SAND2017-9467C

A Spike-Timing Neuromorphic Architecture

Aaron J. Hill*, Jonathon W. Donaldson*, Fredrick H. Rothganger*, Craig M. Vineyard*, David R. Follett†, Pamela L. Follett†‡, Michael R. Smith*, Stephen J. Verzi*, William Severa*, Felix Wang*, James B. Aimone*, John H. Naegle* and Conrad D. James*
*Sandia National Laboratories, Albuquerque, NM 87185 USA
Email:{ajhill, jwdonal, frothga, cmviney, msmith4, sjverzi, wmsever, felwang, jbaimon jhnaegl, cdjame}@sandia.gov
†Lewis Rhodes Labs, Concord, MA 01742 USA
Email:{drfollett, plfollett}@earthlink.net
‡Tufts University, Medford, MA 02155 USA

***Abstract*—Unlike general purpose computer architectures that are comprised of complex processor cores and sequential computation, the brain is innately parallel and contains highly complex connections between computational units (neurons). Key to the architecture of the brain is a functionality enabled by the combined effect of spiking communication and sparse connectivity with unique variable efficacies and temporal latencies. Utilizing these neuroscience principles, we have developed the Spiking Temporal Processing Unit (STPU) architecture which is well-suited for areas such as pattern recognition and natural language processing. In this paper, we formally describe the STPU, implement the STPU on a field programmable gate array, and show measured performance data.**

I. Introduction

Neural-inspired and neuromorphic computing systems have been developed to address data-driven computing challenges, such as pattern matching, largely due to the fact that such problems can not be addressed with explicit numerical programming [1]. In addition, there is a growing consensus that neural approaches will provide significant advantages in power consumption [2]–[4]. This increased interest in neuromorphic computing as a potential source of post-Moore's Law technologies has led to a number of proposed neural architectures that achieve performance benefits against conventional technologies [1].

There are an increasing number of spike-based neuromorphic platforms, but each has had to enforce an independent set of design choices that deviates from the original neural inspiration. One obvious difference from the brain is the lack of on-chip learning, which has increasingly become a focus of technology development [5]. However, two additional aspects of neurobiological systems have continued to be a challenge for efficient hardware implementations that are arguably just as impactful. First, biological neural circuits have a high degree of interconnectivity that is difficult to replicate in two-dimensional hardware. Cortical neurons in mammalian systems often receive over 10^4 synaptic inputs, which stands in stark contrast to most hardware platforms that are either limited to or are only efficient at a fan-in of around 10^2 inputs. Second, biological neural circuits use spike timing, including axonal and dendritic delays, to communicate information [6], [7]. Computationally realizing biological learning processes, ranging from the synaptic (e.g., spike-timing dependent plasticity [8]) to the structural (e.g., adult neurogenesis [7]), requires more complex temporal representations than most neuromorphic hardware currently supports.

Increasingly, there are spiking neural algorithms (SNAs) that take advantage of relative timing differences and utilize more biologically realistic fan-in and fan-out. One of the first spiking machine learning algorithms, the liquid state machine (LSM), relies on exponential-decay synapses that temporally expand the influence of one neuron on another, thus stabilizing the computation [9]. More recently, we have proposed a number of SNAs that utilize spike timing to perform increasingly complex numerical operations such as cross-correlations and optimization [10], [11].

Analogous to the fact that there is no single classifier suitable for all classification tasks [12], there is no single architecture that is optimal for every possible computational task. We recently reported on the design of the Spiking Temporal Processing Unit (STPU) architecture and used a simulation tool to assess the capability of this architecture to process speech data [13]. Here, we formally describe and demonstrate the STPU architecture for implementing SNAs in a field programmable gate array (FPGA) with associated performance measurements such as joules per synaptic event and synaptic events per second.

In Section II we detail the STPU architecture and provide a formal description of the neuron model. In Section III we define the programming interface and in Section IV we report on the FPGA implementation of the STPU architecture with power and computation measurements.

II. STPU Architecture

A. Motivation

In neurobiological systems, each neuron has its own local memory in the form of synaptic connectivity which defines its functionality. The functionality of the larger ensemble of neurons is defined by the combined effect of the specific sparse connectivity, the efficacy of the connectivity, and the latencies imposed by the locality of connections [14], [15].

Aside from a few notable exceptions such as within the retina, one of the differentiating characteristics of neurobiological systems from artificial neural networks used in machine learning is that neurons communicate using action potentials or

"spikes". A spike represents information in the time domain; the time at which a spike occurs contains the critical information while the amplitude of the spike contains no information. In a digital system, a spike event can be represented by a single bit. This presents substantial advantages in energy efficiency for large parallel systems because, at a minimum, only a single bit needs to be communicated for each spike. This potential advantage of spiking has motivated a number of neural computing architectures such as IBM's TrueNorth [16] and University of Manchester's SpiNNaker processor [17]. These architectures and others have leveraged sparse spike-based communication to achieve low energy pattern recognition [4] as well as improved scalability and performance speed-up [18].

While spiking is mostly an agreed upon representation by those interested in low-power hardware, the neuromorphic computing community has focused primarily on how best to implement spiking [19] and communicate spikes at realistic scales [20]. However, it is unclear whether biophysically realistic spiking mechanisms [21], [22] provide a computational advantage over simple discrete spiking models [4], [18]. Our architectural approach leverages a simple and discrete neural model that prioritizes the temporal (precise spike timings and delays) and spatial (robust, generic connectivity) complexity observed within spiking neural systems.

B. Architecture Overview

The STPU is a scalable and highly parallel spike-based neuromorphic architecture with a core processing unit that supports high fidelity spike timing dynamics. The core design principle of the STPU is to support fast and efficient movement of information for computing Spiking Neural Networks. The STPU is very flexible to specifications of neuron connectivity and synaptic delay.

