TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 07/15 pp664–673
DOI: 10.26599/TST.2021.9010010
Volume 26, Number 5, October 2021

Towards "General Purpose" Brain-Inspired Computing System

Youhui Zhang*, Peng Qu, and Weimin Zheng

Abstract: Brain-inspired computing refers to computational models, methods, and systems, that are mainly inspired by the processing mode or structure of brain. A recent study proposed the concept of "neuromorphic completeness" and the corresponding system hierarchy, which is helpful to determine the capability boundary of brain-inspired computing system and to judge whether hardware and software of brain-inspired computing are compatible with each other. As a position paper, this article analyzes the existing brain-inspired chips' design characteristics and the current so-called "general purpose" application development frameworks for brain-inspired computing, as well as introduces the background and the potential of this proposal. Further, some key features of this concept are presented through the comparison with the Turing completeness and approximate computation, and the analyses of the relationship with "general-purpose" brain-inspired computing systems (it means that computing systems can support all computable applications). In the end, a promising technical approach to realize such computing systems is introduced, as well as the on-going research and the work foundation. We believe that this work is conducive to the design of extensible neuromorphic complete hardware-primitives and the corresponding chips. On this basis, it is expected to gradually realize "general purpose" brain-inspired computing system, in order to take into account the functionality completeness and application efficiency.

Key words: brain-inspired computing; neuromorphic computing; computational completeness; hardware/software decoupling; system hierarchy

1 Introduction

Brain-inspired computing is a general term for computing theory, chip architecture, system design, and algorithms, inspired by the processing mode or structure of biological nervous systems or brains. It is considered as one of the most promising technological paths towards the next generation of artificial intelligence[1–5], and brain-inspired computing architecture is also a major development direction of computer architecture in the post-Moore's Law era[6–8]. Thus, Ref. [9] has stated that brain-inspired computing would be the next wave of intelligent computing.

Brain-inspired computing is also called neuromorphic computing. The term of neuromorphic computing first appeared in Ref. [10] written in the 1990s by Professor Carver Mead of computer science at California Institute of Technology. At that time neuromorphic computing system referred to an adaptive and large-scale parallel computing system using very large scale integrated circuits that simulates biological nervous system with electronic devices. It usually used elementary physical phenomena as computational primitives, and represented information by the relative values of analog signals.

With the passage of time, the styles and technologies of brain-inspired computing chips (also known as neuromorphic chips. In this paper these two terms are considered to be synonyms) have changed a lot[4], but their technical routes can be traced back to Prof. Mead's work[10] to some extent, and the following three main

• Youhui Zhang, Peng Qu, and Weimin Zheng are with the Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China. E-mail: zyh02@tsinghua.edu.cn; qup13@mails.tsinghua.edu.cn; zwm-dcs@tsinghua.edu.cn.

∗ To whom correspondence should be addressed.

Manuscript received: 2021-01-18; accepted: 2021-02-04

characteristics are still roughly maintained:

- Inspired by the structure and/or processing mode of the biological brain (or nervous system);
- Simulate the information processing function of biological neurons and neural synapses through micro-/nano-devices;
- Realize a new intelligent computer system with low energy consumption and high efficiency.

Brain-inspired computing systems employ neuromorphic chips as the core to support various applications[11, 12], and basically use Spiking Neural Networks (SNNs)[13] as the main computing paradigm at present. For such chips, whether they are digital or digital-analog-hybrid or based on some novel nonvolatile memory devices[14, 15], a common feature lies in that almost all of them are based on the end-to-end co-design principle; so that each type of chip has its specific software and hardware interfaces, toolchains, and target applications. These characteristics bring two issues:

- It binds applications with the specific system, and then leads to lack of portability and low productivity. That is, it couples software with hardware.
- The application range of brain-inspired computing is developing rapidly, while hardware features are summarized from existing applications. Thus it is difficult to judge whether today's chips can cope with emerging applications, that is, the completeness of computing system is uncertain.

Meanwhile, there have been some efforts to achieve "general purpose" development frameworks for brain-inspired computing, such as STICK[16], FUGU[17], etc., with the goal of providing a high-level programming abstraction and realizing a unified development tool independent of specific chips, i.e., they intend to make hardware specifications/constraints "transparent" to application development. But the problem of not being able to determine whether the hardware is functional enough, i.e., complete, does persist. In other words, a brain-inspired computing system can be said to be "general-purpose" if it can support all brain-inspired computing applications; currently, how to judge whether a brain-inspired computing system is "general-purpose" is an open question.

This article analyzes the existing brain-inspired chips' design characteristics and disadvantages, as well as the "general purpose" application development framework for brain-inspired computing, and introduces the recent work of the proposed "neuromorphic completeness" (also known as "brain-inspired computational completeness") and some of the key features. In the end, a promising technical approach to realize computing systems of neuromorphic completeness is introduced, as well as the work we are doing and the research foundation.

