BRein Memory: A Single-Chip Binary/Ternary Reconfigurable in-Memory Deep Neural Network Accelerator Achieving 1.4 TOPS at 0.6 W

Kota Ando, Kodai Ueyoshi, *Student Member, IEEE*, Kentaro Orimo, Haruyoshi Yonekawa, Shimpei Sato, *Member, IEEE*, Hiroki Nakahara, *Member, IEEE*, Shinya Takamaeda-Yamazaki, *Member, IEEE*, Masayuki Ikebe, *Member, IEEE*, Tetsuya Asai, *Member, IEEE*, Tadahiro Kuroda, *Fellow, IEEE*, and Masato Motomura, *Member, IEEE*

***Abstract*— A versatile reconfigurable accelerator architecture for binary/ternary deep neural networks is presented. In-memory neural network processing without any external data accesses, sustained by the symmetry and simplicity of the computation of the binary/ternaty neural network, improves the energy efficiency dramatically. The prototype chip is fabricated, and it achieves 1.4 TOPS (tera operations per second) peak performance with 0.6-W power consumption at 400-MHz clock. The application examination is also conducted.**

***Index Terms*— Binary neural networks, in-memory processing, near-memory processing, neural networks, reconfigurable array, ternary neural networks.**

I. Introduction

DEEP neural networks (DNNs) have attracted wide attention as they outperform existing inference/recognition techniques. A number of application algorithms employing neural network techniques, for example, image recognition and scene understanding, have been enabled and well-accepted by their high prediction accuracy. However, their computational and memory resource requirements are becoming stricter and stricter, because the application problems are becoming more complex and the growth of the computational power is slowing. To overcome this difficulty, many application-specific hardware architectures have been presented.

In 2016, binary DNNs [1], [2] were proposed; in these studies, all the synapse weights and neuron activations are binarized into either $+1$ or -1. This binarization can drastically reduce their memory/computational requirements with only a marginal degradation in accuracy [3]. The secret for such a success is that the binary DNN can exploit the abundant redundancy inherent in DNNs by constraining "excitatory" and "inhibitory" weights/activations to $+1$ and -1, respectively. In this paper, we present a compact single-chip DNN accelerator architecture, featuring an array of tightly coupled static random-access memory (SRAM)-logic modules. By utilizing the simplicity and potential parallelism of binary DNNs, our architecture can compute DNNs very efficiently both areawise and energywise, excluding energy-consuming external dynamic random-access memory (DRAM) accesses during streaming. All weights are stored in the on-chip SRAMs and computed in parallel through the associated logic. We also extended the binary DNN to cover ternary DNNs [4], and made it versatile and reconfigurable, that is, the single architecture can cover DNNs of various types (full, dense, and sparse), widths (neurons per layer), and depths (a number of layers). In parallel to keenly optimized CNN accelerators [5]–[9], we believe that such a versatile DNN accelerator will become prevalent with the continuous explosive evolution of algorithms and applications for DNNs.

This paper is based on our previous conference papers [10], [11]. The main contributions of this paper are as follows. We explained the base algorithm and architecture components in more detail, compared our architecture to the results of the latest works, introduced training/preprocessing/simulation software environments that we developed for application evaluations, and conducted examinations for practical applications.

The remainder of this paper is organized as follows. Section II lists some prior works, including reconfigurable or purpose-built neural network/neuromorphic accelerators. In Section III, we explain the base idea of our in-memory accelerator architecture by reviewing the basics of conventional and binary neural networks. Section IV shows the entire architecture structure, introduces the reconfigurable extensions for versatile DNN accelerations, and details the mechanisms of our highly parallel computation. In Section V, we show the effectiveness of our proposed system by evaluating the prototyped chip and comparing it to CPU, graphics processing unit (GPU), field-programmable gate array (FPGA), and prior hardware solutions. Section VI shows the application examinations that were conducted on the prototyped chip, including the setup of training/preprocessing software environments. Here,

Manuscript received August 3, 2017; revised October 16, 2017; accepted November 17, 2017. This paper was approved by Guest Editor Makoto Ikeda. This work was supported by JST ACCEL, Japan, under Grant JPMJAC1502. *(Corresponding author: Kota Ando.)*

K. Ando, K. Ueyoshi, K. Orimo, S. Takamaeda-Yamazaki, M. Ikebe, T. Asai, and M. Motomura are with the Graduate School of Information Science and Technology, Hokkaido University, Sapporo 060-0814, Japan (e-mail: ando@lalsie.ist.hokudai.ac.jp).

H. Yonekawa, S. Sato, and H. Nakahara are with the Department of Information and Communications Engineering, School of Engineering, Tokyo Institute of Technology, Tokyo 152-8552, Japan.

T. Kuroda is with the Department of Electronics and Electrical Engineering, Keio University, Yokohama 223-8522, Japan.

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

Digital Object Identifier 10.1109/JSSC.2017.2778702

we also considered the requirements of the extended architectures based on the observations of the current prototyped architecture. Finally, we conclude this paper with a provision toward future works in Section VII.

II. Related Works

Many studies have been conducted on the neural network inference accelerator built on FPGAs or application-specific integrated circuits (ASICs). The Eyeriss architecture [5] proposed by MIT in 2016 is one of the most successful processors for convolutional neural networks (CNNs), and features a single-instruction multiple-data (SIMD)-style array with multilevel on-chip memories and efficient data delivery by a network-on-chip.

Envision [9] by KU Leuven and their previous work [6] are dedicated to CNN processing utilizing an SIMD multiply-accumulator (MAC) array with technical task division and a reconfigurable voltage control to reduce power consumption. In 2016, KAIST proposed an SIMD MAC-based CNN accelerator [7], which utilizes the dynamic precision control, and some techniques to reduce the kernel data amount. Its improved version DNPU [8] is one of the few accelerators with the ability to process recurrent neural networks (RNNs) as well as CNNs.

TrueNorth [12], [13], developed by IBM, is a neuromorphic accelerator, which employs in-memory spiking neuron cores to simulate the asynchronous neural behavior. It works at low power based on the sparsity of the asynchronous synaptic activities in "real time" (the clock frequency of the real brain).

From the viewpoint of the algorithm, our approach originates from the computing of low-precision neural networks. The concept of processing DNNs at low computational precision has been researched over several years [14], exploiting the redundancy in their large amount of weights. The first report on neural networks using binary weights was "BinaryConnect" [15], which restricts all the weight to ± 1 while the activations are still in full precision, proving that this extreme approximation could obtain a high recognition accuracy, thus encouraging the later hardware-friendly low-precision algorithms. The technique binarizing both weights and activations [16], termed "BinaryNet" [1], was reported in 2016, followed by "exclusive NOR (XNOR)-Net" [2] that expresses the weights with binary weights and a few representative real values. In the second half of 2016, "ternary weight network" [4], which restricts the weights to be -1, $+1$, or 0, was presented and proved to achieve an even smaller accuracy degradation compared with binarization in some DNN categories because of its richer numerical representation [4], [17]. The growth of these low-precision neural networks cannot be stated without stating the contribution of the "Batch Normalization" algorithm [18], which tolerates the influence of the changes in the input distribution by applying a statistical transformation in the training phase.

III. Binary DNN Algorithm and Base Architecture

A. Motivation

A modern DNN model is usually composed of dozens of millions of parameters (synapse weights), which are multiplied with the input activations and then accumulated. A large number of MAC operations are needed to generate the output from these input activations. Neural network processing requires high memory capacity and high computational performance.

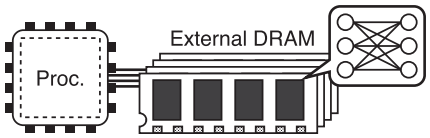


Fig. 1. Conventional DNN accelerator: optimized processor with high-capacity external DRAM.

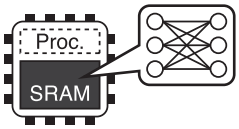


Fig. 2. Our concept: in-memory single chip solution.

