NeuroVM: Dynamic Neuromorphic Hardware Virtualization

Murat Isik
Drexel University
Philadelphia, USA
mci38@drexel.edu

Kayode Inadagbo
Prairie View A&M University
Prairie View, USA
kayodeinadagbo@gmail.com

I. Can Dikmen
Temsa R&D Center
Adana, Turkey
can.dikmen@temsa.com

Abstract

This paper introduces a novel approach in neuromorphic computing, integrating heterogeneous hardware nodes into a unified, massively parallel architecture. Our system transcends traditional single-node constraints, harnessing the neural structure and functionality of the human brain to efficiently process complex tasks. We present an architecture that dynamically virtualizes neuromorphic resources, enabling adaptable allocation and reconfiguration for various applications. Our evaluation, using diverse applications and performance metrics, provides significant insights into the system's adaptability and efficiency. We observed scalable throughput increases across configurations of 1, 2, and 4 Virtual Machines (VMs), reaching up to 5.1 Gibibits per second (Gib/s) for different data transfer sizes. This scalability highlights the system's proficiency in managing data-intensive tasks. Energy consumption analysis in our virtualized accelerator environment showed a near-linear growth with the addition of more NeuroVM accelerators, ranging from 25 to 45 millijoules (mJ) as the number of accelerators increased from 1 to 20. Additionally, our investigation into reconfiguration overheads revealed that partial reconfigurations significantly reduce time compared to full reconfigurations, particularly as the number of VMs increases, with time reductions evident in the logarithmic scale of time measurements.

I. Introduction

The landscape of computational architectures has significantly evolved, transitioning from traditional CPU-based systems to specialized, parallel architectures like GPUs and FPGAs. These advancements have improved processing power, energy efficiency, and adaptability to diverse workloads. However, as computational tasks become more complex, especially in artificial intelligence and machine learning, there is a need for innovative computing paradigms [1]–[3].

Neuromorphic computing, inspired by the neural structures and functions of the human brain, offers a transformative solution. By emulating neural networks and synaptic connections, neuromorphic systems can efficiently process complex cognitive tasks. These systems leverage parallel processing and adaptive learning principles, making them suitable for various applications, including artificial intelligence, robotics, and scientific computations.

The potential of neuromorphic computing is maximized through integrating heterogeneous neuromorphic hardware nodes into a unified, massively parallel system. This integration harnesses the strengths of diverse architectures to handle computations beyond single-node capabilities [4]–[7]. The challenge lies in developing scalable and portable software technologies for large-scale neuromorphic systems and exploring virtualization in these environments.

FPGAs have gained rapid acceptance due to their versatility across applications. The time-consuming nature of feature extraction algorithms poses a drawback for real-time applications. Using dedicated hardware like FPGAs, which perform complex operations in parallel, addresses this issue. FPGA virtualization abstracts FPGA hardware, decoupling the interface and hiding the framework's complexity [8]–[11]. Definitions and methodologies for FPGA virtualization have evolved with changing application requirements. This paper addresses these challenges by proposing a novel neuromorphic architecture leveraging dynamic virtualization to efficiently handle heterogeneous workloads. Through exploring the integration, virtualization, and optimization of neuromorphic hardware nodes, we aim to realize the full potential of neuromorphic computing for a wide range of scientific and computational applications. Our contributions include:

- Proposing a novel neuromorphic architecture that integrates multiple hardware nodes through dynamic virtualization, enhancing the system's ability to handle complex computations and adapt to varying workloads.
- Conducting an extensive analysis of key performance indicators such as throughput, energy efficiency, and resource utilization, and performing a comparative study with traditional single-node neuromorphic systems to highlight the advantages of our multi-node, virtualized architecture.
- Outlining future research directions, focusing on integrating specialized accelerators with the neuromorphic fabric and exploring security implications in virtualized neuromorphic environments.

This paper investigates the union of diverse neuromorphic hardware nodes within a parallel framework. **Section 2** reviews the history and concept of virtualization in neuromorphic computing. **Section 3** explores the architectural design and

the dynamic virtualization essential for resource management. **Section 4** assesses the system's performance using different VM configurations. **Section 5** concludes by summarizing the study's primary insights. Finally, **Section 6** envisions future work on integrating specialized accelerators and securing virtualized neuromorphic environments.

II. Background

Neuromorphic computing, inspired by the human brain, represents a major advancement in artificial cognition. These systems mimic neural networks and synaptic connections, crucial for AI, robotics, and complex data analysis. However, single-node hardware configurations limit their potential. Transitioning to integrated, parallel systems is natural but challenging, especially regarding software compatibility and adaptability with expansive neuromorphic setups.

Our goal is to create a unified system architecture to efficiently manage tasks across neuromorphic hardware nodes, each contributing unique capabilities. The system is designed to be flexible and self-optimizing in response to varying workloads [12], [13]. We are exploring neuromorphic virtualization for dynamic reconfiguration and resource sharing, aiming to enhance the adaptability, efficiency, and performance of neuromorphic systems [14], [15].

