National Science Review
11: nwae102, 2024
https://doi.org/10.1093/nsr/nwae102
Advance access publication 18 March 2024

COMPUTER SCIENCE

Special Topic: Human Brain Computing and Brain-Inspired Intelligence

Darwin3: a large-scale neuromorphic chip with a novel ISA and on-chip learning

De Ma[1,2,3,4,†], Xiaofei Jin[1,2,†], Shichun Sun[2], Yitao Li[1,3], Xundong Wu[2], Youneng Hu[1], Fangchao Yang[2], Huajin Tang[1,2,3,4], Xiaolei Zhu[5,2], Peng Lin[1,3,4] and Gang Pan[1,2,3,4,*]

[1]College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China; [2]Research Center for Intelligent Computing Hardware, Zhejiang Lab, Hangzhou 311121, China; [3]The State Key Lab of Brain-Machine Intelligence, Zhejiang University, Hangzhou 310027, China; [4]MOE Frontier Science Center for Brain Science and Brain-machine Integration, Zhejiang University, Hangzhou 310027, China and [5]College of Micro-Nano College of Micro-Nano Electronics, Zhejiang University, Hangzhou 311200, China

[*]**Corresponding author.** E-mail: gpan@zju.edu.cn

[†]Equally contributed to this work.

Received 27 September 2023; **Revised** 3 February 2024; **Accepted** 23 February 2024

ABSTRACT

Spiking neural networks (SNNs) are gaining increasing attention for their biological plausibility and potential for improved computational efficiency. To match the high spatial-temporal dynamics in SNNs, neuromorphic chips are highly desired to execute SNNs in hardware-based neuron and synapse circuits directly. This paper presents a large-scale neuromorphic chip named Darwin3 with a novel instruction set architecture, which comprises 10 primary instructions and a few extended instructions. It supports flexible neuron model programming and local learning rule designs. The Darwin3 chip architecture is designed in a mesh of computing nodes with an innovative routing algorithm. We used a compression mechanism to represent synaptic connections, significantly reducing memory usage. The Darwin3 chip supports up to 2.35 million neurons, making it the largest of its kind on the neuron scale. The experimental results showed that the code density was improved by up to 28.3× in Darwin3, and that the neuron core fan-in and fan-out were improved by up to 4096× and 3072× by connection compression compared to the physical memory depth. Our Darwin3 chip also provided memory saving between 6.8× and 200.8× when mapping convolutional spiking neural networks onto the chip, demonstrating state-of-the-art performance in accuracy and latency compared to other neuromorphic chips.

Keywords: neuromorphic computing, spiking neural network, instruction set architecture, connectivity compression

INTRODUCTION

Spiking neural networks (SNNs) have garnered significant attention from researchers due to their ability to process spatial-temporal information in an efficient event-driven manner. To exploit the capabilities of SNNs, several spiking neural network simulation platforms have been introduced, such as Brian2 [1], NEST [2] and SPAIC [3]. Nevertheless, the dependence of these platforms on using extensive GPU and CPU resources to mimic the spiking dynamics with a high count of timing steps potentially diminish the intrinsic advantages of SNNs. Neuromorphic chips are designed for efficient execution of spiking neural networks, which have demonstrated promising performance in brain simulation and specific ultra-low power scenarios. However, several limitations prevent them from fully leveraging the advantages of spiking neural networks. To better leverage the benefits of the SNN models, we should emphasize the following three aspects when designing neuromorphic chips.

Flexibility of neural models. One of the key functions of neuromorphic chips is to simulate diverse biological neurons and synapses. However, many neuromorphic chips only support a single type of neuron model, as evidenced in platforms like Neurogrid [4], which is based on analog neuron circuits. Some works introduce a degree of configurability to accommodate various neuron models. Loihi [5] achieved enhanced learning capabilities through configurable sets of traces and delays. FlexLearn [6] has conceived a versatile data path that amalgamates key features from diverse models. Moreover, endeavors have been undertaken to develop fully configurable neuronal models using instructions. SpiNNaker's multi-core processors [7], based on conventional ARM cores, provide significant flexibility. However, it is associated with reduced performance and energy efficiency compared to other accelerators. Loihi2 [8] presents an instruction set incorporating logical and mathematical operations similar to Reduced Instruction Set Computer (RISC) instructions. However, instruction sets designed for conventional neural networks lack efficiency for SNNs despite their flexibility.

Synapse density. To further unlock the potential of SNNs, neuromorphic chips need to support the representation of large-scale SNNs with more complex topologies [9]. However, current neuromorphic chips pay less attention to this aspect, primarily concentrating on simulating the behavior of neurons and synapses. For instance, TrueNorth [10] employs a crossbar design for synaptic connections, but it suffers from limited and fixed fan-in/fan-out capacity. Loihi [5] takes an approach by using axon indexes to encode topology, thereby enhancing flexibility. Loihi2 [8] proposes optimizing for convolutional and factorized connections, but gives less attention to other connection types. Unicorn [11] introduces a technique for merging synapses from multiple cores to extend the synaptic scale of a single core. Improving the synapse density for various topologies under limited storage conditions is thus crucial for optimizing the cost effectiveness of the chip.

On-chip learning ability. Learning capability is a critical feature of biological neural networks. Currently, only a few neuromorphic chips support on-chip learning. Among those, the supported learning rules are pretty restricted. For instance, BrainscaleS2 [12] only accommodates fixed learning algorithms. Loihi [5] supports programmable rules for pre-, post- and reward traces. Loihi2 [8] extends its capabilities of the programmable rules applied to pre-, post- and generalized 'third-factor' traces. However, even with the enhanced flexibility exhibited by Loihi2 [8], it cannot accommodate novel learning rules that might emerge. The latest research achievements in the field of electrochemical memory array [13] also provide new reference solutions.

In this paper, we design a large-scale neuromorphic chip with a domain-specific instruction set architecture (ISA), named Darwin3, to support model flexibility, system scalability and on-chip learning capability of the chip. Darwin3 is the third generation of our Darwin [14] family of neuromorphic chips, which was successfully taped out and lit up in December 2022. Our main contributions are as follows.

(1) We propose a domain-specific ISA for neuromorphic systems, capable of efficiently describing diverse models and learning rules, including the integrate-and-fire family [15], Izhikevich [16] and Spike Timing Dependent Plasticity (STDP)[17], among others. The proposed architecture excels in achieving high parallelism during computational operations, including loading parameters and updating state variables such as the membrane potential and weights.
(2) We design a novel mechanism to represent the topology of SNNs. This mechanism effectively compresses the information required to describe synaptic connections, thereby reducing overall memory usage.

The article is organized as follows. First, we introduce the topic and briefly overview the article's contents. Second, we present the neuromorphic computing domain-specific ISA. Then, we offer the overall architecture of the neuromorphic chip and the implementation of each part, including the architecture of neuron nodes and the mechanism of topology representation. Lastly, we present the experimental results.

THE DARWIN3 DOMAIN-SPECIFIC ISA

Model abstraction of neurons, synapses and learning

Many neuron models have been proposed in the field of computational neuroscience. The leaky integrate-and-fire (LIF) family [15,18–20] is a group of spiking neuron models that can be described by one- or two-dimensional differential equations and were widely implemented on hardware accelerators. These models have been developed for use in many real-world applications. The Hodgkin–Huxley model [21,22] is considered biologically plausible and accurately captures the intricacies of neuron behavior with four-dimensional differential equations that represent the transfer of ions across the neuron membrane. However, this model can cause very high computational costs. The Izhikevich model [16], specifically designed to replicate bursting and spiking behaviors observed in the Hodgkin–Huxley model, is represented with two-dimensional differential equations.

All these neuron models are represented using systems of differential equations, with variations occurring only in the number of equations and the variables and parameters in each equation. The primary operators needed to solve them are the same. Therefore, it can be a practical approach to identify the common features shared by complex LIF models and utilize them to construct more complex models by introducing additional state variables and computation steps. We chose the adaptive leaky integrate-and-fire model [20] as the baseline with relatively more variables and parameters. Mathematically, it can be expressed by the following set of equations, which capture the dynamics of the model and its adaptation properties:

$$\tau_m \frac{dv_m}{dt} = -(v_m - E_L) - \frac{1}{g}(v_{adp} - I), \quad (1a)$$

$$\tau_{adp} \frac{dv_{adp}}{dt} = a(v_m - E_L) - v_{adp}, \quad (1b)$$

$$\text{if}\,(v_m > v_{th}) : v_m = v_0,\ v_{adp} = v_{adp} + b. \quad (1c)$$

Here v_m is the membrane potential, τ_m is the membrane time constant, E_L is the leak reversal potential, g is the synapse conductance, v_{adp} is the adaptation current, τ_{adp} is the time constant of the adaptation current, a is the sensitivity to the sub-threshold fluctuations of the membrane potential, b is the increment of v_{adp} produced by a spike, v_0 is the reset potential after a spike and I is the synaptic spike current.

