Tianjic: A Unified and Scalable Chip Bridging Spike-Based and Continuous Neural Computation

Lei Deng, *Member, IEEE*, Guanrui Wang, Guoqi Li, *Member, IEEE*, Shuangchen Li, Ling Liang, Maohua Zhu, Yujie Wu, Zheyu Yang, Zhe Zou, Jing Pei, Zhenzhi Wu, Xing Hu, Yufei Ding, Wei He, Yuan Xie, *Fellow, IEEE*, and Luping Shi

***Abstract*— Toward the long-standing dream of artificial intelligence, two successful solution paths have been paved: 1) neuromorphic computing and 2) deep learning. Recently, they tend to interact for simultaneously achieving biological plausibility and powerful accuracy. However, models from these two domains have to run on distinct substrates, i.e., neuromorphic platforms and deep learning accelerators, respectively. This architectural incompatibility greatly compromises the modeling flexibility and hinders promising interdisciplinary research. To address this issue, we build a unified model description framework and a unified processing architecture (Tianjic), which covers the full stack from software to hardware. By implementing a set of integration and transformation operations, Tianjic is able to support spiking neural networks, biological dynamic neural networks, multilayered perceptron, convolutional neural networks, recurrent neural networks, and so on. A compatible routing infrastructure enables homogeneous and heterogeneous scalability on a decentralized many-core network. Several optimization methods are incorporated, such as resource and data sharing, near-memory processing, compute/access skipping, and intra-/inter-core pipeline, to improve performance and efficiency. We further design streaming mapping schemes for efficient network deployment with a flexible tradeoff between execution throughput and resource overhead. A 28-nm prototype chip is fabricated with >610-GB/s internal memory bandwidth. A variety of benchmarks are evaluated and compared with GPUs and several existing specialized platforms. In summary, the fully unfolded mapping can achieve significantly higher throughput and power efficiency; the semi-folded mapping can save 30x resources while still presenting comparable performance on average. Finally, two hybrid-paradigm examples, a multimodal unmanned bicycle and a hybrid neural network, are demonstrated to show the potential of our unified architecture. This article paves a new way to explore neural computing.**

***Index Terms*— Deep learning accelerator, hybrid paradigm, neuromorphic chip, unified/scalable architecture.**

Manuscript received July 31, 2019; revised November 2, 2019; accepted January 22, 2020. Date of publication February 13, 2020; date of current version July 23, 2020. This article was approved by Associate Editor Edith Beigne. This work was supported in part by the National Natural Science Foundation of China under Project 61836004 and Project 61327902, in part by the National Key R&D Program of China under Grant 2018YFE0200200, in part by the Brain-Science Special Program of Beijing under Grant Z181100001518006, and in part by the Suzhou-Tsinghua Innovation Leading Program under Grant 2016SZ0102. *(Lei Deng and Guanrui Wang contributed equally to this work.) (Corresponding author: Luping Shi.)*

Lei Deng is with the Center for Brain Inspired Computing Research, Department of Precision Instrument, Tsinghua University, Beijing 100084, China, and also with the Department of Electrical and Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA 93106 USA (e-mail: leideng@ucsb.edu).

Guanrui Wang, Guoqi Li, Yujie Wu, Zheyu Yang, Zhe Zou, Jing Pei, and Luping Shi are with the Center for Brain Inspired Computing Research, Tsinghua University, Beijing 100084, China, also with the Beijing Innovation Center for Future Chip, Tsinghua University, Beijing 100084, China, and also with the Department of Precision Instrument, Tsinghua University, Beijing 100084, China (e-mail: wgr16@mails.tsinghua.edu.cn; wu-yj16@mails.tsinghua.edu.cn; zy-yang17@mails.tsinghua.edu.cn; zouz16@mails.tsinghua.edu.cn; liguoqi@mail.tsinghua.edu.cn; peij@mail.tsinghua.edu.cn; lpshi@mail.tsinghua.edu.cn).

Shuangchen Li, Ling Liang, Maohua Zhu, Xing Hu, and Yuan Xie are with the Department of Electrical and Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA 93106 USA (e-mail: shuangchenli@ucsb.edu; lingliang@ucsb.edu; maohuazhu@ucsb.edu; xinghu@ucsb.edu; yuanxie@ucsb.edu).

Zhenzhi Wu and Wei He are with Lynxi Technology Co., Ltd., Beijing 100097, China (e-mail: zhenzhi.wu@lynxi.com; wei.he@lynxi.com).

Yufei Ding is with the Department of Computer Science, University of California at Santa Barbara, Santa Barbara, CA 93106 USA (e-mail: yufeiding@cs.ucsb.edu).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSSC.2020.2970709

I. Introduction

A. Two Approaches Toward Artificial Intelligence

THROUGHOUT the development history of artificial intelligence, two solution paths have been successfully paved, i.e., neuromorphic computing and deep learning. The former targets at harnessing neuroscience to extract insights for efficient brain-like computing by studying the detailed implementation of neural dynamics, circuits, coding, and learning. Although our understanding of how the brain works is limited and there exists skepticism on neuromorphic computing [1], this bio-plausible way still holds great potential with the brain's advocation. By contrast, the latter aims at solving practical tasks with high accuracy by eschewing most neuroscience details and using brute-force cost optimization (e.g., gradient descent). With the help of big data (e.g., ImageNet), high-performance processors (e.g., GPU), effective models (e.g., artificial neural networks, ANNs), and easy-to-use programming tools (e.g., Pytorch/Tensorflow), it has achieved human-level performance in a broad spectrum of scenarios [2]–[5].

B. Typical Models of Neural Networks

The neural network models can also be correspondingly categorized into neuromorphic ones and deep learning ones. Spiking neural networks (SNNs) [6], [7], mainly governed by continuous dynamics of neuronal membrane potential and

discrete firing of spike signal (0-nothing or 1-spike event), closely mimic the behaviors of the brain cortex, including spatio-temporal dynamics, sparse coding, and probabilistic firing. SNNs are helpful for probabilistic inference and heuristic solution of hard computational problems [8], as well as recognition task [9], [10] or sparse representation [11], [12]. A variant of SNNs, with similar membrane potential dynamics but directly propagating a continuous and normalized firing rate between neurons [13]–[15], is termed biological dynamic neural networks (BDyNNs) in this article. We show that BDyNNs can be used for object tracking in a video stream. On the other side, deep learning models dominate the machine learning field in recent years, especially the ANNs that represent a complex mapping from inputs to outputs using nonlinear neurons with continuous response but without intrinsic dynamics. In the ANN family, multilayered perceptron (MLP) is a feedforward network with only fully connected (FC) layers, while convolutional neural networks (CNNs) combine both convolutional (Conv) layers and FC layers [16], [17]. MLP and CNNs are widely used in image recognition and detection tasks [2], [17]–[19]. By contrast, recurrent neural networks (RNNs) [20] have intra-layer feedback connections, which brings a similar temporal dynamics with BDyNNs. Long short-term memory (LSTM) [21] is a typical RNN model, which is usually adopted in sequence learning for speech recognition and natural language processing [22]–[25].

C. Two Types of Specialized Hardware Substrates

Two branches have been developed for efficient AI hardware. On one hand, neuromorphic platforms support SNNs through massively parallel components (neurons and synapses) and a communication network. They can be constructed by mixed analog-digital circuits [26]–[28], fully digital circuits [29]–[31], or emerging memory devices [32], [33]. On the other hand, deep learning accelerators are designed for ANNs to accelerate execution and save energy compared with general processors. Current solutions usually leverage parallel processing elements (PEs), optimized access locality [34], near-/in-memory computing [35]–[38], and efficient model compression or data reuse [39]–[47]. Because the above-mentioned two substrates have very different paradigms regarding compute, memory, and communication, they follow independent design philosophies.

D. Trend of Cross Research

Although neuromorphic computing and deep learning speak different languages, they recently start interacting with each other via complementary advantages toward the exploration of artificial general intelligence [48]–[50]. Keeping the biological foundation, neuromorphic models introduce the mature training algorithms in deep learning (e.g., error backpropagation) [51]–[60] or directly equip an ANN classifier at the distal end of the SNN network [61] to bridge the accuracy gap between SNNs and ANNs on recognition tasks. Emerging neuronal dynamics [62], [63] or synaptic plasticity [64] try to obtain the compatibility with deep learning framework.

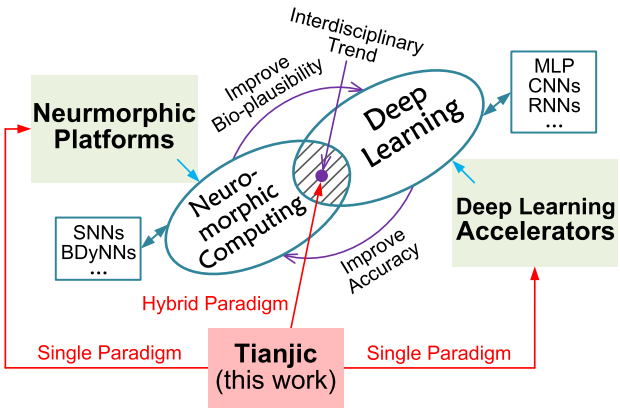


Fig. 1. Main idea of the unified architecture.

Interestingly, the discussion about whether there is deep cortical circuit with "deep learning" in the brain emerges [65]. Correspondingly, deep learning models also begin to borrow ideas from neuroscience, such as neuroscience-inspired measures to study the model generalization [66], neural group coding-based CNN model for robust representation [67], differentiable neural computer with human-like long-term memory [68], and human mind-inspired ToMnet [69]. The direct performance comparison between ANNs and SNNs also comes out [70]. On the hardware side, researchers in both neuromorphic and deep learning areas start paying attention to each other, such as using the same emerging devices for SNNs [32], [33] and ANNs [71] separately, comparing the hardware design of ANNs and SNNs [72], transferring ANNs to SNNs for the running on neuromorphic chips [73] or for the removal of costly ADCs/DACs in memristor-based deep learning accelerators [74], and combining the spatial architecture of neuromorphic chips with the temporal architecture of deep learning accelerators for ANNs [75].

E. Unified Architecture for Hybrid Paradigm

To support the hybrid computing paradigm from software to hardware, Tianjic [76] integrates neuromorphic computing and deep learning into a unified description framework using five building blocks. As shown in Fig. 1, via real fusion of these two paradigms, not simply putting them side by side, Tianjic enables both homogeneous and heterogeneous modeling scalability. In this article, we present the architecture design details of Tianjic. Several optimization methods and flexible streaming mapping schemes are designed, which enable high execution performance. We fabricate a prototype chip to verify our design and achieve better results on average across many single-paradigm models (including SNNs, BDyNNs, MLP, CNNs, and RNNs) for a variety of tasks (including image recognition, language processing, dynamic vision sensor processing, and object tracking). We also provide two examples to present the potential of the hybrid paradigm.

The contributions of this article are listed as follows.

1) We design a unified architecture, i.e., Tianjic, to fuse neuromorphic computing and deep learning from software to hardware. The unified model description framework covers a broad spectrum of neural networks

using five building blocks. Tianjic inherits the many-core scalability, and it allows both homogeneous networks with high performance and the cross modeling of heterogeneous networks. We not only provide a unified substrate for two single domains but also demonstrate a promising solution to promote interdisciplinary research.

2) A set of optimization techniques is implemented to improve execution performance. Specifically, resource sharing between SNN and ANN paradigms reduces the area cost; near-memory processing, data sharing, compute/access skipping, and intra-core pipeline reduce the latency and power consumption.
3) Flexible streaming mapping schemes are developed for efficient network deployment, which allows a streaming dataflow of neuronal activations and the inter-core pipeline execution. The throughput-aware fully unfolded mapping achieves higher performance compared with existing platforms; the resource-aware semi-folded mapping greatly reduces resource overheads while still achieving comparable performance.
4) We fabricate a prototype chip to verify our design, which presents advanced performance on various single-paradigm models involving different scenarios. Two extra application examples, a multimodal unmanned bicycle with individual ANNs/SNNs and a hybrid neural network model with fused ANN/SNN signals, are demonstrated to show the potential of our hybrid paradigm.

II. Preliminaries

Different neural network models are usually coded in different signal formats and governed by different operations of integration/transformation. Note that because it is still challenging to unify the distinct learning rules, we focus on the inference rather than training in this article.

A. Neuroscience-Oriented Models

1) SNNs: SNNs stem from cortex networks that use spike to represent information and communicate between the neurons. A neuron in SNNs is comprised of dendrite, soma, and axon and then connected with other neurons through synapses. The dendrite integrates pre-inputs considering the synaptic influence on the transmission efficacy (i.e., synaptic weight), and the soma nonlinearly transforms the integrated signal to membrane potential and then generates the spike event by comparing with a firing threshold. The axon transmits the spike event to post-neurons. The classic leaky integrate and fire (LIF) [77] model of SNNs is governed by

$$\begin{cases} \tau \dfrac{dV_i(t)}{dt} = -[V_i(t) - V_{r_1}] + \sum\limits_j w_{ij} \sum\limits_{t_j^k \in [t - T_w,\ t]} K\left(t - t_j^k\right) \\ \begin{cases} s_i(t) = 1\ V_i(t) = V_{r_2}, & \text{if } V_i(t) \geq V_i^{th} \\ s_i(t) = 0, & \text{if } V_i(t) < V_i^{th} \end{cases} \end{cases} \tag{1}$$

where (t) denotes the simulation time step, V_i and s_i are the membrane potential and spike signal of the ith neuron, respectively, τ is a time constant, V_{r_1} is the resting potential, w_{ij} is the synaptic weight from the jth neuron to the ith neuron, t_j^k is the time step when the kth spike from the jth input neuron comes during the past integration time window of T_w, $K(\cdot)$ is a kernel function describing the temporal decay effect that a more recent spike should have a greater impact on the post-synaptic membrane potential, V_{r_2} is the reset potential, and V_i^{th} is the neuronal firing threshold of the ith neuron.

The current membrane potential is jointly determined by the inputs within T_w and the historical membrane potential, which endows SNNs the ability to memorize historical information for the temporal association. The spatio-temporal spike pattern further provides rich complexity for information representation. Spike signals are usually sparse, and the processing can be event-driven, i.e., enabling the integration once pre-spikes come while disabling it if nothing received. Furthermore, because a spike is binary (0/1), there is only addition operation during the dendritic integration (if $T_w = 1$ and $K(\cdot) \equiv 1$). All these properties result in low power consumption that has been evidenced by neuromorphic chips.

