Area and Energy Optimization for Bit-Serial Log-Quantized DNN Accelerator with Shared Accumulators

Takumi Kudo†, Kodai Ueyoshi†, Kota Ando†, Kazutoshi Hirose†, Ryota Uematsu†,
Yuka Oba†, Masayuki Ikebe†, Tetsuya Aasai†, Masato Motomura†, Shinya Takamaeda-Yamazaki†
†*Graduate School of Information Science and Technology (IST), Hokkaido University,*
†*Kita 14, Nishi 9, Kita-ku, Sapporo, Hokkaido, 060-0814, Japan*

***Abstract*—In the remarkable evolution of deep neural network (DNN), development of a highly optimized DNN accelerator for edge computing with both less hardware resource and high computing performance is strongly required. As a well-known characteristic, DNN processing involves a large number multiplication and accumulation operations. Thus, low-precision quantization, such as binary and logarithm, is an essential technique in edge computing devices with strict restriction of circuit resource and energy. Bit-width requirement in quantization depends on application characteristics. Variable bit-width architecture based on the bit-serial processing has been proposed as a scalable alternative that allows different requirements of performance and accuracy balance by a unified hardware structure. In this paper, we propose a well-optimized DNN hardware architecture with supports of binary and variable bit-width logarithmic quantization. The key idea is the distributed-and-shared accumulator that processes multiple bit-serial inputs by a single accumulator with an additional low-overhead circuit for the binary mode. The evaluation results show that the idea reduces hardware resources by 29.8% compared to the prior architecture without losing any functionality, computing speed, and recognition accuracy. Moreover, it achieves 19.6% energy reduction using a practical DNN model of VGG 16.**

1. Introduction

Deep neural network (DNN) has made great contributions in various artificial intelligence applications, such as face authentication and self-driving technology, with its high-level recognition capability [1]. Recognition accuracy of DNN is enhanced by utilizing wide and deep structure of neural network elements. Modern neural network models, such as ResNet [2], have a very deep and complicated structure with skip connections. While a larger structure naturally achieves higher accuracy, it also requires a huge amount of multiply-add computations. While GPU is a suitable device to accelerate such applications of numerous multiply-add operations, native implementations of DNN on embedded edge devices is quite difficult, due to the available energy capacity. To realize intelligent IoT systems anywhere, energy-efficient and high-performance DNN processing mechanisms are strongly required.

To this end, many DNN accelerators based on Field Programmable Gate Arrays (FPGAs) and Application-Specific Integrated Circuits (ASICs) have been proposed. Prior accelerator architectures mainly focus on optimizations of processing dataflow [3], optimizations of memory access traffic between on-chip buffer and external DRAMs by exploiting the computation behavior of DNN [4].

In addition to the dataflow and memory system optimizations, quantization has been introduced to reduce the amount of entire data and to simplify the processing hardware by utilizing low bit-width arithmetic units. In various DNN accelerators, fixed-point data representation is often used instead of area- and energy-consuming floating-point representation. The evaluation in [5] indicates 8-bit fixed-point quantization is enough to achieve the recognition accuracy equaling or surpassing the original floating-point. Thus, such the quantization enables light-weight and efficient DNN processing on restricted resources of circuit and energy.

BRein [6] is an in-memory neural network accelerator chip based on BinaryNet, where all the weights and activations are quantized into either '-1' or '1'. The binarization enables to reduce the circuit area of calculation with replacing the expensive multiplications with simple bit-wise XNOR operations.

While the binarization is an ultimate data quantization for the data amount, the recognition accuracy is certainly low compared to the floating-point and other quantization methods. Another size-effective quantization is the logarithmic quantization (log-quantization) [7], that represents numeric values of weights and activations in logarithmic domain. Compared to the fixed-point quantization, the logarithmic quantization can represent a wider range of values within a smaller bit-width. Thus, it is much effective to relax the memory and computational requirements at the high recognition accuracy.

In this paper, we propose an area and energy optimized DNN accelerator based on the bit-serial log-quantized DNN accelerator, such as [8]. The baseline architecture features bit-serial computation for flexible bit precision. However, a part of the computation units has an idle time due to the bit-serial computation. We propose the shared accumulator architecture among multiple bit-serial processing elements for the reduction of the circuit area and energy consumption without any functional degradation. We evaluate the area and energy efficiency of the proposed architecture, and compare between the baseline and proposed architecture to show the advantage of the sharing resources.

This paper is organized as follows: Section 2 explains the baseline architecture to be compared with our proposal. Section 3 presents and discusses our proposed architecture that shares accumulators to reduce the circuit

area. Section 4 shows the evaluation results that compare the baseline and proposed architecture. Section 5 describes related works of this work, and Section 6 finally concludes this paper.

