International Journal of Reconfigurable and Embedded Systems (IJRES)
Vol. 10, No. 1, March 2021, pp. 1~10
ISSN: 2089-4864, DOI: 10.11591/ijres.v10.i1.pp1-10

A design methodology for approximate multipliers in convolutional neural networks: A case of MNIST

Kenta Shirane, Takahiro Yamamoto, Hiroyuki Tomiyama
Graduate School of Science and Engineering, Ritsumeikan University, Japan

Article Info

Article history:

Received Nov 10, 2020
Revised Jan 10, 2021
Accepted Jan 20, 2021

Keywords:

Approximate computing
Approximate multiplier
CNN
MNIST

ABSTRACT

In this paper, we present a case study on approximate multipliers for MNIST Convolutional Neural Network (CNN). We apply approximate multipliers with different bit-width to the convolution layer in MNIST CNN, evaluate the accuracy of MNIST classification, and analyze the trade-off between approximate multiplier's area, critical path delay and the accuracy. Based on the results of the evaluation and analysis, we propose a design methodology for approximate multipliers. The approximate multipliers consist of some partial products, which are carefully selected according to the CNN input. With this methodology, we further reduce the area and the delay of the multipliers with keeping high accuracy of the MNIST classification.

Corresponding Author:

Kenta Shirane
Graduate School of Science and Engineering
Ritsumeikan University
1-1-1 Noji-Higashi, Kusatsu, Shiga 525-8577, Japan
Email: kenta.shirane@tomiyama-lab.org

1. INTRODUCTION

Convolutional Neural Networks (CNNs) are well-known as one of the state-of-the-art approaches in the fields of media processing such as computer vision, speech recognition, and so on. Recently, designs of computing systems with CNNs have been extensively studied by researchers. In general, CNNs are computationally expensive due to repetition of the matrix multiplication that is the most critical bottleneck in CNNs [1]. In order to reduce the computational cost of matrix multiplication, approximate computing attracts increasing attention. Approximate computing is considered as a promising approach to design of area-, performance- or power-efficient circuits for applications which are resistant to a certain amount of computational inaccuracy. Approximate calculation circuits have been actively surveyed [2-5]. According to the work in [6] and [7], which investigated the recent studies in the area of approximate computing, the design of approximate multipliers has extensively been studied. Many approximate multipliers proposed calculate the upper bits exactly and lower bits approximately [8-10]. Yamamoto et al. proposed a methodology to systematically design 8-bit approximate array multipliers by gradually removing adders, and analyzed the trade-offs among circuit area, delay, power and accuracy of the approximate multipliers [11]. Boroumand et al. explored the design space of approximate multipliers using a family of approximate compressors as building blocks for the partial product reduction tree [12]. They presented a tool that allows the user to specify an allowable level of error tolerance, and returns the minimum area, delay, or power approximate multiplier that provides the level of accuracy. Liu et al. proposed a new approximate multiplier with highly efficient partial product accumulation and error reduction [13]. The new approximate multiplier has high accuracy but is more efficient in power and performance. In these works, approximate calculation

circuits are designed with small calculation error. In addition, there are many researches on approximate multipliers that reduce the error in the final output of CNN.

As mentioned above, approximate computing is studied as an effective method specialized to CNN [14-16]. An error resilience analysis was performed in order to determine key constraints for the design of approximate multipliers that are employed in the resulting structure of CNN [17]. Moreover, the paper showed the capability of the back-propagation learning algorithm for CNNs containing the approximate multipliers in MNIST and Street-View House Number dataset. Hammad et al. simulated the VGGNet [18] which employs an approximate multiplier in the layers [19]. The simulation results show that the approximate multiplier keeps the high accuracy of VGGNet. Sim et al. proposed a new Stochastic Computing (SC) multiplication algorithm and its vector extension which is called SC-MVM (Matrix-Vector Multiplier) [20]. The experimental results show that new SC-based CNN is more accurate and 40× to 490× more energy-efficient in computation than the conventional SC-based ones. SC is one of the methods of approximate computing and has been actively studied [21, 22]. Courbariaux et al. proposed a method which trains neural network parameters as binary weight and activation [23]. This network, which is called BinaryNet, can make a reduction of memory usage and use XNOR circuits as a multiplier, so many approaches of acceleration or/and implementation have been proposed [24-27]. Li et al. and Balaji et al. designed 8-bit fixed-point approximate multipliers for LeNet and implemented to FPGA [28, 29]. These are more efficient in delay and resource than exact computation with high accuracy of MNIST classification. For these works, approximate computing can reduce computational resources and CNN's trained parameters.