2 Research Status

2.1 Chips

The next generation of high-performance and low power computer systems might be inspired by the brain. The focus of neuromorphic computing is typically on spiking neural networks, as such systems are more similar to biological neural networks than artificial neural networks, which have been commonly used in modern deep-learning applications. Moreover, most brain-inspired systems share common design principles, such as co-location of the memory and processing units. Some representative studies are given as follows:

TrueNorth[18, 19] is a digital neuromorphic chip that includes 4096 neurosynaptic cores connected via a 2D-mesh NoC, equipped with a toolchain that includes the programming paradigm, Corelet[20]. Through Corelet, a complex algorithm is decomposed into simple ones, which will be repeated till tasks can be completed by one or more cores. In addition, TrueNorth provides an optimized strategy to map logical NNs to physical cores[21]. The Corelet description just matches the organization of the hardware substrate (although the description is agnostic to the physical location of each core in the actual hardware[21]); thus it is bound to the hardware platform.

Neurogrid[22] is an analog/digital hybrid system with a customized network[23]. It uses the Neural Engineering Framework (NEF)[24] to configure neuromorphic chips to implement target functions. This allows the designer to work on a higher level of abstraction and yet still produce a detailed model using spiking neurons. Namely, Neurogrid supports the NEF on the hardware and system levels. The latest successor of Neurogrid, Braindrop[25], also adopted this design philosophy.

Intel's Loihi[26] chip consists of a many-core mesh of 128 neuromorphic cores, as well as three x86 cores for task scheduling and IO. It supports SNN-based inference and learning functions. The neuromorphic core contains a hardware pipeline customized for synaptic computing (including support for synaptic plasticity, which is the basis of SNN learning mechanism),

neural computing, and others. Loihi also provides a development toolchain[27], including compilers and runtime, and exposes hardware-supported brain-inspired computing primitives to developers through interfaces such as Python. Due to the limitation of hardware resources, Loihi currently only supports fixed-point computation and fixed-point weight data.

The FACETS[28] project and its successor BrainScaleS have produced wafer-scale IC systems. It developes a graph model to map a complex biological network to the hardware network. This strategy requires that both have the same underlying structure; thus it is also hardware-specific.

SpiNNaker[29] has a toolchain that is based on the CMPs of ARM cores[30]. Thus, its neural computing is completed by software with some hardware customization for efficiency. The drawback is that the performance will be lower than dedicated hardware.

So, most existing chips are based on the end-to-end design principle. We can summarize this phenomenon with a passage from Ref. [25] of Braindrop: "$\cdots$ achieving this goal required co-designing all layers of the software-hardware stack, keeping the theoretical framework in mind even at the lowest levels of the hardware design". This design principle leads to higher execution efficiency for adapted applications, while the downside is that hardware constraints are often exposed to developers, which usually requires developers to understand substantial knowledge of neural computing or neuromorphic hardware, in other words, impairing portability and productivity. What's worse, because hardware functions have been summarized based on the requirements of existing applications, it is difficult to determine where the functional boundary of hardware is and whether it is complete or not facing the rapid development of this field. Following this trend, various brain-inspired computing chips and systems would become research and development islands.

2.2 Software

On the other hand, in the field of basic software for brain-inspired computing, there are also researchers who have recognized this problem and tried to study a "general purpose" development framework.

For example, Fugu[17] sought to achieve a programming platform to enable the development of neuromorphic applications without substantial knowledge of the substrate. Rather than necessitating a developer attain intricate knowledge of how to program and exploit spiking neural dynamics to utilize the potential benefits of neuromorphic computing, Fugu is designed to provide a higher level abstraction as a hardware-independent mechanism for linking a variety of scalable spiking neural algorithms from a variety of sources.

Another typical work is STICK[16]. Its goal is to develop a new framework for brain-inspired general purpose computation architectures. This works tries to show that the use of neuron-like units with precise timing representation, synaptic diversity, and others could set a complete (Turing complete) computation framework.

These studies are still carried out at the software level, without support for any specific brain-inspired hardware. To a large extent, one sentence from Ref. [17] of Fugu can illustrate the dilemma faced by such software frameworks: "We envision that as these hardware-specific interfaces begin to stabilize $\cdots$".

But, the hardware interfaces are difficult to stabilize until they have been determined to be complete. That is, in the face of the rapid development of AI computing field, it is not enough to summarize the hardware interfaces (functions) only according to the specific application requirements or hardware specifications. Therefore, it is necessary to study the completeness of brain-inspired computing to find out the functional boundary.

