Journal of Systems Architecture 77 (2017) 43–51

Contents lists available at ScienceDirect

Journal of Systems Architecture

journal homepage: www.elsevier.com/locate/sysarc

Darwin: A neuromorphic hardware co-processor based on spiking neural networks

De Ma[a,b], Juncheng Shen[b], Zonghua Gu[b,*], Ming Zhang[b], Xiaolei Zhu[b,*], Xiaoqiang Xu[b], Qi Xu[b], Yangjing Shen[a], Gang Pan[b]

[a] MOE Key Laboratory of RF Circuits and Systems, Hangzhou Dianzi University, 310018, China
[b] College of Computer Science/College of Microelectronics, Zhejiang University, Hangzhou, 310027, China

ARTICLE INFO

Article history:
Received 2 February 2016
Revised 4 June 2016
Accepted 9 January 2017
Available online 17 January 2017

Keywords:
Neuromorphic computing
Spiking neural networks (SNN)
Address-event representation (AER)
Digital VLSI

ABSTRACT

Spiking Neural Network (SNN) is a type of biologically-inspired neural networks that perform information processing based on discrete-time spikes, different from traditional Artificial Neural Network (ANN). Hardware implementation of SNNs is necessary for achieving high-performance and low-power. We present the Darwin Neural Processing Unit (NPU), a highly-configurable neuromorphic hardware co-processor based on SNN implemented with digital logic, supporting a configurable number of neurons, synapses and synaptic delays. The Darwin NPU was fabricated by standard 180 nm CMOS technology with area size of 5×5 mm^2 and 70 MHz clock frequency at the worst case. It consumes 0.84 mW/MHz with 1.8 V power supply for typical applications. Two prototype applications are used to demonstrate the performance and efficiency of the Darwin NPU.

1. Introduction and related work

Spiking Neural Network (SNN) is a type of biologically-inspired neural networks that performs information communication and computation based on discrete-time spikes. When an SNN neuron receives an input spike, its soma's membrane potential is increased momentarily, but gradually drops due to leakage of ion channels. When multiple input spikes are received in rapid succession, the membrane potential may increase to a certain threshold voltage, triggering an output spike to its downstream neurons through connecting synapses. There are multiple possible levels of abstractions for SNN modeling, ranging from the most biologically-realistic Hodgkin-Huxley model to the most simplified Leaky Integrate and Fire (LIF) model, with models of intermediate complexity including the Quadratic Integrate and Fire (QIF), Adaptive Exponential Integrate and Fire (AdEx IF), Izhikevich, FitzHugh-Nagumo, Hindmarsh-Rose, and many others [1].

There are number of well-known software simulators for SNN based on conventional CPUs, including NEST [2], Brian [3], NEURON [4], Nengo [5] and others. Since software simulation can be quite slow, researchers have developed simulation acceleration solutions based on high-performance parallel computing platforms, e.g., the Bluebrain project [6] uses IBM BlueGene/Q supercomputer with 65,536 cores to achieve simulation of large-scale SNN based on multi-compartment Hodgkin-Huxley model; the SpiNNaker system [7] is a custom multicore ARM-based super-computer for real-time and low-power simulation of simplified SNN models such as LIF or Izhikevich. There are also GPU-based simulators, e.g., the CarlSim3 simulator [8] runs on high-performance GPUs to achieve large-scale SNN simulation.

In order to achieve real-time and low-power implementation of SNN simulation, it is necessary to use (digital, analog or mixed-signal) hardware to implement SNN instead of (CPU or GPU-based) software simulation. Hardware implementation of SNN for machine learning applications has the potential to achieve much lower power consumption compared to traditional Artificial Neural Network (ANN) [9]. There are a number of different approaches to developing hardware acceleration of SNN models based on different types of hardware platforms, including digital logic implementation with FPGA or ASIC, e.g., the IBM TrueNorth processor developed in the DARPA SyNAPSE project [9], which supports a maximum number of 1 million neurons and 256 million synapses; or Analog and Mixed-Signal circuit implementation, e.g., the ROLLS processor [10], which supports a maximum of 256 neurons and 128 K synapses with on-line learning algorithms such as Spike Timing Dependent Plasticity (STDP); the NeuroGrid system [11] with the aim of simulating large-scale neural models in real time, and the HICANN wafer-scale system [12] for 10,000-times faster emulation of SNN based on the AdEx IF neuron mode. Besides conventional CMOS-based implementations, there are also

* Corresponding authors.
E-mail addresses: zgu@zju.edu.cn (Z. Gu), zhuxl@vlsi.zju.edu.cn (X. Zhu).

http://dx.doi.org/10.1016/j.sysarc.2017.01.003
1383-7621/

attempts at building systems based on emerging devices such as memristors [13].

For embedded systems, it is typical for neuromorphic processors to be used as co-processors integrated with the CPU in a master-slave configuration; the neuromorphic processor is used to accelerate computation-intensive machine learning algorithms such as image recognition, image segmentation etc., and the CPU is used to run the typical operating systems tasks, including graphical user interface, networking stack, file systems, etc. Compared to special-purpose hardware accelerators designed for a specific function, the neuromorphic co-processor has the advantage of being configurable to support any function that can be implemented with neural networks, which are universal approximators of arbitrary continuous functions. For example, the Qualcomm Zeroth co-processor [14] integrated with the Snagdragon 820A processor implements deep Artificial Neural Networks that can be configured to support any function.