2) BDyNNs: The information representation in SNNs can exploit the spike timing or rate. At the case of rate coding, lots of models ignore the representation with binary spikes and only transmit the continuous spike rate value (such as the activation in ANNs) between neurons, which is termed BDyNNs here. They usually have similar temporal dynamics with SNNs for the update of membrane potential, but adding a nonlinear normalization function to generate the spike rate based on the membrane potential, as follows:

$$\begin{cases} \tau \dfrac{dV_i(t)}{dt} = -[V_i(t) - V_r] + \sum\limits_j w_{ij} r_j(t) + w_{ie} r_{ie}(t) \\ r_i(t) = \text{Norm}(\boldsymbol{V}(t)) \end{cases} \tag{2}$$

where most parameters have the same definition as (1). The rest are defined as follows: r_i and r_j are the spike rate of the ith and jth neuron, respectively, V_r is the resting potential, $r_{ie}(t)$ is the ith external stimulus, and $\text{Norm}(\cdot)$ denotes a normalization function that constrains the dynamic range of spike rate, such as $r_i(t) = V_i^2(t)/[1 + k \sum_j V_j^2(t)]$, where k is a scaling constant.

B. Deep Learning Models

1) MLP: It only includes feedforward FC layers, and the basic computation rule is

$$\mathbf{Y} = \varphi(\mathbf{b} + \mathbf{W}\mathbf{X}) \tag{3}$$

where $\mathbf{Y}$ is the activation of neurons in the current layer, $\mathbf{X}$ is the activation of neurons from the previous layer, $\mathbf{W}$ is the weight matrix connecting adjacent two layers, and $\mathbf{b}$ is a bias vector. $\varphi(\cdot)$ is a nonlinear activation function, e.g., $\varphi(x) = \text{ReLU}(x) = \max(x, 0)$. In MLP, the activation state has nothing to do with the historical state, unlike the SNNs and BDyNNs.

2) CNNs: Different from the 1-D feature organization of MLP, CNNs directly process 2-D feature maps (FMs). First, the operation of Conv layers is governed by

$$\mathbf{FM}_n^{\text{out}} = \varphi\left(b_n + \sum_m \mathbf{FM}_m^{\text{in}} \circledast \mathbf{W}(m, n)\right) \tag{4}$$

where $\mathbf{FM}_n^{\text{out}}$ is the nth output FM, i.e., a 2-D matrix of neuronal activations, b_n is a bias item shared by all neurons in $\mathbf{FM}_n^{\text{out}}$, $\mathbf{FM}_m^{\text{in}}$ is the mth input FM, $\mathbf{W}(m, n)$ is a 2-D weight kernel (e.g., 3×3 size) connecting $\mathbf{FM}_m^{\text{in}}$ and $\mathbf{FM}_n^{\text{out}}$, $\circledast$ is a 2-D convolutional operation, and $\varphi(\cdot)$ is the same as that in MLP. Second, pooling layers are used to downsample the size of FMs, i.e., $\mathbf{FM}_n^{\text{out}} = \text{Pool}(\mathbf{FM}_n^{\text{in}})$, where Pool$(\cdot)$ denotes a pooling operation that generates each output activation based on only the local receptive filed (RF) in the corresponding input FM. Max (average) pooling uses the maximum (average) value of each RF. Finally, FC layers follow the Conv and pooling layers to form a complete CNN model.

3) RNNs: Although MLP and CNNs are powerful in image recognition, it is quite difficult for them to handle sequence learning problems. To this end, RNNs are designed to achieve temporal dynamics by introducing feedback connections. Among various RNN models, LSTM is the most widely used. Its dynamics can be describe as

$$\begin{cases} \mathbf{f}/\mathbf{i}/\mathbf{o}(t) = \sigma(\mathbf{b}_{f/i/o} + \mathbf{W}_{f/i/ox}\mathbf{X}(t) + \mathbf{W}_{f/i/oh}\mathbf{h}(t-1)) \\ \mathbf{g}(t) = \theta(\mathbf{b}_g + \mathbf{W}_{gx}\mathbf{X}(t) + \mathbf{W}_{gh}\mathbf{h}(t-1)) \\ \mathbf{c}(t) = \mathbf{c}(t-1) \odot \mathbf{f}(t) + \mathbf{g}(t) \odot \mathbf{i}(t) \\ \mathbf{h}(t) = \theta(\mathbf{c}(t)) \odot \mathbf{o}(t) \end{cases} \tag{5}$$

where $\mathbf{f}$, $\mathbf{i}$, and $\mathbf{o}$ are the state of forget, input, and output gates, respectively, and $\mathbf{g}$ is the input activation. Each of them has its own bias vector and weight matrices. $\mathbf{c}$ and $\mathbf{h}$ are the cellular and hidden states of the hidden layer, respectively. $\sigma(\cdot)$ and $\theta(\cdot)$ denote sigmoid and tanh functions, respectively. Although LSTM also has temporal dynamics, it does not mimic the spike activity of SNNs and the possible global normalization of BDyNNs.

C. Model Differences

MLP and CNNs usually do well in feature extraction from static data, such as images while meeting difficulty in processing dynamic information; by contrast, SNNs, BDyNNs, and RNNs achieve that capability by introducing temporal dynamics during the integration of inputs. Despite the similar temporal dynamics of the latter three models, they all own unique behaviors to distinguish from each other. For example, SNNs use binary spikes produced by the thresholded firing mechanism at soma for inter-neuron communication, which brings potential for versatile trajectories in response spike pattern, an intrinsic denoising effect on the neuronal inputs, and a better match for sparse feature handling [70]. Although BDyNNs use continuous signals for inter-neuron communication, similar to RNNs, the possible global normalization function they use is very different from the element-wise activation functions in RNNs.

III. Design Challenges

A. Challenge 1

How to fuse the compute, memory, and communication of distinct neural paradigms, especially at a low cost? We revisit most neural network models in these two domains and then build a unified model description framework using five building blocks to construct a unified functional core (UFC). Six vector/matrix operations and several transformations are supported in each core. A routing infrastructure with homogeneous and heterogeneous scalability connects the many-core network. Compared with the naive solution of building different paradigms separately and putting them side by side, we achieve a high resource utilization and support seamless cross modeling.

B. Challenge 2

How to maximize the efficiency of area, latency, and power? We share resources between the two paradigms to shrink the area significantly. The distributed local memory eliminates the costly off-chip access, and the decentralized many-core architecture greatly improves the internal memory bandwidth and processing parallelism. The data sharing and execution pipeline optimize the running throughput. The access/compute skipping techniques further saves power consumption.

C. Challenge 3

How to get a good mapping tradeoff between the performance and overhead? We design streaming mapping schemes to enable a streaming dataflow of neuronal activations and the inter-core execution pipeline. Moreover, we provide two mapping ways, throughput-aware fully unfolded mapping and resource-aware semi-folded mapping, for a flexible tradeoff between the running throughput and resource overhead. The former allows higher performance at the cost of consuming more resources; the latter dramatically reduces the resource overhead while still achieving advanced performance.

IV. Tianjic Architecture

In this section, we build a unified model description for typical neural networks across neuromorphic computing and deep learning using five building blocks and design a UFC and the resulting many-core network with homogeneous/heterogeneous scalability to implement it. Then, two streaming mapping schemes are proposed for efficient model deployment. Compared with [76], we provide more architectural details in this article, including the architecture of individual blocks, the systematic definition of the supported operations/transformations, the overall timing schedule, the performance optimization techniques, and the routing strategies for the first time. Before explaining the details of our design, we summarize the abbreviations used in this section in Table I to improve the readability.

A. UFC

A naive solution to fuse different neural paradigms is to build separate blocks and put them side by side. However, this method suffers from several drawbacks: 1) limited flexibility in determining an appropriate ratio between different paradigms to adapt to diverse real-world workloads; 2) inefficiencies in building extra inter-block circuits for signal conversion; and 3) poor programmability in managing a heterogeneous system. These concerns motivate us to select a unified architecture.

TABLE I
DEFINITION OF ARCHITECTURAL ITEMS

Item	Description
UFC	Unified functional core
HAB	Hybrid activation buffer (axon)
LSM	Local synapse memory (synapse)
SIE	Shared integration engine (dendrite)
NTU	Nonlinear transformation unit (soma)
NC	Network connector (router)
A/S_MEM	Activation (ANN mode) /Spike (SNN mode) memory
s_reg	Recent spike register
P_MEM	Soma parameter memory
RLUT	Routing look up table
P2P	Point-to-point routing
AMC	Adjacent multicast routing
FM	Feature map
T_w	Integration time window: length of spike train for SIE integration every time
Time Phase	Period for executing a round of intra-UFC computation and inter-UFC routing transmission
Time Step	Coarse-grain time unit containing multiple time phases with flexible on/off phase pattern

TABLE II
DEFINED OPERATIONS AND TRANSFORMATIONS

Operation	Mode	Definition	Resource
VMA	SNN	$\mathbf{y} = \mathbf{W} \cdot \mathbf{s}$	Accumulator
VMM	share	$\mathbf{y} = \mathbf{W} \cdot \mathbf{x}$	Multiplier Accumulator
VVA	share	$\mathbf{y} = \sum_i \mathbf{x}_i, i = 0, 1, ..., \frac{N}{2}$	Accumulator
VVM	ANN	$\mathbf{y} = \mathbf{x}_1 \odot \mathbf{x}_2$	Multiplier
VS	ANN	$\mathbf{y} = x_\alpha \mathbf{x}$	Multiplier
VB	share	$\mathbf{y} = \mathbf{x}$	None
Transformation	**Mode**	**Definition**	**Resource**
B_L	share	$\mathbf{y} = \mathbf{x} + \mathbf{x}_b / \mathbf{x}_{leak}$	Adder
Act_Fun	ANN	$\mathbf{y} = \varphi(\mathbf{x})$	LUT
Spi_Gen	SNN	Thresh_comp V_update | (Spike_fire & V_Reset)	Comparator
Out_Trans	share	Send Output to NC	None
Pooling	share	$y_i = \text{max/ave}(\{x_j \mid j \in \text{pool}_i\})$	Comparator Adder/Shifter
Lat_Acc	share	$y_j = x_j + y_{j-1}$	Adder
Out_Copy	share	$\text{Fire\_data}_i = \text{Fire\_data}_j$	None
Others	SNN	Leakage/Threshold Adaption	Multiplier, Adder
		Probabilistic Firing	RNG

1) Unified Model Description: We revisit almost all the typical neural network models used in both neuromorphic computing and deep learning that have been introduced in Section II. To support them in a unified framework, we first define six vector/matrix operations and several additional transformations used by these models, as listed in Table II. Then, we design a UFC[1]) with five building blocks: hybrid activation buffer (HAB), local synapse memory (LSM), shared integration engine (SIE), nonlinear transformation unit (NTU), and network connector (NC). At last, we split each model into several execution steps and map them onto these building blocks, which is illustrated in Table III. Specifically, the operations and transformations lie in SIE and NTU, respectively; HAB and LSM are designed for data storage, including parameters and states; and NC is a routing infrastructure that connects neurons.

2) Architecture Design Philosophy: From the model abstraction in Table III, we propose two operating modes in each UFC: spiking mode for the spike-based models (i.e., SNNs) and non-spiking mode for the continuous value-based models (i.e., BDyNNs, MLP, CNNs, and RNNs), which are, respectively, termed SNN mode and ANN mode in this article. Note that the neuromorphic model, i.e., BDyNNs, is grouped into the ANN mode because the neurons of BDyNNs communicate with each other using continuous rate values rather than binary spikes. In addition to running the single-paradigm models individually, we hope to support two types of hybrid paradigm: coarse grain and fine grain. The former means a hybrid system with separate ANN and SNN modes in different UFCs, while the latter means a hybrid network with both spiking and non-spiking neurons, which needs UFCs with coexisting ANN and SNN modes to perform the ANN-to-SNN or SNN-to-ANN signal conversion.

To this end, we propose three key design features to achieve the expected hybrid paradigm.

a) Independently reconfigurable HAB and NTU: The configurations of HAB and NTU are independent of each other. HAB can receive and organize SNN inputs (binary spikes) or ANN inputs (non-spiking continuous activations) according to its mode configuration; similarly, NTU can also generate SNN outputs or ANN outputs according to its mode configuration. When HAB and NTU are configured into the same operating mode (either ANN or SNN), the UFC processes pure ANN signal or SNN signal, respectively. This is a straightforward implementation that enables the support of single-paradigm workloads or a coarse-grain hybrid system. When HAB and NTU are configured into different operating modes (e.g., HAB in ANN and NTU in SNN, or vice versa), the UFC processes ANN and SNN signals simultaneously, termed a hybrid UFC operating in the hybrid mode. For example, if the UFC receives continuous activations in HAB, conducts the vector-matrix multiplication in SIE, and generates binary spikes via the thresholded comparison in NTU, this "ANN-input and SNN-output" dataflow naturally performs spike encoding as an ANN-to-SNN converter. Similarly, if the UFC processes "SNN-input and ANN-output" dataflow, it behaves as an SNN-to-ANN converter. This is a carefully designed implementation that enables the support of fine-grain hybrid workloads, which helps the exploration of novel neural models.

b) Shared Dendritic Integration in SIE: The dendritic integration under both SNN and ANN modes share the same

[1]The functional core (FCore in [76] or FunC in [78] and [79]) in our previous work is denoted as UFC in this article to emphasize the unified architecture. The axon, synapse, dendrite, soma, and router modules are renamed as HAB, LSM, SIE, NTU, and NC, respectively, to highlight the design features.