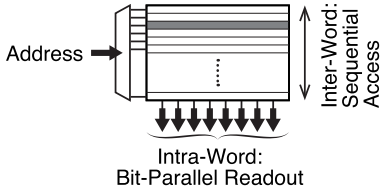


Fig. 3. SRAM has the intra-word bit-parallel readout and inter-word wise serial access.

Many ASIC- and FPGA-based neural network accelerators have been proposed to speed up neural network computation and achieve high energy efficiency. The major structure of such accelerators is an "application-specific processor," as shown in Fig. 1; this is made up of dedicated processing units and an external memory to store the parameters of a neural network. However, the demerit of such a type of structure is that it often suffers from the bottleneck of the external memory (or "Von Neumann bottleneck"), where the throughput and energy efficiency are limited by the external DRAM accesses.

To solve this problem, we considered a single-chip in-memory approach, as shown in Fig. 2, in which all the network parameters are stored in on-chip SRAMs and the computation completes in logic units located near the SRAMs. This "in-memory" concept excludes costly (in time and energy) external DRAM accesses and could improve the efficiency of neural network processing. The key concept of this approach is the integration of the processing circuits that harmonize with the nature of the SRAMs, as well as the method of sufficiently compressing the neural network parameters to store in on-chip SRAMs. We must exploit the essential intra-word (bitwise) parallelism of SRAMs and hide the sequential behavior of inter-word (wordwise) accesses to achieve highly efficient in-memory computation (see Fig. 3).

It should be noted, in addition, that the architectural concept itself does not limit the on-chip memories to SRAMs; it could be utilized with any type of memories such as embedded DRAMs or embedded non-volatile memories, if the area,

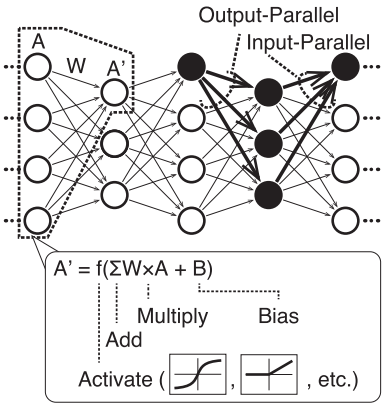


Fig. 4. Computation of a DNN.

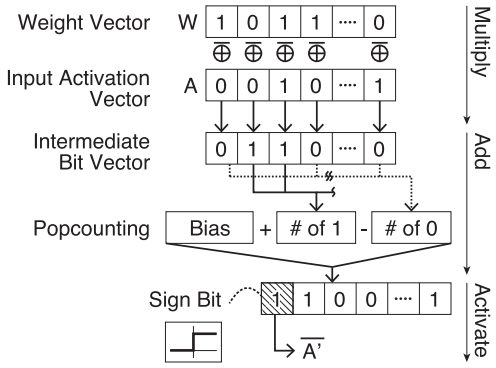


Fig. 5. Neural function in a binary DNN.

process technology, power, and the timing specification were sufficient to its application. It would also be possible to incorporate the proposed architecture into DRAM or non-volatile memory chips for realizing binary/ternary DNN acceleration inside a memory system.

B. Binary Neural Networks and Its Computation

Fig. 4 indicates a general DNN process. A "layer" consists of neurons with inputs from the previous layer and outputs to the next layer. The activation of a neuron is calculated by: 1) multiplying the input activations with their corresponding weights; 2) summing up the products; and 3) applying a nonlinear function ("activation function").

Binary neural networks [2] restrict ("binarize") all the activations and weights to be either $+1$ or -1. When -1 is denoted as 0, a binarized neural function becomes: 1) performing bitwise XNOR of the weight and input activation vectors ("multipling"); 2) counting the numbers of 1s and 0s in the XNORed intermediate bit vector ("adding"); and 3) determining the output activation as 1 (0) when the 1s (0s) are dominant ("activation function"), as shown in Fig. 5.

There are two opportunities of parallel computation in a DNN: output-parallel (a single activation to multiple inputs) and input-parallel (multiple activations to a single input), as shown in Fig. 4. This indicates that if the output-parallel computation (first to second layer) is followed by the input-parallel computation (second to third layer), the second layer results do not need to be explicitly serialized (in a DNN, the second layer activations are only "intermediate activations" used only by the third layer). Based on this observation, we developed a processing-in-memory module (PIM) that can house a 3-layer binary neural network with maximum layer widths of L–H–L as a unit (see Fig. 6).

A PIM consists of two H-bit L-word SRAM arrays with output-parallel neural engine (OPNE) and input-parallel neural engine (IPNE). The OPNE sequentially receives the input activation bit stream A1, which is column-parallel XNORed with H-bit weights W1 read in parallel from the upper SRAM cycle by cycle and accumulated in real values. After the L cycle, the input activation traversal, accumulation, and activation (taking sign) are completed, and the resultant H-bit output activation vector A2 is stored in the registers (flip-flops) between the OPNE and the IPNE. Activation bit vector A2 is then fed in parallel into the IPNE, where column-parallel XNORs and an adder tree process them by using the H-bit weight vector W2 read from the lower SRAM. This produces an output activation every cycle; the activation constitutes the bit-serial output activation stream A3. The OPNE and IPNE process weight matrices in an orthogonally transposed manner, that is, a weight vector used by an output neuron, are stored vertically in the upper SRAM and horizontally in the lower, respectively (in other words, a word of the upper SRAM is the weight *from* an input neuron, while that of the lower SRAM is the weight *to* an output neuron). By using wide SRAMs, long weight/activation vectors can be processed at once, while restricting the input/output of a PIM to a single bit.

IV. Full Architecture (Ternarized/Biased) and Prototype Chip

Recent studies have shown that compared with binary DNNs, ternary DNNs (whose weights take values $+1$, 0, or -1) achieve higher accuracy in several situations [4], as mentioned in Section II, and as discussed in Section VI. In addition, batch normalization techniques [18] are indispensable in binary/ternary neural networks to keep the weights informative under extreme approximation and obtain higher accuracy [1], [2].

In this paper, we extended the base architecture by introducing mask (M) and bias (B) bits to accommodate ternary weights and batch normalization, aiming at handling versatile DNNs by using a single reconfigurable architecture.

A. Ternarized Neural Network

We introduce the mask (M) bits into the base computation model mentioned in Section III-B to allow ternarized DNNs to be mapped onto it. When the mask (M) bit is enabled, the corresponding weight is treated as value "0" (note that logic 0 in the weight bits represents value "-1"; thus, the original binary DNN algorithm does not have any way to represent zero-valued synapse).

Moreover, this extension makes the PIM versatile and reconfigurable, because zero-valued weights are equivalent to "non-existing synapses." There are some algorithms called "pruning" [17], which eliminate some powerless synapses/

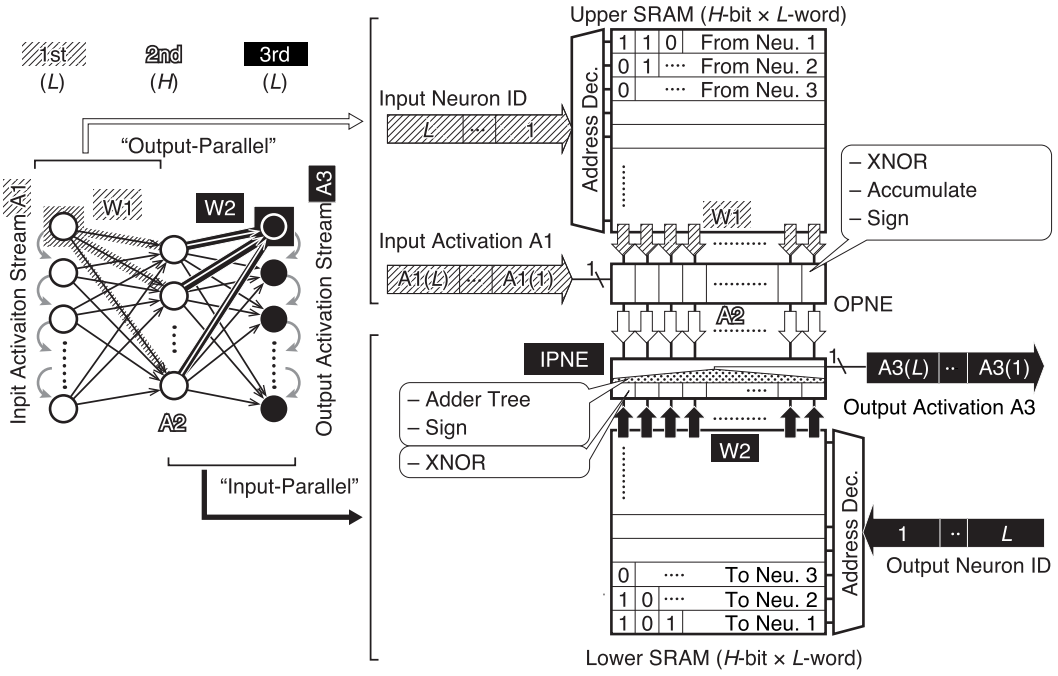


Fig. 6. PIM with OPNE and IPNE for the 3-layer BNN processing.

