Including thousands of RRAM devices in the netlist described by *Verilog-A* model involving differential equations with often bounding constraints on the timestep, can quickly become an unbearable burden. The principal aim of this work is to introduce a model for the 1T1R cell which can depict the switching behaviour of the RRAM devices, while avoiding computational demanding equations. Thus, it is suitable to be used from the application to the circuit level in a top-down design flow of resistive neural hardware accelerators (see Fig. 1). To verify the functionalities of such mode at the circuit level, a custom *Python* wrapper is developed around the ANN *PyTorch* simulation. It generates the *Spectre* stimuli files required to program the 1T1R matrices and to perform inference operations.

This paper is structured as follows: The LUT-based model of IHP's RRAM device for circuit simulation is described and simulated in Section 2. Subsequently, Section 3 proposes the design flow for resistive neural hardware accelerators starting from a system level simulation of ANN architecture, extracting and mapping different parameters to the netlist, simulating the circuit with the developed 1T1R model and comparing the outcome with the achieved *PyTorch* result. Based on the shown design flow an use case is analyzed: the MNIST handwritten digit recognition [6] task is solved with a DNN architecture in Section 4. Section 5 presents conclusions and gives an outlook on how to expand the 1T1R circuit design model and the presented neural hardware accelerator design flow.

2 LUT-BASED RRAM MODEL

2.1 Model Description

To capture the multi-level switching behaviour of the RRAM device without the need of computational demanding equations, three

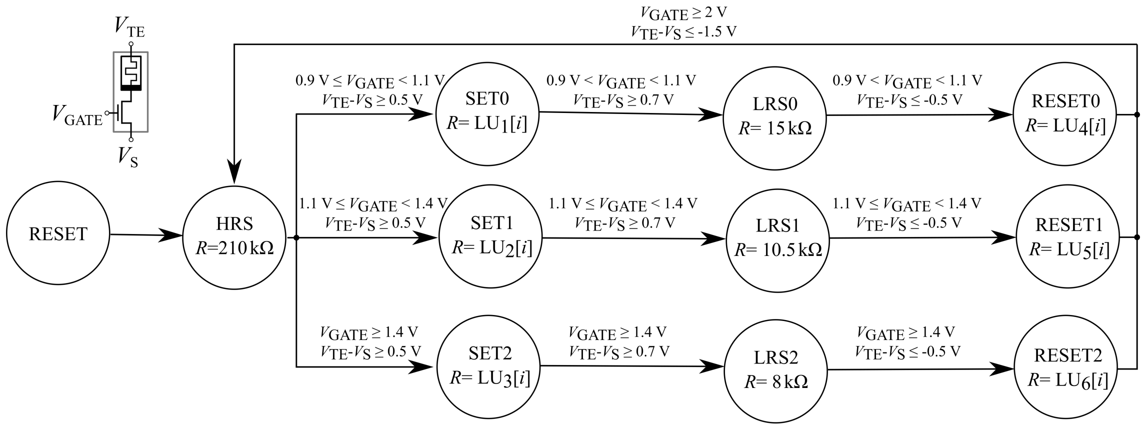
V_{TE}
V_{GATE}
V_S
RESET
HRS R=210 kΩ
$V_{GATE} \geq 2$ V
V_{TE}-$V_S \leq$ -1.5 V
0.9 V ≤ V_{GATE} < 1.1 V
V_{TE}-$V_S \geq$ 0.5 V
SET0 R= LU$_1$[i]
0.9 V < V_{GATE} < 1.1 V
V_{TE}-$V_S \geq$ 0.7 V
LRS0 R= 15 kΩ
0.9 V < V_{GATE} < 1.1 V
V_{TE}-$V_S \leq$ -0.5 V
RESET0 R= LU$_4$[i]
1.1 V ≤ V_{GATE} < 1.4 V
V_{TE}-$V_S \geq$ 0.5 V
SET1 R= LU$_2$[i]
1.1 V ≤ V_{GATE} < 1.4 V
V_{TE}-$V_S \geq$ 0.7 V
LRS1 R= 10.5 kΩ
1.1 V ≤ V_{GATE} < 1.4 V
V_{TE}-$V_S \leq$ -0.5 V
RESET1 R= LU$_5$[i]
$V_{GATE} \geq$ 1.4 V
V_{TE}-$V_S \geq$ 0.5 V
SET2 R= LU$_3$[i]
$V_{GATE} \geq$ 1.4 V
V_{TE}-$V_S \geq$ 0.7 V
LRS2 R= 8 kΩ
$V_{GATE} \geq$ 1.4 V
V_{TE}-$V_S \leq$ -0.5 V
RESET2 R= LU$_6$[i]