Similar to the various designs of neuron models with different computational complexities, there are also multiple synapse models, such as the delta and alpha synapse models [23]. One of the complex and commonly used models is the conductance-based (COBA) dual exponential model [23,24], given by

$$\frac{dh}{dt} = \frac{-h}{\tau_{rise}} + \delta(t_0 - t), \quad (2a)$$

$$\frac{dg}{dt} = \frac{-g}{\tau_{decay}} + h, \quad (2b)$$

$$I = g(v_m - E_L), \quad (2c)$$

where δ is a spike at time t_0, h is the gating variable of the ion channel, g is the synapse conductance, τ_{decay} is the time constant of the synaptic decay phase, τ_{rise} is the time constant of the synaptic rise phase, I is the synaptic spike current, v_m is the membrane potential and E_L is the leak reversal potential. The COBA dual exponential model has a similar computational complexity to that of (1). We chose this model as our representative synapse model.

Synaptic plasticity [25], the ability of synapses to change their strength, was first proposed as a mechanism of learning and memory by Donald Hebb [26]. After that, numerous learning rules have been proposed ever since. The STDP rule [17] and its variants are the most widely used. One relatively complex variant considers triplet [27] interactions and is reward modulated [28]. We select this model as the baseline, and through the selection of different state variables and parameters, the same set of equations can describe most STDP and its variant rules. The rule can be expressed mathematically as

$$\tau_{pre_0} \frac{dx_0}{dt} = -x_0 + a_{pre_0}\delta(t - t_{pre_0}), \quad (3a)$$

$$\tau_{post_0} \frac{dy_0}{dt} = -y_0 + a_{post_0}\delta(t - t_{post_0}), \quad (3b)$$

$$\tau_{post_1} \frac{dy_1}{dt} = -y_1 + a_{post_1}\delta(t - t_{post_1}), \quad (3c)$$

$$\tau_{rwd} \frac{dr}{dt} = -r + a_{rwd}\delta(t - t_{rwd}), \quad (3d)$$

$$dw(t) = A_{pre} r x_0(t)\delta(t - t_{post_0}) - A_{post_0} r y_0 \delta(t - t_{pre}) - A_{post_1} r y_1 \delta(t - t_{pre}), \quad (3e)$$

where x_0 is the pre-synaptic spike trace, y_0 is the first post-synaptic spike trace, y_1 is the second post-synaptic spike trace, r is the reward to modulate the synaptic traces, and the τ_* are time constants of x_0, y_0, y_1 and r.

Equations (1)–(3) describe three representative models. To implement the models using digital circuits, we need to convert the differential equations to discrete form. By applying the Euler method, equations (1) become

$$v_m(t+1) = p_0 v_m(t) + p_1 I(t+1) + p_2 v_{adp}(t+1) + c_0, \quad (4a)$$

$$v_{adp}(t+1) = p_3 v_{adp}(t) + p_4 v(t) + c_1, \quad (4b)$$

$$\text{if}\,(v(t+1) > v_{\text{th}}) : \begin{cases} v(t+1) = v_0, \\ v_{adp}(t+1) = v_{adp}(t+1) + c_2, \end{cases} \quad (4c)$$

and equations (2) become

$$h(t+1) = p_8 h(t) + w_{ij}\mathrm{H}[t - t^s], \quad (5a)$$

$$I(t+1) = \mathrm{g}(t+1)v(t) + p_7 g(t+1), \quad (5b)$$

$$g(t+1) = p_5 g(t) + p_6 h(t+1), \quad (5c)$$

where $p_0, \ldots, p_7$ are fixed coefficient parameters and $c_0, \ldots, c_2$ are constants. Similarly, equations (3) become

$$x_0(t+1) = P_3^* x_0(t) + C_0^* x_2(t), \quad (6a)$$

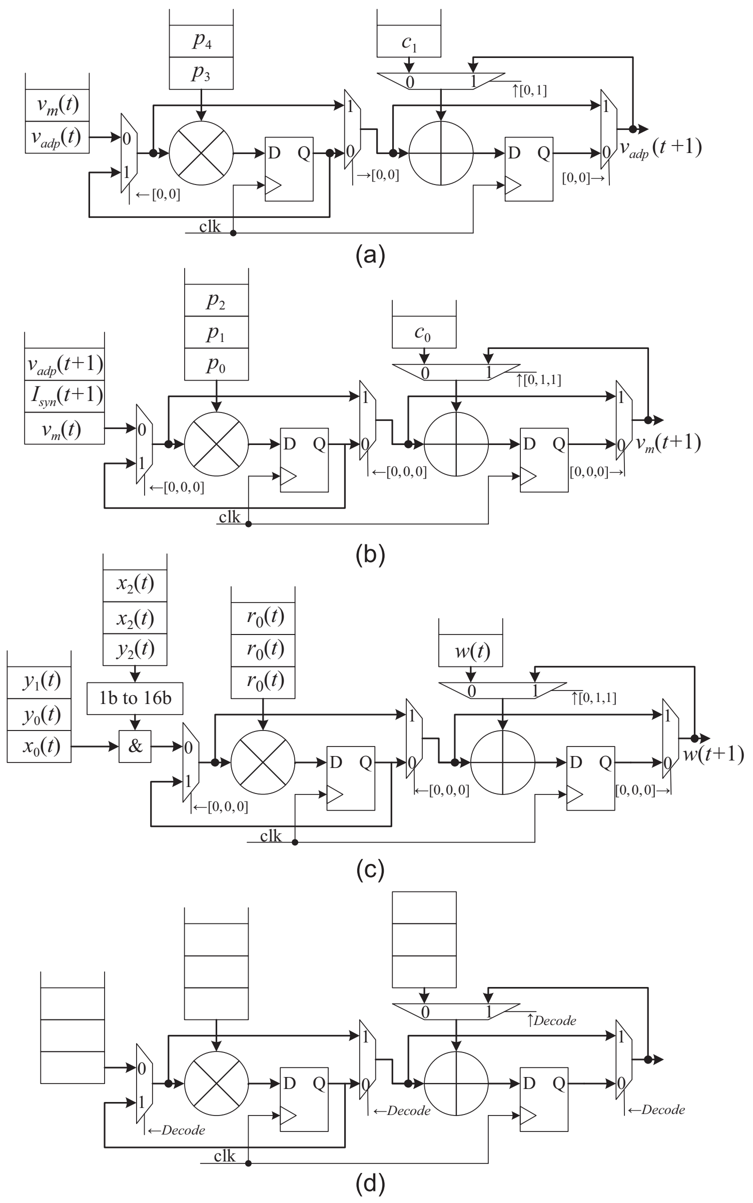


Figure 1. Typical data paths. (a) The data path of v_{adp}. (b) The data path of v_m. (c) The data path of w. (d) The common data path for state variables.

$$y_0(t+1) = P_4^* y_0(t) + C_1^* y_2(t), \tag{6b}$$

$$y_1(t+1) = P_5^* y_1(t) + C_2^* y_2(t), \tag{6c}$$

$$r_0(t+1) = P_6^* r_0(t) + C_3^* r_2(t), \tag{6d}$$

$$\begin{aligned} w(t+1) = w(t) &+ P_0^* r_0(t) x_0(t) y_2(t) \\ &+ P_1^* r_0(t) y_0(t) x_2(t) \\ &+ P_2^* r_0(t) y_1(t) x_2(t), \end{aligned} \tag{6e}$$

where $P_0^*, \ldots, P_6^*$ are fixed coefficient parameters and $C_0^*, \ldots, C_3^*$ are constants.

Table 1. Registers for state variables and parameters.

Name	Description
v_m, S_0	The membrane potential
g, S_1	The synaptic conductance
I, S_2	The synaptic current
h, S_3	The gating variable
v_{adp}, S_4	The adaptive voltage
v_{th}, S_5	The threshold voltage
w	The synaptic weight
v_0	The reset membrane potential value
IP_0–IP_7	Eight parameters for the inference stage, corresponding to p_0–p_7 in equations (4) and (5)
IC_0–IC_2	Three constants for the inference stage, corresponding to c_0–c_2 in equations (4) and (5)
X_0–X_1	Two pre-synaptic spike traces LS_0–LS_1
X_2	Flag for a spike from pre-synapse L_2
Y_0–Y_1	Two post-synaptic spike traces LS_3–LS_4
Y_2	Flag for a spike from post-synapse LS_5
R_0–R_1	Traces to do reward and punishment LS_6–LS_7
R_2	Reward and punishment LS_8
LP_0–LP_7	Eight parameters for the learning stage, corresponding to P^* in equation (6)
LC_0–LC_7	Eight constants for the learning stage, corresponding to C^* in equation (6)
TR_0–TR_7	Temporary registers for parameters and states

Equations (4), (5) and (6) reveal that both complex LIF models and STDP variants can be expressed as polynomials involving multiple multiplication and addition operations. To implement these polynomial computations in digital circuits, we map them to corresponding data paths for further analysis. Figure 1a–c illustrate that the data paths of v_{adp} and v_m in equation (4) and w in equation (6) are almost identical, except for different control signals from selectors and input sources, which allows us to efficiently implement these computations in circuits using a unified data path, where parameters can be pre-configured statically, and state variables are updated continuously over time steps. For more complex cases such as the Izhikevich model [16], the parameter that multiplies the state variables in the computation process is also a state variable. Therefore, we obtain the unified data path shown in Fig. 1d.