Several algorithms exist which could leverage the STPU's flexibility. A temporal-coded SNA was developed to compute the argument of the maximum cross-correlations [10]—a computation closely related to the concept of matched filters. Specifically, the temporal-coded algorithm uses graduated synaptic delays, which utilizes the temporal capabilities of the STPU. We previously simulated an LSM for spoken digit recognition [13] and examined different algorithms for training the readout neurons, various liquid topologies and different firing thresholds for the LIF neurons. While using arbitrarily complex synaptic response functions to model synaptic decay, greater than 90% classification accuracy was achieved using both support vector machines and linear discriminant analysis. Additionally, the STPU was used to provide support for fundamental neural-inspired spiking algorithms for simple optimization [11]. Utilizing the STPU's temporal characteristics, each of these algorithms use some form of temporal coding.

The architectural flow of the STPU is depicted in Figure 1. The Spike Transfer Structure (STS) is responsible for routing spikes through the network structure, managing external input data, performing all synaptic computations and executing the program instructions. Recurrent spike routing is supported by the Output Spike Consolidator (OSC).

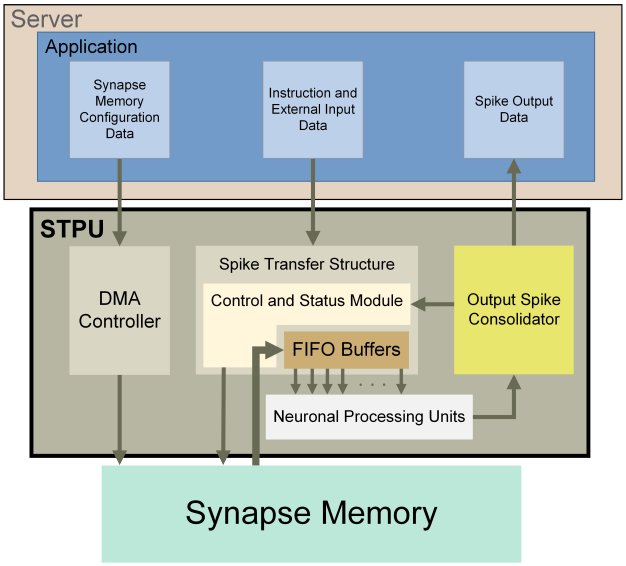

Fig. 1. The Spiking Temporal Processing Unit architectural flow: synaptic information for each neuron is stored in a large off chip memory known as the Synapse Memory. The Neuronal Processing Units interface to the Synapse Memory through the Spike Transfer Structure. Spikes generated in the Neuronal Processing Units are routed to the Spike Transfer Structure through the Output Spike Consolidator.

The Neuronal Processing Units (NPUs) draw two key inspirations from neuroscience: the leaky integrate and fire (LIF) neuron model to facilitate spike generation and a dendritic arbor model to facilitate delayed spike delivery. The LIF model [23] is a simple neuron model that captures several core features of biological neurons. This model accumulates input stimuli received by the neuron. If this accumulated state crosses a defined firing threshold the neuron will "fire". A neuron's fire event transmits a spike to all of the neurons it connects to and resets its accumulated state to a predefined rest state. If the neuron does not fire, the accumulated state from past input stimuli will slowly decrease in an effort to return the accumulated state value back to the resting state. In summary, the neuron performs a leaky integration of input stimuli and if the value of the integrator crosses its firing threshold, the neuron will spike resulting in an integrator reset.

The LIF model is common practice in spiking neural networks, but the dendritic arbor model is often overlooked. The dendritic arbor model defines one aspect of the time delays associated with each post synaptic connection to a neuron. To account for this timing characteristic, each STPU neuron incorporates a temporal buffer. The temporal buffer allows for pre-synaptic neurons to make weighted connections with associated delays. The behavioral dynamics of the temporal buffer can be likened to that of a shift register. Values in the temporal buffer are shifted from top to bottom, where only the bottom value is shifted into the integrator of the LIF neuron. In this sense, the temporal buffer of a single neuron is similar to a compartment and cable model [23] in which the inputs must propagate through the temporal buffer to reach the soma rather than instantaneously arriving at the integrator. The temporal buffer allows for the delayed arrival of

input spike information from other neurons. This is analogous to the notion that synaptic connections within a neuron's dendritic arbor are not uniformly equidistant from the soma. Rather, some connections are shorter than others and affect the post-synaptic neuron sooner than those that are further away. The temporal buffer of each LIF neuron provides the innate ability to perform temporal processing and implement complex synaptic response functions [13].

In many respects, neuromorphic computing architectures are still tied to conventional computing paradigms for data input. This notion is represented in Figure 1 by the Server/Application block. The network configuration information is written to the Synapse Memory (SM) via a Direct Memory Access (DMA) controller. The application supplies the STPU with instructions and external data it may need for computation through a high speed data bus and provides a means to receive any output spike information from the NPUs.

The following sections will provide in depth coverage of the STPU architecture by detailing the SM, the STS, the NPUs, and the OSC.

C. Synapse Memory

The SM is responsible for holding all the synaptic information necessary to perform each synaptic calculation for every spike event. The information stored in this memory includes the pre-synaptic neuron designator, the synaptic weight, the post-synaptic neuron designator, and the synaptic delay associated with the connection. The organization of the SM is done in a way that optimizes access of the data and its respective storage allocation.

The SM allows the STPU to support a fully connected network of neurons and allows for a unique synaptic weight and delay specification for every connection. It also allows any neuron to make multiple connections to a downstream neuron with different weights and time delays. Additionally, the SM can be updated at any point during execution through the DMA controller allowing for adjustments to the neuronal connectivity, synaptic weights, and synaptic delays on the fly.

