Sensitivity-Based Error Resilient Techniques With Heterogeneous Multiply–Accumulate Unit for Voltage Scalable Deep Neural Network Accelerators

Dongyeob Shin, *Student Member, IEEE*, Wonseok Choi, *Student Member, IEEE*, Jongsun Park, *Senior Member, IEEE*, and Swaroop Ghosh, *Senior Member, IEEE*

***Abstract*— With inherent algorithmic error resilience of deep neural networks (DNNs), supply voltage scaling could be a promising technique for energy efficient DNN accelerator design. In this paper, we present an error resilient technique to enable aggressive voltage scaling by exploiting the asymmetric error resilience (sensitivity) with respect to DNN layers, filters, and channels. First-order Taylor expansion is used to evaluate the filter/channel-level weight sensitivities of large scale DNNs which accurately approximates weight sensitivities from actual error injection simulations. We also present the heterogeneous multiply-accumulate (MAC) unit based design approach where some of the MAC units are designed larger with shorter critical path delays for robustness to aggressive voltage scaling while other MAC units are designed relatively smaller. The sensitivity variations among filter weights can be leveraged to design DNN accelerator such that the computations with more sensitive weights are assigned to more robust (larger) MAC units while the computations with less sensitive weights are assigned to less robust (smaller) MAC units. Using dynamic programming, the sizes of MAC units are selected to achieve best DNN accuracy under ISO area constraint. As a result, the proposed voltage scalable DNN accelerator can achieve 34% energy savings in post layout simulations using 65 nm CMOS process with ImageNet dataset using ResNet-18 compared to state-of-the-art timing error recovery technique.**

***Index Terms*— Deep neural network resilience, timing error resilient accelerator, voltage scaling.**

Manuscript received April 29, 2019; revised July 4, 2019; accepted July 27, 2019. Date of publication August 8, 2019; date of current version September 17, 2019. This work was supported in part by the Ministry of Science and ICT (MSIT), South Korea, through the Information Technology Research Center (ITRC) Support Program supervised by the Institute for Information & Communications Technology Promotion (IITP) under Grant IITP-2019-2018-0-01433, in part by the Industrial Strategic Technology Development Program (Development of SoC Technology Based on Spiking Neural Cell for Smart Mobile and IoT Devices) under Grant 10077445, and in part by the Information Technology Research and Development Program of the Korea Evaluation Institute of Industrial Technology (Design Technology Development of Ultralow Voltage Operating Circuit and IP for Smart Sensor SoC) under Grant 10052716. This article was recommended by Guest Editor M. Ziegler. *(Corresponding author: Jongsun Park.)*

D. Shin, W. Choi, and J. Park are with the School of Electrical Engineering, Korea University, Seoul 136-701, South Korea (e-mail: shindy919@korea.ac.kr; sicnarf@korea.ac.kr; jongsun@korea.ac.kr).

S. Ghosh is with the School of Electrical Engineering and Computer Science, Pennsylvania State University, State College, PA 16802 USA (e-mail: szg212@psu.edu).

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

Digital Object Identifier 10.1109/JETCAS.2019.2933862

I. Introduction

DEEP neural networks (DNN) are playing an important role in wide applications e.g., object recognition, tracking in autonomous vehicles [1], [2] and speech recognition [3]. As recent DNNs contain millions of parameters and perform billions of computations involving large-scale dataset, energy efficient acceleration of DNNs for both mobile devices and data centers becomes crucial. Recently, various energy-efficient DNN accelerators have been proposed [4]. Among them, systolic array based architectures such as, Google TPU [5] and DaDianNao [6] are widely adopted, where complex multiplications and additions are performed on large number of 2D MAC arrays.

Supply voltage scaling is one of the most effective design techniques to improve energy-efficiency of DNN accelerators [7], [8] however it increases path delay and lead to timing errors. Although DNN shows the algorithmic error resiliency [9], propagation of timing errors incurs significant accuracy drop especially when the accelerator contains a large number of MAC units [10]. To handle the timing errors while maintaining throughput, TE-Drop [10] detects timing errors using widely used Razor flip-flops [11]–[14], and (if there is an error) drops the update of multiplication results subsequent to the erroneous MAC while bypassing the correct partial sum of shadow latch with delayed clock. However, the above approach did not consider the relative importance of the weights. Therefore, some of the MAC results involving the multiplication of important weights can get ignored.

In this paper, we present an error resilient technique for voltage scalable DNN accelerator based on the sensitivity variations of DNN filters and weights. Here, the sensitivity of a filter weight means the amount of accuracy drop when bit-flip errors occur in the multiplication with the particular filter weight. In other words, the errors in the computations with more sensitive weights give rise to large accuracy drop while the same errors in the computations with less sensitive weights lead to relatively less accuracy drop. For the sensitivity analysis, we propose a fine-grained error profiling method to obtain the filter/channel-level weight sensitivities with reduced simulation time especially for large scale DNNs. Based on the sensitivity analysis, computations with more sensitive weights

are assigned to more robust MAC units, while those with less sensitive weights are assigned to less robust MAC units. The relative robustness (timing error resilience) to supply voltage scaling can be controlled by using heterogeneous sizing of MAC units where more robust MAC units can be upsized while less robust MAC units are downsized. Using dynamic programming [15], the sizes of MAC units are selected to achieve best DNN accuracy under ISO-area constraints. In addition, three sensitivity-maps of 256 × 256 MAC units are drawn considering data flow while ensuring minor area overhead. The proposed DNN accelerator with heterogeneous MAC units allows more aggressive voltage scaling compared to the accelerators with the MAC units of identical size. The simulations with ImageNet dataset show that significant energy savings can be achieved with minor accuracy degradation even with very aggressive supply voltage scaling.

The rest of the paper is organized as follows. Section II presents the preliminary of DNN error resilience. The sensitivity analysis using Taylor expansion is described in Section III. Voltage scalable DNN accelerator with heterogeneous MAC units is proposed in Section IV and the experimental results including the comparisons with previous error resilient techniques are presented in Section V. Finally, conclusions are drawn in Section VI.

II. Related Works and Background

A. Related Works

1) Timing Error Propagation: By using algorithmic error resilience, timing errors are propagated through the partial sum accumulation to reduce the latency overhead for the re-execution. This scheme can only endure very low timing error probability below 0.1% [16], as the partial sums can be changed significantly for timing errors in the high-order bits.

2) Timing Error Drop (Thundervolt): To deal with the timing errors in partial sums without latency overhead, the idea of dropping the multiplication result considering the timing errors is first introduced in [14]. Thundervolt [10] drops one multiplication result whenever timing error is detected in the partial sum. When the timing error occurs in one MAC unit, the MAC next to the erroneous MAC unit sends the delayed output from shadow latch to the following MAC which indicates that the erroneous multiplication is skipped. Since each output of DNN layers is the summation of huge number of multiplication results, the error of individual multiplication results might have less effect on the classification accuracy. In addition, as DNN filter weights are distributed primarily on small values in most cases [17], the effect of each multiplication result becomes smaller. However, although the weight is small, the multiplication result can be large depending on the input activation. Moreover, the number of large weights can be larger when using 8-bit integer computations (widely used in recent DNN inference accelerator such as google TPU [5]).

B. Deep Neural Networks

1) Basic DNN Operations: As shown in Fig. 1, deep convolutional neural networks consist of many subsequent layers

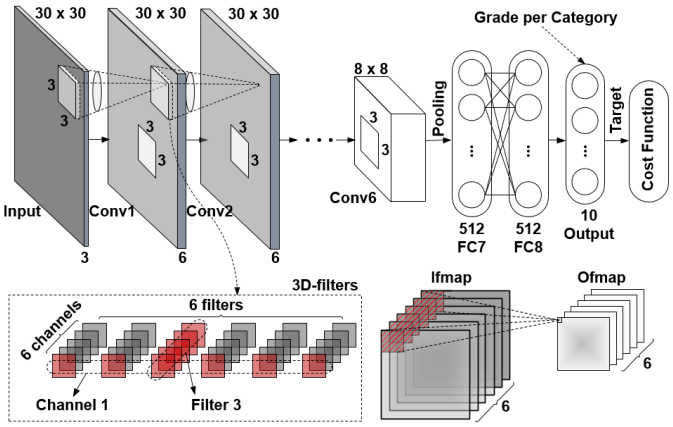

Fig. 1. An example of deep convolutional neural network architecture.

including convolutional layers, pooling layers, fully connected layers, and softmax layer (classification). Among those layers, convolutional layers take up most of the computations. For the convolution operations with input featuremap (Ifmap: input activation), filters are used to generate the output feature map (Ofmap). Here, a filter means one 3D filter and a channel means all 2D filters of the same depth. As presented in Fig. 1, for example, six $3\times3\times6$ filters (6 channels) are used in Conv1 layer. In this work, sensitivities (to error) of those filters and channels will be analyzed.

