2

Benchmarking the Epiphany Processor as a Reference Neuromorphic Architecture

Maarten Molendijk[1,2], Kanishkan Vadivel[2], Federico Corradi[2,1], Gert-Jan van Schaik[1], Amirreza Yousefzadeh[1], and Henk Corporaal[2]

[1]imec, Netherlands
[2]Technical University of Eindhoven, Netherlands

Abstract

This short article explains why the Epiphany architecture is a proper reference for digital large-scale neuromorphic design. We compare the Epiphany architecture with several modern digital neuromorphic processors. We show the result of mapping the binary LeNet-5 neural network into few modern neuromorphic architectures and demonstrate the efficient use of memory in Epiphany. Finally, we show the results of our benchmarking experiments with Epiphany and propose a few suggestions to improve the architecture for neuromorphic applications. Epiphany can update a neuron on average in 120ns which is enough for many real-time neuromorphic applications.

Keywords: neuromorphic processor, spiking neural network, bio-inspired processing, artificial intelligence, edge AI.

2.1 Introduction and Background

Neuromorphic sensing and computing systems mimic the functions and the computational primitives of the nervous systems. Nevertheless, state-of-the-art Deep Neural Networks (DNNs) have exceeded the accuracy of biological brains (including the human brain) in specific tasks like video/audio processing, decision-making, planning and playing games. However, all of these tasks are done without considering one of the main restrictions in bio-evolution, the "energy consumption". The biological restrictions pushed the evolution toward power-efficient algorithms and architectures. The human brain is an extreme example that consumes a considerable portion (around 20%) of the human body's energy while it has less than 3% of the total weight.

Even though the elements of the biological fabric in the brain are not as fast and arguably as power efficient as our modern silicon technologies, no computing platform can get close to the compute efficiency of the biological brain for processing natural signals. The brain is a perfect example of algorithm-hardware co-optimization. As mentioned, the ultimate goal of bio-inspired processing is to process the raw sensory data with the minimum amount of power consumption.

The Epiphany architecture was first introduced back in 2009 [1] as a high-performance energy-efficient many-core architecture suitable for real-time embedded systems. Epiphany's architecture contains many RISC processor cores connected with a packet-based mesh Network-On-Chip (NoC). Figure 5.1 shows the big picture of the Epiphany's architecture. This architecture is different from the mainstream von-Neumann type multi-core processors since in Epiphany, the cores are connected directly via a NoC

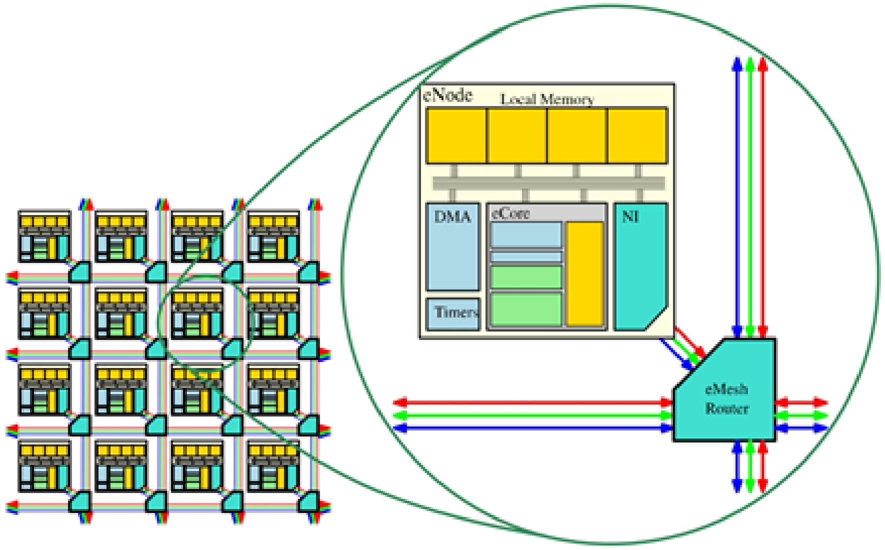

Figure 2.1 Overall scalable architecture of Epiphany-III [1].

without using a single shared memory to communicate. The mesh packet switch network in Epiphany results in highly efficient local data movement between neighbouring processors. However, it introduces a possible non-deterministic behaviour as the order of the packets in the mesh network is not guaranteed. Despite implementing a synchronization mechanism, the RISC processors work individually, and the architecture is not designed for strict synchronous execution (since it harms the scalability feature). Hence, programming epiphany with a conventional programming model is challenging. Therefore, Epiphany has never gained enough attention in the mainstream general-purpose processor market.