In this paper, we present a design methodology of approximate multipliers for MNIST CNN. As a preliminary preparation of deign of approximate multipliers, we design 64 multipliers with simple reduction of the bit-width of an 8-bit multiplier, and evaluate trade-off between the accuracy of MNIST classification, the area and the delay of the multipliers. Based on the analysis, we design approximate multipliers based on two methods. One of the methods is based on comprehensive analysis of upper 2-bit partial products of 8-bit multiplier, and the other is selection of the partial products according to result of first method and 8-bit multiplier's analysis. With these improvements, we further reduce the area and the delay of the multipliers with keeping high accuracy in MNIST classification. The rest of this paper is organized as follows. Section 2. shows outline of general CNN and MNIST CNN in this paper. Section 3 describes simple approach to bit-width reduction and the experiments. Section 4. describes a design methodology of approximate multiplier with two-step approach, and Section 5. concludes this paper with a summary.

2. CNN FOR MNIST CLASSIFICATION

In this section, we explain a general Convolutional Neural Network (CNN), its convolution layer and MNIST CNN in this work. CNN is a neural network which has deep layer including convolution layer. CNN is known as a high-accuracy image classification model. One of the most popular image classification models is Alexnet [30]. In 2012, Alexnet won ILSVRC by a big margin over other models. This victory triggered prevalence of CNN in the world and many of neural networks in image classification are based on CNN today.

2.1. Convolution layer

In this subsection, we explain about a general convolution layer. A Convolution layer performs convolution operation on an input image with a filter (weight), its outputs sum up bias, and the outputs are propagated to the next layer or activation function. The filter and bias are trained and saved in convolution layer. In convolution operation, the products of input and filter are summed up and stored to the corresponding output. Convolution operation performs at an interval which is defined "stride". Figure 1 represents an example of convolution operation. When a 5×5 input, a 3×3 filer and stride value "1" are given, the size of the output is organized as 3×3. In this case, the convolution layer contains 81 multiplications per filter. In general, CNN contains many filters. In this work, CNN contains 30 filters and the convolution layer performs multiplication operations 432,000 times per image.

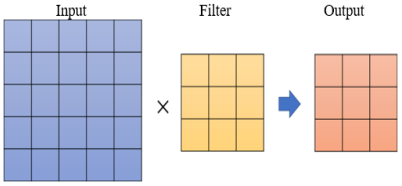


Figure 1. Convolution layer

2.2. An example of CNN for MNIST

In this subsection, we explain about MNIST CNN in this paper. Figure 2 shows the model of MNIST CNN and MNIST dataset. MNIST CNN consists of Convolution (convolution layer), Pooling (max pooling layer), Affine (affine layer), ReLU and Softmax. Next part describes each of the layers or the activation functions, except from the convolution layer.

Pooling selects the maximum number of input elements in filter range and stores the corresponding output. This layer can reduce processing weight with constant approximation. This operation does not contain parameters such as weight or bias, and keeps the spatial information of input image.

Affine multiplies each input element and weight value, sums up these products and stores the corresponding output. This layer connects all spatial information to each class. This operation contains parameters which are weight and bias, and does not keep the spatial information of input image.

ReLU and Softmax are activation functions which prepare input data. ReLU fixes "0" or less value to "0" and stores the corresponding output. It is known as ramp function. Softmax fixes input value to probability value. This function is used before finally output and highest probability value's class is CNN output.

The CNN trained MNIST data set. MNIST data set contains handwritten digits from "0" to "9", which is a collection of 28×28 quasi binary images, and the pixel value is varied from "0" to "1". MNIST is one of the most elementary datasets in image classification of CNN, so we firstly use this dataset to examine the impact of approximate multiplier to image classification.

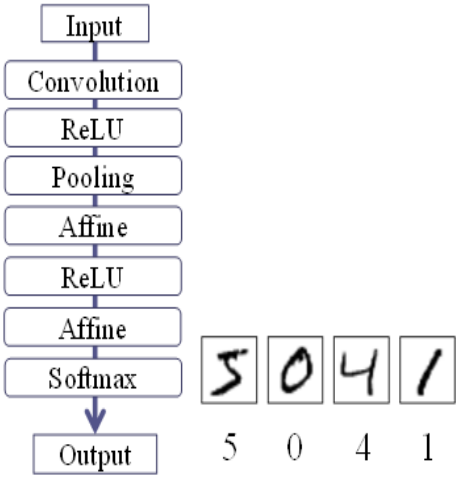


Figure 2. An example of CNN for MNIST

3. PRELIMINARY ANALYSIS ON BIT-WIDTH

This section describes how to reduce bit-width of multiplication and the experiments to Figure out the scale of multiplier which CNN needs. The experiment is the preparation of Section 4. and the purpose is an exploration of the multiplier which keeps high accuracy of image classification.