D. Spike Transfer Structure

The STS's data flow is depicted in Figure 2. The OSC delivers to the STS all spikes generated in the previous time step. These internally routed spikes carry with them the neuron that spiked and the relative spike-time so that any synaptic delays applied to the spike event are applied relative to the time the spike occurred. This notion is critical in applying accurate spike timing dynamics independent of clock frequency. External input data is delivered to the STS from the server-side application and can be in the form of spikes or real-valued input. Because external input data is not generated relative to any spike timing event internal to the STPU, they carry a different temporal parameter known as the temporal shift. This allows external input data to have relative time delays associated to them and not be required to affect the system at the exact time the input data is applied.

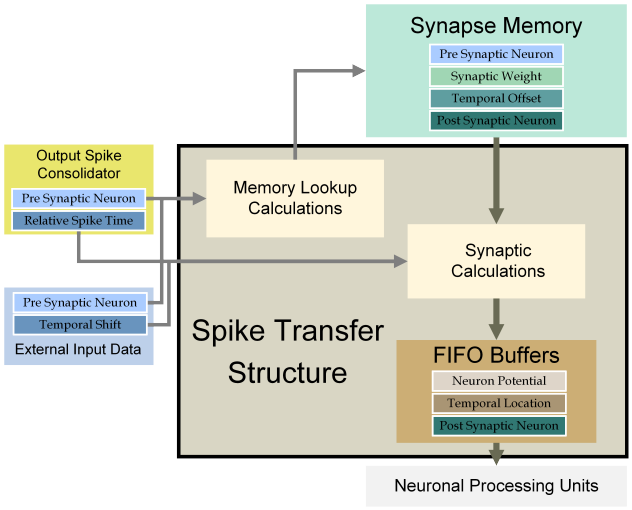

Fig. 2. Spike Transfer Structure: the core engine of the Spiking Temporal Processing Unit architecture responsible for receiving all input data (internal spike data, external input data, and execution instructions), performing all synaptic calculations for the network, and routing the information to the Neuronal Processing Units. The Spike Transfer Structure directly interfaces with the instruction and input data from the server-side application, the Output Spike Consolidator, Synapse Memory, and Neuronal Processing Units.

Access to the SM is performed over a high speed and highly parallel data communication interface. The STS uses the pre-synaptic neuron information from the input data to calculate where in memory the synapse information is stored. The STS fetches the synapse information needed to perform all synaptic calculations which includes the synaptic weight, the temporal offset and the post-synaptic neuron. These calculations are performed in parallel where the number of parallel computation paths is dependent on the size of the parallel data bus to the SM. The first calculation performed is the input potential induced by the spike or external input. The synaptic weight is multiplied by the input value to produce the input potential that will affect the post-synaptic neuron (in the case of a spike, the input value is one). The second calculation performed is the spike delivery delay which determines where in the temporal buffer of the post-synaptic neuron the calculated input potential will be placed. The STS uses the temporal information from the input data by adding it to the temporal offset value from the SM to produce the exact location in the temporal buffer to place the input potential of the post-synaptic neuron. This location cannot be larger than the physical number of temporal buffer indexes. Mathematical formulation of these computational details will be covered in the next section (Section II-E).

The STS delivers to the NPUs the post-synaptic neuron to target, the location in the temporal buffer and the potential induced by the input (input potential) into the post-synaptic neuron.

E. Neuronal Processing Unit

The NPUs contain all the computational dynamics of the LIF neuron model, the dendritic arbor model, and a small set of configurable parameters for each neuron. The STPU has

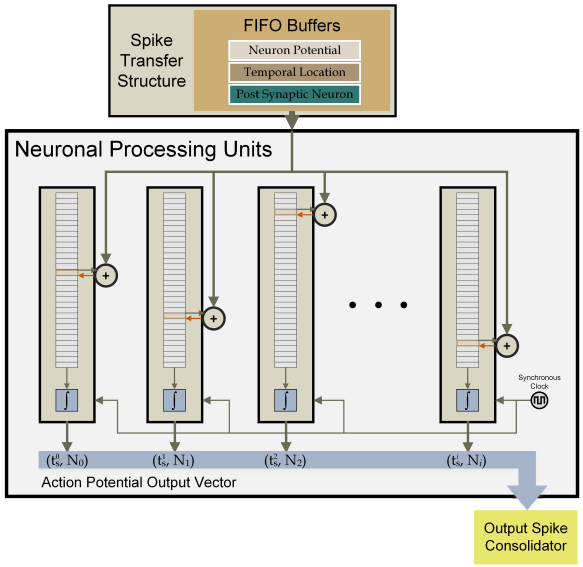

Fig. 3. Neuronal Processing Units: post-synaptic neuron information from the Spike Transfer Structure identifies which neuron to target for a temporal buffer update. The temporal location data identifies where in the temporal buffer to place the input potential data. The input potential is inserted into the temporal buffer by adding it to the current value in the temporal buffer at the designated temporal location. This effectively superimposes spike information that will arrive to the neuron at the same time so that no spike information is lost.

configurable parameters per neuron for the firing threshold, the minimum neuron potential, and the leakage value used to compute leak. A limited set of parameters was chosen to maximize neuron count by reducing computational logic and parameter storage.