neurons in a neural network layer. CNNs can also be seen as a fully connected neural network with sparse and repeating synapse weights. By setting nonexisting synapse to a zero value, we can map the neural network layer with an arbitrary shape (dense or sparse and wide or narrow) under maximum width.

B. Bias Terms and Batch Normalization

The bias term (see Fig. 5) is necessary, especially for emulating batch normalization in binary/ternary neural networks. Batch normalization [18] normalizes the distribution of the activations in each layer statistically, thus speeding up the training and improving the generalization ability of the network. Batch normalization is defined by the following formula [18]:

$$\hat{x} = \gamma \left(\frac{x - \mu}{\sigma} \right) + \beta \tag{1}$$

where x is the sum (i.e., $\sum Wa$ in the previous explanation), μ and σ^2 are, respectively, the mean and variance of x over all the input images (or vectors) in the training set, and γ and β are, respectively, the trainable scaling and offset factors. Then, an output in the binary/ternary neural networks is obtained as

$$a' = \text{sign}(\hat{x}) \tag{2}$$

$$= \text{sign} \left\{ \gamma \left(\frac{x - \mu}{\sigma} \right) + \beta \right\}. \tag{3}$$

According to Yonekawa and Nakahara [19], since binary/ternary networks require only the sign, this equation can be modified into

$$a' = \text{sign} \left\{ x + \left(-\mu + \frac{\sigma}{\gamma} \beta \right) \right\} \tag{4}$$

because γ and σ are positive constants in the inference phase. This implies that all the batch normalization constants can be packed into a bias constant $(-\mu + (\sigma/\gamma)\beta)$ after the completion of the training phase [19].

To represent the real-valued constant bias terms in addition to the weight values, we introduced the "bias" B bits. The bias is coded into B bits in a distributed manner, that is, the absolute value of the bias term of an output neuron is the number of B bits enabled in its input weights, with its sign coded by the M bits (the M bits are diverted as the signs of the bias values when the B bits are asserted). The bias values are added or subtracted to the bit count in the OPNE/IPNE. By accommodating this bias addition mechanism, our architecture can apply batch normalization without needing additional multipliers/dividers to calculate normalized activations. Note that we adopted this distributed bias expression because of the symmetry of the OPNE and IPNE, thus overlooking the loss of the data representation density on the SRAMs. A bias term is needed *per column* in the OPNE, because the weights for an output neuron are represented in a column, while it is required *per row* in the IPNE because an output is represented in a row. Nonetheless, the distributed bias representation can be utilized equally in the upper and lower SRAMs.

C. Architecture Design With Ternary and Bias Extension

All the possible configurations of the extended M and B bits in a weight (synapse) are enumerated in Fig. 7(a). Fig. 7(b) and (c) shows the detailed schematics of the OPNE and IPNE that consider the ternary and bias extensions, respectively. We define a small logic unit as "synapse decoder" (or *SynDec*), which determines the real number (partial sum) to add in the accumulator or adder tree according to the weight W, mask M, bias B, and input activation A bits.

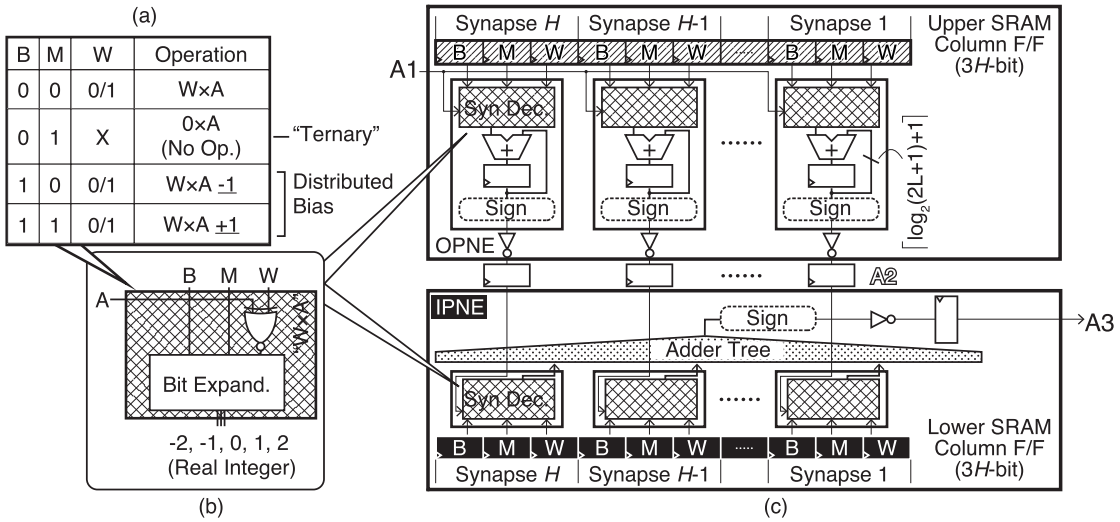


Fig. 7. Synapse decoder (SynDec) and detailed OPNE/IPNE circuit for extended ternary neural networks and bias terms. (a) Synapse configuration. (b) Synapse decoder. (c) Schematic of a PIM (OPNE/IPNE).

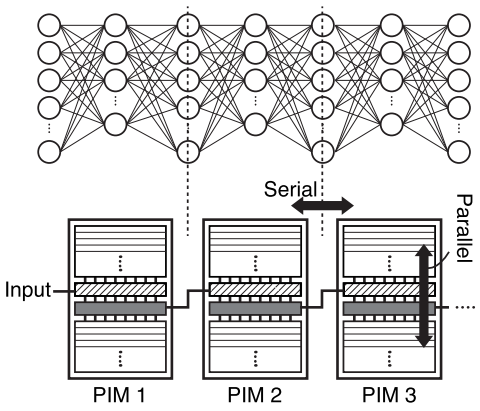


Fig. 8. Processing multilayer neural network onto the cascaded PIMs.

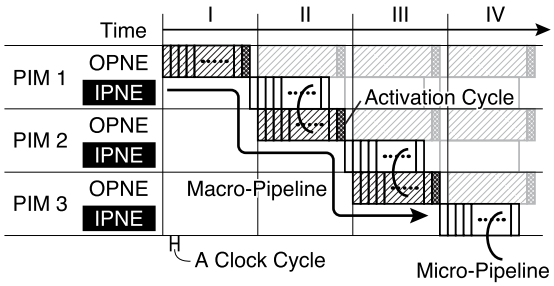


Fig. 9. Macro- and micro-pipelined computation on the cascaded PIMs. An "activation cycle" at the end of every macro-pipeline stage is required for taking the sign of the accumulator values and for issuing the SRAM addresses.

The OPNE is composed of arrayed H SynDecs, followed by accumulators; it computes H "ternarized" multiplications by using one input activation A1 and H accumulations in L clock cycles, and then takes the signs to produce the activation after summing up all the partial sums. The IPNE also consists of the SynDecs; its main difference to the OPNE is the existence of an adder tree. Here, the partial sums are summed up in an adder tree and an output activation A3 is generated in a clock cycle. The OPNE/IPNE and two SRAMs are packed to form a PIM. A PIM conducts bit-parallel computation of two output layers of a neural network by using the bit-serial input and output.