3.1. Bit-Width reduction approach

In this work, we design different bit-width multipliers by reducing partial products of 8-bit multiplier. The 8-bit multiplier is general small scale, but MNIST is known as dataset which can be classified by using small scale circuit. The structure of BinaryNet is different from CNN in this work, but BinaryNet has a high accuracy of image classification and MNIST classification does not seem to need high precision multiplication.

The 8-bit multiplier is based on an exact 8-bit array multiplier since the 8-bit array multiplier is so small, and the 8-bit multiplier does not have to apply Wallace tree and/or booth's multiplication because of bad tradeoff between the overhead and the performance improvement of the multiplier. Figure 3 is an exact 8-bit array multiplier. The multiplication uses multiplicand and multiplier to obtain partial products. Sum of the partial products is a product. Sum operation starts from upper-right to lower-left. The product of 8-bit multiplier becomes 16-bit product.

In this reduction, we replace the multiplicand or the multiplier with "0" in order from lowest-order bit. By this operation, the corresponding multiplier or multiplicand is useless, and this partial product's value becomes '0'. In addition, the multiplicand and multiplier have 8-bit information, and we can reduce different bit-width between the multiplicand and the multiplier. Figure 4 is an example of 7×6-bit multiplier. We replace least bit of multiplicand and lower 2-bit of the multiplier with "0", so we can delete 22 partial products and lower 3-bit of product. We can reduce different bit-width between the 8 multiplicands and 8 multipliers, so we can create 64 multipliers in different bit-width.

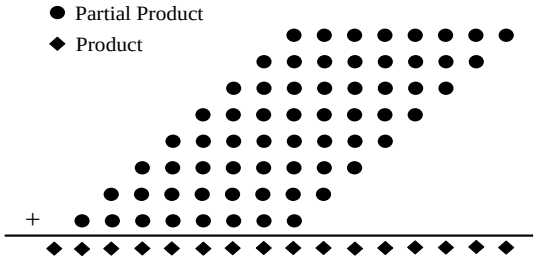


Figure 3. An exact 8-bit multiplier

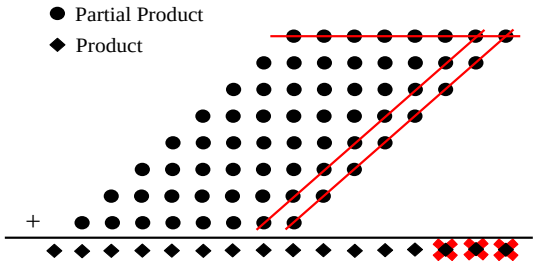


Figure 4. An example of 7×6-bit multiplier

3.2. Preliminary results

We designed MNIST CNN in order to apply self-made multipliers. We applied different bit-width multipliers to MNIST CNN and evaluated the trade-offs of the accuracy of MNIST classification, the area and the delay of the multiplier. The experiments can Figure out the scale of multiplier which CNN needs, and we improve the multiplier, which keeps high accuracy of image classification, by reference to experimental results in the following section. We can reduce different bit-width between the multiplicand and multiplier, so we can create 64 multipliers in different bit-width. We use CNN parameters learned 2000 training images. We measure the image classification accuracy when CNN recognize 1000 test images. The accuracy with exact computation, which is 32-bit floating point multiplication, is 98.4%. The description of multipliers is written in Verilog. We synthesis the 64 multipliers by LeonardoSpectrum and measure the number of gate and critical path delay. We use Nangate45 which is 45nm process library. When we synthesis the circuit optimized minimizing area.

Figure 5 (a)-(c) are the experimental results of bit-width reduction. The filter data are multiplicand and the image data are multiplier. Figure 5 (a) is the result of accuracy, Figure 5 (b) is the result of the area, and Figure 5 (c) is the result of the delay. In Figure 5 (a), accuracy in 8×8-bit is 98.4% and accuracies in 3×2-bit and 4×1-bit are almost the same as the exact one. The accuracy in 2×1-bit is 97.1% and the accuracy in less than 2×1-bit decrease significantly. Whenever the filter bit-width is 1, the accuracy is extremely low. Therefore, the MNIST CNN needs about 2×2- or more bit multiplier to keep high accuracy of image classification. In Figure 5 (b) shows that the area of multiplier decreases in proportion to the scale of multiplier. The area in 3×2-bit is 15 gates that contains HA (half adder) and many AND gates, the area in 4×1-bit is 4 gates which are 4 AND gates and the one in 2×1-bit is 2 gates which are two AND gates. In Figure 5 (c) shows that the delay of multiplier decreases in proportion to the scale of multiplier. The delay in 3×2-bit is 0.10 ns. The delay in 4×1-bit is 0.02 ns as well as the 2×1-bit.