In this paper, we present the *Darwin Neural Processing Unit (NPU)*, a highly-configurable neuromorphic hardware co-processor based on *Leaky Integrate and Fire (LIF)* SNN model [15], implemented with digital logic. It is designed for resource-constrained embedded applications, hence the hardware resource used by the design is very limited. We reduce the computation resource cost by time-multiplexing the physical neuron units, and minimize the memory resource cost by the design of reconfigurable memory subsystem. It has been prototyped on FPGA, and fabricated as ASIC in SMIC's 180 nm process. Since different applications have very different requirements, the Darwin NPU is designed to be highly configurable, with the maximum number of neurons, synapses and synaptic delays all being configurable parameters.

Preliminary results have been reported in our short paper [16]. In this paper, we present more details on the architectural design of the Darwin NPU, including the overall architecture design, main parameters and variables, the off-chip and on-chip memory system design, as well as details on the SNN architecture for the demonstration applications. The rest of the paper is structured as follows: Section 2 presents the neuron model and its optimizations for implementation with digital logic; Section 3 presents the hardware architecture of the Darwin NPU; Section 4 presents two demonstration applications; Finally, Section 5 presents conclusions.

2. The neuron model

The *Leaky Integrate and Fire (LIF)* model is a simplified model of biological neuron, widely used in neuromorphic engineering projects. It represents a good tradeoff between computational complexity and biological realism. The membrane potential V of a LIF neuron is described by the following equation:

$$C_m \frac{dV}{dt} = g_l(V_{rest} - V) + I, \tag{1}$$

Where V_{rest} is the resting membrane potential; $C_m$is the membrane capacitance; $g_l$is the membrane conductance; I is the input current. When the membrane potential V rises up to reach the firing threshold V_{th}, a *spike* (also called an *Action Potential*) is triggered, and V rapidly rises to a large value, then reset to $V = V_{reset}$. Afterwards, there is a refractory period with length of T_{ref}, when the neuron is not responsible to input spikes. At the end of the refractory period, the membrane potential V returns to the resting membrane potential V_{rest}, and starts to be responsive to input spikes again.

To implement the model with digital logic, it is necessary to have a discrete-time version of the LIF model. Consider a post-synaptic neuron with index j, connected to possibly multiple pre-synaptic neurons with indices denoted as i. The membrane potential of neuron j satisfies the following discrete time equations:

$$V_j(t) \leftarrow V_j(t-1)(1 - \Delta t/\tau_m) + \sum_i S_{ij} V_{max} w_{ij}, \tag{2}$$

$$V_j(t) \leftarrow \begin{cases} 0, & if\ t \in [T_f,\ T_f + T_{ref}] \\ H(V_{th} - V_j(t)) \cdot V_j(t), & otherwise \end{cases}, \tag{3}$$

$$S_i(t) \leftarrow H(V_i(t) - V_{th}), \tag{4}$$

where $V_j(t)$ is the membrane potential of neuron j at time step t; V_{max} is a spike's maximum contribution to membrane potential (occurring when $w_{ij} = 1$); the term $\sum_i S_{ij} V_{max} w_{ij}$ denotes the input current I, equal to sum of each input spike current multiplied by the respective synapse weights (we use a per-neuron *Weight-Sum Queue* to store this term at different time steps); Δt is the simulation time step size, with typically value of 0.1 ms; τ_m=C_m/g_l is time constant of the RC circuit model of the cell membrane; $S_{ij} = \{0,\ 1\}$ denotes whether neuron i fires a spike at time step t; V_{max} denotes the maximum voltage change to a neuron caused by receiving an incoming spike; w_{ij} indicates the weight of the synapse that connects pre-synaptic neuron i to post-synaptic neuron j; it is positive if the synapse is excitatory; negative if it is inhibitory; V_{th} is the firing threshold; $H(x) = \{^{1, x \geq 0}_{0, x < 0}$ is the unit step function; V_{rest} and V_{reset} are both assumed to be 0. If the neuron fires an output spike at $t = T_f$, then it remains quiescent for the length of the refractory period during the time interval $[T_f,\ T_f + T_{ref}]$, when its membrane potential stays at $V_{reset} = 0$ and not responsive to input spikes. (The synapse delay does not appear explicitly in Eqs. (2)–(4), but is modeled as a circular buffer, as shown in Fig. 2 later.)

To reduce the computation density, the floating-point variables in Eqs (2)–(4) need to be converted to fixed-point integer variables. We first simplify the status update Eq. (2) by merging parameters. We define the leakage constant $N_{leak} = 1 - \Delta t/\tau_m$, and the equivalent synapse weight $W_i = V_{max} w_{ij}$. We then perform floating-to-fixed-point conversion by defining $v_j(t) = V_j(t) \cdot 2^{\beta_v}$ as the neuron status, $N_{decay} = N_{leak} \cdot 2^{\gamma}$ as the decay constant; β_v, γ are integers in the range [0, 31]. Eqs (2)–(4) are converted into the following fixed-point equations:

$$v_j(t) \leftarrow v_j(t-1) \cdot N_{decay} \ / \ 2^{\gamma} + \sum_i S_{ij} W_{ij} \cdot 2^{\beta_v} \tag{5}$$

$$v_j(t) \leftarrow \begin{cases} 0,\ if\ t \in [T_f,\ T_f + T_{ref}] \\ H(V_{th} \cdot 2^{\beta} - v_j(t)) \cdot v_j(t),\ otherwise \end{cases} \tag{6}$$

$$S_i(t) \leftarrow H(v_j(t) - V_{th} \cdot 2^{\beta_v}) \tag{7}$$

Since the membrane potential $v_j(t)$ and synapse weights W_{ij} have significantly different dynamic ranges, we apply different scaling factors during floating-point to fixed-point conversion, β_v and β_w, respectively. We define $\beta_d = \beta_v - \beta_w$ as the difference between scaling factors, then Eq. (5) turns into:

$$v_j(t) \leftarrow v_j(t-1) \cdot N_{decay} \ / \ 2^{\gamma} + \left(\sum_i S_{ij} W_{ij} \cdot 2^{\beta_w}\right) \cdot 2^{\beta_d} \tag{8}$$

Eqs. (6)-(8) form the set of *kernel equations* that are executed by the NPU to perform simulation of a network of LIF neurons.