3 Inspiration from General-Purpose Computers

Turing completeness and the von Neumann architecture are the key factors in the rapid development of general-purpose computer technologies. Almost all high-level programming languages are Turing complete, and the general-purpose processors based on the von Neumann architecture can realize Turing completeness through some Turing complete instruction set, which means that any Turing computable function written in a programming language can be converted into an equivalent instruction sequence on any Turing complete processor (i.e., program compilation). This is why general-purpose computers are called "general-purpose". Further, the computer hierarchy composed of the software layer, the compiler layer, and the hardware layer can ensure that the application software and the hardware design are compatible with each other while moving forward independently (i.e., software and hardware decoupling), laying a systematic foundation for the

prosperity and development of the entire field.

Inspired by this principle, the concept of "neuromorphic completeness"[31] has been proposed: For any given error gap $\epsilon \geqslant 0$ and any Turing-computable function $f(x)$, a computational system is called neuromorphic complete if it can achieve a function $F(x)$, such that $\|F(x)-f(x)\| \leqslant \epsilon$ for any valid input x.

Compared with Turing completeness, this definition does not require the system to achieve a function through a series of precise computational steps (that is, algorithms). Further, a corresponding computer hierarchy and some hardware primitives have been proposed to ensure the completeness of brain-inspired computing to make full use of the advantages brought by this new concept. The hierarchy has three levels: a Turing complete software model, a neuromorphic complete hardware architecture, and a compilation layer between the above two. A constructive transformation algorithm is designed to convert any Turing computable function into an equivalence on any neuromorphic complete hardware, which brings the following advantages:

First, general-purpose computing applications can be supported. The proposed application-oriented software model is Turing complete and provides a basis for programming to support various applications (not limited to neural networks).

Second, compilation feasibility. Through the proposed hardware primitives and constructive conversion algorithm, the equivalent conversion of "neuromorphic completeness" between the "Turing complete" software and the "neuromorphic complete" hardware primitive sequence is ensured, which realizes the decoupling of software and hardware.

Third, a new dimension of system design and optimization, approximate granularity, is introduced. A reasonable inference of this definition is that a Turing computable function can be implemented by a process of traditional accurate computation (algorithm), by approximations (not limit the specific technical means), or by a hybrid of the two, which enlarges the system design space and is conducive to improving efficiency.

4 Some Highlights of Neuromorphic Completeness

4.1 A neuromorphic complete system is conducive to the realization of a "general-purpose" brain-inspired computing system

The proposed concept of completeness could enable the decoupling of hardware and software while ensuring compatibility. As declaimed by Ref. [32], it is a useful step to unite the work carried out by the many industrial and academic research groups in the field of neuromorphic computing, as it helps researchers to focus on specific aspects of research problems, rather than trying to find entire end-to-end solutions. In other words, any neuromorphic complete system can support any brain-inspired computing application, i.e., it is "general-purpose" (This concept is orthogonal to Artificial General Intelligence (AGI)).

Further, for a computing paradigm (including chips, basic software, applications, etc.) with great and long-term development potential, it is necessary to build an application ecosystem: More and more research and development personnel entering the field, rather than experts with general knowledge (interdisciplinary), is a prerequisite for the establishment of such an ecological environment, otherwise there can only be limited scope of applications, impairing the potential for progress.

4.2 Neuromorphic completeness and Turing completeness

It is necessary to note that the concept of completeness itself defines the functional boundary of a computing system and does not involve the specific implementation. Taking Turing completeness as an example: The universal Turing machine is Turing complete, but there are many various forms of Turing complete systems (including recurrent neural network, lambda calculus, some forms of cellular automata, recursively enumerable languages, etc.), and the common point is that all can simulate a universal Turing machine and have the equivalent computational capability. Thus, a neuromorphic complete system may be not on the neural basis, from an implementation point of view. Moreover, although this definition relaxes the requirement on the computing process (compared with Turing completeness), they are not opposites; any Turing complete system is neuromorphic complete[31]. This is also in line with the actual situation of brain-inspired research: In fact, some well-known computing systems (such as SpiNNaker from the University of Manchester) directly use Turing-complete general-purpose processors (ARM cores with custom extension) for main operations.

4.3 Computational process of a neuromorphic complete system could be a combination of accurate Turing computation and approximation

Traditional computer algorithms are derived from the

Turing machine, which is an accurate description of the specific computing process. In this sense, Turing computation has no errors.

On the basis of Turing completeness, the new completeness is compatible with the ability to approximate computing results. It should be noted that here the concept of approximation refers to approximation capability, not limited to numerical approximation.

Specifically, for an objective function, its realization form may have three ways (but not limited to, because the completeness itself does not limit the concrete method):

First, the entire function is directly approximated by neural network or lookup table, etc.

Second, through reorganizing and approximating some specific steps in the computing process of the objective function (i.e., algorithm), the corresponding intermediate results are approximated, combined with the remaining accurate computation.

Finally, the function as a whole can be realized by some numerical approximation algorithms.

