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†, Cong Xu†, Shuangcheng Li§, Yuan Xie*, Shaojun Wei∗

* Institute of Microelectronics, Tsinghua University, Beijing 100084, China
† University of California, Santa Cruz
§ University of California, Santa Barbara

Abstract—The 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). Such 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°C and the steady temperature by up to 32.27°C 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 [1–4]. 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[1–9], 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 and memory accesses. The acceleration of hybrid-NN requires a large on-chip

![Figure 1. Examples of hybrid neural networks (Hybrid-NN): AlexNet [1] for image classification and LRCN [3] for image understanding.](image-url)
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 NPs [15]. A higher temperature increases the cooling cost [15] and reduces the life-time of 3D memory [20]. For example, Google’s TPU3 used 3D memory to help address the memory bandwidth problem, but has 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% execution phase takes 80% 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 accelerator. 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 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, 25–49] 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 scratchpad-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/TiO2/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, 54–56], 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 [16–18], 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 subsection.

**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, 29–31, 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.

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 gate indicators are iteratively calculated by such kind of operations:

\[
\text{Gate}_t = \delta(W_t \cdot X_t + W_f \cdot F_t + W_o \cdot O_t + \text{bias}),
\]

where, in iteration \(t\), \(\text{Gate}_t\) denotes the input gate, forget gate or output gate. \(X_t\) and \(W_t\) 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 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 (\(LM_1 \sim LM_5\)). 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.

### 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 he 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.

![Figure 7. An example of sparse FC acceleration on PE.](image)

(a) Original dense representation of sparse FC

In sparse FC mode, each PE accelerates the sparse FC as depicted in Fig. 7. Fig. 7(a) shows the original representation of sparse FC. Fig. 7(b) 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.

![Figure 8. An example of CONV acceleration on PE.](image)

(b) Accelerated by one PE

**Figure 7. An example of sparse FC acceleration on PE.**

PE saves the partial sums in LM5 to implement accumulation with future results. The CONV computation in each PE is 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.

![Figure 9.](image)

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. 7(b). 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 multiplies 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
Figure 9. The dataflows of CONV and sparse FC in SDM

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 \(P_{FC}\), the bandwidth demand of sparse FC is

\[
BW_{FC} = \frac{M \cdot N \cdot SPA \cdot DL}{(M \cdot N \cdot SPA) / (PE_{FCNet/RNN})} = \frac{DL \cdot f \cdot PE_{FCNet/RNN}}{PE_{FCNet/RNN}}
\]

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 ConvNet is

\[
T_{CONV} = \frac{(R \cdot C \cdot M \cdot N \cdot K^2)}{(PE_{ConvNet} \cdot f)}
\]

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

\[
BW_{Input} = \frac{R \cdot C \cdot N \cdot DL}{T} = \frac{DL \cdot f \cdot PE_{ConvNet}}{PE_{ConvNet}}
\]

\[
BW_{Output} = \frac{R \cdot C \cdot M \cdot DL}{T} = \frac{DL \cdot f \cdot PE_{ConvNet}}{PE_{ConvNet}}
\]

\[
BW_{Weight} = \frac{M \cdot N \cdot K^2 \cdot DL}{T} = \frac{DL \cdot f \cdot PE_{ConvNet}}{PE_{ConvNet}}
\]

Hence, the bandwidth demand of CONV is

\[
BW_{CONV} = BW_{Input} + BW_{Output} + BW_{Weight} = \left(1 - \frac{DL \cdot f \cdot PE_{ConvNet}}{DL \cdot f \cdot PE_{ConvNet}}\right)
\]

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 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 \((T_r, T_c, T_m, T_w)\) to ensure that the unified SPM buffer can save all the 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 the total number of 3D memory accesses is the product of SPM buffer size. The total amount of 3D memory access in ConvNet is formula

\[
MA_{\text{CONV}} = \min(MA_{\text{CONV,wr}},MA_{\text{CONV,or}},MA_{\text{CONV,ir}}) - \min MA_{\text{CONV}} = \min(MA_{\text{CONV,wr}},MA_{\text{CONV,or}},MA_{\text{CONV,ir}})
\]

s.t.

\[
SPM_{\text{CONV,wr}} + SPM_{\text{CONV,or}} + SPM_{\text{CONV,ir}} \leq SPM_{\text{ConvNet}}\]

By enumerating all possible tiling under the three reuse patterns, the Parana compiler 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 \(MA_{\text{CONV}}\) 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

\[
SPM_{\text{FC,ir}} = T_i \cdot T_b
\]

\[
SPM_{\text{FC,or}} = T_o \cdot T_b
\]

\[
SPM_{\text{FC,wr}} = T_i \cdot T_o \cdot \text{SPA} \cdot 3
\]

