NeuroVP: A System-Level Virtual Platform for Integration of Neuromorphic Accelerators

Melvin Galicia
Institute for Communication Technologies and Embedded Systems
RWTH Aachen University
Aachen, Germany
galicia@ice.rwth-aachen.de

Ali BanaGozar
Electronics Systems
Eindhoven University of Technology
Eindhoven, The Netherlands
a.banagozar@tue.nl

Karl Sturm
Institute for Communication Technologies and Embedded Systems
RWTH Aachen University
Aachen, Germany
sturm@ice.rwth-aachen.de

Felix Staudigl
Institute for Communication Technologies and Embedded Systems
RWTH Aachen University
Aachen, Germany
staudigl@ice.rwth-aachen.de

Sander Stuijk
Electronics Systems
Eindhoven University of Technology
Eindhoven, The Netherlands
s.stuijk@tue.nl

Henk Corporaal
Electronics Systems
Eindhoven University of Technology
Eindhoven, The Netherlands
h.corporaal@tue.nl

Rainer Leupers
Institute for Communication Technologies and Embedded Systems
RWTH Aachen University
Aachen, Germany
leupers@ice.rwth-aachen.de

***Abstract*—Executing neural network (NN) applications on general-purpose processors result in a large power and performance overhead, due to the high cost of data movement between the processor and the main memory. Neuromorphic computing systems based on memristor crossbars, perform the NN main operation i.e., vector-matrix multiplications (VMM) in an efficient way in the analog domain. Thus, they circumvent the costly energy overhead of its digital counterpart. It can be expected that neuromorphic systems will be used initially as complements to current high-performance systems rather than as a replacement. This paper presents NeuroVP, a virtual platform integrating a neuromorphic accelerator, developed in SystemC that can model functionality, timing, and power consumption of the components integrating the system. Using NeuroVP to evaluate performance and power consumption at the electronic system level (ESL), it is corroborated that the execution of NN applications with a neuromorphic accelerator yields of up to 46x higher power efficiency and 26x speedup relative to a general-purpose computing system.**

Keywords—Virtual Platform, SystemC, RISC-V, Neuromorphic Accelerator, Memristor, ESL Power Estimation

I. Introduction

Substantial advances in neural networks (NNs) have made their extensive adoption in solving problems such as image classification [1], face recognition [2], speech recognition [3], and many others possible. Additionally, the ever-increasing data generation and the development of high-performance computing systems have collaborated to the positive feedback loop by allowing more training data and faster training times. However, NN applications running in general-purpose computing architectures are extremely power and performance inefficient, since they require massive data transfers from CPU to memory and vice versa and also require a huge number of multiply-accumulate (MAC) operations mainly [4, 5].

Brain-inspired neuromorphic computing, based on memristive devices in crossbar architectures, is one of the most promising emerging technologies for NN accelerators since it offers solutions to the challenges of massive data transfer and a large number of MAC operations. Memristors not only can serve as memory elements but also perform computation on the stored data [6, 7]. Thereby computations can be performed directly in-memory reducing the data transfer bottleneck. Additionally, memristive crossbar architectures, illustrated in Fig. 1, can be used to efficiently perform vector-matrix multiplications (VMMs) used in deep neural networks (DNNs). They reduce the time complexity of VMM operations from $O(n^2)$ to $O(n)$ or even to $O(1)$ [8] if considering only the analog computation on the crossbar array, therefore facilitating the VMM acceleration and improving the overall power efficiency of the system.

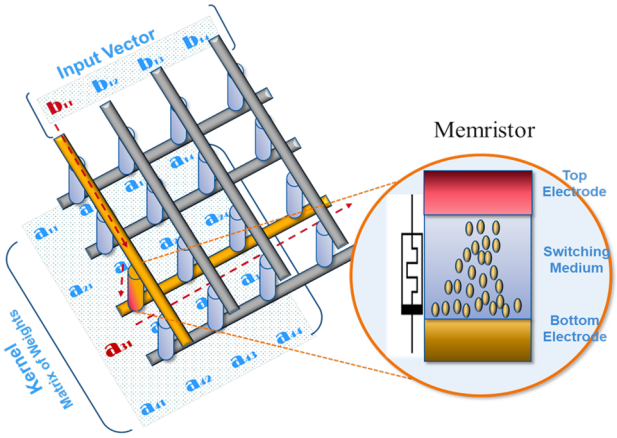

Fig. 1. Memristor-based crossbar array.

