SANDIA REPORT
SAND2022-13310
Printed September 28, 2022

ATHENA: Analytical Tool for Heterogeneous Neuromorphic Architectures

Mark Plagge, Luke Parker, Ben Feinberg, Sapan Agarwal, Fred Rothganger, Clay Hughes and Suma G. Cardwell

Center for Computing Research
Sandia National Laboratories
Albuquerque, NM, 87185

{mplagge,lgparke, bfeinbe, sagarwa, frothga, chughes, sgcardw}@sandia.gov

John McFarland, Amro Awad

North Carolina State University
Raleigh, North Carolina, 27695

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185
Livermore, California 94550

ABSTRACT

The Advanced Simulation and Computing (ASC) program seeks to use machine learning to improve efficiencies in its stockpile stewardship mission. Moreover, there is a growing market for technologies dedicated to accelerating Artificial Intelligence (AI) workloads. Many of these emerging architectures promise to provide savings in energy efficiency, area, and latency when compared to traditional CPUs for these types of applications. In particular, neuromorphic analog and digital technologies provide both low-power and configurable acceleration of challenging artificial intelligence (AI) algorithms. If designed into a heterogeneous system with other accelerators and conventional compute nodes, these technologies have the potential to augment the capabilities of traditional High Performance Computing (HPC) platforms. This expanded computation space requires not only a new approach to physics simulation, but the ability to evaluate and analyze next-generation architectures specialized for AI/ML workloads in both traditional HPC and embedded nuclear deterrence (ND) applications. Developing this capability will enable the ASC program to understand how this hardware performs in both HPC and ND environments, improve our ability to port our applications, guide the development of computing hardware, and inform vendor interactions, leading them toward solutions that address ASC's unique requirements.

This report discusses the progress in developing and expanding the tool ecosystem at Sandia National Laboratories with a focus on the tool ATHENA: Analytical Tool to Evaluate Heterogeneous Neuromorphic Architectures, for fast design space exploration for emerging next-generation machine learning architectures.

ACKNOWLEDGMENT

The authors acknowledge financial support from the DOE Advanced Simulation and Computing program. Sandia National Laboratories is a multi-mission 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-NA0003525. This report describes technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government. SAND Number: SAND2022-13310

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS AND DEFINITIONS

ASC	Advanced Simulation and Computing
ATHENA	Analytical Tool for Heterogeneous Neuromorphic Architectures
ANN	Analog Neural Network
ARIAA	Artificial Intelligence focused Architectures and Algorithms
CMOS	Complementary Metal-Oxide-Semiconductor
CNN	Convolutional Neural Network
GEMM	General Matrix Multiplication
GenSA	General Spiking Architecture
HPC	High Performance Computing
LIF	Leaky Integrate and Fire
MAESTRO	Modeling Accelerator Efficiency via Spatio-Temporal Resource Occupancy
MAERI	Multiply-Accumulate Engine with Reconfigurable Interconnects
NoC	Network On Chip
AI	Artificial Intelligence
CGRA	Coarse Grained Reconfigurable Array
SST	Structural Simulation Toolkit
SWaP	Size Weight and Power
PE	Processing Element
MAC	Multiply Accumulate
SONOS	Silicon-Oxide-Nitride-Oxide-Silicon
MVM	Matrix Vector Multiplication
ERT	Energy Reference Tables
ART	Area Reference Tables
FG	Floating-gate

1. INTRODUCTION

The ASC program seeks to use machine learning to improve efficiencies in its stockpile stewardship mission. Moreover, there is a growing market for technologies dedicated to accelerating AI workloads. Many of these emerging architectures promise to provide savings in energy efficiency, area, and latency when compared to traditional CPUs for these types of applications — neuromorphic analog and digital technologies provide both low-power and configurable acceleration of challenging artificial intelligence (AI) algorithms. If designed into a heterogeneous system with other accelerators and conventional compute nodes, these technologies have the potential to augment the capabilities of traditional High Performance Computing (HPC) platforms [5]. This expanded computation space requires not only a new approach to physics simulation, but the ability to evaluate and analyze next-generation architectures specialized for AI/ML workloads in both traditional HPC and embedded ND applications. Developing this capability will enable ASC to understand how this hardware performs in both HPC and ND environments, improve our ability to port our applications, guide the development of computing hardware, and inform vendor interactions, leading them toward solutions that address ASC's unique requirements.

Currently, there are many viable AI/ML-focused next-generation computer architectures, each focused on different technologies and granularities, e.g. analog neural networks, dataflow accelerators, and spiking neural networks. Yet, they face a common obstacle — how to map an application's operations to individual hardware units and how to evaluate their performance quickly. The large design space, particularly with heterogeneous nodes, necessitates a simulation environment that can quickly eliminate poor accelerator choices. We developed a new tool, ATHENA (Analytical Tool to Evaluate Heterogeneous Neuromorphic Architectures), to quickly evaluate heterogeneous neuromorphic architectures to enable design space exploration of emerging machine learning architectures.

Technical Approach: ATHENA

Neuromorphic accelerators can impact the efficiency of machine learning, scientific computing, and trusted AI applications with two-three orders of magnitude improvement in energy and speed. This technology has the potential to augment the capabilities of traditional HPC platforms [5] when integrated into a heterogeneous system with other accelerators and conventional computing nodes. However, the lack of sufficient benchmarking for neuromorphic hardware is a deterrent to adopting these novel architectures.

ATHENA is an analytic modeling tool to enable design space exploration of neuromorphic accelerators in the context of highly heterogeneous architectures, enabling massively parallel computation and mixed-precision computing as shown in Figure 1-1. Analytical tools are approximate modeling tools that estimate performance of a given architecture (number of memory read/write, number of multiply-accumulate operations, network-on-chip communications), for a given technology node. Adding a non-CMOS or analog hardware element to it involved developing a similar solution in the analog device space. This tool will be extremely beneficial for rapid testing and architecture prototyping of neuromorphic components for deep learning. These results can then be used to provide to Structural Simulation Toolkit (SST) to provide a flexible cycle-approximate simulation foundation and simulate in more detail.

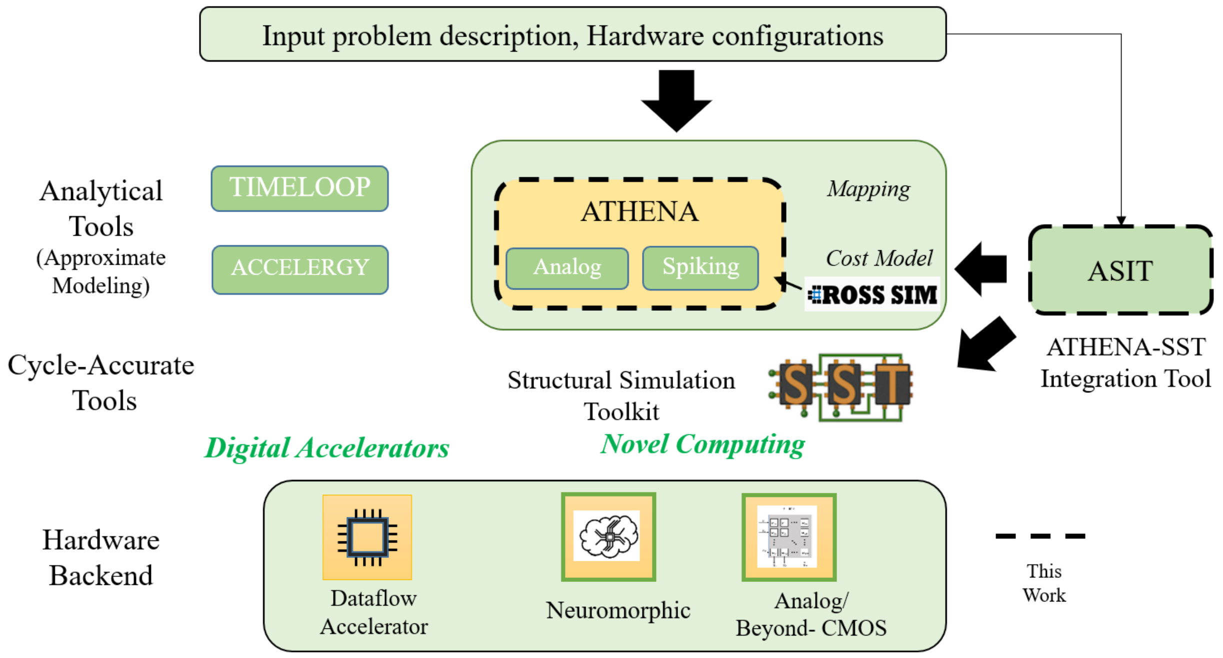


Figure 1-1. ATHENA Overview. ATHENA is a tool that extends existing accelerator tools to model digital/analog/neuromorphic kernels to enable evaluation of heterogeneous architectures for machine learning applications. We have also developed a tool called ASIT: ATHENA-SST Integration Tool, that leverages ATHENA for design space exploration for different hardware configurations (analog/spiking) and based on a metric of choice (energy,latency etc.) selects a hardware configuration to then simulate on SST. Once a 'best candidate' architecture is picked, then detailed simulations can be done on SST.

Chapter 2 reviews few open-source tools leveraged for machine learning dataflow accelerators, reviews an emerging analog architecture and also a brief overview of SST. Chapter 3 gives an overview of the ATHENA tool and specific example of the SONOS based analog accelerator and results. Chapter 4 details the development of the ASIT and reviews leveraging ATHENA to select a hardware configuration to simulate on SST.

2. BACKGROUND

Challenges in power scaling of conventional digital computing have ushered in a new ‘Golden Age in Computer Architecture’ [8]. A wide variety of computer architecture design tools have emerged to facilitate research into these novel and emerging computational architectures. This exploration ranges from less precise analytical assessments to high fidelity simulations. Example analytical approaches include Modeling Accelerator Efficiency via Spatio-Temporal Resource Occupancy (MAESTRO) and Eyeriss Eyexam [11, 6]. Other analytical tools focus upon assessing properties of a hardware architecture such as the utilization of resources and identifying what is an optimal dataflow strategy for the architecture. An example is the Timeloop tool [15]. More accurate but slower tools offer cycle accurate simulation capabilities. This increased fidelity often incorporates component models to attain the cycle accurate analysis and sometimes couples with executable hardware description level simulations. Examples include Systolic CNN AcceLErator Simulator (SCALE Sim) and Nvidia Deep Learning Accelerator (NVDLA) [22, 14]. The above techniques have largely focused on ML accelerator approaches such as systolic arrays and convolutional neural networks (CNN) accelerators. Additional interest is in how emerging neuromorphic architectures may also be modeled. For example, NeMo utilizes the Rensselaer’s optimistic simulation system (ROSS) in a discrete event simulation tool to provide a functional simulation of the IBM TrueNorth spiking neuromorphic architecture [17]. Other capabilities seek to account for the performance of emerging device technologies such as CrossSim and PUMA [1, 3]. Effectively, this spectrum of analytical modeling capabilities help enable co-design and the assessment of the impact of incorporating emerging ML accelerator and neuromorphic architectures into truly heterogeneous HPC systems [5].

