A Neuromorphic Chip Optimized for Deep Learning and CMOS Technology With Time-Domain Analog and Digital Mixed-Signal Processing

Daisuke Miyashita, *Member, IEEE*, Shouhei Kousai, *Member, IEEE*, Tomoya Suzuki, and Jun Deguchi, *Member, IEEE*

***Abstract*— Demand for highly energy-efficient coprocessor for the inference computation of deep neural networks is increasing. We propose the time-domain neural network (TDNN), which employs time-domain analog and digital mixed-signal processing (TDAMS) that uses delay time as the analog signal. TDNN not only exploits energy-efficient analog computing, but also enables fully spatially unrolled architecture by the hardware-efficient feature of TDAMS. The proposed fully spatially unrolled architecture reduces energy-hungry data moving for weight and activations, thus contributing to significant improvement of energy efficiency. We also propose useful training techniques that mitigate the non-ideal effect of analog circuits, which enables to simplify the circuits and leads to maximizing the energy efficiency. The proof-of-concept chip shows unprecedentedly high energy efficiency of 48.2 TSop/s/W.**

***Index Terms*— Analog computing, binarized neural network (BNN), convolutional neural network (CNN), deep learning, neuromorphic computing, time domain.**

I. Introduction

Convolutional neural networks (CNNs) have not only demonstrated state-of-the-art performance in image classification [1] but also empower many other applications and algorithms such as reinforcement learning [2]. However, the computational complexity of CNNs has steadily increased. For example, ResNet [3] set a new record in image classification accuracy using 11.3 billion multiply-accumulate (MAC) or *synaptic* operations for an inference task and 230 MB of memory to store the weights in its 152-layer network. In order to perform the processing for these large networks in mobile and real-time embedded systems, energy-efficient coprocessor that executes full or part of CNN computations is required.

It has recently been reported that binarized neural networks (BNNs) [4] that use binary activations and weights can achieve near state-of-the-art accuracy (e.g., 99.04% for MNIST [5] and 89.85% for CIFAR-10 [6]). This technique is obviously preferable to realizing energy-efficient hardware. By applying the technique, the multiplier can be replaced with simple XNOR (or XOR depending on the definition of the polarity) bit operation. And the nonlinear activation function is performed by just picking up the sign bit of accumulated data [7]. Binarization improves efficiency by reducing bandwidth for activations and weights as well as simplifying computations.

Fig. 1 shows the proposed concept. A standard CNN shown in Fig. 1(a) is composed of several layers, and convolution and activation functions are applied in each layer. In BNN shown in Fig. 1(b), the convolution function is computed by XNOR and summation (SUM) operations instead of MAC operations. Binarization is employed as the activation function. The batch normalization (BatchNorm) technique [8] that is often utilized for accelerating training is simply computed by adding offset [9] in BNN as explained in the following section. Different from [7], which employs conventional digital signal processing for implementing BNN, we employ time-domain analog and digital mixed-signal processing (TDAMS) [10]. We refer to our implementation as a time-domain neural network or TDNN [11]. As shown in Fig. 1(c), we apply the TDNN coprocessor to the part (second layer) of the BNN in this proof-of-concept work. In the TDNN, we not use*voltage* [12], [13] but *time* as the analog signal. As shown in Fig. 1, not only data between layers are represented with 1-bit digital signals, but also intermediate results are represented with analog signals instead of multi-bit digital signals. This leads to further improvement of energy efficiency.

However, it is necessary to discuss whether low-precision analog computing is acceptable for deep learning applications. Although it is empirically demonstrated with experimental results in some papers [11]–[14], it is still an open problem, however, generalized their results may be, and this should be an important research topic in this area. In this paper, we also discuss this issue and offer a new perspective on it. Then, we propose useful training techniques that mitigate the non-ideal effect of analog circuit.

Furthermore, we propose the architecture optimized in terms of energy efficiency. It is worth considering why the human brain is capable of sophisticated perception while consuming very little energy. In the human brain, synaptic weights are obviously built into each synapse and utilized only by that synapse. In contrast, in von-Neumann type processors and most other processors including [15] and [16], which are neuromorphic chips, processing elements (PEs) are spatially

Manuscript received February 10, 2017; revised May 10, 2017; accepted May 27, 2017. Date of publication June 27, 2017; date of current version September 21, 2017. This paper was approved by Guest Editor Jaeha Kim. *(Corresponding author: Daisuke Miyashita.)*

The authors are with Toshiba Corp., Kawasaki 212-8520, Japan (e-mail: daisuke1.miyashita@toshiba.co.jp).

Digital Object Identifier 10.1109/JSSC.2017.2712626

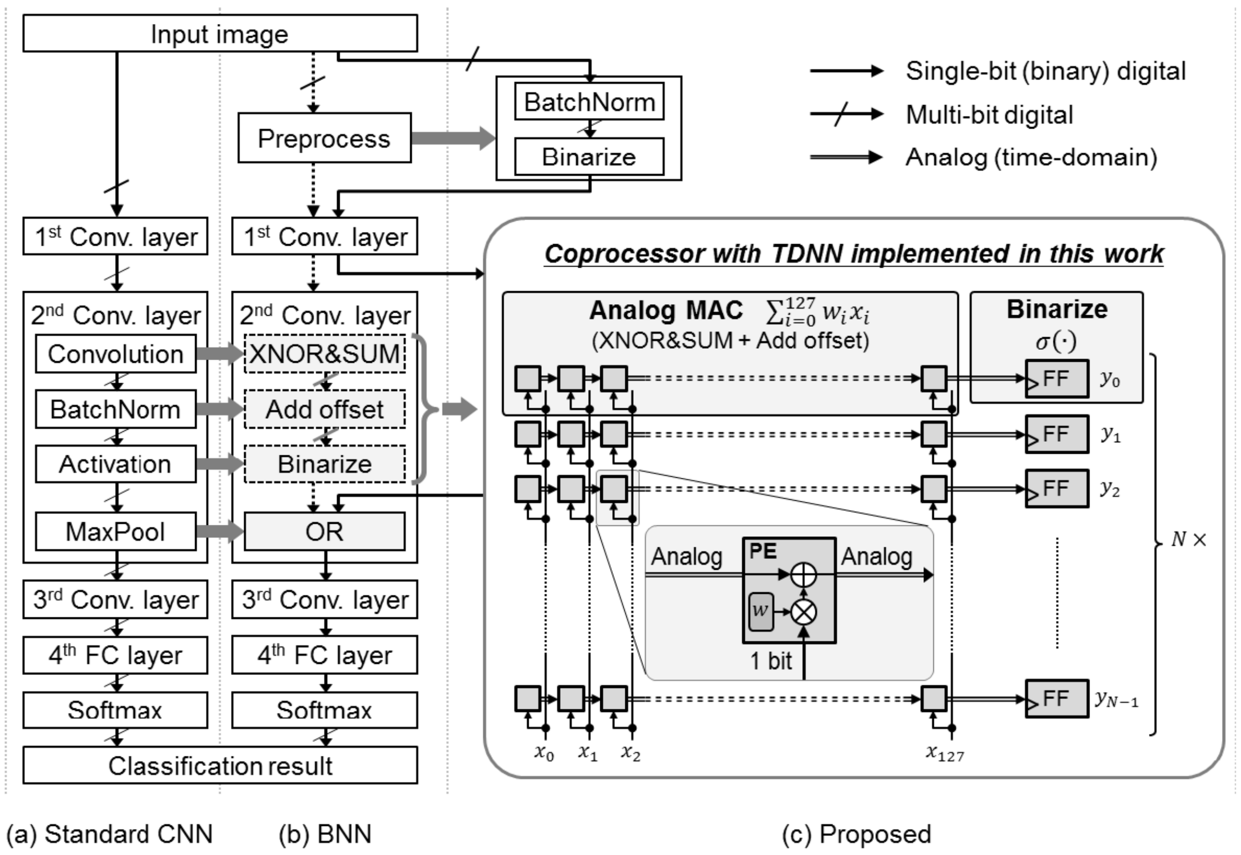


Fig. 1. (a) Standard convolutional neural network (CNN). (b) Binarized neural network (BNN). (c) Proposed TDNN coprocessor. Convolution, batch normalization (BatchNorm), and activation functions in standard CNN are replaced with XNOR and SUM, adding offset, and binarization functions in BNN, and the proposed TDNN coprocessor performs those functions with analog computing.

separated from memory banks that store weights, making the movement of weights from memory to PEs inevitable. In the proposed architecture, mimicking the architecture of the human brain, every weight has its own PE nearby for exclusive use, as shown in Fig. 1(c). Then we can completely eliminate the energy consumption for loading weight from memory. The proposed architecture reduces memory or register accesses for activations as well by broadcasting an activation to multiple PEs that reuse the activation.