Figure 2: Finite state machine model of IHP's 1T1R cell with lookup tables implemented in *Verilog-A*.

assumptions are taken: i) the set/reset voltages required to switch the device are deterministic and fixed before launching the simulation; ii) the non-linear behavior is captured only in the transition between conductive states; and iii) setting the RRAM to a different Low Resistive State (LRS) requires to mandatory reset the device to the High Resistive State (HRS) first. While iii) can be a reasonable design choice depending on the selected programming algorithm, i) and ii) are simplifications of the complex memristive behaviour of the device. Nevertheless, they allow the 1T1R model to fully reproduce the set/reset cycles using only a Finite State Machine (FSM) and Look-up Tables (LUTs). The number of LUTs (M) is bound by the relation $M = 2 * N$, where N is the number of considered low conductance states. The model is calibrated on IHP's 1T1R structure composed by an RRAM device with a TiN/Al:HfO$_2$/Ti/TiN material stack in series with a 130 nm NMOS transistor [10]. A graphical representation of the three-terminals *Verilog-A* implementation of the model can be seen in Fig. 2. The model considers already formed devices, initiating the component to HRS. The state to be programmed is chosen depending on the applied gate voltage V_{GATE} to the transistor, while the voltage drop across the device ($V_{TE} - V_S$) is used as entry key to read the conductance values in the LUTs during switching operations. When $V_{TE} - V_S$ is equal or higher than $0.5\,V$, the cell is considered under programming, and its conductance values are derived from LUTs until the voltage across the device is higher than $0.7\,V$. Afterwards, the programming phase is considered concluded and the resistance is assumed fixed based on the selected LRS. This condition is preserved until the reset phase is started by applying a voltage drop across the device smaller than $-0.5\,V$. While resetting the 1T1R cell, the non-linear changes of the resistance value are again obtained by LUTs. Once $V_{TE} - V_S$ is equal or minor than $-1.5\,V$ with a V_{GATE} of at least $2\,V$, the reset procedure is considered concluded and the device conductance returns to the HRS value. In principle the model is capable of updating its resistive value every simulation step. However, a resistive slew rate parameter R_{Slew} is included to depict a realistic dynamic switching behavior. Analog RRAM devices are characterized by transient switching time at least in the nanosecond order of magnitude [18]. Therefore, for a resistance change of around 100 kΩ in 100 ps, a $R_{Slew} = 10\,\text{P}\Omega/s$ is set.

2.2 Model Simulation

To assess the model's capability of correctly reproducing the device behaviour, a quasi static DC simulation is performed. Fig. 3 shows the resulting IV-characteristics compared with the measured data for each conductive level. The voltage across the 1T1R cell is swept in the range [0 V:1 V] with different V_{GATE} depending on the selected LRS. Afterward, the devices are reset to HRS with a V_{GATE} equal to 2.7 V and a negative polarization across the device. As it can be observed, the model successfully reproduces the switching behaviour with a very accurate fit between −0.4 V and 0.4 V regardless of the programmed state. This is a crucial feature of any RRAM model because the devices are often operated in low voltage range to allow reading operations (eg. during inference) without applying significant stress that may cause undesired change in the resistance level [11, 15]. It should be noted that, as expected, the perfect fit with the measurement data is partially lost during the transition phases between states due to the LUTs simplifications. This is especially true for the setting of the LRSs where an abrupt behaviour is assumed. Nevertheless, the devices are operated in such voltage rage only seldom and for very limited time. Consequently, the presented model can be efficiently used in circuit simulations allowing to capture switching behaviour of the device and without the need of computational demanding mathematical operations. For instance, comparing the required CPU times for one transient simulation of a set-reset cycle for one 1T1R cell modelled with the proposed LUT-based representation or with complex mathematical models [10], the presented approach enables almost 15 × faster simulation when tested in *Spectre* tool.