2. Baseline Architecture

In this section, we define a baseline architecture used for evaluating improvement of area and energy efficiency. The baseline architecture is based on the prior proposed bit-serial log-quantized DNN accelerator called QUEST [8].

2.1. Overall Design

Figure 1 illustrates an overview of the baseline architecture: QUEST [8]. QUEST is constructed as a 3D-stacking module of a DNN processing die and multiple SRAM dies. All dies are tightly connected via Thru-Chip Interface (TCI) which is a wireless communication interface by inductive coupling [9]. Stacking 3D SRAMs make large capacity (96 MB) and high bandwidth (28.8 GB/s) with small latency (three cycles @ 300 MHz). Since all data such as weights and activations are quantized within four bits, QUEST can process large scale DNN without off-chip memories.

The DNN die on the top of the 3D-stacking module is composed of 24 cores which have processing element (PE) arrays to process logarithmic quantized multiply-and-accumulate (MAC) operations in a bit-serial manner. Therefore, all data are stored vertically and read in a time-multiplexed manner as shown in Fig. 1(b). The behavior of each DNN core is completely managed by its own microcontroller (μ ctrl.). The microcontroller is a light-weight RISC processor that supports a few types of instructions to manage the sequencer and the direct memory access controller (DMAC). The microcontroller configures the overall computations on the PE array and memory accesses via the sequencer and DMAC, respectively. The programmable sequencer generates clock-level control signals of the PE array and scratchpad addresses. The DMAC handles chunked data transfers between the PE arrays and the stacked SRAMs via the TCI interface. Each DMAC accesses both other memory blocks on other cores via the interconnections and local memory blocks including SRAM banks on other dies. This allows processing arbitrary DNN models because of its programmability. The DNN die also has interconnection called a local link for communications between neighbor cores.

In this paper, we propose a new PE array architecture based on the baseline architecture of QUEST to improve area and energy efficiency.

2.2. Logarithmic Quantized Deep Neural Network

Herein, we explain a logarithmic quantization (log-quantization) method used in the baseline architecture. The log-quantization is proposed by [7] which approximate both activations and weights in logarithmic domain. The approximation formula is represented as

$$x \approx Sign(x) \times 2^{Quantize(\log_2 |x|)}.$$

The absolute value of x is approximated by power of two. In a digital processing, all data are expressed in

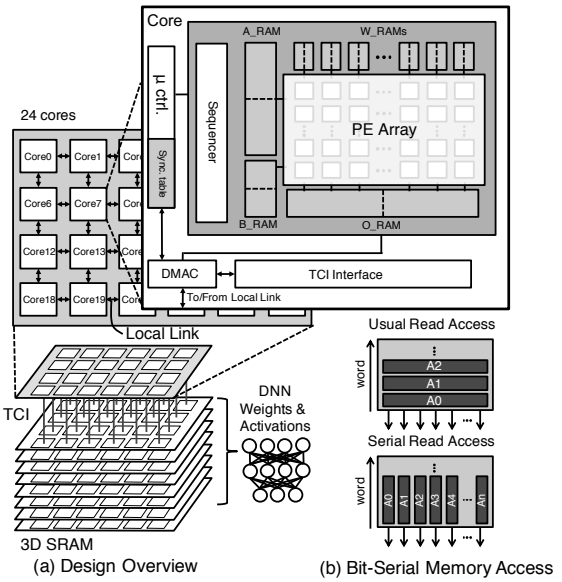

Figure 1. Overall design of log-quantized baseline architecture (QUEST [8]).

binary number, so that logarithmic expression based two is suitable.

The most important feature of log-quantization is that massive-energy-consuming multiplication can be replaced with addition. The log-quantized multiplication of input activations and weights can be approximated as

$$\begin{aligned} a \times w &\approx \{Sign(a) \times 2^{\tilde{a}}\} \times \{Sign(w) \times 2^{\tilde{w}}\} \\ &= Sign(a \times w) \times 2^{\tilde{a}+\tilde{w}}, \end{aligned}$$

where a is input activation, w is weights, $\tilde{a} = Quantize(\log_2 |a|)$ and $\tilde{w} = Quantize(\log_2 |w|)$. Therefore, MAC operations can be processed by simple adder-based calculations.

Another advantage of log-quantization is a low memory footprint and bandwidth requirements because it is enough to maintain high image recognition accuracy of ImageNet dataset within four bits of both activations and weights [8]. Therefore, QUEST architecture supports flexible bit precision within four bits by bit-serial computation.