In 2011, Adapteva, a kick-starter company, introduced the first processor based on the Epiphany architecture (Figure 5.2). It contains a 16 RISC core Epiphany chip, expandable to be used in a 256 multi-chip platform (4096 cores in total). The chip is implemented in a 65nm technology node and consumes less than 2 Watts. A few months later, Adapteva introduced a bigger version of the processor with 64 cores. The latest version of the processor [2] was announced in 2016 and contains 1024 cores.

Despite the failure of Epiphany in the general-purpose compute domain, it has a very similar architecture to the neuromorphic processors which were introduced a few years later (e.g., SpiNNaker in 2013 [3], IBM TrueNorth in 2015 [4], Intel Loihi in 2018 [5], BrainChip AKIDA in 2019 [6] and GML NeuronFlow in 2020 [7]). The main goal of neuromorphic engineering is to build a brain-inspired processor to execute variations of Spiking Neural

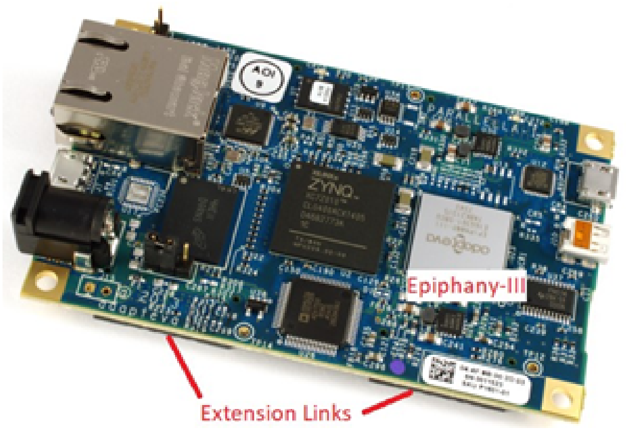

Figure 2.2 Adapteva launched an $99 Epiphany-III based single board computer as their first product.

Network (SNN) algorithms for real-time sensory signal processing. Programming to implement neural networks using conventional programming models and compilers is difficult (and inefficient), which resulted in a new paradigm shift in the programming models. A neural network usually contains neurons (as the processing unit) and weighted synapses/axons to connect the neurons in a graph like architecture. Therefore, several new graph-based programming models (like TensorFlow from Google and PyTorch from Facebook) are introduced to efficiently execute such applications.

The architecture is made up of eNode processing cores and eMesh routers to build connectivity networks. Each eNode contains a RISC processor (1GHz, with an integer and a floating-point ALU and a 64-word register file), 4 memory banks (each $64b \times 1024w$) to store data (like synaptic weights and neuron states), and the instructions (like the neuron model) locally, a Network Interface (NI), Direct Memory Access (DMA) to handle incoming/outgoing packets, a few general timers (for example to implement periodic leakage) and a memory BUS interconnect which allows access to each memory bank simultaneously. The eMesh routers handle 3 separate networks. A high-performance network for sending one packet of data (spike) to the other cores with the maximum speed of one packet per clock cycle) and two lower performance networks (one for reading from another core's memory and one for off-chip communication) are introduced to make the programming easier. These programming models allow for easy splitting of the computational load over several processing units and mapping synaptic connectivity into the NoC. Therefore, they are a good fit for architectures like Epiphany.

Like the other neuromorphic architectures, Epiphany is extremely scalable, performs near memory processing, is optimized for local data movement (local connectivity) and asynchronous processing. The eMesh network is flexible enough to time multiplex any arbitrary synaptic connections. Besides, the eCores are flexible enough to implement different neuron models. Most importantly, the architecture is straightforward, which allows easy design space exploration and benchmarking. Finally, unlike all the other neuromorphic platforms it is accessible and affordable which makes it a suitable platform for benchmarking new neuromorphic platforms and innovative ideas.

