
AFRL-RI-RS-TR-2024-030

# GENESIS: A NEUROMORPHIC CHIP WITH LIFELONG LEARNING ON-DEVICE

UNIVERSITY OF TEXAS AT SAN ANTONIO

*MARCH 2024*

FINAL TECHNICAL REPORT



**AIR FORCE RESEARCH LABORATORY**
**INFORMATION DIRECTORATE**

■ **AIR FORCE MATERIEL COMMAND** ■ **UNITED STATES AIR FORCE** ■ **ROME, NY 13441**

# NOTICE AND SIGNATURE PAGE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09. This report is available to the general public, including foreign nations. Copies may be obtained from the Defense Technical Information Center (DTIC) (http://www.dtic.mil).

AFRL-RI-RS-TR-2024-030 HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION IN ACCORDANCE WITH ASSIGNED DISTRIBUTION STATEMENT.

FOR THE CHIEF ENGINEER:

**/ S /**
CLARE D. THIEM
Work Unit Manager

**/ S /**
NICK P. KOWALCHUK
RIT Chief Engineer
Computing & Communications Division
Information Directorate

This report is published in the interest of scientific and technical information exchange, and its publication does not constitute the Government's approval or disapproval of its ideas or findings.

# REPORT DOCUMENTATION PAGE

| 1. REPORT DATE | 2. REPORT TYPE | 3. DATES COVERED | |
|---|---|---|---|
| | | START DATE | END DATE |
| MARCH 2024 | FINAL TECHNICAL REPORT | SEPTEMBER 2020 | SEPTEMBER 2023 |

**4. TITLE AND SUBTITLE**

GENESIS: A NEUROMORPHIC CHIP WITH LIFELONG LEARNING ON-DEVICE

| 5a. CONTRACT NUMBER | 5b. GRANT NUMBER | 5c. PROGRAM ELEMENT NUMBER |
|---|---|---|
| FA8750-20-2-1003 | N/A | 62788F |
| **5d. PROJECT NUMBER** | **5e. TASK NUMBER** | **5f. WORK UNIT NUMBER** |
| | | R31B |


**6. AUTHOR(S)**

Dhireesha Kudithipudi, Fatima Tuz Zohora, Vedant Karia, Abdullah Zyarah, Nicholas Soures, Peter Helfer

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

University of Texas at San Antonio
One UTSA Circle
San Antonio TX 78249-1644


**8. PERFORMING ORGANIZATION REPORT NUMBER**

| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | 10. SPONSOR/MONITOR'S ACRONYM(S) | 11. SPONSOR/MONITOR'S REPORT NUMBER(S) |
|---|---|---|
| Air Force Research Laboratory/RITB<br>525 Brooks Road<br>Rome NY 13441-4505 | AFRL/RI | AFRL-RI-RS-TR-2024-030 |

**12. DISTRIBUTION/AVAILABILITY STATEMENT**

Approved for Public Release; Distribution Unlimited. This report is the result of contracted fundamental research deemed exempt from public affairs security and policy review in accordance with SAF/AQR memorandum dated 10 Dec 08 and AFRL/CA policy clarification memorandum dated 16 Jan 09.

**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT**


Deep neural networks (DNNs) have been proven to excel in various domains, from image classification to natural language processing. Despite their remarkable performance, these models face a significant limitation—lack of lifelong learning capabilities, impeding their capacity to assimilate new knowledge while retaining previously acquired information. Addressing this challenge proves even more daunting when adapting these capabilities to resource-constrained edge platforms with stringent size, weight, area, and power (SWaP) constraints. This project is dedicated to crafting a resilient lifelong learning agent tailored for operation in such constrained environments. Two state-of-the-art hardware-friendly lifelong learning models, TACOS and MetaplasticNet, were developed, drawing inspiration from neuro-inspired mechanisms like Metaplasticity and synaptic consolidation. In tandem with algorithmic innovations, we introduced two cutting-edge CMOS/Memristor hybrid accelerators—Genesis-v1 and Genesis-v2. These accelerators not only execute lifelong learning tasks in real-time at approximately 20-30 frames per second but also operate with exceptional efficiency, consuming a minimal power of 20mW. This integrated approach pushes the boundaries of lifelong learning in DNNs and provides a robust solution for on-device learning in resource-constrained environments.



**15. SUBJECT TERMS**

Continual Learning, Memristor Crossbar, Metaplasticity, Spiking Neural Networks, Mixed Signal Design, On-Device Learning


| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES |
|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | C. THIS PAGE | | |
| U | U | U | SAR | 71 |

| 19a. NAME OF RESPONSIBLE PERSON | 19b. PHONE NUMBER *(Include area code)* |
|---|---|
| CLARE D. THIEM | N/A |



PREVIOUS EDITION IS OBSOLETE.

## TABLE OF CONTENTS

**SECTION** **PAGE**

**LIST OF FIGURES**

**LIST OF TABLES**

## LIST OF EQUATIONS

## 1.0 SUMMARY

Deep neural networks (DNNs) demonstrate exceptional performance across a variety of domains, including image classification and natural language processing. However, these models lack lifelong learning capabilities, which restricts their ability to acquire new knowledge while preserving old knowledge. Lifelong or continual learning is inherently challenging, and realizing these capabilities in resource-constrained edge platforms poses additional challenges. These devices have stringent size, weight, area, and power (SWaP) budgets. This translates to the constraint that the intelligent agent must continually learn in a scenario where it encounters samples only once, as samples cannot be stored on-chip. This challenging scenario is known as online continual learning [1]. Moreover, lifelong learning requires additional memory and computational overhead, which are challenging to accommodate within the SWaP constraints of edge platforms.

Our work focused on bridging the gap between models with lifelong learning capabilities and their practical integration on resource-constrained platforms. In this effort, we developed (i) state-of-the-art (SOTA) models with lifelong learning capabilities that are resource-efficient and (ii) novel accelerators that can accommodate lifelong learning with energy-efficiency. We developed two novel continual learning models, namely, Task Agnostic Continual Learning in Spiking Neural Networks (TACOS) and MetaplasticNet, which draw inspiration from neuro-inspired mechanisms, particularly activity-dependent metaplasticity and synaptic consolidation, to address catastrophic forgetting effectively. Our approach, TACOS, is the first supervised spike-based continual learning approach in literature. It achieves near-SOTA mean accuracy in two continual learning benchmarks in the challenging online continual learning scenario, while requiring lower memory and computational overhead compared to contemporary work. We also designed lifelong learning solutions that can leverage on-device computation with memristors for energy-efficient lifelong learning accelerators. The developed online continual learning framework utilizes probabilistic metaplasticity which can accommodate non-ideal characteristics of memristor devices. Our results demonstrated near-SOTA performance on two continual learning benchmarks with low-precision (6-bit) weights realized with memristor devices. It also reduces memory overhead by up to ~ 35% compared to the contemporary literature.

Alongside the algorithmic advances, we developed the first-ever lifelong learning accelerator with online learning capability. In this effort, two variants of the accelerator were developed. Accelerators address catastrophic forgetting through spiking local learning models that are optimized for energy efficiency. Specifically, we designed two mixed-signal Complementary Metal-Oxide-Semiconductor (CMOS)/memristor hybrid lifelong learning accelerator chips, Genesis-v1 and Genesis-v2, and an FPGA-based digital accelerator.

The first chip, Genesis-v1 was designed to efficiently perform lifelong learning on binary sequential classification tasks. It supports activity-dependent metaplasticity for lifelong learning, where neurons in the network are assigned a metaplasticity parameter to modulate the plasticity of the weights connected to them. The architecture features a digital block operating synchronously with a memristor crossbar design. The digital block realizes the full lifelong learning network, and the memristor crossbar design interfaces with it to realize the output layer of the network. We leveraged the sparse event-driven computations of the spiking networks and Address Event Representation (AER) for efficient data communication. The digital block features a one-dimensional systolic array of 25 Processing Elements (PEs), equipped with 16-bit fixed-point compute units. The PEs in the accelerator take advantage of the energy efficiency offered by the custom-designed Leaky

Integrate and Fire (LIF) neuron module and the weight update module. The chip stores the network parameters in an on-chip distributed memory of 102 kB. The digital accelerator consumes an average power of 20 mW at 10 MHz frequency. Genesis-v1 also features four $50 \times 7$ crossbar arrays, which interface with its digital counterpart for input spike streams. The crossbar outputs are processed in the on-chip analog Leaky Integrate and Fire (LIF) neuron units, which support different levels of network activity. The crossbar emulates multi-memristor weights with three 1 Transistor 1 Resistor (1T1R) cells, which were evaluated on continual learning benchmarks. The area footprint of the accelerator is 21.62 $mm^2$ which was taped out in the IBM 65nm technology node.

The second accelerator chip, Genesis-v2, was enhanced from the first version to provide greater scalability and reconfigurability. It comprises a digital accelerator and a crossbar assembly, designed to support online continual learning with activity-dependent metaplasticity. The continual learning mechanism on this chip assigns metaplasticity parameters to weights instead of neurons, which enhances the learning ability compared to the model adopted in the previous chip. This design choice led to a larger on-chip memory of 512 kB. We optimized the digital accelerator for higher throughput, with 64 parallel PEs arranged in a 2D array. The design also supports configuring the network structure and parameters, with support for up to eight output neurons. The digital accelerator operates at an average power of 17.1 mW at 10 MHz operating frequency. Genesis-v2 is also equipped with a simpler crossbar assembly with $8 \times 8$ crossbar units, which can emulate weights with precision up to 6 bits. Genesis-v2 has an area footprint of 23.28 mm2. It is currently being fabricated in the IBM 65nm technology node. We also developed an FPGA-based lifelong learning accelerator on the Xilinx ZYNQ-7000 Field Programmable Gate Array (FPGA) platform. We investigated the efficacy of different numerical formats on the performance of continual learning and proposed an 8-bit dual fixed-point format that matches the distribution of network parameters and gradients. With this optimization, the accelerator maximizes continual learning performance at low memory overhead and processes 30 samples of $16 \times 16$ images per second with a 10 MHz operating frequency.

Our work represents a significant step forward in addressing the critical challenge of lifelong learning on resource-constrained platforms, offering promising solutions to achieve lifelong learning effectively in real-world applications where adaptability and knowledge retention are paramount.

## 2.0 INTRODUCTION

When artificial neural networks are presented with new learning tasks, they are prone to forgetting previously acquired knowledge. This problem, known as catastrophic interference (or catastrophic forgetting), was first identified in 1989 [2] and continues to be a challenge in the field of artificial intelligence [3]. By contrast, humans and other mammals do not seem to suffer from this problem. Neuroscientists and machine learning researchers have sought to understand how biological neural systems can avoid catastrophic interference, and several explanations have been proposed. Machine learning research often adopts such neuroscience-inspired mechanisms to alleviate catastrophic forgetting in networks. Although algorithmic progress has achieved varying degrees of success in achieving lifelong learning, designing lifelong learning machines remains difficult. Algorithmic-level innovations are important to address this problem. However, to be of practical value, lifelong learning models must often be deployed on physical hardware at the edge, under strict size, weight, area, and power (SWaP) constraints. The

solutions deployed on these platforms also need to demonstrate online continual learning since the SWaP constraints do not allow the storage of samples for training indefinitely. Hence achieving effective lifelong learning requires efforts on multiple fronts: (i) designing lifelong learning algorithms which are compute- and memory-efficient to be suited to such edge applications, and (ii) designing accelerators which can support lifelong learning abilities.

A crucial consideration when designing lifelong or continual learning accelerators (in this report, we use lifelong learning and continual learning interchangeably) is memory optimization. Off-chip memory access can account for more than 80% of total energy consumption during inference [4]. Consequently, during training the overhead is more pronounced, since calculating gradients can require up to 10× as much memory access as updating the weights [5]. Lifelong learning methods that mitigate catastrophic forgetting naturally exacerbate memory overhead during training, as they require additional mechanisms to deter catastrophic forgetting. This inevitably strains the energy budget of a lifelong learning accelerator. A potential solution to address this issue is to use novel non-volatile memory devices such as memristors for lifelong learning on the edge. These non-volatile devices can emulate weights in a network and perform vector-matrix multiplication in memory, which can lead to up to two orders of magnitude gains in energy efficiency compared to conventional platforms [6]. These devices also offer a compact form factor which is suitable for realizing high-density on-chip memory. These features pose memristors as a suitable substrate for lifelong learning on energy-constrained platforms. However, these devices also exhibit characteristics such as low precision and inherent variability, which impede high performance in lifelong learning applications. During the project, we investigated innovative mechanisms that can be implemented on this substrate for energy-efficient continual learning.

Our efforts in this project were two-pronged, (i) developing novel energy-efficient lifelong learning algorithms and (ii) designing neuromorphic lifelong learning accelerators suited to energy-constrained platforms. In our efforts to develop novel lifelong learning algorithms with low compute and memory overhead, we developed MetapalsticNet, which adopts binary probabilistic metaplastic synapses, embedded in a random projection-based network ensemble. MetaplasticNet utilizes activity-dependent plasticity, or metaplasticity, which stabilizes important weights in the network to reduce catastrophic forgetting. The network adopts binarized weights and activations that lead to training requiring energy-efficient computations and lower memory overhead, achieving up to ∼ 53× lower power consumption per weight update compared to the previous model of [7]. We also proposed TACOS [8] algorithm, which leverages neuro-inspired mechanisms such as synaptic consolidation and metaplasticity to mitigate catastrophic interference in a spiking neural network, using only local information from synapses. It is the first spike-based supervised continual learning model in literature. The model can perform continual learning without task awareness and with a fixed memory size that does not need to be increased when training on new tasks. It combines neuromodulation with complex synaptic dynamics to enable new learning while protecting previous information. We evaluated TACOS in sequential image recognition tasks and demonstrated its effectiveness in reducing catastrophic interference in the challenging online continual learning scenario. Our results showed that TACOS outperforms existing regularization techniques with lower memory overhead. Along the lines of this effort, we also designed lifelong learning algorithms targeted towards memristor devices. While these devices offer compute-in-memory capability which can greatly benefit energy-efficiency, these devices also exhibit

characteristics such as low precision and inherent variability. These non-ideal characteristics greatly impede high performance in lifelong learning applications [9]. We developed an online continual learning framework that uses probabilistic metaplasticity for low-precision (6-bit) weights realized with memristor devices. We proposed a novel mechanism that stabilizes important weights by tuning the probability of weight update rather than the magnitude of weight update, which makes it compatible with low-precision weights. The proposed algorithm shows near SOTA performance on two continual learning benchmarks while reducing memory overhead by ~ 35%.

The second prong of our efforts was to design neuromorphic accelerators with on-device online lifelong learning capability. To accommodate the computational and memory overhead of lifelong learning algorithms in resource-constrained platforms, we explored innovative optimizations such as linear metaplastic functions and dual-fixed point numbers. We designed and fabricated two mixed-signal hybrid CMOS/memristor accelerators, Genesis-v1 and Genesis-v2, which enable real-time online continual learning leveraging our lifelong learning solutions. We also developed a digital lifelong learning accelerator on the Xilinx ZYNQ-7000 FPGA platform.

The first version of our mixed-signal accelerator, Genesis-v1, can accommodate lifelong learning with activity-dependent metaplasticity assigned to neurons in the network. It features a multi-core digital architecture with 25 PEs arranged in a one-dimensional systolic array with on-chip training. The accelerator exploits sparse activity in spike-based networks and uses AER for efficient data communication. The accelerator has 102 kB of distributed static random-access memory (SRAM) on the chip to store network parameters. The chip also features a mixed-signal design that interfaces with the digital design and realizes the continual learning network partially with a modular crossbar architecture and on-chip analog leaky-integrate and fire neurons. Each individual crossbar module was sized as 50 × 7 array to realize multi-memristor synapses with three 1T1R cells. This synaptic weight configuration was also evaluated in lifelong learning setting. The Genesis-v1 chip was fabricated at State University of New York at Albany, College of Nanotechnology, Science and Engineering, using a hybrid IBM 65nm technology node.

Our second lifelong learning accelerator Genesis-v2 is an improved version of its predecessor. It features a higher level of user-configurability and modularity. This version of the accelerator was designed for greater lifelong learning ability with activity-dependent metaplasticity parameters assigned to individual synapses instead of neurons. We also enhanced the scale of the design with larger on-chip memory and systolic array with 64 parallel processing elements, which results in reduced computation latency. The design integrates a pipelined Leaky Integrate and Fire (LIF) neuron to enhance the throughput of the accelerator. Genesis-v2 is also equipped with a crossbar assembly of four 8 × 8 crossbar modules. Each row of the module can emulate multi-memristor synapses with seven 1T1R cells. Our simulation results indicate that this configuration leads to a precision equivalent to ~ 6-bits. The chip was also taped out in the hybrid IBM 65 nm technology node.

In Section 3.1, we cover the fundamentals of lifelong learning and the methods devised to mitigate catastrophic forgetting. In Section 3.2, we elaborate on key capabilities for lifelong learning accelerators. Section 3.3 details our lifelong learning algorithms and their evaluations. Section 4.1 describes the Genesis-v1 chip development and Section 4.2 contains the digital accelerator on FPGA platform. Section 4.3 elaborates the improvements incorporated in Genesis-v2, the probabilistic continual learning algorithm, and the design of the accelerator. Section 4.4 follows showing our work on semi-random deep neural networks.

**Table 1. List of Tasks for the Genesis Project**

| Task | Status |
|---|---|
| Development of software interface for the design flow | Completed |
| Custom lifelong learning algorithm framework development | Completed |
| Incorporation of hardware constraints to the algorithm | Completed |
| Development of the training circuitry | Completed |
| Analysis and optimization of the network architecture | Completed |
| Circuit design of Genesis | Completed |
| Physical layout of Genesis - v1 | Completed |
| Development of built-in self-test circuitry for the chip | Completed |
| Genesis-v1 tapeout | Completed |
| FPGA design of golden model | Completed |
| Evaluation and test framework for post-Silicon analysis of Genesis | In progress |
| Noise, timing and power analysis | Completed |
| Optimizing the physical layout of Genesis-v2 | Completed |
| Characterization of Genesis chip | Completed |
| Identification of the application platform | Completed |
| Benchmark results of Genesis | Completed |
| Genesis-v2 tapeout | Completed |

**Table 2. Physical Design Library Development Tasks for Continual Learning**

| Task | Status |
|---|---|
| Standard cells schematic design | Completed |
| Functional verification of standard cells | Completed |
| Physical layout of standard cells | Completed |
| Post layout verification | Completed |
| Standard cell characterization | Completed |
| SRAM cell optimization | Completed |
| 1T1R cell optimization | Completed |
| Standard cell optimization | Completed |

## 3.0 METHODS ASSUMPTIONS AND PROCEDURES

### 3.1 Fundamentals of Lifelong Learning

The term lifelong learning refers to a system's ability to autonomously operate in, interact with, and learn from its environment [10, 11, 12]. This requires the system to be able to improve its performance through the acquisition of new knowledge and learning to operate under energy and resource constraints. More specifically, a lifelong learning system needs to function in dynamic and noisy environments, rapidly adapt to novel situations, minimize forgetting when learning new tasks, transfer information between tasks, and operate without explicit task identification.

Several learning paradigms have been involved in the process of arriving at this definition of lifelong learning. The concept includes transfer learning in which the goal was to recycle/reuse the learned representations for other tasks [13]. This was followed by multi-task learning (MTL), which had the goal of improving generalization by leveraging domain-specific information in related tasks [14]; however, in this setting, tasks are trained jointly and not sequentially. Transfer learning and MTL evolved into few-shot learning [15], in which the goal is to learn from a limited number of examples with supervised information in the target domain. This drives the approach towards learning to learn (also called meta-learning), where a system learns to optimize the objective on its own; an

approach that aims to achieve the rapid, general adaptation that biological brains can offer [16]. In tandem, studies on biological brains have shown that there is a combination of learning mechanisms that support lifelong learning, rather than a single one [17]. Drawing from this and more, researchers consider that several of these learning paradigms constitute a subset of lifelong learning.

Methods have been devised to address different aspects of lifelong learning. These methods are synaptic consolidation, dynamic architectures, and replay (Figure 1).

**Synaptic consolidation** is a way to preserve synaptic parameters when learning new tasks. The most common method is to add regularization terms to the loss function to maintain previous synaptic strengths or neural activations [18,19]. Another approach is to use more complex synapse models with memory-preserving mechanisms such as metaplasticity [20], multiple weight components operating on different timescales [8], or probabilistic synapses [21].

**Dynamic architectures** expand the capacity of the network to solve a particular task, using top-down control of resources. Expanding the capacity of the network takes different forms, including periodic additions of new neurons or addition of entire new networks or layers dedicated to solving new tasks [22,23]. Another approach is to dynamically gate the network components according to the task identity. This is often achieved by means of a task oracle (that is, an autoencoder capable of separating specific classes and tasks) or by meta-learning a gating function [24].

**Replay** involves the presentation of samples representative of previously learned tasks interleaved with samples from the task currently being trained. Replay helps bring the training data closer to being independent and identically distributed, as would be the case if all tasks had been trained jointly. This allows the network to be trained to maximize performance across all tasks (as opposed to only learning the current task) and to learn inter-task boundaries. To replay data in neural networks, previous training samples (or internal representations) are encoded, stored, and recalled from a memory buffer or recreated by a generative model [25, 26, 27]. Methods are evolving to use more sophisticated algorithms for both the selection of samples to be stored in the replay buffer and for the selection of samples to be replayed from the buffer [28].

Several machine learning models have been developed to address lifelong learning, with a heavy emphasis on the catastrophic forgetting problem using the methods discussed above, as well as some hybrid models that combine features from two or more of them. Increasingly, Artificial Intelligence (AI) researchers are taking inspiration from discoveries in neuroscience that help explain how biological organisms are able to learn continually [17]. In our research, we have developed models that adopt neuroscience-inspired synaptic consolidation methods to achieve lifelong learning.

### 3.2 Key Capabilities of Lifelong Learning Accelerators

Recent research on synaptic consolidation [8, 18, 20, 21], dynamic architectures [22, 23] and replay methods [25, 27] show promise in addressing aspects of lifelong learning. By studying these methods, we have assembled a list of six desirable capabilities for AI accelerators: on-device learning, resource reassignment within budget, model recoverability, synaptic consolidation, structural plasticity, and replay memory.

The first three capabilities are general ones that apply to any lifelong learning method; the last three are specific to one or more of the lifelong learning methods discussed above. The relevance of each capability to a specific design will depend on the type of accelerator and the target application.

### 3.2.1 On-Device Learning

A lifelong learning system needs to continuously learn from non-stationary data distributions over varying time-scales [29]. Since the system must learn in real-time, on-device learning is crucial to avoid latency associated with data transmission to the cloud. Generally, for on-device learning, design considerations are required i) when batching data of large samples on limited memory resources and ii) when storing and mapping a large pool of intermediate model parameters in compact formats to minimize data movement cost. While translating to the context of lifelong learning, additional challenges arise: optimizing complex computations observed in consolidation methods so that the energy cost is reduced and minimizing the latency to access previous samples in replay methods.

### 3.2.2 Resource Reassignment within Budget

Lifelong learning systems must have the ability to reallocate or distribute resources within a SWaP budget at runtime, and quite often this could entail different granularities. This suggests that the system must be capable of parametric, neuronal or memory reassignments during its lifetime [30]. This is a challenging requirement — especially when hardware is tightly optimized for one method, model, or functionality. Furthermore, to allocate resources at fine granularity one needs to identify points of operation that are pareto-optimal, while ensuring that the architecture remains flexible to different tasks. Additional challenges may arise when the size of the input and output layer differs between tasks [31, 32], such as accommodating the model expansion on the fly and/or the inability to encode and store data within a fixed size.

### 3.2.3 Model Recoverability

One challenge in developing a lifelong learning system is to establish confidence in the model, and more so when the system is changing autonomously [33]. In a software environment, it is possible to checkpoint a model's state, preserving the overall model or maintaining a history of updates. This tracking provides a valuable record of a model's past states that can prove essential for either diagnostic purposes or for reverting to a previous state (if an online update led to failure). Several of the design choices that make AI accelerators more efficient make checkpointing a model's configuration dynamically impractical, if not impossible.

### 3.2.4 Synaptic Consolidation

Models incorporating synaptic consolidation typically use multiple internal states to learn at different timescales using several loss functions, probabilistic synapses, reference or target weight values or other synaptic states in addition to magnitude (*i.e.*, metaplastic state, consolidation tag). Each of these consolidation mechanisms entails auxiliary information stored on-device and additional operations performed during any learning process. In general, for consolidation mechanisms, a lifelong learning accelerator needs to store and associate auxiliary information with specific synapses/neurons and support custom loss functions.

