DOI: 10.1177/10943420231178537
FERMILAB-PUB-23-590 (accepted)

Abisko: Deep Codesign of an Architecture for Spiking Neural Networks using Novel Neuromorphic Materials

XX(X):1–23

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Jeffrey S. Vetter[1], Prasanna Date[1], Farah Fahim[3], Shruti R. Kulkarni[1], Petro Maksymovych[1], A. Alec Talin[5], Marc Gonzalez Tallada[1], Pruek Vanna-iampikul[4], Aaron R. Young[1], David Brooks[6], Yu Cao[2], Wei Gu-Yeon[6], Sung Kyu Lim[4], Frank Liu[1], Matthew Marinella[2], Bobby Sumpter[1], and Narasinga Rao Miniskar[1]

Abstract

The Abisko project aims to develop an energy-efficient spiking neural network (SNN) computing architecture and software system capable of autonomous learning and operation. The SNN architecture explores novel neuromorphic devices that are based on resistive-switching materials, such as memristors and electrochemical RAM. Equally important, Abisko uses a deep codesign approach to pursue this goal by engaging experts from across the entire range of disciplines: materials, devices and circuits, architectures and integration, software, and algorithms. The key objectives of our Abisko project are threefold. First, we are designing an energy-optimized high-performance neuromorphic accelerator based on SNNs. This architecture is being designed as a chiplet that can be deployed in contemporary computer architectures and we are investigating novel neuromorphic materials to improve its design. Second, we are concurrently developing a productive software stack for the neuromorphic accelerator that will also be portable to other architectures, such as field-programmable gate arrays and GPUs. Third, we are creating a new deep codesign methodology and framework for developing clear interfaces, requirements, and metrics between each level of abstraction to enable the system design to be explored and implemented interchangeably with execution, measurement, a model, or simulation. As a motivating application for this codesign effort, we target the use of SNNs for an analog event detector for a high-energy physics sensor.

Keywords

microelectronics, codesign, spiking neural networks, neuromorphic materials, LLVM, chiplets

Introduction

Experts predict that computing systems will become more specialized over the coming years (Hennessy and Patterson 2019; Ang et al. 2021; Li et al. 2020a; Dally et al. 2020; Schulte et al. 2015; Vetter et al. 2018), and we are already seeing evidence of this trend within computing architectures for high performance, machine learning, and mobile systems. As this specialization happens, the coordinated development of hardware, software, and algorithms—or *codesign*—is critical to achieving the main goals of improved power, performance, size, and effectiveness (Ang et al. 2021).

To explore the concept of codesign, we are executing the *Abisko* project to engage every level of the computing hierarchy—from materials up through algorithms. The overarching goal of this Abisko project is to develop an energy-efficient spiking neural network (SNN) computing architecture and software system. To this end, we are exploring novel neuromorphic devices that are based on resistive-switching materials, such as tantalum oxide and electrochemical RAM (ECRAM), for implementing these SNN architectures. Taken together, Abisko is engaging experts from across the entire range of disciplines: materials, devices, circuits, architectures, packaging, software, and algorithms.

Our efforts consist of three key objectives. First, we are designing an energy-optimized high-performance neuromorphic accelerator based on SNNs by using new materials. This architecture is being designed as a chiplet that can be deployed in contemporary computer architectures. Second, we are concurrently developing a productive software stack for the neuromorphic accelerator that will also be portable to other architectures, such as field-programmable gate arrays and GPUs. Third, we are creating a new deep codesign methodology and framework for developing clear interfaces, requirements, and metrics between each level of abstraction to enable the system

[1]Oak Ridge National Laboratory
[2]Arizona State University
[3]Fermi National Accelerator Laboratory
[4]Georgia Institute of Technology
[5]Sandia National Laboratories
[6]Harvard University

Corresponding author:
Jeffrey S. Vetter, Oak Ridge National Laboratory
Email: vetter@computer.org

design to be explored and implemented interchangeably with execution, measurement, a model, or simulation.

As a motivation, our effort will examine multiple real-world uses of SNNs. Here, we will describe one of our major design targets: using an SNN as an analog event detector for a high-energy physics (HEP) sensor.

Spiking Neural Networks (SNNs)

The computational paradigm of neuromorphic computing with SNNs offers the opportunity to realize machine learning (ML) operations in hardware with orders of magnitude improvements in energy efficiency when compared with the current state-of-the-art digital hardware (Tavanaei et al. 2019). SNNs are a bio-plausible version of artificial neural networks (ANNs), in which the compute units—or neurons—communicate by using binary-valued spikes. A typical SNN has parameters for the neuron (e.g., firing threshold, leak) and parameters for the synapse (e.g., weight, delay). These SNN features provide the opportunity to explore various material, device, and circuit properties that align with the neuron and synapse dynamics. To fully exploit the benefits of this computational paradigm for real-world applications, the entire compute stack must be redefined for neuromorphic computing. Although there is a tremendous amount of research on the development of devices and materials for neuromorphic computers, this research is often focused on one portion of the computing stack and does not connect to applications, algorithms, or even architectures, in some cases. Without the context of higher levels of the compute stack, it is difficult to evaluate the impact of these new materials and devices on the performance of a full neuromorphic implementation.

Abisko Overview

Our approach has been to assemble experts from each level of the computing stack to design a specific hardware and software system for a chiplet that provides the computational capability of a SNN. Our team will be organized into six research thrusts along layers of the computing stack (See Table 1): algorithms, software, architectures and integration, devices and circuits, and materials. The algorithms thrust will identify computational motifs for SNN algorithms and tailor those motifs for our software and hardware interfaces for SNN implementation. The software thrust will extend the LLVM compiler ecosystem to create a compiler intermediate representation for SNN execution and an asynchronous instruction set architecture for general neuromorphic computing. The architecture and integration thrust will design a conceptual SNN chiplet, including the models and tools needed to integrate it into contemporary packaging technologies. The devices and circuits thrust will model and simulate a range of neuromorphic devices and interconnects by using computer-aided design simulation and TCAD (Stettler et al. 2021) models to provide a standard library of modules to the higher levels. The materials thrust will investigate new resistive-switching materials for energy-efficient neuromorphic devices that offer scalability, CMOS compatibility, and good radiation characteristics while simultaneously investigating more aggressive molecular-based materials and ferroelectric semiconductors. It will also identify new pathways to abstract material performance to the device level by using numerical compact modeling coupled with ML.

Layer	Description
Application	Design applications using SNN as a fundamental computing paradigm
Algorithms	Develop suite of algorithmic motifs for neuromorphic computing; develop simulations of candidate application solutions
Software	Develop meuromorphic programming abstractions implemented as C++ embedded DSL on LLVM/MLIR
Architecture	Design neuromorphic chiplet as integrated in 2.5D or 3D with classic computing components
Devices	Study Neuromorphic devices and circuits as (modified) TCAD models, simulations, and experimental data
Materials	Perform atomic characterization of neuromorphic materials including resistive switching, electrochemical, and more aggressive molecular-based candidates

Table 1. Codesign layers and interfaces for Abisko.

As shown, it is clear that each layer can be refined and optimized internally; however, our deep codesign framework is trying to formalize the abstractions and interfaces across this design space and allow us to drive this process automatically with experiment, simulation, and AI, which is described in the Codesign section.

Motivating Problem: Pixel Detectors for High-Energy Physics (HEP) Collider Experiments

The HEP community is actively searching for a new paradigm to replace the standard model. After the discovery of the Higgs boson in 2012 at the Large Hadron Collider (LHC) (Aad et al. 2012; Chatrchyan et al. 2012), many fundamental questions are still left unanswered, and signals of long-sought new physics have not yet been observed. In preparation for the High Luminosity Large Hadron Collider (HL-LHC), the LHC detectors are undergoing major upgrades, which include the replacement of the tracking systems and the insertion of new subsystems for precision timing to disentangle hard collisions from the background of inelastic interactions per bunch crossing (pileup). The pileup will dramatically increase in the HL-LHC—from an average of 33 up to 200 events. The HEP community is also planning to construct an electron-positron collider for operation soon after the end of the HL-LHC, followed by a hadron collider on a longer time scale. Higgs factories, which can produce the Higgs boson for precision measurements, are also being discussed as tools to discover new physics (of Particles and of the American Physical Society 2021).

Specifically, the pixel detectors currently planned for the HL-LHC experiments include approximately 145,000 pixels per chip and read out an effective 11 bits per hit at 750 kHz, which results in data rates of nearly 3 Gbps in the innermost chips, and nearly 400 pixels are read out per event (Calligaris et al. 2020) (see Figure 1).

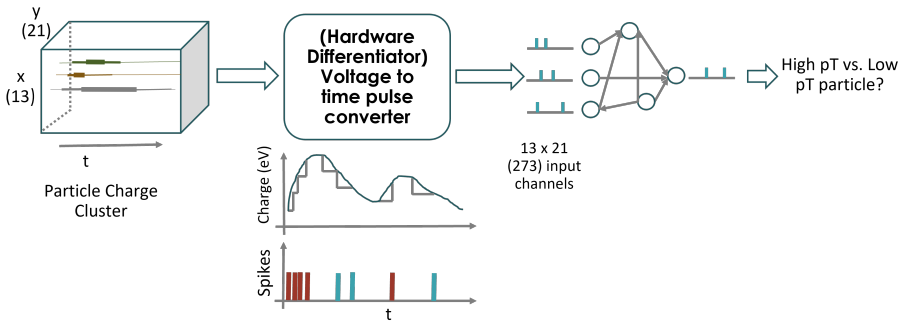


Figure 1. End-to-end in-pixel filtering of particle charge clusters into high pT or low pT samples. Each real-valued incoming signal from the 13×21 array is converted into spike streams. The inter-spike times are related to the rise and fall time of the signal waveform. There are two input spike channels per sensor waveform: one corresponding to the rising edge (in brown) and another for the falling edge (in cyan).

Fermi National Accelerator Laboratory is developing next-generation pixel detectors with high spatial and temporal resolution for future experiments in which data rates will increase by orders of magnitude relative to the HL-LHC. Some approximate numbers are listed here:

- Hit rates in the innermost tracking layers are expected to increase from 3 GHz/cm^2 at the HL-LHC to 10 GHz/cm^2 at a future 100 TeV pp collider (Fleming et al. 2019) (a 3$\times$ increase).
- Consequently, detector granularity will also need to increase by a similar factor with pixel areas going from 2,500 μm^2 to 625 μm^2 (a 4$\times$ increase).
- Reading out precise time information for each pixel in addition to the charge would increase data per pixel from about 4 bits to 16 bits (a 4$\times$ increase for data payload).
- Including full tracking information in the readout or trigger would require an increase in readout rate from 750 kHz to 40 MHz (a 20$\times$ increase).

The eventual solution to this incredible data challenge will involve a carefully optimized combination of many strategies: AI-enabled on-chip clustering that allows readout of reduced cluster-level information rather than raw pixel-level information; AI-enabled lossy data compression; and physics-inspired, in-pixel data filtering.

The power consumption and area of the circuits for readout and data processing must be kept manageable despite the tendency for these to increase with detector resolution and granularity. Strategies for managing power consumption and area include using low-power smaller-geometry technology nodes, power- and area-efficient circuit design, and using novel beyond-CMOS structures, including the memristors described in this work. The competing needs for significant on-detector data processing and minimal power consumption inspire the use of neuromorphic, reconfigurable AI/ML networks for local data processing. The methods and technologies developed to solve the problem will have an impact on various industrial applications that require compact, low-power sensors with edge computing capabilities (e.g., autonomous driving, edge IoT, industry 4.0).

Algorithms

Neuromorphic computers perform computations by using neurons and synapses. An SNN is a network of these neurons connected to each other via synapses. Compared with ANNs used in deep learning applications, SNNs more closely emulate biological neurons and synapses (Schuman et al. 2017). In particular, time is an inherent component of the computation for SNNs, unlike ANNs, in which time has no such significance. In an SNN, neurons can leak charge over time, and it takes time for information to travel from one neuron to another along a synapse; this latency is realized as either synaptic or axonal delay. These temporal characteristics can differ from synapse to synapse and from neuron to neuron. They can therefore add additional dimensions to the processing capability of an SNN.

Unlike deep learning approaches in which back propagation and stochastic gradient descent have been popular training algorithms, a single best training algorithm has not been developed for SNNs. Instead, there are a wide variety of different training algorithms, including those inspired by back propagation but adapted for SNNs (Bohte et al. 2000), evolutionary approaches (Schuman et al. 2016), reservoir computing or liquid state machines (Schliebs et al. 2011), and synaptic plasticity mechanisms, such as Hebbian learning (Ferrari et al. 2008) and spike-timing-dependent plasticity (Caporale et al. 2008). The types of algorithms most suitable for a given neuromorphic implementation depend on the underlying architecture, devices, and materials of the implementation. Each training and learning approach has its own requirements for the neuromorphic implementation, such as dense, feed-forward connectivity for back propagation-like algorithms and sparse, recurrent connectivity for reservoir computing.

The neuromorphic algorithms developed in this work leverage the model of neuromorphic computing proposed by Date et al. (2022b). The neurons in this model are leaky integrate and fire (LIF) neurons that have two parameters: threshold and leak. Each neuron also has an internal state in which signals from incoming synapses are accumulated over time. The neuron leaks value from its internal state based on its leak parameter. A zero leak implies that the neuron instantaneously leaks all of its internal state, and an infinite leak implies that the neuron never leaks its internal state (i.e., it remembers its internal state until it spikes). If at some point

the internal state of a neuron exceeds the neuron threshold, then the neuron spikes. After spiking, the internal state of the neuron is set to the reset state, which is an optional configurable neuron parameter. The spike, considered to be a binary value, is then propagated to all outgoing synapses of the neuron. Each synapse has two parameters: weight and delay. Each synapse multiplies the incoming signal (spike) by its weight, stalls for a time denoted by its delay, and deposits the signal into its post-synaptic neuron. Several applications leverage this model of neuromorphic computing in the literature, including neuromorphic graph algorithms (Kay et al. 2020, 2021; Hamilton et al. 2020b), sparse binary matrix multiplication (Schuman et al. 2021), spiking graph neural networks (Cong et al. 2022), autonomous vehicles (Patton et al. 2021), epidemiological simulations (Hamilton et al. 2020a), classifying supercomputer failures (Date et al. 2018), and many others (Aimone et al. 2022).

SNN Classification

In this work, we employ the Evolutionary Optimization for Neuromorphic Systems (EONS) algorithm to train SNNs for the HEP application (Schuman et al. 2016). EONS considers the parameters of the underlying hardware during the training process, thereby making it quite promising for this neuromorphic codesign effort. Apart from EONS, several other evolutionary algorithms have also been demonstrated for training SNNs (Parsa et al. 2021). EONS begins by evaluating a number of sparse randomly connected SNNs (population of networks), followed by selection and reproduction operations, which include introducing genetic changes (e.g., mutation and crossover) to create the networks (called offsprings) for the next generation of evaluation. The selection operation is determined by the overall fitness score obtained during the evaluation phase, which in a classification task is typically the accuracy measure. This sequence of steps proceeds until the desired fitness score is reached.

As a motivating application, our interest in this algorithm is how it can be applied to a pixel detector for the HEP collider experiment as a demonstration of the neuromorphic codesign effort. With this particular application, we would benchmark the neuromorphic solutions for algorithmic and architecture design requirements for an edge scenario. Although the motivating problem (pixel detector) is from HEP, the methods presented here would be applicable to a broader class of problems that deal with temporal data. The algorithmic approaches developed would inform the underlying software and hardware architecture for an optimized codesign approach.

As discussed in the previous section on pixel detection for the HEP experiments, one of the tasks involves in-pixel filtering: classifying the incoming sensor data into clusters of high- or low-transverse momentum (pT) such that it reduces the overall data transferred to off-pixel processors by retaining and transferring only the high pT clusters.

