



HOKKAIDO UNIVERSITY

Title	Efficiency-Centric Hardware Accelerator for Deep Neural Network Inference
Author(s)	植吉, 晃大
Degree Grantor	北海道大学
Degree Name	博士(工学)
Dissertation Number	甲第14129号
Issue Date	2020-03-25
DOI	https://doi.org/10.14943/doctoral.k14129
Doc URL	https://hdl.handle.net/2115/80359
Type	doctoral thesis
File Information	Kodai_Ueyoshi.pdf



Efficiency-Centric Hardware Accelerator for Deep Neural Network Inference

Gradient School of Information Science and Technology

Hokkaido University



Kodai Ueyoshi

Submitted in partial fulfillment of the requirements of the Ph.D. Degree in
Engineering

February 2020

Efficiency-Centric Hardware Accelerator for Deep Neural Network Inference

Kodai Ueyoshi

Abstract

This study discusses the efficiency-centric hardware architecture for deep neural network (DNN) inference. DNN is a mathematical model inspired by the functionality of the cortex of the brain. Recently, DNN has been devoted growing attention in many fields of artificial intelligence technology, such as image or sound recognition and natural language processing. This is because DNN can achieve high performance and accuracy in the fields. A lot of data can be managed to train DNN because of improve of processor technology. GPU is the most usable devices for DNN training. Recent GPUs achieves highly parallel processing with low cost. Therefore, became to be able to train a lot of data on the real-world devices. On the other hand, trained DNN have to be run on the restricted devices in the real world. Therefore, designing high energy efficient hardware model is required for the embedded devices.

In this study, we explore an optimal approach in the aspect of both algorithm and architecture for highly efficient DNN hardware. Here, I analyze three points, compressed DNN, architecture exploration, and optimal NN model.

At first, we analyze a log-quantization and the benefit. Log-quantization is a multi-bit quantization method that utilizes a power-of-2 logarithmic format. The most important feature of logarithmic quantization (log-quantization) is that multiplier hardware is no longer required because all multiplications in the linear field are represented simply through additions in the logarithmic field. Therefore, LOGNET can potentially achieve a high level of energy efficiency. Another advantage of LOGNET is that the memory footprint and bandwidth requirements are much lower than with linear quantization, but with the same accuracy, because a log-quantization can represent the same numeric range using fewer bits. A key insight here is that most of the weight distributions generally form a Gaussian distribution, in which smaller values appear more frequently than larger values. Log-quantization can represent these types of non-uniform distributions with a lower amount of numerical errors as compared to linear quantization with the same bit width.

Secondly, we propose a novel DNN architecture called QUEST. QUEST is a programmable MIMD parallel accelerator for general-purpose state-of-the-art deep neural networks (DNNs). It features die-to-die stacking with three-cycle latency, 28.8 GB/s, 96 MB, and 8 SRAMs using an inductive coupling technology called the ThruChip Interface (TCI). By stacking the SRAMs instead of DRAMs, lower memory access latency and simpler hardware are expected. This facilitates in balancing the memory capacity, latency, and bandwidth, all of which are in demand by cutting-edge DNNs at a high level. QUEST also introduces log-quantized programmable bit-precision processing for achieving faster (larger) DNN computation (size) in a 3D module. It can sustain a higher recognition accuracy at a lower bit-width region compared to linear quantization. The prototype QUEST chip is integrated in the 40-nm CMOS technology, and it achieves 7.49 tera operations per second (TOPS) peak performance in binary precision, and 1.96 TOPS in 4-bit precision at 300-MHz clock.

Lastly, we propose prediction-based DNN model called Dead Neuron Prediction (DNP). In most DNN models, a large part of neurons finally results in zero (dead neuron) due to activation functions. Computations for such the dead neurons waste huge energy by unnecessary multiply-and-accumulate (MAC) operations. To skip unnecessary computations for dead neurons, we propose DNP to predict liveness of neurons in advance by employing a supportive lightweight neural network. By efficiently pipelining both computations of a main DNN and its prediction, computations for likely dead neurons are dynamically skipped. Experiment results indicate a DNN accelerator with DNP achieves a better energy efficiency than prior approaches at the same accuracy.

Acknowledgements

First and foremost, I would like to thank my advisor, Professor Masato Motomura. He has been my supervisor and has supported me for 6 years. He gave me extremely visionary advice and a lot of opportunities to spread my research around the world. I feel so fortunate to join his laboratory.

I would also thank my co-advisors, Professor Tetsuya Asai and Professor Shinya Takamaeda-Yamazaki. They gave me a great environment for research and insightful discussions.

My sincere thanks are due to the official referees, Professor Akihisa Tomita, Professor Seiya Kasai and Professor Masayuki Ikebe, for their detailed review, constructive criticism and excellent advice during the preparation of this thesis.

It has been an honor to work with many great collaborators outside Hokkaido University. I am deeply grateful to Professor Tadahiro Kuroda and Professor Mototsugu Hamada of Keio University provided me the TCI technology and hardware development technology with a lot of suggestions and discussions. I would like to thank Professor Alexandre Schmid of EPFL and Dr. Takao Marukame of Toshiba Corporation for collaborating research when I stayed EPFL as a visiting scholar. I would also thank to Professor Yutaka Matsuo of the University of Tokyo for insightful discussions for DNP project.

I would also thank my friends and lab mates in LALSIE group: Kasho Yamamoto, Miho Ushida, Dr. Eric-Shun Fukuda, Dr. Kazuyoshi Ishimura, Dr. Itaru Hida, Takanori Ohira, Masafumi Mori, Tsunaki Sadahisa, Toshiyuki Itou, Kentaro Orimo, Kota Ando, Ryota Uematsu, Teruaki Kumazawa, Takeshi Shimada, Kazutoshi Hirose, Prasoon Ambalathankandy, Takumi Kudo, Yuka Oba, Tatsuya

Acknowledgements

Kaneko, Taiga Ikeda, Yuki Hirayama, Shungo Kumazawa, Junnosuke Suzuki, and many others for supporting my research and life in lab.

This work was supported by JSPS Grant Number JP17J00403, JP19J30002, and JST ACT-I Grant Number JPMJPR18U8.

And I give sincere thanks to my parents. I have been supported my long student life mentally and financially by their kindness.

Finally, I would like to express my sincere thanks to Kana, my dear wife.

Contents

Acknowledgements	5
1 Introduction	1
1.1 Motivation	3
1.2 Contribution and Thesis Outline	6
2 Background	7
2.1 Basis of Deep Neural Networks	7
2.2 Network architecture	9
2.3 HW optimization method	12
2.4 HW accelerators	14
2.4.1 Roofline model based exploration	14
2.4.2 DNN dataflow architectures	17
3 HW-oriented Quantization	21
3.1 Introduction	21
3.2 Bit Reduction using Quantized Neural Networks	22
3.2.1 Effects of Log-quantized DNNs	22
3.3 Bit-Parallel Processing vs. Bit-Serial Processing	25
3.4 Related Works for Bit-Serial Processing	26

Contents

3.5	Conclusion	27
4	Log-quantized Neural Network Engine with Stacking Technology	29
4.1	Introduction	29
4.2	Quantized Neural Network Engine with Stacking Technology (QUEST)	31
4.2.1	3D-stacking with Inductive Coupling Technology	32
4.2.2	Short and Uniformed SRAM Access	33
4.3	System Architecture	36
4.3.1	QUEST Core Architecture	36
4.3.2	PE Array Architecture	38
4.3.3	Mapping Convolution and Fully Connected Layers on Unified Architecture	42
4.3.4	PE microarchitecture	43
4.3.5	Programmable Controller	43
4.3.6	Mapping CONV and FC Layers on Unified PE Array	46
4.4	Results and Implementation	47
4.4.1	Setups	47
4.4.2	PE Array Size	48
4.4.3	On-chip Memory Capacity	49
4.4.4	Effect of SRAM Short Latency	50
4.4.5	Performance and Accuracy	53
4.4.6	Cluster Size	53
4.4.7	Power Breakdown Based on the Measurement Results	54
4.4.8	Implementation and Comparison with State-of-the-art General Purpose DNN Accelerators	56
4.5	Conclusion	57
5	Latency Evaluation for 3D integrated Memory	59
5.1	Introduction	59
5.2	Impact of memory access latency	61
5.2.1	Memory Access Pattern	62
5.2.2	Baseline Architecture	63
5.2.3	Effects of Memory Access Latency	64

5.3	Evaluation	65
5.3.1	Methodology	65
5.3.2	Bandwidth of 3D Stacking Memory	67
5.3.3	Performance and energy consumption	68
5.3.4	Hardware implementation	70
5.4	Related Works	72
5.5	Conclusion	72
6	Energy Efficient Inference by Skipping Inefficient Calculation	73
6.1	Introduction	73
6.2	Dead Neuron Prediction	76
6.2.1	How do dead neurons appear?	76
6.2.2	Concept of Dead Neuron Prediction	77
6.2.3	Related Works for Dead Neuron Prediction	78
6.2.4	Predicting Dead Neurons by Binarized NN	79
6.2.5	Optimizing a Tradeoff between Skipping Amount and Accuracy	81
6.3	Accelerator Architecture	82
6.3.1	Output sparse CNN	82
6.3.2	Proposed DNP-based Accelerator Architecture	82
6.4	Experimental results	84
6.4.1	Setups	84
6.4.2	Computation Reduction and Accuracy	85
6.4.3	Energy Consumption	87
6.5	Conclusion	88
7	Conclusion	89
	Bibliography	93
	List of Publications	101

List of Figures

1.1	The mechanism of training and inference for Deep Neural Network, and its demand on the embedded devices.	3
1.2	Algorithm-hardware co-design for optimized DNN accelerator.	4
2.1	Typical DNN architecture.	8
2.2	LeNet architecture [1].	10
2.3	AlexNet architecture [2].	10
2.4	VGG architecture [3].	11
2.5	GoogleNet architecture [4].	11
2.6	ResNet architecture [5].	12
2.7	SENet architecture [6].	12
2.8	Operation and memory access energy [7].	13
2.9	Roofline model.	15
2.10	Characteristics between CONV and FC layers.	15
2.11	Partitioned CONV layer.	16
2.12	Several stationary of CNN Accelerators.	18
3.1	Weight distribution with log-quantization.	23

List of Figures

3.2	Bit width vs. ImageNet classification accuracy of log-quantization and linear quantization on AlexNet and VGG.	24
3.3	MAC energy vs. ImageNet classification accuracy of logarithmic and linear quantizations on AlexNet and VGG16 networks.	25
3.4	(a) Area and (b) energy of MAC: 4-bit log-serial, 4-bit log-parallel, and 8-bit integer-parallel.	26
4.1	Overall design of QUEST.	31
4.2	ThruChip Interconnect (TCI).	32
4.3	Detail layout of TCI channels on QUEST cores and SRAM vaults.	33
4.4	Broadcast access with fixed latency by TCI.	34
4.5	TCI wireless communication circuit diagram.	34
4.6	Time chart of single read access of 3D SRAMs through the TCI.	35
4.7	Interconnect among QUEST cores.	36
4.8	Detailed circuit diagram of core architecture.	37
4.9	Bit-serial PE mechanism.	37
4.10	Detailed computation in PE_COL.	39
4.11	Pipelined shift dataflow in PE_COL for dot-product computations.	40
4.12	Bit-serial memory access.	40
4.13	Types of Neural Networks.	41
4.14	Pseudocode of CONV/FC operation in the homogeneous PE array.	41
4.15	FC/CONV dataflow on a homogeneous PE array.	42
4.16	PE and ACT microarchitecture.	44
4.17	Controller architecture.	45
4.18	Memory access pattern of activation RAM in the CONV layer.	46
4.19	Clustering cores and data movement in the inter-cluster and intra-cluster.	47
4.20	Test chip and setups.	49
4.21	Evaluation results of PE array configuration (x : # of PE_COL, y : # of PE in PE_COL).	50
4.22	Evaluation results of a capacity of on-chip memory.	50
4.23	Performance comparison of SRAM and DRAM latency in different types of DNNs.	51
4.24	Performance comparison of SRAM and DRAM latency in VGG for CIFAR10 with different bit-widths.	51

4.25	Time-chart of CONV5 layer operation in VGG at log 4 bit.	52
4.26	Accuracy and performance results of different types of DNNs.	53
4.27	Performance and required memory capacity when scaling cluster size in all CONV layers in AlexNet.	54
4.28	Power breakdown of QUEST chip and its performance result in AlexNet at 300 MHz and 1.1 V when scaling the number of used core.	55
4.29	Chip microphotograph and layout overview.	55
5.1	Capacity comparisons of 3D SRAM using CACTI 6.5 [8].	61
5.2	Two data categories (activations and weights/biases) in DNN processing: Their access patterns and the influence of memory access latency.	62
5.3	Baseline architecture.	64
5.4	Latency evaluation.	65
5.5	Performance comparison of each layer in AlexNet and VGG16 with respect to different memory bandwidths of 3D SRAMs.	67
5.6	Energy consumption (left-axis, top row), execution time (right-axis, top row), and energy efficiency (bottom row) with different sized PE arrays and on-chip memory per core.	68
5.7	Microphotograph, diagram of TCI interface, and specifications of the fabricated chip.	71
6.1	Prediction-based DNN model.	74
6.2	Dead Neuron Prediction by SW/HW co-design.	75
6.3	Dead neuron rate [9].	76
6.4	Dead neurons produced by ReLU.	77
6.5	Concept of Dead neuron prediction.	77
6.6	HW-efficiency comparison between between the early-termination [9, 10] and the neuron prediction.	78
6.7	Dead neuron prediction mechanism.	79
6.8	DNP training by layer.	80
6.9	Threshold adjustment.	81
6.10	Overview of DNP-based accelerator.	83
6.11	Recall of DNP in each layer.	86

List of Figures

6.12	Classification accuracy and the number of MAC operation.	86
6.13	Trade-off between classification accuracy and MAC reduction by threshold tuning for CifarNet.	87
6.14	Energy comparison for no compressed processing, SCNN, and our DNP approach. .	87
7.1	Summary of the thesis.	91

List of Tables

4.1	Benchmarks.	48
4.2	Comparison with state-of-the-art general purpose DNN accelerators.	56
4.3	Energy comparison between SRAM and DRAM.	57
5.1	Latency and energy comparison between SRAM and DRAM.	65
5.2	Benchmarks.	66
5.3	Hardware configuration.	66
5.4	Comparison of performance and energy efficiency when using a GPU (AlexNet) [11, 12].	69
5.5	Performance and energy efficiency comparison with state-of-the-art ASICs (AlexNet).	70
6.1	The network structure of M-CifarNet for CIFAR10 classification.	84
6.2	Energy setup table.	85

CHAPTER *1*

Introduction

Deep learning (DL) algorithms have been devoted a growing attention, owing to impressive performances recently demonstrated in acceptedly difficult applications, *e.g.*, classification and prediction, exceeding the performances of conventional machine learning algorithms [13]. Above all, deep neural networks (DNNs) show extremely high performance in machine learning algorithms, and are being rapidly developed as practically useful technologies. DNNs are built using mathematical models that mimic the cranial nerves (neurons) of living organisms.

These dramatic improvements are reported daily, causing the third boom in neural networks. In the near future, larger systems are expected to cause change around the world. For example, enhancing service automation in large data centers, predicting drug action, reorganizing brain circuits, and the effects of sudden changes in DNA on genes and diseases.

Then, why is DNN technology so advanced rapidly? The background of this development is the following three points.

- (1) Training data can be easily acquired.

Chapter 1. Introduction

(2) Development of a high-performance computing environment that performs computations in real time.

(3) Growing demand as applied technology in the era of big data.

With the advent of the age of big data in recent years, our world has been formed in which various types of data are valued and used. As a result, the expansion of data centers and other facilities has expanded, and cloud computing has developed (1). This is because of the improvement of integrated circuit technology. As typified by Moore's Law, computing devices have improved their performance twice as fast each three years, and large-scale devices such as graphics processing unit (GPU) can be performed DNN computing in real time (2). Then, algorithms have evolved using this technology, and the results have been published one after another, have gained public demands through industry and academia (3).

The interaction between the social trend and the hardware technology created this innovation. However, in recent years, Moore's law has come to an end in integrated circuit technology, and the mere improvement has entered the post-moore era, where performance cannot be improved. In such a situation, it is difficult to raise the level of this technology evolution simply with the power of integrated circuit technology. Therefore, system-level hardware innovation is needed. In other words, it is necessary to build a seamless hardware model that is aware of the strong connection between the application and the hardware.

So what kind of hardware system does DNN require? DNN has very high performance in prediction and classification, but it is a result of huge training before inference execution, and it takes long calculation time and large power consumption to perform sufficient training. In order for data scientists to perform this training within practical cost and time, it is necessary to build a highly efficient (= performance / energy) hardware system. In a typical CPU-based architecture, the speed is limited by the clock frequency. Therefore, in recent years, approaches have been taken to accelerate calculations on GPUs. One of the technical problems of GPU computing is managing efficient memory access. Data transfer rate among multi-GPUs is bottle neck for the huge training tasks. Even for a single GPU, related data transfer must be controlled carefully between local and shared memory. For this reason, GPU-based computing requires careful management of memory coherence, and new memory and communication technologies are expected to be used for further improvements.

On the other hand, application specific integrated circuit (ASIC) approaches are expected in terms of both high performance and energy efficiency. The ASIC implementation lacks flexibility and

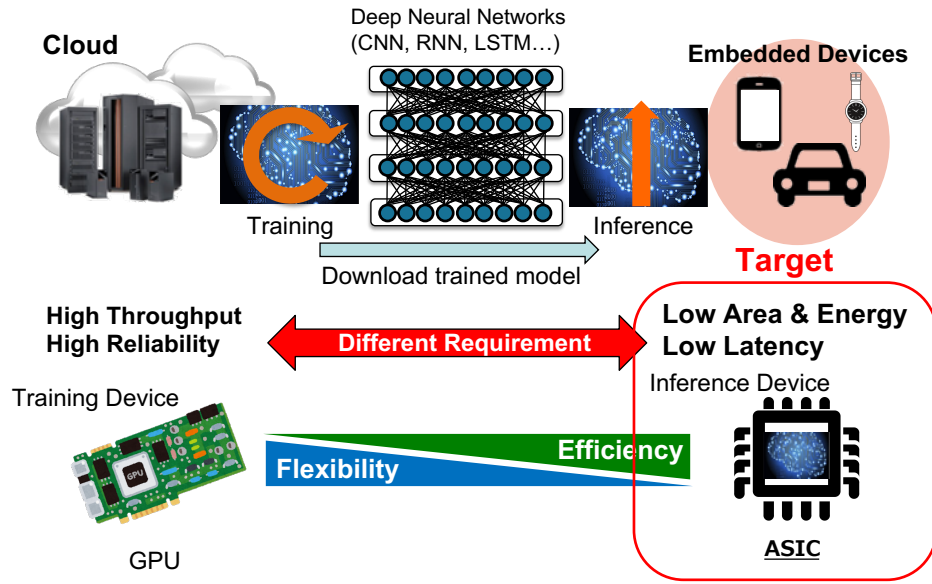


Figure 1.1: The mechanism of training and inference for Deep Neural Network, and its demand on the embedded devices.

versatility. However, DNN calculations are simple repetitions of multiply-accumulate operations. Therefore, ASICs are suitable for inferences where less flexible computing is required. A Field Programmable Gate Array (FPGA) has lower performance than ASICs, but because it can quickly prototype new algorithms, it can flexibly respond to rapidly changing DNN algorithms and different models.

As described above, today's DNN computing is being carried out on each of the CPU, GPU, FPGA, and ASIC, taking advantage of each feature. What can be said consistently is that we have adopted a digital hardware implementation in the current technology and an architecture aimed at improving the efficiency of memory management based on it.

1.1 Motivation

DL algorithms require significant computational time and dissipate large amounts of power in order to reach a state where the data structure is optimized with a set of parameters that are suitable to obtain intelligent operations. Consequently, the execution of training algorithms is considered excessively slow, specifically in the case of large-scale networks. Iterative updates of connection weights of the network use over the major part of the training time.

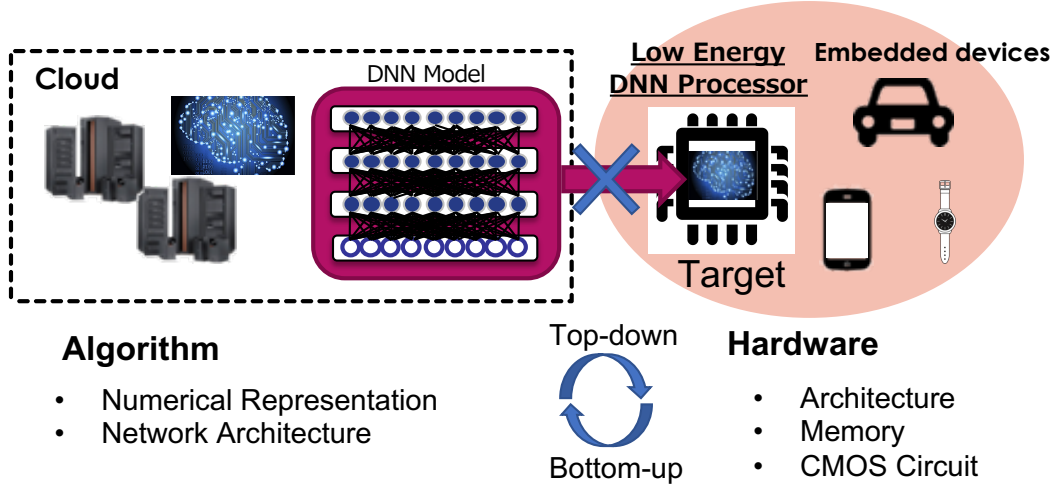


Figure 1.2: Algorithm-hardware co-design for optimized DNN accelerator.

DL hardware systems can be broadly divided into two types as shown in Fig. 1.1. One is high-performance training processing at cloud. The other is a power-saving embedded system using edge computing. This is because the current system collects data from sensor devices that have blended into the real environment and performs learning processing on the cloud. In this training process, features are extracted so that a specific output is given to similar data, and the process is reflected in the structural data of the DNN. In the training phase, high throughput and reliability are required. Therefore, in practical case, DNN training has been performed by cloud servers or high-performance GPU computing. Hence, efficient and reliable hardware systems are required to satisfy the growing requires for DL in the field of data analysis, autonomous system control and cognitive applications. In this area, large companies with data centers are taking the initiative to develop power-efficient and highly reliable hardware.

On the other hand, embedded devices perform inference processing such as recognition and prediction in a real-world environment. The inference is performed for single data using downloaded trained data. This operation is required low are, low energy, and low latency. Therefore, ASIC is suitable for designing high efficient design. The target of this study is the high efficient ASIC design for DNN inference.

In this study, we explore an optimal approach in the aspect of both algorithm and architecture for highly efficient DNN hardware as shown in Fig. 1.2. In previous approaches, hardware engineers worked hard to implement DNN models compressed by machine-learning professionals on realistic

devices (top-down). However, in this study, we proposed more efficient solutions by constructing a DNN model considering hardware side in advance (bottom-up). Therefore, we have optimized the trade-off between computing efficiency and DNN model accuracy combining top-down and bottom-up approaches, instead of the traditional hardware design that considered only computing efficiency which includes the performance and energy.

Here, we analyze three points, compressed DNN, architecture exploration, and optimal NN model.

At first, we analyze a log-quantization and the benefit. Log-quantization is a multi-bit quantization method that utilizes a power-of-2 logarithmic format. The most important feature of logarithmic quantization (log-quantization) is that multiplier hardware is no longer required because all multiplications in the linear field are represented simply through additions in the logarithmic field. Therefore, LOGNET can potentially achieve a high level of energy efficiency. Another advantage of LOGNET is that the memory footprint and bandwidth requirements are much lower than with linear quantization, but with the same accuracy, because a log-quantization can represent the same numeric range using fewer bits. A key insight here is that most of the weight distributions generally form a Gaussian distribution, in which smaller values appear more frequently than larger values. Log-quantization can represent these types of non-uniform distributions with a lower amount of numerical errors as compared to linear quantization with the same bit width.

Secondly, we propose a novel DNN architecture called QUEST. QUEST is a programmable MIMD parallel accelerator for general-purpose state-of-the-art deep neural networks (DNNs). It features die-to-die stacking with three-cycle latency, 28.8 GB/s, 96 MB, and 8 SRAMs using an inductive coupling technology called the ThruChip Interface (TCI). By stacking the SRAMs instead of DRAMs, lower memory access latency and simpler hardware are expected. This facilitates in balancing the memory capacity, latency, and bandwidth, all of which are in demand by cutting-edge DNNs at a high level. QUEST also introduces log-quantized programmable bit-precision processing for achieving faster (larger) DNN computation (size) in a 3D module. It can sustain a higher recognition accuracy at a lower bit-width region compared to linear quantization. The prototype QUEST chip is integrated in the 40-nm CMOS technology, and it achieves 7.49 tera operations per second (TOPS) peak performance in binary precision, and 1.96 TOPS in 4-bit precision at 300-MHz clock.

Lastly, we propose prediction-based DNN model called Dead Neuron Prediction (DNP). In most DNN models, a large part of neurons finally results in zero (dead neuron) due to activation functions. Computations for such the dead neurons waste huge energy by unnecessary multiply-and-accumulate

Chapter 1. Introduction

(MAC) operations. To skip unnecessary computations for dead neurons, we propose DNP to predict liveness of neurons in advance by employing a supportive lightweight neural network. By efficiently pipelining both computations of a main DNN and its prediction, computations for likely dead neurons are dynamically skipped. Experiment results indicate a DNN accelerator with DNP achieves a better energy efficiency than prior approaches at the same accuracy.

1.2 Contribution and Thesis Outline

In this study, we optimize the efficiency-centric hardware accelerator for DNN with a co-design approach between algorithm and hardware. This thesis proposes the architecture, chip design, and reduced calculation model for DNN inference. The contributions of the thesis are:

- To implement large DNN models into low-power devices, we evaluate the quantization method of different numerical representations considering both the hardware computation performance and recognition accuracy.
- Based on the above discussion, we propose a novel DNN architecture by employing SRAM-based 3D-stacked memory and log-quantization for high energy efficiency and a low memory-capacity demand.
- To reduce the number of operation in large DNN models, we propose a prediction-based DNN model which can skip the zero-output neurons.

Chapter 2 provides the back ground for the DNN and hardware platform. We also survey the related works in CNN HW accelerators.

Chapter 3 describes the quantization method and its micro architecture. A logarithmic quantization method is suitable for the digital domain hardware architecture. Also, we analyze serial processing for flexible bit precision for both activations and weights in DNN.

Chapter 4 explains the proposed architecture for the logarithmic quantized DNN with 3D stacking technology. The proposed architecture actually implemented on the chip with 40-nm technology.

Chapter 5 presents the evaluation of the 3d integrated memory using SRAM and DRAM. The result shows the proposed 3D SRAM is suitable for a low latency middle-class capacity memory.

Chapter 6 provides prediction-based DNN inference model. The proposed method called dead neuron prediction reduces the operation of negligible neurons.

Finally, in Chapter 7 we summarize our work and discuss what should be done in the future.

CHAPTER 2

Background

In this chapter, we give a fundamental background of deep neural Network(DNN) and the hardware accelerator with related works. DNN has recently been gaining attention because of its impressive performance in various applications such as image recognition, machine vision, natural language processing, and speech recognition. However, the performance requirements of practical applications cannot be achieved with a conventional CPU. Therefore, several accelerators based on GPUs, FPGAs and ASICs have been applied to obtain effective performance. We also the hardware design methods for DNN with previous studies.

2.1 Basis of Deep Neural Networks

DNNs are artificial neural network models composed of multiple layers. Modern DNNs primarily have two types of layers, i.e., a convolutional (CONV) layer and a fully connected (FC) layer. As shown in Fig. 2.1, the CONV layer receives N input feature maps. Further, the $K \times K$ window is chosen from the input feature maps by the sliding stride (S). It is convoluted by the $K \times K$ kernels

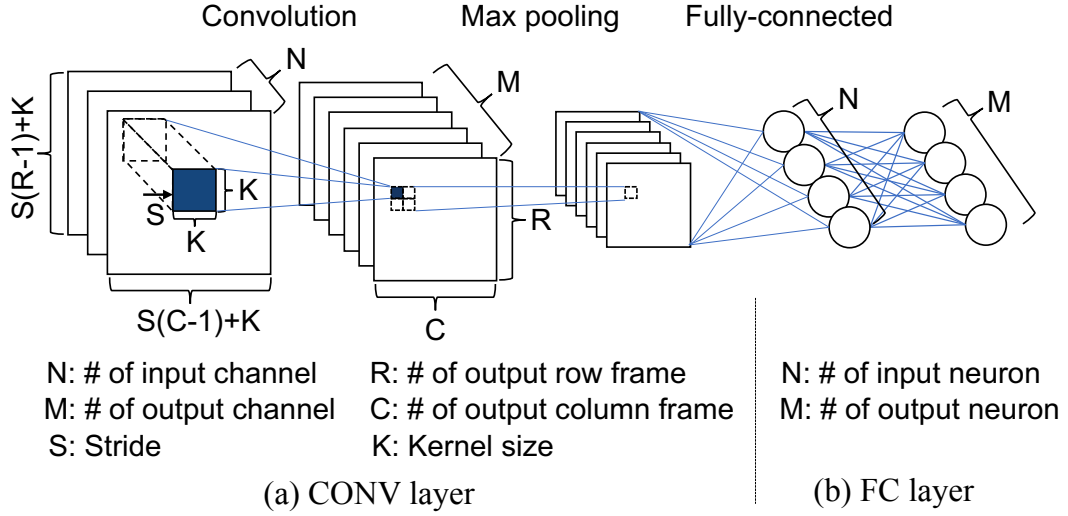


Figure 2.1: Typical DNN architecture.

to obtain one pixel in the output feature maps (M) as the next input feature map. The last convolution layer's feature maps are vectorized to adapt to the FC layer. Finally, the classifications are applied in the last output layer.

The CONV and FC operations are represented as the following formulas, respectively.

$$O_{CONV}[m][r][c] = f\left(\sum_{n=0}^{N-1} \sum_{i=0}^{K-1} \sum_{j=0}^{K-1} A[n][Sc+i][Sr+j] \times W[m][n][i][j] + b[m]\right) \quad (2.1)$$

$$O_{FC}[m] = f\left(\sum_{n=0}^{N-1} A[n] \times W[m][n] + b[m]\right) \quad (2.2)$$

where A is the input activation, W is the weight, S is stride, $0 \leq m < M$, $0 \leq c < C$, $0 \leq r < R$, and f is an activation function i.e. ReLU, binarization, and identity function. For simplicity, we denote the pre-activated multiply-add as $z = \sum A \times W + b$ herein.

CNN is a mathematical model inspired by the functionality of the visual cortex of the brain – a part of the cerebral cortex that plays an important role in processing visual information. CNN has been gaining great attention in the field of computer vision because of its impressive performance.