The proposed Darwin3 ISA

To effectively manage the data path and maximize performance and concurrency, it is crucial to design an efficient controller. First, we map state variables and parameters into a set of registers, as indicated in Table 1. It covers the state variables and parameters related to neurons and synapses, in

Table 2. Neuromorphic specific instruction set.

Opcode (5 bits)	Operand (11 bits)			
LSIS	LS (1 bit)	Reserve	NHIS (6 bits)	
	LS indicates whether an operation is a load or store operation, and NHIS is a 6-hot code corresponding to the state variables S_0–S_5, indicating whether each state variable needs to be loaded or stored during the operation.			
LDIP		NHIP (8 bits)		NHIC (3 bits)
	NHIP is an 8-hot code corresponding to the parameters p_0–p_7, and NHIC is a 3-hot code corresponding to c_0–c_2, indicating whether each needs to be loaded during the operation.			
LSLS	LS (1 bit)		NHLS (10 bits)	
	A 1-hot code LS bit indicates whether an operation is a load or store operation, and a 10-hot code NHLS corresponding to the state variables LS_0 to LS_9 indicates whether each state variable needs to be loaded or stored during the operation.			
LDLP		NHLP (7 bits)		NHLC (4 bits)
	A 7-hot code NHLP corresponding to the parameters LP_0–LP_6 and a 4-hot code NHLC corresponding to LC_0–LC_3 indicate whether each parameter needs to be loaded during the operation.			
UPTIS	Reserve	OHIS (3 bits)	NHIP (6 bits)	
	OHIS is a 1-hot code indicating whether I, g or v_{adp} is to be calculated and NHIP is a 6-hot code corresponding to p_3–p_7 and c_1, indicating whether each needs to participate in the calculation according to equations (4) and (5).			
UPTVM		Reserve		NHVM (4 bits)
	NHVM is a 4-hot code to determine whether v_m, I, v_{adp} or c_0 needs to participate in the calculation to update v_m according to equations (4) and (5).			
UPTLS	k (3 bits)	l (3 bits)	m (3 bits)	n (2 bits)
	The variables k, l, m and n determine the selected state variable LS_k that needs to update according to the equation $LS_k(t+1) = LP_l \times LS_m(t) + LC_n$.			
UPTWT		m (2 bits)		n (9 bits)
	A binary code m and n-hot code n determine the synaptic weight to update according to the equation $WT(t+1) = WT(t) + LP_m \prod LS_n$.			
UPTTS	k (3 bits)	l (3 bits)	m (3 bits)	n (2 bits)
	The variables k, l, m and n determine the selected temporary state variable RT_k that needs to update according to the equation $RT_k(t+1) = P_l \times S_m(t) + C_n$.			
GSPRS		Reserve		NHSP (4 bits)
	A 4-hot code NHSP respectively determines whether to fire a spike, perform a threshold comparison, involve an adaptive operation and whether a membrane potential needs to reset to v_0.			

which constants are static parameters. To provide users with the flexibility to implement different models, we propose a specialized ISA, as shown in Table 2.

The core principle of this ISA is to amalgamate common operations into a single instruction, taking into account the computational characteristics of SNNs. By doing so, it not only reduces the memory usage required for instructions, but also minimizes the time needed for instruction decoding during the computation process. We defined a set of instructions outlined in Table 2. This instruction set comprises 10 primary commands. The first group, which focuses on load and store operations, consists of LSIS, LDIP, LSLS and LDLP. Specifically, LSIS and LSLS cater to loading or writing back state variables for both the inference and learning processes, executed in parallel. On the other hand, LDIP and LDLP are designated for loading parameters in parallel for inference and learning phases, respectively.

The second group, tailored for updating state variables, includes UPTIS, UPTVM, UPTLS, UPTWT and UPTTS. Among these, UPTIS updates state variables, excluding the membrane potential. UPTVM is exclusively for adjusting the membrane potential. UPTLS emphasizes state variables specific to the learning stage, while UPTWT manages the adjustments of synaptic weights. UPTTS oversees the updating of temporary state variables. Lastly, the GSPRS instruction is dedicated to generating spikes. With these instructions, we can effectively manage the computing units and support the process required for constructing flexible SNN models.

Table 3. Extended instruction list.

Type	Instructions
Arithmetic	ADD, SUB, MUL, ADDI
Bitwise	SHIFT, LOGIC
Move	MOV, WMOV
Jump	CMP, JMP
Memory	SA, TS, LOAD/PUSH, STORE/POP, SP
Pseudo	DIV, EXP

Models such as AdEx [20] and HH [21] necessitate intricate operations, including exponential and division functions. These are not directly supported by the instructions outlined in Table 2. The design enables users to perform division and exponentiation operations using computational units like shift, multiplication and lookup tables, thus conserving hardware resources. Consequently, we have augmented our instruction set with several instructions typically found in reduced instruction set architectures, as detailed in Table 3.

In Table 4, we present a range of code examples beyond basic loading and storing operations to illustrate the efficacy of the proposed ISA. This selection demonstrates that simple LIF models and more complex triplet STDP rules can be concisely represented using minimal instructions. Additionally, specialized rules such as SDSP [29] can be efficiently encoded through a strategic combination of instructions. This versatility underscores the high flexibility and effectiveness of our instruction set design, establishing it as a viable tool for researchers and developers engaged in implementing diverse models in SNNs.

THE DARWIN3 CHIP ARCHITECTURE

Overview

The Darwin3 chip architecture is characterized by a two-dimensional mesh of computing nodes, forming a 24 × 24 grid, interconnected via a network on chip (NoC), shown in Fig. 2a. The node at position (0,0) features a RISC-V processing core for chip

Table 4. Code examples for widely used models.

Model	Code	Model	Code
LIF [18]	UPTVM 0×D GSPRS 0×A	Triplet STDP [27]	UPTLS 0×20 UPTLS 0×08 UPTLS 0×04 UPTWT 0×10C UPTWT 0×264 UPTWT 0×454
QIF [15]	UPTTS 0×021 MOV P0, RT0 UPTVM 0×D GSPRS 0×A	RSTDP [28]	UPTLS 0×20 UPTLS 0×08 UPTLS 0×02 UPTWT 0×10C UPTWT 0×264
ExpIF [15]	UPTTS 0×061 EXP RT1 RT0 UPTTS 0×48A MOV C0 RT2 UPTVM 0×D GSPRS 0×A	S-TP [30]	LayerH: UPTTS 0×15A MOV LP0 RT0 UPTWT 0×C0 LayerO: UPTWT 0×228
Izhikevich [16]	UPTTS 0×143 MOV P0, RT0 UPTIS 0×38 UPTVM 0×0F GSPRS 0×E	SDSP [29]	UPTLS 0×20 CMP RT0 S0 JMP Keep CMP RT1 LS1 JMP Keep CMP LS1 RT3 JMP Keep CMP LS1 RT2 JMP UP SUB W RT4 NOP Up: ADD W RT4 Keep: NOP
Basic STDP [17]	UPTLS 0×20 UPTLS 0×08 UPTWT 0×108 UPTWT 0×260		