The pixel detector's simulated dataset consists of charge values (in electrons) of the detected clusters. Each cluster is a sequence of arrays of 13×21 pixels for 20 timestamps every 200 ns. The dataset consists of over 4 million clusters, with pT labels in the range $[-4.9, 4.9]$. The goal of the classification task is to classify the clusters into high pT

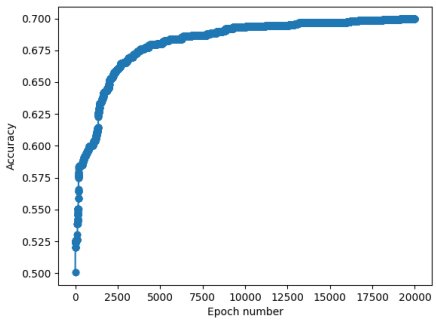


Figure 2. EONS Training curve for the positive pT clusters. In this training simulation, we used only the positive pT samples from the dataset with an equal number of high-pT and low-pT clusters.

(> 0.2 or < -0.2), low positive $[0, 0.2]$ and low negative $[-0.2, 0]$ clusters. For our initial analysis, we consider only the positive samples (i.e., pT ≥ 0). The dataset that we use is not balanced in the two classes. Hence, for the training process, we use only a subset of the high-pT samples equal in number to the low-pT samples.

SNNs operate on binary valued events (or spikes) sparsely distributed in time. As a preprocessing step of the pixel detector dataset, each pixel waveform (real-valued) is translated into temporal spike trains, with the inter-spike interval proportional to the rising or falling time of the waveform. The translation of data into streams of binary events (spikes) would eventually be carried out by the front-end electronics of the pixel. Figure 1 shows the overall scheme of preprocessing the sensor data and classifying it into high- or low-pT clusters with SNNs.

The original dataset provided has charge arrays for each cluster at every 200 ps. To convert the waveform into spike trains, we perform an up-sampling of the original data to capture the timing of the spikes as accurately as possible. The time resolution of the encoding process can be set as a hyperparameter in the classification process. The finer the resolution, the more accurately the data is encoded as spike trains; however, that also results in a higher number of time steps for the SNN simulation. To begin with, we choose 50 ps as the time step size to perform the encoding. Our encoding process results in two spike trains per pixel, with one that encodes the rising points and the other that encodes the falling points in the waveform. There are many details on how the physical pixels generate these rising and falling points. These details can be expressed as parameters to the simulated data encoding process, and through the codesign effort, simulators can be used to evaluate the effect these parameters have on the algorithm and neuromorphic hardware.

After the encoding of real-valued signals into spike trains, each input sample has 273×2 channels (for both rising and falling edges). So, the SNN that we train has 546 input neurons and 2 output neurons. The training phase has several hyperparameters (e.g., EONS configuration parameters, number of epochs, number of time steps for SNN simulation). The key EONS hyperparameters that we consider in our study are population size of the solutions, randomization fraction, and the starting number

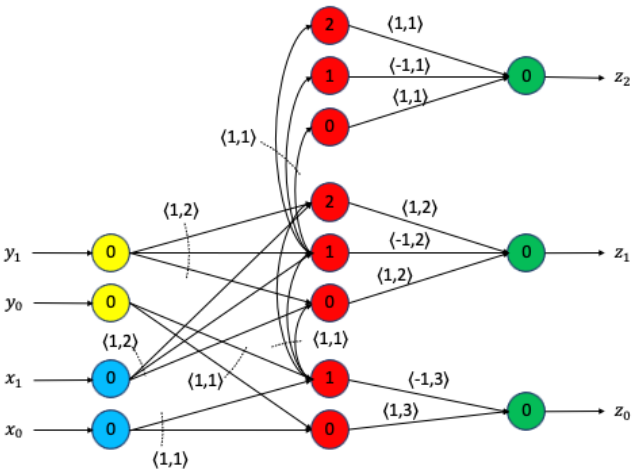


Figure 3. A virtual neuron that takes two 2-bit numbers as input (X and Y) and returns their sum, a 3-bit number, as output (Z).

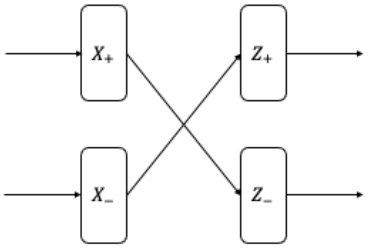


Figure 4. Multiply by -1 function enabled by the virtual neuron.

of hidden nodes and edges within each network. As EONS training proceeds, the networks can potentially grow or shrink depending on the fitness of each generation. The optimization across the entire hyperparameter space is conducted with the Data-efficient Exploration Framework (DEFFE). Our initial EONS training results are shown in Figure 2. Here, we ran the training process for $10,000$ epochs with a population size of 100 networks and a randomization fraction of 0.1. The training accuracy on the balanced dataset reaches 70%. The trained SNN consists of about 700 neurons and $1,000$ synapses.

As next steps, we are exploring the possibility of further reducing the number of input channels passed to the SNN. We noted that the overall number of spikes required to represent each of the signal waveforms is quite sparse, with each channel having at most 4 spikes. Hence, we are studying the secondary encoding process by looking at only the rows or columns of the original sensor charge arrays. This exploration will be carried out using the DEFFE framework.

SNN Regression

The next level of processing for the LHC detectors would require the sensor values to be converted into physics information, such as the particle position and angle, which are real-valued entities. This task will require regression algorithms in which the learning model outputs real-valued data. Although the neurons in an SNN can receive real-valued inputs, their outputs can only be binary. So, the current SNN models cannot return real-valued outputs as

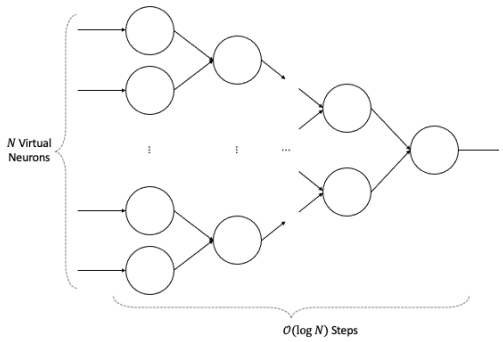


Figure 5. Adding N rational numbers using $\mathcal{O}(N \log N)$ virtual neurons in $\mathcal{O}(N \log N)$ time. Each circle in the figure represents a virtual neuron.

required in a regression task. To effectively perform a regression task, an SNN must be able to (1) encode real-valued numbers up to a certain precision efficiently, (2) add real-valued numbers up to a certain precision efficiently, and (3) multiply real-valued numbers up to a certain precision efficiently.

To address the encoding and the addition functionalities above, we introduce the concept of the virtual neuron, which allows a spike-based neuromorphic computing platform to operate on and output rational-valued data. The virtual neuron is a spatial encoding mechanism that leverages the binary representation of numbers to encode rational numbers on neuromorphic computers. The virtual neuron is also an adder, which takes two rational numbers as input and returns their sum as output. Structurally, the virtual neuron is composed of a group of LIF neurons and synapses that are connected in a particular way. Functionally, the virtual neuron mimics the behavior of an artificial neuron with identity activation. The virtual neuron is an encoding mechanism as well as an adder. It performs the addition operation similar to a ripple carry adder.

Figure 3 shows a virtual neuron that takes two 2-bit numbers X (x_1, x_0) and Y (y_1, y_0) as inputs on the left and returns their sum Z (z_2, z_1, z_0), a 3-bit number, on the right. The output is simply read as a binary-encoded number. For instance, if (z_2, z_1, z_0) equals $(1, 0, 1)$, then the output can be interpreted as the number 5 because $(1, 0, 1)$ is the binary representation of 5. Although other neuromorphic encoding approaches (e.g., rate-based encoding, time-based encoding, binning) take exponential time, exponential space, or exponential energy to encode rational numbers exactly, the virtual neuron takes constant time $(\mathcal{O}(1))$ to encode numbers exactly (Date et al. 2022a). For adding two numbers exactly, the virtual neuron takes linear $(\mathcal{O}(N))$ time, whereas other neuromorphic approaches take either exponential time, space, or energy or are incapable of adding two rational numbers exactly.

We tested the performance of the virtual neuron on 8-, 16- and 32-bit rational numbers and verified that the virtual neuron was able to correctly encode and add two rational numbers (Date et al. 2022a). While the virtual neuron is an essential component for neuromorphic regression, it can also facilitate other applications. Figure 4 shows how the virtual neuron can be used to implement the multiply by -1 function on a neuromorphic computer. Figure 5 shows how

a collection of $\mathcal{O}(N \log N)$ virtual neurons can be used to add N rational numbers in $\mathcal{O}(N \log N)$ time. The virtual neuron is also used to implement a few μ-recursive functions in Date et al. (2022a). Finally, the virtual neuron will be a vital component of neuromorphic compilers and can be easily implemented in any neuromorphic language/compiler, as shown in Figures 6 and 7.

Although the virtual neuron facilitates encoding and addition, we still need an efficient way to multiply two rational numbers exactly on a neuromorphic computer. Multiplication will be the last piece of the neuromorphic regression puzzle. To this extent, our future work consists of developing the virtual synapse, which will be an efficient method of multiplying two rational numbers exactly on a neuromorphic computer. The virtual neuron and the virtual synapse will allow us to perform regression tasks efficiently on the neuromorphic processor. This will allow ML models that run on a neuromorphic computer to output real-valued outputs (e.g., particle position, angle, momentum) as required by the HEP application.

DEFFE (Data-Efficient Exploration Framework)

Domain-specific computing is a viable solution to meet the performance and energy requirements of current and future computing workloads. These computing cores provide several runtime tunable knobs and lead to a co-optimization problem to find the optimal set of architecture knobs for the given workload and workload parameters. One of the challenges of such a co-optimization approach is the size of the search space. By some estimates, the combined search space can easily exceed 100,000 samples.

DEFFE (Liu et al. 2020) addresses this challenge from both methodology and infrastructure perspectives. The ML model in DEFFE can exploit the correlation among different workloads. By casting the performance estimation as transfer learning tasks in ML, the modeling method in DEFFE can drastically reduce the number of samples needed when constructing the performance model of a new workload kernel. On the infrastructure side, DEFFE provides a scalable computing platform to further reduce the run time needed for performance estimation by harnessing the parallelism of HPC clusters. The net outcome is that the run time of a typical performance modeling task, which could take a few months on a single-node computer, is reduced to a few days by using the ML-learning method and further reduced to a few hours when the simulation is executed on a 20-node cluster.

In a codesign environment like the Abisko project, there are many design parameters that are driven by different levels of the design stack. DEFFE automates the process of analyzing these design parameters, thereby extending our ability to explore the trade-offs of various design options. For example, in the case of exploring SNN classification, DEFFE is configured with adjustments for the conversion of the simulated dataset's charge values to spike pulses, adjustments for the hyperparameters for the EONS training process, and adjustments for the design and configuration of the neuromorphic processor. Then, the DEFFE infrastructure can evaluate and collect relevant metrics for each of these configurations by calling our modeling tools with each configuration and then parsing the output to collect metrics from that configuration. DEFFE evaluates the specific configurations in parallel and collects the results into a single table. By using DEFFE, the setup of experiments is simplified. The DEFFE configuration file specifies the adjustments that can be tuned and the metrics to be collected. Scripts are added and used by DEFFE to take a specific configuration and evaluate it. DEFFE then handles the launching of each job on an HPC cluster to explore the design space.

General-Purpose Computing Outlook

Although most algorithms for neuromorphic computers are ML approaches that determine an SNN to perform a particular task, there is also an opportunity to leverage the underlying computational characteristics of neuromorphic systems to perform more general-purpose computation. Neuromorphic computers are inherently massively parallel, perform event-driven computation, and have colocated processing and memory. These characteristics are useful for solving a variety of other types of problems, including graph algorithms, such as shortest path (Schuman et al. 2019; Aimone et al. 2020); modeling epidemics (Hamilton et al. 2020a); constraint satisfaction (Yakopcic et al. 2020); and generating Markov random walks (Severa et al. 2018a). However, the lack of programming abstraction for more general-purpose uses of neuromorphic computing has contributed to the lack of work in this area.

In recent years, several advancements have been made toward the general-purpose applications of neuromorphic computing. First, the Turing-completeness of neuromorphic computing has been proven—this provides a compelling theoretical argument that neuromorphic computers are capable of general-purpose computation (Date et al. 2022b). In fact, from a theoretical standpoint, neuromorphic computers can perform all those operations that today's computers can perform. The interesting question along this line of research is as follows: if a neuromorphic computer is made to do everything that today's CPUs/GPUs can do, then does it retain its energy advantage? The second advancement toward general-purpose neuromorphic computing has been the development of the computational complexity theory for neuromorphic algorithms (Date et al. 2021). This allows us to compare neuromorphic algorithms to their conventional counterparts in the most theoretically fair manner possible. Some efforts have also been made toward developing the theory of energy complexity (Kwisthout and Donselaar 2020) and benchmarking the energy consumption and latency of neuromorphic systems (Kösters et al. 2022). These efforts, along with a strong focus on developing a neuromorphic programming language, a neuromorphic compiler, and a neuromorphic runtime system, are critical to fully realizing the potential of general-purpose neuromorphic computing.

Software

The Abisko software aims to implement a powerful system-level C++ neuromorphic programming infrastructure that enables the integration of conventional and neuromorphic compute architectures. At the same level, the Abisko software has a strong focus on HPC technologies. One

objective for the Abisko software stack is to scale the main neuromorphic processes (e.g., data encoding and decoding, SNN training and inference, and structural analysis of SNNs) to leverage their optimization at HPC scales as an effort to deploy an HPC framework for neuromorphic computing development.

Neuromorphic Programming Language

Within Abisko, the program specification is based on a domain-specific language—the *Aurora* programming language. The *Aurora* language is a functional and declarative language implemented by using the Embedded Compiler Construction (`eCC`) framework (Gonzalez-Tallada et al. 2022). This framework enables the embedding of domain specific languages within the C++ programming language. *Aurora* is defined with the `eCC` API, and the language is embedded in C++ and reuses all the existing C++ infrastructure. This strategy has proven extremely useful in heterogeneous programming because it enables domain-specific compiler technology (e.g., ML (Jin et al. 2020), quantum computing (McCaskey and Nguyen 2021), tensor algebra (mli 2023)).

The *Aurora* language is designed to specify the computation as an SNN. The network is described as a graph composed of nodes that correspond to neuromorphic computing elements such as neurons and synapses. Edges in the graph represent connections among these computing elements. This program representation is widely used across common neuromorphic programming systems. In general, these systems originated from neusoscience frameworks (e.g., Brian (Stimberg et al. 2019), NEST (Gewaltig and Diesmann 2007), Arbor (Abi Akar et al. 2019), Nengo (Sharma et al. 2016), and Rasmussen (2018)) or even from deep learning frameworks (e.g., PyTorch (Paszke et al. 2019), TensorFlow (Chien et al. 2019)), which have been adapted or extended to provide support for neuromorphic computing. Additionally, these neuromorphic systems rely on application-level languages (e.g., Python) that target vendor-specific coprocessor types, thereby limiting benchmarking and portability. The *Aurora* language was designed to incorporate the advances from all these past experiences. For example, the *Aurora* language includes graph data types and operators (e.g., *Node, NodeSet*, *Layer*) to specify the nodes in the graph and native types (e.g., *Edge*, *EdgeType*) to represent connections. *Aurora* also includes specific native types and operators to define data models associated to the elements of the graph (i.e., nodes and edges). Within *Aurora*, both aspects are kept totally separate. In this regard, *Aurora* follows a similar approach as followed by Kozloski and Wagner (2011).