TABLE III
UNIFIED MODEL DESCRIPTION FOR TYPICAL NEURAL NETWORKS WITH FIVE BUILDING BLOCKS

	Neuromorphic Models		**Deep Learning Models**	
	SNNs	**BDyNNs**	**MLP/CNNs**	**RNNs** (taking LSTM as an exmple)
HAB	$\mathbf{s}_e$, $\mathbf{S}^{T_w}$, $\mathbf{s}$	$\mathbf{r}_e$, $\mathbf{r}$, Intermediate Variables	$\mathbf{x}$, $\mathbf{y}$	$\mathbf{x}$, $\mathbf{h}$, Intermediate Variables
LSM	Weight Matrix	Weight Matrix, $\mathbf{V}$ Intermediate Variables	Conv: Weight Kernels Pool: Disabled FC: Weight Matrix	Gates: Weight Matrices Cell State: $\mathbf{f}$/$\mathbf{i}$/$\mathbf{g}$, $\mathbf{c}$, Intermediate Variables Hidden State: $\mathbf{o}$, $tanh(\mathbf{c})$
SIE	$\sum_j w_{ij} \sum_{t_j^k \in S_j^{T_w}} K(t-t_j^k) + w_{ie} s_{ie}(t)$	$V_i(t) = \sum_j w_{ij} r_j(t-1) + w_{ie} r_{ie}(t)$ Parts of spike_rate: $Norm(\cdot)$	Conv: $\sum_m \mathbf{FM}_m^{in} \otimes \mathbf{W}(m,n)$ Pool: Data Copy FC: $\mathbf{WX}$	Gates: $\mathbf{W}_{f/i/o/g x}\mathbf{X} + \mathbf{W}_{f/i/o/g h}\mathbf{h}$ Cell State: $\mathbf{c}(t) = \mathbf{c}(t-1) \odot \mathbf{f}(t) + \mathbf{g}(t) \odot \mathbf{i}(t)$ Hidden State: $\mathbf{h}(t) = tanh(\mathbf{c}(t)) \odot \mathbf{o}(t)$
	Op: `VMA` (`VMM`), `VVA`	Op: `VMM`, `VVA` (`VVM`, `VS`, `VB`)	Op: `VMM`, `VVA` (`VB`)	Op: `VMM`, `VVM`, `VVA`, `VB`
NTU	Leakage: $\tau \frac{dV_i(t)}{dt} = -[V_i(t) - V_{r_1}] + V_{\sum}$ Thresh_comp V_update | (Spike_fire & V_reset)	Parts of spike_rate: $Norm(\cdot)$	Conv: $\varphi(b_n + \mathbf{FM}_n^{\sum})$ (Overlap Copy) Pool: Max/Average (Overlap Copy) FC: $\varphi(b_i + x_i^{\sum})$	Gates $f/i/o$: $sigmoid(\mathbf{b} + (\mathbf{x} \,\&\, \mathbf{h})^{\sum})$ Gate g: $tanh(\mathbf{b} + (\mathbf{x} \,\&\, \mathbf{h})^{\sum})$ Cell State: $tanh(\cdot)$
	Op: `B_L`, `Spi_Gen`, `Out_Trans`	Op: `Act_Fun`, `Out_Trans` (`Lat_Acc`)	Op: `B_L`, `Act_Fun`, `Out_Trans` (`Pooling`, `Out_Copy`)	Op: `B_L`, `Act_Fun`, `Out_Trans`
NC[1]	Feedforward/Recurrent	Recurrent (Feedforward)	Feedforward (Recurrent)	Recurrent (Feedforward)
Mode	SNN (spiking) Mode	ANN (non-spiking) Mode		

[1]The "feedforward/recurrent" outside the parentheses means explicit connections of original networks, while the one inside the parentheses means implicit connections between UFCs introduced during mapping networks onto chip.

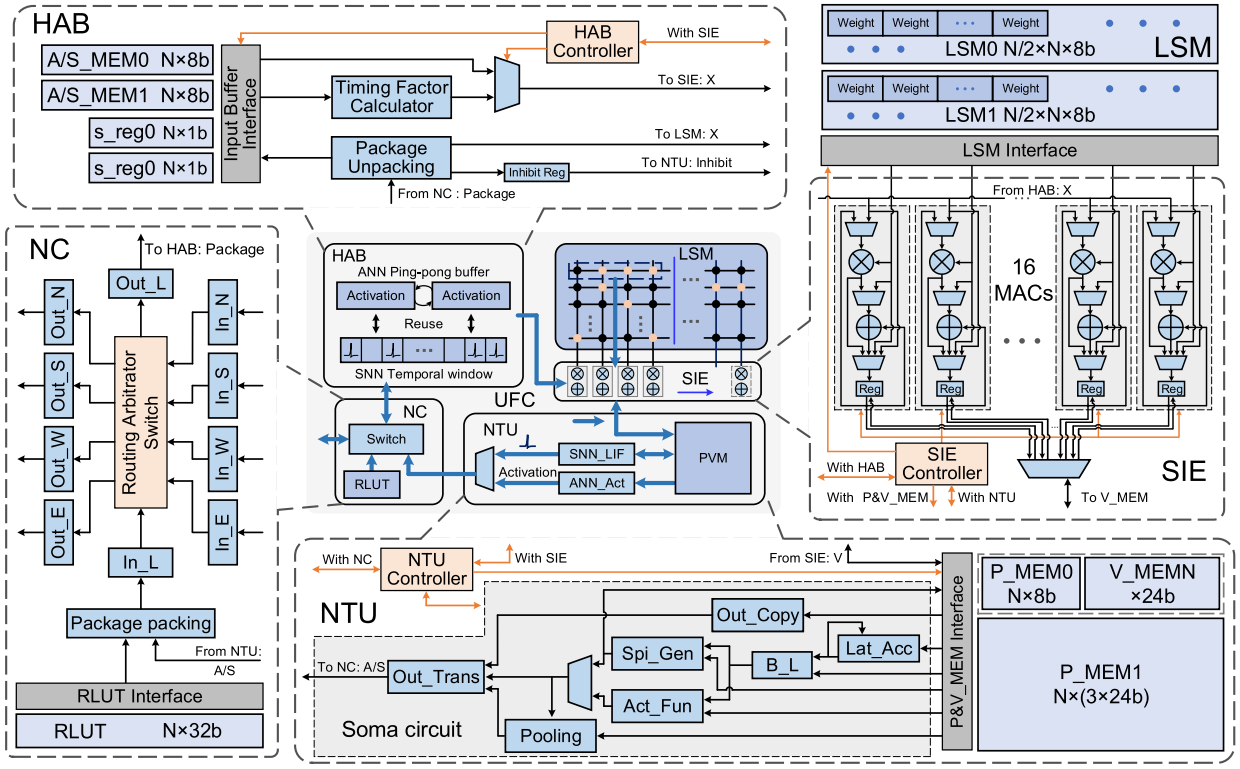


Fig. 2. Overview of the UFC architecture. N is the number of neurons in one single UFC.

calculators [multipliers and accumulators (MACs)] although they might have different processing manner.

c) Unified routing infrastructure (NC): The interconnections of UFCs share the same routing infrastructure to transfer both SNN and ANN packets in a unified format. Based on the needs, pre-neuron NTU can package the output into either a spiking or a non-spiking packet according to the NTU configuration, and the post-neuron HAB parses the routing packet into either the spiking or the non-spiking format based on its HAB configuration.

3) Overview of UFC Architecture: Fig. 2 shows the overall architecture of UFC, whose dataflow follows "remote UFCs or local UFC⇒NC⇒HAB/LSM⇒SIE⇒NTU⇒NC⇒local UFC or remote UFCs." In this section, we assume that the number of neurons in one single UFC is N. Specifically, HAB acts as a data buffer to provide inputs for SIE and also buffers

outputs from NC (generated in NTU). The buffered data are a spike in SNNs, spike rate and membrane potential in BDyNNs, activation in MLP/CNNs, gate output, and cell/hidden state in LSTM. LSM locally stores the synaptic weights, which is placed near to the SIE compute for better memory locality. SIE is an integration engine occupying most MACs to support the operations in Table II. NTU is another compute unit to support the neuronal transformations in Table II. Besides the intra-core compute and data movement, inter-core communication is wired by NCs that are configurable to support arbitrary connection topology.

4) Data Precision Selection: The selection of data precision is mainly determined by the application accuracy. We focus on the inference rather than a training problem, which lowers the precision requirement. In TrueNorth [30] or Loihi [80] for SNNs, the weights adopt 9-bit integers with extra data clustering or a variable bit-width between 1 and 9 bits, respectively. We find that SNNs still work with only 8-bit weights. CNN quantization can reach extremely low bits (e.g., $\leq$8 bits [81] or even 1-2 bits [82]–[84]). RNNs usually need a bit higher precision, e.g., $\geq$4 bits in LSTM [85], [86]. BDyNNs have been evidenced to work well with only 8 bits [79]. With these concerns, we select 8-bit signed integers as the major data precision for activations and weights while representing the intermediate integration results in 24 bits. Actually, many neural network hardwares also adopt this precision level or even lower [30], [35], [37], [42], [46], [75], [80], [87], [88]. Note that we use fixed precision for simplicity in this article as we focus on the functionality of the hybrid paradigm. Flexible precision support is of course allowed on our architecture by slightly modifying the UFC design.

5) HAB: There are two cases for the inputs of HAB received from NC: 8-bit continuous activations in ANN mode or 1-bit binary spikes in SNN mode. As shown in Fig. 3, two SRAM chunks, i.e., A/S_MEM0 and A/S_MEM1, form the main body of HAB. Specifically, A and S denote the activation in ANN mode and the spike in SNN mode, respectively. There are three cases for the outputs of HAB sent to SIE.

a) In ANN mode, the outputs are 8-bit continuous activations, where A/S_MEM0 and A/S_MEM1 work as two ping-pong buffers for SIE read and NC write, respectively. During each time phase, if SIE is enabled, the ping-pong pointer flips once the SIE processing starts.

b) In SNN mode with $T_w = 1$, two extra register vectors of $(N \times 1\text{b})\times 2$ are used to buffer the most recent spikes received at the last time phase and also work in the ping-pong manner. The most recent spike vector [i.e., $\mathbf{s}(t)$] is directly sent to SIE in this case.

c) In SNN mode with $1 < T_w \leq 16$, A/S_MEM0 and A/S_MEM1 are merged to be a whole chunk to contain the spike pattern within a historical temporal window (i.e., T_w time phases), which is updated at each time phase. In this case, a timing factor calculator (TFC) is enabled to convert the T_w-phase spike train at each row into a scalar value through a weighted sum considering the timing decaying effect of $K(\cdot)$ in (1). Thus, the

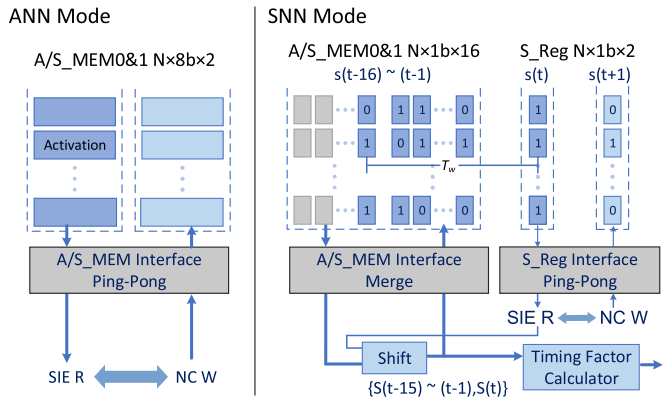


Fig. 3. HAB for buffering input and output data.

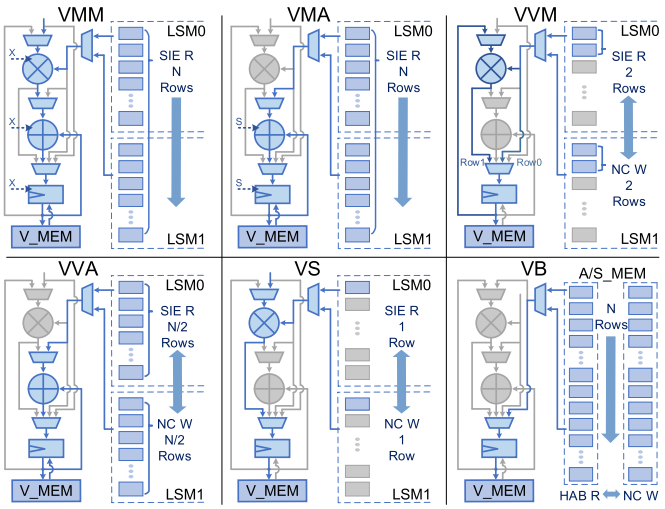


Fig. 4. SIE for the input integration with six vector/matrix operations. "R" and "W" denote read and write, respectively, which indicates the ping-pong manner.

$N \times 16$b spike pattern can be reduced to be $N \times 8$b activations and then be sent to SIE.

6) LSM: LSM is mainly designed for storing the synaptic weights, which is close to the compute block (e.g., SIE) for good access locality. Each UFC has its own LSM without coherence that enhances the parallelism. LSM is comprised of two SRAM chunks, LSM0 and LSM1, with a size of $N/2 \times N \times 8$b for each. For vector-matrix operations (e.g., `VMA` and `VMM`), they merge as a crossbar of $N \times N \times 8$b. For element-wise vector operations (e.g., `VVA`, `VVM`, and `VS`), LSM0 and LSM1 hold dynamic intermediate results from NC and work as two ping-pong buffers. For the `VB` operation, LSM will be disabled, and SIE bypasses LSM to directly copy data from HAB.

7) SIE: SIE dominates the integration for accumulating weighted inputs or processing intermediate results. As shown in Fig. 4, it mainly consists of MACs, as well as multiplexers. By using them in different ways, six vector/matrix operations (see Table II) are supported. There are 16 8-bit multipliers and 16 24-bit accumulators. For the N columns in the LSM crossbar, the calculation is divided into multiple groups with 16 columns for each group. In the `VMM` operation, at each

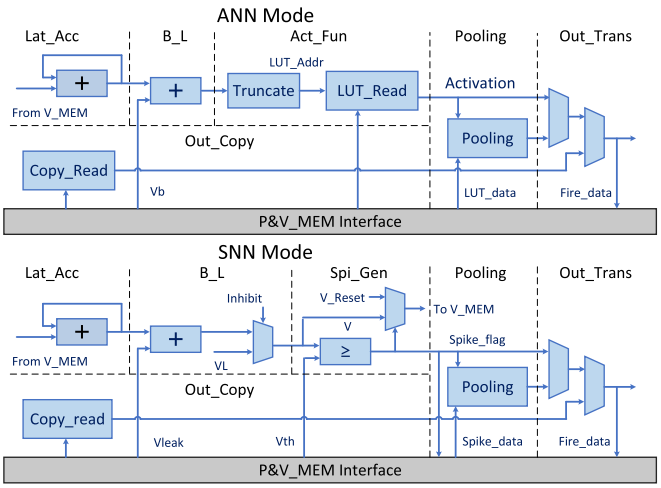


Fig. 5. NTU for neuronal transformations.

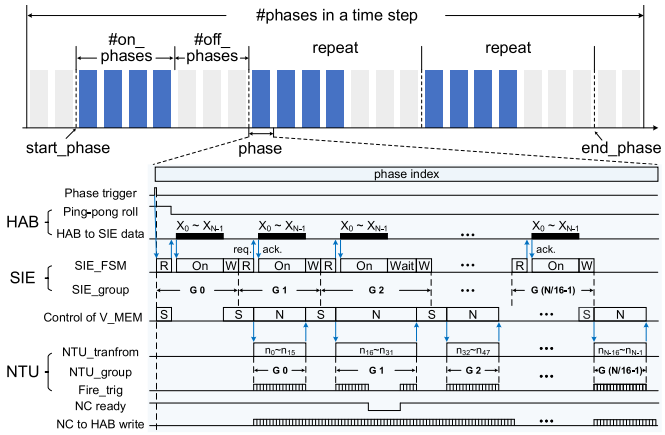


Fig. 6. UFC timing schedule.

cycle, SIE reads one input from HAB, reads 16 weights from LSM at once, and then concurrently executes 16 MACs that share the same HAB input. In the `VMA` operation, the multiplication is disabled, which could save energy with negligible area waste since SIE only occupies 7% area of UFC (to be shown later). In the `VVM` operation, SIE ignores HAB and reads dynamic intermediate data rather than static weights from LSM and executes a variable-variable multiplication. Considering typical models, `VVM` supports only two-vector multiplication. `VVA` bypasses the multipliers, and it supports a reduction of maximum $N/2$ vectors. The `VS` operation requires only one input from HAB and one row of dynamic data from LSM. LSM is disabled in VB operation, and SIE directly copies data from HAB.