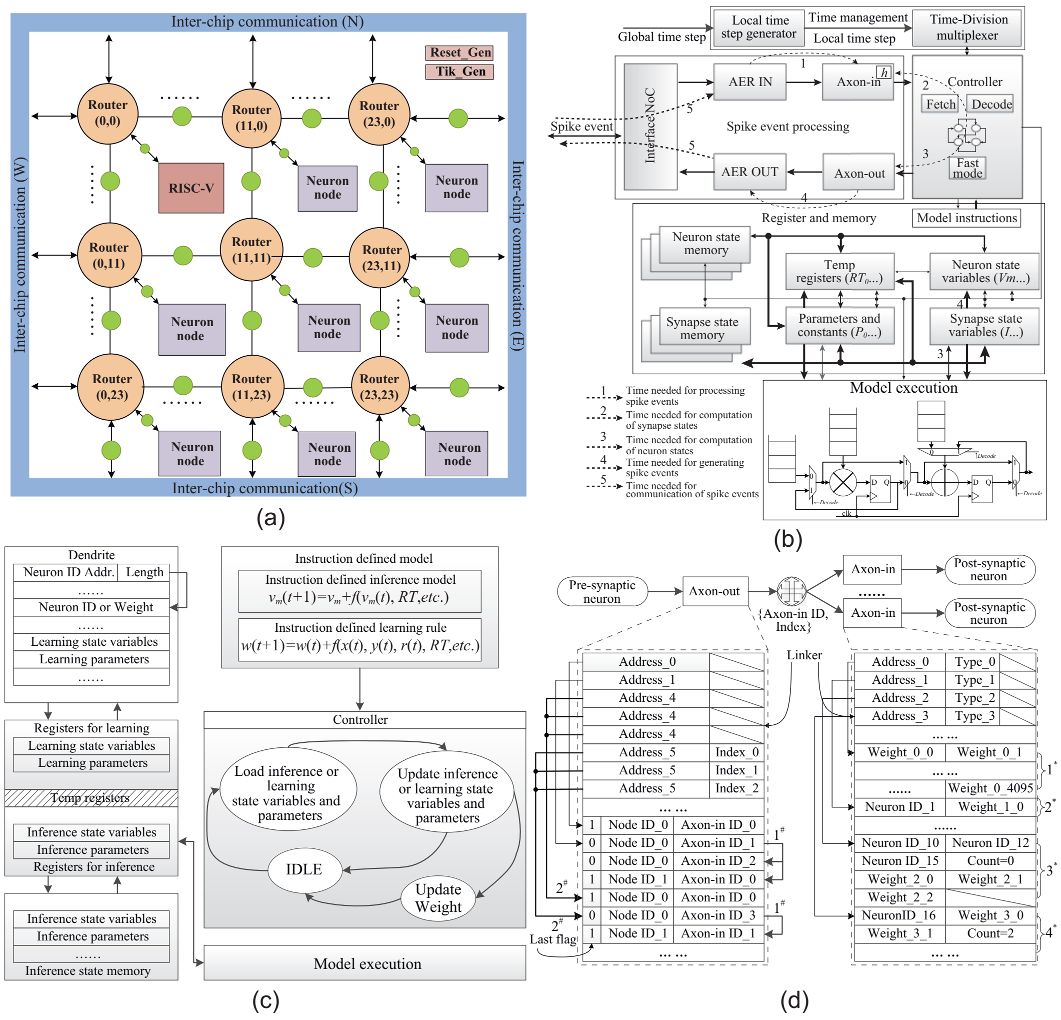


Figure 2. The architecture of the chip top and main blocks. (a) The top architecture of the proposed chip. (b) The architecture of a neuron core. (c) The architecture for the inference and learning processes. (d) The architecture of the synapses.

management, while the other nodes, functioning as neuron cores, handle the majority of computations, with each supporting up to 4096 spiking neurons. Inter-chip communication modules are placed at four edges of the chip connected with peripheral routers, acting as compression and decompression units. This design enables the NoC to extend connections to other chips in all four cardinal directions, enhancing system scalability.

This work implements a low-latency NoC architecture, which employs XY routing as detailed in [31]. The design is further improved by integrating the CXY [32] and OE-FAR [33] routing strategies to tackle congestion issues. Additionally, a new routing algorithm is introduced in this work that uses the relative offsets between the source and destination addresses as the basis for its routing scheme. This strategic decision simplifies the data packet transmission process to neighboring chips, eliminating the need for complex routing protocols and address translation.

The asynchronous communication interfaces, denoted by small circles in Fig. 2, interconnect local synchronous modules, establishing Darwin3 as a global asynchronous local synchronous system. This enhances the capability of each node on the chip to operate independently at a high-performance level.

Architecture of neuron cores

The architecture of a neuron core, illustrated in Fig. 2b, comprises five components: the controller unit, the model execution unit, the time management unit, the register and memory units, and the spike event processing unit.

The controller unit is responsible for fetching, decoding and executing flow control. The model execution unit can perform various arithmetic and logical operations. As defined in Table 1, registers store state variables, parameters, constants and temporary variables.

The time management unit has two primary responsibilities. First, it generates an internal tick signal based on global time-step information, indicating the progression of time steps within the core. Second, it implements time-division multiplexing for 1–4096 logical neurons based on configuration information.

Each neuron core has memories for different things like axon-in, axon-out, neuron state variables, synapse state variables and instructions. Instructions are only used to describe how neurons and synapses work. The neuron's ID determines the address of instructions and related state variables. The memories for axon-in and axon-out store how neurons are connected, and their organization is shown in Fig. 2d. When the chip starts, we need to set up the memories for the working nodes. Configuration data will be transported from the external controller (e.g., a PC or an Field Programmable Gate Array (FPGA)) to the corresponding nodes through the inter-chip communication module.

Unlike conventional processors that fetch instructions in every cycle using a clock, Darwin3's neuron cores are driven by spike events. When a neuron gets a spike, address-event representation (AER) IN queries the corresponding axon-in entry to find the neuron ID and weight, calculating the state variable h. When a time step advances, the controller unit performs computations for each neuron's inference or learning stage based on the instructions. If a neuron fires a spike, the AER OUT gets the address and ID of the post-synaptic neuron from the axon-out, packaging this in a spike data packet.

The dashed lines in Fig. 2b illustrate the process of a neuron core receiving, processing, generating and transmitting spikes. Multiplication operations take two cycles, while addition operations take one cycle. For example, the commonly used LIF neuron model requires four cycles (two multiplication and two addition operations), while the CUBA Delta model requires three cycles (one multiplication and one addition operation). Transmission delay is expressed as $2N + 2(N + 1)$, where $2N$ is for delay through N routers, and $2(N + 1)$ is for delay through $N + 1$ asynchronous interconnections between pre-synaptic and post-synaptic neurons.

Figure 2c shows the architecture tailored for inference and learning based on the proposed ISA. In the inference mode, the controller unit updates the state variables of each neuron described by the instructions within the current time step. In the learning mode, the controller unit extracts learning parameters and state variables to execute necessary calculations and updates, calculating new weights. The axon-in memory area has been reconfigured to accommodate learning-related parameters and state variables to optimize hardware resources.

Table 5. Different connectivity mechanisms.

Connectivity mechanism	Max. fan-in/core	Max. fan-out/core
Crossbar [10]	C	R
Normal index [6]	D_1	D_2
Synaptic expansion [11]	D_1	D_2M
Population-based index [5]	D_1N	D_2
Flexible compression for Darwin3	$(D_1 - 1)N$	$(D_2 - N)N$

D_1 represents the fan-in memory depth (commonly associated with axon-in structures), D_2 represents the fan-out memory depth (commonly linked to axon-out structures), M represents the number of neuron cores and N represents the number of neurons within a neuron core. By R and C we represent the dimensions of the crossbar, with R being equivalent to D_1 (rows) and C being equivalent to D_2 (columns).

Representation of neuronal connections

A flexible connection representation mechanism is essential in pursuing the development of neuromorphic computing chips capable of supporting complex networks. Several connection topologies find frequent application in SNNs.

(1) Multiple neuron groups connect to a group of neurons, similar to the convolutional neural network (CNN) arrangement with shared weights.
(2) A single neuron connects to an entire group of neurons.
(3) A group of neurons fully connects to another group of neurons.

Upon a comprehensive examination of commonly employed connection expression mechanisms (as summarized in Table 5, we discovered that the approach used by Loihi [5] stands out for its exceptional flexibility, featuring substantial fan-in and fan-out capabilities. Combining these advantageous attributes, we have introduced a novel scheme that enables a highly compressed representation of connection topology, as depicted in Fig. 2d. To efficiently represent the topology of connections, each neuron core has independent memories for axon-out and axon-in within this framework.

Spikes are conveyed utilizing the AER method. Following the generation of a spike by a pre-synaptic neuron, it accesses axon-out to retrieve the target node's address and axon-in index information of the target node. Subsequently, the AER OUT module encapsulates and transmits this information to the router through the local connection port. The router, in turn, directs the data packet towards the designated target node. Upon reception of the data packet, the target node queries axon-in to acquire pertinent information concerning the target neuron and connection weights.

Within the framework of the axon-out structure, each operational neuron is associated with a linker. The linker's entries retain the address of the entry containing detailed connection information and the specific index of the neuron. This index distinguishes cases in which multiple neurons are connected to the same target node. This structural configuration optimizes the compression of information for connection types extending beyond point-to-point scenarios.

(1) The last flag is set to 0 when a neuron connects to multiple nodes, indicating non-terminal nodes and facilitating efficient compression of redundant information. As shown in Fig. 2d, the situation is represented by $1^{\#}$.
(2) When multiple neurons connect to the same node(s), a single entry suffices for their connectivity information. As shown in Fig. 2d, the situation is represented by $2^{\#}$.

Each received axon-in index aligns with a corresponding linker within the axon-in structure context. The entries in the linker encapsulate the address of the entry, housing detailed connection information and the type of connection. This structure is strategically designed to optimize information compression, particularly tailored for connection types extending beyond point-to-point scenarios.

(1) When all 4096 neurons within the node are connected to one pre-synaptic neuron, there is no necessity to store neuron indexes individually. In such cases, 4096 weights can be stored sequentially. As depicted in Fig. 2d, the case is denoted 1^{*}.
(2) In instances where multiple neurons are connected to a specific neuron within the node and share the same weights, storing a single neuron index along with the corresponding weight suffices. As depicted in Fig. 2d, the case is denoted 2^{*}.
(3) When neurons from a remote cluster are connected to a group of neurons within the target node, it is necessary to have only one instance of neuron indexes, and weights can be stored systematically based on the order of the source neurons. As depicted in Fig. 2d, the case is denoted 3^{*}.
(4) When the target neurons are organized sequentially, it becomes sufficient to store only the index of the initial neuron and the count of the target neurons, further reducing storage demands. As depicted in Fig. 2d, the case is denoted 4^{*}.

This structure also facilitates the incorporation of weights with different bit widths, allowing diverse weights to be accommodated within a shared entry, consequently improving storage density.