2) Cost Function: DNN maps inputs to output labels by learning to approximate an unknown relationship between inputs and outputs. Generally, let $X \in \mathbb{R}^{D\times N}$ be input data where N is the batch size, i.e., the number of training examples, and D is the dimension of each example. Let $w^l \in \mathbb{R}^{D_l\times D_{l-1}}$ be a matrix to obtain a D_l-dimensional representation of outputs at l^{th} layer, whose outputs $a^l \in \mathbb{R}^{D_l\times N}$ are called activations. Let $\phi^l : \mathbb{R} \rightarrow \mathbb{R}$ be a non-linear activation function, e.g., a rectified linear unit (ReLU), which returns zero output if the input of ReLU is negative and otherwise returns input itself. This non-linearity is applied to each layer's outputs to generate the inputs of next layers. The mathematical formulation of each layer can be represented as

$$a^i = \phi^i\left(w^i a^{i-1} + b^i\right) \quad \forall i \in [1, L] \tag{1}$$

where L is the number of layers in the network. The network's parameters $W = \{w^1, w^2, \ldots w^L\}$ are trained to minimize the cost, $C(X, W)$, which is the difference between the outputs of the network and the target values. In this work, we use a negative log-likelihood function among various version of cost function $C(\cdot)$.

3) Integer Quantization: Recently, 8-bit integer quantization scheme [18] is suggested and widely used for the inference of DNN accelerator [5]. In the scheme, instead of floating or fixed point (32-bit or 16-bit), 8-bit of integer values are used to represent the weights and activations which are multiplied with each other in MAC units. Since 8-bit integer quantization shows comparable classification accuracies with reasonable hardware cost, weights and activations are represented as 8-bit integers in this work.

III. Sensitivity Analysis Using Taylor Expansion

In this Section, we present an approach to analyze the sensitivities of DNN filter weights using Taylor expansion [19]. For developing error resilient design techniques, the first task is to measure the importance of each computational node in DNN. In the traditional fault injection method, a combination of large number of error patterns and bit error rates are injected to worsen the time and cost. The proposed approach can simply obtain the error resiliencies of each node in a neural network with a given BER.

With Taylor expansion, the error resilience of each variable can be expressed as

$$|\Delta C(w_i)| = |C\,(D, w_i) - C\,(D, w_i + \varepsilon)| \tag{2}$$

where $C\,(D, w_i + \varepsilon)$ is the cost if variable w_i is distorted to a certain noise ε, and $C\,(D, w_i)$ is the original cost. Here, we refer to $|\Delta C(w_i)|$ as sensitivity. In other words, if the sensitivity is large, variable w_i is sensitive to noise.

To approximate the sensitivity, we use the first-order Taylor expansion of the cost function. In a function $f\,(x)$, the Taylor expansion around point x = a is expressed as

$$f\,(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}\,(x-a)^n \tag{3}$$

where $f^{(n)}(a)$ is the n-th derivative of $f(x)$ at point a. By adjusting with first-order Taylor expansion, $C\,(D, w_i + \varepsilon)$ can be expressed as

$$\begin{aligned} C\,(D, w_i + \varepsilon) &= C\,(D, w_i) + \frac{\partial C}{\partial w_i}\,((w_i + \varepsilon) - w) + Rem \\ &= C\,(D, w_i) + \frac{\partial C}{\partial w_i}\varepsilon + Rem. \end{aligned} \tag{4}$$

Here, Rem is the remaining term that can be neglected since it is assumed to be small.

Finally, by substituting (3) into (1), we get

$$\begin{aligned} |\Delta C(w_i)| &= |C\,(D, w_i) - C\,(D, w_i + \varepsilon)| \\ &= \left|C\,(D, w_i) - C\,(D, w_i) + \frac{\partial C}{\partial w_i}\varepsilon\right| = \left|\frac{\partial C}{\partial w_i}\varepsilon\right|. \end{aligned} \tag{5}$$

From (5), the sensitivity depends on the values of the gradient and noise itself. As the noise ε can vary depending on the magnitude of the variable and the bit cell which is flipped over, we can express the noise ε as a function of original bit error rate (BER). Suppose that we have a 3-bit integer '010', and BER is p. The target value can be replaced by a probability for a given p for 2^3 cases. For example, if the integer is flipped and replaced by '100', the probability of that event is $p^2(1-p)^1$. The power of p terms can be calculated by the hamming distance of the two binary representations. Thus, the expectation in such a case where the injected noise affects the bit cell at the first and second positions, becomes $p^2(1-p)^1 \cdot val('100')$, where $val\,(\cdot)$ is a function that returns the real value with a given integer. Although the approach is used for integer quantization, it is also applicable to fixed-point or floating-point representations.

Algorithm 1 shows the steps to get the expectation of noise ε and the sensitivity, where $hamming(\cdot)$ is the hamming distance between two binary representations, $bin(\cdot)$ is a function that returns the actual value in its binary representation, and $int(\cdot)$ is a function that returns the integer representation of given values. For gradient measurement step, the gradient is statistically obtained by differentiating and then averaging over the target variable for entire dataset. For noise measurement step, the expectation is obtained by multiplying the flipped probability p_term, non-flipped probability q_term, and the flipped value for all cases. Finally, the sensitivity of a weight is obtained by multiplying the gradient and the expectation. The sensitivity can be used in fine-grained or coarse-grained manners. For coarse-grained, the sensitivity can be expressed as

$$Sen_w = \frac{1}{M}\sum\nolimits_m \left|\frac{\partial C}{\partial w_m}\varepsilon\right|, \tag{6}$$

where M is the length of the vectorised variables w.

We ran the simulations on Conv1 layer filters of LeNet-5 using the proposed Taylor expansion based sensitivity test approach. We also use the traditional fault injection method to obtain the sensitivities. As presented in Fig. 2(a), the sensitivities obtained with the proposed Taylor expansion (plot) and the actual error resilience (histogram) matches well. We can notice from the results that #2 filter, that is composed of 5 × 5 weights, has the largest sensitivity, which means that small errors in #2 filter weights give rise to large cost function change. Here, the sensitivity of the filter is the average sensitivity of 5 × 5 weights. Fig. 2(b) shows the simulated sensitivities of Conv2 layer channels of LeNet-5. Fig. 3 shows the simulated sensitivities of individual weights of Conv2 layer filters in LeNet-5. Interestingly, it is observed that the weights with large amplitudes do not have large sensitivities, and the weights with medium amplitudes show relatively larger sensitivities.

Algorithm 1 Sensitivity Analysis

Input : *Cost*: Cost function of network, D: Training datasets, w_l: The pre-trained integer-type variable, p: Flip probability of each bit-cell, *bit*: Bit-width of integer quantization scheme.
Output : Sen_{w_l}: the sensitivity of variable w_l
Gradient Measurement
$gradient_{w_l} = 0$
for *data* in D **do**
  $gradient_{w_l} = gradient_{w_l} + \frac{\partial}{\partial w_l} Cost(data, w_l)$
$gradient_{w_l} = gradient_{w_l} / range(D)$
Noise Measurement
$expectation_{w_l} = 0$
for $i = 0, 1, ...2^{\text{bit}-1}$ **do**
  $p\_term = p^{hamming(bin(w_l), bin(i))}$
  $q\_term = (1-p)^{(bit - hamming(bin(w_l), bin(i)))}$
  $expectation_{w_l} = expectation_{w_l} + p\_term \cdot q\_term \cdot val(int\,(i))$
$\varepsilon_{w_l} = w_l - expectation_{w_l}$
Sensitivity Calculation
$Sen_{w_l} = \left|gradient_l \cdot \varepsilon_l\right|$

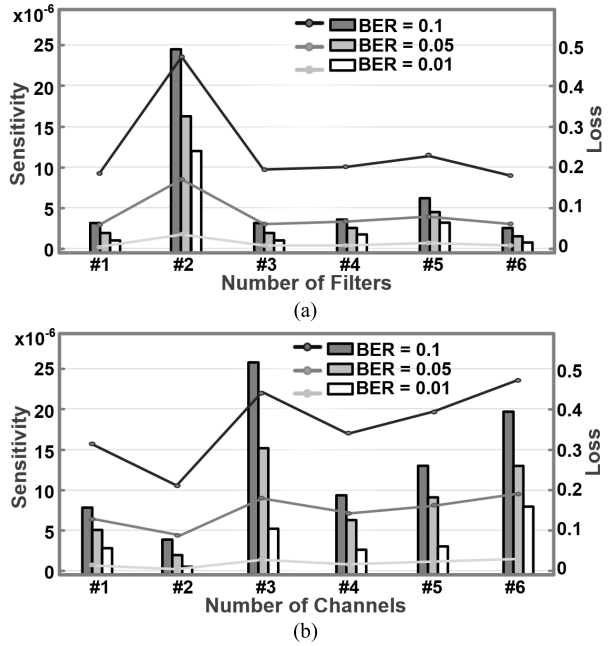

Fig. 2. (a) Sensitivity simulation results with varying BERs to each filter. (b) Sensitivity simulation results with varying BERs to each channel.

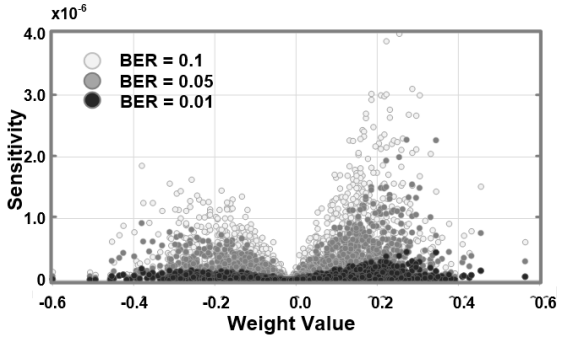

Fig. 3. Sensitivity to each weight value is measured according to the given each bit error rate (BER).