This work was supported by the Federal Ministry of Education and Research (BMBF, Germany) within the NEUROTEC project (No. 16ES1134 and 16ES1133K).

However, all these benefits are commonly expressed relative to stand-alone crossbar array accelerators. A more comprehensive finding might be concluded when examining these types of neuromorphic accelerators integrated within a full system, i.e. CPU, memory cache, data buses, and other peripherals devices altogether. Since, it can be expected that neuromorphic systems will be used initially as complements to current high-performance systems rather than as a replacement, which as well might happen but at a much later point in time.

Analyzing the whole system's behavior at the electronic system level (ESL) with the help of a virtual platform (VP) is an efficient way to carry out design space exploration (DSE) while integrating neuromorphic accelerators in the system. VPs offer high flexibility while enabling hardware and software co-design of the new neuromorphic paradigm. SystemC with transaction-level modeling (TLM) [9] is widely used to build VPs and to perform thorough DSE. This paper introduces Neuromorphic-Virtual-Platform (NeuroVP) developed in SystemC that can model functionality, timing, and power consumption of the components integrating the system, allowing to estimate the system's complete behavior during the early stages of the design. The contributions of NeuroVP are as follows:

- Application of ESL power estimation methodology using a VP that includes a neuromorphic accelerator.
- DSE using a VP for the integration of general-purpose memristive accelerator, with different crossbar sizes, into a complete system.
- Verifying the consistency of power estimation results at the ESL modeling for non-cycle-accurate simulations compared to cycle-accurate ones for neuromorphic systems.

This paper is structured as follows: Section II provides related work and background to this paper. Section III introduces the structure of NeuroVP and its characteristics. Section IV summarizes the used ESL power estimation methodology. In Section V, the significance of NeuroVP is demonstrated by using it to perform simulations. It provides system power results of selected case studies. Finally, in Section VI conclusions are drawn and remarks on future work are provided.

II. Related Work

A good deal of effort is being invested by academia in developing different types of neuromorphic simulators. Ranging from high-performance like PUMA [10], ISAAC [11] to edge-computing such as Tiny-but-Accurate [12] and the one proposed by G. Yuan et al. in [13]. They are also designed as purpose-specific, like Pinatubo [14], MemTorch [15], or general-purpose such as, CIM-SIM [16], PRIME [4] and NMSIM [17]. PUMA proposes a programmable architecture and ISA design organized in three tiers: cores, tiles, and nodes bringing programmability and generality to memristor crossbars. On the other hand, Tiny-but-Accurate takes full advantage of the "alternating direction method of multipliers" algorithm with memristor hardware constraints, thus achieving an extremely high compression rate with minimum accuracy loss. Pinatubo introduces a processor in memristor-based architecture for bulk bitwise operations, including OR, AND, XOR, and INV. Oppositely, CIM-SIM presents a general-purpose accelerator that includes the memristor crossbar with surrounding analog drivers, providing the required interface to the co-processing digital elements, and it presents a micro-instruction set architecture that controls and operates both analog and digital components. All aforementioned simulators use memristive crossbar arrays to simulate the performance of NN required operations and benchmarks within their respective architectures. The execution of those benchmarks drives the system's overall performance. To report the power consumption, due to such executions, of the proposed accelerator architecture or the power consumption of the full system, all proposed works rely on register transfer level (RTL) or lower levels of simulation making the whole estimation process time consuming and very computationally demanding.

Power estimation at abstraction levels above RTL has been extensively investigated by academia already. Approaches like [18] and [19], focus on the integration of various ESL power models into the workflow, while other approaches provide a methodology of how to create the power models, e.g. [20]. A commercial framework like Docea Aceplorer [21] is a good example of ESL power estimation being adopted in the industry due to its advantage for rapid and accurate estimation. Works from academia have developed power models suitable for ESL simulations for different kinds of system components, such as dynamic memories (DRAMs) [22], peripheral cores [23], and communication architectures in general as proposed in [24]. Therefore, NeuroVP uses for the first time, ESL power estimation methodology for obtaining the system-level power consumption of a system that integrates a neuromorphic accelerator. Furthermore, it allows a rapid DSE and performing fast instruction-accurate simulations.