Virtualization has become fundamental in computing, optimally allocating capabilities between hardware and OS. Conceived by IBM in the 1960s to partition mainframes into multiple virtual instances, virtualization has evolved to improve efficiency and reduce costs. The hypervisor, or virtual machine monitor (VMM), abstracts physical resources from the OS, enabling multiple OSs to run simultaneously on a single hardware platform [16]–[18]. This enhances resource utilization and strengthens security, reliability, and resilience. With cloud computing's rise, virtualization bridges hardware and software applications, creating a cohesive operational ecosystem. Various virtualization techniques, including full, OS-layer, hardware, para, application, and resource virtualization, offer distinct benefits in resource sharing, isolation, and efficiency.

FPGA virtualization is not new; various microkernel-based approaches have been explored, enabling mapping and exchange of hardware accelerators to VMs. However, these systems focused mainly on resource utilization, neglecting energy efficiency and diverse guest OS requirements.

To fill this void, the FPGA virtualization layer L4ReC has been developed. It facilitates the shared use of reconfigurable resources by multiple guest OS while considering embedded reconfigurable systems' constraints. L4ReC addresses critical research questions related to limitations of the embedded environment, support for various guest OS requirements, and ensuring system isolation. Isolation includes performance isolation to reduce mutual interference and data isolation to guard against malicious applications. L4ReC components allow customization to suit specific embedded system needs. The L4ReC strategy includes integrating FPGA virtualization into the micro-hypervisor L4Re, an OS framework designed for systems with strict real-time, security, safety, and virtualization requirements. L4ReC introduces virtual FPGAs (vFPGAs) as an abstraction from the physical FPGA, allowing VMs to use hardware accelerators regardless of location. A critical application for virtualized embedded reconfigurable systems is simultaneous execution of RTOS and GPOS, resulting in hardware threads with different real-time priorities. L4ReC provides a dynamic mapping and scheduling strategy for these threads, considering real-time needs and ensuring that threads with less stringent deadlines do not disrupt those with tighter ones. Preliminary results from implementing L4ReC on a Xilinx Ultrascale MPSoC show promise for increased FPGA resource utilization and energy efficiency. The synchronization strategy for hardware threads aims to extend FPGA sleep phases, improving energy efficiency—a vital consideration for battery-powered devices. The L4ReC virtualization layer marks a significant advancement in optimizing the use of reconfigurable resources in embedded systems. Initial findings indicate substantial improvements in resource management and energy savings, depending on application characteristics [19].

Authors [20] explore Linux's foundational concepts and technologies underpinning the Virtio-FPGA solution. They provide background on the Virtio standard, the Linux FPGA Manager component of the Linux kernel, and the VFIO pass-through for ARM in QEMU and Device Tree Overlays technologies. FPGA overlay architectures on computational storage devices enable programmable near-storage processing. These architectures consist of reconfigurable operators, crossbar stream switches, and on-board DRAM, facilitating data movements between operators and the FPGA's DRAM. The storage interfaces in these architectures allow direct access to storage units, adopting the NVMe standard for fast and parallel access.

Software support for these systems includes an abstraction layer simplifying near-storage processing, exposing FPGA overlay architecture operators as executable files within an OS. HLS tools allow users to customize operators using FPGA reconfigurability. An example HLS code snippet demonstrates how users can define a stream data structure and input/output ports for an operator, streamlining near-storage data processing. In I/O virtualization, software-based virtualization presents virtual device instances for device sharing across VMs. Full virtualization uses a trap-and-emulate approach, while paravirtualization creates VM-friendly virtual device interfaces. Hardware-assisted virtualization mechanisms, such as PCIe SR-IOV, allow direct access to PCIe devices, enabling resource sharing at the hardware level without software arbitration.

III. Methodology

A. Neuromorphic Hardware Virtualization

Neuromorphic hardware, evolving from basic silicon neurons to advanced neuromorphic chips, offers significant advantages in dynamic resource management and virtualization. The integration of FPGAs with neuromorphic systems brings new capabilities in virtualization and resource utilization. This section explores methodologies and implications of virtualizing neuromorphic hardware, focusing on design, application, and system architecture. Initial efforts aimed to replicate brain

TABLE I: Comparison of Neuromorphic Virtualization vs. Digital Virtualization

Functional Hierarchy	Key Technology	Neuromorphic Virtualization Solution	Digital Virtualization Solution
Resource Pool Management	Integrated neuromorphic resources, centralized management, dynamic allocation, monitoring, maintenance, and unified scheduling.	Use neuromorphic-specific management tools for resource pools.	Use traditional virtualization management tools like VMware, and Hyper-V.
Virtualization Layer	Neuromorphic virtualization, VM management, and container management.	Employ neuromorphic-specific virtualization technologies for resource abstraction.	Utilize hardware-assisted or software-assisted virtualization technologies.
Resource Isolation Layer	Hardware isolation technology, software isolation technology, or a combination of the two.	Implement neuromorphic-specific isolation technologies to prevent resource contention.	Use established isolation technologies like Intel VT-x, and AMD-V.
Scheduling Layer	Neuromorphic resource scheduling algorithm, load balancing, resource prediction.	Developing and use neuromorphic-aware scheduling algorithms for efficient resource allocation.	Apply conventional scheduling and load balancing algorithms.
Application Layer	Neuromorphic task scheduling algorithm, dynamic migration technology, application deployment, and management.	Utilize algorithms optimized for neuromorphic computing tasks and dynamic resource management.	Leverage general-purpose computing algorithms and migration technologies like live VM migration.