8) NTU: To cover most neuronal transformations, we build seven normal transformations and several other retained transformations in NTU, as given in Table II and Fig. 5. All soma parameters (and a few intermediate states) are stored in P_MEM. NTU and SIE share V_MEM, from which NTU reads the integrated results written by SIE. In ANN mode, the dataflow follows the path of "`B_L`⇒`Act_Fun`⇒`Out_Trans`." A reconfigurable lookup table (LUT) is used to implement an arbitrary activation function. In the SNN mode, the dataflow path changes to "`B_L`⇒`Spi_Gen`⇒`Out_Trans`." Three other transformations, i.e., `Pooling`, `Lat_Acc`, and `Out_Copy` shared by both ANN and SNN modes, might be enabled in some scenarios according to actual needs. When processing the pooling layers in CNNs, `Pooling` will be enabled. `Lat_Acc` is designed for lateral accumulation of the integration results across continuous neurons to form a "big neuron." This helps to extend the `VVA` vector, which reduces along the row dimension to column dimension. For example, BDyNNs require this operation if the global normalization in (2) is used. `Out_Copy` is dedicated for the copy of neuronal output (e.g., overlapped neuron in CNNs to be described in Section IV-C). We also design several retained transformations (e.g., threshold adaption, leakage adaption, and probabilistic firing) to allow more complex SNN models.

9) NC: The inter-core communication is completed by NC, as shown in the left bottom of Fig. 2. NC has an $N \times 32$b routing LUT (RLUT) that stores the addresses of destination UFCs and their intra-UFC memory cells connected to the neurons in the current UFC. RLUT is reconfigurable to support arbitrary connection topology. Each NC has five communication channels: local, eastern, western, southern, and northern.

10) Timing Schedule: There are two levels of temporal execution unit in our design: time step and time phase that work synergistically to support a flexible operating pattern, as depicted in the top of Fig. 6. The time step is a coarse-grain temporal unit. The execution pattern of UFC is identical across time steps after the initial configuration, which limits the variety of execution patterns. By adding a fine-grain time phase (each time step has multiple time phases and the number of phases is configurable), the execution pattern can be diversified in the temporal dimension. For example, by configuring the timing registers, i.e., start_phase, end_phase, #on_phases, and #off_phase, UFC can operate in a flexible pattern with different on/off phase distribution (on: SIE & NTU enabled; off: SIE & NTU disabled) within each time step. This flexibility is useful in some application scenarios, such as the semi-folded mapping (to be explained in Section IV-C).

The bottom of Fig. 6 illustrates the timing schedule within each time phase, taking the `VMM` operation in SIE as an example. First, for each neuron group, SIE with 16 MACs processes 16 synapse columns in parallel but N input rows sequentially. Accordingly, the total number of cycles required for SIE integration is N^2/#MACs. Second, SIE and NTU work in the pipeline at the group grain and access V_MEM alternately. When NTU processes the neuronal transformation of the kth group G(k) and occupies the access permission of V_MEM, SIE can process the integration of next group. SIE must wait for NTU to release the access permission of V_MEM before writing the integrated results into V_MEM. Usually, this wait time is zero because NTU is faster than SIE, while it might be long if routing congestion occurs. With this group-by-group processing, the transmission of routing packets can be more uniform in the temporal dimension to alleviate the communication burden.

11) Performance Optimization: We use several techniques to optimize the running performance, as depicted in Fig. 7.

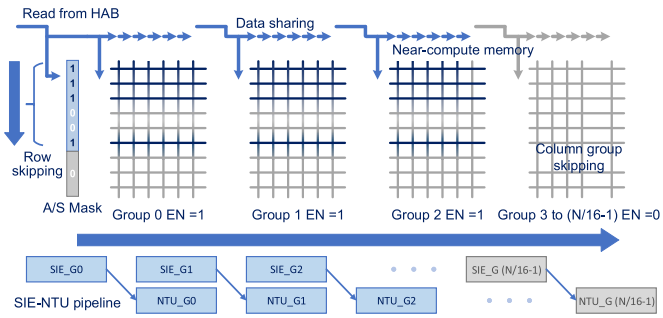


Fig. 7. Performance optimization techniques.

a) Near-memory compute: Each UFC has its own memory hierarchy, and it is allocated near to the compute logic for fast access.

b) Input data sharing: The SIE/NTU compute is divided into multiple groups of neurons (along with the LSM columns), and the HAB activation/spike is shared by multipliers within each group for fewer data accesses, different from the inefficient column-by-column processing.

c) Row skipping: If the activation/spike from HAB is zero, the following LSM accesses and SIE operations will be skipped.

d) Column group skipping: The compute and memory accesses in unutilized column groups (e.g., performing VMM with a non-square matrix, or intentionally emptying out some columns to alleviate the routing burden in VVA UFCs to be shown in Section IV-C) will be skipped to save cycles and power consumption.

e) Inter-group SIE-NTU pipeline is used for higher processing parallelism and uniform routing transmission. Besides, ANN and SNN modes share resources to a great extent, including the HAB memory, LSM, V_MEM, P_MEM, the whole SIE, and parts of NTU, as well as the communication infrastructure, which dramatically reduces the area cost for the fusion of two distinct paradigms.

B. Homogeneous or Heterogeneous Many-Core Network

1) P2P and AMC Routing Strategies: UFC is a small self-contained neural network with $N \times N$ programmable connections. It is easy to construct a larger network by wiring many UFCs together through NCs, as shown in Fig. 8. The architecture is hierarchically scalable, i.e., UFC⇒chip⇒board⇒system. The point-to-point (P2P) routing on the XY mesh is a typical strategy [30], [89]. A routing packet of spike or activation starts from the source neuron to destination neurons through two steps: 1) move to a destination memory cell in an intra- or inter-chip UFC and 2) fan-out to destination neurons when the computation (e.g., VMA, VMM, and VS operation) starts in that UFC. The input data sharing in each UFC saves long-distance connections, and the routing table is reconfigurable for the flexible support of arbitrary topology. A synchronous clock is required within each UFC, while an asynchronous communication with handshaking is enough for the inter-core communication. A global phase synchronization indicating a complete round

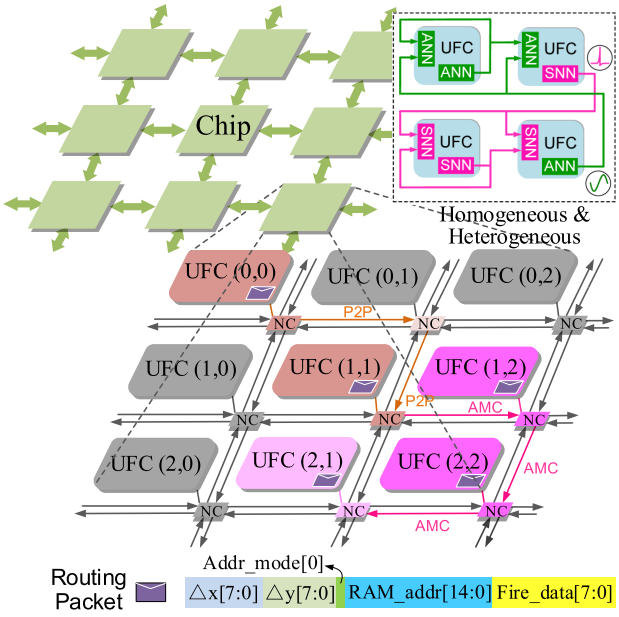


Fig. 8. Scalable routing infrastructure.

of computation and communication is used to ensure a correct timing schedule facing many UFCs.

On the 2-D-mesh UFC plane, we use the coordinate format of (y, x) to represent the UFC location, in which y and x denote the vertical and horizontal locations, respectively. The routing packet has a width of 40 bits, including $Fire\_data$ (8 bits), relative addresses of Δx and Δy (8 bits for each), $Addr\_mode$ (1 bit), and RAM_addr (15 bits). Here, the address mode guides where to put the data in the destination UFC (0–HAB; 1–LSM). When $Addr\_mode = 0$, the lower 8 bits in RAM_addr represent the row index in HAB, but the higher 7 bits are ignored, while when $Addr\_mode = 1$, the higher 7 bits and lower 8 bits record the row index and the column index in LSM, respectively. For the VMM operation in SNN mode, one Inhibit bit indicating the inhibition signal is included in the higher 7 bits of RAM_addr. Besides the P2P routing (e.g., UFC(0, 0)⇒UFC(1, 1) in Fig. 8), we propose an adjacent multicast (AMC) routing to break the fan-out limitation. Because of the restriction on the RLUT capacity, each neuron is only allowed to connect one destination memory cell. Although Out_Copy operation can be leveraged to copy the routing packets for multicast, it decreases the utilization of computation resources, and the number of copies is limited within UFC. By contrast, AMC can address this issue efficiently, which relies on two AMC registers in each UFC, i.e., (AMC_Δy, AMC_Δx). If these registers are configured with non-zero values, all received packets will be indiscriminately copied and sent to the destination UFC indicated by (AMC_Δy and AMC_Δx). For example, in Fig. 8, we initialize the AMC registers of UFC(1, 1), UFC(1, 2), and UFC(2, 2) to be (AMC_Δy=0, AMC_Δx=1), (AMC_Δy=1, AMC_Δx=0), and (AMC_Δy=0, AMC_Δx=−1), respectively. In this way, UFC(1, 2), UFC(2, 2), and UFC(2, 1) are able to share the same packet received by UFC(1, 1). In other words, UFC(0, 0) finally multicasts its output packets to

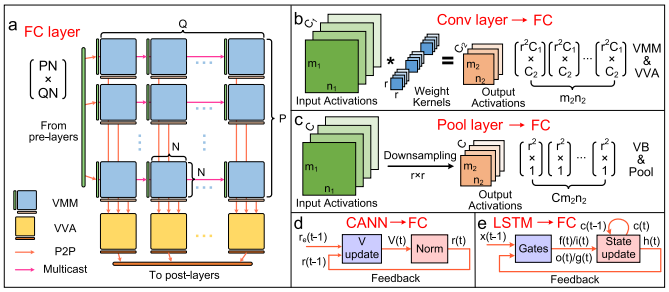


Fig. 9. Throughput-aware fully unfolded mapping.

multiple UFCs. Theoretically, there is no limitation on the number of destination UFCs via this relay-like AMC routing.

In contrast to the intra-chip communication, the packet routing between the chips has to consider the limited number of I/Os and slow off-chip signal transmission. First, merge-split circuits are required for inter-chip communication, leading to inefficient parallel-serial conversion. Second, compared with the ultra-fast signal transmission within chip, the inter-chip data movement heavily depends on the dedicated interface to pursue a high speed. In this article, we use a medium-speed inter-chip interface [low-voltage differential signaling (LVDS) [90]] with four bidirectional lanes on each chip side for prototype verification. In the future, we will replace it with high-speed ones [91].

2) Homogeneous/Heterogeneous Scalability: Our routing infrastructure is compatible between ANN and SNN modes, which supports both the homogeneous and heterogeneous scalability. As mentioned in Section IV-A, HAB and NTU can be independently configured into either ANN or SNN mode, and the network works correctly if the pre-NTU and post-HAB are in the same mode (see Fig. 8). If we configure all the HABs and NTUs into the same mode, the many-core network forms a homogeneous network for single-paradigm workloads, while if they are configured into different modes, a heterogeneous network with ANN-SNN hybrid paradigm can be constructed.

C. Streaming Mapping Schemes

The independent memory hierarchy in each UFC brings challenges in network deployment. In this section, we propose two types of streaming mapping scheme: 1) throughput-aware fully unfolded mapping and 2) resource-aware semi-folded mapping. They allow a good tradeoff between the processing throughput and resource overhead.

1) Throughput-Aware Fully Unfolded Mapping: As illustrated in Fig. 9, the fully unfolded mapping converts all layers to an FC layer and then splits it to sub-FC layers. Fig. 9(a) presents the partition of a large FC layer with $pn \times QN$ size. Here, we assume the synapse crossbar size of each UFC is $N \times N$, so we need a $P \times Q$-size UFC array to complete the VMM operations. The VMM UFCs on each row share inputs through the AMC routing. Each output from VMM UFCs just holds a partial sum of the final activations; therefore, reduced UFCs with the VVA operation are additionally required for the accumulation of partial results. Specifically, VMM UFCs send partial activations to reduced UFCs via the P2P routing, and then, the VVA operation there accumulates them to produce the complete output activations.

Next, we explain how to transform other layers to the FC layer. In fact, Conv and pooling layers can be viewed as many small FC layers across sliding windows, so it is easy to unfold them, as shown in Fig. 9(b) and (c), respectively. Note that the overlapped activations should be copied using extra UFCs with Out_Copy. As for BDyNNs in Fig. 9(d), we first convert the differential format of (2) to its difference format of "$[\mathbf{r}(t-1),\ \mathbf{r_e}(t)] \Rightarrow \mathbf{V}(t) \Rightarrow \mathbf{r}(t)$," similar to the iterative temporal dynamics of LSTM in Fig. 9(e). For these two models, VMM UFCs are needed for the V_update in BDyNNs and gate calculations in LSTM. Besides, extra UFCs are used to execute the vector–vector operations, e.g., Norm$(\cdot)$ for the spike rate generation in BDyNNs, cell/hidden state update in LSTM, and the possible reduce operations. Different from the previous layers, recurrent routing paths are required in BDyNNs and LSTM.

2) Resource-Aware Semi-Folded Mapping: To save resource overhead, we additionally propose a resource-aware semi-folded mapping scheme. It targets the Conv layer where the number of neurons and synapses will greatly increase if we fully unfold it. This scheme is not suitable for SNNs because the membrane potential of spiking neurons has temporal dependence so that UFCs cannot be reused across neurons. Before introducing details, please recall the smart design feature (see Fig. 6) that SIE and NTU can be flexibly switched ON or OFF at each time phase. Briefly speaking, the purpose of the semi-folded mapping is twofold.

a) To exploit the weight reuse of convolution, it assigns UFCs for generating only one row of output FMs and reuses them along the row dimension to save resource overhead.

b) It unfolds the synapses of output neurons along the column dimension using weight copy to enhance the parallelism. In other words, this mapping folds resources in the row dimension and unfolds resources in the column dimension, so it is referred to as "semi-folded."