2.1. Comparison of Existing Accelerator Tools

We evaluated existing accelerator tools to evaluate suitability to extend their functionality for novel accelerators. In particular we reviewed two tools, namely MAESTRO and Timeloop/Accelergy in our evaluation which is presented below.

2.1.1. MAESTRO/MAERI

MAESTRO (Modeling Accelerator Efficiency via Spatio-Temporal Resource Occupancy) is an open-source tool for modeling and evaluating dataflow accelerators. The MAESTRO cost model provides rapid estimation of the performance/energy given a DNN Model, its mapping, and hardware configuration. It can be used for HW-SW co-design by porting it into tools that perform design-space exploration [11]. MAERI (Multiply-Accumulate Engine with Reconfigurable

Interconnects) is a modular design-methodology for building DNN accelerators in RTL. It provides an efficient mapping of neural networks, which covers various DNN layer types and dimensions, state-of-the-art partitioning strategies (inter-layer fusion, intra-layer tiling, etc.) data density optimizations (sparsity, compression, etc.), data reuse strategies (dataflow classes based on stationary data) [12] .

2.1.1.1. Advantages

- MAESTRO has shorter execution time. Maestro determines the estimated energy and output of a network in milliseconds. Given a convolutional neural network, it will give an energy estimate for the entire network as well.
- Ease of use. Maestro is fairly easy to use out of the box. The workflow for an end-user is to gather hardware specification for a systolic style accelerator, convert a PyTorch/Keras network into a model and dataflow.
- Actively supported with frequent updates.

2.1.1.2. Disadvantages

- Hard to customize. The high speed of MAESTRO comes at the cost of assumptions about the underlying hardware. The way MAESTRO computes network energy estimates is limited by the analytical method which works when used against systolic array-based accelerators. The codebase is monolithic and not documented well.
- Assumes NoC based systolic accelerator architecture.
- It uses single lookup-table for energy estimates and does not support dynamic models.
- MAESTRO supports a limited set of operations.

2.1.2. Timeloop/Accelergy

Timeloop and Accelergy are tools developed by Nvidia labs and MIT. The purpose of Timeloop is to provide a system that both finds the optimal dataflow for a given problem and a hardware definition. It does this through an optimized exhaustive search. If the dataflow is already known, Timeloop can output an energy estimate. Timeloop uses, by default, a lookup table for hardware costs. This can be enhanced by using Accelergy as a plugin. Accelergy is a tool developed to provide an easy-to-use architecture-level energy estimation for accelerator designs. Timeloop features three executables. These are a mapper, which optionally runs if a dataflow map is not specified. This requires a problem shape configuration. A model generator, which generates performance metrics from a given workload, architecture, and mapping. And a metrics generator, which gives statistics and information about a particular hardware definition. The following subsection details the advantages and disadvantages of Timeloop/Accelergy tools.

2.1.2.1. Advantages

- Accelergy is able to model a wide variety of hardware designs. Examples include very non-standard hardware including a ‘Process-In-Memory’ design.
- Plugin based: Accelergy can either use a functional-style plugin or a lookup table based plugin to generate energy estimates. This makes integration of novel devices easier.
- Accelergy takes run-time action counts as input and an architecture as an input. So if the user generates a set of operation/action counts, this could be piped into Accelergy.
- Timeloop can use Accelergy to generate energy estimates.
- Given an input problem shape, Timeloop can search valid dataflow mappings to find an optimal mapping.
- Timeloop, as input, takes a layer shape: Dimensions of input, output, and weights, along with operations. Timeloop focuses on convolutional layers and supports GEMM (General Matrix Multiplication) layers as well.
- Operations can be extended easily. There is a projection option that deals with Inputs and can apply dilation and stride modifier.
- Timeloop has a python interface that can call Timeloop and parse the results.

2.1.2.2. Disadvantages

- Timeloop’s execution time is slower if the mapping is not given.
- Timeloop’ s definitions are tricky to figure out (but they are in YAML text).
- Larger codebase. With Timeloop and Accelergy, both the code bases are extremely large.

2.2. Emerging Hardware: SONOS Based Analog accelerator

ATHENA leverages the SONOS (silicon-oxide-nitride-oxide-silicon) floating-gate based accelerator design as an initial exemplar for analog accelerators. This accelerator was developed by hardware groups at Sandia and Infineon Memory Technologies. These devices are fabricated in the embedded 40nm process and enable 8-bit *in situ* matrix multiplications. The SONOS analog memory arrays are optimized for neural network inference and have been shown to achieve 20 TOPS/W on ResNet-50 with a >10X gain in energy efficiency over state-of-the-art digital and analog inference accelerators [24].

SONOS floating-gates are a type of non-volatile memory device that enable matrix computation. Typically, digital dataflow accelerators use arrays of multiply-accumulate (MAC) units, but are limited by memory read/write and data movement costs. In analog MVM (Matrix-Vector Multiplication) arrays as shown in Figure 2-1(a), the input vector is encoded in the applied voltage to the rows $\vec{V}$, the weight matrix W is encoded in the memory cell conductances, and the

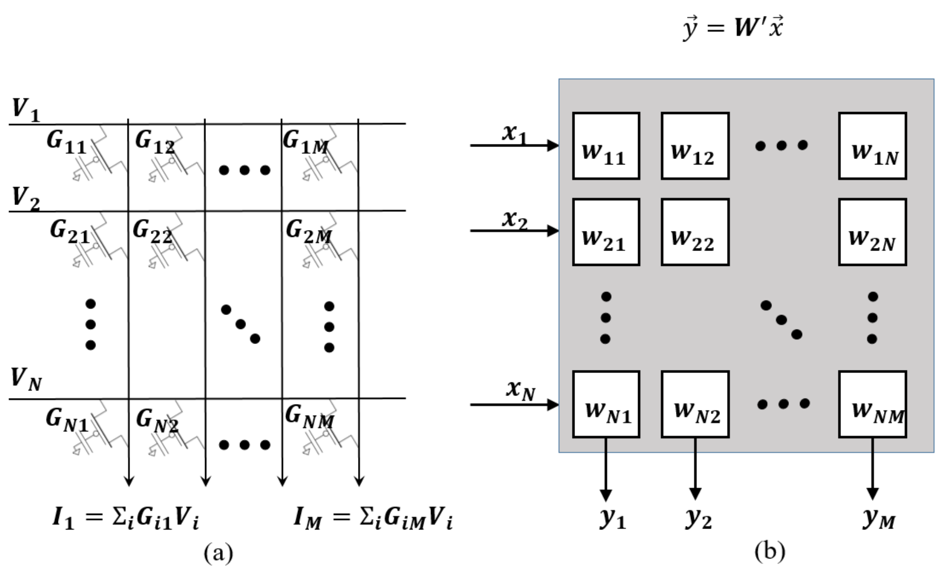

$\vec{y} = W'\vec{x}$
V_1
V_2
V_N
G_{11} G_{12} G_{1M}
G_{21} G_{22} G_{2M}
G_{N1} G_{N2} G_{NM}
$I_1 = \Sigma_i G_{i1} V_i$
$I_M = \Sigma_i G_{iM} V_i$
x_1 x_2 x_N
w_{11} w_{12} w_{1N}
w_{21} w_{22} w_{2N}
w_{N1} w_{N2} w_{NM}
y_1 y_2 y_M
(a)
(b)

Figure 2-1. Analog Crossbar Array (a) Schematic of an analog crossbar using floating-gates. (b) The analog crossbar essentially performs matrix multiplication. Different analog devices afford different precision per device.

dot product is the output $\vec{I}$ n the column currents. Using Kirchhoff's current law, products accumulate on the bit line. The current is then quantized using an analog-to-digital converter (ADC) and sent to the next layer's array. This is equivalent to Figure 2-1(b) matrix multiplication, where $\vec{V}$ is equivalent to $\vec{x}$, G is equivalent to weight matrix W and output $\vec{I}$ is equivalent to $\vec{y}$.

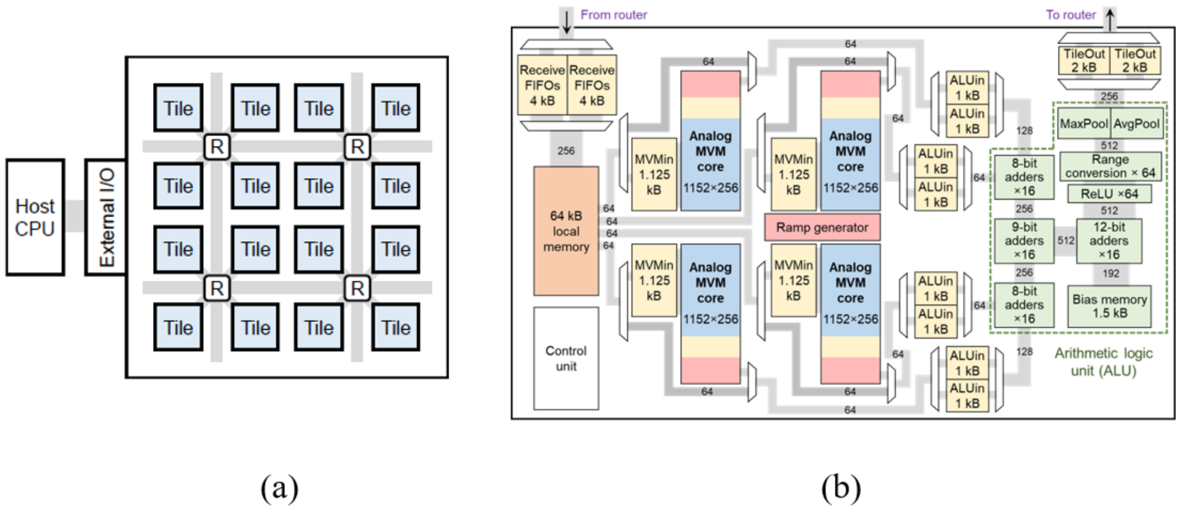


Figure 2-2. Overview of SONOS Floating-gate based analog accelerator architecture and tile. (a) Tile architecture for the SONOS analog inference accelerator (b) Detailed diagram of SONOS accelerator tile. Analog accelerator arrays are computationally denser than a convention digital accelerator PE with four 1152×256 MVM arrays in the same tile along with peripheral circuits, control unit and 64kB local memory. Images reproduced from [24]

Unlike digital dataflow accelerators, the SONOS analog accelerator tile contains multiple MVM arrays as seen in Fig. 2-2. This posed some challenges adapting to this novel accelerator as discussed in Section 3.4.