Table 1 is the detail of upper order 3-bit image bit-width. Cases of 3×2-bit and 4×1-bit are same accuracy and contain five partial products. However, the area of the 3×2-bit multiplier is larger than that of 4×1-bit, and the delay of the 3×2-bit multiplier is longer delay than a case of 4×1-bit. Therefore, a case of 3×2-bit is better power efficiency than a case of 4×1-bit. Besides, filter data (multiplicand) are more important to MNIST CNN than image data (multiplier). In the experimental results, MNIST CNN needs 4 gates to achieve almost the same accuracy as accuracy of classification with exact computation. In addition, when the number of gates is 3 in 3×1-bit, accuracy is 98.0% and the MNIST CNN needs about 3×2, 4×1 or more-bit multiplier to keep high accuracy of image classification. Therefore, in the next section, the proposed design methodology is used to achieve higher accuracy with 3 or fewer gates.

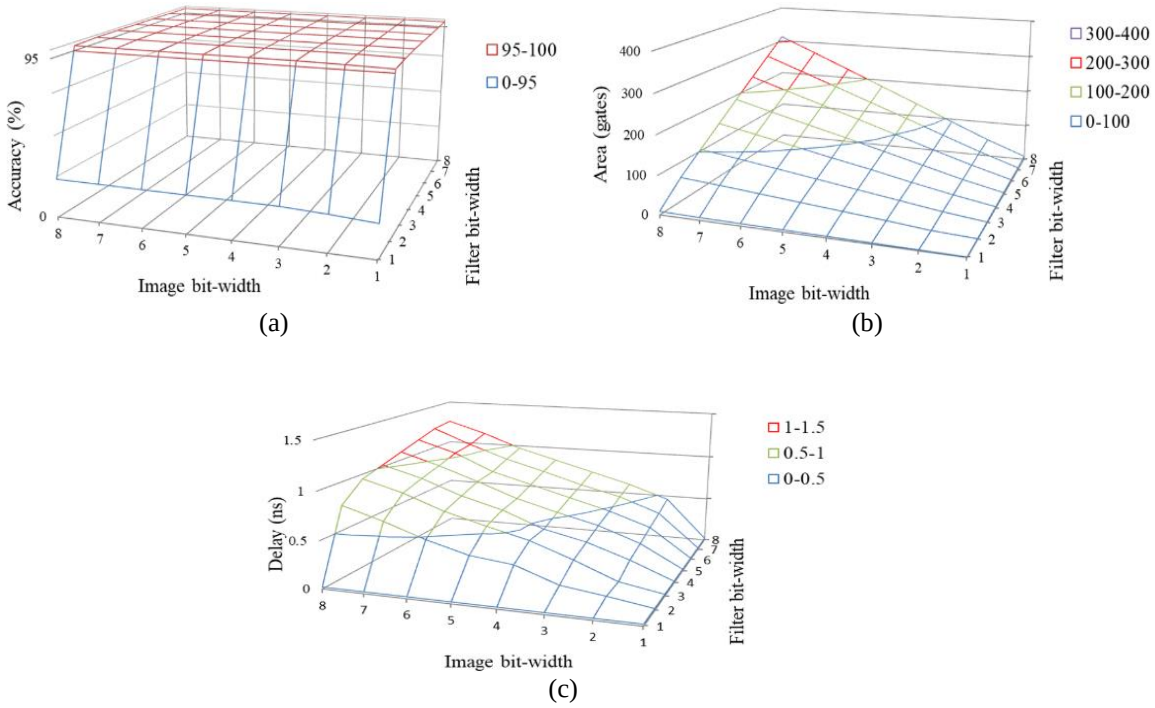


Figure 5. Experimental results of bit-width reduction (a) Accuracy, (b) Area, (c) Delay

Table 1. Detail of upper order 3-bit image bit-width

Filter bit-width	Image bit-width	Accuracy (%)	Area (gates)	Delay (ns)
1	1	20.4	1	0.02
2	1	97.1	2	0.02
3	1	98.0	3	0.02
4	1	98.5	4	0.02
5	1	98.5	5	0.02
6	1	98.4	6	0.02
7	1	98.4	7	0.02
8	1	98.4	9	0.02
1	2	22.1	2	0.02
2	2	97.6	7	0.09
3	2	98.5	15	0.10
4	2	98.7	22	0.20
5	2	98.6	28	0.26
6	2	98.6	35	0.33
7	2	98.6	42	0.39
8	2	98.6	48	0.46
1	3	23.2	3	0.02
2	3	97.9	17	0.15
3	3	98.5	30	0.28
4	3	98.6	42	0.35
5	3	98.6	55	0.42
6	3	98.3	67	0.48
7	3	98.3	80	0.55
8	3	98.3	92	0.61

4. A DESIGN METHODOLOGY FOR APPROXIMATE MULTIPLIERS

This section describes a design methodology of approximate multipliers for MNIST CNN. In the first subsection, we explain an overview of a design methodology of approximate multipliers for CNN. In the next two subsections, we applied the design methodology to MNIST CNN. At every step, we evaluated the trade-offs of the accuracy of MNIST classification, the area and the delay of the multiplier.