CNN is a multi-layered neural network with an input layer, multiple hidden layers, and an output layer. Within CNN, the hidden layers are referred to as convolutional layers that extract the input feature as feature maps. A subsampling layer is also included in the CNN hidden layers. However, convolutional operations occupy 90% of the computation time in the feed-forward process. Firstly,

the convolutional layer receives N input feature maps. Next, the $K \times K$ window is picked out from the input feature maps by sliding stride (S). It is convoluted by the $K \times K$ kernels to obtain the one pixel in the output feature map as the next input feature map. The acquired feature maps represent the previous input features, and the feature maps in the deeper layer represent the higher-dimensional features. The last convolutional layer's feature maps are vectorized to adapt to the fully connected neural network. Finally, the classifications are applied in the last output layer.

CNN employs a feed-forward process as an inference, and a backward process as training, similar to a classical multi-layer neural network. In practical application, the training is processed offline using high performance computing, and the inference is processed online to realize real-time processing. In this study, we treated the feed-forward process for inference processing.

Recent DNNs use batch normalization [14] that normalizes the distribution of the activation in each layer statistically for accelerating the training and achieving high-recognition accuracy, even at a reduced bit precision. Batch normalization is defined by the following formula:

$$\hat{z} = \gamma \left(\frac{z - \mu}{\sigma} \right) + \beta \quad (2.3)$$

where μ and σ^2 are the mean and variance of z over all the input images (or vectors) in the training set, respectively, and γ and β are the trainable scaling and offset factors, respectively.

Further, μ , σ^2 , γ , and β are constants in the inference phase. Therefore, the equation above can be modified to

$$\begin{aligned} \hat{z} &= \frac{\gamma}{\sigma} (\sum A \times W) + \left\{ \beta + \frac{\gamma(b - \mu)}{\sigma} \right\} \\ &= \alpha (\sum A \times W + B) \end{aligned} \quad (2.4)$$

where $\alpha = \frac{\gamma}{\sigma}$ and $B = \frac{\sigma\beta}{\gamma} + b - \mu$. This factorization enables the complicated normalization to be replaced with simple multiply-add operations in the inference phase.

2.2 Network architecture

In this section, we introduce the various type of DNN model architecture. The minimum component of DNN is a neuron as mentioned before section. The neurons are grouped in layered modules. Inter-layer of modules are connected by a weight which is corresponding to synapse in neuroscience. DNN is a multi-layered neural network with an input layer, multiple hidden layers, and an output layer. Different connections and arrangements produce the various type of models. Multi-layer perceptron

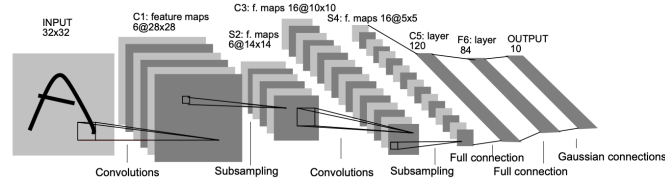


Figure 2.2: *LeNet architecture [1].*

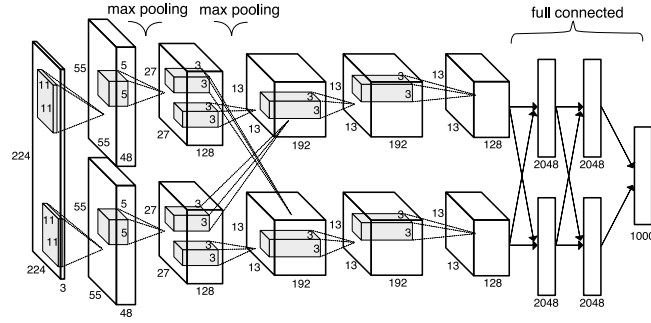


Figure 2.3: *AlexNet architecture [2].*

(MLP) is constructed by multi FC layers. On the other hand, many type of recent DNN is composed of a CNN. CNN has been gaining great attention in the field of computer vision because of its impressive performance. Within CNN, the hidden layers are referred to as convolutional layers that extract the input feature as feature maps. This recognition accuracy has produced very high results at the prestigious image recognition contest ILSVRC2012 [2]. Since then, research has been actively pursued and commercialization has been promoted. However, the scale of the network is increasing year by year, and in order to use this under practical constraints, a dedicated accelerator having excellent area and power efficiency is required.

Figure 2.3 shows Alex-net, which is one of the most famous models of CNN, winning the ImageNet 2012 contest [2]. It is composed of eight layers. The first five layers are convolutional layers, and the next six to eight layers are fully-connected layers. The input images are 224×224 three-channel RGB images. The output of the first layer is partitioned into two sets; the inputs of the third layer and the sixth layer are applied between the sets. The last layer that is vectorized to 1000 elements can classify 1000 categories from the image dataset.

Figure 2.4 illustrates VGG architecture which won the ILSVRC competition in 2014 [3]. VGG forms a deeper and monotonous network and has high versatility. Although there are few model parameters, it is easy to implement, but VGG16 has as many as 138 million parameters, so the calcu-

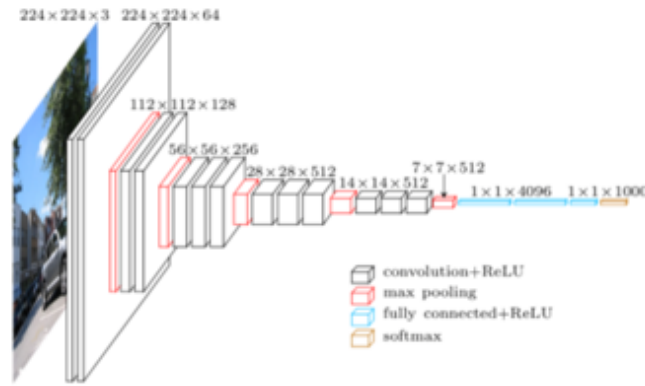


Figure 2.4: VGG architecture [3].

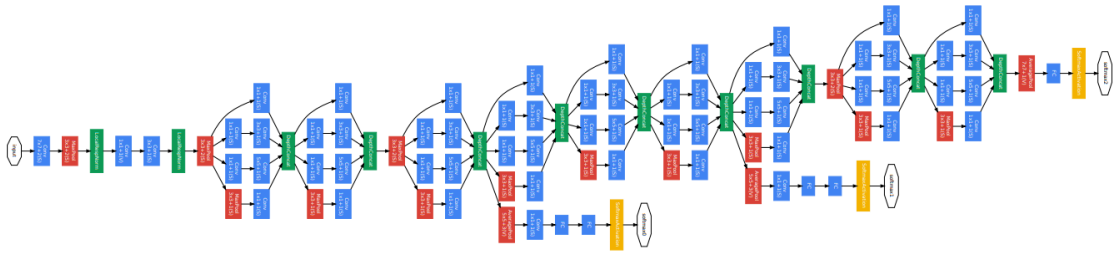


Figure 2.5: GoogleNet architecture [4].

lation cost is very high. On the other hand, GoogLeNet [4] has a configuration called inception layer that has CONV layers having different channels in parallel in order to reduce the number of parameters as shown in Fig. 2.5. The different channels allow a high amount of information to be reproduced in one layer.

Figure 2.6 shows ResNet architecture [5]. ResNet won the ILSVRC competition in 2015. The concept is based on the hypothesis that deeper networks should have better recognition accuracy. Add layers to a shallow network to form a deeper network. At this time, by joining the layers that skip the added layer and learning the residual of the result before and after adding the layer, the recognition accuracy is improved at least compared to the shallow layer. is there.

Figure 2.7 shows Squeeze-and-excitation Network (SENet) [6]. This model won the 2017 ILSVRC.

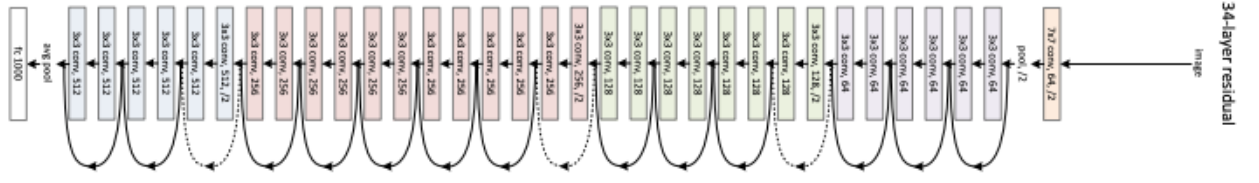


Figure 2.6: ResNet architecture [5].

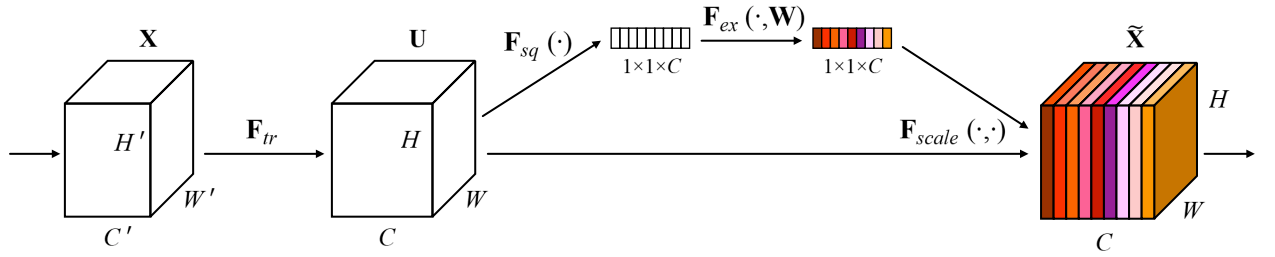


Figure 2.7: SENet architecture [6].

In the above-mentioned ResNet skip joint, the average of the plane of the CNN is taken and squeeze in a one-dimensional direction as a representative value of the channel. After that, the channel is weighted (excitation) by passing through all tie layers that perform dimensional compression. Finally, scale each channel to the original output. As a result, it becomes possible to select a feature of which channel is important.

2.3 HW optimization method

In this section, we introduce techniques for hardware optimization from algorithm side.

Table 2.8 compares different type of operations and memory access. In the operations, it can be seen that the energy consumption differs not only among the different bit widths, but also between different data formats. The key points of energy reduction are the numerical representation and the amount of operations. There are several works called quantization for the former and pruning for the latter.

The numerical representation in computing is crucial to realize efficient calculation. There are several type of representations are determined by IEEE standard, such as integer, floating point. Though the DNN has a robustness against the luck of data, there are trade-off between the accuracy and the bit reduction. The quantization can reduce the bit-width of values in the calculation of DNN. A binary

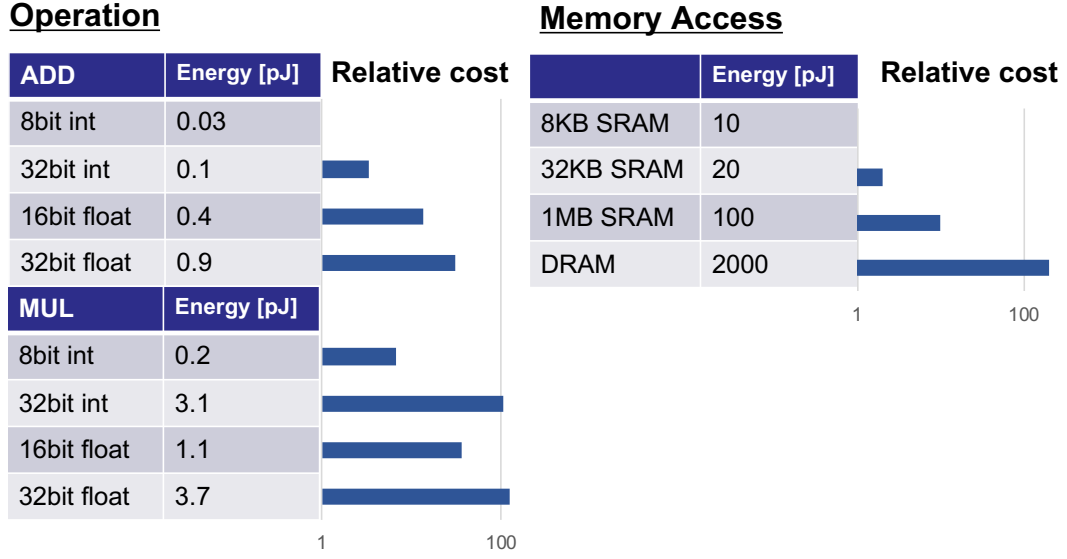


Figure 2.8: Operation and memory access energy [7].

representation realizes the lowest cost [15, 16]. However, it drastically decreases the accuracy, particularly in complicated applications. A dynamic fixed point controls the tradeoff between accuracy and bit precision [17, 18]. Another approach to reduce the hardware cost with acceptable accuracy is logarithmic quantization [19–21]. In chapter 3, we evaluate the energy efficiency for the several type of data representation and bit width.

Given its high importance, several methods have been developed to reduce the amount of data necessary to compose a DNN at an inference phase. Deep compression [22] is a hardware-aware compaction method that prunes and compresses the unnecessary and remaining weights, respectively. It achieves almost same the accuracy as that of the original uncompressed model, and EIE [23] implemented the method on ASIC. SCNN [24] and Scalpel [25] also focused on exploiting the sparsity of weights, and the activations in pruned NNs. The efficiency of these approaches can be improved using SRAM-based 3D memory because the memory access latency issues are exposed in such sparse data structures

In memory access, there is a trade-off between energy and capacity. However, there is a large gap between them, which is caused by device technology. Dynamic random access memory (DRAM) is suitable for high integration, but its energy efficiency is low. On the other hand, static random access memory (SRAM) is energy efficient, but not capacity efficient. In this research, in order to fill this gap, we applied 3D lamination technology to fill this gap in chapter 4.

Chapter 2. Background

There are several type of quantization. The standard is the linear quantization. In the linear quantization, the values can be represented by integer with short bitwidth.

2.4 HW accelerators

In recent years, demand for hardware systems for deep learning has been increasing. This is because in-vehicle applications of intelligent computing technology and the spread of IoT devices require embedded systems with high area and power efficiency. In particular, an accelerator for speeding up classification using a learned convolutional neural network is expected.

CNN is a neural network with a convolutional layer, and enables supervised learning from the result of class classification using forward propagation and backpropagation methods like ordinary neural networks. This recognition accuracy has produced very high results at the prestigious image recognition contest ILSVRC2012. Since then, research has been actively pursued and commercialization has been promoted. However, the network scale is increasing year by year, and in order to use this under practical constraints, a dedicated accelerator having excellent area and power efficiency is required.

2.4.1 Roofline model based exploration

The roofline model demonstrates a computing performance, a memory performance, and a computation to communication (CtC) ratio in a two-dimensional graph shown in Fig. 2.9. The CtC ratio denotes the operations ratio per external memory accessed, that is, how often the read data from external memory is utilized in an internal operation. For example, if the read data can be shared more often, the CtC ratio increases. The attainable performance depends on the memory bandwidth before reaching point A in Fig. 3. At that point, it converges to the computational roof (CR). The attainable performance is given by

$$Attainable\ Perf. = \min \begin{cases} Computational\ Roof \\ CtC\ ratio \times Bandwidth \end{cases} \quad (2.5)$$

The inclination from the origin point to point A denotes I/O bandwidth. Point A is obtained by the CR and CtC ratio, which is determined by a circuit structure.

Here, Fig. 2.10 shows the number of parameters and the number of operations of AlexNet and VGG16. In both cases, the numbers are reversed in CONV and FC. In CONV layer, the number of

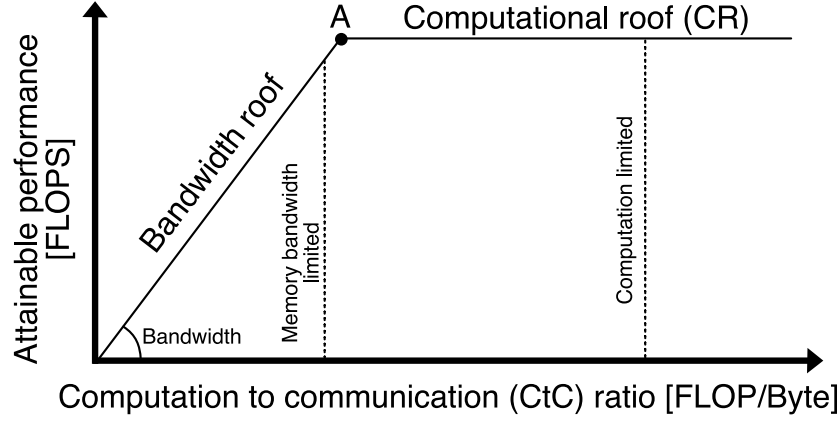


Figure 2.9: Roofline model.

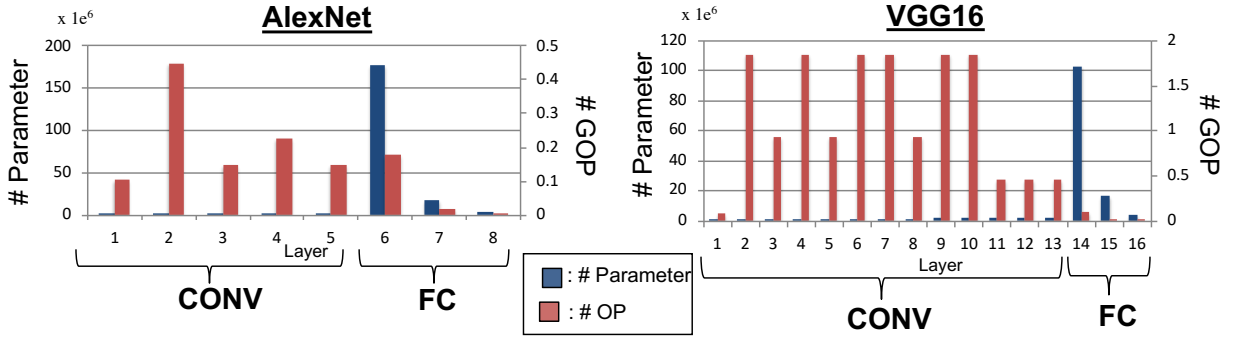


Figure 2.10: Characteristics between CONV and FC layers.

operations is larger, and in FC layer, the number of parameters is larger. Since the CONV layer can share the parameters, attainable performance can reach the computational roof of the roofline model by managing the dataflow. On the other hand, in FC layer, the memory bandwidth is a bottleneck. Therefore, many DNN accelerators are designed for CONV layer based on the data flow analysis [26]. The accelerator we propose in Chapter 4 supports both CONV and FC using three-dimensional stacking technology to ensure sufficient memory bandwidth.

From here, we introduce an optimized architecture exploration method based on FPGA. Figure 2.11 illustrates each parallelism of CONV layer. We defined the parallel parameters as follows: T_n as the input channel, T_m as the output channel, T_c as the output pixel of the column, T_r as the output pixel of the row, and T_k as the kernel. The parallel patterns we analyzed are as follows:

1. Channel parallelism (T_n, T_m)

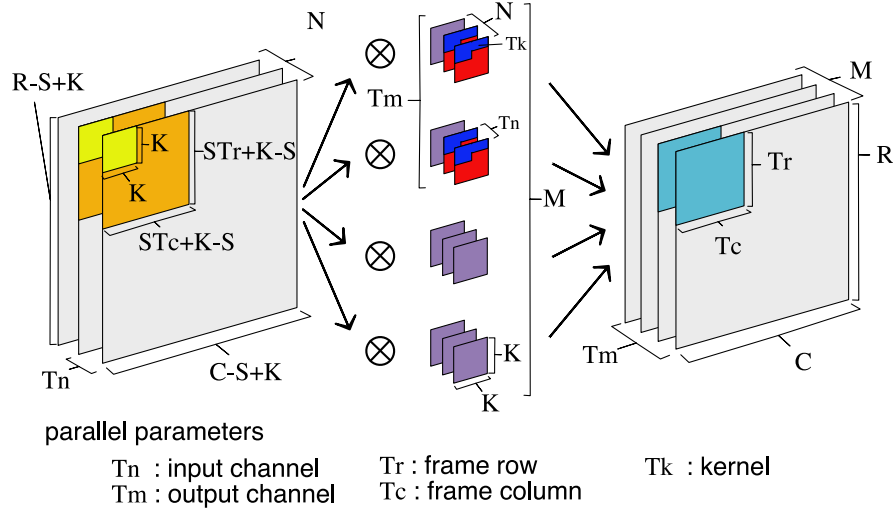


Figure 2.11: Partitioned CONV layer.

2. Pixel parallelism (T_r, T_c, T_k)
3. Channel and pixel parallelism (T_n, T_m, T_r, T_c, T_k).

Zhang *et al.* [27] handled the parallelism of 1), and Matamedi *et al.* [28] added a kernel parallelism to 1). In this study, a pixel parallelism includes the kernel parallelism as an input pixel. We explored the parallelism of 3) which considered all parallelisms in the case of binarized CNN in our previous paper [29]. When these parallelisms are determined, the CR and CtC ratio can be calculated. The most optimal point is obtained by calculating the CR and CtC ratio within the limitation of the target device. The CR and CtC ratio are defined from the following equations:

$$\begin{aligned}
 & \text{Computational Roof} \\
 &= \frac{\text{total number of operations}}{\text{number of execution cycles}} \\
 &= \frac{\text{set} \times R \times C \times M \times N \times K \times K}{\lceil \frac{R}{T_r} \rceil \times \lceil \frac{C}{T_c} \rceil \times \lceil \frac{M}{T_m} \rceil \times \lceil \frac{N}{T_n} \rceil \times \lceil \frac{K}{T_k} \rceil \times \lceil \frac{K}{T_k} \rceil}
 \end{aligned} \tag{2.6}$$

$$\begin{aligned}
& \text{Computation to communication ratio} \\
&= \frac{\text{total number of operations}}{\text{total amount of external memory access}} \\
&= \frac{2 \times R \times C \times M \times N \times K \times K}{A_{in}B_{in} + A_wB_w + A_{out}B_{out}} \tag{2.7}
\end{aligned}$$

where

$$A_{in} = \frac{M}{T_m} \times \frac{N}{T_n} \times \frac{R}{T_r} \times \frac{C}{T_c} \tag{2.8}$$

$$A_w = \frac{M}{T_m} \times \frac{N}{T_n} \tag{2.9}$$

$$A_{out} = \frac{M}{T_m} \times \frac{R}{T_r} \times \frac{C}{T_c} \tag{2.10}$$

$$B_{in} = T_n(S \times T_r + K - S)(S \times T_c + K - S) \times \text{bitwidth} \tag{2.11}$$

$$B_w = T_m \times T_n \times K^2 \times \text{bitwidth} \tag{2.12}$$

$$B_{out} = T_m \times T_r \times T_c \times \text{bitwidth} \tag{2.13}$$

$$B_{in} + B_w + B_{out} \leq \text{BRAM}_{\text{capacity}}. \tag{2.14}$$

The kernel parameters can be shared in a channel. We can explore the optimal point of each parallelism in the case of SIMD architecture with flexible interconnect.

2.4.2 DNN dataflow architectures

General neural network accelerators have a structure in which multiply-accumulate operations are performed in parallel, causing a memory bandwidth bottleneck. However, convolution operations, which account for 90% of CNN operations, require many parameters to be reusable. Therefore, the balance between the arithmetic processing and the memory bandwidth is important. Until now, various dedicated accelerators focusing on this convolution operation have been studied. These can be roughly divided into four types of patterns.

Park et al. [30] and Qiu et al. [18] have weights (Weight) in local registers and store pixels (Activation) or intermediate results (Psum) globally (A). We took the technique of propagating. These are configurations in which the reusability of the weight of the CNN is regarded as the most important, and the number of times of reading the weight can be minimized.

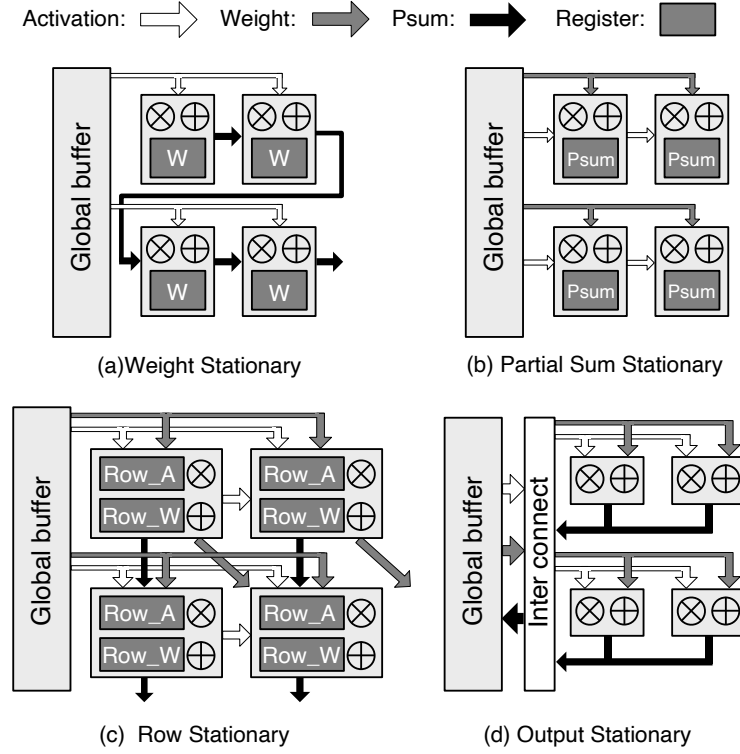


Figure 2.12: *Several stationary of CNN Accelerators.*

Zidong et al. [31] had intermediate results in local registers and propagated pixels or weights globally (B). In this way, while utilizing the continuity of the pixels and the weights, the intermediate calculation is held locally, thereby minimizing the moving cost of the calculation result.

The above (A) and (B) are vulnerable to changes in the network structure because the priorities are biased toward one element. Thus, Chen et al. [26] globally optimizes by grouping weights and pixels by row (C). By providing a local bus for each row and maximizing the use of the two-dimensional structure, the number of times of reading from the global buffer was minimized, thereby realizing power saving. Furthermore, efficiency is improved by providing a compression expression using the characteristic that the output of each neuron is almost 0 and a calculation skip function.

However, these three types lack the versatility because the network structure specializes in convolution. Therefore, this problem is avoided by performing exception processing or using a fixed network. On the other hand, Zhang et al. [27] introduced a search method to find the best perfor-

mance from an arbitrary network structure, and implemented it by FPGA (D). In this search method, an input from a global buffer is arbitrarily selected without having a local register. They combine and identify the roof line model of the computing unit that can perform computations on each layer, and calculate the highest computational performance and structure for a given memory bandwidth and area constraint.

As described above, a method of increasing the speed of CNN by hardware has been proposed. However, in order to cope with the rapidly evolving neural network structure, not only performance and power efficiency but also high-speed technology with high versatility will be important in the future.

CHAPTER 3

HW-oriented Quantization

3.1 Introduction

Algorithmic optimization helps in solving the memory capacity issue. To implement a large DNN model on a resource-constrained device, hardware-aware optimization of the DNN model, such as quantization, is crucial. Quantization represents the numerical values of weights and neuron activations in a simpler format with fewer bits as compared to the original format, and is an effective method for reducing both the memory footprint and computing hardware complexity. In this study, we focus on logarithmic quantization (log-quantization) [19, 20], which is a novel quantization method utilizing a power-of-2 logarithmic format, that can replace power/area hungry multipliers with adders. The required accuracy depends on the types of applications and use-cases applied even when the same hardware accelerator is utilized. To allow the adaptive selection of an appropriate bit width based on the application requirements, the processing elements of the accelerator are organized as reconfigurable arrays using bit-serial data paths, as opposed to parallel data paths. Hence, various

configurations of DNNs are efficiently mapped at a high performance.

3.2 Bit Reduction using Quantized Neural Networks

Recently developed DNNs have succeeded because of their massive large network configuration, which achieves a high level of recognition accuracy [2, 3]. Hence, to hold such a large DNN on SRAM-based 3D-stacked memory, how to reduce the size in terms of bits becomes a critical concern. A potential solution described herein is quantization, which is a major approximation technique for reducing the computing load and required memory footprint, as well as reducing the energy consumption.

3.2.1 Effects of Log-quantized DNNs

The most general quantization method is a linear quantization that represents a value using a statically fixed numeric range and steps. A fixed-point representation, as opposed to a floating-point representation, is used to reduce the hardware complexity such that it improves the computing performance and energy efficiency. Specifically, a narrower quantization is more suitable for a higher performance and energy efficiency.

LOGNET [19, 20] is a multi-bit quantization method that utilizes a power-of-2 logarithmic format. The most important feature of logarithmic quantization (log-quantization) is that multiplier hardware is no longer required because all multiplications in the linear field are represented simply through additions in the logarithmic field. Therefore, LOGNET can potentially achieve a high level of energy efficiency. Another advantage of LOGNET is that the memory footprint and bandwidth requirements are much lower than with linear quantization, but with the same accuracy, because a log-quantization can represent the same numeric range using fewer bits. A key insight here is that most of the weight distributions generally form a Gaussian distribution, in which smaller values appear more frequently than larger values. Log-quantization can represent these types of non-uniform distributions with a lower amount of numerical errors as compared to linear quantization with the same bit width. Figure 3.1 shows the weight distribution with log-quantization applied. Its denser the finer approach allows it to represent weight/activation distributions better.

In log-quantized DNNs, the activations and weights are quantized by the power of two, i.e., $\tilde{A} = \text{Quantize}(\log_2 A)$ and $\tilde{W} = \text{Quantize}(\log_2 W)$. In [19], they quantized only the weight and activation parameters, and calculated them with full-bit precision. However, considering actual

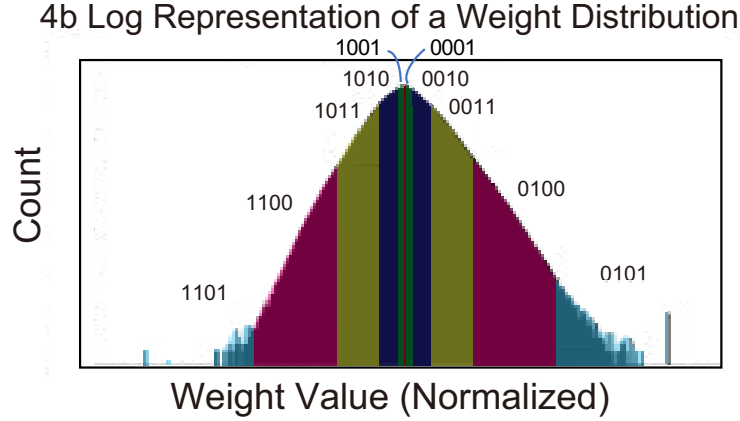


Figure 3.1: *Weight distribution with log-quantization.*

hardware, it is difficult to operate additions in a logarithmic domain. Therefore, we proposed a logarithmic and linear hybrid mechanism by converting the logarithmic domain to a linear domain before the additions. This guarantees the accuracy of addition including biases that also serve as normalization. It achieves a small resource calculation without sacrificing accuracy. Finally, it converts back to the logarithmic domain and approximates the scale parameter as $\tilde{\alpha} = \text{Quantize}(\log_2 \alpha)$. Therefore, equation (1) and (2) can be approximated as

$$\tilde{O} = f[\tilde{\alpha} + \text{Quantize}\{\log_2(\sum 2^{(\tilde{A} + \tilde{W})} + B)\}]. \quad (3.1)$$

This means that all the calculations can be represented by the low-energy consuming adder.