2.3. PE Array Architecture

Each core has a PE array which processes log-quantized MAC operations in parallel. A column of the PE array (PE_COL) processes an output activation of a neuron. Figure 2 shows the relationship between the operation of the output activation and a PE_COL. The output activation is calculated by MAC of input activations (a) from n prior layer's neurons and weights (w), a bias (b), and an activate function (f). The PE_COL processes the MAC operation in logarithmic domain with the 32 PEs that process in a bit-serial manner. These 32 parallel MAC operations are calculated in a time-multiplexed manner until receiving all inputs (n input activations and n weights). Intermediate partial sums from each PEs are shifted by a shift register, and an ACT unit at the bottom of PE_COL sums up the partial sums and bias, and applies the activate function.

Figure 3 shows the overall diagram of PE array. The 16 PE_COLs are arranged in a row and input activations

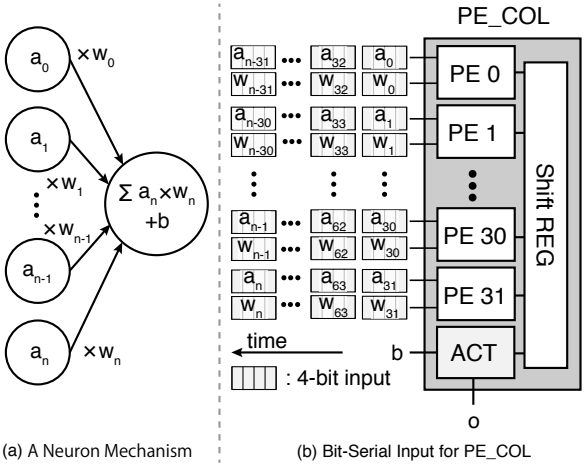

Figure 2. Neuron mechanism and corresponding bit-serial inputs for PE_COL.

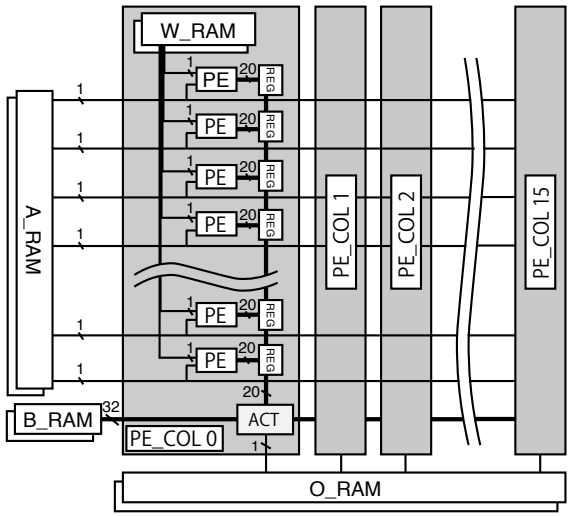

Figure 3. PE array architecture.

are shared. The PE array has four dedicated on-chip RAMs of weights (W_RAM), input activations (A_RAM), biases (B_RAM), and output activations (O_RAM). All RAMs are double buffered to prepare next data in advance to hide the data transfer overhead. The read activations and weights are scattered to one bit in bit-serial manner. Each PE_COL has a different W_MEM independently to provide independent weight to each PE. On the other hand, the input activations of A_MEM are shared among the PEs on the same row. This parallelism allows efficient calculation for both convolution (CONV) and fully-connected (FC) layers which are often used in modern DNNs [2], [10].

2.4. Baseline PE Architecture

In this subsection, we show the baseline PE architecture that processes MAC operations in a logarithmic domain for multiplication and a linear domain for accumulation. Figure 4 shows a detailed block diagram of the baseline PE architecture in QUEST. All weights and input activations are log-quantized and serialized. Therefore, the multiplier can be replaced with a simple serial adder. Since the maximum bit width is 4 (a sign bit and absolute value in three bits), the serial-add output is set as 5 bits (a sign bit and 4 sum bits). To obtain the inputs sequentially, the serial adder processes a sign and MSB of absolute value of sum at the same time. This feature enables the variable bit-width arithmetic operation. The 1-bit mode is a special case that both the input activations and weights have only sign bits, i.e. binary quantization, where '+1' / '-1' are represented as '0' / '1' respectively to realize the multiplication by a simple exclusive OR gate. After finishing the bit-serial addition, the output is converted into linear number to process the next accumulation in high precision. This is because the addition in logarithmic domain is complicated in digital processing [7]. This process is conducted by a simple one-hot decoder because the numbers are quantized into a power of two. Finally, the output is accumulated until REG in shift register becomes acceptable.