The above methods are also shown in the experiments in Ref. [31]. A direct inference is that if a certain objective function (or part of the objective function's algorithm) cannot be approximated (or is not suitable to be approximated), then it can be achieved through precise calculations.

There are many ways to achieve approximation, and "approximate" computation may not introduce errors. For example, certain logic functions can be looked up through truth tables; neural network can achieve Boolean functions accurately[33]. Further, Tianjic chips[5] use lookup tables to achieve certain non-linear functions, without affecting the result of applications.

4.4 Neuromorphic completeness and approximate computation

As mentioned above, the computational process of a neuromorphic complete system could be a combination of accurate computation and approximation; thus, approximate computation is a way to realize a neuromorphic complete system.

On the other hand, the concrete realization means can be diverse (rather than approximate computation), such as the above-mentioned "certain logic functions can be looked up through truth tables" (there is no error in the process), fitting a surface of the high dimensional vector space through a neural network (the target function is unknown), and training neural networks for symbolic computation[34] (scientific computation can be divided into numerical computation and symbolic computation, while approximate computation is applied to the former), and so on.

4.5 Software/hardware decoupling and software/hardware co-design

The principle of software/hardware co-design is widely applied in the field of domain-specific architecture. Although this paper has emphasized the decoupling, these two are not contradictory. Software/hardware co-design is an important means to overcome the challenges faced by brain-inspired computing.

In our opinion, the decoupling method based on completeness is the design foundation, which provides the feasibility of judging the compatibility between hardware and software (i.e., it is useful to determine whether brain-inspired computing systems are general purpose). On this basis, from a performance/energy efficiency perspective, to design dedicated hardware elements/primitives for typical algorithms or operations (i.e., to expose more functionalities of hardware to the programming level for utilization) is necessary, that is, it gives consideration to both functional completeness and application efficiency.

Specifically, the proposed hierarchy[31] decouples programming languages and hardware, and it also benefits the codesign procedure. Owing to the composability of its application-oriented software model, we can abstract a complex operation supported by the hardware into a hardware primitive and further wrap it as an operator on the software level to achieve the equivalent function, that originally requires multiple operators for implementation, without affecting other parts of the software. Thus, it is possible to make full use of new types of hardware without affecting the up-level models, which simplifies the programming.

5 How to Achieve a "Neuromorphic Complete" System Efficiently

As mentioned earlier, the definition of completeness itself is abstract and does not consider how the system is implemented or whether the system itself is neuromorphic; the implementation example[31] is just a reference model. Thus, we should further study the potential technical route towards neuromorphic complete systems.

5.1 Methdology

The premise is to quantify and compare the neuromorphic complete system in the early design, so as to give the target design the right direction and specific optimization guidance. We believe that the early comparison is very necessary, especially when dealing with this type of system with more design space (as neuromorphic completeness introduces a new dimension of system design and optimization).

The feasibility of this idea lies in that between the abstract concept of completeness and the system custom design method, there is sufficient space to occupy the comparison theory and method for computing systems that incorporate specific implementation methods into consideration.

Thus, it is proposed to establish a theory of quantitative analysis of neuromorphic complete system. This theory is to analyze the collection of core operators and patterns from the widely used neuromorphic computation paradigm and representative applications, and then construct a flexible abstract reference model. By analyzing the "differences" between various brain-inspired systems (in the early stage of design) and this abstract model, quantitative analysis and comparison could be made to achieve the research goal. Specifically, this work can be divided into two parts (the workflow is presented in Fig.1):

The software part's input is a dataflow-like graph of the target application (or function); the output is a number of optimal (or approximate optimal) combinations of accurate calculation and approximation (in various ways) to achieve the target function, which are then input to the hardware part for evaluation and further tuning.

Accordingly, the initialization is to construct a design space that can reflect the multi-dimensional evaluation metrics (such as performance, cost, precision, etc.) of brain-inspired computing system, and map the description of the above dataflow-like graph into this space. FSN[31] is such a primary method.

Afterwards, the first step is to analyze the parts that can be achieved in an approximate manner (or manually specified) and find out the optimal (or approximate optimal) combinations, which is followed by replacing approximate part(s) with the corresponding implementations of suitable method (in the following content, we take the approximation in terms of neural networks as the example).

Step 1: Through static analyses, find out the parts that can be approximated, and then carry out neural network training with different hyper-parameters to obtain the approximation scheme. The difficulty lies in that the design space is very huge and then it is not feasible to traverse all possibilities. So it is necessary to accelerate this process by rapid performance prediction and optimal solution search, including (1) how to determine the approximation granularity of the part, as different parts can be disassembled further or reassembled together, (2) how to handle the control flow, and (3) how to judge the merits of many different schemes.