In this paper, we present the detail of the proposed TDNN technique using a concrete example where the proposed TDNN technique is applied to one layer of a BNN designed for MNIST [11], and propose some additional techniques that make it possible to put BNNs with analog computing into practical use. In Section II, we provide an overview of the BNN and propose some techniques such as a method to binarize input data of the first layer. Section III describes how BNN is implemented by TDAMS in detail. We also discuss the negative effect of low-precision analog computing, and propose techniques to mitigate it. Section IV details the proposed fully spatially unrolled architecture that improves the energy efficiency. Section V describes the design of the fabricated prototype chip whose experimental results are presented in Section VI. Finally we conclude in Section VII.

II. Binarized Neural Network

In this section, we first provide an overview of the binarized neural network proposed in [4]. Next, we show a method to apply the batch normalization technique [8] to BNN and propose a way to binarize the input data of the first layer. Then we present the network structure we use in this paper.

A. Overview

BNNs are neural networks with binary (1 bit) weights and activations for inference, approaching state-of-the-art classification accuracy [4]. As is usual in CNNs, each convolution and fully connected layer of BNN performs matrix operations that distill down to computing many operations of the form

$$y = \sigma\left(\sum_{i=0}^{l-1} w_i x_i\right) \tag{1}$$

where $x_i \in \{-1, +1\}$ is the input, $w_i \in \{-1, +1\}$ is the weight for the specified input, l is the accumulation size (e.g., $l = 128$ in Fig. 1), and $\sigma(\cdot)$ is the binarization function defined as follows:

$$\sigma\ (x) = \text{Binarize}(x) = \begin{cases} +1, & \text{if } x \geq 0 \\ -1, & \text{else} \end{cases} \tag{2}$$

and $y \in \{-1, +1\}$ is called activation. Since x_i is also the activation produced at the previous layer, we hereinafter refer to x_i and y as the input activation and the output activation, respectively. Because BNNs use 1-bit representation for the weight and the input activation, a multiplication of $w_i \times x_i$ can be performed with an XNOR gate instead of a bulky digital multiplier. Note that, in our implementation, we use a 2-input 2-output switch, or an analog XNOR circuit, as described later. Here we define digital *high* as "+1," and digital *low* as "−1" (not "0"). Even though the product $w \times x$ is always $+1$ or -1, that is, each input of accumulator contains only 1-bit information, the sum of many 1-bit values becomes a multi-bit value, as shown in Fig. 1(b). Thus, the accumulator's resolution needs to be large enough to avoid saturation. The nonlinear activation function Binarize(x) can be realized by just picking the sign bits from the accumulated results using a single-bit flip-flop.

B. Batch Normalization for BNN

In order to make training of BNN stable, a batch normalization technique [8] is essential. The batch normalization function normalizes input data to be zero mean and unity variance in a mini-batch, and then, scales and shifts them with learnable parameters. By this technique, the distribution of each layer's inputs does not change largely during training, while allowing arbitrary mean and variance values for the next layer's input. In the training phase, mean (μ_B) and variance (σ_B) of input data are calculated for each mini-batch, and scale (γ) and shift (β) parameters are updated. On the other hand, in the test phase, all the μ_B, σ_B, γ, and β are fixed, and the following element-wise function that we defined as $\text{BatchNorm}_{\gamma,\beta}(x)$ is executed:

$$\text{BatchNorm}_{\gamma,\beta}(x) \equiv \frac{\gamma\ (x-\mu_B)}{\sqrt{\sigma_B^2}} + \beta. \tag{3}$$

Combining (3) with the following (2) [see Fig. 1(b)], it becomes equivalent to simply adding constant offset [9] as follows, and this can be performed using the bias term in convolution or fully connected layer

$$\begin{aligned}&\text{Binarize}(\text{BatchNorm}_{\gamma,\beta}(x))\\ &\quad= \text{Binarize}\left(\frac{\gamma}{\sqrt{\sigma_B^2}}\left\{x-\mu_B+\frac{\sqrt{\sigma_B^2}}{\gamma}\beta\right\}\right)\\ &\quad= \text{Binarize}\left(x-\mu_B+\frac{\sqrt{\sigma_B^2}}{\gamma}\beta\right) = \text{Binarize}\,(x+\text{offset})\,.\end{aligned} \tag{4}$$

C. Binarization of First Layer Input

In BNN, how to handle the input of the first layer is an issue because in general it is not binary but multi-bit (e.g., 8-bit gray scale in this paper). In [4], the way to calculate multiplication of a multi-bit input value and a binary weight using XNOR and bit shift operation is provided. In [16], additional convolution filters are employed just for binarizing input data. As shown in Fig. 1(c), we propose to employ a preprocessing layer where batch normalization and binarization are applied to the input data in order to prepare the binarized input data for the first layer. The preprocessing layer is simply implemented by adding offset followed by binarization in the inference phase using the above-explained technique.

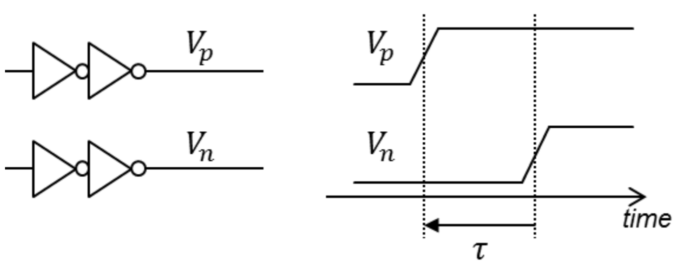


Fig. 2. Definition of the time-domain analog signal.

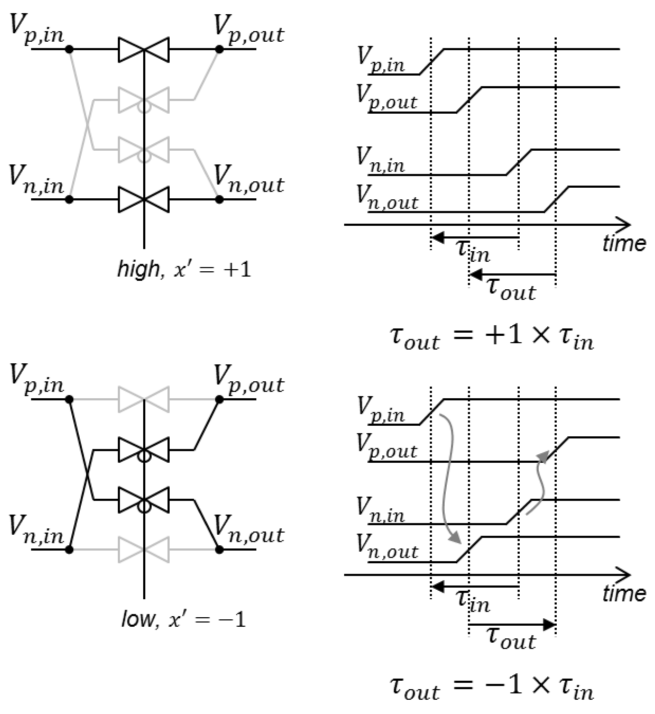


Fig. 3. Circuit diagram for the multiplication function.