3. Architectural design of the NPU

3.1. Architecture overview

Fig. 1 shows the overall microarchitecture of the Darwin NPU. Due to its limited area size, the NPU supports 8 physical *neuron units* on the chip, which are used to implement logical neurons

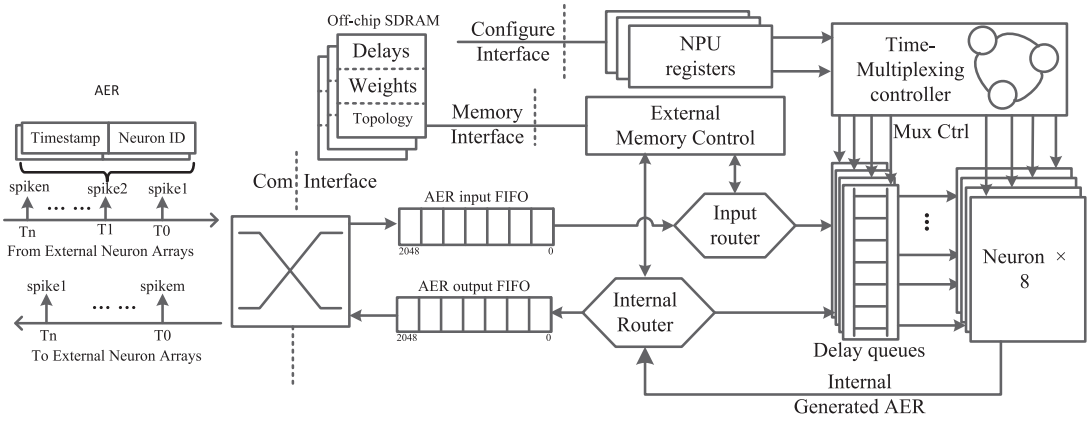


Fig. 1. The overall microarchitecture of the NPU.

with time multiplexing. The total number of logical neurons supported by the NPU is equal to the number of physical neurons multiplied by the degree of time-division multiplexing (which is configurable, as shown in Table 2 later). The *Address-Event Representation (AER)* format is used for information encoding of both input and output spikes. Each spike is represented with an AER packet, which consists of two fields: ID of the neuron that generated the spike and the timestamp when the spike is generated. The NPU works in an event-triggered manner, driven by arrival of input AER packets. Hence if there is no input AER packets, the NPU can be potentially put in sleep mode to achieve low stand-by power consumption. Input AER packets are processed in a streaming and online manner instead of in a batch mode, in order to provide fast response to input stimuli.

The runtime execution consists of the following steps:

1. *Time calibration:* The NPU works in the event-driven mode. When spikes arrive at the input FIFO, the NPU takes an AER packet out of the input FIFO and checks whether the timestamp of the AER matches the current time. If they match, go to spike routing process; otherwise, go to the neuron status update process.
2. *Input spike routing*: Each input spike in the form of an AER packet contains a timestamp and source (pre-synaptic) neuron ID, which is used to look up the IDs of destination (post-synaptic) neuron IDs, and the synapse attributes including their weights and delays stored in off-chip DRAM.
3. *Synapse delay management*: Each synapse has a configurable delay parameter, i.e., the delay from spike generation of the pre-synaptic neuron to spike reception of the post-synaptic neuron. Each entry in the Weight-Sum Queue contains the intermediate result of a weight sum to be sent to the neuron after a certain delay. Assuming a maximum of 15 delay values are supported, the *Weight-Sum Queue* consists of 15 entries, each corresponding to a delay value in the range [1,15]. The k^{th} entry stores the sum of input synapse weights (used to compute the term $\sum_i S_{ij} V_{max} w_{ij}$ in Eq. (2)) with synaptic delay of k. Fig. 2(a) shows the conventional design. Each time an incoming spike on a synapse with delay k (in the range of [1,15]) arrives, the current Weight-Sum at position k is updated by adding the synapse weight to it. At every time step, regardless of whether there is an incoming spike or not, every element of the Weight-Sum Queue is shifted leftward by one position, and the left-most Weight-Sum entry is then sent to the neuron unit. This design is not very efficient, since it incurs a large number of memory accesses. We adopt the more efficient design shown in Fig. 2(b), where the Weight-Sum Queue is replaced by a circular buffer with 15 entries. At each time step, the Current Time-step Pointer (CTP) is shifted by one entry, and the content of the entry previously pointed to by CTP is sent to the neuron.
4. *Neuron status update*: Each neuron performs status update based on Eqs. (2)–(4). The neuron first fetches the current status of the updating biological neuron from local status memory, and then fetches the weight sum of current step from the Weight-Sum Queue. If an output spike is generated, it is sent to the spike router (input or internal router in Fig. 1) in the form of AER packets.
5. *Internal spike routing:* This is similar to the input spike routing process.

Table 1
Some parameters and their bit-widths.

