Qu P, Ji XL, Chen JJ *et al.* Research on general-purpose brain-inspired computing systems. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 39(1): 4-21 Jan. 2024. DOI: 10.1007/s11390-023-4002-3

Research on General-Purpose Brain-Inspired Computing Systems

Peng Qu[1, †] (渠　鹏), Xing-Long Ji[2, †] (纪兴龙), Jia-Jie Chen[1, †] (陈嘉杰), Meng Pang[1, †] (庞　猛)
Yu-Chen Li[1, †] (李宇晨), Xiao-Yi Liu[1, †] (刘晓义), and You-Hui Zhang[1, *] (张悠慧), *Senior Member, CCF*

[1] *Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China*
[2] *Department of Precision Instrument, Tsinghua University, Beijing 100084, China*

E-mail: qp2018@mail.tsinghua.edu.cn; xinglongji@mail.tsinghua.edu.cn; cjj21@mails.tsinghua.edu.cn
pangm20@mails.tsinghua.edu.cn; liyuchen20@mails.tsinghua.edu.cn; xiaoyi-l17@mails.tsinghua.edu.cn
zyh02@tsinghua.edu.cn

Received December 3, 2023; accepted January 5, 2024.

Abstract　Brain-inspired computing is a new technology that draws on the principles of brain science and is oriented to the efficient development of artificial general intelligence (AGI), and a brain-inspired computing system is a hierarchical system composed of neuromorphic chips, basic software and hardware, and algorithms/applications that embody this technology. While the system is developing rapidly, it faces various challenges and opportunities brought by interdisciplinary research, including the issue of software and hardware fragmentation. This paper analyzes the status quo of brain-inspired computing systems. Enlightened by some design principle and methodology of general-purpose computers, it is proposed to construct "general-purpose" brain-inspired computing systems. A general-purpose brain-inspired computing system refers to a brain-inspired computing hierarchy constructed based on the design philosophy of decoupling software and hardware, which can flexibly support various brain-inspired computing applications and neuromorphic chips with different architectures. Further, this paper introduces our recent work in these aspects, including the ANN (artificial neural network)/SNN (spiking neural network) development tools, the hardware agnostic compilation infrastructure, and the chip micro-architecture with high flexibility of programming and high performance; these studies show that the "general-purpose" system can remarkably improve the efficiency of application development and enhance the productivity of basic software, thereby being conductive to accelerating the advancement of various brain-inspired algorithms and applications. We believe that this is the key to the collaborative research and development, and the evolution of applications, basic software and chips in this field, and conducive to building a favorable software/hardware ecosystem of brain-inspired computing.

Keywords　brain-inspired computing, neuromorphic chip, compiler, spiking neural network

1　Introduction

Brain-inspired computing refers to computational theories, computer architectures, and application models/algorithms that draw on the information processing mode and/or structure of the biological nervous system. Its system architecture is one of the heading directions of computer architecture in the post-Moore era[1, 2].

Neuromorphic chips are the kernel of a brain-inspired computing system, which is structurally inspired by the brain organization, and efficiently sup-

Survey

This work was supported by the National Natural Science Foundation of China under Grant Nos. 62250006, 62072266, and 61836004, the National Natural Science Foundation of China Youth Fund under Grant No. 62202254, Beijing National Research Center for Information Science and Technology under Grant No. BNR2022RC01003, the Tsinghua University Initiative Scientific Research Program, and the Suzhou-Tsinghua Innovation Leadership Program.

†Co-First Author (Peng Qu wrote the methodological implications from the field of general purpose computing, Xing-Long Ji wrote the content about the framework for HNNs and the Tianjic chip, Jia-Jie Chen wrote the content of instruction set and micro-architecture, Meng Pang wrote the section of the framework for learning algorithms, Yu-Chen Li wrote the section of compilation infrastructure, and Xiao-Yi Liu wrote the current state of research in this field. The above several have made equal contributions to the paper.)

*Corresponding Author

ports spiking neural networks[3] (SNNs; the mainstream form of the brain-inspired neural network at present) in the computing paradigm, in order to break through the bottleneck of traditional computer architecture for higher computing efficiency.

The basic software of brain-inspired computing (hereinafter referred to as brain-inspired software in abbreviation)[4] is the intermediate layer that bridges the development requirements of applications and the computing capability of chips. Specifically, it not only meets the needs of various applications in this interdisciplinary field by providing flexible and friendly development interface, but also efficiently drives various hardware back-ends, through compilation, resource scheduling and mapping. Basic software is critically important for building the ecosystem of brain-inspired computing.

Countries around the world have fully recognized the importance of brain science and brain-inspired research. The United States, the European Union, Japan, Canada, Australia, and South Korea have carried out relevant research respectively. In China, "Brain and Brain-Inspired Research" (China Brain Project) is included in the "Science and Technology Innovation 2030 Major Projects" and the "New Generation of Artificial Intelligence Development Plan" (Fig.1).

The study on brain-inspired computing systems is a typical interdisciplinary research. Neuroscience, materials science, electronics/microelectronics, computer science and technology, etc. have made unique contributions: the computational needs and inspiration of neuroscience[5] are the source of progress, neuromorphic devices/circuits drive the development of neuromorphic chips with high energy-efficiency, and the integration of deep learning and SNN has greatly accelerated the real applications.

Thus, the following contents analyze the progress trends from multiple perspectives, including neuromorphic chips and learning algorithms, as well as the computational neuroscience, and then condenses the opportunities and challenges faced. In response to these issues, we have drawn methodological enlightenment from general-purpose computers and proposed to construct "general-purpose" brain-inspired computing systems. We also introduce our series of work from this aspect, including the ANN (artificial neural network)/SNN (spiking neural network) development tools, the hardware agnostic compilation infrastructure, and the chip designs.

2 Status Quo of Brain-Inspired Computing Systems

2.1 Diversity of Neuromorphic Chips

Traditional computing hardware, such as CPUs and GPGPUs, often fails to fully utilize the potential energy efficiency or computation density of neuromorphic computations. In contrast, neuromorphic chips are optimized for executing SNNs and other neuromorphic workloads. Their architectures differ vastly, ranging from the traditional von Neumann architecture to emerging neuromorphic architectures, and provide different programming interfaces and/or functional primitives.

• Neuromorphic chips of the von Neumann architecture often start with existing general purpose processors, and augment them with custom functional units for SNNs. One typical case is SpiNNaker[6]. It uses CMP (chip multi-processor) composed of ARM-cores as the baseline architecture, which is equipped with customized NoC (network on chip) to achieve

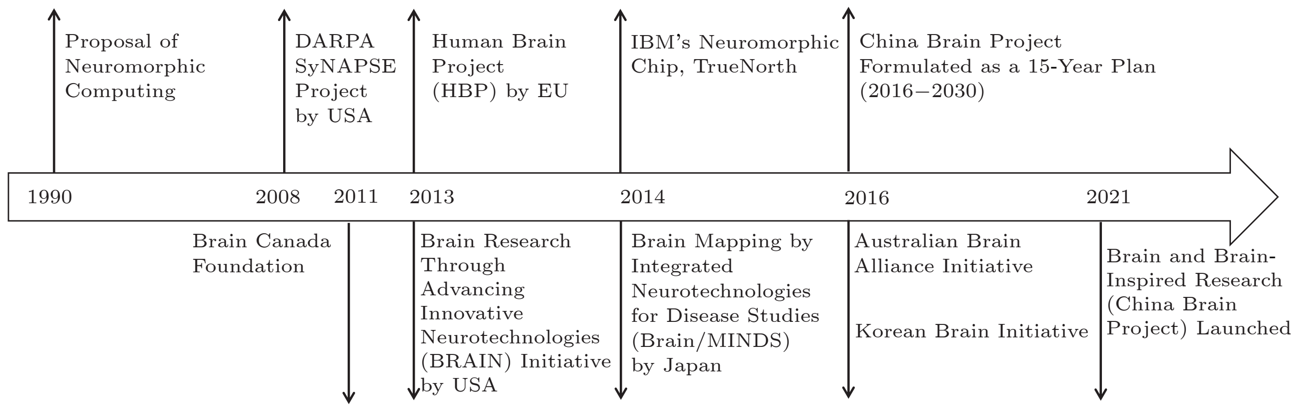

Proposal of Neuromorphic Computing
DARPA SyNAPSE Project by USA
Human Brain Project (HBP) by EU
IBM's Neuromorphic Chip, TrueNorth
China Brain Project Formulated as a 15-Year Plan (2016−2030)
1990
2008
2011
2013
2014
2016
2021
Brain Canada Foundation
Brain Research Through Advancing Innovative Neurotechnologies (BRAIN) Initiative by USA
Brain Mapping by Integrated Neurotechnologies for Disease Studies (Brain/MINDS) by Japan
Australian Brain Alliance Initiative
Korean Brain Initiative
Brain and Brain-Inspired Research (China Brain Project) Launched

Fig.1. Timeline of brain and brain-inspired research supported by countries around the world.

the efficient SNN computation with an event-driven programming model. As the main-body of computation is completed by software, it is easier to support a variety of SNN applications. The second generation SpiNNaker chip[7] is still based on ARM cores and the custom NoC. It uses several techniques to improve the energy efficiency, including near-threshold operation and dynamic voltage-and-frequency scaling. The architectural enhancements also include some accelerators for DNN (deep neural network) processing.

• Emerging neuromorphic architectures usually take a more aggressive approach by directly achieving synaptic processing and/or neuronal computation at the level of circuits (analog/digital).

For example, some[8–10] fully utilize ReRAM-based circuits to complete the vector matrix multiplications (the main body of synaptic computing) in analog, and some[11, 12] further use the dynamic properties of non-volatile memory to complete the more complex dynamics of neural networks, like the STDP rule[13]. Some digital chips, like TrueNorth[14], have dedicated neurosynaptic cores to achieve synaptic processing and neuronal computation. In addition, TrueNorth employs asynchronous-synchronous-hybrid circuits to fully exploit the asynchronous nature of SNNs. Braindrop[15] is an analog-digital-mixed neuromorphic chip, with subthreshold-analog and asynchronous-digital circuits. It is characterized by being programmed at a high level.

Moreover, some chips try to provide more flexible programming while ensuring high performance. For example, Intel's Loihi[16] is an asynchronous neuromorphic chip that supports synaptic plasticity. Each chip contains 128 neuromorphic cores for SNN processing, three x86 cores for control, and an asynchronous network-on-chip for massage transmission. It also provides the Loihi toolchain[17] with a friendly programming interface. GaBAN[18] utilizes a control-flow/data-flow hybrid architecture. It provides an instruction set tailored to neuromorphic computation for friendly programming, as well as a hardware scheduler to improve the efficiency of irregular data access in SNN computation.