2.3. Structural Simulation Toolkit (SST)

SST is a tool that enables cycle-accurate simulations of extreme-scale architectures [20]. It enables the simulating and testing instruction set architectures, memory components, network interfaces, network on chip, dataflow accelerators. It has a modular design and parallel simulation environment based on MPI. This enables simulating large-scale systems on this tool.

The following sections briefly describe the analog and spiking tiles in SST. These can be leveraged for more detailed cycle-accurate simulation after an analytical tool like ATHENA is used for design-space exploration.

2.3.1. SST Analog Tile

An analog tile component was developed for SST, that can encapsulate recent developments in analog crossbars and capture their unique properties. It currently interfaces with CrossSIM, which is an analog crossbar simulation tool [18]. Each analog tile contains one or more array models, and an interface to the memory/NoC. The tile component is a memory request generator for tile memory and for inter-tile communication/synchronization. Different tile functionality can be simulated by modifying tile address mapping functions for data movement and addressing (e.g. producertoArray, arraytoConsumers etc.). It currently implements linear/direct mapping, fully connected layer, and convolutional layers. For more details on this model, please refer to [9].

2.3.2. SST Spiking Tile: Generic Spiking Architecture (GenSA)

SST also contains a spiking neuromorphic processing element called GenSA. This component will be part of the available hardware that ASIT can utilize when constructing heterogeneous machines. The GenSA model is under active development. It can now accept an arbitrary spiking neural network via an external specification file. The architecture configurations are currently limited to a few basic parameters. This set will be expanded in ongoing development work. The following is a summary of the goals of GenSA.

A ‘generic’ architecture must encompass the whole design space, such that all the existing architectures can be represented. The primary machines we consider are IBM’s TrueNorth [13], Intel’s Loihi [7], and University of Manchester’s SpiNNaker [10]. To a lesser extent, we consider Sandia’s own STPU, as well as the mixed-signal systems BrainScaleS and Neurogrid. The following is a notional diagram to illustrate the idea of a design space.

Figure 2-3 illustrates the components that tend appear in some form across a wide range neuromorphic devices. A neuromorphic machine must scale to a very large cluster, possibly with

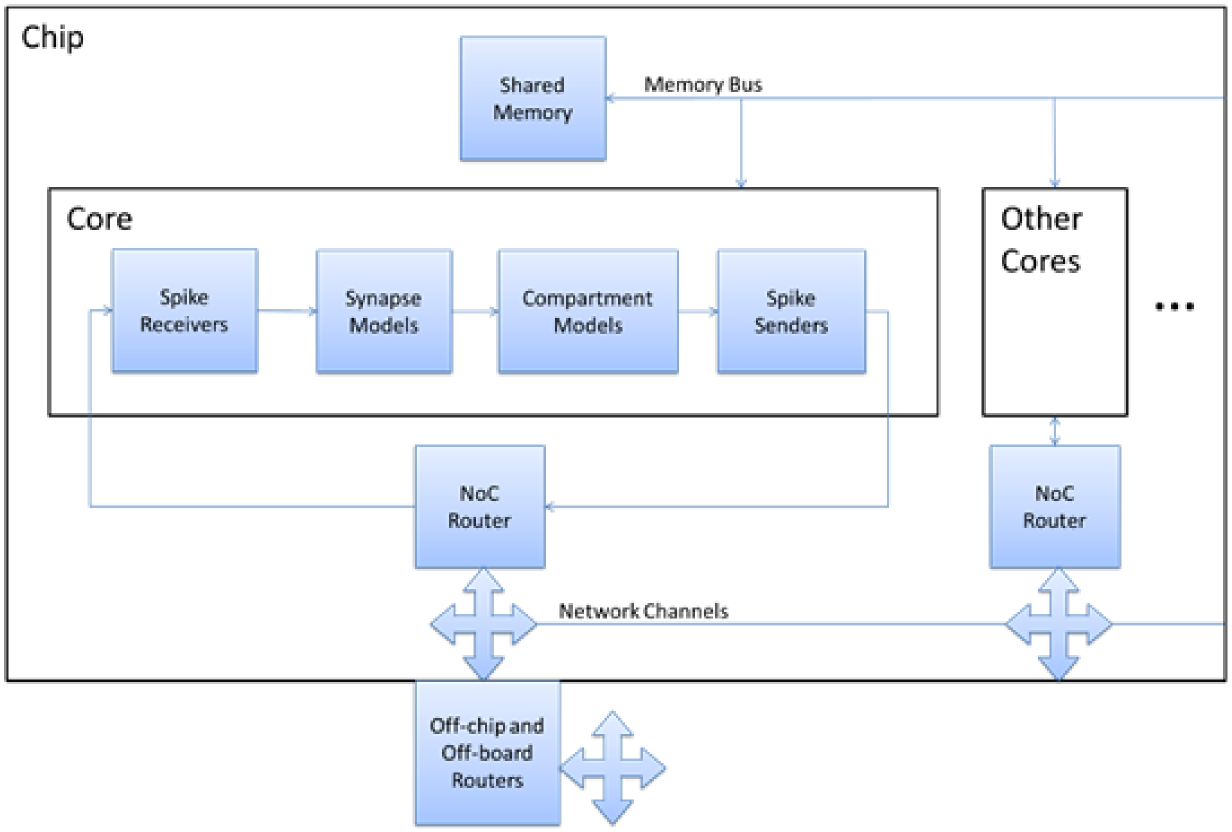


Figure 2-3. GenSA Workflow Overview. Image reproduced from [21].

enough processing units to represent an entire brain. There may be many levels to the machine organization to allow such an enormous system. Starting at the board level and working downward, there are several chips, each of which contains several cores. A network fabric crosses all the machine levels in order to move event messages between neural units. Each core will process several compartments (typically LIF: Leaky Integrate and Fire units). It must accept incoming event messages from the fabric, store them during their delay period and integrate them as they become active. The core must compute the dynamics of each compartment and send event messages out to the network fabric.

GenSA does not need to replicate each architecture exactly. It is sufficient that the generic machine be configured to provide similar performance characteristics under similar inputs (code + data). The SST element will include a Python script to construct a spiking device for a given set of parameters.

3. ATHENA: MODELING NOVEL ANALOG ML ACCELERATORS

Analog and neuromorphic accelerators can impact the efficiency of machine learning, scientific computing, and trusted AI applications with two-three orders of magnitude improvement in energy and speed [1]. The multiply-accumulate (MAC) unit is a core computational kernel of many problems, be it neural networks or partial differential equations (PDEs). Current efforts are focused on accelerating the MAC unit and optimizing data flow. Emerging analog memory devices can provide significant gain in energy efficiency and latency for vector matrix multiplication. However, the lack of sufficient benchmarks for analog and neuromorphic hardware is a deterrent to adopting these novel devices and architectures. Thus, developing tools for design space exploration of analog and neuromorphic accelerators is important to demonstrating their efficacy for deep learning applications.

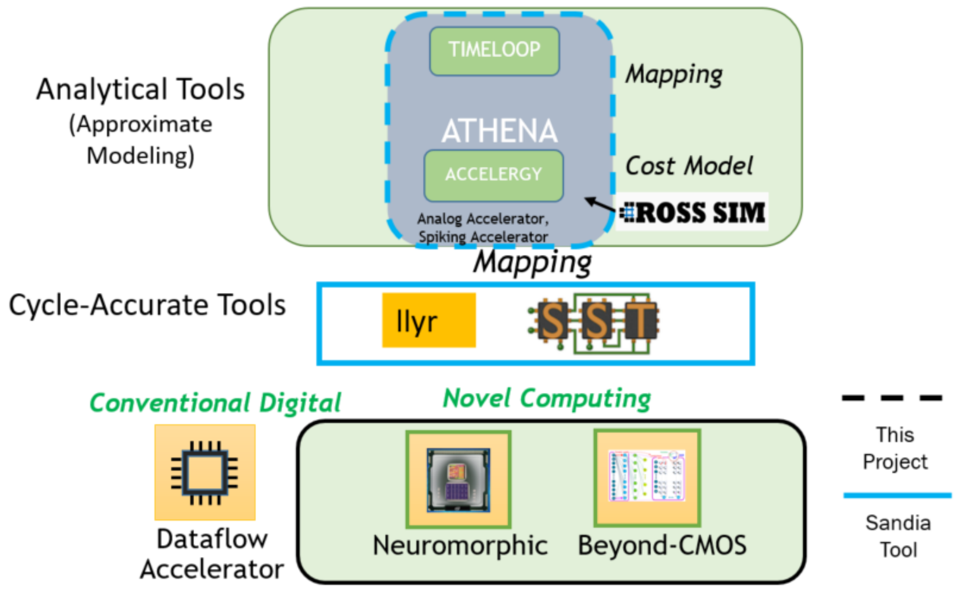


Figure 3-1. ATHENA is an analytical tool that can be leveraged for design space exploration of novel architectures that use emerging devices. It is an approximate tool that be used for performance estimation.

