
FROM RISC-V CORES TO NEUROMORPHIC ARRAYS: A TUTORIAL ON BUILDING SCALABLE DIGITAL NEUROMORPHIC PROCESSORS

Amirreza Yousefzadeh
University of Twente, The Netherlands
a.yousefzadeh@utwente.nl

ABSTRACT

Digital neuromorphic processors are emerging as a promising computing substrate for low-power, always-on EdgeAI applications. In this tutorial paper, we outline the main architectural design principles behind fully digital neuromorphic processors and illustrate them using the SENECA platform as a running example. Starting from a flexible array of tiny RISC-V processing cores connected by a simple Network-on-Chip (NoC), we show how to progressively evolve the architecture: from a baseline event-driven implementation of fully connected networks, to versions with dedicated Neural Processing Elements (NPEs) and a loop controller that offloads fine-grained control from the general-purpose cores. Along the way, we discuss software and mapping techniques such as spike grouping, event-driven depth-first convolution for convolutional networks, and hard-attention style processing for high-resolution event-based vision. The focus is on architectural trade-offs, performance and energy bottlenecks, and on leveraging flexibility to incrementally add domain-specific acceleration. This paper assumes familiarity with basic neuromorphic concepts (spikes, event-driven computation, sparse activation) and deep neural network workloads. It does not present new experimental results; instead, it synthesizes and contextualizes findings previously reported in our SENECA publications to provide a coherent, step-by-step architectural perspective for students and practitioners who wish to design their own digital neuromorphic processors.

1 Introduction

Interpreting complex sensory data patterns in real-time and always-on with minimum power consumption is important for the survival of any living creature. The brain is responsible for performing this computation and has evolved over millions of years to be efficient in power consumption and processing speed. A honey bee's brain uses less than 1 milliwatt of power, yet it can perform a wide range of complex tasks such as navigation, communication, learning, and memory in real-time. The brain seamlessly integrates memory and learning and constantly evolves through experience. Despite the fact that ions, which are the signal carriers in the brain, move a million times slower than electrons, biological brains outperform electronic computers in terms of power consumption and latency. This document promotes the biological brain's architecture as a blueprint for future computing technologies, especially in the field of EdgeAI.

The brain is composed of a complex network of neurons that work together to create a highly efficient dataflow architecture. This architecture is designed for distributed processing and memory, enabling efficient parallel processing across a large three-dimensional structure. Neurons communicate through spikes, which allows for reliable digital data exchange to overcome the noisy environment. Moreover, this communication is sparse, which allows data to propagate and be processed quickly and efficiently. Although the brain is capable of executing any computational task, it specializes in processing natural signals such as audio and video.

1.1 Promises of Neuromorphic engineering

Neuromorphic engineering is a field that aims to enhance the speed and efficiency of computers by processing data intelligently. The main principle behind this approach is activation sparsity, which involves processing only the most

significant data while avoiding energy consumption on ineffective operations. This approach is similar to how our brains work, and not only saves power but also enables these systems to respond quickly, making them ideal for fast-acting applications.

Neuromorphic processors are particularly useful for applications that deal with sparse data. Sparsity can be found in natural signals such as audio and video. In these signals, not every point in time holds important information. Similarly, the neurons in the brain are usually only 1 to 10% active at any given time [1].

To mimic this efficiency, neuromorphic processors are designed to be event-driven. They only process data when an event occurs, and in the absence of events, the processor idles. This feature allows them to consume power proportionally to the number of events they receive. This is known as event-driven power consumption.

The co-localization of memory and processing is a key feature that improves the efficiency of neuromorphic processors. This is accomplished by placing memory and processing units near each other, which is similar to how the human brain functions. This design significantly reduces the amount of data that needs to be moved, which is the primary factor that consumes power in most computing platforms [2].

Both sparsity exploitation and distributed parallel processing pose several challenges in developing algorithms, programming, and fabricating platforms. This is mainly because the mature processing ecosystem stack does not support such models efficiently. For instance, training Spiking Neural Networks in GPUs, the most accessible deep learning accelerators, is an extremely slow process. These challenges have hindered the growth and integration of neuromorphic systems into today's market [3].

1.2 Digital or Analog

In this tutorial, we will be discussing the development of digital neuromorphic processors. While "neuromorphic processing" is commonly associated with "analog computing", it has been shown that integrating neuromorphic principles in the design of traditional digital processing systems can significantly improve their performance [4]. Moreover, digital neuromorphic processors can pave the way for analog neuromorphic computing and can be scaled and developed in areas that are currently not feasible with analog computing.

Digital design offers several advantages over analog design. One of the most significant benefits is its compatibility with advanced technology nodes. As semiconductor manufacturing progresses, digital systems can be built using smaller, more efficient, and faster transistors. Digital neuromorphic systems can thus take full advantage of the latest manufacturing techniques, which can enhance their performance and reduce costs. In contrast, analog designs require a complete redesign to adapt to each new technology node, and as the size of analog components decreases, they become more susceptible to noise, which can significantly affect their performance [5].

Digital logic blocks can be easily duplicated and reused in different designs, which simplifies the process of creating complex architectures. This means that once a digital circuit is designed for a neuromorphic function, it can be effortlessly integrated into multiple systems. Consequently, digital neuromorphic processors are highly reconfigurable, making them more adaptable than analog systems and better equipped to support a wide range of applications.

The main benefit of using analog neuromorphic designs is their low power consumption. Analog circuits can operate with extremely low power, which is especially useful when energy conservation is a priority. However, when it comes to overall performance and energy efficiency, digital systems may actually be more effective [6]. This is because digital systems can utilize superior technology nodes and employ more complex and precise designs. Combining digital and analog systems in a single neuromorphic processor can sometimes provide the best of both worlds, exploiting the power efficiency of analog circuits with the precision and scalability of digital design [7]. However, choosing between digital or analog implementations should be based on the specific needs of the application, as each approach has unique contributions to the development of neuromorphic engineering.

The field of digital neuromorphic chips is rapidly advancing, with various companies and research groups developing successive generations of technology. SpiNNaker[8], a chip developed as part of the Human Brain Project, now has an updated version called SpiNNaker2[9]. Intel's neuromorphic chip, Loihi[10], has also seen an upgrade in the form of Loihi2[11], indicating significant progress in digital neuromorphic technology. IBM has made significant contributions to the development of neuromorphic computing through its TrueNorth chip [12]. The company has introduced a successor to the TrueNorth chip named NorthPole [13]. IMEC's initial venture into this space with the uBrain processor [14] has led to the development of a more advanced chip called SENECA [15].