D. Network Structure

Table I shows the network structure employed in this paper for the MNIST handwritten digit recognition data set. In the training phase, we apply max pooling, which downsamples the input by picking the maximum value within a 3×3-size filter with a stride of two, before binarization to avoid losing information. In the test phase, however, we apply the max pooling after binarization because Binarize(max $(x_0, \ldots, x_8)$) = max (Binarize(x_0), ..., Binarize(x_8)) when $x_i \in \{+1, -1\}$ and because the max function for binary arguments is more efficiently executed with a logical OR function since $\max(x_0^B, \ldots, x_8^B) = \text{or}(x_0^B, \ldots, x_8^B)$ [9], where superscript B indicates binary.

III. TIME-DOMAIN ANALOG AND DIGITAL MIXED-SIGNAL PROCESSING

In this section, we detail how the computations (1) for BNNs are efficiently performed by the TDAMS [10] technique.

TABLE I

CONVOLUTIONAL NEURAL NETWORK STRUCTURE ON MNIST EMPLOYED IN THIS WORK. NOTATIONS ARE (OUTPUT CHANNEL, ROW, COLUMN) FOR THE INPUT SHAPE, (OUTPUT CHANNEL, INPUT CHANNEL, KERNEL HEIGHT, KERNEL WIDTH) FOR THE KERNEL SHAPE OF CONVOLUTION AND FULLY CONNECTED LAYERS, AND (KERNEL HEIGHT, KERNEL WIDTH) FOR THE KERNEL SHAPE OF POOLING LAYERS. SINCE THE SECOND LAYER IS DIVIDED INTO 4 GROUPS, THE ACCUMULATION LENGTH IS $(32/4) \times 3 \times 3 = 72$

	Training	Inference	Input shape	Kernel shape	
Preprocess	BatchNorm	AddOffset	(1, 28, 28)		
	Binarize	Binarize	(1, 28, 28)		
1st layer	Convolution	Convolution	(1, 28, 28)	(1, 32, 3, 3)	
	MaxPool	AddOffset	(32, 26, 26)		
	BatchNorm	Binarize	(32, 26, 26)		
	Binarize	MaxPool	(32, 26, 26)	(3, 3)	stride=2
2nd layer	Convolution	Convolution	(32, 13, 13)	(64, 32, 3, 3)	group=4
	MaxPool	AddOffset	(64, 11, 11)		
	BatchNorm	Binarize	(64, 11, 11)		
	Binarize	MaxPool	(64, 11, 11)	(3, 3)	stride=2
3rd layer	Convolution	Convolution	(64, 5, 5)	(128, 64, 3, 3)	
	MaxPool	AddOffset	(128, 3, 3)		
	BatchNorm	Binarize	(128, 3, 3)		
	Binarize	MaxPool	(128, 3, 3)	(3, 3)	stride=2
4th layer	Fully connected	Fully connected	(128, 1, 1)	(10, 128, 1, 1)	
	Softmax	Softmax			

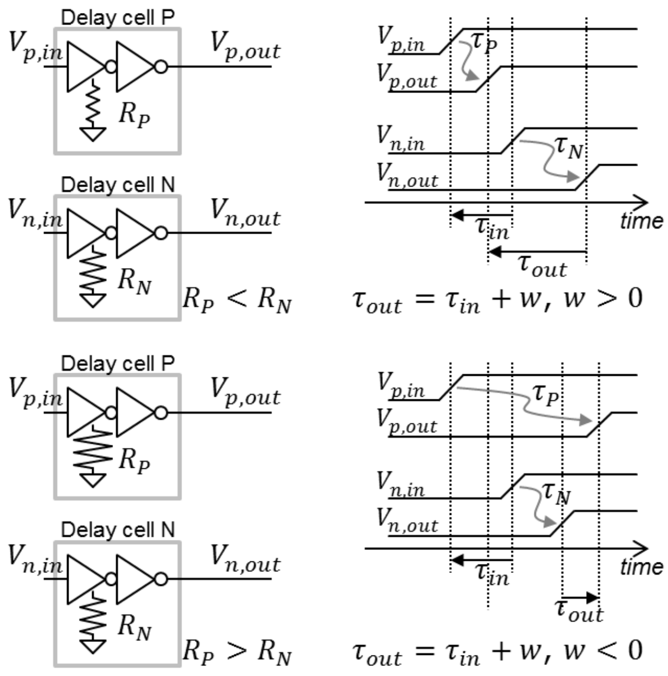


Fig. 4. Circuit diagram for the addition and subtraction functions.

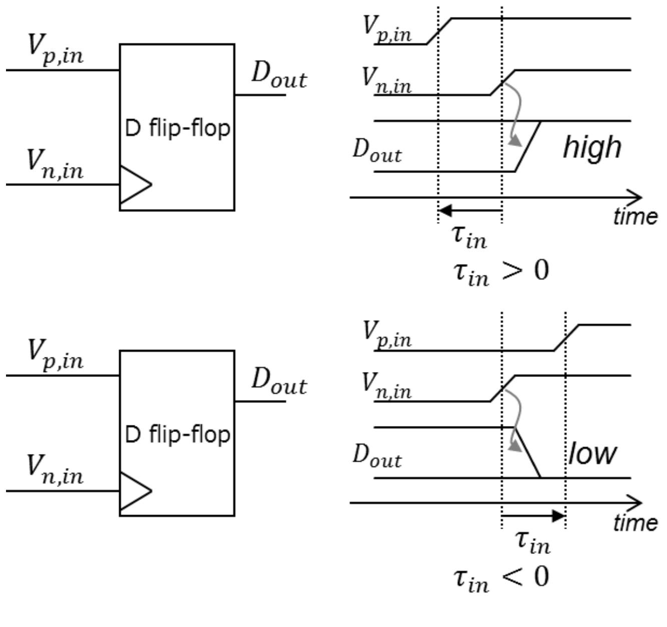


Fig. 5. Circuit diagram for the binarization function.

TDAMS uses not *voltage* [12], [13] but *time* as analog signal, and has the following features: 1) the same as for voltage domain analog computing, energy efficiency, and hardware efficiency are higher than for digital signal processing in low-precision computation [17]; 2) better suited to technology scaling than voltage domain analog computing; and 3) because TDAMS circuitry is composed of the same logic gates as digital signal processing, design methodology with existing EDA tools for digital signal processing is applicable, and therefore, large-scale integration is easier [10].

A. Computations in TDAMS

In this paper, as shown in Fig. 2, we use the time difference of rising edges of two nodes for representing a value.

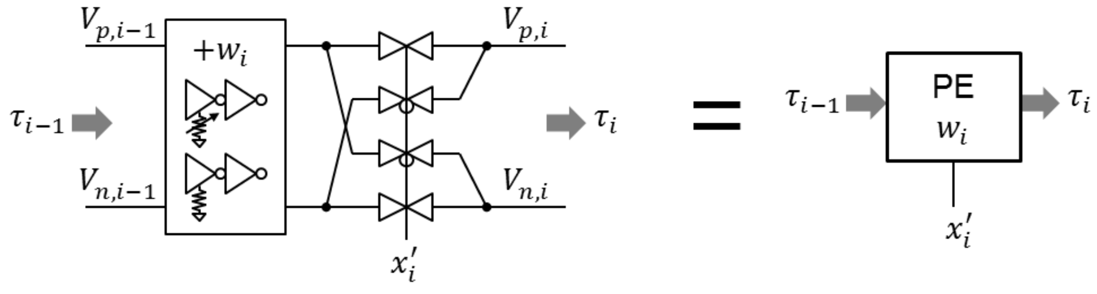
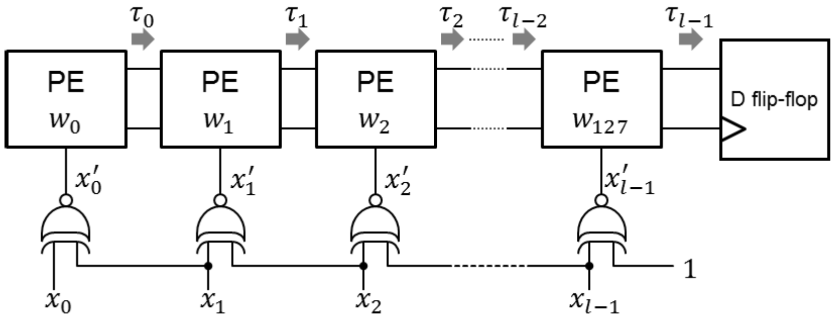


Fig. 6. Circuit diagram of the proposed PE.



x_i	x_{i+1}	$x_i{}'$
-1 Low	-1 Low	$+1$ High
-1 Low	$+1$ High	-1 Low
$+1$ High	-1 Low	-1 Low
$+1$ High	$+1$ High	$+1$ High

Fig. 7. Circuit diagram for performing $y = \text{Binarize}\left(\sum_{i=0}^{l-1} w_i x_i\right)$. The table on the right is the truth table of an XNOR preprocessing circuit.