The code in Figure 6 shows an extract of the *Virtual Neuron* specification. Several layers are defined (lines 1–2), and then a 1D view is used to create the neurons in each layer (lines 4–7). Data views are generated when a *Layer* is used with `operator []`. The *View* data type is introduced to define a logical grid organization of the nodes within a layer. In this case, the 1D view is only used to describe how the nodes in the layers must be created. But when using a view, no data layout in memory is assumed. The compiler will later derive which data layout is optimal for the final graph implementation. *Aurora* also

```
(1)  def x_pos("x_pos", Layer); // positive number X

(2)  def bits_pos("bits_pos", Layer); // positive bit neurons

(3)  use create_neuron("create_neuron", (Real=0, Real=-1.0) ->* LIF);

(4)  x_pos[0,positive_precision-1] = create_neuron(0);

(5)  bits_pos[0,positive_precision][0] = create_neuron(0);
(6)  bits_pos[0,positive_precision][1] = create_neuron(1);
(7)  bits_pos[1,positive_precision][2] = create_neuron(2);

(8)  var range(Range);
(9)  range = (0, positive_precision-1);

(10) Connect(x_pos[range],
                bits_pos[range][0,1]) = Synapse("weight"_m = 1.0,
                                                "delay"_m = Real(range+1));
(11) Connect(x_pos[1, positive_precision-1],
                bits_pos[range][2]) = Synapse("weight"_m = 1.0,
                                              "delay"_m = Real(range+1));
```

Figure 6. Extract of Virtual Neuron graph specification in *Aurora*. Layers are defined and combined with *Range* type variables, which generates *Views* of the nodes in the graph layers. Usage of *Connect* operator to connect elements in the graph. Usage of *Synapse* type, which is a derived type from *EdgeType* native type in *Aurora*.

```
def("create_neuron", (Real=0, Real=-1.0)  ->* LIF,

  Begin(in V_th, in internal_state) {

    var neuron(LIF);

    neuron["V_th"] =  V_th;
    neuron["V_m"] = internal_state;
    neuron["tau_m"] = -1e-6;
    neuron["t_ref"] = 0.0;
    neuron["E_L"] = neuron["V_m"];

    Return(neuron);

  } ); // def
```

Figure 7. *Aurora* definition of a function to generate a LIF instance and update internal parameters of the LIF data model.

includes the *Range* data type to represent value ranges of primitive types such as natural, integer numbers. Ranges become useful when describing data views. In the example, nodes in layers are connected with edges (lines 8–9). Ranges appear in a 2D view. In frameworks based on application-level languages (e.g., Python), this would require the use of iterative structures in which in each iteration of the graph elements would have been connected. Moreover, this would usually determine the memory layout to implement the graph. In *Aurora*, data views remove all of this, and the code is not used to implement the graph; instead, it is used to describe the graph and allow the compiler to find an optimal implementation. *Aurora* includes specific constructs and operators to connect the elements in the graph. For example, the *Connect* operator connects a pair of nodes (lines 10–11). A 2D view and a *Range* are combined to relate the nodes from two layers (e.g., *x_pos* and *bits_pos*). The nodes correspond to *Neurons* generated in the *create_neuron* function. The edges correspond to *Synapses*. In both cases, these two types correspond to data models and are completely orthogonal to the actual graph structure.

Data models in *Aurora* follow a similar approach as in NEST. Figure 7 shows the code for the function in

which a *Neuron* is created. The *Neuron* corresponds to the LIF data model. A model instance is generated with a variable of type *LIF*. The data members of the LIF model are updated by using dictionary-based access methods (Figure 7). This is a common feature used in many previous neuromorphic frameworks, and *Aurora* includes it. A data model specification is divided into three sections (Figure 8). The first two sections correspond to constants and global variables associated with the model, which are shared by all instances of the model. Section *Data* corresponds to values for each instance of the model (i.e., per each *Neuron* in the example). The *Aurora* data models allow the compiler to identify read-only data, data shared among all model instances, and per-instance data. This is essential to enabling optimal data layouts (e.g., Struct-of-Arrays instead of using Array-of-Structs) or using specific memory levels within the memory hierarchy (i.e., map constant values to GPU-specific memory components such as constant and texture memory and data values to shared memory). In general, data models in *Aurora* allow for introducing any neuromorphic functionality (e.g., spike recorders, encoding and decoding schemes). These can be introduced as data models and instantiate them within the graph as layers connected with other elements such as neurons and synapses. These functionalities and others are supported as libraries that contain data models similar to how high-level programming languages support data type abstractions (e.g., *vectors*, *lists*, *queues*, STL in C++, modules in Python).

In *Aurora*, the computations associated with a data model are specified as *Phases*. A *Phase* identifies a computation that by default is totally parallel among the instances of the data model. In the case of the *LIF* model, the *update* phase includes the code that describes the dynamic of this neuron type (we omit its code for brevity). Similarly, the *Synapse* model includes its computations as phases. *Aurora* includes `forall` operators to indicate the execution of a phase across a subset of graph elements (e.g., a *NodeSet*, a *Layer*, or a single *Node*). Consequently, the computation of an *Aurora* program corresponds to the execution of a sequence of phases over the graph elements. Each phase activates a set of graph elements in parallel. The *Aurora* language deploys a massively parallel execution model that matches the inherent parallelism in SNNs. This makes it possible for common tasks within the neuromorphic domain (e.g., data encoding and decoding, inference and training processes) to be accelerated with HPC technologies. Abisko targets the acceleration of existing optimization algorithms (e.g., EONS (Schuman et al. 2020), Spike Timing-Dependent Plasticity (Caporale et al. 2008)) or back propagation–based training algorithms (e.g., Slayer (Shrestha and Orchard 2018), Whetstone (Severa et al. 2018b)).

Overall, *Aurora* presents two main advances within the neuromorphic computing domain. First, its declarative approach totally decouples the graph and model descriptions from their actual implementation. The memory layout of graphs and models and when memory is actually allocated is never under the programmer's control. Second, its massively parallel execution model makes explicit the available parallelism inherent in SNNs. With these two features, it is possible to design the necessary compiler technology for high-performance neuromorphic computing.

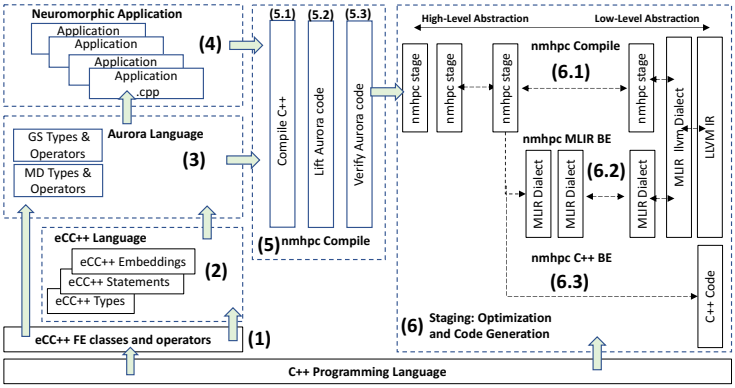
```
def LIF("LIF", Model(Neuron), Begin() {

  def Constants("Constants", Begin() {
    def("tau", Real=10.0);
    def("c", Real=250.0);
    def("t_ref", Real=2.0);
    def("E_L", Real=-70.0);
    def("I_e", Real=0.0);
    def("V_th", Real=(-55.0 - E_L));
    def("V_min", Real=-min(Real));
    def("V_reset", Real=-70.0 -E_L);
    def("with_refr_input", Bool=false);
  } ); // Constants

  def Variables("Variables", Begin() {
    def("P30", Real);
    def("P33", Real);
    def("RefractoryCounts", Int);
  } ); // Variables

  def Data("Data", Begin() {
     def("y0", Real);
     def("y3", Real);
     def("r", Int;
     def("refr_spikes_buffer", Real);
   } ); // Data

   def("Phases", Begin() {
      def("update", Phase);
   } ); // Phases

} ); // Model LIF
```

Figure 8. *Aurora* example of a model specification for a LIF neuron type. Constants, global variables, and per-neuron data are separated in different sections.

Neuromorphic Compiler

The Abisko compiler, `nmhpc`, targets a distributed-memory heterogeneous architecture composed of nodes with conventional processors (i.e., CPUs, GPUs) and neuromorphic coprocessors. For the former case, `nmhpc` implements analysis, optimization, and code generation phases for three main tasks. `nmhpc` must analyze the graph that represents the neuromorphic *Aurora* program to generate an optimal memory layout for the graph implementation. Second, the `nmhpc` compiler must generate code for all computational kernels associated with model phases described within the neuromorphic program. Finally, the `nmhpc` compiler must elaborate the graph cuts to distribute the graph computations across the computational resources. This work distribution happens at two levels: between the nodes and then inside the node, thereby distributing work between the conventional processors and coprocessors. The graph cutting process implies the use of performance models to estimate both the weight of computations and the communications. The design of the `nmhpc` compiler follows a strategy based on targeting different runtime systems. The Abisko runtime systems (1) implement the required functionalities to instantiate the graph that represents the program, (2) operate with it (e.g, allocate the graph, deploy a namespace for nodes and edges and primitives to access the data models), and (3) support the execution of phases (e.g, threading, SIMD, GPU acceleration), message passing primitives for communication across the graph cuts, and graph cut algorithms based on performance models.

The Abisko compilation process is shown in Figure 9. The actual *Aurora* embedding happens in steps 1, 2, and 3 in Figure 9, where C++ libraries implement the *Aurora* language as an API. *Aurora* code (step 4 in Figure 9) is



(1) eCC++ classes and operators embedded in C++.

(2) eCC++ language entities describing the Types, Embeddings, Statements and other extensions.

(3) Definition of the Aurora neuromorphic language build on top of the eCC++ framework. Graph Specification (GS) and Model Definition (MD) types and operators compose the Aurora neuromorphic language.

(4) Application coded in C++ but using Aurora front-end as a C++ API.

(5) Compilation process of the application: C++ compile the sources that implement the eCC++ framework, the Aurora neuromorphic language and the application implementation (5.1). Execution of the Abisko binary which lifts the program representation (5.2). Verify correctness of the lifted program (5.3).

(6) Staging of the optimization process and code generation. (6.1) Staging happens within the nmhpc++ compiler framework in several stages. Each stage lowers the program representation. (6.2) At some stage it is possible to generate code targeting an MLIR dialect and continue the lowering within MLIR. (6.3) At some stage it is possible to generate C++ code targeting the Abisko RTLs.

Figure 9. Abisko compilation process. Abisko is embedded in C++ using the eCC framework. *Aurora* programmers do the actual programming using the C++ API that implements the Aurora language within C++.

compiled with the C++ standard compiler to generate an executable that links with the libraries that implement the *Aurora* language. Its execution instantiates the neuromorphic program representation by using a multilevel intermediate representation (IR) similar to that of the MLIR (Lattner et al. 2021) compiler infrastructure. Program verification is based on this IR (step 5 in Figure 9), which is later lowered for optimization and final code generation (step 6 in Figure 9). After the program is verified, `nmhpc` is designed to allow two paths within the compilation process. One option is to target the MLIR/LLVM infrastructure for optimization and final binary generation. For this purpose, `nmhpc` includes a back-end code generation phase that emits MLIR code and allows the output of `nmhpc` to enter the MLIR compiler infrastructure. This makes it possible to implement lowering stages of the `nmhpc` IR to reach a program representation using the MLIR LLVM dialect (path 6.1 in Figure 9). It is also possible to lower the `nmhpc` IR to reach an MLIR-based representation by using other dialects located at higher levels of abstraction within MLIR (path 6.2 in Figure 9). Finally, `nmhpc` includes code generators that target C++ (path 6.3 in Figure 9).

When targeting a neuromorphic coprocessor, the main task of the `nmhpc` compiler is to generate bit streams from the actual program representation in *Aurora*. These bit streams conform to the specifications of the neuromorphic architecture. The `nmhpc` compiler must generate configuration bit streams or packets to configure the neuromorphic coprocessor. Similary, the `nmhpc` compiler generates packets that result from encoding/decoding processes to be sent to and from the neuromorphic coprocessor. The `nmhpc` also targets a runtime system in which the primitives for configuring and operating the neuromorphic coprocessor are implemented. In this regard, the neuromorphic coprocessor is treated as a conventional coprocessor, meaning that the Abisko software includes similar functionalities as CUDA/HIP to move data to and from the host and device and to activate and offload a computation.

Architectures

One hallmark of contemporary microelectronics research is heterogeneous integration. By allowing different semiconductor technologies (e.g., digital, analog, DRAM, carbon nanotube) to be closely integrated by 2.5D (e.g., silicon bridge and interposer) or 3D (e.g., TSV, hybrid bonding), heterogeneous integration can achieve better performance at a lower cost compared with monolithic integration. Our objective in this task is to use heterogeneous integration to realize an agile codesign platform. More specifically, we plan to use 3D integration to implement traditional von Neumann processing cores (e.g., open-source RISC-V cores) with neuromorphic coprocessors. At a high level, our heterogeneous integration approach has the following distinct advantages: (1) it enables rapid feedback between the realistic applications and the material and device research choices, (2) it enables the intelligent design-of-experiment to explore novel material science and device research by leveraging intelligent ML sampling techniques, and (3) as a demonstration vehicle of neuromorphic computing, the traditional von Neumann cores can perform crucial data prepossessing tasks for neuromorphic training in real time.

The architecture task is positioned in the middle of the codesign stack and ties together the higher-level neuromorphic algorithms thrust with the lower-level neuromorphic circuits thrust. The architecture thrust takes the circuit-level implementation of a neuromorphic element and is tasked with defining the architecture for how multiple core circuits will be structured to build up a complete neuromorphic chip. There are many ways a neuromorphic chip could be designed when considering impacts on the operation of the devices and algorithmically how the system can be used. As part of the codesign effort, the architectures thrust will evaluate different architectural decisions concerning the design choices in the software stack, circuit design, and device properties. We are leveraging modern tools (e.g., SIAM (Krishnan et al. 2021)) and high-level design languages (e.g., high-level synthesis and Chisel) to rapidly evaluate and prototype architecture design options. Additionally, we are leveraging DEFFE to further automate and accelerate design space exploration.

Multiple neuromorphic architectures exist, including mixed-analog architectures (e.g., Neurogrid (Benjamin et al. 2014), Braindrop (Neckar et al. 2019), and BrainScaleS (Schemmel et al. 2010)) and digital neuromorphic designs (e.g., TrueNorth (Akopyan et al. 2015) and Loihi (Davies et al. 2018)). However, the Abisko architecture effort is novel based on the codesign integration with a unique software compiler, the exploration of new neuromorphic devices, and

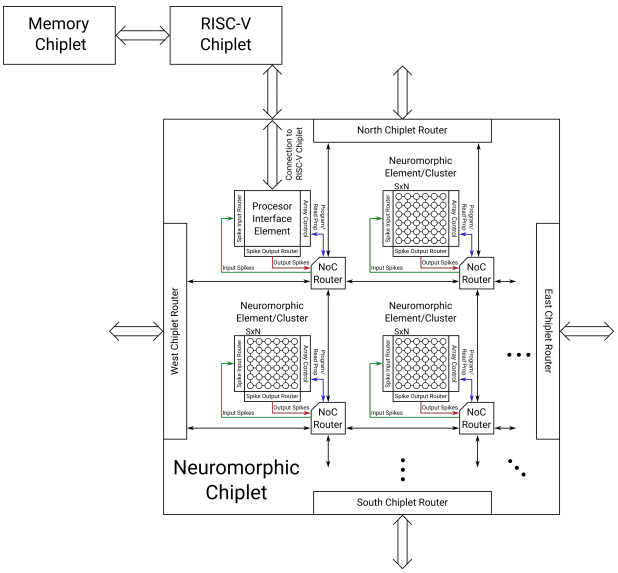


Figure 10. Overview of Abisko chiplet design.