• Most chips come with their own toolchain, following the software and hardware coupled methodology. The toolchain includes a compiler or a mapper to convert an SNN program into the executable to drive the chip, and/or some runtime libraries to provide high-level APIs for programming. Usually, a toolchain is only designed for the target chip, that is, the development interface and intermediate representations of the toolchain are bound to the target[17, 19]. While this methodology brings high performance to the target applications, it impairs the programming efficiency and portability, and increases the development difficulty of new toolchain.

Despite different architectures, some emerging chips show the common characteristics of supporting the hybrid of SNN and DNN.

The second generation of Tianjic[20, 21] initially uses a unified processing architecture to support SNNs, biological dynamic neural networks, multilayered perceptron, convolutional neural networks (CNNs), and recurrent neural networks (RNNs) efficiently, through multiple integration and transformation operations. SpiNNaker 2's architectural enhancements include the acceleration for DNN processing. Loihi 2[22]'s extensions[①] in this regard include two main points, the generalized event-based messaging and the enhanced learning capabilities. The former permits spikes to carry integer-valued payloads (while Loihi originally supported only binary-valued spike) to provide greater numerical precision. The latter provides support for many neuro-inspired learning algorithms, including approximations of the error back-propagation algorithm (the essential of deep learning).

2.2 Refinement of Computational Neuroscience

The neuroscience community has accumulated extensive experimental data across a wide range of scales, from sub-cellular structures and neurons to circuits and networks. This wealth of data has inspired the research interest of biological neural network models, aiming to unveil the functional mechanisms of nervous systems at multiple scales. Contrary to the prevailing trend in the domain of deep learning, which tends to simplify the biological neurons as an elementary point, experimental results show that a detailed neuron model has complex internal structures and functionalities, and it is able to learn to undertake complex tasks.

The detailed neuron model introduces more biological authenticities, such as dendritic tree and iron channels. Moreover, in contrary to the simulation of

[①]https://www.intel.com/content/www/us/en/research/neuromorphic-computing-loihi-2-technology-brief.html, Jan. 2024.

point models which only needs to solve a few simple ordinary differential equations (ODEs), the simulation of detailed models must solve a linear system of equations. In the detailed neuron model, the dendritic tree is decomposed as a series of short cylindric cables, and the circuit diagram of each piece of cable is modeled through the general cable equation and its extensions. These equations, together with the corresponding dynamic equations of non-linear ion channels, constitute the linear system of equations that needs to be solved.

As the detailed neuron models a complex system of dendritic tree, iron channels, and the soma, it provides considerable functionality and computational power. Research work[23] shows that the computational capability of detailed neurons is comparable to a 5–8 layers deep neural network. This advantage of detailed models has attracted widespread research interest, from high-performance simulation frameworks[24] to efficient learning algorithms[25].

2.3 SNN/ANN Hybrid

Besides the trend of mutual influence and gradual integration between SNNs and ANNs at the hardware level, many training algorithms for SNNs are drawing inspiration from the gradient descent algorithm employed by ANNs. These algorithms can be divided into four categories.

• *ANN-to-SNN Conversion.* Training a deep spiking neural network, which involves the learning of synaptic weights, poses significant challenges. An alternative strategy is to transform a pre-trained neural network into an SNN. The initial network is referred to as an ANN due to its real-valued activations, which represent spike rates. After the conversion, the original weights are retained, while the analog (rate) neurons of the ANN are substituted with integrate-and-fire (IF) spiking neurons. This approach is effective: throughout the simulation, the average firing rate of the spiking neurons gradually approximates the activation of the corresponding neurons in the original ANN. Extensive efforts[26–28] have been made to explore diverse methods for accomplishing this conversion efficiently, thereby facilitating the training of progressively larger SNNs.

• *Training ANNs with SNNs' Constraints.* DNNs have attained exceptional success across various tasks. This achievement implies that these models may provide valuable insights into human problem-solving processes. Consequently, substantial efforts have been directed towards increasing the biological realism of DNNs by introducing neural "spiking". In pursuit of this goal, some researchers[29] have explored the use of softened leaky integrate-and-fire (LIF) models, which involves the utilization of continuous bounded gradients achieved by smoothing activation thresholds. This approach aims to replicate the spiking behavior of SNNs within ANNs.

• *Direct Training SNNs with Surrogate Gradients.* Because SNNs use spikes for signaling, this model inherently lacks gradients at the neuron level. As a result, it does not naturally accommodate the direct application of the gradient descent method commonly employed in ANNs. To address this challenge, some researchers have introduced surrogate gradients at the neuron level to ensure the entire network differentiable[30, 31]. This approach enables SNNs to be directly trained within the frameworks of ANNs, such as PyTorch[32] and TensorFlow[33]. Furthermore, customized gradients can incorporate specific features of SNNs, such as time coding and rate coding, enhancing their adaptability and expressiveness[34].

• *Hybrid Usage of SNNs and ANNs.* Different from drawing inspiration from ANNs to SNNs (or vice versa), another promising way is to design hybrid neural networks (HNNs) by integrating SNNs and ANNs to leverage both strengths. Some researches[35–37] have proposed to employ some features of ANNs and SNNs to solve given intelligent tasks. Recent work[38] proposes a framework to support the construction and processing of HNNs at multiple scales and multiple domains by separating ANNs and SNNs and then combining them through specific interfaces. The interfaces are designable and learnable. Demonstrations have shown that this research can achieve cross-paradigm modeling for a variety of intelligent tasks.

At the same time, SNN is being gradually applied to non-AI/non-nervous-system-simulation applications, including scientific computing, signal processing, multi-class optimization problems, etc., making applications further diversified.

2.4 New Challenges

Based on the above analysis, the challenges faced by brain-inspired computing systems come from two aspects.

First, the chip micro-architectures span largely, with the diversity of hardware primitives and constraints, while the software toolchain is basically bound to the target chip, which increases the difficulty of programming/porting applications and the difficulty of basic software development.

Second, now brain-inspired applications include not only brain simulation and intelligent applications but also some common applications. Thus, the requirements for computing accuracy and flexibility, as well as learning algorithms, are getting higher, which are given as follows:

- *Computational Refinement.* Multi-precision calculation and flexible modeling are needed to meet the requirement of multi-scale and multi-dimensional nervous system simulation.
- *Broad Spectrum of Learning Algorithms.* There are multiple algorithms with different biological authenticities, as well as the SNN/DNN hybrid algorithms.

Accordingly, the applications have different computational characteristics from DNNs, including computing sparsity and the poor scale-out feature. For example, the firing rate of a biological neuron is usually no more than 100 Hz, or even only a few hertz or less[39], while a time step within the numerical simulation is usually 0.1 milliseconds. So the percent of neurons that fire in a time step does not exceed 1%, and the distribution can be regarded as random. Accordingly, because the communication between neurons is sparse and irregular, and the SNN simulation usually adopts time-step synchronization, the scale out of SNNs (Fig.2) is poor[40].

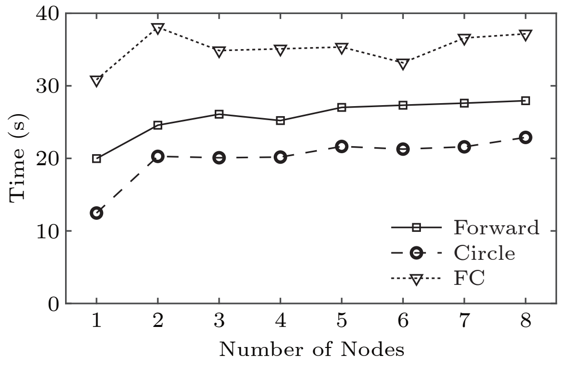


Fig.2. Strong scalability of SNNs with different topologies on a GPGPU cluster[40]. "Forward", "Circle", and "FC (full-connection)" stand for different topologies. The three SNNs run on GPU clusters with different numbers of nodes, and the network size remains the same.

There is ongoing work to address these challenges. In view of the fragmentation of software toolchain, the Lava[②] framework released by Intel at the end of 2021 abstracts SNN applications into computing models of CSP (communication sequence process). Lava is intended as a cross-platform development framework. Currently, it provides the runtime libraries that support CPU/GPU and Loihi chips.

For the efficient SNN development with multiple learning algorithms, Spikingjelly[41] was proposed. It modifies PyTorch[32] to achieve parallel computation acceleration for training and inference of deep SNNs, while providing friendly development interfaces.

Another recent work is DeepDendrite[24], which integrates a high-performance simulation algorithm of biophysically detailed compartment models and the NEURON[42] simulator (a widely-used biological neural network simulation tool). DeepDendrite demonstrates its applications in neuroscience tasks and the potential to enable the efficient training for AI tasks, that is, to bridge AI tasks and the compartment models.

3 Design Methodology of General Purpose Computers and Enlightenment

The problem that brain-inspired computing systems face is the fragmentation of software and hardware. The root cause is that current computing systems are mostly developed around their respective target chips, and the design of the interface between each layer of the system is bound to the target chip. Although this design principle will improve the efficiency of targeted applications, it increases the development difficulty of other common applications, impairs portability, and even fails to meet the need of some applications.

Accordingly, we draw experience from several design methodologies of general-purpose computers, to develop "general-purpose" basic brain-inspired software and hardware.

3.1 Computational Completeness

Alan Turing proposed an ideal computational model consisting of an infinitely long paper tape and a read/write head. This simple and intuitive model is Turing Machine[43]. Based on the Turing machine, an important concept was introduced—Turing completeness. If a computing system can simulate a Turing

[②]https://lava-nc.org/, Jan. 2024.

machine, the system is Turing-complete. Any general-purpose programming language can be used to write arbitrary algorithms as long as it satisfies Turing-completeness. Similarly, if any hardware is Turing-complete, it can simulate the execution of any general-purpose programming language.

Further, the hardware abstraction of the von Neumann architecture is called the Instruction Set Architecture (ISA), which is similar to a general-purpose Turing machine and is more flexible and efficient. For general-purpose CPUs, ISA decouples hardware implementation from software. Thus, hardware engineers can focus on the efficient micro-architecture implementation of ISA. At the same time, high-level programming languages conceal hardware details of ISA, making software development much more efficient. To some extent, this also implies the meaning of general-purpose computing, that is, software and hardware can be developed independently and efficiently while compatible with each other.