Our approach leverages existing deep learning accelerator modeling tools like Timeloop [15] and Accelergy [23] to explore analog and neuromorphic accelerator architectures. ATHENA extends the design space supported by these tools by incorporating hardware description and energy tables for analog kernels. Analytical tools work through computing energy related operations (number of memory read/write, number of multiply-accumulate operations, network-on-chip communications) given a certain technology node. Adding non-conventional analog and emerging devices involves developing a similar solution in the analog device space for characterization. This tool can be leveraged for rapid testing and architecture prototyping of novel analog accelerators. While Sandia's Structural Simulation Toolkit (SST) can yield cycle-accurate simulations, analytical tools provide quicker analysis for a given architecture. Other tools that account for the performance of emerging device technologies are CrossSim and PUMA [1, 3]. We leverage data from the CrossSim [19] tool within ATHENA to model emerging memories. In this paper we will present results based on a SONOS floating-gate based analog accelerator using ATHENA.

3.1. ATHENA Overview

ATHENA is intended to be an end-to-end tool that enables searching across a wider variety of hardware designs. ATHENA provides the ability to quickly examine the performance in terms of latency, energy requirements, and network traffic limitations of novel analog neuromorphic hardware. In addition, ATHENA will provide the ability to generate estimates of hardware from problems implemented in PyTorch, Tensorflow, and MLIR. By providing rapid performance estimates, ATHENA will enable us to quickly prototype new hardware designs. In addition to the rapid performance prototyping provided by ATHENA, we will also leverage more traditional simulation tools as the neuromorphic architecture matures.

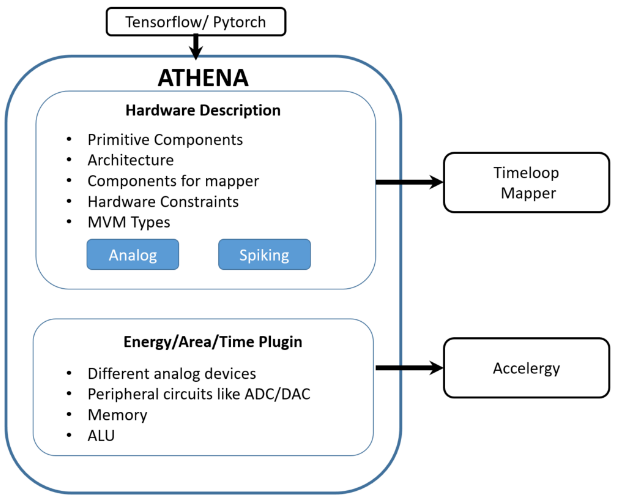


Figure 3-2. ATHENA is an analytical tool that can be leveraged for design space exploration of novel architectures that use emerging devices. It describes the hardware specifications and energy/area/timing tables for beyond-CMOS analog accelerators.

Our work leverages the existing work by [15], Timeloop, a dataflow style mapping tool that can estimate energy for CMOS logic based ML acceleration devices. Timeloop supports single layer mapping of a neural network. ATHENA generates multiple hardware/problem inputs for Timeloop. ATHENA adapts Accelergy hardware design descriptions to generate a hardware layout file for each layer of the run (changing active rows and columns). This allows for more dynamic energy estimates based on the problem size. We have adapted Timeloop to provide estimates of a tiled analog ML acceleration device [24]. This first implementation is a proof of concept of the suitability of analytical methods for estimating performance of analog devices.

Given the preliminary results estimating the energy and latency of this hardware against a deep learning network task, we hope to further ATHENA's capabilities by adding spiking neuromorphic hardware estimates as well as combining the ATHENA tool with an ecosystem of novel hardware design space exploration tools that interface with LLVM and MLIR for broad applicability.

3.2. Activation Functions

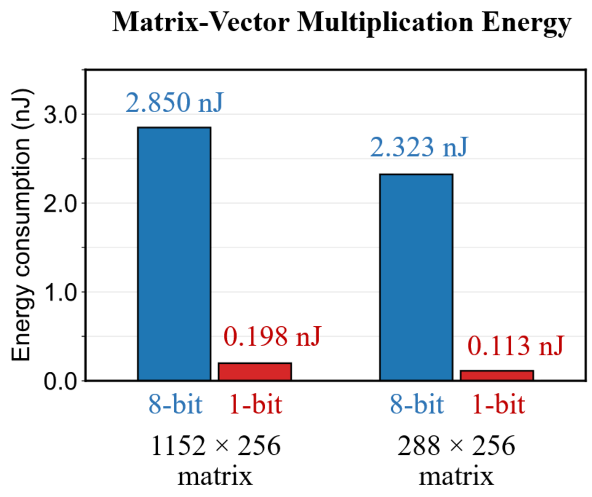
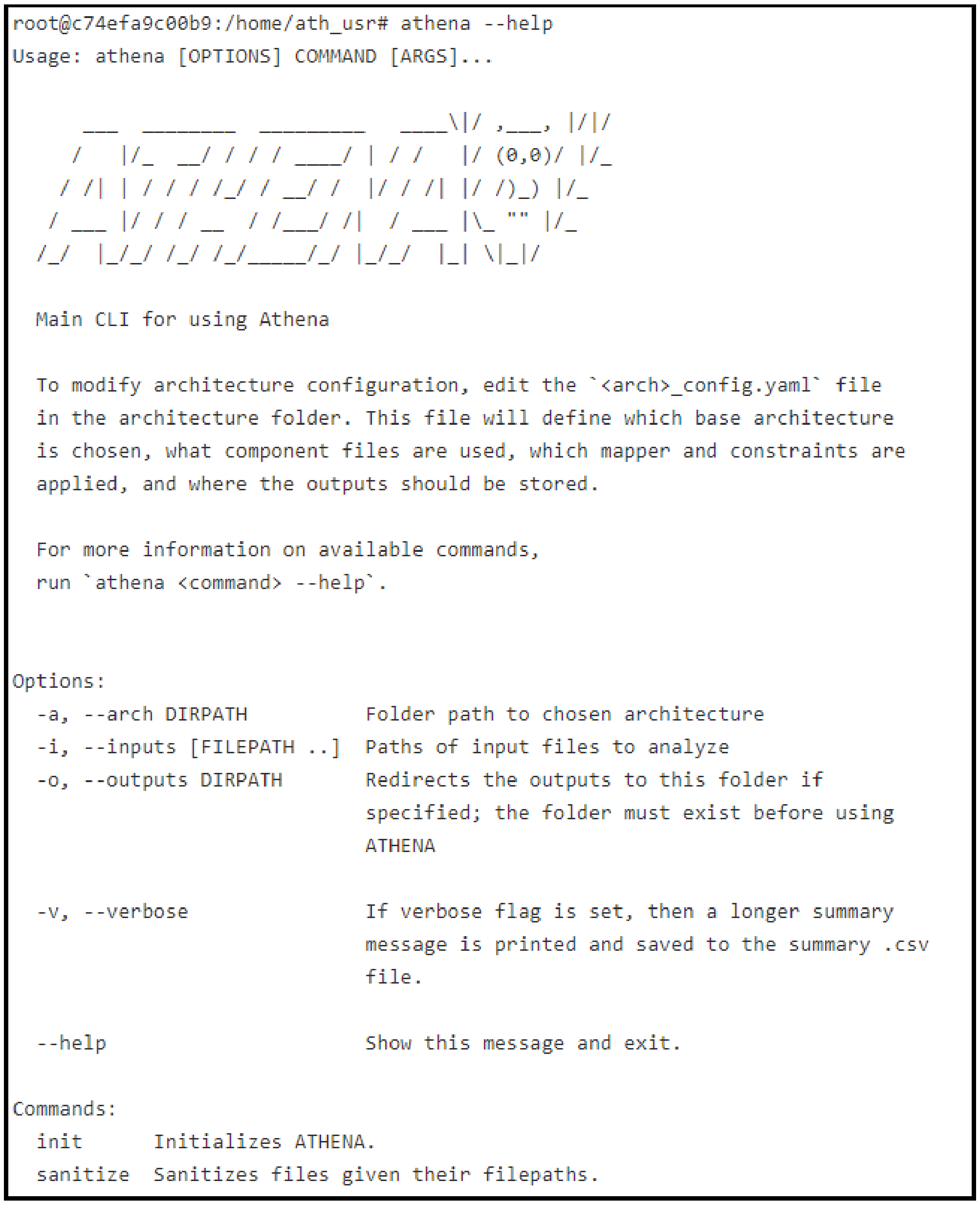


Figure 3-3. Impact of binary (1-bit) versus 8-bit activations in an analog MVM array using Resistive Random Access Memory (RRAM) devices. Reproduced from [2]

Typically, in analytical tools for dataflow accelerators, the cost of computing activation functions are largely ignored. When examining the energy of dataflow accelerator hardware, the cost of activation functions is relatively small when compared to the cost of the large MAC operation energy cost. This however is important when considering binary neural networks or spiking neural networks. Binary neural networks leverage simplified activation functions which could affect the total energy of an analog accelerator device. In Figure 3-3, the difference in energy between 8-bit and 1-bit activation functions computed on an analog array are compared. 1-bit activation functions are computed on the MVM array, allowing for significantly higher energy efficiency.

ATHENA has preliminary support for activation function hardware, but this is still a work in progress. To add support for activation functions, we first compute the size of the output dimensions of the running layer. Given a word-size in an ALU and a bit-precision from the network, we can determine the count of activation function operations that need to be completed for a given input layer. This allows ATHENA to compute the energy required to run the activation function circuitry for a given input layer. This functionality will be fully integrated in a future release of ATHENA.