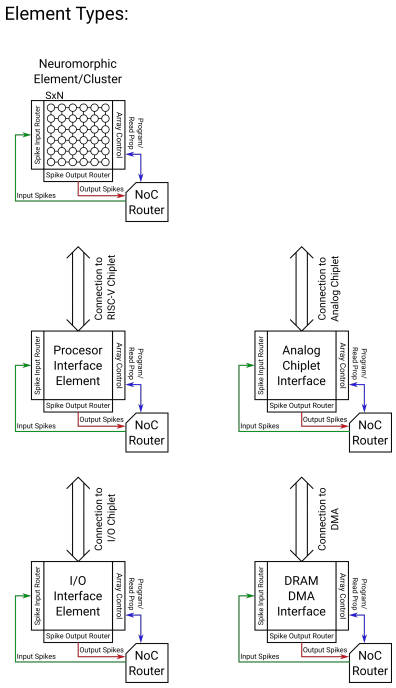


Figure 11. Abisko chiplet design with novel neuromorphic materials.

the exploration of new microelectronic design trends and standard interfaces. Specifically, the architecture thrust is exploring three main fronts in the design of a neuromorphic processor: (1) the use of chiplets, (2) the interface between the neuromorphic and traditional von Neumann processors, and (3) the benefits of new 3D chip processes. The following sections elaborate further on each of the research fronts.

Chiplet-Based Design

A chiplet-based design uses smaller functional dies, called chiplets, that are built independently and then are connected together with an interposer to form a single 2.5D or 3D chip. As highlighted by Stark (2019), there are many benefits to a chiplet-based design, including reduced investment costs, lower cost of specialization, lower production costs, shorter time-to-market, lower supply risks for OEMs, and simpler architectural partitioning. Many of these benefits come from the reusability and functional partitioning that chiplets provide. The ability to use mixed-process nodes is of great benefit to reduce cost and for novel device designs, which benefit from specialized process nodes. Mixed-process nodes also provides additional benefit in the Abisko project as we explore various neuromorphic devices and materials, which can be far less mature than traditional semiconductor devices.

As part of the Abisko architecture effort, we explore how a neuromorphic architecture could benefit from the use of chiplets. From a high-level perspective, there are three main chiplet types that constitute a chip (as illustrated in Figure 10): traditional von Neumann chiplet(s), neuromorphic chiplet(s), and memory chiplet(s). The von Neumann chiplet manages the operation of the neuromorphic chiplet and could be used to configure networks, interface with traditional computers and sensors, and implement more advanced on-chip learning algorithms. The memory chiplet provides shared storage for the von Neumann and neuromorphic chiplets.

As shown in Figure 11, the neuromorphic chiplet evaluates the SNN and is built from digital or mixed-signal circuits that implement the neurons and synapses. There are other chiplets that could also be included in the design. For example, a hardened I/O interface could directly convert a data stream into spikes to be fed directly into the neuromorphic chiplet, thereby bypassing slower software-based spike conversion by the von Neumann processor.

There are many open questions that we are addressing in this effort, including how the neuromorphic chiplet should be structured. The new devices are used to build neuromorphic circuits, then the circuits are used to construct a neural core that implements some number of neurons, and finally the cores must be connected together to form the chiplet. There are also scalability questions. How many neurons and synapses should a chiplet implement? How can networks scale to multiple chiplets on a chip? How about across multiple chips in an accelerator? Once the chip is built, how will the software stack compile code to be mapped onto the hardware? We plan to answer these questions by modeling different architectural designs and analyzing the designs for power, area, scalability, and performance.

Interface and Communication

There are multiple challenges with designing the interfaces between chiplets and between traditional and neuromorphic processors and still more challenges with communication at multiple scales within the neuromorphic processor.

Chiplets are still emerging as a standard in chip design, and multiple chiplet interfaces and standards are being proposed, including UCIe (Sharma 2022), BoW (Ardalan et al. 2020), AIB (Kehlet et al. 2017), OpenHBI (McLellan 2020), and Cadence Ultra-Link. As part of the chiplet-based design, we are evaluating if one of the emerging standards can be used to connect the chiplets or if a custom neuromorphic interface would be better suited for spike-based communication. For each chiplet in the complete design, the physical interface between the components must

be defined. The interface will impact the number of vias required to make the connection and the bandwidth and latency available over the link.

There are also multiple options on how to interface between a von Neumann and a neuromorphic processor. One commonly used method is to define a packet structure and have the traditional processor generate configuration, input, and execution packets. These packets are then sent to the neuromorphic processor, which parses and executes the packets and generates output fire packets and status packets. A DMA engine is commonly used to send commands stored in memory as packets to the neuromorphic processor. Another method would be to treat the neuromorphic processor as a memory-mapped accelerator with addresses defined to control the operation of the accelerator. A third method could be adding custom instructions to the von Neumann processor and having these instructions directly executed on the neuromorphic processor for a very tight integration.

There is also the challenge of how to route the native spiking packets at different scales, internally within the chiplet, between chiplets, and between chips as the design is scaled. Spike-based communication is different from traditional communication (Young et al. 2019). The challenge comes both from the large number of messages that contain a small amount of data and from the time-sensitive nature of the spike packets. SNNs rely on binary fire events, which must be routed to potentially thousands of destinations per fire event. These spikes are typically encoded into packets through Address-Event Representation (AER). Traditional AER packets only store the address or destination of the event, and the presence of the packet represents a fire. Neuromorphic communication systems can be categorized by the network structure of the routers and by the information used to route packets, as shown in Figure 12. The most common router structures are grid and hierarchical tree routers. Packets can then be routed based on a source address or a destination address. Some neuromorphic systems have combinations of these options with different levels of routing using different network structures or routing methods. The challenge for the Abisko design is how to handle spike packet routing within a chiplet, between chiplets, and between chips as the scale of the network increases.

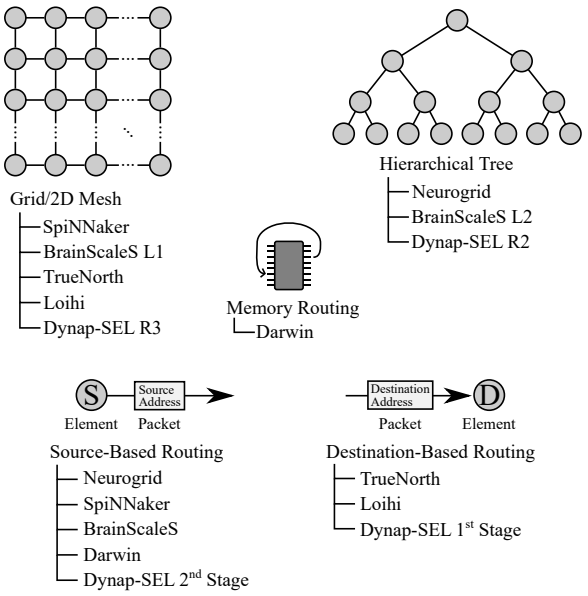


Figure 12. Graphical summary of neuromorphic hardware communication systems. The top half of this figure summarizes the routing schemes used by the neuromorphic systems, and the bottom half summarizes the routing methods (Young et al. 2019).

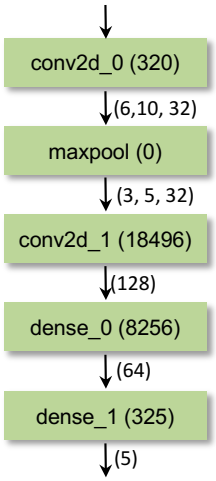


Figure 13. Topology of the CNN model. The numbers in each module indicate the number of weights, and the numbers between each component indicate the sizes of I/O tensors. The CNN model is implemented as a five-stage pipeline.

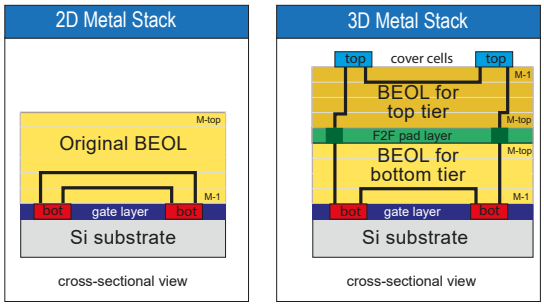


Figure 14. Cross-sectional view of the material stack.

Benefits of Face-to-Face 3D Integration

As a preliminary study, we evaluate the benefit of face-to-face die bonding. To demonstrate the benefits of codesign, we implement a digital convolutional neural network (CNN) model from a pixel detector used in HEP experiments. The top-level of the CNN implementation is a five-stage pipeline, in which each stage implements a CNN layer as shown in Figure 13. The implemented CNN model has a fixed network structure, but the weights can be configured at run time via a scan chain. The implementation target is a 28 nm CMOS technology that uses a customized design flow. In the 2D case, only six metal layers are used. In the 3D case, face-to-face bonding is used, as illustrated in Figure 14.

An in-house model generator is implemented in Python, and it flattens a pretrained CNN model and generates high-level synthesis (HLS) code for each layer. The generated HLS code is then synthesized with the HLS tool to generate the RTL for the ASIC design. Commercial tools from Synopsys and Cadence are used for logic synthesis and

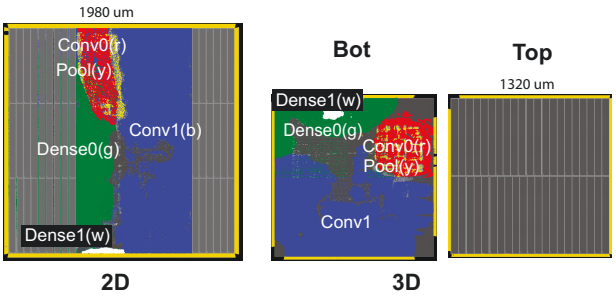


Figure 15. Placement comparison between 2D and 3D designs. SRAM arrays are placed on the sides in 2D and on the top tier in the 3D case. The modules are colored as conv0 (red), pool (yellow), conv1 (blue), dense0 (green), and dense1 (white).

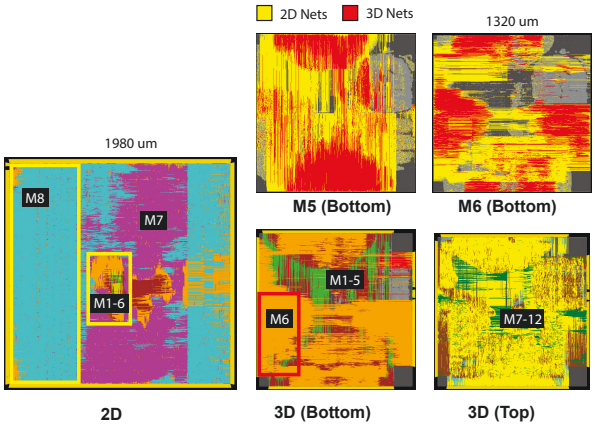


Figure 16. Top-level shape comparison between 2D and 3D designs. The 3D layout includes two metal layers of the bottom tier to illustrate 3D nets (in red), when compared with 2D nets (in yellow).

physical design. To explore the benefits between 2D and 3D design flows, we use the pseudo-3D partitioning approach (Park et al. 2020). The partitioning step uses the same RTL, which generated the tier location. For 2D design, this step is simply skipped. Figure 15 shows the placement results of the 2D and 3D design flows. Notably, for the 2D flow, the SRAM components are placed on the sides of the logic. For the 3D design, the SRAM arrays are placed on the top tier. The comparison of the routine is shown in Figure 16.

The performance comparison between 2D and 3D cases are tabulated in Table 2. In addition to simple metrics such as wire length, leakage, and switching power, we also list comprehensive metrics such as PDP (power delay product) and EDP (energy delay product). The comprehensive results show that the 3D design approach achieved 30%–40% improvements over the 2D implementation.

Devices

Neural network inference and training based on analog in-memory computing (AIMC) can achieve energy efficiency (i.e., performance per watt) 2 orders of magnitude beyond what is possible with digital CMOS (Xiao et al. 2022b). This is made possible by implementing common matrix operations (e.g., vector matrix multiply [VMM]) with a technique that is more natively matched to this operation than traditional von Neumann approaches (see Figure 17). In this

Table 2. Full-chip power, performance, and area comparison. The percentage values in the last column indicate the improvements with the 3D design.

Design	Full-chip design		
	2D	*3D*	*Imp. 3D*
Effective freq. (MHz)	742	881	18.7%
Footprint (mm^2)	3.9	1.7	56.4%
No. of cells	892K	806K	10.0%
Wire length (m)	64.7	40.6	37.2%
Total power (mW)	1,954	1,549	20.7%
→ Internal power (mW)	995.9	826	17.1%
→ Switching power (mW)	957.5	688	28.1%
→ Leakage power (mW)	40.7	34	16.4%
PDP	2,633	1,758	33.2%
EDP	3,549	1,996	43.8%

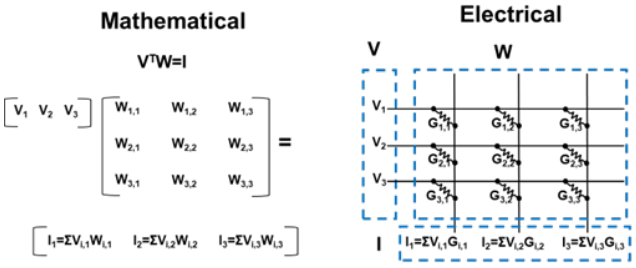


Figure 17. (a) Mathematical and (b) electrical vector matrix multiplication (Marinella et al. 2022).

technique, the neural network weights are stored in an analog array of tunable resistors. The VMM operation requires a multiply, which is accomplished using ohm's law, and a sum over each column, which is provided by Kirchoff's current law, as illustrated in Figure 17b. A complete VMM operation can be accomplished in a single parallel step, thereby avoiding the complex data shuttling between registers and execution units that is required by von Neuman systems.

One of the key challenges for AIMC is that accuracy is typically degraded compared to a digital processor because analog tunable resistors in Figure 17b must be programmed to a conductance that represents a corresponding 8-bit weight in the neural network. However, it is typically not possible to program a large array of memory devices precisely to 256 discrete levels. Instead, devices are programmed to conductance values that represent the weight as closely as possible but include an error tolerance that is a function of device attributes (e.g., stochasticity and short-term drift). This programming error depends on the specific device being use for neural network weight storage (sometimes referred to as a synaptic device). Despite this programming error, it is possible to obtain high inference accuracy even on modern deep CNNs that use AIMC circuits thanks to the averaging over many values in each column. The accuracy possible for a particular CNN and dataset (e.g., ImageNet running on ResNet-50) depends on the details of the device being used to represent the weight and the error at a given conductance (Xiao et al. 2022b, 2021b).

Resistive random access memory (ReRAM) (Marinella et al. 2018), phase-change memory (PCM) (Burr et al. 2010; Raoux et al. 2008), and SONOS (Xiao et al. 2022b) are the most mature in-production nonvolatile memory devices being considered as the tunable resistor in Figure 17b. The programming error of each of these devices is typically

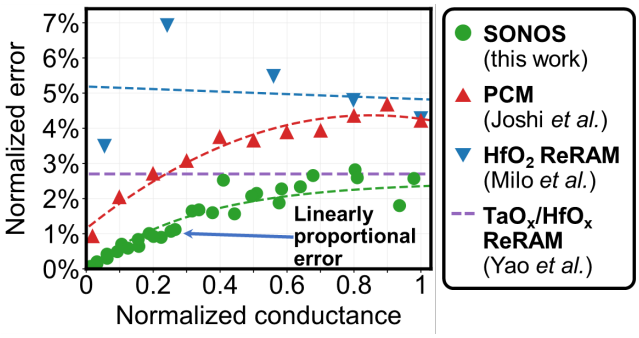


Figure 18. Normalized error vs. conductance for SONOS, PCM, HfO_2, and TaO_x/HfO_x ReRAM.

characterized as a function of conductance, which is plotted in Figure 18 for ReRAM, SONOS, and PCM. The SONOS (green curve) has the lowest error at low conductance, and this is ideal behavior and provides the highest CNN inference accuracy (Xiao et al. 2022b, 2021b). A drawback of SONOS is the slow drift of the state following initial programming, but this can be significantly accelerated under modest ionizing radiation (Xiao et al. 2021a). Metal-oxide based ReRAM is generally robust to ionizing radiation (Marinella 2021) but has a higher programming error, especially at low conductances (blue and violet points in Figure 18). Notably, this type of programming error benchmarking is in early stages and does not yet represent the full potential of ReRAM. As part of the Abisko project, we are investigating new write-verify and filament-forming techniques that are expected to reduce the programming error and enable radiation-hard AIMC.