Note that the baseline architecture has no module to convert linear numbers into logarithmic numbers at the input layer. Thus, the input layer is preprocessed and converted into the logarithmic representation by CPU. The converted data are then fed into A_RAM and W_RAM in the core.

TABLE 1. BINARY UNIT OUTPUTS

Adder_0 output	Adder_1 output	Binary Unit output
0 (+1)	0 (+1)	+2
0 (+1)	1 (-1)	0
1 (-1)	0 (+1)	0
1 (-1)	1 (-1)	-2
logic (linear number)		linear number

3. Area and Energy Optimization by Shared Accumulator

In this section, we propose an advanced architecture based on the baseline architecture by sharing the accumulator in each PE.

In the baseline architecture, each PE has a serial adder in the logarithmic domain and a parallel accumulator in the linear domain. Therefore, the numbers of required clock cycles for an operation between the serial part and the parallel part are different. Figure 5 shows a timing chart of the baseline PE architecture when 2-bit serial inputs. The serial adder takes as many cycles as the bit width (herein two cycles) in contrast to the accumulator which takes only a cycle in parallel due to the conversion from serial logarithmic domain to parallel linear domain. Therefore, an idle time is occurred at the accumulation cycles. We aim to reduce the idle time of the accumulator without losing the function of the baseline architecture. We found the accumulator can be shared between neighbor PEs. The proposed PE architecture with a shared accumulator is shown in Fig. 6. Since the proposed PE shares accumulator with adjacent PE, two PEs in Fig. 4 are combined into a single PE with the shared accumulator. Here, the number of combined PEs is limited to two to support the same flexible bit precision as the one to four bit baseline architecture. The proposed architecture has a buffer at the both outputs of the serial adders to hold one output for waiting the other output's accumulation.

Figure 7 shows a timing chart of the proposed PE architecture when 2-bit serial inputs. The two serial adders process the 2-bit serial addition at the same time, and the

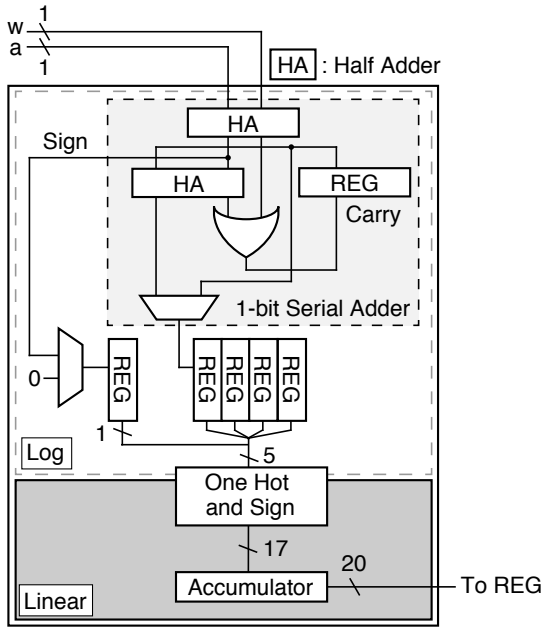

Figure 4. Baseline PE architecture.

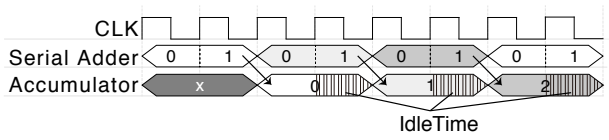

Figure 5. Timing chart of MAC in baseline PE architecture when 2-bit inputs.

two buffers hold them. The accumulator receives these buffer's data one by one. Therefore, the proposed PE can process sequentially without a collision of inputs. It is the same for three and four bits.

On the other hand, in the case for binary inputs, the two parallel inputs collide in the accumulator. To avoid this problem, we designed a preprocess unit named Binary Unit in Fig. 6. The unit is composed tiny circuit of multiplexers, which receives two sign bits from the serial adders, and calculates a summation result (Table 1).

Since the proposed PE architecture halved the number of PEs in PE_COL by shared accumulator, the length of shift register is also halved. As a result, our proposed PE architecture can reduce the area of PE array without losing the function of baseline architecture.

While the proposed method introduces two multiplexers between the shift register and the accumulator in the baseline PE architecture, it usually does not increase the critical path length. The baseline architecture has some longer paths outside the PEs for broadcasting the input activations. Since these additional multiplexers have little impact to the critical path, the proposed architecture can operate at the same clock frequency as the baseline architecture [8].