3.3. Command Line Interface

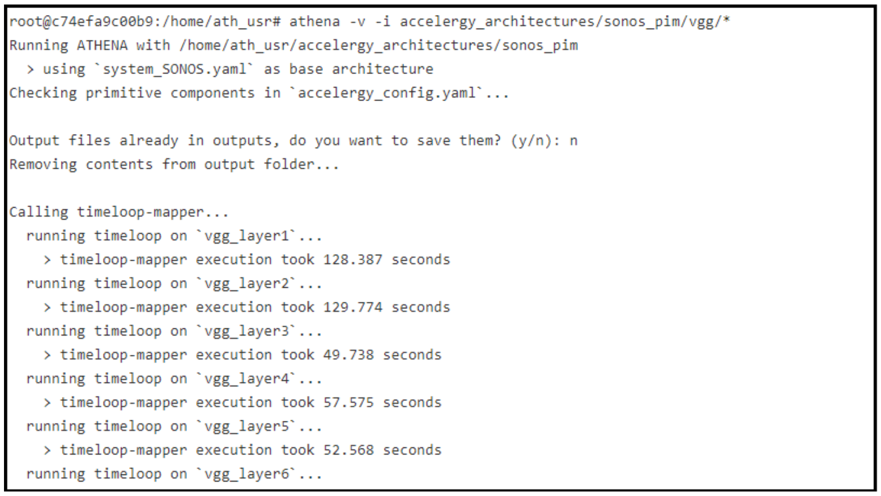


```
root@c74efa9c00b9:/home/ath_usr# athena --help
Usage: athena [OPTIONS] COMMAND [ARGS]...


      ___  ________  _________   ____\|/ ,___, |/|/
     /   |/_  __/ / / / ____/ | / /   |/ (0,0)/ |/_
    / /| | / / / /_/ / __/ /  |/ / /| |/ /)_) |/_
   / ___ |/ / / __  / /___/ /|  / ___ |\_ "" |/_
  /_/  |_/_/ /_/ /_/_____/_/ |_/_/  |_| \|_|/

  Main CLI for using Athena

  To modify architecture configuration, edit the `<arch>_config.yaml` file
  in the architecture folder. This file will define which base architecture
  is chosen, what component files are used, which mapper and constraints are
  applied, and where the outputs should be stored.

  For more information on available commands,
  run `athena <command> --help`.


Options:
  -a, --arch DIRPATH             Folder path to chosen architecture
  -i, --inputs [FILEPATH ..]     Paths of input files to analyze
  -o, --outputs DIRPATH          Redirects the outputs to this folder if
                                 specified; the folder must exist before using
                                 ATHENA

  -v, --verbose                  If verbose flag is set, then a longer summary
                                 message is printed and saved to the summary .csv
                                 file.

  --help                         Show this message and exit.

Commands:
  init      Initializes ATHENA.
  sanitize  Sanitizes files given their filepaths.
```

Figure 3-4. ATHENA Command Line Interface (CLI)

The ATHENA CLI is the main interface between the user and the ATHENA tool. It currently supports a few options for the user to choose from, such as the path to the chosen architecture (select from the architectures' folder), the input files to analyze, the output folder to store the outputs/logs (currently defaults to output), and a verbose flag that determines the verbosity of the final output message. The input files to the tool are formatted as problem shapes in a YAML file like those found in the Timeloop examples. These problem shapes define the dimensionality of the data being processed in a particular layer of a given network. Work is currently being done on a tool for a user to convert a model created using a tool such as TensorFlow or PyTorch that will convert the model to this particular format.

The ATHENA CLI is setup as a built-in command in the docker container, and requires an initialization to generate the necessary Accelergy dependencies, like an Accelergy configuration file, in order for the tool's backend to work. Once initialized, the user can run the tool directly on SONOS PIM architecture because it is the default architecture used. The user only needs to specify the verbosity and particular problem shape inputs when running ATHENA. This is shown as an example in the following figures.



```
root@c74efa9c00b9:/home/ath_usr# athena -v -i accelergy_architectures/sonos_pim/vgg/*
Running ATHENA with /home/ath_usr/accelergy_architectures/sonos_pim
  > using `system_SONOS.yaml` as base architecture
Checking primitive components in `accelergy_config.yaml`...

Output files already in outputs, do you want to save them? (y/n): n
Removing contents from output folder...

Calling timeloop-mapper...
  running timeloop on `vgg_layer1`...
    > timeloop-mapper execution took 128.387 seconds
  running timeloop on `vgg_layer2`...
    > timeloop-mapper execution took 129.774 seconds
  running timeloop on `vgg_layer3`...
    > timeloop-mapper execution took 49.738 seconds
  running timeloop on `vgg_layer4`...
    > timeloop-mapper execution took 57.575 seconds
  running timeloop on `vgg_layer5`...
    > timeloop-mapper execution took 52.568 seconds
  running timeloop on `vgg_layer6`...
```

Figure 3-5. ATHENA CLI execution

After initialization, we can use the tool to analyze multiple input files and control the verbosity of the summary statement at the end. Here, we set the verbosity flag -v and set the inputs using -i by passing all the VGG layers, as shown in Figure 3-5. The tool alerts the users of which base architecture is chosen, and checks for additional primitive components to add to the Accelergy config file. It then checks if there are outputs in the output folder (default is used here, but can otherwise be specified), and asks the user if they want to save the previously found files. ATHENA then runs the Timeloop-mapper on all inputs and prints a summary statement at the end. The summary statement is saved to a .csv file and the details of each layer's run are stored in a .pkl file. Logs for each layer are stored in /outputs/logs.

3.4. Modeling the SONOS Analog Accelerator in ATHENA

ATHENA models analog MVM array based hardware leveraging the Accelergy+Timeloop software tools. Adaptation of the analog hardware required emulating the MVM array in the context of a CMOS-based dataflow hardware acceleration device. To accomplish this, we implemented a plugin and wrapper based system around the Accelergy+Timeloop framework. An overview of the ATHENA system is shown in Figure 3-6. ATHENA acts primarily as a "wrapper" to Accelergy and Timeloop, providing a user interface entry point as well as analysis tools. Furthermore, ATHENA provides an energy estimate plugin system to Accelergy, providing energy tables that the Timeloop mapper can use to estimate analog hardware performance, shown in a high level in Figure 3-7. Combined with the wrapper functionally, ATHENA is able to coerce Timeloop into analyzing tiled analog accelerator hardware.

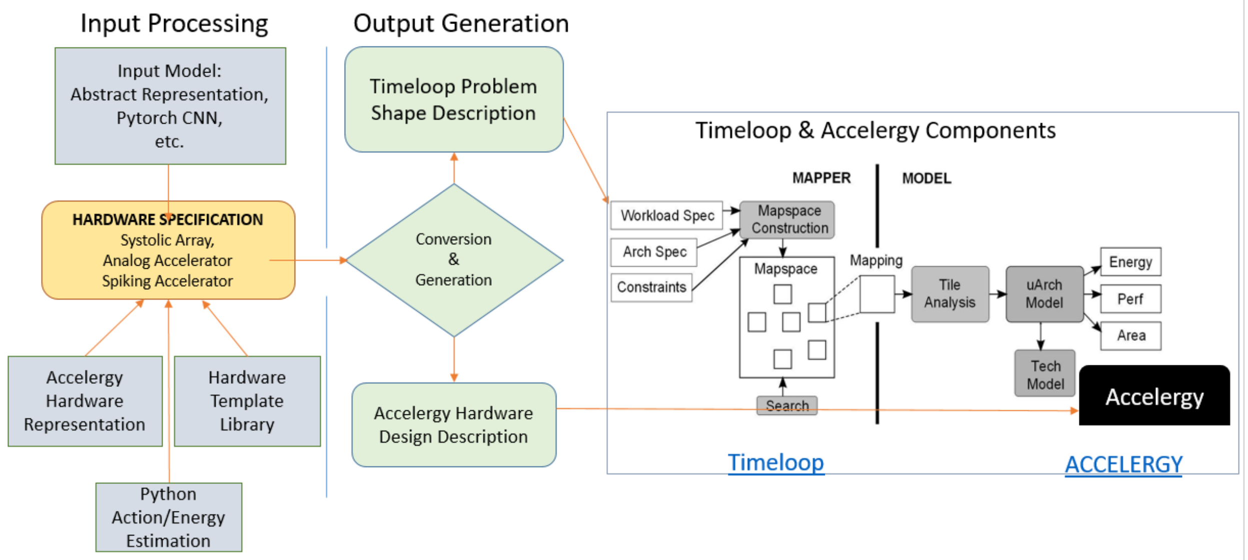

Input Processing
Output Generation
Input Model: Abstract Representation, Pytorch CNN, etc.
HARDWARE SPECIFICATION
Systolic Array, Analog Accelerator Spiking Accelerator
Accelergy Hardware Representation
Hardware Template Library
Python Action/Energy Estimation
Timeloop Problem Shape Description
Conversion & Generation
Accelergy Hardware Design Description
Timeloop & Accelergy Components
MAPPER
MODEL
Workload Spec
Arch Spec
Constraints
Mapspace Construction
Mapspace
Mapping
Search
Tile Analysis
uArch Model
Tech Model
Energy
Perf
Area
Accelergy
Timeloop
ACCELERGY

Figure 3-6. Flowchart describing ATHENA's process flow.

Adapting a dataflow-centric analytical performance estimation tool to enable analog hardware estimation required several design changes. ATHENA works as a wrapper and plugin for Timeloop and Accelergy, allowing for the mapping of multi-component SONOS hardware tiles using Timeloop. A high level overview of ATHENA is shown in Figure 3-6, detailing the program's flow from input processing, output generation, and wrapping over Timeloop and Accelergy.

Within the SONOS hardware, MVM Arrays are combined into structures called "Tiles" as shown in Figure 2-2. Athena represents tiles as "Fat PEs", given these tiles are much larger than a typical dataflow accelerator PE, wrapping energy and performance into a single logical cluster of PEs.

3.4.1. Hardware Description in ATHENA

More precisely, to provide energy and latency estimates, Timeloop computes data movement across buffer levels. Once the PE level is reached, Timeloop estimates the cycles needed to complete the computation based on the number of available PEs for computation coupled with available buffer memory.

Representing a tile within a dataflow-centric tool requires several adaptations. First, the tool must be aware of both the number of available compute units and the limits of the MVM array sizes. To represent a tile, with these constraints, a PE cluster was constructed within Timeloop. This cluster contains a set of "dummy" PEs, "dummy" buffers, along with peripheral components that make up the tile. To represent the MVM array within the cluster, a group of PEs coupled to scratchpad memory exist. The scratchpad memory connects to the ATHENA energy estimation tables, which provide data on a SONOS array's performance. The PEs exist to allow Accelergy to successfully map the input problem to hardware, however they report zero energy.