The field of digital neuromorphic processing technology is being revolutionized by a number of startups that are introducing EdgeAI solutions to the market. Among these startups is GraI Matter Labs, which was acquired by Snap. They have developed the NeuronFlow architecture [16], followed by GraI-VIP as a second generation. Synsense, another

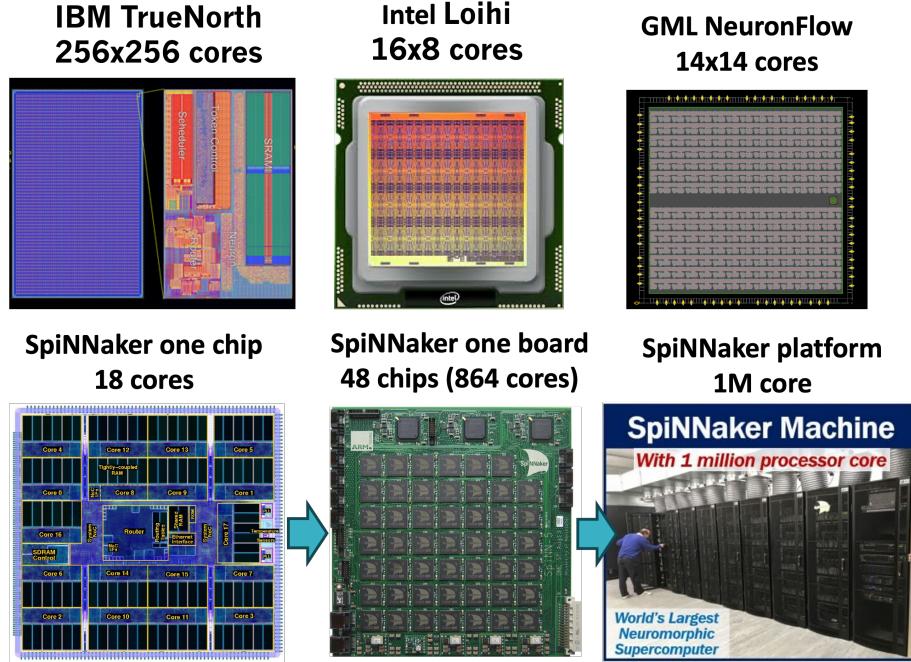


Figure 1: Examples of several digital neuromorphic chips. All follow the same template of connecting many tiny processing elements to build a scalable distributed processing platform.

neuromorphic startup, initially designed analog neuromorphic processors and now also offers digital neuromorphic processors [17]. BrainChip is continuously scaling and improving its AKIDA architecture, which is a fully digital neuromorphic processor [18]. These examples illustrate a trend of continuous development in digital neuromorphic chips, with each new release aiming to provide better performance and energy efficiency.

1.3 The SENECA Project

This document is a compilation of the findings of the SENECA project, an ongoing initiative in the field of neuromorphic engineering that has been in progress for several years. It presents the results from previously published papers, which reflect the continuous efforts and progress made by the project's team of researchers and university collaborators [15, 19, 20, 21, 22, 23].

The SENECA project began in the summer of 2020 and has so far been funded by seven research grants from the European Union (€1.7M), The Netherlands Enterprise Agency (€0.8M), and IMEC (€1.5M): TEMPO[24], ANDANTE[25], MNEMOSEN[26], DAIS[27], MeM-Scales[28], REBECCA[29], NEUROKIT2E[30] and NimbleAI[31]. Although it is still in progress, the project has already made significant contributions to the field, and further advancements and insights are expected in the future.

2 The main architectural template: Array of Tiny Processors

Advanced digital neuromorphic processors are typically composed of a network-on-chip that interconnects many small processing elements. The image in Fig.1 displays several digital neuromorphic processors that were designed using this same configuration. This structure emulates the decentralized architecture of the biological brain, resulting in distributed memory and processing power across the processor. In neuromorphic platforms, each core communicates with others by sending and receiving data packets called spikes or events. Thus, to create neuromorphic processors, two essential components are required: the tiny processing core and the Network on Chip (NoC). The processing cores function as the fundamental units of computation, similar to neurons in the human brain, while the NoC serves as the communication backbone, analogous to the synaptic connections within the brain's network.

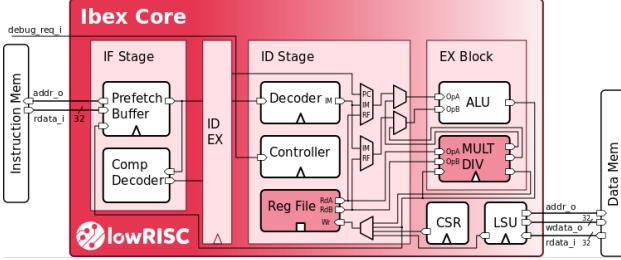


Figure 2: Internal pipeline and architecture of IBEX, from iis-projects.ee.ethz.ch.

2.0.1 Tiny processor core

Neuromorphic processors have independent and asynchronous cores that can process input events and generate output events in parallel¹. Each core has its own memory and processing logic, and some may also have control units that manage the overall process.

The functionality of a platform is mainly determined by the design of its neuromorphic processing core. These functionalities include the neuron models and activation functions (e.g., LIF, ReLU, LSTM, and GRU), the neural network architectures (e.g., Dense, Conv, Recurrent, and Transformer), the number of neurons and weights, as well as the precision of these weights. The architecture of a processing core has a significant impact on the performance, area, and energy consumption of the system. Designing such a core involves various trade-offs, which are listed in [21] and will be briefly discussed here.

A neuromorphic processing core can emulate several neurons in a neural network by **time-multiplexing** their fast data path. This is possible because a single processing core can work much faster than a biological neuron. Though time-multiplexing is not biologically plausible, it is commonly used in the design of neuromorphic cores, as it improves the area efficiency of the system. Designers have the option to increase the time-multiplexing ratio to enhance area efficiency and reduce communication overhead, or decrease it to improve performance and energy efficiency. This trade-off is an essential consideration for designers.

When developing a processing core, there is a crucial decision to be made between **flexibility and performance**. A core that prioritizes extremely high performance may be limited to a narrower range of applications and cannot benefit from future software optimizations. Therefore, this decision isn't always straightforward and is often misunderstood. This is because it is possible to improve system performance by leveraging the flexibility of an architecture. Several recent works demonstrate superior performance achieved by deploying neural networks on commercial-off-the-shelf CPUs/GPUs, compared to domain-specific deep learning accelerators or neuromorphic processors (e.g., [32], [33], [34], [35]).

In this tutorial, we start with a highly flexible neuromorphic core. We then integrate domain-specific accelerators to enhance performance for specific benchmarks, such as fully connected and convolutional layers. This approach allows us to continually enhance the performance of our neuromorphic platform and keep up with the dynamic changes in the neuromorphic algorithm domain. With this methodology, the core can meet the high-performance requirements of target algorithms while remaining flexible enough to deploy new algorithms or software optimizations. However, the drawback is that it may require more silicon area compared to fully dedicated cores.

SpiNNaker is an excellent example of a neuromorphic platform that adopted such an approach. The initial version of the chip utilized tiny and low-power ARM processors as the neuromorphic processing cores. As the next step, SpiNNaker2 has been developed with several new accelerators added to the ARM processors to improve its performance for specific applications while maintaining its general applicability for new applications. SpiNNaker proved to be a valuable endeavor, as the scaled-up platform is versatile enough to handle all kinds of neuromorphic computations, including various neuron models [36], neural networks [37], and learning algorithms [38].

As the core of the SENECA project, we used tiny RISC-V processors as our processing cores. Our main goal was to quickly build a flexible neuromorphic platform capable of running various applications. After conducting some benchmarking on the Epiphany platform (which is also built by connecting an array of RISC processors with NoC) and the SpiNNaker platform (which uses ARM processors), we decided to select the IBEX core (Fig.2). It demonstrated a reasonable area/performance trade-off and had excellent open-source repositories.

¹Here, asynchrony refers to a processing model where individual threads execute on each processing core without explicit synchronization barriers. Neuromorphic processors may have synchronized circuits but retain an asynchronous execution model.