2.2 Comparison with a Few Well-Known Digital Neuromorphic Platforms

Probably the SpiNNaker architecture [3] (introduced in 2013) is the most similar neuromorphic platform to Epiphany. SpiNNaker contains several ARM cores as the processing units connected through an advanced asynchronous packet-switched network.

Therefore, like Epiphany, the processing core is very flexible and can implement different neuron models with various mapping schemes. Unlike Epiphany, each SpiNNaker chip contains only one router, with a higher complexity level than the Epiphany's eMesh router. The SpiNNaker's NoC allows for multi-casting (using source-based addressing with a programmable routing table), which is an optimization on top of the plain mesh NoC.

Contrary to SpiNNaker, IBM TrueNorth [4] (introduced in 2015) uses a plain mesh packet-switched network but with optimized processing cores. Therefore, the NoC in IBM TrueNorth is very similar to the Epiphany. Each core in the TrueNorth architecture is fixed to emulate 256 neurons, and each neuron with 256 input synapse (a crossbar architecture) and a single output axon (connectable to 256 neurons in any other core). The cores update all the neurons every $1ms$. The synaptic weights are limited to be binary. This optimized processing core resulted in an ultra-low-power neuron update (about 26pJ). However, having such constrained cores makes the deployment of many neuromorphic applications either impossible or inefficient.

In Intel Loihi [5] (introduced in 2018), the processing cores are more flexible than TrueNorth, and the interconnect is a simple packet-switched mesh. Each core in Loihi emulates 1024 neurons with a fixed neuron model, but the number of input synapses to each neuron and their resolution is flexible (1kb of synaptic memory per neuron). The number of output axons is also flexible, and one axon can be shared among many neurons. Loihi cores accelerate a bio-inspired learning algorithm. The cost of these flexibility is having a higher neuron update energy (about 80pJ) in comparison with the TrueNorth (while using a better technology node).

In addition to the three previous research platforms, many companies started to build neuromorphic processors for commercial purposes. For example, BrainChip AKIDA (introduced in 2019) and GML NeuronFlow (introduced in 2020) have similar architectures to Loihi.

One of the features in the research of neuromorphic chips is asynchronous processing and communication. In Loihi, the asynchronization level is inside the core's logic blocks. In SpiNNaker and TrueNorth, the cores are working asynchronously with each other in a Globally Asynchronous Locally Synchronous (GALS) structure. In Epiphany, NeuronFlow, and AKIDA, the asynchronousity level is pushed toward the boundaries of the chip (asynchronous chip to chip connectivity). Despite where is the boundary of asynchronousity, it is essential for scalability.

Nevertheless, in all the mentioned architectures, the cores still work individually with each other. Therefore, the implementation of a globally synchronous algorithm is not optimal in neuromorphic architectures.

2.3 Major Challenges in Neuromorphic Architectures

Since neuromorphic architecture design aims to follow the principles of bio-inspired processing mechanism in the available nano-electronic technologies, facing several challenges that result from the platform constraints is normal. Many innovative schemes have been introduced to overcome the difficulties of developing neuromorphic technology and spiking neural network algorithm design. These challenges are discussed below.

2.3.1 Memory Allocation

One of the main challenges in neuromorphic design is the available amount of local memory near or inside the processing element where the data is consumed. In the brain, there is no separation between memory and computation. This feature eliminates a) the memory bandwidth bottleneck issue and b) the high cost of data movement between the processing and a far-away memory block. To mimic this feature, neuromorphic chips use distributed memories near or inside the processing elements (to keep the synaptic weights and neuron states close to the processing unit). However, the onchip memory made by using the conventional SRAM memory technology is not area-efficient (compared to DRAM and Flash) and therefore expensive. Besides using a new denser memory technology [8], one of the solutions to overcome this problem is the proper memory management and maximum reuse of the memory bits.

The important elements to be stored in each processing core are the spike queue(s), synaptic weights, neuron states, and axons (destination addresses). The depth and width of these memories heavily depend on the executable neuron model and supported connectivities. Table 5.1 shows the memory allocations in different neuromorphic chips.