IV. Voltage Scalable DNN Accelerator With Heterogeneous MAC Units

This section presents the DNN accelerator design with heterogeneous sizing of MAC units to enable aggressive supply voltage scaling by exploiting the sensitivities of filters and weights. To detect timing errors induced by voltage scaling, the modification of MAC unit design is first presented. Then, we present the timing error probability model of MAC units with supply voltage scaling. The sensitivity map of MAC units which is the key factor to decide heterogeneous sizes of MAC units is also presented in this section. Finally, the sizes of MAC units are selected using dynamic programming to achieve best DNN accuracy under ISO-area constraints.

A. Modification to MAC Unit Design

Fig. 4(a) shows the typical systolic array based DNN accelerator architecture [5], [10]. The 2D MAC array contains 256×256 MAC units where each MAC unit computes 24-bit partial sum with 8-bit of weights and activations. For the computations, weights are preloaded into the MAC array from weight FIFO and remain stationary until all the related input activations are computed. In the systolic operation, the scheduled input activations are streamed into MAC array from the activation buffers, and the MAC outputs (partial sums) are accumulated sequentially through top-down direction in 2D array. In the conventional systolic array based DNN accelerator design [5], even though an error occurs in the MAC unit that processes less sensitive weights, the accuracy drop can be significant since the error occurs in partial sum. For error resilience, the multiplication with less sensitive weight should be dropped before accumulating with partial sum when error occurs. The modified MAC unit is shown in Fig. 4(b). In the MAC unit, additional 16 bits of flip-flops with 10 bits of Razor flip-flops [11] are inserted right after multiplier as an additional pipeline stage to detect timing errors that occur during the multiplication. As many noncritical flip-flops do not need Razor flip-flops [11], only 10 bits are used to protect 16 bits in the critical multiplication data-path. The power overhead of the additional flip-flops is around 6.86% compared to conventional design [5] which is paid for timing error detection. Since the path delays of the non-Razor paths (adder and MUX) are around half of the MAC unit critical path delays, the probability of the timing errors in the non-Razor paths is extremely low even under aggressive supply voltage scaling. When timing error occurs in the multiplication, the result can be instantaneously discarded and the non-updated partial sum goes through the next (bottom) MAC unit. The latency increases by one clock cycle with the modified MAC but the throughput of the systolic MAC array is maintained.

In order to exploit the sensitivity variations among filter/channel-level weights, we assign the computations with more sensitive weights to more robust MAC units, while those with less sensitive weights are assigned to less robust MAC units. The approach can effectively reduce the number of timing errors that can occur in the multiplications with more sensitive weights. The relative robustness (timing error resilience) to voltage scaling among MAC units can be controlled by using heterogeneous sizing of MAC units where more robust MAC units are upsized while less robust MAC units are downsized. In the next subsection, we present the path delay model of MAC units through which we can easily model the timing error rates of MAC units with supply voltage scaling.

B. Timing Error Probability Model of MAC Units

When critical path delay d_{path} is modeled as a Gaussian random variable over the input vector space, it can be modeled as an inverter chain of length T [21] if d_{path} has the worst-case delay of T FO4 (fan-out of 4 inverter). Delay d_{path} can be also obtained by the simulation of the target module using Synopsys Prime time [23]. For the single FO4 delay, the mean μ with the standard deviation σ can be obtained using 2K Monte Carlo simulations with HSPICE. If FO4 delay of a single inverter is d_{inv}, we can calculate the length T as d_{path} divided by d_{inv}. If the clock period is T_{clk}, an input dependent timing error probability of one critical path can be represented

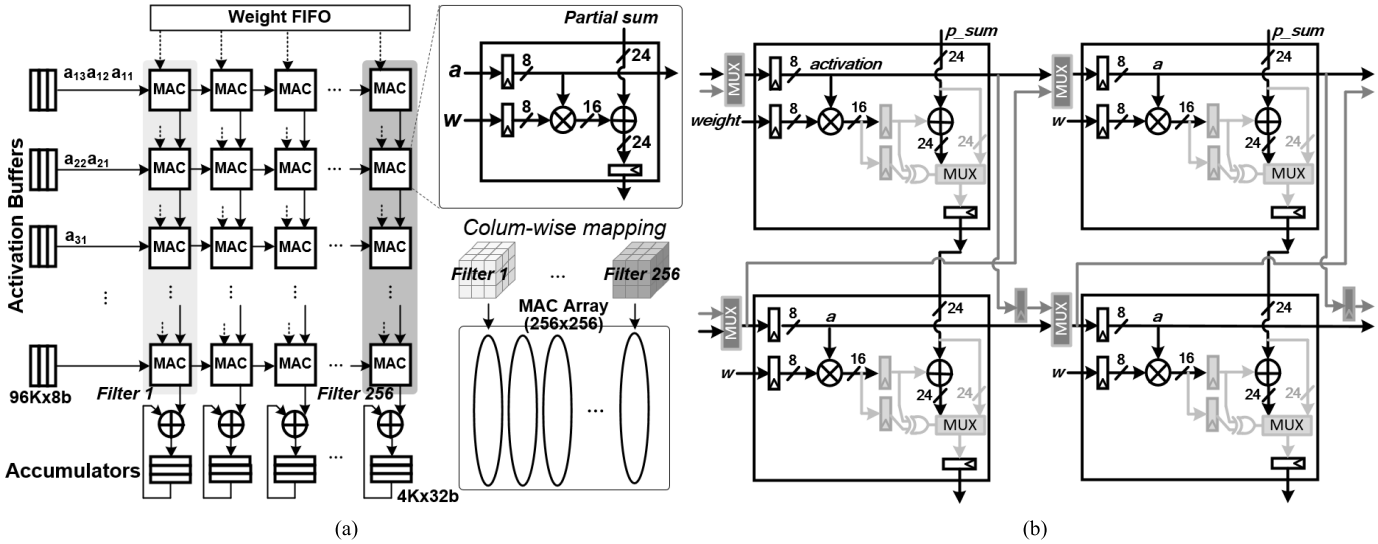

Fig. 4. (a) Baseline hardware architecture, (b) proposed error resilient techniques.

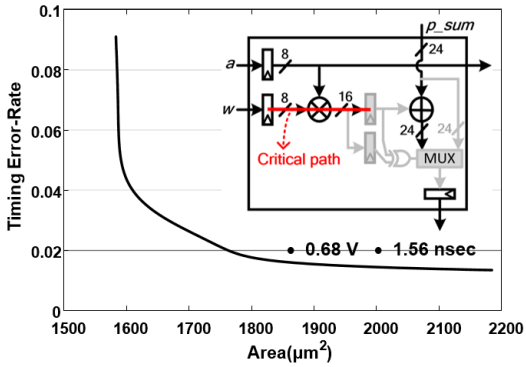

Fig. 5. Input dependent timing error probabilities with varying areas of MAC unit designed with 65nm CMOS.

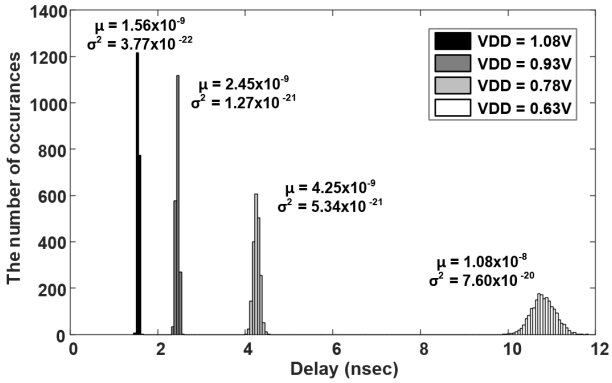

Fig. 6. Inverter delay distributions (48 stages) due to process variations for different VDDs scaled down from 1.08V to 0.63V.

as follows [21]:

$$P_I = \sum_{N=T_{clk}/d_{inv}}^{T} p^N(1-p), \tag{7}$$

where p is the probability that the input signal of the critical path propagates by one inverter when the critical path is composed of T inverter chains. We use $p = 0.5$ [21] in this paper. If we assume that input dependent timing error probabilities of all critical paths have the same distribution as (7), an input dependent timing error probability considering N (=16 in this work) critical paths can be expressed [21] as following:

$$P_{I,N} = 1-(1-P_I)^N. \tag{8}$$

Here, $P_{I,N}$ is the probability of one of N critical paths exceeding a given clock period T_{clk} and can be modeled as the cumulative distribution function (CDF) of a normal distribution [21]. Fig. 5 presents the input-dependent timing error probabilities (8) with varying MAC unit areas in 65nm CMOS process under 0.68V supply voltage and 1.56nsec clock period.