4.1. A design methodology

This subsection describes overview of the proposed design methodology of approximate multipliers for CNN. Figure 6 is flowchart of the proposed design methodology. In the first harf of the methodology, we analyze partial products of exact multiplier and select significant partial products for CNN as an approximate multiplier. If the accuracy of CNN is not significantly higher or the area or delay don't satisfy user requirements, we analyze a broader scope again.

In the second harf of the methodology, we analyze in more detail partial products of exact multiplier for combining to the approximate multiplier designed in the first half of the methodology. If the accuracy of CNN is lower than exact one, we analyze a broader scope again. In the design methodology, we can design approximate multipliers that maintain a high recognition rate of CNN without compromising the user's requirements. In the next two subsections, we applied the design methodology to MNIST CNN.

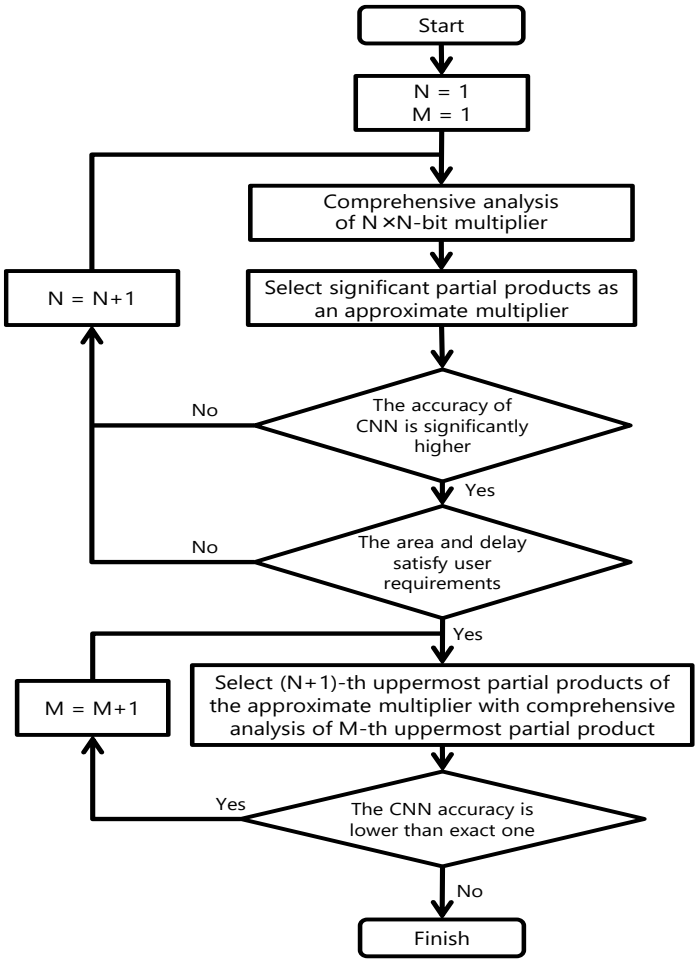


Figure 6. Flowchart of the proposed design methodology

4.2. Comprehensive analysis of 2-bit partial products

This subsection describes the example of the comprehensive analysis of 2×2-bit multiplier. According to the experimental results of Section 3, we analyzed only higher 2-bit partial products. Figure 7 is an exact 2×2-bit multiplier. 'a' is filter bit, 'b' is image bit, 'p' is product, and HA is half-adder circuit. We reduced partial products from lower bit in Section 3, but we selected one to four partial products in our

design methodology. We analyzed the accuracy of MNIST classification, the area and the delay of the approximate multiplier which has one to four partial products of 2×2-bit multiplier comprehensively. In this experiment, we can explore the combination of partial products which keeps high accuracy of image classification. The experimental conditions are based on Section 3.

Table 2 is the result of comprehensive analysis of the 2-bit partial products. When the number of partial products is 1, the case of a_7b_6 is the highest accuracy, but it is not enough high accuracy. When the number of partial products is 2, the case of a_6b_7 and a_7b_7 is highest accuracy, 97.1%. In this case, the number of gates is 2 and delay is 0.02 ns. When the number of partial products is 3 or 4, no other case of partial products is more efficiency than the case of a_6b_7 and a_7b_7. Figure 8 is the logic level circuit of a_7b_7 and a_6b_7. HAs are removed. In the second harf of the methodology, we try to achieve higher accuracy by a_6b_7, a_7b_7 and another partial product.