3 NEURAL ACCELERATOR CO-DESIGN FLOW

Although an holistic simulation at the circuit level of the whole neural hardware accelerator could be unpractical, the validation of the circuit under realistic conditions is of paramount importance before fabrication for an accurate performance assessment. Consequently, designers focus their testing on the most critical sub systems, while abstracting the components which have been already verified. At the application level, for example, in case of a recognition task, the final verification cannot neglect a careful

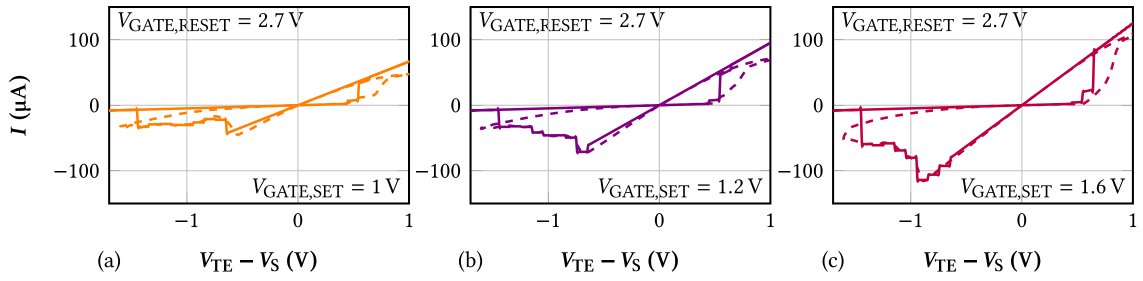

Figure 3: IV sweep comparison of model (solid line) and measurement (dashed line) results for SET and RESET operation of the 1T1R cell with different gate voltages applied.

Table 1: Artificial Neural Network Architecture for the MNIST Dataset implemented in *PyTorch*

Layer	*Type*	*Input Shape*	*Output Shape*	*Kernel Shape*	*Stride*	*Padding*	*Activation*
1	Conv2D	(28,28,1)	(24,24,10)	(5,5,10)	1	0	ReLU
2	MaxPooling2D	(24,24,10)	(12,12,10)	(2,2,10)	2	0	/
3	Conv2D*	(12,12,10)	(8,8,20)	(5,5,20)	1	0	ReLU
4	MaxPooling2D	(8,8,20)	(4,4,20)	(2,2,20)	2	0	/
5	Flatten	(4,4,20)	(320)	/	/	/	/
6	FullyConnected*	(320)	(50)	/	/	/	ReLU
7**	FullyConnected	(50)	(10)	/	/	/	LogSoftmax

*Dropout *p=0.5*

**Layer parameters extracted and simulated in *Cadence Spectre*

evaluation of the hardware implementation for the last fully connected (FC) layers which perform the decision-making step. This requirement is often hindered by the difficulties of translating input and output signals across abstraction layers, resulting in the adoption of simplified test-bench rather then real use case scenario. To overcome this challenge and to verify the proposed RRAM model under realistic conditions, we developed a *Python* wrapper that can generate custom stimuli files to reproduce the programming and inference operations in *Spectre* circuit simulations. The design flow spanning all abstraction levels is shown in Fig. 1. After defining the network architecture for the specific task, we apply specific FAT techniques for quantized network [3] to prepare the synaptic weights for the translation in discrete conductance values without hindering the system accuracy. This operation is performed in *Brevitas* [7], a tool developed on top of *PyTorch* by *Xilinx* that offers great customization to the user while remaining at the application level. When an acceptable accuracy is achieved, the quantized weights are mathematically converted in resistance states that can be stored in RRAM components. Based on this information and on the proposed RRAM model, our *Python* wrapper generates the input stimuli required to program the devices. Moreover, the wrapper can extract during inference operations the inputs values from any layers and convert them in voltage levels to apply to the matrices in the circuit simulation. The stimuli files are generated in the *Spectre* dialect of the SPICE language, thus they can be easily included in any netlist file. Therefore, if a coherent naming scheme is followed by the circuit designer for the pins nomenclature, the resulting waveforms are applied to the circuit allowing to reproduce the programming phase of the entire array, followed by inference steps useful to assess the hardware impact on the network accuracy at the circuit level. The obtained output currents from the circuit simulation can then be re-converted in digital values and compared to the results achieved in *PyTorch* at the application level. The design choice of developing the *Python* wrapper for the generation of the input stimuli rather than for the creation of the entire netlist is driven by the requirement of giving a high degree of freedom to the circuit designer in the definition of the sub-system for the circuital analysis. Moreover, although some changing may be necessary to tune the custom wrapper for the input signals of a different circuit, the implementation effort is still significantly lower compared to the task of adapting the wrapper to generate a new netlist.