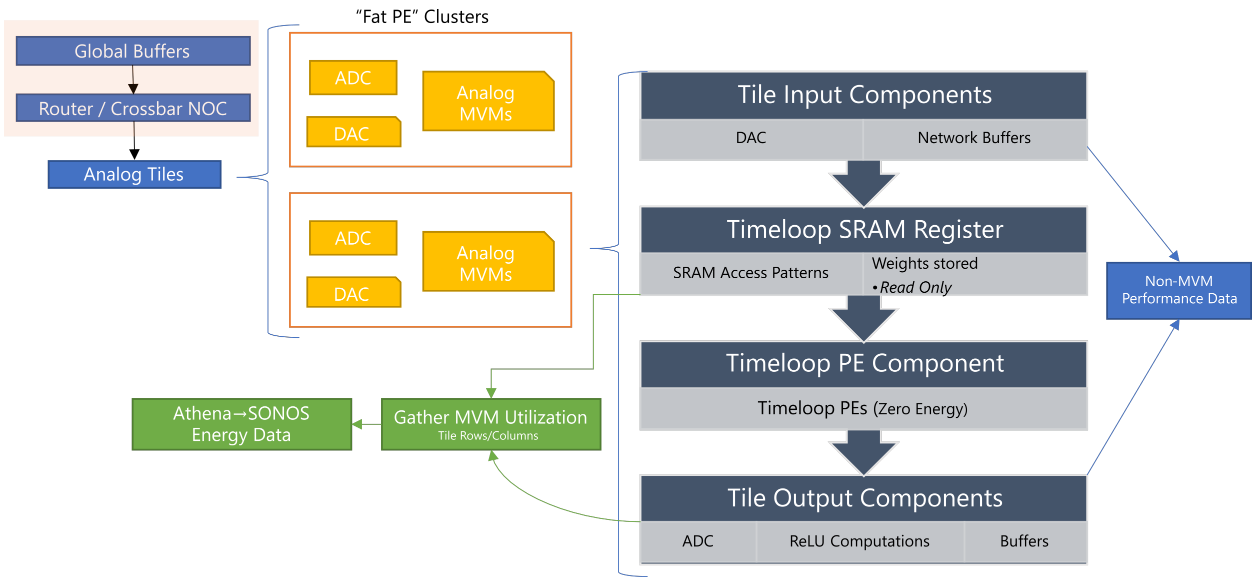


Figure 3-7. Overview of ATHENA's hardware mapping and interface to Accelergy and Timeloop.

Each group of MVM cores, in the ATHENA hardware design mapping, contains an extra "scratchpad" memory buffer. This scratchpad memory buffer serves two purposes. First, the memory represents the analog MVM array's stored weight values. The memory is configured as read-only, and can only store the weights of the input problem.

One constraint of Accelergy and Timeloop is that they are unable to dynamically change the energy required for a single MAC operation when finding a valid and optimal mapping for a given hardware configuration. However, Accelergy will look up different energy values based on memory access patterns. Athena uses these memory layers to identify and compute the active rows and columns in the SONOS array, providing energy values to Timeloop that can be incorporated into the mapping cost model.

Each tile, from the perspective of Accelergy, is a cluster of PEs. This cluster can be mapped similarly to a standard dataflow-centric hardware design. ATHENA's design takes the memory access patterns reported by Accelergy, and uses them to represent tile access energy. This technique enables Timeloop's mapping algorithm to receive more dynamic energy costs when exploring the mapspace. In standard Timeloop, energy costs are fixed at runtime; each MAC operation cost is fixed based on the hardware class and definition within the energy look up tables or calling functions.

In Accelergy, sub-trees define a set of connected components at a specific "level" within the processor structure. In the case of the SONOS definition, subtrees are defined for the MVM core with attached MVM-in and ALU-in buffers and the tile structure. Figure 3-8 is a simplified example of the hardware definition file used by Athena and Accelergy. Line 6 is the end of the non-MVM components of the device. Each tile has 4 MVM arrays, defined on line 7. Each MVM array has an SRAM buffer component, which is attached to a `sonos_access` element. This element is treated as a zero energy memory buffer by Timeloop. However, the memory access patterns are used to find energy used based on the number of active SONOS rows and columns for a particular computation.

```
1  # Fat PE simplified example - MVM array
2
3  subtree:
4        # Non MVM Components of a tile
5        # Fat PE:
6          subtree: # Virtual cluster of MVM arrays
7            - name: MVMArray[0..4]
8            - local:
9              - name: mvm_in
10               class: SRAM
11               attributes:
12                 sizeKB: 8
13             - name: sonos_access
14               class: sonos_array_pattern
15             - name: scratchpad[0..294911] # MVM Array Weights
16               class: sonos_dummy # Scratchpad containing weights
17               attributes:
18                 action_name: read
19                 network_drain: sonos_output_network
20             - name: MVM[0..294911]
21               class: compute
22               subclass: sonos_array #sonos array
23               attributes:
24                 fat: 1
25                 action_name: compute
26                 n_mvm_rows: 1152
27                 n_mvm_cols: 256
28             - name: sonos_output_network
29               class: sonos_tile_network
30             - name: ALUin
31               class: SRAM
32               network_fill: sonos_net_output
33               network_drain: alu_network
```

Figure 3-8. Simplified example of a SONOS tile definition using Accelergy's hardware definitions with ATHENA's extensions.

To allow Timeloop to map computation, the `sonos_access` element is attached to a scratchpad memory element which represents the intrinsic memory of the analog array. We restrict the data mapped to this element such that the weights or read-only portion of the input problem are stored within these values. There is one scratchpad entry per MVM cell which provides the weight table for Timeloop's mapping. Next, the MVM array is defined as a set of generic "compute" component classes. The compute class is a standard PE compute element in Timeloop. In this mapping, the ATHENA energy table reports zero energy for using this MVM array, as the inherent cost of computation is already reported by the ATHENA energy tables based on the memory access patterns via the component defined in line 13. The results of these values are sent via the on-tile network to the `ALUin` SRAM buffer, which connects to the non-MVM component network. Figure 3-7 also provides a graphical overview of the structure of the SONOS hardware mapping in the ATHENA system. This diagram illustrates where non-MVM, and thus non-ATHENA components are connected to the ATHENA based hardware.

3.4.2. Energy/Area Tables

To enable fast energy and area estimation, Athena uses energy and area lookup tables. ATHENA generates these tables before each run, providing a fast way to estimate how much energy a

```
ERT:
  version: 0.3
  tables:
  - name: system_arch.chip.tile[0..255].Core[0..3].sonos_array_pattern
    actions:
    - name: read
      arguments:
        active_cols: 0
        active_rows: 0
      energy: 4.80143808e-07
    - name: read
      arguments:
        active_cols: 1
        active_rows: 0
      energy: 4.92697728e-07
    - name: read
      arguments:
        active_cols: 2
        active_rows: 0
      energy: 5.05251648e-07
    - name: read
      arguments:
        active_cols: 3
        active_rows: 0
      energy: 5.17805568e-07
    - name: read
      arguments:
        active_cols: 4
        active_rows: 0
      energy: 5.30359488e-07
```

Figure 3-9. Small selection of an ATHENA+Accelergy ERT, with memory access patterns showing as `active_rows` and `active_columns` which provide energy values for the underlying MVM array.

particular input problem will consume. Data from this table was extracted from a simulation of the SONOS hardware, discussed in [24]. Energy provided by this simulation data contains per-array MVM energies based on the number of active rows and columns. Using this data, we generated energy reference tables for use with the ATHENA and Timeloop system.

Energy reference tables, or ERTs, consist of a set of values based on the actions performed by a particular hardware component. Each action has a corresponding value in the table, and energies are provided. The end result file is generated by Accelergy, using Athena as a plugin. A small sample of this end result data file is shown in Figure 3-9.

ATHENA uses the number of MAC operations required to compute a problem as the basis for estimating energy required for computation. Within the ERT table, as can be seen in Figure 3-9, each entry for a particular hardware element has a corresponding activity entry. Since we are adapting Timeloop's energy system to support dynamic MAC energies, MAC compute values are stored as part of the memory read operations. As MAC operations occur while running the mapper, the energy values in the ERT are added to the running total.

The Area Reference Table (ART) is similar to the ERT, in that it is also a generated lookup table. Instead of providing energy-action values it provides area estimates. As an example, Figure 3-10 shows some components of an ART. This file is generated by using Accelergy, with the ATHENA plugin providing other area estimates.

```
ART:
  version: 0.3
  tables:
  - name: system_arch.chip.tile[0..255].D2A_NoC
    area: 84.992
  - name: system_arch.chip.tile[0..255].A2D_NoC
    area: 1972.25
  - name: system_arch.chip.chip_net
    area: 181
```

Figure 3-10. Small selection of an ATHENA+Accelergy ART, with various example components and thier corresponding areas.

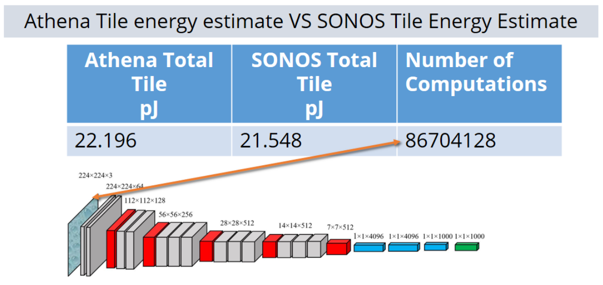
When running ATHENA, the ART is used to provide estimates of the area of the processor. This functionality has the potential to be leveraged for a design space exploration tool. The total area of the processor could be added as a constraint when finding efficient hardware designs. Currently, in Athena the ART is an informational tool. Accurate area estimations need to be gathered for specific subcomponents. Furthermore, using ATHENA as a design space exploration tool will be examined as future work.

An additional tunable element of ATHENA is an activation circuit module. In this work, we added the activation element as an element to the ALU as a tunable component. Currently, the circuit estimates a ReLU function. This ReLU uses energy values gathered from the CrossSim [1] simulation tool, or from a MUX based system described in [4, 16]. The CrossSim energy data provided values for 13 and 14 bit ReLU operations, and the MUX based system provided 16 bit ReLU operations. For values larger than these ranges, the ReLU circuit assumes that the largest component is an atomic building block, and computes how many of them are required to fulfill the needed word size.

3.5. Results

To examine ATHENA's performance, we ran two major accuracy tests. The first test involved testing a layer of the VGG-16 network and comparing the tile energy for an ATHENA based mapping versus a SONOS based mapping. In Figure 3-11, we show the network layer, ATHENA's total tile energy, and the SONOS reported total tile energy. This comparison was generated using the full dataflow problem mapping features of ATHENA, compared to the hand-tuned mapping used in the SONOS array. The results show that the techniques used in ATHENA have the potential to provide fast and close to accurate results when measuring tile energies.