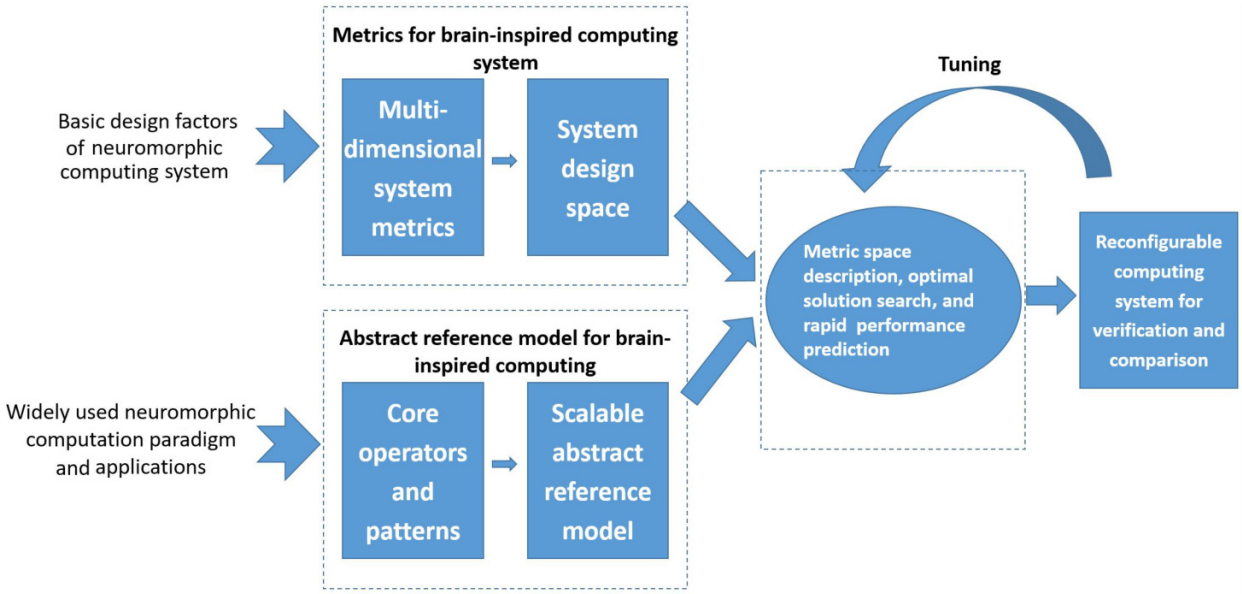


Fig. 1 Workflow to quantify and compare neuromorphic complete systems in the early design.

Step 2: Replace approximate parts. We should link the cost function to specific hardware for overall evaluation, rather than simply replacing the part with Neural Networks (NN).

In terms of hardware, what needs to be studied is a prototype of a reconfigurable brain-inspired computing system for verification and co-design. It contains the following two jobs:

(1) Based on the reconfigurable properties of FPGA, a scalable and flexible simulation prototype is helpful. On one hand, it can fully realize all primary elements of brain-inspired computational core operators, and all kinds of elements should be extensible and composable. On the other hand, it is necessary to make full use of on-chip resources of FPGA to realize the features of in-situ and event-driven computation, so as to reflect the features of brain-inspired computing paradigm more accurately and make the behavior of the system more representative.

(2) The automatic design tool for brain-inspired system is needed, too. Based on the reconfigurable properties, automation (or semi-automation with manual intervention) is implemented according to the given optimization direction and guidance scheme for verification of the theoretical results.

5.2 Reserach foundation

Our research team has been designing custom chips, basic software, and computing systems since 2015 to efficiently support the brain-inspired computing paradigms and algorithms. We also participated in the research and development of the series of Tianjic chip[5] led by the Center for Brain-inspired Computing Research of Tsinghua University. The main work is centered on the establishment of the brain-inspired computer hierarchy (Fig. 2), including the following aspects:

(1) Spiking neural network programming language

We proposed the general SNN description language, E-PyNN, as well as its compiler and simulator[35] for parallel heterogeneous systems; the latter are optimized for the inherent characteristics of SNN. Compared with some state-of-the-art SNN simulators on General-Purpose Graphics Processing Unit (GPGPU), the proposed simulator is 1.41 to 9.33 times faster than the state-of-the-art SNN simulator on GPGPU. Moreover, its performance is 1 to 2 orders of magnitude faster than other commonly-used simulators on CPU. The work laid the foundation for the current application-oriented software model[31].

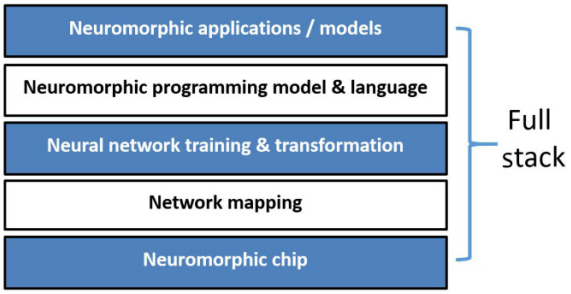


Fig. 2 Hierarchy of neuromorphic computing system.