### 3.2.5 Structural Plasticity

Structural plasticity relates to physical changes in the model, including the addition/removal of synapses (e.g. synaptogenesis or synaptic pruning) or of neurons (e.g. neurogenesis/neural pruning), gating and attention mechanisms, and mixture of experts [23, 34, 35]. While static allocation of pools of neurons and/or synapses is possible at design time, it is still challenging to model and train such highly dynamic architectures. The underlying accelerators should support fine-grain runtime reconfigurability to add or reallocate a new pool of memory and computational resources. This problem is exacerbated when reallocation has to occur under a limited SWaP budget.

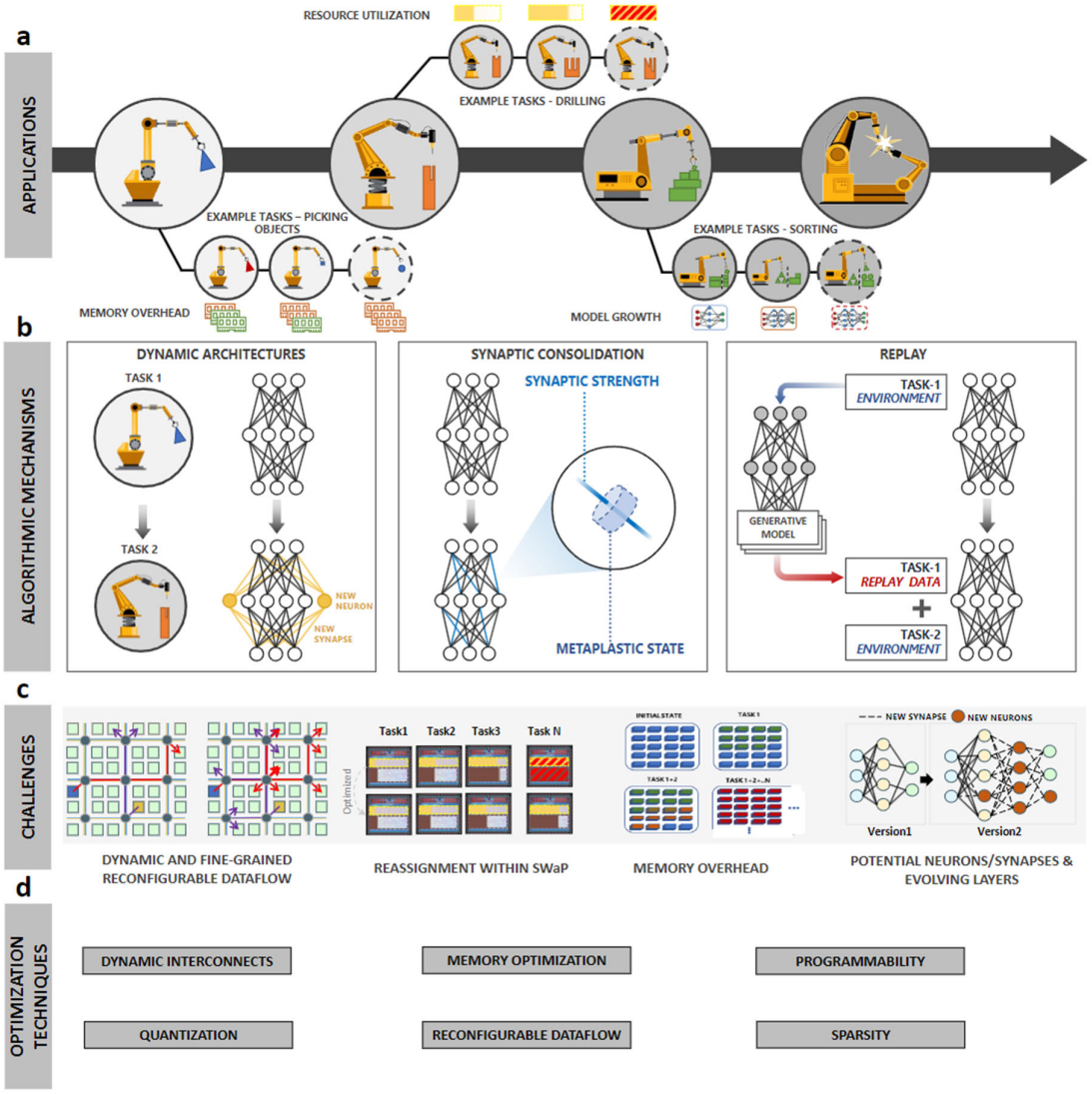



**Figure 1. Addressing Lifelong Learning in AI Systems – (a) Applications, (b) Algorithmic Mechanisms, (c) Hardware Challenges, (d) Optimization Techniques**

### 3.2.6 Replay Memory

Replay mechanisms might require a continuously growing memory as the model learns new tasks. Replay comes in two varieties, known as waking and sleeping replay, both of which need to be supported by accelerators. In waking replay, to prevent forgetting previously learned tasks, selected samples representative of the earlier tasks are interleaved while training a new task. During sleep replay, the system rehearses only samples from previous tasks to consolidate knowledge. Memory access during sleep replay can be at a slower clock cycle, since the model does not need to respond to real-time stimuli. Memory storage can be off-chip. Waking replay, on the other hand, requires on-chip buffers that can update the system state at a faster rate without disrupting learning on the current task. Overall, memory storage and access patterns for replay mechanisms are of mixed latency and are distributed.

The current generation of accelerators includes some highly programmable devices, but there are not any that support the full set of capabilities listed above. In this project, we have developed lifelong learning accelerators that include on-device learning and synaptic consolidation. The following sections detail the lifelong learning algorithm and accelerator design and their evaluation.

## 3.3 Lifelong Learning Algorithm Development

### 3.3.1 Problem Formulation

Continual learning is the ability to learn tasks sequentially, without suffering severe performance loss on previously learned tasks when new ones are learned. Formally, we define a task, $T^t$, to be a set $\{X^t, Y^t\}$ of ordered pairs of input data points and their corresponding class labels. The continual learning problem is formulated as maximizing the performance of a system across all tasks, $T$, when trained sequentially.

### 3.3.2 MetaplasticNet

We developed MetaplasticNet, which is a continual learning network incorporating probabilistic metaplastic synapses with discrete metastates. The network uses low-precision weights which can be trained using compute-lite operations. The following sections describe the details of the network architecture and evaluation results.

#### 3.3.2.1 Probabilistic Metaplastic Synapse Model

A key component of MetaplasticNet is the probabilistic metaplastic synapse. It is a binary parameter that can only assume efficacy values 0 or 1, but it has an associated hidden metaplastic variable representing the metalevel of the synapse. When a synapse is faced with repeated similar plasticity events (potentiation or depression), it transits to a higher metalevel, which is more stable, and the synapse is less likely to change its weight value. A synapse with n synapses can be in 2n metastates, as shown in Figure 2 for n = 4. The high (1) and low (0) efficacies are shown with pink and blue. The red and blue arrows show the transition due to potentiation and depression plasticity events respectively.

In the cascade model proposed by Fusi et. al. [36] (shown in Figure 2(a)), the synaptic transitions are probabilistic. In response to stimuli, a synapse can transit from metalevel $i$ to the next higher metalevel of the same efficacy with probability $p_i$, or to the lowest metalevel of the opposite efficacy with probability $d_i$. To ensure that the synapse is less

plastic at higher metalevels, the transition probabilities have lower values for higher values of $i$. On the contrary, in the serial model [37] (Fig. 2(b)), the metastates are connected in a serial manner and the transitions are deterministic. Thus, in the serial model, the greater synaptic stability of higher states is achieved by the greater number of similar successive plasticity events required to cause a change in efficacy. Although the cascade and serial models perform reasonably well when subjected to random plasticity events [36, 37], we found that their memory retention in continual learning tasks was poor, see Table 3 and Table 4.

We define the probabilistic metaplastic synapse model with state-dependent transition probabilities to introduce greater stability. The transition dynamics of the synapse is a combination of serial connectivity with probabilistic transitions (Figure 2(c)). Let, $p_{pot}$ and $p_{dep}$ are base probabilities of transition to higher and lower metalevels, respectively. The transition probabilities are defined such that, at metalevel i, a synapse will transition to metastate i+1 with probability $p_i = (p^i_{pot})_{\{0\leq i<n-1\}}$, and to metastate i-1 with probability $d_i = (p^i_{dep})_{\{0\leq i<n-1\}}$, where $n$ is the number of metalevels. As a result, at metastate $i$, on

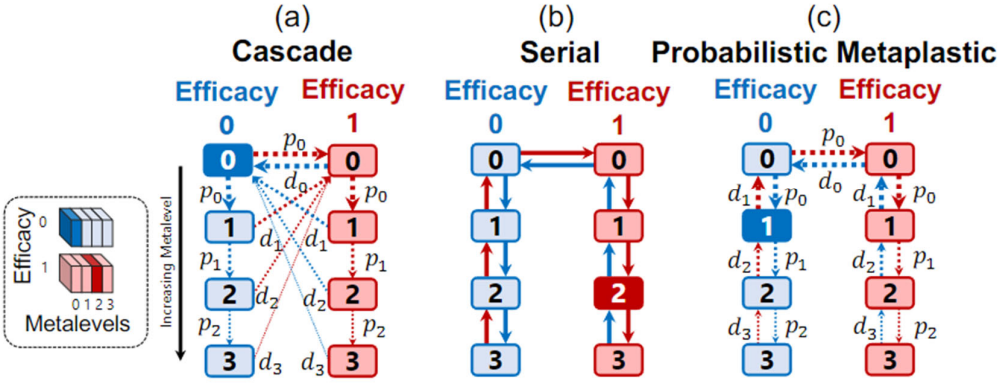



**Figure 2. Transition Dynamics of the (a) Cascade [1], (b) Serial [2], and Proposed (c) Probabilistic Metaplastic Models**

average $\Sigma(1/d_j)_{\{0\leq j\leq i\}}$ events are required, which is greater than the cascade model with similar state-dependent probabilistic transition and the serial model, which results in a higher retention of information.

### 3.3.2.2 Network Architecture and Training Methodology

The proposed two-layer network, MetaplasticNet, integrates the probabilistic metaplastic synapse in its output layer. The network architecture is shown in Figure 3. The hidden layer of the network employs fixed random binary weights to project the input to a high-dimensional space. The output layer consists of an ensemble of classifiers which are fully connected to the hidden layer neurons. The hidden layer activations are connected to each classifier through probabilistic metaplastic synapses and constant inhibitory weights. The prediction of the network is decided through a majority unit based on the majority classified label.

The hidden layer employs a Heaviside step function ($H$) to binarize the input projections. Since the hidden weights and the activations are binary, multiplication operations can be replaced with addition/subtraction which significantly reduces energy consumption [38]. The output layer classifiers in the ensemble are independently trained

using a stochastic learning rule proposed in [39]. The training algorithm maintains that the mean pre-activation at the McCulloch-Pitts output neurons with label 1 should be above its threshold by a margin, otherwise the prediction is considered erroneous with error $e = 1$. For the output neurons with label 0, the mean pre-activation should be below the threshold by a margin, otherwise the error $e$ at that neuron is set to $-1$. The margin around the threshold reduces overfitting in the network. Given the error signal, the probabilistic metaplastic synapse is updated according to Algorithm 1. Probabilistic transitions in the synapse are induced by comparing a random number drawn from a uniform distribution to the state-dependent transition probability of the weight. The weights are updated with an increment/decrement operation, which leads to power-efficient training.

Finally, during inference, the global prediction of the ensemble $y_{global}$ is determined by majority function; each network votes by predicting a class and the class with the most votes represents the ultimate output of the model.

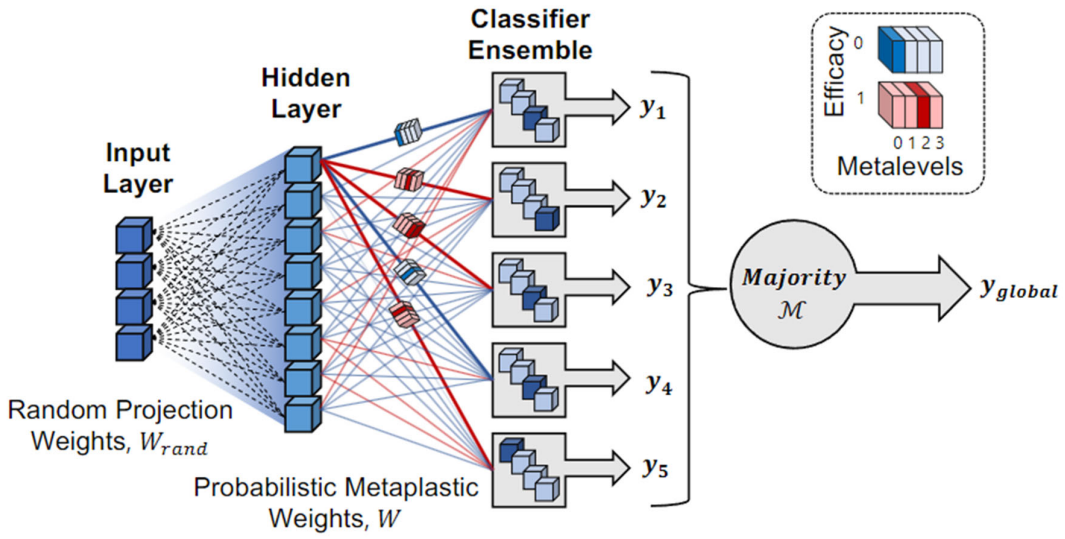



**Figure 3. Network Architecture in MetaplasticNet with Probabilistic Meta-plastic Synapses**

### 3.3.2.3 Results

**Continual Learning Evaluation:** We test MetaplasticNet on two continual learning benchmarks, the Split-Modified National Institute of Standards and Technology (MNIST) [40] task and the MNIST-Fashion MNIST [41] sequential image classification task in domain-incremental learning scenario [42, 43]. The domain incremental scenario is a challenging continual learning set-up where the sequential tasks share the same output layer and during training and testing, the network is not given information about the task identity. In both continual learning benchmarks, the network is trained on a sequence of tasks. After sequential training, we observe the mean accuracy across all tasks to assess the continual learning ability of the network. The split-MNIST benchmark comprises five tasks, each containing two classes of the MNIST dataset. The MNIST-Fashion MNIST sequential task comprises two tasks where the network is first trained on the MNIST dataset followed by the Fashion-MNIST dataset. We evaluated MetaplasticNet configured with 2000 hidden neurons and an ensemble of five classifiers. The cascade and serial synapse models are also evaluated in the same network architecture. All discrete

metaplastic synapses are considered to have 16 metalevels. We also compared MetaplasticNet with Binarized Neural Network (BNN) [44] and metaplastic BNN [7] that were set to have a single hidden layer of the same size as MetaplasticNet.

Table 3 shows that the MetaplasticNet has ~ 18% mean accuracy improvement across tasks over serial and cascade models, while also outperforming BNN and metaplastic BNN. We observe that MetaplasticNet shows better retention of older tasks with an overall higher mean accuracy. In the MNIST-Fashion MNIST sequential task (Table 4), our network improves the mean accuracies of the other discrete metaplastic synapses by ~ 19%, while performing similarly to the metaplastic BNN. We also simulated MetaplasticNet digital architecture with mixed-precision (inputs: 8-bit; accumulator: 16-bit, hidden weights/output: 1-bit, output weights: 5-bit). We observed 3.47% and 0.92% degradation in the mean accuracy for Split-MNIST and MNIST-Fashion MNIST sequential tasks, respectively, due to the low precision fixed-point representation of the network inputs and accumulations.

**Table 3. Individual and Mean Task Accuracies with Standard Deviations of Different Binarized Networks on the Split-MNIST Task**

| Task | BNN [45] | Metaplastic BNN[9] | Serial [2] | Cascade [1] | MetaplasticNet | MetaplasticNet *(Digital)* |
|---|---|---|---|---|---|---|
| Class 0,1 | 27.62% ± 8.97 | 50.43% ± 4.07 | 37.69% ± 0.72 | 37.34% ± 1.78 | 53.49% ± 2.91 | **58.74%± 2.89** |
| Class 2,3 | 58.39% ± 4.03 | 75.28% ± 2.46 | 52.42% ± 0.69 | 55.88% ± 0.60 | **92.89%** ± **1.24** | 92.074%± 0.45 |
| Class 4,5 | 12.01% ± 2.94 | 54.93% ± 2.46 | 8.61% ± 0.69 | 17.77% ± 2.12 | **79.79%** ± **2.36** | 70.03%± 1.44 |
| Class 6,7 | 82.81% ± 4.95 | 94.25% ± 0.73 | 78.51% ± 1.94 | 91.77%± 0.83 | 94.91% ± 0.39 | **96.13%± 0.09** |
| Class 8,9 | 97.43% ± 0.65 | 87.53% ± 1.05 | **97.87%** ± **0.29** | 95.85% ± 0.17 | 67.48% ± 1.29 | 74.05%± *1.37* |
| Mean | 55.64% ± 1.84 | 72.49% ± 1.64 | 55.02% ± 0.64 | 59.72%± 0.21 | **78.20%** ± **0.57** | **78.20%± 0.57** |

**Computational Analysis:** MetaplasticNet is computationally efficient compared to other gradient-based approaches, including BNNs [45] and metaplastic BNN [9]. This can be attributed to: (i) Lower asymptotic time and space complexities: Considering $i$ = number of input neurons, $h$ = number of hidden neurons, $o$ = number of output neurons, $N$ = number of samples, M = number of ensembles, the time and space complexities of our model scale in O(|M|$Nho$) and O(|M|$ho$) respectively, whereas the complexities of BNN and metaplastic BNN scale in O($N(ih + ho)$) and O($ih + ho$). (ii) Replacing multiplications with simple accumulations: The weight update in our model requires a fixed-point comparison with a value generated by a linear feedback shift register (LFSR), and an integer increment/decrement operation. However, the metaplastic BNN requires four multiplications and two subtractions for the same. When comparing the average power consumption for a single weight update operation, our approach consumes ~ 53× lower power than metaplastic BNN with 16-bit fixed-point precision implemented in 32nm node (34.51 $\mu W$ for our model vs. 1844.18 $\mu W$ for the latter).

### 3.3.3 TACOS

Our developed continual learning algorithm, "TACOS: Task Agnostic Continual Learning in Spiking Neural Networks" [46], is targeted toward spike-based networks. TACOS uses neuro-inspired synapse-local mechanisms (metaplasticity and synaptic consolidation) to reduce catastrophic forgetting in spiking neural networks in task-agnostic continual learning settings. The publication based on this work was published in the International Conference on Machine Learning (ICML) 2021 workshop on Theory and Foundation of Continual Learning.

### 3.3.3.1 Training Methodology & Continual Learning Mechanisms

**Preliminaries:** We describe the TACOS framework for continual learning in Spiking Neural Networks (SNN) in discrete time. The basic building block is a spiking neuron model, LIF neuron, with dynamics described by

$$V(t+1) = V(t) + \frac{\Delta t}{T_{mem}} \left( \left( V_{rest} - V(t) \right) + I(t)R \right)$$

**Equation 1. LIF Neuron Voltage Integration**

where $t$ is the simulation time step counter, $\Delta t$ is the length of a time step, the membrane potential $V(t)$ integrates synaptic current $I(t)$ over time, $R$ is the membrane resistance, and $T_{mem}$ is the membrane time constant which controls the rates of integration and decay.

**Algorithm 1:** Weight update for probabilistic metaplastic model in Metaplastic-Net

**Input** : $\mathbf{W}$, $\mathbb{E}$
**Require**: $w_{min}, w_{max}, p_{pot}, p_{dep}, \eta$
**Procedure** `WeightUpdate(W, E)`
  $\mathbf{W}_{\mathbf{meta}} = abs(\mathbf{W}) - H(-H(\mathbf{W}))$
  **foreach** $e \in \mathbb{E}$ **do**
    `# Random subset of activations (size η%)`
    $N_{update} \sim \binom{N_{active}}{\eta |N_{active}|}$
    **for** $n \in N_{update}$ **do**
      $\epsilon \sim uniform(0,1)$
      **if** $H(\boldsymbol{W}(e,n)) == 1$ **then**
        **if** $\epsilon < p_{pot}^{\mathbf{W}_{\mathbf{meta}}(e,n)}$ ***and*** $e > 0$ **then**
          $\mathbf{W}(e,n)$ += 1
        **end**
        **if** $\epsilon < p_{dep}^{\mathbf{W}_{\mathbf{meta}}(e,n)}$ ***and*** $e < 0$ **then**
          $\mathbf{W}(e,n)$ -= 1
        **end**
      **else**
        **if** $\epsilon < p_{dep}^{\mathbf{W}_{\mathbf{meta}}(e,n)}$ ***and*** $e > 0$ **then**
          $\mathbf{W}(e,n)$ += 1
        **end**
        **if** $\epsilon < p_{pot}^{\mathbf{W}_{\mathbf{meta}}(e,n)}$ ***and*** $e < 0$ **then**
          $\mathbf{W}(e,\ n)$ -= 1
        **end**
      **end**
    **end**
    $\mathbf{W} \leftarrow Clip(\mathbf{W}, w_{min}, w_{max})$
  **end**
**Output** : Updated Weights ($\mathbf{W}$)

**Table 4. Individual and Mean Task Accuracies with Standard Deviations of Different Binarized Networks on the MNIST-Fashion MNIST Sequential Task**

| Task | BNN[45] | Metaplastic BNN[9] | Serial[2] | Cascade[1] | MetaplasticNet | MetaplasticNet *(Digital)* |
|---|---|---|---|---|---|---|
| MNIST | **90.06%** ± **1.32** | 89.53% ± 3.62 | 89.60% ± 0.27 | 84.76% ± 0.69 | 89.37% ± 0.08 | 89.30% ± 0.06 |
| FMNIST | **79.73%** ± **1.55** | 73.55% ± 0.05 | 79.69% ± 0.27 | 73.16% ± 0.72 | 72.45% ± 0.24 | 69.18% ± 0.63 |
| MNIST continual | 29.65% ± 1.46 | **57.64%** ± **1.53** | 6.00% ± 0.37 | 9.39% ± 1.16 | 53.47% ± 1.61 | 57.13% ± 1.54 |
| Mean | 54.69% ± 1.00 | **65.60%** ± **0.83** | 42.85% ± 0.25 | 41.28% ± 0.51 | *62.96% ± 0.69* | 63.16% ± 1.00 |

The synaptic current is a summation of the presynaptic action potentials integrated according to:

$$I(t+1) = I(t) + \frac{\Delta t}{T_{syn}}\left(\sum_{j=1}^{N} w_j S_j(t) - I(t)\right)$$

**Equation 2. Current Summation**

At each time step, the synaptic input current, $I(t)$, is incremented by the summation of incoming $N$ presynaptic action potentials, $S_j$, which have the value 1 for neurons that have fired and 0 otherwise, weighted by the synaptic strengths, $w_j$, and decays exponentially towards zero with the synaptic time constant $T_{syn}$.

When the membrane potential crosses a threshold $V_{th}$, an action potential is emitted ($S = 1$) and the membrane potential is reset to zero ($V = 0$). Any neuron that has recently fired undergoes a short time period where its membrane potential is frozen at zero as a simple model of a refractory period. The neural network used in this work is strictly feedforward, as shown in Figure 4. Input is received as spike trains, one per input neuron, generated from input data, e.g. by Poisson or population encoding.

**Surrogate Gradient Learning:** To train the model, we implement a surrogate gradient learning rule known as event-driven Random BackPropagation (eRBP) [47]. eRBP relies on presynaptic firing, postsynaptic surrogate gradients, and error feedback through fixed random weights, approximating backpropagation. In eRBP, each neuron has a second compartment $U$ (in addition to the membrane potential $V$) for integrating the error feedback as described in the following.

At each time step, the difference between the output neurons' spiking activity, $S^{out}$, and a target spiking activity or label, $L$, is used to generate an error current:

$$I^{err}(t) = S^{out}(\mathrm{t}) - \mathrm{L}(\mathrm{t})$$

**Equation 3. Error Current Computation**

**The error current is passed as input to two sets of error-encoding neurons; one set for false positives (neurons that fire when they should not), which receives $I^{err}$, and a second set for false negatives (neurons that do not fire when they should), which receives $-I^{err}$. The two sets of error-encoding neurons are implemented using integrate-and-fire dynamics and emit spikes $S^{fp}$ and $S^{fn}$ when the accumulated error crosses their membrane threshold. The $i^{th}$ output neuron receives the error signal $E^o(t)$ at time t, while the $j^{th}$ hidden neuron in layer $h$ receives the error signal $E^h(t)$:**

$$\mathrm{E}_{\mathrm{i}}^{\mathrm{o}}(\mathrm{t}) = \mathrm{S}_{\mathrm{i}}^{\mathrm{fp}}(\mathrm{t}) - \mathrm{S}_{\mathrm{i}}^{\mathrm{fn}}(\mathrm{t})$$

**Equation 4. Output Layer Error**

$$E_j^h(t) = \sum_{i=1}^{O} \left( w_{i,j}^{fp,h} S_i^{fp}(t) - w_{i,j}^{fn,h} S_i^{fn}(t) \right)$$

**Equation 5. Hidden Layer Error**

where $O$ is the number of output neurons, $w_{i,j}^{fp,h}$ and $w_{i,j}^{fn,h}$ are random feedback weights from the error neurons encoding false positives and false negatives, respectively.