4. Evaluation

In this section, we evaluate our proposed architecture from the viewpoint of area and energy efficiency comparing with the baseline architecture of QUEST. Furthermore, we show the result of energy efficiency for a recent practical DNN in our proposed architecture.

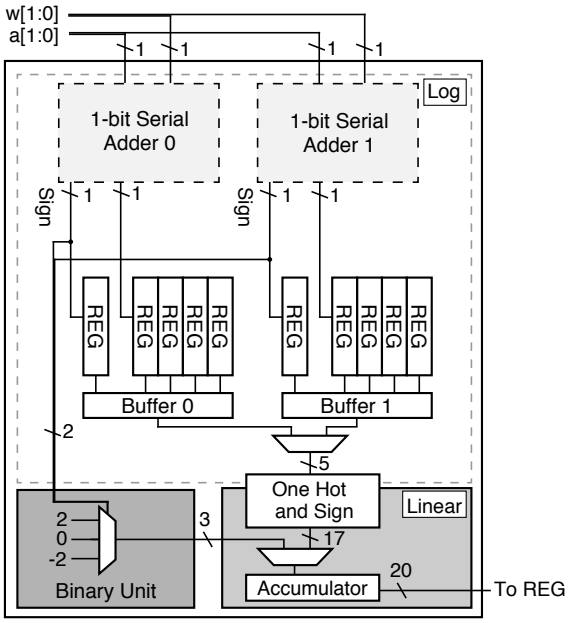

Figure 6. Proposed accumulator-sharing PE architecture.

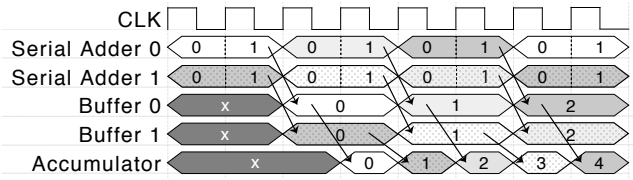

Figure 7. Timing chart of MAC in proposed PE architecture when 2-bit inputs.

4.1. Methodology

At first, we implemented an RTL design for both baseline and proposed architecture with Verilog HDL. The area and energy of these designs are estimated by Synopsys Design Compiler [11] in a 40-nm CMOS technology at 300-MHz clock frequency. Characteristics of SRAM components are evaluated by CACTI6.5 [12] for the fair comparison. Here, we evaluate only the core which includes PE array, ACT unit, sequencer and on-chip RAMs (A_RAM, W_RAM, B_RAM, O_RAM) because our improvements affect only these parts. Therefore, the area and power of interconnects, microcontrollers and 3D SRAMs are ignored. Finally, we selected a recent typical DNN models of VGG 16 [10] for ImageNet as a benchmark.

4.2. Area and Power Efficiency

Table 2 and Table 3 summarize the area and power of each module for both baseline and proposed architecture estimated by Synopsys Design Compiler. Since the proposed PE shares the accumulator between two baseline PEs, we show the result of the one proposed PE and the two baseline PEs for a fair comparison. Therefore, the size of the proposed PE array architecture composes 16×16 PEs, whereas the baseline is 32×16 PEs to normalize the number of MAC.

Figure 8 compares the area of logic parts of core which includes the PE array and sequencer. The proposed architecture achieved 29.8% reduction in core area from

TABLE 2. AREA AND POWER OF BASELINE ARCHITECTURE

	Area [$(mm)^2$]	Power [mW]
Sequencer	5.92×10^{-3}	1.09
PE_Array (PE_COL×16)	3.84×10^{-1}	71.5
PE_COL (PE×32 + ACT×1)	2.40×10^{-3}	4.47
$PE_{Baseline}$	7.27×10^{-4}	1.34×10^{-1}
ACT	7.12×10^{-4}	1.83×10^{-1}
Total	3.90×10^{-1}	72.6

TABLE 3. AREA AND POWER OF PROPOSED ARCHITECTURE

	Area [$(mm)^2$]	Power [mW]
Sequencer	5.92×10^{-3}	1.09
PE_Array (PE_COL×16)	2.68×10^{-1}	54.7
PE_COL (PE×16 + ACT×1)	1.67×10^{-3}	3.42
$PE_{Proposal}$	1.00×10^{-3}	2.02×10^{-1}
ACT	7.12×10^{-4}	1.83×10^{-1}
Total	2.74×10^{-1}	55.8

the baseline architecture. The graph also shows that almost all the logic parts of core are occupied by PE array, so that the area reduction of PEs are most effective to optimizing core area.