Fig. 10 shows an example with 2×2 Conv kernel. The three-row–five-column input FM generates an output FM with two rows and four columns (padding 0 and stride 1), as shown in Fig. 10(a). Actually, generating each output row only requires two input rows. The weight kernel is shared in the row dimension, which enables the row-wise reuse of UFCs. Note that the FM column size, in reality, might be quite large (e.g., 224 for ImageNet data set [92]) that the number of fan-ins of single UFC will be exceeded even if we only handle limited rows. In those cases, a column-wise slicing method is proposed, where the FMs can be split into multiple slices with independent UFCs for concurrent processing. Every two adjacent slices might have several overlapped columns that need to be copied through Out_Copy in the pre-layer. In order to realize the row-wise reuse, x is injected row by row at each time phase with an extra UFC shown in Fig. 10(b) to buffer it in the same row-by-row manner. The row buffer is implemented by the intra-core P2P routing with interleaved feedback connections. The following Conv UFCs share the inputs with the buffer UFC through the AMC routing. SIE of

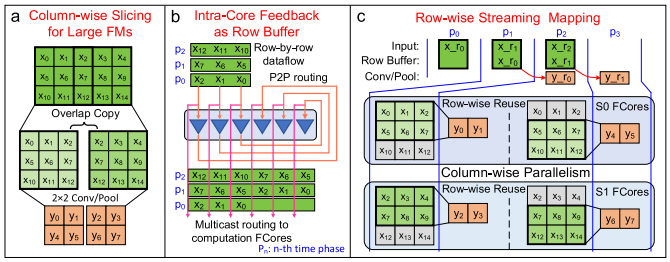


Fig. 10. Resource-aware semi-folded mapping.

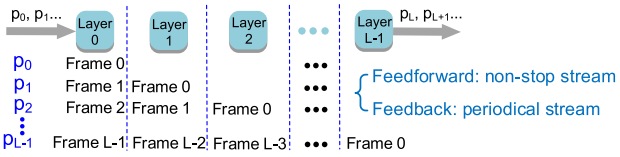


Fig. 11. Inter-core streaming pipeline.

the buffer UFCs works in VB mode; therefore, LSM there can be disabled. Once the required input rows are completely organized, the Conv UFCs start their SIE and NTU at next time phase to generate one row of y. The timing configuration (on/off phases of SIE and NTU) is shown in Fig. 10(c). The entire dataflow behaves as a row-wise stream, which can accommodate FMs unlimited rows.

For the case of multiple FMs, we use more UFCs to buffer the input rows and generate the output row across all FMs. Extra VVA UFCs are usually needed to accumulate the partial sums of activations calculated from only a part of input FMs. The mapping of pooling layers is similar to the Conv layers, while it uses VB operation in SIE and Pooling transformation in NTU. The reduced UFCs with VVA operation can be removed due to the one-to-one FM correspondence between inputs and outputs.

3) Inter-Core Pipeline: It is easy to realize the execution pipeline across UFCs with our streaming mapping schemes. As depicted in Fig. 11, each layer is allocated in independent UFCs that can be processed within one time phase for simplicity (actually, more phases are required for the accumulation of partial activations, row buffering, or several sub-execution steps in one recurrent iteration). Once each layer generates its outputs and sends them to the next layer, it is able to process the next frame. In the case of only feedforward routing, the inter-core pipeline behaves as a non-stop stream, which enables ultrahigh throughput (e.g., one time phase per frame). In the case of feedback routing, the entire execution presents as a periodical stream.

4) Discussion on Existing Mapping Schemes: Conventional neuromorphic chips for only SNNs use the fully unfolded mapping with massive parallel cores [30], while the deep learning accelerators for only ANNs usually select the folded mapping in different formats to reuse the PE array [44], [46], [47]. A recent chip [75] combines the above-mentioned two types of architecture, as mentioned in Section I. At the inter-core/inter-chip level, its mapping is similar to ours using a decentralized manner with pinned weights, spatially propagated activations, inter-PE activation multicast, and cross-PE accumulation of partial integration results, while at the intra-core level, its resources are temporally reused in a fashion different from our schemes proposed earlier. There is no absolute answer that the mapping scheme is better, and it is actually a tradeoff between the processing speed and resource overhead.

TABLE IV
GENERAL COMPARISON WITH EXISTING PLATFORMS

Platform	TrueNorth [30]	EIE [42]	DRISA [37]	Eyeriss [44]	DNA [45]	ESE [43]	Tianjic
Type	ASIC	ASIC	PIM	ASIC	ASIC	FPGA	ASIC
Model	SNNs	MLP	CNNs	CNNs (Conv)	CNNs	LSTM	Unified
Bitwidth	9W[a]-1S	4W-4A	1W-8A	16W-16A	16W-16A	12W-16A	8W-8A/1S
On-chip	SRAM (53.5 MB)	SRAM (10.125 MB)	DRAM	SRAM (181.5 KB)	SRAM (280 KB)	SRAM (4.7 MB)	SRAM (3.3 MB)
Off-chip	N. A.	N. A.	N. A.	DRAM	DRAM	DRAM	N. A.
Technology	28 nm	45 nm	22 nm	65 nm	65 nm	22 nm	28 nm
Clock (MHz)	Async	800	125	100-250	200	200	300
Area (mm^2)	430	40.8	258.2	16	16	N. A.	14.44[b]
Power (W)	0.063-0.3	0.59	103.8	0.2-0.3	0.48	41	0.95
GOPS/W[c]	N.A.	w: 174[d]	w: 15.9	w/o[e]: 246	w/o: 406 w: 153	w: 6.9	w: 1278
GSyOPS/W[f]	w: 400 [89]	N. A.	N. A.	N. A.	N. A.	N. A.	w: 649
Chip Test	Mixed	Simulated	Simulated	Measured	Simulated	Measured	Mixed

[a] The weight precision here means the data bitwidth itself without considering the data clustering technique. [b] In the real chip, we compressed LSM by using weight sharing mechanism [30], [42] to save the fabrication cost. While in latter experiments, we show the uncompressed results and all the measurement data are accordingly scaled. [c] "GOPS/W" denotes giga operations per second per watt (ANN mode), [d] "w" denotes power consumption considering all on-chip memories and the off-chip memory access (if applicable), [e] "w/o" denotes power consumption without considering the off-chip memory access, and [f] "GSyOPS/W" denotes giga synaptic operations per second per watt (SNN mode).

V. EVALUATION

A. Fabrication and Measurement

We fabricate a prototype chip, i.e., Tianjic, in UMC 28-nm HLP CMOS process for basic functional verification and data collection. Table IV lists the specification summary compared with several existing neuromorphic platforms and deep learning accelerators. Tianjic is the first unified ASIC that covers most neural network models across neuromorphic computing and deep learning. Considering the fabrication cost, we set $N = 256$ in each UFC and integrate only 156 UFCs in one single chip. Fig. 12 shows the chip layout and a multichip board equipped with an Altera Arria 10 GX FPGA. Tianjic requires 5050 clock cycles to complete a round of computation and communication, which reflects the minimum latency of the time phase. This is consistent with our real measurement in Fig. 13 at 300 MHz and 0.85 V. Therefore, the peak throughput is 5.95×10^4 frames/s (F/S). For a single chip, the effective peak power efficiency is 1.28 TOPS/W (ANN mode) and 649 GSyOPS/W (SNN mode), and the internal memory bandwidth could reach >610 GB/s. Note that these basic results are measured using a random rather than specific workload to avoid workload dependence. Our configuration philosophy is to make the computation as full-scale as possible and measure the power consumption at the extreme conditions.

In detail, Fig. 14 presents the area breakdown of one single UFC. LSM occupies 69% area, SIE and NTU together

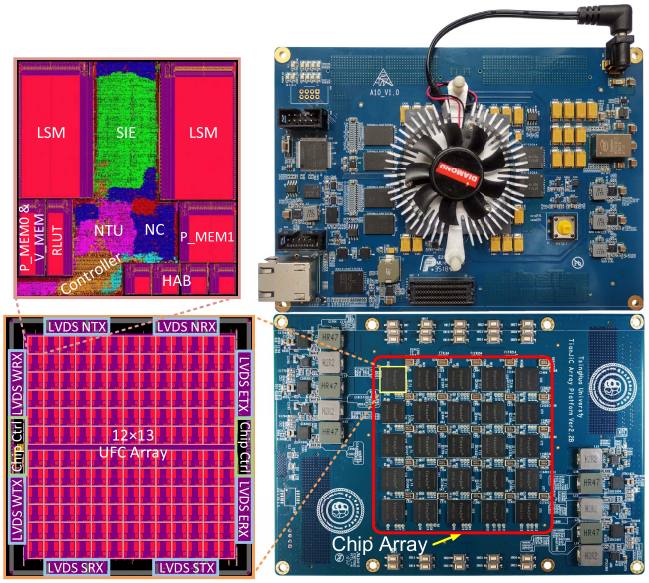


Fig. 12. Chip layout and multichip board.

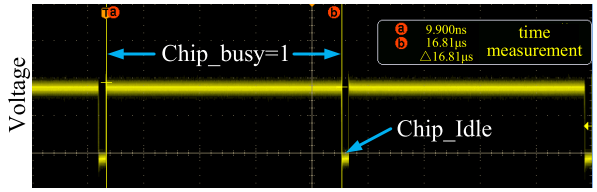


Fig. 13. Measurement of time phase latency.

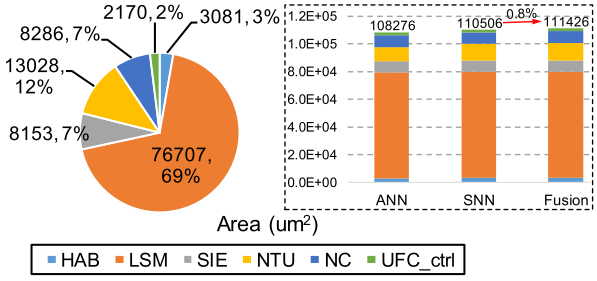


Fig. 14. Area breakdown of a single UFC.

occupy 19%, and HAB, NC, and other control logics occupy the rest 12%. Compared with the single paradigm of ANN or SNN, Tianjic consumes only a slight more area (<1%) by greatly sharing resources between the two modes. Fig. 15 gives the power optimization results under different SIE operation in ANN mode. UFC will shift its power consumption if it works in different SIE operation. The column group skipping technology on unutilized groups can linearly reduce the UFC power consumption except for the VVM, VS, and VB operations where SIE consumes little power. The row skipping technology also saves power consumption when excluding zero inputs under VMA and VMM operations.

B. Experiment Setup

Fig. 16 illustrates our simulation tool chain. First, the quantized training should be completed before network

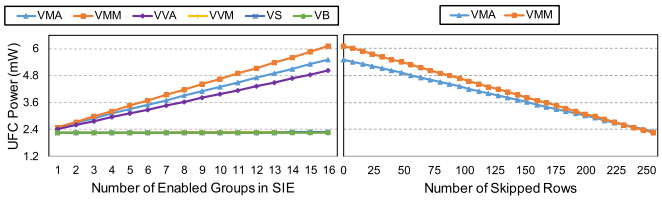


Fig. 15. Power optimization for SIE operations.

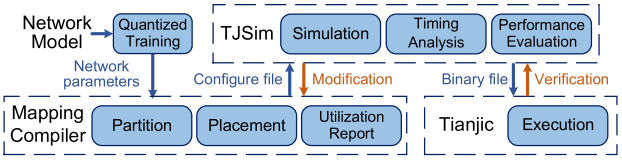


Fig. 16. Illustration of the simulation tool chain.

deployment. Then, we develop a mapping compiler in MATLAB for network partition and resource placement and a C++ cycle-accurate simulator, i.e., TJSim, for hardware simulation that considers all chip-specific constraints. We use a mixed software-hardware methodology [30], [73], [93] for performance evaluation on various workloads, including both the single-paradigm and hybrid-paradigm ones. For small-scale models (e.g., the networks on MNIST [16] or NMNIST [94], and the networks in Figs. 23–26), we report the measurement results, while for those that exceed the resources of Tianjic, we run them in TJSim whose timing can be one-to-one corresponding to the hardware and power consumption is extrapolated based on the basic chip measurement with power scaling.

Note that our focus in this article is to demonstrate the feasibility of the hybrid paradigm on a unified chip rather than a comprehensive performance optimization and physical scaling up. Therefore, we relax two chip constraints from fabrication cost: removing the weight-sharing technique to recover a full degree ($N \times N$) of synapses and ignoring the inter-chip communication overhead by treating it identical with the intra-chip one. In fact, for a large-scale many-chip system, inter-chip communication will become a bottleneck. With this concern, we recommend three solutions for scaling up without compromising the overall performance in the future: 1) integrating more cores into a single chip to accommodate most models within the one chip without inter-chip communication [30]; 2) using high-speed interface between chips to decrease the communication latency [75]; and 3) designing communication-aware network mapping scheme to reduce the cross-chip connections from the software perspective [89].

Besides the general baseline of GPUs (Titan Xp and Tesla V100) with the default FP32 precision, we also compare with several existing specialized platforms, including TrueNorth [30] (SNNs), EIE [42]/DRISA [37] (MLP), DRISA/Eyeriss [44]/DNA [45] (CNNs), and ESE [43] (LSTM). The batch size is set to one in default as suggested by [95] for inference tasks, while for some baseline accelerators that support batching, such as Eyeriss and DNA, our comparisons with them use their batched results. We use Pytorch to measure the running performance of SNNs, MLP, CNNs, and

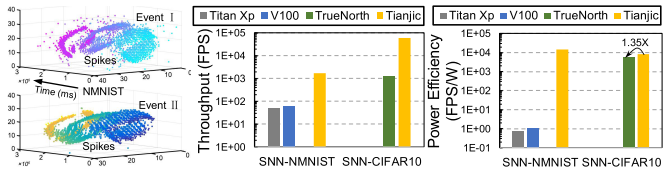


Fig. 17. Evaluation of SNNs.

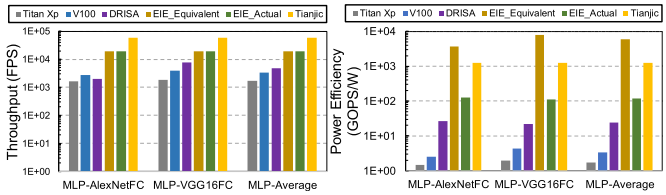


Fig. 18. Evaluation of MLP.

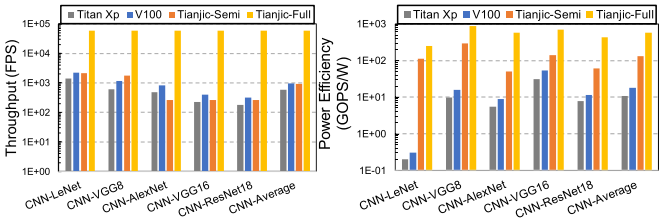


Fig. 19. Evaluation of CNNs (versus GPUs).