EXPERIMENTAL RESULTS

To evaluate the proposed ISA and architecture, we first implemented the entire architecture in Verilog at the register transfer level. Using the GLOBAL FOUNDRIES 22-nm Fully Depleted Silicon On Insulator process, we generated a GDSII file that meets the sign-off requirements after completing physical design and verification.

After the initial chip-on-board testing in December 2022, the chip was repackaged using flip-chip Ball Grid Array, and a dedicated test system board featuring a Xilinx seven-series FPGA was assembled. Figure 3a illustrates the system board, chip layout and main blocks. The chip's structure is organized into a grid of 6 × 6 groups, each consisting of 4 × 4 tiles. Each tile comprises a node connected to a router. Except for the RISC-V node, all nodes on the chip are neuron cores, collectively driving their computational functions. Notably, two distinct tile types exist, primarily differing in the size of their axon-in memory.

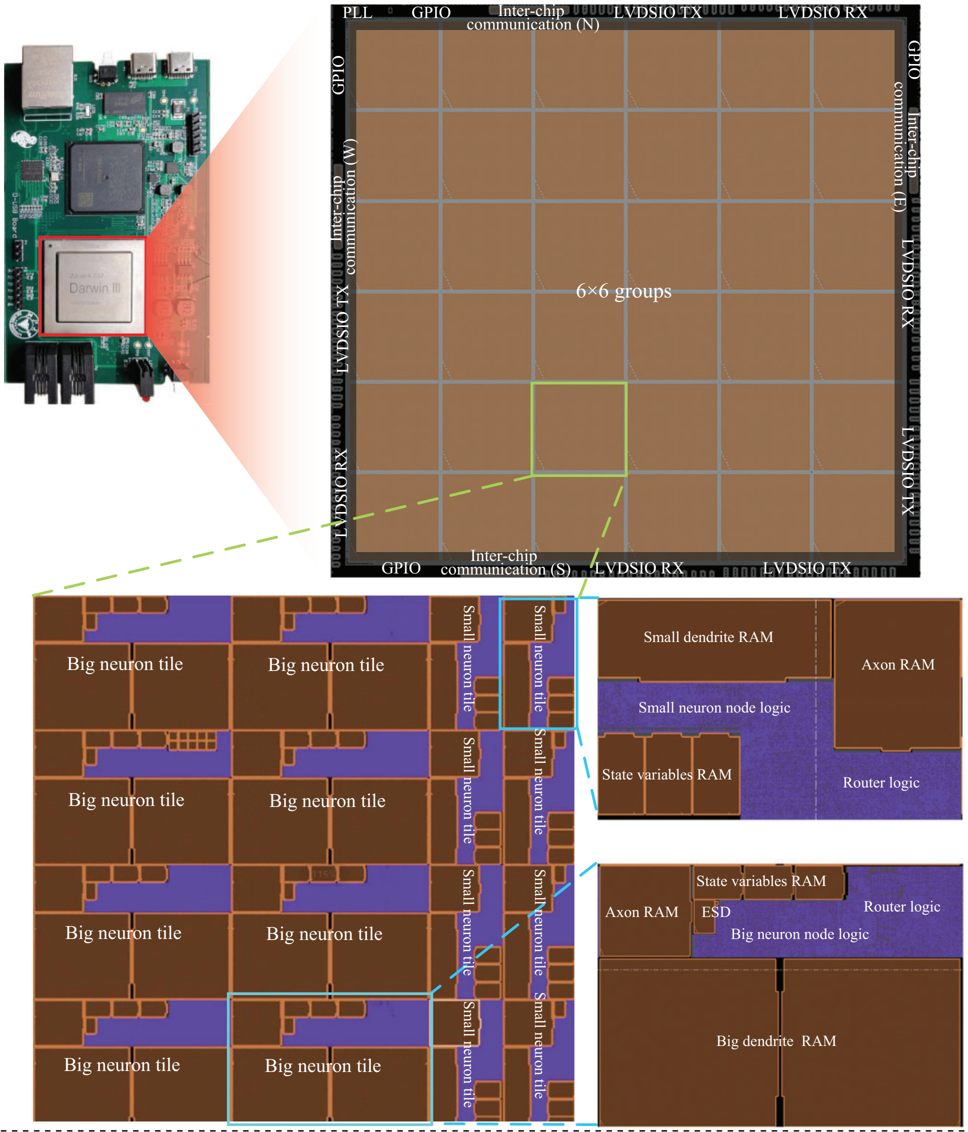
The Wuyuan framework is a customized software toolchain designed to meet the requirements of Darwin3 based on the reference software architecture [41]. Figure 3b illustrates the primary stages of application development. Users can use this toolchain to convert their existing artificial neural network (ANN) models to SNN models or create new SNN models from scratch. The development process involves merging different business logic. In addition, the toolchain allows users to compile and deploy applications that can run on Darwin3 chips.

We first compare some important metrics with the current state-of-the-art works, and then we run some application demonstrations to verify the chip's functionalities and performance.

Comparison with the state-of-the-art neuromorphic chips

Table 6 summarizes the performance and specifications of state-of-the-art neuromorphic chips. Mixed-signal designs with analog neurons and synapse computation and high-speed digital peripherals are grouped on the left [4,12,34], and digital designs, including Darwin3, are grouped on the

(a)



(b)

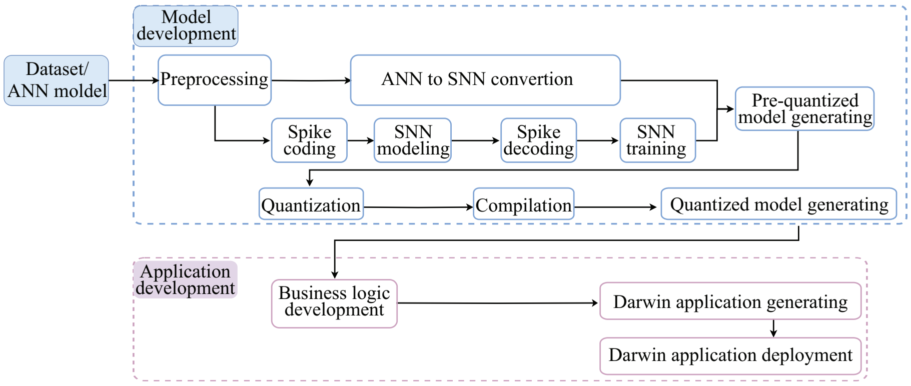


Figure 3. Test environment. (a) The test chip and system board. (b) Application development process.

right [5–8,10,11,30,35–37]. The critical metrics for efficient spiking neuromorphic hardware platforms are the scale of neurons and synapses, model construction capabilities, synaptic plasticity and the energy per synaptic operation.

Neuron number

The quantity of neurons and synapses directly determines the size and complexity of the spiking neural network that a neuromorphic chip can support, which is extremely important. However, a

Table 6. Performance and specifications of state-of-the-art neuromorphic chips.

Chip name	Neurogrid [4]	DYNAPs [34]	Brainscales2 [12]	SpiNNaker [7]	SpiNNaker2[#2] [35]	TrueNorth [10]	Loihi [5]	FlexLearn [6]	ISSCC 2019 [36]	ODIN [37]	Loihi2 [8]	Unicorn [11]	ANP-I [30]	Darwin3[#1]
Implementation	Mixed	Mixed	Mixed	Digital	Digital	Digital	Digital	Digital	Digital	Digital	Digital	Digital	Digital	Digital
Technology (nm)	180	180	65	130	22	28	14	45	65	28	7[#3]	28	28	22
Die area (mm^2)	–	43.79	–	102	8.76	430	60	410.5	10.08[#4]	–	31	500	1.628	358.527
Neuron cores		4	4	18	8	4K	128	128	1	1	128	36	64	575
Neurons per core	64K	256	128	Prog.	Prog.	256	max. 1K	max. 35	400	256	max. 8K	1K	8	max. 4K
Fan in/out per core		64/4K	512/256	Prog.	Prog.	256/256	4K–4M/4K	–	–	–	–[#5]	256–256K/-	–	28K–256M/12K–48M
Synaptic weight	4 bits	12 bits	6 bits	Prog.	Prog.	1 bit	1 to 9 bits	–	14 bits	3 bits	1 to 9 bits	4 bits	8, 10 bits	1/2/4/8/16 bits
Neuron models	LIF	AdEx-IF	AdEx-IF	Prog.	Prog.	LIF	LIF	Config.	–	LIF	Prog.	LIF	LIF[#6]	Prog.
Synapse models	COBA Delta	COBA NMDA	CUBA/COBA Alpha	Prog.	Prog.	CUBA Delta	CUBA Delta	Config.	CUBA Delta	CUBA Delta	–	CUBA Delta	CUBA Delta	Prog.
On-chip learning	No	No	STP/STDP/R-STDP	Prog.	Prog.	No	STDP based	Config.	Mod.SD	SDSP	Prog.	No	S-TP	Prog.
Energy per SOP	941pJ @3.0 V	417 fJ [38] @1.8 V	–	11.3 nJ [39] @1.2 V	10 pJ @0.5V	26 pJ [40] @0.775 V	23.6 pJ[#7] @0.75 V	–	–	8.4 pJ @0.55 V	–	–	1.5 pJ @0.56 V	5.47 pJ @0.8 V

#1 Darwin3 allows nodes to operate at different frequencies, with internal modules typically running at 300–400 MHz.