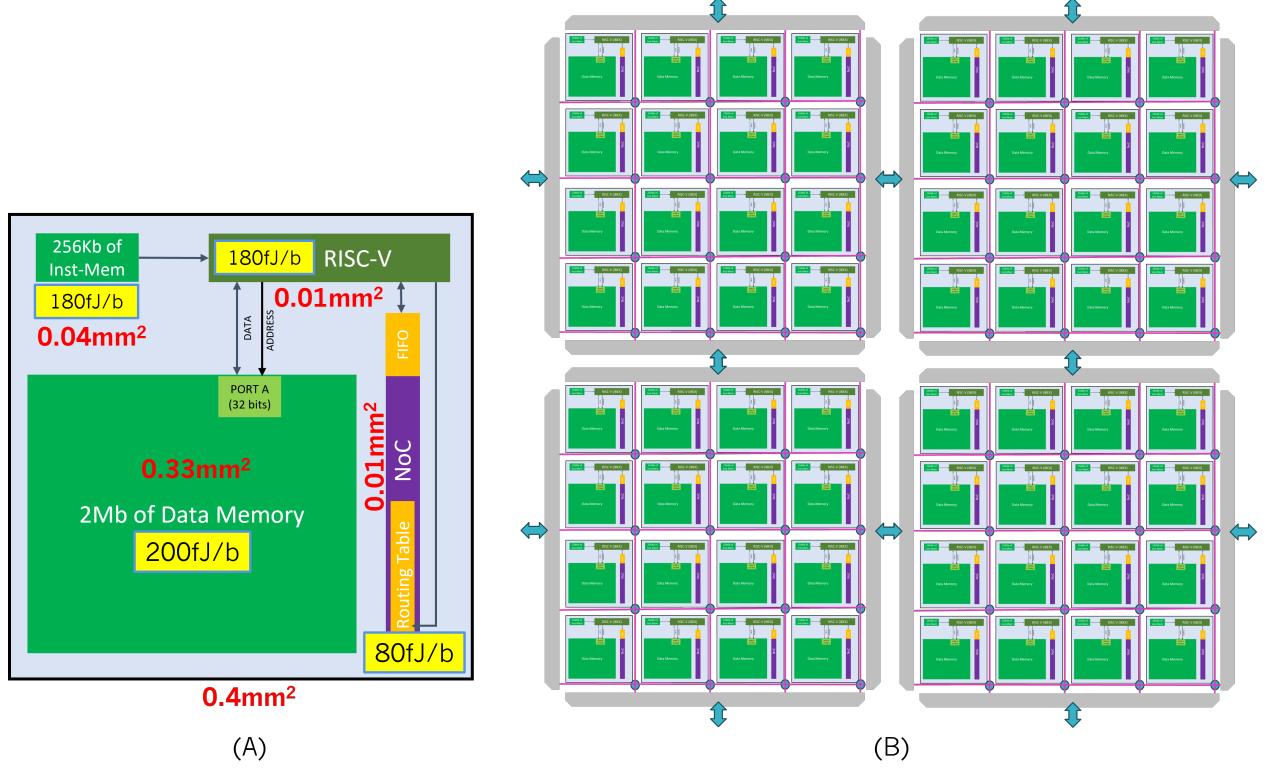


Figure 3: On the left, you can see the initial version of our neuromorphic processing core, while on the right, you can see the complete platform created by linking the cores together. The core architecture is annotated with the area and energy figures in the GF-22nm FDSOI technology node.

The simplified architecture of the proposed neuromorphic processor is illustrated in Fig. 3. It consists of several interconnected processing cores, each of which includes RISC-V and NoC (will go through it in the next subsection), as well as instruction and data memory. The execution model and neural network mapping are stored in the instruction memory, while the data memory stores the synaptic weights and neuron states. As planned, we began with the most basic and adaptable neuromorphic processor we could conceive.

2.0.2 Network on Chip

The Network on Chip, also known as NoC, is a technology that allows for efficient, flexible, and scalable connections between neuromorphic processing cores. In biological brains, processing cores can connect and disconnect during operation through flexible plastic wires. However, in electronic circuits, it is not possible to form completely new connections after fabrication. Therefore, all cores must be connected through wires by default. These wires are time-multiplexed among cores since the electrical communication speed is much faster than ions in the brain. NoC handles this time multiplexing to ensure an effective exchange of information between processing cores.

In neuromorphic platforms, the neurons are connected via shared wires. As a result, the generated spikes cannot be a single electrical pulse. Instead, each spike leaving the core must carry its unique source neuron ID to ensure proper decoding in the destination core. This means that each spike traveling through the NoC (Network on Chip) is actually a packet of data rather than a single pulse. This data packet is commonly known as AER (Address Event Representation) in many neuromorphic processors [39].

The main purpose of the Network on Chip (NoC) is to ensure that data (spike) is delivered promptly to enable the processing core to perform its functions as quickly as possible. After observing various neuromorphic platforms such as NeuronFlow, Loihi, SpiNNaker, and Epiphany, we found that the NoC has never been a performance bottleneck in those platforms. The high level of sparsity in our algorithms and the high operation density² for each spike could be the reason for this.

²Operation density is defined as the number of operations required to process a data packet.

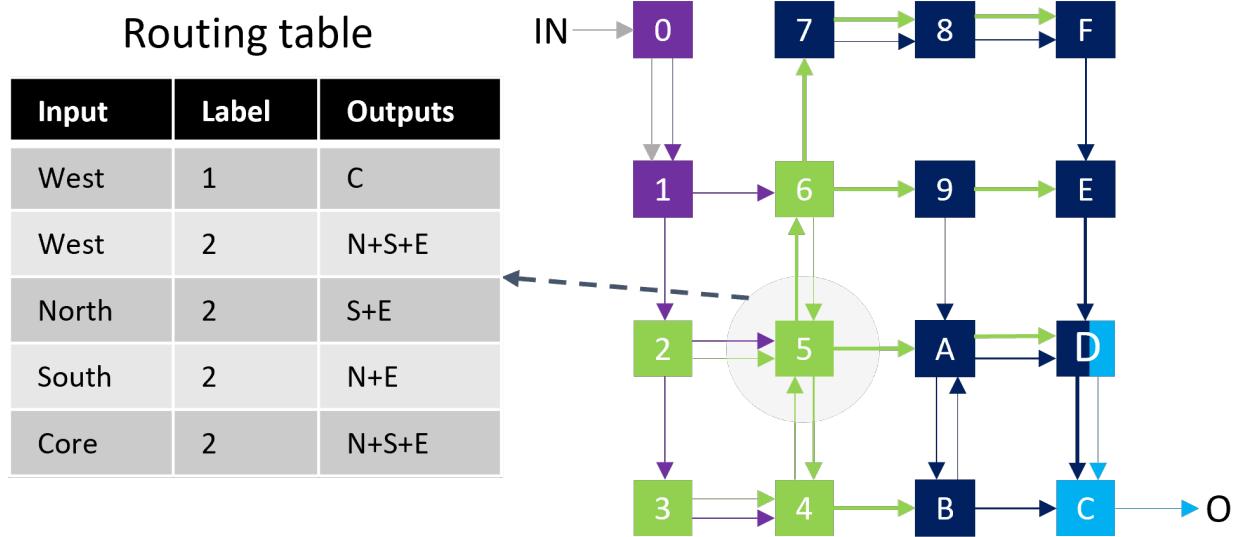


Figure 4: On the right, you can see a 4-layer neural network being mapped on 16 cores (color-coded). On the left, there is a routing table of the NoC in the core 5. The NoC processes the incoming packet based on the input port and the label it carries. It then routes them to one or several output ports, depending on the content of the routing table. The label is a part of the spike packet.