The hidden and output neurons integrate the received error signal in a separate compartment $U$, in a manner similar to the way their membrane potentials integrate forward spikes:

$$U(t+1) = U(t) + \frac{\Delta t}{T_{mem}} \left( \left( -U(t) \right) + E(t)R \right)$$

**Equation 6. Error Accumulation**

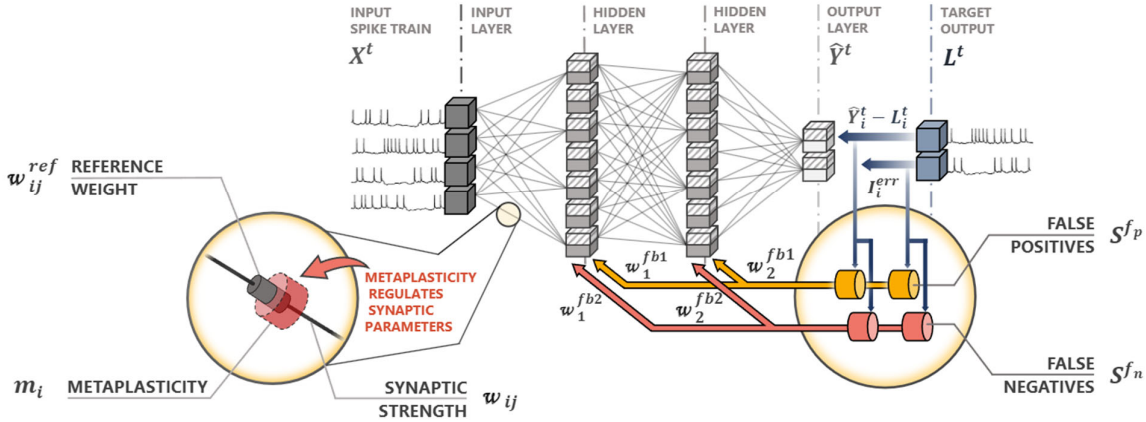



**Figure 4. Supervised Spiking Neural Network Architecture in TACOS**

Whenever a presynaptic neuron fires, the weights of synapses connecting that neuron to postsynaptic neurons are updated, provided the total synaptic current flowing into the postsynaptic neuron falls in a specified interval, $I_{post} \in \{I_{min}, I_{max}\}$. This condition on post-synaptic current is an approximate surrogate gradient for the LIF neuron based on the relation between LIF firing rates and a Rectified Linear Unit (ReLU) activation. The weight update is proportional to the accumulated error in the corresponding post-synaptic neuron's second compartment $U$ (positive or negative):

$$\mathrm{w_{i,j}(t+1) = w_{i,j}(t) - \eta S_j(t) U_i \Theta\left(I_i(t)\right)}$$

**Equation 7. Weight Update**

where $S_j$ is a binary value representing if a neuron has fired, $U_i$ is the accumulated error at the post-synaptic neuron of synapse $w_{i,j}$, and $\Theta(I_i)$ is a boxcar function equal to 1 when $I_{post} \in \{I_{min}, I_{max}\}$ and 0 otherwise.

**Lifelong Learning with eRBP:** To enable continual learning by preserving synaptic knowledge, TACOS uses two mechanisms. The first mechanism, activity-dependent metaplasticity, is inspired by a technique proposed by Laborieux et al. [7], where the plasticity of synapses depends on their weight $w$ and a fixed metaplasticity parameter $m$.

As the magnitude of weight *w* increases, its plasticity decreases, making it more difficult for subsequent training to modify the synapse. The value of the parameter m controls the slope of the function: with higher *m* values, the plasticity diminishes faster. In TACOS, the metaplastic parameter m is a synapse-local variable instead of a global parameter. At the synapse connected to the pre-synaptic neuron *j* and post-synaptic neuron *i*, the metaplastic parameter $m_{ij}$ is updated as follows:

$$\mathrm{m_{ij}(t+1)} = \begin{cases} \mathrm{m_{ij}(t+1)} + \Delta\mathrm{m}, & \text{if } \mathrm{X_j^{tr}} \geq \mathrm{m_{th}^{pre}} \\ \mathrm{m_{ij}(t+1)}, & \text{Otherwise} \end{cases}$$

**Equation 8. Metaplasticity Parameter Update**

Here $\Delta m$ is a fixed increment added to *m* at qualifying synapses determined by $X^{tr}$, the trace of neuron activity which reflects recent firing activity, and the thresholds for presynaptic activity ($\mathrm{m_{pre}^{th}}$) and post-synaptic activity ($\mathrm{m_{post}^{th}}$).

As opposed to using a fixed metaplasticity level that restricts synapses based on magnitude, dynamic metaplastic states allow for greater stability in important synapses without requiring inherently high metaplastic states at initialization, which would significantly slow learning. This is especially important when extending the applications to real-world scenarios. To ensure that the metaplastic parameter does not grow without bounds, a limit is imposed. The parameter *m* modulates the plasticity of the synapse with the following equation:

$$f(m, w) = \exp\left(-abs(mw)\right)$$

**Equation 9. Metaplasticity Function**

The factor is used to modulate the weight update.

The second mechanism to enable continual learning is synaptic consolidation. TACOS uses a two-component synaptic model in which in addition to the actual synaptic weight *w*, each synapse has a hidden slower-changing reference weight $w^{ref}$ that tracks the actual weight according to:

$$w^{ref}(t+1) = w^{ref}(t) + \frac{\Delta t}{T_{ref}}\Big(w(t) - w^{ref}(t)\Big)$$

**Equation 10. Reference Weight Update**

where $T_{ref}$ is the time constant for the evolution of $w^{ref}(t)$. The change in $w^{ref}(t)$ is driven by its difference from $w(t)$. Whenever the postsynaptic neuron spikes, the actual weights of all its inbound synapses drift towards their reference weights through heterosynaptic plasticity according to:

$$\Delta\mathrm{w_{ij}(t+1)} = -\alpha\Big(\mathrm{w_{ij}(t)} - \mathrm{w_{ij}^{ref}(t)}\Big)$$

**Equation 11. Heterosynaptic Decay**

where $\alpha$ is the decay rate. This decay back toward the reference weight, dependent on the postsynaptic activity, acts as a regularizer for neurons that are actively learning. This helps to maintain a balance between current synaptic changes and consolidated knowledge stored in the reference weight.

In summary, while metaplasticity slows the changes in synaptic weight, it does not prevent long-term shifts in synaptic strengths. Synaptic consolidation works with metaplasticity to regulate changes in synaptic values.

#### 3.3.3.2 Experimental Results

**Continual Learning Performance:** TACOS is evaluated on split image classification benchmarks using 5-Split MNIST [48] and Fashion-MNIST [41] datasets under the task agnostic domain-incremental continual learning (Domain-IL) scenario [31], where tasks share the same output layer while the model is unaware of task identity during both training and inference. The network configuration was fixed to 200 neurons per hidden layer (1-2 hidden layers), and a two-neuron output layer. Unlike most models, TACOS only sees each data sample once (i.e., one training epoch), as would be the case in a real-world scenario [49]. To assess the performance of TACOS and other continual learning models, the mean accuracy across all tasks is measured after training on them sequentially. The memory overhead of the different networks is also evaluated.

From Table 5, we can see that TACOS outperforms its similar rate-based counterparts in the Domain-IL scenarios (the maximum accuracies are bolded). TACOS is able to achieve this performance with a memory overhead lower than the other models, except for Learning without Forgetting (referred to as LwF in Table 5) [50] and Stochastic Synapses (referred to as SS in Table 5) [21]. It is, however, important to note that the number of parameters or memory overhead of TACOS does not grow with the number of tasks.

**Plasticity-Stability Trade-off:** We studied the plasticity-stability trade-off by observing learning in three different models. The first model is TACOS, where the maximum metaplastic state is set to different values (Table 6). The second model is where the metaplastic state is fixed as proposed in [7] with synaptic consolidation (referred to as Fixed in Table 6). The third and last model is the baseline SNN with no metaplasticity or consolidation incorporated.

**Table 5. Mean Task Accuracies and Memory Overhead of TACOS and Other Algorithms in Split-MNIST and Split-Fashion MNIST (FMNIST) Tasks**

| Model | FMNIST(%) | MNIST(%) | Memory Overhead |
|---|---|---|---|
| Baseline eRBP [48] | 75.52 ± 1.31 | 60.69 ± 0.6 | 1x |
| TACOS | **93.22 ± 0.22** | 82.56 ± 1.12 | 2.5x |
| LwF [51] | 71.02 ± 0.46 | 71.5 ± 1.63 | 2x |
| MAS [52] | 68.57 ± 6.85 | 66.42 ± 2.47 | 3x |
| Online EWC [53] | 65.55 ± 3.30 | 64.32 ± 2.48 | 3x |
| BGD [54] | 89.73 ± 0.88 | 80.44 ± 0.45 | 3.44x |
| SS [24] | 91.98 ± 0.12 | **82.90 ± 0.01** | 2x |

First, we compare knowledge retention (i.e. stability) across models, by measuring the degradation in accuracy on prior tasks. We also monitor the degree of perturbation of the network by measuring the cosine similarity between network representations. of the same input after learning each task. As seen in Figure 5 (the dashed grey line indicates when the model has been trained on the task specified on the x-axis), TACOS shows lower degradation of accuracy than either the fixed model or the SNN. This result is dramatic when TACOS has a

maximum metaplastic state greater than 25; a setting with which the fixed model is incapable of learning.

Similarly, the cosine similarity of network representations for the digits in task 1 (0,1), after learning task 1 and after learning task 5 is 0.98 vs 0.86 on class 1 in TACOS (m=25) compared to the baseline SNN and 0.98 vs 0.85 compared to fixed metaplasticity (m=10) as shown in Table 6. Equally important in addressing catastrophic forgetting is the plasticity of the model, ensuring that the model is capable of learning future tasks. To assess the plasticity of each model, we compare the single-task accuracy on the most recently trained task, as well as the average weight change during learning. For single-task accuracy (see Table 6), it can be observed that the gap between TACOS and the baseline SNN performance becomes progressively larger as the number of tasks learned increases — with TACOS performance culminating between 6% to 30% lower than the baseline SNN on the final task. This can be attributed to metaplasticity and consolidation restricting weight changes to preserve knowledge on prior tasks. This is also reflected in Figure 6, where

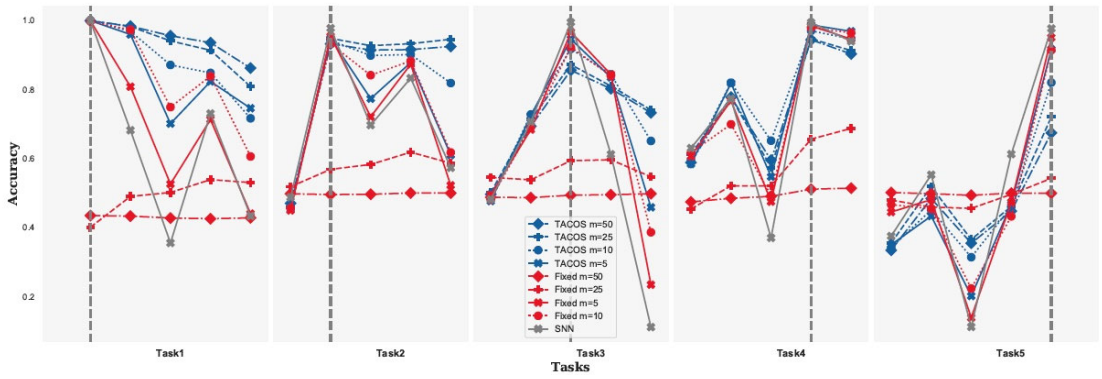



**Figure 5. Accuracy of Tasks over Tune for TACOS in Different Settings, and a Baseline SNN on Split-MNIST**

the SNN is seen to have ~ 3x the plasticity of a TACOS model. In comparison to fixed metaplasticity, TACOS allows for significantly more plasticity.

Finally, when varying the metaplastic limit in TACOS, there is little change to the average plasticity induced in the network. When considering the optimal trade-off between plasticity and stability there are three main takeaways - (i) using a dynamic metaplastic state as proposed in TACOS shows both greater plasticity and better retention of knowledge than a fixed metaplastic state, (ii) Although the plasticity of TACOS is ~ 3x lower than that of the baseline SNN, the mean accuracy is ~ 22% higher and catastrophic interference is ~ 5x lower and (iii) the stability-plasticity trade-off in TACOS on the split-MNIST and split-FMNIST tasks shows an inflection point in mean accuracy when the maximum metaplastic state is set to m = 25.

**Table 6. Cosine Similarity of Average Hidden Layer Activity Pattern for Classes 0 and 1 after Learning Task 1 and Task 5**

| | TACOS m =50 | TACOS m =25 | TACOS m =10 | TACOS m =5 | Fixed m = 50 | Fixed m = 25 | Fixed m = 10 | Fixed m = 5 | SNN |
|---|---|---|---|---|---|---|---|---|---|
| Class 0 | 0.9641 | 0.9620 | 0.9332 | 0.9188 | 0.9784 | 0.9781 | 0.9756 | 0.9169 | 0.9211 |
| Class 1 | 0.9890 | 0.9872 | 0.9718 | 0.9538 | 0.9775 | 0.9538 | 0.8532 | 0.8240 | 0.8632 |

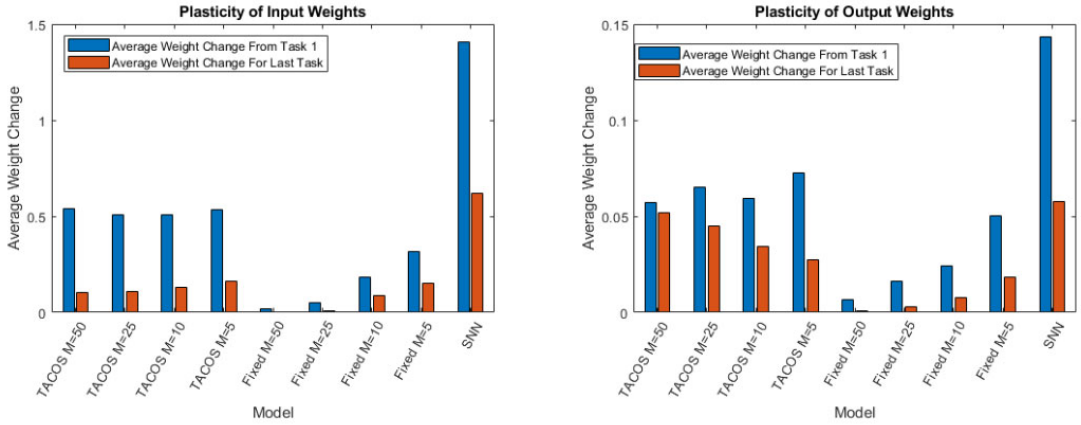



**Figure 6. The Average Weight Change Induced after Learning Task 1 and the Final Task for Input and Output Weights Standard Cell Library Development**

## 4.0 RESULTS AND DISCUSSION

### 4.1 Lifelong Learning Accelerator Design

#### 4.1.1 Standard Cell Library Development

Our lifelong learning chip Genesis was developed using the IBM 65nm technology node. We collaborated with Dr. Garrett Rose's team at University of Tennessee, Knoxville to develop the standard cell library for this technology node. The standard cell library consists of 181 cells listed in Table 7. All cells were verified for correct functionality. We have also worked on the physical layouts to ensure each of them complies with the Design Rule Check (DRC) and the Layout Versus Schematic (LVS) verification.

#### 4.1.2 Standard Cell Library Optimization

We optimized some of the standard cell library components in the IBM 65nm technology node to support the continual learning capabilities that were developed in the Genesis project. Design modifications were made to the existing memory cells which were characterized to support the auto-routing feature required to tape out the Genesis chip.

**Static Random Access Memory (SRAM):** SRAM cells are commonly used to store data for faster read and write access on the chip. It is designed using a series of 6T transistor cells. The digital design block in the Genesis project uses distributed memory architecture to access parameters (weights) in parallel, which need to be stored in on-chip Random Access Memory (RAM). To facilitate SRAM design, a 6T cell was extracted

**Table 7. List of Cells in the IBM 65nm Standard Cell Library**

| Cell Name | Count | Cell Name | Count |
|---|---|---|---|
| andx xx | 12 | oai2bb2 xx | 4 |
| aoi2bbx xx | 8 | clken xx | 4 |
| aoi2x xx | 8 | dlatch xx | 4 |
| addf xx | 4 | oai21 xx | 4 |
| addfi xx | 4 | oai22 xx | 4 |
| addh xx | 4 | or2 xx | 4 |
| buf xx | 7 | or3 xx | 4 |
| inv xx | 7 | xnor2 xx | 4 |
| ffxx xxx | 11 | xnor3 alt xx | 4 |
| mx2 xx | 4 | xnor3 xx | 4 |
| mxi2 xx | 4 | xnor4 alt xx | 4 |
| nor2 xx | 4 | xor2 xx | 4 |
| nor2b xx | 4 | xor3 alt xx | 4 |
| nor3 xx | 4 | xor3 xx | 4 |
| nor3b xx | 4 | xor4 xx | 4 |
| oai2bb1 xx | 4 | nandx xx | 28 |

from a TSMC 65nm technology RAM and then modified to IBM 65nm node. The modified cell was DRC and LVS clean. However, when multiple SRAM cells were integrated as an SRAM module, the space between two adjacent cells was less than the minimum distance according to the design rule check (DRC), leading to DRC violations. We modified the SRAM cell layout as shown in Figure 8 to resolve the issue.

**Standard Cell Characterization:** Layout of a large-scale design such as an accelerator relies on auto-routing tools. To carry out the automatic routing, the tools need Library Exchange Format (LEF) files of the components of a design. LEF files hold the information of the manually routed basic component cells like the metal routing coordinates input/output pins and the transistor details of the cell. For the Genesis chip layout, the SRAM module, mixed-signal design controller block and the crossbar architecture modules were characterized and the LEF files were generated which were further used by the auto-router to create the final tape-out.

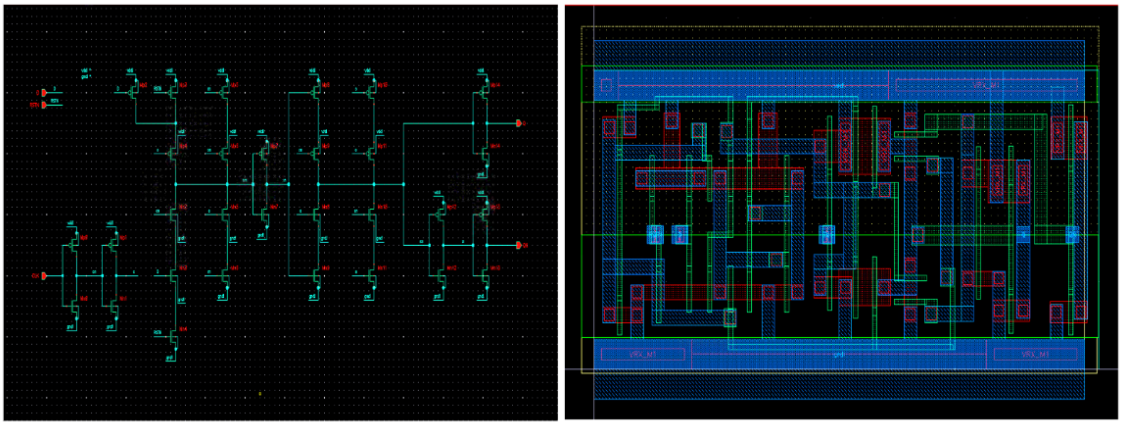


**Figure 7. Schematic and Layout of a D Flip-Flop Cell in the IBM 65nm Standard Cell Library**

**1T1R Cell:** In the mixed-signal design for Genesis chip, we used Hafnium Oxide (HfOx) based 1T1R cells based on IBM 65nm CMOS 10LPL Low Power (CMS10LPe) process, developed by our collaborator Dr. Nathaniel C. Cady's group at SUNY Polytechnic

Institute's Center for Semiconductor Research [51]. In the technology library, the body terminals of the transistors in the 1T1R cells were connected to the source. However, since transistors are symmetric devices, during reset the source reverses and the body terminal can be at a higher voltage compared to the source. To avoid this, for Genesis the 1T1R cell was updated to have a separate body terminal which is always connected to the lowest voltage supply. This change also leads to threshold increase in the transistor due to the body effect. The set and reset voltages during training Genesis were updated to accommodate this effect. The transistor in the 1T1R cell was chosen to have width of 1.6$\mu$m and length of 500nm as shown in Figure 9.

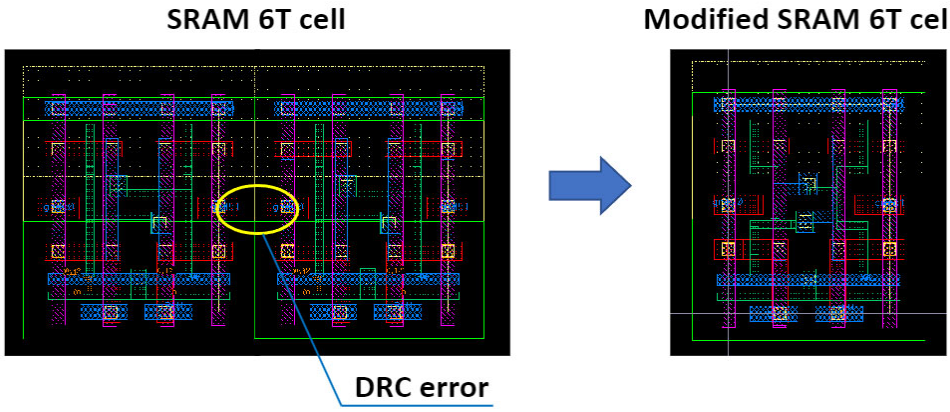



**Figure 8. Modification of 6T SRAM Cell**

### 4.1.3 MetaplasticNet Digital Accelerator Design

#### 4.1.3.1 System Architecture

The digital architecture of our model MetaplasticNet, shown in Figure 10, was implemented in IBM 65nm technology node. It consists of a systolic array, global controller, training unit, and on-chip distributed memory. The systolic array consists of 8×8 processing elements (PEs) dedicated to compute neurons' activations. The output weight SRAM holds synaptic weights, which are continuously regulated by the training unit. The control unit governs the data flow and generates the necessary control signals. During the inference phase, the input features are transmitted to the systolic array elements and stored simultaneously in the input SRAM. Each processing element computes the activation by accumulating (either adding/subtracting) the input features based on the weight generated by 16-bit LFSR. Unlike conventional designs, MetaplasticNet uses LFSR to generate random weights in the hidden layer, thus saving memory. The sign bit of the accumulators is stored as binary activations in activation SRAM. Since the systolic array can only compute 64 activations concurrently, the hidden layer is folded [52]. This design scheme requires the inputs to be broadcasted $N/64$ times, where $N$ represents the hidden layer size. Additionally, to reduce the resource utilization, the same systolic array is used for computing the output activations as well.