neural structures in silicon, progressing to sophisticated neuromorphic chips for various fields. FPGAs are ideal for neuromorphic systems due to their flexibility and reconfigurability, aligning with efficient computing models. Virtualization enables dynamic resource management, meeting variable computational demands by reconfiguring resources. Studies have explored design methodologies for dynamic management and reconfiguration of neuromorphic systems [12], [21]–[23].

Neuromorphic hardware, designed to emulate biological neurons, is pivotal in computing. As networks become intricate, computational demands, particularly for Spiking Neural Network (SNN) inference, grow, intensifying trade-offs between hardware resources, power consumption, and performance. Neuromorphic Hardware Virtualization addresses these challenges by introducing dynamic resource allocation and reconfiguration. Neurons communicate using spikes, an efficient mechanism reducing overall logic occupation. This is crucial for NeuroVM, where dynamic resource allocation and virtualization demand efficient communication protocols. This framework integrates hardware design, security algorithms, and neuromorphic computing principles within a virtualized environment, creating robust neuromorphic systems capable of adapting to various tasks while ensuring data integrity.

The virtualization of neuromorphic hardware is central to our research, promising new paradigms of efficiency and flexibility. Through careful task profiling, memory, and interconnect optimization, and integrating a sophisticated kernel controller driver, we aim to advance neuromorphic computing. Figure 1 illustrates the intricate architecture of a neuromorphic computing system, showing the arrangement and interconnection of multiple 'Neurocores.' These neurocores are fundamental processing units designed to emulate brain neural circuits, providing unique cognitive and processing capabilities. The hardware is modular, allowing configurations tailored to specific tasks. 'Local Interconnects' facilitate communication between neurocores, essential for parallel processing and neural network simulations. This parallelism is key to high-speed, energy-efficient computation in neuromorphic hardware.

B. FPGA Virtualization

Neuromorphic computing necessitates hardware platforms capable of emulating neural architectures. FPGAs are well-suited for this due to their reconfigurable nature and parallel processing capabilities. FPGA virtualization abstracts physical FPGA resources to enable multiple applications on a single chip. This section details methodologies and strategies in FPGA virtualization, emphasizing Dynamic Function Exchange (DFX) for adaptability and efficiency, conceptually illustrated in Figure 2.

1) Dynamic Function Exchange: DFX is an FPGA feature facilitating runtime reconfiguration of hardware functions without disrupting system operation. DFX allows selective activation and deactivation of hardware modules, enabling the FPGA to adapt to new tasks on-the-fly. This is crucial in neuromorphic computing, where dynamic reconfiguration in response to neural network demands is essential.

Our approach leverages DFX to implement a virtualized neuromorphic hardware environment, enabling real-time instantiation and reconfiguration of neural network models. This dynamic reconfiguration is managed by a dedicated controller orchestrating the loading and unloading of DFX modules.

C. Neuromorphic Hardware Virtualization

Our research explores innovative strategies for virtualizing neuromorphic hardware. This involves dynamic partitioning of hardware resources for concurrent execution of diverse tasks while maintaining strict data isolation and security protocols, ensuring task integrity. Virtualization enables dynamic reconfiguration and resource sharing, enhancing neuromorphic computing systems' adaptability and performance. In I/O virtualization, software-based mechanisms offer virtual device instances for device sharing across multiple VMs. Full virtualization and paravirtualization are two such mechanisms, with the latter minimizing VM exits and memory-mapped I/O operations. Hardware-assisted virtualization mechanisms, such as PCIe SR-IOV, allow direct PCIe device access, enabling resource sharing at the hardware level while maintaining isolation.