Using an ERT as a lookup method has the potential to lose some dynamical behavior of the underlying hardware. This is a trade-off between accuracy and modeling speed. Look-up-tables will provide the most performance, but with potentially the least accuracy, especially when compared with lower-level simulation tools. In Table 3-1, the results of using ATHENA's method of computing energy values is compared with raw results from the SONOS simulator. These values are computed by first computing the size of the MVM arrays on each tile, modifying the MAC count with this number, then multiplying out the energy per MAC array against the number of MACs computed per array. These energy values are a total sum of just the MVM analog arrays, which have significantly smaller total energy values than the peripheral circuits on a tile.


Athena Tile energy estimate VS SONOS Tile Energy Estimate

Athena Total Tile pJ	SONOS Total Tile pJ	Number of Computations
22.196	21.548	86704128

Figure 3-11. ATHENA analytical tool results compared to the Analog FG-based model [24].

As such, these values are much more susceptible to small inaccuracies, and provide a deeper look into possible errors in ATHENA's methods of computing energy.

We found that the MVM energy array inaccuracy ranged from approximately 67 % to 2 % over all layers. A major source of the inaccuracy in these results stems from the more dynamic way that the SONOS simulation engine maps workloads across the available compute resources on-chip. The SONOS simulation engine uses a hand-tuned mapping scheme, where weight layers are duplicated to provide more efficient processing. Furthermore, the SONOS simulation uses activity over time, using clock-cycles to compute energy usage. Enabling fine-grained MVM array re-use in ATHENA is part of our ongoing work, and should dramatically increase accuracy.

Table 3-1. MVM energy estimates from ATHENA's MAC operation count method versus SONOS simulation against the VGG-16 convolutional neural network. Only convolutional layers are compared. Each energy result is from the MVM arrays in all tiles, and does not contain peripheral circuits.

Layer	Total MACs	Athena Result	SONOS Result	Difference
Conv. 1	86 704 128	2.1431 pJ	1.0652 pJ	67.1968 %
Conv. 2	1 849 688 064	8.5201 pJ	3.7520 pJ	77.7058 %
Conv. 3	924 844 032	2.1295 pJ	2.1042 pJ	1.1940 %
Conv. 4	1 849 688 064	4.0181 pJ	3.9704 pJ	1.1940 %
Conv. 5	924 844 032	1.0647 pJ	1.0395 pJ	2.3951 %
Conv. 6	1 849 688 064	2.1295 pJ	2.0791 pJ	2.3951 %
Conv. 7	1 849 688 064	2.1295 pJ	2.0791 pJ	2.3951 %
Conv. 8	924 844 032	1.0647 pJ	1.0146 pJ	4.8186 %
Conv. 9	1 849 688 064	2.1295 pJ	2.0293 pJ	4.8186 %
Conv. 10	1 849 688 064	2.1295 pJ	2.0293 pJ	4.8186 %
Conv. 11	462 422 016	0.532 37 pJ	0.482 88 pJ	9.7503 %
Conv. 12	462 422 016	0.532 37 pJ	0.482 88 pJ	9.7503 %
Conv. 13	462 422 016	0.532 37 pJ	0.482 88 pJ	9.7503 %

Another source of potential inaccuracy is the inherent differences in low-level simulation and the underlying Timeloop architecture. In [15], the accuracy of timeloop when compared against the NVDLA [14] architecture ranged from 78 % to 99 %. Those results were gathered running Timeloop against a dataflow architecture, something that Timeloop was tuned to perform well against. We expect ATHENA, in general, to have similar accuracy results against analog hardware. As we enable more advanced mapping patterns, ATHENA should reach these levels of accuracy.

Table 3-2. This table compares evaluating activation functions as part of the energy costs for different architectures.

VGG Convolutional Layer	Eyeriss Energy (pJ)	SONOS Energy (pJ)	
		No ReLU	14-Bit ReLU
Conv. 1	925.06	42.235	42.574
Conv. 2	12196.60	139.119	146.344
Conv. 3	5636.16	75.684	79.296
Conv. 4	11384.15	139.694	146.919
Conv. 5	5524.96	37.442	41.054
Conv. 6	10739.76	74.273	81.498
Conv. 7	10739.76	74.273	81.498
Conv. 8	5174.99	23.995	27.607
Conv. 9	10710.05	71.804	79.029
Conv. 10	10710.05	71.947	79.172
Conv. 11	3015.38	13.763	15.569
Conv. 12	3015.38	13.617	15.423
Conv. 13	3015.38	13.617	15.423

We further investigated the effect of the ReLU components on the total energy of the SONOS system. Using ATHENA and plain Timeloop + Accelergy, we computed the energy cost of running VGG-16 convolutional layers on the Eyeriss digital system against SONOS with and without a 14 bit ReLU. In Table 3-2, the results of these evaluations are shown. We found that a ReLU activation circuit adds a non-insignificant effect on the energy of running these layers. Table 3-2 also compares the reported performance of Eyeriss against the SONOS estimations. We ran this hardware design using Timeloop + Accelergy to examine the effect of the ATHENA plugin, wrappers, and modifications on non-analog device accuracy. As the table shows, the modifications to Timeloop and Accelergy did not change the underlying digital estimation performance.

4. ATHENA-SST INTEGRATION TOOL (ASIT)

ASIT allows for the user to pass in a specific input problem, to be tested against different hardware architectures. ASIT identifies the series of operations needed to run these problems, then builds a set of possible compatible hardware architectures that could theoretically execute the problem set. ASIT then utilizes ATHENA in order to evaluate the performance execution of the input problem set over the specified hardware.

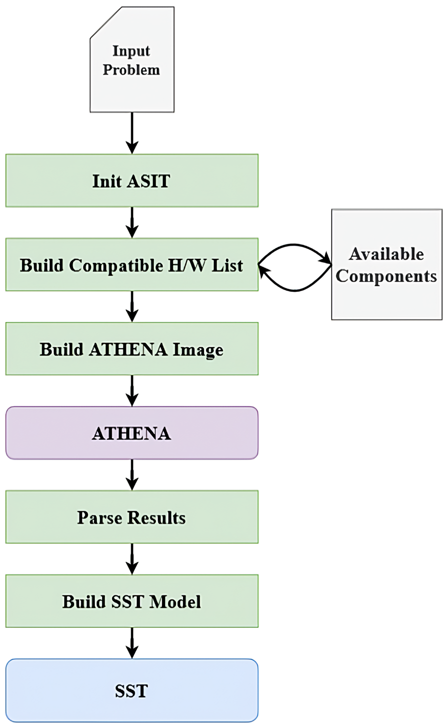
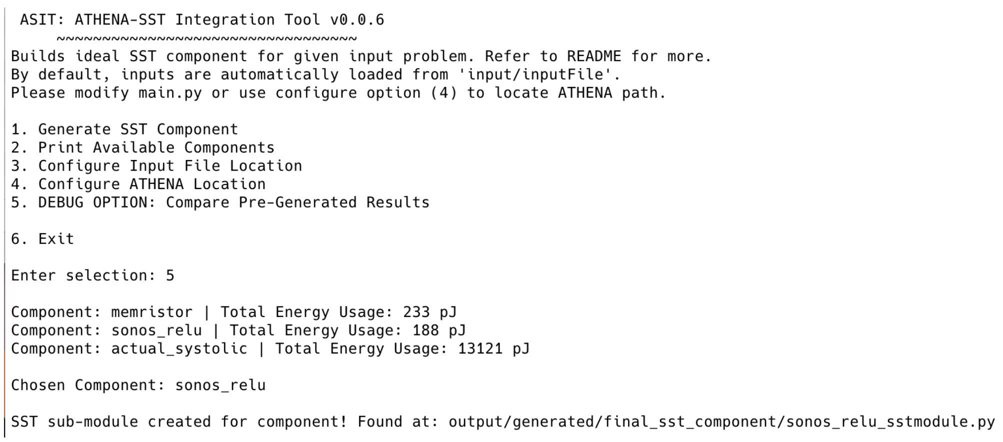


Figure 4-1. ASIT Functional Diagram

ATHENA is then used to evaluate the candidate hardware’s performance based on the given input. Currently, ASIT supports selecting the hardware with the lowest estimated total energy, however, it is possible to add other optimization targets including latency, area, and cost of the devices. ASIT is designed to take the selected “best” hardware configuration, and generate an SST component based on the hardware. This is implemented via a set of configuration parameters which define the component’s capabilities in terms of compute size, memory capacity, and other values. In the case of the SONOS system the parameters are tile size, number of tiles, and cache sizes. As ATHENA is estimating novel analog hardware performance, the design of SST components to support these devices is critical to increase supported hardware in SST. A supported component must have a set of configurable parameters which ASIT can populate based on the ATHENA hardware design. To increase the number of supported hardware devices, more

SST components need to be added, along with a corresponding set of ATHENA hardware and ASIT output configuration parameters.

4.1. Results



```
 ASIT: ATHENA-SST Integration Tool v0.0.6
     ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Builds ideal SST component for given input problem. Refer to README for more.
By default, inputs are automatically loaded from 'input/inputFile'.
Please modify main.py or use configure option (4) to locate ATHENA path.

1. Generate SST Component
2. Print Available Components
3. Configure Input File Location
4. Configure ATHENA Location
5. DEBUG OPTION: Compare Pre-Generated Results

6. Exit

Enter selection: 5

Component: memristor | Total Energy Usage: 233 pJ
Component: sonos_relu | Total Energy Usage: 188 pJ
Component: actual_systolic | Total Energy Usage: 13121 pJ

Chosen Component: sonos_relu

SST sub-module created for component! Found at: output/generated/final_sst_component/sonos_relu_sstmodule.py
```

Figure 4-2. Example output of the ASIT tool

We evaluated ASIT on three different hardware configurations: A SONOS-based analog accelerator, a memristor crossbar accelerator, and a digital systolic accelerator configuration which were then presented to ATHENA to evaluate over. ATHENA then evaluated two layers of the VGG network previously examined to find performance values. After evaluation the ASIT tool examined the devices, picked the SONOS accelerator as the best performance option, and generated a set of SST component configuration settings. Figure 4-2 shows the result of the run as a demonstration of the software interface.

4.2. Extending ASIT

ASIT is intended to provide a link between an analytical performance estimation tool like ATHENA, and the SST simulation engine. In the future, ASIT will be able to generate an SST model based on more primitive SST component templates, allowing for full simulation of a variety of hardware. ASIT will allow for integration of any device supported both by ATHENA and SST template enabled components, specifically targeting architectures including: tile-based digital components, analog accelerators, and neuromorphic accelerators. With the addition of these hardware components to ATHENA and SST, ASIT will allow for the evaluation of efficiency of these hardware designs before implementing them into SST. This would allow for an array of different architectures to be filtered through so the best-fit option is the one chosen to be fully simulated.