The STPU assigns subsets of neurons of equal size to each parallel computation path from the STS. For simplicity, only a single computation path is depicted in Figure 3. All the STPU neurons operate on a synchronous clock. One neural computation cycle involves the following operations. First, all temporal buffers are updated with the information from the STS. Then, the bottom value of the temporal buffer is added into the accumulator. The neurons then check their accumulator values against their respective firing thresholds. If the accumulator value (neuron potential) is below the firing threshold the accumulator value will have a multiplicative leak operation performed on it. If the accumulator value is equal to or greater than the firing threshold then the neuron will spike and reset the accumulator to its resting state. Each neuron that spikes will package its neuron designator value and the time that the neuron spiked and deliver this information to the OSC. We refer to this process as one LIF operation (LIFop) which is composed of the update, shift, integrate, check, and spike/reset or leak sub-operations.

More formally we define the neuron computation cycle through Figure 4. Let D be the number of registers in the temporal buffer and N be the number of neurons in the STPU system. Let $\mathbf{R}$ be a $D \times N$ matrix where each column in $\mathbf{R}$ represents the temporal buffer for neuron j and is indexed by $R_{d,j}$ with $d \in \{0, \cdots, D-1\}$. Synaptic connection weights from input neuron k to destination neuron j at the d^{th} index of R_j is denoted as $W_{d,j,k}$. Therefore, $\mathbf{W}$ is a $D \times N \times N$ rank-3 tensor. Internally recurrent spikes are denoted by the output vector O and external input data is denoted by the input vector E. The integration register of the j^{th} neuron is denoted by A_j, where A is an $N \times 1$ vector of integration registers. The leak function performed on the integration register is denoted as $\Lambda(A_j)$. Note that $\hat{\mathbf{R}}$ and $\bar{A}$ are intermediary states for the temporal buffer and the integration register, where $\hat{\mathbf{R}}$ defines the temporal buffer after an update of the temporal buffer and prior to a shift and integrate operation, and $\bar{A}$ defines the integration register after the shift and integration but prior to the threshold check and leak sub-operations. Equations 1, 2 and 3 below define the computation model of the STPU neuron.

SYNAPTIC INPUT UPDATE

$$\hat{R}_{d,j}(t) = R_{d,j}(t) + \sum_k i_k(t) W_{d,j,k}(t) \tag{1}$$

TEMPORAL INTEGRATION

$$\begin{aligned} R_{\bar{d},j}(t+1) &= \hat{R}_{\bar{d}+1,j}(t) \\ R_{D-1,j}(t+1) &= 0 \\ \bar{A}_j &= A_j(t) + \hat{R}_{0,j}(t) \end{aligned} \tag{2}$$

where $\bar{d} \in \{0, \cdots, D-2\}$.

THRESHOLD, FIRE, RESET

$$\begin{aligned} A_j(t+1) &= \begin{cases} \Lambda(\bar{A}_j) & \text{if } \bar{A}_j < T_j \\ 0 & \text{if } \bar{A}_j \geq T_j \rightarrow \text{SPIKE} \end{cases} \\ \Lambda(\bar{A}_j) &= \bar{A}_j(1-\lambda) \end{aligned} \tag{3}$$

where $A_j(0) = 0$. Note that $A_j(t+1)$ is the value of the integration register after a complete LIFop has been performed, T_j is the threshold to fire value of the j^{th} neuron, and λ is the leakage value.

F. Output Spike Consolidator

Spike routing is a necessary function in all neuromorphic architectures. TrueNorth implemented their own routing dynamics for on chip and off chip spike routing [16] while others use address event representation (AER). There is also hierarchical address event representation (HiAER) which is a multiscale tree-based extension of single-bus AER that offers scalable throughput without restrictions on spatial range [24]. The STPU implements a simplified hierarchical spike routing mechanism termed the OSC. An example of a three-stage OSC is illustrated in Figure 5.

Stage one is directly coupled to a subset of the neurons of the system. For efficiency each stage one consolidator is allocated the same number of neurons. The stage one consolidators operate in parallel with each other compactly placing any spike data from its associated neurons into an output buffer. These output buffers are serviced by the stage two consolidation node. In this illustration only a pair of

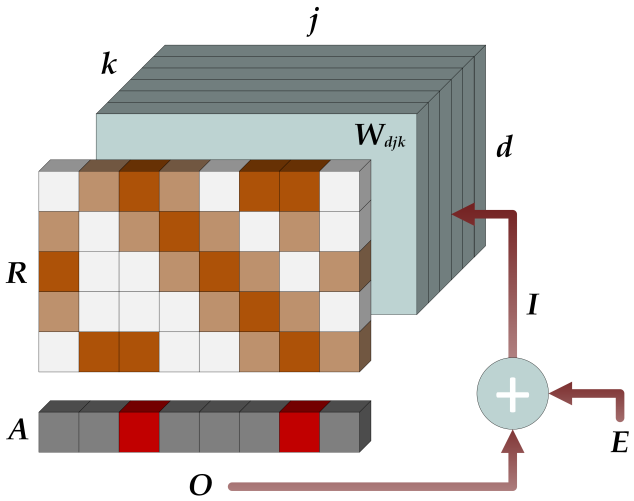

Fig. 4. Neuron computation model: the output spikes vector (O) are combined with external input data vector (E) to form an input data vector (I) that is applied to the rank-3 tensor ($\mathbf{W}$) which defines the connection parameters for synaptic weight and delay. The resulting computation of $I \times \mathbf{W}$ (I is iteratively multiplied with the 2D matrices formed by the k^{th} dimension of $\mathbf{W}$) is added element-wise with the existing values of $\mathbf{R}$. Which is then accumulated column-wise to produce A. Elements of A are checked against the threshold to fire to produce the output vector (O).

stage two consolidation nodes are depicted each servicing eight stage one consolidation nodes. Each stage two node operates in parallel taking data from the stage one output buffers and compactly placing it into a single stage two output buffer. These output buffers are serviced by the stage three consolidation node.