Meanwhile, the research community of brain-inspired computing has envisioned that "as these hardware-specific interfaces begin to stabilize…"[44], while the implication we get from the field of general-purpose computer is that the premise of interface stability is a reasonable hardware abstraction and the definite capability boundary. Accordingly, our preliminary work[45] proposed the theory of neuromorphic completeness, which combines Turing completeness with the universal approximation theorem and then allows any Turing-computable application to be converted to a non-exact equivalence on neuromorphic hardware with controllable accuracy loss. To be specific, general-purpose computers regard "computation" as a precise and concrete operational process, while we view "computation" as a combination of "memory" and the traditional computation (where "memory" refers to specific means such as neural network fitting, lookup table implementation of transcendental functions, and underlying logical functions implemented by lookup tables within FPGA; the commonality is that a large amount of storage resources are needed to store relevant parameters for fitting).

This work thereby theoretically makes it feasible to design a brain-inspired system for general-purpose computing applications, based on the principle of software and hardware decoupling. Fig.3 shows the correspondence between Turing completeness and neuromorphic completeness. Another benefit is that it increases the space for system design and optimization. Simply put, for a target application, the best combination of "precise computation" and "memory" should be found, which can guide the optimization of basic software and chip architecture.

3.2 Compilation Infrastructure

In addition to separating software and hardware by ISA, high-level programming languages further separate algorithms from specific hardware instructions, and compilers complete the conversion from programs in high-level languages to machine instructions. This design ensures that software applications and hardware chips can progress independently: experts of software/algorithms can develop applications without understanding hardware details, while chip architects also do not need to be proficient in specific

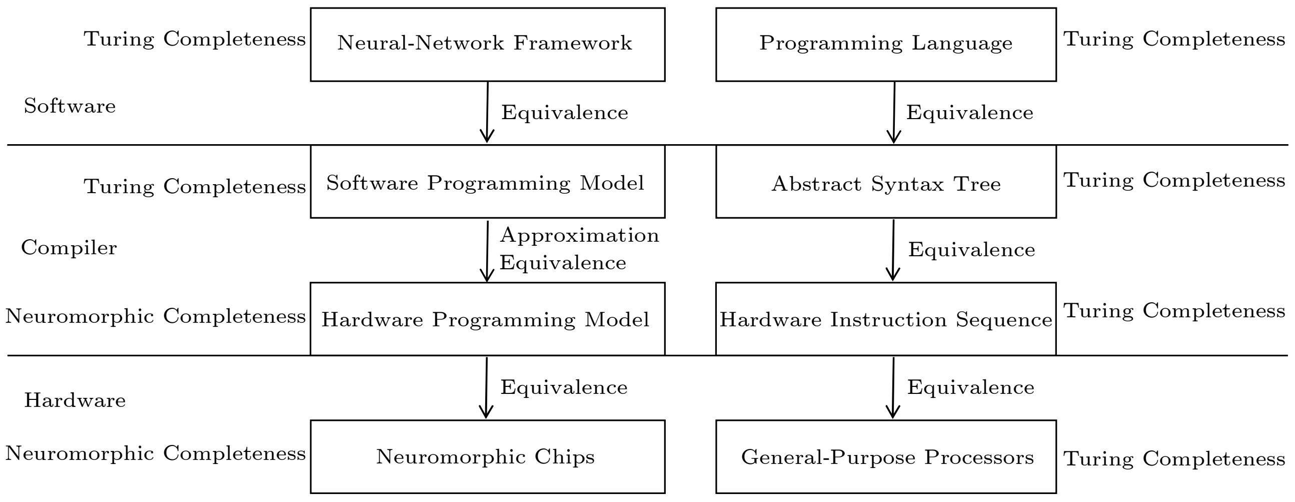


Fig.3. Turing completeness and neuromorphic completeness[45].

algorithms for various application areas[③].

It should be pointed out that the design of modern compilers themselves also reflects the hierarchical and SW (software)/HW (hardware) decoupled characteristics. Taking LLVM[46] as an example, LLVM is a modular and reusable compiler infrastructure that supports multiple front-ends (high-level languages) and back-ends (general-purpose processors). Specifically, LLVM converts the language code into the intermediate code as an intermediate representation (IR), which can be converted into machine code for any supported hardware architecture (Fig.4). The IR is independent of the source and target languages, which looks like the assembly but offers richer type annotations and user-friendly syntax. This design methodology greatly reduces the cost of developing compilers, as the infrastructure's IR, language conversion processes, and optimization techniques can be reused.

One of LLVM's latest extension efforts is MLIR[47]. MLIR is a compilation infrastructure for the problem of fragmentation, which helps improve the efficiency of building domain-specific compilers with lower cost. It provides a specification for IRs and offers a framework to do the progressive lowering of IRs. One of its features is partial lowering, that is, in the lowering path it supports mixing different levels of abstraction and concepts in the same layer. MLIR intends to solve a similar problem to that the brain-inspired computing is facing, that is, to efficiently design the compiler for DSA (domain specific architecture). Thus, we can make full use of the compilation resources of traditional computing systems to solve the fragmentation problem in the brain-inspired field.

3.3 Reduced Instruction Set Computer (RISC)

The principle of Reduced Instruction Set Computer (RISC) had an important impact on the design of modern computer processors. It further clarifies the focus of different layers of the entire computer system hierarchy: ISA is mainly oriented to hardware, focused on streamlined and efficient hardware implementation, while programming languages are oriented to software developers, focused on development flexibility and convenience. The equivalence of programming languages and instruction sets can be guaranteed by Turing-completeness theoretically, while the conversion of them is achieved by compilation. This hierarchical design decouples software and hardware requirements, avoiding the situation where the two are pinned down. RISC has become an important design philosophy for contemporary processors. Professor David A. Patterson and Professor John L. Hennessy won the 2017 Turing Award for this contribution.

Our preliminary work[48, 49] also reflected the design philosophy of RISC, that is, we proposed a set of basic primitives to realize arbitrary precision approximation of arbitrary functions (including zero-error approximation), as shown in Fig.5. These primitives are widely compatible with most current neuromorphic hardware, which can obtain accurate transformations with high complexity and approximate transformations with low complexity. For recent work [18], especially when designing chip architectures that combine flexible programming and high performance, we still draw on this philosophy, try to provide reduced hardware primitives, and leave complex functions to software (compiler).

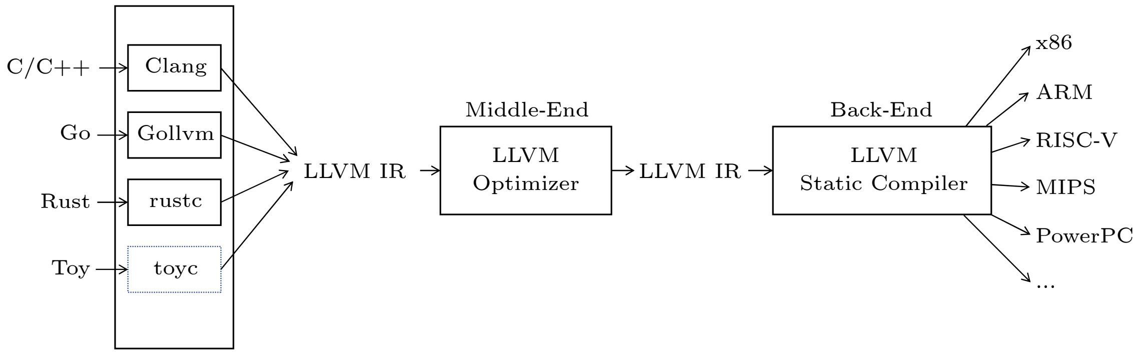


Fig.4. LLVM diagram[④].

[③]Decoupling of hardware and software is an important design principle. In practice, software and hardware co-design is also a widely-used method, and how to get a balance between the two is the key point.

[④]https://www.alibabacloud.com/blog/compilation-optimization-llvm-code-generation-technology-details-and-its-application-in-databases_598408, Jan. 2024.

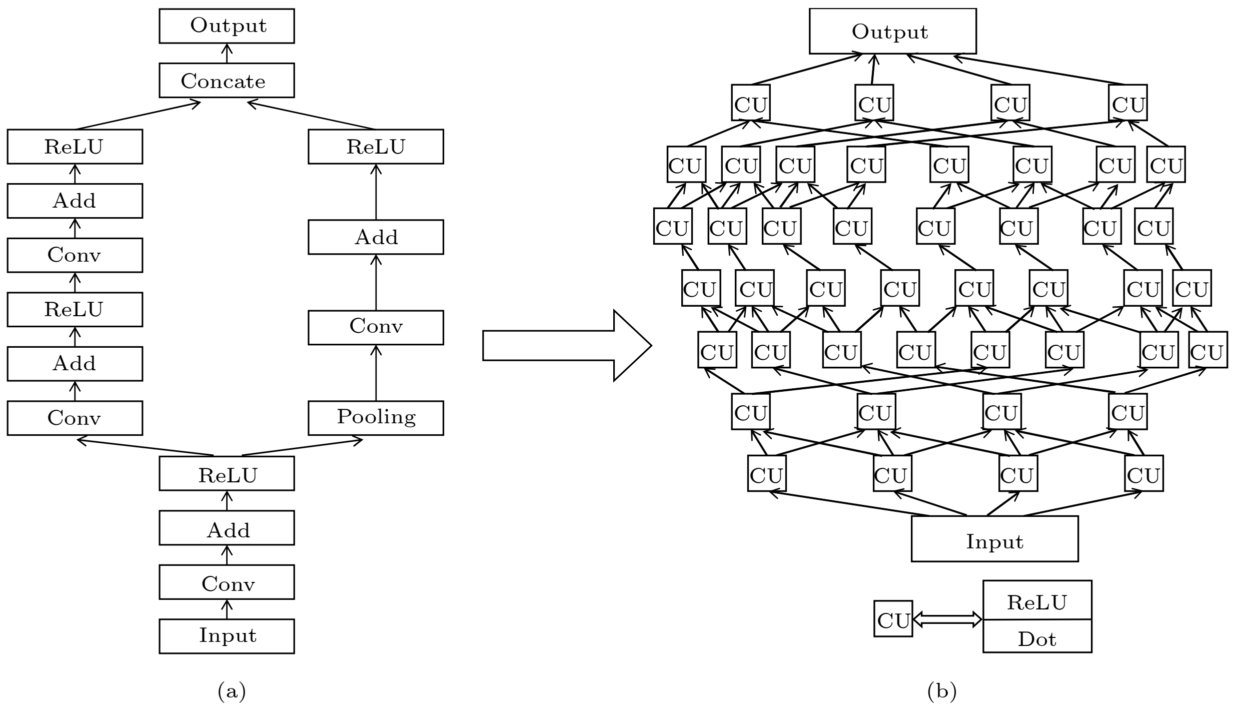


Fig.5. Different functions can be approximated by finite and basic primitives. (a) Software programming model. (b) Hardware execution model. CU: computer unit.