Eventually, ASIT will enable rapid prototyping of analog devices for specific applications with a higher fidelity simulation step in the loop. This could pave the way to a true DSE software system for analog and analog-based neuromorphic hardware.

5. CONCLUSION

We delivered an initial API for Athena in a docker environment for an analog machine learning accelerator. The ATHENA framework is now set up to enable design space exploration of different accelerators that leverage emerging devices. Future work in ATHENA will support a wider array of devices and architectures. As shown in Figure 5-1, the overarching goal is to support and develop tools as part of the codesign tool ecosystem at Sandia, thereby impacting multiple mission areas.

Future work on the ATHENA tool will be to add a hardware design library that would enable a ‘lego’ style method of testing analog hardware design. We also plan to generate candidate hardware configurations by multi-objective optimization to reduce the analog search space. We will also continue design space exploration with SST in the loop, leveraging ASIT.

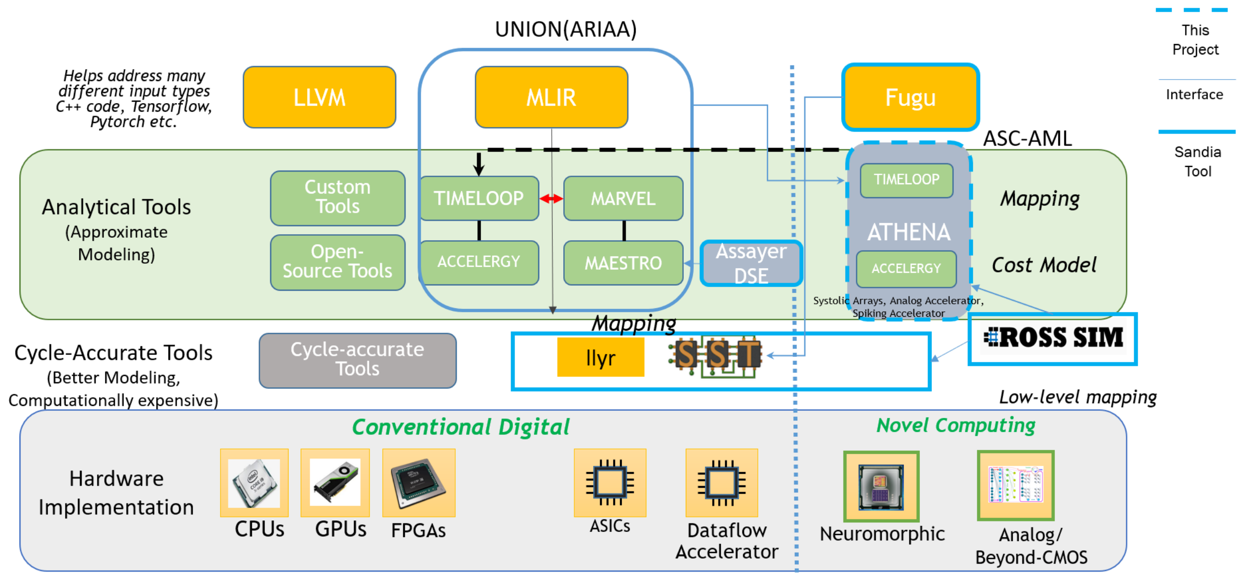


Figure 5-1. A snapshot of codesign tool ecosystem for machine learning with a focus on projects at Sandia.

ATHENA-SST integration was especially important to be able to do design space exploration for many different accelerator backends. We demonstrated an initial example of integration of ATHENA with SST, to enable approximate modeling of performance before detailed simulation in SST.

As shown in Figure 5-1, there are a number of different projects at Sandia that target different areas of the stack ranging from analytical tools that achieve approximate performance modeling to cycle-accurate and hardware accelerated simulations and many different low-level backends ranging from conventional computing (CPU, GPU, FPGAs) to leveraging dataflow accelerators, neuromorphic and beyond-CMOS computing. Our future in particular will focus on heterogeneous novel computing paradigms like neuromorphic and analog computing and contributing to the codesign tool ecosystems.

The two tracks for mapping and architectural exploration are complimentary and critical in our co-design methodology to design heterogeneous architectures that incorporate novel computing paradigms. The outcome of this work are the first steps toward a formalized design stack methodology for mission-oriented hardware/software co-design for heterogeneous node and system architectures. This expanded computation space requires not only a new approach to application development, but the ability to identify, evaluate, design, and analyze next-generation architectures specialized for specific workloads.

BIBLIOGRAPHY

[1] Sapan Agarwal, Alexander Hsia, Robin Jacobs-Gedrim, David R Hughart, Steven J Plimpton, Conrad D James, and Matthew J Marinella. Designing an analog crossbar based neuromorphic accelerator. In *2017 Fifth Berkeley Symposium on Energy Efficient Electronic Systems & Steep Transistors Workshop (E3S)*, pages 1–3. IEEE, 2017.

[2] James Bradley Aimone, Christopher H Bennett, Suma George Cardwell, Ryan Anthony Dellana, and Patrick Xiao. Mosaic, the best of both worlds: Analog devices with digital spiking communication to build a hybrid neural network accelerator. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2020.

[3] Aayush Ankit, Izzat El Hajj, Sai Rahul Chalamalasetti, Geoffrey Ndu, Martin Foltin, R Stanley Williams, Paolo Faraboschi, Wen-mei W Hwu, John Paul Strachan, Kaushik Roy, et al. Puma: A programmable ultra-efficient memristor-based accelerator for machine learning inference. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 715–731, 2019.

[4] Christopher H. Bennett, Vivek Parmar, Laurie E. Calvet, Jacques-Olivier Klein, Manan Suri, Matthew J. Marinella, and Damien Querlioz. Contrasting advantages of learning with random weights and backpropagation in non-volatile memory neural networks. *IEEE Access*, 7:73938–73953, 2019.

[5] Suma George Cardwell, Craig Vineyard, Willam Severa, Frances S Chance, Frederick Rothganger, Felix Wang, Srideep Musuvathy, Corinne Teeter, and James B Aimone. Truly heterogeneous hpc: Co-design to achieve what science needs from hpc. In *Smoky Mountains Computational Sciences and Engineering Conference*, pages 349–365. Springer, 2020.

[6] Yu-Hsin Chen, Tien-Ju Yang, Joel Emer, and Vivienne Sze. Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 9(2):292–308, 2019.

[7] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, et al. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

[8] Jeff Dean, David Patterson, and Cliff Young. A new golden age in computer architecture: Empowering the machine-learning revolution. *IEEE Micro*, 38(2):21–29, 2018.

[9] Ben Feinberg, Sapan Agarwal, Mark Plagge, Fredrick H Rothganger, Suma Cardwell, and Clayton Hughes. Modeling Analog Tile-Based Accelerators Using SST. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2022.

[10] Steve B Furber, Francesco Galluppi, Steve Temple, and Luis A Plana. The spinnaker project. *Proceedings of the IEEE*, 102(5):652–665, 2014.

[11] Hyoukjun Kwon, Michael Pellauer, and Tushar Krishna. Maestro: an open-source infrastructure for modeling dataflows within deep learning accelerators. *arXiv preprint arXiv:1805.02566*, 2018.

[12] Hyoukjun Kwon, Ananda Samajdar, and Tushar Krishna. MAERI: Enabling flexible dataflow mapping over DNN accelerators via reconfigurable interconnects. *SIGPLAN Not.*, 53(2):461–475, March 2018.

[13] Paul A Merolla, John V Arthur, Rodrigo Alvarez-Icaza, Andrew S Cassidy, Jun Sawada, Filipp Akopyan, Bryan L Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[14] NVDLA, 2020.

[15] Angshuman Parashar, Priyanka Raina, Yakun Sophia Shao, Yu-Hsin Chen, Victor A Ying, Anurag Mukkara, Rangharajan Venkatesan, Brucek Khailany, Stephen W Keckler, and Joel Emer. Timeloop: A systematic approach to dnn accelerator evaluation. In *2019 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 304–315. IEEE, 2019.

[16] Vivek Parmar and Manan Suri. Design exploration of iot centric neural inference accelerators. In *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, GLSVLSI '18, page 391–396, New York, NY, USA, 2018. Association for Computing Machinery.

[17] Mark Plagge, Christopher D Carothers, Elsa Gonsiorowski, and Neil Mcglohon. Nemo: A massively parallel discrete-event simulation model for neuromorphic architectures. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 28(4):1–25, 2018.

[18] Steven J Plimpton, Sapan Agarwal, Richard Schiek, and Isaac Richter. Crosssim. Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2016.

[19] Steven J Plimpton, Sapan Agarwal, Richard Schiek, and Isaac Richter. Crosssim. Technical report, 2016.

[20] A. F. Rodrigues, K. S. Hemmert, B. W. Barrett, C. Kersey, R. Oldfield, M. Weston, R. Risen, J. Cook, P. Rosenfeld, E. Cooper-Balis, and B. Jacob. The structural simulation toolkit. *SIGMETRICS Perform. Eval. Rev.*, 38(4):37–42, March 2011.

[21] Fredrick H Rothganger and Arun F Rodrigues. Generic spiking architecture (gensa). Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2020.

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

[23] Yannan Nellie Wu, Joel S Emer, and Vivienne Sze. Accelergy: An architecture-level energy estimation methodology for accelerator designs. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 1–8. IEEE, 2019.

[24] T Patrick Xiao, Ben Feinberg, Christopher H Bennett, Vineet Agrawal, Prashant Saxena, Venkatraman Prabhakar, Krishnaswamy Ramkumar, Harsha Medu, Vijay Raghavan, Ramesh Chettuvetty, et al. 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, 2022.

DISTRIBUTION

Hardcopy—Internal

Number of Copies	Name	Org.	Mailstop
1	Clay Hughes	01422	1318
1	Ron Oldfield	01441	1327
1	John S. Wagner	01421	1327
1	Suma G. Cardwell	01421	1327

Email—Internal (encrypt for OUO)

Name	Org.	Sandia Email Address
Technical Library	01911	sanddocs@sandia.gov