D. Macro- and Micro-Pipelined Execution and Reconfigurable Array Extension

Connecting PIMs inline is a straightforward extension. Since the input and output of a PIM are 1-bit serial streams, multiple PIMs can be easily cascaded for deeper neural network processing, thus keeping the computation inside a PIM parallel (see Fig. 8). Each of the OPNEs and IPNEs processes a layer; thus, two layers are hosted by a PIM.

The input bit stream is processed by OPNEs/IPNEs in a macro- and micro-pipeline parallel manner (see Fig. 9) in addition to the internal column-parallel processing.

The output of the OPNE is calculated and determined after $(L + 1)$ cycles (all L input activations pass and the sign bit is taken), then the IPNE functions by using this OPNE result (stored in pipeline registers). While the IPNE is computing, the OPNE can start the processing of the next input frame ("transaction"). We call this a "macro-pipeline," since the activations move *every* $(L + 1)$ *cycles*.

The output of the IPNE is a single bit that is generated every clock cycle. This output is used immediately as the OPNE serial input at the next PIM. We call this a "micro-pipeline," because the activations flow in the processing engines *every cycle*.

A PIM can compute layers smaller than the maximum size L–H–L, as shown in Fig. 10. For the first layer (serial input) or the third layer (serial output), the computation can be performed by limiting the address range of the weights in SRAMs. For the second layer (intermediate result between the OPNE and IPNE), we can "mask" the surplus synapses to prevent those unused values from being summed.

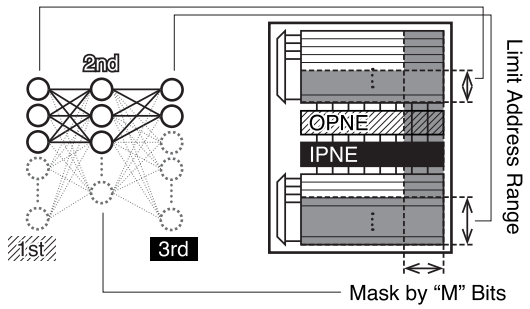


Fig. 10. Mapping a smaller layer onto a PIM. We can use smaller layers by limiting the address range (for the input and output activations) and by masking unused weights (for the intermediate layer).

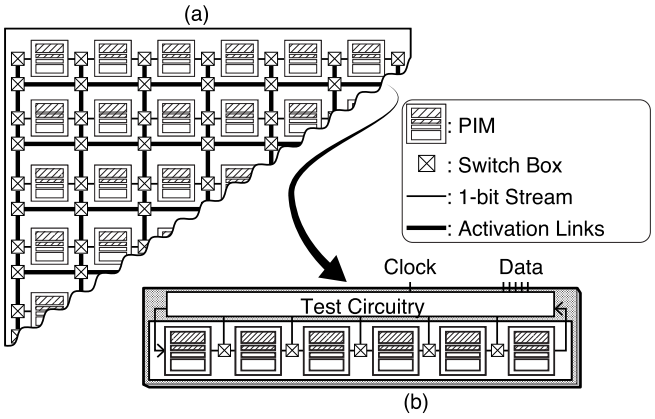


Fig. 11. (a) Reconfigurable 2-D PIM array. (b) Prototype chip with inline-arranged six PIMs.

By arranging PIMs in a reconfigurable regular array, the architecture becomes transformable and expandable (see Fig. 11). Here, activation links among PIMs are bundled bitwise signals and are circuit-switched at the switch boxes (like in FPGA) so that the activations can be looped back (for RNNs), duplicated (for wider DNNs), or chained (for deeper DNNs).

V. Experimental Results and Comparisons

A. Prototyped Chip Evaluation

We prototyped a portion of the reconfigurable array by using the TSMC 65-nm GP process (see Fig. 12). SRAM sizes L and H were set to 484 and 144, respectively, based on the availability of an SRAM library. We successfully integrated six PIMs capable of hosting a maximum 13-layer, 4.2k-neuron/0.8M-synapse binary/ternary DNN in 3.9 mm^2. In every clock cycle, 3456 operations (144×2[OPNE + IPNE] $\times$ 6[PIMs] $\times$ 2[Add + Mul]) are conducted in parallel. The chip operates at 400 MHz, achieving 1.4 TOPS (tera operations per second; 1 synapse MAC = 2 operations) with 0.6 W.

We measured the power consumption and critical path delay on several supply voltages (see Fig. 13). The chip works at the range from 110 MHz (min. 50 MHz) at 0.55-V supply to 400 MHz at 1-V supply, with a power consumption from 0.06 to 0.58 W. The critical path of this architecture is in

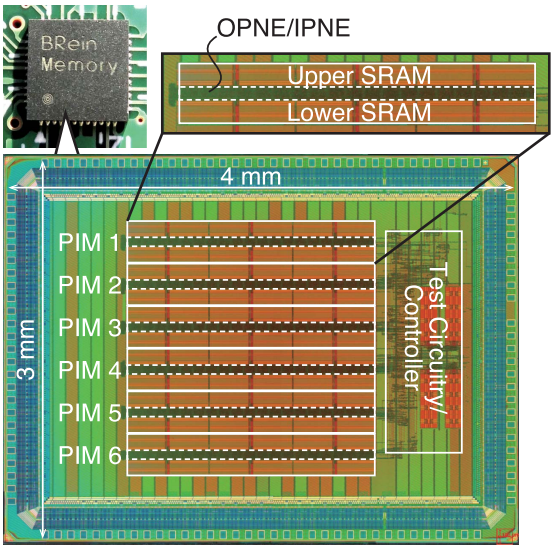


Fig. 12. Prototyped chip.

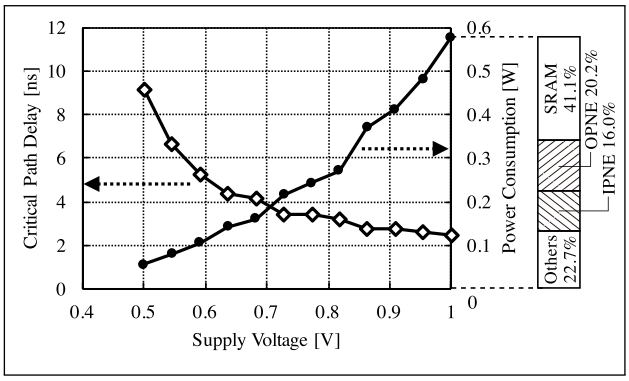


Fig. 13. Critical path delay and power consumption over the supply voltage. The chip achieves 400 MHz and 0.58 W at 1-V power supply. The critical path of the architecture is the path from the OPNE output register to the IPNE output register. The SRAMs (with the flip-flops), OPNEs, and IPNEs account for 41.1%, 20.2%, and 16.0% of the total power consumption, respectively.

the adder tree in the IPNE, that is, the path from the OPNE output registers to the IPNE output register, where the sum of the L intermediate products and the sign of the sum are calculated within a cycle. When we scale up the architecture, the maximum clock frequency will be limited by the width (i.e., the depth of the adder tree) of the IPNE. The ratio of the power consumption of the SRAMs (including the readout flip-flops), OPNEs, and IPNEs to the total is 41.1%, 20.2%, and 16%, respectively.