In addition to the input dependent timing error probability model, the process variations of critical paths should be also considered. Since the critical path delay is modeled as the inverter chain of length T, and each inverter delay follows the Gaussian distribution with mean μ and standard deviation σ, the delay of the entire critical path becomes [21]

$$D_{path} = \sum_{k=1}^{T} D_{inv,k} \sim N(T\mu, T\sigma^2) \tag{9}$$

where D_{path} and $D_{inv,k}$ are the delays of the entire critical path and single inverter, respectively. To calculate the final timing error probability of $P_{I,N}$ in (8), the length T inverter chain can be obtained from the randomly sampled path delay D_{path} of each MAC unit using delay distribution in (9). In Fig. 6, a 48 stage inverter delay distributions $N(\mu, \sigma^2)$ with supply voltage scaling is presented which shows the larger variations and increasing critical path delays of MAC units from $N(T\mu, T\sigma^2)$ as supply voltage is scaled down. Fig. 7 also shows the timing error probabilities considering both input vectors and process variations with varying MAC unit areas under 0.68V supply voltage and 1.56 nsec clock period.

As mentioned earlier, the relative robustness (timing error resilience) to voltage scaling among MAC units is con-

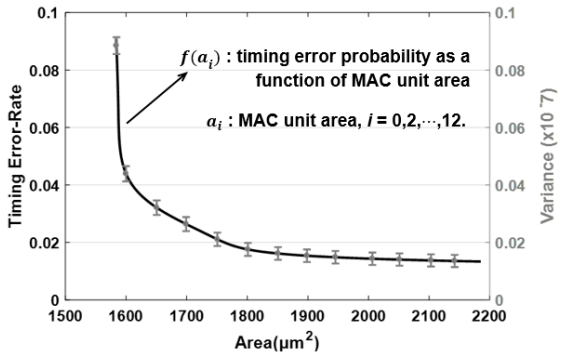

Fig. 7. Timing error probabilities of MAC units designed with 65nm CMOS considering process variations, varying input vectors and varying areas of MAC unit.

trolled by different sizing of MAC units with the timing error probability modeling. A set of possible MAC unit sizes for heterogeneous MAC sizing is represented as $A = \{a_0, a_1, a_2, \ldots, a_{n-1}\}$, where a_0 is the minimum area and a_{n-1} is the maximum area of MAC units. Sizes a_0 and a_{n-1} are decided by the maximum and minimum timing error probability, respectively, which are determined by the simulations shown in Fig. 7. In addition, a set of timing error probabilities of the corresponding A is represented as $P = \{f(a_0), f(a_1), f(a_2), \ldots, f(a_{n-1})\}$, where $f(a_k)$ is the timing error probability of the MAC unit of area a_k, obtained from Fig. 7. Here, the grid (minimum increment step) between different MAC unit areas is selected as d which satisfies $f(a_{n-1} - d) - f(a_{n-1}) \geq P_{th}$, where $P_{th} = 10^{-6}$. For example, as shown in Fig. 7, A is chosen as {1585.04, 1635.53, 1685.00, 1735.67, 1785.42, 1835.23, 1883.02, 1935.10, 1980.18, 2035.52, 2084.93, 2142.16, 2185.15} with $d = 50$ and P is $\{1.33\times10^{-2}, 1.35\times10^{-2}, 1.38\times10^{-2}, 1.41\times10^{-2}, 1.45\times10^{-2}, 1.49\times10^{-2}, 1.56\times10^{-2}, 1.65\times10^{-2}, 1.83\times10^{-2}, 2.26\times10^{-2}, 2.79\times10^{-2}, 3.44\times10^{-2}, 9.09\times10^{-2}\}$ under 0.68 V of supply voltage with 1.56 nsec clock period. The number of elements in the set A (set P) is 13.

Based on the timing error probability models of MAC units, we select the sizes of 256 × 256 MAC units among the possible MAC unit sizes in $A = \{a_0, a_1, a_2, \ldots, a_{n-1}\}$. Here, the timing error probabilities of the MAC units which are $P = \{f(a_0), f(a_1), f(a_2), \ldots, f(a_{n-1})\}$ and those that impact the accuracy of DNN should be carefully considered. In the following subsection, the 2D MAC array sensitivity map which represents the sensitivity (importance) of each of the 256×256 MAC unit considering the data flow reflected weight sensitivities (as the multiplications with weights are mapped to MAC units) and timing error probabilities is presented.

C. Sensitivity Map for 2D MAC Array

As depicted in Fig. 8, the sensitivities of all the weights that are processed in a same MAC unit should be considered to determine the size (area) of MAC units since 256×256 MAC units process different 256×256 filter weights many times to obtain the accuracy of DNNs. Here, *the sensitivity of a MAC unit* is defined as "the summation of the sensitivities of all the

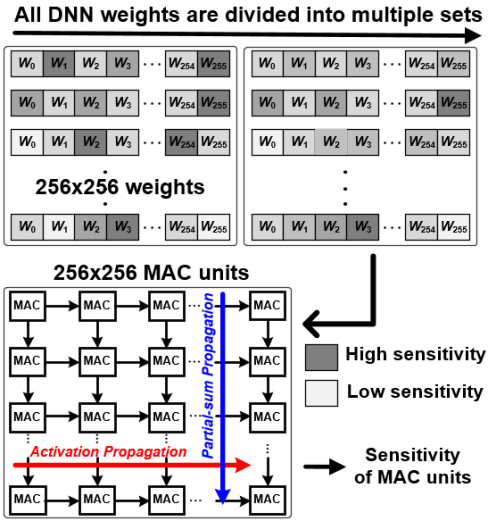

Fig. 8. Sensitivity based DNN weights mapping to MAC units.

weights that are processed in the MAC unit". The sensitivity of each of MAC unit is scaled to a constant to reflect the relative importance of different layers. The mapping of those weights to each of MAC unit is decided by the data flow and the architecture of DNN accelerator.

For conventional data flow which does not consider the weight sensitivities, the sensitivity variations among different MAC units can be small due to averaging effect as shown in Fig. 9. When 36 sets (X axis in Fig. 9(a)) of 256×256 weights in CONV18 layer of ResNet-18 are mapped to MAC array, the sensitivity variance (σ) of 256×256 MAC units is as small as 2.263×10^{-9}. For the extreme case, if the sensitivities of all MAC units are same (zero sensitivity variance), heterogeneous sizing of MAC units does not show any improvement over ISO-area MAC units. In the proposed heterogeneous MAC design, we want to increase the sensitivity gap (variance) between different MAC units so that the MAC units with high sensitivities can be designed with larger size (robust) while the MAC units with low sensitivities can be designed with smaller size. The sensitivity variance can become large by using the following three mapping strategies where column-wise and intra-column sorting is considered.

1) Column-Wise: As shown in Fig. 4(a), different DNN filters are mapped to different columns of MAC array (column-wise mapping). To increase the sensitivity difference of the weights, the columns are sorted using the average values of the weight sensitivities in each column. For example, in the column-wise mapping, the column with the weights having the highest sensitivities (for the sensitivity sorting, the average values of weight sensitivities in each column are used) is mapped to the rightmost column of 2D MAC array while the column with the weights having the lowest sensitivities is mapping to the leftmost column. As shown in Fig. 9(b), the sensitivity variance of 256 × 256 MAC units increases to 2.796×10^{-9} when column-wise sorting is used. Since sorting the weights between different columns of MAC array can be performed off-line, column-wise sorting does not incur any additional cost and hardware overhead.

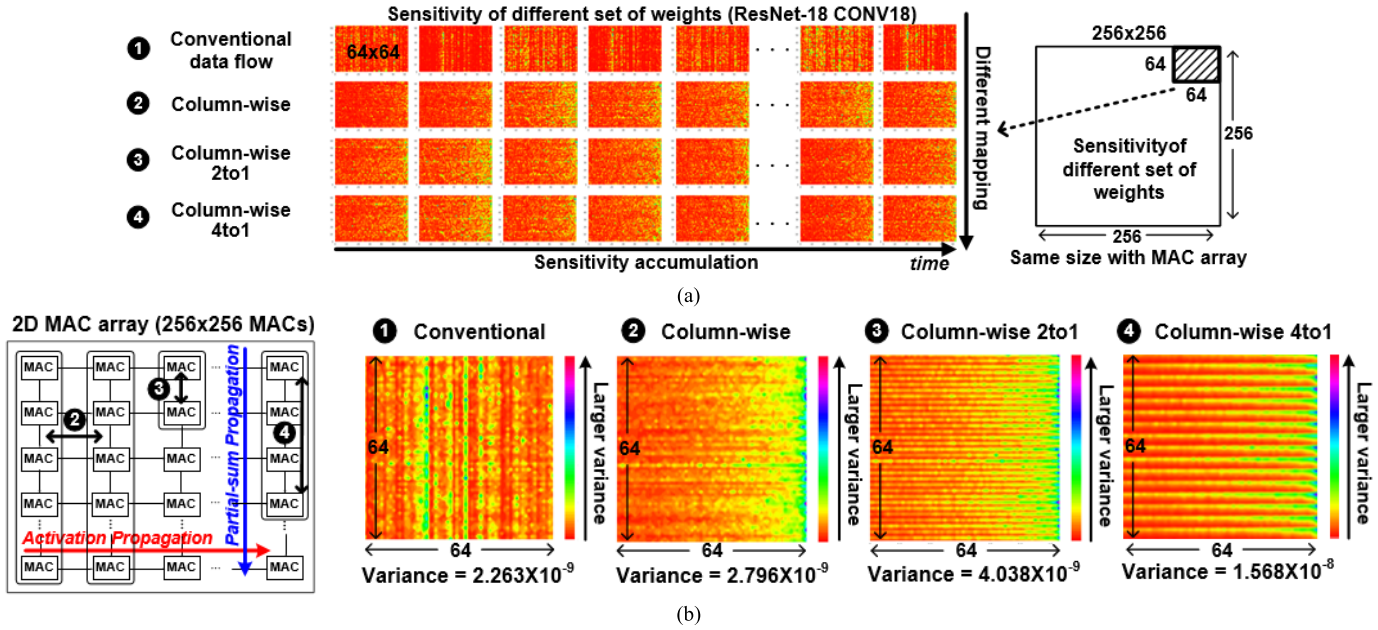

Fig. 9. The sensitivity variations depending on different mapping strategies: (a) an example of sensitivity accumulation of 64 × 64 MAC units according to time, (b) the accumulation results and sensitivity variance of conventional scheme and the proposed three mapping strategies.