To compute the output activations, the efficacy of the output layer weights and the hidden layer activations are broadcasted to the systolic array. Since the output layer weights are modified during training, instead of generating them using LFSR, they are fetched from the on-chip SRAM. To accommodate the ensembles in the output layer, each ensemble layer is mapped to a row in the systolic array, which allows the architecture to

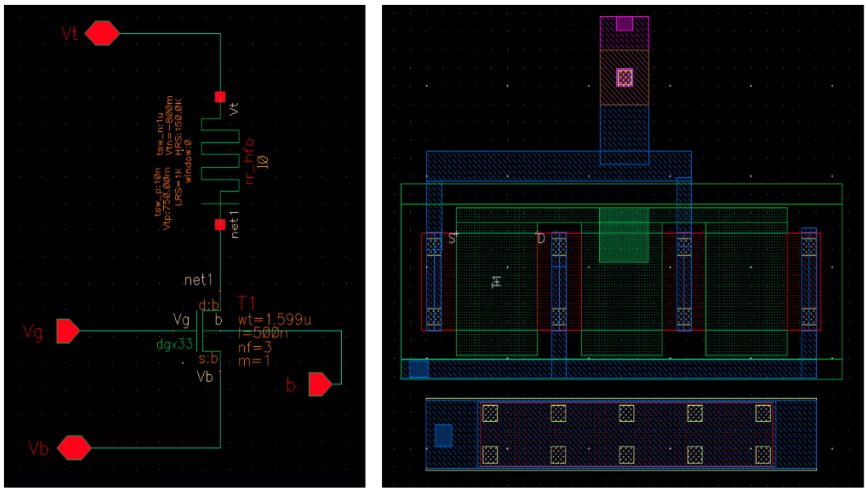



**Figure 9. Schematic and Layout of 1T1R Cell**

utilize the PEs efficiently. The activations of each ensemble layer are fetched from the PE and stored in a register file. During the training phase, the generated output layer activations are transferred to the training unit which computes the error for each neuron. The training unit consists of five training elements (TEs), each assigned to one ensemble layer. Every training element has 10 compartments corresponding to 10 output neurons. Each training compartment consists of a probability Look Up Table (LUT) and a 16-bit LFSR which generates a random number to be compared with the probability of the state of the weight. The weights are updated as a result of the comparison. It is important to mention here that output weights are updated in a stochastic manner, governed by hidden neuron activations. The fraction of hidden neuron activations utilized in the training are randomly chosen using two 8-bit LFSRs. The LFSRs define the position of the PEs in the systolic array, which model the active hidden neurons involved in training.

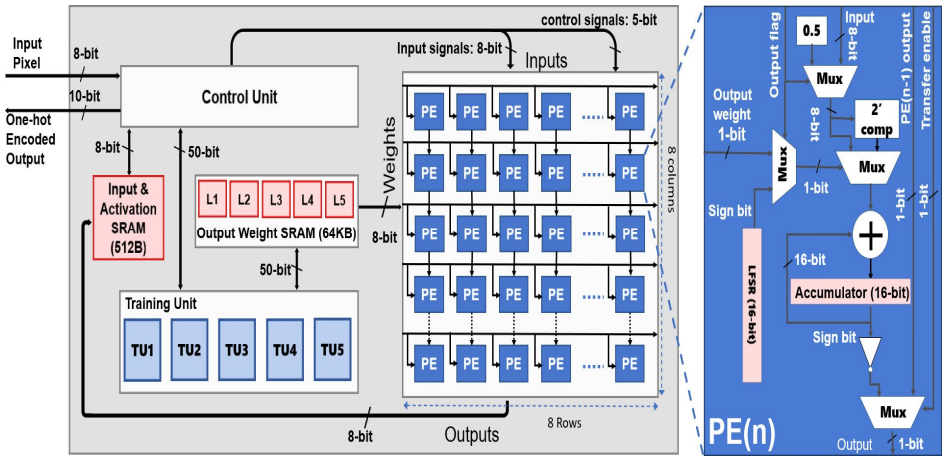



**Figure 10. System Architecture for MetaplasticNet**

### 4.1.3.2 Latency Evaluation

We analyzed the latency of training one sample using MetaplasticNet architecture for different configurations of the systolic array as illustrated in Figure 11. Having multiple columns of PEs reduces the number of steps to read the activations from the systolic array. Generally, increasing the total number of PEs improves the latency, however this

improvement does not scale linearly. For instance, when doubling the number of PEs from 8×8 to 8×16 array, the latency of the architecture does not scale proportionally due to the inefficient utilization of the PEs.

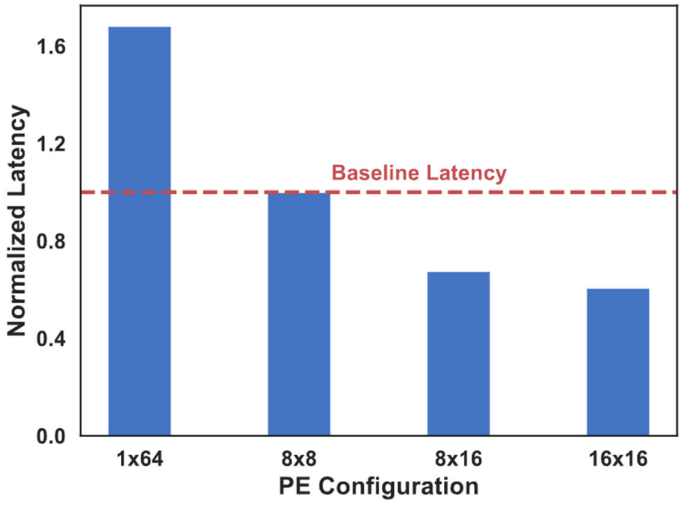



**Figure 11. Normalized Training Latency per Sample on the MetaplasticNet Architecture for Different Systolic Array Configurations**

### 4.1.4 Genesis-v1 Digital Lifelong Learning Accelerator

The developed continual learning accelerator architecture, shown in Fig. 13, consists of four main modules: i) Spike encoder, ii) Processing Elements (PE) matrix, iii) LIF neuron module, and iv) Metaplasticity block. A dedicated config/control module interfaces to all the modules and controls the dataflow by transferring inputs from the data transfer block to the encoder module and then to the PE matrix. The data transfer block reads the input layer spikes at every time step and stores them in the spike memory.

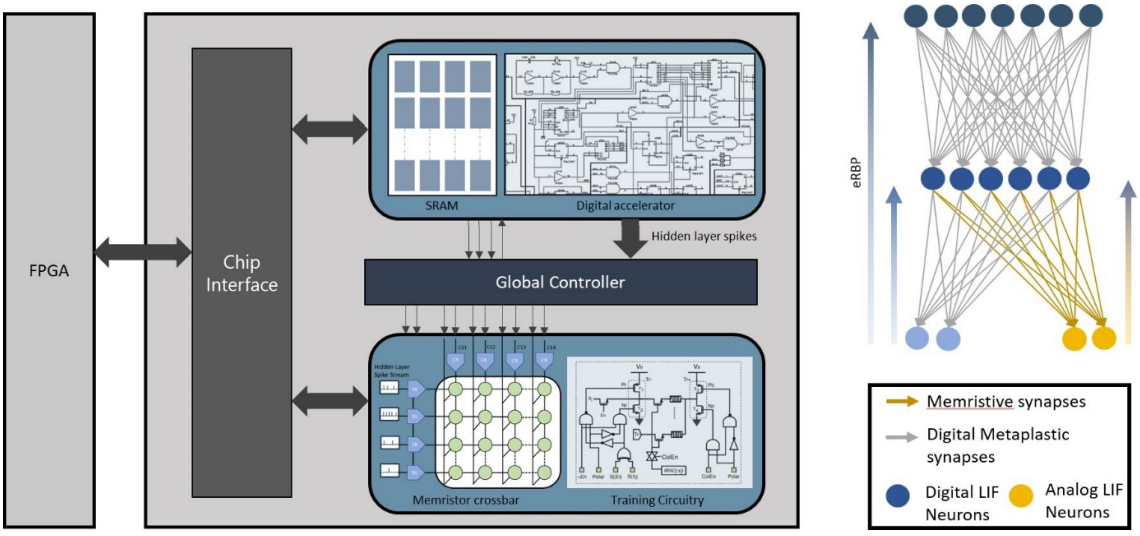



**Figure 12. System Architecture of Genesis-v1 Accelerator**

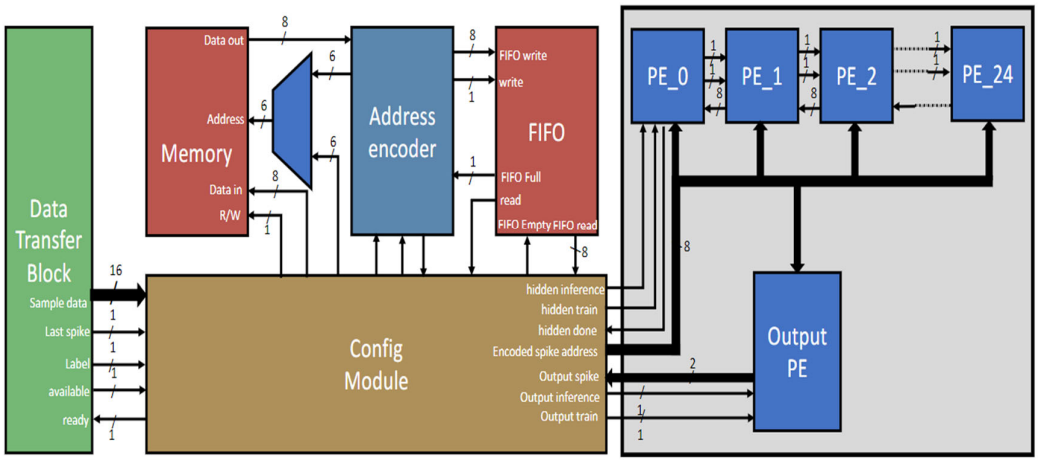



**Figure 13. System Architecture of the Digital Accelerator in Genesis-v1**

Network training is divided into two steps, the forward pass and the error backpropagation. During the forward pass, the address encoder reads the spikes from the memory and encodes the spikes to the index and forwards the index of the neuron that has spiked. This design choice reduces memory read operations by reading only the parameters associated to the neuron that spiked. Unlike skip zero operations, this architecture does not require a zero comparison and reduces the number of operations. The spike indices are stored in the First-In-First-Out buffer (FIFO), which is further broadcast to the PE array.

PEs are arranged in a one-dimensional structure, which is more area efficient (compared to a two-dimensional array of PEs) for a fully connected network topology [53]. Each PE in the array computes ($N_{fold} = N_h/N_{pe}$) neurons where $N_h$ is the number of hidden neurons and $N_{pe}$ represents the number of reconfigurable PEs. The PE matrix operates under an output stationary and input stationary combined dataflow configuration. Unlike the output stationary dataflow where the input is transferred in multiple iterations, here it is broadcast once and the computation of $N_{fold}$ neurons is performed sequentially in each PE. This reduces the number of spike memory reads but requires additional memory to store the partial sums for multiple neurons. On final accumulation of the partial sums, the PEs compute the currents, voltages, and output spikes of $N_{fold}$ neurons in the LIF neuron module. The output spikes are transferred to the spike memory sequentially in a pipelined fashion which can be further used by the address encoder to input the hidden-layer spike indices to the output PE.

During backpropagation, false positive and false negative errors are transferred to each hidden layer PE where the error spikes are integrated. According to the eRBP training algorithm, only neurons associated with spiked neurons are trained, giving the benefit of reusing the address encoder. In a similar manner as the forward pass, the spike indices are broadcast to the PEs where the synapses associated with the neuron are read from the Block Random Access Memory (BRAM) and forwarded to the metaplasticity block. To encode the input layer spikes the config module verifies the availability of input spike train data from data transfer block. The data transfer block is the initial block that interacts with the external processor to read the data. The data transfer block facilitates the system with the input spike train and other information such as the label spike and the control signal if the read data is to perform inference or to train the network as shown in Figure 14.

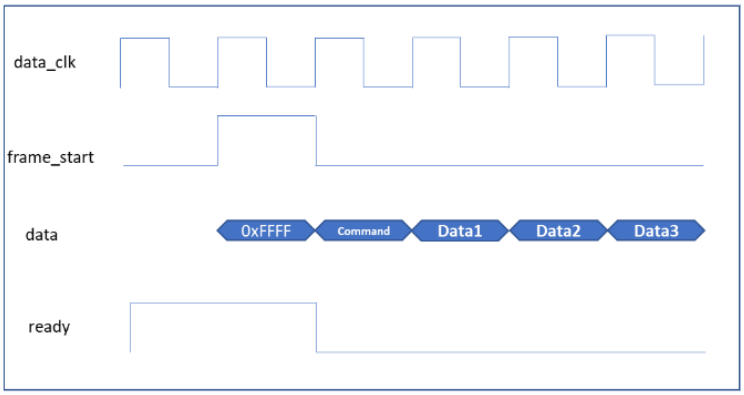



**Figure 14. Data Transfer Protocol Used in Genesis-v1**

### 4.1.5 Genesis-v1 Mixed Signal Design

#### 4.1.5.1 Memristor Device

In the Genesis project, we adopted the memristor device fabricated and characterized at SUNY Polytechnic Institute's Center for Semiconductor Research in the IBM 65nm 10LPe CMOS/Memristor process [51]. The memristor is a hafnium oxide-based 1T1R cell. The device consists of TiN/Ti/$HfO_2$/Ti stack connected to a transistor. The device can be used in binary mode by setting and resetting to high and low conductive states by controlling the compliance current with the help of the transistor. The set process is controlled by the compliance current, whereas the reset depends on the voltage applied across the device. During set operation, modulation of the compliance currents with the help of the transistor has been shown to give rise to 10 distinct levels in the High Conductive State within the range of ~ 40 µ S -286 µS (resistance range ~ 3.5 kΩ - 25 kΩ) with an average conductance resolution of ~ 27 uS [51]. The different resistance levels are shown in Figure 15. The conductive states give rise to a precision of ~ 3-bits.

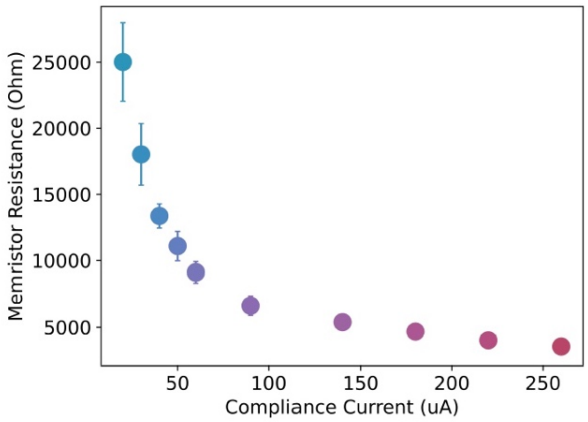



**Figure 15. Mean and Standard Deviation of 1T1RCell Resistance Levels vs. the Compliance Current**

#### 4.1.5.2 Synaptic Weight Configuration and Continual Learning Evaluation

The low precision of the 1T1R device is a challenge to achieve high performance in continual learning setting. The ability to precisely modulate the weights of the network is a prerequisite to achieve high accuracy in classification tasks [9], and it is critical for a network with on-device training without off-chip post-training modulation of weights. However, the precision of weight modulation is limited by the small number of states achievable in the device. Previously, researchers have proposed multi-memristor synapses to increase the resolution of weight modulation in networks [9]. In this setting, multiple memristor devices are connected in parallel to form the network weight. During inference, all devices participate, but during training, only one device is updated. The update process is arbitrated by a global counter, which reduces the peripheral overhead. It has been shown that in a multi-memristor synapse, if n number of memristors are connected in parallel, the weight resolution increases by a factor of n. However, increasing the precision with multiple memristors in a weight comes with the trade-off of an increased size of the crossbar. In Genesis-v1, we choose to incorporate three 1T1R cells to realize synaptic weight. In the crossbar, three adjacent 1T1R cells in row form a synaptic weight. To realize both positive and negative weights, we assign a bias column in the crossbar. The current through the bias column is subtracted from the current through the multi-memristor synaptic weight, and it is combined and scaled with an inverting summing amplifier.

Figure 16 shows the weight configuration for the multi-memristor synapse in a crossbar. The bias current is subtracted through the bias resistance $R_b$ and the feedback resistance $R_f$ scales the output. The weight, $w_{mem}$, is given by Equation 12.

$$w_{mem} = R_f\left(\frac{1}{R_{mem}} - \frac{1}{R_b}\right)$$

**Equation 12. Memristor Weight Configuration**

Figure 17 shows the distribution of weights in a given range for a single memristor weight, when we program 1T1R cells from the highest resistive state to the lowest sequentially. The right panel shows the same for a multi-memristor weight with 3 memristors. We see that the multi-memristor weight leads to 3 times gain in then weight resolution.

We incorporated the proposed weight scheme into a modified TACOS algorithm optimized for lower hardware cost. TACOS has two mechanisms that contribute to continual learning, activity-dependent metaplasticity, and synaptic consolidation. Activity-dependent metaplasticity requires maintaining the activity trace of neurons, and heterosynaptic plasticity requires a second set of weights, which are both compute-intensive and memory-intensive. Therefore, in the mixed signal design, we simplified the algorithm by only incorporating activity-independent metaplasticity for continual learning. We evaluated it on the split MNIST continual learning benchmark. We considered a spiking neural network with a single hidden layer of 200 neurons. The MNIST digits were resized from 28×28 to 16×16 to reduce the number of input neurons. The resulting network has dimensions 256×200×2. The evaluation shows that with activity-dependent metaplasticity in memristor weights, we achieve 11% increase in mean accuracy across tasks compared to a baseline network with memristor weights that do not incorporate activity-dependent metaplasticity. We also achieve an improvement 6% over a baseline spiking neural network with full precision weights.

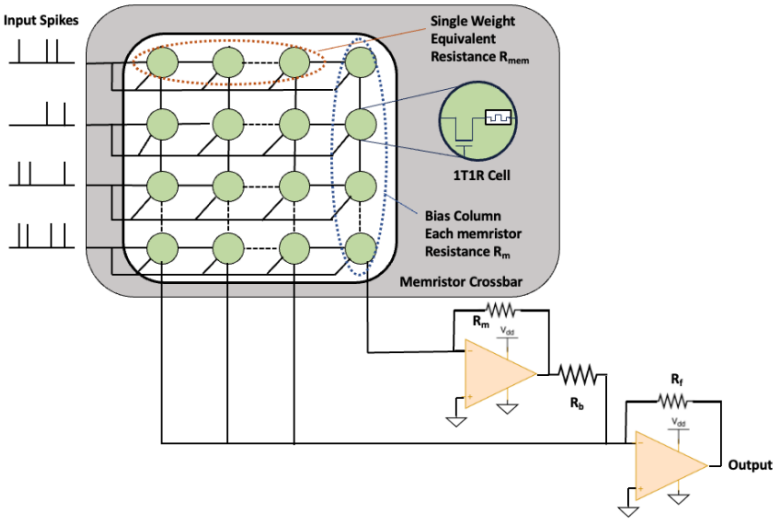



**Figure 16. Genesis-v1 Multi-Memristor Synaptic Weight Configuration with 3 1T1R Cells and Inverting Summing Amplifiers**

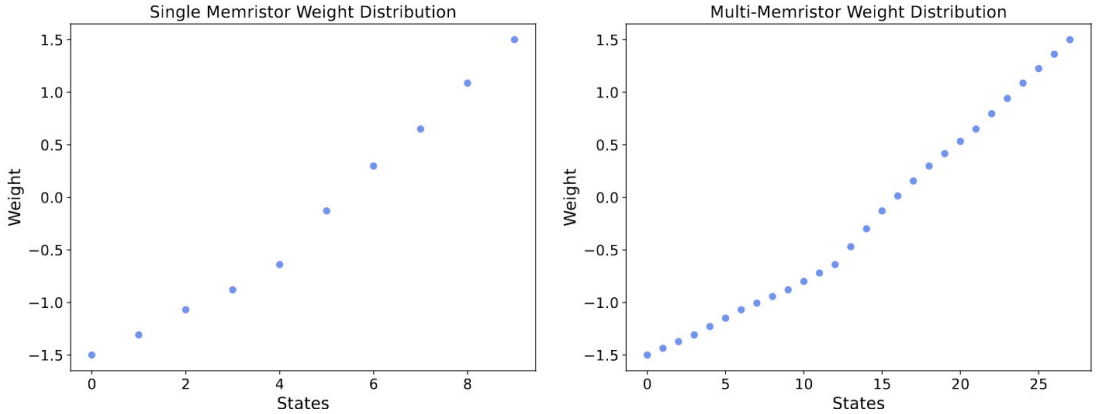



**Figure 17. Weight Distribution with a Single Memristor and a Multi-Memristor Weight with Three 1T1R Cells**

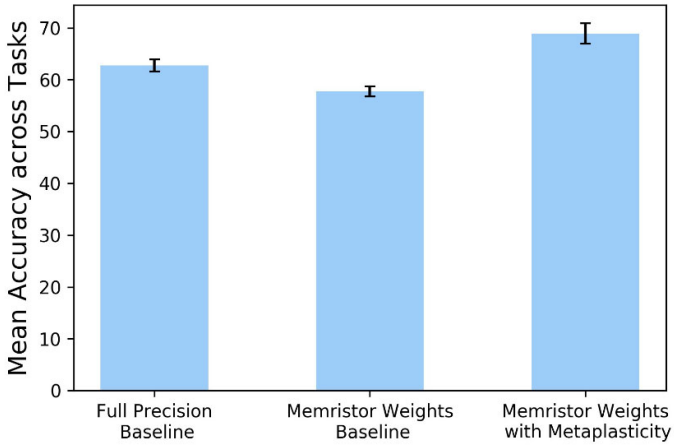



**Figure 18. Individual and Mean Task Accuracies with Standard Deviations on the Split-MNIST Task**

#### 4.1.5.3 Mixed Signal Design Architecture

Genesis-v1 incorporates a mixed-signal design which implements part of the spiking neural network and interfaces with the digital accelerator. The hidden to output layer is realized through a cascaded modular crossbar architecture and analog LIF output neurons. The design receives hidden-layer activations from the digital design and the output spikes from the analog neurons interface with the digital error neurons. Figure 19 shows an overview of the mixed signal design.

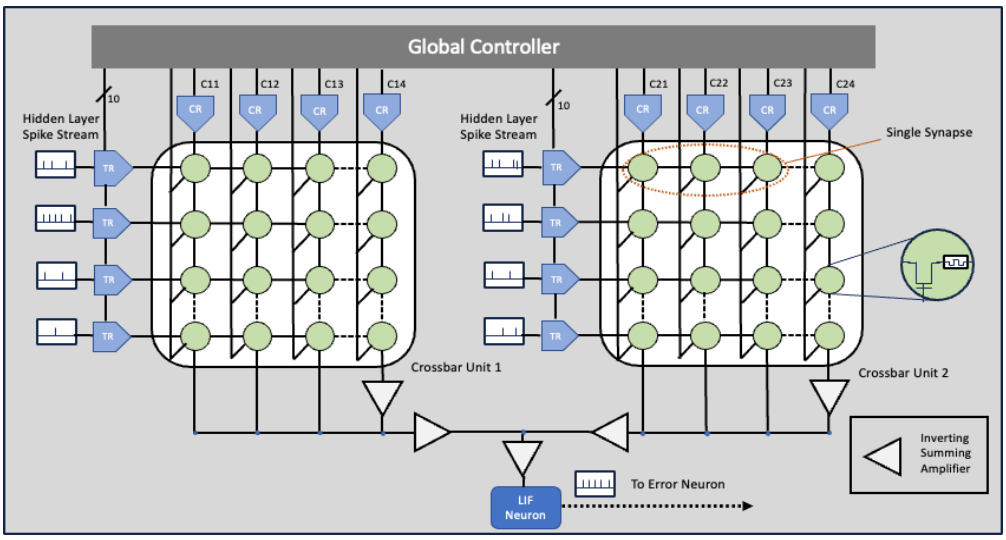



**Figure 19. Mixed-Signal Design Configuration of Genesis-v1 Chip with Modular Crossbar Architecture**

The weights from the hidden layer to the output which form a 200 × 2 matrix translate to a 200 × 7 crossbar in the weight scheme discussed in the previous subsection. Such a large crossbar leads to high sneak path currents which leads to undesirable power dissipation. To reduce this effect, we divided this crossbar into blocks of 50 × 7 [52]. The blocks are isolated from each other, and their outputs are combined with inverting summing amplifiers and connected to the LIF output neurons. The LIF (Figure 20) neurons consist of three parts, an inverting integrator, a comparator, and a pulse generator.

The inverting comparator integrates the input currents from the crossbar, which is compared to the threshold voltage in the comparator. The pulse generator module is made of a circuit with resistance and capacitor which controls the pulse width of the output neuron spike when the integrated voltage crosses the threshold. The pulse width of the neuron is controlled by a bias voltage which is set as an input to the design to ensure tunability. Figure 21 shows the schematic of the neuron in Cadence Virtuoso and its functionality.

The design was functionally verified on IBM 65nm technology node in Cadence Virtuoso Suite. A key feature of the design is its flexibility. The input voltages of the crossbar along with the threshold voltage of the LIF neuron were assigned individual pads in the chip frame. It ensures that the network activity can be tuned based on input data.