Figure 3.2 shows the results of the recognition accuracy of the ImageNet [32] classification in AlexNet [2] and VGG [3] for log or linear quantization. The recognition accuracy is evaluated using TensorFlow [33] and compared with a floating point (FP 32-bit) which is the state-of-the-art accuracy in ILVSRC [32]. From both results, log-quantization maintains a higher accuracy in a low bit-width than linear quantization. The log-quantization also enables low-energy (area) computing because all multiplications can be replaced with additions, as shown in equation (5). Not only is the hardware implementation low in cost, but the bit-serial processing is also achievable. This can optimize the performance-accuracy trade-off for different application levels.

The relationship between the quantization method used and the recognition accuracy has been widely investigated [34, 35]. Nevertheless, the relationship between energy efficiency and recognition accuracy has yet to be adequately examined. The purpose of the present study involves realizing a highly accurate and energy-efficient DNN accelerator for embedded systems, and thus most energy-

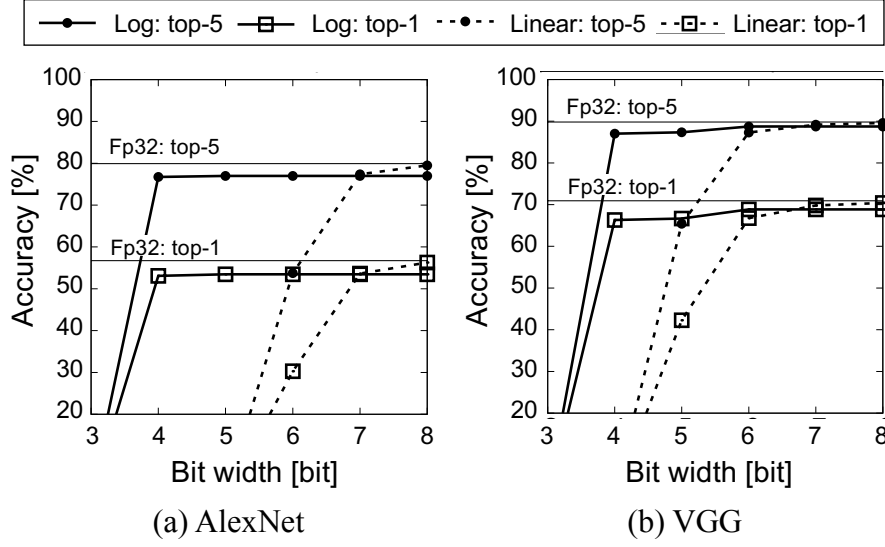


Figure 3.2: Bit width vs. ImageNet classification accuracy of log-quantization and linear quantization on AlexNet and VGG.

efficient quantization approaches must be identified. We analyzed the relationship between the energy consumption and recognition accuracy of several quantization methods, as well as the bit precision, through a software simulation. Fig. 3.3 shows the relationship between the energy consumption of a MAC unit and the recognition accuracy in fixed-point quantization (Linear) and log-quantization (Log). The energy consumption of each MAC is estimated using Synopsys Design Compiler [36] with 40-nm UMC technology. The recognition accuracy is evaluated using TensorFlow [33] with the ImageNet dataset [32], and compared with a 32-bit floating-point (FP32) approach. It should be noted that the multiplier used in log-quantization is realized as an adder, and the original adder is synthesized as a combination of a standard adder, a log-to-linear converter, and a linear-to-log converter.

Both methods with wide quantization widths achieve a level of accuracy nearly identical to that of the floating-point method. In contrast, narrower quantization widths degrade the accuracy in both the quantization methods. In terms of the tradeoff between energy and accuracy, the energy consumption of 4-bit log-quantization is lower than that of 6-bit linear quantization even though the recognition accuracy of the former exceeds that of the latter. Moreover, 4-bit log-quantization achieves a level of accuracy similar to that of 7-bit linear quantization under significantly lower energy consumption. The results indicate that log-quantization at a narrow bit width has the potential to provide high accuracy with low energy consumption. Needless to say, a narrower bit width is useful for lower energy

3.3. Bit-Parallel Processing vs. Bit-Serial Processing

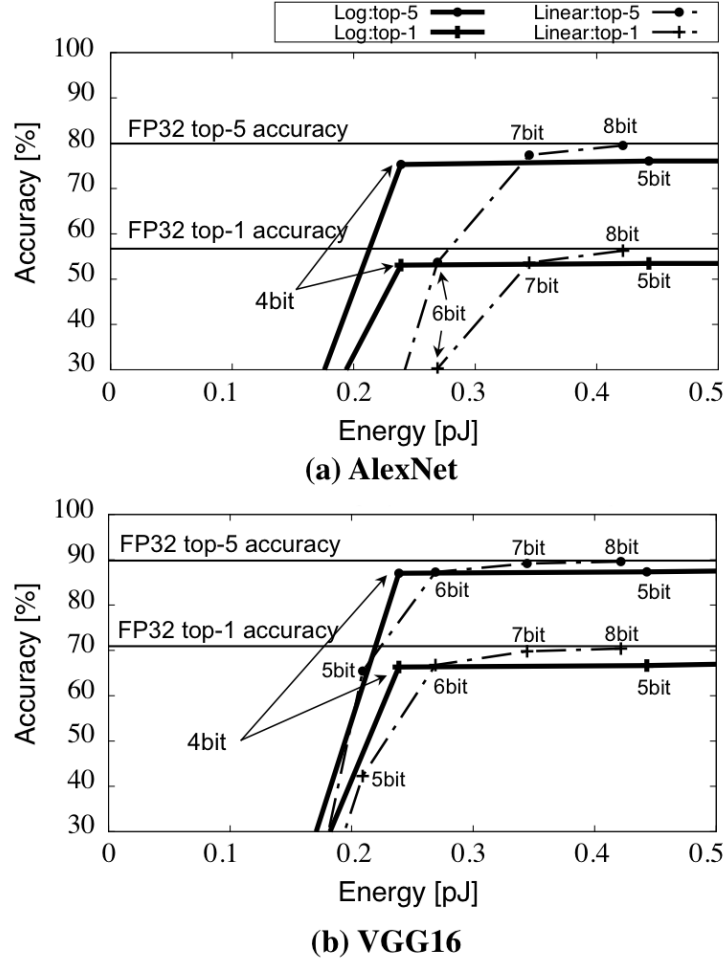


Figure 3.3: MAC energy vs. ImageNet classification accuracy of logarithmic and linear quantizations on AlexNet and VGG16 networks.

consumption and for reducing the memory footprint and bandwidth requirements.

3.3 Bit-Parallel Processing vs. Bit-Serial Processing

In quantized neural networks, an appropriate bit width depends on the workload complexity. Bit-serial computation is generally well suited to efficiently support a variety of quantization bit widths. In log-quantization, its multiplier-free nature allows one-pass bit serial computations. That is, every bit simply flows through the computation units without requiring a buffer to store the previous bits. Fig. 3.4 compares the circuit areas and energy consumption of 8-bit bit-parallel linear quantization,

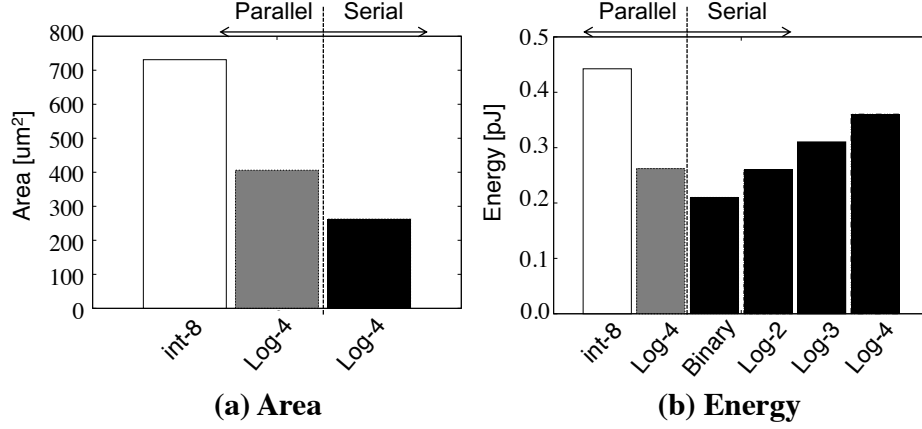


Figure 3.4: (a) Area and (b) energy of MAC: 4-bit log-serial, 4-bit log-parallel, and 8-bit integer-parallel.

4-bit bit-parallel log-quantization, and 4-bit bit-serial log-quantization data paths. Bit-parallel fixed-point quantization uses a general multiplier, and therefore both the circuit area and energy consumption are relatively high. The circuit area of bit-parallel log-quantization is smaller than that of linear quantization. Although the energy consumption of 4-bit bit-serial log-quantization is higher than that of bit-parallel log-quantization, the advantage of the former lies in its versatility mentioned above. In DNN processing, there is sufficient data-level parallelism, such as matrix and vector multiplications. Thus, a high computing performance is achieved by allocating numbers of bit-serial log-quantization data paths in parallel.

3.4 Related Works for Bit-Serial Processing

Stripes [37] and Bit-pragmatic [38] proposed a bit-serial operation architecture based on DaDian-Nao [39]. This architecture supports flexible bit-width operations to optimize the structure of the processing elements for fewer computations at the bit level. These proposals apply a bit-serial structure only to the activations. In contrast, our proposed architecture adopts a bit-serial manner to both the activations and weights for better efficiency.

3.5 Conclusion

we describes the hardware-oriented quantization method for DNN inference. Logarithmic quantization based serial processing realizes the low energy and area with flexible bit precision. The flexibility is important to optimize a trade-off between accuracy and calculation cost. Because required minimum bit width is different among with different application level.

CHAPTER 4

Log-quantized Neural Network Engine with Stacking Technology

4.1 Introduction

Deep neural networks (DNNs) have been rapidly developing in various fields of machine learning applications [13, 40]. The size of modern DNN models is growing in order to obtain higher recognition accuracy and robustness for a wide range of applications [5, 41]. These DNNs require a large memory footprint and they consume high energy. To implement an accelerator for this powerful computing on limited energy and resources, it is important to combine a new device technology and an optimization algorithm.

A large and high-bandwidth external memory is a key for DNN computing that requires a huge amount of data. These issues are solved by the 3D-stacking of memories. Recently, a high bandwidth memory (HBM) and a hybrid memory cube (HMC) are used for DRAM-based 3D-stacking memories [42, 43]. However, the long DRAM latency remains problematic and limits the perfor-

Chapter 4. Log-quantized Neural Network Engine with Stacking Technology

mance. Moreover, recent algorithm-level optimizations, such as network pruning and compression, have shown success in reducing the DNN memory footprint. However, they cause the DNN construction to become irregular and sparse. They induce an additional requirement for agile random accesses to the memory systems. Therefore, we introduce SRAM-based 3D-stacking memories (3D SRAMs) with wireless communication using inductive coupling interfaces to realize both high memory bandwidth and low and uniform access latency.

To implement a large DNN model on constraint models, the algorithm-level hardware-aware optimization of DNN models, such as quantization, is also crucial. Quantization represents the numerical values of weights and neurons by a simpler format with fewer bits compared to the original format. It is an effective method to reduce both the memory footprint and computing hardware complexity. A floating-point unit (FPU) generally consumes a larger circuit area and higher energy than integer units in all hardware systems. Compared to the floating point, a fixed-point linear quantization achieves a better performance with less memory footprint and energy budget. Moreover, logarithmic quantization (log-quantization) is more suitable for digital system architectures because a power-of-two logarithmic format simplifies the power-hungry multiplications. Further, we show that the log-quantization can maintain a high recognition accuracy with a lower bit-width than linear quantization.

From the application viewpoint, the recognition accuracy requirement depends on the application and environment, even if we used the same hardware accelerator. Evidently, a lower bit-width accelerator achieves better energy efficiency and computing speed. However, fixed bit-width computing cannot adapt to the most efficient energy bit-width of the other applications, which have different accuracy requirements. A serial operation can adapt to the different bit-widths for each application and each layer [37, 38]. By combining the adaptive control of the efficient bit-width computation and the quantization intensity, we can maximize the accuracy-balanced energy efficiency.

In this paper, we present a log-quantized neural network accelerator with programmability for runtime energy and accuracy optimization called QUEST. QUEST employs log-quantization to improve memory bandwidth, area, and energy efficiency. To allow the adaptive control of a bit-width requirement for each application, the accelerator is organized as reconfigurable arrays by a bit-serial data path, as opposed to parallel data paths. Different bit-precision configurations are efficiently supported by changing the processing latency of each operation. The approach also enables the efficient support of different layer types of neural networks, i.e., convolution (CONV) and fully connected (FC) layers, by a unified hardware structure. The behavior of the unified hardware is programmed via a simple

4.2. Quantized Neural Network Engine with Stacking Technology (QUEST)

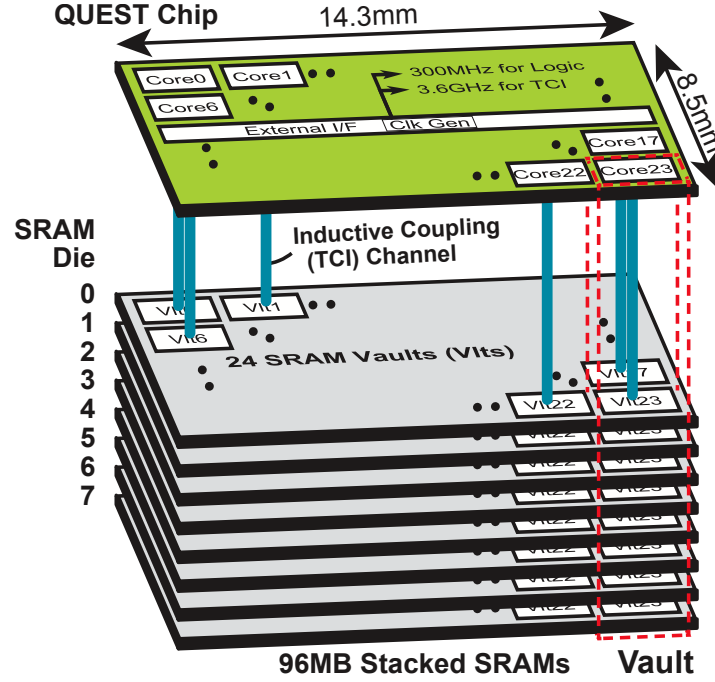


Figure 4.1: Overall design of QUEST.

dedicated instruction set architecture (ISA). Hence, various configurations of neural networks are efficiently mapped at a high performance. A recently proposed DNN accelerator chip called BRein memory [44] features binary or ternary DNNs. It can also reduce memory usage and computational resources drastically. However, the bit precision is fixed only to binary or ternary, and the chip mainly focuses on the FC layers. This observation motivated us toward the development of QUEST.

Finally, we developed a prototype LSI based on the proposed architecture. High memory bandwidth and short latency are critical to improve the absolute performance and energy efficiency. The accelerator employs 3D SRAMs as external memories with inductive coupling interfaces [45] that enable wireless data communication at low integration cost. By stacking SRAMs instead of DRAMs, a significantly lower memory access latency is expected than in the case of DRAMs.

4.2 Quantized Neural Network Engine with Stacking Technology (QUEST)

A key consideration for deep neural network (DNN) inference accelerators is the need for large and high-bandwidth external memories. Although an architectural concept for stacking a DNN accelerator with DRAMs was proposed, long DRAM latency remains problematic and limits performance improvement [43]. Recent algorithm-level evolutions such as network pruning and compression have

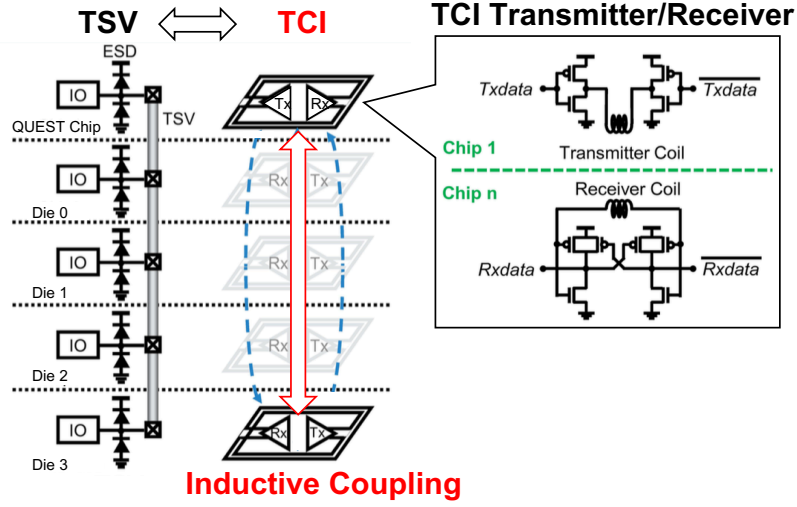


Figure 4.2: ThruChip Interconnect (TCI).

shown success in reducing the DNN memory size [24]; however, since networks become irregular and sparse, they induce an additional need for agile random accesses to the memory systems.

Chapter 3 touched on the fact that an efficient bit reduction from log-quantization may allow all parameters required for DNN computing to be stored at SRAM-level capacity. In addition to these baseline discussions, we propose a log-quantized bit-serial DNN accelerator with 3D-stacked SRAM modules.

Figure 4.1 illustrates our proposed DNN accelerator chip (QUEST) with multi-vault SRAMs using inductive-coupling die-to-die wireless communication technology, known as the ThruChip Interface (TCI) [45]. Eight SRAM dies are stacked, achieving an SRAM capacity of 96 MB in total.

QUEST has 24 processing cores running at 300 MHz, and each core is associated with one 32 bit-width 4-MB SRAM vault. Each block of the DNN core or SRAM has a built-in TCI controller that independently communicates with other TCI controllers. The controller transfers data at 3.6 GHz in a bit-serial manner. Thus, 32-bit data are read out in three clock cycles at 300 MHz on the DNN chip. Utilizing the stacked SRAMs provides sufficient memory bandwidth with low latency.

4.2.1 3D-stacking with Inductive Coupling Technology

QUEST and eight SRAMs are TCI-stacked as a single $14.3 \times 8.5 \text{ mm}^2$ 3D module. Figure 4.2 illustrated the abstract model of TCI. TSV is commonly used for die stacking, but it is known to experience

4.2. Quantized Neural Network Engine with Stacking Technology (QUEST)

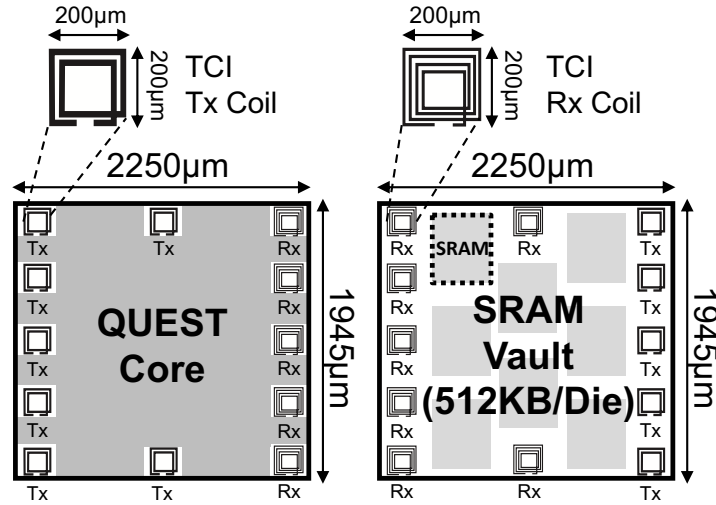


Figure 4.3: Detail layout of TCI channels on *QUEST* cores and *SRAM* vaults.

open-contact failure when it becomes fine-pitched. TCI has a simple transmitter and receiver composed of CMOS digital circuits so that it is scalable and is integrated at a low cost. Parallel TCI channels placed in a planar manner provide the QUEST with multiple independent high-bandwidth access points to the stacked SRAMs. TSV technology, used commonly for die stacking, is known to experience open-contact failure. In our design, however, because all signal transmissions are conducted by wireless TCI channels, the presented 3D module can limit the usage of TSVs to power/ground grids, where numerous parallel connections negate this concern. TCI enables broadcast access to all stacked dies by fixed latency. It achieves 3.6 Gbps transmission per TCI channel in this design.

4.2.2 Short and Uniformed SRAM Access

SRAMs can provide random access capability with extremely low latency (an order of magnitude lower than DRAMs), whereas 3D stacking facilitates SRAMs in achieving a reasonably large memory capacity. An SRAM vault (512 kB/die) is the same size as a QUEST core, as shown in Fig. 4.3. The TCI channel in the QUEST core has seven Tx and five Rx coils composed of 200 μm square, whereas the SRAM vaults have five Tx and seven Rx coils. Operating at 3.6 GHz, the TCI channel realizes 9.6 Gb/s/vault, combined with 28.8 GB/s/module, and an R/W data bandwidth in a source-synchronous manner. The TCI Rx circuits require 90-mV signal amplitude to receive a reliable data less than 10^{-12} bit-error rate (BER) that is sufficient for product-level designs with additionally including error correction circuits. The thickness of each SRAM die is 10 μm . Therefore, the supply voltage of the

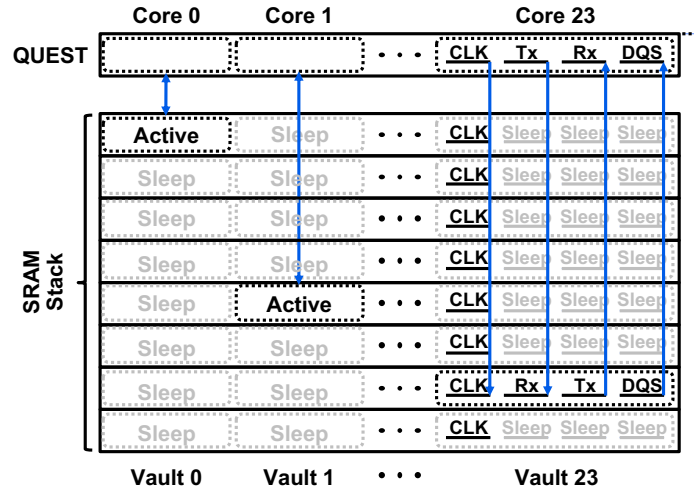


Figure 4.4: Broadcast access with fixed latency by TCI.

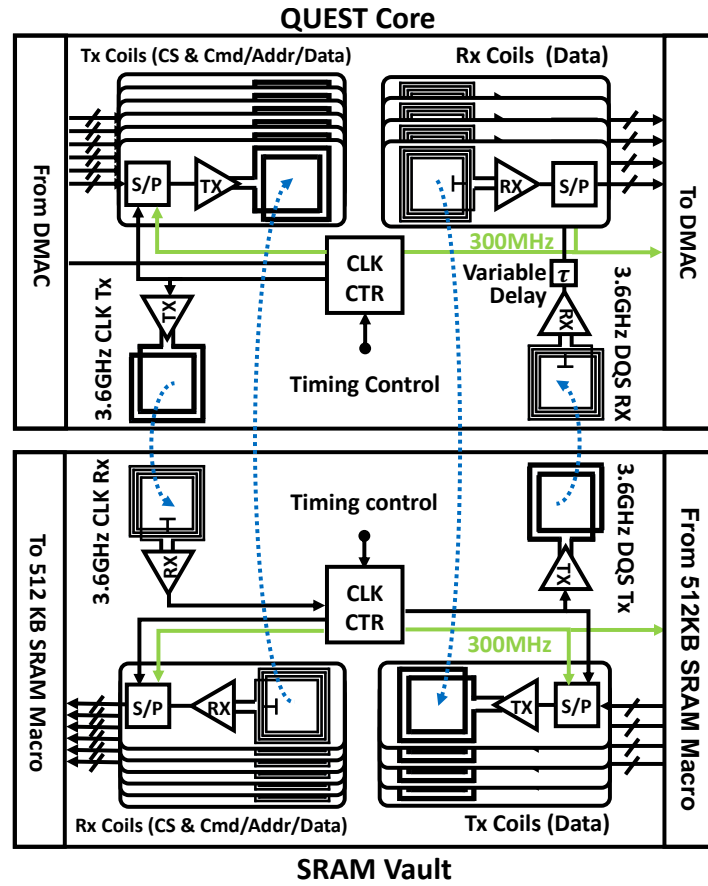


Figure 4.5: TCI wireless communication circuit diagram.

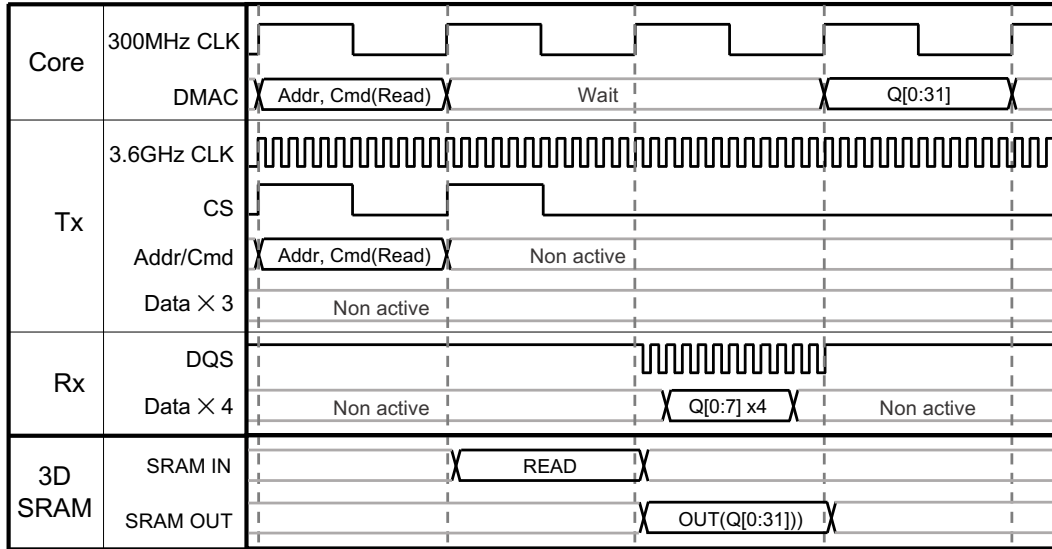


Figure 4.6: Time chart of single read access of 3D SRAMs through the TCI.

TCI Tx circuits should be at least 0.6 V to achieve reliable communication via the 8-stacking SRAM dies. In this case, the TCI circuits consume 1.6 pJ/bit at Tx and 0.4 pJ/bit at Rx. A 3.6-GHz TCI channel contains the CLK, CS, and five Cmd/Addr/Data coils for the Tx, and the DQS and four data coils for the Rx, as shown in Fig. 4.5. Each SRAM die has its own static ID with which an address broadcast from QUEST is matched for activation. The DQS variable delay in the QUEST Rx core is used for assuring the data setup/hold timing. The SRAM features the same die size as that of QUEST (14.3 mm × 8.5 mm), the same frequency (300 MHz), and is fabricated using the same 40-nm CMOS process.

Figure 4.6 shows a time chart of the TCI channel in the QUEST core for a single SRAM read. The two Tx coils in the QUEST core are used for the request read access and address, then the data is accessed from the SRAM and communicated by four Rx coils. Each sequence is executed within a core cycle. Therefore, the R/W latency including the TCI trip time is three cycles, which is uniform over the eight SRAMs.

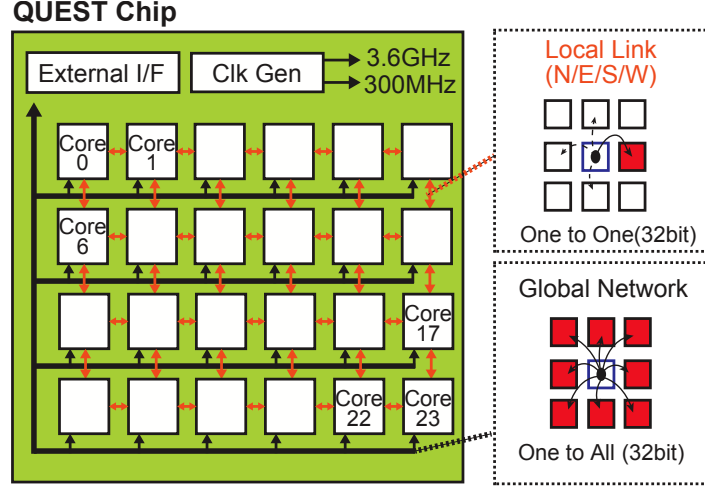


Figure 4.7: Interconnect among QUEST cores.

4.3 System Architecture

4.3.1 QUEST Core Architecture

As shown in Fig. 4.7, the 24 cores operate in a multiple instruction, multiple data (MIMD)-parallel manner, where the intercore communication is handled either with a mesh-structured local link between the neighboring cores or the tree-structured global network between one and all other cores. The synchronization among the cores is managed through a synchronization table (sync. table) composed of 24-bit read/write registers for writing a synchronization flag of the target core. Each core has a microprogrammed sequencer for setting and controlling the data flow.

A detailed QUEST core architecture is shown in Fig. 4.8. Each core has a processing element array (PE array) with on-chip double buffers, a direct memory access controller (DMAC), a TCI interface, and a sequencer with a programmable microcontroller (μ ctrl). The sequencer contains an address generator for the on-chip buffers and a PE array controller that is a programmable finite state machine (FSM) generating clock-level control signals of the PE array. As is widely known, the computation and memory access patterns of DNN inference processing can be adequately predicted because almost all patterns are statically determined by model parameters. Hence, our architecture employs scratchpad memories and DMAC as opposed to complicated cache systems. Each DMAC accesses both the other memory blocks on the other cores via the interconnections and the local memory blocks, including the SRAM banks on the stacked dies.

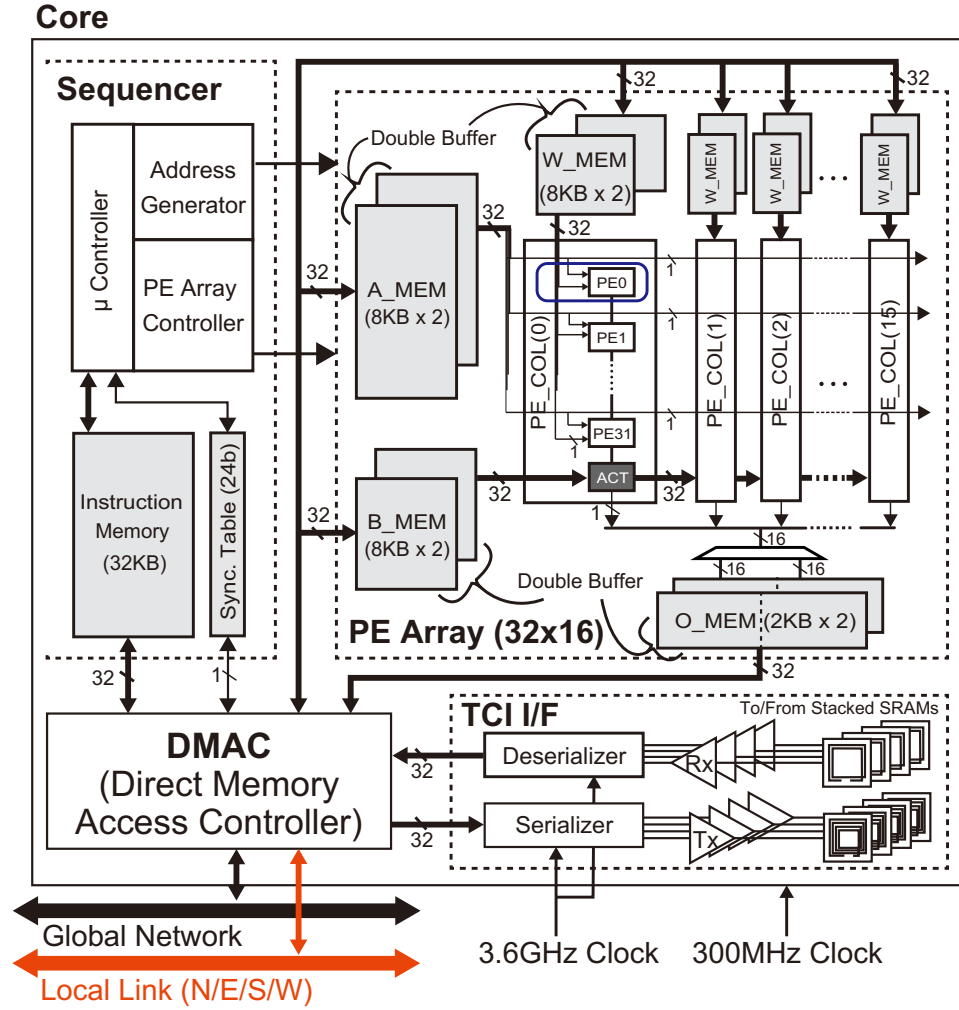


Figure 4.8: Detailed circuit diagram of core architecture.