Flexibility in the memory allocations allows for optimized mapping of a neural network in the processor. Different neurons in the neural network have a different number of inputs/outputs and different amounts of activities. Some neuromorphic chips allow flexible parameter resolution to trade-off accuracy and SNN size [5]. Since the range of the parameters is sometimes more important than the resolutions of the parameters, using smaller floating-point representations (like BrainFloat16 [9]) may results in better accuracy

Table 2.1 Memory fragmentations in some digital large-scale neuromorphic chips

Architecture	Total Memory	Spike queue	Neurons	Synapses	Axons
TrueNorth [4]	110kb fixed allocations (426b per neuron)	256*16b	256 fixed neuron type	256*256*1b	256*26b (1 per neuron)
Loihi [5]	2Mb	N/A	1024 fixed neuron type	1Mb Flexible resolution (1b to 9b) Weight sharing	4k flexibly shared
NeuronFlow [7]	120kb	N/A	1024 few neuron types	1k*8b Weight sharing	1k flexibly shared
SpiNNaker [3]	768kb 256kb instruction memory 512kb data memory	Flexible	Flexible pro-grammable neuron type	Flexible resolution only integer ALU	Flexible
Epiphany [1]	256kb in 4 banks, each with 64b data width	Flexible	Flexible pro-grammable neuron type	Flexible resolution Int/float ALUs	Flexible

and power/area performance than using a larger inter (like int32) format. Therefore, it is possible to trade-off the memory footprint and complexity of the operations.

Another method to use the memory space efficiently is to store a compressed form of the parameters when there is a high amount of sparsity in the synaptic weight tensor [10]. Weight sharing is another method to efficiently use the memory for spiking Convolutional Neural Networks (sCNN) [5] [7].

The Epiphany contains 256kb of memory per core and is the most flexible architecture in Table 5.1. In the table N/A means we could not find the data publicly. Axons are the destination core addresses to route spikes from a neuron. All the numbers in this table are for a single processing core inside the mentioned neuromorphic chip. All the above-mentioned schemes can be implemented in Epiphany to optimally use the memory space. To demonstrate

Table 2.2 Mapping LeNet-5 neural network (with binary weights) in different neuromorphic architectures

Architecture	**Number of used neurons**	**Average number of synapses per neuron**	**Number of individual stored weights**	**Number of used cores**	**Total memory used**
LeNet-5 (before deployment)	6518	144.5	61k	-	-
TrueNorth [4]	40k	256	941k (144.5×6518)	155	17Mb
Loihi [5]	6518	1024	61k	7	14Mb
NeuronFlow [7]	6518	1024	61k	7	840kb
SpiNNaker [3]	6518	144.5	61k	1	768kb
Epiphany	6518	144.5	61k	1	256kb

the value of flexibility for efficient use of memory, in Table 5.2 we show the result of mapping the binary LeNet-5 [11] into the above-mentioned neuromorphic architecture. The average pooling layers are optimized out in the mapping (as average pooling is a linear operation and does not consume stateful neurons). The mappings are hand optimized with only memory constraint. In TrueNorth, several neurons need to be combined to make a single neuron with enough synapses and axons. Also, since weight-sharing is not used, the weight for each synapse needs to be stored individually. In the flexible architectures, the neuron states are assumed to be 16b, without refractory mechanism and with a single threshold per channel. Mapping in SpiNNaker is done with the "Convnet Optimized Implementation" which is described in [12]. Total memory used is (*number of cores*×*memory per core*).

2.3.2 Efficient Communication

Using a packet to communicate spikes between cores can be very inefficient. A spike packet that carries a single bit of data (spike) contains several bits for the address. For example, a spike packet in SpiNNaker contains 44b of data to communicate a single binary spike in the AER format [13]. There are several possible solutions to reduce the number of bits for communicating spikes. One solution is to use a more complex neuron model (for example [14] and [15]) with a lower firing rate (trading off operation complexity with the number of packets). Another solution is to compress several spikes into one event. For instance, when the destination core for several packets is the same, we can compress them easily in one hyper-packet. Epiphany's packets are fixed in size (104b packet with a 64b payload data) but the format of payload data is programmable.