(2) Neural network training and transformation

We proposed the "training before constraint" strategy and the corresponding workflow of neural network for neuromorphic chips[36]. The input is Deep Neural Network (DNN)/SNN trained by the existing third-party NN development framework (no hardware constraints are involved), and the output is the equivalent neural network that conforms to hardware constraints of the target chip. Afterwards, we enhanced the above work to support a broader range of NN functions[37]. These works are the foundation for compilation based on neuromorphic complete primitives[31].

(3) Neuromorphic chip design

We proposed and designed the brain-inspired chip with "reduced instruction set", FPSA[38, 39], which can give full play to the advantages of Resistive Random Access Memory (ReRAM). Similar with the Reduced Instruction Set Computer (RISC) general-purpose computer, FPSA only provides limited and compact hardware primitives, while the software (compiler) supports the full functions of different neural networks through conversion or approximation methods. This principle is in contrast to most existing chips, existing brain-inspired chips tend to support a wide variety of complex primitives, because the hardware interface is defined by specific software requirements or hardware specifications. One of the hardware platforms (simulation) used in the experiments in Ref. [31] is FPSA. FPSA also investigated mechanisms for efficiently mapping the results of neural network training and transformation onto chip.

We have also developed SNN simulators based on FPGA[40] and software simulation tools[41] for novel non-volatile memory devices.

6 Conclusion and Outlook

The next decade will be the golden age of computer architecture development[42], and brain-inspired computing is one of the most promising

solutions. The design philosophy that this article focuses on is to construct a flexible, adaptive, and software/hardware decoupling system hierarchy of brain-inspired computing based on the proposed completeness, which is conducive to the promotion of the collaborative development of this interdisciplinary field.

From the perspective of the development history of general-purpose computers, the computational completeness and software/hardware decoupling hierarchy laid the theory and architecture foundation for vigorous development. Our work has been inspired by this historical process and is conducive to enabling all kinds of personnel participating in this interdisciplinary research to focus on their professional fields and improve the efficiency of research and development, i.e., promotes the progress of this field from isolated research to collaborative and iterative development. This would be one of the keys to the rapid development of brain-inspired computing systems and the formation of scale industries, which will also facilitate the development of high-efficiency computing systems, including supercomputing systems[43]. We are going to open-source the implementation to try our best to promote this process.

Acknowledgment

This work was partly supported by the National Natural Science Foundation of China (Nos. 62072266 and 62050340) and Beijing Academy of Artificial Intelligence (No. BAAI2019ZD0403).

References

[1] D. Hassabis, D. Kumaran, C. Summerfield, and M. Botvinick, Neuroscience-inspired artificial intelligence, *Neuron*, vol. 95, no. 2, pp. 245–258, 2017.

[2] A. H. Marblestone, G. Wayne, and K. P. Kording, Toward an integration of deep learning and neuroscience, *Front. Comput. Neurosci.*, DOI: 10.3389/fncom.2016.00094.

[3] B. A. Richards, T. P. Lillicrap, P. Beaudoin, Y. Bengio, R. Bogacz, A. Christensen, C. Clopath, R. P. Costa, A. de Berker, S. Ganguli, et al., A deep learning framework for neuroscience, *Nat. Neurosci.*, vol. 22, no. 11, pp. 1761–1770, 2019.

[4] K. Roy, A. Jaiswal, and P. Panda, Towards spike-based machine intelligence with neuromorphic computing, *Nature*, vol. 575, no. 7784, pp. 607–617, 2019.

[5] J. Pei, L. Deng, S. Song, M. G. Zhao, Y. H. Zhang, S. Wu, G. R. Wang, Z. Zou, Z. Z. Wu, W. He, et al., Towards artificial general intelligence with hybrid Tianjic chip architecture, *Nature*, vol. 572, no. 7767, pp. 106–111, 2019.

[6] M. M. Waldrop, The chips are down for Moore's law, *Nature*, vol. 530, no. 7589, pp. 144–147, 2016.

[7] G. S. Wu, Ten fronties for big data technologies (Part B), (in Chinese), *Big Data Res.*, vol. 1, no. 3, pp. 113–123, 2015.

[8] G. S. Wu, Ten fronties for big data technologies (Part A), (in Chinese), *Big Data Res.*, vol. 1, no. 2, pp. 109–117, 2015.

[9] J. D. Kendall and S. Kumar, The building blocks of a brain-inspired computer, *Appl. Phys. Rev.*, vol. 7, no. 1, p. 011305, 2020.

[10] C. Mead, Neuromorphic electronic systems, *Proc. IEEE*, vol. 78, no. 10, pp. 1629–1636, 1990.