SONOS stores analog states as a charge trapped in the nitride layer between the channel and gate of a MOSFET, as shown in Figure 19b. SONOS is particularly attractive due to its technological maturity and CMOS compatibility. Motivated by these factors, high accuracy inference on 40 nm SONOS arrays was recently demonstrated at 8 bits (Agarwal et al. 2019). However, exposure of these arrays to a total ionizing dose (TID) of 10–100 krad(Si) can degrade the accuracy of ANNs from >90% to <10% due to shifts in the SONOS MOSFET threshold voltage and therefore the channel conductance, as shown in Figure 19c (Xiao et al. 2022a). The reason for state loss and accuracy degradation stems from TID being very effective at generating charge in the trap layer, particularly holes, and this forces the MOSFET threshold voltage in the program 0 state to migrate toward lower values.

The sensitivity of SONOS to relatively low levels of radiation render it and other charge-trapping memory (CTM) variants (e.g., flash) as unlikely candidates for in-sensor data processing for high-radiation environments. Nevertheless, developing and experimentally validating a compact model for SONOS represents a valuable knowledge-generating activity for modeling other three-terminal analog nonvolatile memories such as ECRAM. To this end, we have recently developed a well-poised physics-based compact model for a SONOS analog synaptic element in collaboration with Yi et al. (2022). The model contains a nonvolatile memristor with the state variable Q_M to represent the memristor charge under the gate of the three-terminal element. By incorporating the exponential dependence of the memristance on Q_M and the applied bias V for the gate, the compact model agrees quantitatively with the results from Technology CAD simulations as well as experimental measurements for the drain current (Figure 19d). This compact model can now be used for design of analog ANNs based on CTMs.

In addition to ReRAM, PCRAM, and SONOS, other device types have been investigated as potential solutions to enable energy-efficient analog ANNs, including electrochemical metallization cells (Valov et al. 2011; Liu et al. 2013; Barbera et al. 2015) and ferroelectric memristive devices (Chanthbouala et al. 2012; Yang et al. 2013a). Performance limitations that currently prevent these technologies from challenging digital solutions include excessive write noise (Yang et al. 2013a; Terai et al. 2010; Close et al. 2010), write nonlinearities (Strukov and Williams 2009; Menzel et al. 2011; Athmanathan et al. 2016; Chen et al. 2015b; Burr et al. 2015), high switching voltages and currents (Wong et al. 2010; Kim et al. 2010; Chen et al. 2015a), and, most problematically, high read currents (Marinella et al. 2018). A promising avenue to address these challenges is to add a third electrode, the *gate*, to decouple the read and write operations. Two prominent examples of three-terminal nonvolatile analog memory devices are SONOS CTM (Agarwal et al. 2019) and ECRAM (Talin et al. 2022). ECRAM was pioneered by Talin and colleagues at Sandia National Laboratories with the demonstration of the Li-ion synaptic transistor in 2017 and has emerged as another promising platform for neuromorphic computing (Talin et al. 2022). In ECRAM, a gate is used to drive defects (e.g., Li ions, protons, or oxygen vacancies) from a reservoir layer in and out of a channel to control its electronic properties (Figure 20a). An electronically insulating solid-state electrolyte that allows only the motion of the mobile defects is what separates the channel and the reservoir. Just like SONOS, ECRAM decouples the read and write operations to enable linear, symmetric switching with low read and write currents. A principle advantage of ECRAM compared to a CTM is the much higher charge density available for tuning analog states. Our recent analysis of SONOS reveals a charge trap density of $\approx 10^{19}/cm^3$, or approximately 1 out of every 1,000 atoms[2]. For ECRAM based on transition metal oxide channels (e.g., Li_xCoO_2, TiO_x, or WO_{3-x}), every transition metal in the lattice, or $\approx 10^{22}/cm^3$, is capable of changing its redox state and is thus available for tuning and storing analog states (Talin et al. 2022). Long-term retention, however, remains a major challenge for ECRAM. We define true retention as measured at 85°C with the gate grounded (or shorted to the source). Previously, we demonstrated a retention of ≈3 hours for TiO_x ECRAM that utilizes YSZ solid electrolyte (Li et al. 2020b). More recently, we demonstrated in collaboration with Y. Li's group at the University of Michigan that a retention of ≈10 years is feasible for WO_{3-x}/YSZ ECRAM, as shown in Figure 20 (Kim et al. 2022).

Materials

Neuromorphic algorithms can be implemented by a combination of nonlinear conductance with time-retention (memory) that can be further programmed with voltage

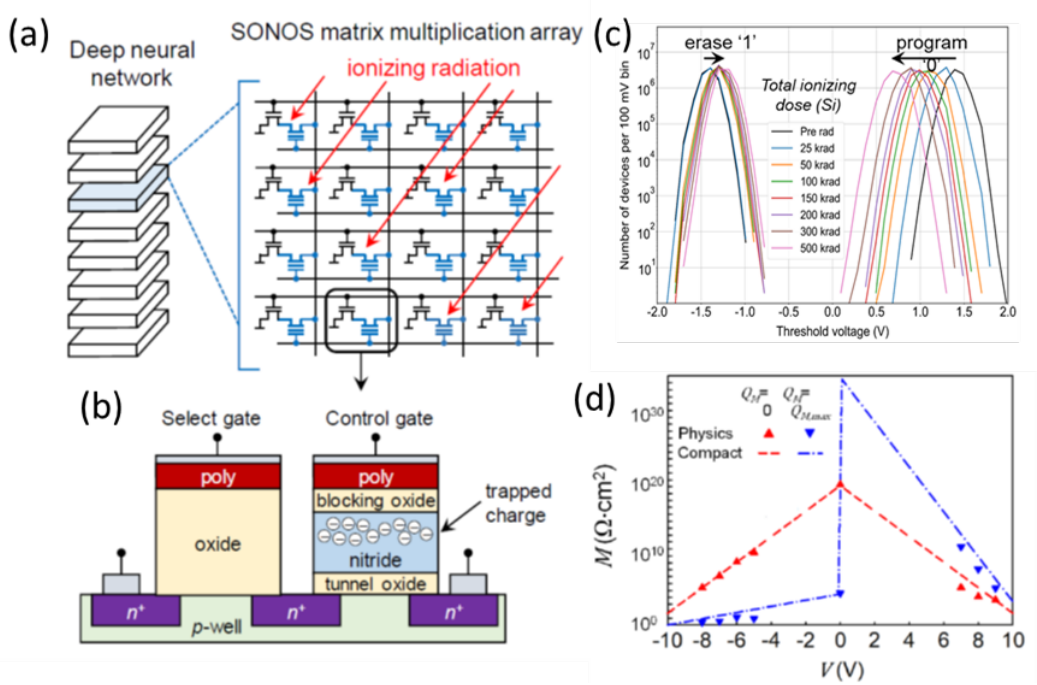


Figure 19. SONOS Device. (a) A SONOS memory array can store a matrix of weights for one layer of a DNN. (b) Memory cell consisting of a SONOS transistor and an access transistor. When exposed to ionizing radiation (red arrows), the state of the SONOS cells may be perturbed. (c) Measured VT distribution of the SONOS test chips at varying levels of TID. Reproduced with permission (Xiao et al. 2021a). Copyright 2021, IEEE. (d) Memristance, M, as a function of voltage. Upward and downward triangles represent the memristance values at two extremes (see ref. for details). Adapted with permission (Yi et al. 2022). Copyright 2017, Wiley.

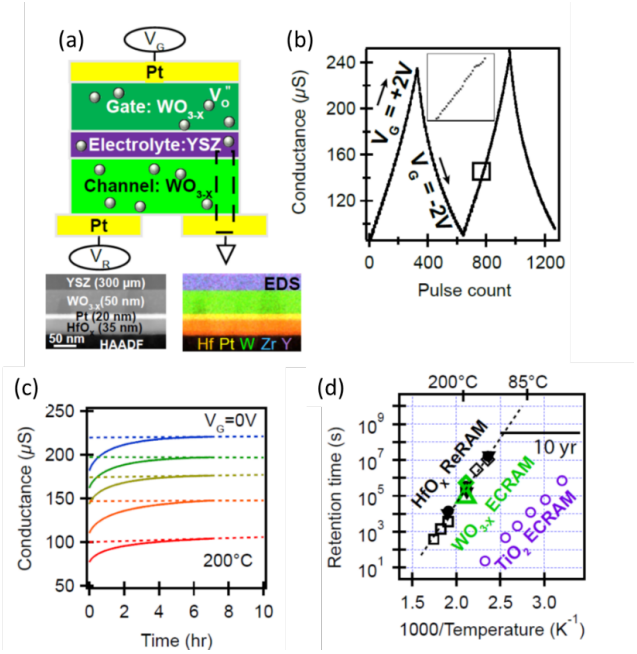


Figure 20. ECRAM device. (a) Schematic and cross-section TEM of WO_{3-x} ECRAM cell. (b) Analog switching characteristics that demonstrate high state density. (c) ECRAM retention characteristics when the gate and channel are shorted at 200°C. (d) Comparison of retention times of WO_{3-x} ECRAM with filament-based ReRAM and past TiO_x-based ECRAM. (Kim et al. 2022). Copyright 2022, Wiley.

waveforms (Di Ventra and Traversa 2018; Yang et al. 2013b). By contrast, semiconductors in ideal circumstances exhibit no memory effects. Although a fairly large number of devices that exhibit both nonlinearity and memory have been demonstrated, the descriptive and predictive understanding of the enabling phenomena is not on the same mature level of semiconductor physics (Nili et al. 2020b; Brown et al. 2022). Given that codesign of modern semiconductor circuitry presently occurs across all contributing length scales (i.e., from atoms to architectures (Stettler et al. 2021)), the argument in favor of fundamental understanding of material mechanism that can best implement specific neuromorphic functions is very compelling. However, the intrinsic challenge is that both nonlinear and memory functions imply that the desired properties originate from non-equilibrium behavior (generally transient, metastable, and temporal), so that many trusted methodologies for materials design will, at best, have limited applicability to the neuromorphic paradigm. Some of the specific challenges that arise from incomplete understanding of nonlinear device mechanisms include the limited level of reproducibility and predictability over resistive switching devices, the challenge of long-term retention of their programmed characteristics, and even well-defined, physics-based compact models that abstract the device properties for further integration with computing architectures (Gao et al. 2021; Dao and Koch 2020; Hamdioui et al. 2017; Brown et al. 2022) .

In the Abisko project, we are pursuing a multimodal approach to observe nanoscale processes in candidate neuromorphic materials and connect the observation to device performance and eventually device models. The first task is to develop a combination of quantifiable nanoscale characterization techniques that can directly reveal the connection between conductance and material structure with nanoscale spatial resolution. The most promising

emerging experimental techniques for this task are scanning microwave impedance microscopy, cathodoluminescence microscopy, and spatially resolved time-of-flight secondary ion mass spectroscopy. Notably, these techniques have not been extensively applied to neuromorphic materials, particularly in combination. However, their advantages stem from the ability to gain complementary signals with high spatial resolution and the ability to quantify the observed signals.

Microwave microscopy directly measures local conductance with nanoscale resolution while minimizing the effects of poor contact resistance (Barber et al. 2022; Chu et al. 2020). Some teams have successfully applied this methodology for quantitative measurements of local domain wall conductivity in ferroelectric materials (Burns et al. 2022; Tselev et al. 2016). From the functional point of view, the locally conductive domain walls in ferroelectrics and filamentary structures in resistive switching materials are reminiscent of each other, including their characteristic nanoscale dimensions, two-terminal nonlinear conductance, and memory effects. Meanwhile, cathodoluminescence enables direct characterization of electronic structure, defects, and trap states (Coenen and Haegel 2017). Investingating the electronic structure of most practically relevant materials, including amorphous solids, dirty semiconductors, and even metal-insulator systems, has been a challenging goal for a very long time, and any new insights will be highly relevant for neuromorphic applications. Finally, another technique called Time-of-Flight Secondary Ion Mass Spectrometry (TOF-SIMS) complements the above measurements with direct chemical sensitivity to ionic diffusion and elemental modification (Belianinov et al. 2018).

As candidate model systems, the Abisko project is pursuing resistive switching in TaO_x, electrochemical switching in VO_x devices, and also the search for new materials, which could offer distinct advantages over binary oxides in terms of energy efficiency and architectural design for accelerated feedback between materials and device levels. For example, molecular-based materials and ferroelectric semiconductors could avoid costly and stochastic filamentary mechanisms while enabling multilevel, temporal, and scalable switching properties.

A crucial ingredient that will enable connecting microscopy observations to electronic properties relevant for neuromorphic properties is the long-sought ability to control electronic effects uniformly across active volume of the material. The filamentary conductors ubiquitous in memristor devices are notoriously difficult to observe directly because stochastic and highly energetic processes behind electroforming are fundamentally localized due to the high local energy density required for chemical transformation into an electronically conducting state. To this end, we are pursuing the methods of ionic modification of materials, whether by high ion irradiation or low-energy ion intercalation, in an attempt to create electroforming-free control over material conductance and therefore enable both effective characterization and prospectively on-demand control over electroresistive properties.