TrueNorth [4] and NeuronFlow [7] use a relative addressing scheme which allows reducing the number of bits for the destination address in the packet when a limited communication range is acceptable. For example, in a platform with 4096 cores, if the destination address contains only 4b, a core can only communicate with 16 neighbouring cores which might be sufficient for many applications. This results in a saving of 8b per packet (from 12b address in a 4096-cores system to 4b-address). Another method to reduce the number of packets is the multi-casting feature which is used in SpiNNaker [3]. In this case, a core can only send one spike out and this spike will be multicasted in the NoC and near the destination cores. Epiphany uses the basic mesh NoC interconnect which is a shortcoming but contributes to its simplicity.

2.3.3 Mapping SNN onto Hardware

An optimized mapping algorithm can reduce the memory footprint (by performing maximum sharing of parameters), balance the loads in different cores (as not all the neurons in an SNN are equally active) and reduce the core-to-core communications (since it is expensive in terms of power consumption and latency). Having a flexible number of neurons per core and synapses per neuron allows the mapping optimizer to find a better solution. The Epiphany platform can be used to benchmark different mapping algorithms in the neuromorphic domain because of its flexible and unified memory architecture.

2.3.4 On-chip Learning

On-chip learning is supported as a futuristic feature in some neuromorphic chips (like Loihi [5] and AKIDA [16]). However, implementation a hardware acceleration for on-chip learning is challenging. First, because the algorithm domain is very dynamic (experimental), it is difficult to find a suitable algorithm for a wide range of applications. Second, many applications can be pre-trained and only require fine-tuning after deployment. Therefore, the learning acceleration might be used only for a few last layers of the neural network (after general feature extraction layers). Epiphany does not have a hardware accelerated learning engine, but it allows for software implementation of those algorithms and therefore benchmarking the new learning algorithms.

2.3.5 Idle Power Consumption

One of the challenges in event-based neuromorphic processors is the power consumption when the cores are in the idle state (no event to be processed). It is reported that around 30% of power consumption for TrueNorth [4] and Loihi [17] is the idle power. This can be even worse when the application is sparser. It is possible to reduce idle power consumption by using asynchronous design or clock gating when no input spike is processed. Also, using a non-volatile memory technology helps to reduce leakage in the memory cells (since neuromorphic chips are mostly memory dominant). Epiphany supports dynamic clock gating for the processing cores. In this case, a core can only wake up by an interrupt (for example, receiving a new input packet).

2.4 Measurements from Epiphany

In this section, we present some of our measurements using the Epiphany processor to provide a sense of its performance for possible neuromorphic applications.

We implemented a neural network with a Leaky integrate and Fire (LIF) neuron model with different parameters in Epiphany and measured processing time for different processes, which can be used as a reference for assessment of Epiphany when one wants to use it as a neuromorphic processor. Our measurements in this work consider the processing time (no power measurement) and are performed using the hardware timers inside the cores.

The compiled instructions (not hand-optimized) for our experiments took around 52kb of the used cores' memory. Since the instruction code is almost similar for all the cores, it will be copied in each core's memory. It is therefore recommended to use bigger cores (more memory), so instruction memory takes only a small fraction of the total memory and is used for a higher number of neurons.

Figure 5.3 shows a flowchart of our neuron model with the processing cycle time attached to each block, where N is the number of neurons, F is the number of firings, X is the neuron state, W is the synaptic weight, Thr is the firing threshold, Time is the current time (read from Timer), LFT is the last firing time (stored per neuron), Ref is the refractory time and LR is the leak rate.

An input spike enters the eCore through the DMA and interrupts the RISC core. Then a process handles this spike and puts it in a FIFO (made with