[11] C. D. Schuman, T. E. Potok, R. M. Patton, J. D. Birdwell, M. E. Dean, G. S. Rose, and J. S. Plank, A survey of neuromorphic computing and neural networks in hardware, arXiv preprint arXiv: 1705.06963, 2017.

[12] N. Wang, G. G. Guo, B. N. Wang, and C. Wang, Traffic clustering algorithm of urban data brain based on a hybrid-augmented architecture of quantum annealing and brain-inspired cognitive computing, *Tsinghua Sci. Technol.*, vol. 25, no. 6, pp. 813–825, 2020.

[13] W. Maass, Networks of spiking neurons: The third generation of neural network models, *Neural Netw.*, vol. 10, no. 9, pp. 1659–1671, 1997.

[14] M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov, Training and operation of an integrated neuromorphic network based on metal-oxide memristors, *Nature*, vol. 521, no. 7550, pp. 61–64, 2015.

[15] C. Q. Yin, Y. X. Li, J. B. Wang, X. F. Wang, Y. Yang, and T. L. Ren, Carbon nanotube transistor with short-term memory, *Tsinghua Sci. Technol.*, vol. 21, no. 4, pp. 442–448, 2016.

[16] X. Lagorce and R. Benosman, STICK: Spike time interval computational kernel, a framework for general purpose computation using neurons, precise timing, delays, and synchrony, *Neural Comput.*, vol. 27, no. 11, pp. 2261–2317, 2015.

[17] J. B. Aimone, W. Severa, and C. M. Vineyard, Composing neural algorithms with Fugu, in *Proc. Int. Conf. Neuromorphic Systems*, Knoxville, TN, USA, 2019, pp. 1–8.

[18] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura, et al., A million spiking-neuron integrated circuit with a scalable communication network and interface, *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[19] S. K. Esser, A. Andreopoulos, R. Appuswamy, P. Datta, D. Barch, A. Amir, J. Arthur, A. Cassidy, M. Flickner, P. Merolla, et al., Cognitive computing systems: Algorithms and applications for networks of neurosynaptic cores, in *Proc. 2013 Int. Joint Conf. on Neural Networks*, Dallas, TX, USA, 2013, pp. 1–10.

[20] A. Amir, P. Datta, W. P. Risk, A. S. Cassidy, J. A. Kusnitz, S. K. Esser, A. Andreopoulos, T. M. Wong, M. Flickner, R. Alvarez-Icaza, et al., Cognitive computing programming

paradigm: A corelet language for composing networks of neurosynaptic cores, in *Proc. 2013 Int. Joint Conf. on Neural Networks*, Dallas, TX, USA, 2013, pp. 1–10.

[21] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G. J. Nam, 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.*, vol. 34, no. 10, pp. 1537–1557, 2015.

[22] B. V. Benjamin, P. R. Gao, E. McQuinn, S. Choudhary, A. R. Chandrasekaran, J. M. Bussat, R. Alvarez-Icaza, J. V. Arthur, P. A. Merolla, and K. Boahen, Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations, *Proc. IEEE*, vol. 102, no. 5, pp. 699–716, 2014.

[23] P. Merolla, J. Arthur, R. Alvarez, J. M. Bussat, and K. Boahen, A multicast tree router for multichip neuromorphic systems, *IEEE Trans. Circuits Syst. I Reg. Pap.*, vol. 61, no. 3, pp. 820–833, 2014.

[24] C. Eliasmith and C. H. Anderson, *Neural Engineering: Computation, Representation, and Dynamics in Neurobiological Systems*. Cambridge, MA, USA: The MIT Press, 2004.

[25] A. Neckar, S. Fok, B. V. Benjamin, T. C. Stewart, N. N. Oza, A. R. Voelker, C. Eliasmith, R. Manohar, and K. Boahen, Braindrop: A mixed-signal neuromorphic architecture with a dynamical systems-based programming model, *Proc. IEEE*, vol. 107, no. 1, pp. 144–164, 2019.

[26] M. Davies, N. Srinivasa, T. H. Lin, G. Chinya, Y. Q. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, et al., Loihi: A neuromorphic manycore processor with on-chip learning, *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[27] C. K. Lin, A. Wild, G. N. Chinya, Y. Q. Cao, M. Davies, D. M. Lavery, and H. Wang, Programming spiking neural networks on Intel's Loihi, *Computer*, vol. 51, no. 3, pp. 52–61, 2018.

[28] K. Wendt, M. Ehrlich, and R. Schüffny, A graph theoretical approach for a multistep mapping software for the facets project, in *Proc. 2nd WSEAS Int. Conf. on Computer Engineering and Applications*, Capulco, Mexico, 2008, pp. 189–194.

[29] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown, Overview of the SpiNNaker system architecture, *IEEE Trans. Comput.*, vol. 62, no. 12, pp. 2454–2467, 2013.