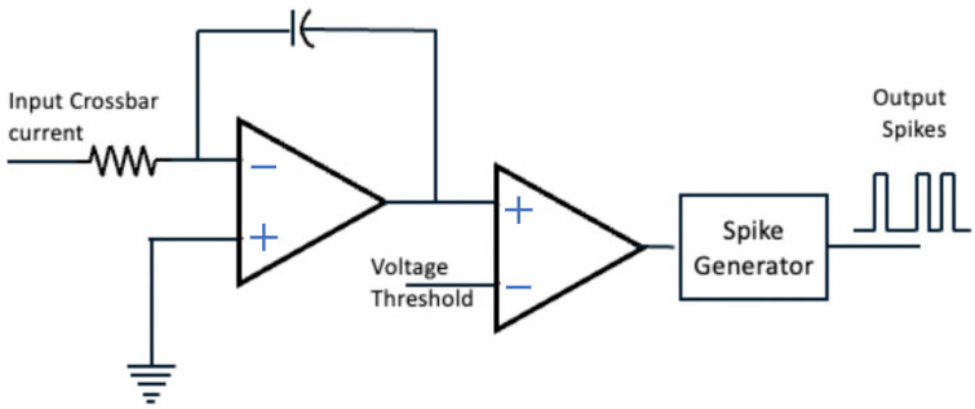



**Figure 20. LIF Neuron Comprising of an Inverting Integrator, Comparator and Spike**

#### 4.1.5.4 Mixed Signal Design Layout

**Modular Crossbar Architecture Layout**

The rows of the crossbar modules are connected to transmission gate-based control circuitry. The transmission gates consist of transistors with a high threshold voltage to enable passing of the high voltages required for programming the memristors. During inference and read operations, the rows are connected to diodes to ensure the current flows towards the output neurons and minimize leakage to other input neurons. The width of the metal wires in the crossbar were carefully designed to ensure that the voltage drop across them is not substantial. The crossbar rows and column metal widths were chosen such that the path resistance is below 10% the minimum resistance of the memristor. Since metal1 in this technology node has 6 higher resistivity compared to metal2, to minimize voltage drop and crossbar area, we designed the crossbar rows with metal1 and the columns were designed with metal2. Each crossbar module was interfaced with inverting amplifiers to collect their inputs, which were applied to the inverting neurons. The outputs of the inverting neurons are fed to the error computing units, which were implemented in the digital domain.

**Leaky Integrate and Fire Neuron Layout**

The layout of the leaky integrate and fire neuron required considering several design constraints are as follows-

- Network size: The hidden layer size in our network is 200, with moderate amount of neuron activity. The weight resistance in our design is also low since we used multiple 1T1R cells in parallel. Both factors lead to a high input current to the LIF neuron. However, if the input current is too high, the neuron will fire continuously and fail to learn anything meaningful. Hence, the parameters of the neurons were designed to ensure moderate neuron activity at the output neurons, i.e. higher capacitance, and resistance.

- Area budget: The network dynamics required large capacitance to ensure moderate neuron activity, however, capacitors consume large area which is prohibitive.

Considering these constraints, to decrease the input current to the neuron, we chose a large input resistance to the neuron. The neuron capacitance was realized with metal3 and

metal1 as top and bottom layer respectively to achieve higher capacitance in a lower area. To ensure that network activity can be tuned we kept the threshold voltage of the neuron as well as the input voltage to the network as input voltages to the chip. The design layout is shown in Figure 22. The area of the mixed signal block is 690 $\mu m$ × 520 $\mu m$.

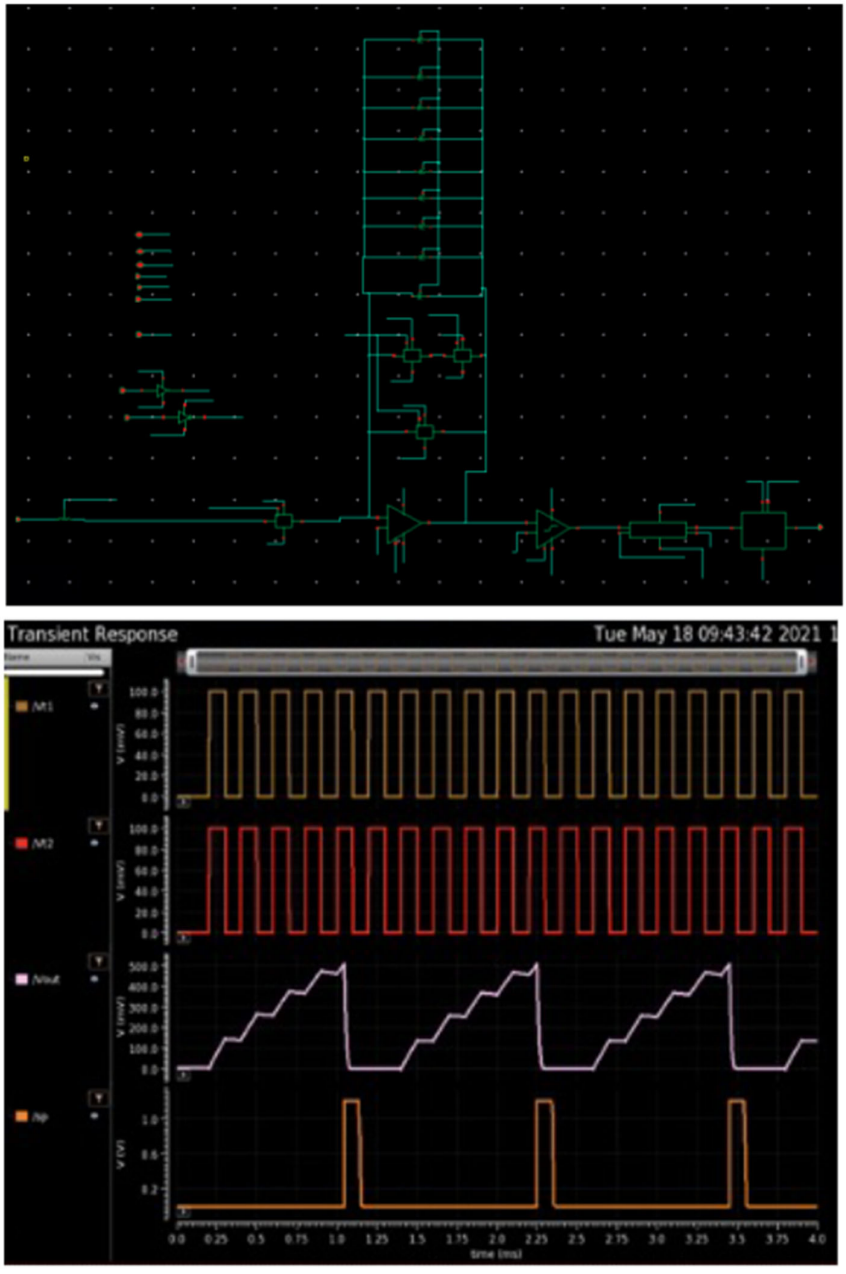



**Figure 21. LIF Neuron Schematic in Cadence Virtuoso and Its Functionality**

**Layout design of Genesis-v1**

The digital accelerator was combined with the mixed signal design and connected to operate synchronously. The manual layout of memristor crossbar architecture was connected to the auto-routed digital design using Cadence design kit tools. The final design consists of four major parts as show in the Figure 23 covering the total area of 4.7 *mm*×

4.6 *mm*. The majority of the chip area is dedicated to the on-chip SRAM, followed by the digital design due to the larger network size compared to the mixed-signal design.

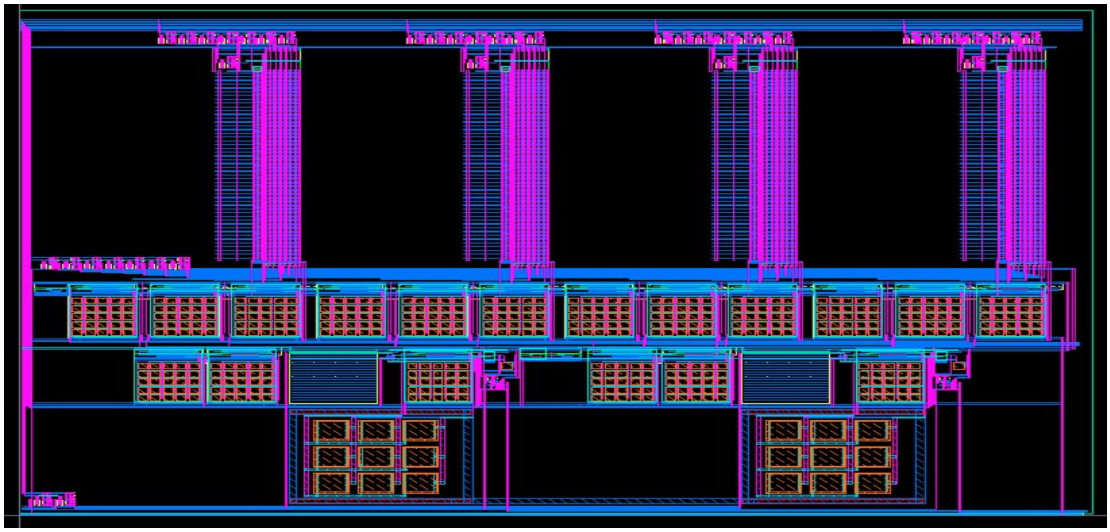


**Figure 22. Layout of Mixed-Signal Design of Genesis-v1**

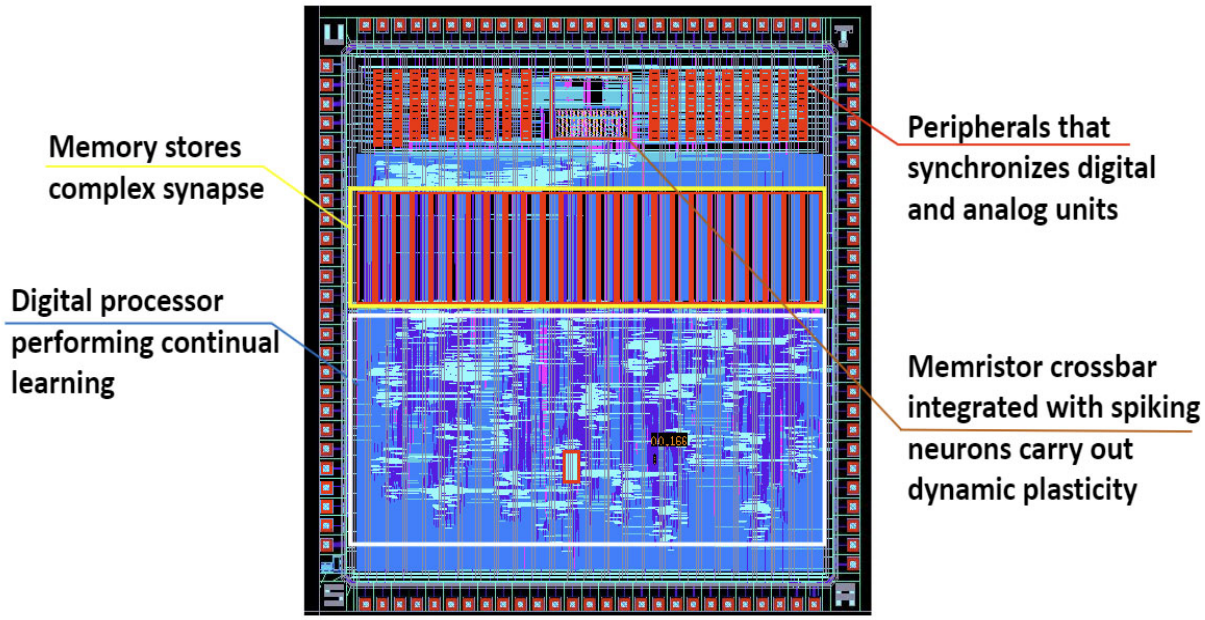



**Figure 23. Layout of Genesis-v1 Chip**

### 4.1.6 PCB Design and Verification

To verify the Genesis-v1 chip a test bed a printed circuit board (PCB) was designed which allowed us to program the chip. To design the board, we identified pins in the pad frame for the input-output signals controlling the state of the Genesis chip as shown in Figure 24 and connected them to the peripherals of the board.

The designed PCB was verified before the chip was placed on the board. Two major tests are needed to verify its functionality. Every pin needs to be tested for short circuits

to ground or supply rail. Any short circuit connection from a pin to either ground or supply rail can greatly damage a chip. Second, the inter-pin shorts need to be tested. Every pin must be tested against other pins using a multimeter short detector.

The goal of the testing procedure is to ensure that there are no open connections and functionality of the chip is correct. A common way to test the connections is by driving the pins on one end using an FPGA board and testing the signal on the other end of the connection. We equipped the test PCB board with level shifters to convert the output of the FPGA board to the power supply level of our chip. The setup was designed to accommodate any processor or FPGA to drive the Genesis chip. We used the PYNQ-z2 FPGA board for our testing procedure.

### 4.1.7 Genesis-v1 Post Silicon Verification

The testing of the chip was carried out in two phases. The first phase of the chip is to test the basic setup designed on the chip during layout of the chip as shown in the Figure 26. An AND gate was placed in the chip before it was sent to the tape out. The input of the AND gate was tied to the two least significant bits (LSBs) of the data signals going to the accelerator for data transfer. If the test pin on the chip is set to high the output of the chip should give the AND operation of the data0 and data1 signals.

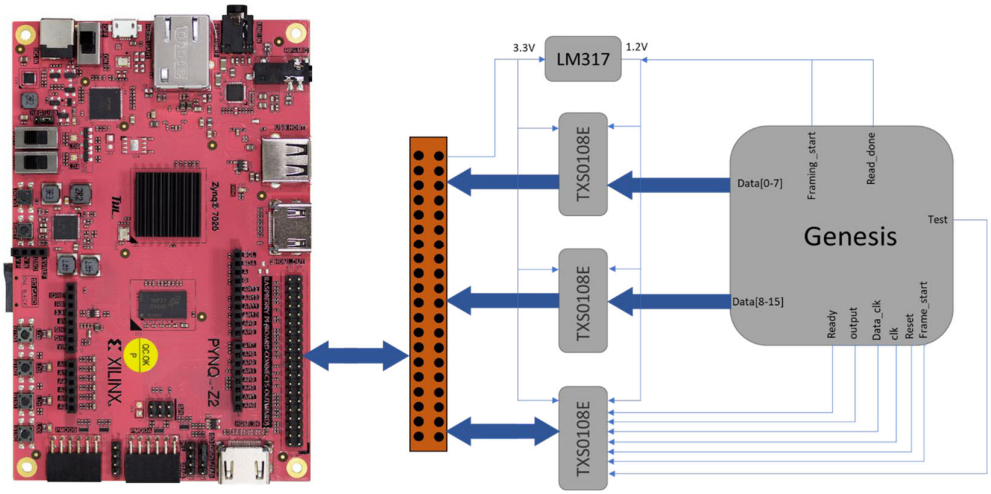



**Figure 24. Block Diagram of the PCB for Genesis-v1 Test Set-up**

After the AND gate operation verification on the chip, the accelerator working was be tested. Golden vectors that were used to carry out the functional verification will be used to test the chip. Current tape out does not have additional pins setup to monitor the status of the accelerator to verify the chip.

The primary choice to verify the chip is by considering it as a black box. The accelerator designed was programmed on the Programmable logic (PL) of the PYNQ-z2 board and the test vectors were interfaced and sent to both the PL part of PYNQ-z2 board and the Genesis chip. The timing variations in the PL logic and the PYNQ-z2 board were synchronized with the Genesis chip by slowing it down.

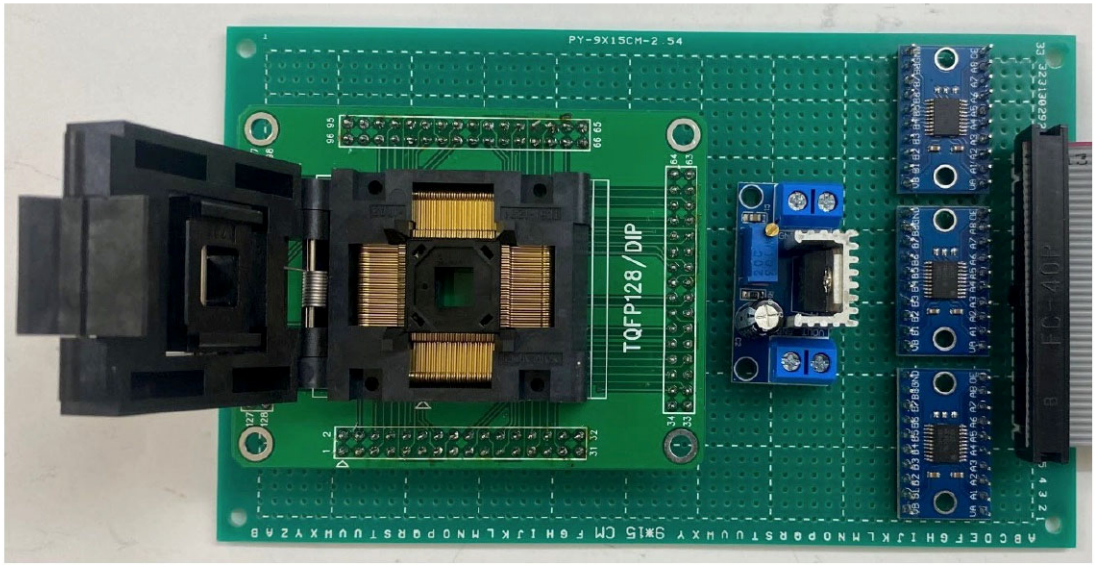



**Figure 25. Snapshot of the PCB Designed to Connect Genesis-v1 to a Processor/PYNQ-z2 Board**

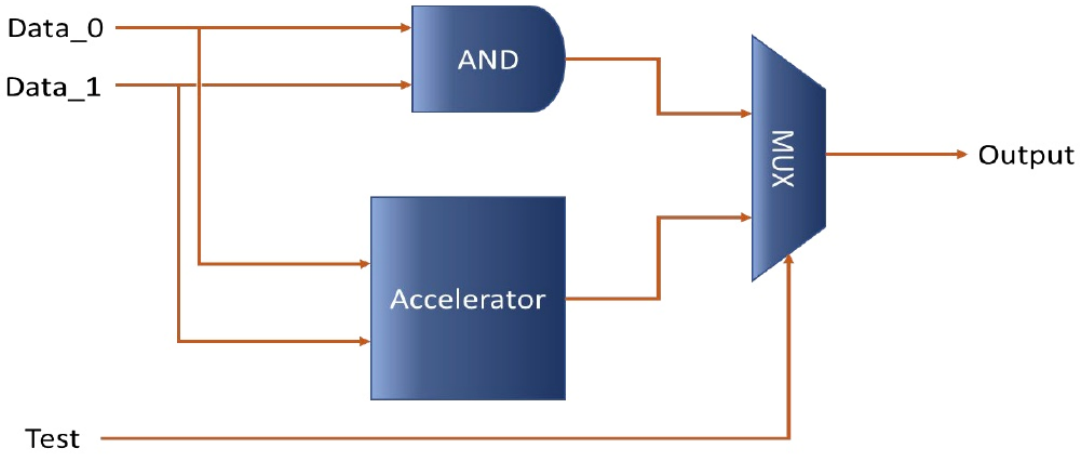



**Figure 26. Connectivity of AND Gate in Genesis-v1 Operating in Parallel to the Accelerator**

### 4.2 Digital Accelerator Prototype On FPGA Platform

#### 4.2.1 Genesis Prototype

The RTL design of Genesis-v1 was developed with focus on compatibility and optimization for FPGA (Field-Programmable Gate Array) prototypes. To demonstrate the accelerator capabilities on an edge device, the PYNQ-z2 FPGA was selected as shown in Figure 27. ZYNQ Xilinx ZYNQ 7000 series SoCs features an ARM A9 Processor operating at 650 MHz, which allows the users to integrate a co-processor with the accelerator. The PYNQ-z2 development kit was chosen due to its low-operational power, high computational throughput of ~27 GOps/sec and in-built 256 kB Block RAM.

The accelerator design described in the digital system architecture [55] with 25 Processing elements and the control block was synthesized Xilinx XC7Z020-1CLG400C mounted on the PYNQ-z2 board. The SRAMs in the Genesis-v1 were replaced with the BRAM in the RTL design. The spike memory and FIFOs used register files to store the memory were kept unchanged due to faster access requirements. The integration of the RTL design with the processor was carried out in two routes, i) designing the custom transfer block on the processor and ii) modifying the transfer block on accelerator to AXI-lite protocol. To implement the former choice to connect the processor and the Programmable Logic (RTL design), General Purpose Input/Outputs (GPIOs) were used, which can be programmed individually as shown in Figure 28. The advantage of Python-enabled GPIO reconfiguration, facilitated by the PYNQ-Z2 board, significantly strengthens the choice of using the evaluation kit. Though the processor provides additional user configurability but it adds significant delay simulating the data transfer protocol. To reduce the data transfer latency of the system

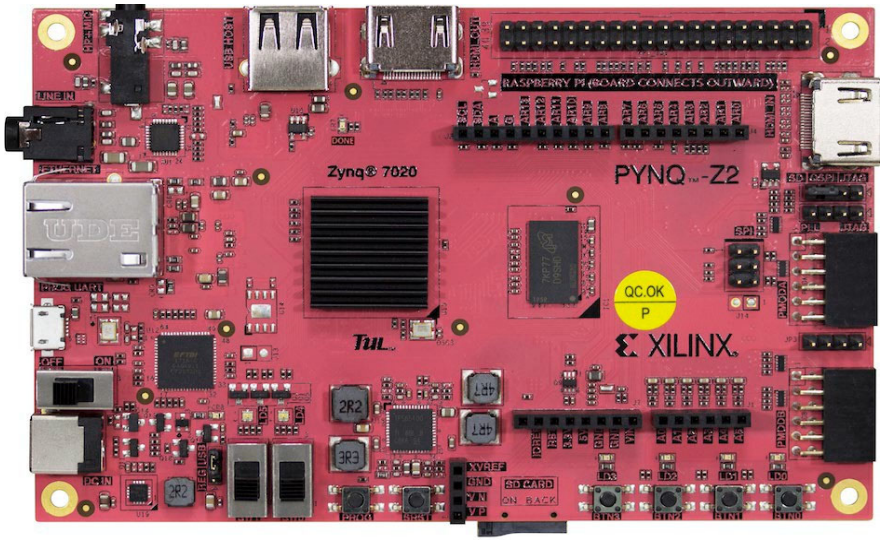



**Figure 27. PYNQ-z2 Board Diagram**

**Table 8. Comparison between Two Versions of FPGA Design**

| Design | Version-1 | Version-2 |
|---|---|---|
| Protocol | 16-bit parallel data transfer | 32-bit AXI-lite |
| Training Latency | 4 seconds | 100 milliseconds |
| Power consumption | 1.52 W | 1.61 W |

the second version was adapted which AXI-lite protocol to write and read data from the accelerator. The AXI bus used internally by the ARM A9 process is extended externally which is then interfaced with the accelerator. Due to bypassing GPIO modules on the FPGA and configuring a single pin, this version transfers 32-bit data using AXI bus into the register files on the accelerator. This update in data transfer protocol reduced the latency by ~100× with minimal increase in power consumption from 1.52Watt to 1.61W.

To reduce the overall computational resources and the memory storage we incorporated the following optimizations.

### 4.2.2 Dyadic Scaling

Multipliers and dividers are power-hungry and add significant latency to the critical path. Previous studies have shown that multipliers consume more than seven times the power of adders [38]. To reduce power consumption, we used dyadic scaling for the time constants of the LIF neurons, as well as for the learning rate. This allowed us to replace the multipliers/dividers with shift registers.

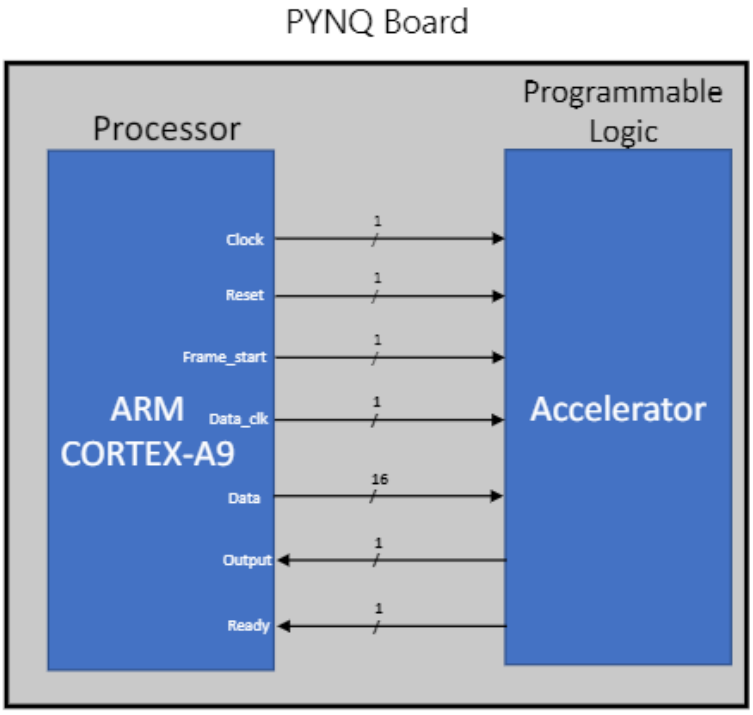



**Figure 28. Block Diagram of the Connections from Processor to Programmable Logic**

### 4.2.3 Metaplasticity Function Approximation

To achieve continual learning, our algorithm uses an exponential function to regulate the synapses. Typically, the exponential function is realized using a Taylor series expansion or an LUT [53].