On the device level, successful control of resistive switching by ion irradiation has been demonstrated (Vogel et al. 2022), including by several members of the Abisko team (Marinella et al. 2012; Jacobs-Gedrim et al. 2019). Meanwhile, the fundamental mechanism of ionic transport across interfaces is pursued within Abisko as the acting principle behind ECRAM devices (Talin et al.), with an implicit assumption of largely uniform transformations within the active volume of the material. Electronic characterization of ionically modified materials forms one of the central codesign links between the device and materials teams. Figure 21 demonstrates preliminary results of cathodoluminescence and microwave imaging of the He-irradiated amorphous TaO_x films. Figure 21(a) shows the CL spectrum with discrete signatures of oxygen vacancy states in the amorphous lattice (red arrows). The energies of these defect states can be mapped onto the structures using state-of-the-art first-principles calculations of large amorphous unit-cells. Inset shows the schematic of the tantalum oxide film, involving the TaOx active layer, Ta and TiN metallic contact layers, all grown atop silicon wafer. Figure 21(b) shows the spectral decomposition of the hyperspectral array of cathodoluminescence spectra carried

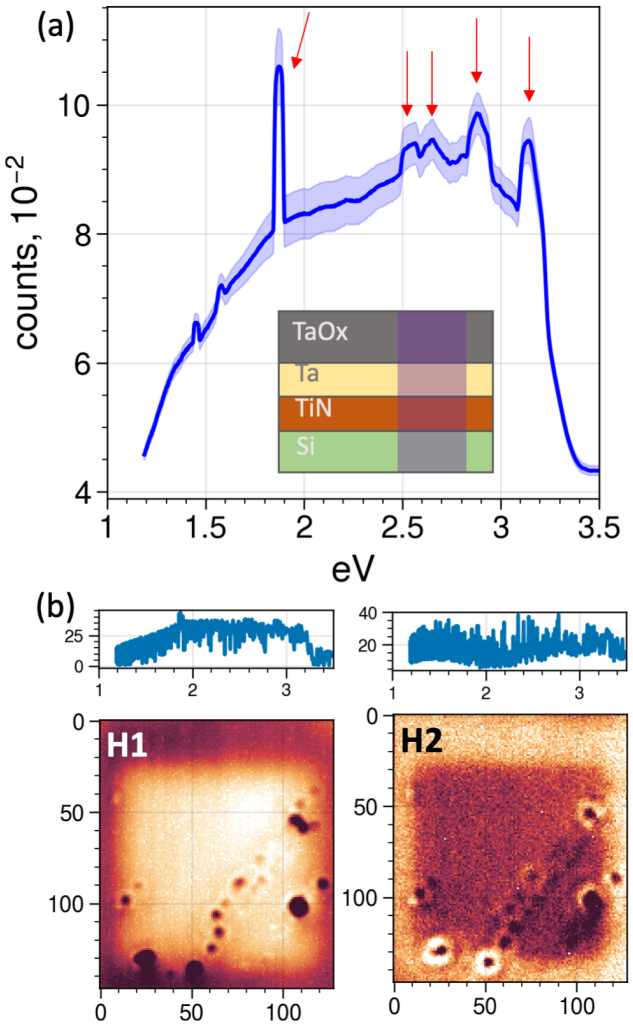


Figure 21. Cathodoluminescence (CL) probe of amorphous tantalum oxide, and the effects of He-ion irradiation measured at the Center for Nanophase Materials Sciences utilizing the films grown at Sandia National Laboratory. (a) CL spectrum showing discrete signatures of oxygen vacancy states in the amorphous lattice (red arrows). (b) Spectral decomposition of the hyperspectral array of cathodoluminescence spectra.

out with non-negative matrix factorization (NMF). The maps shows the spatial intensity of each matrix factor component (H1 and H2), while the plots at the top show the components themselves. NMF analysis very clearly reveals the square region of the He-irradiated film, as well as other defect sites in the film. Note that the modification due to He-ion is remarkably subtle. The components only contribute to about 2-5% of the luminescence intensity (see also the error bars in (a)), and the matrix factors themselves are quite noisy. Yet the patterns and defects are very clear due to effectiveness of NMF (and similar) techniques, enabling structural modification and concomitant electronic analysis of neuromorphic materials. The eventual goal of these studies is to identify reproducible paths to electroforming-free activation and subsequent control of film conductance using ion-beam irradiation.

Both techniques successfully reveal the irradiated areas through both electronic and capacitive signatures. Remarkably, the films maintain a high local resistivity even at comparatively large He-irradiation doses, likely owing to their amorphous structures. Meanwhile, by using TOF-SIMS, we unambiguously detected the signatures of ionic redistribution within TaO_x under specific action of local electric fields applied from an atomic force microscopy tip. The measurements of local resistive switching characteristics of TaO_x as a function of ion irradiation and localized field-induced electrochemical modification are currently under way.

The second goal of the materials thrust is to abstract the experimental observations from spatially resolving microscopy techniques into a compact representation. Fulfilling this task will enable effective multimodal analysis of the experimental observations. Even more intriguing is the prospect of deriving data-driven low-dimensional device and compact models (Messaris et al. 2018; Nili et al. 2020a) by using heterogeneous nanoscale measurements. Indeed, microscopy techniques are intrinsically data rich. The aforementioned analysis of He-ion irradiation demonstrates how well-established methods of statistical data analysis (e.g., non-negative matrix factorization) are sensitive to even the slightest signatures of He-ion irradiation. Such methods are widely applicable to hyperspectral datasets and will enable us to derive statistically meaningful trends from all the spatially resolving techniques, including chemical imaging and electronic and conductive measurements. Data-driven modeling may bridge the critical gap between material and device and, in turn, enable an effective codesign strategy, even in the absence of a detailed physical understanding of the mechanisms behind field-induced properties of neuromorphic devices.

Codesign

As described in Abisko Overview, *codesign* is an overarching goal of Abisko. Our motivating example will be the abstractions and interfaces presented in Table 1. For true codesign, we need a vertically integrated codesign framework that helps to systematize relationships among layers beyond informal correspondence and interaction. More specifically, this codesign activity cross-cuts the computing stack (Table 1) and we are working to capture the overall design by extending existing modeling tools (e.g., ASPEN/FLAME) to specify interfaces for each layer as well their interfaces across this stack. This representation will facilitate automatic design space exploration. Moreover, neuromorphic computing provides an excellent vehicle to study our deep codesign framework because its computational paradigm can be represented in low-level devices and materials that can, if properly designed, offer tremendous benefits in terms of energy efficiency and performance to applications at the highest levels of abstraction.

In Abisko, we approach deep codesign in three ways. First, we are identifying concrete interfaces across the conceptual compute layers to capture idealized forms of information exchange. Examples of these interfaces include using the TCAD compact modeling language for modeling material and device characteristics (e.g., Stettler et al. (2021)), which can, in turn, be used to define new architectures, and using the MLIR programming infrastructure to describe primary concepts for SNNs as first class objects in the programming language.

Second, we are extending our Aspen/FLAME domain-specific language to capture hierarchical details of system design. To address many of these concerns, ORNL created the domain-specific languages called Aspen (Spafford and Vetter 2012; Spafford et al. 2013; Spafford and Vetter 2015; Umar et al. 2016, 2018; Peng and Vetter 2018; Lee et al. 2015) and FLAME (Belviranli and Vetter 2019) to model applications and architectures. Aspen allows users to create formal written descriptions of the applications and the architectures. Prior Aspen work focused on system architecture, including CPUs, GPUs, networks, and memory systems.

Third, we are investigating formalisms built on top of Aspen and FLAME that can automatically optimize and refine abstractions across the stack with AI techniques to expedite the exploration of these massive design spaces. Our initial activity will use DEFFE's transfer learning (see DEFFE (Data-Efficient Exploration Framework)) capability to reduce the computation cost of these explorations.

Summary

This paper describes an approach to deep codesign of an architecture for spiking neural networks using novel neuromorphic materials. The *deep codesign* approach engages experts from across the entire range of disciplines: materials, devices and circuits, architectures and integration, software, and algorithms. The novel neuromorphic devices are based on resistive-switching materials, such as memristors and electrochemical RAM (ECRAM). Abisko has three key objectives. First, we are designing an energy-optimized high-performance neuromorphic accelerator based on SNNs. This architecture is being designed as a chiplet that can be deployed in contemporary computer architectures and we are investigating novel neuromorphic materials to improve its performance and energy-efficiency. Second, we are concurrently developing a productive software stack for the neuromorphic accelerator that will also be portable to other architectures, such as field-programmable gate arrays and

GPUs. Third, we are creating a new deep codesign methodology and framework for developing clear interfaces, requirements, and metrics between each level of abstraction to enable the system design to be explored and implemented interchangeably with execution, measurement, a model, or simulation. As a motivating application for this codesign effort, we target the use of SNNs for an analog event detector for a high-energy physics sensor.

References

(2023) Mlir tensor dialect. URL https://mlir.llvm.org/docs/Dialects/TensorOps/.

Aad G et al. (2012) Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC. *Phys. Lett. B* 716: 1. DOI:10.1016/j.physletb.2012.08.020.

Abi Akar N, Cumming B, Karakasis V, Küsters A, Klijn W, Peyser A and Yates S (2019) Arbor — A Morphologically-Detailed Neural Network Simulation Library for Contemporary High-Performance Computing Architectures. In: *2019 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*. pp. 274–282. DOI:10.1109/EMPDP.2019.8671560.

Agarwal S, Garland D, Niroula J, Jacobs-Gedrim RB, Hsia A, Heukelom MSV, Fuller E, Draper B and Marinella MJ (2019) Using floating-gate memory to train ideal accuracy neural networks. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits* 5(1): 52–57. DOI:10.1109/JXCDC.2019.2902409.

Aimone J, Date P, Fonseca-Guerra G, Hamilton K, Henke K, Kay B, Kenyon G, Kulkarni S, Mniszewski S, Parsa M et al. (2022) A review of non-cognitive applications for neuromorphic computing. *Neuromorphic Computing and Engineering* .

Aimone JB, Ho Y, Parekh O, Phillips CA, Pinar A, Severa W and Wang Y (2020) Provable neuromorphic advantages for computing shortest paths. In: *Proceedings of the 32nd ACM Symposium on Parallelism in Algorithms and Architectures*. pp. 497–499.

Akopyan F, Sawada J, Cassidy A, Alvarez-Icaza R, Arthur J, Merolla P, Imam N, Nakamura Y, Datta P, Nam G, Taba B, Beakes M, Brezzo B, Kuang JB, Manohar R, Risk WP, Jackson B and Modha DS (2015) Truenorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34(10): 1537–1557. DOI:10.1109/TCAD.2015.2474396.

Ang J, Chien AA, Hammond SD, Hoisie A, Karlin I, Pakin S, Shalf J and Vetter J (2021) Reimagining codesign for advanced scientific computing: Unlocking transformational opportunities for future computing systems for science. Technical report. DOI:10.2172/1822198.

Ardalan S, Cirit H, Farjad R, Kuemerle M, Poulton K, Subramanian S and Vinnakota B (2020) Bunch of wires: An open die-to-die interface. In: *2020 IEEE Symposium on High-Performance Interconnects (HOTI)*. pp. 9–16. DOI:10.1109/HOTI51249.2020.00017.

Athmanathan A, Stanisavljevic M, Papandreou N, Pozidis H and Eleftheriou E (2016) Multilevel-cell phase-change memory: A viable technology. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 6(1): 87–100. DOI:10.1109/JETCAS.2016.2528598.

Barber ME, Ma EY and Shen ZX (2022) Microwave impedance microscopy and its application to quantum materials. *NATURE REVIEWS PHYSICS* 4(1): 61–74. DOI:10.1038/s42254-021-00386-3.

Barbera SL, Vuillaume D and Alibart F (2015) Filamentary switching: Synaptic plasticity through device volatility. *ACS Nano* 9(1): 941–949. DOI:10.1021/nn506735m.

Belianinov A, Ievlev AV, Lorenz M, Borodinov N, Doughty B, Kalinin SV, Fernandez FM and Ovchinnikova OS (2018) Correlated materials characterization via multimodal chemical and functional imaging. *ACS NANO* 12(12): 11798–11818. DOI:10.1021/acsnano.8b07292.

Belviranli ME and Vetter JS (2019) FLAME: Graph-based hardware representations for rapid and precise performance modeling. In: *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. pp. 1775–1780. DOI:10.23919/DATE.2019.8747521.

Benjamin BV, Gao P, McQuinn E, Choudhary S, Chandrasekaran AR, Bussat JM, Alvarez-Icaza R, Arthur JV, Merolla PA and Boahen K (2014) Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations. *Proceedings of the IEEE* 102(5): 699–716.

Bohte SM, Kok JN and La Poutré JA (2000) Spikeprop: backpropagation for networks of spiking neurons. In: *ESANN*, volume 48. Bruges, pp. 419–424.

Brown TD, Kumar S and Williams RS (2022) Physics-based compact modeling of electro-thermal memristors: Negative differential resistance, local activity, and non-local dynamical bifurcations. *APPLIED PHYSICS REVIEWS* 9(1). DOI:10.1063/5.0070558.

Burns SR, Tselev A, Ievlev AV, Agar JC, Martin LW, Kalinin SV, Sando D and Maksymovych P (2022) Tunable microwave conductance of nanodomains in ferroelectric pbzr0.2ti0.8o3 thin film. *ADVANCED ELECTRONIC MATERIALS* 8(3). DOI:10.1002/aelm.202100952.

Burr GW, Breitwisch MJ, Franceschini M, Garetto D, Gopalakrishnan K, Jackson B, Kurdi B, Lam C, Lastras LA, Padilla A, Rajendran B, Raoux S and Shenoy RS (2010) Phase change memory technology. *Journal of Vacuum Science & Technology B* 28(2): 223–262. DOI:10.1116/1.3301579.

Burr GW, Shelby RM, Sidler S, Nolfo C, Jang J, Boybat I, Shenoy RS, Narayanan P, Virwani K, Giacometti EU, Kurdi BN and Hwang H (2015) Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element. *IEEE Transactions on Electron Devices* 62(11): 3498–3507. DOI:10.1109/TED.2015.2439635.

Calligaris L, collaboration C et al. (2020) Status of the phase-2 tracker upgrade of the cms experiment at the hl-lhc. In: *Journal of Physics: Conference Series*, volume 1690. IOP Publishing, p. 012039.

Caporale N, Dan Y et al. (2008) Spike timing-dependent plasticity: a hebbian learning rule. *Annual review of neuroscience* 31(1): 25–46.

Chanthbouala A, Garcia V, Cherifi RO, Bouzehouane K, Fusil S, Moya X, Xavier S, Yamada H, Deranlot C, Mathur ND, Bibes M, Barthélémy A and Grollier J (2012) A ferroelectric memristor. *Nature Materials* 11(10): 860–864. DOI:10.1038/

nmat3415.

Chatrchyan S et al. (2012) Observation of a New Boson at a Mass of 125 GeV with the CMS Experiment at the LHC. *Phys. Lett. B* 716: 30. DOI:10.1016/j.physletb.2012.08.021.

Chen PY, Kadetotad D, Xu Z, Mohanty A, Lin B, Ye J, Vrudhula S, Seo J, Cao Y and Yu S (2015a) Technology-design co-optimization of resistive cross-point array for accelerating learning algorithms on chip. In: *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. pp. 854–859.

Chen PY, Lin B, Wang IT, Hou TH, Ye J, Vrudhula S, Seo J, Cao Y and Yu S (2015b) Mitigating effects of non-ideal synaptic device characteristics for on-chip learning. In: *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. pp. 194–199. DOI:10.1109/ICCAD.2015.7372570.

Chien SWD, Markidis S, Olshevsky V, Bulatov Y, Laure E and Vetter JS (2019) Tensorflow doing HPC. *CoRR* abs/1903.04364.

Chu Z, Zheng L and Lai K (2020) Microwave microscopy and its applications. In: Clarke D (ed.) *ANNUAL REVIEW OF MATERIALS RESEARCH, VOL 50, 2020*, *Annual Review of Materials Research*, volume 50. ISBN 978-0-8243-1750-8, pp. 105–130. DOI:10.1146/annurev-matsci-081519-011844.

Close GF, Frey U, Breitwisch M, Lung HL, Lam C, Hagleitner C and Eleftheriou E (2010) Device, circuit and system-level analysis of noise in multi-bit phase-change memory. In: *2010 International Electron Devices Meeting*. pp. 29.5.1–29.5.4. DOI:10.1109/IEDM.2010.5703445.

Coenen T and Haegel NM (2017) Cathodoluminescence for the 21st century: Learning more from light. *APPLIED PHYSICS REVIEWS* 4(3). DOI:10.1063/1.4985767.

Cong G, Lim SH, Kulkarni S, Date P, Potok T, Snyder S, Parsa M and Schuman C (2022) Semi-supervised graph structure learning on neuromorphic computers. In: *Proceedings of the International Conference on Neuromorphic Systems 2022*. pp. 1–4.

Dally WJ, Turakhia Y and Han S (2020) Domain-specific hardware accelerators. *Commun. ACM* 63(7): 48–57. DOI:10.1145/3361682.

Dao NC and Koch D (2020) Memristor-based reconfigurable circuits: Challenges in implementation. In: *2020 International Conference on Electronics, Information, and Communication (ICEIC)*. pp. 1–6. DOI:10.1109/ICEIC49074.2020.9051174.