[30] O. Rhodes, P. A. Bogdan, C. Brenninkmeijer, S. Davidson, D. Fellows, A. Gait, D. R. Lester, M. Mikaitis, L. A. Plana, A. G. D. Rowley, et al., sPyNNaker: A software package for running PyNN simulations on SpiNNaker, *Front. Neurosci.*, vol. 12, p. 816, 2018.

[31] Y. H. Zhang, P. Qu, Y. Ji, W. H. Zhang, G. R. Gao, G. R. Wang, S. Song, G. Q. Li, W. G. Chen, W. M. Zheng, et al., A system hierarchy for brain-inspired computing, *Nature*, vol. 586, no. 7829, pp. 378–384, 2020.

[32] O. Rhodes, Brain-inspired computing boosted by new concept of completeness, *Nature*, vol. 586, no. 7829, 364–366, 2020.

[33] B. Steinbach and R. Kohut, Neural networks – A model of Boolean functions, in *Proc. 5th Int. Workshop on Boolean Problems*, Freiberg, Germany, 2002.

[34] G. Lample and F. Charton, Deep learning for symbolic mathematics, in *Proc. 8th Int. Conf. on Learning Representations*, Addis Ababa, Ethiopia, 2020.

[35] P. Qu, Y. H. Zhang, X. Fei, and W. M. Zheng, High performance simulation of spiking neural network on GPGPUs, *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 11, pp. 2510–2523, 2020.

[36] Y. Ji, Y. H. Zhang, S. C. Li, P. Chi, C. H. Jiang, P. Qu, Y. Xie, and W. G. Chen, NEUTRAMS: Neural network transformation and co-design under neuromorphic hardware constraints, in *Proc. 49th Ann. IEEE/ACM Int. Symp. on Microarchitecture*, Taipei, China, 2016, pp. 1–13.

[37] Y. Ji, Y. H. Zhang, W. G. Chen, and Y. Xie, Bridge the gap between neural networks and neuromorphic hardware with a neural network compiler, in *Proc. 23rd Int. Conf. on Architectural Support for Programming Languages and Operating Systems*, Williamsburg, VA, USA, 2018, pp. 448–460.

[38] Y. Ji, Y. Y. Zhang, X. F. Xie, S. C. Li, P. Q. Wang, X. Hu, Y. H. Zhang, and Y. Xie, FPSA: A full system stack solution for reconfigurable ReRAM-based NN accelerator architecture, in *Proc. 24th Int. Conf. on Architectural Support for Programming Languages and Operating Systems*, New York, NY, USA, 2019, pp. 733–747.

[39] Y. Ji, Z. X. Liu, and Y. H. Zhang, A reduced architecture for ReRAM-based neural network accelerator and its software stack, *IEEE Trans. Comput*., vol. 70, no, 3, pp. 316–331, 2021.

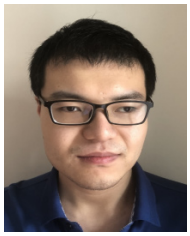
[40] J. H. Han, Z. L. Li, W. M. Zheng, and Y. H. Zhang, Hardware implementation of spiking neural networks on FPGA, *Tsinghua Sci. Technol.*, vol. 25, no. 4, pp. 479–486, 2020.

[41] X. Fei, Y. H. Zhang, and W. M. Zheng, XB-SIM*: A simulation framework for modeling and exploration of ReRAM-based CNN acceleration design, *Tsinghua Sci. Technol.*, vol. 26, no. 3, pp. 322–334, 2021.

[42] J. L. Hennessy and J. L. Hennessy, A new golden age for computer architecture: Domain-specific hardware/software co-design, enhanced security, open instruction sets, and agile chip development, in *Proc. 2018 ACM/IEEE 45th Ann. Int. Symp. on Computer Architecture*, Los Angeles, CA, USA, 2018, pp. 27–29.

[43] W. M. Zheng, Research trend of large-scale supercomputers and applications from the TOP500 and Gordon Bell Prize, *Sci. China Inf. Sci.*, vol. 63, no. 7, p. 171001, 2020.

Youhui Zhang received the BEng and PhD degrees in computer science from Tsinghua University, Beijing, China in 1998 and 2002, respectively. He is currently a professor in the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His research interests include computer architecture and neuromorphic computing. He is a member of CCF, ACM, and IEEE.



Peng Qu received the BEng and PhD degrees in computer science from Tsinghua University, Beijing, China in 2013 and 2018, respectively. He is currently a postdoctoral fellow in the Department of Computer Science and Technology, Tsinghua University, Beijing. His research interests include computer architecture and neuromorphic computing.

Weimin Zheng received the MEng degree in computer science from Tsinghua University, Beijing, China in 1982. Currently he is an academician of Chinese Academy of Engineering and a professor at the Department of Computer Science and Technology, Tsinghua University, Beijing, China. His research interests include high performance computing, network storage, and parallel compiler.