4 Our Work

The research on completeness theoretically makes feasible the "general-purpose" brain-inspired computing system. Further, we draw on the compilation infrastructure and the RISC design philosophy and adopt the hierarchical and decoupling technologies to develop "general-purpose" basic software and hardware. We have supported SNN development with broad-spectrum learning algorithms, achieved SW/HW decoupling and non-chip-specific compilation infrastructure, and designed the chip micro-architecture with reduced hardware primitives, flexible programming, and high performance.

Around the above principles, we have carried out the following work.

4.1 Framework for HNNs

HNNs combine SNNs from the neuroscience paradigm with ANNs from the computer science paradigm, offering flexible building blocks to advance the development of Artificial General Intelligence (AGI). To facilitate the systematic construction of HNNs, a framework[38] was proposed for their general design and computation, as illustrated in Fig.6. The framework introduces hybrid units (HUs) as the linkage interface. HUs are both customizable and trainable, enabling support for the transmission and modulation of hybrid information flows within HNNs. Building upon this foundation, diverse HNN architectures characterized by hybrid serial, parallel, and feedback structures, along with various hybrid information flows, can be constructed. These architectures enable the realization of advanced sensing, cognition, and learning tasks.

As a representative, the hybrid sensing network utilizes a parallel structure with diverse transmission paths, enabling multi-pathway sensing. This design allows for extra high-speed tracking capabilities while maintaining high accuracy[38]. Moreover, the hybrid modulation network adopts a hierarchical feed forward structure and employs hybrid modulation to achieve multi-level abstraction of task information,

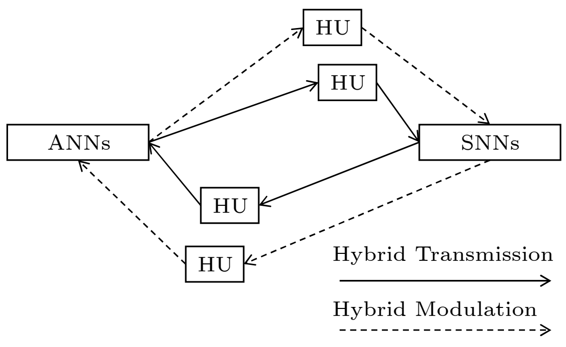


Fig.6. Architecture and information flow of the HNN framework.

which effectively mitigates catastrophic forgetting in continual learning scenarios[38]. The hybrid reasoning network implements a comprehensive neuro-symbolic system, utilizing heterogeneous transmission for interpretable and robust multimodal reasoning[38]. These innovative architectures contribute to the advancement of intelligent systems by combining the strengths of different paradigms and achieving superior performance in various intelligent tasks.

Several HNNs explore advanced functionalities like adversarial robustness and the binding problem. A typical example is the hybrid top-down attention network, which combines a feed forward SNN with a feedback ANN to efficiently allocate processing resources towards informative sensory inputs. This attention mechanism achieves remarkable resilience against adversarial noise while maintaining computational efficiency[50]. Additionally, an HNN inspired by the human cortex addresses the binding problem through an architecture employing an ANN for top-down mean-field attention and an SNN for bottom-up coincidence detection. The HNN produces synchronous coding patterns, providing a biologically plausible solution to the fundamental binding problem in neuroscience[51].

Moreover, in order to tackle intricate tasks and effectively orchestrate ANNs and SNNs, Hybrid Neural State Machine (HNSM)[52] has been devised. This neuro-inspired state machine governs the information flow among multiple networks, offering encouraging progress in control logic for such systems. One compelling illustration of the HNSM's capabilities is manifested in the hybrid neural state tracker[53], which leverages this approach to achieve high-speed tracking objectives. By combining ANN-based detection with correlation filter tracking, it demonstrates a substantial enhancement in both tracking accuracy and speed.

Merits. Aiming at the trend of SNN/ANN hybrid, the HNN framework has achieved cross-paradigm modeling, enabling the tackling of a wide array of intelligent tasks.

4.2 Framework for Various Brain-Inspired Learning Algorithms

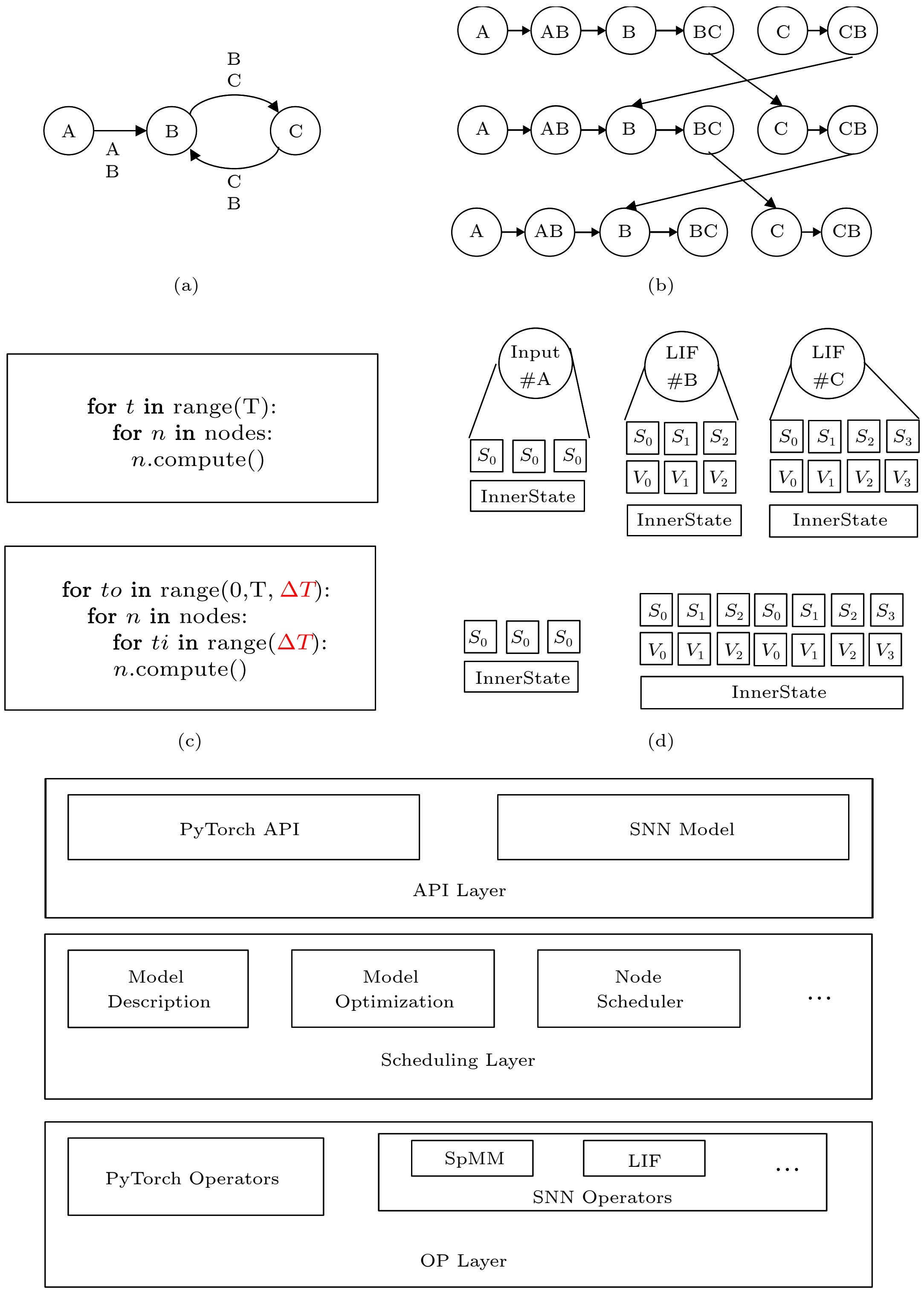
The research on the training algorithms of SNN has not yet converged[54], in contrast with the well-established gradient descent and error backpropagation utilized in ANNs. It exhibits diverse approaches, including those inspired by DNNs like D-SNN, spike-based backpropagation algorithms such as STBP[55], and biomimetic unsupervised learning through synaptic plasticity[56]. The challenge lies in balancing biological authenticity with performance. Existing development tools, like SNN simulators (NEST[57] and NEURON[42], etc.) and ANN development frameworks (PyTorch[32] and TensorFlow[33], etc.), are suboptimal for SNN training. The former is designed for biological neural network (BNN) simulation, while the development and running efficiency of the latter is relatively low due to SNNs' different computational characteristics from ANNs. Thus, an adaptable, and high-performance SNN development framework is imperative, addressing various learning algorithms while offering usability and performance enhancement. FABLE[58] was introduced as a comprehensive three-level framework designed to meet these requirements effectively.

First, it presents a computing model for various SNN training algorithms and optimizations. Based on the synchronous data flow (SDF), this model can decouple algorithm definition and optimized implementation by representing them through simple deformations. Specifically, this approach introduces the time dimension into the scheduling process while effectively representing parallelism between neural computations. Thus, it can incorporate essential SNN optimization methods that facilitate computation fusion both across time steps and within time steps.

To achieve cross-time-step optimization, we represent the SNN computational process with a two-level loop that supports the fusion of synapses and neurons at different time steps. Intra-time steps optimization, on the other hand, involves the analysis of data dependency between nodes (neurons/synapses) and then fuses nodes of the same type without dependency.

Moreover, we have integrated operators that are finely tuned for widely-used SNN neurons and optimized for sparse computing to yield higher running speed. Specifically, for neuron-related operators, several fine-grained operators within the neuron computation process over multiple time steps will be merged into one coarse-grained operator. In terms of synaptic spike delivery, we have harnessed an optimized sparse matrix multiplication algorithm for superior results. This algorithm involves the compression of the spike matrix (synaptic connection matrix) into a redundant format resembling GCOO[59], and then divides the matrix into tiles for better data locality and parallelism. Finally, multi-stage prefetching for both sparse and dense data is employed to improve the efficiency further.