The stage three consolidation node operates in much the same way as the stage two nodes by servicing each stage two output buffer and compactly placing the spike information into a final stage three output buffer. This final buffer is serviced by the STS to pull data in and complete the computation loop of the STPU. All consolidation stages work with pipelined efficiencies so that as soon as data arrives into a buffer the downstream stage will begin consolidating it into its own output buffer.

It is important to note that Figure 5 is an implementation example. The STPU architecture supports any number of staged consolidation nodes and any number of consolidation nodes attached to downstream nodes. Increasing the number of stages or number of nodes attached per stage will increase parallelization and performance at the cost of hardware resources.

G. STPU Computation Modes

The STPU supports two modes of computation: streaming mode and wave mode. Streaming mode supports tightly coupled recurrent networks and high frequency input data and output data. In streaming mode, every LIFop requires all generated spikes to be routed to the STS before the next LIFop is performed and can accept input data into the STS before each LIFop begins.

In contrast to streaming mode, wave mode is a more efficient form of computation for some algorithms. In wave mode one

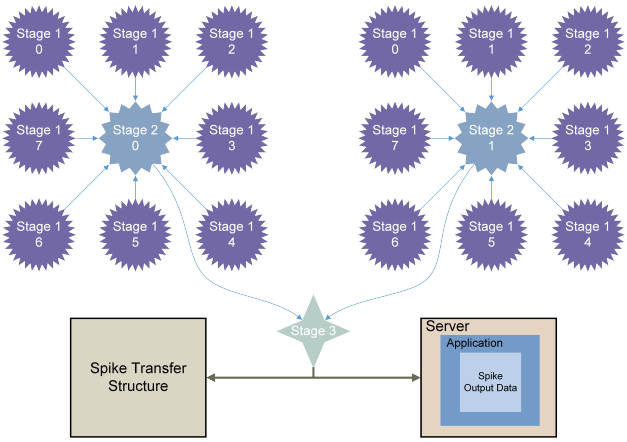

Fig. 5. Three-stage Output Spike Consolidator: the Output Spike Consolidator is a series of staged buffers that efficiently routes all the spike information resulting form a LIFop from the Neuronal Processing Units to the Spike Transfer Structure in a highly parallel way. The design choice of the Output Spike Consolidator plays a critical role in the placement of neurons in the underlying hardware medium. Neurons are tightly coupled to their respective stage one output buffers to reduce routing complexity and routing distance.

computation step is defined to be D LIFops. This forces the STPU to block I/O while computations on the temporal buffer are performed. All spikes are stored in a small local memory of D bits for each neuron. After the computation step is completed, all spikes generated are released to the OSC with their relative time of spike. The STS uses this relative spike time information to appropriately place the spike information into the temporal buffer of the post synaptic neuron; this preserves accurate timing for all spikes and increases computational throughput.

III. Software API

A crucial part of any architecture is the software stack that supports programming. A typical software stack for a von Neumann machine includes at least: 1) an operating system to abstract the hardware, manage resources and schedule code execution, 2) a loader to read object files into memory and start program execution, 3) a compiler to turn source code into object code, and 4) source files to specify procedures in a language that is both human and machine readable.

Neuromorphic platforms also need a software stack to run applications. For example, IBM's TrueNorth comes with the Corelet Programming Environment. Sandia National Labs has developed a general purpose software stack called "Neurons to Algorithms" (N2A) [25], which can run a network specification on a range of platforms with minimal changes.

A. Neurons To Algorithms

The N2A software stack compiles a neural system description into object code for a given device, loads it, and manages input and output. The specifics are determined by back end modules, just as traditional compilers have back ends to output object code for various architectures. N2A represents neural components as bundles of equations and associated parameters. Each bundle is called a "part". Because

everything is in the form of "name=value" pairs, parts are data, not code in the traditional sense. The language is declarative, not procedural. This makes it easy for one part to inherit from another and extend it. Parts may also include other parts, assembling them into large hierarchical models.

We observe that most physically-realized neuromorphic platforms (*e.g* SpiNNaker [17], TrueNorth [4], BrainScaleS [18], Neurogrid [26], and Darwin [27]) implement some form of LIF neuron combined with spike-based communication, where spikes may have some delay and add some amount of activation to the receiving neuron. This abstraction (LIF neurons + spike events) is thus a common language for programming these devices. The STPU also uses this abstraction, with fine-grained support for spike timing.

N2A supports network design for the STPU via a library LIF part and an event predicate. The LIF neurons are described in traditional neuroscience terms, using electrical units such as Volts, Amps, Siemens, and Farads. By expressing neuron configurations in a target-agnostic form, the same model can run across a wide range of platforms. At present, N2A supports several software simulators along with the STPU. Support for TrueNorth and SpiNNaker may become available in future releases.

The STPU back end translates each neuron configuration into the units used by the device, and makes calls to a user-space library to send the parameters onto the FPGA. The back end processes the connections between neurons and packs them into Synapse Memory Lists. Not all neurons get loaded onto the STPU. Some are necessary for reading input files and/or receiving output spikes. An entire neural network can be set up for pre- and post-processing of spikes flowing through the STPU. The back end transparently recognizes these and allocates them on the CPU side.

IV. Hardware Implementation

The architecture of the STPU was implemented in an FPGA.The current implementation exists on a Nallatech 385A. The 385A is a Low Profile 8-lane PCI-Express (PCIe) 3.0 card with an Intel®Arria®10 GX 1150 FPGA[1]. The FPGA on the 385A is connected to two banks of DDR3L SDRAM[2]. Each bank is 72 bits wide with 4 GB of memory operating at 2133 MT/s. This 8 GB of memory serves as the SM of the STPU. Using 8 bits per parallel computation path in the STS, this instantiation of the STPU has a total of 16 parallel computations paths—8 on each memory bank. The Aria®product line was specifically chosen for their patented memory logic array block (MLAB). The MLAB is a general-purpose dual-port memory SRAM array. We utilize this special memory for the temporal buffer.