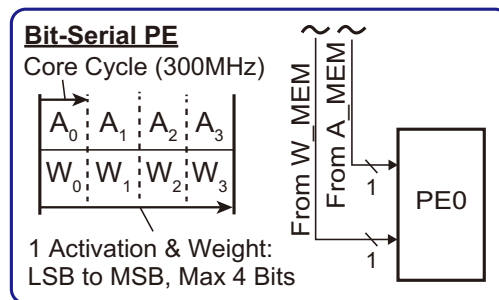


Figure 4.9: Bit-serial PE mechanism.

Chapter 4. Log-quantized Neural Network Engine with Stacking Technology

The PE array has 16 PE columns (PE_COLs) composed of 32 PEs and an activation unit (ACT) at the column tail. The 32×16 PE array features a bit-serial architecture: the PE conducts binary computation in a single cycle, and N-bit log-quantization in N cycles ($N < 5$). Weights double-buffered in W_MEMs are distributed to the PE array in a fully parallel manner, whereas incoming activations also double-buffered in A_MEMs are broadcast in a row-parallel manner. The ACT unit at the tail of each column apply bias (shifted-in from B_MEMs), scaling, and activation function; and then write the output activations into O_MEMs. Each PE conducts "multiply and accumulate" in a bit serial manner as shown in Fig. 4.9. Therefore, all inputs; *i.e.*, activations and weights, are serialized to a bit stream. This bit-serial operation realizes flexible bit precision which can be optimized for each DNN application. Our chip design supports 1 to 4-bit precision.

4.3.2 PE Array Architecture

A PE array consists of multiple PEs that receive the weight and activation values from the RAM modules in a bit-serial manner. A PE has two different numerical domains, namely, a logarithmic domain, and a linear domain. All data are represented in a bit-serial logarithmic representation, and hence multipliers are not required. With respect to the processing of radical multiplications in the logarithmic domain, bit-serial weight and activation data in the logarithmic representation are accumulated at a serial adder, as opposed to multiplier. A product (specifically, a sum in the logarithmic domain) is then converted into a linear value for radical accumulations through a one-hot decoder to calculate a partial sum by its accumulator (ACC). The bit width of the quantized data is adaptively selected based on the workload characteristics and application requirements. In addition to log-quantization, the PE naturally supports binary quantization.

The partial sums of the PEs in the same column are then delivered to the activation unit (ACT) at the bottom of the column through shift registers. An ACC in ACT accumulates all partial sums from the PEs in the same column. The accumulator then appends a bias value loaded from the bias RAM in advance, which is added to the accumulation result. In recent DNNs, batch normalization is a key component for highly accurate and robust recognition [14]. In terms of inference processing, batch normalization is represented as $Ax + B$, where A denotes a scaling value, and B indicates a bias value. Bias additions are conducted in the linear domain, as described above, and scaling multiplications are executed in the logarithmic domain. After the accumulation in ACT, the resulting value is again converted into the logarithmic format using a priority encoder. A scaling operation in the logarithmic

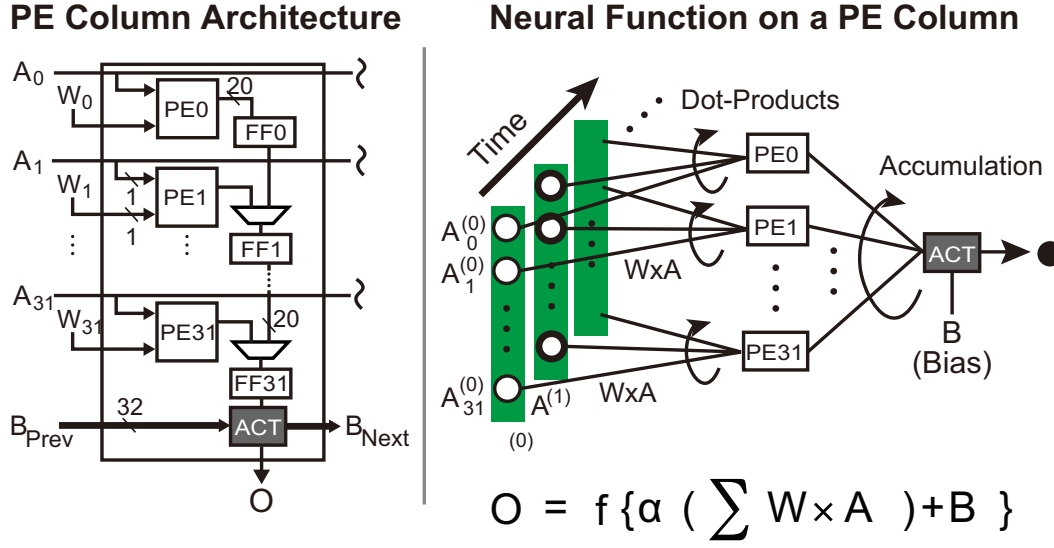


Figure 4.10: Detailed computation in PE_COL.

domain is then simply calculated through the addition of the resulting value and a scaling value loaded from the bias RAM. After passing through the activate function unit and the pooling unit, the serialized values are finally combined into a single word and written into the output RAM.

These mechanisms are key enablers for handling various DNNs on a single homogeneous PE array ([46, 47] use hybrid cores and hybrid PEs, respectively, for different DNN types): e.g., for a FC layer in CNN, MLP, or RNN, max 32 fan-in for a neuron is mapped on a PE column at a time and then time-multiplexed on the same column. For a CONV layer in CNN, on the other hand, max 32 input channels are mapped on a PE column at a time and then time-multiplexed. The filter kernel is stored vertically in the W_MEMs and processed in an element-by-element, kernel-parallel manner. UNPU [48] also composes the unified PE alignment for various DNNs by providing unique weight bits to all PEs. They quantize only weights by LUT-based quantization which improves performance and flexibility. Meanwhile, we constructed high-dense PE array in parallel by quantizing and serializing both activations and weights in logarithmic domain.

Unlike other array-structured DNN accelerators, all PEs receive unique weight bits, whereas the PEs in a row receive an identical activation bit. In a PE column, partial dot products are first generated in parallel in PEs and then simply shifted towards ACT where they are accumulated as detailed in Fig. 4.10. In a PE_COL, the partial dot products are first generated in parallel in the PEs and then simply shifted towards the ACT where they are accumulated. The pipelined shifts hide behind the

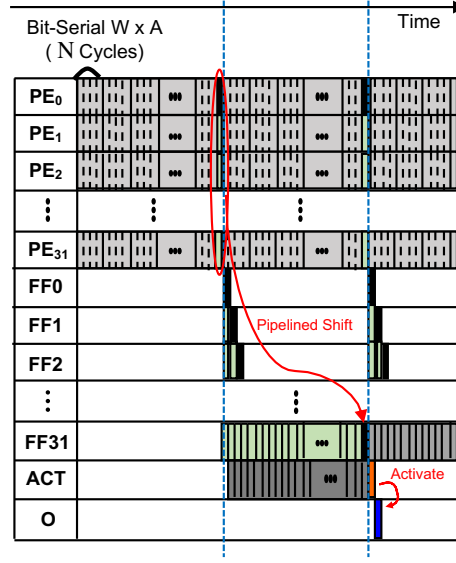


Figure 4.11: Pipelined shift dataflow in PE.COL for dot-product computations.

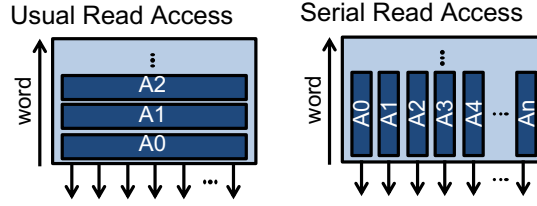


Figure 4.12: Bit-serial memory access.

PE-parallel, bit-serial, and dot-product computations, as shown in Fig. 4.11.

As shown in Fig. 4.8, each PE array has four dedicated on-chip RAM modules for weight, activation, bias, and output. Each RAM has two banks for a double-buffered transfer of data between other memory components. The accelerator employs a bit-serial logarithmic numerical format. Therefore, data arrangement on the memory spaces is significantly different from that in standard bit-parallel processing, as shown in Fig. 4.12. Each bit of multiple values is collectively combined into a single word in the bit-serial format of this accelerator. Because of the lightweight serialized interconnects, the weight value of each PE can be directly distributed from the weight RAM. In contrast to the weight, activation values are shared among the PEs in the same row.

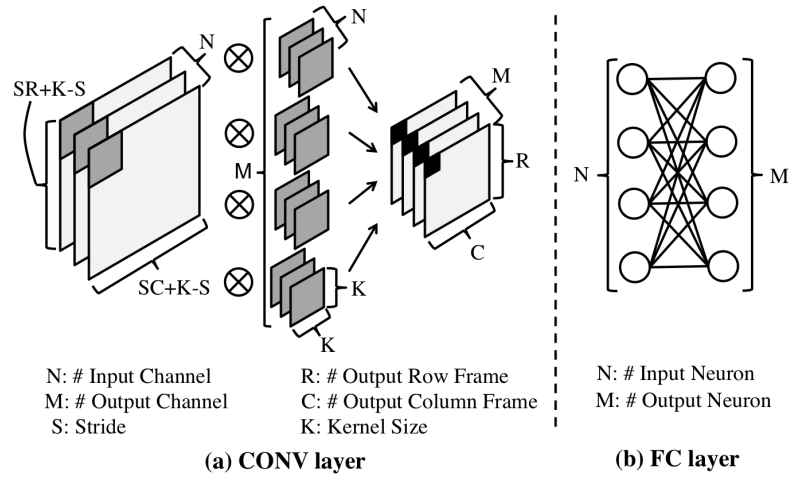


Figure 4.13: Types of Neural Networks.

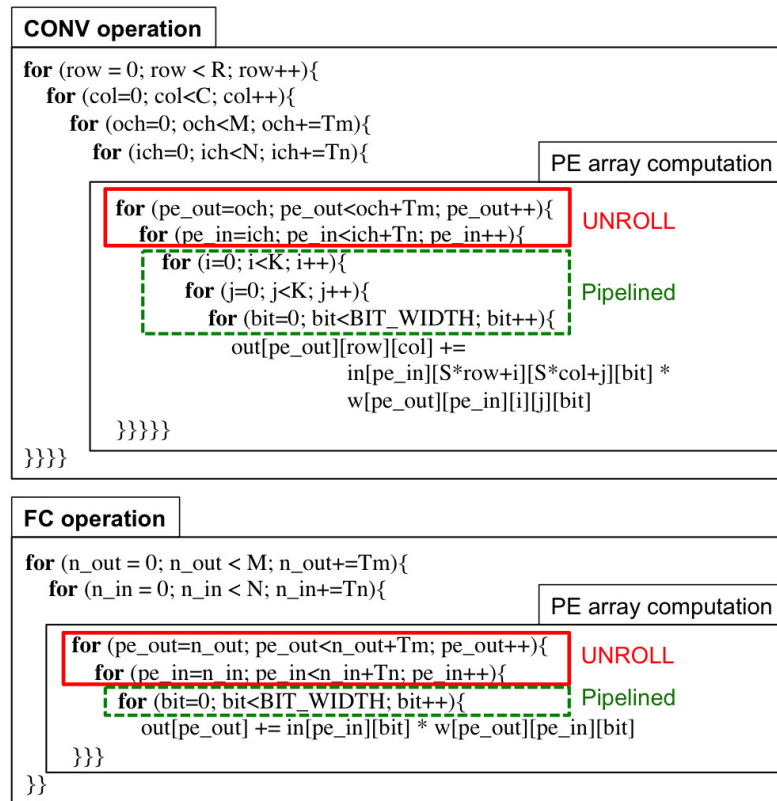


Figure 4.14: Pseudocode of CONV/FC operation in the homogeneous PE array.

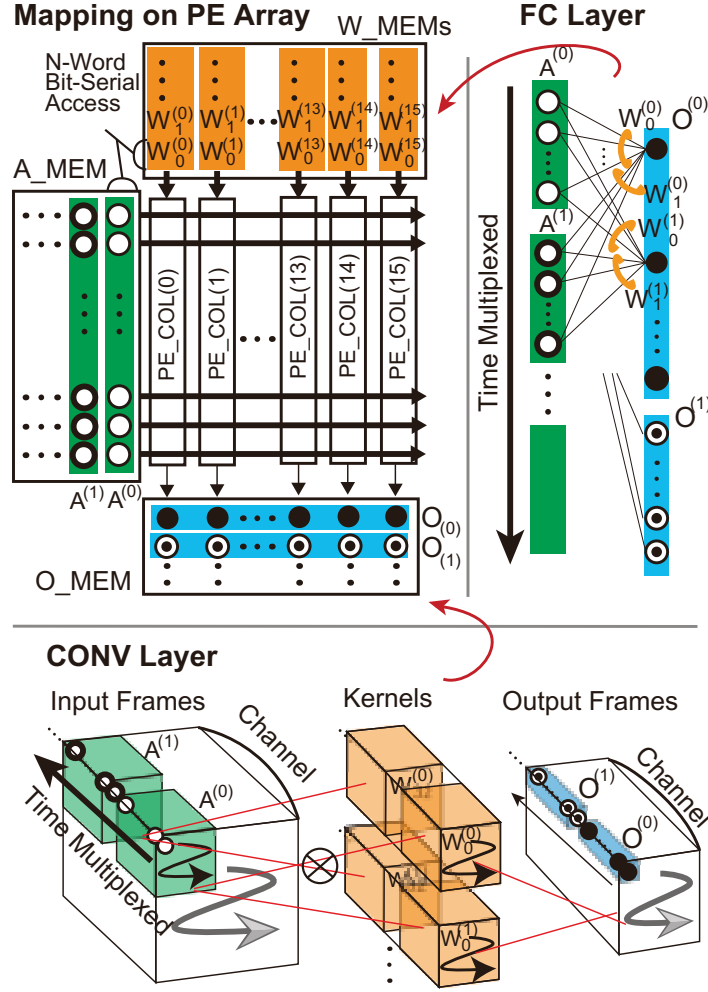


Figure 4.15: FC/CONV dataflow on a homogeneous PE array.

4.3.3 Mapping Convolution and Fully Connected Layers on Unified Architecture

A convolutional neural network (CNN) generally consists of CONV layers, pooling (POOL) layers, and FC layers. Most of the previous approaches for general DNN accelerators install a number of MAC units [46, 49], or dedicated hardware units, for the CONV and FC layers, respectively. In contrast, the PE array of the proposed accelerator employs a unified architecture that supports both CONV and FC layers with the same hardware, thereby reducing the area of silicon required.

Fig. 4.13 shows the computing patterns of (a) CONV and (b) FC layers, and Fig. 4.14 shows the corresponding pseudo codes of these layers Fig. 4.13.

In CONV layer processing, computations are partitioned in both the input and output channel

directions, where T_n denotes the chunk size of the input channel direction, and T_m denotes the chunk size of the output channel direction. In addition, pe_in and pe_out are iterators tied to T_n and T_m , respectively. Chunked operations of the input and output channels are unrolled inside a PE array, and T_m operations are arranged in parallel in the column direction. Because a CONV computation contains repeatedly reused activations, such reused activations are delivered from the activation RAM modules each time through a programmable address controller. As in the CONV layer, computations are partitioned in both the input and output neuron directions in FC layer processing. Although no weight value is reused in the FC layer processing, the patterns of parallel computations and activation sharing are almost identical to those of the CONV layer. In this way, the proposed PE array computes both the CONV and FC layers in the same structure by sharing the same input elements, i.e., the input channel (N) and kernel size (K^2) in the case of CONV, and the input neuron (N) in the case of FC.

As shown in Fig. 4.15, our proposed architecture shares the activation values with multiple PEs in the same row, whereas the weight values are independently assigned for every PE. In both computing patterns, each activation value is repeatedly consumed to determine multiple output values. Therefore, both are executed on the unified architecture through row-wise activation sharing. Finally, both the FC and CONV layers can be handled by using the same PE array structure.

4.3.4 PE microarchitecture

The detailed block diagram of PE and ACT is shown in Fig. 4.16. At first, the serialized inputs of the activations and weights are conducted as a serial addition in the logarithmic domain instead of multiplication in the linear domain. After that, the sum of the serial addition is converted in the linear domain by a one-hot decoder, and finally added by the accumulator. The partial sum in each PE is accumulated at the own PE until the finishing pipeline shift of previous partial sums from every PE to the ACT. The ACT applies the final accumulation, bias (shifted-in from B_MEMs), scaling, and activation function; and then writes the output activations into O_MEMs.

4.3.5 Programmable Controller

Although most of the computations of a DNN are multiplication and addition operations, data access patterns have significantly large variations according to the layer structure. To compute all types of layers in a unified architecture, a programmable μ controller is installed, which manages the overall behavior of the PE array and DMA controller, as shown in Fig. 4.17. The μ controller has a synchro-

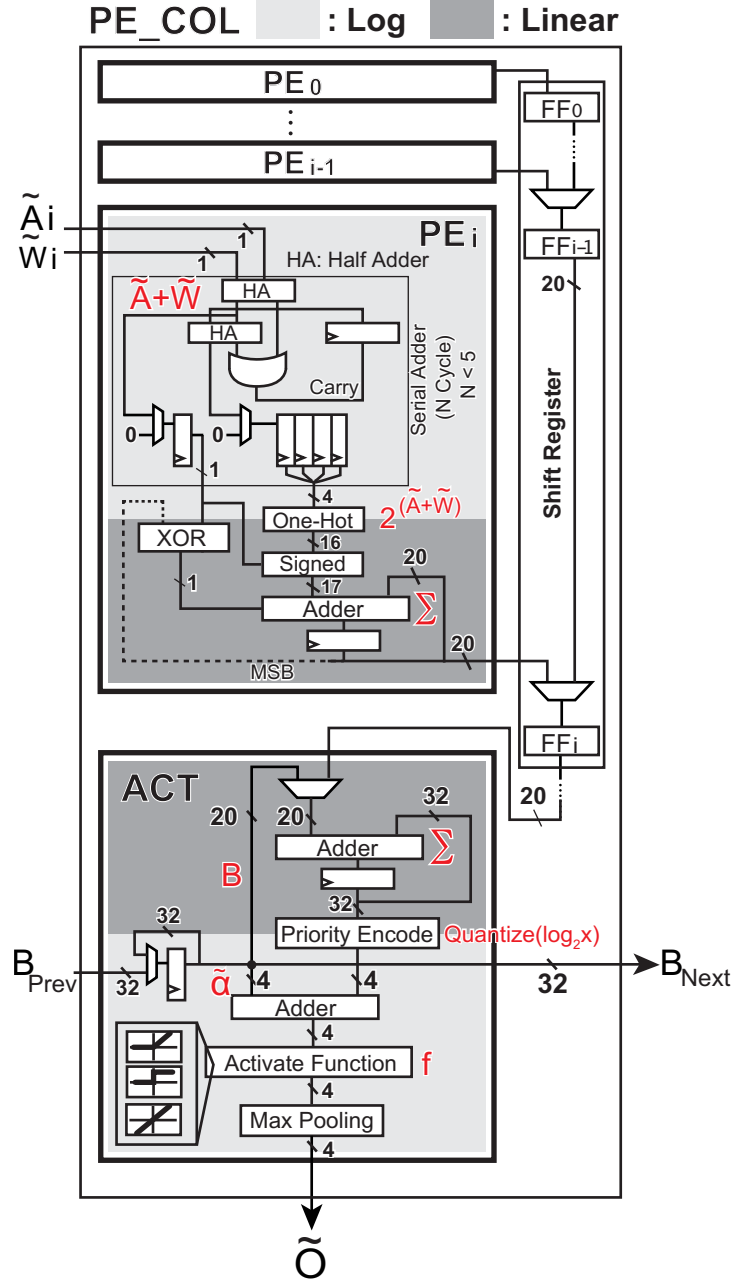


Figure 4.16: PE and ACT microarchitecture.

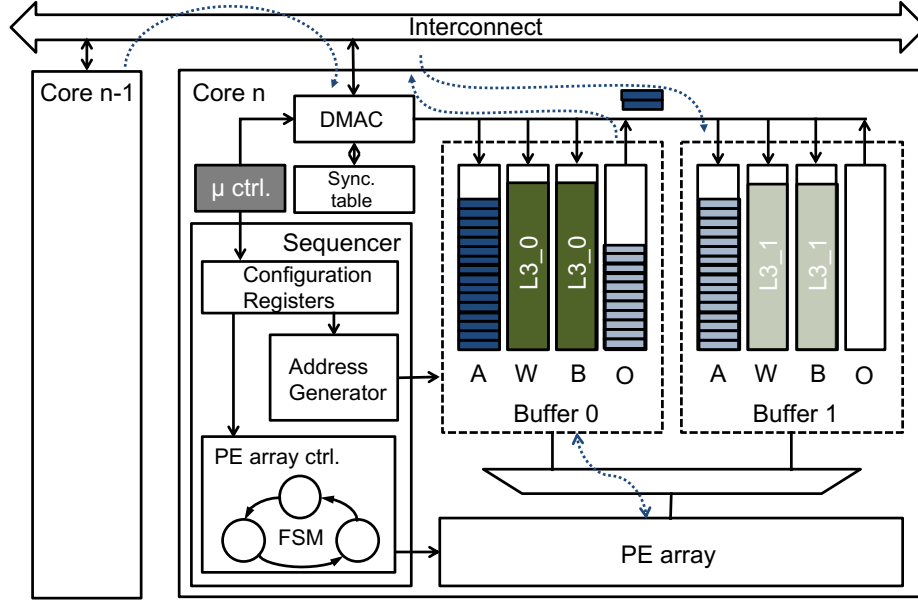


Figure 4.17: *Controller architecture.*

nization table (Sync. table) of the atomic read/write registers for concurrent parallel programming among the cores.

Fig. 4.18 presents the addressing mechanism of the activation RAM for the convolution layer operations. The locations of the activation values on the on-chip RAM modules are dynamically configured at runtime. The activation data used in each computation in a PE array for the convolution layer have four dimensions. When the shape (width and height) of the output activations is identical to that of the input activations, padding to the input activations is used. The memory access patterns can be summarized as follows:

$$\begin{aligned}
 Addr &= pad_col + pad_row + stride_col \\
 &+ stride_row + k_col + k_row + bit_width
 \end{aligned}
 \tag{4.1}$$

Based on this equation, the address generator is composed of simple counters. The μ controller writes the counter parameters to configure arbitrary memory access patterns.

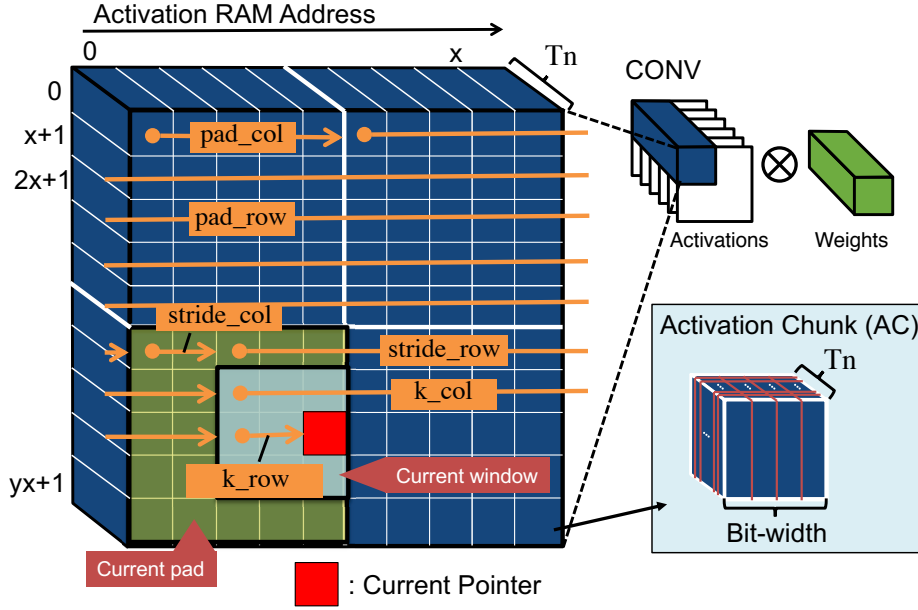


Figure 4.18: Memory access pattern of activation RAM in the CONV layer.

4.3.6 Mapping CONV and FC Layers on Unified PE Array

Figure 4.19 presents the task mapping patterns of the CONV and FC layers. The 24 parallel cores process the inference in a layer-by-layer manner, producing/reading intermediate results to/from the SRAMs, respectively. For a CONV layer, the weight values are reused multiple times such that they are shared by a cluster of multiple cores. Computations by shared weight values require different activation values; the activation values are divided in the pixel direction and distributed in the cluster. Each core in a cluster has all corresponding weight values on the on-chip or stacked SRAMs. Otherwise, as some typical layers in recent DNNs contain a narrow pixel with deep channels, the pixel-direction partitioning is not sufficient to efficiently exploit the hardware parallelisms. The computations of such a layer are also divided into output channel direction. In this situation, the cores in a cluster share the same activation value with different weight values. Meanwhile, for an FC layer, the output neurons are mapped onto all the cores evenly, requiring the all-to-all shuffling data-distribution patterns. Each core computes different output neurons from the entire shared activations and different distributed weight values. In both cases, the computation in a core must read the activations from the other SRAM vaults that are delivered through the TCI channels and the interconnects. The accesses are scattered across individual memory spaces, and the burst lengths are very short (1 to 4 for this

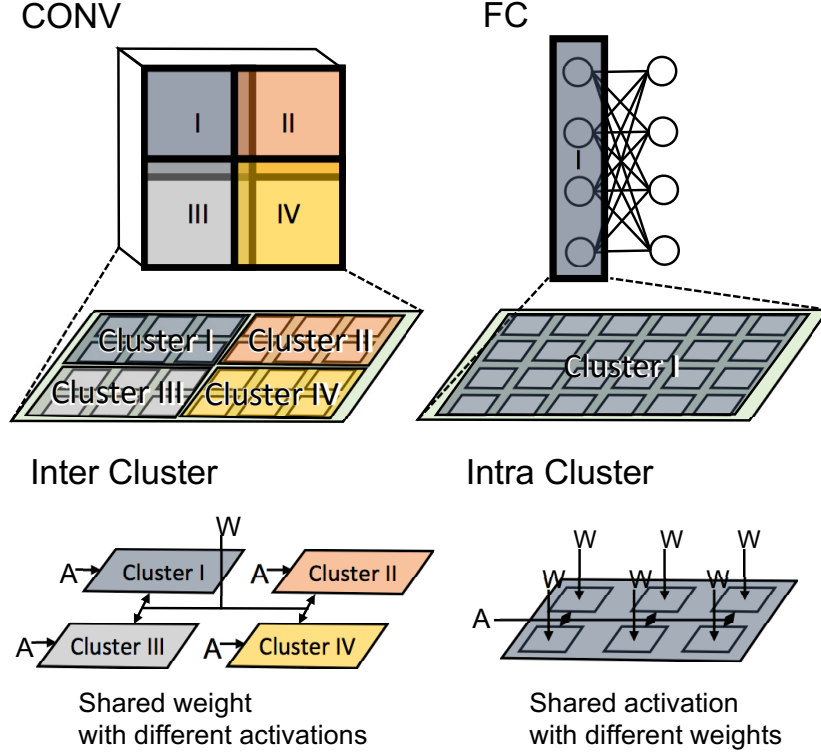


Figure 4.19: Clustering cores and data movement in the inter-cluster and intra-cluster.

mapping). To utilize the available hardware and energy resource efficiently, a task-mapping scheme on multiple computing nodes is critical.

4.4 Results and Implementation

In this section, we present the evaluations of the architecture characteristics based on the measurement results and the implementation results.

4.4.1 Setups

Our benchmarks are summarized in Table 5.2. We used the pre-trained DNN model for the ImageNet classification obtained from the Caffe Model Zoo [50]. It is noteworthy that the results do not include the computations of the input layer.

Figure 4.20 illustrates test running system on a test board for evaluations. Binary and log quantized DNN training environment is built upon TensorFlow with GPUs. Log-quantized DNN models for QUEST were obtained by fine-tuning from existing models or training from scratch. In both cases,

Table 4.1: *Benchmarks.*

Network	Benchmark	Bitwidth	Accuracy [%]	Required memory capacity [MB]
AlexNet [2]	ImageNet	4	76.72	38.0
VGG16 [3]			87.34	69.2
LeNet [1]	MNIST	1	97.23	0.6

parameters were trained through quantized forward propagation and normal backward propagation. The trained models were transformed into numerical data on the 3D SRAM and micro-controller programs. A QUEST module is evaluated by using a test board with a power supply, a clock signal generator, and a host computer connected via an external interface.

For evaluation, we implemented an in-house cycle simulator to estimate the performance of our proposed architecture. The simulator calculates the amount of clock cycles to be consumed by the computations and the data transfers.

4.4.2 PE Array Size

Figure 4.21 shows the performances in the cases of scaling the PE array size where the number of PE_COL per a core is x and the number of PE in the PE_COL is y . The total number of PEs is normalized by 12,288, such that the number of cores is also scaled under the total PE count constraint. Figure 4.21(a) presents the performances when the input activation parallelisms (y) are fixed to 32, and Figure 4.21(b) presents the performances when the output activation parallelisms (x) are fixed to 16.

According to the results in Fig. 4.21(a), output parallelisms of PE_COL slightly affect the performance. On the other hand, in Figure 4.21(b), a certain amount of input parallelism is essential to achieve high performance. However, excessive input parallelism does not contribute to performance improvement, owing to an increase in the required memory bandwidth.

Based on these results, the fabricated chip employs the near-optimal configurations of $x=16$ and $y=32$.