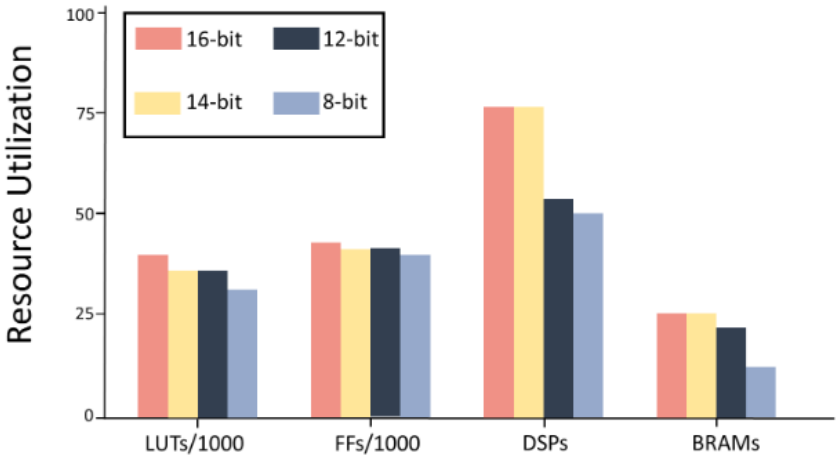



**Figure 29. Resource Utilization of the Accelerator on the PYNQ-z2 Development Board**

However, these methods are computationally expensive and memory-intensive. To reduce the complexity, the exponential function is approximated with a two-piece linear dyadic function as shown in Figure 30. The two-piece approximation requires fine-tuning the slope of the function which affects the continual learning performance and also the bit shift operations. We explore various slopes of the function and select the best-performing function.

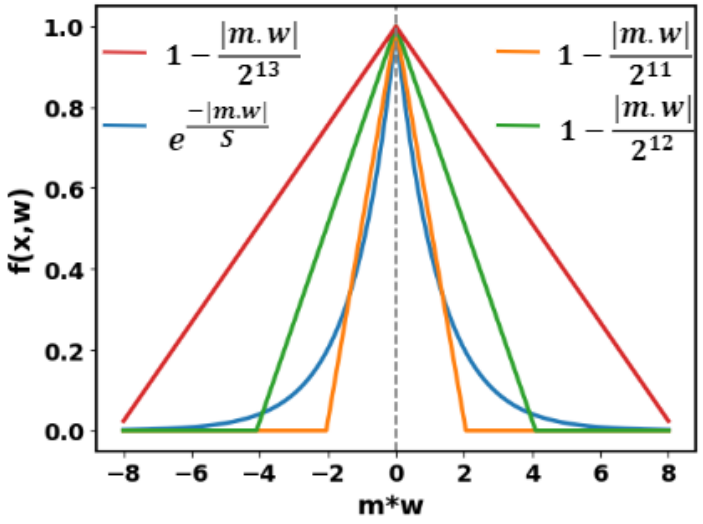



**Figure 30. Piecewise Linear Metaplasticity Functions that Resemble the Scaled Exponential Function**

### 4.2.4 Dual Fixed-Point Representation

In system architecture, parameters require high precision (resolution), because a large proportion of them take values close to zero, and they also require high dynamic range. This can be achieved by storing parameters at high bit-width. However, the parameters are stored in BRAM available in the FPGA and with high bit width scaling up the network will significantly increase the memory footprint of the accelerator and the performance of the continual learning algorithm which can be observed in Table 9.

To overcome this challenge, the weights may be quantized to a lower bit width, e.g. 8-bits. However, if parameters are represented using uniform fixed-point format, such a reduction of bit width will significantly exacerbate the mean accuracy.

**Table 9. Individual and Mean Task Accuracies with Standard Deviations on the Split-MNIST Task for Weight Representation with Different Bit Precisions**

| Task | Baseline SNN | Without Optimization | 16-bit | 12-bit | 8-bit | 8-bit DXP |
|---|---|---|---|---|---|---|
| Class 0,1 | 45.53% ± 2.67 | 62.06% ± 18.92 | 78.59% ± 3.54 | 65.41% ± 2.96 | 36.66% ± 0.00 | **84.91% ± 6.28** |
| Class 2,3 | 57.47% ± 1.85 | 65.81% ± 2.08 | **78.43% ± 8.70** | 61.76% ± 4.30 | 55.22% ± 0.00 | 74.40% ± 12.40 |
| Class 4,5 | 23.34% ± 5.03 | 50.77% ± 7.45 | 61.22% ± 8.01 | 52.35% ± 7.60 | 52.27% ± 0.00 | **63.37% ± 12.65** |
| Class 6,7 | 94.19% ± 0.07 | 91.78% ± 1.03 | 92.67% ± 1.14 | 74.75% ± 5.81 | 45.33% ± 0.00 | **95.29% ± 0.52** |
| Class 8,9 | 93.12% ± 0.07 | **86.38% ± 4.04** | 71.95% ± 1.53 | 60.08% ± 3.92 | 39.70% ± 0.00 | 79.35% ± 8.8 |
| Mean | 62.73% ± 1.13 | 71.36% ± 4.30 | 76.57% ± 5.43 | 62.87% ± 6.90 | 45.93% ± 0.00 | **79.46% ± 2.68** |

We analyzed the accelerator resource utilization and power utilization on the PYNQ-z2 board which are illustrated in Figure 29. The design utilizes from 28000 to 32000 out of 53,200 LUTs available on the FPGA based on the bit-width selection. This indicates that there is room for expansion in the design on board. After implementing the design, only a quarter of the available BRAM on the FPGA was utilized when the bit precision of the system was chosen to 16-bit. This leaves room for the system to expand and incorporate more neurons and synapses or increase the precision of the network.

The operating frequency of the design is 10 MHz, and the design can train MNIST images of size 16x16 at a speed of 30 samples per second. Although the reported throughput of the accelerator suggested higher performance, the actual implementation exhibits higher latency. This slowdown is attributed to the processor's GPIOs emulating the custom interface protocol, leading to increased polling operations necessitated by various concurrent processes running on the processor and impacting the monitoring of the accelerator's ready signal.

### 4.3 Genesis-v2 Accelerator

#### 4.3.1 Insights from Genesis-v1 for Improvements in Genesis-v2

During the design and testing of Genesis-v1, we gained valuable insights that helped us make improvements in the design of Genesis-v2.

There were three modifications made to the mixed signal design. Our first modification is regarding the continual learning algorithm developed for memristor weights. During Genesis-v1 development, the continual learning performance with low-precision memristor weights was much lower than that observed in TACOS. We realized the continual learning mechanisms developed for high-precision weights may be ill-suited to low-precision memristor weights. Hence, we developed a novel continual learning algorithm. Research in computational neuroscience shows that cascaded probabilistic metaplasticity in binarized weight leads to higher memory retention [36]. Inspired by this insight, we proposed a probabilistic metaplasticity mechanism that can help memory retention in spiking networks with low precision weights. We considered a spiking neural network which is trained based on an error threshold that drastically reduces the memory footprint of the continual learning algorithm. We evaluated the proposed algorithm on two continual learning benchmarks considering a 6-bit multi-memristor weight. The results show that it

achieves performance comparable to state-of-the-art networks with ~ 35% lower memory overhead.

The second modification involves the design of the crossbar module. Our continual learning simulation results showed that a multi-memristor weight with seven memristors in parallel achieves ~6-bit precision. Hence, in Genesis-v2, we incorporate crossbar modules that can support this arrangement. However, while designing Genesis-v1, we noticed that the parasitic resistance of the crossbar module rows and columns can be high, which can impact the read accuracy of devices. Hence, in Genesis-v2, we opt for a smaller 8 × 8 crossbar module to improve the read margin of the devices. The crossbar assembly consists of four such modules.

Our last modification was to increase the testability of the module. In Genesis-v1, the digital accelerator was connected to the crossbar modules. This made the module less accessible for testing protocols. It also led to interdependence issues during the test procedure since some operations in the digital accelerator depend on the output from the mixed-signal module. To alleviate these issues, in Genesis-v2, we have separated the crossbar module from the digital design. The crossbar module nodes can also be directly accessible from the pad frame. This design approach allows for various testing scenarios for the crossbar assembly.

Moreover, the digital design choices incorporated in the Genesis-v1 architecture did not include a few design considerations. One major design challenge was the configurability/reconfigurability which made the first design a rigid architecture. The hyperparameters and the network topology were not configurable by the user in the first design. Another major challenge in architecture was limited memory and limited parallel cores. The sequential character of the network gives additional benefits of reusing the resources, which were not explored in Genesis-v1. Additionally, the functionality verification of the RTL design was a major step during the design process. The functionality testing was done by generating the golden test vectors from the software model. This verification strategy can be upgraded to verify the design against a reference model which does not require generating vectors before the process.

Table 10 shows the design features of the two chips side by side for the primary updates.

**Table 10. Design Features of Genesis-v1 and Genesis-v2**

| Design Feature | Genesis-v1 | Genesis-v2 |
|---|---|---|
| Systolic Array Architecture | One-dimensional Array | Two-Dimensional Array |
| Processing Elements | 25 | 64 |
| LIF Neurons | hard-wired | Reconfigurable parameters |
| Scalability | 200 hidden neurons | 256 hidden neurons |
| On-Chip Memory | 102 kB | 512 kB |
| User Configurability | No configurability | Supports configurability |
| Crossbar Array Size | 50 × 7 | 8 × 8 |
| Multi-memristor Weight Configuration | 3 1T1R cells | up to 7 1T1R cells |

### 4.3.2 Memory-Efficient Continual Learning with Probabilistic Metaplasticity

#### 4.3.2.1 Continual Learning Mechanism

We employed two insights in our work to achieve memory-efficient continual learning with low-precision memristor weights. The first involves an error-based spiking neural network training mechanism. The network is trained with eRBP as described in section 3.3.3.1. We modified the training mechanism to enable training with low-precision weights. Weight changes corresponding to small errors cannot be incorporated to low-precision weights due to the limited tunability. We avoided this issue by incorporating the insight that the error could be accumulated in the dendritic compartment of neurons. The error accumulation was modified as follows:

$$U_{j(t+1)} = U_j(t) + \frac{\Delta t}{T_u} E(t) R_u$$

**Equation 13. Non-leaky Error Integration**

Here $U_j$ is the error accumulated at the dendritic compartment of the jth post- synaptic neuron. Once the error accumulates to a level that requires a change in weight corresponding to its precision, the weight is updated. A similar concept of accumulation of gradients has been proposed in earlier work [54], however, our approach leads to lower memory overhead since errors are accumulated in neurons instead of a separate high-precision memory corresponding to each weight. Once the error reaches the threshold, similar to the eRBP, weights connected to the active pre-synaptic spikes are selected for update if the post-synaptic current magnitude is within a given range.

The second insight was to leverage probabilistic metaplasticity to enable continual learning with memristor synapses. In this scheme, weights are updated in a probabilistic manner. Throughout learning, the probability of weight updates evolves so that important weights have a lower probability of updating compared to less important weights. Thus, the important weights maintain their state, to preserve learned information. The less important weights maintain a higher probability to update during subsequent training to allow plasticity in the network. The transition or update probability of weight depends on its magnitude and the activity of the surrounding neurons. For the weight $w_{ij}$, connected between presynaptic neuron $i$ and postsynaptic neuron $j$, the transition probability is given by:

$$p_{update}(m_{ij}, w_{ij}) = exp\left(-abs(m_{ij} w_{ij})\right)$$

**Equation 14. Transition Probability of Weights**

Here, $m_{ij}$ is the metaplastic factor of the weight that depends on the activity of the adjacent neurons. This ensures that weights with higher m value and magnitude will be preserved during the training which helps in preservation of previously learned information. The learning mechanism is detailed in Algorithm 2.

**Table 11. Mean Accuracies across Tasks with Standard Deviations of Different Algorithms on the Split-MNIST Task and the Split-Fashion MNIST Task**

| Algorithm | Split-MNIST Task | Split-Fashion MNIST Task | Memory Overhead |
|---|---|---|---|
| LwF [51] | 71.50% ± 1.63 | 71.02% ± 0.46 | 2x |
| MAS [52] | 66.42% ± 2.47 | 68.57% ± 6.85 | 3x |
| BGD [54] | 80.44% ± 0.45 | 89.73% ± 0.88 | 3.44x |
| SS [24] | **82.90% ± 0.01** | 91.98% ± 0.12 | 2x |
| TACOS [10] | 82.56% ± 1.12 | **93.22% ± 0.22** | 2.5x |
| Gradient Accumulation | **83.06% ± 1.62** | **93.42% ± 0.28** | 2.5x |
| Proposed Probabilistic Algorithm | **82.73% ± 1.70** | **92.97% ± 0.50** | 1.625x |

#### 4.3.2.2 Experimental Results

We evaluated the proposed probabilistic algorithm on two continual learning benchmarks, the Split-MNIST [40] task and the MNIST-Fashion MNIST [41] sequential image classification task in domain-incremental learning scenario [42, 43]. Each benchmark consists of five tasks (two classes in each class) presented to the network sequentially. We consider the spiking neural network to have a single hidden layer of 200 neurons and two output neurons. As a control experiment, we also train a spiking neural network with activity-dependent metaplasticity [8] in which weight gradients accumulate. The performances of the networks are then compared with contemporary continual learning algorithms, as shown in Table 11. As we see, the probabilistic algorithm achieves continual learning performance comparable to state-of-the-art networks using full-precision weight. The gradient accumulation approach with activity-dependent metaplasticity also achieves similar results. However, the probabilistic algorithm requires ~ 35% less memory overhead compared to the gradient accumulation-based approach. This greatly increases the feasibility of the proposed continual learning mechanism on energy-constrained platforms.

**Algorithm 2:** Continual Learning Algorithm with Probabilistic Metaplasticity

```
for t ∈ 𝒯 do
    for {x^t, y^t} ∈ {X^t, Y^t} do
        for t ∈ T_sim do
            Forward Pass: y(t) ← f (x(t), W(t))
            Random Error Feedback: τ_U ∂U/∂t = E R_U
            Update Neuron Trace: d/dt X_tr = −X_tr/τ_tr
            if U_j > U_th then
                ε ∼ uniform(0, 1)
                for i ∈ {X_i(t) == 1} do
                    if I_min < I_j < I_max then
                        p_update = exp(−abs(m_ji(t) w_ji(t)))
                        if p_update < ε then
                            c_ij = c_ij − sign(Uj)
                        end
                        U_j = 0
                    end
                end
            end
        end
        Update memristor weights: W ← Program(R_mem, c)
        Update metaplastic co-efficient: m ← m + Δm
    end
end
```

#### 4.3.3 Genesis-v2 Reconfigurable Digital Lifelong Learning Accelerator

The initial iteration of the digital accelerator encountered significant limitations, marked by bottlenecks such as a small processing element (PE) count, lack of reconfigurability, and scalability issues. Recognizing the critical importance of a scalable and reconfigurable architecture in enhancing overall performance, the second version of the digital accelerator aimed to overcome the initial bottlenecks, ensuring a more robust and versatile platform capable of accommodating evolving computational demands. The first architecture had 25 PEs which process 200 hidden layer neurons. Corresponding to 25 hidden layer PEs, the system has a dedicated output layer PE which is dedicated to two output neurons. Each PE comprises of 4 kB memory which stores the fan-in synapses of the corresponding neuron. PEs were also accommodated with a LIF neuron compartment with hyperparameters such as R and C values of the neurons hard-wired. To reduce the complexity of the architecture, the system was equipped with neuron activity-dependent metaplasticity and did not include the synaptic consolidation mechanism.

The second version of tape out shown in Figure 31 introduces a significant advancement in its architecture, featuring a multicore parallel processing design with cores arranged in an 8x8 systolic array configuration. A crucial modification was made to transition the system from a 1D to a 2D array, effectively reducing the fan-in and fan-out of the core and benefits from low node-to-node distance implying faster data transfer to the PEs. To

enhance reconfigurability and scalability, the distributed memory was consolidated into a single block. The accelerator exhibits an impressive capability to process 256 neurons in a single layer, with the memory requirement for storing synapses in one layer with 256 input and output neurons requiring 256 kB. In the broader context, for processing two layers of networks, a minimum on-chip memory of 512 kB is indispensable. Notably, the challenge posed by additional memory needs due to continual learning was addressed by strategically placing metaplasticity parameters of 16-bits and synapses of 16-bits in a common block, providing an elegant solution to accommodate evolving requirements in memory constraints.

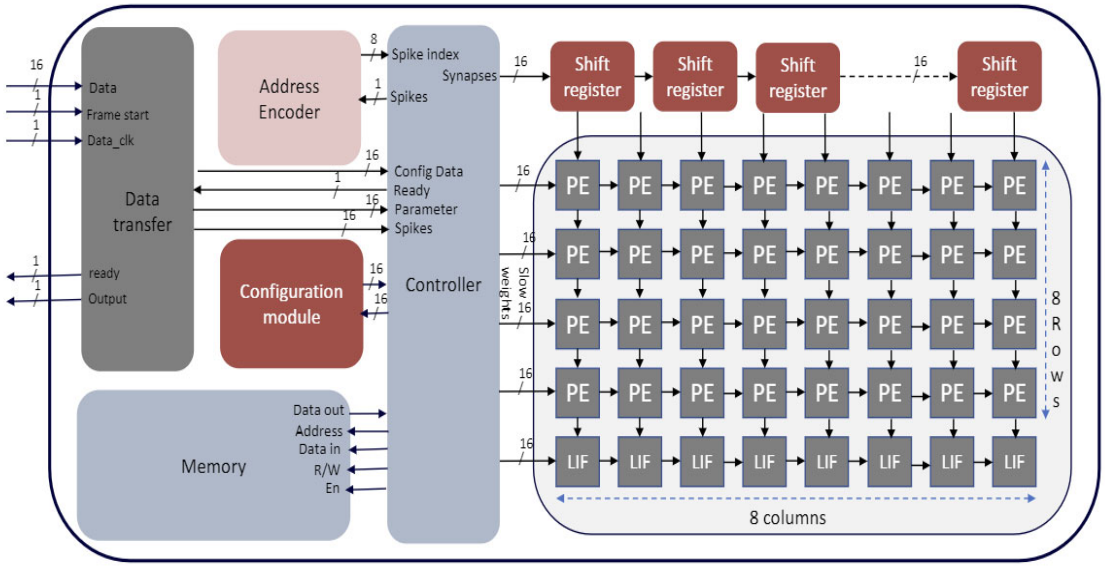



**Figure 31. System Architecture for Reconfigurable Genesis-v2 Digital Accelerator**

**Table 12. ALU Instructions in the Digital Accelerator**

| Instruction | 3-bit Opcode |
|---|---|
| Reset PE | 000 |
| Move data | 001 |
| Initialize accumulator | 010 |
| Accumulate data | 011 |
| Dendrite update | 100 |
| Metaplastic synapse update | 101 |
| Synapse load | 110 |
| Reset accumulator | 111 |

In the initial iteration, the processing elements of the first version were seamlessly integrated, featuring a comprehensive ensemble of a metaplastic weight update block, synaptic memory, LIF (Leaky Integrate-and-Fire) neuron, and an accumulator. This amalgamation of components played a pivotal role in the overall functionality of the system. The original design employed an explicit state machine structure, a paradigm that was subsequently enhanced with the incorporation of an Arithmetic Logic Unit (ALU). The ALU brought a heightened level of sophistication to the processing architecture, providing versatility and efficiency in computation. Notably, the ALU was equipped with

eight instructions, each meticulously detailed in Table 12. Furthermore, the accumulator embedded within the system was a 16-bit fixed-point accumulator, adding precision to the numerical operations performed. Recognizing the need for adaptability, the weight update block underwent a strategic modification to accommodate the reconfigurability of a bilinear function. This adaptation not only underscored the commitment to flexibility but also showcased a proactive approach to meeting evolving computational demands in the ever-evolving landscape of processing technology.

#### 4.3.3.1 Custom Designed Pipelined LIF Neuron Architecture

The pipelined LIF module is a parameter-reconfigurable module that allows users to change hyperparameters in the accelerator. The programmable parameters in the LIF module are the resistance and capacitance values and the maximum and minimum voltage thresholds of the LIF as shown in Figure 32. This allows us to modulate the integration speed of the LIF neuron, and consequently the level of activity in the neurons.

The pipelined LIF module is optimized to consume low energy compared to its sequential version. The updated design can be efficient if the total number of neurons exceeds a count of 40 as shown in Figure 33.

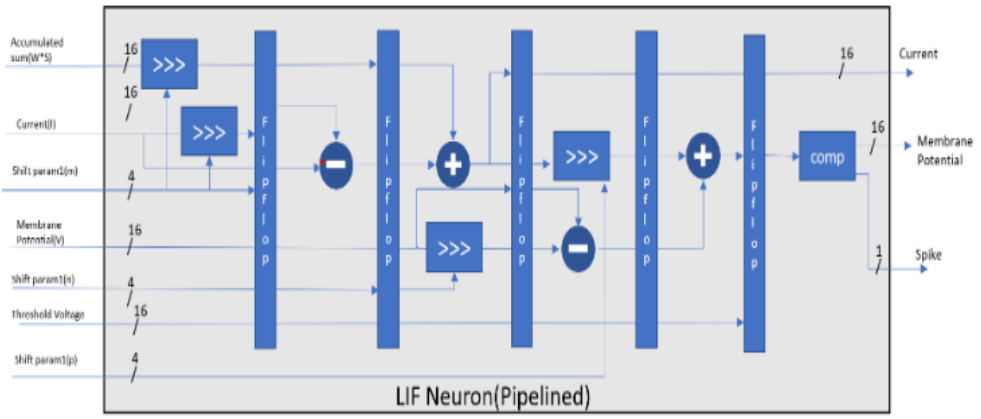



**Figure 32. Pipelined LIF Module Equipped with User-Configurable Hyperparameters**

### 4.3.4 Genesis-v2 Crossbar Assembly

#### 4.3.4.1 Crossbar Assembly Overview and Schematic

Genesis-v2 features a crossbar assembly that consists of four 8 × 8 1T1R crossbar units and three decoder units. The configuration of the assembly is shown in Figure 34. Each crossbar unit has eight rows, eight columns, and eight row-select or gate terminals. All the terminals of the crossbar units are connected to the chip frame pins to improve testability and accessibility. To optimize the number of pins connected to the assembly, the four crossbar units share these pins. Each crossbar terminal is connected to the pins through transmission gates that are activated with the decoder signals. Transmission gates were designed to ensure that the voltage drop across is less than 10% of the applied voltage under the worst load condition. Since the wide transmission gates form a high capacitive load, we designed three decoder units that are dedicated to selecting the rows, columns, and gate terminals of the selected crossbar, respectively.

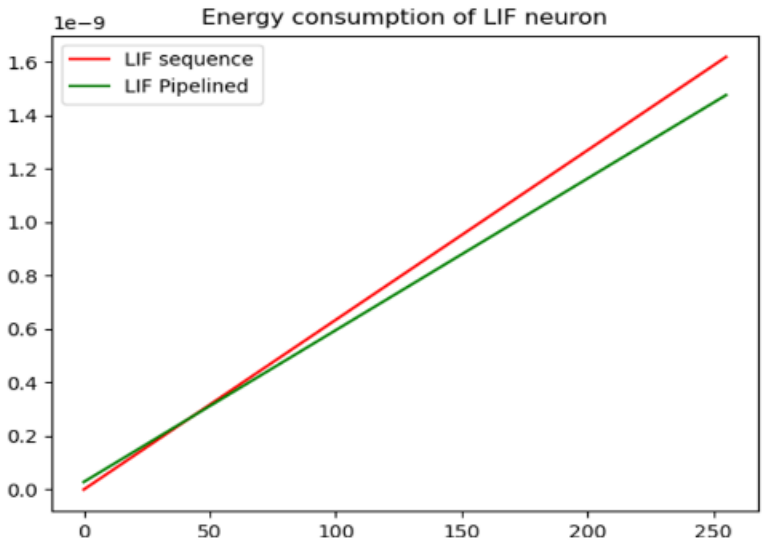



**Figure 33. Energy Consumption of Pipelined LIF Module and Sequential LIF Neuron vs. the Total Number of LIF Neurons**

### 4.3.4.2 Crossbar Module Layout

The crossbar module layout was carried out focusing on the reduction of parasitic resistance, since it degrades the accuracy of vector-matrix multiplication and read operations. The rows of the crossbars relate to metal1 (M1), which has ~ 6 times higher sheet resistance compared to metal2 (M2), which connects the columns. The interconnects that connect the 1T1R cells in the crossbars were sized so that each cell (shown in Figure 35) is connected to ~8 Ω of resistance due to row and column connections. This leads to a maximum parasitic resistance of ~64 Ω seen by the 1T1R cells in the 8th row and 8th column, which is less than 2% of the minimum 1T1R resistance. The transmission gates that control the signal flow in each crossbar feature multiple fingers to reduce mismatch and improve gate resistivity. The entire assembly layout is (shown in Figure 35) area is 650 µm× 750 µm.