Through an extensive adaption of PyTorch, we implemented FABLE (as shown in Fig.7), incorporating the aforementioned features while maintaining compatibility with PyTorch. To achieve this, we have leveraged PyTorch's fundamental components, such as the tensor data types, GPU and CPU operators, and basic parallel computing primitives, extended



```
for $t$ in range(T):
    for $n$ in nodes:
        $n$.compute()
```

```
for $to$ in range(0,T, $\Delta T$):
    for $n$ in nodes:
        for $ti$ in range($\Delta T$):
        $n$.compute()
```

Fig.7. Overview of FABLE[58]. (a) Original SNN. (b) Unrolled dataflow graph. (c) Cross-time-step optimization. (d) Intra-time-step optimization. (e) Implementation architecture.

them to accommodate custom operators, and then implemented our scheduler. Specifically, in the operator (OP) layer, we have designed a variety of coarse-grained operators for neuronal computation, including multiple variants of LIF (leaky integrate-and-fire) employed by different algorithms, as well as sparse computing operators for spike delivery. Within the scheduling layer, we have established a uniform scheduling strategy for the data flow graph, effectively separating the definitions of neurons and synapses from concrete computations. At the API layer, we have crafted friendly interfaces for custom neurons and synapses, along with the pre-defined neuron and synaptic models used in multiple integrated learning algorithms.

Merits. This work demonstrates the decoupling of SNN training algorithm and development framework. To illustrate the flexibility, five learning algorithms with different biological authenticity have been ported with less programming efforts (i.e., less coding) compared with their original implementations. Experiments reveal that FABLE outperforms the original implementations, achieving up to a 2.61x improvement in computational performance.

4.3 Compiler Infrastructure

The evolution of research in neuromorphic architecture has brought a variety of hardware function primitives. Thus, efficiently translating applications of varied characteristics into executables to drive assorted hardware back-ends poses a significant challenge. Even though existing neuromorphic chips possess their distinct software toolchains, these are designed for the respective target chips. In essence, the development interface and intermediate representations (IRs) within the toolchain frequently exhibit a close binding to the target, i.e., a tight coupling between software and hardware. Therefore, it is necessary to design a software-hardware decoupled compilation framework.

The proposed compilation framework uses MLIR[47] extensively, which supports partial lowering. Thus, the same top-level code can be transformed into a variety of hardware-specific code at the bottom level, to support different back ends, in which multiple lowering paths can share common optimizations, decreasing the effort and difficulty of developing compilers for new back ends.

Specifically, our compilation framework has three layers of IRs. The top level is SNN IRs, which contain the definition and initialization of neurons and synapses, the neuron update process, and the synaptic delivery process. They go through several lowing paths and generate either fine-grained or coarse-grained IRs at the second layer before being translated into the hardware-specific IRs at the third layer. During the lowering process (Fig.8), some optimizations and transformations can be shared.

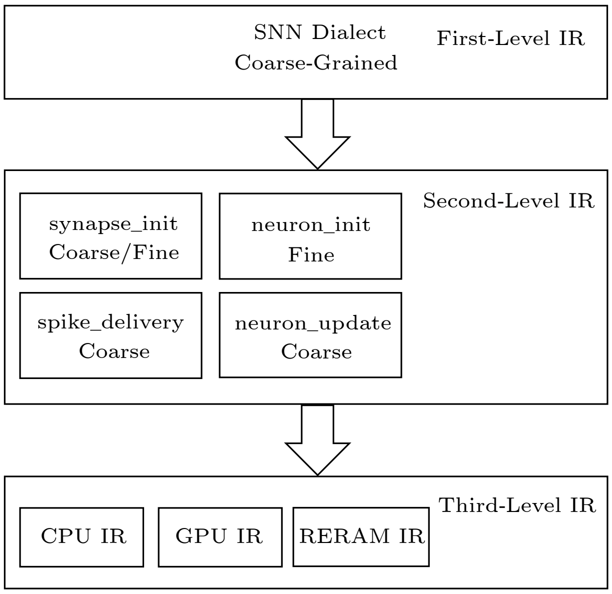


Fig.8. Multiple lowering paths.

To generate efficient executables for broad-spectrum applications and hardware, our compilation framework uses the following techniques.

- *Attribute to Describe SNN Characteristics.* Different SNN applications vary greatly, e.g., synaptic connections in computational neurology are generally sparse, whereas D-SNNs are denser. We introduce some new attributes to describe the sparsity of synapses, to give the compiler directives to produce more efficient code.
- *Optimized Merging Pass.* SNN applications (especially computational neuroscience applications) often have many neuron populations. Since their synapse connections are inherently sparse, multiple populations make the distribution of sparse synaptic connections more irregular. Therefore, we design a specific merging pass to merge neurons of different populations into a larger matrix, allowing more room for parallel optimization (like SIMD (single instruction multiple data) and multi-threading).
- *Algorithm for Spiking Delivery.* There are propagation delays in the synaptic delivery process while storing separate weights for each delay would lead to an increase in the sparsity. Therefore we proposed an

extended CSR format, which supported storing the weights of different delays in a single CSR matrix, and an efficient spike delivery algorithm based on this format was proposed further.

Merits. Owing to the HW/SW decoupled design, a hardware-agnostic SNN program is written once, and the compilation framework can translate it into high performance executables for different chips along multiple lowering paths. For example, it demonstrates 7.1x performance improvement on GPU compared with a widely-used SNN simulator. Now four kinds of back-ends have been supported, and we are extending the scope. Further, this design will reduce the burden of developing compilers for new chips, as these lowering paths share common optimizations or transformations. For example, although the ReRAM-based back-end belongs to the data flow architecture, it can still share the same high-level dialect for SNN and three main functional compilation modules.

4.4 Research on Instruction Set and Micro-Architecture

As the general abstraction of neuromorphic workloads, SNNs have some special and interesting computing characteristics, including high parallel computations, irregular and sparse memory accesses, event-driven computational mode, etc.

These characteristics provide optimization opportunities at the micro-architecture level. Through exploiting the sparse and parallel nature of SNN, we proposed the ISA-based neuro-processor, GaBAN[18].

GaBAN aims to provide both high-performance and flexible programming for SNN simulations, using a custom instruction set. The design motivation comes from the observation that although memory accesses of SNN simulation are sparse, the address of the accesses does not depend on complex computations or conditions. Thus, GaBAN introduces a hardware module, Buffets[5] (the central module in Fig.9), to fully decouple the following three aspects of SNN simulation: address generation, memory access, and computation. This allows memory loads and write-backs to happen concurrently with computations, as well as an extremely large number of outstanding memory accesses, resulting in the full overlap of computation and memory access.

The programming model of GaBAN is centered

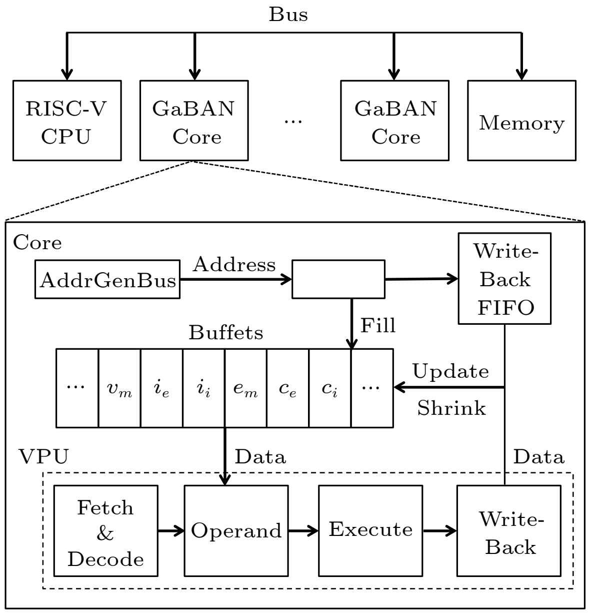


Fig.9. GaBAN's overall design (multiple core configuration)[18].

around "iteration", and different steps within SNN simulations can be seen as iterations on different entities: neuron updates iterate on the data of each neuron, spike propagations iterate on firing neurons and their corresponding rows within the synapse matrix. Thus, the kernel of a GaBAN program specifies a computation step, which contains instructions and multiple types of data (constants, variables, and memory values). The parameters for a single instance of computation step includes the iteration length, the initial value of constants and variables, and the base address of different memory values. These data are written by the control core (the GaBAN architecture contains a general-purpose core for control and initialization, and the vector cores that are responsible for computation) during startup, and may change between runs.

As mentioned above, GaBAN employs an asynchronous buffer called Buffets[60] that supports data prefetch and delayed write-back. For each iteration, a memory frame is allocated from buffets, in the form of a circular buffer. The integrated LSU (load-store unit) uses the type of each memory value and the base address to compute the valid address, and then load data into the frame. After all data within a frame have been loaded, the computation for that corresponding iteration will happen, and the LSU will continue to process the next iteration. Writebacks happen in a

[5]Buffets[60] is a new layer of the memory hierarchy between the host and accelerator, originally proposed by NVIDIA at ASPLOS 2019, which we have extended to make it programmable and applied to GaBAN.

similar manner after the computations on those data have been done. To support different steps within SNN simulations, GaBAN's memory subsystem can handle three types of memory accesses, which have been individually optimized:

- *Strided Memory Access*. It is used by neuron updates. When the stride step and the data size coincide, multiple strided accesses can be combined into a single burst of continuous memory reads for higher bandwidth.
- *Indexed Memory Access*. It usually uses a strided memory access's result as an offset to another memory location. Indexed accesses happen in synaptic updates and spike propagations.
- *Fire Tables*. The table is used for storing indexes of firing neurons.

The instruction set of GaBAN aims to provide a set of common and basic computational operations for SNN. Three flavors of arithmetic operations are provided: floating point, fixed point, and integer arithmetic, while conditional operations are implemented using a mux instruction. These instructions are executed by a VPU (vector processing unit), handling multiple data in parallel. Because no looping and branching instruction is needed, scheduling within the VPU is easy and energy efficient.

Merits. The prototype is implemented on a Xilinx VCU128 FPGA. Tests show that this ISA-based design can bring high performance while maintaining programming flexibility. It provides better performance (1.36x–1.56x) and programmability (supporting more neuron models and floating-point computation) than the SOTA work[61] also based on ISA, while managing to consume fewer hardware resources. Compared with software simulators, GaBAN is much faster than the CPU-based counterpart (1.44x–3.0x), and its performance is comparable with that of GPGPU. In addition, for D-SNN workloads, GaBAN can perform comparably with some D-SNN accelerators based on FPGA.