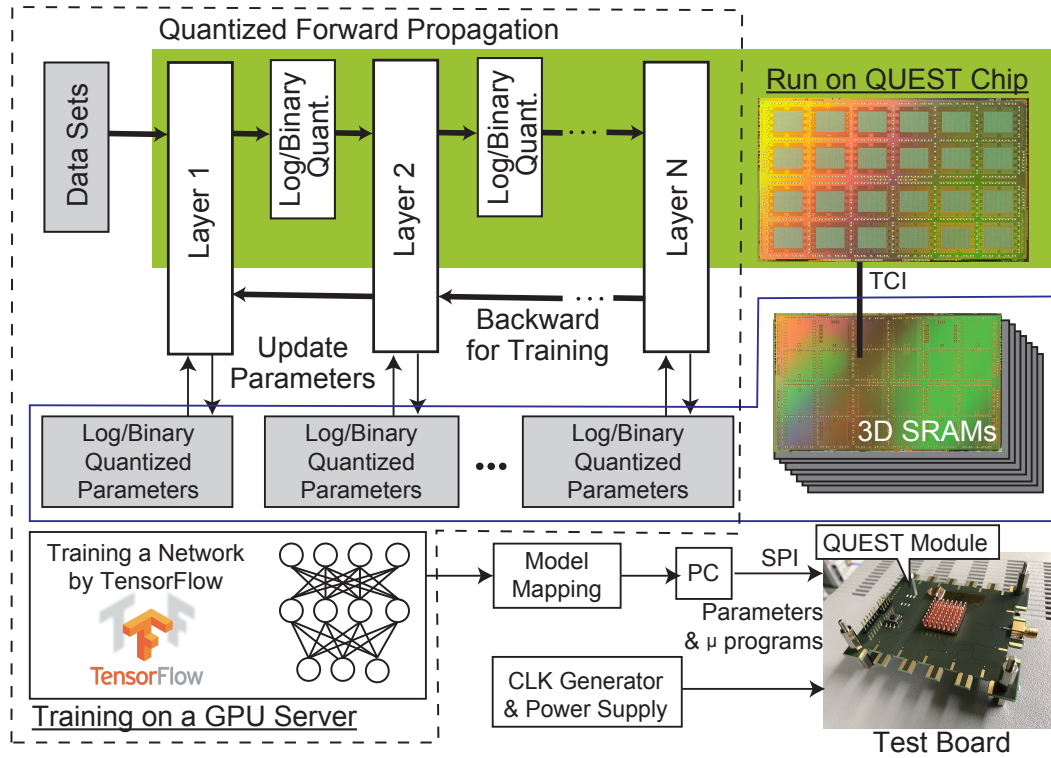


Figure 4.20: Test chip and setups.

4.4.3 On-chip Memory Capacity

Figure 4.22 shows the performances with different on-chip memory capacities. In the evaluation, the activation memory (A_MEM) and the weight memory (W_MEM) in each core are scaled. Figure 4.22(a) presents the performances when the capacity of W_MEM is scaled with a fixed capacity of A_MEM. The results indicate that the performances in AlexNet and VGG16 are affected by the capacity of W_MEM because they have a large amount of reusable weights. Therefore, a sufficiently large capacity of W_MEM is necessary for large DNNs.

Meanwhile, in Figure 4.22(b), scaled to the capacity of A_MEM, the performances in AlexNet and VGG16 are slightly affected because these complicated models generally have many activations for each layer. As the smaller capacity of A_MEM increases the amount of data transfers between the on-chip RAMs and 3D RAMs, the absolute performances reduce, even if double buffering is utilized

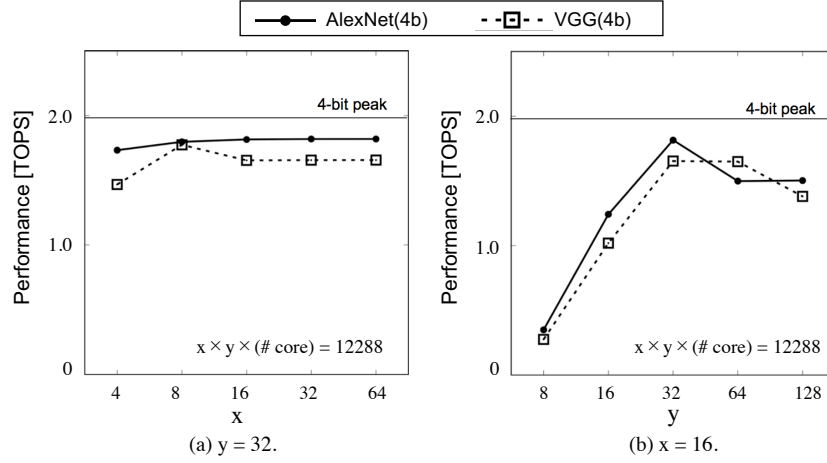


Figure 4.21: Evaluation results of PE array configuration (x : # of PE_COL, y : # of PE in PE_COL).

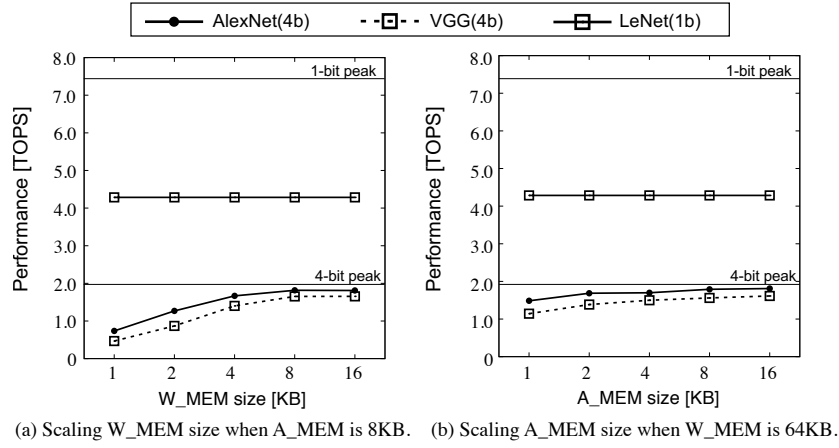


Figure 4.22: Evaluation results of a capacity of on-chip memory.

as shown in section V.A.3.

4.4.4 Effect of SRAM Short Latency

Figure 4.23 shows the evaluation result of the different types of memory access latencies for various types of DNNs.

Our proposed QUEST chip can access the 3D SRAMs uniformly with three-cycle latency. Meanwhile, the DRAM latency is assumed the burst memory access with a 30-cycle initial latency, which is estimated as an order of magnitude longer than the SRAM latency. This result shows that the long latency in DRAM actually affected the performance for all benchmarks. For more detail, Fig. 4.24

4.4. Results and Implementation

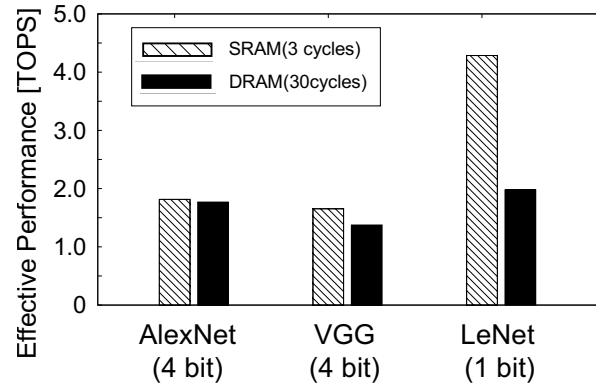


Figure 4.23: Performance comparison of SRAM and DRAM latency in different types of DNNs.

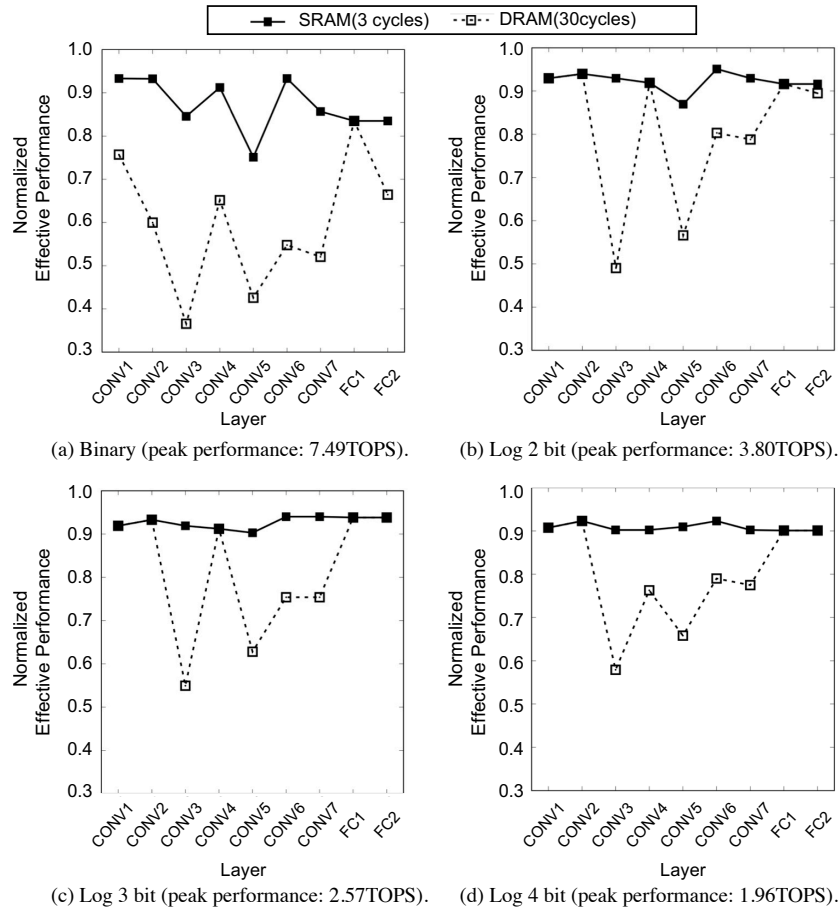


Figure 4.24: Performance comparison of SRAM and DRAM latency in VGG for CIFAR10 with different bit-widths.

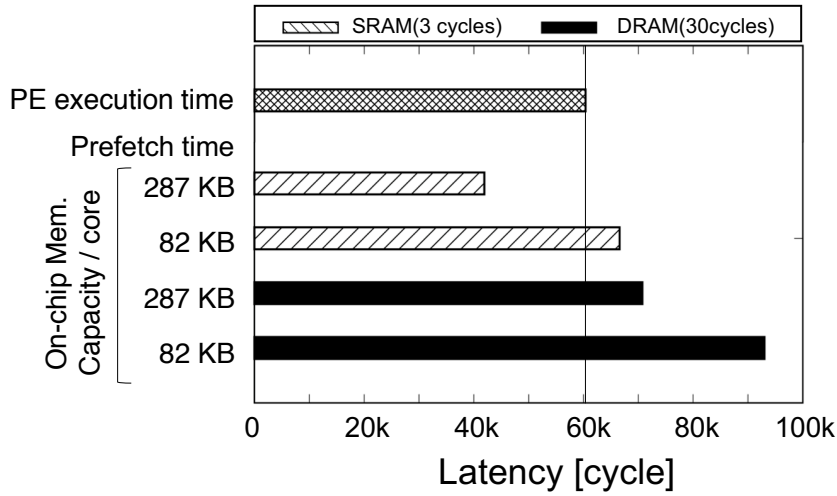


Figure 4.25: Time-chart of CONV5 layer operation in VGG at log 4 bit.

shows the layer-by-layer characteristics for various bit-widths in VGG for CIFAR10 [51]. From this result, the 3-cycle short random-access latency is indispensable for achieving highly-effective performance. It is also noticeable that the lower bit-width is more effective because lower bit-width data access produces smaller burst access than larger bit-width in serial processing. Larger DNNs such as ResNet, moreover, require aggressive pruning to fit the limited memory space, so the presented random-access capability of the 3D module will become even more indispensable.

Figure 4.25 shows a time chart of execution of PEs and prefetch from external memory in the CONV5 layer of VGG with 4 bit width. The figure compares SRAM and DRAM access latency, and the size of on-chip RAM for each memory. From the result, SRAM achieves a short prefetch time because of the uniformly short latency access. In contrast, DRAM cannot conceal the prefetch time due to a long access latency. Moreover, the result indicates that a sufficient capacity of on-chip RAM helps to conceal the prefetch time. When the size of the on-chip RAM is smaller, the prefetch time increases due to limited access length. DRAM requires a larger on-chip RAM than SRAM because DRAM takes advantage of the long burst access. The large on-chip RAM squeezes the area of PE array. Therefore there is a tradeoff between latency and on-chip RAM size. This fact affects the performance under the limited area devices.

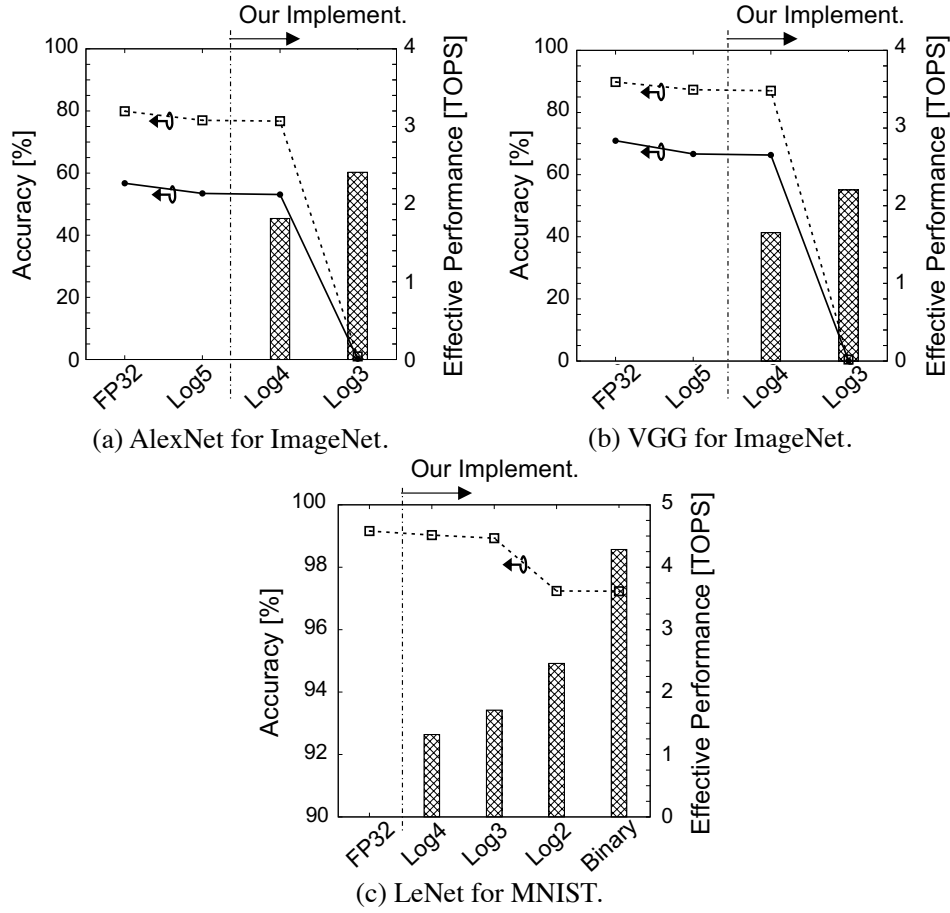


Figure 4.26: Accuracy and performance results of different types of DNNs.

4.4.5 Performance and Accuracy

Figure 4.26 shows the trade-off between performance and accuracy. The log-quantization inference accuracy on both log-4 (4bit log-quantized) AlexNet and VGG for ImageNet shows only marginal degradations compared with FP-32, whereas Log-3 was destructive. Meanwhile, even a binary representation can attain a reasonable accuracy for LeNet-5 for MNIST [53]. Therefore, we can select the optimal bit precision to achieve the level required by the application.

4.4.6 Cluster Size

Figure 4.19 shows the performances of the different cluster sizes using log-4 AlexNet. Cluster size is the number of cooperating cores that share the activation values, as described in section IV.B. The results indicate that the performance increases when multiple cores are clustered because the

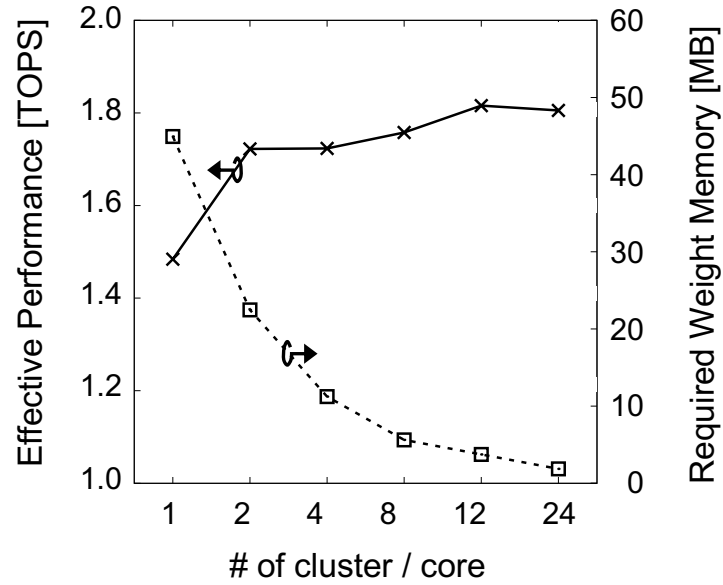


Figure 4.27: Performance and required memory capacity when scaling cluster size in all CONV layers in AlexNet.

computations of the output channels (M in Fig. 2.1) are unrolled more efficiently. The weight values can be duplicated for multiple cores in a cluster and redundantly stored in advance on the 3D SRAMs of all the cluster cores for performance improvement. Therefore, the memory capacity of the weights will be sensitive to the performance when the cluster size is increased. If the cluster size is excessively large, the amount of data transfers among the cluster cores is increased, as shown in Fig. 4.19. Thus, performance improvement by clustering is limited by the interconnection and the bandwidth of the on-chip RAMs.

4.4.7 Power Breakdown Based on the Measurement Results

Figure 4.28 shows the breakdown of power consumption and its performance results based on the measurement when processing AlexNet. The power is broken down for the static power of the whole chip, CLK and miscellaneous circuits, TCI data communications, PE arrays, access power of the on-chip memories and controllers which include μ controllers and DMACs. The results are scaled by the number of used cores. QUEST cores consume over half the total power when the number of used cores is 24. From this graph, a trade-off between the power consumption and performance is observed. As the number of used cores increases, the performance improves, but the power consumption of the PE

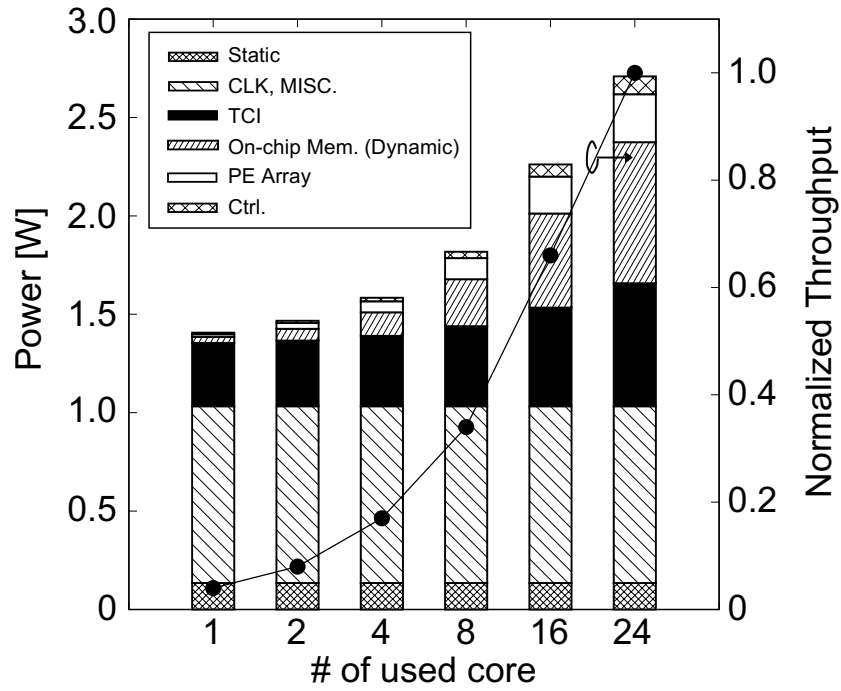


Figure 4.28: Power breakdown of QUEST chip and its performance result in AlexNet at 300 MHz and 1.1 V when scaling the number of used core.

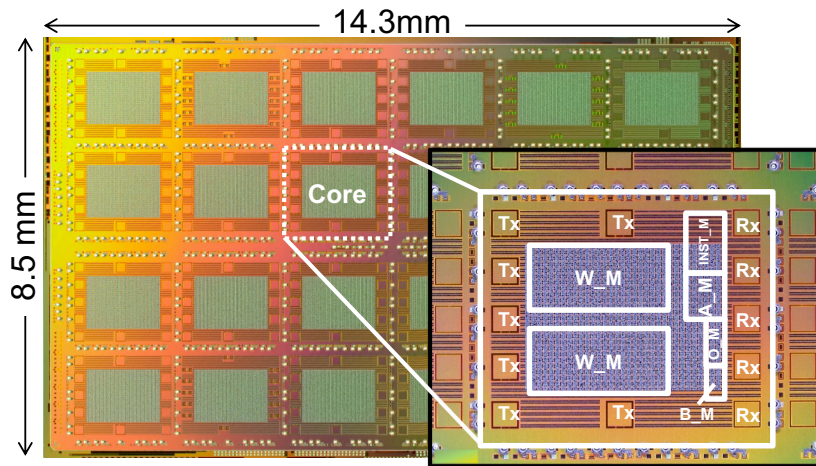


Figure 4.29: Chip microphotograph and layout overview.

array and TCI also increase. Since QUEST is a MIMD processor, this trade-off can be easily selected by a programmer.

Chapter 4. Log-quantized Neural Network Engine with Stacking Technology

Table 4.2: Comparison with state-of-the-art general purpose DNN accelerators.

	ISSCC2016 [26]	ISSCC2017 [46]	ISSCC2017 [52]	VLSIC2017 [47]	This work
Technology [nm]	65	65	28	65	40
Core Area [mm ²]	3.5×3.5	4.0×4.0	1.29×1.45	3.8×3.8	14.3×8.5
Bit-precision [bit]	16	4/8/16	4/8/16	8/16	1 - 4
Target	CONV	CONV/FC/RNN	CONV	CONV/FC/RNN	CONV/FC/RNN
Quantization	Linear	LUT-based	Linear	Linear	Logarithmic
Clock Frequency [MHz]	100 - 250	50 - 200	25 - 200	10 - 200	75 - 330
External Memory	Not Discussed				3D-stacked SRAM
Peak performance [TOPS]	0.084	1.2@4b (CONV) 0.025@4b (FC)	0.102@16b 0.408@4b	0.410@8b	1.96@4b 7.49@1b
AlexNet Benchmark					
Voltage[V]	1.0	1.1	1.0	1.2	1.1
Clock Frequency [MHz]	200	200	200	200	300
Bit-precision [bit]	16	4	4	4	4
Performance [TOPS]	0.046	1.088	0.076	0.368	1.825
Power [W]	0.278	0.279	0.044	0.29	2.083 (w/o TCI coils)
Energy Efficiency [TOPS/W]	0.165	3.900	1.727	1.269	0.877
Top-5 Accuracy [%]	Not Shown				76.7 (-3.2% from FP32)

4.4.8 Implementation and Comparison with State-of-the-art General Purpose DNN Accelerators

Figure 5.7 shows a QUEST prototype microphotograph and the layout. The chip is fabricated with UMC 40-nm low-power technology. The peak performance of the chip is 1.96 TOPS at 4 bits and 7.49 TOPS at 1 bit, respectively.

Table 4.2 compares the recently reported multipurpose (CNN/FC/RNN, etc.) DNN accelerators [26], [46], [52] and [47], using LUT-based or linear quantization, with this work. For a fair comparison, we use a same benchmark of AlexNet as other works. As a result, QUEST chip achieves 1.825-TOPS effective performance and consumes 2.083-W power, corresponding to an energy efficiency of 0.877 TOPS/W. The effective performance achieves 1.7-24.0× better than other works at 4 bit precision. However, the energy efficiency is a little lower than the state-of-the-art works due to the large die size. Actual processing power of PE array is 9% of the total power as shown in Fig. 4.28. The other works integrated a limited capacity of on-chip SRAMs (approximately 300 kB) and did not

Table 4.3: *Energy comparison between SRAM and DRAM.*

	Energy [pJ/access]
100MB SRAM	86.9
100MB DRAM	204.9

include the external memory for power estimation. QUEST, meanwhile, includes a 7.68-MB large on-chip SRAM (sufficient for AlexNet on-chip buffering) in the power measurement in addition to the 96-MB 3D SRAM.

Table 4.3 compares the energy consumption of a typical on-chip 100-MB SRAM and a DRAM. The energy is estimated by CACTI 6.5 [8] on a 32-nm process library. The SRAM read access energy is more efficient than the DRAM under the same technology and capacity. As external memory accesses are responsible for the majority of the power dissipation, and as the 3D SRAM can substantially reduce the external memory power in comparison to DRAMs, the system-level energy efficiency favors the proposed solution.

4.5 Conclusion

QUEST is aimed toward rapidly revolutionizing highly compressed (bit-reduced, pruned, etc.) DNNs with three main architectural features: 1) 3D integration with large capacity/bandwidth yet low-latency random access SRAM, 2) flexible dataflow support in the PE array for CONV/FC and other types of DNN layers, 3) a bit-serial PE architecture for binarized and log-quantized DNN representations. It achieves 7.49 TOPS peak performance in binary precision and 1.825 TOPS for the actual AlexNet processing. The results indicate that logarithmic quantization and bit-serial processing exhibit highly-effective performances in both the CONV and FC layers with efficient flexibility to balance the computing performance and recognition accuracy. Additionally, the results revealed that stacking SRAMs provided high energy efficiency compared to stacking DRAMs. Furthermore, we implemented our proposed architecture on the fabricated chip to verify our simulation results.

CHAPTER 5

Latency Evaluation for 3D integrated Memory

5.1 Introduction

Various emerging intelligent applications are being brought about by the recent rapid advances in machine learning technologies, particularly the use of deep neural networks (DNNs) [13, 40]. Modern DNN models are shifting toward deeper and more complicated structures for achieving higher accuracy and robustness in a wide range of applications [5, 41]. These types of DNN models typically require high computing power compared to other simple machine-learning approaches. Therefore, the hardware-based acceleration of DNNs is emerging in both cloud and edge environments.

The accelerators typically store the huge amounts of data required for DNN processing on external DRAM. Therefore, several approaches have attempted to utilize large on-chip buffers [54] to minimize the read/write access frequency of DRAM, and hence increase the performance while reducing the energy consumption. However, the improvement in performance is limited by the relatively narrow off-chip DRAM bandwidth. Obviously, 3D-stacked memory systems are attractive candidates for

Chapter 5. Latency Evaluation for 3D integrated Memory

realizing both high bandwidth and large capacity at the same time.

In DRAM-based 3D-stacked memory, a DRAM controller has the responsibility to manage all read and write requests and refresh operations at high throughput. Thus, DRAM access usually takes several tens of clock cycles, and the access latency cannot be accurately predicted. Owing to this long access latency, which is strongly dependent on the locality of the requested data, DRAM-based 3D memory is also ineffective when data access patterns are scattered across a wide memory space. Conventional solutions exploit a memory hierarchy, that is, they employ long burst transfers in combination with a large on-chip SRAM buffer, such as a scratchpad or cache memory, for hiding the long latency.

In DNN processing, intermediate data, such as neuron activation, are short and sparse. Moreover, recent DNN optimization techniques, such as pruning [22], tend to enhance the sparsity, which requires random fine-grained memory access in a wide working set. Using this type of situation, the aforementioned latency hiding technique does not operate properly. In contrast, trying to integrate a larger on-chip SRAM buffer to hold the entire working set sacrifices the performance by limiting the resource allocation for the processing elements.

In this study, we propose a novel DNN accelerator using SRAM-based 3D-stacked memory. Needless to say, the SRAM access latency is an order of magnitude shorter than that of DRAM access. In addition to being short, SRAM read/write access is achieved through a simple single command, featuring a flat latency over the entire memory space, and allowing fine-grained randomized access without incurring additional latency. No refresh operations are required as well. That is, in clear contrast to DRAM, SRAM allows frequent data transfers of fine-grained data that can be fully controlled statically using computation threads.

Finally, we developed a prototype LSI based on the proposed architecture. High memory bandwidth and a short latency play important roles in improving the absolute performance and energy efficiency in recent accelerators. As a challenging feature, the accelerator is associated with 3D-stacked SRAM with an inductive coupling interface [45].

The main contributions of this study are as follows:

- We first evaluate the impact of memory access latency in general DNN computing.
- To implement large DNN models into the SRAM-based 3D-stacked memory, we evaluate the quantization method of different numerical representations considering both the hardware computation performance and recognition accuracy.

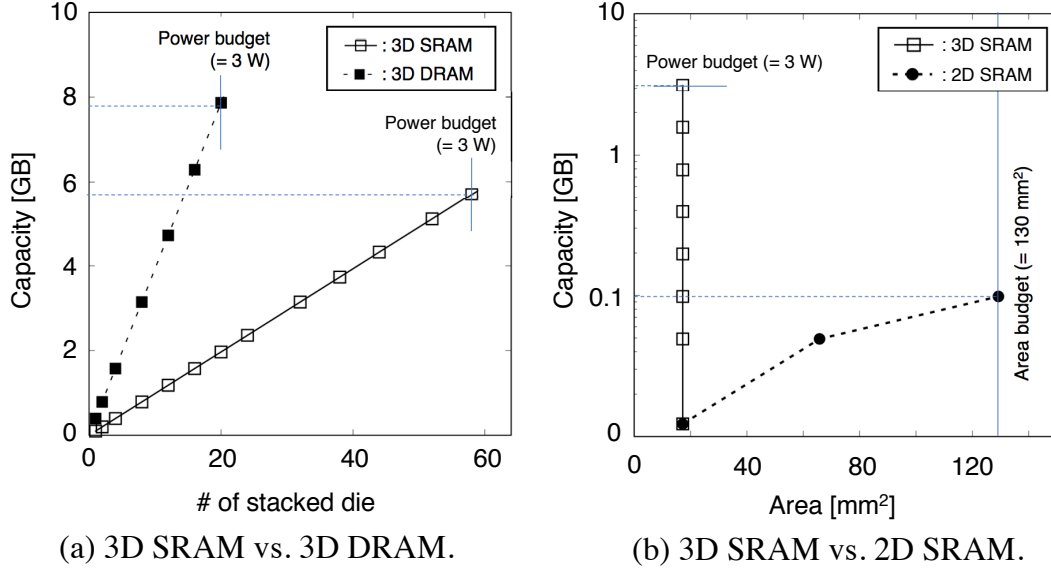


Figure 5.1: Capacity comparisons of 3D SRAM using CACTI 6.5 [8].

- Based on the above discussion, we propose a novel DNN architecture by employing SRAM-based 3D-stacked memory and log-quantization for high energy efficiency and a low memory-capacity demand.
- We also present the prototype LSI of the proposed architecture using 3D-stacked SRAM.