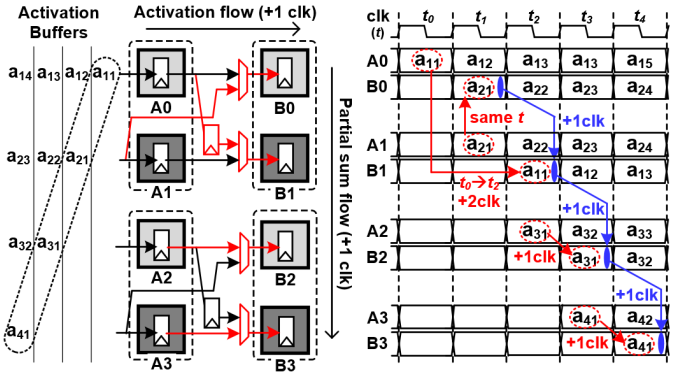

Fig. 10. An example of the activation propagation in column-wise 2to1 sorting.

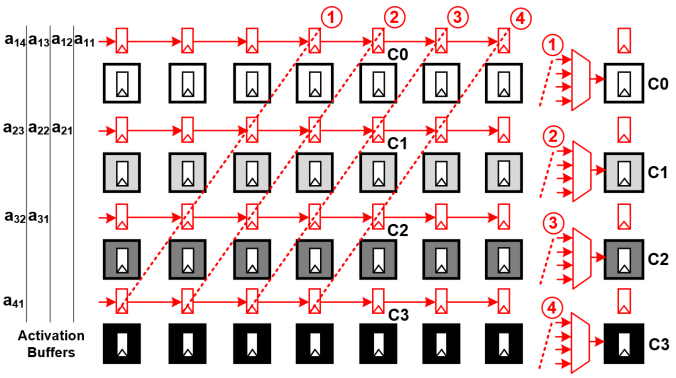

Fig. 11. The data flow and hardware overhead of the column-wise 4to1 sorting.

2) Column-Wise 2 to 1: The column-wise 2to1 sorting sorts the weights assigned to two consecutive MAC units in a same column of MAC array. As presented in the data flow in Fig. 4, since all the MAC units in the same row share same input activations and sequentially accumulate partial sums from top MAC to bottom MAC, separately sorting each filters (within each column) messes up the data flow in systolic array. To sort the weights of same column while preserving the systolic data flow with small area overhead, the input activations between two rows of MAC units are switched as well. As shown in Fig. 4(b) (dark grey color modules), two 2to1 MUXs and one 8bit flip-flop are added to the pair of MAC units with additional interconnect routing for the activation paths. Fig. 10 shows an example of the switched activation propagation. In Fig. 10, A0 ∼ A3 and B0 ∼ B3 are heterogeneous MAC units and A0/A1, A2/A3, B0/B1, B2/B3 (grey/dark grey) are the pairs which can switch the activations using the modified MAC units shown in Fig. 4(b). Within one pair of MAC units, dark grey MAC units have larger sizes (more resilience to timing errors) than grey MAC units using column-wise 2to1 sorting. In Fig. 10, when the activations are switched between two successive pairs of MAC units (A0/A1 and B0/B1), a_{21} is used in both B0 and A1 at the same clock cycle while a_{11} for B1 is delayed by 2 clock cycles. As shown in the timing diagram of Fig. 10, the partial sum accumulation (blue arrow) is not affected by the activation switching. As presented in Fig. 9(b), the sensitivity variance of 256 × 256 MAC units is 4.038×10^{-9} with column-wise 2 to 1 mapping which is about 2 times larger than that of the conventional mapping.

3) Column-Wise 4 to 1: Column-wise 2to1 scheme can be extended to sort more rows of MAC units with larger area overhead. To allow the sorting of four rows of MAC units, 2to1 MUX is replaced with 4to1 MUX, and additional activation FIFO is needed in each row of MAC units (one 8bit flip-flop per MAC unit) to keep the original data flow. Fig. 11 shows the connection between different rows of MAC

units and the flip-flops in the activation FIFO to provide right activations to each MAC unit. In Fig. 11, ① ∼ ④ flip-flops are connected to 4to1 MUXs for C0 ∼ C3 MAC units, respectively. Depending on different selection signals for the MUXs, the activations are selected from one of the four rows while the partial sum accumulation is maintained. Fig. 9(b) shows the sensitivity variance of the column-wise 4to1 sorting significantly improves to 1.568×10^{-8} (7.5 times larger than that of conventional mapping) with an area overhead of one 4to1 MUX and one 8bit flip-flop per MAC unit.

D. Heterogeneous MAC Unit Sizing Approach

The sizes of 256 × 256 MAC units in 2D array of DNN accelerator are selected using the sensitivity maps of 2D MAC array, possible MAC unit sizes in $A = \{a_0, a_1, a_2, \ldots, a_{n-1}\}$ and their corresponding timing error probabilities $P = \{f(a_0), f(a_1), f(a_2), \ldots, f(a_{n-1})\}$. The MAC array should be carefully designed such that the sizes of more sensitive MAC units in 2D sensitivity map are larger, while those of less sensitive MAC units are selected smaller.

With given area constraint of whole MAC array, the problem can be formulated as follows:

$$Minimize \sum_{i=o}^{MN-1} s_i \times f(a_i)$$
$$s.t. \sum_{i=o}^{MN-1} a_i \leq A_{max}, (M, N = 256) \qquad (10)$$

where s_i is the sensitivity of i_{th} MAC unit, a_i is the area of i_{th} MAC unit, $f(a_i)$ is the timing error probability as a function of MAC unit area obtained from Fig. 7, and A_{max} is the given area constraint. In this work, the area constraint is set to ISO-area with the conventional MAC array for the comparison with conventional MAC array with homogeneous MAC units. The objective function $\sum_{i=o}^{MN-1} s_i \times f(a_i)$ (8) is referred to *the expected accuracy degradation* in the rest of the paper.

E. Dynamic Programming Based MAC Array Design

Algorithm 2 shows the overall flow of the proposed heterogeneous MAC unit sizing process. The algorithm minimizes the expected accuracy degradation Exp, which is $\sum_{i=o}^{MN-1} s_i \times f(a_i)$ in (8), under a given area constraint A_{max}. $Exp_{(i,m)}$ represents the minimum sum of the expected accuracy degradation of $i+1$ MAC units. In the initialization step, $Exp_{(0,m)}$ can be easily found by a simple table look-up with one candidate, $Exp_{MN-1}[m]$. Here, m is the intermediate area constant to solve the sub-problem of current iteration. For other table contents, $Exp_{(i>0,m)}$ and $MAC_{(i>0,m)}$ are initialized as an infinite number and zero, respectively. As i becomes larger, $Exp_{(i,m)}$ can be obtained by accumulating the Exp value each MAC size, where $Exp_{(i,m)}$ has the partial optimal MAC sizes of $MAC_{(i,m)} = \{MAC_{MN-1}, \cdots, MAC_{MN-1-i}\}$ after iterations. To compute $Exp_{(i,m)}$ while using the previously computed $MAC_{(i-1,m-a)}$, Exp of current iteration step $Exp_{MN-1-i}[u]$ is calculated, where u is the newly added area of the $(MN-1-i)_{th}$ MAC unit. $Exp_{MN-1-i}[u]$ is also computed by the sensitivity s_{MN-1-i} corresponding to $(MN-1-i)_{th}$ MAC unit multiplied by the timing error probability at given supply voltage $f_{V_{dd}}(u)$ and MAC sizes. In other words, the partial MAC sizing $MAC_{(i,m)}$ that has the minimum sum of Exp is determined by varying u in the i_{th} iteration to compute $MAC_{(i,m)}$, where the $MAC_{(i-1,m-u)}$ is obtained from the previous step. The minimum sum of Exp is iteratively searched by solving the sub-problems.