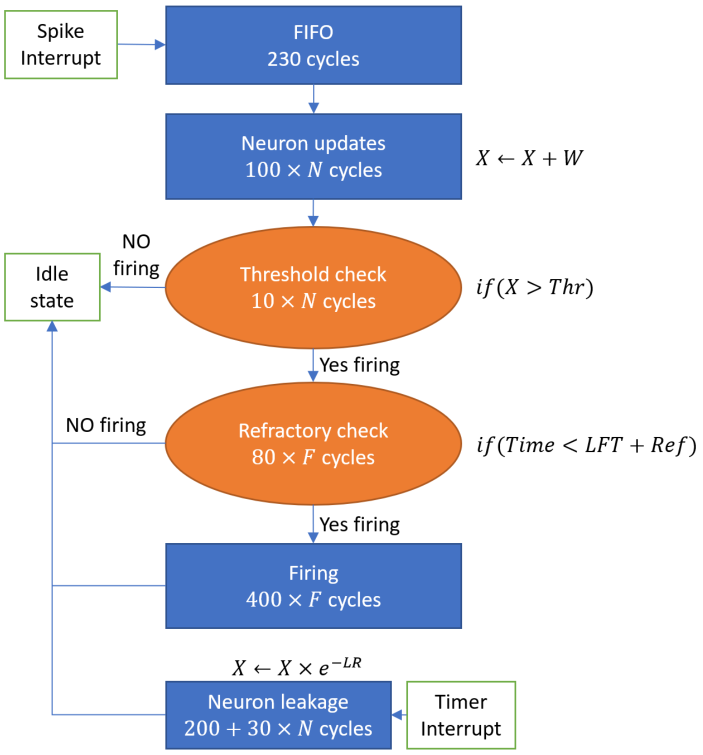
Spike Interrupt
FIFO 230 cycles
Neuron updates $100 \times N$ cycles
$X \leftarrow X + W$
NO firing
Idle state
Threshold check $10 \times N$ cycles
$if(X > Thr)$
Yes firing
NO firing
Refractory check $80 \times F$ cycles
$if(Time < LFT + Ref)$
Yes firing
Firing $400 \times F$ cycles
$X \leftarrow X \times e^{-LR}$
Neuron leakage $200 + 30 \times N$ cycles
Timer Interrupt

Figure 2.3 Flow chart of processing a LIF neuron with processing time measured in Epiphany.

a software process). Thereafter, the target neurons will get updated. After updates, the threshold of neurons is checked, and the refractory check is executed for each firing. If both checks pass, the firing process starts, and the RISC core commands to the DMA to transmit a spike packet. Membrane leakage is also an independent process that starts with a timer interrupt.

Each cycle takes 1ns when using a 1GHz clock frequency. For example, processing a single spike from the first convolutional layer of LeNet-5 to the second convolutional layer requires to update $16 \times 5 \times 5$ neurons. When the second layer is implemented in a core and 1% of the updated neurons fire, the processing time takes around 46us. The leak process on all these 400 neurons takes around 12us. Our measurements are averaged over many experiments and therefore the numbers in this figure are reasonable estimations. Since the neuron model is programmable, one may decide to remove some of the components (like refractory) or make it more complex (for example by introduction of an individual threshold for every neuron)

In Figure 5.3 we showed that updating a neuron with a single spike takes around 120ns on average. We know that TrueNorth can update all of the neurons in a core every 1ms, to be suitable for real-time neuromorphic applications. If we assume a reasonable sparsity in the input spikes in each time-step (32 input spikes per neuron with 256 input synapses), with 120ns update time, Epiphany can also process the 256 neurons in less than 1ms.

2.5 Conclusion

This article demonstrates that the Epiphany processor is compatible with neuromorphic computing. Overall, it has a similar architecture to the well-known neuromorphic processors and is flexible enough for the implementation of new ideas. Unlike Epiphany, all the mentioned neuromorphic processors contain optimized elements that add complexity to the architecture and make it less flexible to be a reference benchmarking architecture (flexibility vs efficiency trade-off). For example, having a fixed number of neurons per core (in TrueNorth, Loihi, and NeuronFlow) does not allow for optimized resource management during mapping. Also, having an accelerated learning mechanism (in Loihi) may be unnecessary for many applications. Additionally, suppose one wants to know the performance improvement of the SpiNNaker processor due to its optimized NoC. In that case, Epiphany is an excellent platform to compare to, due to its simplicity and flexibility.

As mentioned, not having any accelerator makes the epiphany less efficient compared to the accelerated architectures (like Loihi), but it increases its value for benchmarking the performance improvement of any accelerators.