In addition to area optimization, Fig. 9 shows the power estimation per core of the baseline and proposed architecture from the results of Synopsys Design Compiler. From this result, the proposed architecture reduces power consumption by 23.1% from the baseline. The result comes from the reduction in the number of accumulators and the size of shift registers being halved.

From these evaluations, the proposed PE architecture sharing accumulator can efficiently reduce the area and power consumption without losing the function of the baseline architecture.

4.3. Energy Efficiency on a Practical DNN

In this subsection, we evaluate the energy efficiency of practical DNN application of 1000-class ImageNet classification [13] using VGG 16 [10]. This DNN model consists of both CONV and FC layers. We obtained the pre-trained DNN model from the Caffe Model Zoo [14], and all parameters are log-quantized. As a result, the recognition accuracy can maintain 66.32% for Top-1 and 87.02% for Top-5. Note that the evaluation results do not include computations of the input layer. We use an in-house cycle-level simulator to estimate the cycle consumption for each part of the baseline and proposed cores. The dataflow among the cores are same between the baseline and proposal because the proposed updates are inside of the core. Therefore, a throughput of processing is also same.

Combining the result of Section 4.2 and the cycle-level simulator, we evaluated the energy consumption of an

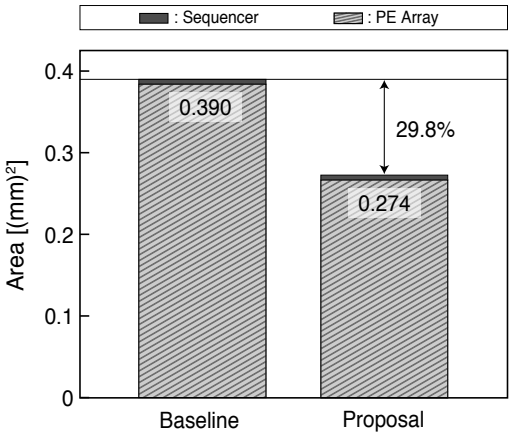

Figure 8. Area comparison between baseline and proposed architecture per core.

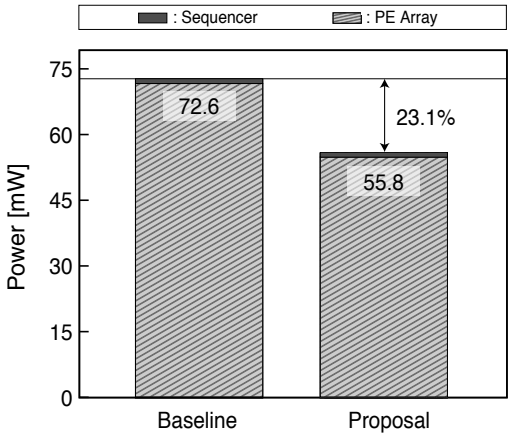

Figure 9. Power comparison between baseline and proposed architecture per core.

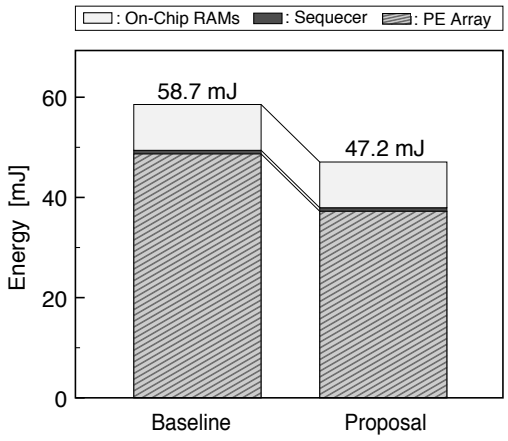

Figure 10. Energy comparison between baseline and proposed architecture at an inference in VGG 16 for ImageNet classification.

image classification by VGG 16. The result is shown in Figure 10. These results includes the energy consumption of on-chip RAMs, sequencer and PE array per core for VGG 16. As a result, the energy consumption of PE array that occupies a majority of total core are reduced by 11.5 mJ. This is 19.6% of the total energy consumption of each core. From these results, we showed our proposed PE architecture with shared accumulators enables not only the area optimization but also the energy consumption in a practical DNN application.

5. Related Work

Several previous studies performed NN accelerations on ASIC or FPGA [5], [15]. Given their high importance, there are several methods to reduce the amount of data necessary to compose DNN at an inference phase to reduce the cost of hardware acceleration. Deep compression [16] is a hardware-aware compaction method that prunes unnecessary weights and compresses the remaining weights by quantization and Huffman coding. It achieves almost same the accuracy as that of the original uncompressed model, and EIE [17] implemented this method on ASIC. SCNN [18] and Scalpel [19] also focus on exploiting the sparsity of pruned NNs.