Variable	Note	Bit Width
N_{decay}	Decay constant.	32 bits
β_d	Scaling factor difference.	5 bits
V_{th}	Firing threshold voltage.	32 bits
T_{ref}	Refractory period.	8 bits

3.2. Main parameters and variables

1) *Global parameters:* including all parameters in Eqs. (2)–(4), which are shared by all neurons (after floating-to-fixed point conversion and parameter consolidation). Since these parameters are used at every time step, and their sizes are small, they are stored in the on-chip registers to maximize performance. Table 1 shows the variables and their bit width
2) *Per-neuron variables:* including each neuron's membrane potential $V_j(t)$, remaining length of the refractory period, and delay queue, stored in an on-chip SRAM of size 16 KB for each physical neuron. Hence the NPU has a total of 8*16 KB = 128 KB of on-chip SRAM.
3) *Synapse attributes:* including a table of Boolean values encoding the neural network's connection topology, as well as weight and delay attributes of each synapse. Each synapse's attributes are accessed when the synapse is activated by a spike. Its average access frequency is low, since a specific synapse in a large SNN has low average probability of being activated. The synapse attributes can be very large, ranging from several MBs up to GBs depending on the SNN size. Therefore, they are stored in off-chip SDRAM to archive high storage density.

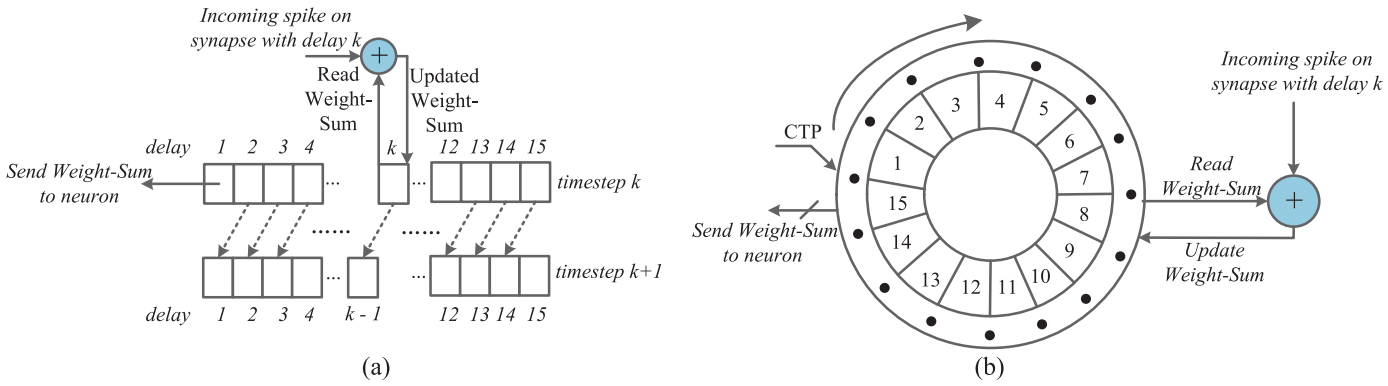


Fig. 2. Circular buffer design for the Weight-Sum Queue (refer to Fig. 4 for the overall design of the Weight-Sum queues).

3.3. Off-chip memory system

The spike router is responsible for translating the spikes in AER format to Weight-Delay information by accessing off-chip SDRAM that stores SNN topology, and Weight-Delay information for each synapse. We now introduce the off-chip memory data structure that stores this information. Suppose the system supports a maximum number of N neurons and M synapses. We define the following data structures:

1) *Topology Table TT[N][N]*: a 2-dimentional N-by-N matrix of 0–1 values; each 1 entry $TT[i][j]=1$ denotes a connection, or a synapse, from the pre-synaptic neuron with ID i to post-synaptic neuron with ID j.
2) *Weight-Delay Array WD[M]*: an array with M entries, each storing a 32-bit word consisting of a tuple *(Weight[k], Delay[k])*, where *Weight[k]* denotes the synapse weight, and *Delay[k]* denotes the synapse delay in the range of [0,15] for the k^{th} synapse. Both *Weight[k]* and *Delay[k]* are 16-bit numbers. All of each neuron's outgoing synapses are stored consecutively in *WD[M]* for easy lookup.
3) *Address Array A[N]*: an array with N entries; each entry *A[i]* storing the address (index) of the first outgoing synapse (among all synapses of neuron with ID i) of the neuron with ID i in the Weight-Delay Array *WD[M]*.

Fig. 3 illustrates the process of looking up the Weight-Delay entry for each of the outgoing synapses of a pre-synaptic neuron, with the following steps (each step is numbered corresponding to the number marked in the figure):

1) When the router receives an incoming AER packet with pre-synaptic neuron with ID i, e.g., $i=2$, read the i^{th} row of the Topology Table *TT[N][N]*.
2) Index into the array *A[N]* to obtain *A[i]*.
3) For each entry $T[i][j]=1$ in the i^{th} row of *TT[N][N]*, increment a counter m (initialized to 0).
4) The entry with index $k=A[i]+m$ in the Weight-Delay Array *WD[M]* belongs to the neuron with ID j.

3.4. On-chip memory system

The degree of time-multiplexing of logical neurons on physical neurons is a configurable parameter. If we use a higher degree of time-multiplexing to support a higher maximum number of logical neurons, we will need more memory resource to store the neuron status information, since each logical neuron needs a circular buffer to store its Weight-Sums (Fig. 2); if we want to support a higher maximum number of possible synaptic delays for each neuron, in order to have more fine-grained spike timing resolution, then we need a larger size circular buffer to store to store Weight-Sums of each neuron. Given a finite size of total memory space, we can make a tradeoff decision between the maximum number of logical neurons, and the maximum number of synaptic delay values. Therefore, we present a configurable on-chip memory system to support the tradeoff decision that can be made by the application designer.