III. SystemC Virtual Platform

The NeuroVP consists of several SystemC modules: a main processor, L1 instruction and data caches, a bus, main memory, and a neuromorphic accelerator. Each module can work as a transaction initiator and a transaction target depending on its current function or just as a forwarder in the case of the bus module. The modules have their corresponding sockets for

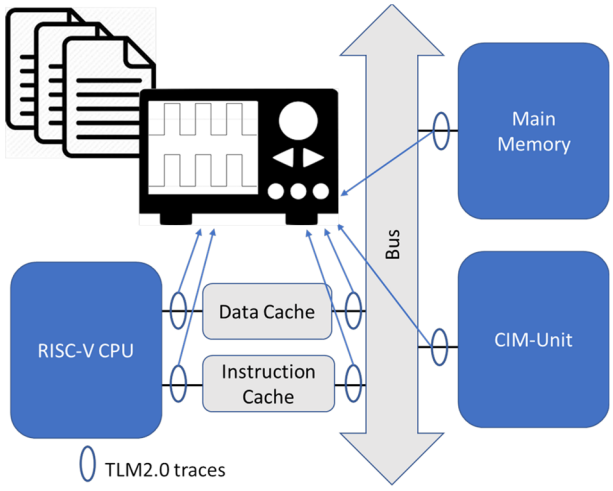

Fig. 2. NeuroVP high level architecture.

TLM2.0 transaction-based communication where the tracing of each transaction is being monitored and stored. Fig. 2 depicts NeuroVP's high level architecture and illustrates the tracing approach. By collecting all the transactions, it is possible to have a histogram and the total number of transactions that occurred during the execution of any particular benchmark.

A. RISC-V

NeuroVP uses RISC-V [25], an open-source Instruction Set Architecture (ISA) which is license-free and royalty-free, as the main system processor. The RISC-V core implemented in [26] using SystemC and TLM-2.0 is the one integrated into NeuroVP. This implementation offers a 32/64-bit RISC-V core supporting the IMAC instruction set with different privilege levels, CLINT interrupt controller with a set of peripherals. It is designed as an extensible and configurable platform with a generic bus system. Additionally, it provides a sensor peripheral and extension debug functionality from the software application perspective. By adding this SystemC RISC-V 64IMAC, NeuroVP achieves a significantly faster simulation compared to using an RTL implementation, while being more accurate than other commercially available Instruction Set Simulators (ISSs). Lastly, the aforementioned RISC-V implementation is fully open source following the MIT licensing model.

The RISC-V processor is used to execute different software applications that exploit the available neuromorphic accelerator by offloading specific types of operations, such as VMMs, meanwhile being free to perform other computational tasks. It is important to point out that the RISC-V core does not make use of any standard extension instructions to offload operations to the accelerator but rather it is the Computation In-Memory (CIM)-Unit that interprets and handles the executions of the operations using its internal controller and own set of micro-instructions.

B. CIM-Unit

NeuroVP integrates the CIM-Unit, proposed in [27], since it provides built-in interfaces that allow interaction with SystemC-based virtual platforms in several abstraction levels. The CIM-Unit consists of two main parts Calculator and its digital surrounding, the Micro-engine as shown in Fig. 3. The calculator is a simulator that not only replicates the functional behavior of memristor crossbars but also covers the operation of surrounding mixed-signal circuitries, e.g., ADC, DAC, S+H, which are essential for driving the memristor crossbar. The micro-engine, on the other hand, comprises the digital components, e.g., controller, registers files, and buffers, that [27] finds necessary for operating the memristor crossbar. In addition to these, the CIM-Unit offers a micro-instruction set that allows the unit to communicate with the main processor, if it is deployed as an accelerator --as it is the case in this work-- or other functional units, in a coarse grain reconfigurable architecture. For the CIM-Unit to operate it should first be supplied with some necessary parameters, e.g., size of the matrix that is to be mapped on the crossbar, I/O resolution, etc. The controller stores these parameters, which are passed to the CIM-Unit through several instructions, into the configuration register. As soon as all the necessary parameters are received, Controller --which is a state-machine-- transitions into the state IN signaling that is ready to accept input data. It stays in IN until all