#2 A test chip contains two QPEs with eight PEs, while a full SpiNNker2 has 38 QPEs with 152 PEs.

#3 Loihi2 has been implemented in Intel 4, equivalent to the 7 nm process.

#4 These data are obtained through a digital synthesis flow, not from the final silicon tape-out data.

#5 Loihi2's "Axon Routing", which refers to fan-out or fan-in, has a topology compression of 256 ×.

#6 Its output layer consists of 10 integrate-and-fire (IF) neurons.

#7 A minimum synaptic operation (SOP) energy of 23.6 pJ at 0.75 V is extracted from pre-silicon simulations.

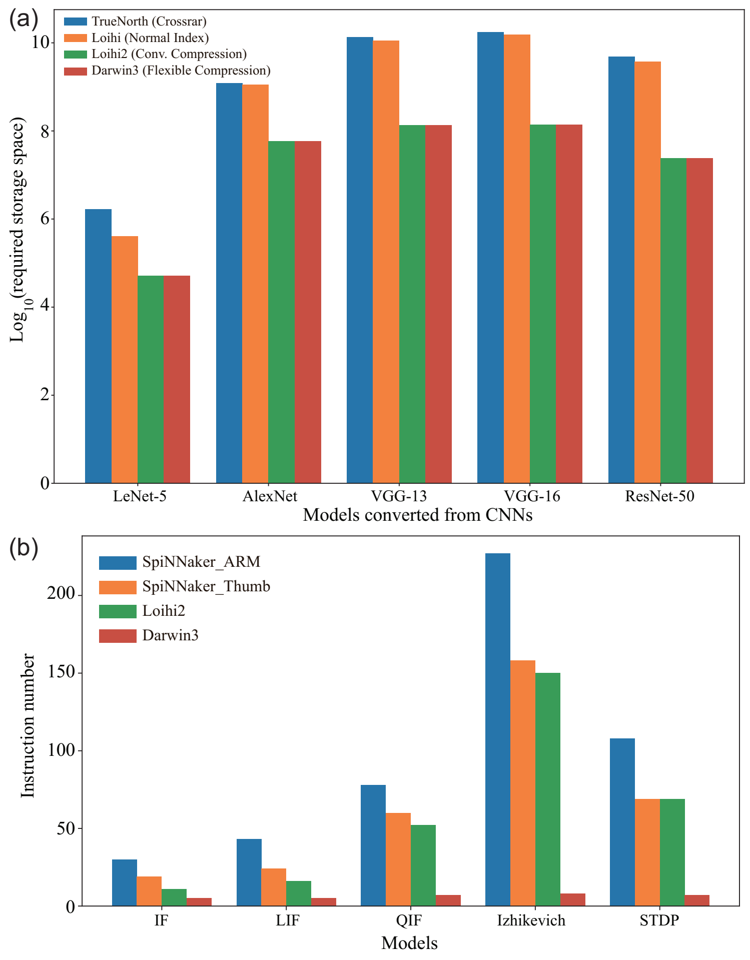


Figure 4. Comparison of the code density and memory usage. (a) Comparison of the required weight memory across typical networks. (b) Comparison of the code density.

direct comparison with the SpiNNaker chips [7,35] is not feasible due to its use of ARM processors, where the scale is tied to the size of the off-chip memory. Among other chips, NeuroGrid [4] has the largest number of neurons in a single neuron core, reaching 64K. Loihi [5], Unicorn [11], Loihi2 [8] and Darwin3 are at a similar level, boasting neuron counts exceeding 1K. At the chip level, Darwin3 can support up to 2.35 million neurons, surpassing the scale of TrueNorth [10] and Loihi2 [8] by more than two times.

Synapse capacity

The capabilities of fan-in and fan-out within each neuron core profoundly impact the chip's overall capacity of synapses, as detailed in Table 5. Darwin3 distinguishes itself with its adaptive axon-out and axon-in memory configurations, coupled with efficient compression mechanisms, enabling remarkable fan-in and fan-out capacities of up to $(D_1 - 1)MN$ and $(D_2 - N)N^2$, respectively. In the case of Darwin3, the compression mechanism yields a maximum fan-in improvement of 1024 × and a maximum fan-out improvement of 2048 × when compared to the physical memory depth.

While the previous discussion delved into fan-in and fan-out capabilities, focusing on the synaptic connectivity potential, the challenge of efficiently storing synaptic weight parameters remains crucial. In Fig. 4(a), we present a comparative analysis of weight storage requirements, highlighting the stark contrast between Darwin3 and existing approaches when applied to typical networks converted from CNNs. Chips lacking specialized compression mechanisms exhibit dense weight matrices, making memory usage 6.8 × to 200 × larger than the original approach. In crossbar designs, neurons consistently occupy their unique space, contributing to additional inefficiencies.

Darwin3 employs a versatile mechanism by classifying convolutional connections into weight-sharing multi-to-multi forms and obtains storage parity with the initial parameters, thereby achieving efficiency comparable to Loihi2 [8]. Importantly, this advantage extends to non-convolutional connections featuring shared weight parameters. Darwin3 enables instructional access to the complete axon-in, thus realizing the factorized attribute, which is also supported by Loihi2 [8], through multiplication operations. Furthermore, Darwin3 offers compatibility with diverse weight-bit widths, enhancing its adaptability and storage efficiency.

Code density

Code density is a meaningful ISA metric, so we compare the code density of Darwin3 with the SpiNNaker chips [7,35] and Loihi2 [8], the outstanding neuromorphic chips based on ISA. We use the C code to describe a model and the spinnaker tools integrated by SpyNNaker [48] to generate assembly code for the SpiNNaker chips. Then, we compare the length of the assembly code in Fig. 4(b). Loihi2's RISC instruction set is similar to ARM's Thumb, where spike instructions aid in curtailing spike-related instruction codes, offering a slight edge over SpiNNaker [7]. Darwin3 shows an advantage in code density because of our proposed instructions. This instruction set concurrently loads parameters and expedites multiplication and addition with multiple parameters. Impressively, Darwin3 gets a remarkable 2.2 × to 28.3 × code density advantage across distinct models.

Inference and learning performance

For researchers working on SNNs, after finalizing the model, the primary focus lies on evaluating the chip's performance during application execution,

Table 7. Performance comparison with other chips: learning mode.

Platform	SpiNNaker [39]	Loihi [47]	ISSCC 2019 [36]	ODIN [37]	ANP-I [30]	Darwin3
Frequency	150 MHz	–	20 MHz	150 MHz	40 MHz	333MHz
Dataset	MNIST					
Learning algorithm	CD	EMSTDP	Mod.SD	SDSP	S-TP	RSTDP
Weight precision[#1]	16 bits	8 bits	14 bits	3 bits	8 bits[#2]	16 bits
Network topology	784-500-500-10	–	(784)-200-200-10	(256)-10	(1024)-512-10	(784)-100-100-10
Accuracy	95.01%	94.70%	97.83%	84.50%	96.00%	96.00%

#1 The weight precision here refers to the precision of the network run in the experiment rather than the maximum weight precision of the chip.
#2 The synaptic weights of the 10 neurons in the output layer are 10 bits.

with particular attention to latency and accuracy. To evaluate the capabilities of Darwin3, we conducted several experiments under two distinct scenarios: inference and learning. Table 7 compares the performance of Darwin3 to state-of-the-art neuromorphic chips in typical applications. These applications were SNNs converted from trained and quantified ANNs. We implemented the same type of network models on Darwin3, and the performance metrics indicate that Darwin3 is in the leading position regarding accuracy and latency. The accuracy is up to 6.76% higher, and the latency is up to 4.5 $\times$ better than others. Darwin3 exhibits advantages because it has a flexible and efficient connection construction ability, which is very friendly to the converted convolutional networks. Because of the high efficiency of connection storage, it does not increase redundant spike transmission latency. The asynchronous interconnection method employed by Darwin3 has significantly reduced the communication delay between neuron cores. Darwin3 utilizes click elements [49] to construct a cross-clock domain structure, enabling the completion of cross-clock domain data transfer in just two cycles. Furthermore, the related topological structures can be split and computed in parallel with more neuron cores. We attribute the observed discrepancies to the quantization operations while mapping these models to hardware, and the quantization methods employed may vary among different approaches. It is important to note that there is still room for improving latency performance by optimizing the mapping approach.

To further evaluate the on-chip learning capability of Darwin3, we constructed a network based on the architecture proposed by Diehl and Cook [50]. We added a supervision layer, which provides positive or negative rewards based on comparing the network's output and the target during the training process, achieving the overall implementation of the RSTDP rule. The network was trained directly on Darwin3 with a weight precision of 16 bits and we achieved a classification accuracy of 96.0%. Table 8 presents the experimental results compared with prior works, demonstrating that Darwin3 is in the leading position regarding accuracy. The Mod.SD algorithm is hardware-specific and performs slightly better, while Darwin3 allows flexible construction of multiple learning rules. We plan to optimize the current learning algorithm or introduce new ones to improve performance.