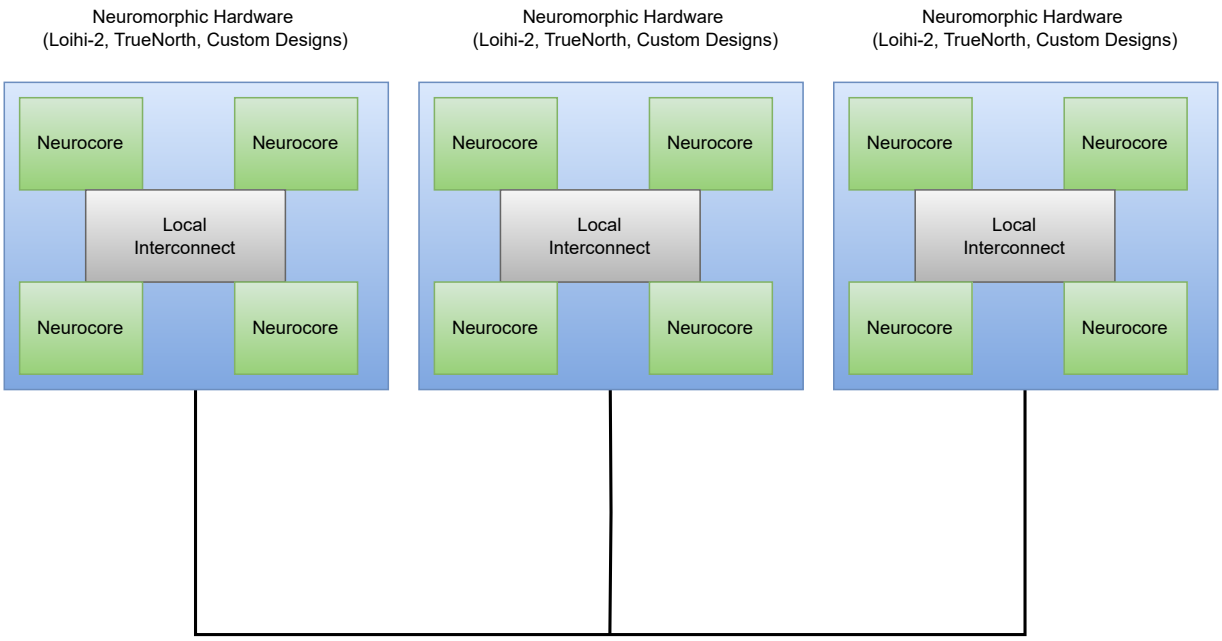


Fig. 1: Schematic of Neurocore Interconnectivity in Neuromorphic Hardware

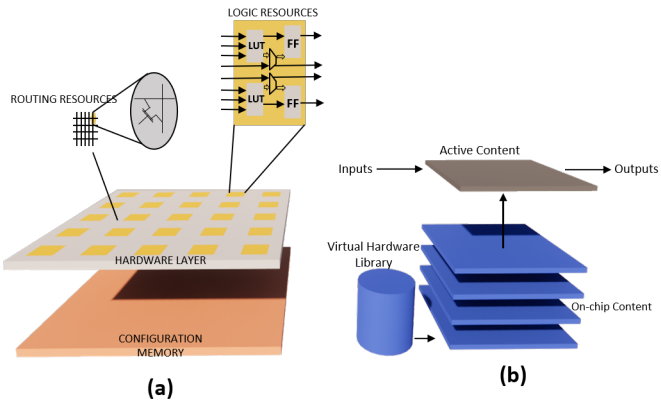


Fig. 2: FPGA Virtualization Concept

1) Task Profiling and Allocation: Our methodology includes creating an advanced task profiling system, assessing tasks based on processing demands and parallel execution suitability. This guides task allocation to appropriate neuromorphic nodes, optimizing resource utilization and computational throughput.

2) Memory and Interconnect Optimization: We design specialized memory hierarchies and interconnect architectures tailored to neuromorphic hardware nuances. The goal is to establish a high-bandwidth, low-latency communication framework for efficient data exchange across the neuromorphic network.

3) Kernel Controller Driver Integration and Mapping: The kernel controller driver manages data influx and control over neuromorphic processor cores. Inspired by existing paradigms like the Coyote driver, our design focuses on virtualizing neuromorphic processors, replacing traditional VFPGAs with bespoke designs. The driver interfaces with user space through C++ constructs, ensuring seamless hardware interaction.

4) FPGA Virtualization and C++ Integration: FPGA virtualization segments and manages FPGA capabilities to mimic multiple discrete processing units. This is pivotal for resource management, akin to virtual machines in conventional computing. Our project leverages the Vitis software platform for FPGA development, utilizing C++ and embedded systems expertise to realize PS-PL relationships within the FPGA architecture. We focus on high-level synthesis in Vitis for neuromorphic virtualization needs.

A summary of the resource utilization report for the Zynq UltraScale+ XCZU7EV MPSoC is presented in Table II. The table includes resources such as Logic Cells, Memory, DSP Slices, and I/O Pins. The "Utilization" column indicates resources used by the design, while the "Available" column reflects total FPGA resources. Utilization percentages are calculated for reference.

TABLE II: Resource utilization summary

Zynq UltraScale+ XCZU7EV			
Resource	**Utilization**	**Available**	**% Utilization**
LUT	151,200	504,000	30
Memory	11.4MB	38MB	30
IO	139	464	29.19
DSP	518	1,728	29.94

Figure 3 presents a detailed view of the task management and processing architecture within a neuromorphic computing system. At the forefront is the 'Driver,' managing system inputs and outputs, interfacing between the external environment and internal processing units. Beneath the driver lies the 'Service Scheduler,' distributing tasks across processing units.