Each physical neuron unit contains a local SRAM with the size of 4096 words * 4Bytes/word = 16 KB. Each word is an entry that may contain either a weight sum or neuron status (membrane voltage and refractory status). By changing the proportion of entries used to store neuron statuses and Weight-Sum Queues, the NPU can be set to different configurations to meet the requirements of different application scenarios and to avoid memory waste. Table 2 shows four possible configurations with different tradeoffs between the maximum number of neurons and synaptic delay precision. Configuration 1 supports a maximum of 2048 neurons, and 15 possible synaptic delay values (from 1 to 15); while Configuration 4 supports a maximum of 32,768 neurons, but only 1 possible synaptic delay value of 1. In all 4 configurations, the maximum number of synapses is equal to square of the maximum number of neurons, i.e., $20{,}48^2 = 4$ million synapses for configuration 1; $327{,}68^2 = 1$ billion synapses for Configuration 4, but the actual number is likely to be limited by external DRAM size. If an SNN does not need different synaptic delay values, then Configuration 4 can be adopted to support more neurons in the SNN.

Fig. 4 illustrates how the memory system can be configured to support different configurations, in order to achieve flexible tradeoff between the maximum number of neurons and the maximum number of synaptic delay values. In Fig. 4(a), each physical neuron unit is time-multiplexed 256 times, hence $8*256=2048$ logical neurons are supported with 8 physical neuron units; each Weight-Sum Queue has 15 entries, hence a maximum of 15 synaptic delays in the range [1,15] are supported. In Fig. 4(b), each neuron unit is time-multiplexed 512 times, hence $8*512=4096$ neurons are supported with 8 physical neuron units; each Weight-Sum Queue has 7 entries, hence a maximum of 7 synaptic delays in the range [1,7] are supported.

4. Experiment results

The Darwin NPU has been implemented with synthesizable Verilog, and fabricated with standard 180 nm CMOS technology with area size of $5 \times 5\text{mm}^2$ and 70 MHz clock frequency at the worst case. It consumes 0.84 mW/MHz with 1.8 V power supply for typical applications. A RISC CPU core is integrated on-chip, based on openrisc1200 from the open source project Open-

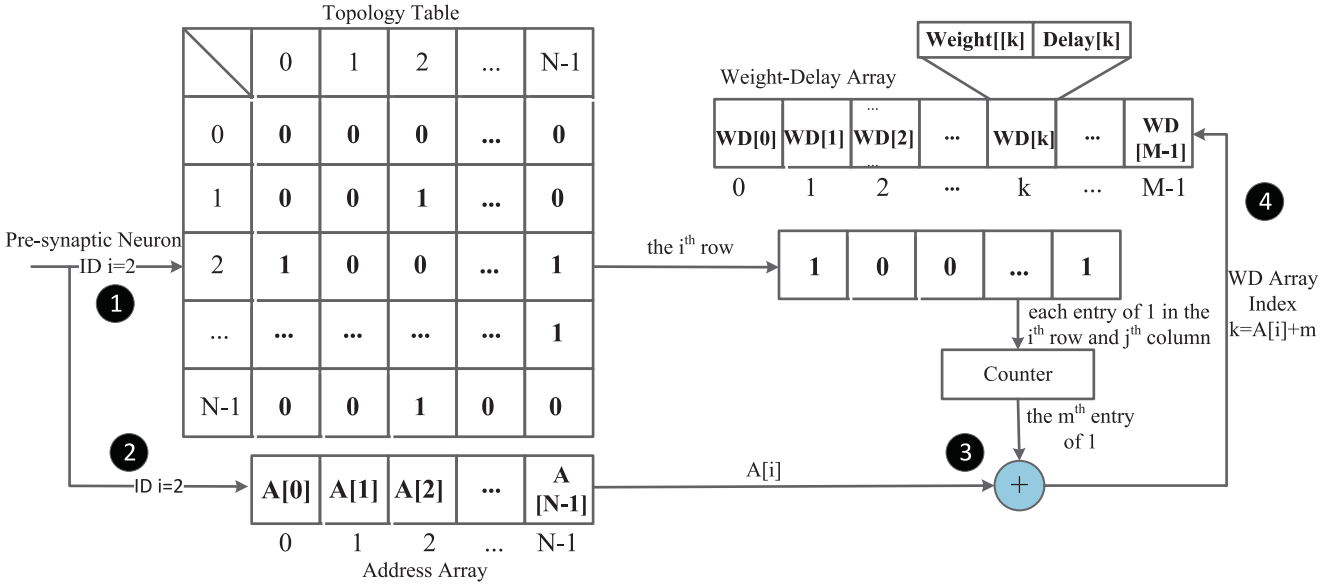
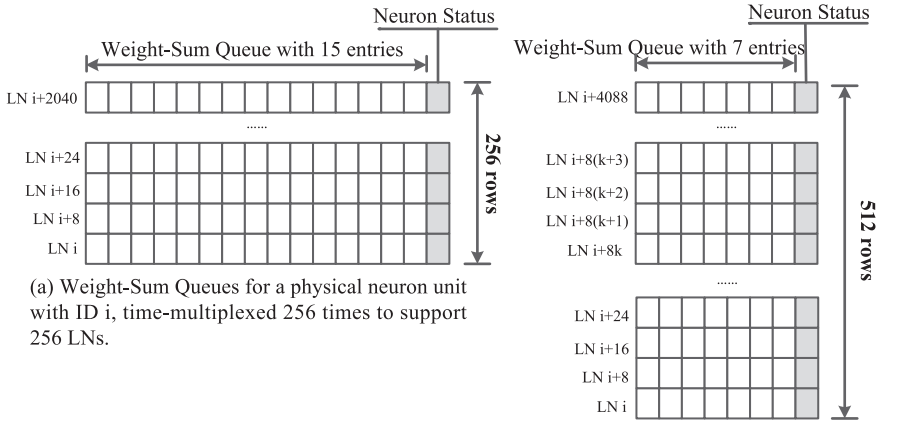


Fig. 3. The process of looking up the Weight-Delay entry for a given neuron in the off-chip memory system.