Regarding polarity, we define that if V_p rises earlier than V_n, it means the value is positive, and otherwise the value is negative. And the amount of time difference τ represents the absolute value. Fig. 3 shows circuit diagrams for multiplication. Because the activations are either $+1$ or -1 in BNN, all we need is $\times\ (+1)$ and $\times(-1)$. From the definition of the polarity, the computations of $\times\ (+1)$ and $\times\ (-1)$ are realized by making the input–output path straight connection and cross connection, respectively. Fig. 4 shows circuit diagrams for addition and subtraction functions. τ_P and τ_N are the delay time of the delay cell P and that of the delay cell N, respectively. When $\tau_P - \tau_N$ is defined as w taking into account the polarity, this circuit calculates $\tau_{\text{out}} = \tau_{\text{in}} + w$. Since τ_P and τ_N are approximately proportional to the foot resistance R_P and R_N, respectively, we can set w to a value corresponding to either $+1$ or -1 by using a variable resistor. Fig. 5 shows circuit diagrams for the binarization function. A standard D flip-flop is employed with D port and CK port connected to V_p and V_n, respectively. This circuit produces $high$ $(+1)$ or low (-1) depending on whether the rise edge of V_p is earlier or later than that of V_n. Then, we show how to compute (1) using the elements explained in Figs. 3–5. As shown in Fig. 6, τ_i is represented in a recurrence relation form of $\tau_i = (\tau_{i-1} + w_i)x'_i$, where we define the time difference of the rise edges of ith nodes $V_{p,i}$ and $V_{n,i}$ as τ_i. Let $\tau_0 = 0$, this recurrence relation is solved to $\tau_{l-1} = \sum_{i=0}^{l-1} w_i(\prod_{j=i}^{l-1} x'_j)$. On the other hand, what we would like to calculate is $\tau_{l-1} = \sum_{i=0}^{l-1} w_i x_i$ from (1). Then we need a conversion of $x_i = \prod_{j=i}^{l-1} x'_j$. Since $x_i = \prod_{j=i}^{l-1} x'_j = x'_i \prod_{j=i+1}^{l-1} x'_j = x'_i x_{i+1}, x'_i = x_i/x_{i+1} = x_i \times x_{i+1}$. The second equal sign is due to the fact that x_{i+1} is either $+1$ or -1. Thus, x'_i can be produced from x_i and x_{i+1} using XNOR gate, as shown in Fig. 7. These XNOR gates for input activation preprocessing are shared across multiple rows as detailed in the following section. Finally, the overall circuit shown in Fig. 7 performs (1). Note that the accumulation function is performed in the time domain, which is analog. Therefore, although accumulating many 1-bit values produces a multi-bit value, we need only a single pair of wires to contain the multi-bit value. In digital implementation, multi-bit adders are required for the same function. Besides, adding offset function (4) is performed with the same circuit by making some of xs fixed to $+1$, as shown in Fig. 8.

B. *Effect of Analog Computing*

Applying analog computing for deep learning hardware has been proposed in many papers such as [11]–[14], [18] on the premise that local computations in deep learning do not require high precision and analog computation is more efficient for low-precision computation [17]. However, regarding the former premise, to what extent we can reduce the precision is still an open issue.

Analog computing introduces two categories of additional non-idealities. One category includes the non-idealities depending on individual chips or individual PEs. Dominant errors in this category are caused by device mismatches. The other category includes the temporally varying non-idealities. A dominant factor is temporal noise from the circuits. Hereinafter, we refer to the first category as mismatch error and the second category as temporal noise.

It is helpful to discuss the quantization error at first although this is not the error due to analog computing. As long as the input data are the same, the quantization errors do not change regardless of the time and place at which the computation is

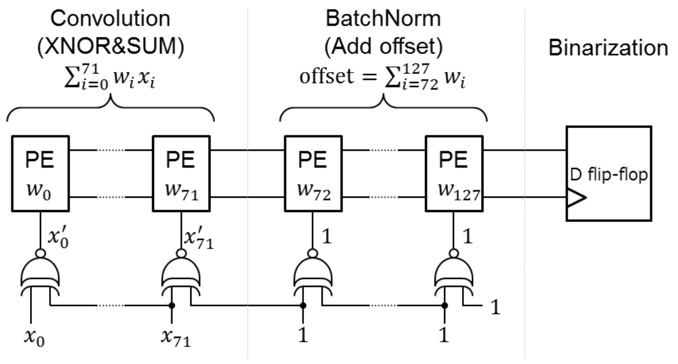


Fig. 8. Circuit diagram for the batch normalization. Since the accumulation length of the second layer is 72, remaining 56 PEs are used for batch normalization function by setting $\{x_i : i \in [72, 127]\}$ to 1.

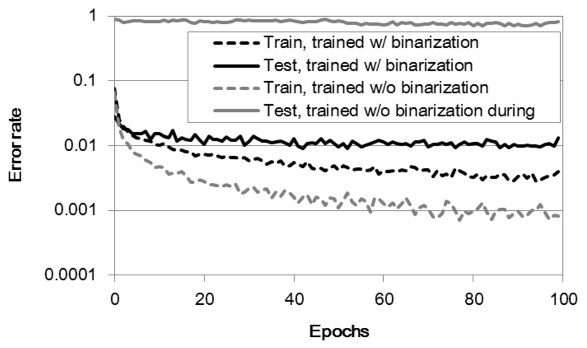


Fig. 9. Training curve of the BNN shown in Table I on the MNIST data set.

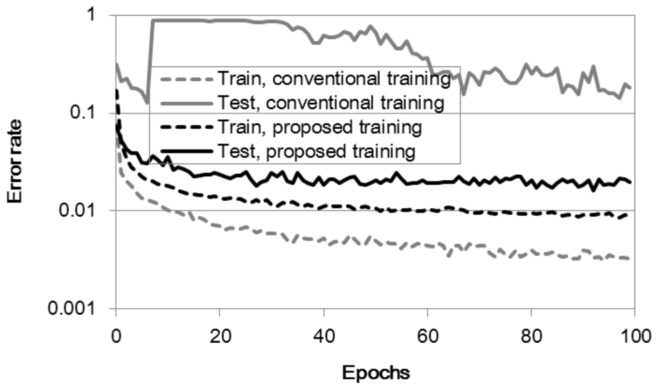


Fig. 10. Training curve of the BNN with mismatch error.