4 RESISTIVE NEURAL ACCELERATOR SIMULATION & CIRCUIT VERIFICATION

4.1 ANN Architecture

As a small although relevant workload, we select the MNIST handwritten digit recognition task to employ the proposed design flow. After an initial normalization, the 28×28 input images are directly fed to the first convolutional layer. The structure of the considered DNN is reported in Table 1. The LogSoftmax activation function is adopted in the last layer for the decision-making step to ensure higher numerical stability compared to the standard Softmax function. *Brevitas* framework is used during the training phase to apply a signed 3-bits quantization to each convolutional and fully connected layer. During this step, the weights are still assumed to be mapped on purely linear devices. With such QAT procedure, the network is able to converge to an accuracy level of 97.5 %.

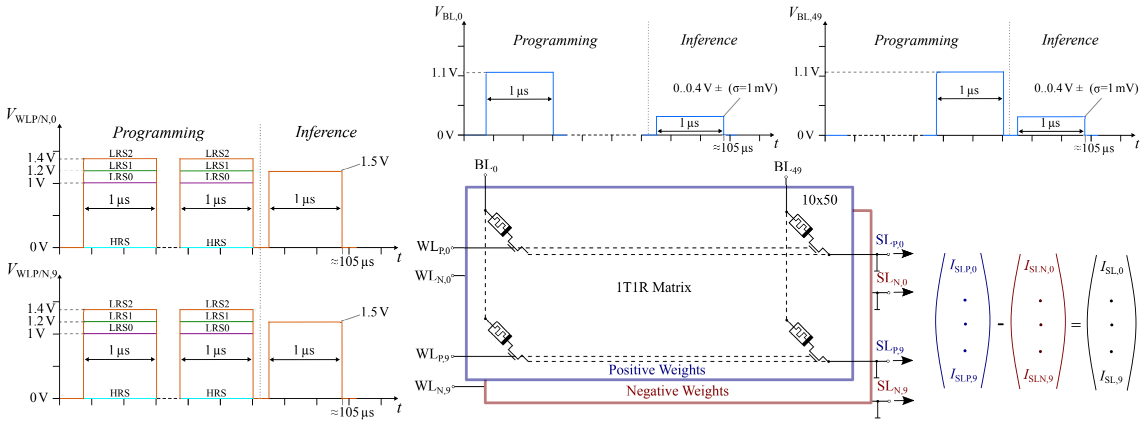

Figure 4: Test-bench & stimuli for 1T1R matrix simulation in *Cadence Spectre.*

Table 2: Mapping of Synaptic Weights of the FC Output Layer into Hardware Parameters

RRAM States	Conductance	Integer Rep.	Bits
LRS2-HRS	+120 µS	+3	+"11"
LRS1-HRS	+90 µS	+2	+"10"
LRS0-HRS	+60 µS	+1	+"01"
HRS-HRS	0 µS	0	"00"
HRS-LRS0	−60 µS	−1	-"01"
HRS-LRS1	−90 µS	−2	-"10"
HRS-LRS2	−120 µS	−3	-"11"

4.2 Mapping to Hardware Parameters

To correctly reproduce the signed weights in the RRAM levels, a differential approach with two paired crossbars is considered [1]. The available 4 conductance states are therefore combined to represent the signed 3-bits integer values in the range [-3:+3] (Table 2). While the two least significant bits (LSBs) define the two conductance levels required to store the weight, the most significant bit (MSB) represents the sign of the weight and it states in which crossbar the two levels are set. It should be noted that with the available conductance levels, the considered differential states are not evenly distributed. In particular, there is a 60 μS gap between the conductances representing the integers -1, 0, and +1, while all other states are divided by 30 μS. The resulting non-linearity will inevitable add an error factor in the final result compared to theoretical output of the *PyTorch* simulation. The input signals in the digital domain are extracted by the *Python* wrapper during the inference phase. Coming from a rectified linear unit (ReLU) activation function, only positive inputs have to be translated in voltage levels. This is performed with an analog linear mapping in the [0 V : 0.4 V] range.

To translate back to the digital domain with a linear transformation the output of the matrices, it is necessary to calculate the theoretical maximum and minimum output currents during inference with Equations 1 and 2.

$$I_{\max} = \Delta G_{\max} \times V_{\max} \times n_{\text{cols}} \tag{1}$$

$$I_{\min} = \Delta G_{\min} \times V_{\max} \times n_{\text{cols}} \tag{2}$$

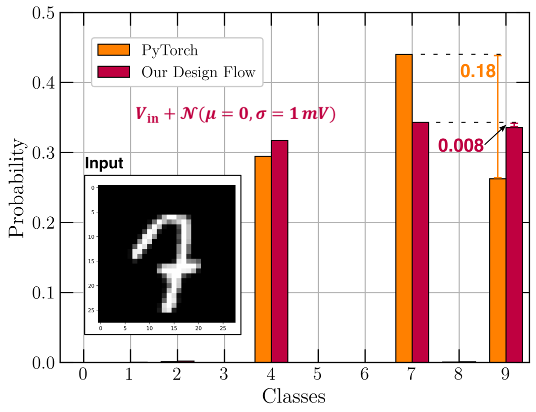

Figure 5: Output predictions with *PyTorch* simulation (orange) and in circuit simulation with deviation of the inference voltage (red) for the '7' digit reported in the inset.