Table 2
Four possible configurations with different network size and delay precision.

Configuration.	Max # Neurons	# Synaptic Delays	Max # Synapses
1	2048	15	4M
2	4096	7	16.7M
3	8192	3	67M
4	32,768	1	1000 M (Limited by DRAM size)



(a) Weight-Sum Queues for a physical neuron unit with ID i, time-multiplexed 256 times to support 256 LNs.

(b) Weight-Sum Queues for a physical neuron unit with ID i, time-multiplexed 512 times to support 512 LNs.

Fig. 4. Reconfigurable design for Weight-Sum Queues. LN stands for *Logical Neuron*.

Cores minsoc (http://opencores.org/project,minsoc). In addition to the NPU and CPU, the SoC also contains an on-chip 32-bit Wishbone bus, 64 KB SRAM, SPI flash, UART controller and SDRAM controller. The CPU core acts as the interface for system configuration and transfer of AER packets between the Darwin NPU and the outside world. It mainly performs memory load and store operations, not computation-intensive operations, we do not need a high-performance CPU core. Since the NPU currently supports a relatively small number of neurons in the SNN, the CPU core and the Wishbone bus connecting the CPU core and NPU are not performance bottlenecks. For connection to the outside, off-chip world, we perform protocol conversion between UART and USB, so that the entire SoC can be used as an USB device and attached to a host PC. Fig. 3 shows the die photo, and the prototype PCB board.

The NPU was first design and verified by prototyping on a Xilinx Spartan 6 XC6SLX45 FPGA. The FPGA prototype was logical synthesized by Synopsys Synplify Premier with Design Planner, placed and routed by Xilinx ISE. The FPGA prototype runs under a 25 MHz system clock with the external SDRAM. Table 3 shows the utilization of various FPGA resources.

Next, we present two application case studies.

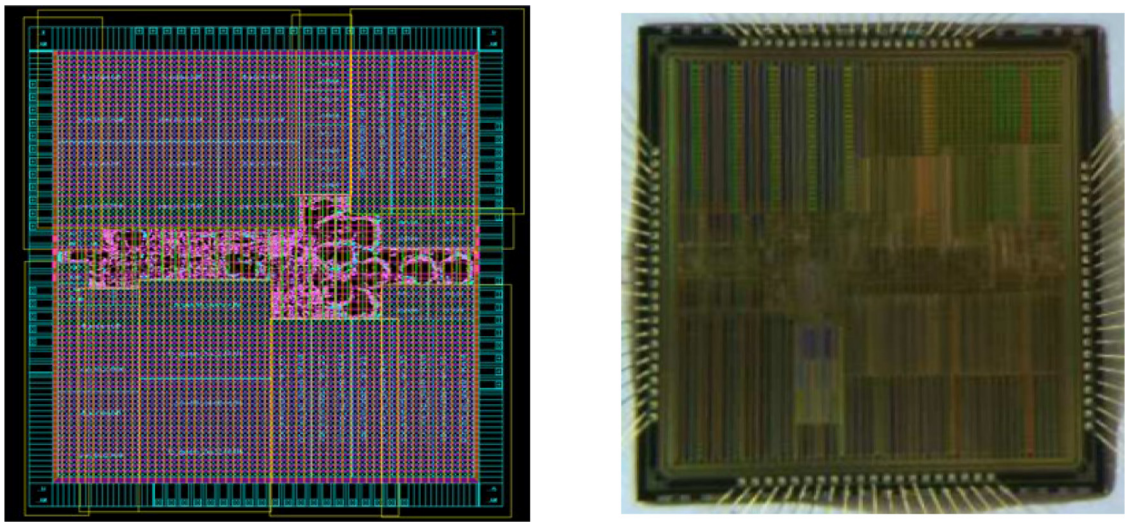


Fig. 5. Photos of the die and the PCB board.

Table 3
Resource utilization on the FPGA by the entire SoC, and by the NPU.

Resource	Used by SoC	Used by NPU	Total on FPGA
Slice LUTs	11489	6214	27288
Slice Flip-Flops	4705	1676	54576
18K Block RAMs	110	72	116
DSP48s	36	32	58