5.2 Impact of memory access latency

Recently, DRAM-based 3D-stacked memory modules are adopted to obtain a high bandwidth and large capacity. However, it is well known that DRAM consumes much more energy than SRAM. Fig. 5.1 (a) shows the number of stackable 3D DRAM dies and 3D SRAM modules under the same power budget constraint. The number of stacked dies is limited owing to the thermal issue, which is proportional to the power consumption. Whereas the capacity of a single DRAM die is larger than that of typical SRAM, the number of stackable 3D DRAM dies is small owing to the power budget restriction. In contrast, although the capacity of a single SRAM die is smaller than that of DRAM, the number of stackable SRAM dies is higher than that of DRAM. Therefore, there is no critical capacity gap between 3D SRAM and 3D DRAM under a power budget constraint. Fig. 5.1 (b) shows a capacity comparison between 3D SRAM and 2D SRAM. Conventional 2D SRAM clearly requires a larger area

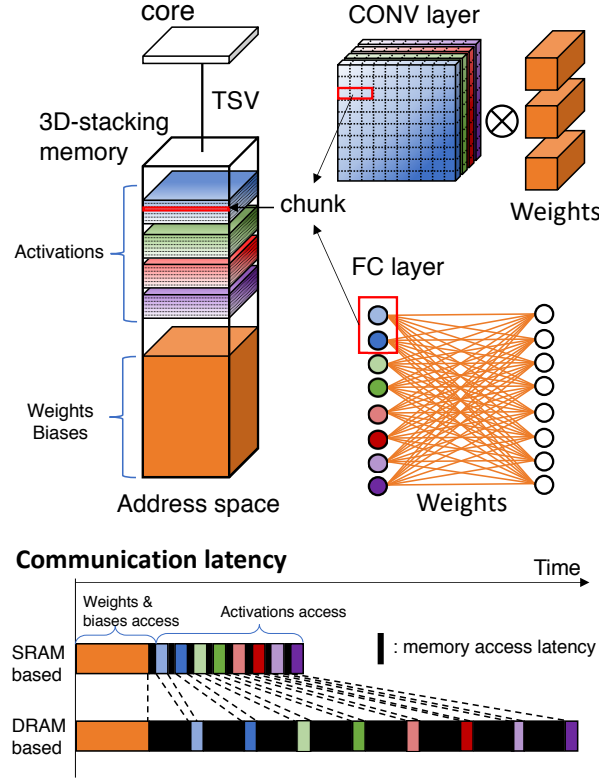


Figure 5.2: Two data categories (activations and weights/biases) in DNN processing: Their access patterns and the influence of memory access latency.

than 3D SRAM. The large area decreases the yield rate. Thus, 3D SRAM is a more feasible solution to support both a sufficient capacity and low latency.

5.2.1 Memory Access Pattern

In DNN inference computing, we can identify two types of memory access patterns, as shown in Fig. 5.2. One is for pre-stored parameters such as weights and biases. These parameters are static for inference, and thus we can store them into memory in a conveniently ordered manner prior to running a program. This means they can be accessed in sequence, and the latency is hidden by burst accesses even if we use DRAM. In contrast, intermediate data of neuron activations are dynamically produced. These data are read/written in a scattered manner from/to multiple small chunks during DNN inference computing. In addition, low bit-width quantization shrinks the size of the data transfer chunks, whereas long-burst transfers are crucial for exploiting the maximum bandwidth of DRAM.

In contrast, the memory latency of SRAM is independent of the burst length. Thus, SRAM is highly suitable for quantized neural network processing. As shown in Fig. 5.2, both types of layers, namely, a convolution (CONV) layer and a fully connected (FC) layer, have a small chunk of activations of neurons. In a multicore DNN processing system, each core outputs its own activations of neurons, and stores them to its own memory in an individually controlled manner. Hence, the sizes of the data chunks tend to become fine-grained. This behavior will be more enhanced in pruned DNNs that bear sparse networks. Simplified access latency can be represented through the following formula.

$$\begin{aligned} & \text{Access latency per data} \\ &= \{(\text{access cmd. latency}) + (\text{sequential length}) \\ & \quad + (\text{mem. access latency})\} / (\# \text{ data}). \end{aligned} \tag{5.1}$$

The formula shows that the latency is lengthened by the short-burst random accesses, which in turn causes DNN computing stalls and reduces the performance.

5.2.2 Baseline Architecture

We first define a baseline architecture used for evaluating the DNN processing performance with different memory access latency. Fig. 5.3 provides an overview of our baseline architecture. The accelerator is constructed as a 3D-stacked module made up of a DNN processing die and multiple memory dies. All dies are tightly connected using conventional TSVs. The DNN die on top of the 3D-stacked module employs a multicore architecture of processing element (PE) arrays, which process multiply and accumulate (MAC) operators. The behavior of each DNN core is completely managed by its own microcontroller. The microcontroller is a lightweight RISC processor that supports a few types of instructions to manage the sequencer and DMA controller (DMAC). The microcontroller configures the overall computations on the PE array and memory accesses through 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 stacked memory modules through the TSV interface. The chip has interconnection links, called local links, for communication among neighbor cores.

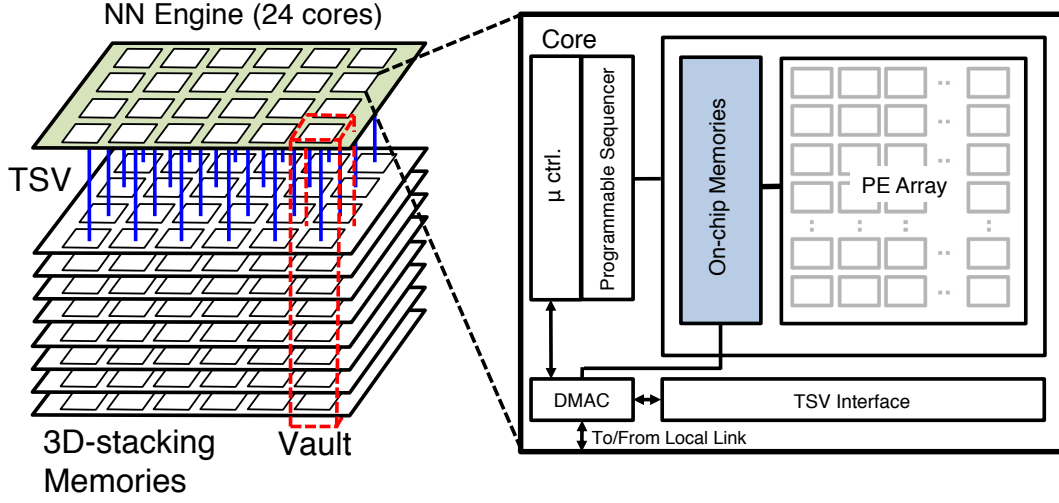


Figure 5.3: *Baseline architecture.*

5.2.3 Effects of Memory Access Latency

Fig. 5.4 summarizes the evaluation of the execution time when the memory access latency varies for various types of DNNs. We compared ideal latency (one cycle/data), SRAM latency (three cycles/data), and emulated DRAM latency (30 cycles/data) cases for 3D-stacked memory, when the cores function at a clock frequency of 500 MHz, as estimated from Table 5.1. The number of PEs and the size of the on-chip buffers in this comparison are estimated using a Synopsys Design Compiler [36] with 40-nm UMC technology under the same resource construction: 16 PEs, with each PE containing 660 kB of memory, and 32 PEs containing 330 kB memory each. We can see that the DRAM-based long access latency affects the execution time in every type of DNN computation. The influence becomes more apparent when the size of the per-PE memory becomes smaller because the accesses to 3D-stacked memory become more frequent.

The short SRAM access latency enables uniform access without an unpredictable hitmiss latency. This means that all access patterns are predictable and can be placed under fully static control. Hence, the programmable sequencer can be aggressively optimized without complicated access patterns such as monitoring the access failure caused in DRAM access. In addition, the DRAM access energy consumption is 45 times higher than that of a conventional SRAM, as shown in Table 5.1. These results motivated us toward the use of 3D-stacked SRAM instead of DRAM.

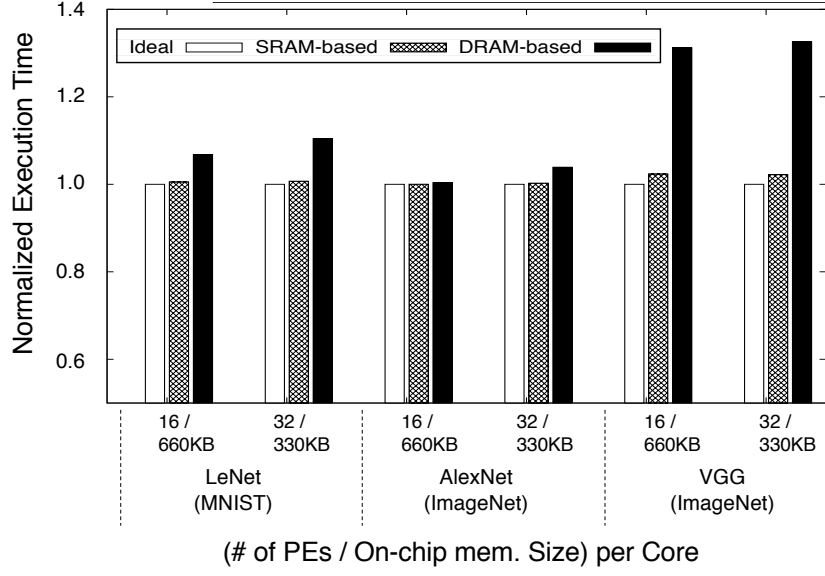


Figure 5.4: Latency evaluation.

Table 5.1: Latency and energy comparison between SRAM and DRAM.

	Latency [ns]	Energy [pJ/access]
2Mb SRAM [8]	0.84	4.44
16Gb DRAM [55]	45	201.0
	$(t_{RCD} + t_{RP} + t_{CL})$ [56]	

5.3 Evaluation

To reveal the advantage of our proposal, we evaluated the performance, energy consumption, and hardware resources of several hardware configurations. Table 5.3 shows the baseline configuration of these evaluations.

5.3.1 Methodology

We selected several typical DNN models and benchmarks, including AlexNet [2] and VGG16 [3] for ImageNet, and LeNet [1] for MNIST [53] (Table 5.2). These DNN models consist of both CONV and FC layers. We used a pre-trained DNN model for the ImageNet classification obtained from the Caffe Model Zoo [50]. Note that the evaluation results do not include computations of the input layer. We used an in-house simulator to evaluate the performance and energy consumption of the

Chapter 5. Latency Evaluation for 3D integrated Memory

Table 5.2: *Benchmarks.*

Network	Benchmark	Bit-width	Accuracy [%]	Memory cap. [MB]
AlexNet [2]	ImageNet	4	76.72	38.0
VGG16 [3]			87.34	69.2
LeNet [1]	MNIST	1	97.23	0.6

Table 5.3: *Hardware configuration.*

Factor	Baseline
Area / core	$2.5mm^2$
# of core	24
Clock frequency	500MHz
3D memory bandwidth	28.8GB/s
Ideal 3D mem. latency	1 cycle
3D SRAM latency	3 cycles
3D DRAM latency	30 cycles

proposed architecture. The simulator simulates the circuit behavior in a cycle-accurate manner, and thus provides several performance statistics, such as the number of execution clock cycles and its breakdown of the computations and data transfer. Based on the performance statistics and energy consumption of each element, we evaluated the overall energy consumption of the system. The energy consumption of each circuit primitive was estimated based on an RTL design using Synopsys Design Compiler [36] in 40-nm CMOS technology. The characteristics of the memory components, such as the SRAM and DRAM modules, were evaluated using CACTI6.5 [8] and a DDR4 system-power calculator [55] for a fair comparison. We compared three types of 3D memory integrations: an ideal situation in which the memory access latency to any locations is always a single cycle (Ideal), a situation in which the proposed SRAM-based 3D integration can read and write data in three cycles (Tx, R/W, and Rx) (SRAM), and a conventional situation of DRAM-based 3D integration, which assumes that each memory access requires 30 cycles (DRAM), as estimated in Table 5.3.

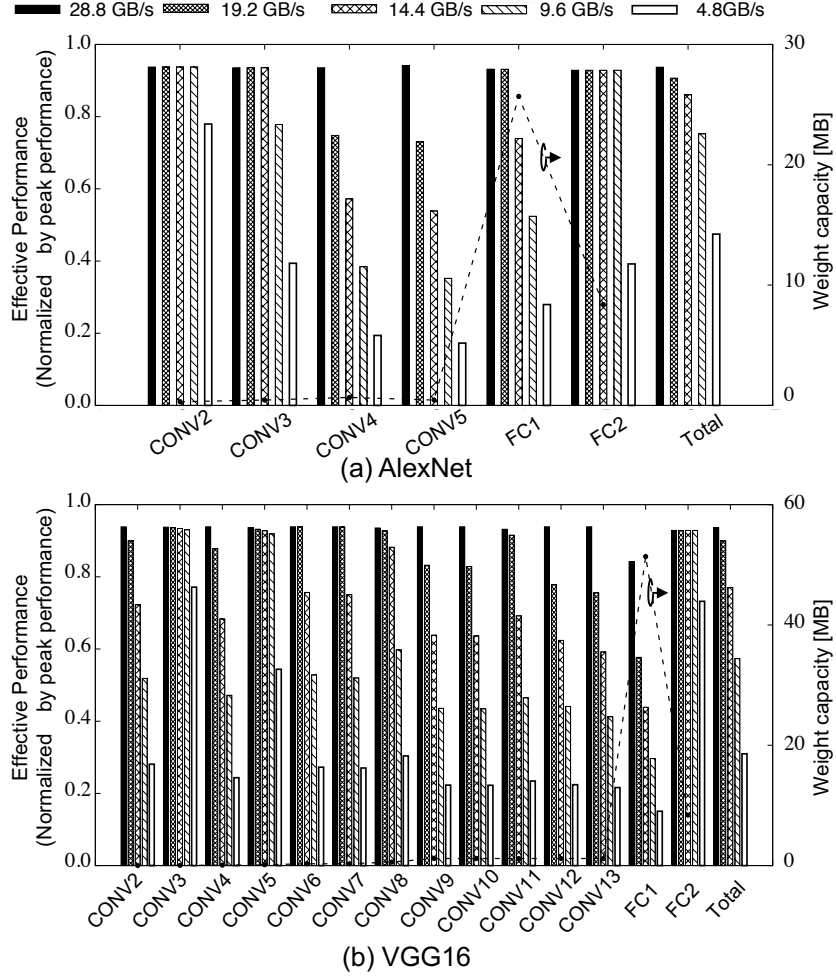


Figure 5.5: Performance comparison of each layer in AlexNet and VGG16 with respect to different memory bandwidths of 3D SRAMs.

5.3.2 Bandwidth of 3D Stacking Memory

External memory bandwidth is a crucial factor for the performance of DNN accelerators. Each of CONV layer and FC layer has different properties represented by a roofline model. While CONV layer generally reuses more data of activations and weights than FC layer, so that memory bandwidth demand of CONV layer is lower than FC layer, sufficiently bandwidth helps to achieve higher performance in both CONV and FC layers.

Fig. 5.5 shows the performance fluctuations by AlexNet and VGG16 by scaling the available memory bandwidth of SRAM-based 3D stacking memory. In the both the networks, the performance of the deeper layer is decreased due to the higher amount of the total data transfer for weights of each

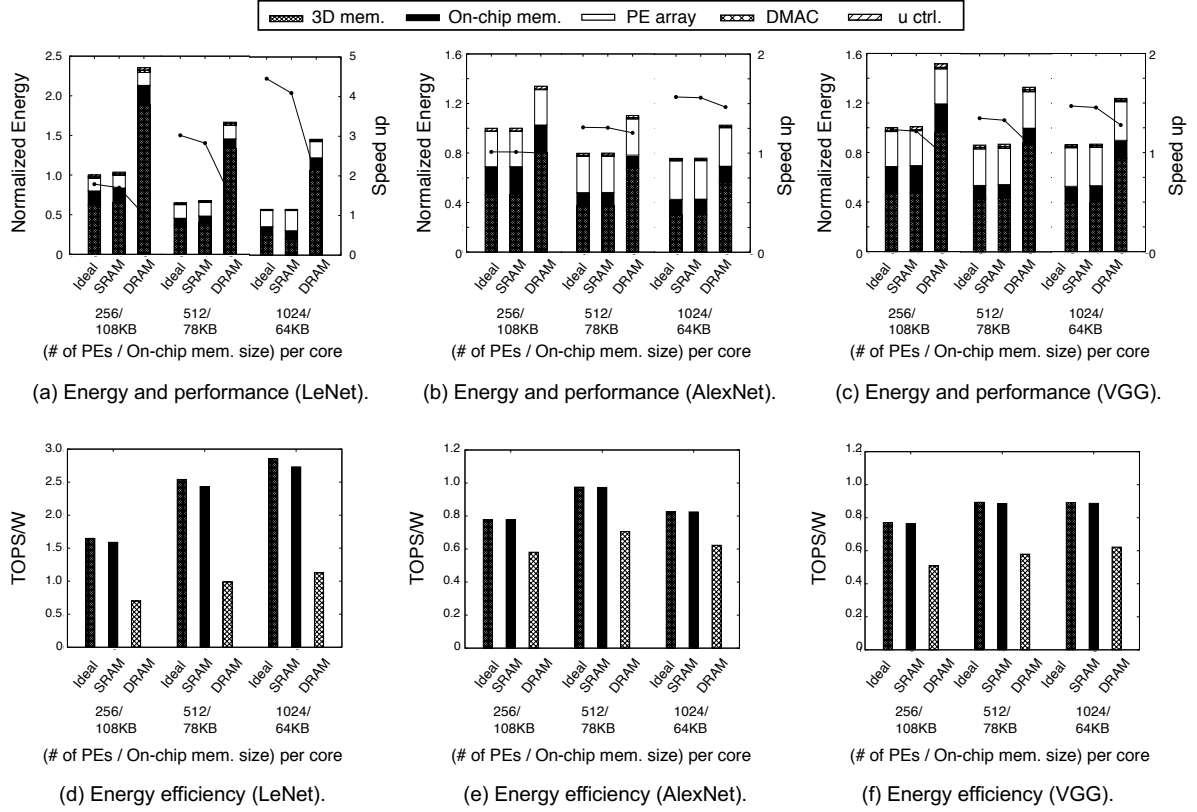


Figure 5.6: Energy consumption (left-axis, top row), execution time (right-axis, top row), and energy efficiency (bottom row) with different sized PE arrays and on-chip memory per core.

layer. Thus a low memory bandwidth affects the performance badly even if double buffering method is employed to hide data transfer overheads. As FC layers are just large matrix multiplications that have less reusability of data than CONV layers, the performance of FC layers is obviously restricted by the memory bandwidth.

These discussions finally indicate that providing a sufficient memory bandwidth is crucial for relatively high performance in any kinds of layers. As described, memory access latency will become an important factor to further improve the performance.

5.3.3 Performance and energy consumption

Fig. 5.6 (a)-(c) show the energy consumption of each hardware element, and the processing performance of each evaluation configuration, namely, LeNet of binary mode, and AlexNet and VGG of 4-bit logarithmic mode, respectively. We evaluated the efficiency of several hardware configurations

Table 5.4: Comparison of performance and energy efficiency when using a GPU (AlexNet) [11, 12].

	Titan X (FP32)		Tegra X2 (FP16)		This work (LOG4)
Batch	1	128	1	128	1
Power[W]	164.0	227.0	2.3	3.2	1.94
TOPS	0.586	4.657	0.172	0.615	1.890
TOPS/W	0.004	0.021	0.075	0.192	0.974
Efficiency comparison	272.44x	47.49x	13.00x	5.07x	1x

with different numbers of PEs and on-chip buffer capacity under die area constraints.

In all configurations, the energy consumption of the 3D memory occupies a majority of the total consumption. In particular, DRAM-based 3D memory consumes more energy than SRAM-based 3D memory, and uses over 65% of the total energy consumption in all configurations.

In addition to an energy drawback, the DRAM-based approach degrades the computing speed owing to its long latency, as described in section II. This characteristic is enhanced when the on-chip memory capacity decreases because the amount of data traffic between 3D memory and on-chip memory increases.

Fig. 5.6 (d)-(e) show comparisons of DRAM- and SRAM-based 3D integrations in terms of their performance and energy efficiency. In all DNN models applied in this evaluation, the decrease in performance from the ideal performance of SRAM-based integration is quite small. In contrast, the decrease in performance of the DRAM-based approach is significant owing to the longer memory access latency.

Table 5.4 summarizes the comparison results when using a conventional and embedded GPU in terms of the performance and energy efficiency of AlexNet computing. Note that the proposed architecture does not support a batch size of over 1 because it is optimized for real-time processing on an embedded system. For a fair comparison, we employed the same batch size as in the proposed architecture. The numerical representation of the proposed architecture is quantized in 4-bit log, which results in almost the same recognition accuracy as ImageNet classification, as described in section II. Because a conventional GPU (Titan X) requires a large power budget, the proposed architecture achieves over 270 times greater energy efficiency. Compared to a modern embedded GPU (Tegra

Chapter 5. Latency Evaluation for 3D integrated Memory

Table 5.5: Performance and energy efficiency comparison with state-of-the-art ASICs (AlexNet).

			[57]'17	[42]'16	[58]'18	This work
Technology[nm]			65	28	65	40
Bitwidth[bits]			16	16	8	4
Quantization			Linear	Linear	Linear	Log
Voltage [V]			1.0	1.2	1.2	1.1
# of PEs			168	16	512	12288
TOPS			0.046	0.008	0.37	1.89
Power [W]	Ext.	w/	N/A	1.86	N/A	1.94
	Mem.	w/o	0.28	0.25	0.29	1.02
TOPS/W	Ext.	w/	N/A	0.0004	N/A	0.974
	Mem.	w/o	0.165	0.032	1.269	1.853

X2), the proposed architecture achieves 13 times greater efficiency. Note that the power value of an embedded GPU does not include any power consumption of the external DRAM, whereas that of the proposed architecture includes the power consumed by the SRAM-based 3D memory.

Table 5.5 shows a comparison with state-of-the-art ASIC designs that can compute both the CONV and FC layers. Our architecture consists of massively parallel PEs and achieves an order-of-magnitude higher throughput against the other implementations. Because the PEs of our architecture are composed of a multiplier-free bit-configurable circuitry using log-quantization, more PEs can be installed within a limited circuit area. Because many other designs do not include the power consumption of external memory, for a fairer comparison, we also showed the power consumption of the external memory is omitted. Our proposed design achieves the highest energy efficiency because of the multiplier-free and bit-configurable microarchitecture achieved through log-quantization. Note that it achieved a 0.974 TOPS/W even when including the external memory.

5.3.4 Hardware implementation

To verify the feasibility of the proposed architecture, we developed a prototype chip based on this architecture. Fig. 5.7 shows a micro-photograph of the fabricated chip, and the chip specifications. The chip was fabricated using 40-nm UMC low-power technology. The peak performances of the

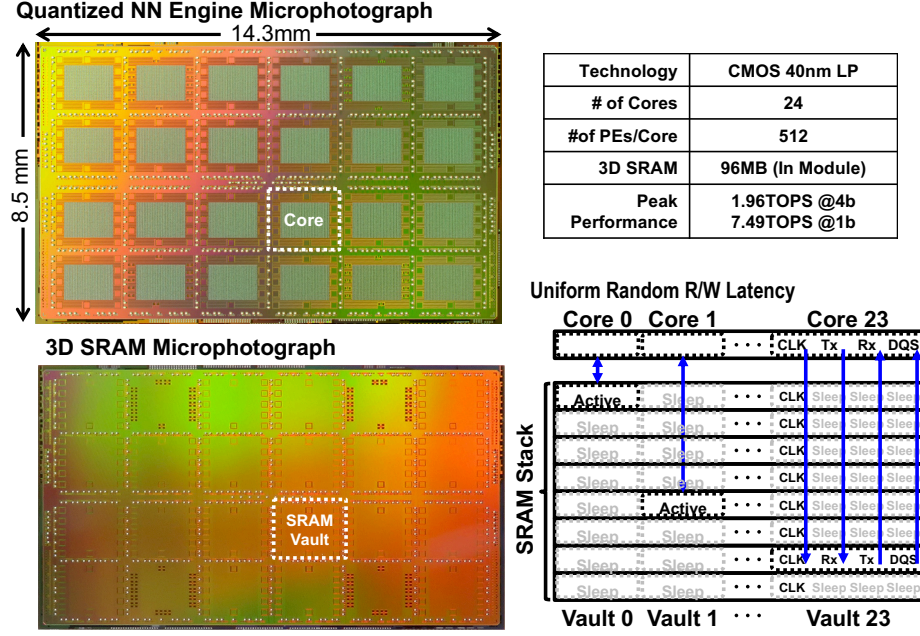


Figure 5.7: Microphotograph, diagram of TCI interface, and specifications of the fabricated chip.

chip are 1.96 TOPS in 4-bit logarithmic quantization mode, and 7.49 TOPS in 1-bit binary mode. The upper photograph shows the layout of the DNN accelerator chip, and the lower photograph shows the SRAM chip for 3D integration. These two types of chips are stacked, and are connected using a thru-chip interface (TCI) [45], as opposed to TSV. TCI is a wireless inter-chip communication technology with inductive coupling between multiple coils, and is well known for its smaller area required for the inductive coils, as compared to via holes. Thus, TCI achieves a better yield ratio than TSV. Note that multiple SRAM chips can be stacked, whereas only one DNN chip can be combined with SRAM chips and must be located on the top of the stack.

Each block of the DNN core or SRAM chip has a built-in TCI controller that independently communicates with other TCI controllers. The controller transfers data sequences at 3.6 GHz in a bit-serial manner. Thus, all 32-bit data are read out in three clock cycles at 300 MHz on the DNN chip, and the throughput is always one 32-bit data per clock cycle. Utilizing stacked SRAM chips provides a sufficient memory bandwidth and low access latency simultaneously.

We evaluated the thermal characteristics using [59] based on the performance statistics obtained from our in-house simulator. For 40-nm technology, the thermal effect is negligible because the chip consumes 1.94 W at 500 MHz, which increases the chip temperature by only 10 K at maximum.

5.4 Related Works

Neurocube [42] and TETRIS [60] proposed an NN accelerator using a 3D-stacked memory of HMC. These studies focus on the high-bandwidth feature of HMC without paying significant attention to memory access latency. Under a sufficient bandwidth, however, the impact of the memory latency becomes relevant. In our approach, SRAM modules are stacked instead of DRAM modules to reduce the energy consumption from the external memory under high bandwidth and short access latency.

5.5 Conclusion

This paper presented a novel approach for DNN accelerator architecture that uses logarithmic quantization along with 3D-stacked SRAM chips. Log quantization enables low area- and energy-consuming DNN processing and bit-serial processing for controlling the bit precision to obtain the highest energy efficiency under a high recognition accuracy constraint. We evaluated the proposed architecture in the various type of stacked RAM chips used in contemporary DNNs. The logarithmic quantization and bit-serial processing exhibit a highly effective performance in both the CONV and FC layers with efficient flexibility to balance the computing performance and recognition accuracy. The evaluation results also show that the stacking of SRAM chips can provide a high DNN processing performance as well as greater energy efficiency when compared to the stacking of DRAM chips, that is, both bandwidth and latency are important aspects in DNN inference processing. We also reported the implementation of our prototype module.

CHAPTER 6

Energy Efficient Inference by Skipping Inefficient Calculation

6.1 Introduction

Various emerging intelligent applications are being brought by the recent rapid advances in machine learning technologies, particularly the use of deep neural networks (DNNs). Modern DNN models require high computing power for the massive amount of the neurons. While many DNN accelerators have been developed for high performance and energy-efficient computing, there are still big demands for reducing the energy consumption of DNN computing to implement them on power-constrained embedded devices such as automotive and wearable devices.

For lower computation loads of DNN, there are a lot of works for compact model exploration [61]. However, they consume huge computation power to obtain a dedicated model for an application in a training phase. Pruning is a major approach to reduce model sizes and computation amounts of DNNs [22].

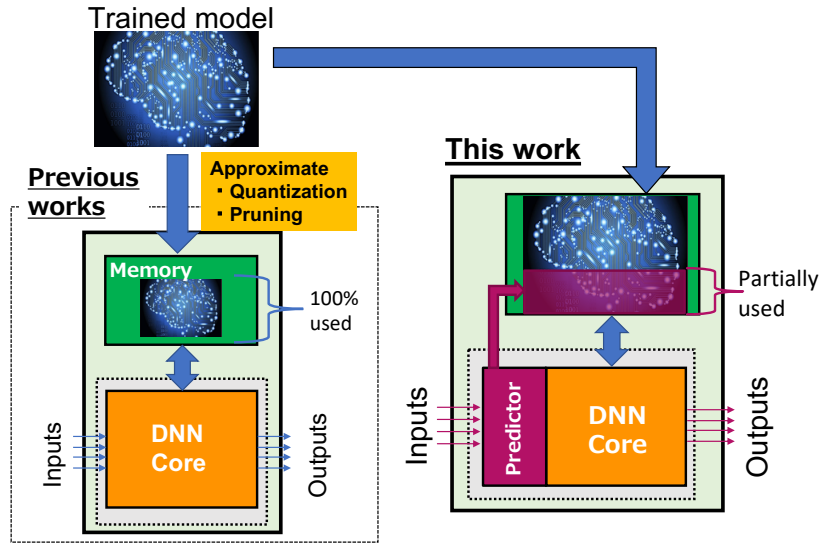


Figure 6.1: *Prediction-based DNN model.*

Even if a compact model is obtained by the model exploration and pruning, there is still a large space to reduce the amount of computations by exploiting dynamic behaviors of neurons. Most recent DNN models utilize a rectified linear function (ReLU) as an activation function, which filters the negative value of a MAC operation as zero. It generates a certain number of zero-output neurons. In this paper, we refer to them as dead neuron. Multiply-and-accumulate (MAC) operations for such dead neurons just waste energy and do nothing to its next layers.

If we know the dead neurons before the actual MAC operations, more aggressive computation reduction will be available by skipping such the wasted MAC operations. There are several studies for predicting results after ReLU to reduce the number of MAC operations. V. Akhlaghi et al. [9] proposed the early activation by re-ordering the multiplications of weights and activations to know the signs of accumulation results in advance. M. Song et al. [10] proposed the prediction method by calculating quantized high-order bits of weights. These approaches judge predicted signs in the middle of each output neuron computation by the early termination. So the computation time for each output neuron is not deterministic, and it brings some difficulties in scheduling and synchronization of massive-parallel processing elements.

In the conventional quantization and pruning, the calculation was reduced by compressing the model. However, actually, the calculated model is used all parameters (Fig. 6.1). Therefore, the purpose of this research is not to use all the learned parameters but to predict where to use them. As a

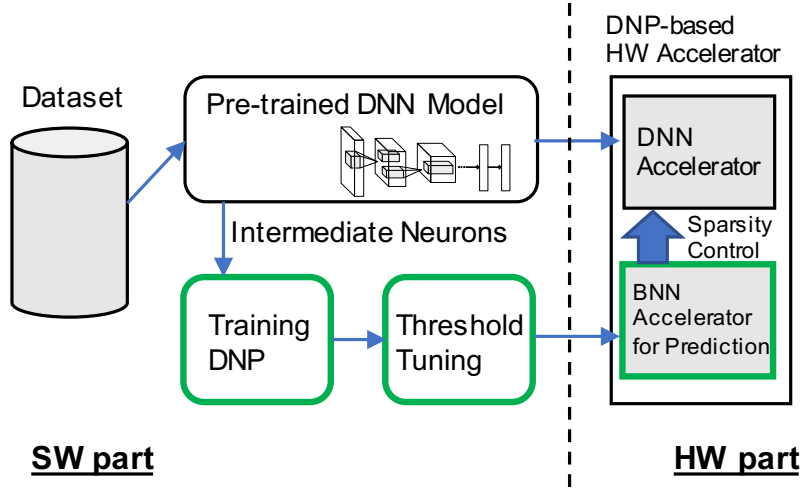


Figure 6.2: *Dead Neuron Prediction by SW/HW co-design.*

result, the amount of calculation can be reduced to a necessary minimum, and the memory access can be reduced.