Algorithm 2 MAC Unit Size Selection Algorithm Based on Dynamic Programming

1:**Input** : Area Constraint (A_{max}), Voltage (V_{dd}), Row Numbers of MAC array (M), Column Numbers of MAC array (N), Expected Accuracy Degradation (Exp), Sensitivity (S = $\{s_0, \cdots, s_{MN-1}\}$), Possible MAC unit sizes (U = $\{u_0, \cdots, u_{n-1}\}$), Timing Error Probability Function ($f_{V_{dd}}$)
2: **Output**: Optimal MAC sizing MAC_{opt}
3: **Initial:**
4: Initialize $Exp_{(0,m)} \leftarrow Exp_{MN-1}[m]$
5: Initialize other sizing sub-problems $Exp_{(i,m)} \leftarrow \infty$
5: **for** $i \leftarrow 1 to MN-1$ **do**
6: **for** m $\leftarrow (i+1) \cdot u_0 to A_{max}$ **do**
7: **for** $u \leftarrow u_0 tom$ **do**
8: $Exp_{MN-1-i}[u] \leftarrow s_{MN-1-i} \cdot f_{V_{dd}}(u)$
9: $Exp_{temp} \leftarrow Exp_{(i-1,m-u)} + Exp_{MN-1-i}[u]$
10: **If** $Exp_{(i,m)} > Exp_{temp}$ **then**
11: $Exp_{(i,m)} \leftarrow Exp_{temp}$
12: $MAC_{(i,m)} \leftarrow MAC_{(i-1,m-u)} \cup \{u\}$
13: $MAC_{opt} \leftarrow MAC_{(i,m)}$

V. Experimental Results

A. Simulation Results

In this section, we present the numerical comparisons on the accuracy of DNN when supply voltage is scaled down in DNN accelerators with identical MAC units and heterogeneous MAC units. With identical MAC units, all the weights with different sensitivities are processed under same timing error probabilities. However, with heterogeneous MAC units, the MAC unit areas are determined by Algorithm 2 considering sensitivity map, where more sensitive weights can be processed with relatively lower timing error probabilities while less sensitive weights are processed with larger error probabilities. Fig. 12(a) shows the sensitivities (log scale) of 256 × 256 MAC units and their corresponding areas obtained from Algorithm 2 for three mapping strategies, under ISO-area condition of $1885um^2$ (per one MAC unit) at the supply voltage of 0.68V. As presented in Fig. 12(a), the sensitivities of 256 × 256 MAC units vary depending on the mapping strategy, and the areas of 256 × 256 MAC units are also changed following the sensitivities. As it goes from column-wise mapping to column-wise 4to1 mapping, the sensitivity variance increases and the distribution of MAC unit areas become diversified as well. Fig. 12(b) shows the distributions of 256 × 256 MAC unit areas for three mapping strategies. We notice from Fig. 12(b) that most diverse sizes of MAC units are found in the column-wise 4to1 mapping where the largest error resilience improvement has been observed. Fig. 12(c)

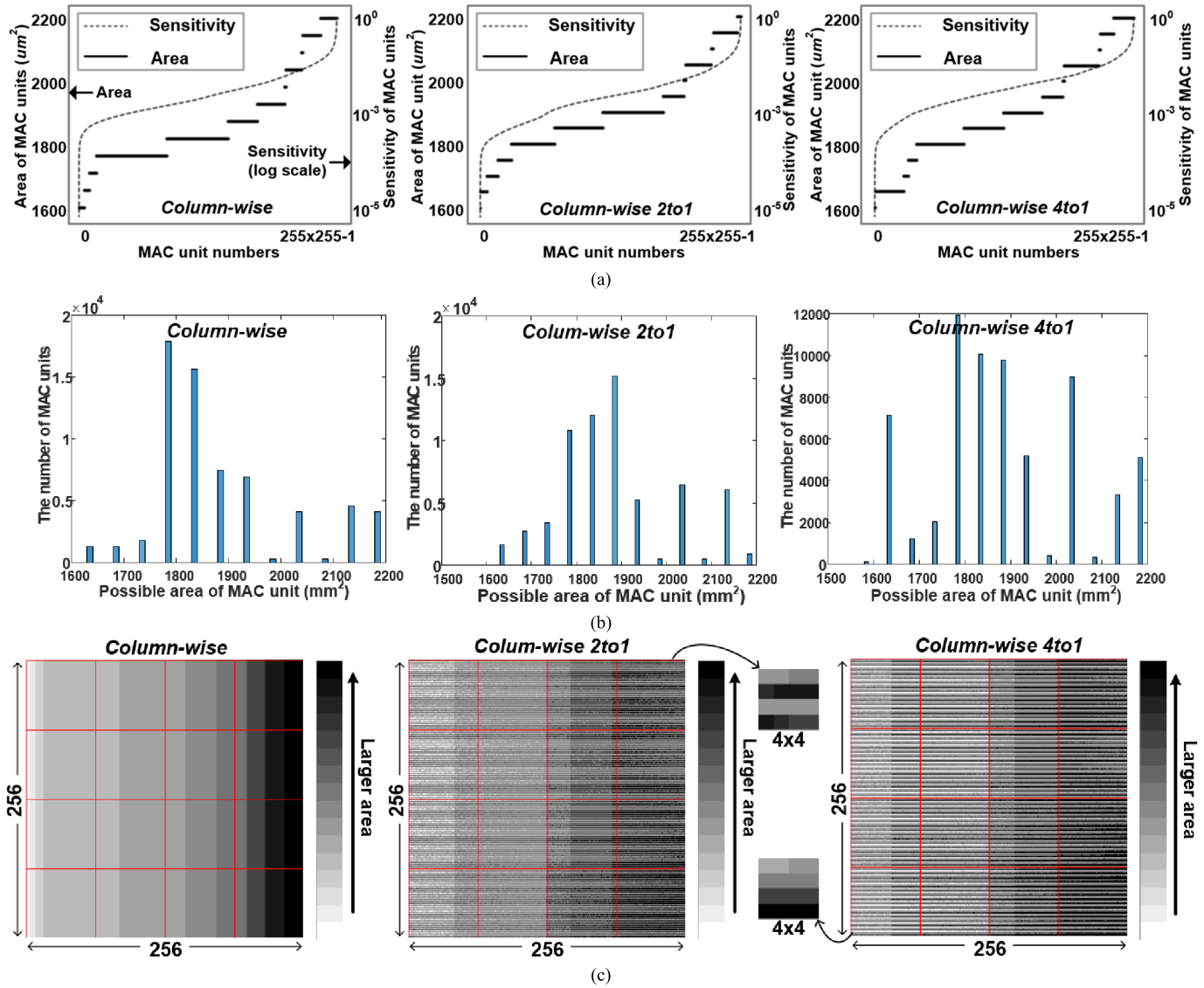

Fig. 12. (a) Sensitivities (log scale) of 256 × 256 MAC units and their corresponding areas obtained from Algorithm 2 for three mapping strategies. (b) Distribution of 256 × 256 MAC unit areas for three mapping strategies. (c) Grey scale representation of 256 × 256 MAC unit areas over 2D array for three mapping strategies.

shows the grey scale representation of 256 × 256 MAC unit areas over 2D array to check the effect of column/row sorting. In the figure, darkest dots show the MAC units with largest area while white dots mean smallest ones. Fig. 12(c) also shows that the heterogeneous sizing MAC units are allocated differently in 2D array depending on the mapping strategies, which is consistent with the 2D sensitivity map shown in Fig. 9(b). As shown in 4 × 4 pixels of the middle and right figures in Fig. 12(c), MAC areas are sorted by 2 rows and 4 rows in the column-wise 2to1 and column-wise 4to1 schemes, resepectively.

In order to evaluate the accuracy of DNN using various timing error probabilities of MAC units, three convolutional neural network benchmarks including ResNet-18, ResNet-34 and ResNet-50 are used with ImageNet large scale visual recognition challenge 2012 (ILSVRC2012) validation data set (50,000 images) [20]. We use pre-trained networks for all benchmarks. All of the experiments are built on TensorFlow [20] for adjusting the sensitivity analysis and timing error injection on the above benchmarks, which is based on the timing error probabilities of 256 × 256 MAC units. To reflect the timing errors of MAC units in TensorFlow simulation, we first find the error rate of each weight considering the mapping between weights and MAC units. According to the error rate of each weight, binary values representing errors (0: error, 1: no error) are sampled from Bernoulli distribution. To get the accuracy drop due to timing errors, weights multiplied with those binary values are used as new weights for the convolutions in DNN inference. In order to prevent only one weight error from affecting many following computations (weight×activation), every stride of the convolutions are processed in parallel (different *conv2d* function in TensorFlow) with different binary values. For more accurate simulation, the average accuracy drop of 100 inferences is used. The proposed three mapping strategies are compared with other error resilient approach [10]. Fig. 13(a) shows

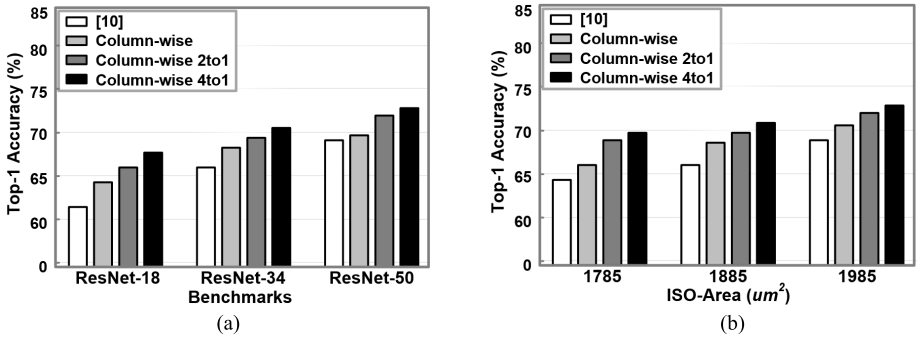

Fig. 13. Accuracy comparisons between the proposed mapping strategies and other error resilient approach [10] at 0.68V for (a) different benchmarks with 1885um^2 (one MAC unit area) ISO-area condition, (b) with varying ISO-area conditions for ResNet-34.