4.5 Tianjic-X

The research team at the Center for Brain-Inspired Computing Research (CBICR) at Tsinghua University, Beijing, has developed three generations of Tianjic series chips shown in Fig.10: Tianjic-1[62], Tianjic-2[20], and Tianjic-X[63]. The Tianjic series chips all adopt the dual-driven brain-inspired computing paradigm[64], which is inspired by both computer science and neuroscience. The goal is to build a unified and hybrid computing platform that efficiently supports the development of AGI.

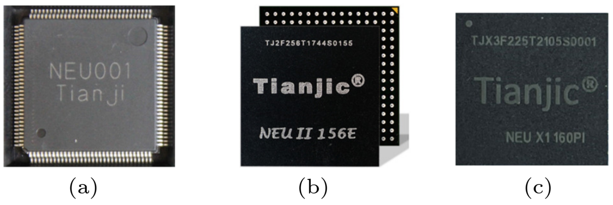


Fig.10. Tianjic series chips. (a) Tianjic-1 IEDM 2015. (b) Tianjic-2 Nature 2019. (c) Tianjic-X Science Robotics 2022.

Tianjic-1[62] is the initial exploration of hybrid architecture, implemented using 110 nm process digital circuits. The single chip has six neuromorphic functional cores, with each core having a synaptic storage size of only 256 Kb. Tianjic-1 achieved the first heterogeneous integration of computer-science-oriented ANNs and neuroscience-oriented SNNs on the functional cores by configuring the mode selection through pattern registers. This allows all modules within the functional core to operate in either ANN or SNN mode. The first-generation Tianjic chip aims to support different types of neurons at the functional level, without excessive optimization of chip area, speed, or supporting network scale. Additionally, due to its support for a simplified LIF model, the range of SNN types it supports is limited.

Tianjic-2[20] achieved the first implementation of a hybrid brain-inspired computing architecture, supporting both independent processing of ANNs and SNNs, and their hybrid modeling. It was manufactured using the UMC 28 nm CMOS process and can model up to 40k neurons and 10M synapses on a single chip. The hybrid architecture of Tianjic-2 features high resource sharing and reconfigurability, with adjustable ANN-SNN ratios to dynamically optimize resource allocation for various algorithms. Compared with Tianjic-1, Tianjic-2 optimizes resource sharing, computation within modules, and highly shared storage resources. It increases fan-in and fan-out, enabling the processing of larger-scale neural networks. The design of Soma modules was enriched to support various SNN models. The modules within the functional core can be independently configured, resulting in a unified cross-paradigm functional core architecture that efficiently supports hybrid neural networks. Compared with the IBM TrueNorth chip with the same manufacturing process, Tianjic-2 achieved a 20% increase in functional core density[65] and approximately a 10-fold increase in synaptic processing capability[14]. Currently, leading brain-inspired computing research teams worldwide have adopted similar hybrid architectures, such as SpiNNaker2[7], Loihi-2[22], and BrainScaleS-2[66]. The CBICR research team also developed an autonomous bicycle that utilizes only one Tianjic-2 chip to achieve intelligent tasks such as target tracking, image recognition, speech recognition, and decision control[20]. This demonstrates outstanding performance capabilities.

Based on the proposed hybrid architecture, the CBICR research team has developed the Tianjic-X[63] chip, which is the world's first brain-inspired computing chip designed specifically for intelligent robots. The team addressed the strict requirements of low power consumption, low latency, and multi-modal multi-tasking parallelism for robots by proposing a system solution at various levels, including execution models, the chip architecture, the software toolchain, and robot systems. At the execution model level, they proposed "Rivulet" multi-intelligent-tasking model inspired by the collaborative processing mechanisms of multiple brain regions in the human brain, which endow hardware with spatio-temporal elasticity. This allows robots to parallelly handle multiple intelligent tasks in complex and dynamic environments while adjusting resource allocation based on task performance and environmental changes. Based on the "Rivulet" model, the team developed Tianjic-X, a multi-source asynchronous-event-driven brain-inspired computing chip[67]. They proposed and implemented an architecture design with a neuromorphic-complete instruction set, which greatly enhances the programmability of the chip. Multiple intelligent tasks can be executed in parallel on the Tianjic-X chip through space-sharing, time-sharing and spatio-temporal sharing, achieving better load balancing and compact execution. The NorthPole chip[68], the newest generation of IBM's brain-inspired computing chip published recently, also adopts a similar many-core architecture with spatial-temporal pattern and elastic resource sharing.

Additionally, a hierarchical and complete software toolchain[69] has been developed to flexibly deploy different paradigms of neural networks on the chip, enabling synergistic optimization between software and hardware. Based on Tianjic-X, the team has also built a four-legged robot development platform[70]. They developed a brain-inspired general place recognition system for robotics, with the core relying on the proposed brain-inspired multi-modal hybrid neural network. This intelligent robotic system systematically addresses issues such as perception aliasing, motion blur, and large-scale dynamic changes in unmanned systems operating in open environments. Compared with existing technologies, this system exhibits higher robustness under conditions such as changes in lighting and weather, while also offering advantages such as low power consumption and low latency.

Merits. The Tianjic series of chips embodies the advantages of SNN/ANN hybrid at the hardware level. Taking its second-generation chip as an example, compared with IBM's TrueNorth chip, it supports more intelligent models in function, improves neuronal density by 20%, and increases the computing capability by 10 times. Combined with the cross-paradigm multi-tasking asynchronous parallelism, it is an efficient hardware platform for brain-inspired computation.

5 Conclusions

To a large extent, the current research of brain-inspired computing faces the major challenge of system fragmentation. Some work has started with the development software of different functions, and tried to solve such problems by supporting multiple types of applications, training algorithms or hardware.

We drew inspiration from the development history and the design philosophy of general-purpose computing, and proposed to design a "general-purpose" brain-inspired computing system. Based on the SW/HW decoupling system hierarchy, it includes the application development framework, compilation infrastructure, and flexible programmable neuromorphic chips, which can support broad-spectrum SNN training algorithms and different kinds of back-end hardware, while simplifying the development of applications and compilers. We believe that this series of work is conducive to the construction of a brain-inspired computing ecological environment, and then promote its development and industrialization.

Conflict of Interest You-Hui Zhang is an editorial board member for Journal of Computer Science and Technology and was not involved in the editorial review of this article. All authors declare that there are no other competing interests.

References

[1] Roy K, Jaiswal A, Panda P. Towards spike-based ma-

chine intelligence with neuromorphic computing. *Nature*, 2019, 575(7784): 607–617. DOI: 10.1038/s41586-019-1677-2.

[2] Waldrop M M. The chips are down for Moore's law. *Nature*, 2016, 530(7589): 144–147. DOI: 10.1038/530144a.

[3] Maass W. Networks of spiking neurons: The third generation of neural network models. *Neural Networks*, 1997, 10(9): 1659–1671. DOI: 10.1016/S0893-6080(97)00011-7.

[4] Qu P, Yang L, Zheng W M, Zhang Y H. A review of basic software for brain-inspired computing. *CCF Trans. High Performance Computing*, 2022, 4(1): 34–42. DOI: 10.1007/s42514-022-00092-1.

[5] Kass R E, Amari S I, Arai K, Brown E N, Diekman C O, Diesmann M, Doiron B, Eden U T, Fairhall A L, Fiddyment G M, Fukai T, Grün S, Harrison M T, Helias M, Nakahara H, Teramae J N, Thomas P J, Reimers M, Rodu J, Rotstein H G, Shea-Brown E, Shimazaki H, Shinomoto S, Yu B M, Kramer M A. Computational neuroscience: Mathematical and statistical perspectives. *Annual Review of Statistics and Its Application*, 2018, 5: 183–214. DOI: 10.1146/annurev-statistics-041715-033733.

[6] Plana L A, Clark D, Davidson S, Furber S, Garside J, Painkras E, Pepper J, Temple S, Bainbridge J. SpiNNaker: Design and implementation of a GALS multicore system-on-chip. *ACM Journal on Emerging Technologies in Computing Systems*, 2011, 7(4): 17. DOI: 10.1145/2043643.2043647.

[7] Höppner S, Yan Y X, Dixius A, Scholze S, Partzsch J, Stolba M, Kelber F, Vogginger B, Neumärker F, Ellguth G, Hartmann S, Schiefer S, Hocker T, Walter D, Liu G T, Garside J D, Furber S, Mayr C. The SpiNNaker 2 processing element architecture for hybrid digital neuromorphic computing. arXiv: 2103.08392, 2021.https://arxiv.org/abs/2103.08392, Jan. 2024.

[8] Zhang W B, Yao P, Gao B, Liu Q, Wu D, Zhang Q T, Li Y K, Qin Q, Li J M, Zhu Z H, Cai Y, Wu D B, Tang J S, Qian H, Wang Y, Wu H Q. Edge learning using a fully integrated neuro-inspired memristor chip. *Science*, 2023, 381(6663): 1205–1211. DOI: 10.1126/science.ade3483.

[9] Yao P, Wu H Q, Gao B, Tang J S, Zhang Q T, Zhang W Q, Yang J J, Qian H. Fully hardware-implemented memristor convolutional neural network. *Nature*, 2020, 577(7792): 641–646. DOI: 10.1038/s41586-020-1942-4.

[10] Nguyen A, Nguyen H, Venimadhavan S, Venkattraman A, Parent D, Wong H Y. Fully analog ReRAM neuromorphic circuit optimization using DTCO simulation framework. In *Proc. the 2020 International Conference on Simulation of Semiconductor Processes and Devices (SISPAD)*, Sept. 2020, pp.201–204. DOI: 10.23919/SISPAD49475.2020.9241635.

[11] She X Y, Long Y, Mukhopadhyay S. Improving robustness of ReRAM-based spiking neural network accelerator with stochastic spike-timing-dependent-plasticity. In *Proc. the 2019 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2019, pp.1–8. DOI: 10.1109/IJCNN.2019.8851825.

[12] Kim C H, Lee S, Woo S Y, Kang W M, Lim S, Bae J H, Kim J, Lee J H. Demonstration of unsupervised learning with spike-timing-dependent plasticity using a TFT-type NOR flash memory array. *IEEE Trans. Electron Devices*, 2018, 65(5): 1774–1780. DOI: 10.1109/TED.2018.2817266.