B. Comparison With CPU, GPU, and FPGA

We tested the proposed chip through an experimental handwritten digit recognition by using a fully connected binary DNN (see Fig. 14). To measure the core power consumption and execution time, we prepared a 13-layer binary DNN, which occupies all the PIM computational resources and the SRAM area, except the last layer (that is, the output layer; only ten neurons corresponding to the classes are enabled). We trained the 13-layer DNN by using the

TABLE I
COMPARISON ON THE SAME 13-LAYER BINARY DNN (MEASURED)

	Process Tech.	Clock Freq. [GHz]	Exec. Time [sec]	Power [W]	Energy Consum. [J]	Effective Perf. [GOPS]	Energy Efficiency [GOPS/W]	Ratio			
								Clock Freq.	Exec. Time	Power	Energy Consum.
CPU	14nm	2.2	124	155	19.2k	12.4	0.08	5.5	102	266	27.1k
GPU	28nm	1.0	13.8	152	2,098	111.3	0.73	2.5	11	261	2,966
FPGA	28nm	0.2	53	11	583	29.0	2.64	0.5	44	19	825
This Work	65nm	0.4	1.22	0.6	0.73	1264.4	2172.42	1	1	1	1

Measured using 1 million transactions of the binary DNN handwritten digit recognition. We assumed that 1 MAC is composed of 1 ADD and 1 MUL; thus, 1 MAC is equivalent to 2 OPs.

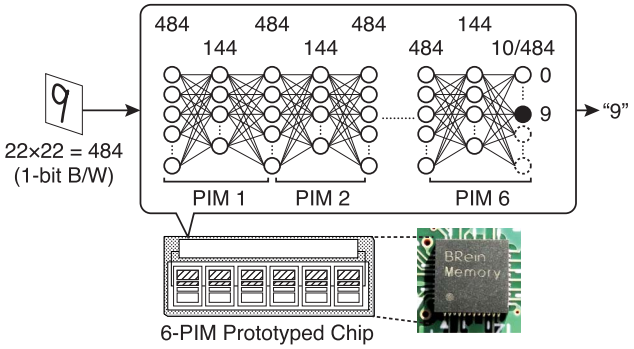


Fig. 14. Testing handwritten digit recognition on a 13-layer fully connected binary MLP.

trainer software we developed using Chainer and TensorFlow; this is explained in Section VI. The training data set is not MNIST-compatible, but is generated from the MNIST data set by resizing the input images to 22×22 (because of the limit of the SRAM depth 484, which corresponds to the input data size), and binarizing these images into black-and-white images through thresholding. The recognition accuracy reached 90.1% at 13 layers (details will be explained in Section VI).

When compared with a CPU,[1] GPU,[2], and FPGA[3] running the same binary DNN, the chip achieved 1–2 and 2–4 orders of magnitude better performance and energy efficiency, respectively, with the 0.5×–5.5× lower clock frequency and 1–2 orders lower power consumption (see Table I). In addition, we examined a binary DNN inverted pendulum robot and game player through reinforcement learning.

C. Comparison With Prior Accelerators

Recent hardwired binary neural network implementations [20], [21] are custom solutions with insufficient versatility in their scope. The comparison of recent state-of-the-art CNN accelerators [5], [8], [9] with this paper (see Table II) is not straightforward: they are nonbinary (versus binary) and are built for being used in the CNN (versus targeted for a wide variety of DNNs). For the metrics in Table II, the former (latter) performs positively (negatively) to the proposed method. With this precaution, the peak performance (TOPS) of our chip outperformed that of the prior studies. The energy efficiency (TOPS/W) and the area efficiency (TOPS/mm^2) were in better or comparable range, considering our older CMOS process technology (we also estimated the 65-nanometer-normalized power and area efficiency of the prior works). This is mainly because the wide SRAMs cost the 66% area and 41% power, respectively. However, considering that the prior CNN accelerators access off-chip memories whose area and power are not included, the balance is actually much more favorable to the proposal.

VI. TRAINING ENVIRONMENT AND ALGORITHM EVALUATION

We built the neural network trainer and emulator software tuned to BRein Memory for three purposes: 1) to examine the effectiveness of the refined BRein Memory architecture; 2) to run application evaluation awareness of the hardware features; and 3) to consider and determine the requirements of the future architecture. We developed the custom operation library of Google TensorFlow by using the aforementioned extensions, such as weight ternarization and bias-term-based batch normalization. We used the customized TensorFlow framework in the construction of the application described in this section.

A. Application Test Using the MNIST Handwritten Digit Data Set

First, we implemented the handwritten digit recognition on the multilayer perceptron (MLP) that we mentioned in Section V-B with three types of data representations, that is, binary, ternary, and floating point, for the comparison. The flexibility in the depth (a number of layers) of the neural network is one of the features of our architecture; thus we conducted a parameter exploration to show the flexibility on the layer number and to determine the optimal layer number.

[1]The test environment is a CentOS 6.4 running on a 128-GB DDR4 main memory and two CPUs, each of which is Intel Xeon E5-2650v4, 2.20 GHz, 12 cores, with 30M cache. The vector computation library was not enabled.

[2]The GPU we used is NVIDIA GeForce GTX TITAN X, which has 3072 cores and a 12-GB VRAM, installed on the same host system used for the CPU evaluation.

[3]The FPGA is Xilinx Virtex-7 XC7V690T FFG1761-3 mounted on the Digilent NetFPGA-SUME board with 8-GB DDR3 and 216-Mbit SRAM. The board is installed on the same system used for the CPU evaluation.

TABLE II
COMPARISON AND CHIP SUMMARY

	ISSCC 2016 [5]	ISSCC 2017 [8]	ISSCC 2017 [9]	This Work
Technology	65nm LP	65nm	28nm FD-SOI	65nm GP
Target	CNN	CNN + RNN	CNN	Versatile DNN[1]
W/A Precision [bits]	16	4–16 (Conv), 4–7 (FC)	4–16	Bin/Ter
Operating Frequency [MHz]	200	50–200	200 (Typ.)	100–400
Voltage [V]	1.0	0.77–1.1	0.6–1.1	0.55–1.0
Power [W]	0.3	0.03–0.28	0.075–0.3	0.05–0.6
Core Area [mm^2]	12.3	7.4	0.95	3.9
Peak Performance[2] [TOPS]	0.07	1.25	0.41	1.38
Energy Efficiency [TOPS/W]	0.2	1.0–8.1	0.26–10 [0.05–1.86][3]	2.3–6.0
Area Efficiency [TOPS/mm^2]	0.006	0.045–0.169	0.108–0.431 [0.02–0.08][3]	0.089–0.365
Dataset (Network) for Evaluation	(AlexNet)	(AlexNet)	Hierarchical Face Recognition (AlexNet, VGG-16)	Resized MNIST (Binary/Ternary MLP)

[1] Although our architecture is not specially optimized for CNNs, it can still house CNNs by mapping convolutional layers as fully connected layers, as mentioned in Section VI-C.
[2] Note that the definition of OP differs chip-wise. We estimated the performance by assuming that 1 MAC for 1 synapse is composed of 2 operations (addition and multiplication).
[3] The efficiency assuming the system is integrated on a 65nm technology, that is calculated using the scaling law (Power, Area $\propto$ (Gate Length)^2).

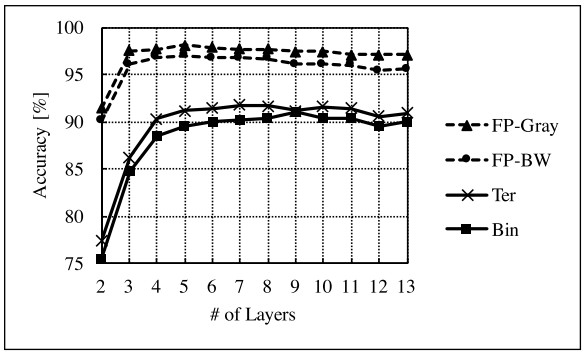


Fig. 15. Accuracy behavior when the number of layers varies (including the input layer). "Ter" and "Bin" are the results of ternarized and binarized weights, respectively, which can be mapped to the prototyped chip, while "FP-Gray" and "FP-BW" are grayscale and black-and-white images, respectively, used for comparison. The sizes of intermediate layers (the layers except for the input and output layers) are 484–144 and the input image is 22×22 pixel black and white due to the SRAM width of the prototyped chip.