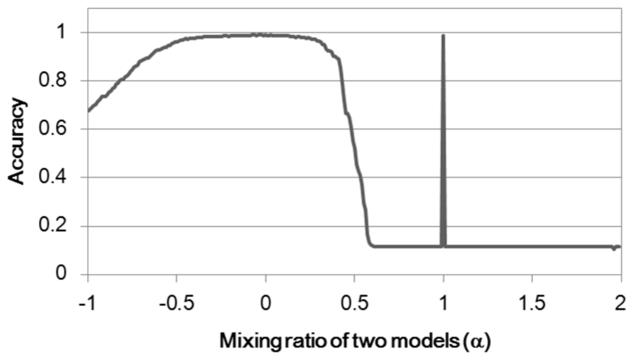
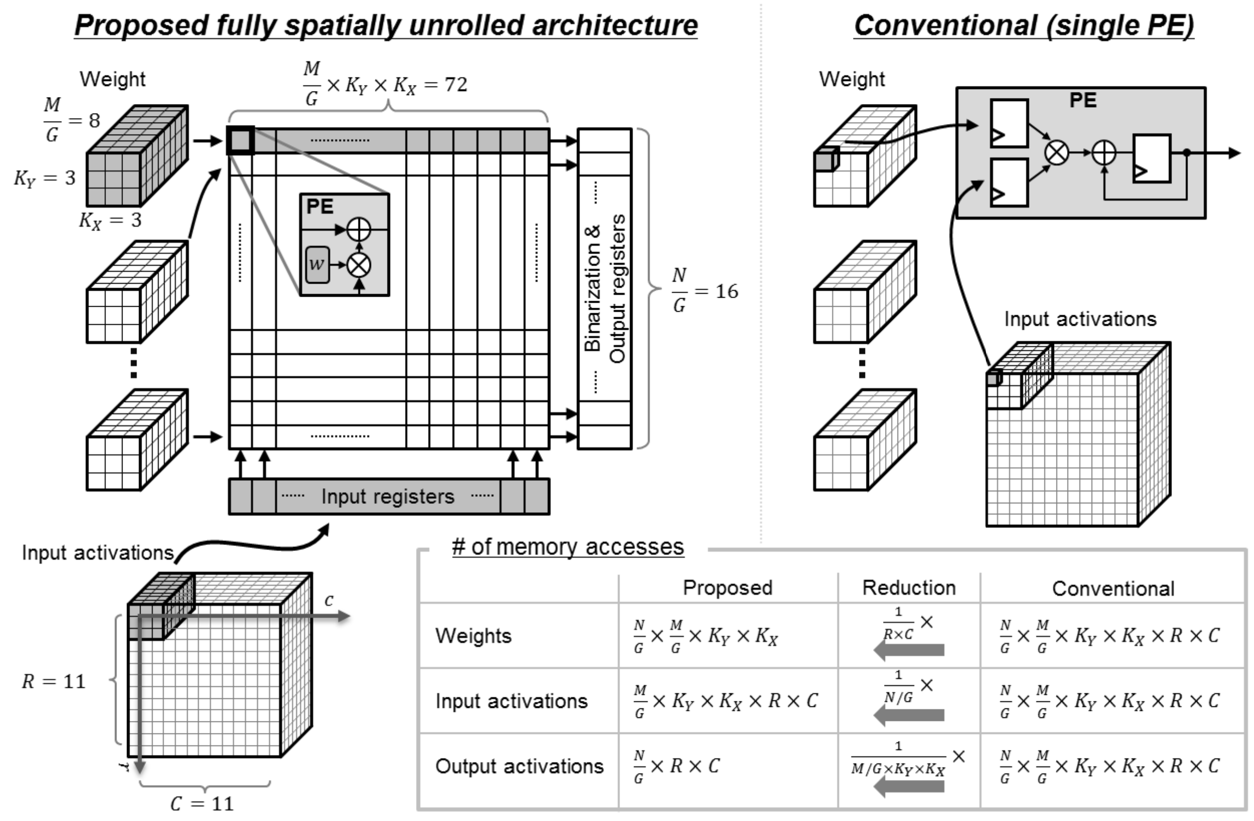


Fig. 11. Accuracy versus the mixing ratio of two models.

executed. In other words, this error is completely reproducible. This means that it is possible to train the network while taking quantization errors into account. By applying binarization during forward path computation in the training phase, the network is optimized for the condition where quantization errors are introduced. When it is sufficiently generalized, the feature of robustness to the quantization error is maintained during the test phase, whereas all the input data are unseen in the training phase [4]. Fig. 9 shows the simulation results when this technique is applied to the network shown in Table I. This type of figure is referred to as a training curve, which is common in deep learning community. The horizontal axis represents the training "epoch," which means how many times all the training data are processed. For example, since there are 60 000 training images in the MNIST data set, all 60 000 images are processed in each epoch. The vertical axis represents the classification error rate of the trained neural network, or the *model*, over the training data set and the test data set. Fig. 9 compares the training curves where binarization of weights is applied and not applied during the training phase using training data set, labeled as trained w/binarization and trained w/o binarization, respectively. In the test phase using test data set, binarization of weights is applied for both cases. When the model is trained without binarization, the error rate is very high for the test data. On the other hand, the model trained with binarization maintains low error rate for the test data because it learns how to mitigate the effect of the quantization error.

Next, we discuss the mismatch error introduced by analog computing. Different from quantization errors, even when the input data are the same, the mismatch errors vary depending on the chip (or PE) where the computations run. If training is performed for an individual chip rather than sharing a specific pre-trained model with many chips, the effect of this mismatch error is the same as that of the quantization errors, and we can take the same measure to alleviate it. However, we do not think individual training is practical because the conventional training of deep learning with a huge amount of data is too costly. On the other hand, designing a circuit with sufficiently small mismatch error, which is conventional approach in analog circuit design, requires area penalty because mismatches depend inversely on area.

In this paper, we present another approach, namely, to train the network so that it is robust to mismatch errors. In order to realize this, we propose to introduce mismatch errors in forward propagation during the training phase of the stochastic gradient descent algorithm with back propagation [4]. Mismatch errors are modeled by adding random values to weights. The random values are sampled from a normal distribution with standard deviation of 0.7, which is determined to be large enough compared to the Monte-Carlo simulation results for the circuit shown in Fig. 6; the standard deviation over the average of the delay is 0.47 (20.2 ps/43.4 ps). Note that random values are not fixed, but new ones are sampled every iteration. Fig. 10 shows the training curves comparing the error rate curves where mismatch errors are introduced and not introduced during training. In the test phase, mismatch



	Proposed	Reduction	Conventional
Weights	$\frac{N}{G} \times \frac{M}{G} \times K_Y \times K_X$	$\frac{1}{R \times C} \times$	$\frac{N}{G} \times \frac{M}{G} \times K_Y \times K_X \times R \times C$
Input activations	$\frac{M}{G} \times K_Y \times K_X \times R \times C$	$\frac{1}{N/G} \times$	$\frac{N}{G} \times \frac{M}{G} \times K_Y \times K_X \times R \times C$
Output activations	$\frac{N}{G} \times R \times C$	$\frac{1}{M/G \times K_Y \times K_X} \times$	$\frac{N}{G} \times \frac{M}{G} \times K_Y \times K_X \times R \times C$

Fig. 12. Proposed fully spatially unrolled architecture. Due to the massively parallel architecture, the weights and activations are efficiently reused and the number of memory accesses reduces compared to a conventional single PE architecture.

errors, which are different values from those in the training phase, are introduced for both cases. In training, the error rate without mismatch is slightly better than that with mismatch. However, in the test phase, the error rate with mismatch is even better. Fig. 11 shows the simulated accuracy of the trained model given by the formula $(1 - \alpha)w_{\text{proposed}} + \alpha w_{\text{conv}}$ [19] versus α. w_{proposed} is the model trained with mismatch (proposed) and w_{conv} is the model trained without mismatch (conventional). Thus, the value at $\alpha = 0$ is the accuracy of the model trained with the proposed method, the value at $\alpha = 1$ is the accuracy of the model trained with the conventional method, and the values at other α are the accuracies of the models that are generated by linear combination of w_{proposed} and w_{conv}. Fig. 11 shows that around the model trained with the proposed method, the classification accuracy curve is flat and immune to the fluctuation of weight values, while around the model trained with the conventional method, the curve is very sharp and even slight fluctuation ruins the accuracy. These results show that the model trained with the proposed method is more robust to mismatch errors; in other words, the network learns how to mitigate the effect due to mismatch errors by the proposed method. Although emerging nonvolatile memories such as ReRAM [20] is attractive candidates for the resistor in Fig. 4, the resistance variation of ReRAM could be an issue. However, the proposed training technique has the potential to largely relax the requirement of the resistance variation.