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

\[
R_{\text{PT}} = \frac{B \cdot I \cdot T_i \cdot O \cdot T_o}{T_o}
\]

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

\[
MA_{\text{FC,ir}} = I \cdot (2 \cdot SPM_{\text{FC,ir}} + SPM_{\text{FC,wr}}) \cdot R_{\text{PT}}
\]

\[
MA_{\text{FC,or}} = O \cdot (2 \cdot SPM_{\text{FC,or}} + SPM_{\text{FC,wr}}) \cdot R_{\text{PT}}
\]

\[
MA_{\text{FC,wr}} = I \cdot O \cdot \text{SPA} \cdot 3 \cdot SPM_{\text{FC,wr}} \cdot R_{\text{PT}}
\]

The optimization problem for the tiling and scheduling of FC is

\[
\min MA_{\text{FC}} = \min(MA_{\text{FC,ir}},MA_{\text{FC,or}},MA_{\text{FC,wr}})
\]

s.t.

\[
SPM_{\text{FC,ir}} + SPM_{\text{FC,or}} + SPM_{\text{FC,wr}} \leq SPM_{\text{FCNet/RNN}}\]

With \(MA_{\text{FC}}\) of all full-connection layers, the compiler 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 \(MA_{\text{CONV}}\), which has been discussed in Section 5.1. Assuming \(PE_{\text{CONV}}\) PEs are assigned for CONV layer, the inference time of CONV layer is

\[
T_{\text{CONV}} = (R \cdot C \cdot M \cdot N \cdot K^2) / (PE_{\text{CONV}} \cdot f)
\]

Hence, the bandwidth demand of the CONV layer is

\[
BW_{\text{CONV}} = MA_{\text{CONV}} / T_{\text{CONV}}
\]

Assuming \(PE_{\text{FCNet/RNN}}\) PEs are assigned for the sparse FC/RNN layer, the inference time is

\[
T_{\text{FC}} = (M \cdot N \cdot \text{SPA}) / (PE_{\text{FCNet/RNN}} \cdot f)
\]

Hence, the bandwidth demand of the FC layer is

\[
BW_{\text{FC}} = MA_{\text{FC}} / T_{\text{FC}}
\]

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

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

\[
\min T_{\text{Hybrid-NN}} = \max(T_{\text{CONV}(t)},T_{\text{FC}(t)} + T_{\text{RNN}})
\]

\[
\max\{BW_{\text{t}}||0 \leq t \leq T_{\text{Hybrid-NN}}\}
\]

s.t.

\[
PE_{\text{FCNet}} + PE_{\text{FCNet/RNN}} \leq PE
\]

\[
SPM_{\text{ConvNet}} + SPM_{\text{FCNet/RNN}} \leq SPM
\]

where \(T_{\text{Hybrid-NN}}\) is the inference time of the whole hybrid-NN and \(T_{\text{CONV}(t)} / T_{\text{FC}(t)} / T_{\text{RNN}}\) are inference times for ConvNet/FCNet/RNN. They are

\[
T_{\text{CONV}} = \sum_{t=1}^{L_{\text{CONV}}} T_{\text{CONV}}(t)
\]

\[
T_{\text{FCNet}} = \sum_{t=1}^{L_{\text{FCNet}}} T_{\text{FC}(t)}
\]

\[
T_{\text{RNN}} = \sum_{t=1}^{L_{\text{RNN}}} T_{\text{RNN}}(t)
\]
Figure 10. Compiler workflow and execution workflow.

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, e.g. 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 (HBM) stacked memory [64]. The DRAM controller integrates a cycle-accurate NPU + HBM simulator, a power model and a thermal model from HotSpot 6.0, 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 4
Features of each NPU in comparison

<table>
<thead>
<tr>
<th>NPU</th>
<th>SDM</th>
<th>Sparse</th>
<th>U-SPM</th>
<th>Scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neurocube [13]</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>EIE [41]</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Eyeriss [8]</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>TETRIS [14]</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>IW, OW, or IO</td>
</tr>
<tr>
<td>Parana + TDM</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>OPT</td>
</tr>
<tr>
<td>Parana</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>OPT</td>
</tr>
</tbody>
</table>

Table 3 lists the configuration of the simulation. In the configuration, Parana works at 1GHz 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. 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 256KB.

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) 28nm 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 0.96ns. The delay of the critical path in 256 KB SPM with 64 banks is 0.85 ns. Therefore, the highest frequency of Parana is 1GHz. 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% power and 89.8% area. Our NPU has a total power of 4.9W. The power is much lower than the power of the 3D DRAM, which is 13.6W 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. Fig. 11 (a)∼(d) respectively show the results on peak temperature, steady temperature, energy costs and the inference time.