Date P, Carothers CD, Hendler JA and Magdon-Ismail M (2018) Efficient classification of supercomputer failures using neuromorphic computing. In: *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, pp. 242–249.

Date P, Kay B, Schuman C, Patton R and Potok T (2021) Computational complexity of neuromorphic algorithms. In: *International Conference on Neuromorphic Systems 2021*. pp. 1–7.

Date P, Kulkarni S, Young A, Schuman C, Potok T and Vetter J (2022a) Encoding integers and rationals on neuromorphic computers using virtual neuron. *arXiv preprint arXiv:2208.07468* .

Date P, Potok T, Schuman C and Kay B (2022b) Neuromorphic computing is turing-complete. In: *Proceedings of the International Conference on Neuromorphic Systems 2022*. pp. 1–10.

Davies M, Srinivasa N, Lin T, Chinya G, Cao Y, Choday SH, Dimou G, Joshi P, Imam N, Jain S, Liao Y, Lin C, Lines A, Liu R, Mathaikutty D, McCoy S, Paul A, Tse J, Venkataramanan G, Weng Y, Wild A, Yang Y and Wang H (2018) Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro* 38(1): 82–99. DOI:10.1109/MM.2018.112130359.

Di Ventra M and Traversa FL (2018) Perspective: Memcomputing: Leveraging memory and physics to compute efficiently. *Journal of Applied Physics* 123(18): 180901. DOI:10.1063/1.5026506. URL https://doi.org/10.1063/1.5026506.

Ferrari S, Mehta B, Di Muro G, VanDongen AM and Henriquez C (2008) Biologically realizable reward-modulated hebbian training for spiking neural networks. In: *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*. IEEE, pp. 1780–1786.

Fleming B et al. (2019) Basic research needs for high energy physics detector research & development. URL https://science.osti.gov/hep/Community-Resources/Reports.

Gao L, Ren Q, Sun J, Han ST and Zhou Y (2021) Memristor modeling: challenges in theories, simulations, and device variability. *JOURNAL OF MATERIALS CHEMISTRY C* 9(47): 16859–16884. DOI:10.1039/d1tc04201g.

Gewaltig MO and Diesmann M (2007) Nest (neural simulation tool). *Scholarpedia* 2(4): 1430.

Gonzalez-Tallada M, Valero-Lara P, Denny J and Veter J (2022) ecc++ : An embedded compiler construction frameworkfor domain-specific languages [manuscript submitted for publication] .

Hamdioui S, Kvatinsky S, Cauwenberghs G, Xie L, Wald N, Joshi S, Elsayed HM, Corporaal H and Bertels K (2017) Memristor for computing: Myth or reality? In: *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. pp. 722–731. DOI:10.23919/DATE.2017.7927083.

Hamilton K, Date P, Kay B and Schuman D C (2020a) Modeling epidemic spread with spike-based models. In: *International Conference on Neuromorphic Systems 2020*. pp. 1–5.

Hamilton K, Mintz T, Date P and Schuman CD (2020b) Spike-based graph centrality measures. In: *International Conference on Neuromorphic Systems 2020*. pp. 1–8.

Hennessy JL and Patterson DA (2019) A new golden age for computer architecture. *Commun. ACM* 62(2): 48–60. DOI:10.1145/3282307.

Jacobs-Gedrim RB, Hughart DR, Agarwal S, Vizkelethy G, Bielejec ES, Vaandrager BL, Swanson SE, Knisely KE, Taggart JL, Barnaby HJ and Marinella MJ (2019) Training a neural network on analog taox reram devices irradiated with heavy ions: Effects on classification accuracy demonstrated with crosssim. *IEEE TRANSACTIONS ON NUCLEAR SCIENCE* 66(1, 1, SI): 54–60. DOI:10.1109/TNS.2018.2886229. 55th IEEE Nuclear and Space Radiation Effects Conference (NSREC), Kona, HI, JUL 16-20, 2018.

Jin T, Bercea GT, Le TD, Chen T, Su G, Imai H, Negishi Y, Leu A, O'Brien K, Kawachiya K and Eichenberger AE (2020) Compiling onnx neural network models using mlir.

Kay B, Date P and Schuman C (2020) Neuromorphic graph algorithms: Extracting longest shortest paths and minimum

spanning trees. In: *Proceedings of the Neuro-inspired Computational Elements Workshop*. pp. 1–6.

Kay B, Schuman C, O'Connor J, Date P and Potok T (2021) Neuromorphic graph algorithms: Cycle detection, odd cycle detection, and max flow. In: *International Conference on Neuromorphic Systems 2021*. pp. 1–7.

Kehlet D et al. (2017) Accelerating innovation through a standard chiplet interface: The advanced interface bus (aib) .

Kim DS, Watkins VJ, Cline LA, Li J, Sun K, Sugar JD, Fuller EJ, Talin AA and Li Y (2022) Nonvolatile electrochemical random-access memory under short circuit. *Advanced Electronic Materials* n/a(n/a): 2200958. DOI:https://doi.org/10.1002/aelm.202200958.

Kim MJ, Baek IG, Ha YH, Baik SJ, Kim JH, Seong DJ, Kim SJ, Kwon YH, Lim CR, Park HK, Gilmer D, Kirsch P, Jammy R, Shin YG, Choi S and Chung C (2010) Low power operating bipolar tmo reram for sub 10 nm era. In: *2010 International Electron Devices Meeting*. pp. 19.3.1–19.3.4. DOI:10.1109/IEDM.2010.5703391.

Kösters DJ, Kortman BA, Boybat I, Ferro E, Dolas S, de Austri R, Kwisthout J, Hilgenkamp H, Rasing T, Riel H et al. (2022) Benchmarking energy consumption and latency for neuromorphic computing in condensed matter and particle physics. *arXiv preprint arXiv:2209.10481* .

Kozloski JR and Wagner J (2011) An ultrascalable solution to large-scale neural tissue simulation. *Frontiers Neuroinformatics* 5: 15. DOI:10.3389/fninf.2011.00015. URL https://doi.org/10.3389/fninf.2011.00015.

Krishnan G, Mandal SK, Pannala M, Chakrabarti C, Seo JS, Ogras UY and Cao Y (2021) Siam: Chiplet-based scalable in-memory acceleration with mesh for deep neural networks 20(5s). DOI:10.1145/3476999. URL https://doi.org/10.1145/3476999.

Kwisthout J and Donselaar N (2020) On the computational power and complexity of spiking neural networks. In: *Proceedings of the Neuro-inspired Computational Elements Workshop*. pp. 1–7.

Lattner C, Amini M, Bondhugula U, Cohen A, Davis A, Pienaar J, Riddle R, Shpeisman T, Vasilache N and Zinenko O (2021) Mlir: Scaling compiler infrastructure for domain specific computation. In: *2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*. pp. 2–14. DOI:10.1109/CGO51591.2021.9370308.

Lee S, Meredith JS and Vetter JS (2015) COMPASS: A framework for automated performance modeling and prediction. In: *Proceedings of the 29th ACM on International Conference on Supercomputing*. Newport Beach, California, USA: ACM, pp. 405–414. DOI:10.1145/2751205.2751220.

Li T, Hou J, Yan J, Liu R, Yang H and Sun Z (2020a) Chiplet heterogeneous integration technology—status and challenges. *Electronics* 9(4): 670.

Li Y, Fuller EJ, Sugar JD, Yoo S, Ashby DS, Bennett CH, Horton RD, Bartsch MS, Marinella MJ, Lu WD and Talin AA (2020b) Memory devices: Filament-free bulk resistive memory enables deterministic analogue switching (adv. mater. 45/2020). *Advanced Materials* 32(45): 2070339. DOI:https://doi.org/10.1002/adma.202070339.

Liu D, Cheng H, Zhu X, Wang G and Wang N (2013) Analog memristors based on thickening/thinning of ag nanofilaments in amorphous manganite thin films. *ACS Applied Materials & Interfaces* 5(21): 11258–11264. DOI:10.1021/am403497y.

Liu F, Miniskar NR, Chakraborty D and Vetter JS (2020) Deffe A data-efficient framework for performance characterization in domain-specific computing. In: *Proceedings of the 17th ACM International Conference on Computing Frontiers*, CF '20. New York, NY, USA: Association for Computing Machinery, p. 182–191.

Marinella MJ (2021) Radiation effects in advanced and emerging nonvolatile memories. *IEEE Transactions on Nuclear Science* 68(5): 546–572. DOI:10.1109/TNS.2021.3074139.

Marinella MJ, Agarwal S, Hsia A, Richter I, Jacobs-Gedrim R, Niroula J, Plimpton SJ, Ipek E and James CD (2018) Multiscale co-design analysis of energy, latency, area, and accuracy of a reRAM analog neural training accelerator. *Ieee Journal on Emerging and Selected Topics in Circuits and Systems* 8(1): 86–101. DOI:10.1109/jetcas.2018.2796379.

Marinella MJ, Dalton SM, Mickel PR, Dodd PED, Shaneyfelt MR, Bielejec E, Vizkelethy G and Kotula PG (2012) Initial assessment of the effects of radiation on the electrical characteristics of $\mathrm{tao_x}$ memristive memories. *IEEE Transactions on Nuclear Science* 59(6): 2987–2994. DOI:10.1109/TNS.2012.2224377.

Marinella MJ, Xiao TP, Feinberg B, Bennett C, Agrawal V, Puchner H and Agarwal S (2022) Achieving accurate in-memory neural network inference with highly overlapping nonvolatile memory state distributions. In: *2022 6th IEEE Electron Devices Technology & Manufacturing Conference (EDTM)*. pp. 330–332. DOI:10.1109/EDTM53872.2022.9797919.

McCaskey A and Nguyen T (2021) A mlir dialect for quantum assembly languages.

McLellan P (2020) Hbi, a new standard to connect your chiplets. URL https://community.cadence.com/cadence_blogs_8/b/breakfast-bytes/posts/hbi-a-new-standard-to-connect-your-chiplets.

Menzel S, Waters M, Marchewka A, Böttger U, Dittmann R and Waser R (2011) Origin of the ultra-nonlinear switching kinetics in oxide-based resistive switches. *Advanced Functional Materials* 21(23): 4487–4492. DOI:https://doi.org/10.1002/adfm.201101117.

Messaris I, Serb A, Stathopoulos S, Khiat A, Nikolaidis S and Prodromakis T (2018) A data-driven verilog-a reram model. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 37(12): 3151–3162. DOI:10.1109/TCAD.2018.2791468.

Neckar A, Fok S, Benjamin BV, Stewart TC, Oza NN, Voelker AR, Eliasmith C, Manohar R and Boahen K (2019) Braindrop: A mixed-signal neuromorphic architecture with a dynamical systems-based programming model. *Proceedings of the IEEE* 107(1): 144–164. DOI:10.1109/JPROC.2018.2881432.

Nili H, Vincent AF, Prezesio M, Mahmoodi MR, Kataeva I and Strukov DB (2020a) Comprehensive compact phenomenological modeling of integrated metal-oxide memristors. *IEEE Transactions on Nanotechnology* 19: 344–349. DOI:10.1109/TNANO.2020.2982128.

Nili H, Vincent AF, Prezioso M, Mahmoodi MR, Kataeva I and Strukov DB (2020b) Comprehensive compact phenomenological modeling of integrated metal-oxide memristors. *IEEE TRANSACTIONS ON NANOTECHNOLOGY* 19: 344–349. DOI:10.1109/TNANO.2020.2982128.

of Particles D and of the American Physical Society F (2021) The particle physics community planning exercise (snowmass). URL https://snowmass21.org/.

Park H, Ku BW, Chang K, Shim DE and Lim SK (2020) Pseudo-3D approaches for commercial-grade RTL-to-GDS tool flow targeting monolithic 3D ICs. In: *Proceedings of the 2020 International Symposium on Physical Design*. pp. 47–54.

Parsa M, Kulkarni SR, Coletti M, Bassett J, Mitchell JP and Schuman CD (2021) Multi-objective hyperparameter optimization for spiking neural network neuroevolution. In: *2021 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, pp. 1225–1232.

Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z, Gimelshein N, Antiga L, Desmaison A, Kopf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J and Chintala S (2019) Pytorch: An imperative style, high-performance deep learning library. In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., pp. 8024–8035. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Patton R, Schuman C, Kulkarni S, Parsa M, Mitchell JP, Haas NQ, Stahl C, Paulissen S, Date P, Potok T et al. (2021) Neuromorphic computing for autonomous racing. In: *International Conference on Neuromorphic Systems 2021*. pp. 1–5.

Peng IB and Vetter JS (2018) Siena: exploring the design space of heterogeneous memory systems. In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*. Dallas, Texas: IEEE Press, pp. 1–14.

Raoux S, Burr GW, Breitwisch MJ, Rettner CT, Chen YC, Shelby RM, Salinga M, Krebs D, Chen SH, Lung HL and Lam CH (2008) Phase-change random access memory: A scalable technology. *IBM Journal of Research and Development* 52(4.5): 465–479. DOI:10.1147/rd.524.0465.

Rasmussen D (2018) NengoDL: Combining deep learning and neuromorphic modelling methods. *arXiv* 1805.11144: 1–22. URL http://arxiv.org/abs/1805.11144.

Schemmel J, Briiderle D, Griibl A, Hock M, Meier K and Millner S (2010) A wafer-scale neuromorphic hardware system for large-scale neural modeling. In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. pp. 1947–1950. DOI:10.1109/ISCAS.2010.5536970.

Schliebs S, Mohemmed A and Kasabov N (2011) Are probabilistic spiking neural networks suitable for reservoir computing? In: *The 2011 International Joint Conference on Neural Networks*. IEEE, pp. 3156–3163.

Schulte MJ, Ignatowski M, Loh GH, Beckmann BM, Brantley WC, Gurumurthi S, Jayasena N, Paul I, Reinhardt SK and Rodgers G (2015) Achieving exascale capabilities through heterogeneous computing. *Micro, IEEE* 35(4): 26–36. DOI:10.1109/MM.2015.71.

Schuman CD, Hamilton K, Mintz T, Adnan MM, Ku BW, Lim SK and Rose GS (2019) Shortest path and neighborhood subgraph extraction on a spiking memristive neuromorphic implementation. In: *Proceedings of the 7th Annual Neuro-inspired Computational Elements Workshop*. pp. 1–6.

Schuman CD, Kay B, Date P, Kannan R, Sao P and Potok TE (2021) Sparse binary matrix-vector multiplication on neuromorphic computers. In: *2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, pp. 308–311.

Schuman CD, Mitchell JP, Patton RM, Potok TE and Plank JS (2020) Evolutionary optimization for neuromorphic systems. In: *Proceedings of the Neuro-inspired Computational Elements Workshop*. pp. 1–9.

Schuman CD, Plank JS, Disney A and Reynolds J (2016) An evolutionary optimization framework for neural networks and neuromorphic architectures. In: *2016 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 145–154.

Schuman CD, Potok TE, Patton RM, Birdwell JD, Dean ME, Rose GS and Plank JS (2017) A survey of neuromorphic computing and neural networks in hardware. *arXiv preprint arXiv:1705.06963* .

Severa W, Lehoucq R, Parekh O and Aimone JB (2018a) Spiking neural algorithms for markov process random walk. In: *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 1–8.

Severa W, Vineyard CM, Dellana R, Verzi SJ and Aimone JB (2018b) Whetstone: A method for training deep artificial neural networks for binary communication. *CoRR* abs/1810.11521. URL http://arxiv.org/abs/1810.11521.

Sharma DD (2022) Universal chiplet interconnect express (ucie)®: Building an open chiplet ecosystem. Technical report, Universal Chiplet Interconnect Express.

Sharma S, Aubin S and Eliasmith C (2016) Large-scale cognitive model design using the nengo neural simulator. *Biologically Inspired Cognitive Architectures* 17: 86–100. DOI:https://doi.org/10.1016/j.bica.2016.05.001. URL https://www.sciencedirect.com/science/article/pii/S2212683X16300317.