$\Delta G_{\max}$ and $\Delta G_{\min}$ are the maximum and minimum conductance difference (±120 µS), $V_{\max}$ is the maximum voltage applied to read the device (0.4 V), and n_{cols} is the number of columns that contribute to the output current. Assuming that both the analog differential current range and the digital range are symmetric, the digital result of the VMM can be obtained from the output current applying Equation 3 with $y_{\max}$ and $y_{\min}$ maximum and minimum values observed during the *PyTorch* ANN simulation and back-annotated by the proposed *Python* wrapper.

$$y = (I_y - I_0)\frac{\Delta I}{\Delta y} + y_0 = (I_y - I_0)\frac{I_{\max} - I_{\min}}{y_{\max} - y_{\min}} + y_0 \tag{3}$$

4.3 Circuit Simulation Results

In an ideal situation with RRAM devices having evenly distributed conductance levels, the results coming from the *Python* evaluation and the circuit simulation are expected to be identical. This can change when the real behaviour of the RRAM devices is considered.

As a demonstration of the presented 1T1R model and design flow, the last layer of the aforementioned ANN is simulated in *Spectre* after being entirely mapped in two 10×50 1T1R matrices. To showcase an additional possibility, the robustness to input variations is assessed applying a Gaussian noise ($\mu = 0, \sigma = 1\,\text{mV}$) to the generated voltage signals for the inference phase. Fig. 4 shows the input waveforms with the considered circuits. The differential output currents are then translated from the circuit simulation to the *PyTorch* environment to compare the results with the expected recognition accuracy. It is interesting to note how even a moderate number of non-idealities can be problematic for certain critical input data. As an example, when the digit '7' presented in Fig. 5 is applied as input, the network is able to correctly identify it in both our design flow and *PyTorch*. Nevertheless, the confidence interval between the correct class '7' and the wrong '9' is reduced more than one order of magnitude when a more realistic representation of the hardware is considered thanks to the circuit simulation. This entails that with higher input variability, the wrong prediction will be given. In the figure, the log probability obtained at the output of the ANN is translated in the standard $[0, 1]$ unit interval via exponential operation. The more precise assessment of the network performance once deployed in a resistive neural hardware accelerator can, moreover, be achieved with a reasonable simulation time. The circuital simulation of the entire programming phase including 50 pulses plus one inference operation of the fully-connected layer required 7.49 s on a single *Intel Xeon*® E5-4627 v4 CPU @ 2.60 GHz clock frequency with conservative accuracy settings.

5 CONCLUSION

In this work, we present a *Verilog-A* LUT-based model for a 1T1R cell that can depict the full switching behaviour of the RRAM devices, while achieving $15\times$ faster simulation time compared to physical-based models based on computational demanding equations with possible convergence issues. To verify its functionalities, a design flow for resistive neural hardware accelerators is proposed. The required interaction between *PyTorch*, an ANN high-abstraction framework, and *Spectre*, state-of-the-art tool for circuital analysis, is achieved via a custom *Python* wrapper that generates the stimuli files for the circuit simulation.

As a representative use case, an ANN for the MNIST handwritten digit recognition task is designed and analyzed with the proposed design flow. While the rest of the network runs in *PyTorch*, the last layer is fully mapped and simulated at the circuit level on two matrices employing the presented 1T1R model. The obtained results not only prove the low computational requirement of our approach (7.49 s for two 10×50 matrices), but they also highlight how the performance determined by the *PyTorch* analysis can be much more optimistic compared to the actual circuit level simulation. This underlines the importance of including circuital analysis for the performance assessment of a neural network hardware accelerator. The current version of the model aims to reproduce the device's deterministic behavior. Nevertheless, the stochastic nature of the underlying physial mechanism can make the inclusion of variability effects relevant for the reliability circuit assessment. Future interesting research may focus on the integration in the model of stochastic behaviour (e.g., set/reset voltages, conductance values), as well as the inclusion in the design flow of larger mixed-signal system with more sophisticated programming and inference circuitry.