We propose dead neuron prediction (DNP), a novel approach to predict dead neurons in advance and to utilize sparsity-aware DNN accelerators efficiently. The key idea is to employ a supportive lightweight neural network model and its tiny hardware for prediction of liveness. As a predictive model and hardware, binarized neural network (BNN) is used, which is composed of binarized weights and activations for small memory footprint and low hardware cost, and the BNN model is dedicatedly trained from an original pre-trained DNN model. DNP can skip entire computations for predicted dead neurons, and it achieves higher computation efficiency by a simpler hardware than the previous works which compute halfway. We evaluate only using the weights in the pre-trained DNN is not enough to predict the dead neurons, but the retrained BNN dedicated for predicting the sign of pre-activation value of ReLU is enough. Totally, DNP by BNN can aggressively reduce the computational cost.

We also propose the hardware accelerator for DNP. The DNP is composed of in-memory accelerator which achieves extremely low energy and suitable for BNN. The efficient-pipelined accelerator between sparse multi-bit DNN accelerator and in-memory DNP achieved highly energy efficiency.

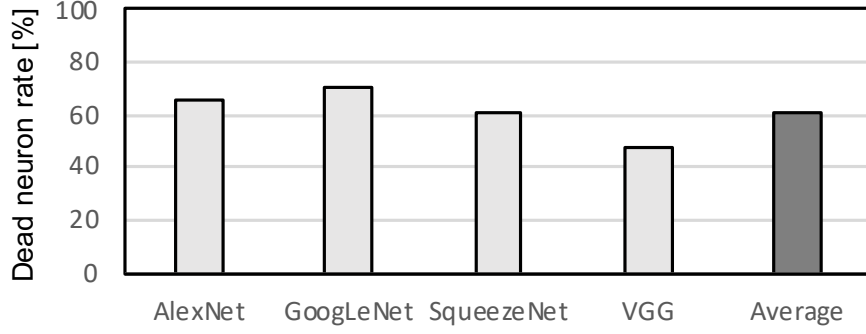


Figure 6.3: *Dead neuron rate [9].*

6.2 Dead Neuron Prediction

6.2.1 How do dead neurons appear?

Figure 6.3 shows the dead neuron rates at several type of convolutional neural network (CNN) for ImageNet classification task which is often used for the benchmark of DNN computation. The dead neuron is zero-activated neuron. Therefore, it is unnecessary for the CNN computing because it does not affect the calculation of next layer. In Fig. 6.3, over half of computation includes the wasted dead neurons in every modern CNN models. It means the skipping computation of dead neurons can significantly reduce the amount of computations of CNNs. In this subsection, we explain how the dead neurons appear in CNN computing.

Widely used DNNs have rectified linear unit (ReLU) as a activation function because it can make loss converge faster [62]. The dead neurons are occurred due to the ReLU function. The ReLU function is defined as follows:

$$f_{ReLU}(x) = \begin{cases} x & (x > 0) \\ 0 & (otherwise) \end{cases} \quad (6.1)$$

where x is a final accumulated value of pre-activation. When the value of x is negative, the ReLU filters x as zero. This is because the dead neuron does not affect the next-layer computations as shown in Fig. 6.4. Whether the ReLU activation is zero or not is determined by the sign of final pre-activated value. That is, if the sign of the pre-activated value (x) can be predicted, a large part of the calculations can be excluded.

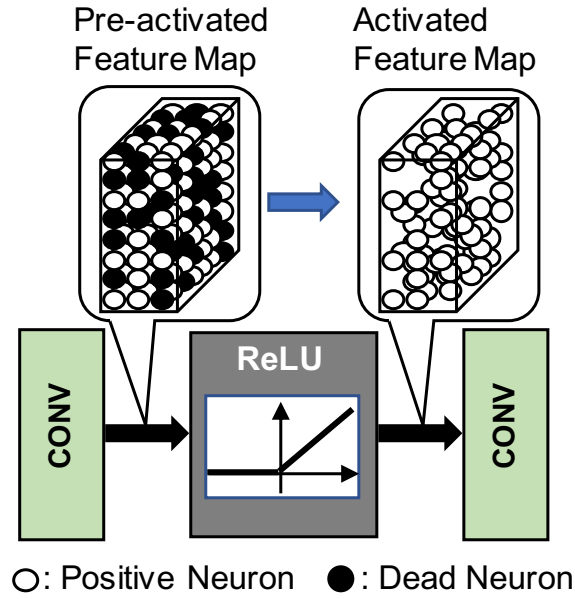


Figure 6.4: *Dead neurons produced by ReLU.*

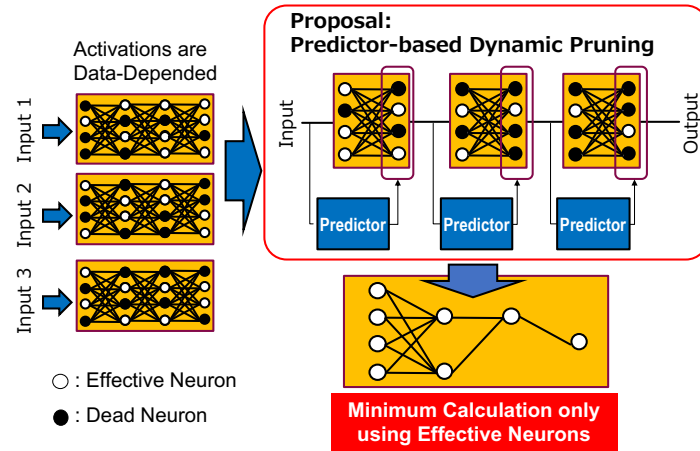


Figure 6.5: *Concept of Dead neuron prediction.*

6.2.2 Concept of Dead Neuron Prediction

As a new technique to reduce the amount of computation, we propose dead neuron prediction. Since the dead neuron does not affect the calculation of neurons in the next layer, the amount of calculation can be greatly reduced if the calculation can be omitted in advance by predicting it. Neuron pruning was used as a method to reduce invalid neurons. On the other hand, in our proposal, instead of

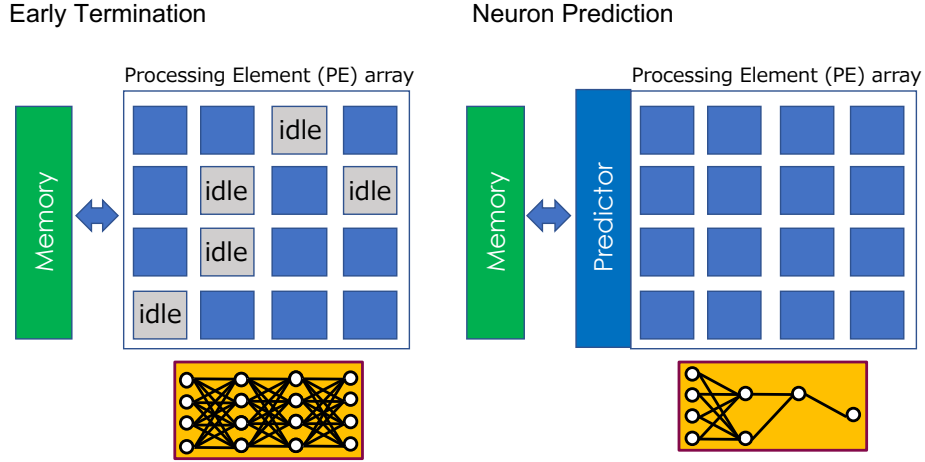


Figure 6.6: HW-efficiency comparison between the early-termination [9, 10] and the neuron prediction.

compressing the model itself, we apply a predictor to the already trained model per layer according to the input during inference, which performs only the minimum required calculations without sacrificing the accuracy (Fig. 6.5). It differs from the conventional calculation reduction method in that it is operated externally to make the calculation more efficient.

6.2.3 Related Works for Dead Neuron Prediction

The dead neuron problem has a large space of improving a performance of DNN-inference computation. There are several related works for skipping the wasted MAC operations by stopping in the middle of neuron computing [9, 10]. However, they just terminate the calculation of dead neurons, not skip them.

Dynamic pruning approach is also similar as our proposal. J. Lin et al. [63] introduce RNN as predictor and X. Gao et al. [64] introduce channel saliency predictor. Both approaches subsample the 2-D pixels into 1-D channel. They keep the accuracy by re-training whole DNN including predictor and executor. [65] also presents the channel gating during the inference. They develop a network architecture itself which is suitable for the channel gating. Meanwhile, our approach does not re-train the original pre-trained DNN. Therefore, our proposal is much lower training cost than the re-training or re-making network approaches.

S. Cao et al. [66] propose a method of skipping the dead neurons using the quantized original

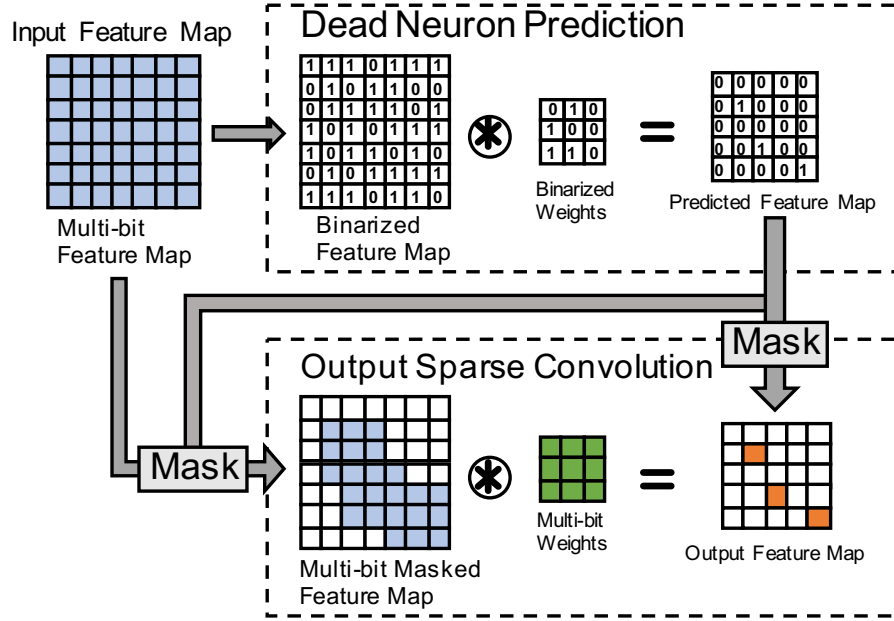


Figure 6.7: Dead neuron prediction mechanism.

pre-trained weights without training, and M. Song et al. [10] also use high-order bits of the original pre-trained weights. However, they show binarization which uses only most significant bit (MSB) is not enough to predict the dead neurons. We will show the training dedicated for DNP have a space of recovering the accuracy with skipping much more dead neurons.

Figure 6.6 illustrates the HW-efficiency comparison between early-termination [9, 10] and our proposal. Our proposal which predicts before the neuron calculation can operate the PE unit that calculates the neurons with higher utilization than the early termination that calculates all the neurons. This is because neuron prediction is performed as scheduler of DNN accelerator. Furthermore, it can be used an IP outside the conventional DNN accelerator.

6.2.4 Predicting Dead Neurons by Binarized NN

The output-feature map (OFM) of CNN has many zero-valued dead neurons due to the ReLU. This section explains how to train the dead neuron prediction (DNP) by yet another tiny neural networks.

Though the ReLU filters the negative pre-activation value, DNP means a prediction of a sign bit of the pre-activation value. Figure 6.7 shows the mechanism of our proposed DNP. A multi-bit input-feature map (IFM) is binarized, and computes a same operation as original pre-trained DNN

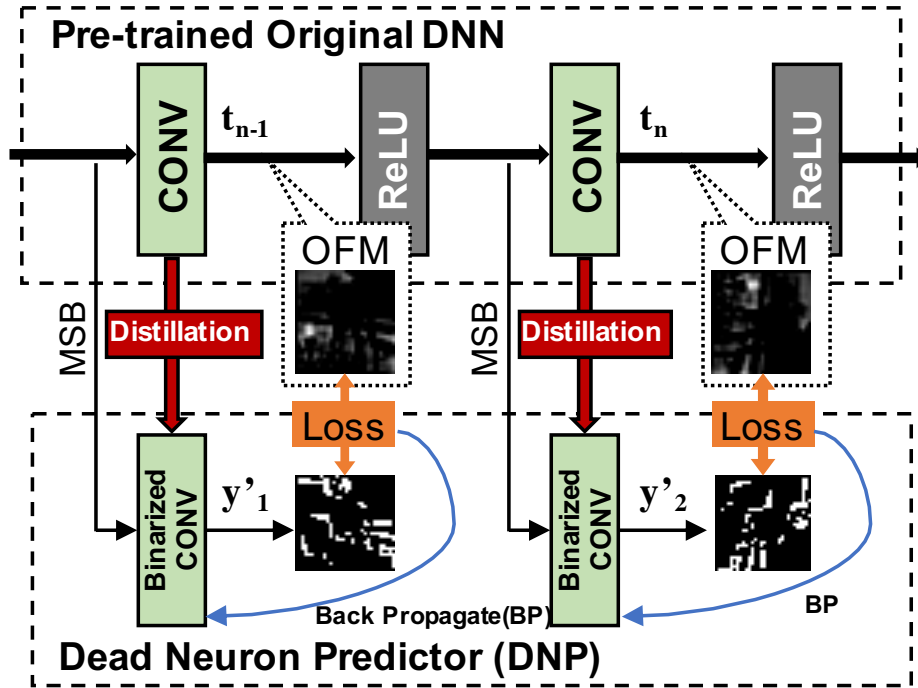


Figure 6.8: DNP training by layer.

at a layer. Note that the operations are binarized all weights such as -1, +1. The result of binarized operation becomes a predicted dead neuron mask. DNP trains the distribution of the predicted mask to be same as a OFM of the original pre-trained DNN. The trained DNP can remove the dead neurons in the multi-bit original DNN computing. By calculating DNP before the original DNN computing by layer, the original DNN computing can be calculated with the minimum amount of calculation. This is because DNP-based DNN computing can reduce multiply-and-accumulate (MAC) operations by DNP-based DNN computing.

Figure 6.8 shows the proposed training model for DNP. Each layer of the pre-trained DNN has a sparse OFM due to dead neuron as mentioned before. The proposed DNP trains to minimum losses between the OFM of original pre-trained DNN and the predicted feature maps of DNP. The DNP is composed of the same structure of the original pre-trained DNN model by layer. At the beginning of the DNP training, the binarized weights at DNP are initialized from the original pre-trained weights by binarization. Since each layer of DNP is composed of the same structure as a layer of the original DNN, the OFM of BDNP corresponds to the OFM of the original. Therefore, it can be trained from a simple difference of each OFM by a back propagation. As a loss function for the DNP training, we

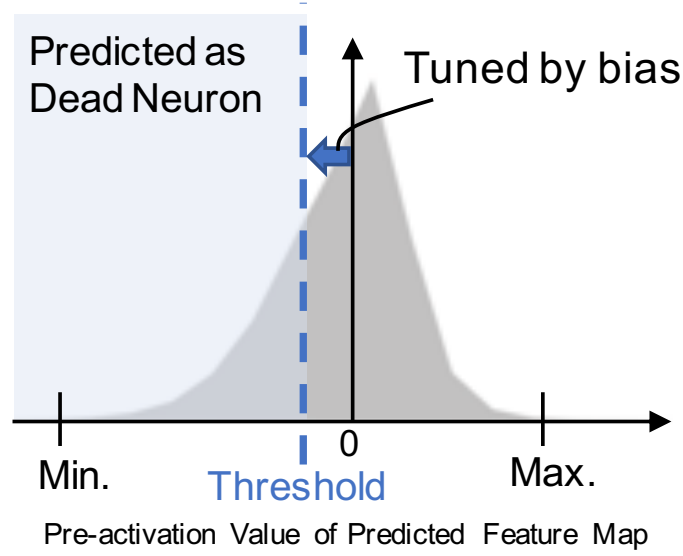


Figure 6.9: *Threshold adjustment.*

used pairwise mean squared error (PMSE) defined by the following formula:

$$L_{PMSE} = \frac{1}{(N-1)^2} \sum_{i=1}^N \sum_{j=1}^N ((y_i - y_j) - (t_i - t_j))^2 \quad (6.2)$$

where N is batch size, y is predicted feature map of DNP, and t is OFM of original pre-trained DNN. The loss is adopted because it is able to express the relationship among batches and heuristically succeeded.

6.2.5 Optimizing a Tradeoff between Skipping Amount and Accuracy

In this subsection, we propose the tuning method for a trade-off between the classification accuracy and the computation performance. Prediction of the dead neurons cannot be ensured even after enough training. It may sacrifice the classification accuracy. However, our proposal does not change the parameters of the original pre-trained DNN model. Therefore, the trade-off can be adjusted by how much the DNP results are reflected in the original pre-trained DNN computing.

In the BNN, the weights are binarized, but the biases are not binarized. In this paper, they have 16 bits. This is because the BNN should have a batch normalization [14] which is widely used to maintain high accuracy even for binarized weights. At an inference phase, a calculation for the batch normalization can be omitted by calculating in advance and including in biases [67]. Therefore, it is

Chapter 6. Energy Efficient Inference by Skipping Inefficient Calculation

necessary to prepare enough numerical representation for the biases in the BNN.

Since the biases have high numerical representation, the threshold of binarization for the predicted feature maps at DNP can be adjusted using the shift by the biases. This adjustment can be artificially operable after training. Figure 6.9 illustrates the mechanism of the adjusting threshold for pre-activated value. The histogram shows a pre-activated neuron at DNP. The pre-activated neuron will be binarized to predict whether the neuron will be dead or not. Before the binarization, the threshold can be tuned by adding bias. Therefore, no additional computing cost needs to adjust the threshold.

6.3 Accelerator Architecture

In this section, we propose a DNP-based hardware architecture combined an output sparse DNN accelerator and an in-memory BNN accelerator.

6.3.1 Output sparse CNN

Since the DNP predicts the sparse OFMs, the calculation process is different with the general sparse CNN accelerators. In general sparse CNN accelerators, the zero-valued IFMs and weights can be skipped, but all OFMs should be calculated because the values of OFMs cannot know before calculation. On the other hand, in the DNP-based accelerator, all predicted dead neurons can be skipped. Therefore, the conventional array architecture is not efficient due to the OFM sparsity which includes pixel and channel sparsity. Predicted neurons show different distributions in each OFM. Therefore, each OFM should be calculated independently not be chunked. Our proposed architecture introduces the weight stationary dataflow to take advantage of the reusability of weights. In addition to the output sparsity, the input and weight sparsity is also considered by an indexing approach in our proposed architecture. This means minimal input feature maps and weights are fetched for the selected output feature maps.

6.3.2 Proposed DNP-based Accelerator Architecture

Figure 6.10 shows an overview of our proposed DNP-based DNN accelerator architecture. This architecture composed of mainly two parts: sparse-output multi-bit DNN accelerator and DNP accelerator by computing-in-memory for BNN.

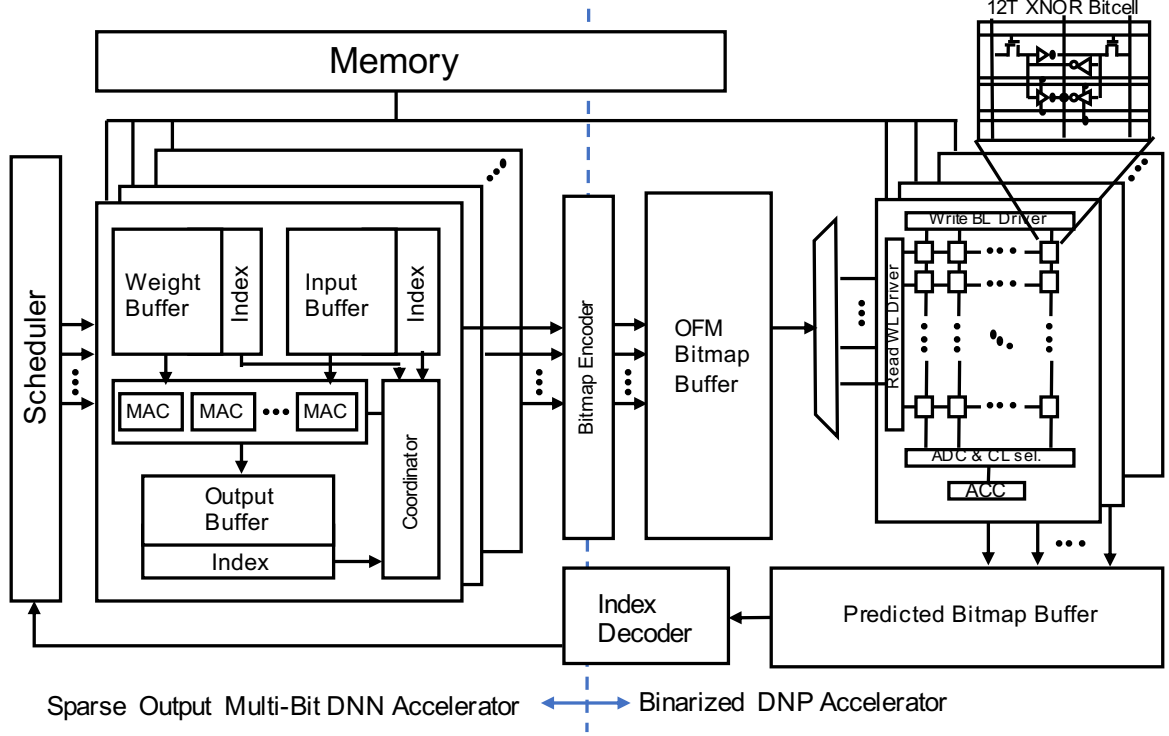


Figure 6.10: Overview of DNP-based accelerator.

The sparse-output multi-bit DNN accelerator is composed of an architecture based on SCNN [24], but our architecture has a benefit of the output sparsity by DNP. Our proposed architecture supports not only output sparsity, but also input and weight sparsity. Therefore, each buffer for inputs, outputs, and weights have index buffer to compress zero-value data. Only non-zero values are stored into the buffers with indexes which indicate the address of neurons and weights. The coordinator is data-driven controller for input and weight sparsity, which manages MAC operations by judging whether the input and weight indices match. The scheduler manages which output neurons should be calculated. This can be controlled because of the predicted feature maps by DNP.

The DNP accelerator is composed of bitmap buffers and XNOR-SRAM macros proposed by Z. Jiang et al. [68] which is one of the well-designed computing-in-memory SRAM macro dedicated for the BNN. The XNOR-SRAM macro has 64x256 bit cell which is able to process XNOR operation in parallel read accesses. Word lines drives at the same time to calculate XNOR and accumulate all the bit line at one cycle. The XNOR-SRAM computes all binarized OFMs produced from multi-bit DNN accelerators. Always-on DNP computing by energy saving computing-in-memory SRAMs manages

Table 6.1: *The network structure of M-CifarNet for CIFAR10 classification.*

Layer	Shape	# MACs
conv0	30×30	1.5M
conv1	30×30	33.2M
conv2	15×15	16.6M
conv3	15×15	33.2M
conv4	8×8	33.2M
conv5	8×8	14.1M
conv6	8×8	21.2M
conv7	8×8	21.2M
fc	1×1	1.9k
Total		174.3M

the minimum operation of the output-sparse multi-bit DNN accelerator.

The data between the output-sparse multi-bit DNN accelerator and DNP accelerator has been converted from/to bitmap to/from indices. The outputs of multi-bit DNN accelerator are binarized and encoded to bitmap of OFM. The OFM is chunked by channel to compute convolution operation at XNOR-SRAM macro. On the other hand, the outputs of XNOR-SRAM are converted to indices which indicate the address of output neuron to compute at DNN accelerator. These operations are pipelined by layer.

6.4 Experimental results

In this section, we evaluate the effect of DNP training and energy consumption of proposed DNP-based architecture.

6.4.1 Setups

We selected typical CNN model of VGG for CIFAR10 as benchmark (called M-CifarNet in [64]) which has eight layer of convolution layer. The network structure is shown in Table 6.1. Before DNP training, we trained the benchmark model with a 0.01 learning rate and a 256 batch size using PyTorch. We reduced the learning rate by a factor of 10 times for every 100 epochs, and trained for

Table 6.2: *Energy setup table.*

Operation	Energy (pJ/bit)	Relative Cost
Register File Access	0.2	1
Index Decoder	0.2	1
Bitmap Encoder	0.2	1
16-bit Fixed Point MAC	0.3	1.5
256 kB SRAM	1.2	6
DDR4 Memory Access	15.0	75
XNOR-SRAM (fast: 1.2GHz) [68]	0.015	0.075
XNOR-SRAM (std: 500MHz) [68]	0.008	0.04

a maximum of 300 epochs which is enough to achieve at 92.2 % classification accuracy. We trained DNP using this pre-trained DNN models.

We used an in-house simulator to evaluate the energy consumption of the proposed architecture. The simulator simulates the circuit behavior in a cycle-accurate manner, and thus provides several performance statistics. The energy consumption of each circuit primitive was estimated based on an RTL design using Synopsys Design Compiler [36] in UMC 40-nm CMOS technology. The characteristics of the memory components, such the SRAM and DRAM modules, were evaluated using CACTI6.5 [8] and a DDR4 system-power calculator [55] for a fair comparison. Table 6.2 lists the energy consumption per microarchitecture component used in our evaluation.

6.4.2 Computation Reduction and Accuracy

Figure 6.11 shows the recall in each layer between initialized from the original pre-trained DNN and after DNP training. After training, the recall of DNP recovers because the positive neuron can be surely predicted. This means the trained DNP will not affect the final classification accuracy. Figure 6.12 shows the trade-off between the classification accuracy and the number of MAC operation. Before DNP training, the final classification accuracy significantly degrades because many non-dead neurons cannot be predicted by just binarized weights from the original pre-trained DNN. After DNP training, however, it can be improved almost same as original classification accuracy with enough reduction of the number of MAC operation. From this observation, it is revealed that DNP by binarized

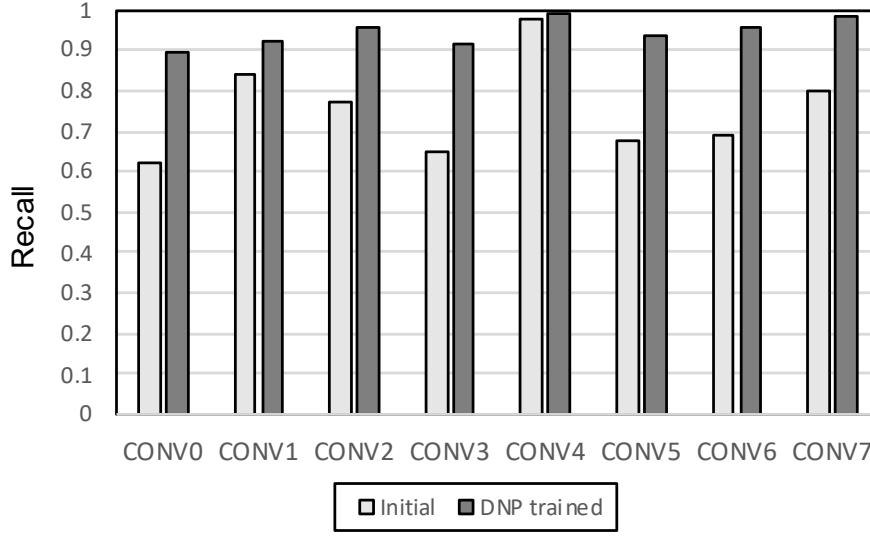


Figure 6.11: Recall of DNP in each layer.

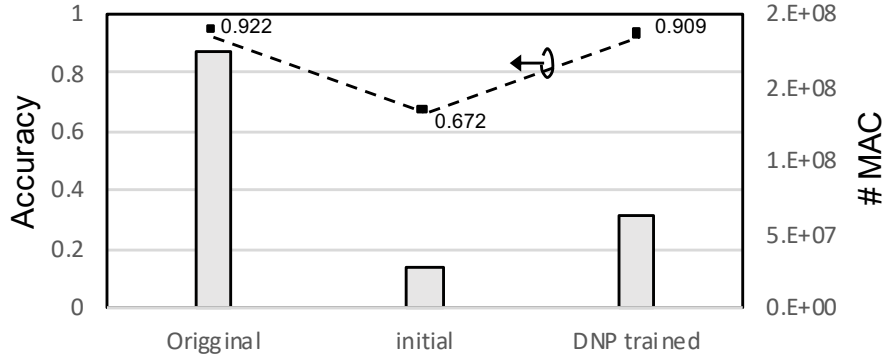


Figure 6.12: Classification accuracy and the number of MAC operation.

weights requires different features with pre-trained weights. Therefore, a dedicated predictor trained by yet another neural network is more effective than the exist works using original weights [10, 66] for aggressive prediction using only one bit. Note that the DNP training does not tuning original pre-trained DNN the related works [63–65] do it.

Figure 6.13 shows behaviors of trade-off between the classification accuracy and the number of MAC reduction by threshold adjustments. All pre-activated neurons in DNN model are tuned by fixed additional bias shifting for each data point. Therefore, it is quite easy to control after DNP training with monitoring the degradation of accuracy. From this observation, we can select the aggressiveness

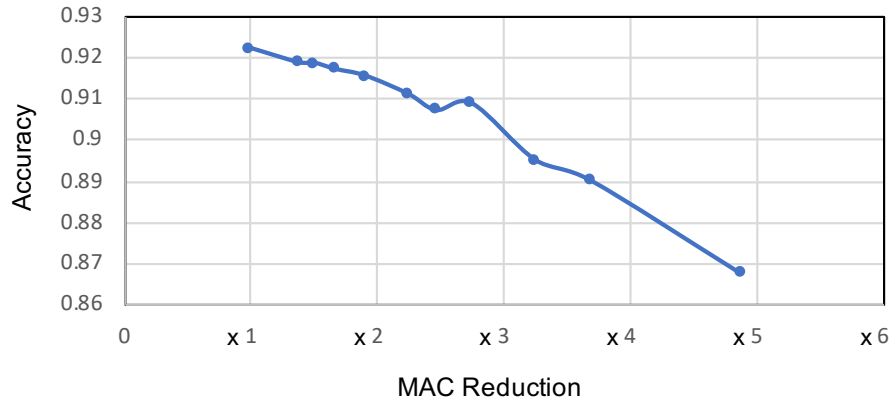


Figure 6.13: Trade-off between classification accuracy and MAC reduction by threshold tuning for CifarNet.