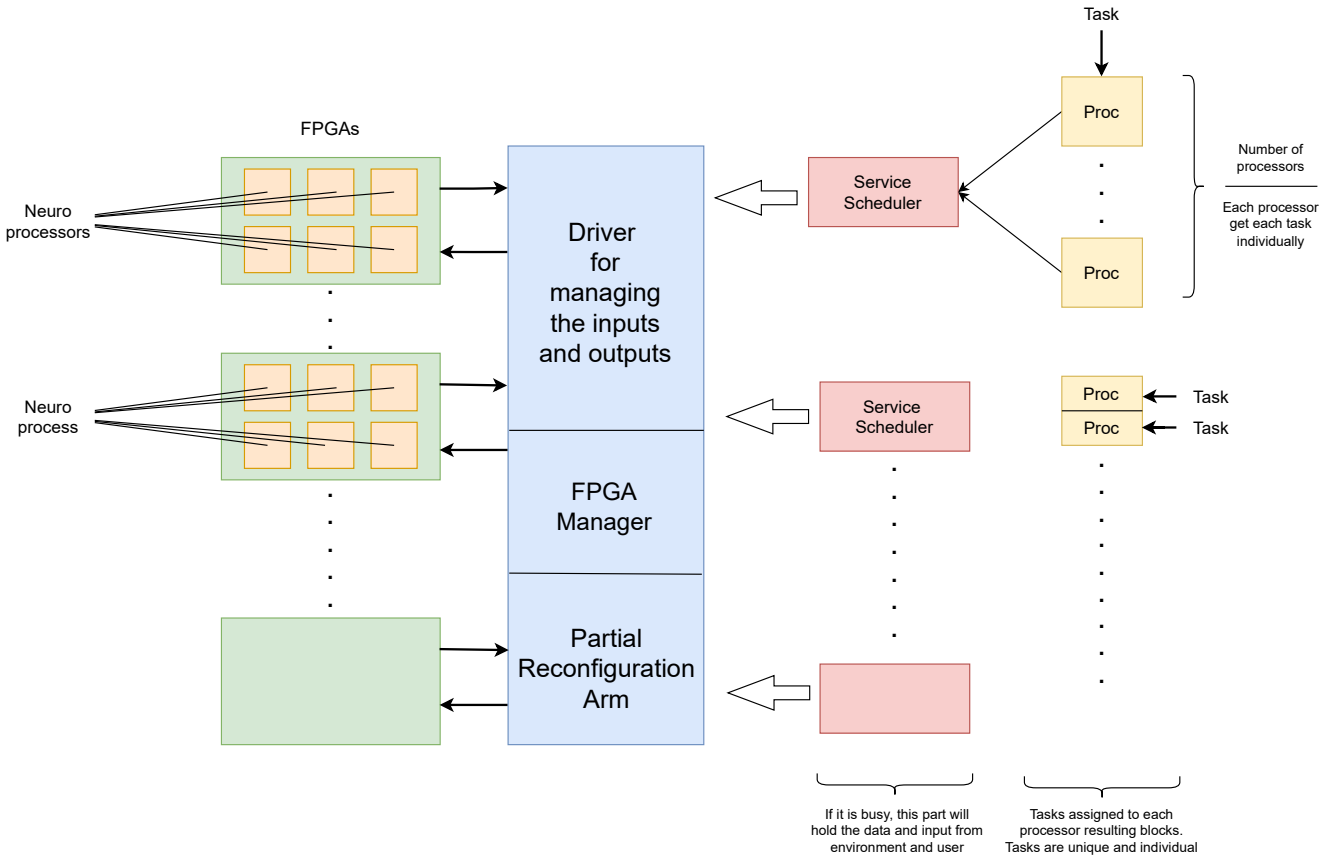


Fig. 3: Task Management and Processing Architecture in Neuromorphic Systems

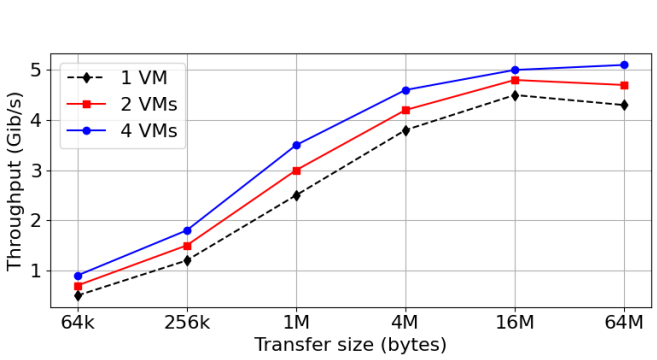
This scheduler efficiently allocates tasks to 'Neuroprocessors,' optimized for neural network simulations and brain-inspired computations. The 'Proc' blocks represent individual processing units working with neuroprocessors to execute tasks. The 'FPGA Manager' oversees 'Partial Reconfiguration' of FPGAs, adapting hardware configurations on-the-fly for different tasks. The 'Arm Proc' blocks may represent general-purpose processors providing additional support. The figure highlights a buffering mechanism to manage data during peak processing times, ensuring no loss of information. 'Tasks' are depicted as blocks assigned to each processor, showcasing the system's ability to handle multiple concurrent tasks, processed by dedicated hardware units. This architecture allows high-throughput, efficient computation, leveraging neuroprocessors and FPGAs' unique properties for flexibility and efficiency.