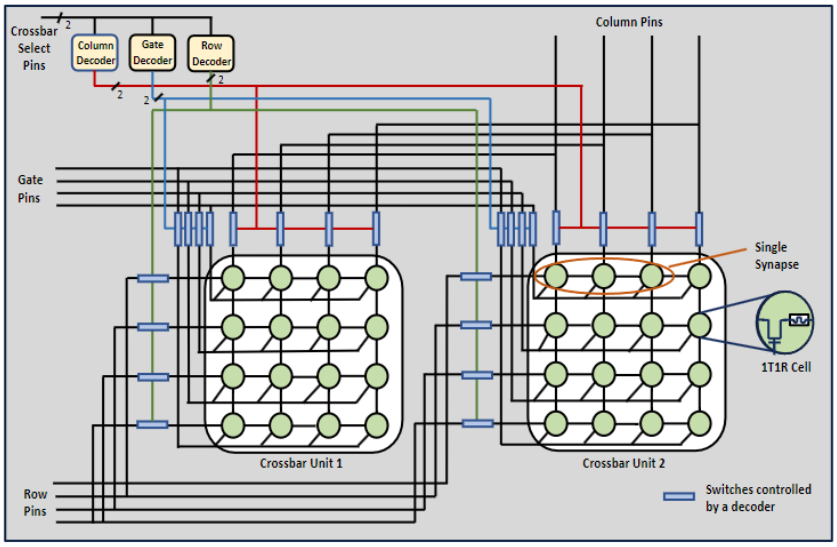
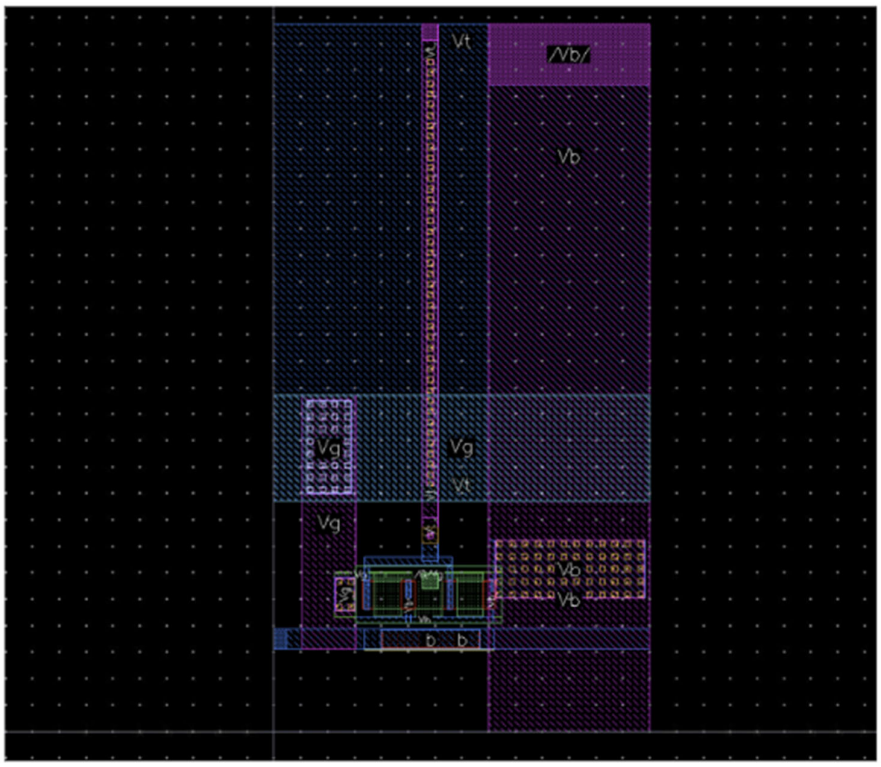
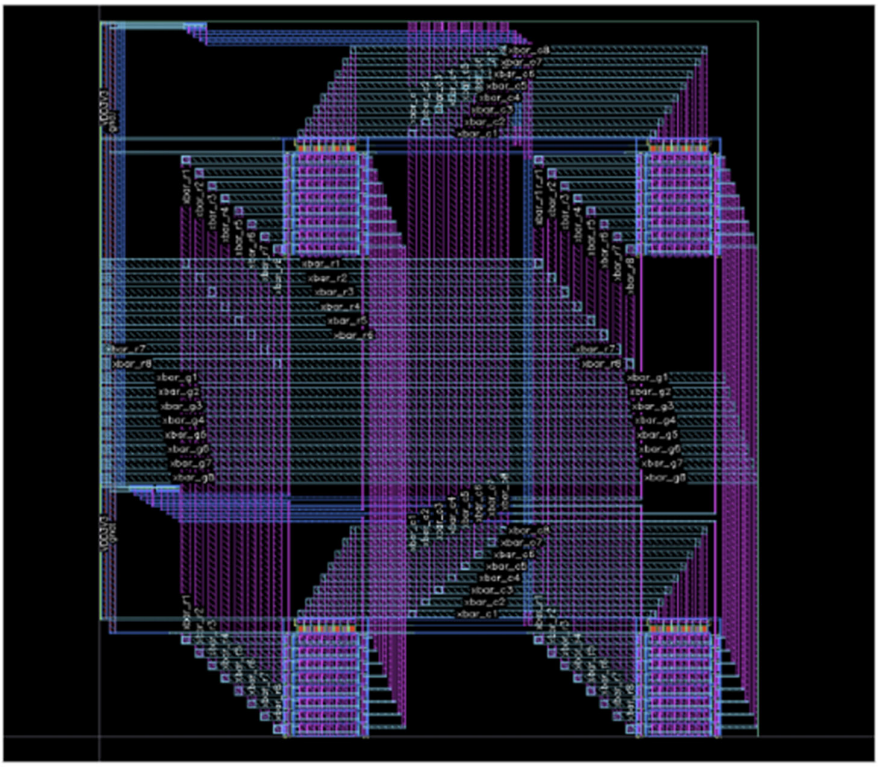



**Figure 34. Genesis-v2 Crossbar Assembly Configuration for Two Crossbar Units**




(a)



(b)

**Figure 35. Layout of a Single 1T1R Cell (a) and the Crossbar Assembly (b)**

#### 4.3.4.3 Performance Analysis

We designed test benches that verify the programming and reading operation of the crossbar assembly. To emulate the testing scenario, we initialize the crossbars at the maximum resistance specified in the Verilog-A model. The parasitics of the crossbar are also extracted in this scenario. Then testbench input stimuli sequentially programs (i.e., carries out set operation) 1T1R cells of a crossbar unit at a time. Following this, we conduct read and inference operations with the test stimuli and record the outputs. The results were compared with ideal expected outputs. In Genesis-v2, the output of the read and inference operation is sensed externally. In the testbenches, we emulated the external sensing with an ideal operational amplifier which is configured in an inverted amplifier set-up.

**Read Operation Analysis**

During read operation, the read voltage is applied to the row of the target 1T1R cell, and its column is connected to the operational amplifier. The gate terminals of the rest of the rows are grounded to minimize leakage current and improve read accuracy. This procedure was carried out for the 10 resistive states of the 1T1R cell, and the outputs were observed. The read operation accuracy is calculated as-

$$\boldsymbol{Percentage\ Error} = \frac{V_{ideal} - V_{readout}}{V_{ideal}} \text{ x } 100\%$$

**Equation 15. Read Error Calculation**

In the crossbar assembly, the read operation output known as $V_{readout}$ is compared to the ideal voltage, $V_{ideal}$. To analyze the impact of parasitic resistance, we examine the read outputs of 1T1R cells located in the first and eighth rows and columns (position (1,1) and (8,8), respectively) of the 8 × 8 crossbar. Although the (1,1) cell and the (8,8) cell face the lowest and highest amount of parasitic resistance, the difference between their read outputs is negligible, indicating an accurate read operation. The error of the read outputs increases with increasing conductance, which is expected because of the higher loading effect at lower resistivity. Furthermore, we observed a higher error rate at position (8,8) due to its higher parasitic resistance. However, as shown in Figure 37, the percentage error remains within 7% of the ideal output.

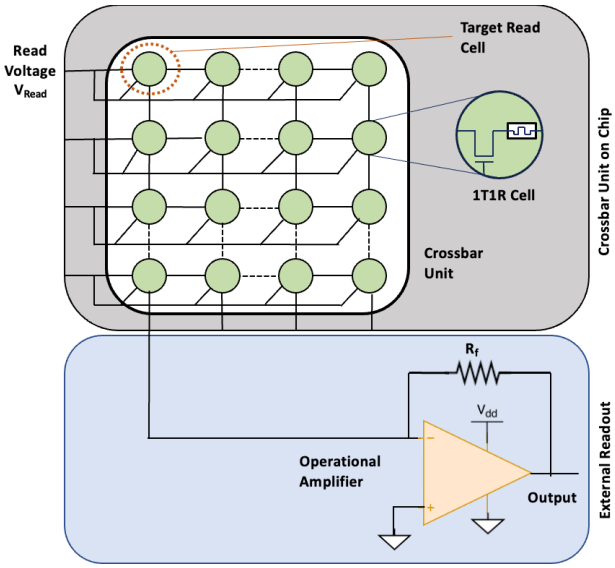



**Figure 36. Test Structure for Read Operation of the Crossbar Assembly**

### Inference Operation Analysis

During inference, we set up seven memristors in a row as a single weight while using the eighth as a bias column. To improve the inference accuracy, gate terminals of the rows that do not have input spikes are grounded. The external readout through an inverting summing amplifier has two resistive parameters, a feedback resistance $R_f$ and a resistance $R_b$ connecting the bias column input to the op-amp (shown in Figure 38). These two parameters can be tuned to scale and shift the crossbar output. Since the input spiking activity is sparse in our continual learning network, we tuned the resistive parameters considering three active input spikes to the crossbar.

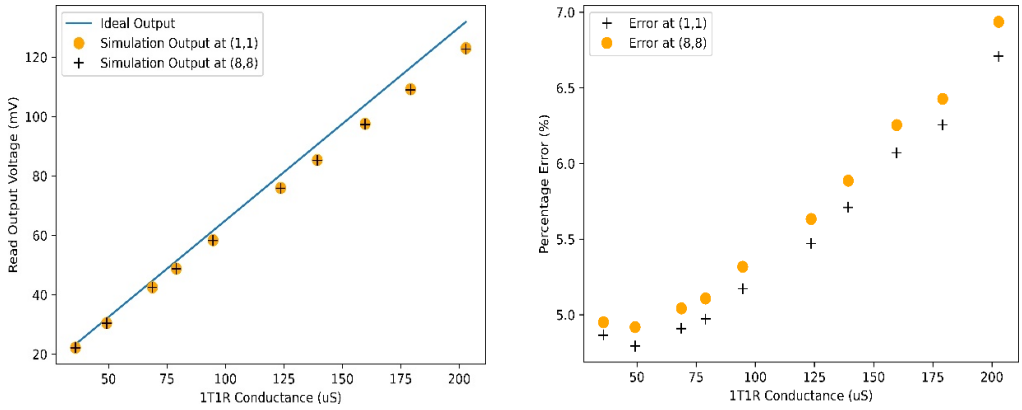



**Figure 37. Read Operation Verification, Ideal Output (Left) and Percentage Error (Right)**

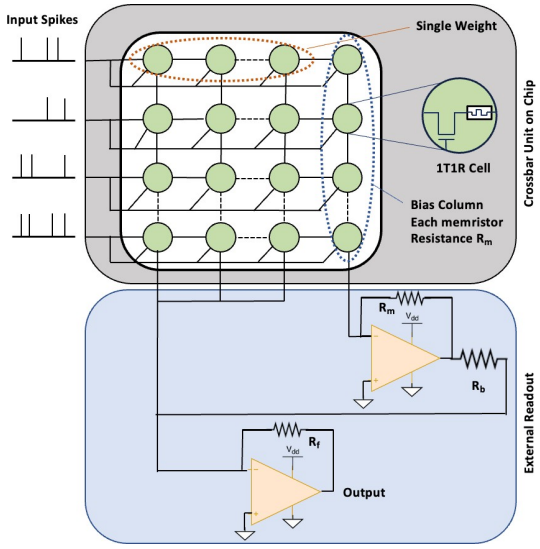



**Figure 38. Test Structure for Inference Operation of the Crossbar Assembly**

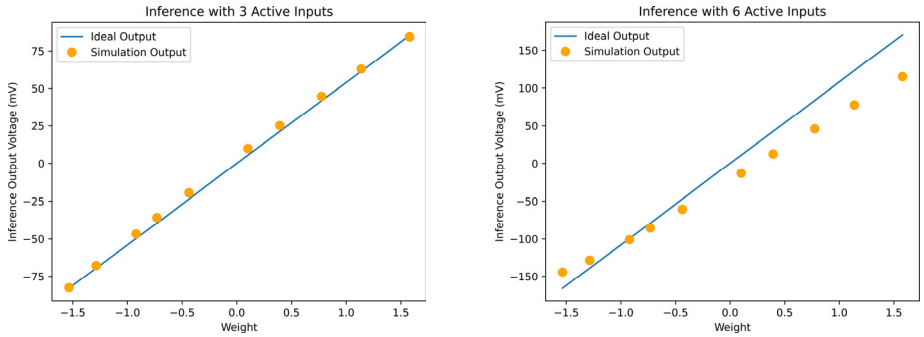



**Figure 39. Inference Operation Verification with 3 (Left) and 6 (Right) Input Spikes**

Figure 39 shows that the observed output in the op-amp for three active inputs, despite including crossbar parasitic, follows the ideal expected output. However, the error increases when the number of active inputs increases to six. This is to be expected since the different numbers of active inputs will lead to different loading conditions. A high variation in the level of input activity might also affect the precision of the inference. However, the external readout allows the option to configure the scaling of the output dynamically in case the input activity varies significantly. The initial scaling parameters can be adjusted according to the expected level of activity and then adjusted according to the observed error.

### Layout design of Genesis-v2

The digital design was enhanced by incorporating a higher parallel processing core count, 5x more memory compared to the Genesis-v1 design and additional user features such as user reconfigurability and scalability were accommodated which can be observed in Figure 40, the SRAM size was increased significantly. The chip also houses four 8x8 memristor crossbars which can be used to evaluate the continual learning algorithm by integrating multiple chips. The final design consists of four major parts covering the total area of $4.85mm \times 4.8mm$. The majority of the chip area is dedicated to the on-chip SRAM,

followed by the digital design due to high parallel computation capabilities and larger network size compared to the mixed-signal design.

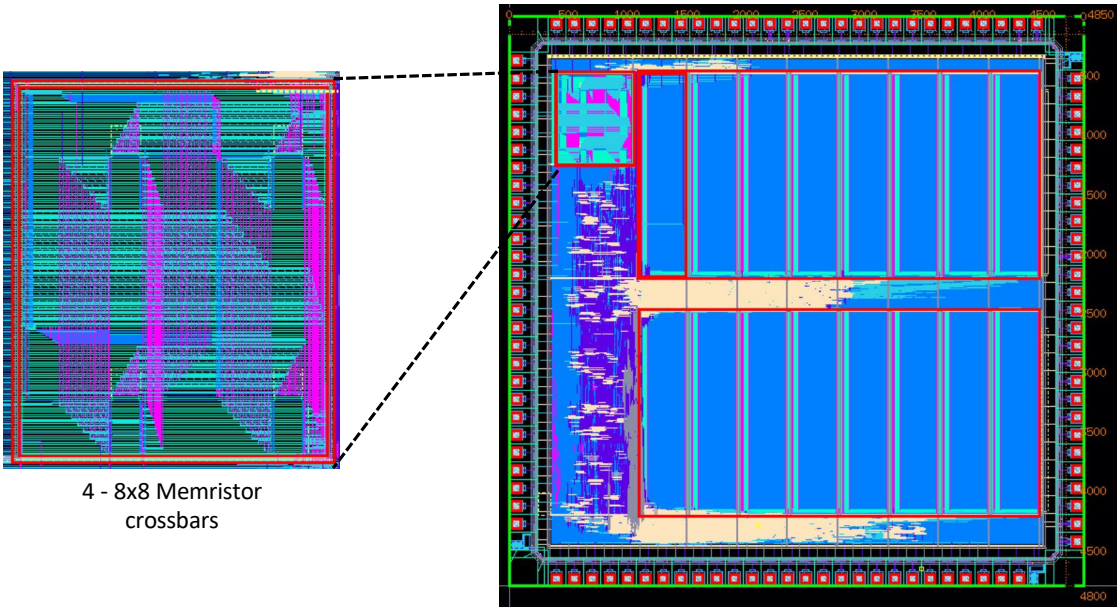



**Figure 40. Genesis-v2 Chip Layout**

### 4.4 Semi Random Deep Neural Networks

Recent studies show that training AlexNet on ImageNet for 100 epochs and a batch size of 512 on the powerful DGX-1 station requires 6 h and 10 min [6]. Training ResNet-50 on ImageNet for 90 epochs and a batch size of 256 on a DGX-1 station requires 21 h. Alternatively, with enough computational resources, such as 3456 Tesla V100 Graphical Processing Units (GPUs), ResNet-50 was trained in 2 min. Such long training times and massive resource demand are impractical for a growing subset of applications, such as lifelong learning systems [8] and edge applications. To boost the performance of DNNs, random projection networks are extended further with additional layers, containing fixed-random weights, as well as concatenating skipped connectivity, similar to that of DenseNet. The use of additional random projection layers is shown to create compositional internal representations of the input data, thereby increasing the separability of the inputs. These representations can act as a method for scaling the proposed learning mechanisms to larger datasets. Likewise, the use of skipped concatenating connectivity allows for earlier layer features to be utilized in the training process to obtain optimal weights. By concatenating the skipped connections, the model is able to alleviate the problem of vanishing gradients, enabling enhanced feature propagation. In addition to these architectural designs, a tensor decomposition technique, known as tensor-train (TT), is leveraged to decrease the memory required of random projection models.

Extreme learning machine (ELM) networks are popularly known for state-of-the-art image classification with minimal complexity and hardware-friendly deployment. The learning methodology of ELM enables it to compute efficiently, as it requires just one single step to train the model. However, it has challenges when training on large datasets, and compression techniques would be necessary to scale to large datasets. Although it is limited to a single hidden layer, the architecture is extended with additional layers, such as

convolution, pooling, normalization, and fully connected layers. However, with each additional layer, all weights are left with their initialized random values and training occurs only on the weights of the output layer using the Moore-Penrose pseudoinverse, similar to the ELM's training. It is believed that by adding these layers, a higher accuracy model can be achieved with insignificant additions to training time. Three architectures explored to address the issue are Convolutional ELM, Convolutional Random Vector Functional Link (CRVFL) Network-Fully Connected and Tensor-Train ELM and Random Vector Functional Link (RVFL).

### 4.4.1 Results

The MSTAR dataset is the first to be experimented with to evaluate these architectures. The set of 5499 training images is used to train these networks and the results of these experiments are shown in Fig. 41. In these plots, the accuracies, training times, Giga Floating-point Operations per second (GFLOPs), and memory size required to store the parameters of the model are presented. In terms of time to train each model, the difference is negligible across all models. As these times are fractions of a minute, the difference between 0.13 and 0.18 is only three s. This fast-training time is due to the less than 2.5 GFLOPs required to train the network. In addition to the quick training time, high performance of more than 90% in all cases can be obtained with these networks on the grayscale 28 × 28 images. The network that achieved the highest performance is most notably the CELM model with skipped connectivity to the first and second fully connected layers. Additional feature space is available to train on in this model. The original input as well as the features of each successive layer are utilized in a large feature space that can become separable. This claim is also supported when observing how the RVFL and CELM with skipped connectivity to the second fully connected layer also achieved comparable accuracies.

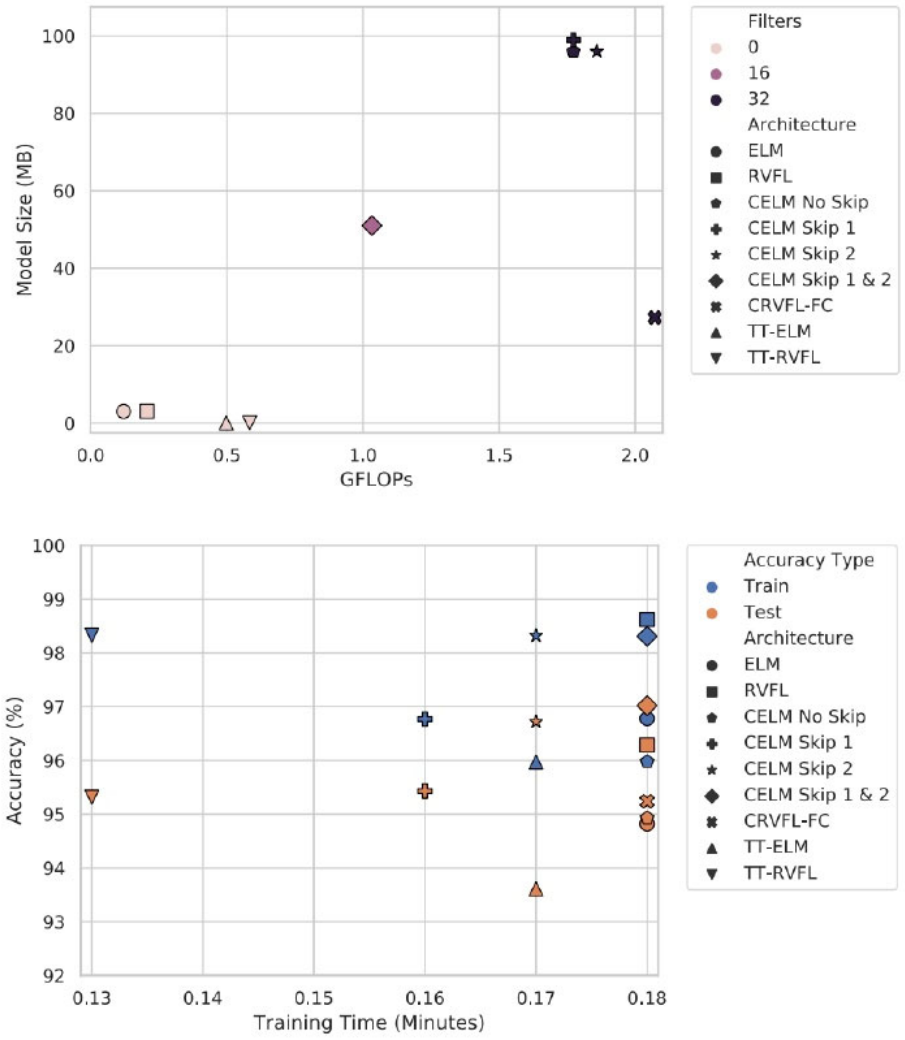



**Figure 41. Training (Blue) and Testing (Orange) Performance of the Random Projection Architectures on the MSTAR Dataset**

## 5.0 CONCLUSIONS

In conclusion, this research report summarizes our efforts to develop lifelong learning solutions suited to energy-constrained edge platforms. During the project, we developed lifelong learning algorithms that are resource-efficient. Our lifelong learning solutions incorporate brain-inspired mechanisms such as activity-dependent metaplasticity and synaptic consolidation to help balance the retention of previous knowledge with learning. Our spike-based continual learning algorithm, TACOS, demonstrates state-of-the-art performance on two continual learning benchmarks with low memory overhead. We also proposed a continual learning mechanism using probabilistic metaplasticity for low-precision memristor weight to enable continual learning accelerators with compute- in-memory capabilities. We developed two mixed-signal lifelong learning accelerators, Genesis-v1 and Genesis-v2, which accommodate optimized versions of our developed lifelong learning solutions for energy-efficient on-device online lifelong learning. Both accelerators support a spike-based network with activity-dependent metaplasticity as the

continual learning mechanism. Genesis-v1 features a digital accelerator with 25 parallel processing elements and 102 kB on-chip SRAM to support network operations. It also contains a mixed-signal design with four 50 × 7 crossbars, closely integrated with the digital accelerator. The crossbar modules support multi-memristor synapses with three 1T1R cells with HfOx-based memristors. The mixed-signal design features on-chip analog LIF neurons to process the output of the crossbar module. The second accelerator, Genesis-v2, features a digital accelerator that improves the capabilities of the previous accelerator with higher user configurability. It has 64 parallel processing units arranged in a two-dimensional array and 512 kB of on-chip memory. Genesis-v2 also features a crossbar assembly with four 8 × 8 crossbars which can emulate multi-memristor weights with up to 6-bits of precision. We also developed a digital lifelong learning accelerator which adopts an 8-bit dual fixed-point number format to maximize continual learning performance at low memory overhead. The accelerator was developed and characterized at the Xilinx ZYNQ-7000 FPGA platform, where it showed the ability to process 30 16 × 16 images per second at 10 MHz operating frequency. Our research efforts towards on-chip online continual learning made progress towards meeting the demands of lifelong learning within the constraints of edge platforms and paved the way towards viable and efficient continual learning.