[13] Shouval H Z, Wang S S H, Wittenberg G M. Spike timing dependent plasticity: A consequence of more fundamental learning rules. *Frontiers in Computational Neuroscience*, 2010, 4: 19. DOI: 10.3389/fncom.2010.00019.

[14] Akopyan F, Sawada J, Cassidy A, Alvarez-Icaza R, Arthur J, Merolla P, Imam N, Nakamura Y, Datta P, Nam G J, Taba B, Beakes M, Brezzo B, Kuang J B, Manohar R, Risk W P, Jackson B, Modha D S. TrueNorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip. *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, 2015, 34(10): 1537–1557. DOI: 10.1109/TCAD.2015.2474396.

[15] Neckar A, Fok S, Benjamin B V, Stewart T C, Oza N N, Voelker A R, Eliasmith C, Manohar R, Boahen K. Braindrop: A mixed-signal neuromorphic architecture with a dynamical systems-based programming model. *Proceedings of the IEEE*, 2019, 107(1): 144–164. DOI: 10.1109/JPROC.2018.2881432.

[16] Davies M, Srinivasa N, Lin T H, Chinya G, Cao Y Q, Choday S H, Dimou G, Joshi P, Imam N, Jain S, Liao Y Y, Lin C K, Lines A, Liu R K, Mathaikutty D, Mccoy S, Paul A, Tse J, Venkataramanan G, Weng Y H, Wild A, Yang Y, Wang H. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 2018, 38(1): 82–99. DOI: 10.1109/MM.2018.112130359.

[17] Lin C K, Wild A, Chinya G N, Cao Y Q, Davies M, Lavery D M, Wang H. Programming spiking neural networks on Intel's Loihi. *Computer*, 2018, 51(3): 52–61. DOI: 10.1109/MC.2018.157113521.

[18] Chen J J, Yang L, Zhang Y H. GaBAN: A generic and flexibly programmable vector neuro-processor on FPGA. In *Proc. the 59th ACM/IEEE Design Automation Conference*, Jul. 2022, pp.931–936. DOI: 10.1145/3489517.3530561.

[19] Amir A, Datta P, Risk W P, Cassidy A S, Kusnitz J A, Esser S K, Andreopoulos A, Wong T M, Flickner M, Alvarez-Icaza R, McQuinn E, Shaw B, Pass N, Modha D S. Cognitive computing programming paradigm: A Corelet language for composing networks of neurosynaptic cores. In *Proc. the 2013 International Joint Conference on Neural Networks (IJCNN)*, Aug. 2013, pp.1–10. DOI: 10.1109/IJCNN.2013.6707078.

[20] Pei J, Deng L, Song S, Zhao M G, Zhang Y H, Wu S, Wang G R, Zou Z, Wu Z Z, He W, Chen F, Deng N, Wu S, Wang Y, Wu Y J, Yang Z Y, Ma C, Li G Q, Han W T, Li H L, Wu H Q, Zhao R, Xie Y, Shi L P. Towards artificial general intelligence with hybrid Tianjic chip architecture. *Nature*, 2019, 572(7767): 106–111. DOI: 10.1038/s41586-019-1424-8.

[21] Deng L, Wang G R, Li G Q, Li S C, Liang L, Zhu M H, Wu Y J, Yang Z Y, Zou Z, Pei J, Wu Z Z, Hu X, Ding Y F, He W, Xie Y, Shi L P. Tianjic: A unified and scalable chip bridging spike-based and continuous neural computation. *IEEE Journal of Solid-State Circuits*, 2020, 55(8): 2228–2246. DOI: 10.1109/JSSC.2020.2970709.

[22] Orchard G, Frady E P, Rubin D B D, Sanborn S, Shrestha S B, Sommer F T, Davies M. Efficient neuromorphic signal processing with Loihi 2. In *Proc. the 2021*

IEEE Workshop on Signal Processing Systems (SiPS), Oct. 2021, pp.254–259. DOI: 10.1109/SiPS52927.2021.00053.

[23] Beniaguev D, Segev I, London M. Single cortical neurons as deep artificial neural networks. *Neuron*, 2021, 109(17): 2727–2739.e3. DOI: 10.1016/j.neuron.2021.07.002.

[24] Zhang Y C, He G, Ma L, Liu X F, Hjorth J J J, Kozlov A, He Y T, Zhang S J, Kotaleski J H, Tian Y H, Grillner S, Du K, Huang T J. A GPU-based computational framework that bridges neuron simulation and artificial intelligence. *Nature Communications*, 2023, 14(1): 5798. DOI: 10.1038/s41467-023-41553-7.

[25] Bicknell B A, Häusser M. A synaptic learning rule for exploiting nonlinear dendritic computation. *Neuron*, 2021, 109(24): 4001–4017.e10. DOI: 10.1016/j.neuron.2021.09.044.

[26] Rueckauer B, Lungu I A, Hu Y H, Pfeiffer M, Liu S C. Conversion of continuous-valued deep networks to efficient event-driven networks for image classification. *Frontiers in Neuroscience*, 2017, 11: 682. DOI: 10.3389/fnins.2017.00682.

[27] Ding J H, Yu Z F, Tian Y H, Huang T J. Optimal ANN-SNN conversion for fast and accurate inference in deep spiking neural networks. arXiv: 2105.11654, 2021.https://arxiv.org/abs/2105.11654, Jan. 2024.

[28] Gao H R, He J X, Wang H B, Wang T X, Zhong Z Q, Yu J Y, Wang Y, Tian M, Shi C. High-accuracy deep ANN-to-SNN conversion using quantization-aware training framework and calcium-gated bipolar leaky integrate and fire neuron. *Frontiers in Neuroscience*, 2023, 17: 1141701. DOI: 10.3389/fnins.2023.1141701.

[29] Hunsberger E, Eliasmith C. Spiking deep networks with LIF neurons. arXiv: 1510.08829, 2015.https://arxiv.org/abs/1510.08829, Jan. 2024.

[30] Wu Y J, Deng L, Li G Q, Zhu J, Xie Y, Shi L P. Direct training for spiking neural networks: Faster, larger, better. In *Proc. the 33rd AAAI Conference on Artificial Intelligence*, Jan. 27–Feb. 1, 2019, pp.1311–1318. DOI: 10.1609/aaai.v33i01.33011311.

[31] Shrestha S B, Orchard G. SLAYER: Spike layer error reassignment in time. In *Proc. the 32nd International Conference on Neural Information Processing Systems*, Dec. 2018, pp.1419–1428.

[32] Paszke A, Gross S, Massa F, Lerer A, Bradbury J, Chanan G, Killeen T, Lin Z M, Gimelshein N, Antiga L, Desmaison A, Köpf A, Yang E, DeVito Z, Raison M, Tejani A, Chilamkurthy S, Steiner B, Fang L, Bai J J, Chintala S. PyTorch: An imperative style, high-performance deep learning library. In *Proc. the 33rd International Conference on Neural Information Processing Systems*, Dec. 2019, Article No. 721.

[33] Abadi M, Barham P, Chen J M, Chen Z F, Davis A, Dean J, Devin M, Ghemawat S, Irving G, Isard M, Kudlur M, Levenberg J, Monga R, Moore S, Murray D G, Steiner B, Tucker P, Vasudevan V, Warden P, Wicke M, Yu Y, Zheng X Q. TensorFlow: A system for large-scale machine learning. In *Proc. the 12th USENIX Symposium on Operating Systems Design and Implementation*, Nov. 2016, pp.265–283.

[34] Kim J, Kim K, Kim J J. Unifying activation- and timing-based learning rules for spiking neural networks. In *Proc. the 34th International Conference on Neural Information Processing Systems*, Dec. 2020, Article No. 1639.

[35] Lobov S, Mironov V, Kastalskiy I, Kazantsev V. A spiking neural network in sEMG feature extraction. *Sensors*, 2015, 15(11): 27894–27904. DOI: 10.3390/s151127894.

[36] Chancán M, Hernandez-Nunez L, Narendra A, Barron A B, Milford M. A hybrid compact neural architecture for visual place recognition. *IEEE Robotics and Automation Letters*, 2020, 5(2): 993–1000. DOI: 10.1109/LRA.2020.2967324.

[37] Lee C, Kosta A K, Zhu A Z, Chaney K, Daniilidis K, Roy K. Spike-FlowNet: Event-based optical flow estimation with energy-efficient hybrid neural networks. arXiv: 2003.06696, 2020. https://arxiv.org/abs/2003.06696, Jan. 2024.

[38] Zhao R, Yang Z Y, Zheng H, Wu Y J, Liu F Q, Wu Z Z, Li L K, Chen F, Song S, Zhu J, Zhang W L, Huang H Y, Xu M K, Sheng K F, Yin Q B, Pei J, Li G Q, Zhang Y H, Zhao M G, Shi L P. A framework for the general design and computation of hybrid neural networks. *Nature Communications*, 2022, 13(1): 3427. DOI: 10.1038/s41467-022-30964-7.

[39] Roxin A, Brunel N, Hansel D, Mongillo G, Vreeswijk C V. On the distribution of firing rates in networks of cortical neurons. *Journal of Neuroscience*, 2011, 31(5): 16217–16226. DOI: 10.1523/JNEUROSCI.1677-11.2011.

[40] Qu P, Lin H, Pang M, Liu X F, Zheng W M, Zhang Y H. ENLARGE: An efficient SNN simulation framework on GPU clusters. *IEEE Trans. Parallel and Distributed Systems*, 2023, 34(9): 2529–2540. DOI: 10.1109/TPDS.2023.3291825.

[41] Fang W, Chen Y Q, Ding J H, Yu Z F, Masquelier T, Chen D, Huang L W, Zhou H H, Li G Q, Tian Y H. SpikingJelly: An open-source machine learning infrastructure platform for spike-based intelligence. *Science Advances*, 2023, 9(40): eadi1480. DOI: 10.1126/sciadv.adi1480.

[42] Hines M L, Carnevale N T. The NEURON simulation environment. *Neural Computation*, 1997, 9(6): 1179–1209. DOI: 10.1162/neco.1997.9.6.1179.

[43] Turing A M. On computable numbers, with an application to the entscheidungsproblem. *Journal of Mathematics*, 1936, 58: 345–363. DOI: 10.112/plms/s2-42.1.230.

[44] Aimone J B, Severa W, Vineyard C M. Composing neural algorithms with Fugu. In *Proc. the International Conference on Neuromorphic Systems*, Jul. 2019, Article No. 3. DOI: 10.1145/3354265.3354268.