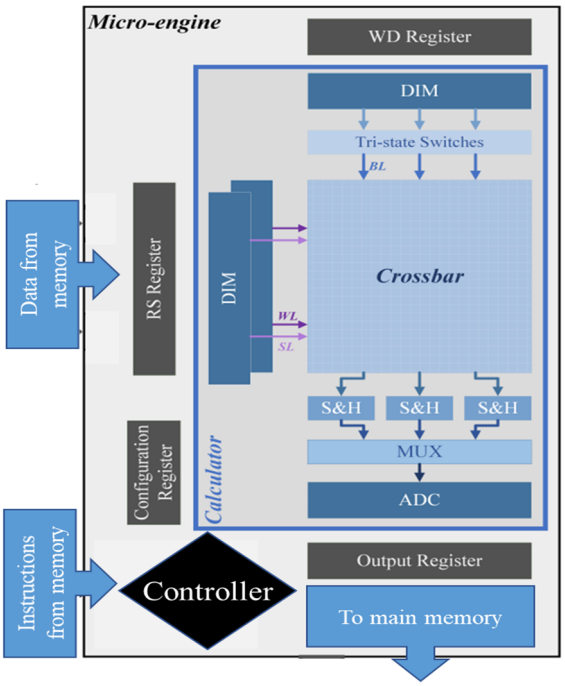

Fig. 3. CIM-Unit architecture.

the input data is received. The number of cycles that it takes to be ready to move to the next state depends on the width of the input ports, the resolution of the input data, and the size of the input vector. In the next state, OP, the Controller executes the operation that is specified in the configuration register. In the end, the Controller moves to the final state, OUT, where it sends out the processed information and goes back to the initial state, IDLE, as soon as all the output is sent out.

C. Memory Modules

Main memory (DRAM) and instruction and data cache (SRAM) modules are taken from an in-house library. In addition to commonly modeling memory in SystemC with a read and write access delay, further delays for page switches and write-to-read switches are introduced. Each module has its annotated power model for leakage power, sequential and random dynamic power for writing and reading operations.

IV. System Power Estimation

The ESL power estimation methodology relies on obtaining a reference power trace for the component of interest in one arbitrary scenario. There are a few options for obtaining this reference; by using a power simulation at lower levels such as gate or layout level, by using direct power measurements from a device containing the component of study, or ultimately, by using reference values reported by the research community. In all previous cases, there is always the possibility of further refinement and subsequent calibration.

The available power information from the reference scenario is then back-annotated to the ESL model, thus creating the ESL power model for the component and then storing it into the component's ESL library. Afterward, it can be used to estimate the power consumption of the component in the same system running different scenarios or even in different systems

with different scenarios. If power models for all components of the system are available, the power consumption of the entire system can be predicted using the ESL simulation. This power estimation methodology is based on the work first presented in [18] and is briefly summarized as follows:

The power consumption of CMOS devices is a combination of leakage and dynamic power. Leakage power can be assumed as constant as long as temperature and supply voltage changes are not taken into consideration. Dynamic power, on the other hand, depends on short circuit and switching power caused by the total activity performed by the device. The dynamic power depends on control signals which initiate actions or instructions performed by the device. Therefore, to estimate the dynamic power consumption at ESL, it is sufficient to trace those control signals, as illustrated in Fig. 2, to calculate the power. A total of N traces of control signals will be recorded over T sampling periods. All traces can be represented by a matrix $\mathbf{Q} \in \mathbb{N}^{T \times N}$. Where the first trace is always set to 1, i.e. $Q_{t,1} = 1$, to take into consideration, the constant part of the power consumption. The power estimate is then calculated as:

$$\mathbf{P}_{\text{est}} = \mathbf{Q} \cdot \mathbf{f} \quad \text{with} \quad \mathbf{f} \in \mathbb{R}^{N} \tag{1}$$

where $\mathbf{f}$ denotes a vector of power model factors, which allows for the model's calibration. A reference power trace $\mathbf{P}_{\text{ref}} \in \mathbb{R}^{T}$ recorded at a lower level than ESL simulation, or using a real hardware sample, should represent closely the ESL traces as much as possible. Hence, $\mathbf{f}$ can be calculated using the pseudo-inverse matrix $\mathbf{Q}^{+}$as follows:

$$\mathbf{f} \coloneqq \mathbf{Q}^{+} \cdot \mathbf{P}_{\text{ref}} \tag{2}$$

In general, more activity in the internals of a component corresponds to higher dynamic power consumption, with an approximately linear relationship. Therefore, all in-house developed ESL models have been extended to deliver all available information of their internal signals to the linear power model that will compute the power estimation utilizing (1). A reference power trace necessary for the RISC-V processor was obtained performing an RTL simulation using the 64 bit 6-stage CVA6 (formerly Ariane core) from [28]. In the case of the main memory and cache memories values from in-house libraries reported in [19] were used.