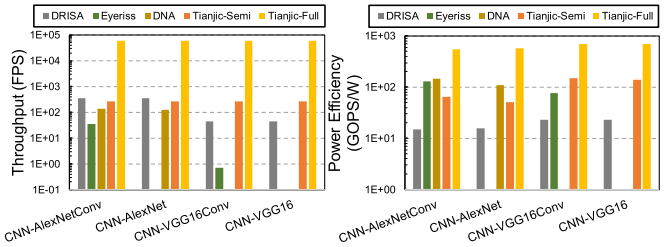


Fig. 20. Evaluation of CNNs (versus accelerators).

BDyNNs on GPUs, while for LSTM, we use DeepBench [96]. We report the average results on Titan Xp and Tesla V100 for simplicity when claiming the improvements over GPUs. In the end, we demonstrate two measured examples to show the potential of our unified architecture with a hybrid paradigm.

C. Benchmarks for Single Paradigm

1) SNNs: Here, we use an MLP on NMNIST data set [94] and a CNN on the spiking version of CIFAR10 data set [97]. On Tianjic, the former is a regular SNN [56] with 98.22% accuracy, and the latter is a variant of ternary ANN [73] with 89.49% accuracy. Because no platforms have reported the execution performance of the former model, we only compare it with GPUs, and we compare TrueNorth on the latter model. As shown in Fig. 17, Tianjic achieves 32× and 48× speedup over GPUs and TrueNorth, respectively. Correspondingly, $1.65 \times 10^4\times$ and 1.35× power efficiency improvements are obtained. Note that compared with the 83.41% accuracy on TrueNorth [73], we can usually achieve higher accuracy at the same network scale due to two reasons.

a) Tianjic uses higher synapse freedom, while TrueNorth suffers a strong weight sharing constraint (the synapses on each neuron only share several weights [30]).

b) Tianjic supports a flexible fan-in extension with the help of ANN/SNN hybrid paradigm (to be mentioned in Section V-D), while TrueNorth has to adopt the connection-partitioned training with severe accuracy loss [73] to avoid the inter-core accumulation of partial membrane potentials in low-precision spikes.

2) MLP: FC layers of AlexNet [98] (9216-4096-1000) and VGG16 [17] (25088-4096-1000) are evaluated, shown in Fig. 18. On average, Tianjic achieves speedup of 23×, 12×, and 3× compared with GPUs, DRISA, and EIE, respectively. The power efficiency improvement is 487× and 51× compared with GPUs and DRISA, respectively. As is well known, EIE compresses the original models to be extremely sparse networks with $< 10\%$ parameters and operations. It is still challenging to design a sparse engine on a unified and flexible architecture. Therefore, for fair comparisons, we list both the equivalent and actual performance of EIE. Here, the actual one represents the physical execution performance, while the equivalent one reflects the virtual execution performance corresponding to the original uncompressed model. Although EIE achieves 4.7× efficiency improvement if considering the equivalent performance, Tianjic outperforms its actual efficiency by 10.5×.

3) CNNs: Fig. 19 shows the CNNs performance compared with GPUs, where we use LeNet-variant [84] (on MNIST data set [16]), VGG8 [84], [99] (on CIFAR10 data set [97]), and AlexNet [98]/VGG16 [17]/ResNet18 [2] (on ImageNet data set [92]) as benchmarks. The top-1 accuracy results on Tianjic are 99.48%, 93.52%, 56.91%, 70.83%, and 68.32%, respectively. Here, we implement both the fully unfolded mapping and the semi-folded mapping, which demonstrates 76× and 1.2× speedup and 39× and 9× power efficiency improvement, respectively, on average. Furthermore, we compare with three existing accelerators, as shown in Fig. 20. DRISA is a DRAM-based processing-in-memory (PIM) architecture; Eyeriss and DNA are based on normal circuits. The reported power values of Eyeriss and DNA do not include the static power of external memory, but Tianjic reports the whole power consumption. The access power of DNA is explicitly given in [45], while the access power of Eyeriss is estimated according to the reported number of accessed bits in [44] and the 70-pJ/bit base energy suggested by [45]. Because Eyeriss and DNA provided their results on only AlexNet and VGG16 (even with only Conv layers), we do not show the average performance. The batch sizes of AlexNet and VGG16 are four and three, respectively, on Eyeriss, while the batch size of AlexNet is 16 on DNA. Specifically, Tianjic-Full achieves much higher improvement in both throughput and power efficiency.

a) Regarding Tianjic-Semi, DRISA achieves a little higher throughput (1.3×) on AlexNet because of the high bandwidth of the PIM architecture; however, it presents lower throughput on VGG16 (5.8×) due to the huge data movement, as well as lower power efficiency in all cases. On average, compared with DRISA,

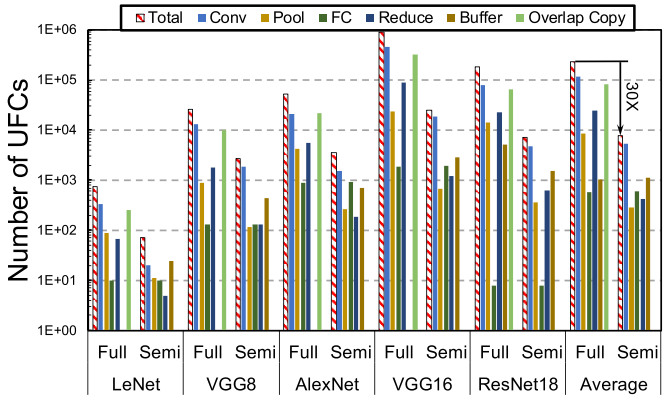


Fig. 21. Resource overhead of two mapping methods.

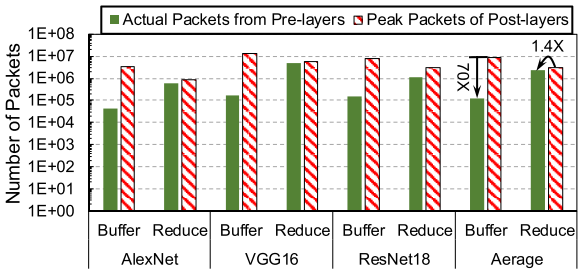


Fig. 22. Analysis of routing capability.

Tianjic-Semi shows 1.3× speedup and 5.3× efficiency improvement.

b) Compared with Eyeriss, Tianjic-Semi achieves 7.6× and 376× speedup on AlexNetConv and VGG16Conv, respectively. Although Eyeriss presents higher power efficiency on AlexNetConv, we outperform it on VGG16Conv (1.9×). The lower efficiency of AlexNet on Tianjic is caused by the large kernel size and channel grouping that decrease the UFC utilization.

c) Compared with DNA, Tianjic-Semi achieves 1.9× and 2.1× speedup on AlexNetConv and AlexNet, respectively. Similar to Eyeriss, DNA presents higher power efficiency on AlexNet, while it did not provide the results on VGG16 where Tianjic-Semi could probably perform better. In summary, even if we consider the semi-folded mapping by sacrificing the performance, Tianjic is still able to achieve better or comparable results.

Fig. 21 shows the distribution of UFC overheads. The semi-folded mapping can save 30× UFC resources on average compared with the fully unfolded one. Furthermore, Fig. 22 presents the overall routing analysis under the semi-folded mapping by considering the actual pre-packets and post-allowable peak packets in all layers. Due to the decentralized communication without the central bottleneck, Tianjic can easily handle the regular activation routing with 70× average redundancy. Here, the routing redundancy means that the allowable peak number of the input routing packets at each layer is larger than the actual number of packets received from the previous layer. This indicates our scalable architecture can support correct running of these benchmarking workloads. However, as the number of FM channels increases, the number of reduced UFCs with VVA operation

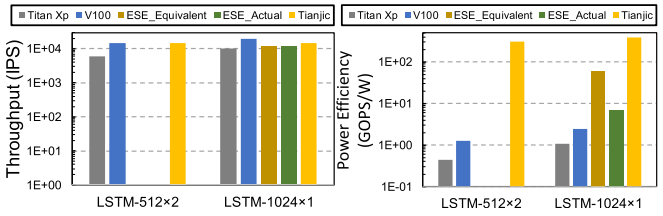


Fig. 23. Evaluation of LSTM.

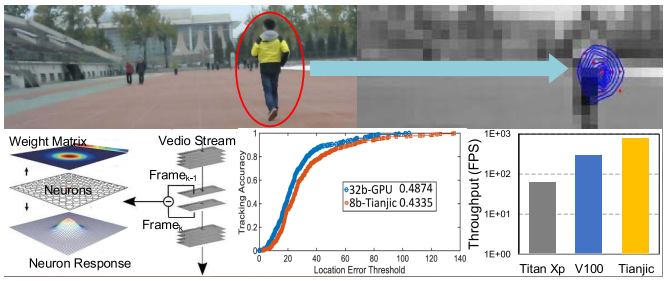


Fig. 24. Evaluation of CANN. The tracking accuracy reflects the percentage of frames whose predicted location of the object correctly lies within a given threshold distance (center location error in pixels) to the ground truth. The reported overall tracking accuracy in the legend is actually a typical value at the threshold of 25 pixels.

for accumulating the partial sums of activations also grows dramatically, which would exacerbate the routing burden. Fortunately, by intentionally using only a part of neurons and LSM columns in the reduced UFCs, the routing overflow can be avoided (actually with 1.4× redundancy) at the cost of slightly more resource overheads.

4) LSTM: Two networks with 1024×1 and 512×2 LSTM hidden layers are used as benchmarks that are tested on WikiText-2 data set [100] with 116.92 perplexity/word (P/W) and on Tiny-Shakespeare data set [101] with 10.06 P/W, respectively. As shown in Fig. 23, on average, 1.2× speedup and 268× efficiency improvement are achieved compared with GPUs. As for ESE with only one hidden layer reported, we also present both the actual and effective performance, such as in the EIE evaluation, because it also leverages a similar extreme compression technique. The throughputs of ESE and Tianjic are nearly the same, while Tianjic achieves better power efficiency compared with both the effective and actual efficiencies of ESE (6.4× and 57×, respectively).

5) BDyNNs: Here, we select a neuroscience-inspired tracking model (continuous attractor neural network (CANN) [15]) for BDyNNs evaluation, as shown in Fig. 24. It is a fully recurrent network with 30×56 neurons that form a 2-D plane to receive external video frames (resized differential signals from a customized human tracking video data set [56]). The synaptic weights are configured as a Gaussian bump, where the weight is determined by the neuronal distance (closer distance → larger weight). The spike rate, generated from the global nonlinear normalization of membrane potential, produces a response bump reflecting the object position. Tianjic achieves a comparable tracking accuracy compared with the GPU version with 4.4× speedup. The tracking accuracy, here, is a typical evaluation methodology to measure the overall tracking performance in a video with multiple frames [102].

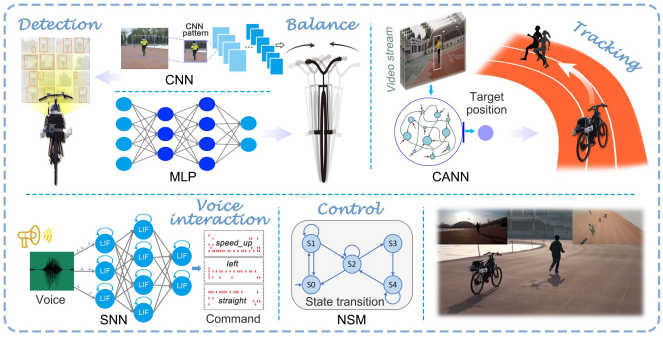


Fig. 25. Multimodal unmanned bicycle.

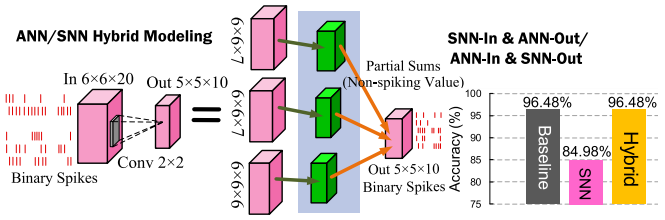


Fig. 26. Example of ANN/SNN hybrid modeling.

D. Examples for Hybrid Paradigm

In order to demonstrate the potential of our unified architecture, we provide two small-scale examples including a multimodal bicycle system and an ANN/SNN hybrid neural network that has been validated on the real chip. Although the examples have been described in [76], we also give them here to showcase the hybrid-paradigm applications of Tianjic, thus presenting our chip in an integrity manner. Because the focus of this article is to disclose the design and evaluation details, we only, briefly, summarize the key points as follows.

1) Multimodal System: We develop a multimodal unmanned bicycle that uses an SNN for voice command recognition, a CNN for object detection, a CANN for object tracking, an MLP for balance control, and a neural state machine (NSM) for decision making, as illustrated in Fig. 25. The SNN recognizes the voice commands from human, the CNN receives resized images from the camera and detects the initial location, and the CANN continuously tracks the object based on the initial location detected by the CNN. The MLP receives sensor inputs and the target inclination angle reflected by the CANN and then generates the target rotation angle for the handlebar motor controller to maintain balance. Moreover, an SNN-based NSM plays the role of decision-maker to integrate the networks in the system. The UFCs between the NSM and the MLP work in a hybrid paradigm with spiking inputs and non-spiking outputs, which contain trajectory patterns in the LUT of NTU to convert spiking action signals into real-valued target inclination angle sequences. All these distinct-paradigm models can be put onto one single Tianjic chip, and then, a real-time autonomous tracking without human intervention is realized.

2) ANN-SNN Hybrid Modeling: Another example in Fig. 26 shows how ANN helps to build large-scale SNNs without accuracy loss. Here, we use a network with the structure of "Input-20C3-AP2-20C2-AP2-10C2-10" on the spiking version of MNIST data set, where "C" denotes Conv and "AP" is average pooling. Although the spiking version of MNIST has insufficient temporal components, we still use it here to keep consistent with [76], where a comprehensive measurement and analysis on the ANN-only, SNN-only, and ANN-SNN hybrid models were presented and MNIST was used because the ANN-only model meets difficulty in handling dynamic data sets (e.g., NMNIST). Usually, one neuron receives thousands of inputs in large-scale SNNs, which exceeds the fan-in number of one single UFC and each neuron can output only a partial rather than complete result. This is easy to address in ANN mode by directly accumulating the non-spiking partial membrane potentials in higher precision using the VVA operation, whereas it becomes challenging in the SNN mode because the intermediate UFCs would naturally truncate the partial membrane potentials to low-precision spikes before sending them to reduced UFCs. This forced spike truncation leads to severe precision loss, only reaching 84.98% accuracy (11.5% lower than the GPU baseline). As mentioned earlier, TrueNorth can use connection-partitioned training to avoid this situation but requiring much more neurons to maintain the accuracy [73].