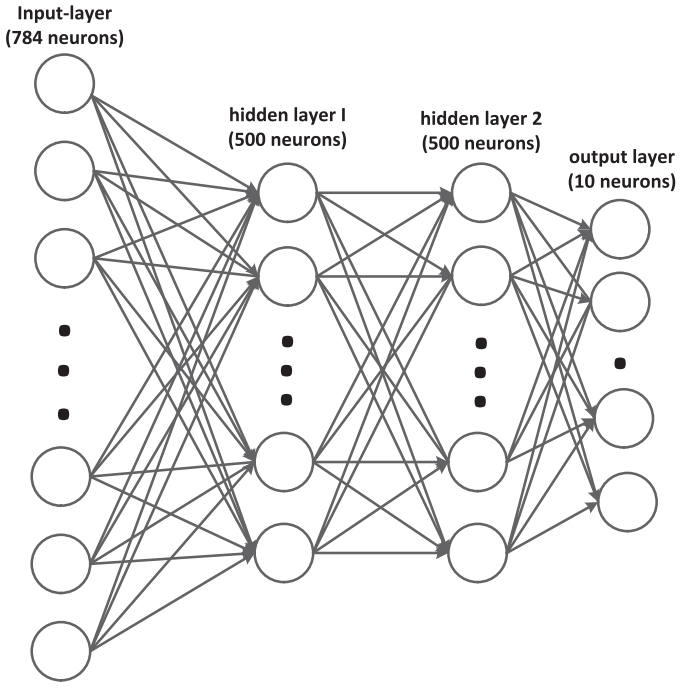


Fig. 6. The SNN for handwritten digit recognition.

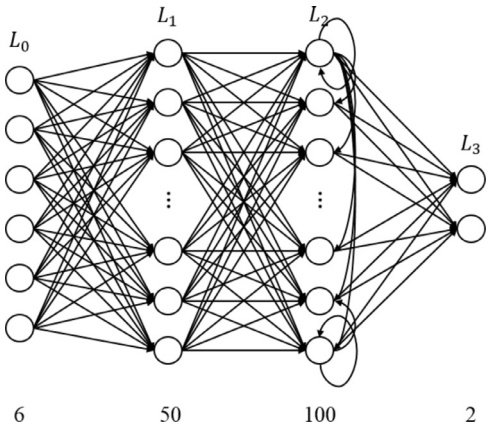


Fig. 7. The SNN for the motor-imagery system.

4.1. Application case 1: handwritten digit recognition

The first application case study is the Spiking Deep Belief Network (DBN) from [17] for handwritten digit recognition. Fig. 6 shows the SNN network architecture. It is a 4-layer SNN, with full feedforward connection between layers. The input layer consists of 784 neurons, each representing an image pixel in a 28×28 pixel input image; each input neuron emits a spike train that uses firing rate coding to encode pixel intensity. Each of the two hidden layers consists of 500 neurons; and the output layer consists of 10 neurons, each representing a digit of 0–9. The output neuron with the highest firing rate is selected as the classification output. Since the input spikes are processed in a streaming and online manner, the NPU starts processing as soon as the first spike of the input spike train arrives, instead of waiting for the entire spike train that encodes the input image.

We consider another SNN hardware accelerator *Minitaur* 7 [17], which implemented the same Spiking DBN on a Xilinx Spartan-6 FPGA. The Darwin NPU is configured to have clock speed of 25 MHz, with average latency of 0.16 s for recognizing each digit, and overall classification accuracy of 93.8%. In comparison, Minitaur has clock speed of 75 MHz, average latency of 0.152 s, and classification accuracy of 92%. The Darwin NPU achieved similar average latency with a lower clock speed, and slightly better classification accuracy.

4.2. Application case 2: EEG decoding of motor imagery

The second application case study is decoding of *Electroencephalogram (EEG)* signals. We use an *Emotiv* headset to collect EEG signals for real-time classification of the user's motor imagery, i.e., whether the user is thinking of left or right direction, and use the result to control the movement of a virtual ball on the screen. Fig. 7 shows the SNN network architecture. The

SNN has 4 layers, with full feedforward connection between layers. The input layer consists of 6 neurons, each representing an EEG channel; each input neuron emits a spike train that uses firing rate coding to encode EEG signal amplitude. The first hidden layer consists of 50 neurons, and the second hidden layer consists of 100 neurons, with full recurrent connections within the layer; the output layer consists of 2 neurons, each representing a binary decision of either left or right motor imagery. The output neuron with the highest firing rate is selected as the classification output. The training set consists of 4000 EEG signal fragments, and the test set consists of 4000 EEG signal fragments captured and decoded in real-time. The classification accuracy is 92.7%.

5. Conclusions

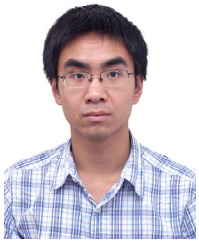
We have presented the Darwin NPU, a neuromorphic hardware co-processor based on SNN. While the NPU is currently a single-chip system, the standard communication interface defined by the AER format enables future extensions to a multi-chip distributed system with AER bus connection. The NPU can also be use it as a Processing Element in a Network-on-Chip (NoC) architecture, using the AER format for input and output spikes, in order to scale up size of the SNN on the chip to potentially millions of neurons instead of mere thousands.

Acknowledgments

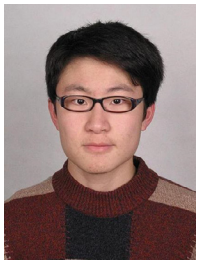
This work was supported by National Key Basic Research Program of China (2013CB329504), National Science Foundation of China (No. 61404041) and Zhejiang Provincial Natural Science Foundation of China (LR15F020001).