V. Methodology and Simulations

A. System Hardware Characteristics

Table I lists the components that play an active role in the power and performance of the entire system. Main RAM memory and cache memories have small sizes just enough to handle bare-metal applications. Adding larger memory sizes would mainly add static power consumption to the overall system. Reference for the energy values for ADC, DAC, sample and hold (S+H) components of the CIM-Unit are the ones reported in ISAAC [11]. As for the crossbar and the Micro-engine digital components are the ones reported in [27].

B. NeuroVP Operation flow

A RISC-V bare-metal application triggers the VMM operations required by the NN. If the neuromorphic accelerator is enabled, the system will proceed to offload those operations otherwise it will execute them using the main processor and the main memory. To offload the VMM operations, all micro-instructions, required by the CIM-Unit's Micro-engine to perform the calculations on the memristor crossbar, are pushed to a reserved program address space on main memory. Similarly, it is done for the matrix and vector values being pushed to the respective data address space reserved for the CIM-Unit. Afterwards, a start signal is sent to the Micro-engine controller and it will fetch all data from memory without the help of the RISC-V processor. The aforementioned process flow is illustrated in Fig. 4. The system's main processor is now free to execute other applications if any was included on the benchmark, which is not the case in this work for an adequate comparison. The operation flow inside the CIM-Unit is described in [27] in detail. In summary, the weight values will be written once in the memristor crossbars and the convolution operation with the input vectors happen sequentially and results are stored in the main memory. The process repeats until the last program instruction is encountered and then a done message is

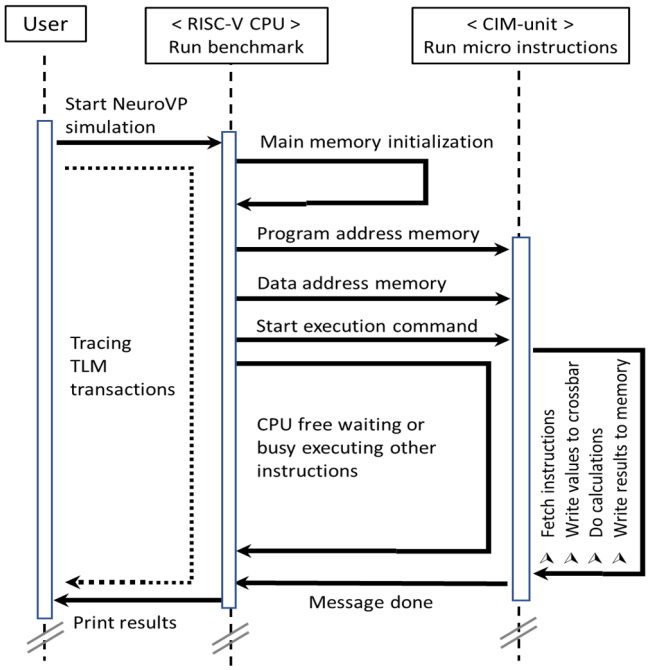

Fig. 4. NeuroVP simulation flow.

TABLE I. System Hardware Characteristics

Component	Parameters
Processor	1 core 64-bits; 1.7 GHZ; in order
L1 I&D cache	SRAM 16KB & 32KB
RAM	DRAM 128MB
CIM-Unit (values per 8 bits) [11] [27]	
DAC	3.3 pJ
ADC	13 pJ
S+H	8.3 fJ
Crossbar -- Compute	200 fJ
Crossbar -- Write	200 pJ
Crossbar sizes	32x32, 64x64, 128x128, 256x256
Micro-engine (digital)	64.8 pJ

sent to the RISC-V. Finalizing the bare-metal application will also trigger the end of the NeuroVP simulation and subsequent dumping of all collected data to an external file.

All operations and TLM transactions between each component at any point during the simulation are stored for post-processing and estimation of the entire system power and performance. Internal operations and statistics are as well collected, for components that deliver that information.

C. Benchmarks

Convolutional layers from Googlenet [29], ImageNet [1], and MobileNets [30] NN are selected as benchmarks, to show the acceleration within layers that requires several vector multiplications. The layers are listed in Table II, where **m** and **n** are the matrix height and width respectively and **p** is the number of vectors. The matrix and vector sizes varied to assess different scenarios; some matrices fit into the crossbar, whereas some do not. When the number of columns or rows of the weight matrix is bigger than that of the memristor crossbar, the task is then divided evenly over time to fit the crossbar. A straightforward algorithm of a nested loop for a VMM operation was implemented. This implementation is used in both cases, for RISC-V plus main memory case and also for the case when the host system offloads the operations to the CIM-Unit. However, for the latest case, the internal handling is left to the Micro-engine.

D. Results

In this work, power efficiency is represented by the number of operations (for every 8-bit) performed per watt (GOPS/W) similar to [11]. Simulation results of the power efficiency of the complete system, for the benchmarks listed in Table II, are presented in Fig. 5. From all cases, up to 46× higher power efficiency was registered for ImageNet_conv_1. The CIM-Unit was configured with only one crossbar array of 128 ×128. Fig. 6 shows results for different sizes of the crossbar array implementation for one benchmark. As expected the efficiency increases with the size of the crossbar, which is more evident for the cases when the tasks do not need to be divided over time anymore. Additionally, Fig. 7 shows the speedup achieved when using the CIM-Unit relative to the main processor, using simulation time as a measurement metric. The main factors that influence the speedup achieved by the CIM-Unit are crossbar write and calculate latencies, where updating the matrix weights takes the biggest penalty. However, this is still faster than the overall data movement and latencies to access the main memory to update the partial results incurred by the main processor. In the best case up to a 26× speedup was registered as result of utilizing the neuromorphic accelerator.

TABLE II. NETWORK LAYER BENCHMARKS

Network Name	Layer Type _id	m	n	p
Googlenet	Conv_1	224	224	7
Googlenet	Conv_2	56	56	3
ImageNet	Conv_1	224	224	11
ImageNet	Conv_2	207	207	5
MobileNets	Conv_1	224	224	3
MobileNets	Conv_2	112	112	3

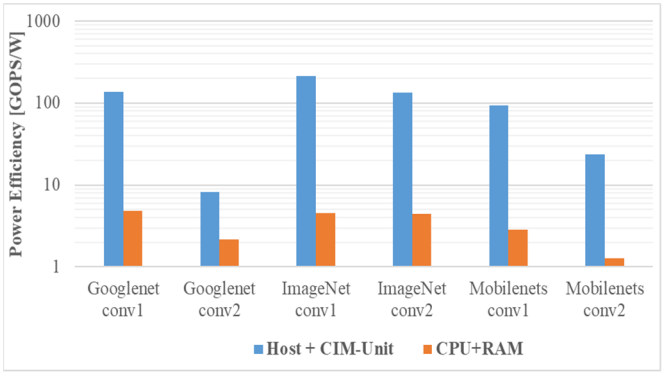

Fig. 5. Overall system power efficiency.

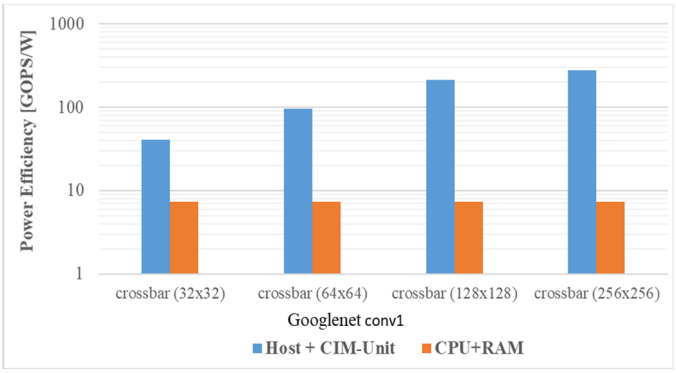

Fig. 6. System power efficiency for different crossbar sizes.

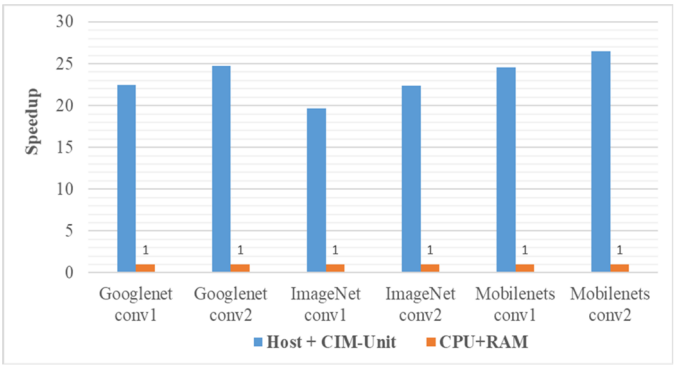

Fig. 7. System speedup from simulation time.

It is important to notice that in all cases the CIM-Unit still requires access to the main memory. Furthermore, all components of the VP are active during the benchmark execution, even if they are not being specifically utilized by the benchmark. All this considered, reported power efficiency and speedup using NeuroVP are in line with figures reported in other works that use cycle-accurate simulations [10, 11, 27]. This corroborates the fitness of the ESL methodology for modeling this type of systems. Discrepancies can be rationalized due to other accelerators relying much less on accessing the main memory, and including software techniques for optimizing VMM operations. Other case studies like configuring more than one crossbar array inside the CIM-Unit or integrating more than one CIM-Unit inside the virtual platform were left out of this work, but certainly, such DSE is possible using NeuroVP.

VI. Conclusion

This paper presented NeuroVP, a SystemC-based system-level virtual platform that integrates a neuromorphic accelerator. It uses a RISC-V 64IMAC as the main processor unit and offloads NN operations to a neuromorphic accelerator. The introduced virtual platform allows the verification of power and performance estimation results at the ESL modeling using TLM2.0 tracing while offering a rapid DSE. The execution of selected NN applications using a neuromorphic accelerator yields up to 46× higher power efficiency and 26× speedup relative to a general-purpose computing system.

Future work might extend the virtual platform to integrate distinct neuromorphic accelerators, and compare performance between them or explore complementary acceleration when they are designed as purpose-specific.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton. 2012. ImageNet classification with deep convolutional neural networks. In Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'12). Curran Associates Inc., Red Hook, NY, USA, 1097–1105.

[2] F. Schroff, D. Kalenichenko, and J. Philbin. 2015. FaceNet: A Unified Embedding for Face Recognitionand Clustering. CoRR abs/1503.03832 arXiv:1503.03832 http://arxiv.org/abs/1503.03832.

[3] Z. Q. Lin, A. G. Chung, and A. Wong. 2018. EdgeSpeechNets: Highly Efficient Deep Neural Networks for Speech Recognition on the Edge. CoRR abs/1810.08559 (2018).

[4] P. Chi, *et al*., 2016. PRIME: A Novel Processing-in-memory Architecture for Neural Network Computation in ReRAM-based Main Memory. SIGARCH Comput. Archit. News 44, 3 (June 2016), 27–39. https://doi.org/10.1145/3007787.3001140.

[5] X. Qiao, X. Cao, H. Yang, L. Song, and H. Li. 2018. Atomlayer: A Universal reRAM-based CNN Accelerator with Atomic Layer Computation. In Proceedings of DAC (San Francisco, California) (DAC '18). ACM, New York, NY, USA, Article 103, 6 pages. https://doi.org/10.1145/3195970.3195998.

[6] H. Akinaga and H. Shima. 2010. Resistive Random Access Memory (ReRAM) Based on Metal Oxides. Proc. IEEE 98, 12 (Dec 2010), 2237–2251. https://doi.org/10.1109/JPROC.2010.2070830.

[7] M. A. Lastras-Montaño and K. Cheng. 2018. Resistive random-access memory based on ratioed memristors. Nature Electronics 1 (08 2018), 466–472. https://doi.org/10.1038/s41928-018-0115-z.

[8] C. Lammie, O. Krestinskaya, A. James and M. R. Azghadi, "Variation-aware Binarized Memristive Networks," 2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Genoa, Italy, 2019, pp. 490-493, doi: 10.1109/ICECS46596.2019.8964998.

[9] SystemC. [Online] http://www.accellera.org/downloads/standards/systemc (accessed 03/2021).

[10] A. Ankit, *et al*., "PUMA: A Programmable Ultra-efficient Memristor based Accelerator for Machine Learning Inference," CoRR, vol. abs/1901.10351, 2019. http://arxiv.org/abs/1901.10351.

[11] A. Shafiee, *et al.,* 2016. ISAAC: A Convolutional Neural Network Accelerator with In-Situ Analog Arithmetic in Crossbars. In 2016 ISCA. 14–26. https://doi.org/10.1109/ISCA.2016.12.

[12] X. Ma et al., "Tiny but Accurate: A Pruned, Quantized and Optimized Memristor Crossbar Framework for Ultra Efficient DNN Implementation," 2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC), Beijing, China, 2020, pp. 301-306, doi: 10.1109/ASP-DAC47756.2020.9045658.

[13] G. Yuan *et al*., (2019). An Ultra-Efficient Memristor-Based DNN Framework with Structured Weight Pruning and Quantization Using ADMM. 1-6. 10.1109/ISLPED.2019.8824944.

[14] S. Li, C. Xu, Q. Zou, J. Zhao, Y. Lu, and Y. Xie. (2016). Pinatubo: a processing-in-memory architecture for bulk bitwise operations in emerging non-volatile memories. 1-6. 10.1145/2897937.2898064.

[15] C. Lammie and M. R. Azghadi, "MemTorch: A Simulation Framework for Deep Memristive Cross-Bar Architectures," 2020 IEEE International Symposium on Circuits and Systems (ISCAS), Seville, Spain, 2020, pp. 1-5, doi: 10.1109/ISCAS45731.2020.9180810.

[16] A. BanaGozar, K. Vadivel, S. Stuijk, H. Corporaal, S. Wong, M. Abu Lebdeh, J. Yu, and S. Hamdioui. 2019. "CIM-SIM: Computation In Memory SIMuIator," in Proceedings of the 22nd International Workshop on Software and Compilers for Embedded Systems (SCOPES '19). Association for Computing Machinery, New York, NY, USA, 1–4. https://doi.org/10.1145/3323439.3323989.

[17] Li. Xia, et al., (2017). MNSIM: Simulation Platform for Memristor-Based Neuromorphic Computing System. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. PP. 1-1. 10.1109/TCAD.2017.2729466.

[18] S. Schürmans, G. Onnebrink, R. Leupers, G. Ascheid and X. Chen, "ESL power estimation using virtual platforms with black box processor models," 2015 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), Samos, Greece, 2015, pp. 354-359, doi: 10.1109/SAMOS.2015.7363698.

[19] G. Onnebrink, R. Leupers, and G. Ascheid. 2018. "ESL Black Box Power Estimation: Automatic Calibration for IEEE UPF 3.0 Power Models," in Proceedings of the Rapido'18 Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools (RAPIDO '18). Association for Computing Machinery, New York, NY, USA, Article 1, 1–6. https://doi.org/10.1145/3180665.3180667

[20] S. K. Rethinagiri, R. ben Atitallah, and J.-L. Dekeyser. A system level power consumption estimation for MPSoC. In 2011 Intl. Symposium on System on Chip. IEEE, 2011.

[21] Docea Aceplorer. [Online] http://www.doceapower.com/products-services/aceplorer.html (accessed 03/2021).

[22] M. Jung, C. Weis, P. Bertram, and N. Wehn. Power modelling of 3D stacked memories with TLM2.0 based virtual platforms. In Synopsys User Group Conference (SNUG), 2013.

[23] T. Givargis, F. Vahid, and J. Henkel. Instruction-based system-level power evaluation of system-on-a-chip peripheral cores. Very Large Scale Integration (VLSI) Systems, 2002.

[24] S. Schurmans, D. Zhang, D. Auras, R. Leupers, G. Ascheid, X. Chen, and L. Wang. Creation of ESL power models for communication architectures using automatic calibration. In 50th Design Automation Conference, DAC '13. ACM, 2013.

[25] A. Waterman and K. Asanovic,´ The RISC-V Instruction Set Manual; Volume I: User-Level ISA, SiFive Inc. and CS Division, EECS Department, University of California, Berkeley, 2017.

[26] V. Herdt, D. Grosse, P. Pieper, and R. Drechsler, RISC-V based virtual prototype: An extensible and configurable platform for the system-level, Journal of Systems Architecture, Volume 109, 2020, 101756, ISSN 1383-7621, https://doi.org/10.1016/j.sysarc.2020.101756.

[27] A. BanaGozar, K. Vadivel, J. Multanen, P. Jääskeläinen, S. Stuijk, H. Corporaal. 2020. "System Simulation of Memristor Based Computation in Memory Platforms," in Embedded Computer Systems: Architectures, Modeling, and Simulation. SAMOS 2020. Lecture Notes in Computer Science, vol 12471. Springer, Cham. https://doi.org/10.1007/978-3-030-60939-9_11

[28] F. Zaruba and L. Benini, "The Cost of Application-Class Processing: Energy and Performance Analysis of a Linux-Ready 1.7-GHz 64-Bit RISC-V Core in 22-nm FDSOI Technology," in IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 27, no. 11, pp. 2629-2640, Nov. 2019, doi: 10.1109/TVLSI.2019.2926114.

[29] C. Szegedy et al., "Going deeper with convolutions," 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 2015, pp. 1-9, doi: 10.1109/CVPR.2015.7298594.

[30] A. Howard et al., (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. arXiv:1704.04861 http://arxiv.org/abs/1704.04861