In SENECA, we observed that the Processing core (RISC-V) shown in Fig.3 was always our primary bottleneck in terms of both performance and energy consumption. We did not find that NoC significantly contributed to power consumption or performance. This is because when the operation density is high, transferring a data packet consumes much less energy/time than processing it. Based on these observations, we believe that any simple NoC that can flexibly connect the neuromorphic cores would be sufficient for our purpose. For instance, LOIHI [10], TrueNorth[12], and NeuronFlow [16] use a simple packet-switched mesh type NoC. There are a number of open-source NoCs that can be used to construct a neuromorphic platform directly [40, 41, 42].

NoC, despite not affecting performance and energy consumption in the system, can have various side effects on the performance of other system elements. For instance, it can create a high memory overhead to store the routing table. Each neuron in the core requires its set of destination addresses, defined by the network architecture and mapping. For instance, the TrueNorth [12] uses 26 bits for each neuron to designate a single destination core out of a possible 64 million cores in the system. This approach not only requires a considerable amount of memory but also lacks flexibility since each neuron can only be assigned to a single destination core. Another example is NeuronFlow [16]. Despite the flexible routing and axon-sharing mechanism in NeuronFlow, the NoC routing table still occupies about 25% of the total memory.

SpiNNaker [8] uses a routing method called source-based addressing. Instead of the core knowing the destination address, each NoC has a routing table with enough information to forward the packet to its correct output ports. Therefore, the information about the final destination addresses of each neuron is distributed over many routers and can be optimized to reduce data movement and the size of routing tables. As a result, the routing information required for a structured neural network architecture (like CNN and Dense neural networks) can be much smaller.

A source-based addressing NoC has a second advantage of allowing efficient multicasting of the spike when the spike has multiple destination cores. Without multicasting, the core that generated the spikes would need to send multiple copies of the packet from the source to the destination, which could lead to network congestion. In the source-based addressing scheme, the packet can be copied to multiple output ports of the nearest router to reach the destinations efficiently. However, the improvement provided by this multicasting is negligible when the NoC is not congested.

SENECA has implemented a similar source-based addressing scheme as SpiNNaker, but with a simpler NoC design. We opted for a mesh scheme with one NoC per core. The SENECA NoC concept is illustrated in Fig. 4, which features a small routing table per NoC. Consequently, each spike packet must carry a label as its source address. Here, the label represents the source layer ID as shown in Fig. 4.

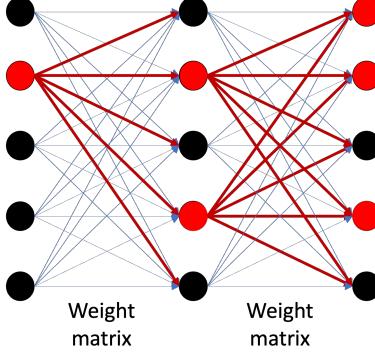


Figure 5: Event-driven processing of a fully connected layer. Neurons with non-zero activations that generate a spike (event) are shown in red. These events move to the next layer and update the corresponding receiving neurons.

Processor	Inference time (μs)	Energy (μJ)
Loihi1	3378	372
SpiNNaker2	1000	7.1
SENECA(V1)	7000	34

Table 1: Comparison of SENECA version 1 (RISC-V and NoC) with other neuromorphic platforms using a fully connected neural network application. More details about this experiment are available in [23].

3 Event-driven processing in our neuromorphic platform

We now have all the necessary components to build our neuromorphic platform, including tiny processing cores and a network-on-chip. We can proceed to implement a simple event-driven, fully connected neural network and measure its performance. Although fully connected neural networks are basic, they serve as a good representation of neural network accelerators' performance, as they form the foundation of any neural network. These networks require vector-matrix multiplication, where the vector represents the activation output of the previous layer and the matrix represents the synaptic weights that connect the previous layer to the next layer (as shown in Fig. 5).

In the computation of vector-matrix multiplication using an event-driven approach, we first break down the activation vector into its non-zero elements. Then, we perform several scalar-vector multiplications where the vector represents the weights of synapses that connect this activation to the next layer (these synapses are highlighted in bold red in Fig. 5). Each non-zero activation forms a spike packet that reaches the destination neurons and updates them based on the synaptic weight. These spike packets can be binary (like their biological counterparts) or can have values. In the case of binary spikes, the amount of synaptic weight will be added to the neuron state of the next layer (known as the partial sum of the neuron in DNN terminology). However, if the spike packet contains a value, this value needs to be multiplied by the synaptic weight before being added to the neuron state of the next layer.

There is an ongoing debate regarding the use of spikes with value, commonly known as valued or graded spikes. These types of spikes are controversial because they are not directly biologically plausible. However, using valued spikes can increase the accuracy of neural networks. This improvement comes at the cost of sending more bits over the NoC and performing a multiplication operation. Despite this drawback, valued spikes can potentially enable neural networks to achieve the same level of performance as binary spikes while using fewer spikes overall. This is why the second generations of Loihi, TrueNorth, and SpiNNaker all allow the use of graded spikes. In our case, since we are using a RISC-V, both options are available to us.

As a first step in the SENECA project, we used a simple benchmarking task that was previously deployed and benchmarked using Loihi1 [43] and SpiNNaker2 [44]. This activity allowed us to measure the performance of this simple neuromorphic chip in comparison to other, more advanced versions. The results are reported in [23] and can be found in Table 1. Please note that benchmarking on Loihi2 is not available.

In comparison to Loihi, our neuromorphic platform is slower by a factor of 2. This is because Loihi has a custom-made processing core with specialized instructions for the neural network. On the other hand, RISC-V requires many execution cycles to finish one neuron update. However, surprisingly, the energy consumption of SENECA(V1) is much better than that of Loihi. We believe the main reason is that Loihi's processing cores emulate a complex neuron model along with learning capabilities, which were not utilized in this benchmark. This fact indicates that a more dedicated

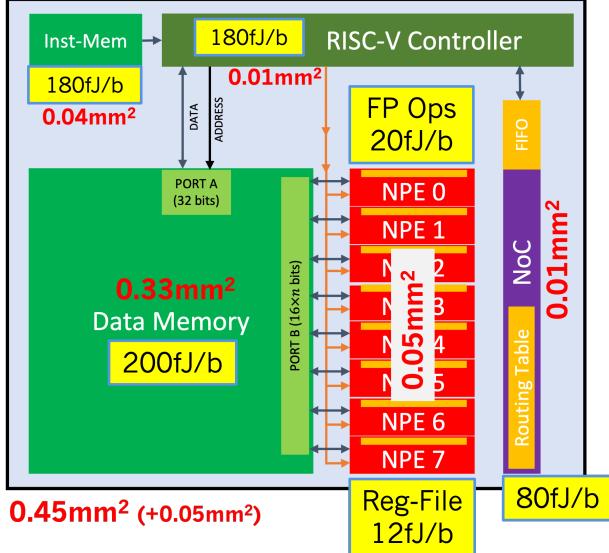


Figure 6: Neural Processing Elements (NPEs) have been integrated into our neuromorphic processors to build SENECA(V2). The figure now displays the energy and area consumption of the processing core after adding the NPEs. Additional details about the benchmarking results can be found in [22].

(less flexible) processing core can only result in good performance if there is a high degree of matching between the application and the hardware. In Loihi2, the neuron model is programmable to increase system flexibility.

When compared to SpiNNaker2, our SENECA(V1) platform consumes significantly more energy and takes longer to complete inference. The primary reason for this difference is that SpiNNaker2 has a dedicated neural processing accelerator that can update neurons much faster than a general-purpose processor. This observation has provided us with valuable insights into how we can optimize our neuromorphic platform.