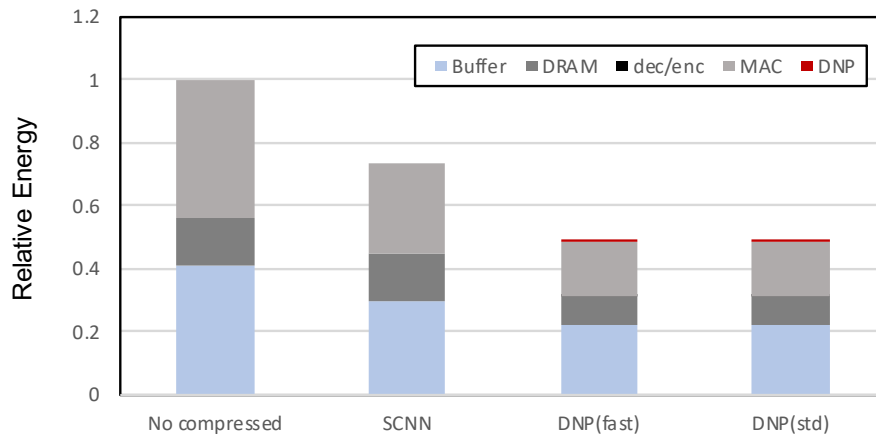


Figure 6.14: Energy comparison for no compressed processing, SCNN, and our DNP approach.

of the prediction even for more difficult tasks which have sensitive results.

6.4.3 Energy Consumption

Figure 6.14 shows a comparison with no compressed processing which calculated all neurons and weights, SCNN [24], and DNP. They are estimated using energy consumption per microarchitecture component shown in Table 6.2. SCNN skips the calculation of the set of zero input and weights. This is effective to reduce the energy by skipping the part of the neuron computing. On the other

Chapter 6. Energy Efficient Inference by Skipping Inefficient Calculation

hand, our proposed DNP-based accelerator shows 2.06x energy reduction over SCNN approach. This is because whole predicted output neurons can be removed from the computing. In addition, the inputs and weights related with predicted dead neurons are not read from the external memory or buffers. This is significantly more effective than only reduce the part of MAC operation at one neuron computing.

6.5 Conclusion

This paper presented a novel SW-HW co-designed approach for energy-efficient DNN accelerator by DNP composed of BNN. The DNP enables to predict the dead neuron which does not affect the next layer before computing of the original pre-trained DNN. The DNP is composed of binarized weights and activations so that the computing cost is extremely low. We proposed a DNP-based HW accelerator composed of pipelined DNP and 16-bit DNN accelerator. Finally, our proposed HW accelerator achieves 2.06x energy reduction against modern sparse DNN accelerator without sacrificing accuracy.

CHAPTER 7

Conclusion

With the development of artificial intelligence, the demand for fast and low power consumption for DNNs is rapidly increasing. In order to realize highly efficient computing for DNNs on real-world devices with restricted energy constraints, we need to reconsider co-optimization both of algorithms and architecture design.

In this study, we aim at efficiency-centric hardware design for DNN inference. While DNN achieves high recognition accuracy, the network model size is increasing. To solve this problem, it is necessary to reduce the network size appropriately. In previous studies, hardware engineers worked hard to implement DNN models compressed by machine-learning professionals on realistic devices. However, in this study, we proposed more efficient solutions by constructing a DNN model considering hardware side in advance. Therefore, we have optimized the trade-off between computing efficiency and DNN model accuracy, instead of the traditional hardware design that considered only computing efficiency which includes the performance and energy.

In chapter 3, we discussed hardware-oriented quantization. Logarithmic quantization has higher

Chapter 7. Conclusion

resolution for values near zero as compared to linear quantization. Therefore, it is finer fitting for the distribution of the weight and activation at higher density. The logarithmic quantization can maintain high accuracy with a small bit width. In the ImageNet classification task, the bit precision can be reduced to 4 bits while the linear quantization is 7 bits without re-training. A benefit of the logarithmic quantization is a multiplier-free processing. This is able to replace the power-hungry multiplication with simple addition. Therefore, we proposed the logarithmic-quantization-based serial processing realizes the low energy and area with flexible bit precision. Since required minimum bit width is different among with different application level, a flexibility is important role to optimize a trade-off between the accuracy and calculation cost. The energy-based evaluation also shows high energy efficiency of 0.24 pJ for equivalent calculation to multiplication against 0.43 pJ in linear quantization, and high selectivity by bit serialization.

Based on the evaluations in chapter 3, we proposed log-quantized DNN inference engine with stacking technology called QUEST in chapter 4. The proposed design has three characteristic. First is multiplier-free multiply-and-accumulation by logarithmic quantization. This can be reduce the energy and be friendly for the bit-serial processing. Second is the homogenous micro architecture both for convolutional layer and fully-connected layer. Combined with MIMD parallel cores can realize a high performance computing with fully-utilized processing. Last is a 3D SRAM integration. This can be realize cycle-level random access for middle class memory capacity of 96 MB. The optimized design for these characteristic showed high efficiency both for performance and energy against other works do not consider the external memory which is the most power consuming operation.

Generally speaking, the computation performance is limited by memory bandwidth, which is explained by roofline model described in chapter 2. Our goal is versatile computing for both CONV and FC layer. Therefore, enough memory bandwidth with high energy efficiency is required to achieve peak performance for different arithmetics operations. In this study, we introduced die-stacking technology using inductive-coupling technology called Thru chip interface (TCI). TCI has a simple transmitter and receiver composed of CMOS digital circuits so that it is scalable and is integrated at a low cost. In this study, we stacked the 10- μ m SRAM dies for middle-range capacity cache for practical size of DNN inference such as AlexNet and VGG16. The low-latency access of TCI which achieved 9.6 Gb/s/vault enables broadcast access to all stacked SRAM dies by fixed latency at 300 MHz system clock frequency.

Finally, it achieves 7.49 TOPS peak performance in binary precision and 1.825 TOPS for the

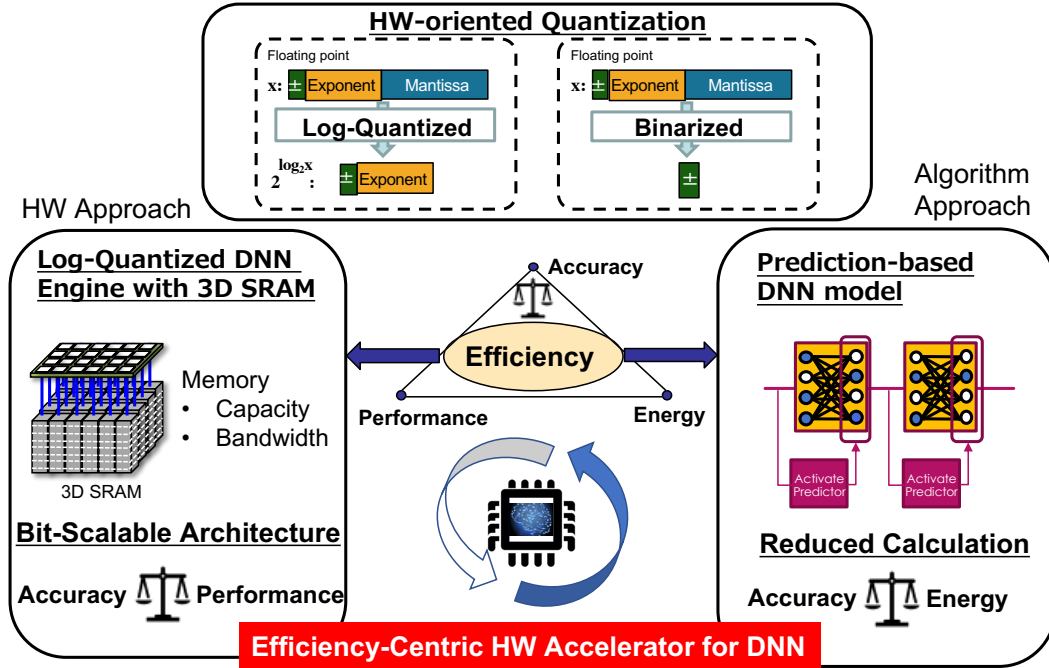


Figure 7.1: Summary of the thesis.

actual AlexNet processing. The results indicate that logarithmic quantization and bit-serial processing exhibit highly-effective performances in both the CONV and FC layers with efficient flexibility to balance the computing performance and recognition accuracy. Additionally, the results revealed that stacking SRAMs provided high energy efficiency compared to stacking DRAMs. Furthermore, we implemented our proposed architecture on the fabricated chip to verify our simulation results.

In addition, we evaluated the energy and performance for different memory in the case of 3D integrated memory with logic cores in chapter 5. The experiment is based on the QUEST architecture but through silicon via (TSV) is used as the 3D die stacking technology. The result shows the low-latency SRAM is better than DRAM integration for DNN inference. The evaluation results show that the stacking of SRAM chips can provide a high DNN processing performance as well as greater energy efficiency when compared to the stacking of DRAM chips, that is, both bandwidth and latency are important aspects in DNN inference processing.

In chapter 6, we proposed the prediction-based DNN inference model to skip inefficient neurons which result in zero called dead neurons. Since the dead neuron does not affect the calculation of neurons in the next layer, the amount of calculation can be greatly reduced if the calculation can be omitted in advance by predicting it. Our proposal applies a predictor to the already trained model

Chapter 7. Conclusion

per layer according to the input during inference, which performs only the minimum required calculations without sacrificing the accuracy. Additional low-calculation costed binarized neural network predicts the sign of pre-activation value that produces the zero output. This can be worked with existed general DNN engine. The proposed approach showed the efficient DNN inference by skipping the calculation itself. The result achieved the 2.77x MAC reduction with 1.1 % accuracy degradation. We also evaluated a dead-neuron-prediction-based HW accelerator composed of pipelined binarized in-memory DNN accelerator and 16-bit conventional DNN accelerator. Finally, our proposed HW accelerator achieves 2.06x energy reduction against modern sparse DNN accelerator without sacrificing accuracy.

Through the thesis, we expanded the world of both performance and energy efficiency from the point of view of accuracy by introducing HW-oriented quantization and a new prediction-based model. This study will play an important role in expanding the platform of advanced intelligent information processing and suggesting the possibility of creating new applications.

Bibliography

- [1] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov 1998.
- [2] 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.
- [3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.
- [4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: <http://arxiv.org/abs/1409.4842>
- [5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.
- [6] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," *CoRR*, vol. abs/1709.01507, 2017. [Online]. Available: <http://arxiv.org/abs/1709.01507>

Bibliography

- [7] M. Horowitz, “1.1 computing’s energy problem (and what we can do about it),” in *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, Feb 2014, pp. 10–14.
- [8] N. N. Author, “Cacti 6.5,” <http://www.hpl.hp.com/research/cacti>, accessed: 2018-09-28.
- [9] V. Akhlaghi, A. Yazdanbakhsh, K. Samadi, R. K. Gupta, and H. Esmailzadeh, “Snapea: Predictive early activation for reducing computation in deep convolutional neural networks,” in *Proceedings of the 45th Annual International Symposium on Computer Architecture*, ser. ISCA ’18. Piscataway, NJ, USA: IEEE Press, 2018, pp. 662–673. [Online]. Available: <https://doi.org/10.1109/ISCA.2018.00061>
- [10] M. Song, J. Zhao, Y. Hu, J. Zhang, and T. Li, “Prediction based execution on deep neural networks,” in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, June 2018, pp. 752–763.
- [11] N. N. Author, “Whitepaper gpu-based deep learning inference: A performance and power analysis,” 2015.
- [12] —, “Technical overview nvidia deep learning platform giant leaps in performance and efficiency for ai services, from the data center to the network’s edge,” 2017.
- [13] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, pp. 436 EP –, 05 2015.
- [14] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *ICML*, ser. JMLR Workshop and Conference Proceedings, F. R. Bach and D. M. Blei, Eds., vol. 37. JMLR.org, 2015, pp. 448–456.
- [15] 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.
- [16] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, “XNOR-Net: Imagenet classification using binary convolutional neural networks,” *CoRR*, vol. abs/1603.05279, 2016.
- [17] B. Reagen, P. Whatmough, R. Adolf, S. Rama, H. Lee, S. K. Lee, J. M. Hernandez-Lobato, G. Y. Wei, and D. Brooks, “Minerva: Enabling low-power, highly-accurate deep neural network accel-

- erators,” in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, June 2016, pp. 267–278.
- [18] 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.
- [19] D. Miyashita, E. H. Lee, and B. Murmann, “Convolutional neural networks using logarithmic data representation,” *CoRR*, vol. abs/1603.01025, 2016.
- [20] 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.
- [21] 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.
- [22] 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.
- [23] 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.
- [24] 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.
- [25] 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.

Bibliography

- [26] Y. H. Chen, T. Krishna, J. Emer, and V. Sze, “Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks,” in *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, Jan 2016, pp. 262–263.
- [27] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, “Optimizing fpga-based accelerator design for deep convolutional neural networks,” in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, ser. FPGA ’15. New York, NY, USA: ACM, 2015, pp. 161–170.
- [28] M. Motamedi, P. Gysel, V. Akella, and S. Ghiasi, “Design space exploration of fpga-based deep convolutional neural networks,” in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2016, pp. 575–580.
- [29] K. Ueyoshi, K. Ando, K. Orimo, M. Ikebe, T. Asai, and M. Motomura, “Exploring optimized accelerator design for binarized convolutional neural networks,” in *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017, pp. 2510–2516.
- [30] S. Park, S. Choi, J. Lee, M. Kim, J. Park, and H. J. Yoo, “14.1 a 126.1mw real-time natural ui/ux processor with embedded deep-learning core for low-power smart glasses,” in *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, Jan 2016, pp. 254–255.
- [31] Z. Du, R. Fasthuber, T. Chen, P. Ienne, L. Li, T. Luo, X. Feng, Y. Chen, and O. Temam, “Shidiannao: Shifting vision processing closer to the sensor,” in *Proceedings of the 42Nd Annual International Symposium on Computer Architecture*, ser. ISCA ’15. New York, NY, USA: ACM, 2015, pp. 92–104.
- [32] 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.
- [33] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasude-

- van, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org.
- [34] P. Yin, S. Zhang, J. Xin, and Y. Qi, “Training ternary neural networks with exact proximal operator,” *CoRR*, vol. abs/1612.06052, 2016.
- [35] S. Zhou, Z. Ni, X. Zhou, H. Wen, Y. Wu, and Y. Zou, “Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients,” *CoRR*, vol. abs/1606.06160, 2016.
- [36] N. N. Author, “Synopsys dc ultra,” <https://www.synopsys.com/implementation-and-signoff/rtl-synthesis-test/dc-ultra.html>, accessed: 2018-09-28.
- [37] 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.
- [38] 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.
- [39] 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.
- [40] D. Amodei, R. Anubhai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, E. Elsen, J. Engel, L. Fan, C. Fougner, T. Han, A. Y. Hannun, B. Jun, P. LeGresley, L. Lin, S. Narang, A. Y. Ng, S. Ozair, R. Prenger, J. Raiman, S. Satheesh, D. Seetapun, S. Sengupta, Y. Wang, Z. Wang, C. Wang, B. Xiao, D. Yogatama, J. Zhan, and Z. Zhu, “Deep speech 2: End-to-end speech recognition in english and mandarin,” *CoRR*, vol. abs/1512.02595, 2015.
- [41] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

Bibliography

- [42] D. Kim, J. Kung, S. Chai, S. Yalamanchili, and S. Mukhopadhyay, “Neurocube: A programmable digital neuromorphic architecture with high-density 3d memory,” in *2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA)*, June 2016, pp. 380–392.
- [43] M. Gao, J. Pu, X. Yang, M. Horowitz, and C. Kozyrakis, “Tetris: Scalable and efficient neural network acceleration with 3d memory,” *SIGARCH Comput. Archit. News*, vol. 45, no. 1, pp. 751–764, Apr. 2017.
- [44] 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.
- [45] 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.
- [46] D. Shin, J. Lee, J. Lee, and H. J. Yoo, “Dnpu: An 8.1tops/w reconfigurable cnn-rnn processor for general-purpose deep neural networks,” in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2017, pp. 240–241.
- [47] S. Yin, P. Ouyang, S. Tang, F. Tu, X. Li, L. Liu, and S. Wei, “A 1.06-to-5.09 tops/w reconfigurable hybrid-neural-network processor for deep learning applications,” in *2017 Symposium on VLSI Circuits*, June 2017, pp. C26–C27.
- [48] J. Lee, C. Kim, S. Kang, D. Shin, S. Kim, and H. J. Yoo, “Unpu: A 50.6tops/w unified deep neural network accelerator with 1b-to-16b fully-variable weight bit-precision,” in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, Feb 2018, pp. 218–220.
- [49] S. Venkataramani, A. Ranjan, S. Banerjee, D. Das, S. Avancha, A. Jagannathan, A. Durg, D. Nagaraj, B. Kaul, P. Dubey, and A. Raghunathan, “Scaleddeep: A scalable compute architecture for learning and evaluating deep networks,” in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ser. ISCA ’17. New York, NY, USA: ACM, 2017, pp. 13–26.

-
- [50] Y. Jia, “Caffe model zoo,” http://caffe.berkeleyvision.org/model_zoo.html, accessed: 2018-09-28.
- [51] A. Krizhevsky, V. Nair, and G. Hinton, “Cifar-10 (canadian institute for advanced research).”
- [52] B. Moons, R. Uytterhoeven, W. Dehaene, and M. Verhelst, “14.5 envision: A 0.26-to-10tops/w subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm fdsoi,” in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2017, pp. 246–247.
- [53] Y. LeCun and C. Cortes, “MNIST handwritten digit database,” 2010.
- [54] 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.
- [55] N. N. Author, “Micron technology inc. ddr4 sdram system-power calculator.” <https://www.micron.com/support/tools-and-utilities/power-calc>, accessed: 2018-09-28.
- [56] —, “Micron technology inc. twindie 1.2v ddr4 sdram features.” <https://www.micron.com/products/dram/ddr4-sdram/16Gb/>, accessed: 2018-09-28.
- [57] Y. H. Chen, T. Krishna, J. S. Emer, and V. Sze, “Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks,” *IEEE Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Jan 2017.
- [58] S. Yin, P. Ouyang, S. Tang, F. Tu, X. Li, S. Zheng, T. Lu, J. Gu, L. Liu, and S. Wei, “A high energy efficient reconfigurable hybrid neural network processor for deep learning applications,” *IEEE Journal of Solid-State Circuits*, vol. 53, no. 4, pp. 968–982, April 2018.
- [59] A. Sridhar, A. Vincenzi, D. Atienza, and T. Brunschweiler, “3d-ice: A compact thermal model for early-stage design of liquid-cooled ics,” *IEEE Transactions on Computers*, vol. 63, no. 10, pp. 2576–2589, Oct 2014.
- [60] M. Gao, J. Pu, X. Yang, M. Horowitz, and C. Kozyrakis, “Tetris: Scalable and efficient neural network acceleration with 3d memory,” in *Proceedings of the Twenty-Second International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS ’17. New York, NY, USA: ACM, 2017, pp. 751–764.

Bibliography

- [61] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, “Mobilenetv2: Inverted residuals and linear bottlenecks,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018, pp. 1–10.
- [62] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee, “Understanding deep neural networks with rectified linear units,” *CoRR*, vol. abs/1611.01491, 2016. [Online]. Available: <http://arxiv.org/abs/1611.01491>
- [63] J. Lin, Y. Rao, J. Lu, and J. Zhou, “Runtime neural pruning,” in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 2181–2191. [Online]. Available: <http://papers.nips.cc/paper/6813-runtime-neural-pruning.pdf>
- [64] X. Gao, Y. Zhao, L. Dudziak, R. Mullins, and C.-Z. X. 0001, “Dynamic channel pruning: Feature boosting and suppression,” in *ICLR*. OpenReview.net, 2019.
- [65] W. Hua, Y. Zhou, C. De Sa, Z. Zhang, and G. E. Suh, “Boosting the performance of cnn accelerators with dynamic fine-grained channel gating,” in *Proceedings of the 52Nd Annual IEEE/ACM International Symposium on Microarchitecture*, ser. MICRO ’52. New York, NY, USA: ACM, 2019, pp. 139–150. [Online]. Available: <http://doi.acm.org/10.1145/3352460.3358283>
- [66] S. Cao, L. Ma, W. Xiao, C. Zhang, Y. Liu, L. Zhang, L. Nie, and Z. Yang, “Seernet: Predicting convolutional neural network feature-map sparsity through low-bit quantization,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [67] H. Yonekawa and H. Nakahara, “On-chip memory based binarized convolutional deep neural network applying batch normalization free technique on an FPGA,” in *2017 IEEE International Parallel and Distributed Processing Symposium Workshops, IPDPS Workshops 2017, Orlando / Buena Vista, FL, USA, May 29 - June 2, 2017*, 2017, pp. 98–105. [Online]. Available: <https://doi.org/10.1109/IPDPSW.2017.95>
- [68] Z. Jiang, S. Yin, J.-s. Seo, and M. Seok, “Xnor-sram: In-bitcell computing sram macro based on resistive computing mechanism,” in *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, ser. GLSVLSI ’19. New York, NY, USA: ACM, 2019, pp. 417–422. [Online]. Available: <http://doi.acm.org/10.1145/3299874.3319458>

List of Publications

1. Journal Papers

1. **K. Ueyoshi**, K. Ando, K. Hirose, S. Takamaeda-Yamazaki, M. Hamada, T. Kuroda, and M. Motomura, "QUEST: Multi-purpose log-quantized DNN inference engine stacked on 96-MB 3-D SRAM using inductive coupling technology in 40-nm CMOS," IEEE Journal of Solid-State Circuits, vol. 54, no. 1, pp. 186-196 (2019).
2. **K. Ueyoshi**, T. Marukame, T. Asai, M. Motomura, and A. Schmid, "FPGA implementation of a scalable and highly parallel architecture for restricted Boltzmann machines," Circuits and Systems, vol. 7, no. 9, pp. 2132-2141 (2016).
3. **K. Ueyoshi**, T. Marukame, T. Asai, M. Motomura, and A. Schmid, "Robustness of hardware-oriented restricted Boltzmann machines in deep belief networks for reliable processing," Non-linear Theory and Its Applications, vol. E7-N, no. 3, pp. 395-406 (2016).

2. Journal Papers (Co-authored)

1. K. Ando, **K. Ueyoshi**, Y. Oba, K. Hirose, R. Uematsu, T. Kudo, M. Ikebe, T. Asai, S. Takamaeda-Yamazaki, and M. Motomura, "Dither NN: hardware/algorithm co-design for accurate quantized neural networks," IEICE Transactions on Information and Systems, vol. E102, pp. 2341-2353 (2019).

List of Publications

2. K. Hirose, R. Uematsu, K. Ando, **K. Ueyoshi**, M. Ikebe, T. Asai, M. Motomura, and S. Takamaeda-Yamazaki, "Quantization error-based regularization for hardware-aware neural network training," *Nonlinear Theory and Its Applications*, vol. E9-N, no. 4, pp. 453-465 (2018).
3. 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 1.4TOPS at 0.6W," *IEEE Journal of Solid-State Circuits*, vol. 53, no. 4, pp. 983-994 (2018).
4. T. Marukame, **K. Ueyoshi**, T. Asai, M. Motomura, A. Schmid, M. Suzuki, Y. Higashi, and Y. Mitani, "Error tolerance analysis of deep learning hardware using restricted Boltzmann machine towards low-power memory implementation," *IEEE Transactions on Circuits and Systems II*, vol. 64, no. 4, pp. 462-466 (2017).

3. International Conference Proceedings

1. **K. Ueyoshi**, "Log-Quantized DNN Inference Engine Stacked on 3D SRAM Using Inductive Coupling Technology," The 2nd GI-CoRE GSQ, GSB, & IGM Joint Symposium -Quantum, Informatics, Biology & Medicine-, Hokkaido University, Sapporo, Japan (Aug. 7-8, 2018).
2. **K. Ueyoshi**, K. Ando, K. Hirose, S. Takamaeda-Yamazaki, J. Kadomoto, T. Miyata, M. Hamada, T. Kuroda, and M. Motomura, "QUEST: A 7.49-TOPS Multi-Purpose Log-Quantized DNN Inference Engine Stacked on 96MB 3D SRAM using Inductive-Coupling Technology in 40nm CMOS," 2018 International Solid-State Circuits Conference (ISSCC 2018), San Francisco Marriott Marquis, San Francisco, US (Feb. 11-15, 2018).
3. **K. Ueyoshi**, M. Ikebe, T. Asai, S. Takamaeda-Yamazaki, and M. Motomura, "Hardware accelerator design for convolutional neural networks with low bit precision," GI-CoRE GSQ, GSB, & IGM Joint Symposium -Quantum, Informatics, Biology, & Medicine -, Hokkaido University, Sapporo, Japan (Jul. 10-11, 2017).
4. **K. Ueyoshi**, T. Marukame, T. Asai, M. Motomura, and A. Schmid, "Feature extraction system using restricted Boltzmann machines on FPGA," 2017 IEEE International Symposium on Circuits & Systems, A4P-O, Baltimore Marriott Waterfront, Baltimore, USA (May 28-31, 2017).

5. **K. Ueyoshi**, K. Ando, K. Orimo, M. Ikebe, T. Asai, and M. Motomura, "Exploring optimized accelerator design for binarized convolutional neural networks," The 2017 International Joint Conference on Neural Networks, William A. Egan Civic and Convention Center, Alaska, USA (May 14-19, 2017).
6. **K. Ueyoshi**, T. Marukame, T. Asai, M. Motomura, and A. Schmid, "Memory-error tolerance of scalable and highly parallel architecture for restricted Boltzmann machines in deep belief network," IEEE International Symposium on Circuits and Systems, Montreal Sheraton Center, Montreal, Canada (May 22-25, 2016).
7. **K. Ueyoshi**, T. Asai, and M. Motomura, "Scalable and highly-parallel architecture for restricted boltzmann machines," 2015 RISP International Workshop on Nonlinear Circuits, Communications and Signal Processing, pp. 369-372, Universiti Teknologi Malaysia, Kuala Lumpur, Malaysia (Feb. 27-Mar. 2, 2015).

4. International Conference Proceedings (Co-authored)

1. K. Ando, **K. Ueyoshi**, Y. Oba, K. Hirose, R. Uematsu, T. Kudo, M. Ikebe, T. Asai, S. Takamaeda-Yamazaki, and M. Motomura, "Dither NN: an accurate neural network with dithering for low bit-precision hardware," The 2018 International Conference on Field-Programmable Technology (FPT'18), Tenbusu-Naha Hall, Naha, Japan (Dec. 10-14, 2018).
2. T. Kudo, **K. Ueyoshi**, K. Ando, K. Hirose, R. Uematsu, Y. Oba, M. Ikebe, T. Asai, M. Motomura, and S. Takamaeda-Yamazaki, "Area and energy optimization for bit-serial log-quantized DNN Accelerator with shared accumulators," IEEE 12th International Symposium on Embedded Multicore/Many-core Systems-on-Chip, Vietnam National University, Hanoi, Vietnam (Sep. 12-14, 2018).
3. R. Uematsu, K. Ando, **K. Ueyoshi**, K. Hirose, M. Ikebe, T. Asai, S. Takamaeda-Yamazaki, and M. Motomura, "Exploring CNN accelerator design space on a dynamically reconfigurable hardware platform," The 21st Workshop on Synthesis And System Integration of Mixed Information Technologies (SASIMI 2018), Kunibiki Messe, Matsue, Japan (Mar. 26-27, 2018).
4. S. Takamaeda-Yamazaki, **K. Ueyoshi**, K. Ando, R. Uematsu, K. Hirose, M. Ikebe, T. Asai, and M. Motomura, "Accelerating Deep Learning by Binarized Hardware," Asia-Pacific Signal

List of Publications

- and Information Processing Association Annual Summit and Conference 2017 (APSIPA ASC 2017), Aloft Kuala Lumpur Sentral Sentral, Kuala Lumpur, Malaysia (Dec. 12-15, 2017).
5. K. Hirose, K. Ando, **K. Ueyoshi**, M. Ikebe, T. Asai, M. Motomura, and S. Takamaeda-Yamazaki, "Quantization Error-based Regularization in Neural Networks," Thirty-seventh SGAI International Conference on Artificial Intelligence (SGAI 2017), Peterhouse College, Cambridge, England (Dec. 12-14, 2017).
 6. I. Hida, **K. Ueyoshi**, S. Takamaeda-Yamazaki, M. Ikebe, M. Motomura, and T. Asai, "Sign-invariant unsupervised learning facilitates weighted-sum computation in analog neural-network devices," 2017 International Symposium on Nonlinear Theory and Its Applications, Cancun International Convention Center, Cancun, Mexico (Dec. 4-7, 2017).
 7. K. Hirose, R. Uematsu, K. Ando, K. Orimo, **K. Ueyoshi**, M. Ikebe, T. Asai, S. Takamaeda-Yamazaki, and M. Motomura, "Logarithmic Compression for Memory Footprint Reduction in Neural Network Training," 5th International Workshop on Computer Systems and Architectures (CSA 2017), Aomori Prefecture Tourist Center, Aomori, Japan (Nov. 19-22, 2017).
 8. K. Hirose, R. Uematsu, K. Ando, **K. Ueyoshi**, M. Ikebe, T. Asai, M. Motomura, and S. Takamaeda-Yamazaki, "A Regularization Approach for Quantized Neural Networks," International Workshop on Highly Efficient Neural Networks Design (HENND 2017), Lotte Hotel City Center, Seoul, Korea (Oct. 20-20, 2017).
 9. K. Ando, **K. Ueyoshi**, K. Hirose, K. Orimo, H. Yonekawa, S. Sato, H. Nakahara, M. Ikebe, S. Takamaeda-Yamazaki, T. Asai, T. Kuroda, and M. Motomura, "In-Memory Area-Efficient Signal Streaming Processor Design for Binary Neural Networks," 60th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS 2017), Tufts University, Boston, USA (Aug. 6-9, 2017).
 10. K. Ando, **K. Ueyoshi**, K. Orimo, H. Yonekawa, S. Sato, H. Nakahara, M. Ikebe, T. Asai, S. Takamaeda-Yamazaki, T. Kuroda, and M. Motomura, "BRein memory: a 13-layer 4.2 K neuron/0.8 M synapse binary/ternary reconfigurable in-memory deep neural network accelerator in 65 nm CMOS," 2017 Symposia on VLSI Technology and Circuits, Rihga Royal Hotel, Kyoto, Japan (Jun. 5-8, 2017).

11. K. Ando, **K. Ueyoshi**, K. Orimo, M. Ikebe, S. Takamaeda-Yamazaki, T. Asai, and M. Motomura, "Throughput analysis of a data-flow reconfigurable array architecture for convolutional neural networks," The 5th RIEC International Symposium on Brain Functions and Brain Computer, Tohoku University, Sendai, Japan (Feb. 27-28, 2017).
12. K. Orimo, K. Ando, **K. Ueyoshi**, M. Ikebe, T. Asai, and M. Motomura, "FPGA architecture for feed-forward sequential memory network targeting long-term time-series forecasting," 2016 International Conference on Reconfigurable Computing and FPGAs, Iberostar Cancun hotel, Cancun, Mexico (Nov. 30-Dec. 2, 2016).
13. K. Ando, K. Orimo, **K. Ueyoshi**, M. Ikebe, T. Asai, and M. Motomura, "Reconfigurable processor array architecture for deep convolutional neural networks," The 20th Workshop on Synthesis And System Integration of Mixed Information Technologies, Kyoto Research Park, Kyoto, Japan (Oct. 24-25, 2016).