Bit-precision is also an important factor in reducing the amount of data. Binary representation realizes the lowest cost because it replaces multipliers with a single XNOR logic gate [20], [21]. However, it decreases accuracy in complicated applications. Therefore, several fixed-point methods focused on obtaining a minimized bit-precision without accuracy reduction. A dynamical fixed-point controls the tradeoff between accuracy and bit-precision [5], [22]. These data representation was analyzed by [23]. They consider the relationship of hardware cost and bit precision using CIFAR-10 dataset, a 10-class classification task [24]. However, there is the most recent method that reduced bit-precision without multipliers corresponds to logarithmic representation [7], [25], [26] that can realize logarithmic representation for both of activation. and weights. We use this representation to reduce bit-width and analyzed using ImageNet dataset.

Stripes [27] proposed a bit-serial operation architecture based on DaDianNao [28]. It performed flexible bit-width operation by layer to realize minimum calculation. Bit-pragmatic accelerator updated it to skip the operation of zero-bit [29]. Their bit-serial operation is used for only activations.

In point of bit-serial processing, some works examined optimizations of bit-serial processing circuits for small footprint microprocessors [30]–[32]. Unlike these works, the proposed architecture employs not only bit-serial units but also bit-parallel units for accumulations. Thus, the proposal focuses on the redundant circuit of bit-parallel processing in order to reduce the circuit area.

6. Conclusion

The paper presented an area and energy optimized architecture based on the shared accumulators among multiple bit-serial inputs for the bit-serial log-quantized DNN accelerator. We focused on the idle time of the accumulators during the bit-serial addition instead of multiplications in the linear domain. The accumulators are only activated after each pack of bits are arrived and added. As the result, the accumulators are not well utilized. The proposed the shared accumulator architecture that shares the accumulators among the neighbor processing units. The evaluation results indicates the proposed architecture achieved 29.8% area reduction in the logic part of each core against the baseline architecture. Furthermore, it also achieved 19.6% energy reduction of each core for the practical DNN of VGG 16.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436 EP –, 05 2015.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385

[3] W. Lu, G. Yan, J. Li, S. Gong, Y. Han, and X. Li, "Flexflow: A flexible dataflow accelerator architecture for convolutional neural networks," in *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, Feb 2017, pp. 553–564.

[4] Y. H. Chen, J. Emer, and V. Sze, "Eyeriss: A spatial architecture for energy-efficient dataflow for convolutional neural networks," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, June 2016, pp. 367–379.

[5] J. Qiu, J. Wang, S. Yao, K. Guo, B. Li, E. Zhou, J. Yu, T. Tang, N. Xu, S. Song, Y. Wang, and H. Yang, "Going deeper with embedded fpga platform for convolutional neural network," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '16. New York, NY, USA: ACM, 2016, pp. 26–35. [Online]. Available: http://doi.acm.org/10.1145/2847263.2847265

[6] K. Ando, K. Ueyoshi, K. Orimo, H. Yonekawa, S. Sato, H. Nakahara, S. Takamaeda-Yamazaki, M. Ikebe, T. Asai, T. Kuroda, and M. Motomura, "Brein memory: A single-chip binary/ternary reconfigurable in-memory deep neural network accelerator achieving 1.4 tops at 0.6 w," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 4, pp. 983–994, April 2018.

[7] D. Miyashita, E. H. Lee, and B. Murmann, "Convolutional neural networks using logarithmic data representation," *CoRR*, vol. abs/1603.01025, 2016. [Online]. Available: http://arxiv.org/abs/1603.01025

[8] K. Ueyoshi, K. Ando, K. Hirose, S. Takamaeda-Yamazaki, J. Kadomoto, T. Miyata, M. Hamada, T. Kuroda, and M. Motomura, "Quest: A 7.49tops multi-purpose log-quantized dnn inference engine stacked on 96mb 3d sram using inductive-coupling technology in 40nm cmos," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, Feb 2018, pp. 216–218.

[9] D. Ditzel, T. Kuroda, and S. Lee, "Low-cost 3d chip stacking with thruchip wireless connections," in *2014 IEEE Hot Chips 26 Symposium (HCS)*, Aug 2014, pp. 1–37.

[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, ser. NIPS'12. USA: Curran Associates Inc., 2012, pp. 1097–1105. [Online]. Available: http://dl.acm.org/citation.cfm?id=2999134.2999257

[11] "Synopsys dc ultra," https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html.

[12] "Cacti 6.5," http://www.hpl.hp.com/research/cacti.