IV. Evaluation

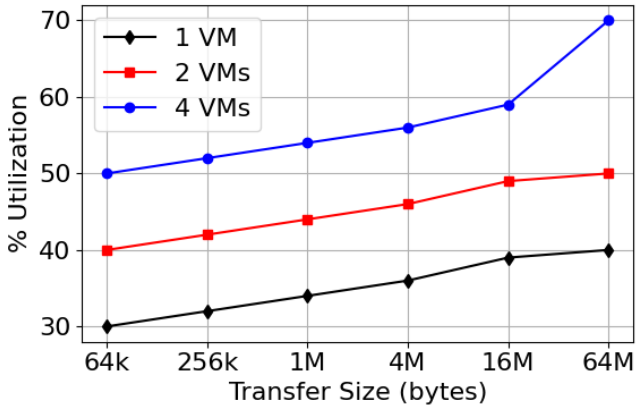
To rigorously assess the efficacy and practicality of our proposed neuromorphic architecture, we structured our evaluation around the following pivotal components. Our system underwent an exhaustive battery of tests, utilizing a comprehensive suite of applications spanning multiple domains such as machine learning, data analytics, and signal processing. This diverse workload spectrum enabled us to meticulously analyze the adaptability and performance of our architecture across a broad range of scenarios, ensuring its versatility in handling varied computational tasks.

The success of our proposed architecture was quantitatively measured against a set of key performance indicators, including throughput, energy efficiency, resource utilization, and the overhead associated with reconfiguration. These metrics were evaluated under various operational conditions and configurations to provide a clear, objective assessment of the system's overall performance and efficiency, highlighting its strengths and areas for potential improvement. This comparison, focusing on specific features such as scalability, adaptability, and energy consumption, served to underscore the tangible benefits and advantages that our multi-node, virtualized architecture brings to the table, setting it apart from its predecessors.

Figure 4 (a) illustrates the relationship between transfer size and throughput in Gibibits per second (Gib/s) for 1, 2, and 4 Virtual Machines (VMs). The data points, represented in distinct colors and markers for each VM configuration, reveal how throughput scales with increasing transfer sizes. The black dashed line (1 VM), red solid line (2 VMs), and blue dotted line (4 VMs) collectively provide insights into the efficiency and scalability of the VMs under varying data transfer loads. This analysis is crucial for understanding the performance dynamics of neuromorphic computing systems in data-intensive scenarios. Figure 4 (b) depicts the percentage



(a) Throughput



(b) Hardware Resource Utilization

Fig. 4: Comparative analysis of NeuroVM performance.

of resource utilization against different transfer sizes for 1, 2, and 4 VMs. Each line, differentiated by color and marker style, indicates the utilization efficiency of the VMs. The plot highlights how resource utilization varies with the size of data transfers, providing valuable insights into the operational efficiency of neuromorphic VMs. The increasing trend in utilization with larger transfer sizes, especially noticeable in the configurations with more VMs, underscores the impact of VM density on resource management in neuromorphic computing environments.

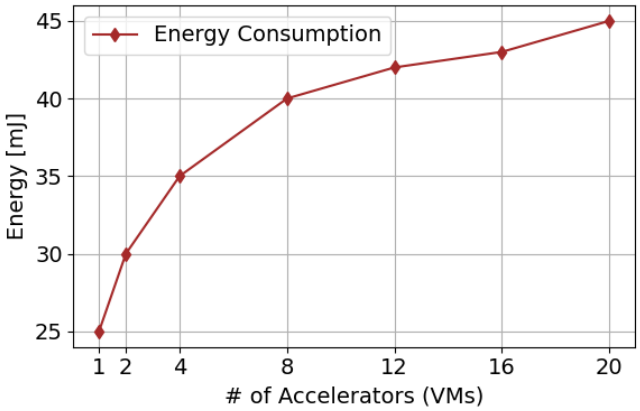


Fig. 5: NeuroVM Energy Consumption.

Figure 5 illustrates the energy consumption trends in a virtualized accelerator environment. The data reveals a pattern of increasing energy demand correlating with the rising number of NeuroVM accelerators. This trend is crucial for understanding the energy efficiency of the system, particularly in the context of neuromorphic computing where energy management is a key performance metric. The near-linear increase in energy consumption with additional accelerators underscores the importance of optimizing resource allocation to balance computational power and energy efficiency.

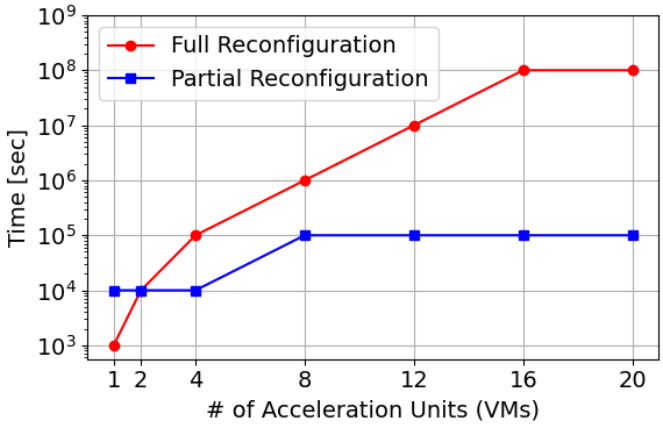


Fig. 6: NeuroVM implementation overhead with full and partial reconfigurations.

Figure 6 presents a comparative analysis of the implementation overhead for full and partial reconfigurations in a NeuroVM environment. The red line with circle markers depicts the time taken for full reconfigurations, and the blue line with square markers represents partial reconfigurations. Notably, the partial reconfiguration demonstrates a significantly reduced time overhead compared to full reconfiguration, especially as the number of VMs increases. This observation is pivotal for optimizing the performance of neuromorphic virtual machines, where reconfiguration times play a critical role in overall system efficiency and responsiveness. The data underscores the potential benefits of partial reconfiguration strategies in dynamic computing environments where rapid adaptation is essential.

Through this comprehensive evaluation, we aim to demonstrate the robustness, versatility, and efficiency of our neuromorphic computing architecture. Our architecture is designed to tackle a diverse array of applications and harness the benefits of virtualization, showcasing the potential of virtualized neuromorphic hardware in shared computational environments. This evaluation will not only validate the performance of our

system but also highlight its potential in fostering collaborative and adaptive computing solutions.

V. Conclusion

This study has presented a comprehensive evaluation of a novel neuromorphic computing architecture, demonstrating its potential to revolutionize the field of high-performance computing. Our proposed system, characterized by its multi-node, virtualized neuromorphic architecture, has been rigorously tested across various performance metrics, including throughput, energy efficiency, resource utilization, and reconfiguration overhead. The results, as illustrated in the accompanying figures, underscore the significant advancements our architecture offers over traditional single-node neuromorphic systems. The integration of virtualization in neuromorphic hardware opens new avenues for scientific exploration and lays the groundwork for further research and development in this field. As we continue to explore the intricacies of integrating various neuromorphic hardware configurations and optimizing software technologies, we are poised to unlock new possibilities for scientific discovery and technological advancements, further cementing the role of neuromorphic computing as a pivotal tool in computational science.

VI. Future Works

Our future endeavors will focus on several key areas to further enhance the capabilities and applications of our proposed architecture.

A. Integration of Specialized Accelerators

A primary focus will be on the integration of specialized accelerators into the neuromorphic fabric. This integration aims to significantly enhance computational efficiency and performance, enabling the system to adeptly handle a broader spectrum of complex and computationally intensive tasks. We anticipate that this amalgamation will not only improve processing speeds but also expand the range of feasible applications, from advanced machine learning algorithms to real-time data analytics.

B. Security in Virtualized Neuromorphic Environments

With the shift towards a more virtualized neuromorphic environment, a thorough investigation into the security aspects becomes crucial. We plan to conduct an in-depth analysis of potential security threats and vulnerabilities inherent in shared computing ecosystems. This will involve developing and implementing robust security protocols and mechanisms to safeguard the system against unauthorized access and other forms of cyber threats, ensuring a secure and trustworthy computing environment.

C. Optimization for Diverse Applications

Continual refinement and optimization of our neuromorphic architecture will be a persistent effort. This optimization will be tailored to meet the specific demands and requirements of various applications, ranging from scientific simulations to industrial processing. Our goal is to create a versatile and adaptive system that can efficiently cater to the unique challenges posed by different computational tasks.

D. Advancing Neuromorphic Computing as a Mainstream Technology

Our vision is to establish neuromorphic computing as a cornerstone technology across various scientific and industrial sectors. We aim to demonstrate its potential in revolutionizing approaches to complex computations and simulations, thereby contributing significantly to advancements in these fields. Our efforts will be directed towards not only enhancing the performance of neuromorphic systems but also in making them more accessible and user-friendly, paving the way for widespread adoption and utilization.

References

[1] C. Bobda, J. M. Mbongue, P. Chow, M. Ewais, N. Tarafdar, J. C. Vega, K. Eguro, D. Koch, S. Handagala, M. Leeser *et al.*, "The future of fpga acceleration in datacenters and the cloud," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 15, no. 3, pp. 1–42, 2022.

[2] E. Nurvitadhi, J. Sim, D. Sheffield, A. Mishra, S. Krishnan, and D. Marr, "Accelerating recurrent neural networks in analytics servers: Comparison of fpga, cpu, gpu, and asic," in *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2016, pp. 1–4.