We have implemented a neural network system and measured the processing time for different components of the LIF neuron model. It is already visible that some small improvements (like having a hardware FIFO) can improve the performance of the system. Increasing the size of the core results in better memory saving, but the designer should scale the performance of the cores as well (by the implementation of the schemes like multi-threading [5] and SIMD, as it is implemented in the forthcoming SpiNNaker2.0 platform [18]). Other improvements (like adding a more suitable interconnect) can be examined and is a topic for our future research. All source code used to benchmark the system and perform hands-on experiments is freely available upon request ({amirreza.yousefzadeh, gert-jan.vanschaik}@imec.nl)

Acknowledgements

This technology is partially funded and initiated by the Netherlands and European Union's Horizon 2020 research and innovation projects **TEMPO** (ECSEL Joint Undertaking under grant agreement No 826655) and **ANDANTE** (ECSEL Joint Undertaking under grant agreement No 876925).

References

[1] A. Olofsson, et al., Kickstarting high-performance energy-efficient manycore architectures with epiphany, in 2014 48th Asilomar Conference on Signals, Systems and Computers, IEEE, 2014, pp. 1719–1726.

[2] A. Olofsson, Epiphany-v: A 1024 processor 64-bit risc system-on-chip, arXiv preprint arXiv:1610.01832.

[3] E. Painkras, et al., Spinnaker: A 1-w 18-core system-on-chip for massively-parallel neural network simulation, IEEE Journal of Solid-State Circuits 48 (8) (2013) 1943–1953.

[4] F. Akopyan, 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) (2015) 1537–1557.

[5] M. Davies, et al., Loihi: A neuromorphic manycore processor with on-chip learning, IEEE Micro 38 (1) (2018) 82–99.

[6] M. Demler, Brainchip akida is a fast learner, spiking-neural-network processor identifies patterns in unlabeled data, Microprocessor Report (2019).

[7] O. Moreira, et al., Neuronflow: a neuromorphic processor architecture for live ai applications, in 2020 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, 2020, pp. 840–845.

[8] E. Miranda, J. Suñé, Memristors for neuromorphic circuits and artificial intelligence applications (2020).

[9] N. P. Jouppi, et al., In-datacenter performance analysis of a tensor processing unit, in: Proceedings of the 44th Annual International Symposium on Computer Architecture, 2017, pp. 1–12.

[10] V. Sze, Y.-H. Chen, T.-J. Yang, J. S. Emer, Efficient processing of deep neural networks, Synthesis Lectures on Computer Architecture 15 (2) (2020) 1–341.

[11] Y. LeCun, et al., Lenet-5, convolutional neural networks, URL: http://yann. lecun. com/exdb/lenet 20 (5) (2015) 14.
[12] A. Yousefzadeh, et al., 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), IEEE, 2018, pp. 1–4.
[13] A. Yousefzadeh, et al., Fast predictive handshaking in synchronous FPGAs for fully asynchronous multisymbol chip links: Application to spinnaker 2-of-7 links, IEEE Transactions on Circuits and Systems II: Express Briefs 63 (8) (2016) 763–767.
[14] A. Yousefzadeh, et al., Asynchronous spiking neurons, the natural key to exploit temporal sparsity, IEEE Journal on Emerging and Selected Topics in Circuits and Systems 9 (4) (2019) 668–678. doi:10.1109/JETCAS.2019.2951121.
[15] B. Yin, et al., Effective and efficient computation with multiple-timescale spiking recurrent neural networks, in International Conference on Neuromorphic Systems 2020, ICONS 2020, Association for Computing Machinery, New York, NY, USA, 2020. doi:10.1145/3407197.3407225.
[16] S. Thorpe, et al., Method, digital electronic circuit, and system for unsupervised detection of repeating patterns in a series of events, US Patent App. 16/349,248 (Sep. 19, 2019).
[17] P. Blouw, et al., Benchmarking keyword spotting efficiency on neuromorphic hardware, in: Proceedings of the 7th Annual Neuro-inspired Computational Elements Workshop, 2019, pp. 1–8.
[18] C. Mayr, S. Höppner, and S. Furber (2019). SpiNNaker 2: a 10 million core processor system for brain simulation and machine learning-keynote presentation. In Communicating Process Architectures 2017 & 2018 277-280, IOS Press, 2019.