As shown in Fig. 11(a) and 11(b), 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 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. 11(a), 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% 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.11(c), 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. 11(b), 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 Fig.11(b) and Fig. 11(c) 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.11(d) 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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2018.2858230, IEEE Transactions on Parallel and Distributed Systems
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% 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%. Removing SDM does not impact the steady state temperature, because SDM is designed to reduce peak temperature.

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 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.
As the size of SPM buffer increasing, the benefits are becoming different number of PEs and different size of SPM buffer. The smaller. In the sensitivity analysis, Parana always achieves the lowest temperature among various NPU designs. produce less energy consumption and finally reduce the temperature. reduces the total 3D memory accesses. Fewer memory accesses unified buffer is beneficial to reduce the steady temperature. This 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 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 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 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.

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 array 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, including: spatial division mapping, sparsity supporting, and buffer-oriented tiling/scheduling. Parana exploits sparsity, unified SPM and tiling/scheduling to reduce 3D DRAM accesses, which 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.

References


[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 *2016 53nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, June 2016.
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TPDS.2018.2858230, IEEE Transactions on Parallel and Distributed Systems
Shouyi Yin received the B.S., M.S. and Ph.D. degrees in Electronic Engineering from Tsinghua University, China, in 2000, 2002 and 2005, respectively. He has worked in Imperial College London as a research associate. Currently he is with Institute of Microelectronics at Tsinghua University as an associate professor. His research interests include SoC design, reconfigurable computing and mobile computing. Prof. Yin 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 B.S. degree in Computer Science and Technology from Shandong University, China, in 2010 and the M.S. degree in the Institute of Microelectronics, Tsinghua University, Beijing, China, in 2016. Currently he is working toward the Ph.D. degree in the Institute of Microelectronics, Tsinghua University, Beijing, China. His research interests include computer architecture, cache coherence, on-chip memory system, deep learning, and neural network acceleration.

Xinhan Lin received the B.S. degree in Computer Science and Technology from Sichuan University, Chengdu, China, in 2010 and the M.S. degree in the Institute of Microelectronics, Tsinghua University, Beijing, China, in 2015. He is currently a PhD student at Institute of Microelectronics, Tsinghua University. His research interests include VLSI SoC design, EDA methodology, and ASIC design.

Peng Ouyang received the B.S. degree in electronic and information technology from center south university, Changsha, Hunan China, in 2008. And received the PH.D. degree in electronic science and technology from Tsinghua university in 2014. During 2015-2017, he worked as a post doctor in school of information of Tsinghua University. His research interests include the computer vision, deep learning and reconfigurable computing.

Fengbin Tu received the B.S. degree in electronic science and technology from Beijing University of Posts and Telecommunications, Beijing, China, in 2013. He is currently pursuing the Ph.D. degree with Institute of Microelectronics, Tsinghua University. His research interests include deep learning, computer architectures, reconfigurable computing, and approximate computing.

Leibo Liu received the B.S. degree in electronic engineering from Tsinghua University, Beijing, China, in 1999 and the Ph.D. degree in Institute of Microelectronics, Tsinghua University, in 2004. He now serves as an Associate Professor in Institute of Microelectronics, Tsinghua University. His research interests include Reconfigurable Computing, Mobile Computing and VLSI DSP.

Jishen Zhao is an Assistant Professor in Department of Computer Engineering at University of California, Santa Cruz. His research interests span and stretch the boundary between computer architecture and system software, with a particular emphasis on memory and storage systems, domain-specific acceleration, and high-performance computing. His 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). Before joining UCSC, she was a Research Scientist at HP Labs. She received her Ph.D. from Pennsylvania State University in 2014. She is a member of IEEE and ACM.

Cong Xu received his B.S. degree in microelectronics from Peking University, Beijing, in 2005, and his Ph.D. degree in computer science and engineering from Pennsylvania State University, University Park, PA, in 2015. He is currently a research scientist with Hewlett Packard Labs. 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 ACM.

Shuangchen Li is a Ph.D student from University of California, Santa Barbara. He works on memory related computer architecture, with emphasis on processing-in-memory architectures, emerging non-volatile technologies, and deep learning accelerators.

Yuan Xie is a Professor at UCSB. His research interests include computer architecture, design automation, and VLSI. He received Ph.D. from Princeton University, and has worked for IBM, AMD, and Penn State, before he joined UCSB in 2014. He is a IEEE Fellow.

Shaojun Wei was born in Beijing, China in 1958. He received Ph.D. degree from Faulte Polytechnique de Mons, Belgium, in 1991. He became a professor in Institute of Microelectronics of Tsinghua University in 1995. He is senior member of Chinese Institute of Electronics. His main research interests include VLSI SoC design, EDA methodology, and ASIC design.