[13] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[14] "Caffe model zoo," http://caffe.berkeleyvision.org/model_zoo.html.

[15] Y. Umuroglu, N. J. Fraser, G. Gambardella, M. Blott, P. Leong, M. Jahre, and K. Vissers, "Finn: A framework for fast, scalable binarized neural network inference," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA '17. New York, NY, USA: ACM, 2017, pp. 65–74. [Online]. Available: http://doi.acm.org/10.1145/3020078.3021744

[16] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural network with pruning, trained quantization and huffman coding," *CoRR*, vol. abs/1510.00149, 2015. [Online]. Available: http://arxiv.org/abs/1510.00149

[17] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: Efficient inference engine on compressed deep neural network," in *Proceedings of the 43rd International Symposium on Computer Architecture*, ser. ISCA '16. Piscataway, NJ, USA: IEEE Press, 2016, pp. 243–254. [Online]. Available: https://doi.org/10.1109/ISCA.2016.30

[18] A. Parashar, M. Rhu, A. Mukkara, A. Puglielli, R. Venkatesan, B. Khailany, J. Emer, S. W. Keckler, and W. J. Dally, "Scnn: An accelerator for compressed-sparse convolutional neural networks," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ser. ISCA '17. New York, NY, USA: ACM, 2017, pp. 27–40. [Online]. Available: http://doi.acm.org/10.1145/3079856.3080254

[19] J. Yu, A. Lukefahr, D. Palframan, G. Dasika, R. Das, and S. Mahlke, "Scalpel: Customizing dnn pruning to the underlying hardware parallelism," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ser. ISCA '17. New York, NY, USA: ACM, 2017, pp. 548–560. [Online]. Available: http://doi.acm.org/10.1145/3079856.3080215

[20] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 4107–4115. [Online]. Available: http://papers.nips.cc/paper/6573-binarized-neural-networks.pdf

[21] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," *CoRR*, vol. abs/1603.05279, 2016. [Online]. Available: http://arxiv.org/abs/1603.05279

[22] B. Reagen, P. Whatmough, R. Adolf, S. Rama, H. Lee, S. K. Lee, J. M. Hernndez-Lobato, G. Y. Wei, and D. Brooks, "Minerva: Enabling low-power, highly-accurate deep neural network accelerators," in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, June 2016, pp. 267–278.

[23] S. Hashemi, N. Anthony, H. Tann, R. I. Bahar, and S. Reda, "Understanding the impact of precision quantization on the accuracy and energy of neural networks," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, March 2017, pp. 1474–1479.

[24] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)." [Online]. Available: http://www.cs.toronto.edu/ kriz/cifar.html

[25] E. H. Lee, D. Miyashita, E. Chai, B. Murmann, and S. S. Wong, "Lognet: Energy-efficient neural networks using logarithmic computation," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, March 2017, pp. 5900–5904.

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

[27] P. Judd, J. Albericio, T. Hetherington, T. M. Aamodt, and A. Moshovos, "Stripes: Bit-serial deep neural network computing," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Oct 2016, pp. 1–12.

[28] Y. Chen, T. Luo, S. Liu, S. Zhang, L. He, J. Wang, L. Li, T. Chen, Z. Xu, N. Sun, and O. Temam, "Dadiannao: A machine-learning supercomputer," in *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, Dec 2014, pp. 609–622.

[29] J. Albericio, A. Delmás, P. Judd, S. Sharify, G. O'Leary, R. Genov, and A. Moshovos, "Bit-pragmatic deep neural network computing," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO-50 '17. New York, NY, USA: ACM, 2017, pp. 382–394. [Online]. Available: http://doi.acm.org/10.1145/3123939.3123982

[30] J. Robinson, S. Vafaee, J. Scobbie, M. Ritche, and J. Rose, "The supersmall soft processor," in *Programmable Logic Conference (SPL), 2010 VI Southern*, March 2010, pp. 3–8.

[31] H. Nakatsuka, Y. Tanaka, T. V. Chu, S. Takamaeda-Yamazaki, and K. Kise, "Ultrasmall: The smallest MIPS soft processor," in *Field Programmable Logic and Applications (FPL), 2014 24th International Conference on*, Sept 2014, pp. 1–4.

[32] S. Takamaeda-Yamazaki, H. Nakatsuka, Y. Tanaka, and K. Kise, "Ultrasmall: A tiny soft processor architecture with multi-bit serial datapaths for fpgas," *IEICE Transactions on Information and Systems*, vol. E98.D, no. 12, pp. 2150–2158, 2015.