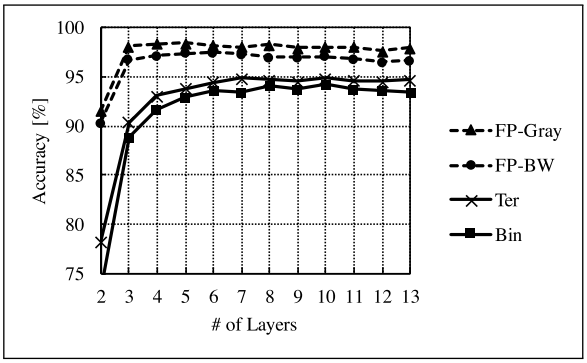


Fig. 16. Accuracy behavior when the number of layers varies (including the input layer) in the case the SRAMs are wider (having more bits per word) than that in the prototype chip. The intermediate layer sizes are fixed to 1024 and the input image is 22×22 pixel black and white.

We trained the MLPs with different numbers of layers by using their shapes, which agree well with the current PIM design (484–144 style). The input images were resized to 22×22 ($= 484$) and binarized through thresholding. The result of the layer-number sweep is shown in Fig. 15, where "Ter" is a ternalized network by fixed thresholds; "Bin" is a binary network, which does not allow zero-valued weights; and "FP-BW" and "FP-Gray" are 32-bit floating-point weights including and excluding the input binarization, respectively. The accuracy increases and saturates when the layer number scales. For this task, a 6-layer (or 5-layer if counting the input layer out) MLP seemed to be sufficient.

We mentioned that the 484–144 style OPNEs/IPNEs are only used for chip prototyping and the architecture itself could be utilized with a larger/smaller memory. To observe the behavior of the case in which the layer sizes were uniform, we tested the same handwritten digit recognition task by using an MLP with 1024 neurons per intermediate layer. Although this assumption is above the capacity of the test chip, it could be realized if we utilized a (1024×3)-bit 1024-word SRAM macro. For comparison, the input size was fixed at 22×22. According to Fig. 16, the accuracy behavior when the number of layers varies is quite similar to that of the 484–144 case, with an improvement in the final accuracy. A 6-layer MLP is still sufficient for this extended layer size assumption.

The two above-mentioned parameter sweeps have shown that a 6-layer MLP is sufficient both in the current 484–144 test chip and the extended 1024-neuron model. Next, we tested the effect of the layer width with the layer number fixed to 6. Fig. 17 shows that the recognition accuracy still improves when we utilize richer layer/input width configurations; thus, there is room for employing wider OPNEs/IPNEs paired with wider SRAMs. Even though the accuracy of the binary/ternary networks in this test is a few percent lower than that of FP32 networks, there are some applications where the efficiency, throughput, and shorter response time are much more important than a slightly higher accuracy, such as mobile or embedded always-on security systems.

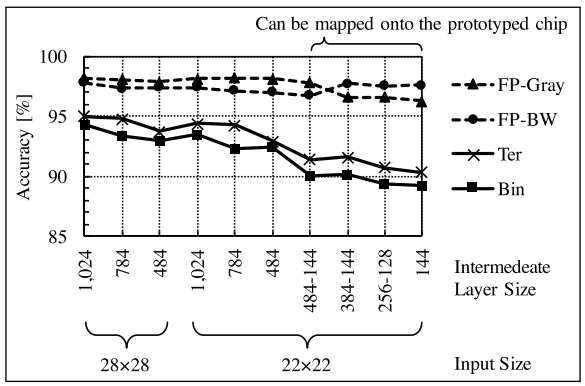


Fig. 17. Accuracy behavior when the intermediate layer width varies.

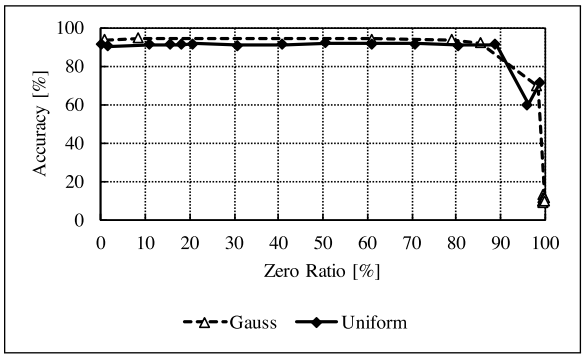


Fig. 18. Accuracy depends on the zero ratio of the total "ternarized" parameters. Over 90% accuracy is obtained when 85% weights are zero. "Gauss" and "Uniform" are the initial distributions of the weights.

B. Threshold and Zero Ratio

The ratio of zero-valued weights can be controlled by simply setting the thresholds. Fig. 18 shows the relationship between the accuracy and zero ratio of a handwritten digit recognition MLP. We tested two methods for parameter initialization at the beginning of the training phase: uniform and Gaussian distributions. In our small application model, the difference between the initial values affected the number of training steps (or "epochs") and the final distribution (and zero ratio) of the weights. However, the effects of zero ratio on the accuracy are similar. This result shows that over 90% accuracy is maintained even if 85% of the weights are valued zero in this task. This suggests an opportunity for data compression (or *pruning*), that is, reducing memory requirement by eliminating zero-valued meaningless weights. Although our current architecture cannot exploit data sparsity, this observation, expecting network compression, would be key in the future reconfigurable architectures.

C. Experimental CNN Mapping

The proposed architecture is not optimized for CNNs; however, it has the capability of housing a CNN by representing a convolutional layer as a fully connected layer. The maximum configuration of a convolutional layer is only limited by the layer size (i.e., the sizes and numbers of feature maps). The kernels are statically preprocessed and reshaped into fully connected forms, with zero-valued weights representing the outside of the receptive field of a certain output neuron. As shown in Fig. 19, this method enables us to map a convolutional layer to our PIMs but the efficiency may be limited; a considerable number of "zero-valued" synapses account for memory capacity, and some regularly duplicated values arise from the same kernel but different input/output neurons.

We conducted an experiment of a simple CNN mapping onto the prototyped 484–144 chip, with the same MNIST digit recognition task. To map a CNN onto the experimental chip, we chose the layer configurations shown in Table III ($K_h \times K_w \times K_c$ implies a kernel shaped $K_h \times K_w$ with K_c output channels). As shown in Fig. 20, the highest accuracy was observed at $5 \times 5 \times 4$ for the first layer and $5 \times 5 \times 12$ for the second layer; however, it resulted in a lower accuracy

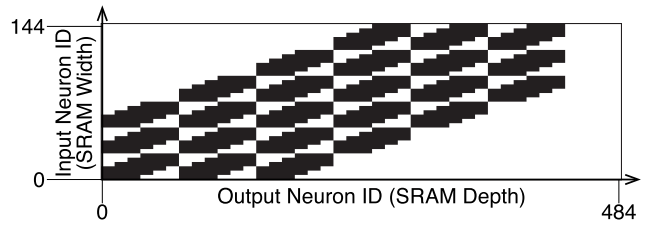


Fig. 19. Expanding a convolutional layer as a fully connected layer (used the second layer of Table III sized $6 \times 6 \times 12$ as an example). The horizontal axis is the addresses, and the vertical axis is the bits of the SRAM (actually, the addresses are tripled for 3-bit synapse representation). The black pixels are used as kernel weights, and the white area is filled with 0s. Since a convolutional layer applies a single kernel to multiple output neurons, many of the black blocks are duplicated.