Utilizing the Nallatech 385's fabric logic and auto-place and route tools, we have currently instantiated 2048 neurons with a temporal buffer size of 32. This instantiation utilizes 4 GB of the SM with 18 bits for each synaptic weight, 6 bits for each synaptic delay, and 8 bits to define the post-synaptic neuron. Technically, 11 bits would be needed to define each of the 2048 possible post-synaptic neurons, but due to hierarchical organization of the SM those 3 additional bits and the pre-synaptic neuron information are encoded in the SM address bits.

[1]FPGA part number: 10AX115N3F40E2SG in a F1517 package
[2]DDR3L part number: MT41K512M8RG-093:N (Micron)
[3]One neuron in the system is utilized to start the spiking of the network.

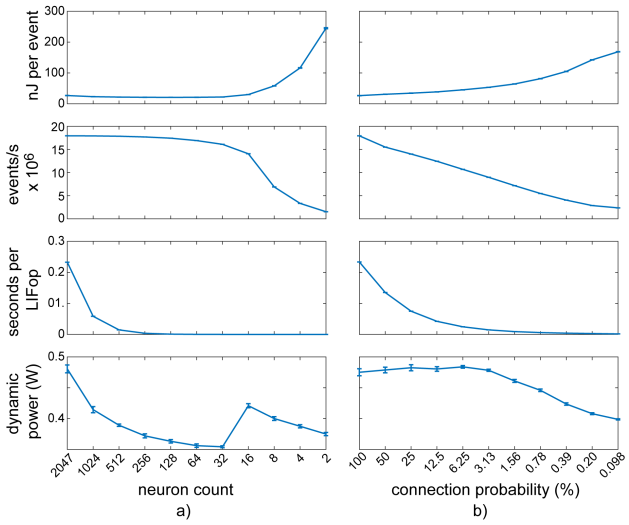

Fig. 6. (a) Eleven trials are conducted on fully connected networks in which the total synaptic event count is constant, but the neuron count in the system is reduced by a factor of two for each trial. The LIFop count is increased by a factor of four for each trial to maintain a constant synaptic event count. Every neuron spikes on every LIFop. All trials were executed ten times and error bars are plotted. (b) The neuron count is kept constant at 2047 and the probability of a neuron connecting to each other neuron is reduced by a factor of two each trial[3]. Eleven trials are conducted in which the synaptic event count is constant, but the synapse count is approximately reduced by a factor of two for each trial. Every neuron spikes on every LIFop. All trials were executed ten times and error bars are plotted.

A. Power Measurements

Power data was collected using a 2048 neuron instantiation of the STPU. A PCIe riser cable was modified to expose the power rails of the PCIe connector. A Keysight N6705B DC Power Analyzer was used with a N6785A Source/Measure Unit in line with the power rail of the PCIe connector to measure current and voltage simultaneously. Only the dynamic power was considered for this analysis so that only the power consumption of the execution of the network dynamics is considered and not the overhead of the entire Nallatech 385A. The Nallatech 385A had a static power of approximately 25W. However, it was noted that the static and dynamic power varied significantly between builds of the FPGA code, a characteristic of the random nature of the place and route tool.

Three experiments were conducted to collect data for computation of the average dynamic power, energy per synaptic event, synaptic events per second, and seconds per LIFop (Figures 6 and 7). In Experiment 1 (Figure 6a), the dynamic power decreases as the neuron count falls, with an anomaly when transitioning from 32 to 16 neurons and an average dynamic power of 392mW. The nJ per synaptic event remained relatively stable for neuron counts between 2047 and 32 with a minimum of 20.8nJ per synaptic event at 128 neurons. For