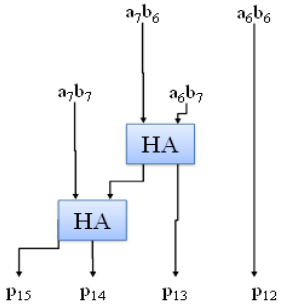


Figure 7. 2-bit exact multiplier

Table 2. Comprehensive analysis in upper order 2-bit partial products

Number of Partial Product	Partial Product	Accuracy (%)	Area (gates)	Delay (ns)
1	a_6b_6	43.7	1	0.02
	a_6b_7	17.8	1	0.02
	a_7b_6	63.6	1	0.02
	a_7b_7	20.4	1	0.02
2	a_6b_6, a_6b_7	69.5	2	0.02
	a_6b_6, a_7b_6	86.6	2	0.02
	a_6b_6, a_7b_7	94.9	2	0.02
	a_6b_7, a_7b_6	91.0	4	0.06
	a_6b_7, a_7b_7	97.1	2	0.02
	a_7b_6, a_7b_7	22.1	2	0.02
3	a_6b_6, a_6b_7, a_7b_6	90.5	5	0.08
	a_6b_6, a_6b_7, a_7b_7	90.0	3	0.02
	a_6b_6, a_7b_6, a_7b_7	96.9	3	0.02
	a_6b_7, a_7b_6, a_7b_7	97.5	6	0.08
4	a_6b_6, a_6b_7, a_7b_6, a_7b_7	97.6	7	0.09

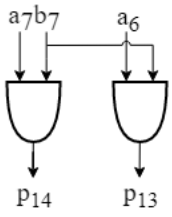


Figure 8. Logic-level circuit of a_7b_7 and a_6b_7

4.3. Selection of the third uppermost product

This subsection describes the example of selection of the third uppermost products. Figure 9 is a logic-level circuit of a_6b_7, a_7b_7 and a_ib_j. a_ib_j is one of the ten partial products which is possible to output upper than p_{11} which is the fifth uppermost product. In the ten cases, we analyze the accuracy of MNIST classification, the area and the delay of the multiplier. The experimental conditions are based on Section 3.

Table 3 is the result of selection of the third significant product. The number of gates and the delay always keep 3 gates and 0.02 ns, respectively. The highest accuracy achieves 98.4% in a_5b_6, and this is the same result as exact computation. Therefore, we achieve the same accuracy as MNIST classification with exact computation by using approximate multiplier composed of a_7b_7, a_6b_7 and a_5b_6.

Figure 10 is the logic-level circuit of a_6b_7, a_7b_7 and a_5b_6. In the case of MNIST CNN, the proposed design methodorogy is efficient for designing approximate multipliers with small scale, low delay and high image recognition of CNN. Because a_6b_7 and a_7b_7 contain neither a_5 nor b_6, we consider that many varieties of multiplicand and multiplier improve CNN accuracy of classification. In addition, a_5b_6 is usually output to p_{11}, but it is output to p_{12} in this case. Therfore, we considered that higher bit information is not always more important to image classification with CNN than lower bit information.

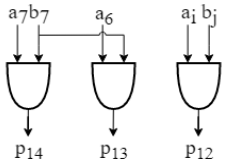


Figure 9. Logic-level circuit of a_7b_7, a_6b_7 and a_ib_j

Table 3. Selection of the third significant product

p_{12}	Accuracy (%)	Area (gates)	Delay (ns)
a_4b_7	98.2	3	0.02
a_5b_6	98.4	3	0.02
a_5b_7	98.0	3	0.02
a_6b_5	96.6	3	0.02
a_6b_6	96.9	3	0.02
a_6b_7	96.2	3	0.02
a_7b_4	97.3	3	0.02
a_7b_5	97.4	3	0.02
a_7b_6	97.3	3	0.02
a_7b_7	97.3	3	0.02

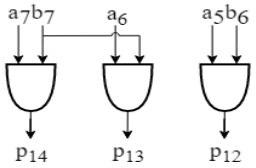


Figure 10. Logic-level circuit of a_7b_7, a_6b_7 and a_5b_6

4.4. Comparison

Table 4 is a comparison table of multipliers with simple bit-width reduction and approximate multipliers based on the proposed methodology. The labels 8×8, 3×2, 2×2 and 2×1 are based on analysis of Section 3 the label AP (approximate) 2×1 is based on subsection 4.2, and the label AP 3×2 is based on subsection 4.3. AP 3×2-bit multiplier has the same accuracy of MNIST recognition as the exact 8×8 and 3×2-bit multipliers. In addition, the area of AP 3×2-bit multiplier improved by 80% over the area of exact 3×2-bit multiplier, and the delay of AP 3×2-bit multiplier improved by 80% over the area of exact 3×2-bit multiplier. Therefore, the design methodology of approximate multipliers is efficient for MNIST CNN.

Table 4. Comparison table of multipliers with simple bit-width reduction and approximate multipliers

Multiplier	Accuracy (%)	Area (gates)	Delay (ns)
8×8	98.4	312	1.26
3×2	98.5	15	0.10
2×2	97.6	7	0.09
2×1	97.1	2	0.02
AP 2×1 (a_7b_7 and a_6b_7)	97.1	2	0.02
AP 3×2 (a_6b_7, a_7b_7 and a_5b_6.)	98.4	3	0.02

5. CONCLUSIONS

In this paper, we proposed a design methodology of approximate multipliers for CNN. We achieved high accuracy of MNIST classification with approximate multipliers based on the proposed methodology. With the design methodology, by using the approximate multiplier, we achieved the same accuracy as MNIST classification with exact computation and improved the area and delay performance. In future, we plan to apply the proposed methodology of approximate multipliers to more complex and larger scale CNN. Besides, we plan to implement approximated CNN on FPGA and plan to analyze the impact on FPGA.

ACKNOWLEDGEMENTS

This work is in part supported by KAKENHI 19H04081 and 20H00590.

REFERENCES

[1] K. Guo, S. Zeng, J. Yu, Y. Wang and H. Yang, "A survey of FPGA based neural network accelerator," *ACM Transactions on Reconfigurable Technology and Systems,* vol. 1, no. 1, 2018.

[2] F. M. del Campo, A. Morales-Reyes, R. Perez-Andrade, R. Cumplido, A. G. Orozco-Lugo and C. Feregrino, "A multi-cycle fixed point square root module for FPGAs," *IEICE Electronics Express,* vol. 9, no. 11, pp. 971-977, 2012.

[3] D. Shin and S.K. Gupta, "Approximate logic synthesis for error tolerant applications," *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*, 2010.

[4] N. Zhu, W.L. Goh and K.S. Yeo, "An enhanced low-power high-speed adder for error-tolerant application," *Proceedings of the 12th International Symposium on Integrated Circuits (ISIC'09)*, 2009, pp. 69-72.

[5] V. Gupta, D. Mohapatra, A. Raghunathan and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 32, no. 1, pp. 124-137, 2013.

[6] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *2013 18th IEEE European Test Symposium (ETS)*, 2013, pp. 1-6.

[7] Q. Xu, N. Kim and T. Mytkowicz, "Approximate computing: A survey," *IEEE Design & Test*, vol. 33, no. 1, pp. 8-22, 2015.

[8] B. Shao, and P. Li, "Array-based approximate arithmetic computing: A general model and applications to multiplier and squarer design," *International Symposium on Low Power Electronics and Design (ISLPED),* vol. 64, no. 4, pp. 1081-1090, 2014.

[9] T. A. Drane, T. M. Rose, and G. A. Constantinides, "On the systematic creation of faithfully rounded truncated multipliers and arrays," *IEEE Transactions on Computers*, vol. 63, no. 10, pp. 2513-2525, Oct. 2014.

[10] F. Farshchi, M. S. Abrishami, S.M. Fakhraie, "New approximate multiplier for low power digital signal processing," in *The 17th CSI International Symposium on Computer Architecture & Digital Systems (CADS 2013)*, 2013, pp. 25-30.

[11] T. Yamamoto, I. Taniguchi, H. Tomiyama, S. Yamashita and Y. Hara-Azumi, "A systematic methodology for design and worst-case error analysis of approximate array multipliers," *IEICE Trans. on Fundamentals*, vol. 100, no. 7, pp. 1496-1499, 2017.

[12] S. Boroumand, H. P. Afshar, P. Brisk and S. Mohammadi, "Exploration of approximate multipliers design space using carry propagation free compressors," in *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2018, pp. 611-616.

[13] C. Liu, J. Han and Fabrizio Lombardi, "A Low-Power, High-performance approximate multiplier with configurable partial error recovery," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2014, pp. 1-4.

[14] J. Chang and J. Sha, "An efficient implementation of 2D convolution in CNN," *IEICE Electronics Express*, vol. 14, no. 1, pp. 1-8, 2017.

[15] B. Liu, H. Qin, Y. Gong, W. Ge, M. Xia and L. Shi," EERA-ASR: An energy-efficient reconfigurable architecture for automatic speech recognition with hybrid DNN and approximate computing," *IEEE Access*, vol. 6, pp. 52227-52237, 2018.

[16] D-A. Nguyen, H-H. Ho, D-H. Bui and X-T Tran, "An efficient hardware implementation of artificial neural network based on stochastic computing," in *NAFOSTED Conference on Information and Computer Science (NICS)*, 2018, pp. 237-242.

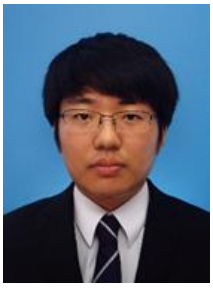
[17] V. Mrazek, S. S. Sarwar, L. Sekanina, Z. Vasicek and K. Roy, "Design of power-efficient approximate multipliers for approximate artificial neural networks," in *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2016, pp. 1-7.

[18] K. Simonyan and A. Zisserman," Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2014.

[19] I. Hammad and K. El-Sankary, "Impact of approximate multipliers on VGG deep learning network," *IEEE Access*, vol. 6, pp. 60438-60444, 2018.

[20] H. Sim and J. Lee, "A new stochastic computing multiplier with application to deep convolutional neural networks," in *IEEE Design Automation Conference (DAC)*, 2017, pp. 1-6.
[21] E. Schkufza, R. Sharma, A. Aiken, "Stochastic optimization of floating-point programs with tunable precision", *ACM SIGPLAN*, vol. 49, no. 6, pp. 53-64, 2014.
[22] A. Alaghi and J. P. Hayes, "Survey of stochastic computing," *ACM Trans. Embedded Computing Systems*, vol. 12, no. 2s, pp. 1-19, 2012.
[23] M. Courbariaux and Y. Bengio "BinaryNet: Training deep neural networks with weights and activations constrained to +1 or −1," *arXiv*, 2016.
[24] N. J. Fraser, Y. Umuroglu, G. Gambardella, M. Blott, P. Leong, M. Jahre and K. Vissers, "Scaling binarized neural networks on reconfigurable logic," in *ACM International Conference Proceeding Series (ACM-ICPS)*, 2017, pp. 25-30.
[25] E. Nurvitadhi, D. Sheffield, J. Sim, A. Mishra, G. Venkatesh and D. Marr, "accelerating binarized neural networks: Comparison of FPGA, CPU, GPU, and ASIC," in *International Conference on Field-Programmable Technology (FPT)*, 2016, pp. 77-84.
[26] Y. Zhang, Z. Zhou, P. Huang, M. Fan, R. Han, W. Shen, L. Liu, X. Liu and J. Kang, "An improved hardware accelaration architecture of binary neural network with 1T1R array based forward/backward propagation module," in *Silicon Nanoelectronics Workshop (SNW)*, 2019, pp. 1-2.
[27] P. K. Chundi, P. Liu, S. Park. S. Lee and M. Seok "FPGA-based acceleration of binary neural network training with minimized off-chip memory access," in *IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 2019, pp. 1-6.
[28] Z. Li, L. Wang, S. Guo, Y. Deng, Q. Dou, H. Zhou and W. Lu, "Laius: An 8-bit fixed-point CNN hardware inference engine," in *IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*, 2017, pp. 143-150.
[29] A. Balaji, S. Ullah, A. Das, and A. Kumar," Design methodology for embedded approximate artificial neural networks," in *Great Lakes Symposium on VLSI (GLSVLSI)*, 2019, pp. 489-494.
[30] A. Krizhevsky, I. Sutskever and E. Hinton, "ImageNet classification with deep convolutional neural networks," in *International Conference on Neural Information Processing Systems*, vol. 60, no. 6, pp. 84-90, 2017.

BIOGRAPHIES OF AUTHORS



Kenta Shirane received his BE degree in electronic and computer engineering from Ritsumeikan University in 2019. He is in the Master's degree program at Ritsumeikan University. His research interests include design methodologies for embedded systems.

Takahiro Yamamoto received his BE and ME degrees in electronic and computer engineering from Ritsumeikan University in 2016 and 2018, respectively. At present, he works for Mitsubishi Electric Corporation. His research interests include design methodologies for embedded systems.

Hiroyuki Tomiyama received his BE, ME, and DE degrees in computer science from Kyushu University in 1994, 1996, and 1999, respectively. He worked as a visiting researcher at UC Irvine, as a researcher at ISIT/Kyushu, and as an associate professor at Nagoya University. Since 2010, he has been a full professor with the College of Science and Engineering, Ritsumeikan University. He has served on program and organizing committees for a number of premier conferences including DAC, ICCAD, DATE, ASP-DAC, CODES+ISSS, CASES, ISLPED, RTCSA, FPL, and MPSoC. He has also served as an editor-in-chief for IPSJ TSLDM; an associate editor for ACM TODAES, IEEE ESL, and Springer DAEM; and a chair for the IEEE CS Kansai Chapter and IEEE CEDA Japan Chapter. His research interests include, but are not limited to, design methodologies for embedded and cyber-physical systems.