Energy efficiency

The energy consumption of each synaptic operation (SOP) is the most critical energy consumption metric for neuromorphic chips. We measured the energy consumption of the Darwin3 chip when running a two-layer neural network, where the neurons in the first layer can fire spikes without inputs, and the neurons in the second layer receive spikes and perform calculations. We select the common approach [39] to evaluate energy consumption, as detailed in

$$P_{total} = P_I + P_B + (P_N \times n) + (P_S \times s), \quad (7)$$

where P_I is the power dissipated by a Darwin3 chip after the power-up process with no applications configured, P_B is the baseline power, which consists of the power dissipated by all nodes enabled without running any neurons on it, P_N is the power required to simulate an LIF neuron with a 1-ms time step, n is the total number of neurons, P_S is the energy consumed per synaptic event (activation of neural connections) and s is the total synaptic events. The chip operates at a frequency 333 MHz with a core voltage supply of 0.8 V and an IO voltage supply of 1.8 V, as shown in Table 6. The measured average SOP power consumption is 5.47 pj/SOP. This metric is directly influenced by factors such as the manufacturing process, power supply voltage and operating frequency, making fair comparison challenging. However, based on the data released by prior works under typical scenarios, Darwin3 boasts a leading achievement. Darwin3's advantage lies in its internal asynchronous interconnection circuit, which enables the chip to consume very low power when there is no spike transmission or calculation.

Table 8. Performance comparison with other chips: inference mode.

Platform	TrueNorth [42,43]		Loihi [44,45]		SpiNNaker [39]	ReckOn [46]	ANP-I [30]		Darwin3				
Weight precision[#1]	8 bits	8 bits	16 bits	9 bits	8 bits	8 bits	8, 10 bits[#2]		8 bits	16 bits	8 bits	8 bits[#3]	8 bits
Frequency	–		–		150 MHz	13MHz	210MHz		333MHz				
Dataset	MNIST	CIFAR-10	MNIST	IBM Gesture	MNIST	IBM Gesture	N-MNIST	IBM Gesture	MNIST	MNIST	MNIST	IBM Gesture	CIFAR-10
Network topology	LeNet	Mod. VGG	VGG-9	cNet	DBN	RNN	–		LeNet	VGG-9	RNN	cNet	Mod. VGG
Accuracy	99.40%	83.41%	99.79%	89.64%	95.01%	87.30%	96.00%	92.00%	99.10%	99.79%	87.51%	89.60%	90.17%
Latency	5.74 ms	–	6.13 ms	–	20 ms	15 ms	–		5.7 ms	6.48 ms	2.7 ms	6.08 ms	9.88 ms

#1 The weight precision here refers to the precision of the network run in the experiment rather than the maximum weight precision of the chip.

#2 The synaptic weights of the 10 neurons in the output layer are 10 bits.

#3 Darwin3 cannot support 9 bits of weight precision.

Additionally, all memories of Darwin3 will shut down during the idle phase, reducing power consumption.

Applications with a million of neurons

To further illustrate the chip's efficacy, we developed two extensive applications implemented on Darwin3, spiking VGG-16 ensembling and directly trained [51] SNN-based maze solving, shown in Fig. 5. We ensembled outputs of five VGG-16 models obtained through ANN2SNN [52] using a voting mechanism, culminating in a composite model comprising approximately 1.05 million neurons and employing an 8-bit weight precision. We applied random transformations to the input and used five independent VGGs in the hidden layers for the classification tasks. The voting layer produces the final classification outcome based on the collective votes from the individual outputs of the hidden layers. Compared to the original single VGG-16, accuracy testing on the CIFAR-10 dataset witnessed an increase from 92.98% to 93.48%.

We also developed an application for maze solving. We mapped the maze onto a set of neurons [9], where excitatory neurons represent the free-walking grid points, and inhibitory neurons represent obstacles. The interconnected excitatory neurons can transmit spikes in sequence, and under the action of STDP rules, the synaptic weights are continuously increased to form a stable synaptic strength. However, synapses connected to inhibitory neurons cannot be strengthened, and the transmission of spikes will be terminated when encountering inhibitory neurons. After learning, the model can quickly find the path by observing the path along which the spikes propagate. We conducted experiments using mazes of different sizes, comparing the time it takes to search for a path on our chip versus a CPU server. A map of size 15 434 $\times$ 1534 requires over 2.35 million neurons, approaching the upper limit of neurons that Darwin3 can simulate. The result is shown in Table 9. With the STDP-based SNN method, the time consumed increases linearly with the maze size. In contrast, the traditional search method on the CPU server consumes a lot of time because it relies on many recursive operations.

CONCLUSION

The article proposes a new instruction set and a connectivity compression mechanism to create a chip that can support large-scale neural networks. This chip has been designed to be more efficient in terms of the number of neurons it can accommodate and its synaptic computing performance, compared

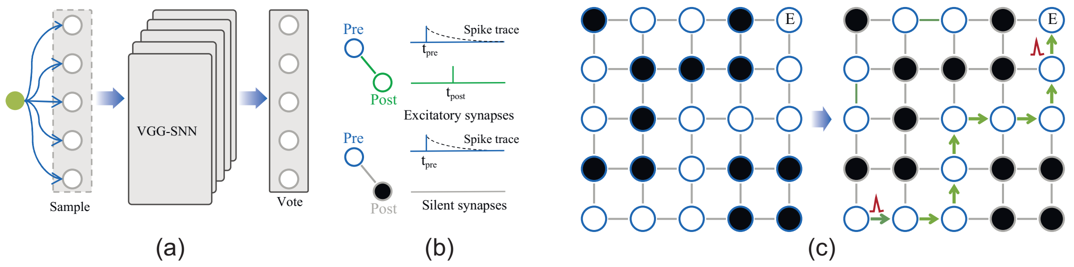


Figure 5. Two large-scale applications with a million neurons. (a) Spiking VGG-16 ensembling. (b) Directly trained SNN-based maze solving.

Table 9. Time cost for the maze-solving application.

Maze size (neuron #)	Time of Intel Xeon Gold 6248R @ 3 GHz, 205 W (ms)	Time of Darwin3 @ 400 MHz, 1.8 W (ms)
63 × 63	83	128
125 × 125	296	412
250 × 250	1132	1375
500 × 500	4744	4200
600 × 600	7968	6248
700 × 700	11 283	7706
900 × 900	FAIL	9654
1534 × 1534	FAIL	22 089

The mazes are randomly generated, and the running time is an average of five measurements.

to existing works. The experimental results show that the chip has reached the same leading level as the state-of-the-art works in terms of accuracy and latency performance metrics, both for inference and learning modes. The practical effectiveness of the chip has also been demonstrated by running a maze-searching application on it.

Because of the chip's versatile chip communication mechanism, different Darwin3 chips can be integrated onto a single board and we can interconnect several boards to configure a big chassis. These chassis can be interconnected through a network infrastructure to support the construction of extensive SNNs. This configuration can support the construction of extensive SNNs when coupled with suitable software frameworks.

FUNDING

This work was supported by the National Key R&D Plan of China (2022YFB4500100), the National Natural Science Foundation of China (61925603, U20A20220 and 62076084), the Key Research Project of Zhejiang Lab (2021KC0AC01), the Key Research and Development Program of Zhejiang Province in China (2020C03004) and the Key R&D Program of Zhejiang (2022C01048).

AUTHOR CONTRIBUTIONS

G.P. and D.M. proposed the idea. D.M., X.J., S.S. and Y.L. proposed the chip's architecture and implemented the design. X.J., S.S., Y.H., F.Y., H.T., P.L. and X.Z. built the experimental system. D.M., X.J. and S.S. completed the experiment. D.M., X.J. and X.W. wrote the article. All authors analyzed the data, discussed the results and commented on the manuscript.

Conflict of interest statement. None declared.

REFERENCES

1. Stimberg M, Brette R, Goodman DF. Brian 2, an intuitive and efficient neural simulator. *Elife* 2019; **8**: e47314.
2. Diesmann M and Gewaltig MO. NEST: an environment for neural systems simulations. *GWDG-Bericht* 2003; **58**: 43–70.
3. Hong C, Yuan M, Zhang M *et al.* SPAIC: a spike-based artificial intelligence computing framework. *IEEE Comput Intell Mag* 2024; **19**: 51–65.
4. Benjamin BV, Gao P, McQuinn E *et al.* Neurogrid: a mixed-analog-digital multichip system for large-scale neural simulations. *Proc IEEE* 2014; **102**: 699–716.
5. Davies M, Srinivasa N, Lin TH *et al.* Loihi: a neuromorphic many-core processor with on-chip learning. *IEEE Micro* 2018; **38**: 82–99.
6. Baek E, Lee H, Kim Y *et al.* Flexlearn: fast and highly efficient brain simulations using flexible on-chip learning. In: *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. New York: Association for Computing Machinery 2019, 304–18.
7. Painkras E, Plana LA, Garside J *et al.* SpiNNaker: a 1-W 18-core system-on-chip for massively-parallel neural network simulation. *IEEE J Solid-State Circuits* 2013; **48**: 1943–53.
8. Orchard G, Frady EP, Rubin DBD *et al.* Efficient neuromorphic signal processing with loihi 2. In: *2021 IEEE Workshop on Signal Processing Systems (SiPS)*, Piscataway: IEEE Press, 2021, 254–9.
9. Jin O, Xing Q, Li Y *et al.* Mapping very large scale spiking neuron network to neuromorphic hardware. In: *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, Vol. 3. New York: Association for Computing Machinery, 2023, 419–32.