TABLE III
LAYER CONFIGURATIONS OF THE EXPERIMENTAL MNIST CNN

#	Type	Kernel	Output
0	Input	–	$22 \times 22 \times 1$
1	Conv	$2 \times 2 \times 4$, $3 \times 3 \times 4$, $4 \times 4 \times 4$, or $5 \times 5 \times 4$ with Stride 4	$6 \times 6 \times 4$
2	Conv	$2 \times 2 \times 12$, $3 \times 3 \times 12$, $4 \times 4 \times 12$, or $5 \times 5 \times 12$	$6 \times 6 \times 12$
3	F.C.	(F.C. layer from $432 (= 6 \times 6 \times 12)$ to 144)	144
4	F.C.	(F.C. layer from 144 to 10)	10

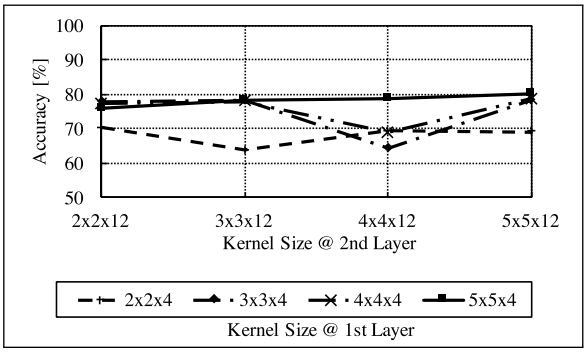


Fig. 20. Accuracy behavior when the kernel sizes of the first and second layers vary. Each line corresponds to the first-layer kernel and the horizontal axis is the second-layer kernel size.

than the fully connected evaluation. This is mainly because of the resized input image and narrow intermediate layer due to the narrow SRAM width of the prototyped chip. The accuracy will improve if we use a wide SRAM with wide OPNE/IPNE that can store many channels of the feature map.

VII. Conclusion and Future Work

In this paper, we proposed an efficient in-memory accelerator architecture for binary/ternary neural networks. The architecture realized efficient highly parallel computation near the SRAMs under the serial input/output data, exploiting the parallelism that the neural networks originally have. By arranging the serial inputs/outputs of processing units and configuring the synapse weights on the SRAMs, this architecture has the versatility to host any type of network with any depth, such as fully connected DNNs, recurrent DNNs, and sparsely connected DNNs (CNNs and pruned DNNs), in a reconfigurable manner.

We fabricated the prototyped chip on 65-nm CMOS with twelve 484-word (144×3)-bit SRAMs, and examined the availability of the architecture through an application test by using our developed training/testing environment. The chip measurement and comparison proved that the energy and area efficiencies achieved 2.3 TOPS/W and 0.089 TOPS/mm^2, respectively, which are higher or comparable to the latest highly dedicated accelerators. In parallel to keenly optimized CNN accelerators, such a versatile DNN accelerator will become prevalent as algorithms and applications for DNNs continue to evolve.

The application test also indicated the issues to be solved for the next generation architecture. Although the sparse DNNs and CNNs can be mapped onto the PIM array, the computational efficiency degrades compared with fully connected dense DNNs. For further improvement of the efficiency, the native support is needed for mapping the sparse DNNs onto SRAMs and skipping the zero-weighted multiplication.

The reconfigurable 2-D PIM array is also being designed and will be published elsewhere. In addition to the architecture and execution model explained in this paper, network model division techniques to construct multiple smaller networks that are virtually equivalent to the original network will be required, as well as hardware consideration for achieving low-latency and general-purpose neural network processing.

References

[1] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to $+1$ or -1," *CoRR*, pp. 1–11, Feb. 2016. [Online]. Available: http://arxiv.org/abs/1602.02830

[2] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," *CoRR*, pp. 1–17, Mar. 2016. [Online]. Available: http://arxiv.org/abs/1603.05279

[3] L. Hou, Q. Yao, and J. T. Kwok, "Loss-aware binarization of deep networks," *CoRR*, pp. 1–11, Nov. 2016. [Online]. Available: http://arxiv.org/abs/1611.01600

[4] F. Li, B. Zhang, and B. Liu, "Ternary weight networks," *CoRR*, pp. 1–5, May 2016. [Online]. Available: http://arxiv.org/abs/1605.04711

[5] Y. H. Chen, T. Krishna, J. Emer, and V. Sze, "14.5 Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Jan. 2016, pp. 262–263.

[6] B. Moons and M. Verhelst, "A 0.3–2.6 TOPS/W precision-scalable processor for real-time large-scale ConvNets," in *Symp. VLSI Circuits Dig. Tech. Papers (VLSI)*, Jun. 2016, pp. 178–179.

[7] 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 *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Jan. 2016, pp. 264–265.

[8] D. Shin, J. Lee, J. Lee, and H.-J. Yoo, "14.2 DNPU: An 8.1TOPS/W reconfigurable CNN-RNN processor for general-purpose deep neural networks," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 240–241.

[9] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, "14.5 envision: A 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28 nm FDSOI," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2017, pp. 246–247.

[10] K. Ando *et al.*, "BRein memory: A 13-layer 4.2 K neuron/0.8 M synapse binary/ternary reconfigurable in-memory deep neural network accelerator in 65 nm CMOS," in *Proc. IEEE Symp. VLSI Circuits (VLSI-Circuits)*, Kyoto, Japan, Jun. 2017, pp. C24–C25.

[11] H. Yonekawa *et al.*, "In-memory area-efficient signal streaming processor design for binary neural networks," in *Proc. 60th IEEE Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Boston, MA, USA, Aug. 2017, pp. 116–119.

[12] P. A. Merolla *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 2014. [Online]. Available: http://science.sciencemag.org/content/345/6197/668

[13] 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. (CICC)*, Sep. 2011, pp. 1–4.

[14] M. Courbariaux, Y. Bengio, and J.-P. David, "Training deep neural networks with low precision multiplications," *CoRR*, pp. 1–10, Dec. 2014. [Online]. Available: http://arxiv.org/abs/1412.7024

[15] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," *CoRR*, pp. 1–9, Nov. 2015. [Online]. Available: http://arxiv.org/abs/1511.00363

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

[17] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *CoRR*, pp. 1–29, Sep. 2016. [Online]. Available: http://arxiv.org/abs/1609.07061

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, pp. 1–11, Feb. 2015. [Online]. Available: http://arxiv.org/abs/1502.03167

[19] H. Yonekawa and H. Nakahara, "On-chip memory based binarized convolutional deep neural network applying batch normalization free technique on an FPGA," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, May 2017, pp. 98–105.

[20] E. Nurvitadhi, D. Sheffield, J. Sim, A. Mishra, G. Venkatesh, and D. Marr, "Accelerating binarized neural networks: Comparison of FPGA, CPU, GPU, and ASIC," in *Proc. Int. Conf. Field-Program. Technol. (FPT)*, Dec. 2016, pp. 77–84.

[21] H. Nakahara, H. Yonekawa, T. Sasao, H. Iwamoto, and M. Motomura, "A memory-based realization of a binarized deep convolutional neural network," in *Proc. Int. Conf. Field-Program. Technol. (FPT)*, Dec. 2016, pp. 277–280.

Kota Ando received the B.E. and M.E. degrees in electronics from Hokkaido University, Sapporo, Japan, in 2016 and 2018, respectively, where he is currently pursuing the Ph.D. degree.

His current research interests include reconfigurable architectures and hardware-aware algorithms for efficient deep learning processing.

Mr. Ando is a member of the Institute of Electronics, Information and Communication Engineers, Japan. He received a Research Fellowship for Young Scientists from the Japan Society for the Promotion of Science in 2018. He received the Best Student Presentation Award from the Technical Committee on Reconfigurable Systems, Institute of Electronics, Information and Communication Engineers, Japan, in 2016 and 2017.

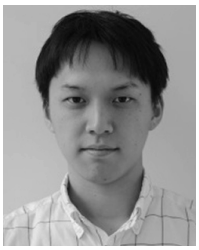
Kodai Ueyoshi (S'15) received the B.E. and M.E. degrees from Hokkaido University, Sapporo, Japan, in 2015 and 2017, respectively, where he is currently pursuing the Ph.D. degree in efficient hardware architecture for machine learning systems.

His current research interests include deep learning hardware accelerators and memory-error tolerance analysis for deep learning system.

Mr. Ueyoshi is a member of the Institute of Electronics, Information and Communication Engineers, Japan. He received a Research Fellowship for Young Scientists from the Japan Society for the Promotion of Science in 2017 and a Silkroad Award at the 2018 IEEE International Solid-State Circuits Conference.

Kentaro Orimo received the B.E. and M.E. degrees in electronics from Hokkaido University, Sapporo, Japan, in 2016 and 2018, respectively.

His current research interests include recurrent neural networks and its hardware-aware algorithms.