Last, we discuss the temporal noise. Although the temporal noise is completely unreproducible, its effect is the same as that of the mismatch error. Therefore, we propose to design a circuit so that temporal noise is sufficiently less than the mismatch error. In TDAMS, this criterion is naturally satisfied. The ratio of noise to signal (N/S) from mismatch and N/S from temporal noise are 0.47 and 0.03 from the simulation of the circuit shown in Fig. 6.

Global variations such as process including R and C, supply voltage, and temperature (PVT) do not directly deteriorate the computation precision because we employ differential signaling.

Regarding the interference due to capacitive coupling, because the input activations and weights are all static during computation and the differential signal of $\tau_i$is the only dynamic signal to be taken care of in Fig. 6, we can easily mitigate the effect by shielding and common centroid layout techniques.

IV. Fully Spatially Unrolled Architecture

As widely reported in the literature, such as [21]–[24], data moving accounts for most of the energy consumption. For example, optimizing ordering of computation so that the number of memory accesses is minimized allows us to reduce the overall energy consumption significantly [21]–[24]. The improvement from these techniques is often much larger than that from techniques that reduce energy consumption for computations [15], [16]. We propose to employ the fully spatially unrolled architecture where each weight memory has its own processing element, as shown in Figs. 1(c) and 12. Since

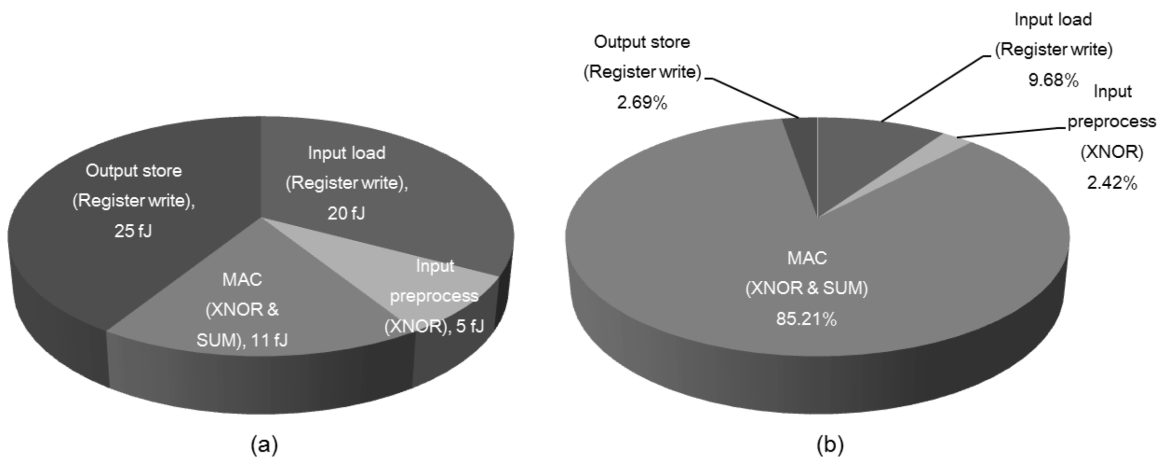


Fig. 13. Pie chart of simulated energy consumptions. (a) Energy consumption of each operation. (b) Energy consumption for the entire second layer processing of Fig. 1.

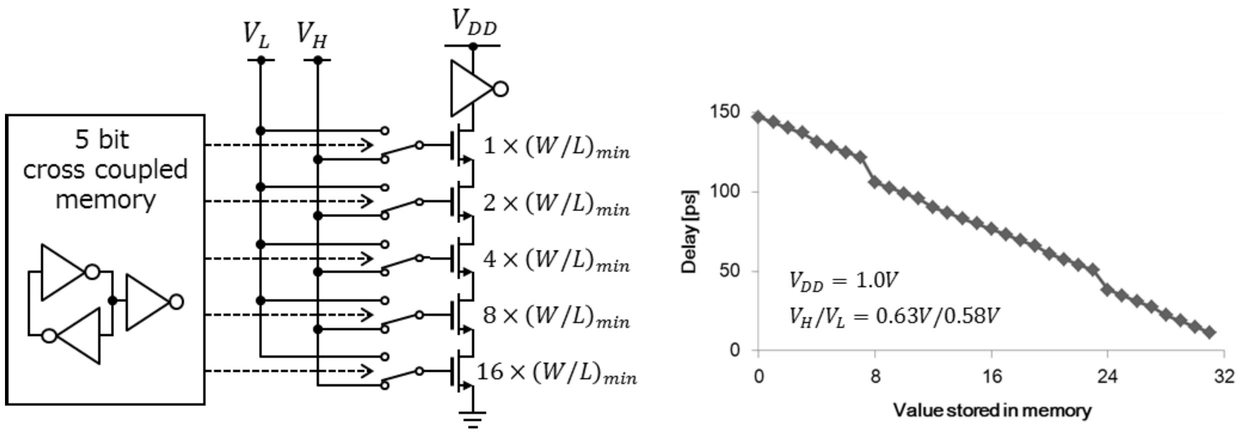


Fig. 14. Left: Schematic of the variable delay cell using 5-bit cross-coupled memory cells and nMOS resistors implemented in the test chip. Right: Simulated delay time of the variable delay cell.

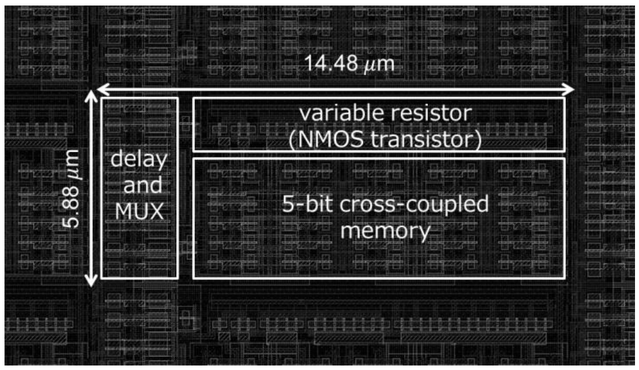

14.48 μm
5.88 μm
delay and MUX
variable resistor (NMOS transistor)
5-bit cross-coupled memory

Fig. 15. Layout design of one PE.

weights are placed close to the PEs, there is no need to move weight from memories to PEs. As well as weight, data moving for activations is also reduced by this architecture. We explain the flow for computing the one group [25] of the second layer of Table I using Fig. 12. The adding offset part is not shown for simplicity. At first, the whole weights, the size of which is $(N/G) \times (M/G) \times K_Y \times K_X$, where N, M, G, K_Y, and K_X are the numbers of output channels, input channels, groups [25], kernel height, and kernel width, respectively, are loaded to distributed weight memories. Then, the highlighted input activations within the first chunk at $(r, c) = (0, 0)$ whose size is $(M/G) \times K_Y \times K_X$ are loaded to input registers and each of them is broadcasted to N/G PEs through an XNOR gate for input activation preprocessing described in Section III-A. After completing the MAC (XNOR & SUM) computation and the binarization, N/G output activations are produced and stored to the output registers. Next, the input activations within the second chunk at $(r, c) = (0, 1)$ are loaded to input registers and the output activations are produced. This sequence is repeated $R \times C$ cycles, where R and C are the numbers of rows and columns of the output activations, to produce $(\frac{N}{G}) \times R \times C$ output activations. Compared with the conventional single PE case shown on the right in Fig. 12, where we need to load a weight and an input activation and store an intermediate accumulation result every cycle, the number of data moving for weights, input activations, and output results are reduced by $1/(R \times C)$, $1/(N/G)$, and $1/(M/G \times K_Y \times K_X)$, respectively.