[45] Zhang Y H, Qu P, Ji Y, Zhang W H, Gao G R, Wang G R, Song S, Li G Q, Chen W G, Zheng W M, Chen F, Pei J, Zhao R, Zhao M G, Shi L P. A system hierarchy for brain-inspired computing. *Nature*, 2020, 586(7829): 378–384. DOI: 10.1038/s41586-020-2782-y.

[46] Lattner C. LLVM: An infrastructure for multi-stage optimization [Master's Thesis]. University of Illinois at Urbana-Champaign, Champaign-Urbana, 2002.

[47] Lattner C, Amini M, Bondhugula U, Cohen A, Davis A, Pienaar J, Riddle R, Shpeisman T, Vasilache N, Zinenko O. MLIR: Scaling compiler infrastructure for domain specific computation. In *Proc. the 2021 IEEE/ACM International Symposium on Code Generation and Optimization*

(CGO), Feb. 2021, pp.2–14. DOI: 10.1109/CGO51591.2021.9370308.

[48] Ji Y, Zhang Y Y, Xie X F, Li S C, Wang P Q, Hu X, Zhang Y H, Xie Y. FPSA: A full system stack solution for reconfigurable ReRAM-based NN accelerator architecture. In *Proc. the 24th International Conference on Architectural Support for Programming Languages and Operating Systems*, Apr. 2019, pp.733–747. DOI: 10.1145/3297858.3304048.

[49] Ji Y, Liu Z X, Zhang Y H. A reduced architecture for ReRAM-based neural network accelerator and its software stack. *IEEE Trans. Computers*, 2021, 70(3): 316–331. DOI: 10.1109/TC.2020.2988248.

[50] Liu F Q, Zhao R. Enhancing spiking neural networks with hybrid top-down attention. *Frontiers in Neuroscience*, 2022, 16: 949142. DOI: 10.3389/fnins.2022.949142.

[51] Zheng H, Lin H, Zhao R, Shi L P. Dance of SNN and ANN: Solving binding problem by combining spike timing and reconstructive attention. In *Proc. the 36th International Conference on Neural Information Processing Systems*, Nov. 28–Dec. 9, 2022, pp.31430–31443.

[52] Tian L, Wu Z Z, Wu S, Shi L P. Hybrid neural state machine for neural network. *Science China Information Sciences*, 2021, 64(3): 132202. DOI: 10.1007/s11432-019-2988-1.

[53] Zou Z, Wu Y J, Zhao R. HNST: Hybrid neural state tracker for high speed tracking. In *Proc. the 7th International Conference on Control, Automation and Robotics (ICCAR)*, Apr. 2021, pp.231–235. DOI: 10.1109/ICCAR52225.2021.9463460.

[54] Eshraghian J K, Ward M, Neftci E, Wang X X, Lenz G, Dwivedi G, Bennamoun M, Jeong D S, Lu W D. Training spiking neural networks using lessons from deep learning. arXiv: 2109.12894, 2021. https://arxiv.org/abs/2109.12894, Jan. 2024.

[55] Wu Y J, Deng L, Li G Q, Zhu J, Shi L P. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers in Neuroscience*, 2018, 12: 331. DOI: 10.3389/fnins.2018.00331.

[56] Grossberg S. Competitive learning: From interactive activation to adaptive resonance. *Cognitive Science*, 1987, 11(1): 23–63. DOI: 10.1016/S0364-0213(87)80025-3.

[57] Gewaltig M O, Diesmann M. NEST (neural simulation tool). *Scholarpedia*, 2007, 2(4): 1430. DOI: 10.4249/scholarpedia.1430.

[58] Pang M, Li Y C, Li Z L, Zhang Y H. FABLE: A development and computing framework for brain-inspired learning algorithms. In *Proc. the 2023 International Joint Conference on Neural Networks (IJCNN)*, Jun. 2023. DOI: 10.1109/IJCNN54540.2023.10192026.

[59] Shi H, Wang Q, Chu X W. Efficient sparse-dense matrix-matrix multiplication on GPUs using the customized sparse storage format. In *Proc. the 26th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, Dec. 2020, pp.19–26. DOI: 10.1109/ICPADS51040.2020.00013.

[60] Pellauer M, Shao Y S, Clemons J, Crago N, Hegde K, Venkatesan R, Keckler S W, Fletcher C W, Emer J. Buffets: An efficient and composable storage idiom for explicit decoupled data orchestration. In *Proc. the 24th International Conference on Architectural Support for Programming Languages and Operating Systems*, Apr. 2019, pp.137–151. DOI: 10.1145/3297858.3304025.

[61] Oltra-Oltra J A, Madrenas J, Zapata M, Vallejo B, Mata-Hernandez D, Sato S. Hardware-software co-design for efficient and scalable real-time emulation of SNNs on the edge. In *Proc. the 2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2021. DOI: 10.1109/ISCAS51556.2021.9401615.

[62] Shi L P, Pei J, Deng N, Wang D, Deng L, Wang Y, Zhang Y H, Chen F, Zhao M G, Song S, Zeng F, Li G Q, Li H L, Ma C. Development of a neuromorphic computing system. In *Proc. the 2015 IEEE International Electron Devices Meeting (IEDM)*, Dec. 2015, pp.4.3.1–4.3.4. DOI: 10.1109/IEDM.2015.7409624.

[63] Ma S C, Pei J, Zhang W H, Wang G R, Feng D H, Yu F W, Song C H, Qu H Y, Ma C, Lu M S, Liu F Q, Zhou W H, Wu Y J, Lin Y H, Li H Y, Wang T Y, Song J R, Liu X, Li G Q, Zhao R, Shi L P. Neuromorphic computing chip with spatiotemporal elasticity for multi-intelligent-tasking robots. *Science Robotics*, 2022, 7(67): eabk2948. DOI: 10.1126/scirobotics.abk2948.

[64] Zhang B, Shi L P, Song S. Creating more intelligent robots through brain-inspired computing. *Science Robotics*, 2016, 354(6318): 1445.

[65] Merolla P A, Arthur J V, Alvarez-Icaza R, Cassidy A S, Sawada J, Akopyan F, Jackson B L, Imam N, Guo C, Nakamura Y, Brezzo B, Vo I, Esser S K, Appuswamy R, Taba B, Amir A, Flickner M D, Risk W P, Manohar R, Modha D S. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 2014, 345(6197): 668–673. DOI: 10.1126/science.1254642.

[66] Pehle C, Billaudelle S, Cramer B, Kaiser J, Schreiber K, Stradmann Y, Weis J, Leibfried A, Müller E, Schemmel J. The BrainScaleS-2 accelerated neuromorphic system with hybrid plasticity. *Frontiers in Neuroscience*, 2022, 16: 795876. DOI: 10.3389/fnins.2022.795876.

[67] Li H Y, Ma S C, Wang T Y, Zhang W H, Wang G R, Song C H, Qu H Y, Lin J F, Ma C, Pei J, Zhao R. HASP: Hierarchical asynchronous parallelism for multi-NN tasks. *IEEE Trans. Computers*, 2024, 73(2): 366–379. DOI: 10.1109/TC.2023.3329937.

[68] Modha D S, Akopyan F, Andreopoulos A, Appuswamy R, Arthur J V, Cassidy A S, Datta P, DeBole M V, Esser S K, Otero C O, Sawada J, Taba B, Amir A, Bablani D, Carlson P J, Flickner M D, Gandhasri R, Garreau G J, Ito M, Klamo J L, Kusnitz J A, Mcclatchey N J, Mckinstry J L, Nakamura Y, Nayak T K, Risk W P, Schleupen K, Shaw B, Sivagnaname J, Smith D F, Terrizzano I, Ueda T. Neural inference at the frontier of energy, space, and time. *Science*, 2023, 382(6668): 329–335. DOI: 10.1126/science.adh1174.

[69] Lin J F, Qu H Y, Ma S C, Ji X L, Li H Y, Li X C, Song C H, Zhang W H. SongC: A compiler for hybrid near-memory and in-memory many-core architecture. *IEEE Trans. Computers*. DOI: 10.1109/TC.2023.3311948.

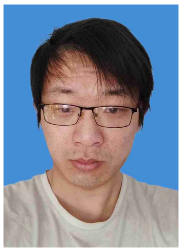
[70] Yu F W, Wu Y J, Ma S C, Xu M K, Li H Y, Qu H Y, Song C H, Wang T Y, Zhao R, Shi L P. Brain-inspired

multimodal hybrid neural network for robot place recognition. *Science Robotics*, 2023, 8(78): eabm6996. DOI: 10.1126/scirobotics.abm6996.

Peng Qu received his B.E. degree in computer science and technology from Tsinghua University, Beijing, in 2013, and his Ph.D. degree in the same field in 2018. Currently, he is an assistant professor in the Department of Computer Science and Technology at Tsinghua University, Beijing. His primary research interests include computer architecture and neuromorphic computing.

Xing-Long Ji received his B.E. degree from Huazhong University of Science and Technology, Wuhan, in 2011, and his Ph.D. degree in microelectronics from University of Chinese Academy of Sciences, Beijing, in 2016. He is currently an associate professor in the Department of Precision Instrument at Tsinghua University, Beijing. His current research interests include brain-inspired computing, edge computing, and non-volatile memory technologies.

Jia-Jie Chen received his B.E. degree in computer science from Tsinghua University, Beijing, in 2021. He is currently a Ph.D. student in the Department of Computer Science and Technology at Tsinghua University, Beijing. His research interests include computer architecture and neuromorphic computing.



Meng Pang received his B.E. degree in computer science from Jilin University, Changchun, in 2020. He is currently a Ph.D. student in the Department of Computer Science and Technology at Tsinghua University, Beijing. His research interests include compilation optimization and automatic parallelization.

Yu-Chen Li is currently an undergraduate student in the Department of Computer Science and Technology at Tsinghua University, Beijing. His research interests include neuromorphic computing and compilation technology.

Xiao-Yi Liu received his B.E. degree in computer science from Tsinghua University, Beijing, in 2023. He is currently a Master student in the Department of Computer Science and Technology at Tsinghua University, Beijing. His research interests include computer architecture and neuromorphic computing.

You-Hui Zhang received his B.E. degree in computer science from Tsinghua University, Beijing, in 1998, and his Ph.D. degree in computer science from the same university in 2002. Currently, he is a professor in the Department of Computer Science and Technology at Tsinghua University, Beijing. His research interests include computer architecture, processor microarchitecture, and neuromorphic computing.