Tianjic can solve this problem well by using the hybrid modeling of ANN and SNN. We configure the intermediate UFCs working in a hybrid paradigm with HAB in SNN mode to receive binary spike inputs and with NTU in ANN mode to generate non-spiking partial membrane potentials and then the reduced UFCs also working in a hybrid paradigm but reversely with HAB in ANN mode to accumulate non-spiking partial membrane potentials and with NTU in SNN mode to generate the final spike outputs. This configuration guarantees an immediate and high-precision accumulation of the partial membrane potentials in SNNs with the help of our hybrid paradigm. In this way, we achieve a lossless accuracy of 96.48% compared with the original SNN baseline on GPU. Please note that here only the last Conv layer is configured into the hybrid paradigm for simplicity.

Our focus in this example is not to explore the SNNs' advantages, while we just want to show that introducing ANN mode into an SNN model indeed helps improve the accuracy without inter-core precision loss. Since this improvement exists for all SNNs no matter which data set, the use of the MNIST data set is acceptable. Besides this example, for the high-accuracy SNNs with an extra ANN classifier equipped at the end of the network [61], Tianjic can easily accommodate the entire model without relying on a heterogeneous system.

VI. Conclusion

To proactively support the promising integration of neuromorphic computing and deep learning, we build a unified model description framework and the corresponding hardware architecture with five building blocks. A set of dendrite operations and neuronal transformations are implemented, wherein several memory/compute optimization methods are incorporated. A decentralized many-core network is connected via a routing infrastructure with homogeneous/heterogeneous scalability. Flexible streaming mapping schemes are further

designed for efficient network deployment, enabling a streaming dataflow and the inter-core pipeline. In this way, a variety of single-paradigm models (e.g., SNNs, BDyNNs, MLP, CNNs, and RNNs) and the hybrid-paradigm models are well supported. A 28-nm prototype chip, i.e., Tianjic, is fabricated with $>$ 610-GB/s internal memory bandwidth. Tianjic presents significant improvement in throughput and power efficiency under the fully unfolded mapping and still achieves comparable results under the semi-folded mapping that can save 30$\times$ resources. Two hybrid examples demonstrate the potential of our unified architecture. This article provides a solution for the exploration of novel neural computing.

REFERENCES

[1] Y. LeCun, "Deep learning hardware: Past, present, and future," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2019, pp. 12–19.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.

[3] W. Xiong *et al.*, "The Microsoft 2016 conversational speech recognition system," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 5255–5259.

[4] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.

[5] D. Silver *et al.*, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017.

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

[7] S. Ghosh-Dastidar and H. Adeli, "Spiking neural networks," *Int. J. Neural Syst.*, vol. 19, no. 4, pp. 295–308, Aug. 2009.

[8] W. Maass, "Noise as a resource for computation and learning in networks of spiking neurons," *Proc. IEEE*, vol. 102, no. 5, pp. 860–880, May 2014.

[9] C. Eliasmith *et al.*, "A large-scale model of the functioning brain," *Science*, vol. 338, no. 6111, pp. 1202–1205, Nov. 2012.

[10] N. Kasabov and E. Capecci, "Spiking neural network methodology for modelling, classification and understanding of EEG spatio-temporal data measuring cognitive processes," *Inf. Sci.*, vol. 294, pp. 565–575, Feb. 2015.

[11] I. E. Ohiorhenuan, F. Mechler, K. P. Purpura, A. M. Schmid, Q. Hu, and J. D. Victor, "Sparse coding and high-order correlations in fine-scale cortical networks," *Nature*, vol. 466, no. 7306, pp. 617–621, Jul. 2010.

[12] P. Knag, J. K. Kim, T. Chen, and Z. Zhang, "A sparse coding neural network ASIC with on-chip learning for feature extraction and encoding," *IEEE J. Solid-State Circuits*, vol. 50, no. 4, pp. 1070–1079, Apr. 2015.

[13] D. Sussillo and L. Abbott, "Generating coherent patterns of activity from chaotic neural networks," *Neuron*, vol. 63, no. 4, pp. 544–557, Aug. 2009.

[14] G. Li *et al.*, "Hierarchical chunking of sequential memory on neuromorphic architecture with reduced synaptic plasticity," *Frontiers Comput. Neurosci.*, vol. 10, p. 136, Dec. 2016.

[15] S. Wu, K. Hamaguchi, and S.-I. Amari, "Dynamics and computation of continuous attractors," *Neural Comput.*, vol. 20, no. 4, pp. 994–1025, Apr. 2008.

[16] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: https://arxiv.org/abs/1409.1556

[18] O. Abdel-Hamid, A.-R. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Trans. Audio, Speech Language Process.*, vol. 22, no. 10, pp. 1533–1545, Oct. 2015.

[19] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 7263–7271.

[20] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," 2015, *arXiv:1506.00019*. [Online]. Available: https://arxiv.org/abs/1506.00019

[21] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[22] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 6645–6649.

[23] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.

[24] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3156–3164.

[25] D. Wang and E. Nyberg, "A long short-term memory model for answer sentence selection in question answering," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguistics 7th Int. Joint Conf. Natural Lang. Process.*, vol. 2, 2015, pp. 707–712.

[26] J. Schemmel, D. Briiderle, A. Griibl, M. Hock, K. Meier, and S. Millner, "A wafer-scale neuromorphic hardware system for large-scale neural modeling," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2010, pp. 1947–1950.

[27] B. V. Benjamin *et al.*, "Neurogrid: A mixed-analog-digital multichip system for large-scale neural simulations," *Proc. IEEE*, vol. 102, no. 5, pp. 699–716, May 2014.

[28] N. Qiao *et al.*, "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128k synapses," *Frontiers Neurosci.*, vol. 9, p. 141, Apr. 2015.

[29] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The SpiNNaker project," *Proc. IEEE*, vol. 102, no. 5, pp. 652–665, May 2014.

[30] P. A. Merolla *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 2014.

[31] L. Shi *et al.*, "Development of a neuromorphic computing system," in *IEDM Tech. Dig.*, Dec. 2015, pp. 4.3.1–4.3.4.

[32] O. Bichler, M. Suri, D. Querlioz, D. Vuillaume, B. Desalvo, and C. Gamrat, "Visual pattern extraction using energy-efficient '2-PCM synapse' neuromorphic architecture," *IEEE Trans. Electron Devices*, vol. 59, no. 8, pp. 2206–2214, Aug. 2012.

[33] D. Garbin *et al.*, "Variability-tolerant convolutional neural network for pattern recognition applications based on OxRAM synapses," in *IEDM Tech. Dig.*, Dec. 2014, pp. 4–28.

[34] Y. Chen *et al.*, "DaDianNao: A machine-learning supercomputer," in *Proc. 47th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2014, pp. 609–622.

[35] P. Chi *et al.*, "PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 27–39.

[36] A. Shafiee *et al.*, "Isaac: A convolutional neural network accelerator with *in-situ* analog arithmetic in crossbars," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 14–26, 2016.

[37] S. Li, D. Niu, K. T. Malladi, H. Zheng, B. Brennan, and Y. Xie, "DRISA: A DRAM-based reconfigurable *in-situ* accelerator," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, 2017, pp. 288–301.

[38] D. Kim, J. Kung, S. Chai, S. Yalamanchili, and S. Mukhopadhyay, "Neurocube: A programmable digital neuromorphic architecture with high-density 3D memory," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 380–392.

[39] J. Qiu *et al.*, "Going deeper with embedded FPGA platform for convolutional neural network," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays (FPGA)*, 2016, pp. 26–35.

[40] B. Reagen *et al.*, "Minerva: Enabling low-power, highly-accurate deep neural network accelerators," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 267–278.

[41] J. Albericio, P. Judd, T. Hetherington, T. Aamodt, N. E. Jerger, and A. Moshovos, "Cnvlutin: Ineffectual-neuron-free deep neural network computing," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 1–13.

[42] S. Han *et al.*, "EIE: Efficient inference engine on compressed deep neural network," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 243–254.

[43] S. Han *et al.*, "ESE: Efficient speech recognition engine with sparse LSTM on FPGA," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays (FPGA)*, 2017, pp. 75–84.

[44] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan. 2017.

[45] F. Tu, S. Yin, P. Ouyang, S. Tang, L. Liu, and S. Wei, "Deep convolutional neural network architecture with reconfigurable computation patterns," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 8, pp. 2220–2233, Aug. 2017.

[46] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. 44th Annu. Int. Symp. Comput. Archit.*, 2017, pp. 1–12.

[47] S. Yin *et al.*, "A high energy efficient reconfigurable hybrid neural network processor for deep learning applications," *IEEE J. Solid-State Circuits*, vol. 53, no. 4, pp. 968–982, Apr. 2018.

[48] A. H. Marblestone, G. Wayne, and K. P. Kording, "Toward an integration of deep learning and neuroscience," *Frontiers Comput. Neurosci.*, vol. 10, p. 94, Sep. 2016.

[49] B. Zhang, L. Shi, and S. Song, "Creating more intelligent robots through brain-inspired computing," *Science*, vol. 354, pp. 4–9, 2016.

[50] S. Ullman, "Using neuroscience to develop artificial intelligence," *Science*, vol. 363, no. 6428, pp. 692–693, Feb. 2019.

[51] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2015, pp. 1–8.

[52] E. Hunsberger and C. Eliasmith, "Spiking deep networks with LIF neurons," 2015, *arXiv:1510.08829*. https://arxiv.org/abs/1510.08829

[53] P. O'Connor and M. Welling, "Deep spiking networks," 2016, *arXiv:1602.08323*. [Online]. Available: https://arxiv.org/abs/1602.08323

[54] J. H. Lee, T. Delbruck, and M. Pfeiffer, "Training deep spiking neural networks using backpropagation," *Frontiers Neurosci.*, vol. 10, p. 508, Nov. 2016.

[55] A. Tavanaei and A. Maida, "BP-STDP: Approximating backpropagation using spike timing dependent plasticity," *Neurocomputing*, vol. 330, pp. 39–47, Feb. 2019.

[56] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Frontiers Neurosci.*, vol. 12, p. 331, May 2018.

[57] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: VGG and residual architectures," *Frontiers Neurosci.*, vol. 13, p. 95, Mar. 2019.

[58] Y. Hu, H. Tang, Y. Wang, and G. Pan, "Spiking deep residual network," 2018, *arXiv:1805.01352*. [Online]. Available: https://arxiv.org/abs/1805.01352

[59] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L. Shi, "Direct training for spiking neural networks: Faster, larger, better," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 1311–1318.

[60] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 787–797.

[61] G. Srinivasan and K. Roy, "Restocnet: Residual stochastic binary convolutional spiking neural network for memory-efficient neuromorphic computing," *Frontiers Neurosci.*, vol. 13, p. 189, Mar. 2019.

[62] J. Sacramento, R. P. Costa, Y. Bengio, and W. Senn, "Dendritic error backpropagation in deep cortical microcircuits," 2017, *arXiv:1801.00062*. [Online]. Available: https://arxiv.org/abs/1801.00062

[63] J. Guerguiev, T. P. Lillicrap, and B. A. Richards, "Towards deep learning with segregated dendrites," *eLife*, vol. 6, Dec. 2017, Art. no. e22901.

[64] T. P. Lillicrap, D. Cownden, D. B. Tweed, and C. J. Akerman, "Random synaptic feedback weights support error backpropagation for deep learning," *Nature Commun.*, vol. 7, no. 1, p. 13276, 2016.

[65] P. R. Roelfsema and A. Holtmaat, "Control of synaptic plasticity in deep cortical networks," *Nature Rev. Neurosci.*, vol. 19, no. 3, pp. 166–180, Mar. 2018.

[66] A. Morcos, D. G. Barrett, N. C. Rabinowitz, and M. Botvinick, "On the importance of single directions for generalization," in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–15.

[67] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3859–3869.

[68] A. Graves *et al.*, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, Oct. 2016.

[69] N. C. Rabinowitz, F. Perbet, H. F. Song, C. Zhang, S. Eslami, and M. Botvinick, "Machine theory of mind," 2018, *arXiv:1802.07740*. [Online]. Available: https://arxiv.org/abs/1802.07740

[70] L. Deng *et al.*, "Rethinking the performance comparison between SNNS and ANNS," *Neural Netw.*, vol. 121, pp. 294–307, Jan. 2020.

[71] P.-Y. Chen, X. Peng, and S. Yu, "NeuroSim: A circuit-level macro model for benchmarking neuro-inspired architectures in online learning," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 12, pp. 3067–3080, Dec. 2018.

[72] Z. Du *et al.*, "Neuromorphic accelerators: A comparison between neuroscience and machine-learning approaches," in *Proc. 48th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Dec. 2015, pp. 494–507.

[73] S. K. Esser *et al.*, "Convolutional networks for fast, energy-efficient neuromorphic computing," *Proc. Nat. Acad. Sci. USA*, vol. 113, no. 41, pp. 11441–11446, 2016.

[74] L. Song, X. Qian, H. Li, and Y. Chen, "PipeLayer: A pipelined ReRAM-based accelerator for deep learning," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2017, pp. 541–552.

[75] B. Zimmer *et al.*, "A 0.11 pJ/Op, 0.32-128 TOPS, scalable multi-chip-module-based deep neural network accelerator with ground-reference signaling in 16 nm," in *Proc. Symp. VLSI Circuits*, Jun. 2019, pp. C300–C301.

[76] J. Pei *et al.*, "Towards artificial general intelligence with hybrid Tianjic chip architecture," *Nature*, vol. 572, no. 7767, pp. 106–111, Aug. 2019.

[77] W. Gerstner, W. M. Kistler, R. Naud, and L. Paninski, *Neuronal Dynamics: From Single Neurons to Networks and Models of Cognition*. Cambridge, U.K.: Cambridge Univ. Press, 2014.

[78] L. Deng *et al.*, "SemiMap: A semi-folded convolution mapping for speed-overhead balance on crossbars," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 39, no. 1, pp. 117–130, Jan. 2020.

[79] L. Deng *et al.*, "Fast object tracking on a many-core neural network chip," *Frontiers neuroscience*, vol. 12, p. 841, Nov. 2018.

[80] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018.

[81] J. L. McKinstry *et al.*, "Discovering low-precision networks close to full-precision networks for efficient embedded inference," 2018, *arXiv:1809.04191*. [Online]. Available: https://arxiv.org/abs/1809.04191

[82] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to $+1$ or -1," 2016, *arXiv:1602.02830*. [Online]. Available: https://arxiv.org/abs/1602.02830

[83] C. Leng, Z. Dou, H. Li, S. Zhu, and R. Jin, "Extremely low bit neural network: Squeeze the last bit out with ADMM," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 3466–3473.

[84] L. Deng, P. Jiao, J. Pei, Z. Wu, and G. Li, "GXNOR-Net: Training deep neural networks with ternary weights and activations without full-precision memory under a unified discretization framework," *Neural Netw.*, vol. 100, pp. 49–58, Apr. 2018.