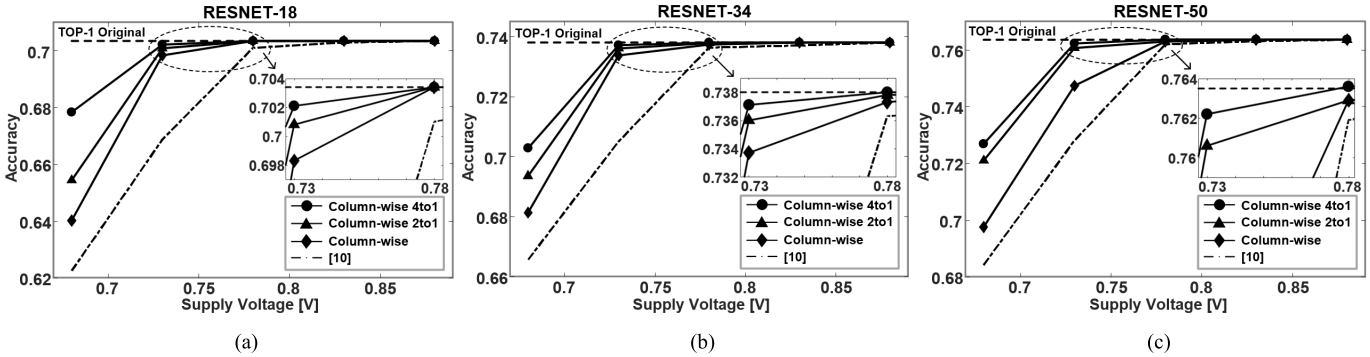

Fig. 14. Accuracy comparisons among various error resilient architectures with aggressive voltage scaling (from 1.08V to 0.68V) on different benchmarks: (a) ResNet-18, (b) ResNet-34, (c) ResNet-50.

TABLE I
THE ACCURAY COMPARISONS AMONG VARIOUS ERROR RESILIENT TECHNIQUES

VDD	ISO Area (One MAC unit size)	ResNet-18 (Original accuracy : 0.7033)						ResNet-34 (Original accuracy : 0.7380)						ResNet-50 (Original accuracy : 0.7635)					
		1785		1885		1985		1785		1885		1985		1785		1885		1985	
	Scheme	Top1	Top5	Top1	Top5	Top1	Top5	Top1	Top5	Top1	Top5	Top1	Top5	Top1	Top5	Top1	Top5	Top1	Top5
0.68	[10]	0.6118	0.8322	0.6224	0.8401	0.6334	0.8484	0.6498	0.8582	0.6656	0.8723	0.6857	0.8834	0.6748	0.8762	0.6841	0.8829	0.6934	0.8894
	Column-wise	0.6310	0.8516	0.6401	0.8531	0.6501	0.8617	0.6660	0.8691	0.6813	0.8812	0.7027	0.8927	0.6856	0.8857	0.6975	0.8920	0.7264	0.9108
	Column-wise 2to1	0.6499	0.8611	0.6543	0.8630	0.6649	0.8689	0.6851	0.8832	0.6935	0.8892	0.7168	0.9039	0.7165	0.9040	0.7209	0.9074	0.7253	0.9107
	Column-wise 4to1	0.6719	0.8755	0.6783	0.8780	0.6885	0.8847	0.6998	0.8926	0.7028	0.8950	0.7194	0.9083	0.7259	0.9103	0.7268	0.9110	0.7276	0.9110
0.73	[10]	0.6591	0.8628	0.6687	0.8768	0.6892	0.8780	0.6930	0.8856	0.7050	0.8927	0.7210	0.8983	0.7213	0.9032	0.7280	0.9066	0.7292	0.9199
	Column-wise	0.6974	0.8914	0.6983	0.8917	0.6997	0.8927	0.7338	0.9117	0.7337	0.9115	0.7344	0.9120	0.7571	0.9276	0.7473	0.9274	0.7578	0.9291
	Column-wise 2to1	0.7004	0.8932	0.7008	0.8933	0.7015	0.8938	0.7360	0.9130	0.7360	0.9129	0.7363	0.9132	0.7604	0.9295	0.7606	0.9294	0.7610	0.9303
	Column-wise 4to1	0.7019	0.8941	0.7021	0.8941	0.7024	0.8944	0.7371	0.9137	0.7371	0.9136	0.7373	0.9138	0.7621	0.9305	0.7622	0.9304	0.7626	0.9309
0.78	[10]	0.7006	0.8934	0.7010	0.8936	0.7010	0.8936	0.7361	0.9130	0.7363	0.9130	0.7363	0.9132	0.7617	0.9301	0.7619	0.9303	0.7622	0.9304
	Column-wise	0.7033	0.8944	0.7034	0.8963	0.7037	0.8946	0.7378	0.9140	0.7373	0.9140	0.7373	0.9142	0.7627	0.9312	0.7629	0.9312	0.7632	0.9314
	Column-wise 2to1	0.7033	0.8942	0.7034	0.8976	0.7035	0.8948	0.7380	0.9142	0.7378	0.9142	0.7378	0.9143	0.7633	0.9313	0.7630	0.9314	0.7635	0.9317
	Column-wise 4to1	0.7033	0.8946	0.7034	0.8955	0.7034	0.8949	0.7381	0.9143	0.7380	0.9143	0.7380	0.9143	0.7636	0.9314	0.7636	0.9314	0.7637	0.9314

the accuracies of three benchmark CNN models (ResNet-18, ResNet-34, and ResNet-50) applied to the optimal MAC unit areas under ISO-area condition of 1885um^2 with the supply voltage of 0.68V (shown in Fig. 12). For ResNet-34, when column-wise/column-wise 2to1/column-wise 4to1 are applied to calculate the sensitivities of MAC units, top-1 accuracy of three mapping strategies are a way higher than previous error resilient scheme [10]. The accuracy of column-wise 4to1 is same as the original (without voltage scaling) top-1 accuracy of ResNet-34. In Fig. 13(b), the accuracies of three mapping strategies and previous one are compared according to different ISO-area conditions (1785um^2, 1885um^2, 1985um^2), where it is shown that the proposed approaches achieve better accuracies for different ISO-area conditions. In Fig. 14, the accuracies of the proposed schemes and previous one [10] are compared under different supply voltages from 1.08V to 0.68V for ResNet-18/ ResNet-34/ ResNet-50 models. As shown in the figure, even with 0.73V of supply voltage, the proposed three mapping strategies show negligible accuracy degradation compared to the original accuracy. In Table I, all the accuracy

TABLE II
THE COMPARISON OF VARIOUS SYSTOLIC ARRAY BASED DNN ACCELERATORS

Designs	[5] (redesigned)	[10] (redesigned)	Proposed		
			Column-wise	Column-wise 2to1	Column-wise 4to1
Technology	65nm	65nm	65nm	65nm	65nm
Bitwidth [bits]	8	8	8	8	8
Number of MAC	65536	65536	65536	65536	65536
Frequency [MHz]	526	526	641	641	641
Throughput [10^9 bits/sec]	16.8	16.8	20.5	20.5	20.5
Area [mm^2]	241.3	282.4	214.2	220.7	228.7
Power Consumption [W]	66.5 @1.08V	78.5 @1.08V 46.4$^+$ @0.78V*	72.5 @1.08V 33.2$^+$ @0.73V*	74.6 @1.08V 34.1$^+$ @0.73V*	77.2 @1.08V 30.5$^+$ @0.68V*

$^+$ Scaled Power Consumption = Power Consumption × (Scaled Voltage^2/1.08^2)
* Minimum supply voltage with same accuracy loss ≤ 3% for ResNet-18

results of the proposed three mapping strategies are presented with different the benchmark CNN models, varying ISO-area conditions and supply voltages scaling.