Because the occurrences of data moving become much less than that of MAC (XNOR & SUM) computations by the proposed architecture, even though a single computation

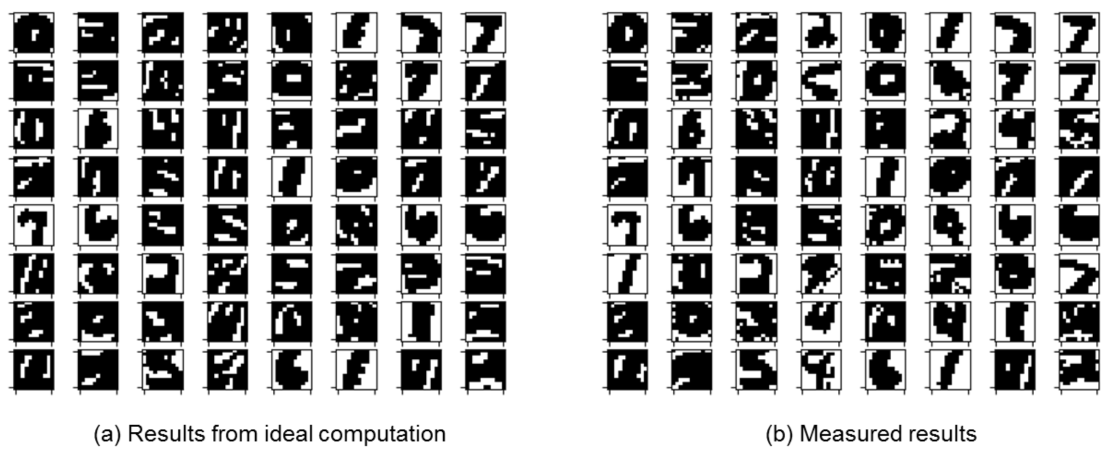


Fig. 16. Output pattern of the second layer. (a) Results from ideal computation on PC. (b) Measured result on the test chip.

consumes less energy than moving a single activation as shown in Fig. 13(a), 85% of the total energy consumption is attributable to MAC computations, as shown in Fig. 13(b). As a result, the energy efficiency reaches 12.9 fJ per synaptic operation, which is corresponding to 77 TSop/s/W.

Note that while this architecture is attractive in terms of energy efficiency, it obviously requires a huge amount of hardware resources and chip area [26]. The feature of TDNN that PEs are tiny compared to digital implementation is preferable from this viewpoint as well. For larger networks such as ResNet [3], even if there are not enough PEs to store the all weights for a layer and therefore memory accesses are required for replacing weights, the proposed architecture is still energy efficient because the memory accesses for loading weights is reduced by $1/(R \times C)$ compared to the conventional case as explained above.

V. Implementation

Each PE shown in Fig. 6 is composed of 4 inverters, 4 transmission gates, and a variable resistor with a memory. The variable resistor with a memory may possibly be realized using e.g., ReRAM [20] in the future. In this paper, we use nMOS transistors in the triode region as the variable resistor whose gates are connected to either V_L or V_H according to the digital data stored in 5-bit cross-coupled memory cells, as shown in Fig. 14. This 5-bit variable resistor is used for realizing 1-bit weight for BNN. The auxiliary bits can be used to calibrate the delay mismatch [11]. In this paper, we do not apply the calibration because the network is already trained so that the effect due to mismatch error is mitigated as proposed in Section III-B. All the data stored in 5-bit cross-coupled memory cells are, therefore, fixed to either all zero “00000” or all one “11111” depending on whether the w value is -1 or $+1$. Fig. 15 shows the layout pattern of a processing unit. As shown, each memory is placed close to its dedicated processing element.

VI. Experimental Results

A. Classification

In order to verify the capability of TDNN to perform the inference computation, we run the BNN shown in Table II

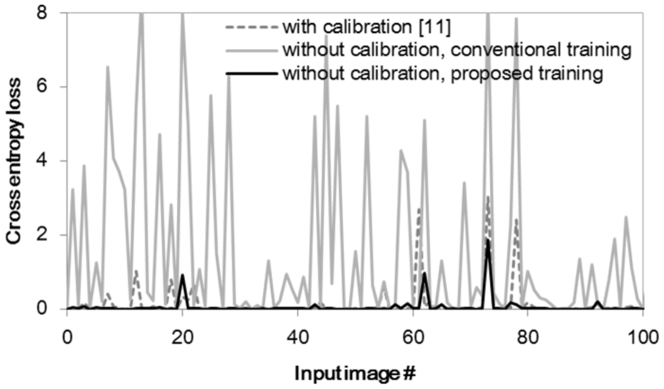


Fig. 17. Cross-entropy loss values for 100 images computed with the measured results.

on the MNIST handwritten digit recognition dataset. After training using the proposed method, the accuracy for test data is 98.4%. The second layer is performed using the test chip and the other layers are processed on a PC. Fig. 16 shows some randomly sampled measured results of the output from the second layer on the test chip and the corresponding simulated result on a PC. They are similar to each other, but not bit-exact because of the non-ideal analog computation. However, as shown in Fig. 17, the cross-entropy loss values, which indicate the distance between the output of the neural network and the correct answer, are sufficiently small and 99 out of 100 samples are correctly classified. As a reference, we test the network trained without mismatch during the training. The loss values of the reference case tend to become even larger than those of the proposed case. This result shows the effectiveness of the proposed training method.

B. Efficiency

Table II shows the performance summary of the test chip and comparison with other papers. Although the integration levels are different, this comparison supports the validity of our proposal. With 3.61 mm^2 core area in 65 nm process, our test chip has 32K PEs. By applying the proposed fully spatially unrolled architecture, the energy efficiency is 12$\times$ better than state-of-the-art [13]. Hardware efficiency defined by the

TABLE II
PERFORMANCE COMPARISON

	This work	[27]	[15, 16]	[13]
Technology [nm]	65	65	28	40
Chip area [mm^2]	3.61[a]	1.31[a]	430	0.012
Energy efficiency [TSop/s/W]	48.2[b]	0.402	0.039[15] 0.4[16]	3.86[c]
Hardware efficiency[d] [GE[e]/PE]	76.5	4641[a]	6.5	288

a. Core area including SRAM.
b. excludes external I/O, c. excludes CML.
d. We measure hardware efficiency by the gate number for a PE.
e. We assume 1GE: 1.44 μm^2 (65 nm), 0.65 μm^2 (40 nm), 0.49 μm^2 (28 nm).

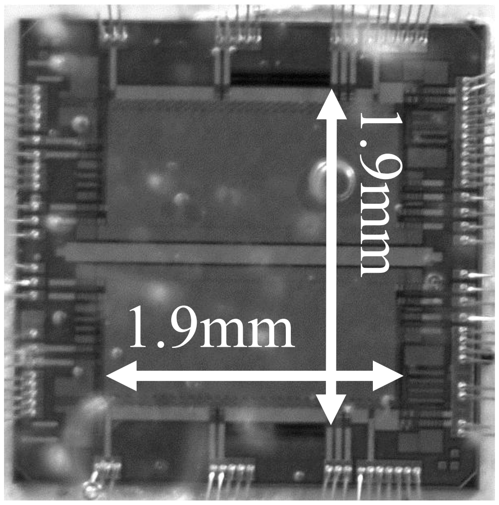


Fig. 18. Chip photograph.

equivalent gate counts for a PE, which is an important number for enabling the fully spatially unrolled architecture, is roughly 4× better than [13]. Compared to [15], [16], although hardware efficiency is 12× worse partly because of the redundant 5-bit cross-coupled memory cells, the energy efficiency is 100× better. When we assume that the variable resistor with memory is replaced with ReRAM and the ReRAM cell is ideally stacked on each PE without any area penalty, the hardware efficiency improves by 25×. Fig. 18 shows the chip photograph.