4 Adding Hardware Accelerators

Although our neuromorphic platform is already complete with RISC-V processing cores and NoC, we want to demonstrate how adding dedicated accelerators can improve the platform's performance without sacrificing its programmability. Adding accelerators is beneficial if there's extra space in the silicon area, as they boost the system's performance when used and can be turned off when not in use.

4.1 Neuron Processing Elements

In SENECA, the first hardware accelerator added was a set of Neural Processing Elements (NPEs). These NPEs work synchronously, executing the same instructions on various data to perform efficient vector operations. Fig. 6 shows the architecture of the processing core, with 8 NPEs added to it. The figure is also annotated with the energy and area consumption [22]. However, NPEs are not a complete processor and require a controller to fetch and decode operations for them. Therefore, in SENECA(V2), RISC-V serves as the controller in the system and does not perform the neural computations anymore.

NPEs (Neural Processing Elements) execute Brain-Float (16b) operations by default, even though parameters can be stored in lower resolution. Using a 16-bit integer for neuron states (partial sums) was insufficient due to the limited range. BrainFloat16 operations consume more power than integer operations but provide a significantly higher range to avoid overflowing of neuron states. Floating point operations provide a mixture of linear and logarithmic quantization schemes that can be optimally tuned to obtain the best accuracy/performance trade-off. As shown in Fig. 6, the floating point operations in NPEs still consume 10x less power compared to data memory access. Due to the nature of being dedicated compute elements, they also consume much less than executing INT32 operations in RISC-V. You can find the full instruction set and measurements in [21].

The eight parallel NPEs (Neural Processing Elements) can perform eight complex neural operations simultaneously, which is expected to significantly enhance the system's performance. Furthermore, using BF16 instead of INT32 and

Processor	Inference time (μs)	Energy (μJ)
Loihi1	3378	372
SpiNNaker2	1000	7.1
SENECA(V1)	7000	34
SENECA(V2)	1100	7

Table 2: Deployment of the benchmarking application of Table 1 on SENECA(V2)

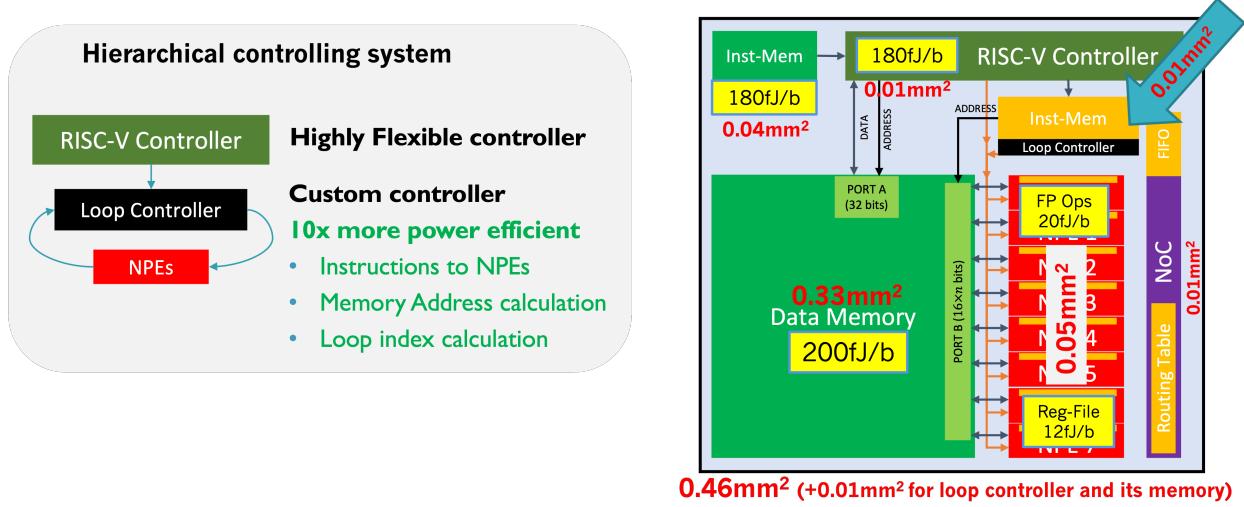


Figure 7: Adding loop controller to SENECA to make a new version: SENECA(V3)

the ability of a single controller to handle all eight NPEs significantly reduced the overall power consumption. The benchmarking results when using the NPEs in the SENECA(V2) platform are presented in Table 2. As shown, we have achieved almost the same performance as SpiNNaker2. This is not surprising, as the main difference between SENECA(V1) and SpiNNaker2 was the existence of the vector processing unit in SpiNNaker2. In [23], it has been demonstrated that the averaged energy consumption of RISC-V for one inference has reduced from $30\mu s$ to $3\mu s$, while NPEs only added 2us to execute all the neural updates.

The NPEs used in the SENECA project can be improved. One issue is the long pipeline (4 stages) caused by the use of floating-point ALU, resulting in inefficiency when there is a pipeline hazard. Another area for improvement is data conversion. Although the NPEs support parameters and operations in lower resolution (integer and flex point), converting between formats adds significant overhead. The final challenge is related to the compilation of NPE instructions. While RISC-V has its own advanced computer that could be used for this project, NPEs with their own instruction sets require their own compiler if the user wants to write code in higher-level languages. Currently, we are using assembly code for NPEs in the SENECA project, which slows down the development of new applications.

4.2 Loop Controller

After we added the Processing elements, we noticed that RISC-V utilized $3\mu J$ of energy, while NPEs only used $2\mu J$ for all the neural operations. Even though RISC-V is more flexible, executing instructions in it is quite expensive. So, we decided to analyze the C program of RISC-V to gain greater insight. During profiling, we discovered that RISC-V provides a small set of instructions to NPEs in a nested loop. Although the instructions were very similar, each instruction needed to be individually loaded from instruction memory, fetched and decoded, which added a lot of overhead to the system.

To improve the efficiency of executing loops in computer architecture, a solution known as a loop buffer has been traditionally used. Inspired by this, we developed a new independent controller, which we named the loop controller. This controller is specifically designed to handle neural network loops and has its own small instruction memory constructed using a register file that consumes less power than SRAM. We later referred to this concept as a hierarchical control system [21], where the RISC-V controls the Loop controller, which in turn controls the NPEs.

Processor	Inference time (μs)	Energy (μJ)
Loihi1	3378	372
SpiNNaker2	1000	7.1
SENECA(V1)	7000	34
SENECA(V2)	1100	7
SENECA(V3)	550	3

Table 3: Benchmarking result for SENECA(V3) which includes loop controller.

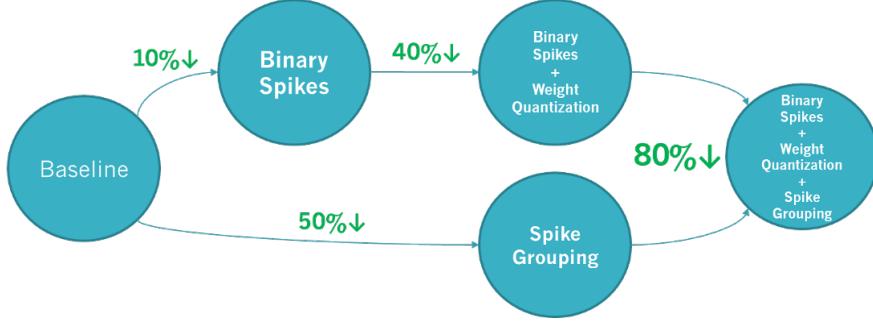


Figure 8: Performance (both energy and latency) improvement by using various optimization techniques. By using spike-grouping alone, the performance of SENECA(V3) can be improved by a factor of two in both energy and time. Additionally, using binary spikes can reduce power consumption by 10%, while weight quantization (to 4 bits) can reduce power consumption by 40%.