[3] C. M. Wyant, C. R. Cullinan, and T. R. Frattesi, "Computing performance benchmarks among cpu, gpu, and fpga," *Computing*, 2012.

[4] S. Qin, F. Wang, Y. Liu, Q. Wan, X. Wang, Y. Xu, and et al., "A light-stimulated synaptic device based on graphene hybrid phototransistor," *2D Mater.*, vol. 4, no. 3, p. 035022, 2017.

[5] Z. Du, D. D. Ben-Dayan Rubin, Y. Chen, L. He, T. Chen, L. Zhang, C. Wu, and O. Temam, "Neuromorphic accelerators: A comparison between neuroscience and machine-learning approaches," in *Proceedings of the 48th International Symposium on Microarchitecture*, 2015, pp. 494–507.

[6] A. Balaji, S. Song, A. Das, N. Dutt, J. Krichmar, N. Kandasamy, and F. Catthoor, "A framework to explore workload-specific performance and lifetime trade-offs in neuromorphic computing," *IEEE Computer Architecture Letters*, vol. 18, no. 2, pp. 149–152, 2019.

[7] O. Moreira, A. Yousefzadeh, F. Chersi, A. Kapoor, R.-J. Zwartenkot, P. Qiao, G. Cinserin, M. A. Khoei, M. Lindwer, and J. Tapson, "Neuronflow: A hybrid neuromorphic–dataflow processor architecture for ai workloads," in *2020 2nd IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 2020, pp. 01–05.

[8] A. Vaishnav, K. D. Pham, and D. Koch, "A survey on fpga virtualization," in *2018 28th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2018, pp. 131–1317.

[9] M. H. Quraishi, E. B. Tavakoli, and F. Ren, "A survey of system architectures and techniques for fpga virtualization," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 9, pp. 2216–2230, 2021.

[10] S. Zeng, G. Dai, H. Sun, K. Zhong, G. Ge, K. Guo, Y. Wang, and H. Yang, "Enabling efficient and flexible fpga virtualization for deep learning in the cloud," in *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2020, pp. 102–110.

[11] S. Yazdanshenas and V. Betz, "Quantifying and mitigating the costs of fpga virtualization," in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2017, pp. 1–7.

[12] M. Nilsson, O. Schelén, A. Lindgren, U. Bodin, C. Paniagua, J. Delsing, and F. Sandin, "Integration of neuromorphic ai in event-driven distributed digitized systems: Concepts and research directions," *Frontiers in Neuroscience*, vol. 17, p. 1074439, 2023.

[13] P. K. Huynh, M. L. Varshika, A. Paul, M. Isik, A. Balaji, and A. Das, "Implementing spiking neural networks on neuromorphic architectures: A review," *arXiv preprint arXiv:2202.08897*, 2022.

[14] L. Concha Salor and V. Monzon Baeza, "Harnessing the potential of emerging technologies to break down barriers in tactical communications," *Telecom*, vol. 4, no. 4, pp. 709–731, 2023. [Online]. Available: https://www.mdpi.com/2673-4001/4/4/32

[15] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: Opportunities and challenges," *Frontiers in neuroscience*, vol. 12, p. 774, 2018.
[16] R. Schmid, M. Plauth, L. Wenzel, F. Eberhardt, and A. Polze, "Accessible near-storage computing with fpgas," in *Proceedings of the Fifteenth European Conference on Computer Systems*, 2020, pp. 1–12.
[17] X. Li and D. L. Maskell, "Time-multiplexed fpga overlay architectures: A survey," *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, vol. 24, no. 5, pp. 1–19, 2019.
[18] A. Brant and G. G. Lemieux, "Zuma: An open fpga overlay architecture," in *2012 IEEE 20th international symposium on field-programmable custom computing machines*. IEEE, 2012, pp. 93–96.
[19] C. Wulf and D. Göhringer, "Virtualization of embedded reconfigurable systems," in *2022 32nd International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE, 2022, pp. 460–461.
[20] S. Bandara, A. Sanaullah, Z. Tahir, U. Drepper, and M. Herbordt, "Enabling virtio driver support on fpgas," in *2022 IEEE/ACM International Workshop on Heterogeneous High-performance Reconfigurable Computing (H2RC)*. IEEE, 2022, pp. 1–8.
[21] E. Johnson, C. Pronovost, and J. Criswell, "Hardening hypervisors with ombro," in *2022 USENIX Annual Technical Conference (USENIX ATC 22)*, 2022, pp. 415–436.
[22] B. Pachideh, C. Zielke, S. Nitzsche, and J. Becker, "Towards hardware-software self-adaptive acceleration of spiking neural networks on reconfigurable digital hardware," in *2023 IEEE 36th International System-on-Chip Conference (SOCC)*. IEEE, 2023, pp. 1–6.
[23] C. U. Kumar, P. Saravanan, N. Thiyagarajan, and V. Raj, "Digital implementation of neural network by partial reconfiguration," in *Neuromorphic Computing Systems for Industry 4.0*. IGI Global, 2023, pp. 226–260.