Haruyoshi Yonekawa received the B.E. and M.E. degrees from the Tokyo Institute of Technology, Tokyo, Japan, in 2016 and 2018, respectively.

His current research interests include reconfigurable architecture, embedded systems, and machine learning.

Mr. Yonekawa received the 24th Reconfigurable Architectures Workshop Best Demo Award in 2017.

Shimpei Sato (M'16) received the B.E., M.E., and Ph.D. degrees in engineering from the Tokyo Institute of Technology, Tokyo, Japan, in 2007, 2009, and 2014, respectively.

From 2014 to 2016, he was involved in high performance computing area as a Post-Doctoral Researcher, where he investigated an application performance analysis/tuning method. He is currently an Assistant Professor with the Department of Information and Communications Engineering, Tokyo Institute of Technology. His current research interests include approximate computing realization by architecture design and circuit design and their applications.

Dr. Sato is a member of the ACM, the Institute of Electronics, Information and Communication Engineers, and the Information Processing Society of Japan.

Hiroki Nakahara (M'05) received the B.E., M.E., and Ph.D. degrees in computer science from the Kyushu Institute of Technology, Fukuoka, Japan, in 2003, 2005, and 2007, respectively.

He has held research/faculty positions at the Kyushu Institute of Technology, Kagoshima University, Kagoshima, Japan, and Ehime University, Ehime, Japan. He is currently an Associate Professor with the Tokyo Institute of Technology, Tokyo, Japan. His current research interests include logic synthesis, reconfigurable architecture, digital signal processing, embedded systems, and machine learning.

Dr. Nakahara is a member of the ACM and the Institute of Electronics, Information and Communication Engineers. He received the 8th IEEE/ACM MEMOCODE Design Contest 1st Place Award in 2010, the SASIMI Outstanding Paper Award in 2010, the IPSJ Yamashita SIG Research Award in 2011, the 11th FIT Funai Best Paper Award in 2012, the 7th IEEE MCSoC-13 Best Paper Award in 2013, and the ISMVL2013 Kenneth C. Smith Early Career Award in 2014. He was the Workshop Chairman for the International Workshop on Post-Binary Ultra Large-Scale Integration Systems in 2014, 2015, 2016, and 2017. He served as the Program Chairman for the International Symposium on 8th Highly Efficient Accelerators and Reconfigurable Technologies in 2017.

Shinya Takamaeda-Yamazaki (M'13) received the B.E., M.E., and D.E. degrees from the Tokyo Institute of Technology, Tokyo, Japan, in 2009, 2011, and 2014, respectively.

From 2014 to 2016, he was an Assistant Professor with the Nara Institute of Science and Technology, Ikoma, Japan. Since 2016, he has been an Associate Professor with Hokkaido University, Sapporo, Japan. His current research interests include field-programmable gate array computing, high-level synthesis, and machine learning.

Mr. Takamaeda-Yamazaki is a member of the Institute of Electronics, Information and Communication Engineers and the Information Processing Society of Japan. From 2011 to 2014, he was a Japan Society for the Promotion of Science Research Fellow (DC1).

Masayuki Ikebe (M'14) received the B.S., M.S., and Ph.D. degrees in electrical engineering from Hokkaido University, Sapporo, Japan, in 1995, 1997, and 2000, respectively. From 2000 to 2004, he was with the Electronic Device Laboratory, Dai Nippon Printing Corporation, Tokyo, Japan, where he was involved in the research and development of wireless communication systems and image processing systems. He is currently an Associate Professor with the Graduate School of Information Science and Technology, Hokkaido University. His current research interests include CMOS image sensor and analog circuits.

Dr. Ikebe is a member of the Institute of Electronics, Information and Communication Engineers.

Tetsuya Asai (M'99) received the B.S. and M.S. degrees in electronic engineering from Tokai University, Hiratsuka, Japan, in 1993 and 1996, respectively, and the Ph.D. degree from the Toyohashi University of Technology, Toyohashi, Japan, in 1999.

He is currently a Professor with the Graduate School of Information Science and Technology, Hokkaido University, Sapporo, Japan. His current research interests include developing intelligent integrated circuits and their computational applications, emerging research architectures, deep learning accelerators, and device-aware neuromorphic very large-scale integrations.

Tadahiro Kuroda (M'88–SM'00–F'06) received the Ph.D. degree in electrical engineering from The University of Tokyo, Tokyo, Japan, in 1999.

In 1982, he joined Toshiba Corporation, Kawasaki, Japan, where he designed CMOS, SRAMs, and application-specific integrated circuits (ASICs). From 1988 to 1990, he was a Visiting Scholar with the University of California at Berkeley, Berkeley, CA, USA, where he conducted research in the field of VLSI CAD. In 1990, he was back to Toshiba, and involved in the research and development of BiCMOS/ECL ASICs, high-speed CMOS large-scale integrated circuits (LSIs) for telecommunications, and low-power CMOS LSIs for mobile applications. He invented a variable threshold-voltage CMOS technology to control threshold voltage through substrate bias, and applied it to a discrete cosine transformation core processor in 1995. He also developed a variable supply-voltage scheme to control power supply voltage by an embedded dc–dc converter, and employed it to a microprocessor core and an MPEG-4 chip in 1997. In 2000, he moved to Keio University, Yokohama, Japan, where he has been a Professor since 2002. He was a MacKay Professor with the University of California at Berkeley in 2007. He has authored over 400 technical publications, including the 37 International Solid-State Circuits Conference (ISSCC) papers, the 28 VLSI Symposia papers, the 19 Custom Integrated Circuits Conference (CICC) papers, and the 17 Asian Solid-State Circuits Conference (A-SSCC) papers. He wrote 22 books/chapters and filed over 200 patents. His current research interests include low-power, high-speed CMOS design, 3-D integration using near-field coupling, and artificial intelligence.

Dr. Kuroda is an elected AdCom Member and an Institute of Electronics, Information and Communication Engineers Fellow. He was a recipient of the 2005 P&I Patent of the Year Award, the 2007 ASP-DAC Best Design Award, the 2009 Institute of Electronics, Information and Communication Engineers (IEICE) Achievement Award, and the 2011 IEICE Society Award. He serves as the Executive Committee Chair for the Symposium on VLSI Technology and Circuits and the Steering Committee Chair for A-SSCC. He served as the Vice Chair for the Asia and South Pacific Design Automation Conference (ASP-DAC), the Sub-Committee Chair for A-SSCC, the International Conference on Computer-Aided Design, the Solid State Devices and Materials, and the VLSI Design, Automation and Test, and the Technical Program Committee Member for ISSCC, the Symposium on VLSI Circuits, CICC, DAC, ASP-DAC, the International Symposium on Low Power Electronics and Design, SSDM, the International Symposium on Quality Electronic Design, and other international conferences. He was a Distinguished Lecturer and Representative of Region 10 for the IEEE Solid-State Circuits Society.

Masato Motomura (M'11) received the B.S. and M.S. degrees and the Ph.D. degree in electrical engineering from Kyoto University, Kyoto, Japan, in 1985, 1987, and 1996, respectively.

He joined NEC Central Research Laboratories, Kawasaki, Japan in 1987, where he was involved in various hardware architectures, including string search engines, multithreaded on-chip parallel processors, DRAM-field-programmable gate array hybrid systems, memory-based processors, and reconfigurable systems. From 2001 to 2008, he was with NEC Electronics, Kawasaki, where he led business development of dynamically reconfigurable processor that he invented. He was also a Visiting Researcher with the MIT Laboratory for Computer Science, Cambridge, MA, USA, from 1991 to 1992. He has been a Professor with Hokkaido University, Sapporo, Japan, since 2011. His current research interests include reconfigurable and parallel architectures for deep neural networks and intelligent computing.

Dr. Motomura is a member of the Institute of Electronics, Information and Communication Engineers (IEICE), the Information Processing Society of Japan (IPSJ), and Engineering Academy of Japan. He was a recipient of the IEEE Journal of Solid-State Circuits Annual Best Paper Award in 1992, the IPSJ Annual Best Paper Award in 1999, and the IEICE Achievement Award in 2011.