Fig.7 illustrates the SENECA(V3) architecture with the addition of a loop controller and its own instruction memory. The loop controller is highly power-efficient, being 10 times more efficient than RISC-V when executing loops and providing instructions to NPEs. It is designed to handle a limited set of instructions, specifically for managing loop indexes, nested loops, and memory address calculations.

Table 3 displays the performance improvement of SENECA(V3) when using the loop controller, as reported in [23]. The loop controller reduces the load on RISC-V, resulting in a decrease in energy consumption from $3\mu J$ to $0.2\mu J$.

After implementing the loop controller, we added several other accelerators to our system to boost its performance. You can find more details about these improvements in [21]. As explained in the article, these accelerators improved the system's performance in neural network inference, which was our target benchmark. However, they did not compromise the flexibility of our SENECA(V1) platform. Therefore, we can follow the same incremental approach to enhance the performance of our platform for any other benchmarking applications. For instance, we already reported our initial results for on-device learning (using e-Prob) in [21], and the SENECA team is currently working on designing hardware accelerators to enhance the performance of on-device learning further.

It is possible to perform pre- and post-processing of applications directly on the RISC-V. Depending on the expected performance of the system, it may or may not be necessary to design an accelerator for these glue algorithms. An example illustrating this point will be discussed in the next section.

5 Software optimizations

One of the key advantages of the flexible processing core is its ability to enhance system performance by implementing advanced algorithmic and software optimizations. In addition to typical hardware-aware algorithm optimizations such as quantization and sparsification, this section will outline three other techniques that we utilized to improve system performance.

5.1 Spike Grouping

Neuromorphic systems consume a lot of energy during memory access. When executing neural networks, it is common for the same data to be read and written multiple times in loops. To reduce memory access, GPUs use batch processing, reusing data as much as possible before discarding it. Similarly, in SENECA, a technique called spike-grouping is used to process several corresponding spikes at once, further reducing memory access.

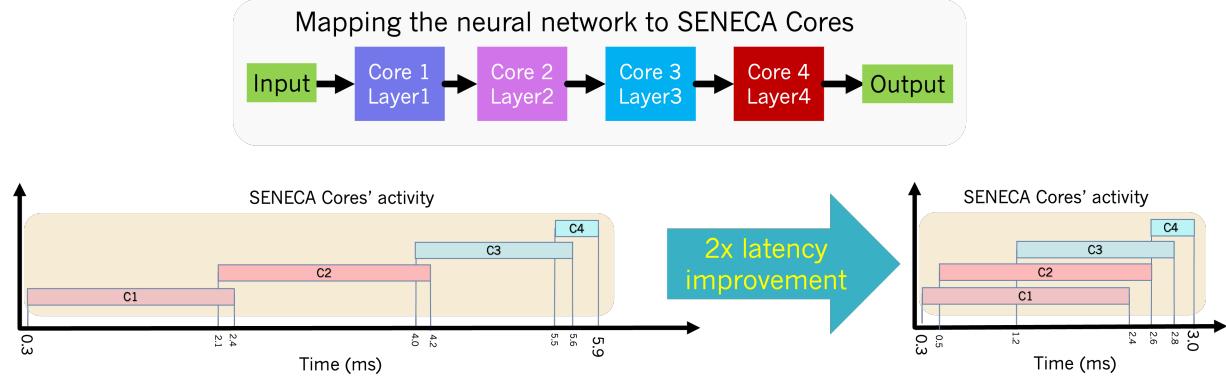


Figure 9: Using the event-driven depth-first method to map a 4-layer CNN on SENECA can reduce memory consumption and latency.

In Fig.5, we can observe that layer-2 produces two spikes which update all the neurons in layer-3. To process each spike, the neuron state of the neurons in layer 3 needs to be read, updated and written back. If we process both spikes simultaneously, the neurons in layer-3 will be read only once, updated twice and written back only once. Therefore, grouping two spikes reduces the memory access by reading and writing the neuron states only once for both spikes. We tried to group four spikes together (whenever possible), which resulted in a 50% reduction in both energy consumption and inference time, as demonstrated in Fig.8.

5.2 Depth-First inference

Neuromorphic platforms face a problem with Convolutional Neural Networks (CNNs) that results in excessive memory usage for neuron states. CNNs are popular because they can use a few individual parameters for all synaptic weights in the network, reducing the number of parameters that need to be stored compared to the number of synapses. As a result, CNNs typically have a much larger number of neurons than the number of individual parameters. However, when CNNs are used in neuromorphic platforms, they often consume significantly more memory (e.g., 200 times) compared to their deployment in a DNN accelerator. This is because neuromorphic processors allocate dedicated memory to each individual neuron.

Despite their memory limitations, Convolutional Neural Networks (CNNs) are crucial in the field of neuromorphic computing when it comes to processing visual data. However, the SENECA platform offers flexibility in implementing hybrid networks where some neurons do not have dedicated memory. An event-driven depth-first inference was developed to avoid using dedicated memory for each neuron in the feedforward CNN layers. This allows for efficient deployment of CNNs on multi-core mapping, resulting in a significant reduction in both latency and memory usage.

The Event-driven Depth-first method is explained in detail in [23]. Fig.9 (top) illustrates how four CNN layers are mapped onto four SENECA cores (one layer per core). The depth-first inference fuses the operations of layers, allowing the process of the next layer to start before finishing the previous layer. This results in reduced latency. Our benchmarking results demonstrate two orders of magnitude reduction in memory consumption using the event-driven depth-first convolution.

5.3 Hard attention

Processing high-resolution event-based cameras in neuromorphic platforms is challenging due to the size of neural networks and the excessive amount of operations which is required to finish a task.

One method for processing high-resolution images efficiently is called hard attention [45]. In this approach, a smaller version of the input is processed in a neural network to identify the Regions of Interest (RoI). Then, in parallel neural network(s), these RoIs are zoomed in (using the original high-resolution image) and analyzed to determine the content of the RoI. Implementing such a neural network requires significant pre- and post-processing mechanisms, which can be computationally heavy. To address this issue, we utilized the flexibility of SENECA and implemented a full hard attention solution in the platform. Our experiments with a neuromorphic benchmark (IBM Hand Gesture Recognition) showed that hard attention can improve the energy consumption of the application by 2x and the latency by 3x without sacrificing accuracy. The detailed results of this work are currently under review and will be published soon.

6 And more

The SENECA platform was created as a test bed for exploring and innovating in the field of neuromorphic (both hardware and software) technologies. In addition to what has been mentioned in this paper, many other explorations have been undertaken or are ongoing. For example, using new memory technologies, mapping neural networks with shared memory, exploring 3D architecture, and in-memory and in-material processing.

We have identified certain challenges with the architecture that need to be addressed. One significant challenge arises when using small-scale neural networks, where the incoming event only updates a few neurons. In this case, the number of operations in NPEs is not significant enough to compensate for the overheads. In this case, the per-event preprocessing in RISC-V takes more time than the task's execution. This pushes us back to the domain where RISC-V was the main bottleneck of energy and latency.

This situation becomes more severe when using depth-first convolution, as it requires more complex preprocessing. Therefore, we have observed that for CNN layers with small channel sizes, a considerable amount of time/energy goes to the RISC-V during inference. This issue can be resolved by introducing specialized depth-first accelerators.

Additionally, new accelerators need to be designed to handle operations for on-device learning and new types of neural networks, such as transformers. Exploiting weight sparsity is also an ongoing work that needs to be added to SENECA. Weight sparsity exploitation is not trivial when using a simple synchronous vector of NPEs.