Shrestha SB and Orchard G (2018) SLAYER: Spike layer error reassignment in time. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N and Garnett R (eds.) *Advances in Neural Information Processing Systems 31*. Curran Associates, Inc., pp. 1419–1428. URL http://papers.nips.cc/paper/7415-slayer-spike-layer-error-reassignment-in-time.pdf.

Spafford K, Vetter JS, Benson T and Parker M (2013) Modeling synthetic aperture radar computation with aspen. *International Journal of High Performance Computing Applications* 27(3): 255–262. DOI:10.1177/1094342013488262.

Spafford KL and Vetter JS (2012) Aspen: A domain specific language for performance modeling. In: *SC12: International Conference for High Performance Computing, Networking, Storage and Analysis*. Salt Lake City, pp. 1–11. DOI:10.1109/SC.2012.20.

Spafford KL and Vetter JS (2015) Automated design space exploration with aspen. *Scientific Programming* 2015: 10. DOI:10.1155/2015/157305.

Stark J (2019) Chiplets: The path to iot diversity. Technical report, Cambridge Consultants.

Stettler MA, Cea SM, Hasan S, Jiang L, Keys PH, Landon CD, Marepalli P, Pantuso D and Weber CE (2021) Industrial tcad: Modeling atoms to chips. *IEEE TRANSACTIONS ON ELECTRON DEVICES* 68(11): 5350–5357. DOI:10.1109/

TED.2021.3076976.

Stimberg M, Brette R and Goodman DF (2019) Brian 2, an intuitive and efficient neural simulator. *eLife* 8: e47314. DOI:10.7554/eLife.47314.

Strukov DB and Williams RS (2009) Exponential ionic drift: fast switching and low volatility of thin-film memristors. *Applied Physics A* 94(3): 515–519. DOI:10.1007/s00339-008-4975-3.

Talin AA, Li Y, Robinson DA, Fuller EJ and Kumar S (????) Ecram materials, devices, circuits and architectures: A perspective. *Advanced Materials* n/a(n/a): 2204771. DOI:https://doi.org/10.1002/adma.202204771. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/adma.202204771.

Talin AA, Li Y, Robinson DA, Fuller EJ and Kumar S (2022) ECRAM materials, devices, circuits and architectures: A perspective. *Advanced Materials* n/a(n/a): 2204771. DOI:https://doi.org/10.1002/adma.202204771.

Tavanaei A, Ghodrati M, Kheradpisheh S, Masquelier T and Maida A (2019) Deep learning in spiking neural networks. *Neural Networks* 111: 47–63. DOI:https://doi.org/10.1016/j.neunet.2018.12.002. URL https://www.sciencedirect.com/science/article/pii/S0893608018303332.

Terai M, Sakotsubo Y, Kotsuji S and Hada H (2010) Resistance controllability of ta2O5/tiO2 stack reRAM for low-voltage and multilevel operation. *Electron Device Letters, IEEE* 31: 204–206. DOI:10.1109/LED.2009.2039021.

Tselev A, Yu P, Cao Y, Dedon LR, Martin LW, Kalinin SV and Maksymovych P (2016) Microwave a.c. conductivity of domain walls in ferroelectric thin films. *NATURE COMMUNICATIONS* 7. DOI:10.1038/ncomms11630.

Umar M, Meredith JS, Vetter JS and Cameron KW (2016) A study of power-performance modeling using a domain-specific language. In: *2016 28th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)*. pp. 84–92. DOI:10.1109/SBAC-PAD.2016.19.

Umar M, Moore SV, Vetter JS and Cameron KW (2018) Prometheus: Coherent exploration of hardware and software optimizations using aspen. In: *2018 IEEE 26th International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. pp. 244–250. DOI:10.1109/MASCOTS.2018.00032.

Valov I, Waser R, Jameson JR and Kozicki MN (2011) Electrochemical metallization memories—fundamentals, applications, prospects. *Nanotechnology* 22(25): 254003. DOI:10.1088/0957-4484/22/25/254003.

Vetter JS, Brightwell R, Gokhale M, McCormick P, Ross R, Shalf J, Antypas K, Donofrio D, Humble T, Schuman C, Essen BV, Yoo S, Aiken A, Bernholdt D, Byna S, Cameron K, Cappello F, Chapman B, Chien A, Hall M, Hartman-Baker R, Lan Z, Lang M, Leidel J, Li S, Lucas R, Mellor-Crummey J, Jr PP, Peterka T, Strout M and Wilke J (2018) Extreme heterogeneity 2018 - productive computational science in the era of extreme heterogeneity: Report for DOE ASCR workshop on extreme heterogeneity. Technical report, USDOE Office of Science (SC) (United States). DOI:10.2172/1473756.

Vogel T, Zintler A, Kaiser N, Guillaume N, Lefevre G, Lederer M, Serra AL, Piros E, Kim T, Schreyer P, Winkler R, Nasiou D, Olivo RR, Ali T, Lehninger D, Arzumanov A, Charpin-Nicolle C, Bourgeois G, Grenouillet L, Cyrille MC, Navarro G, Seidel K, Kaempfe T, Petzold S, Trautmann C, Molina-Luna L and Alff L (2022) Structural and electrical response of emerging memories exposed to heavy ion radiation. *ACS NANO* DOI:10.1021/acsnano.2c04841.

Wong HSP, Raoux S, Kim S, Liang J, Reifenberg JP, Rajendran B, Asheghi M and Goodson KE (2010) Phase change memory. *Proceedings of the IEEE* 98(12): 2201–2227. DOI:10.1109/JPROC.2010.2070050.

Xiao TP, Bennett CH, Agarwal S, Hughart DR, Barnaby HJ, Puchner H, Prabhakar V, Talin AA and Marinella MJ (2021a) Ionizing radiation effects in SONOS-based neuromorphic inference accelerators. *Ieee Transactions on Nuclear Science* 68(5): 762–769. DOI:10.1109/tns.2021.3058548.

Xiao TP, Bennett CH, Agarwal S, Hughart DR, Barnaby HJ, Puchner H, Talin AA and Marinella MJ (2022a) Single-event effects induced by heavy ions in SONOS charge trapping memory arrays. *Ieee Transactions on Nuclear Science* 69(3): 406–413. DOI:10.1109/tns.2021.3127549.

Xiao TP, Feinberg B, Bennett CH, Agrawal V, Saxena P, Prabhakar V, Ramkumar K, Medu H, Raghavan V, Chettuvetty R, Agarwal S and Marinella MJ (2022b) An accurate, error-tolerant, and energy-efficient neural network inference engine based on SONOS analog memory. *IEEE Transactions on Circuits and Systems I: Regular Papers* 69(4): 1480–1493. DOI:10.1109/TCSI.2021.3134313.

Xiao TP, Feinberg B, Bennett CH, Prabhakar V, Saxena P, Agrawal V, Agarwal S and Marinella MJ (2021b) On the accuracy of analog neural network inference accelerators. *CoRR* abs/2109.01262. URL https://arxiv.org/abs/2109.01262.

Yakopcic C, Rahman N, Atahary T, Taha TM and Douglass S (2020) Solving constraint satisfaction problems using the loihi spiking neuromorphic processor. In: *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, pp. 1079–1084.

Yang JJ, Strukov DB and Stewart DR (2013a) Memristive devices for computing. *Nat Nanotechnol* 8(1): 13–24. DOI:10.1038/nnano.2012.240.

Yang JJ, Strukov DB and Stewart DR (2013b) Memristive devices for computing. *NATURE NANOTECHNOLOGY* 8(1): 13–24. DOI:10.1038/NNANO.2012.240.

Yi S, Talin AA, Marinella MJ and Williams RS (2022) Physical compact model for three-terminal SONOS synaptic circuit element. *Advanced Intelligent Systems* 4(9). DOI:10.1002/aisy.202200070.

Young AR, Dean ME, Plank JS and Rose GS (2019) A review of spiking neuromorphic hardware communication systems 7: 135606–135620. DOI:10.1109/ACCESS.2019.2941772. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8843969&isnumber=8600701.

Acknowledgements

This research is funded by the DOE Office of Science Research Program for Microelectronics Codesign (sponsored by ASCR, BES, HEP, NP, and FES) through the Abisko Project with program managers Robinson Pino (ASCR), Hal Finkel (ASCR), and Andrew Schwartz (BES). We thank Olha Popova, Liangbo Liang and Anton Ievlev at ORNL CNMS for their on-going role in the building and development of the materials thrust of the project.

This manuscript has been authored by UT-Battelle LLC under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for US government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan)

This manuscript has been authored by Sandia National Laboratories that is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

Author Biographies

Jeffrey S. Vetter is a Corporate Fellow and the Section Head for Advanced Computer Systems Research at Oak Ridge National Laboratory (ORNL). Vetter earned his Ph.D. in Computer Science from the Georgia Institute of Technology. He is the Principal Investigator of the Abisko Microelectronics Codesign project.

Prasanna Date is a Research Scientist at the Oak Ridge National Laboratory. He received his PhD from Rensselaer Polytechnic Institute in 2019. He explores novel AI and machine learning techniques on neuromorphic and quantum computing platforms. For his research, Date was featured on the 2022 Forbes 30 Under 30 Asia list. He is an Associate Editor of IEEE TNNLS journal and an Editorial Board Member for Nature Scientific Reports. Date is on the organizing committee of marquee conferences in neuromorphic and quantum computing such as ACM ICONS and IEEE Quantum Week.

Farah Fahim (farah@fnal.gov) received her M.Tech research degree from the University of Limerick, Ireland, in 2011 and her Ph.D. degree in electrical engineering from Northwestern University, Evanston, Illinois, USA, in 2019. She has been with the Fermi National Accelerator Laboratory, Batavia, Illinois, 60510, USA, since 2009, specializing in mixed-signal ASIC design, and is currently the Division Head of Microelectronics. She is also an adjunct professor at Northwestern University Department of Electrical and Computer Engineering. For over 15 years she has been developing low-noise, high-speed readout and control electronics for detectors that operate in harsh environments such as high-ionizing radiation for a wide range of applications. She has been awarded five patents and has coauthored more than 40 publications.

Shruti Kulkarni is a research scientist at the Oak Ridge National Laboratory in the Learning Systems group. Her research spans different aspects of neuromorphic computing including algorithms, applications, and hardware codesign. She earned her PhD in 2019 from New Jersey Institute of Technology supervised by Dr. Bipin Rajendran, where she worked on bio-inspired learning and hardware acceleration with emerging memories. She was a postdoctoral research associate with Dr. Catherine Schuman at ORNL studying evolutionary optimization for spiking neural networks and scaling up SNN simulations to HPC systems.

Dr. Petro Maksymovych is a Distinguished Staff Scientist at Oak Ridge National Laboratory and a Theme Leader at the Center for Nanophase Materials Sciences. He received his Ph. D. from the University of Pittsburgh in 2007, focusing on hot-electron dynamics and surface chemical reactions as building blocks of molecular electronic circuits. His research interests at ORNL are aimed at the emergence of phase transitions in classical and quantum materials and the fundamental material properties that can enable future computational paradigms, including ferroelectrics, complex and correlated vdW materials, and the development of new scanning probe techniques for nanoscale characterization of phase-ordered media.

Alec Talin received Ph.D. in Materials Science and Engineering from UCLA in 1995. He is a Distinguished Member of Technical Staff at Sandia National Laboratories, an Adjunct Associate Professor of Materials Science at the University of Maryland, College Park, and is a Fellow of the American Physical Society. Prior to joining Sandia, Alec spent 6 years at Motorola Labs in Phoenix, AZ and 3 years at the National Institute of Standards and Technology in Gaithersburg, MD. His interests focus on nanoelectronics and nanoionics, with applications to energy-efficient computing, energy conversion, energy storage and national security.

Marc Gonzalez Tallada is a Senior Research Scientist at the Programming Systems Group at ORNL. He received the degree in computer science in 1996 and the PhD degree in computer science in 2003, both from the Universitat Politècnica de Catalunya (UPC). His research interests are related to parallel programming models, languages, and compilers for High Performance Computing technologies.

Pruek Vanna-iampikul received the B.E degree in computer engineering from King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand in 2012, and the M.E. degree in microelectronics and embedded systems from Asian Institute of Technology, Pathumthani, Thailand in 2017. He is currently pursuing the Ph.D. degree under Prof. S. K. Lim's guidance. His current research focuses on design algorithms and methodology for energy efficient 2.5D and 3D ICs.

Aaron Young is a Software Engineer in the Architectures and Performance Group at Oak Ridge National Laboratory. He received his Ph.D. in Computer Engineering from the University of Tennessee in 2020, where he completed his dissertation on a scaled-up neuromorphic array communications controller that allowed neural networks to scale across multiple neuromorphic processors implemented using FPGAs. His research interests include neuromorphic computing, high-performance computing, heterogeneous computing, computer architectures, embedded systems, and high-speed communication.

David Brook the Haley Family Professor of computer science with the School of Engineering and Applied Sciences, Harvard University. His research interests include resilient and power-efficient computer hardware and software design for high-performance and embedded systems.

Yu Cao is a Professor of Electrical Engineering at Arizona State University. He received his Ph.D. in 2002 from the University of California, Berkeley. His research interests include neural-inspired computing, hardware design for on-chip learning, and reliable integration of nanoelectronics.

Dr. Wei received his B.S.E.E., M.S., and Ph.D. in Electrical Engineering from Stanford University in 1994, 1997, and 2001, respectively. In 2000, he joined Accelerant Networks (now a part of Synopsys) in Beaverton, Oregon as a Senior Design Engineer. In 2002, he joined Harvard University. His research interests span a variety of topics such as integrated voltage regulators,

flexible voltage stacking, power electronics, low-power computing architectures and circuits, auto-parallelizing compilers, and more.

Dr. Sung Kyu Lim is the Motorola Solutions Foundation Professor at Georgia Institute of Technology's School of Electrical and Computer Engineering, and he previously has served as the principal investigator for multiple DARPA programs. Lim received his bachelor's, master's, and doctorate degrees from the Computer Science Department, University of California, Los Angeles in 1994, 1997, and 2000, respectively.

Frank Liu is the research manager (Group Leader) of the Architecture and Performance Group at Oak Ridge National Lab. His research interests include scientific machine learning and its influence on computer architecture

Matthew Marinella is an Associate Professor of Electrical Engineering at Arizona State University. From 2010 to 2021, Matthew J. Marinella was with Sandia's Microsystems S&T Center, where he was a Distinguished Member of the Technical Staff. At Sandia, Dr. Marinella led numerous internal and externally funded research projects involving neuromorphic and low-power computing with emerging electronic devices. He has served in technical advising and leadership roles in various Lab- and DOE-level initiatives on next generation computing for government applications. Dr. Marinella is a member of the SRC Decadal Plan Executive Committee, chairs the Emerging Memory Devices Section for the IRDS Roadmap Beyond CMOS Chapter, and serves on various technical program committees.

Bobby Sumpter is a Corporate Fellow and the Section Head for Theory and Computing at the Center for Nanophase Materials Science, Oak Ridge National Laboratory (ORNL).He received his Ph.D. in Physical Chemistry from Oklahoma State University. Sumpter's research is focused on a fundamental understanding of self-assembly processes, interactions at interfaces, the structure and dynamics of molecular-based materials including multi-component polymers and composites, and the physical, mechanical and electronic properties of nanostructured materials.

Narasinga Rao Miniskar is research software engineer in architecture performance group at the Oak Ridge National Laboratory with broad experience in heterogeneous computing, hardware/software codesign, reconfigurable architectures and high performance computing. With a strong interest in Neuromorphic computing, he currently focuses on the development of Neuromorphic accelerators at ORNL.