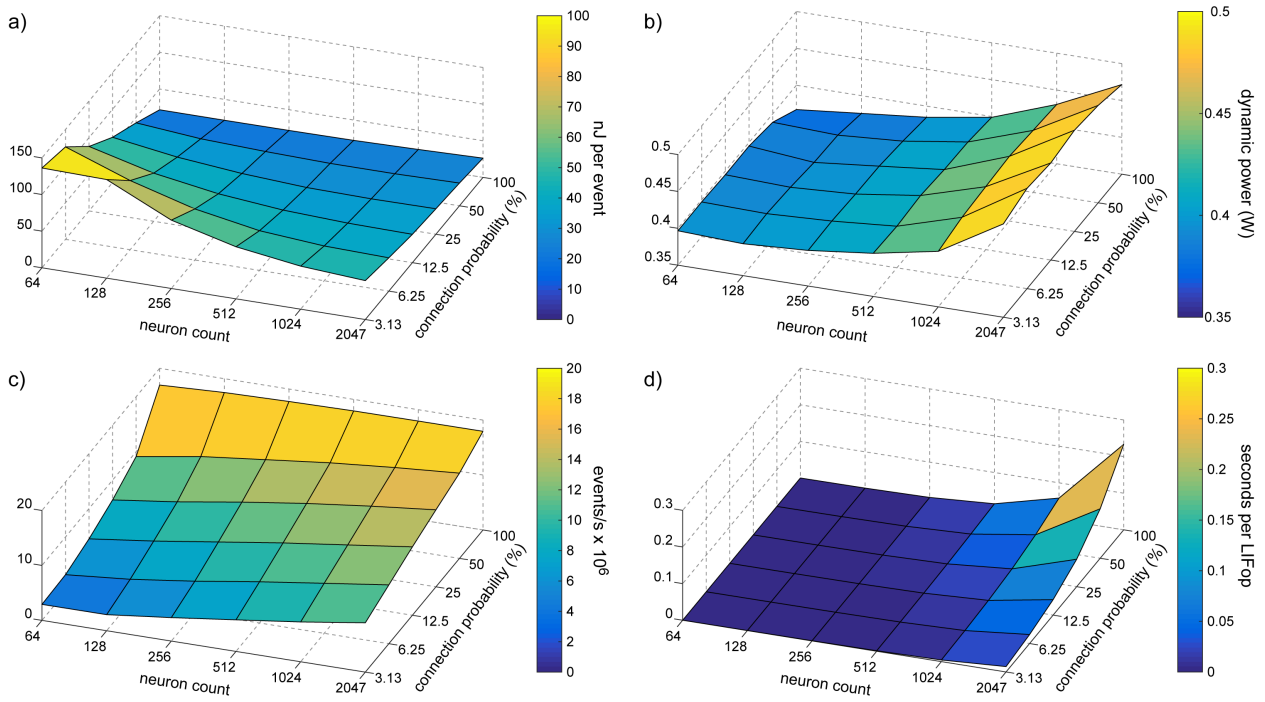

Fig. 7. The neuron count and the connection probability are varied and the synaptic event count is allowed to float. (a) nJ per synaptic event increases as neuron count decreases and connection sparsity increases. (b) dynamic power decreases as neuron count decreases and remains stable over connection probability. (c) synaptic events per second decreases as neuron count decreases and connection sparsity increases. (d) seconds per LIFop decrease with diminishing returns as neuron count decreases and connection sparsity increases.

neuron counts below 32 the nJ per synaptic event increases non-linearly. A similar trend is present for synaptic events per second. We report a peak throughput in Experiment 1 of 17.95 million events per second. The reason for the divergence of performance in low neuron count is attributed to the diminishing returns in the seconds per LIFop. In a fully connected topology the synapse count is reduced by a factor of 4 when the neuron count is reduced by a factor of 2. This presents a drastic reduction in the synaptic calculations performed for each LIFop. However, every neuron is spiking every LIFop. Therefore, the routing time becomes the major factor in the total time to complete one LIFop. Experiment 1 reports a 1ms LIFop time for 128 neurons.

The second experiment maintained a constant neuron count while the connectivity between neurons was varied (Figure 6b). Because Experiment 2 maintains a consistently high spike density and only reduces the number of synapses by a factor of 2, the effects of the LIFop timing are more definable. Even though the synapse count is reduced by a factor of 2 with each trial, the seconds per LIFop is not reduced by the same factor. As a result the nJ per synaptic event (min of 26.5nJ) steadily increases and the synaptic events per second throughput (max of 17.94×10^6) slowly decreases. Again, we see a generally decreasing trend in dynamic power as the number of synapses decreases. However, in this experiment the dynamic power remains stable until the connection probability reaches 3.13%, and then begins to decrease. The average dynamic power in Experiment 2 was 456mW.

The third experiment (Figure 7) shows that there is a diminishing return on the time savings for a single LIFop as the amount of computation per LIFop decreases (Figure 7d). This timing effect is the major factor in why the nJ per synaptic event increases and the synaptic events per second decreases along similar trends in Figure 7a and 7c. The dynamic power in Figure 7b indicates that the major factor in decreasing the power is a reduction in neuron count.

Based on the work of [28] we can place the STPU architecture performance in the context of related neuromorphic computing platforms. In regards to energy per event the STPU reported 21nJ in contrast to Neurogrid (100pJ), BrainSaleS (100pJ), TrueNorth (25pJ), and SpiNNaker (10nJ). For dynamic power the STPU reported an average dynamic power across all trials of 424mW in contrast to Neurogrid (150mW), BrainSaleS(1.3W), TrueNorth (72mW), and SpiNNaker (1W). An important distinction is that the STPU performance data was collected on an experimental FPGA implementation while the other neuromorphic computing systems are optimized ASIC designs.

V. Conclusion

The STPU is designed as a biologically-inspired spiking neural network architecture. The architecture is based on three fundamental neuroscience principles: the significance of neuronal connectivity, the efficacy of the connectivity, and the temporal latency imposed by the connectivity. While other neuromorphic hardware platforms provide some limited capabilities for configuring the relative timing within neural algorithms, the STPU architecture centers around the integra-

tion of spike-timing and information processing in SNAs by providing direct hardware support to efficiently utilize spike delays [10], [11]. The innate temporal processing capabilities of the architecture allow for delayed arrival of spikes with high fidelity, providing a flexible platform for transforming dynamic temporal data. These features make this architecture well suited for SNAs such as liquid state machines [13]. We have implemented the STPU architecture on an FPGA platform and measured performance metrics related to power and execution speed. In terms of power efficiency and throughput, the STPU performs optimally with large and densely connected networks while naturally executing faster for smaller sparsely connected networks. The STPU was developed to implement SNAs that utilize temporal domain representations, as we believe that the timing dynamics of SNAs will play a critical role in future neuromorphic applications.

ACKNOWLEDGMENT

The authors acknowledge financial support from Sandia National Laboratories' Laboratory Directed Research and Development Program, and specifically the Hardware Acceleration of Adaptive Neural Algorithms (HAANA) Grand Challenge Project. Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energys National Nuclear Security Administration under contract DE-NA0003525.

REFERENCES

[1] C. D. James, J. B. Aimone, N. E. Miner, C. M. Vineyard, F. H. Rothganger *et al.*, "A historical survey of algorithms and hardware architectures for neural-inspired and neuromorphic computing applications," *Biologically Inspired Cognitive Architectures*, vol. 19, pp. 49–64, 2017.