ACKNOWLEDGMENTS

This work was funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 434434223 – SFB 1461 and 441921944.

REFERENCES

[1] Ping Chi et al. 2016. PRIME: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-Based Main Memory. In *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*. 27–39. https://doi.org/10.1109/ISCA.2016.13

[2] Charlotteet Frenkel et al. 2023. Bottom-Up and Top-Down Approaches for the Design of Neuromorphic Processing Systems: Tradeoffs and Synergies Between Natural and Artificial Intelligence. *Proc. IEEE* 111, 6 (2023), 623–652. https://doi.org/10.1109/JPROC.2023.3273520

[3] Benoit Jacob et al. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2704–2713. https://doi.org/10.48550/arXiv.1712.05877

[4] Arya Jagath et al. 2019. An Insight Into Physics Based RRAM Models – A Review. *The Journal of Engineering* 2019 (05 2019). https://doi.org/10.1049/joe.2018.5234

[5] Corey Lammie et al. 2022. MemTorch: An open-source simulation framework for memristive deep learning systems. *Neurocomputing* 485 (2022), 124–133. https://doi.org/10.48550/arXiv.2004.10971

[6] Yann LeCun et al. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324. https://doi.org/10.1109/5.726791

[7] Alessandro Pappalardo. 2022. Xilinx/brevitas. (2022). https://doi.org/10.5281/zenodo.3333552

[8] Xiaochen Peng et al. 2020. DNN+ NeuroSim V2. 0: An end-to-end benchmarking framework for compute-in-memory accelerators for on-chip training. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40, 11 (2020), 2306–2319. https://doi.org/10.48550/arXiv.2003.06471

[9] Antonio J. Pérez-Ávila et al. 2020. Behavioral modeling of multilevel HfO 2-based memristors for neuromorphic circuit simulation. In *2020 XXXV Conference on Design of Circuits and Integrated Systems (DCIS)*. IEEE, 1–6. https://doi.org/10.1109/DCIS51330.2020.9268652

[10] Emilio Pérez-Bosch et al. 2021. Toward Reliable Compact Modeling of Multilevel 1T-1R RRAM Devices for Neuromorphic Systems. *Electronics* 10 (03 2021), 645. https://doi.org/10.3390/electronics10060645

[11] Emilio Perez-Bosch Quesada et al. 2023. Experimental Assessment of Multilevel RRAM-Based Vector-Matrix Multiplication Operations for In-Memory Computing. *IEEE Transactions on Electron Devices* 70, 4 (2023), 2009–2014. https://doi.org/10.1109/TED.2023.3244509

[12] Kamilya Smagulova et al. 2023. Resistive Neural Hardware Accelerators. *Proc. IEEE* 111, 5 (2023), 500–527. https://doi.org/10.1109/JPROC.2023.3268092

[13] Weier Wan et al. 2022. A compute-in-memory chip based on resistive random-access memory. *Nature* 608 (08 2022), 504–512. https://doi.org/10.1038/s41586-022-04992-8

[14] Jianan Wen et al. 2021. Behavioral Model of Dot-Product Engine Implemented with 1T1R Memristor Crossbar Including Assessment. In *2021 24th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*. 29–32. https://doi.org/10.1109/DDECS52668.2021.9417070

[15] Jianan Wen et al. 2022. Evaluating Read Disturb Effect on RRAM based AI Accelerator with Multilevel States and Input Voltages. In *2022 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)*. 1–6. https://doi.org/10.1109/DFT56152.2022.9962345

[16] Cheng-Xin Xue et al. 2020. 15.4 A 22nm 2Mb ReRAM Compute-in-Memory Macro with 121-28TOPS/W for Multibit MAC Computing for Tiny AI Edge Devices. In *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*. 244–246. https://doi.org/10.1109/ISSCC19947.2020.9063078

[17] Ussama Zahid et al. 2020. FAT: Training Neural Networks for Reliable Inference Under Hardware Faults. In *2020 IEEE International Test Conference (ITC)*. 1–10. https://doi.org/10.1109/ITC44778.2020.9325249

[18] Furqan Zahoor et al. 2020. Resistive Random Access Memory (RRAM): an Overview of Materials, Switching Mechanism, Performance, Multilevel Cell (MLC) Storage, Modeling, and Applications. *Nanoscale Research Letters* 2020 (05 2020), 15. https://doi.org/10.1186/s11671-020-03299-9