7 Lessons learned and design guidelines

The SENECA project has progressed from a simple array of RISC-V cores with a NoC to a richer architecture that includes Neural Processing Elements (NPEs), a loop controller, and several software optimizations. In this section, we distill the main lessons learned into a set of design guidelines for future digital neuromorphic processors.

7.1 Core and NoC design

- **Start from a flexible core.** A tiny general-purpose core (e.g., RISC-V) provides a fast path to a working neuromorphic platform and makes it easy to explore new neuron models, network topologies, and learning rules. Even if such a core is not optimal for a specific benchmark, it is an excellent vehicle for architectural exploration.
- **Keep the NoC simple.** For the workloads we considered, the NoC was never a performance or energy bottleneck. Event sparsity and high operation density per spike imply that the cost of moving a packet is much smaller than the cost of processing it. A simple mesh with source-based routing and modest routing tables was sufficient in our experiments.
- **Exploit time-multiplexing judiciously.** Time-multiplexing many logical neurons onto one physical core improves area efficiency, but reduces performance and energy efficiency. Designers should tune the time-multiplexing ratio to balance silicon area, throughput, and latency for their target application.

7.2 Hardware acceleration

- **Add accelerators incrementally.** Comparing SENECA(V1) with Loihi and SpiNNaker2 showed that dedicated neural accelerators can dramatically improve performance and energy efficiency, especially for dense workloads such as fully connected and convolutional layers. Adding NPEs in SENECA(V2) closed most of the gap while preserving programmability.
- **Separate control and computation.** Introducing a loop controller to orchestrate the NPEs reduced instruction-fetch and control overhead on the RISC-V core, improving system efficiency. A hierarchical control structure (RISC-V → loop controller → NPEs) is a practical pattern for future designs.
- **Target accelerators to dominant kernels.** In our benchmarks, vector-matrix and convolution operations dominated the computation. Focusing accelerators on these kernels provided the largest benefits, while leaving pre- and post-processing on the general-purpose core.

7.3 Software and mapping

- **Exploit event-driven sparsity in software.** Techniques such as spike grouping can significantly reduce control overhead for fully connected layers by processing multiple events together on the same data. These optimizations are often available “for free” once the architecture is flexible enough.

- **Use depth-first convolution to reduce memory footprint.** Event-driven depth-first convolution allows processing CNNs with substantially lower memory requirements than conventional layer-by-layer schemes. In our experiments, this led to orders-of-magnitude reductions in activation memory, which is critical for on-chip deployment.
- **Leverage architectural flexibility for complex models.** Implementing hard-attention style networks for high-resolution event-based vision required substantial pre- and post-processing around the core CNN. The flexibility of the SENECA architecture made it possible to integrate the entire pipeline, leading to both energy and latency improvements without sacrificing accuracy.

7.4 Open challenges

- **Efficiency for small workloads.** For small neural networks or CNN layers with few channels, the per-event preprocessing on the RISC-V core can dominate the total cost, even in the presence of NPEs. Dedicated accelerators for depth-first convolution and other control-heavy patterns are a promising direction.
- **Support for new model families.** Extending digital neuromorphic processors to on-device learning and transformer-style architectures will likely require new accelerator blocks and execution models that go beyond the current focus on spiking and convolutional networks.
- **Weight sparsity and advanced memories.** Exploiting weight sparsity efficiently with synchronous vectors of NPEs remains non-trivial and is an active topic in SENECA. Integrating new memory technologies and 3D architectures offers additional opportunities to reduce data movement and further improve energy efficiency.

These guidelines are not exhaustive, but they summarize the main architectural insights gained from building and iterating on the SENECA platform. We hope they can serve as a practical starting point for students and designers who wish to develop their own digital neuromorphic processors.

Acknowledgments

The results used in this tutorial were extracted from several published papers during the SENECA project. They are the result of the hard work of many colleagues, students and co-authors.

References

- [1] Rodrigo Quiroga and Gabriel Kreiman. Measuring sparseness in the brain: comment on bowers (2009). 2010.
- [2] Giacomo Indiveri and Shih-Chii Liu. Memory and information processing in neuromorphic systems. *Proceedings of the IEEE*, 103(8):1379–1397, 2015.
- [3] Dennis V Christensen, Regina Dittmann, Bernabe Linares-Barranco, Abu Sebastian, Manuel Le Gallo, Andrea Redaelli, Stefan Slesazeck, Thomas Mikolajick, Sabina Spiga, Stephan Menzel, et al. 2022 roadmap on neuromorphic computing and engineering. *Neuromorphic Computing and Engineering*, 2(2):022501, 2022.
- [4] Catherine D Schuman, Shruti R Kulkarni, Maryam Parsa, J Parker Mitchell, Bill Kay, et al. Opportunities for neuromorphic computing algorithms and applications. *Nature Computational Science*, 2(1):10–19, 2022.
- [5] Steve Furber. Digital neuromorphic technology: current and future prospects. *National Science Review*, page nwad283, 2023.
- [6] Jiacong Sun, Pouya Houshmand, and Marian Verhelst. Analog or digital in-memory computing? benchmarking through quantitative modeling. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 1–9. IEEE, 2023.
- [7] Manuel Le Gallo, Riduan Khaddam-Aljameh, Milos Stanisavljevic, Athanasios Vasilopoulos, Benedikt Kersting, Martino Dazzi, Geethan Karunaratne, Matthias Brändli, Abhairaj Singh, Silvia M Mueller, et al. A 64-core mixed-signal in-memory compute chip based on phase-change memory for deep neural network inference. *Nature Electronics*, 6(9):680–693, 2023.
- [8] Steve B Furber, Francesco Galluppi, Steve Temple, and Luis A Plana. The spinnaker project. *Proceedings of the IEEE*, 102(5):652–665, 2014.
- [9] Christian Mayr, Sebastian Hoeppner, and Steve Furber. Spinnaker 2: A 10 million core processor system for brain simulation and machine learning. *arXiv preprint arXiv:1911.02385*, 2019.