## 6.0 REFERENCES


[1] Hayes, Tyler L., Kanan, Christopher., " Online continual learning for embedded devices". *arXiv preprint arXiv:2203.10681*, 2022.

[2] McCloskey, Michael., Cohen, Neal J.., "Catastrophic interference in connectionist networks: The sequential learning problem," In *Psychology of Learning and Motivation*, **volume 24**, Academic Press, 1989, pp.109–165

[3] Parisi, German I., Kemker, Ronald., Part, Jose L., Kanan, Christopher., Wermter, Stefan., " Continual lifelong learning with neural networks: A review". *Neural networks*, **113:54–71**, 2019.

[4] Li, Jiajun., Yan, Guihai., Lu, Wenyan Lu., Jiang, Shuhao., Gong, Shijun., Wu, Jingya., Li, Xiaowei., "SmartShuttle., "Optimizing Off-Chip Memory Accesses for Deep Learning Accelerators.," In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, IEEE, 2018, pp. 343–348.

[5] Jouppi, Norman P., Yoon, Doe Hyun., Kurian, George., Li, Sheng., Patil, Nishant., Laudon, James., Young, Cliff., Patterson, David., " A domain-specific supercomputer for training deep neural networks". *Communications of the ACM*, **63(7):67–78**, 2020.

[6] Cheng, Ming., Xia, Lixue., Zhu, Zhenhua, Cai, Yi., Xie, Yuan., Wang, Yu.,Yang, Huazhong., "Time: A training-in-memory architecture for memristor-based deep neural networks.," In *Proceedings of the 54th Annual Design Automation Conference 2017*, DAC '17, New York, NY, USA, 2017. Association for Computing Machinery.

[7] Laborieux, Axel., Ernoult, Maxence., Hirtzlin, Tifenn., Querlioz, Damien., "Synaptic metaplasticity in binarized neural networks". *arXiv preprint arXiv:2003.03533*, 2020.

[8] Soures, Nicholas., Helfer, Peter, Daram, Anurag., Pandit, Tej., Kudithipudi, Dhireesha., "Tacos: Task agnostic continual learning in spiking neural networks.," In *Theory and Foundation of Continual Learning Workshop at ICML'2021*, July 2021.

[9] Boybat, Irem., Gallo, Manuel Le., Nandakumar, SR., Moraitis, Timoleon., Parnell, Thomas., Tuma, Tomas., Rajendran, Bipin., Leblebici, Yusuf., Sebastian, Abu., Eleftheriou, Evangelos., " Neuromorphic computing with multi-memristive synapses." *Nature communications*, **9(1):1–12**, 2018.

[10] Thrun, Sebastian., Mitchell, Tom M., " Lifelong robot learning.," *Robotics and autonomous systems*, **15(1-2):25–46**, 1995.

[11] McCloskey, Michael., Cohen, Neal J., " Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem.," In *Psychology of learning and motivation*, **volume 24**. Elsevier, 1989, pp. 109–165

[12] McClelland, James L., McNaughton, Bruce L., O'Reilly, Randall C., " Why there are complementary learning systems in the hippocampus and neocortex: Insights from the successes and failures of connectionist models of learning and memory.," *Psychological review*, **102(3):419**, 1995.

[13] Pratt, Lorien Y., Mostow, Jack., Kamm, Candace A, Kamm, Ace A, et al., "Direct transfer of learned information among neural networks.," In *Aaai*, **volume 91**, 1991, pp.584–589

[14] Caruana, Rich. "Multitask Learning.," *Machine learning*, **28(1):41–75**, 1997.

[15] Fei-Fei, Li., Fergus, Robert., Perona, Pietro. "One-Shot Learning of Object Categories.," *IEEE transactions on pattern analysis and machine intelligence*, **28(4):594–611**, 2006.

[16] Thrun, Sebastian., Pratt, Lorien., "Learning to Learn: Introduction and Overview". In *Learning to learn,* Springer, 1998, pp. 3–17.

[17] Kudithipudi, Dhireesha., Aguilar-Simon, Mario., Babb, Jonathan., Bazhenov, Maxim., et al. "Biological underpinnings for lifelong learning machines.," *Nature Machine Intelligence*, **4(3):196–210**, 2022.

[18] Kirkpatrick, James., Pascanu, Razvan., Rabinowitz, Neil., Joel Veness, et al. "Overcoming catastrophic forgetting in neural networks." *Proceedings of the national academy of sciences*, 2017. (pp.) 201611835

[19] Zenke, Friedemann., Poole, Ben., Ganguli, Surya., " Continual learning through synaptic intelligence.," In *International Conference on Machine Learning*, PMLR, 2017, pp.3987– 3995.

[20] Laborieux, Axel., Ernoult, Maxence., Hirtzlin, Tifenn., Querlioz, Damien., "Synaptic metaplasticity in binarized neural networks." *Nature communications*, **12(1):1–12**, 2021.

[21] Schug, Angelika Steger Simon., Benzing, Frederik., "Task Agnostic Continual Learning via Stochastic Synapses." https://sites.google.com/view/cl-icml/ accepted-papers?authuser, 2020.

[22] Ebrahimi, Sayna., Meier, Franziska., Calandra, Roberto., Darrell, Trevor., Rohrbac, Marcus., "Adversarial Continual Learning.," In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16,* Springer, 2020. pp.386–402.

[23] Pandit, Tej., Kudithipudi, Dhireesha., "Relational Neurogenesis for Lifelong Learning Agents.," In *Proceedings of the Neuro-inspired Computational Elements Workshop*, 2020, pp. 1–9.

[24] Masse, Nicolas Y., Grant, Gregory D., Freedman, David J., "Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization.," *Proceedings of the National Academy of Sciences*, **115(44):E10467–E10475**, 2018.

[25] Rebuffi, Sylvestre-Alvise, Kolesnikov, Alexander, Sperl, Georg. Lampert, Christoph H., "iCaRL: Incremental Classifier and Representation Learning." In *2017 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR'17, 2017, pp.5533–5542.

[26] LopezPaz, David., Ranzato, Marc 'Aurelio., "Gradient Episodic Memory for Continual Learning.," In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, USA, 2017, pp. 6470–6479.

[27] Ven, Gido M van de., Siegelmann, Hava T., Tolias, Andreas S., "Brain-inspired replay for continual learning with artificial neural networks.," *Nature Communications*, **11(1):1–14**, 2020.

[28] Hayes, Tyler L., Krishnan, Giri P., Bazhenov, Maxim., Siegelmann, Hava T., Sejnowski, Terrence J., Kanan, Christopher., "Replay in deep learning: Current approaches and missing biological elements." *Neural Computation*, **33(11):2908–2950**, 2021.

[29] Mund, Martin., Hong Yongwon., Pliushch, Iuliia., Ramesh, Visvanathan., "A wholistic view of continual learning with deep neural networks: Forgotten lessons and the bridge to active and open world learning." *Neural Networks*, **160:306–336**, 2023.

[30] Kwon , Young D., Chauhan, Jagmohan., Kumar, Abhishek., HKUST, Pan Hui., Mascolo¸ Cecilia., " Exploring System Performance of Continual Learning for Mobile and Embedded Sensing Applications." In *2021 IEEE/ACM Symposium on Edge Computing (SEC)*, IEEE, 2021, pp.319–332.

[31] Van de Ven, Gido M., Tolias, Andreas S., " Three scenarios for continual learning." *arXiv preprint arXiv:1904.07734*, 2019.

[32] Gupta, Vibhor., Narwariya, Jyoti, Malhotra, Pankaj., Vig, Lovekesh., Shroff, Gautam., "Continual Learning for Multivariate Time Series Tasks with Variable Input Dimensions.," In *2021 IEEE International Conference on Data Mining (ICDM)*, IEEE, 2021, pp. 161–170

[33] Seshia, Sanjit A., Sadigh, Dorsa., Sastry, S Shankar., " Toward verified artificial intelligence." *Communications of the ACM*, **65(7):46–55**, 2022.

[34] Fernando, Chrisantha., Banarse, Dylan., Blundell, Charles., et al, "Evolution channels gradient descent in super neural networks." *arXiv preprint arXiv:1701.08734*, 2017.

[35] Lee, Soochan., Ha Junsoo., Zhang, Dongsu., Kim, Gunhee., "A Neural Dirichlet Process Mixture Model for Task-Free Continual Learning." *arXiv preprint arXiv:2001.00689*, 2020.

[36] Fusi. S, Drew. P. J., Abbott. L.F., "Cascade models of synaptically stored memories. Neuron", 45(4):599–611, February 2005.

[37] Leibold Christian., Kempter, Richard., "Sparseness constrains the prolongation of memory lifetime via synaptic metaplasticity". Cerebral Cortex, 18(1):67–77, January 2008.

[38] Horowitz, Mark. " 1.1 Computing's energy problem (and what we can do about it)." In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, IEEE, 2014, pp. 10–14

[39] Senn, Walter., Fusi, Stefano., " Convergence of stochastic learning in perceptrons with binary synapses." *Physical Review E*, **71(6):061907**, 2005.

[40] LeCun, Yann., Bottou, Leon., Bengio, Yoshua., Haffner, Patrick, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE*, **86(11):2278–2324**, November 1998.

[41] Xiao, Han., Rasul, Kashif., Vollgraf, Roland., "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms." *arXiv preprint arXiv:1708.07747*, 2017.

[42] van de Ven¸ Gido M., Tolias, Andreas S., " Three scenarios for continual learning."*arXiv:1904.07734*, April 2019.

[43] Hsu, Yen-Chan., Liu, Yen-Cheng., Ramasamy, Anita., Kira, Zsolt., "Re-evaluating continual learning scenarios: A categorization and case for strong baselines." In *NeurIPS Continual Learning Workshop*, 2018.

[44] Courbariaux, Matthieu., Bengio, Yoshua., and David, Jean-Pierre., "Binary connect: Training deep neural networks with binary weights during propagations." In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2015, pp.3123–3131

[45] Hubara, Itay., Courbariaux, Matthieu., Soudry, Daniel., El-Yaniv, Ran., Bengio, Yoshua., "Binarized neural networks." NIPS'16, Red Hook, NY, USA, 2016, pp.4114–4122

[46] Soures, Nicholas., Helfer, Peter, Daram, Anurag., Pandit, Tej., Kudithipudi, Dhireesha Kudithipudi., "Tacos: Task agnostic continual learning in spiking neural networks." *https://www.nuailab.com/publications.html*, 2021.

[47] Neftci, Emre O., Augustine, Charles., Paul, Somnath., Detorakis, Georgios., "Event-driven random back-propagation: Enabling neuromorphic deep learning machines." *Frontiers in neuroscience*, **11:324**, 2017.

[48] LeCun, Yann., "The mnist database of handwritten digits." *http://yann. lecun. com/exd- b/mnist/*, 1998.

[49] Lopez-Paz, David., Ranzato, Marc'Aurelio., "Gradient episodic memory for continual learning." *Advances in neural information processing systems*, **30:6467–6476**, 2017.

[50] Li, Zhizhong., Hoiem, Derek., "Learning without forgetting." *IEEE transactions on pattern analysis and machine intelligence*, **40(12):2935–2947**, 2017.

[51] Liehr, Maximilian., Hazra, Jubin., Beckmann, Karsten., Rafiq, Sarah., Cady, Nathaniel., "Impact of switching variability of 65nm cmos integrated hafnium dioxide-based reram devices on distinct level operations." In *2020 IEEE International Integrated Reliability Workshop (IIRW)*. IEEE, 2020, pp.1-4

[52] Zyarah, Abdullah M., Kudithipudi, Dhireesha,. "Resource sharing in feed forward neural networks for energy efficiency." In *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*. IEEE, 2017, pp. 543–546

[53] Samajdar, Ananda., Zhu, Yuhao., Whatmough, Paul., Mattina Matthew., Krishna, Tushar., "Scale-sim: Systolic cnn accelerator simulator." *arXiv preprint arXiv:1811.02883*, 2018.

[54] Yakopcic, Chris., Taha, Tarek M., "Energy efficient perceptron pattern recognition using segmented memristor crossbar arrays.," In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2013, pp.1-8

[55] Karia, Vedant., Zohora, Fatima Tuz., Daram, Anurag., Soures, Nicholas., Kudithipudi, Dhireesha., "SCOLAR: A spiking digital accelerator with dual fixed point for continual learning." In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS)*, 2022.

[56] Nilsson, Peter., Shaik, Ateeq Ur Rahman., Gangarajaiah, Rakesh., Hertz, Erik., "Hardware implementation of the exponential function using taylor series." In *2014 NORCHIP*, IEEE, 2014, pp. 1-4

[57] Nandakumar, S. R., Gallo, Manuel Le., Piveteau, Christophe., Joshi, Vinay., et al. "Mixed-precision deep learning based on computational memory." *Frontiers in Neuroscience*, **14**, 2020.

## APPENDIX – PUBLICATIONS AND PRESENTATIONS

### List of Publications

| | |
|---|---|
| Title | Metaplasticnet: Architecture with probabilistic metaplastic synapses for continual learning |
| Authors | F. T. Zohora, V. Karia, A. R. Daram, A. M. Zyarah, and D. Kudithipudi |
| Publication date | 27 April 2021 |
| Venue | IEEE International Symposium on Circuits and Systems (ISCAS), 2021 |

| | |
|---|---|
| Title | Tacos: task agnostic continual learning in spiking neural networks |
| Authors | Nicholas Soures, Peter Helfer, Anurag Daram, Tej Pandit, Dhireesha Kudithipudi |
| Publication date | July 2021 |
| Venue | Theory and Foundation of Continual Learning Workshop at ICML'2021 |

| | |
|---|---|
| Title | Towards near real-time training with semi-random deep neural networks and tensor-train decomposition |
| Authors | H. Syed, R. Bryla, U. Majumder, and D. Kudithipudi |
| Publication date | 21 July 2021 |
| Venue | IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 2021 |

| | |
|---|---|
| Title | SCOLAR: A spiking digital accelerator with dual fixed point for continual learning |
| Authors | V. Karia, F. T. Zohora, Daram, N. Soures, and D. Kudithipudi |
| Publication date | 21 July 2021 |
| Venue | IEEE International Symposium on Circuits and Systems (ISCAS), 2022 |

| | |
|---|---|
| Title | Biological underpinnings for lifelong learning machines |
| Authors | D. Kudithipudi, M. Aguilar-Simon, J. Babb, M. Bazhenov, D. Blackiston, J. Bongard, A. P. Brna, S. Chakravarthi Raja, N. Cheney, J. Clune et al. |
| Publication date | 23 March 2022 |
| Venue | Nature Machine Intelligence |

| | |
|---|---|
| Title | Design Principles for Lifelong Learning AI Accelerators |
| Authors | Dhireesha Kudithipudi, Anurag Daram, Abdullah M Zyarah, Fatima Tuz Zohora, James B Aimone, Angel Yanguas-Gil, Nicholas Soures, Emre Neftci, Matthew Mattina, Vincenzo Lomonaco, Clare D Thiem, Benjamin Epstein |
| Publication date | Accepted |
| Venue | Nature Electronics |

**List of Presentations**

| | |
|---|---|
| Description | Kick off meeting |
| Date of Presentation | November 2020 |
| Presenters | Dhireesha Kudithipudi, Fatima tuz Zohora, Vedant Karia |
| Name of Presentation | Genesis: A neuromorphic chip with lifelong learning on-device |

| | |
|---|---|
| Description | ISCAS 2021 |
| Date of Presentation | May 2021 |
| Presenters | Fatima tuz Zohora |
| Name of Presentation | Metaplasticnet: Architecture with probabilistic metaplastic synapses for continual learning |

| | |
|---|---|
| Description | Theory and Foundation of Continual Learning Workshop at ICML'2021 |
| Date of Presentation | July 2021 |
| Presenters | Nicholas Soures |
| Name of Presentation | TACOS: Task agnostic continual learning in spiking neural networks |

| | |
|---|---|
| Description | International Conference on Neuromorphic Systems 2021 |
| Date of Presentation | August 2021 |
| Presenters | Vedant karia, Anurag Daram |
| Name of Presentation | Metaplasticnet: Architecture with probabilistic metaplastic synapses for continual learning |

| | |
|---|---|
| Description | Intelligence in Chip: Tomorrow of Integrated Circuits 2021 (ICTIC) |
| Date of Presentation | August 2021 |
| Presenters | Dhireesha Kudithipudi |
| Name of Presentation | Neuro-inspired ai accelerators for online & continual learning |

| | |
|---|---|
| Description | Intel Neuromorphic Research Community |
| Date of Presentation | September 22, 2021 |
| Presenters | Dhireesha Kudithipudi |
| Name of Presentation | Neural plasticity for continual learning in spiking neural networks |

| | |
|---|---|
| Description | Energy Consequences of Information Workshop |
| Date of Presentation | February 17, 2022 |
| Presenters | Dhireesha Kudithipudi |
| Name of Presentation | Neuro-inspired architectures for continual learning at the edge: The what, how, and why? |

| | |
|---|---|
| Description | Project update meeting, AFRL |
| Date of Presentation | February 10, 2022 |
| Presenters | Dhireesha Kudithipudi, Fatima Tuz Zohora, Vedant Karia |
| Name of Presentation | GENESIS: A neuromorphic chip with lifelong learning on-device |

| | |
|---|---|
| Description | Neuro inspired Computing Elements Workshop'2022 |
| Date of Presentation | March 31, 2022 |
| Presenters | Dhireesha Kudithipudi |
| Name of Presentation | Neuro-inspired architectures for lifelong learning |

| | |
|---|---|
| Description | Coldspring Harbor Laboratory |
| Date of Presentation | April, 2022 |
| Presenters | Dhireesha Kudithipudi |
| Name of Presentation | Neuro-inspired architectures for lifelong learning |

| | |
|---|---|
| Description | Project update meeting |
| Date of Presentation | May, 2022 |
| Location | Airforce research Lab, Rome, NY |
| Presenters | Dhireesha Kudithipudi, Fatima Tuz Zohora, Vedant Karia |
| Name of Presentation | GENESIS: A neuromorphic chip with lifelong learning on-device |

| | |
|---|---|
| Description | International Symposium on Circuits and Systems, 2022 |
| Date of Presentation | May, 2022 |
| Presenters | Vedant Karia |
| Name of Presentation | SCOLAR: A spiking digital accelerator with dual fixed point for continual learning |

| | |
|---|---|
| Description | NeuroPipe ARAP Program Review |
| Date of Presentation | August, 2022 |
| Location | SUNY Polytechnic Institute,Albany,NY |
| Presenters | Dhireesha Kudithipudi |
| Name of Presentation | Building lifelong learning machines with memristors |

| | |
|---|---|
| Description | Nature Conference on AI, Neuroscience, and Hardware, DZNE |
| Date of Presentation | September, 2022 |
| Location | Bonn, Germany |
| Presenters | Dhireesha Kudithipudi |
| Name of Presentation | Brains, Plasticity Silicon, the pursuit of lifelong learning ai |

| | |
|---|---|
| Description | Energy Efficiency Scaling (EES2) Technical Workshop |
| Date of Presentation | September, 2022 |
| Presenters | Dhireesha Kudithipudi |
| Name of Presentation | Energy-efficient continual learning systems |

| | |
|---|---|
| Description | Lifelong Learning Pathways to Silicon Machines |
| Date of Presentation | October, 2022 |
| Presenters | Dhireesha Kudithipudi |
| Venue | Ellis/crc student retreat |


| | |
|---|---|
| Description | Lifelong Learning Pathways to Silicon Machines |
| Date of Presentation | October, 2022 |
| Presenters | Dhireesha Kudithipudi |
| Venue | Ellis/crc student retreat |


| | |
|---|---|
| Description | Meeting the Energy Demands of Large-Scale Machine Learning Models |
| Date of Presentation | April, 2023 |
| Location | RICE University, Houston, TX |
| Presenters | Dhireesha Kudithipudi |
| Name of Presentation | Large scale machine learning |

| | |
|---|---|
| Description | Project update meeting, AFRL |
| Date of Presentation | December, 2022 |
| Presenters | Dhireesha Kudithipudi, Fatima Tuz Zohora, Vedant Karia |
| Venue | GENESIS: A neuromorphic chip with lifelong learning on-device |

| | |
|---|---|
| Description | Neuroscience and Continual Learning |
| Date of Presentation | March, 2023 |
| Location | Dagstuhl, Germany |
| Presenters | Dhireesha Kudithipudi |
| Name of Presentation | Deep continual learning, dagstuhl seminar |

| | |
|---|---|
| Description | Neuro-inspired Computing Elements 2023 |
| Date of Presentation | April, 2023 |
| Location | San Antonio, Texas, USA |
| Presenters | Dhireesha Kudithipudi |
| Name of Presentation | Panel Talk |

| | |
|---|---|
| Description | Neuro-inspired Computing Elements 2023 |
| Date of Presentation | April, 2023 |
| Location | San Antonio, Texas, USA |
| Presenters | Fatima Tuz Zohora, Dhireesha Kudithipudi |
| Name of Presentation | Memory-Efficient Continual Learning on Memristor Crossbar with Probabilistic Metaplasticity |

| | |
|---|---|
| Description | International conference on neuromorphic systems, 2023 |
| Date of Presentation | August, 2023 |
| Location | Santa Fe, New Mexico |
| Presenters | Dhireesha Kudithipudi |
| Name of Presentation | Unleashing the Power of Neuromorphic Systems: Continual learning in Brains and Machines |

## LIST OF ACRONYMS

| | |
|---|---|
| **1D** | **One-Dimensional** |
| **1T1R** | **1 Transistor 1 Resistor** |
| **2D** | **Two-Dimensional** |
| **AER** | **Address Event Representation** |
| **AI** | **Artificial Intelligence** |
| **ALU** | **Arithmetic Logic Unit** |
| **BNN** | **Binarized Neural Network** |
| **BRAM** | **Block Random Access Memory** |
| **CMOS** | **Complementary Metal-Oxide-Semiconductor** |
| **CMS10LPe** | **CMOS 10LPL Low Power Technology** |
| **CRVFL** | **Convolutional Random Vector Functional Link** |
| **DNN** | **Deep Neural Network** |
| **Domain-IL** | **Domain Incremental Learning** |
| **DRC** | **Design Rule Check** |
| **ELM** | **Extreme Learning Machine** |
| **eRBP** | **Event-driven Random Backpropagation** |
| **FIFO** | **First-In-First-Out Buffer** |
| **FPGA** | **Field Programmable Gate Array** |
| **GFLOP** | **Giga Floating-point Operations per Second** |
| **GPIO** | **General Purpose Input/Output** |
| **GPU** | **Graphical Processing Unit** |
| **HfOx** | **Hafnium Oxide** |
| **ICML** | **International Conference on Machine Learning** |
| **LEF** | **Library Exchange Format** |
| **LFSR** | **Linear Feedback Shift Register** |
| **LIF** | **Leaky Integrate and Fire** |
| **LSB** | **Least Significant Bit** |
| **LUT** | **Look Up Table** |
| **LVS** | **Layout Versus Schematic** |
| **LwF** | **Learning without Forgetting** |
| **MNIST** | **Modified National Institute of Standards and Technology** |
| **MTL** | **Multi-task Learning** |
| **PCB** | **Printed Circuit Board** |
| **PE** | **Processing Element** |
| **PL** | **Programmable Logic** |
| **RAM** | **Random Access Memory** |
| **ReLU** | **Rectified Linear Unit** |
| **RVFL** | **Random Vector Functional Link** |
| **SNN** | **Spiking Neural Network** |
| **SOTA** | **state-of-the-art** |
| **SRAM** | **Static Random Access Memory** |
| **SS** | **Stochastic Synapses** |

| | |
|---|---|
| **SWaP** | **Size, Weight, Area, and Power** |
| **TACOS** | **Task Agnostic Continual Learning in Spiking Neural Networks** |
| **TE** | **Training Elements** |
| **TT** | **Tensor-Train** |