[85] Q. He *et al.*, "Effective quantization methods for recurrent neural networks," 2016, *arXiv:1611.10176*. [Online]. Available: https://arxiv.org/abs/1611.10176

[86] C. Xu *et al.*, "Alternating multi-bit quantization for recurrent neural networks," 2018, *arXiv:1802.00150*. [Online]. Available: https://arxiv.org/abs/1802.00150

[87] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H.-J. Yoo, "UNPU: A 50.6 tops/w unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2018, pp. 218–220.

[88] Y. Ji, Y. Zhang, W. Chen, and Y. Xie, "Bridge the gap between neural networks and neuromorphic hardware with a neural network compiler," *ACM SIGPLAN Notices*, vol. 53, no. 2, pp. 448–460, 2018.

[89] F. Akopyan *et al.*, "TrueNorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 10, pp. 1537–1557, Oct. 2015.

[90] S. B. Huq and J. Goldie, "An overview of LVDS technology," Nat. Semicond., Santa Clara, CA, USA, Appl. Note 971, 1998, pp. 1–6.

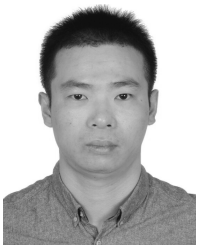
[91] A. Roshan-Zamir, O. Elhadidy, H.-W. Yang, and S. Palermo, "A reconfigurable 16/32 Gb/s dual-mode NRZ/PAM4 SerDes in 65-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 52, no. 9, pp. 2430–2447, Sep. 2017.

[92] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[93] S. Ambrogio *et al.*, "Equivalent-accuracy accelerated neural-network training using analogue memory," *Nature*, vol. 558, no. 7708, pp. 60–67, Jun. 2018.

[94] G. Orchard, A. Jayawant, G. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers Neurosci.*, vol. 9, p. 437, Nov. 2015.

[95] J. Fowers *et al.*, "A configurable cloud-scale DNN processor for real-time AI," in *Proc. ACM/IEEE 45th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2018, pp. 1–14.
[96] S. Narang and G. Diamos. (2017). *Baidu Deepbench*. [Online]. Available: https://github.com/baidu-research/DeepBench
[97] A. Krizhevsky, "Learning multiple layers of features from tiny images," M.S. thesis, Univ. Toronto, Toronto, ON, Canada, 2009.
[98] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[99] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 3123–3131.
[100] S. Merity, C. Xiong, J. Bradbury, and R. Socher, "Pointer sentinel mixture models," 2016, *arXiv:1609.07843*. [Online]. Available: https://arxiv.org/abs/1609.07843
[101] V. Krakovna and F. Doshi-Velez, "Increasing the interpretability of recurrent neural networks using hidden Markov models," 2016, *arXiv:1606.05320*. [Online]. Available: https://arxiv.org/abs/1606.05320
[102] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 2411–2418.



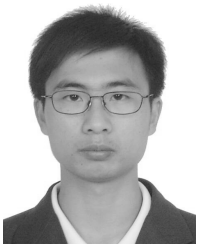
Lei Deng (Member, IEEE) received the B.E. degree from the University of Science and Technology of China, Hefei, China, in 2012, and the Ph.D. degree from Tsinghua University, Beijing, China, in 2017.

He is currently a Post-Doctoral Fellow with the Department of Electrical and Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA, USA. His research interests span the areas of brain-inspired computing, machine learning, neuromorphic chip, computer architecture, tensor analysis, and complex networks. He has authored or coauthored over 40 refereed publications.

Dr. Deng was a PC Member of the International Symposium on Neural Networks (ISNN) 2019. He was a recipient of the MIT Technology Review Innovators Under 35 China 2019. He also serves as a Guest Associate Editor for *Frontiers in Neuroscience* and *Frontiers in Computational Neuroscience* and a reviewer for a number of journals and conferences.

Guanrui Wang received the B.E. degree in electronic engineering from Jilin University, Jilin, China, in 2016. He is currently pursuing the Ph.D. degree in instrumentation science and technology with the Department of Precision Instrument, Tsinghua University, Beijing, China.

His current research interests include neuromorphic chip, computer architecture, and network-on-chip systems.



Guoqi Li (Member, IEEE) received the B.E. degree from the Xi'an University of Technology, Xi'an, China, in 2004, the M.E. degree from Xi'an Jiaotong University, Xi'an, in 2007, and the Ph.D. degree from Nanyang Technological University, Singapore, in 2011.

He was a Scientist with the Data Storage Institute and the Institute of High Performance Computing, Agency for Science, Technology and Research (ASTAR), Singapore from 2011 to 2014. He is currently an Associate Professor with the Center for Brain Inspired Computing Research (CBICR), Tsinghua University, Beijing, China. His current research interests include machine learning, brain-inspired computing, neuromorphic chip, complex systems, and system identification.

Dr. Li is also an Editorial Board Member of *Control and Decision* and a Guest Associate Editor of *Frontiers in Neuroscience* and *Frontiers in Neuromorphic Engineering*. He was a recipient of the 2018 First Class Prize in Science and Technology of the Chinese Institute of Command and Control, the Best Paper Awards (IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS) 2012 and Non-Volatile Memory Technology Symposium (NVMTS) 2015), and the 2018 Excellent Young Talent Award of the Beijing Natural Science Foundation.

Shuangchen Li received the B.S. and M.S. degrees in electrical engineering from Tsinghua University, Beijing, China, in 2011 and 2014, respectively, and the Ph.D. degree from the Department of Electrical and Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA, USA, in 2018.

He was a Post-Doctoral Fellow with the University of California at Santa Barbara. He joined the Alibaba DAMO Academy, Sunnyvale, CA, USA, in 2019. His research interests include memory-centric architecture and domain-specific hardware.

Ling Liang received the B.E. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2015, and the M.S. degree from the University of Southern California, Los Angeles, CA, USA, in 2017. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA, USA.

His current research interests include machine learning security, tensor computing, and computer architecture.

Maohua Zhu received the B.S. and M.S. degrees from the Electrical Engineering Department, Tsinghua University, Beijing, China, in 2007 and 2011, respectively. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA, USA.

He joined AMD Research Beijing, Beijing, as a Research Engineer. His research interest is in the hardware acceleration of machine learning algorithms.

Yujie Wu received the B.E. degree in mathematics and statistics from Lanzhou University, Lanzhou, China, in 2016. He is currently pursuing the Ph.D. degree with the Center for Brain Inspired Computing Research (CBICR), Department of Precision Instrument, Tsinghua University, Beijing, China.

His current research interests include spiking neural networks, neuromorphic devices, and brain-inspired computing.

Zheyu Yang received the B.E. degree in electronic engineering from Sichuan University, Sichuan, China, in 2017. He is currently pursuing the Ph.D. degree with Center for Brain Inspired Computing Research (CBICR), Department of Precision Instrument, Tsinghua University, Beijing, China.

His current research interests include neuromorphic chip, computer vision, and brain-inspired computing.

Zhe Zou received the B.E. degree in electrical engineering and automation from the Beijing Institute of Technology, Beijing, China, in 2016. She is currently pursuing the Ph.D. degree with Center for Brain Inspired Computing Research (CBICR), Department of Precision Instrument, Tsinghua University, Beijing.

Her current research interests include brain-inspired computing, neuromorphic systems, and intelligent robotics.

Jing Pei received the B.E. and M.E. degrees in instrument science and technology from Tsinghua University, Beijing, China, in 1987 and 1989, respectively.

He has been with Tsinghua University since 1990, where he is currently an Associate Professor with the Department of Precision Instrument. He has published over 50 articles in journals and conferences. He holds over 40 invention patents. His research interests include optical information storage systems and optical signal processing. Recently, he focuses on specialized chips for brain-inspired computing based on neuromorphic engineering.

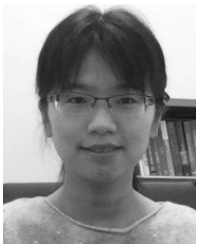
Mr. Pei was a recipient of the Second Prize of the National Invention Award and the National Scientific and Technological Progress Award.

Zhenzhi Wu received the B.S. and Ph.D. degrees from the Beijing Institute of Technology (BIT), Beijing, China, in 2008 and 2015, respectively.

He was a Visiting Researcher with the Division of Computer Engineering, Linköping University, Linköping, Sweden, from 2012 to 2014. He was an Assistant Researcher with Center for Brain Inspired Computing Research (CBICR), Tsinghua University, Beijing, from 2016 to 2018. He is currently the Technical Director of Lynxi Technology Co., Ltd., Beijing. His research interests include high-performance chip design, brain-inspired algorithms, and signal processing.

Xing Hu received the B.S. degree from the Huazhong University of Science and Technology, Wuhan, China, in 2009, and the Ph.D. degree from the University of Chinese Academy of Sciences, Beijing, China, in 2014.

She is currently a Post-Doctoral Fellow with the Department of Electrical and Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA, USA. Her current research interests include emerging memory systems, domain-specific hardware, and machine learning security.



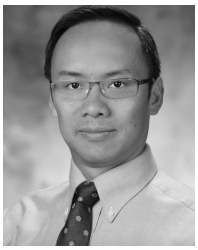
Yufei Ding received the B.S. degree in physics from the University of Science and Technology of China, Hefei, China, in 2009, the M.S. degree from the College of William and Mary, Williamsburg, VA, USA, in 2011, and the Ph.D. degree in computer science from North Carolina State University, Raleigh, NC, USA, in 2017.

She joined the Department of Computer Science, University of California at Santa Barbara, Santa Barbara, CA, USA, as an Assistant Professor, in 2017. Her research interest resides at the intersection of compiler technology and (big) data analytics, with a focus on enabling high-level program optimizations for data analytics and other data-intensive applications.

Dr. Ding was a recipient of the NCSU Computer Science Outstanding Research Award in 2016 and the Computer Science Outstanding Dissertation Award in 2018.

Wei He received the B.S. degree from the Department of Electrical Engineering, Harbin Institute of Technology, Harbin, China, in 2000, and the Ph.D. from the Department of Electrical and Computer Engineering, National University of Singapore, Singapore, in 2010.

He has worked in the semiconductor industry for more than ten years with rich experience on memory fabrication, testing, and design, including resistive memory, phase-change memory, flash memory, and so on. Since 2012, he has been working on neuromorphic devices and circuits at the Agency for Science, Technology and Research (ASTAR), Singapore, and Tsinghua University, Beijing, China. He is currently with Lynxi Techonolgies Co., Ltd., Beijing.



Yuan Xie (Fellow, IEEE) received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 1997, and the M.S. and Ph.D. degrees in electrical engineering from Princeton University, Princeton, NJ, USA, in 1999 and 2002, respectively.

He was an Advisory Engineer with the IBM Microelectronics Division, Burlington, NJ, USA, from 2002 to 2003. He was a Full Professor with Pennsylvania State University, State College, PA, USA, from 2003 to 2014. He was a Visiting Researcher with Interuniversity Microelectronics Centre (imec), Leuven, Belgium, from 2005 to 2007 and in 2010. He was a Senior Manager and Principal Researcher with the AMD Research China Lab, Beijing, from 2012 to 2013. He is currently a Professor with the Department of Electrical and Computer Engineering, University of California at Santa Barbara, Santa Barbara, CA, USA. His interests include VLSI design, electronic design automation (EDA), computer architecture, and embedded systems.

Dr. Xie is also an AAAS Fellow and an ACM Fellow. He is an expert in computer architecture who has been inducted to the IEEE/ACM International Symposium on Computer Architecture (ISCA)/the IEEE/ACM International Symposium on Microarchitecture (MICRO)/IEEE International Symposium on High-Performance Computer Architecture (HPCA) Hall of Fame. He was a recipient of the Best Paper Awards at the HPCA 2015, the International Conference On Computer Aided Design (ICCAD) 2014, the ACM Great Lakes Symposium on VLSI (GLSVLSI) 2014, the IEEE Computer Society Annual Symposium on VLSI (ISVLSI) 2012, the ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED) 2011, the Asia and South Pacific Design Automation Conference (ASPDAC) 2008, and the International Conference on ASIC (ASICON) 2001, the Best Paper Nominations at the ASPDAC 2014, MICRO 2013, the Design, Automation and Test in Europe Conference (DATE) 2013, ASPDAC 2010–2009, and ICCAD 2006, the 2016 IEEE Micro Top Picks Award, the 2008 IBM Faculty Award, and the 2006 NSF CAREER Award. He has served as the TPC Chair for ICCAD 2019, HPCA 2018, ASPDAC 2013, ISLPED 2013, and the International Forum on Embedded MPSoC and Multicore (MPSOC) 2011, a Committee Member for the IEEE Design Automation Technical Committee (DATC), the Editor-in-Chief of the *ACM Journal on Emerging Technologies in Computing Systems*, and an Associate Editor for the *ACM Transactions on Design Automations for Electronics Systems*, the IEEE TRANSACTIONS ON COMPUTERS, the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND SYSTEMS, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, the *IEEE Design & Test of Computers*, and the *IET Computers & Digital Techniques*. Through extensive collaboration with industry partners (e.g., AMD, HP, Honda, IBM, Intel, Google, Samsung, IMEC, Qualcomm, Alibaba, Seagate, and Toyota), he has helped the transition of research ideas to industry.

Luping Shi received the B.S. and M.S. degrees in physics from Shandong University, Jinan, China, in 1981 and 1988, respectively, and the Ph.D. degree in physics from the University of Cologne, Cologne, Germany, in 1992.

He was a Post-Doctoral Fellow with Fraunhofer Institute for Applied Optics and Precision Instrument, Jela, Germany, in 1993, and a Research Fellow with the Department of Electronic Engineering, City University of Hong Kong, Hong Kong, from 1994 to 1996. From 1996 to 2013, he was a Senior Scientist and the Division Manager with Data Storage Institute (DSI), Singapore, and led nonvolatile solid-state memory (NVM), artificial cognitive memory (ACM), and optical storage researches. He joined Tsinghua University, Beijing, China, as a National Distinguished Professor, in 2013. By integrating seven departments, he established Center for Brain Inspired Computing Research (CBICR), Tsinghua University, in 2014, where he served as the Director. His research interests include brain-inspired computing, neuromorphic chip, and memory devices. He has published more than 200 articles in prestigious journals, including *Nature*, *Science*, *Nature Photonics*, *Advanced Materials*, and the *Physical Review Letters*.

Dr. Shi is also an SPIE Fellow. He is also an Editor Board Member of *Scientific Reports* and an Associate Editor of *Frontiers in Neuroscience* and *Frontiers in Neuromorphic Engineering*. He was a recipient of the 2004 National Technology Award of Singapore, the unique awardee that year. He has served as a General Co-Chair of the Asia-Pacific Conference on Near-field Optics 2013, Non-Volatile Memory Technology Symposium (NVMTS) 2011–2015, the East-West Summit on Nanophotonics and Metal Materials 2009, and the Industrial Optical Devices and Systems (ODS) 2009.