- [10] 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.
- [11] Garrick Orchard, E Paxon Frady, Daniel Ben Dayan Rubin, Sophia Sanborn, Sumit Bam Shrestha, Friedrich T Sommer, and Mike Davies. Efficient neuromorphic signal processing with loihi 2. In *2021 IEEE Workshop on Signal Processing Systems (SiPS)*, pages 254–259. IEEE, 2021.
- [12] Filipp Akopyan, Jun Sawada, Andrew Cassidy, Rodrigo Alvarez-Icaza, John Arthur, Paul Merolla, Nabil Imam, Yutaka Nakamura, Pallab Datta, Gi-Joon Nam, 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*, 34(10):1537–1557, 2015.
- [13] Dharmendra S Modha, Filipp Akopyan, Alexander Andreopoulos, Rathinakumar Appuswamy, John V Arthur, Andrew S Cassidy, Pallab Datta, Michael V DeBole, Steven K Esser, Carlos Ortega Otero, et al. Neural inference at the frontier of energy, space, and time. *Science*, 382(6668):329–335, 2023.
- [14] Jan Stuijt, Manolis Sifalakis, Amirreza Yousefzadeh, and Federico Corradi. μ brain: An event-driven and fully synthesizable architecture for spiking neural networks. *Frontiers in neuroscience*, 15:664208, 2021.
- [15] Amirreza Yousefzadeh, Gert-Jan Van Schaik, Mohammad Tahghighi, Paul Detterer, Stefano Traferro, Martijn Hijdra, Jan Stuijt, Federico Corradi, Manolis Sifalakis, and Mario Konijnenburg. Seneca: Scalable energy-efficient neuromorphic computer architecture. In *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pages 371–374. IEEE, 2022.
- [16] Orlando Moreira, Amirreza Yousefzadeh, Fabian Chersi, Gokturk Cinserin, Rik-Jan Zwartenkot, Ajay Kapoor, Peng Qiao, Peter Kievits, Mina Khoei, Louis Rouillard, et al. Neuronflow: a neuromorphic processor architecture for live ai applications. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 840–845. IEEE, 2020.
- [17] Ole Richter, Yannan Xing, Michele De Marchi, Carsten Nielsen, Merkourios Katsimpris, Roberto Cattaneo, Yudi Ren, Qian Liu, Sadique Sheik, Tugba Demirci, et al. Speck: A smart event-based vision sensor with a low latency 327k neuron convolutional neuronal network processing pipeline. *arXiv preprint arXiv:2304.06793*, 2023.
- [18] Mike Demler. Brainchip akida is a fast learner, spiking-neural-network processor identifies patterns in unlabeled data. *Microprocessor Report*, 2019.
- [19] Maarten Molendijk, Kanishkan Vadivel, Federico Corradi, Gert-Jan van Schaik, Amirreza Yousefzadeh, and Henk Corporaal. Benchmarking the epiphany processor as a reference neuromorphic architecture. In *Industrial Artificial Intelligence Technologies and Applications*, pages 21–34. 2022.
- [20] Amirreza Yousefzadeh, Jan Stuijt, Martijn Hijdra, Hsiao-Hsuan Liu, Anteneh Gebregiorgis, Abhairaj Singh, Said Hamdioui, and Francky Catthoor. Energy-efficient in-memory address calculation. *ACM Transactions on Architecture and Code Optimization (TACO)*, 19(4):1–16, 2022.
- [21] Guangzhi Tang, Kanishkan Vadivel, Yingfu Xu, Refik Bilgic, Kevin Shidqi, Paul Detterer, Stefano Traferro, Mario Konijnenburg, Manolis Sifalakis, Gert-Jan van Schaik, et al. Seneca: building a fully digital neuromorphic processor, design trade-offs and challenges. *Frontiers in Neuroscience*, 17, 2023.
- [22] Guangzhi Tang, Ali Safa, Kevin Shidqi, Paul Detterer, Stefano Traferro, Mario Konijnenburg, Manolis Sifalakis, Gert-Jan van Schaik, and Amirreza Yousefzadeh. Open the box of digital neuromorphic processor: Towards effective algorithm-hardware co-design. *arXiv preprint arXiv:2303.15224*, 2023.
- [23] Yingfu Xu and et al. Optimizing event-based neural networks on digital neuromorphic architecture: A comprehensive design space exploration. *Frontiers in Neuroscience*, 2024.
- [24] Technology and hardware for neuromorphic computing. <https://cordis.europa.eu/project/id/826655>.
- [25] Ai for new devices and technologies at the edge. <https://cordis.europa.eu/project/id/876925>.
- [26] Computation-in-memory architecture based on resistive devices. <https://cordis.europa.eu/project/id/780215>.
- [27] Distributed artificial intelligent systems. <https://cordis.europa.eu/project/id/101007273>.
- [28] Memory technologies with multi-scale time constants for neuromorphic architectures. <https://cordis.europa.eu/project/id/871371>.
- [29] Reconfigurable heterogeneous highly parallel processing platform for safe and secure ai. <https://cordis.europa.eu/project/id/101097224>.
- [30] Open source deep learning platform dedicated to embedded hardware and europe. <https://cordis.europa.eu/project/id/101112268>.

- [31] Ultra-energy efficient and secure neuromorphic sensing and processing at the endpoint. <https://cordis.europa.eu/project/id/101070679>.
- [32] Neural magic scales up mlperfTM inference v3.0 performance with demonstrated power efficiency; no gpus needed. <https://neuralmagic.com/blog/neural-magic-scales-up-mlperf-inference-performance-with-demonstrated-power-efficiency-no-gpus-needed>
- [33] James C Knight and Thomas Nowotny. Gpus outperform current hpc and neuromorphic solutions in terms of speed and energy when simulating a highly-connected cortical model. *Frontiers in neuroscience*, 12:941, 2018.
- [34] Beidi Chen, Tharun Medini, James Farwell, Charlie Tai, Anshumali Shrivastava, et al. Slide: In defense of smart algorithms over hardware acceleration for large-scale deep learning systems. *Proceedings of Machine Learning and Systems*, 2:291–306, 2020.
- [35] Rohit Shukla, Mikko Lipasti, Brian Van Essen, Adam Moody, and Naoya Maruyama. Remodel: Rethinking deep cnn models to detect and count on a neurosynaptic system. *Frontiers in neuroscience*, 13:4, 2019.
- [36] Amirreza Yousefzadeh, Mikel Soto, Teresa Serrano-Gotarredona, Francesco Galluppi, Luis Plana, Steve Furber, and Bernabe Linares-Barranco. Performance comparison of time-step-driven versus event-driven neural state update approaches in spinnaker. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–4. IEEE, 2018.
- [37] Florian Kelber, Binyi Wu, Bernhard Vogginger, Johannes Partzsch, Chen Liu, Marco Stolba, and Christian Mayr. Mapping deep neural networks on spinnaker2. In *Proceedings of the 2020 Annual Neuro-Inspired Computational Elements Workshop*, pages 1–3, 2020.
- [38] Amirhossein Rostami, Bernhard Vogginger, Yixin Yan, and Christian G Mayr. E-prop on spinnaker 2: Exploring online learning in spiking rnns on neuromorphic hardware. *Frontiers in Neuroscience*, 16:1018006, 2022.
- [39] Amirreza Yousefzadeh, Mirosław Jabłoński, Taras Iakymchuk, Alejandro Linares-Barranco, Alfredo Rosado, Luis A Plana, Steve Temple, Teresa Serrano-Gotarredona, Steve B Furber, and Bernabé Linares-Barranco. On multiple aer handshaking channels over high-speed bit-serial bidirectional lvds links with flow-control and clock-correction on commercial fpgas for scalable neuromorphic systems. *IEEE transactions on biomedical circuits and systems*, 11(5):1133–1147, 2017.
- [40] Ravenoc - configurable network-on-chip. <https://github.com/aignacio/ravenoc>.
- [41] Michael K Papamichael and James C Hoe. The connect network-on-chip generator. *Computer*, 48(12):72–79, 2015.
- [42] Awesome-mesh. <https://github.com/moarpepes/awesome-mesh>.
- [43] Peter Blouw, Xuan Choo, Eric Hunsberger, and Chris Eliasmith. Benchmarking keyword spotting efficiency on neuromorphic hardware. In *Proceedings of the 7th annual neuro-inspired computational elements workshop*, pages 1–8, 2019.
- [44] Yixin Yan, Terrence C Stewart, Xuan Choo, Bernhard Vogginger, Johannes Partzsch, Sebastian Höppner, Florian Kelber, Chris Eliasmith, Steve Furber, and Christian Mayr. Comparing loihi with a spinnaker 2 prototype on low-latency keyword spotting and adaptive robotic control. *Neuromorphic Computing and Engineering*, 1(1):014002, 2021.
- [45] Athanasios Papadopoulos, Paweł Korus, and Nasir Memon. Hard-attention for scalable image classification. *Advances in Neural Information Processing Systems*, 34:14694–14707, 2021.