References

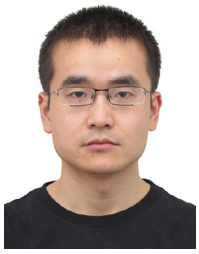
[1] E. Izhikevich, Which model to use for cortical spiking neurons? Neural Netw. IEEE Trans. 15 (5) (2003) 1063–1070.
[2] H.E. Plesser, M. Diesmann, M.O. Gewaltig, et al., NEST: the neural simulation tool, in: Encyclopedia of Computational Neuroscience, 2015, pp. 1849–1852.
[3] M. Stimberg, D.F.M. Goodman, V. Benichoux, et al., Equation-Oriented Specification of Neural Models for Simulations, 2012.
[4] M.L. Hines, N.T. Carnevale, The NEURON simulation environment, Neural Comput. 9 (6) (1997) 1179–1209.
[5] C. Eliasmith, A unified approach to building and controlling spiking attractor networks, Neural Comput. 17 (6) (2005) 1276–1314.
[6] S. Hill, H. Markram, The blue brain project[C]//engineering in medicine and biology society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE, 2008.
[7] S.B. Furber, F. Galluppi, S. Temple, et al., The spinnaker project, Proc. IEEE 102 (5) (2014) 652–665.
[8] M. Beyeler, K.D. Carlson, T.S. Chou, et al., CARLsim 3: a user-friendly and highly optimized library for the creation of neurobiologically detailed spiking neural networks, in: Neural Networks (IJCNN), 2015 International Joint Conference on, IEEE, 2015, pp. 1–8.
[9] F. Akopyan, J. Sawada, A. Cassidy, et al., TrueNorth: design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip[J], Comput. Aided Des. Integr. Circuits Syst. IEEE Trans. 34 (10) (2015) 1537–1557.
[10] N. Qiao, H. Mostafa, F. Corradi, et al., A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128 K synapses, Frontiers Neurosci. 9 (2015) 141.
[11] B.V. Benjamin, P. Gao, E. McQuinn, et al., Neurogrid: a mixed-analog-digital multichip system for large-scale neural simulations, Proc. IEEE 102 (5) (2014) 699–716.
[12] J. Schemmel, D. Bruderle, A. Grubl, et al., A wafer-scale neuromorphic hardware system for large-scale neural modeling[C], in: Circuits and systems (ISCAS), proceedings of 2010 IEEE international symposium on, IEEE, 2010, pp. 1947–1950.
[13] T. Prodromakis, C. Toumazou, L. Chua, Two centuries of memristors, Nat. Mater. 11 (6) (2012) 478–481.
[14] J. C Gehlhaar, Neuromorphic processing: a new frontier in scaling computer architecture, ACM SIGPLAN Notices 49 (4) (2014) 317–318.
[15] P. P Dayan, L.F. Abbott, Theoretical Neuroscience, MIT Press, Cambridge, 2001.
[16] J. Shen, D. Ma, Z. Gu, et al., Darwin: a neuromorphic hardware co-processor based on spiking neural networks, Sci. China Inf. Sci. (2016) 1–5.
[17] D. Neil, S.C. Liu, Minitaur, an event-driven FPGA-based spiking network accelerator, Very Large Scale Integr. (VLSI) Syst. IEEE Trans. 22 (12) (2014) 2621–2628.



De Ma received the B.S degree in electrical and information engineering from Zhejiang University, Hangzhou, China, in 2004. Then he pursued his P.H.D. study in the Institute of VLSI Design, Zhejiang University. He is now a faculty member in the Department of Electronic and Information, Hangzhou Dianzi University. His research interests include VLSI design, SoC architecture exploration and implementation, etc.



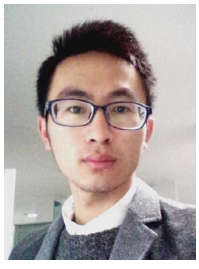
Juncheng Shen received his Bachelor's degree in electronic science and technology from Zhejiang University in 2014, and is currently a PhD student at Zhejiang University.



Zonghua Gu received his Ph.D. degree in Computer Science and Engineering from the University of Michigan at Ann Arbor in 2004. He is currently an associate professor in the College of Computer Science, Zhejiang University. His research area is real-time and embedded systems.

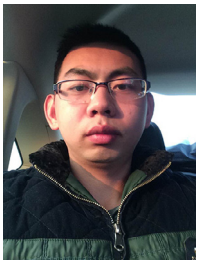
Ming Zhang received his Bachelor's degree in software engineering from Zhejiang University in 2010, and is currently a Ph.D. student at Zhejiang University.

Xiaolei Zhu received the M.S. and Ph.D. degree in electrical engineering from Zhejiang University, China, in 2003 and Keio University, Japan in 2012 respectively. From 2003 to 2006, he was with the Institute of Micro-electronics, Tsinghua University, China, as an assistant researcher. From 2007 to 2012, he was with Fujitsu Laboratories Ltd., Kawasaki, Japan where he designed high performance ADC with calibration. Since 2013, He has been with the College of Electrical Engineering and College of Microelectronics of Zhejiang University, as an assistant professor. His current research interests are in the field of mixed-signal integrated circuits and system for the neural morphic computing and medical healthcare application.



Xiaoqiang Xu received his Bachelor's degree in computer Science from Northwestern Polytechnical University in 2010, and is currently a Master's student at Zhejiang University.

Qi Xu received his Bachelor's degree in Software Engineering from Zhejiang University of Technology in 2015, and is currently a PhD student at Zhejiang University.



Yangjing Shen received his Bachelor's degree in electrical engineering from Hangzhou Dianzi University in 2015, and is currently a Master's student at Hangzhou Dianzi University.

Gang Pan received his B.*Sc.* and Ph.D. in Computer Science in 1998 and 2004, respectively, from Zhejiang University. He is a professor of the College of Computer Science and Technology, Zhejiang University, China. His research interests include pervasive computing, computer vision, and pattern recognition. He has published more than100 refereed papers.