B. Implementation Results

To verify the effectiveness of the heterogeneous MAC unit sizing approach, the proposed voltage scalable DNN accelerator has been implemented using 65 nm CMOS technology. The proposed architecture is coded in Verilog and synthesized using Synopsys Design Compiler. IC complier is used for auto placing and routing to get the post-layout results including clock tree and hold buffers. To obtain the power/energy consumption results, Synopsys Primetime PX and ILSVRC2012 validation data set image inputs (50,000 images) are used in the post-layout simulations in TYPICAL 1.08V (without voltage scaling), 25° corner @641 MHz. Please note that activation buffers (SRAM) and weight FIFOs are not counted in the implementation results since the proposed work is focused on the datapath (MAC units) design. The baseline TPU [5] without additional Razor pipeline stage is implemented for comparison. As presented in Table II, the proposed column-wise 4to1 design can achieve 54.1% of energy savings compared to the baseline TPU [5], at minimum supply voltage (with same accuracy loss $\leq$ 3%). In [5] and [10], due to the longer critical paths of [5] and [10] compared to the proposed pipelined designs, the area increases to meet the operating frequency. The previous error-resilient scheme based architecture [10] is also implemented for comparison. Table II presents the implementation results and comparisons of the proposed voltage scalable accelerator and the previous one [10]. The architecture of [10] which is based on the systolic array of 256 × 256 MAC units with a Razor flip-flop and a MUX per one MAC unit is redesigned using 65nm CMOS technology. As presented in Table II, the proposed accelerator using column-wise 4to1 sorting achieves 34.2% energy savings compared to that of [10] under same 3% accuracy loss (compare to original accuracy) for ResNet-18. Using column-wise sorting, the proposed accelerator can achieve 28.5% energy savings compared to that of [10] with comparable accuracy (accuracy loss $\leq$ 0.5%).

VI. CONCLUSION

In this paper, we present the error resilient techniques for voltage scalable DNN accelerator. We present sensitivity analysis using Taylor expansion to reduce the simulation time. When conventional approach is used, the simulation time is prohibitive to perform the fine-grained error injection for large scale DNNs. Based on the sensitivity analysis, baseline systolic array DNN accelerator is redesigned using heterogeneous MAC units with the sensitivity map for all 256x256 MAC units, where more sensitive weights are allocated to more robust MAC units, and less sensitive weights are assigned to less robust MAC units. Using dynamic programming, the sizes of MAC units are selected to achieve best DNN accuracy under ISO-area constraints. The proposed DNN accelerator with heterogeneous MAC units allows more aggressive voltage scaling compared to the accelerators with identical MAC units. The simulations with ImageNet dataset show that significant energy savings can be achieved with minor accuracy degradation even with very aggressive supply voltage scaling.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2012, pp. 1097–1105.
[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.
[3] D. Amodei *et al.*, "Deep speech 2: End-to-end speech recognition in English and mandarin," in *Proc. Int. Conf. Mach. Learn.*, vol. 48, Jun. 2016, pp. 173–182.
[4] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 51, no. 1, pp. 127–138, Jan. 2017.
[5] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. ACM/IEEE 44th Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2017, pp. 1–12.
[6] Y. Chen *et al.*, "Dadiannao: A machine-learning supercomputer," in *Proc. 47th Annu. IEEE/ACM Int. Symp. Microarchitecture*, 2014, pp. 609–622.
[7] P. N. Whatmough, S. K. Lee, D. Brooks, and G.-Y. Wei, "DNN engine: A 28-nm timing-error tolerant sparse deep neural network processor for IoT applications," *IEEE J. Solid-State Circuits*, vol. 53, no. 9, pp. 2722–2731, Sep. 2018.
[8] B. Reagen *et al.*, "Minerva: Enabling low-power, highly-accurate deep neural network accelerators," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Archit. (ISCA)*, Jun. 2016, pp. 267–278.
[9] B. Reagen *et al.*, "Ares: A framework for quantifying the resilience of deep neural networks," in *Proc. 55th ACM/ESDA/IEEE Design Automat. Conf. (DAC)*, Jun. 2018, pp. 1–6.
[10] J. Zhang, K. Rangineni, Z. Ghodsi, and S. Garg, "Thundervolt: Enabling aggressive voltage underscaling and timing error resilience for energy efficient deep learning accelerators," in *Proc. 55th Annu. Design Automat. Conf.*, 2018, p. 19.
[11] D. Ernst *et al.*, "Razor: Circuit-level correction of timing errors for low-power operation," *IEEE Micro*, vol. 24, no. 6, pp. 10–20, Nov./Dec. 2004.
[12] P. N. Whatmough, S. Das, and D. M. Bull, "A low-power 1-GHz razor FIR accelerator with time-borrow tracking pipeline and approximate error correction in 65-nm CMOS," *IEEE J. Solid-State Circuits*, vol. 49, no. 1, pp. 84–94, Jan. 2014.
[13] P. N. Whatmough, S. Das, D. M. Bull, and I. Darwazeh, "Circuit-level timing error tolerance for low-power DSP filters and transforms," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 6, pp. 989–999, Jun. 2013.

[14] P. Whatmough, S. Das, D. M. Bull, and I. Darwazeh, "Error-resilient low-power DSP via path-delay shaping," in *Proc. 48th Design Automat. Conf.*, 2011, pp. 1008–1013.
[15] C. E. Leiserson, C. Stein, R. Rivest, and T. H. Cormen, "Dynamic programming," in *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, 2009, pp. 323–369.
[16] X. Jiao, M. Luo, J.-H. Lin, and R. K. Gupta, "An assessment of vulnerability of hardware neural networks to dynamic voltage and temperature variations," in *Proc. 36th Int. Conf. Comput.-Aided Design*, 2017, pp. 945–950.
[17] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, May 2016, pp. 1–14.
[18] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2704–2713.
[19] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2017, pp. 1–17.
[20] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," 2016, *arXiv:1603.04467*. [Online]. Available: https://arxiv.org/abs/1603.04467
[21] D. Jeon, M. Seok, Z. Zhang, D. Blaauw, and D. Sylvester, "Design methodology for voltage-overscaled ultra-low-power systems," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 59, no. 12, pp. 952–956, Dec. 2012.
[22] Synopsys. (2017). *PrimeTime*. [Online]. Available:https://www.synopsys.com/implementation-and-signoff/signoff/primetime.html
[23] W. Choi, D. Shin, J. Park, and S. Ghosh, "Sensitivity based error resilient techniques for energy efficient deep neural network accelerators," in *Proc. 56th Annu. Design Automat. Conf.*, 2019, p. 204.

Dongyeob Shin (S'13) received the B.S. degree in electronics engineering from Korea University, Seoul, South Korea, in 2013. He is currently pursuing the integrated master's and Ph.D. degree with the VLSI Signal Processing Research Laboratory, Korea University. His research interests include low-power and energy-efficient VLSI design.

Wonseok Choi (S'16) received the B.S. degree in electronics engineering from Chung-Ang University, Seoul, South Korea, in 2016. He is currently pursuing the master's degree with the VLSI Signal Processing Research Laboratory, Korea University. His research interests include low-power and high-performance neural network processor designs in advanced technologies.

Jongsun Park (M'05–SM'13) received the B.S. degree in electronics engineering from Korea University, Seoul, South Korea, in 1998, and the M.S. and Ph.D. degrees in electrical and computer engineering from Purdue University, West Lafayette, IN, USA, in 2000 and 2005, respectively.

He joined the Electrical Engineering Faculty, Korea University, Seoul, in 2008. He was with the Digital Radio Processor System Design Group, Texas Instruments, Dallas, TX, USA, in 2002. From 2005 to 2008, he was with the Signal Processing Technology Group, Marvell Semiconductor Inc., Santa Clara, CA, USA. His research interests focus on variation-tolerant, low-power, high-performance VLSI architectures and circuit designs for digital signal processing, and digital communications.

Dr. Park is currently a member of the Circuits and Systems for Communications Technical Committee of the IEEE Circuits and Systems Society. He has served as a Guest Editor for the IEEE TRANSACTIONS ON MULTI-SCALE COMPUTING SYSTEMS. He has also served in the technical program committee of various IEEE/ACM conferences, including ICCAD, ISLPED, ISCAS, ASP-DAC, HOST, VLSI-SoC, ISOCC, and APCCAS. He is also an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS II: EXPRESS BRIEFS.

Swaroop Ghosh (SM'13) received the B.E. degree (Hons.) from IIT Roorkee, Roorkee, India, in 2000, and the Ph.D. degree from Purdue University in 2008.

He was a Senior Research and Development Engineer in advanced design with Intel Corporation from 2008 to 2012. He was with the faculty of University of South Florida (USF) from 2012 to 2016. Since 2016, he has been an Assistant Professor with Pennsylvania State University. His research interests include emerging memory technologies, hardware security, and digital testing. He was a recipient of the Intel Technology and Manufacturing Group Excellence Award in 2009, the Intel Divisional Award in 2011, the Intel Departmental Awards in 2011 and 2012, the USF Outstanding Research Achievement Award in 2015, the College of Engineering Outstanding Research Achievement Award in 2015, the DARPA Young Faculty Award (YFA) in 2015, the ACM SIGDA Outstanding New Faculty Award in 2016, the YFA Director's Fellowship in 2017, the Monkowsky Career Development Award in 2018, and the Lutron Spira Teaching Excellence Award in 2018. He is currently a Senior Member of the National Academy of Inventors (NAI). He has served in the technical program committee of ACM/IEEE conferences, such as DAC, ICCAD, CICC, DATE, ISLPED, GLSVLSI, Nanoarch, and ISQED. He has served as the Program Chair of ISQED 2019 and DAC Ph.D. Forum 2016 and the Track (Co)-Chair of CICC 2017–2019, ISLPED 2017–2018, and ISQED 2016–2017. He has served as an Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I and the IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN and a Senior Editorial Board Member of the IEEE JOURNAL OF EMERGING TOPICS ON CIRCUITS AND SYSTEMS (JETCAS). He has also served as a Guest Editor for the IEEE JETCAS and the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS.