Parana: A Parallel Neural Architecture Considering Thermal Problem of 3D Stacked Memory

Shouyi Yin, Shibin Tang, Xinhan Lin, Peng Ouyang, Fengbin Tu, Leibo Liu, Jishen Zhao, *Member, IEEE*, Cong Xu, *Member, IEEE*, Shuangcheng Li, Yuan Xie, *Fellow, IEEE*, and Shaojun Wei

Abstract—Recent advances in deep learning (DL) have stimulated increasing interests in neural networks (NN). From the perspective of operation type and network architecture, deep neural networks can be categorized into full convolution-based neural network (ConvNet), recurrent neural network (RNN), and fully-connected neural network (FCNet). Different types of neural networks are usually cascaded and combined as a hybrid neural network (Hybrid-NN) to complete real-life cognitive tasks. Such hybrid-NN implementation is memory-intensive with large number of memory accesses, hence the performance of hybrid-NN is often limited by the insufficient memory bandwidth. A "3D + 2.5D" integration system, which integrates a high-bandwidth 3D stacked DRAM side-by-side with a highly-parallel neural processing unit (NPU) on a silicon interposer, overcomes the bandwidth bottleneck in hybrid-NN acceleration. However, intensive concurrent 3D DRAM accesses produced by the NPU lead to a serious thermal problem in 3D DRAM. In this paper, we propose a neural processor called *Parana* for hybrid-NN acceleration in consideration of thermal problem of 3D DRAM. Parana solves the thermal problem of 3D memory by optimizing both the total number of memory accesses and memory accessing behaviors. For memory accessing behaviors, Parana balances the memory bandwidth by spatial division mapping hybrid-NN onto computing resources, which efficiently avoids that masses of memory accesses are issued in a short time period. To reduce the total number of memory accesses, we design a new NPU architecture and propose a memory-oriented tiling and scheduling mechanism to exploit the maximum utilization of on-chip buffer. Experimental results show that Parana reduces the peak temperature by up to 54.72 ℃ and the steady temperature by up to 32.27 ℃ over state-of-the-art accelerators with 3D memory without performance degradation.

Index Terms—Neural processor, hybrid neural networks, thermal problem, 3D memory

1 Introduction

After almost two decades of stagnation, Artificial Intelligence (AI) now is embracing another peak in its history. The fast-developing deep learning (DL) techniques and their wide deployment serves as the main driving horse of this new technology wave [1], [2], [3], [4]. Evolved from conventional neural network (NN) and machine learning (ML) algorithms, DL involves many layers with complex structures and/or multiple nonlinear transformations to model a high-level abstraction of data. The ability to learn tasks from examples makes DL particularly powerful in handling so-called cognitive applications such as image understanding, object detection, voice recognition, scene labeling, and video surveillance [1], [2], [3], [4], [5], [6], [7], [8], [9], etc.

From the perspective of operation type and network architecture, deep neural networks can be categorized into full convolution-based neural network (ConvNet), recurrent neural network (RNN) and fully-connected neural network (FCNet). ConvNet is composed of convolutional layers and pooling layers, which is good at visual feature extraction. FCNet is composed of multiple fully connected layers, which is good at classification. RNN is composed of fully connected layers with feedback paths, which is good at sequential data processing. In a practical DL application, several neural networks are usually cascaded as a hybrid neural network (Hybrid-NN). For example, AlexNet [1] and VGGNet-16 [10] which are proposed for image classification are actually the cascade of a ConvNet and FCNet. LRCN [3] is another representative hybrid-NN for visual recognition and description. Fig. 1 summaries the network structures of AlexNet and LRCN. PathNet[2], proposed by Google-DeepMind, demonstrates the general-purpose artificial intelligence with hybrid-NN. Google-DeepMind also proposed a unified hybrid-NN to implement various intelligent tasks [11].

Hybrid-NNs are usually memory-intensive high-throughput algorithms with large amounts of parallel computation

- *S. Yin, S. Tang, X. Lin, P. Ouyang, F. Tu, L. Liu, and S. Wei are with the Institute of Microelectronics, Tsinghua University, Beijing 100084, China. E-mail: yinsy@tsinghua.edu.cn.*
- *J. Zhao is with the University of California, San Diego, CA 92093-0021.*
- *C. Xu, S. Li, and Y. Xie are with the University of California, Santa Barbara, CA 93106.*

Manuscript received 20 Aug. 2017; revised 13 June 2018; accepted 22 June 2018. Date of publication 0 . 0000; date of current version 0 . 0000.
(Corresponding author: Shouyi Yin).
Recommended for acceptance by M. Huebner.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TPDS.2018.2858230

1045-9219 © 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

IEEE Print

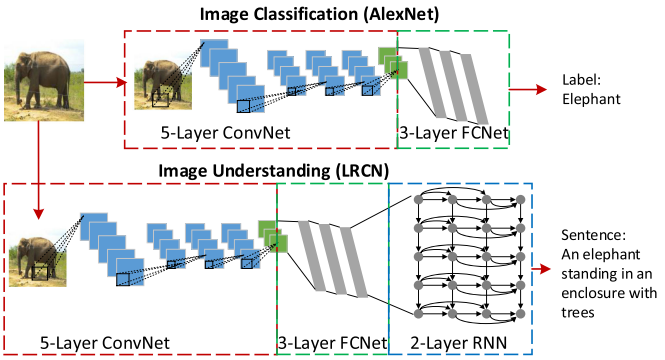


Fig. 1. Examples of hybrid neural networks (Hybrid-NN): AlexNet [1] for image classification and LRCN [3] for image understanding.

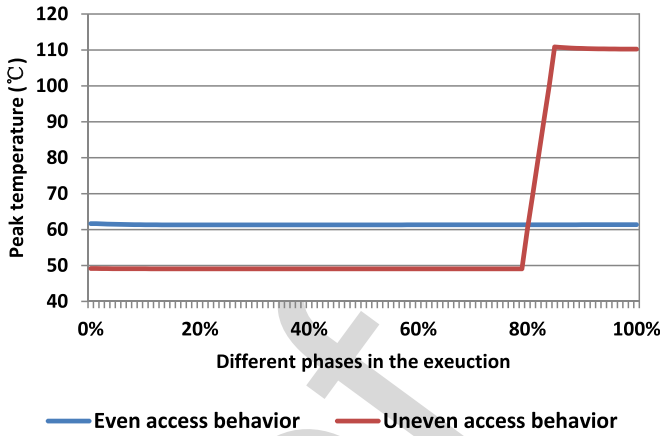


Fig. 2. The temperature affected by memory access behaviors. The data is calculated with CACTI-3DD [22] and HotSpot [23].

and memory accesses. The acceleration of hybrid-NN requires a large on-chip memory and a high memory bandwidth. 3D stacked memory [12] provides the large capacity and the high bandwidth for the high-throughput computation. 3D stacking neural processors *Neurocube* [13] and *TETRIS* [14] integrate a 3D DRAM on top of the neural processor unit (NPU). However, the vertical 3D stacking integration easily suffers from customization of through-silicon vias in 3D DRAM based on the layout of the NPU [15]. Compared to the vertical 3D stacking, a "3D + 2.5D" integration system integrates a high-bandwidth 3D stacked DRAM side-by-side with a highly-parallel NPU on a silicon interposer. It avoids the customization of 3D DRAM. Benefited from the advantages of "3D + 2.5D" integration, high-bandwidth modern CPU/GPU are implemented with 2.5D integration, such as AMD's Fiji [16], NVIDIA's Pascal [17] and Intel's Knights Landing [18]. Such "3D + 2.5D" integration is also explored for neural acceleration, demonstrated by the Nervana Engine [19]. The "3D + 2.5D" integration system provides a promising solution to accelerate the high-throughput neural network. However, this 3D memory architecture may come out a serious thermal problem for integration with NPUs [15]. A higher temperature increases the cooling cost [15] and reduces the life-time of 3D memory [20]. For example, the latest Google's TPU3 used 3D memory to help address the memory bandwidth problem, but had to use liquid-cooling to address the thermal challenge [21]. A straightforward approach to lower the average temperature of 3D memory is to reduce the number of memory accesses over a certain period of time so that the dynamic energy of accessing memory is decreased. However, the peak temperature is also affected by the transient memory bandwidth. Fig. 2 shows different peak temperature over time caused by two different memory access behaviors. The total number of memory accesses are the same in the two cases. With evenly distributed memory access in different phases of the execution, the peak temperature stays almost constant over time. In contrast, with the uneven access behavior, the last 20 percent execution phase takes 80 percent memory accesses, resulting in a much higher the peak temperature. When the temperature exceeds the maximum operation temperature (85 °C), the error rates of DRAM can go up dramatically.

In this paper, we propose a "3D + 2.5D" integration neural processor called Parana for thermal-efficient hybrid-NN acceleration. Parana integrates 3D memory and NPU with the "3D + 2.5D" solution and tackles the thermal problem of 3D memory by reducing the total number of memory accesses and reshaping memory access behaviors. Specially, Parana balances the memory bandwidth with the help of spatial division mapping by avoiding bursty memory traffic. To reduce the total number of 3D memory accesses, we design a novel NPU architecture and optimize the tiling and scheduling policy to maximize the utilization of on-chip buffer. The key contributions are summarized as following:

- For the first time, we consider the thermal problem of 3D Memory in NN accelerator, and jointly optimize the total number of memory accesses and the memory access pattern, to reduce the steady temperature and peak temperature.
- For the first time, we design a spatial division mapping (SDM) mechanism to utilize the complementarity between different NNs, resulting in the smoothness of the memory access pattern. We propose a divisible processing element (PE) array for concurrently accelerating different NNs in parallel.
- We simultaneously use SDM, sparsity, unified buffer and dataflow scheduling techniques to achieve joint optimization. We propose an analysis framework to find the most profitable tiling and scheduling solutions.

2 Background and Related Work

This section presents the background on hybrid-NN, spatial architecture, 3D-stacked memory, "3D + 2.5D" integrated systems, and the related work.

2.1 Background

Hybrid Neural Network (Hybrid-NN). Hybrid-NN typically consists of ConvNet, FCNet, and RNN.

- *ConvNet:* The main operation of ConvNet is convolution. In each convolutional layer, a 3D filter is used to extract features from the input data. The output feature map is generated by means of sliding the 3D filter on the input data. For instance, it receives n_c feature maps as input data and outputs m_c feature maps. The m_c feature maps correspond to the filtered results of m_c 3D filters. Each 3D filter is operated

IEEE Proof

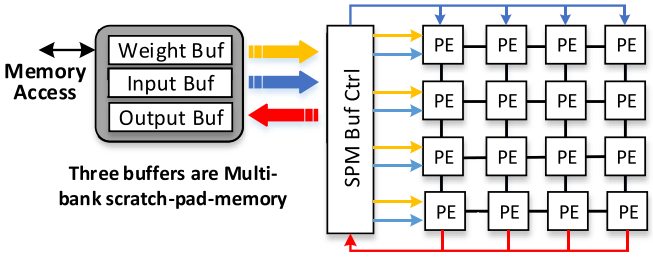


Fig. 3. Overview of the spatial architecture. The common spatial architecture is composed of a PE array and three isolated buffers for temporal input, output, and weight in a neural network.

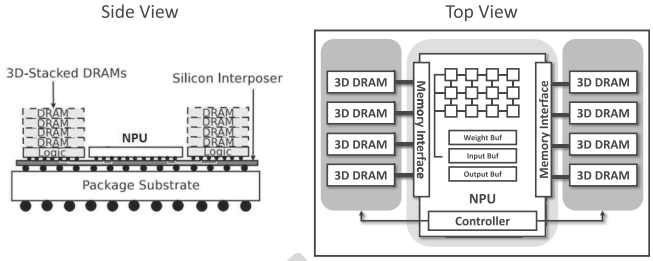


Fig. 4. "3D + 2.5D" system architecture for neural network acceleration.

on n_c input feature maps to generate one output feature map.

- *FCNet:* FCNet mainly performs matrix multiplications. The input data include a vector and a weighting matrix while the output data is a vector.
- *RNN:* Generally, Long-Short-Term-Memory (LSTM) networks are the best performing RNNs [24]. In LSTM-based RNNs, primary operations are full connection operations like FCNet and element-wise vector multiplications that perform the gating function. *In this paper, we use RNN to represent LSTM-based RNN.*

Spatial Architecture. Many prior work [7], [8], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40], [41], [42], [43], [44], [45], [46], [47], [48], [49] have used spatial architectures to accelerate neural networks in ASIC and FPGA implementations. Fig. 3 reviews the spatial architecture in ShiDianNao [29], Eyeriss [8], and ENVISION [50]. The key of the spatial architecture is the PE array which exploits high compute parallelism by using direct communication between simple PEs. Input, output and weight buffers are designed as the scratch-pad-memory (SPM) to save temporal data in the acceleration of hybrid-NN on PE array. The NPU loads inputs and weights from DRAM and stores them into input and output buffer. The PE array reads them from these buffers, carries out the neural computation, and writes outputs back to output buffers. The final outputs are transmitted from output buffer to DRAM. To meet the high parallel degree of PE array, each SPM buffer is composed of multiple memory banks.

3D Memory and "3D + 2.5D" Integration. 3D memory is a promising solution to meet the high capacity and bandwidth demands in the acceleration of high-throughput hybrid-NN. In the early days of neuromorphic, 3D wafer scale integration(3D WSI) has been used in neural network processor design, such as [51]. The work showed that 3D WSI technology can provide a hardware medium for constructing massively parallel computing processors like neural network accelerator. In more recent applications, 3D integration is generally used with memristors in neuromorphic system [52]. This work proposed a 3D synaptic architecture based on Ta/TaOx/TiO_2/Ti RRAM for analog neuromorphic computation. High-bandwidth Memory (HBM) is a typical 3D stacked memory standard proposed by JEDEC [53] for high-bandwidth applications. HBM is composed of 4 DRAM dies and one single logic die, and provides maximum 256 GBps memory bandwidth and 4GB memory capacity. There are two major integration solutions with 3D memory and NPU: vertical 3D integration and "3D + 2.5D" integration (depicted in Fig. 4). Vertical 3D integration places the 3D memory directly on top of the processing unit [13], [14], [54], [55], [56], while "3D + 2.5D" solution integrates wide-interface 3D memory side-by-side with the processing unit on a silicon interposer. The latter one has been proved to be a good solution for high-throughput computation [16], [17], [18], and is explored for neural acceleration recently [19], [57].

2.2 Other Related Work

In addition to the related work described in previous subsection, other related work on ConvNet accelerators, and FCNet/RNN accelerators are briefly described in this subsession.

ConvNet Accelerators. In general, ConvNets are computationally intensive, ConvNet acceleration has focused on improving the peak performance and the energy efficiency. Some accelerators [7], [9], [25], [29], [30], [31], [39], [42], [43], [46] exploit the dataflow to activate more PEs to improve the computing parallelism. Others [8], [26], [36], [58] exploit the locality in dataflow to improve energy efficiency.

FCNet/RNN Accelerators. FCNet/RNN acceleration are usually memory-bounded. Consequently, most of the prior work of FCNet/ConvNet acceleration focus on quantization, sparsity, and 3D stacked design. Previous researches [7], [9], [28], [40], [59] have retrained the neural network with a low-precision quantization. Some accelerators [41], [60] accelerate sparse FCNet with several PEs in SIMD. Because the sparsity of weight matrix is random, some PEs are always wasted in acceleration. The accelerators [13], [14], [19] improve performance of FCNet/RNN with high-bandwidth 3D memory.

Eyeriss [8], EIE [41], Neurocube [13] and TETRIS [14] are the most relevant accelerators. Eyeriss is designed with a optimized dataflow on the PE array and optimizes the memory accesses with a global buffer. EIE supports sparse fully connected networks with a SIMD architecture. Different from Parana, EIE distributes non-zero weights distributed in local storage of all PEs. The input neurons are broadcasted and each PE calculates parts of sparse FC with its local non-zero weights and the broadcasted inputs. Because the number of non-zero weights in each PE are different, EIE suffers from the load imbalance. Parana overcomes the load imbalance by storing input/output neurons in each PE and averagely spreading non-zero weights among all PEs. Neurocube integrates 3D memory on the NPU vertically. TETRIS improves the performance of 3D stacked NPU by trading off the size of on-chip buffer capacity and the number of used PEs, and also proposes a bypassing scheduling scheme to reduce memory accesses.

IEEE Proof

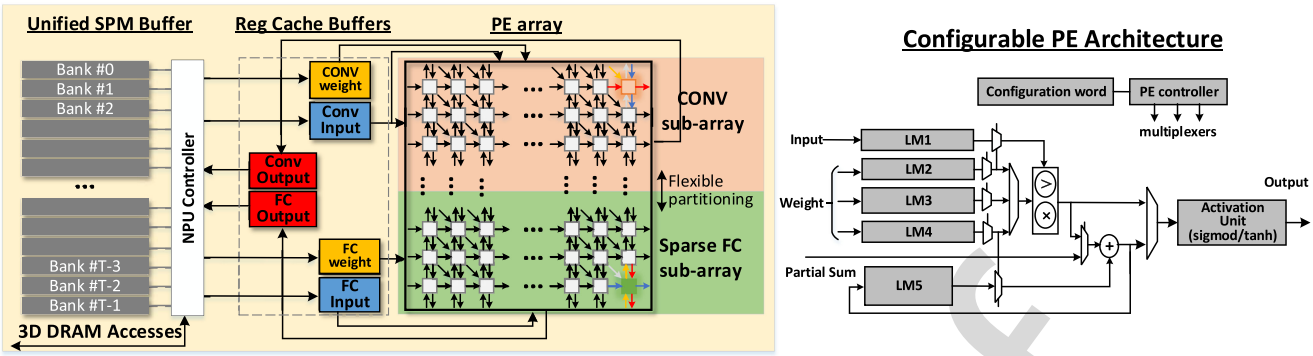


Fig. 5. Parana architecture overview.

Parana is different from the existing accelerator designs. In purpose, Parana jointly optimizes the total number of memory accesses and the memory access pattern, instead of merely avoiding masses of memory accesses, so that both the steady temperature and the peak temperature can be optimized. In methodology, our work not only adopts data reuse to reduce the number of memory accesses, but also utilizes the complementarity between different NNs to smooth the memory access pattern.

3 Motivation

We motivate our design by discussing the inefficiencies in prior NPU designs.

Thermal issue remains a critical concern towards feasible "3D + 2.5D" integrated systems [15]. Higher temperature can lead to substantially increased cooling and packaging costs in "3D + 2.5D" systems [61]. Furthermore, the increase of temperature can degrade DRAM reliability and performance. For example, JEDEC stipulates that systems running beyond 85 °C need to double memory self-refresh rate [62]; the rate continues to double for every ~ 10 °C degree beyond 85 °C [63]. Hence, to tackle the thermal problem of 3D memory, we need to consider both the total number of memory accesses and the memory access behaviors.

The state-of-the-art NPU designs focus on improving the peak performance and the energy efficiency. The design choices in state-of-the-art NPUs are not optimized for the thermal problem in 3D memory. We summarize three inefficiencies in NPU for the thermal problem in "3D + 2.5D" integration, as follows:

First, the time division mapping (TDM) scheme, which is widely adopted by the state-of-the-art NPUs [9], [7], [29], can generate substantial amount of heat by executing FCNet and RNN. Hybrid-NN commonly employs a cascaded network architecture. As such, ConvNet, FCNet and RNN are accelerated one-by-one under TDM. While the memory footprint of ConvNet is relatively small, FCNet and RNN are memory intensive. As a result, such uneven memory access behavior across ConvNet, FCNet and RNN can create hotspots in 3D memory of "3D + 2.5D" integrated NPUs, harmful to the peak temperature.

Second, most previous designs can lead to substantial underutilization of on-chip storage resources. Most prior spatial architecture designs employ isolated on-chip buffers to store input, output, and weight of a neural network, respectively [7], [9], [28], [29], [39], [40]. Yet, the memory footprint varies across input, output and weight in various layers of hybrid-NN. Fig. 6 summaries typical input, output and weight sizes of convolutional layers in hybrid-NNs. Due to the disparate sizes, the isolated buffer design cannot fully utilize all on-chip storage resources to reduce 3D memory accesses. A recent study Eyeriss [8] employs a global buffer design to improve the utilization of on-chip storage resources. Eyeriss also incorporates an analysis framework that optimizes system energy efficiency. However, Eyeriss does not take into account the thermal issue imposed by 3D + 2.5D integrated NPUs.

Third, the sparse fully connected layers are usually accelerated by the SIMD PEs based NPUs [41], [60]. So far for the 2-D PE array based NPU, it has to store the dense representation in the on-chip buffer by filling the pruning connections in FCNet and RNN with zero. The dense representation occupies more on-chip buffer,resulting in extra DRAM accesses, harmful to the steady temperature of 3D memory.

4 Parana

In "3D + 2.5D" integration, the proposed neural processor solves the thermal problem by reducing the total number of memory accesses and controlling the memory access behaviors. This section presents the Parana architecture and the optimization scheme behind Parana, which reduces the temperature of 3D memory. We also propose a analysis framework to fully exploit the advantage of Parana architecture to reduce temperature. In ConvNet, each pooling operation can be treated as a specific convolution by swapping MAC in convolution with MAX comparison. In RNN, The gate indicators are iteratively calculated by such kind of operations: $Gate_t = \delta(W_x \cdot X_t + W_h \cdot H_{t-1} + bias)$, where, in iteration t, $Gate_t$ denotes the input gate, forget gate or output gate. Xt and Wx are the corresponding input data and weighting matrix, respectively. $\delta()$ represents the nonlinear operation like sigmoid and tanh. The matrix operation "$\cdot$" can be regarded as fully connected operations. Therefore, we introduce the implementation of ConvNet, FCNet, RNN on Parana only with convolution computation (CONV) and fully connected computation (FC).

4.1 Architecture Overview

Parana reshapes the memory access behavior with spatial division mapping (SDM), which is supported by a divisible and configurable PE array. To reduce the total memory accesses, a tiling and scheduling mechanism for unified SPM buffer is used to maximize the data reuse. Fig. 5 depicts the overview of

IEEE Proof

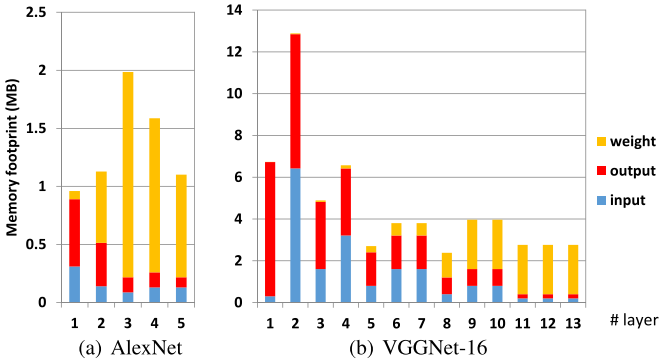


Fig. 6. Input, output and weight sizes of various convolutional layers in AlexNet [1] and VGGNet-16 [10].

Parana architecture. It consists of four major components: a divisible and configurable PE array, a unified SPM buffer, multiple register cache buffers and a dataflow controller.

The Divisible PE Array. To enable SDM, the PE array is allowed to be partitioned into two parts, in which one part is used to accelerate CONV and the other part is used to accelerate sparse FC. In the divisible PE array, the network on-chip (NoC) supports two fixed dataflows: one for CONV and the other for sparse FC. Each PE includes one PE controller (include configuration words), one MAC unit for CONV and FC computation, one MAX comparison for pooling operation and five blocks of local memory (LM1 $\sim$ LM5). To support flexible partitioning solutions in SDM, each PE supports two working modes: CONV mode and sparse FC mode. The logic resources and local memories in the PE are reused for the two modes.

The Unified SPM Buffer. It is implemented with multiple SRAM banks to meet the high parallelism of the PE array and occupies an independent address space in the system address space. The key of the unified SPM buffer is maximizing the data reuse in the acceleration of hybrid-NN. The buffer provides inputs and weights for the computation in the PE array and stores the outputs from the PE array. Inputs, outputs, and weights are saved in different address spaces of the buffer, which is calculated by the Parana compiler in the compiling phase. For different hybrid-NN layers, the boundaries of different address spaces are allowed to be shifted to maximize the utilization of the SPM buffer capacity.

Register Cache Buffers. Different with the SPM buffer, the cache buffers do not occupy any space in the system address space. The temporal data in register cache buffers is the subset of the unified SPM buffer. Each register cache buffer is implemented with register files and organized as a FIFO. Although the unified SPM buffer is implemented as multi-bank which provide multiple data accesses in each cycle, it is hard to serve the requests for different types of data simultaneously, i.e., inputs/outputs/weights. Register cache buffers are used to bridge the dataflow between the unified SPM buffer and the PE array. For each access to the unified SPM buffer, one register cache buffer is filled up with the SPM buffer access and the register cache buffer provides the data to the PE array for each cycle. The unified SPM buffer services six register cache buffers in time division multiplexing.

Parana Controller. Parana is a data-driven neural processor. The controller is used to configure the PE array, manage the unified SPM buffer and schedule the dataflow for efficient hybrid-NN acceleration with a low temperature in 3D memory. The controller first loads the configuration to configure the data path in PE array. Then, the controller loads data from 3D DRAM, saves different types of the data in independent spaces of the unified SPM buffer and feeds the PE array with the data. When the data is worn out, the acceleration is finished.

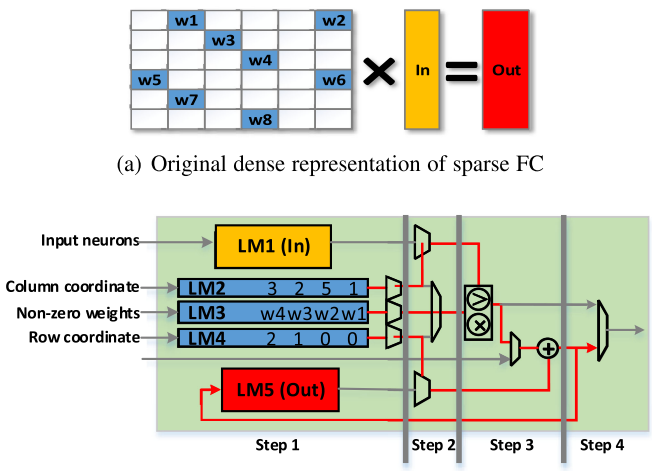


Fig. 7. An example of sparse FC acceleration on PE.

4.2 Configurable PE for Both CONV and Sparse FC Computation

To provide different partitioning solutions in SDM, each PE in Parana can be configured into either CONV mode or sparse FC mode. Fig. 5 depicts the PE architecture, in which the working mode of each PE is decided by the configuration word in the PE and the computation of sparse FC or CONV is controlled by the PE controller. The local memories and logic resources in each PE are shared by both modes.

In sparse FC mode, each PE accelerates the sparse FC as depicted in Fig. 7. Fig. 7a shows the orignal representation of sparse FC. Fig. 7b presents the sparse FC acceleration on the PE. In sparse FC acceleration, inputs and outputs are respectively saved in LM1 and LM5 of the PE. LM2, LM3, and LM4 are implemented by FIFO and used to buffer the non-zero weights and the corresponding row/column indices of each non-zero weight. The sparse FC computation on the PE is performed in four steps, under the control of PE controller: (1) The PE reads non-zero weight ($weight$) and its column/row coordinates from LM3, LM2, and LM4; (2) The PE uses the column coordinate to index the input operand ($in[col]$) in LM1 and uses row coordinate to index the partial sum ($out[row]$) saved in LM5; (3) The PE executes multiply-and-accumulating operation $out[row] = in[col] * weight$; (4) The produced partial sum is written back into LM5.

Under convolutional mode, the computing process of convolution in Parana follows the systolic dataflow [8]. As depicted in Fig. 8, in each PE, LM1 saves one row of input feature maps and LM2/LM3/LM4 saves one row of weight kernels. The logic resources in the PE take the inputs to calculate one row of partial convolutional sums and the PE saves the partial sums in LM5 to implement accumulation with future results. The CONV computation in each PE is

IEEE Proof

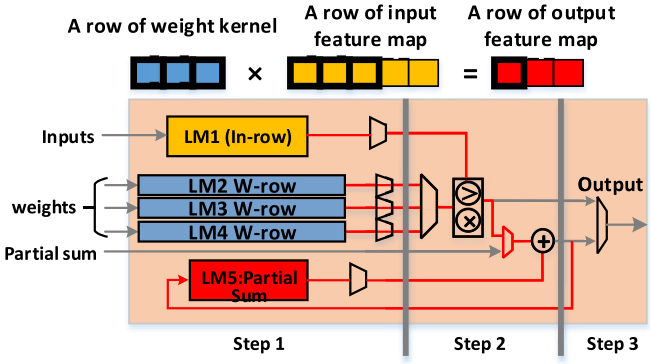


Fig. 8. An example of CONV acceleration on PE.

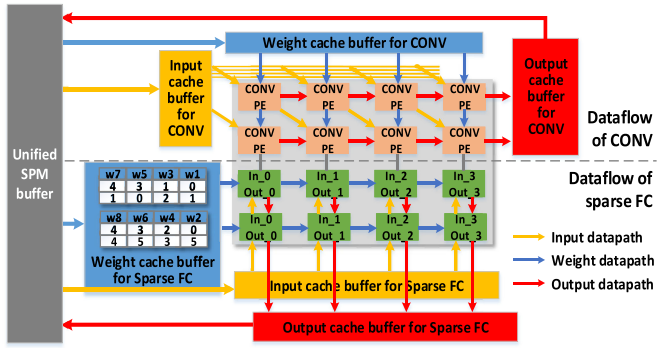


Fig. 9. The dataflows of CONV and sparse FC in SDM.

performed in three steps, under the control of PE controller: (1) The PE loads one kernel weight ($weight$) from LM2/LM3/LM4, one input (in) from LM1 and one partial sum from LM5; (2) The PE executes multiply-and-accumulating operation $out = in * weight$; (3) The produced partial sum is written back into LM5.

4.3 Divisible PE Array for Spatial Division Mapping (SDM)

To enable array partitioning, Parana NOC is configurable and divisible. To enable CONV and sparse FC acceleration on the PE array simultaneously, it provides independent input/output ports around each edge of PE array and avoids the datapath conflicts in the computation of CONV and sparse FC.

Fig. 9 demonstrates the SDM partitioning and the dataflows of hybrid-NN on the partitioned PE array. The demo shows an equal partitioning for a CONV sub-array and a sparse FC sub-array. The CONV sub-array accelerates ConvNet and the sparse FC sub-array accelerates FCNet and RNN. Each sub-array works with three independent register cache buffers (input, output and weight).

To accelerate the sparse fully connected layers on the sparse FC sub-array depicted in Fig. 9, all PEs in the column i duplicate the input vector (in_i) and output vector (out_i). The non-zero weights are uniformly divided into three vectors, which are equal to the number of rows in the PE array. Each row of the PE array computes with one subvector of non-zero weights. The sparse FC acceleration in each PE is as the same depicted as Fig. 7b. The dataflow of sparse FC computation includes three stages: input operand duplication, sparse FC computation, and the accumulation of outputs. At the first stage, input operands are transmitted into the PE array from the bottom PEs and saved into LM1 in each PE. Input neurons are reused across PEs vertically. At the second stage, non-zero weights are uniformly divided into three parts and transmitted into the corresponding rows of the PE array one by one. Each PE multiples the non-zero weight by the saved input operand and accumulates the product with the partial sum saved in the PE. At the third stage, partial sums for the same output at the same column are accumulated and sent out from the PE array.

The convolution computation with the CONV sub-array follows the systolic dataflow in [8]. The CONV sub-array loads weights from the PEs at the top edge. The inputs are transmitted into the PE array along vertical direction. It loads inputs from the PEs at the left and top edges and transmits the inputs along diagonal direction. The outputs are sent out from left to right and written back to the register cache buffer.

To illustrate why SDM can reduce memory bandwidth in the acceleration of hybrid-NN on NPU. We assume that NPU adopts an optimal scratch-pad-memory buffer (OPT-Buffer) which fully exploits the locality in neural networks and 3D memory provides sufficient bandwidth for neural computation. Table 1 lists all parameters used in this paper.

For the acceleration of $(M \times N)$ full-connection layer in FCNet and RNN, M and N are respectively the numbers of output neurons and input neurons. Assuming the number of PEs is PE_{FC}, the bandwidth demand of sparse FC is

$$BW_{FC} = \frac{M \cdot N \cdot SPA \cdot DL}{(M \cdot N \cdot SPA)/(PE_{FCNet/RNN} \cdot f)} = DL \cdot f \cdot PE_{FCNet/RNN} \quad (1)$$

in which f is the frequency of the NPU. For the acceleration of (R, C, M, N, K) convolutional layer in ConvNet, (R, C) are the size of output feature map, (M, N) are respectively the number of output and input feature maps, and K is the kernel size. With P_{CONV} PEs, the inference time of CONV is

$$T_{CONV} = (R \cdot C \cdot M \cdot N \cdot K^2)/(PE_{ConvNet} \cdot f) \quad (2)$$

in which, $(R \cdot C \cdot M \cdot K^2)$ is the MAC operations in the convolutional layer. The bandwidth demands for input, output, and weight are respectively

$$BW_{Input} = \frac{R \cdot C \cdot N \cdot DL}{T} = \frac{DL \cdot f}{M \cdot K^2} \cdot PE_{ConvNet} \quad (3)$$

$$BW_{Output} = \frac{R \cdot C \cdot M \cdot DL}{T} = \frac{DL \cdot f}{N \cdot K^2} \cdot PE_{ConvNet} \quad (4)$$

$$BW_{Weight} = \frac{M \cdot N \cdot K^2 \cdot DL}{T} = \frac{DL \cdot f}{R \cdot C} \cdot PE_{ConvNet}. \quad (5)$$

Hence, the bandwidth demand of CONV is

$$BW_{CONV} = BW_{Input} + BW_{Output} + BW_{Weight} \quad (6)$$

$$= \left(\frac{1}{M \cdot K^2} + \frac{1}{N \cdot K^2} + \frac{1}{R \cdot C}\right) \cdot DL \cdot f \cdot PE_{CONV} \quad (7)$$

$$\leq DL \cdot f \cdot PE_{ConvNet}. \quad (8)$$

IEEE Proof

TABLE 1
NotationDescription

Hybrid-NN Parameters	
B, DL	Batch size and data length of hybrid-NN
(R, C, M, N), K, $L_{ConvNet}$	Parameters of CONV: (R, C) are rows and columns of feature maps, M is the number of output maps, N is the number of input maps, K is the kernel size, $L_{ConvNet}$ is the number of layers in ConvNet
(I, O), SPA, $L_{FCNet/RNN}$	Parameters of FC: I is the number of input neurons, O is the number of output neurons, SPA is the degree of sparsity, L_{FCNet} is the number of layers in FCNet/RNN
Architecture Configuration	
SPM	Storage size of the unified SPM buffer.
PE	the number of PEs in the PE array.
Partition	
$SPM_{ConvNet}$, $SPM_{FCNet/RNN}$	The SPM buffer size of ConvNet and FCNet/RNN in SDM
$PE_{ConvNet}$, $PE_{FCNet/RNN}$	The number of PEs for ConvNet and FCNet/RNN in SDM
Tiling and Scheduling	
$(T_b, T_r, T_c, T_m, T_n)$	Tiling parameters for CONV
(T_b, T_i, T_o)	Tiling parameters for FC
$CONV{-}ir$, $FC{-}ir$	reuse inputs
$CONV{-}wr$, $FC{-}wr$	reuse weights
$CONV{-}or$, $FC{-}or$	reuse partial sums

From the Equations (1) and (8), we find that the bandwidth demands in CONV is smaller than sparse FC with the same row of PEs. Hence SDM, which accelerate ConvNet and sparse FC in parallel, is beneficial to reducing the maximum bandwidth demand in NPU and the peak temperature in 3D memory.

Different partitioning solutions of the PE array and the SPM buffer leads to different thermal and performance effects. Section 5.2 introduces the optimization framework which achieves the most profitable spatial allocation for the lowest peak temperature without performance degradation.

5 Optimization Framework

To accelerate a certain hybrid-NN on Parana, we propose a joint spatial resource partitioning and tiling/scheduling mechanism to minimize both peak temperature and steady temperature of 3D memory. The spatial resource partitioning can reshape the 3D memory access behavior and results in even distribution of memory access over time, which can lower peak temperature. The tiling and scheduling of hybrid-NN can improve the utilization of on-chip buffer and reduce the number of 3D memory accesses, which can lower steady temperature.

We first introduce the analysis model to achieve the most profitable tiling/scheduling of hybrid-NN and resource partitioning of PE array. Then we present the compiling workflow and the execution workflow with Parana.

5.1 Buffer Oriented Tiling and Scheduling to Minimize 3D Memory Accesses

Because the configuration of different layers is largely different in hybrid-NN, we analyze the tiling and scheduling scheme for CONV and sparse FC to maximize the utilization of the unified SPM buffer and data reuses in hybrid-NN. In previous NPUs, tiling is usually used to extract a sub-CONV or sub-FC which can be mapped on the PE array to optimize the computation flow [9], [39]. For Parana, we propose a buffer-oriented tiling method. It tiles the sub-CONV and sub-FC of hybrid-NN to match the size of SPM buffer. For a CONV layer (R, C, M, N) and the kernel size is K, we tile the CONV layer with parameters (T_r, T_c, T_m, T_n) to ensure that the unified SPM buffer can save the all inputs/outputs/weights in the tiled convolution. A high utilization of the unified SPM buffer capacity can be achieved by optimizing the tiling parameters.

Scheduling is used to maximize the data reuse between the tiled neural network layers. There are three classic reuse patterns used by state-of-the-art NPUs [7], [26], [28], [29], [43], including input reuse (ir), weight reuse (wr) and partial sum reuse for output (or). Take the input reuse of CONV as an example, the inputs in one tiled sub-CONV keep being on the chip until all tiled sub-CONVs which need the inputs have been finished by the NPU. The input reuse is suitable to the layer in which the memory footprint of inputs is larger than outputs and weights. Because different layers own different memory footprints of inputs, outputs, and weights, one reuse pattern cannot suit to all layers in hybrid-NN. A dynamic scheduling mechanism for different layers is beneficial to maximize the data reuse in hybrid-NN.

To achieve the most profitable tiling and scheduling for each CONV and fully connected layer, we setup the models for CONV and FC to analyze the memory accesses. In the models, Parana works in batch mode. For ConvNet, Parana accelerate a group of inputs in each inference one by one. For FCNet and RNN, a batch of inputs are processed together. The tiling and scheduling models for one CONV and a batch of FCs are discussed as follows.

Tiling and Scheduling Models for One CONV. Convolutional (CONV) tiling is used to select the most profitable parameters (T_r, T_c, T_m, T_n), which allow the tiled convolution to be saved in SPM buffer directly. With these tiling parameters, we calculate SPM demands for CONV inputs, outputs, and weights, as follows:

$$SPM_{CONV-i} = T_r \cdot T_c \cdot T_n \tag{9}$$

$$SPM_{CONV-o} = T_r \cdot T_c \cdot T_m \tag{10}$$

$$SPM_{CONV-w} = T_m \cdot T_n \cdot K \cdot K. \tag{11}$$

Under the tiling mode, Parana completes the CONV computation by loading different tiled convolutions Rpt times, and the Rpt is

$$Rpt = \left\lceil \frac{R}{T_r} \right\rceil \cdot \left\lceil \frac{C}{T_c} \right\rceil \cdot \left\lceil \frac{M}{T_m} \right\rceil \cdot \left\lceil \frac{N}{T_n} \right\rceil. \tag{12}$$

In a simple scheduling without data reuses between tiled convolutions, the total number of 3D memory accesses is the product of SPM buffer size (SPM for output need to be calculated two times for loading the partial sum and writing

IEEE Proof

back the updated partial sum) and *Rpt*. Efficient scheduling reduces the 3D memory accesses by fixing one kind of data (e.g., tiled input, weight or the partial sum of output) in SPM buffer and reuses the data for different tiled CONV computation. Hence, the total number of memory accesses under input reuse, partial sum reuse and weight reuse scheduling are respectively

$$MA_{CONV-ir} = R \cdot C \cdot N + (2 \cdot SPM_{CONV-o} + SPM_{CONV-w}) \cdot Rpt \quad (13)$$

$$MA_{CONV-or} = R \cdot C \cdot M + (SPM_{CONV-i} + SPM_{CONV-w}) \cdot Rpt \quad (14)$$

$$MA_{CONV-wr} = M \cdot N \cdot K \cdot K + (SPM_{CONV-i} + 2 \cdot SPM_{CONV-o}) \cdot Rpt \quad (15)$$

in which, the first term of each equation is the original data size for input, output, and weight and other terms are calculated by multiplying the SPM buffer size by Rpt.

The minimal number of 3D memory accesses for the CONV layer is achieved by resolving the following optimization problem:

$$min \quad MA_{CONV} = \min(MA_{CONV-ir}, MA_{CONV-or}, MA_{CONV-wr}) \quad (16)$$

s.t.

$$SPM_{CONV-i} + SPM_{CONV-o} + SPM_{CONV-w} \leq SPM_{ConvNet} \quad (17)$$

By enumerating all possible tiling under the three reuse patterns, the Parana complier achieves the most profitable tiling and scheduling configuration for the CONV layer with limited SPM buffer size. The total amount of 3D memory access in ConvNet can be calculated with the MA_{CONV} of all CONV layers.

Tiling and Scheduling Models for a Batch of FCs. The tiling and scheduling model in FC is similar to CONV. The SPM storage demands of input, output, and weight for FC are respectively

$$SPM_{FC-i} = T_i \cdot T_b \quad (18)$$

$$SPM_{FC-o} = T_o \cdot T_b \quad (19)$$

$$SPM_{FC-w} = T_i \cdot T_o \cdot SPA \cdot 3 \quad (20)$$

in which SPA is the degree of sparsity in FC and the factor of 3 is the storage cost for the representation of sparse FC. The repeating times for tiled FCs computation is

$$Rpt = \left\lceil \frac{B}{T_b} \right\rceil \cdot \left\lceil \frac{I}{T_i} \right\rceil \cdot \left\lceil \frac{O}{T_o} \right\rceil. \quad (21)$$

The total number of 3D memory accesses under input reuse, partial sum of output reuse and weight reuse are respectively

$$MA_{FC-ir} = I \cdot B + (2 \cdot SPM_{FC-o} + SPM_{FC-w}) \cdot Rpt \quad (22)$$

$$MA_{FC-or} = O \cdot B + (SPM_{FC-i} + SPM_{FC-w}) \cdot Rpt \quad (23)$$

$$MA_{FC-wr} = I \cdot O \cdot SPA \cdot 3 + (SPM_{FC-i} + 2 \cdot SPM_{FC-o}) \cdot Rpt. \quad (24)$$

The optimization problem for the tiling and scheduling of FC is

$$min \ MA_{FC} = \min(MA_{FC-ir}, MA_{FC-or}, MA_{FC-wr}) \quad (25)$$

s.t.

$$SPM_{FC-i} + SPM_{FC-o} + SPM_{FC-w} \leq SPM_{FCNet/RNN}. \quad (26)$$

With MA_{FC} of all full-connection layers, the complier can calculate the number of 3D memory accesses for FCNet and RNN with assigned SPM storage.

5.2 Spatial Resource Partitioning to Minimize Peak Bandwidth

By partitioning the PE array and SPM buffer, multiple NNs can be accelerated on Parana in parallel. Then the neutralization of peak bandwidth of different NNs results more smooth memory access behavior. To find the best resource partitioning, we formulate an bi-objective optimization problem. Because FCNet and RNN are mainly compose of FC layers, in the operation problem we calculate the inference time of FCNet and RNN by summing up the inference time of all FC layers.

For the CONV layer (R, C, M, N, K), the memory access under the most profitable tiling scheduling is MA_{CONV}, which has been discussed in Section 5.1. Assuming PE_{CONV} PEs are assigned for CONV layer, the inference time of CONV layer is

$$T_{CONV} = (R \cdot C \cdot M \cdot N \cdot K^2)/(PE_{ConvNet} \cdot f). \quad (27)$$

Hence, the bandwidth demand of the CONV layer is

$$BW_{CONV} = MA_{CONV}/T_{CONV.} \quad (28)$$

Assuming $PE_{FCNet/RNN}$ PEs are assigned for the sparse FC/RNN layer, the inference time is

$$T_{FC} = (M \cdot N \cdot SPA)/(PE_{FCNet/RNN} \cdot f). \quad (29)$$

Hence, the bandwidth demand of the FC layer is

$$BW_{FC} = MA_{FC}/T_{FC.} \quad (30)$$

At the time of t, CONV layer $CONV(t)$ and fully connected layer $FC(t)$ are simultaneously accelerated by the PE array. The transient bandwidth BW_t is the sum of the $BW_{CONV(t)}$ and $BW_{FC(t)}$.

To reduce the peak bandwidth in hybrid-NN acceleration, we formulate the optimization problem as follows:

$$min \quad T_{Hybrid-NN} = max(T_{ConvNet}, T_{FCNet} + T_{RNN}) \quad (31)$$

$$min \quad max\{BW_t | 0 \leq t \leq T_{Hybrid-NN}\} \quad (32)$$

s.t.

$$PE_{ConvNet} + PE_{FCNet/RNN} \leq PE \quad (33)$$

$$SPM_{ConvNet} + SPM_{FCNet/RNN} \leq SPM, \quad (34)$$

where $T_{Hybrid-NN}$ is the inference time of the whole hybrid-NN and $T_{ConvNet}/T_{FCNet}/T_{RNN}/$ are inference times for ConvNet/FCNet/RNN. They are

$$T_{ConvNet} = \sum_{l=1}^{L_{ConvNet}} T_{CONV}[l] \quad (35)$$

IEEE Proof

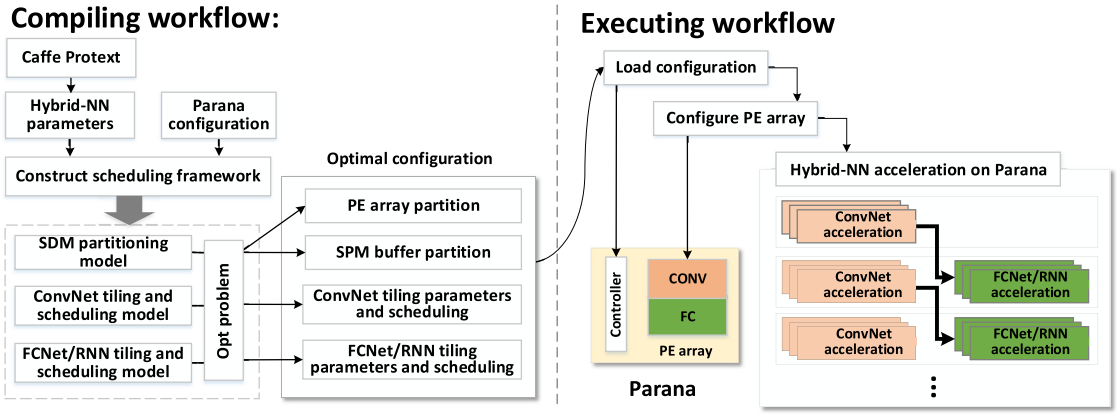


Fig. 10. Compiler workflow and execution workflow.

$$T_{FCNet} = \sum_{l=1}^{L_{FCNet}} T_{FC}[l] \tag{36}$$

$$T_{RNN} = \sum_{l=1}^{L_{RNN}} T_{FC}[l]. \tag{37}$$

Because ConvNet and FCNet/RNN are accelerated in parallel, the inference time of the hybrid-NN is the larger one between them. In the optimization problem, the first goal is the performance optimization in Equation (31) which minimizes the inference time of the whole hybrid-NN. The second goal is to minimize the maximum bandwidth demands in hybrid-NN acceleration.

This optimization problem needs to be jointly resolved with the optimization problems of tiling and scheduling for CONV and FC in Section 5.1. By resolving this optimization problem, we get the best PE array partitioning and SPM buffer allocation for CONV layers and FC layers.

5.3 Workflow of Parana

Fig. 10 summaries the workflow for the compiler and execution of Parana. In compiling workflow, the compiler optimizes the resource partitioning (e.g., PE array and SPM buffer) and the scheduling for ConvNet, FCNet, and RNN. It loads the parameters of hybrid-NN (converted from Caffe protext) and Parana configuration as input to construct the scheduling framework for the most profitable resources partition and execution scheduling. The key of scheduling framework are three models: resources partitioning model for SDM, tiling and scheduling model for ConvNet, and tiling and scheduling model for FCNet. The SDM partitioning model is used to guide the partitioning of computing and memory resources in Parana. With assigned memory resources, the tiling and scheduling models for ConvNet, FCNet, and RNN are exploited to maximize usage of assigned memory and data reuse in each neural network. With the scheduling framework, the compiler produces the most profitable configuration for hybrid-NN acceleration on Parana.

In execution workflow, Parana first loads the most profitable configuration. Then, Parana controller uses the partitioning configuration to configure the partitioning in the PE array and the function of each PE. The controller also manages the memory assignment of the SPM buffer with the SPM buffer partitioning configuration. The scheduling configuration is used to control the dataflow between NPUs and 3D DRAM.

6 Evaluation

In this section, we present the experimental setup and the evaluation results on thermal, performance, and energy improvement.

6.1 Experimental Setup

Benchmarks. We select three representative neural networks, AlexNet [1], VGGNet-16 [10], and LRCN [3], as the hybrid-NN benchmarks in our evaluation. The three benchmarks represent various mixes of neural network layers: 1) VGGNet-16 is CONV computation dominant, 2) LRCN is FC computation dominant, 3) AlexNet has similar CONV and FC computation. Table 2 lists the network structures of our benchmarks, along with the sparsities of FCNet and RNN.

Simulation Framework. We evaluate the temperature of 3D DRAM in the "3D + 2.5D" integration system on a simulator, which integrates a cycle-accurate NPU + HBM simulator, a power model from CACTI-3DD [22], and a thermal model from HotSpot 6.0 [23]. The configuration of 3D DRAM follows the High-Bandwidth-Memory stacked memory [64]. The DRAM controller adapts the open-page policy, because of the high row buffers hit rate. CACTI-3DD is used to generate the detail energy costs of HBM access. The NPU + HBM simulator takes these energy costs and other parameters of the accelerator, HBM and hybrid-NN to produce the power traces for all modules in the 3D DRAM. HotSpot 6.0 analyzes the temperature with power traces as input.

Table 3 lists the configuration of the simulation. In the configuration, Parana works at 1 GHz and the size of PE array is 32×32. It is the maximum size which can be supported by the peak bandwidth of 3D memory [64], when Parana is used to accelerate memory intensive FCNet and RNN. Each PE uses 16-bit fixed-point arithmetic units. The quantized representation of input and output include: 1 bit signed bit, 7 bits integer, and 8 bits fractional parts. The representation of weight includes: 1 bit signed bit and 15 bits fractional parts.

TABLE 2
Description of the Benchmarks

Hybrid-NNs	ConvNet+FCNet+RNN	Spa-FCNet	Spa-RNN
AlexNet [1]	5+3+-	10.12%	-
VggNet-16 [10]	16+3+-	4.63%	-
LRCN [3]	5+3+3	10.12%	11.2%

IEEE Proof

TABLE 3
Configurations of the Simulation

Parana	
Frequency	1 GHz
PE Array Size	32 × 32
MAC Precision	16-bit Fixed Point
# of Banks in SPM Buffer	64
SPM Buffer Size	256 KB
Each Cache Buffer Size	1 KB
HBM [64]	
Process Technology	29 nm DRAM process
Capacity	8 Gb
Chip Size	5.10 mm × 6.91 mm
# of Stack	4 memory dies + 1 logic die
TSV IO	1024
Peak Bandwidth	128 GB/s
Supply Voltages	VDD = 1.2 V, VPP = 2.5 V
Energy [22] and Thermal Parameters	
Activation Energy	3.65 nJ
Read/Write Energy	10.11 nJ
Precharge energy	3.44 nJ
TSV Energy	0.57 nJ
Logic die Energy	18.52 nJ
Ambient Temperature	318.15 (Kelvin)

TABLE 4
Features of Each NPU in Comparison

NPUs	SDM	Sparse	U-SPM	Scheduling
Neurocube [13]	No	No	No	No
EIE [41]	No	Yes	No	No
Eyeriss [8]	No	No	Yes	R-stationary
TETRIS [14]	No	No	Yes	IW, OW, or IO
Parana + TDM	No	Yes	Yes	OPT
Parana	Yes	Yes	Yes	OPT

The unified SPM buffer includes 64 banks and each bank provides one 16 bits read/write port. Unless otherwise specified, the default SPM buffer size is 256 KB.

State-of-the-Art NPU Baselines.We compare our design with four state-of-the-art NPUs, including Neurocube [13], EIE [41], Eyeriss [8], and TETRIS [14], which are representative NPU designs. Table 4 summarizes the features of each NPU. In particular, TETRIS [14] supports three scheduling schemes, including output bypassing (IW), input bypassing (OW), and weight bypassing (IO); we select the one that generates the lowest number of memory accesses.

To make this a fair comparison, we analyze key schemes adopted by state-of-the-art NPUs and implement each of the schemes in our simulator. We perform our evaluation with various schemes under the same chip configuration listed in Table 3.

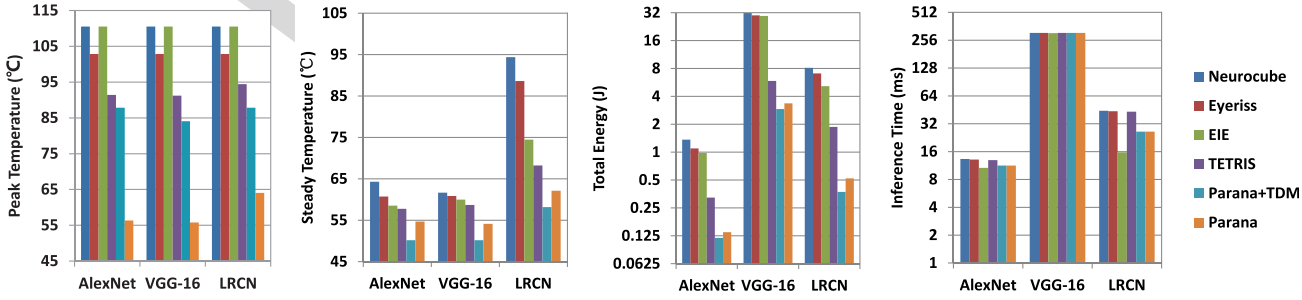
6.2 Chip Area and Power

We implement our Parana architecture design with the configuration in Table 3 and synthesize our design in Taiwan Semiconductor Manufacturing Company (TSMC) 28 nm high performance compact mobile computing (HPC) technology. For a high working frequency, the function of each PE is implemented with at most six-stage pipelines, where local memory reading takes two stages, multiplier takes two stages, adder takes one stage and local memory writing takes one stage. From the synthesized results, we observe that the delay of the critical path in PE array is 0.96 ns. The delay of the critical path in 256 KB SPM with 64 banks is 0.85 ns. Therefore, the highest frequency of Parana is 1 GHz. Table 5 breakdown power and area of Parana with key components under $0.9V$ working voltage. The results show that the most power and area are dominated by the PE array, which takes over 85.5 percent power and 89.8 percent area. Our NPU has a total power of 4.9 W. The power is much lower than the power of the 3D DRAM, which is 13.6 W as calculated using CACTI-3DD [22]).

6.3 Analysis on Temperature, Energy, and Performance

We evaluate Parana against the simulators of Neurocube, Eyeriss, EIE and TETRIS on all hybrid-NNs listed in Table 2. The required bandwidth (GB/s) for AlexNet, VGG and LRCN are 26.92 ∼ 119.07, 24.01 ∼ 128.00 and 63.16 ∼ 126.53, respectively. Figs. 11a, 11b, 11c, and 11d respectively show the results on peak temperature, steady temperature, energy costs and the inference time.

As shown in Figs. 11a and 11b, Parana achieves both lower peak temperature and lower steady temperature than Neurocube, Eyeriss, EIE and TETRIS. This is a combined effect of sparsity, SDM and the unified SPM buffer with most profitable tiling and configuration. Sparsity and the unified SPM buffer reduce the number of memory accesses, so that the steady temperature is lowered. SDM optimizes the memory access behavior, so that the peak temperature is further reduced. If Parana employs TDM instead of SDM, the peak temperature would dramatically rise, because TDM cannot



(a) Peak temperature (b) Steady temperature (c) Total energy (d) Inference time

Fig. 11. Peak/steady temperature, energy and inference time of Neurocube [13], Eyeriss [8], EIE [41], TETRIS [14], Parana+TDM, and Parana, evaluated on benchmarks AlexNet [1], VGGNet-16 [10] and LRCN [3].

IEEE Proof

TABLE 5
Power and Area Breakdown of Parana

Components	power (mW)	area (μm^2)
Controller	10.332	245554.9538
SPM buffer	662.592	1544372.4288
Reg cache buffers	9.396	10642.4638
PE array	4037.12	15866265.0882
Total	4719.44	17666834.9346

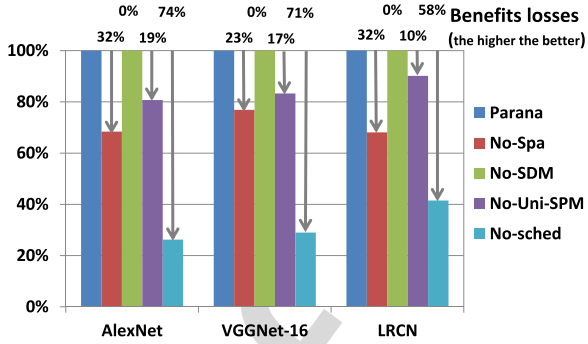


Fig. 12. Benefits losses caused by removing arbitrary scheme in Parana (the higher the better).

evenly distribute memory access over time. Compared to TDM, SDM leads to a bit higher steady temperature, because the granularity of resource partition in SDM cannot be infinitely fine and that brings a little loss.

For the peak temperature in Fig. 11a, Neurocube and EIE both reach the the highest peak temperature, and Eyeriss with the unified SPM buffer only reduces the temperature by 7.66 °C. Parana achieves the lowest peak temperature, which is benefited largely from SDM. This is because that the peak temperature is more related to the maximum bandwidth. SDM merges ConvNet and FCNet/RNN works together and efficiently reduces the maximum bandwidth by 54.16, 72.5 and 26.4 percent for AlexNet, VGGNet-16 and LRCN respectively. In the acceleration of FCNet and RNN, both Neurocube and EIE work with the maximum bandwidth and the maximum bandwidth leads to the peak temperature. Although sparsity supporting in EIE can effectively reduce the total consumed energy as shown in Fig. 11c, EIE still uses the full 3D DRAM bandwidth to load the data for FCNet/RNN. Since the unified SPM buffer can exploit the data reuse in hybrid-NN, Eyeriss with the unified SPM buffer can reduce the maximum bandwidth and reduce the peak temperature. However, the row stationary scheduling in Eyeriss cannot exploit the full capacity of SPM buffer. Parana improves the scheduling with the most profitable tiling and scheduling for each layer and also reduce the peak temperature. Table 6 takes VGGNet-16 as an example to show the most profitable tiling scheduling configurations in Parana.

For the steady temperature in Fig. 11b, Neurocube achieves the highest temperature for all evaluated benchmarks. The reduction of steady temperature in Parana, EIE, and Eyeriss is benefited from the energy reduction. Different with the peak temperature, results in Figs. 11b and 11c show that the trend of steady temperature follows the trend of the energy consumption. Compared to Eyeriss, EIE saves more energy by pruning the useless connections in neural networks. Parana achieves more energy reduction than EIE, because the unified SPM buffer with the most profitable tiling and scheduling reduces memory accesses further. Due to our tiling and scheduling mechanisms, Parana substantially reduces system energy consumption compared with TETRIS.

Fig. 11d shows the average inference time per image for different NPU designs. Because 3D DRAM provides enough bandwidth for PE computation, Neurocube and Eyeriss complete each benchmark with the similar time. EIE and Parana achieve the performance improvement benefited from the sparsity. EIE achieves a higher performance than Parana, because input neurons are also sparse in EIE. In "3D+2.5" integration system, the most energy is consumed by the intensive memory accesses. For an inference of hybrid-NN, the energy cost (or memory access) is more related to the optimization schemes. In contrast, the inference time is largely affected by the number of PEs.

To evaluate the effect of each individual optimization scheme, i.e., SDM, sparsity supporting, unified SPM and data-reuse scheduling, we remove arbitrary scheme from Parana and compare their temperature reduction to Parana. Fig. 12 depicts the results. The baseline is the steady state temperature reduction achieved by Parana with all optimizations, which is set to 100 percent in the comparison. Benefits losses are used to evaluate each optimization and the higher benefits losses mean that the better effect of the removing optimization scheme. The results show that the scheduling mechanism and the sparsity supporting contribute more temperature reduction than SDM and the unified SPM buffer. Removing the scheduling mechanism can bring in the benefits losses over 50 percent. Removing SDM does not impact the steady state temperature, because SDM is designed to reduce peak temperature.

6.4 Peak Temperature under SDM and TDM

Fig. 13 depicts the peak temperature for each layer of the three benchmarks under temporal-division-mapping and spatial-

TABLE 6
The Most Profitable Tiling and Scheduling for VGG16

CONV	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	FC	17	18	19
Tr	224	224	112	112	56	56	56	56	28	28	28	28	1	1	1	1	Tb	32	32	64
Tc	8	8	8	8	14	14	14	14	28	28	28	28	1	1	1	1	Ti	1	1	1
Tm	64	64	128	128	128	128	128	128	128	128	128	128	76	76	76	76	To	410	410	200
Tn	1	1	1	1	1	1	1	1	1	1	1	1	171	171	171	171	-	-	-	-
OPT-Reuse	OR	IR	OR	OR	OR	OR	OR	OR	OR	OR	OR	OR	WR	WR	WR	WR	-	OR	OR	OR

IR: input reuse, WR: weight reuse, OR: output reuse.

IEEE Proof

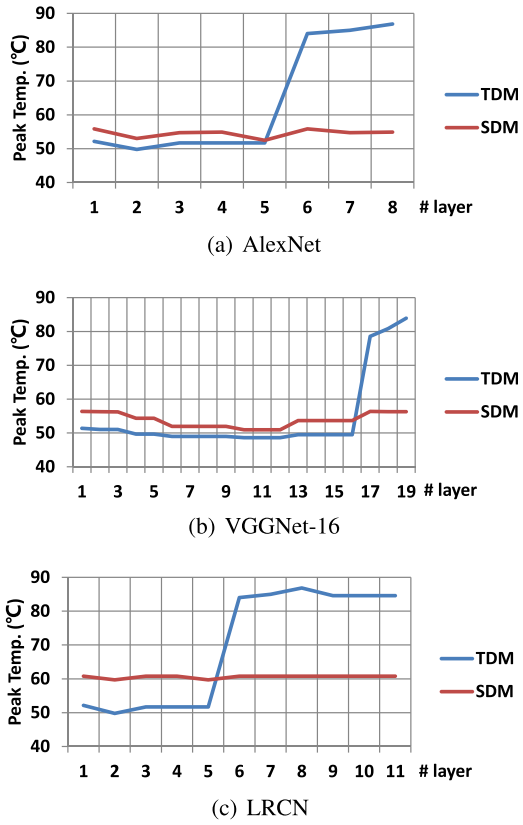


Fig. 13. Temperature of each layer in AlexNet, VGGNet and LRCN on Parana with different mapping strategy (SDM and TDM).

division-mapping. The three curves in TDM show that ConvNet works at a low temperature and FCNet/RNN work at a high temperature. The ConvNets are computation-intensive (e.g., layers [1 ∼ 5] of AlexNet and LRCN, layers [1 ∼ 16] of VGGNet-16 in Fig. 13), hence ConvNet can fully exploit all PEs in Parana with only a small part of the 3D DRAM bandwidth and the low bandwidth leads to a low working temperature. FCNet/RNN are memory intensive. They largely increase the bandwidth demands of Parana and result in a much higher temperature than ConvNet. In contrast, SDM partitions the PE array into two parts, which are respectively used to accelerate ConvNet and FCNet/RNN. Hence, SDM merges the bandwidth demands of ConvNet and FCNet/RNN into a balanced one. For SDM, which may parallelly execute more than one layers, each layer's peak temperature in Fig. 13 represents the maximal temperature in this layer's execution period, and it may include the contribution of other layers. The curves in SDM demonstrate that SDM can produce a lower peak temperature than TDM on all evaluated benchmarks.

6.5 3D Memory Access Analysis on the Most Profitable Data Reuse Scheduling Scheme

Fig. 14 compares the data-reuse scheduling schemes on three benchmarks and three classic layers in VGGNet16 (e.g., convolutional layer 1, 4, and 16): layer 1 represents output dominated layer, layer 4 represents both input and output dominated layer, layer 16 represents the weight dominated layer. In the comparison, the tiling parameters are the most profitable configuration. The results show that the scheduling of no data reuse produces the most memory

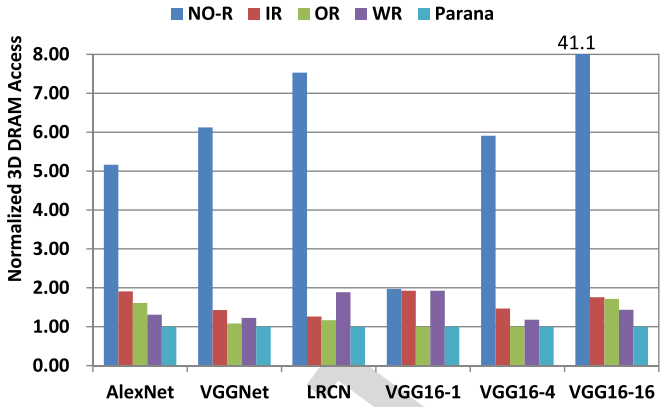
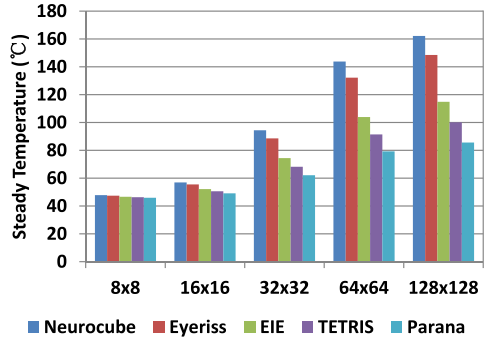
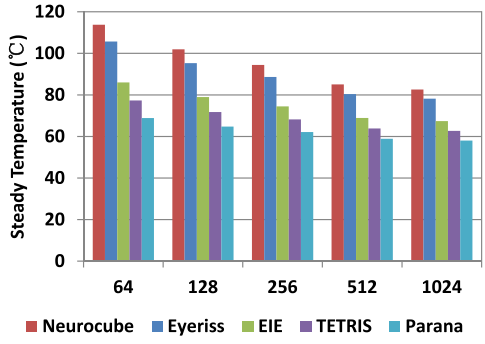


Fig. 14. The 3D DRAM accesses normalized to Parana. (NO-R: no reuse, IR: input reuse, WR: weight reuse, OR: output reuse, Parana: the most profitable reuse scheduling)

accesses and the most profitable scheduling in Parana always achieves the least memory accesses. For some specific layers, input reuse (IR), weight reuse (WR) and partial sum of output reuse (OR) can obtain the least memory accesses as the most profitable scheduling. But for any hybrid-NN, single reusing pattern cannot always achieve the least memory access in all layers. For example, OR is suitable to VGG16-CONV-1 and VGG16-CONV-4, but it is better to adopt weight reuse in VGG16-CONV-16. Hence the results demonstrate that the most profitable scheduling for each layer is better than single reusing pattern for all layers. In "3D + 2.5D" integration system, a small number of memory accesses is still beneficial to reduce the temperature in 3D DRAM.



(a) The number of PEs



(b) The size (KB) of SPM buffer

Fig. 15. Sensitivity analysis on the number of PEs and the size of the unified buffer, evaluated on LRCN.

IEEE Proof

6.6 Sensitivity Study

Fig. 15 depicts the sensitivity analysis of steady temperature on different number of PEs and different size of SPM buffer. The results on different number of PEs show that the steady temperature increases quickly with a larger PE array. A larger PE arrays means more concurrent 3D memory accesses, then the higher temperature is caused by producing the same energy in a smaller period of time. The results on different size of SPM buffer show that a large unified buffer is beneficial to reduce the steady temperature. This is because a larger unified buffer increases the data reusability, then reduces the total 3D memory accesses. Fewer memory accesses produce less energy consumption and finally reduce the temperature. As the size of SPM buffer increasing, the benefits are becoming smaller. In the sensitivity analysis, Parana always achieves the lowest temperature among various NPU designs.

7 Conclusion

This paper proposed a "3D + 2.5D" integration neural processor with 3D-stacked memory for hybrid-NN acceleration. Three key optimization methods are used to tackle the thermal problem in 3D DRAM: spatial division mapping, sparsity supporting, and buffer-oriented tiling/scheduling. Parana exploits sparsity, unified SPM and tiling/scheduling to minimize 3D DRAM accesses and reduce the steady temperature in 3D DRAM. The spatial division mapping achieves a balanced bandwidth and reduces the peak temperature by accelerating convolution and sparse fully connected layers in parallel. The experimental results show that the proposed neural processor reduces the peak temperature by up to 54.72 °C and the steady temperature by up to 32.27 °C over state-of-the-art accelerators with 3D memory.

Acknowledgments

This work was supported in part by the NSFC under Grant 61774094, the China Major S&T Project (No. 2018ZX01031101-002) and the NSF under Grant 1718158/1725447/1719160.

References

[1] 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, 2012.
[2] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra, "Pathnet: Evolution channels gradient descent in super neural networks," *arXiv preprint arXiv:1701.08734*, 2017.
[3] J. Donahue, L. Anne Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell, "Long-term recurrent convolutional networks for visual recognition and description," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 2625–2634.
[4] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *ACM Multimedia*, 2014, pp. 675–678.
[5] S. Han, J. Pool, J. Tran, and W. J. Dally, "Learning both weights and connections for efficient neural networks," *Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.
[6] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," *Int. Conf. Learn. representations*, 2016.
[7] T. Chen, Z. Du, N. Sun, J. Wang, C. Wu, Y. Chen, and O. Temam, "Diannao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," *ACM Sigplan Notices*, vol. 49, pp. 269–284, 2014.
[8] Y.-H. Chen, T. Krishna, J. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," in *Proc. IEEE Int. Solid-State Circuits Conf.*, 2016, pp. 262–263.
[9] Z. Chen, F. Zhenman, Z. Peipei, P. Peichen, and C. Jason, "Caffeine: Towards uniformed representation and acceleration for deep convolutional neural networks," in *Proc. Int. Conf. Comput. Aided Des.*, 2016, Art. no. 12.
[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. conf. Learn. Representations*, vol. abs/1409.1556, 2015.
[11] L. Kaiser, A. N. Gomez, N. Shazeer, A. Vaswani, N. Parmar, L. Jones, and J. Uszkoreit, "One model to learn them all," *arXiv preprint arXiv:1706.05137*, 2017.
[12] G. H. Loh, "3d-stacked memory architectures for multi-core processors," in *Proc. 35th Annu. Int. Symp. Comput. Archit.*, 2008, pp. 453–464.
[13] 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.*, Jun. 2016, pp. 380–392.
[14] M. Gao, J. Pu, X. Yang, M. Horowitz, and C. Kozyrakis, "Tetris: Scalable and efficient neural network acceleration with 3d memory," in *Proc. 22nd ACM Int. Conf. Architectural Support Program. Lang. Operating Syst.*, 2017, pp. 751–764.
[15] Y. Xie and J. Zhao, "Die-stacking architecture," in *Synthesis Lectures on Computer Architecture*, vol. 2. San Rafael, CA, USA: Morgan & Claypool Publishers, Jun. 2015, pp. 1–127.
[16] AMD, "Amd's next generation gpu and memory architecture - hot chips," in *Proc. Hot Chips Symp.*, 2015, pp. 1–26.
[17] NVIDIA, "Pascal gpu architecture." 2018. [Online]. Available: https://www.nvidia.com/en-us/data-center/pascal-gpu-architecture/
[18] Intel, "Knights landing (knl): 2nd generation intel xeon phi processor," in *Proc. Hot Chips*, 2015, pp. 1–24.
[19] Intel, "Intel Nervana Neural Network Processor" 2018. [Online]. Available: https://ai.intel.com/intel-nervana-neural-network-processor
[20] A. Das, A. Kumar, and B. Veeravalli, "Temperature aware energy-reliability trade-offs for mapping of throughput-constrained applications on multimedia mpsocs," in *Proc. Conf. Des., Autom. Test Eur.*, 2014, Art. no. 102.
[21] I. Thomson, "Meet TPU 3.0: Google teases world with latest math coprocessor for AI," 2018. [Online]. Available: https://www.theregister.co.uk/2018/05/09/google_tpu_3/
[22] K. Chen, S. Li, N. Muralimanohar, J. H. Ahn, J. B. Brockman, and N. P. Jouppi, "Cacti-3dd: Architecture-level modeling for 3d die-stacked dram main memory," in *Proc. Des. Autom. Test Eur. Conf. Exhib.*, Mar. 2012, pp. 33–38.
[23] R. Zhang, M. R. Stan, and K. Skadron, "Hotspot 6.0: Validation, acceleration and extension," University of Virginia, Charlottesville, VA, USA, Tech. Rep. CS-2015-04, 2015.
[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[25] S. Chakradhar, M. Sankaradas, V. Jakkula, and S. Cadambi, "A dynamically configurable coprocessor for convolutional neural networks," in *Proc. ACM SIGARCH Comput. Archit. News*, 2010, pp. 247–257.
[26] M. Peemen, A. A. Setio, B. Mesman, and H. Corporaal, "Memory-centric accelerator design for convolutional neural networks," in *Proc. IEEE 31st Int. Conf. Comput. Des.*, 2013, pp. 13–19.
[27] F. Conti and L. Benini, "A ultra-low-energy convolution engine for fast brain-inspired vision in multicore clusters," in *Proc. Des. Autom. Test Eur. Conf. Exhib.*, 2015, pp. 683–688.
[28] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun, et al., "Dadiannao: A machine-learning supercomputer," in *Proc. 47th Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2014, pp. 609–622.
[29] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam, "Shidiannao: Shifting vision processing closer to the sensor," *ACM SIGARCH Comput. Archit. News*, vol. 43, pp. 92–104, 2015.
[30] S. Liu, Z. Du, J. Tao, D. Han, T. Luo, Y. Xie, Y. Chen, and T. Chen, "Cambricon: An instruction set architecture for neural networks," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit.*, Jun. 2016, pp. 393–405.
[31] L. Song, Y. Wang, Y. Han, X. Zhao, B. Liu, and X. Li, "C-brain: A deep learning accelerator that tames the diversity of cnns through adaptive data-level parallelization," in *Proc. 53nd ACM/EDAC/IEEE Des. Autom. Conf.*, Jun. 2016, pp. 1–6.

IEEE Proof

[32] B. Reagen, P. Whatmough, R. Adolf, S. Rama, H. Lee, S. K. Lee, J. M. Hernandez-Lobato , G. Y. Wei, and D. Brooks, "Minerva: Enabling low-power, highly-accurate deep neural network accelerators," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit.*, 2016, pp. 267–278.

[33] S. Park, S. Choi, J. Lee, M. Kim, J. Park, and H. J. Yoo, "14.1 a 126.1mw real-time natural ui/ux processor with embedded deep-learning core for low-power smart glasses," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Jan. 2016, pp. 254–255.

[34] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. S. Modha, "A digital neurosynaptic core using embedded crossbar memory with 45 pj per spike in 45 nm," in *Proc. IEEE Custom Integr. Circuits Conf.*, Sep. 2011, pp. 1–4.

[35] J. S. Seo, B. Brezzo, Y. Liu, B. D. Parker, S. K. Esser, R. K. Montoye, B. Rajendran, J. A. Tierno, L. Chang, D. S. Modha, and D. J. Friedman, "A 45nm cmos neuromorphic chip with a scalable architecture for learning in networks of spiking neurons," in *Proc. IEEE Custom Integr. Circuits Conf.*, Sep. 2011, pp. 1–4.

[36] J. Sim, J. S. Park, M. Kim, D. Bae, Y. Choi, and L. S. Kim, "14.6 a 1.42tops/w deep convolutional neural network recognition processor for intelligent ioe systems," in *Proc. IEEE Int. Solid-State Circuits Conf.*, Jan. 2016, pp. 264–265.

[37] 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.*, 2016, pp. 1–13.

[38] A. Rahman, J. Lee, and K. Choi, "Efficient fpga acceleration of convolutional neural networks using logical-3d compute array," in *Proc. Des. Autom. Test Eur. Conf. Exhib.*, 2016, pp. 1393–1398.

[39] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing fpga-based accelerator design for deep convolutional neural networks," in *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*, 2015, pp. 161–170.

[40] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song, et al., "Going deeper with embedded fpga platform for convolutional neural network," in *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*, 2016, pp. 26–35.

[41] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: Efficient inference engine on compressed deep neural network," in *Proc. ACM/IEEE 43rd Ann. Int. Symp. Comput. Architecture*, Seoul, 2016, pp. 243–254.

[42] C. Farabet, C. Poulet, J. Y. Han, and Y. LeCun, "Cnp: An fpga-based processor for convolutional networks," in *Proc. Int. Conf. Field Program. Logic Appl.*, 2009, pp. 32–37.

[43] M. Sankaradas, V. Jakkula, S. Cadambi, S. Chakradhar, I. Durdanovic, E. Cosatto, and H. P. Graf, "A massively parallel coprocessor for convolutional neural networks," in *Proc. 20th IEEE Int. Conf. Appl.-Specific Syst. Archit. Processors*, Jul. 2009, pp. 53–60.

[44] V. Gokhale, J. Jin, A. Dundar, B. Martini, and E. Culurciello, "A 240 g-ops/s mobile coprocessor for deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2014, pp. 696–701.

[45] H. Li, X. Fan, L. Jiao, W. Cao, X. Zhou, and L. Wang, "A high performance fpga-based accelerator for large-scale convolutional neural networks," in *Proc. 26th Int. Conf. Field Program. Logic Appl.*, Aug. 2016, pp. 1–9.

[46] Y. Shen, M. Ferdman, and P. Milder, "Overcoming resource underutilization in spatial cnn accelerators," in *Proc. 26th Int. Conf. Field Program. Logic Appl.*, 2016, pp. 1–4.

[47] C. Wang, L. Gong, Q. Yu, X. Li, Y. Xie, and X. Zhou, "Dlau: A scalable deep learning accelerator unit on fpga," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 36, no. 3, pp. 513–517, Mar. 2017.

[48] J. Ouyang, "Sda: Software-defined accelerator for large-scale deep learning system," in *Proc. Int. Symp. VLSI Des. Autom. Test*, Apr. 2016, pp. 1–1.

[49] L. He, Y. Luo, and Y. Cao, "Accelerator of stacked convolutional independent subspace analysis for deep learning-based action recognition," in *Proc. IEEE 22nd Annu. Int. Symp. Field-Programmable Custom Comput. Mach.*, May 2014, pp. 104–104.

[50] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "Envision: A 0.26-to-10tops/w subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm fdsoi," in *Proc. IEEE Int. Solid-State Circuits Conf.*, 2017, pp. 246–248.

[51] M. L. Campbell, S. T. Toborg, and S. L. Taylor, "3d wafer stack neurocomputing," in *Proc. IEEE Int. Conf. Wafer Scale Integr.*, 1993, pp. 67–74.

[52] I. T. Wang, Y. C. Lin, Y. F. Wang, C. W. Hsu, and T. H. Hou, "3d synaptic architecture with ultralow sub-10 fj energy per spike for neuromorphic computation," *Electron Devices Meeting .iedm.technical Digest.international*, vol. 2015, pp. 28.5.1–28.5.4, 2014.

[53] JEDEC, "JEDEC 3D ICs Interface," 2016. [Online]. Available: https://www.jedec.org/category/technology-focus-area/3d-ics

[54] T. Kgil, S. D'Souza, A. Saidi, N. Binkert, R. Dreslinski, T. Mudge, S. Reinhardt, and K. Flautner, "Picoserver: Using 3d stacking technology to enable a compact energy efficient chip multiprocessor," in *Proc. 12th Int. Conf. Architectural Support Programming Lang. Operating Syst.*, 2006, pp. 117–128.

[55] G. L. Loi, B. Agrawal, N. Srivastava, S.-C. Lin, T. Sherwood, and K. Banerjee, "A thermally-aware performance analysis of vertically integrated (3-d) processor-memory hierarchy," in *Proc. 43rd Annu. Des. Autom. Conf.*, 2006, pp. 991–996.

[56] G. H. Loh, "3d-stacked memory architectures for multi-core processors," in *Proc. 35th Annu. Int. Symp. Comput. Archit.*, 2008, pp. 453–464.

[57] W. Computing, "Wave Computing's Native Dataflow Technology," 2018. [Online]. Available: https://wavecomp.ai/technology

[58] M. Alwani, H. Chen, M. Ferdman, and P. Milder, "Fused-layer cnn accelerators," in *Proc. 49th Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2016, pp. 1–12.

[59] N. Suda, V. Chandra, G. Dasika, A. Mohanty, Y. Ma, S. B. K. Vrudhula, J. Seo, and Y. Cao, "Throughput-optimized opencl-based FPGA accelerator for large-scale convolutional neural networks," in *Proc. ACM/SIGDA Int. Symp. Field-Programmable Gate Arrays*, 2016, pp. 16–25.

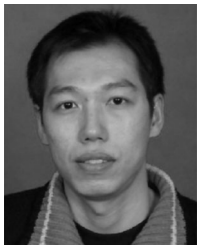
[60] S. Zhang, Z. Du, L. Zhang, H. Lan, S. Liu, L. Li, Q. Guo, T. Chen, and Y. Chen, "Cambricon-x: An accelerator for sparse neural networks," in *Proc. IEEE/ACM Int. Symp. Microarchitecture*, 2016, pp. 1–12.

[61] J. Zhao, X. Dong, and Y. Xie, "Cost-aware three-dimensional (3D) many-core multiprocessor design," in *Proc. 47th Des. Autom. Conf.*, 2010, pp. 126–131.

[62] JEDEC, *DDR3 SDRAM Specification, JESD79–3E*, Jul. 2010, https://www.jedec.org/standards-documents/docs/jesd-79-3d.

[63] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu, "RAIDR: Retention-aware intelligent DRAM refresh," in *Proc. 39th Annu. Int. Symp. Comput. Archit.*, 2012, pp. 1–12.

[64] D. U. Lee, K. W. Kim, K. W. Kim, H. Kim, J. Y. Kim, Y. J. Park, J. H. Kim, D. S. Kim, H. B. Park, J. W. Shin, J. H. Cho, K. H. Kwon, M. J. Kim, J. Lee, K. W. Park, B. Chung, and S. Hong, "25.2 a 1.2 v 8 gb 8-channel 128 gb/s high-bandwidth memory (HBM) stacked dram with effective microbump i/o test methods using 29 nm process and tsv," in *Proc. IEEE Int. Solid-State Circuits Conf. Digest Tech. Papers*, Feb. 2014, pp. 432–433.

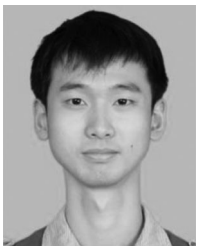


Shouyi Yin received the BS, MS, and PhD degrees in electronic engineering from Tsinghua University, China, in 2000, 2002, and 2005, respectively. He has worked with Imperial College London as a research associate. Currently he is with the Institute of Microelectronics, Tsinghua University as an associate professor. His research interests include SoC design, reconfigurable computing and mobile computing. He has published more than 40 refereed papers, and served as TPC member or reviewer for the international key conferences and leading journals.



Shibin Tang received the BS degree in computer science and technology from Shandong University, in 2008, the PhD degree in computer architecture from the Institute of Computing Technology, Chinese Academy of Sciences, in 2014. He is currently a postdoctoral researcher with the Institute of Microelectronics, Tsinghua University. His research interests include computer architecture, cache coherence, on chip memory system, deep learning, and neural network acceleration.

IEEE Proof



Xinhan Lin received the BS degree in computer science and technology from Sichuan University, Chengdu, China, in 2010, and the MS degree from the Institute of Microelectronics, Tsinghua University, Beijing, China, in 2016. Currently he is working toward the PhD degree in the Institute of Microelectronics, Tsinghua University, Beijing, China. His research interests include reconfigurable computing and optimization of compiler for reconfigurable computing.



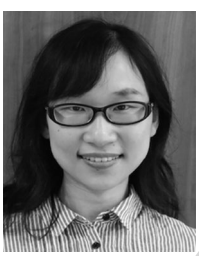
Peng Ouyang received the BS degree in electronic and information technology from Center South University, Changsha, Hunan China, in 2008, and the PhD degree in electronic science and technology from Tsinghua University, in 2014. During 2015 to 2017, he worked as a post doctor with the School of Information of Tsinghua University. Now he is an assistant professor with Bei Hang University. His research interests include the computer vision, deep learning and reconfigurable computing.



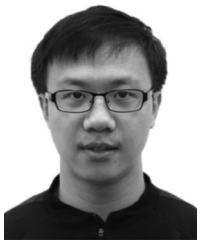
Fengbin Tu received the BS degree in electronic science and technology from Beijing University of Posts and Telecommunications, Beijing, China, in 2013. He is currently working toward the PhD degree in the Institute of Microelectronics, Tsinghua University, Beijing, China. His current research interests include deep learning, computer architecture, VLSI Design, approximate computing and reconfigurable computing.



Leibo Liu received the BS degree in electronic engineering from Tsinghua University, Beijing, China, in 1999, and the PhD degree in the Institute of Microelectronics, Tsinghua University, in 2004. He now serves as an associate professor with the Institute of Microelectronics, Tsinghua University. His research interests include reconfigurable computing, mobile computing and VLSI DSP.



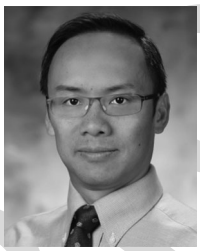
Jishen Zhao received the PhD degree from Pennsylvania State University, in 2014. She is an assistant professor in the Department of Computer Engineering, University of California, San Diego. Her research spans and stretches the boundary between computer architecture and system software, with a particular emphasis on memory and storage systems, domain-specific acceleration, and high-performance computing. Her research is driven by both emerging device/circuit technologies (e.g., 3D integration, nonvolatile memories) and modern applications (e.g., big-data analytics, machine learning, and scientific computing). She is a member of the IEEE and ACM.



Cong Xu received the PhD degree in computer science and engineering from Pennsylvania State University, in 2015, and worked as a postdoc with UCSB before joining Hewlett Packard Labs as a research scientist, in 2016. His research interests include deep learning acceleration, high performance computing, and distributed file system. He has published more than 30 papers on non-volatile memory and deep learning acceleration. He is a member of the IEEE and ACM.



Shuangchen Li received the PhD degree in ECE from the University of California, Santa Barbara, in 2018. He works on memory related computer architecture, with emphasis on processing-in-memory architectures, emerging non-volatile technologies, and deep learning accelerators.



Yuan Xie received the PhD degree from Princeton University, and has worked with IBM, AMD, and Penn State, before he joined UCSB, in 2014. He is a professor with UCSB. His research interests include computer architecture, design automation, and VLSI. He is a fellow of the IEEE.



Shaojun Wei received the PhD degree from Faulte Polytechnique de Mons, Belguim, in 1991. He became a professor with the Institute of Microelectronics of Tsinghua University, in 1995. He is senior member of the Chinese Institute of Electronics. His main research interests include VLSI SoC design, EDA methodology, and ASIC design.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.

IEEE PROOF