[2] S. Agarwal, T.-T. Quach, O. Parekh, A. H. Hsia, E. P. DeBenedictis *et al.*, "Energy scaling advantages of resistive memory crossbar based computation and its application to sparse coding," *Frontiers in neuroscience*, vol. 9, 2015.

[3] J. Hasler and H. B. Marr, "Finding a roadmap to achieve large neuromorphic hardware systems," *Frontiers in neuroscience*, vol. 7, p. 118, 2013.

[4] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[5] J. Schemmel, L. Kriener, P. Müller, and K. Meier, "An accelerated analog neuromorphic hardware system emulating nmda-and calcium-based nonlinear dendrites," *arXiv preprint arXiv:1703.07286*, 2017.

[6] J. Jaramillo and R. Kempter, "Phase precession: a neural code underlying episodic memory?" *Current Opinion in Neurobiology*, vol. 43, pp. 130–138, 2017.

[7] L. M. Rangel, L. K. Quinn, A. A. Chiba, F. H. Gage, and J. B. Aimone, "A hypothesis for temporal coding of young and mature granule cells," *Frontiers in neuroscience*, vol. 7, p. 75, 2013.

[8] C. Clopath, L. Büsing, E. Vasilaki, and W. Gerstner, "Connectivity reflects coding: a model of voltage-based stdp with homeostasis," *Nature neuroscience*, vol. 13, no. 3, pp. 344–352, 2010.

[9] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural computation*, vol. 14, no. 11, pp. 2531–2560, nov 2002.

[10] W. Severa, O. Parekh, K. D. Carlson, C. D. James, and J. B. Aimone, "Spiking network algorithms for scientific computing," in *Rebooting Computing (ICRC), IEEE International Conference on*. IEEE, 2016, pp. 1–8.

[11] S. J. Verzi, C. M. Vineyard, E. D. Vugrin, M. Galiardi, C. D. James *et al.*, "Optimization-based computation with spiking neurons," in *Proceedings of the IEEE International Joint Conference on Neural Network*. IEEE, 2017, p. in press.

[12] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley & Sons, 2012.

[13] M. R. Smith, A. Hill, K. D. Carlson, C. M. Vineyard, J. Donaldson *et al.*, "A novel digital neuromorphic architecture efficiently facilitating complex synaptic response functions applied to liquid state machines," in *Proceedings of the IEEE International Joint Conference on Neural Networks IJCNN'17*, 2017, p. in press.

[14] P. L. Follett and D. Follett, "White matter injury impairs activity-dependent plasticity in a developmental connectivity model," in *Society for Neuroscience*, vol. 28, no. 6, 2008.

[15] P. L. Follett, C. Roth, D. Follett, and O. Dammann, "White matter damage impairs adaptive recovery more than cortical damage in an in silico model of activity-dependent plasticity," *Journal of Child Neurology*, vol. 24, no. 9, pp. 1205–1211, 2009.

[16] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur *et al.*, "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*, vol. 34, no. 10, pp. 1537–1557, 2015.

[17] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras *et al.*, "Overview of the spinnaker system architecture," *IEEE Transactions on Computers*, vol. 62, no. 12, pp. 2454–2467, 2013.

[18] J. Schemmel, J. Fieres, and K. Meier, "Wafer-scale integration of analog neural networks," in *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*. IEEE, 2008, pp. 431–438.

[19] G. Indiveri, B. Linares-Barranco, T. J. Hamilton, A. Van Schaik, R. Etienne-Cummings *et al.*, "Neuromorphic silicon neuron circuits," *Frontiers in neuroscience*, vol. 5, p. 73, 2011.

[20] K. A. Boahen, "Point-to-point connectivity between neuromorphic chips using address events," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 5, pp. 416–434, 2000.

[21] A. G. Andreou, K. A. Boahen, P. O. Pouliquen, A. Pavasovic, R. E. Jenkins *et al.*, "Current-mode subthreshold mos circuits for analog vlsi neural systems," *IEEE Transactions on neural networks*, vol. 2, no. 2, pp. 205–213, 1991.

[22] J. Park, S. Ha, T. Yu, E. Neftci, and G. Cauwenberghs, "A 65k-neuron 73-mevents/s 22-pj/event asynchronous micro-pipelined integrate-and-fire array transceiver," in *Biomedical Circuits and Systems Conference (BioCAS), 2014 IEEE*. IEEE, Oct 2014, pp. 675–678.

[23] P. Dayan and L. F. Abbott, *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural System*. Cambridge, MA: MIT Press, 2001.

[24] J. Park, T. Yu, S. Joshi, C. Maier, and G. Cauwenberghs, "Hierarchical address event routing for reconfigurable large-scale neuromorphic systems," *IEEE transactions on neural networks and learning systems*, 2017.

[25] F. Rothganger, C. E. Warrender, D. Trumbo, and J. B. Aimone, "N2A: A computational tool for modeling from neurons to algorithms," *Frontiers in neural circuits*, vol. 8, 2014.

[26] S. Choudhary, S. Sloan, S. Fok, A. Neckar, E. Trautmann *et al.*, "Silicon neurons that compute," *Artificial neural networks and machine learning–ICANN 2012*, pp. 121–128, 2012.

[27] D. Ma, J. Shen, Z. Gu, M. Zhang, X. Zhu *et al.*, "Darwin: A neuromorphic hardware co-processor based on spiking neural networks," *Journal of Systems Architecture*, p. in press, 2017.

[28] S. Furber, "Large-scale neuromorphic computing systems," *Journal of neural engineering*, vol. 13, no. 5, p. 051001, 2016.