VII. CONCLUSION

In this paper, we present TDNN, which is an energy-efficient coprocessor dedicated for the inference task of deep neural networks using CMOS technology. TDNN employs time-domain analog signal processing and fully spatially unrolled architecture where every memory has its dedicated PE. In addition, we propose a training method that makes employing analog signal processing for deep learning hardware practical. The test chip demonstrates that the proposed architecture achieves energy efficiency of 48.2 TSop/s/W and hardware efficiency of 76.5 GE/PE.

ACKNOWLEDGMENT

The authors would like to thank T. Miyaba, F. Tachibana, and M. Morimoto for their support and contributions.

REFERENCES

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *Proc. IEEE Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.
[2] D. Silver *et al.*, “Mastering the game of Go with deep neural networks and tree search,” *Nature*, vol. 529, pp. 484–489, Jan. 2016.
[3] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proc. IEEE Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 770–778.
[4] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, “Binarized neural networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 29. 2016, pp. 4107–4115.
[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[6] A. Krizhevsky, “Learning multiple layers of features from tiny images,” M.S. thesis, Dept. Comput. Sci., Univ. Toronto, ON, Canada, 2009.
[7] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, “XNOR-Net: ImageNet classification using binary convolutional neural networks,” in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 525–542.
[8] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.
[9] Y. Umuroglu *et al.* (2016). “FINN: A framework for fast, scalable binarized neural network inference.” [Online]. Available: https://arxiv.org/abs/1612.07119
[10] D. Miyashita *et al.*, “An LDPC decoder with time-domain analog and digital mixed-signal processing,” *IEEE J. Solid-State Circuits*, vol. 49, no. 1, pp. 73–83, Jan. 2014.
[11] D. Miyashita, S. Kousai, T. Suzuki, and J. Deguchi, “Time-domain neural network: A 48.5 TSOp/s/W neuromorphic chip optimized for deep learning and CMOS technology,” in *Proc. IEEE Asian Conf. Solid-State Circuits*, Nov. 2016, pp. 25–28.
[12] D. Bankman and B. Murmann, “An 8-bit, 16 input, 3.2 pJ/op switched-capacitor dot product circuit in 28-nm FDSOI CMOS,” in *Proc. IEEE Asian Conf. Solid-State Circuits*, Nov. 2016, pp. 21–24.
[13] E. H. Lee and S. S. Wong, “A 2.5 GHz 7.7 TOPS/W switched-capacitor matrix multiplier with co-designed local memory in 40 nm,” in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Jan./Feb. 2016, pp. 418–419.

[14] J. Holleman, I. Arel, S. Young, and J. Lu, "Analog inference circuits for deep learning," in *Proc. IEEE Biomed. Circuits Syst. Conf. (BioCAS)*, Oct. 2015, pp. 1–4.

[15] 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.

[16] S. K. Esser *et al.*, "Convolutional networks for fast, energy-efficient neuromorphic computing," *Proc. Nat. Acad. Sci. USA*, vol. 113, pp. 11441–11446, Oct. 2016.

[17] R. Sarpeshkar, "Analog versus digital: Extrapolating from electronics to neurobiology," *Neural Comput.*, vol. 10, no. 7, pp. 1601–1638, 1998.

[18] C. Mead, *Analog VLSI and Neural Systems*. Reading, MA, USA: Addison-Wesley, 1989.

[19] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. (2016). "On large-batch training for deep learning: Generalization gap and sharp minima." [Online]. Available: https://arxiv.org/abs/1609.04836

[20] S. Yu, B. Gao, Z. Fang, H. Yu, J. Kang, and H.-S. P. Wong, "A neuromorphic visual system using RRAM synaptic devices with sub-pJ energy and tolerance to variability: Experimental characterization and large-scale modeling," in *IEDM Tech. Dig.*, Dec. 2012, pp. 10.4.1–10.4.4.

[21] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28. 2015, pp. 1135–1143.

[22] B. Murmann, D. Bankman, E. Chai, D. Miyashita, and L. Yang, "Mixed-signal circuits for embedded machine-learning applications," in *Proc. IEEE Asilomar Conf. Signals, Syst. Comput.*, Nov. 2015, pp. 1341–1345.

[23] B. Moons and M. Verhelst, "A 0.3–2.6 TOPS/W precision-scalable processor for real-time large-scale ConvNets," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2016, pp. 1–2.

[24] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, 2016, pp. 161–170.

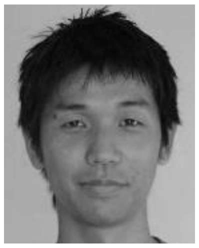
[25] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 25. 2012, pp. 1106–1114.

[26] D. Monroe, "Neuromorphic computing gets ready for the (really) big time," *Commun. ACM*, vol. 57, no. 6, pp. 13–15, 2014.

[27] L. Cavigelli and L. Benini, "A 803 GOp/s/W convolutional network accelerator," *IEEE Trans. Circuits Syst. Video Technol.*, to be published, doi: 10.1109/TCSVT.2016.2592330.

Daisuke Miyashita (M'10) received the B.S. and M.S. degrees in electronic engineering from the University of Tokyo, Tokyo, Japan, in 2001 and 2003, respectively.

In 2003, he joined the Center for Semiconductor Research and Development, Toshiba Corporation, Kawasaki, Japan, where he was involved in the design of RF and analog circuits for wireless communications. From 2015 to 2016, he was a Visiting Scholar at Stanford University, Stanford, CA, USA, where he was involved in research on the efficient implementation of deep learning algorithms on hardware. In 2017, he joined Toshiba Memory Corporation, Kawasaki, Japan. His current research interests include efficient mixed-signal/digital hardware for machine learning applications and algorithms designed for such hardware.



Shouhei Kousai (M'08) received the B.S. and M.S. degrees in electronic engineering from the University of Tokyo, Tokyo, Japan, in 1996 and 1998, respectively, and the Ph.D. degree in physical electronics from the Tokyo Institute of Technology, Meguro, Japan, in 2011.

In 1998, he joined Toshiba Corporation, Kanagawa, Japan. Since then he has been involved in the design of analog and RF circuits for wireless communications. From 2007 to 2009, he was a Visiting Scholar at the California Institute of Technology, Pasadena, CA, USA, where he was involved in research on linear power amplifiers.

Dr. Kousai served as a member of the technical program committee (TPC) of the IEEE International Solid-State Circuits Conference. He was also a TPC member of the Asian Solid-State Circuits Conference and the International Electron Devices Meeting.

Tomoya Suzuki was born in 1982. He received the M.E. degree in information systems from the Nara Institute of Science and Technology, Ikoma, Japan, in 2006.

From 2006 to 2017, he was with the Center for Semiconductor Research and Development of Toshiba Corporation as a Wireless HW/SW Engineer. In 2017, he joined the Memory Technology Research and Development Center, Toshiba Memory Corporation, Kawasaki, Japan.

Jun Deguchi (M'10) received the B.S. and M.S. degrees in machine intelligence and systems engineering and the Ph.D. degree in bioengineering and robotics from Tohoku University, Sendai, Japan, in 2001, 2003, and 2006, respectively.

In 2004, he was a Visiting Scholar at the University of California, Santa Cruz, CA, USA. In 2006, he joined Toshiba Corporation, and was involved in design of analog/RF circuits for wireless communications, CMOS image sensors, high-speed I/O, and neuromorphic chips. From 2014 to 2015, he was a Visiting Scientist at the MIT Media Lab, Cambridge, MA, USA, and was involved in research on brain/neuro science. In 2017, he was transferred to Toshiba Memory Corporation, and has been a Research Lead of an advanced circuit design section. His current research interests include semiconductor technologies required for recording/analyzing neural connections and dynamic neural activity within the brain.

Dr. Deguchi has served as a member of the technical program committee of the IEEE International Solid-State Circuits Conference since 2016.