10. Akopyan F, Sawada J, Cassidy A *et al.* TrueNorth: design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip. *IEEE Trans Comput-Aided Des Integr Circuits Syst* 2015; **34**: 1537–57.
11. Yang Z, Wang L, Wang Y *et al.* Unicorn: a multicore neuromorphic processor with flexible fan-in and unconstrained fan-out for neurons. In: *Proceedings of the 59th ACM/IEEE Design Automation Conference*. New York: Association for Computing Machinery, 2022, 943–8.
12. Pehle C, Billaudelle S, Cramer B *et al.* The brainscales-2 accelerated neuromorphic system with hybrid plasticity. *Front Neurosci* 2022; **16**: 795876.
13. Chen P, Liu F, Lin P *et al.* Open-loop analog programmable electrochemical memory array. *Nat Commun* 2023; **14**: 6184.
14. Ma D, Shen J, Gu Z *et al.* Darwin: a neuromorphic hardware co-processor based on spiking neural networks. *J Syst Archit* 2017; **77**: 43–51.
15. Burkitt AN. A review of the integrate-and-fire neuron model: I. Homogeneous synaptic input. *Biol Cybern* 2006; **95**: 1–19.
16. Izhikevich EM. Simple model of spiking neurons. *IEEE Trans Neural Netw* 2003; **14**: 1569–72.
17. Caporale N and Dan Y. Spike timing–dependent plasticity: a Hebbian learning rule. *Annu Rev Neurosci* 2008; **31**: 25–46.
18. Smith GD, Cox CL, Sherman SM *et al.* Fourier analysis of sinusoidally driven thalamocortical relay neurons and a minimal integrate-and-fire-or-burst model. *J Neurophysiol* 2000; **83**: 588–610.
19. Ermentrout B. Type I membranes, phase resetting curves, and synchrony. *Neural Comput* 1996; **8**: 979–1001.
20. Brette R and Gerstner W. Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *J Neurophysiol* 2005; **94**: 3637–42.
21. Hodgkin A and Huxley A. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Bull Math Biol* 1990; **52**: 25–71.
22. Hodgkin AL. The local electric changes associated with repetitive action in a non-medullated axon. *J Physiol* 1948; **107**: 165–81.
23. Choquet D and Triller A. The dynamic synapse. *Neuron* 2013; **80**: 691–703.
24. Roth A and van Rossum MC *et al. Modeling synapses*. In: De Schutter E (ed) *Computational Modeling Methods for Neuroscientists*. Cambridge: MIT Press, 2009, 139–60.
25. Citri A and Malenka RC. Synaptic plasticity: multiple forms, functions, and mechanisms. *Neuropsychopharmacology* 2008; **33**: 18–41.
26. Sejnowski TJ and Tesauro G. The Hebb rule for synaptic plasticity: algorithms and implementations. In: Byrne JH and Berry WO (eds). *Neural Models of Plasticity*. New York: Academic Press, 1989, 94–103.
27. Cai W, Ellinger F, Tetzlaff R. Neuronal synapse as a memristor: modeling pair-and triplet-based STDP rule. *IEEE Trans Biomed Circuits Syst* 2014; **9**: 87–95.
28. Quintana FM, Perez-Pena F, Galindo PL. Bio-plausible digital implementation of a reward modulated STDP synapse. *Neural Comput Appl* 2022; **34**: 15649–60.
29. Brader JM, Senn W, Fusi S. Learning real-world stimuli in a neural network with spike-driven synaptic dynamics. *Neural Comput* 2007; **19**: 2881–912.
30. Zhang J, Huo D, Zhang J *et al.* ANP-I: a 28nm 1.5 pJ/SOP asynchronous spiking neural network processor enabling sub-0.1 μJ/sample on-chip learning for edge-AI applications. In: *2023 IEEE International Solid-State Circuits Conference (ISSCC)*, Piscataway: IEEE Press, 2023, 21–3.
31. Monemi A, Ooi CY, Marsono MN. Low latency network-on-chip router microarchitecture using request masking technique. *Int J Reconfig Comput* 2015; **2015**: 570836.
32. Ezz-Eldin R, El-Moursy MA, Hamed HF. Process variation delay and congestion aware routing algorithm for asynchronous NoC design. *IEEE Trans Very Large Scale Integr VLSI Syst* 2015; **24**: 909–19.
33. Liu L, Zhu Z, Zhou D *et al.* A fair arbitration for network-on-chip routing with odd-even turn model. *Microelectron J* 2017; **64**: 1–8.
34. Moradi S, Qiao N, Stefanini F *et al.* A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (DYNAPs). *IEEE Trans Biomed Circuits Syst* 2017; **12**: 106–22.
35. Höppner S, Yan Y, Dixius A *et al.* The SpiNNaker 2 processing element architecture for hybrid digital neuromorphic computing. arXiv: 2103.08392.
36. Park J, Lee J, Jeon D. A 65 nm 236.5 nJ/classification neuromorphic processor with 7.5% energy overhead on-chip learning using direct spike-only feedback. In: *2019 IEEE International Solid-State Circuits Conference (ISSCC)*, Piscataway: IEEE Press, 2019, 140–2.
37. Frenkel C, Lefebvre M, Legat JD *et al.* A 0.086-mm^2 12.7-pJ/SOP 64k-synapse 256-neuron online-learning digital spiking neuromorphic processor in 28-nm CMOS. *IEEE Trans Biomed Circuits Syst* 2018; **13**: 145–58.
38. Indiveri G, Corradi F, Qiao N. Neuromorphic architectures for spiking deep neural networks. In: *2015 IEEE International Electron Devices Meeting (IEDM)*, Piscataway: IEEE Press, 2015, 4.2.1–4.
39. Stromatias E, Neil D, Galluppi F *et al.* Scalable energy-efficient, low-latency implementations of trained spiking deep belief networks on SpiNNaker. In: *2015 International Joint Conference on Neural Networks (IJCNN)*, Piscataway: IEEE Press, 2015.
40. Merolla PA, Arthur JV, Alvarez-Icaza R *et al.* A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* 2014; **345**: 668–73.
41. Deng S, Lv P, Jin O *et al.* Darwin-s: a reference software architecture for brain-inspired computers. *IEEE Computer* 2022; **55**: 51–63.
42. Zou C, Cui X, Kuang Y *et al.* Mapping convolutional neural networks onto neuromorphic chip for spike-based computation. In: *2021 China Semiconductor Technology International Conference (CSTIC)*, Piscataway: IEEE Press, 2021, 1–3.
43. Steven K, Merolla P, Arthur J *et al.* Convolutional networks for fast, energy-efficient neuromorphic computing. *Proc Natl Acad Sci USA* 2016; **113**: 11441–6.
44. Chandarana P, Mohammadi M, Seekings J *et al.* Energy-efficient deployment of machine learning workloads on neuromorphic hardware. In: *2022 IEEE 13th International Green and Sustainable Computing Conference (IGSC)*, Piscataway: IEEE Press, 2022.
45. Massa R, Marchisio A, Martina M *et al.* An efficient spiking neural network for recognizing gestures with a DVS camera on the Loihi neuromorphic processor. In: *2020 International Joint Conference on Neural Networks (IJCNN)*, Piscataway: IEEE Press, 2020.
46. Frenkel C and Indiveri G. ReckOn: A 28nm sub-mm2 task-agnostic spiking recurrent neural network processor enabling on-chip learning over second-long timescales. In: *2022 IEEE International Solid-State Circuits Conference (ISSCC)*, Piscataway: IEEE Press, 2022.
47. Shrestha A, Fang H, Rider DP *et al.* In-hardware learning of multilayer spiking neural networks on a neuromorphic processor. In: *2021 58th ACM/IEEE Design Automation Conference (DAC)*, Piscataway: IEEE Press, 2021, 367–72.
48. Rhodes O, Bogdan PA, Brenninkmeijer C *et al.* sPyNNaker: a software package for running PyNN simulations on SpiNNaker. *Front Neurosci* 2018; **12**: 816.
49. Peeters A, Te Beest F, De Wit M *et al.* Click elements: An implementation style for data-driven compilation. In: *2010 IEEE Symposium on Asynchronous Circuits and Systems*, Piscataway: IEEE Press, 2010, 3–14.
50. Diehl PU and Cook M. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Front Comput Neurosci* 2015; **9**: 99.

51. Feng L, Liu Q, Tang H *et al.* Multi-level firing with spiking ds-resnet: Enabling better and deeper directly-trained spiking neural networks. In: *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, Red Hook: Curran Associates, 2022, 2471–7.

52. Hu Y, Zheng Q, Jiang X *et al.* Fast-SNN: fast spiking neural network by converting quantized ANN. *IEEE Trans Pattern Anal Mach Intell* 2023; **45**: 14546–62.