Kobe University Repository : Kernel

PDF issue: 2026-11-26

Study on Performance Optimization of Monitoring System with Edge AI

大原，遼太郎

(Degree)
博士（工学）

(Date of Degree)
2025-03-25

(Date of Publication)
2026-03-01

(Resource Type)
doctoral thesis

(Report Number)
甲第9217号

(URL)
https://hdl.handle.net/20.500.14094/0100496498

※ 当コンテンツは神戸大学の学術成果です。無断複製・不正使用等を禁じます。著作権法で認められている範囲内で、適切にご利用ください。

Doctoral Dissertation

Performance Optimization of Monitoring System with Edge AI

エッジ AI を用いたモニタリングシステム

の高性能化に関する研究

(January, 2025)

Graduate School of System Informatics,

Kobe University

Ryotaro Ohara

Abstract

This paper is about optimizing the performance of monitoring systems using ultrasounds.

Since AlexNet was first reported in 2012, deep neural networks (DNNs) have developed rapidly, achieving significant results in a variety of tasks, including image processing, natural language processing, and speech processing. DNNs excel in tasks requiring real-time operation and function in real-world environments, such as robotics, anomaly detection, and autonomous driving. However, implementing deep-learning models in embedded systems is challenging because of computational resource limitations. Most deep-learning models involve many convolutional and fully connected layers. These layers require many MAC operations and significant memory bandwidth. Therefore, deep-learning models are typically implemented on high-performance servers equipped with high-performance CPUs, GPUs, and large memories. One common method to reduce the computational load on edge devices involves sending sampled data from the edge device to a server, which then processes the data. However, this approach had several limitations. First, it requires a constant network connection, which makes it unsuitable for real-time applications owing to communication delays. Second, it transfers large amounts of data, which consumes a significant amount of communication bandwidth. Furthermore, privacy concerns arise when measured data are uploaded to a server.

Edge AI provides an effective solution to these problems. Edge AI leverages edge computing technology to implement deep-learning models in embedded systems. Owing to their high computational resource requirements, implementing DNNs in typical microcontrollers or single-board computers is challenging. Solutions to these challenges include the quantization of DNN models and use of deep learning processors. Quantization reduces the number of DNN parameters and activations by converting them from FP32 precision to lower-bit precision. This method significantly reduces the memory bandwidth usage and computational resource requirements. A deep learning processor supports numerous sum-of-products operations and the substantial on-chip memory required for deep learning models, enabling the power-efficient and effective implementation of these models in edge devices.

The Internet of Things (IoT) is strongly associated with edge AI. The IoT technology connects various electronic devices through a network, enabling seamless collaboration to support a wide range of applications. The IoT technology has already been widely implemented in the society. IoT technologies for personal use include healthcare devices, such as smartwatches, which collect and analyze vital data. Smart home systems use smart keys and plugs to enhance convenience. Additionally, IoT services can be used to monitor elderly behavior using hot water pots and sensors, and smart meters can track electricity usage. The IoT technology is also applied in industry, where it is referred to as the Industrial IoT (IIoT). Its applications in industry include monitoring factory equipment, predicting failures, and improving efficiency. In agriculture, it is used to monitor greenhouse conditions and optimize crop growth. Thus, IoT technologies drive efficiency and enable

automation across various sectors, including society and industry. It is particularly indispensable in Japan, where a declining birthrate and an aging population are causing a significant workforce reduction.

Monitoring systems are among the most important applications of IoT technology. They facilitate rapid and appropriate responses by continuously observing the environment and detecting anomalies or emergencies in the environment or subjects. Sensors installed in residential areas monitor road and traffic conditions during disasters, provide guidance on safe evacuation routes, and assist in dispatching rescue personnel to isolated areas. Sensors installed in homes can detect accidents and injuries involving elderly people living alone. As discussed, monitoring technologies offer significant benefits to the society.

Obtaining resident consent to install sensors in their living spaces is challenging because of privacy concerns. In particular, installing optical cameras in living spaces is problematic because they can capture detailed personal information that individuals wish to keep private. Therefore, there is a need for sensors that minimize privacy risks by collecting only the data necessary for applications. Three-dimensional measurement technologies, such as LiDAR and millimeter-wave radar, are effective in addressing this need. However, unlike surveillance cameras, outdoor LiDAR systems are expensive, and their rotating mechanisms make long-term maintenance-free operation difficult. Millimeter-wave radar is costly and lacks the ability to capture high-resolution images that are comparable to those of LiDAR. As a solution, we propose a noncontact sensing technology based on ultrasonic waves and deep learning. Ultrasonic waves can be detected using inexpensive microphones, enabling three-dimensional environmental measurements through their analysis.

Three studies have been conducted to develop a monitoring system based on edge AI and ultrasound. The first focuses on reducing the main memory bandwidth of deep learning processors to enhance the speed and conserve the power of edge AI applications. In this study, we proposed two approaches to reduce the DRAM bandwidth in deep learning: enhancing the memory capacity with an MRAM and compressing the memory bandwidth with codebook memory. The second approach involves the development of a 3D reconstruction model using ultrasound images. In 3D ultrasound measurements, sound waves received by a microphone array are beamformed, and the time-of-flight (ToF) is calculated to construct an ultrasound image. In this study, a variational autoencoder (VAE) was trained with ultrasound images as the input, enabling the three-dimensional reconstruction of a three-cubic-meter space divided into voxels. The third is a wireless time-synchronization technology with ultralow power consumption. To synchronize multiple sensors in a monitoring system, we developed a time synchronization system that achieved sub-microsecond accuracy using Bluetooth Low Energy (BLE).

This paper is comprised of seven chapters. Chapter 1 introduces the study. Chapter 2 provides the background for this study. This section describes the proposed method, algorithm, and purpose of this research.

In Chapters 3 and 4, we explore methods for reducing the main memory bandwidth in deep-learning processors, aiming to save power in edge AI applications. Deep-learning models predominantly consist of convolutional or fully connected layers, which involve a large number of inner product operations.

Consequently, a significant number of parameters and activations must be read and written, leading to increased memory bandwidth. Memory access constitutes a significant portion of the power consumption of a processor. The power consumptions of the on-chip SRAM and external DRAM for memory access differ by two orders of magnitude. Maximizing on-chip computation and minimizing DRAM access can significantly reduce the power consumption. In this study, we used NVDLA, an open-source deep-learning processor developed by NVIDIA, as the reference processor. Using this processor as a reference, we conducted two studies: on-chip memory augmentation with MRAM (Chapter 3) and main-memory bandwidth compression with the codebook SRAM (Chapter 4).

In Chapter 4, we evaluate the performance improvements achieved by replacing conventional memory with MRAM. Magnetoresistive random access memory (MRAM) records information based on the magnetization direction. Unlike conventional SRAM, which consists of 6T transistors, the MRAM consists of 1T or 2T transistors and a single MTJ. MRAM is superior to SRAM in terms of area efficiency. In this study, we estimated the memory expansion achieved by configuring on-chip memory, such as the cache in the NVDLA, with MRAM. Additionally, we evaluated the effects the increased on-chip memory on the main memory bandwidth and power reduction.

In Chapter 5, we describe the development of an ultra-multiport SRAM designed to accelerate codebook compression in hardware. The codebook SRAM reduces the main memory bandwidth usage by employing a deep learning quantization algorithm known as deep compression. Codebook compression is a technique that compresses data by clustering deep learning parameters and activations, storing them as indexes corresponding to each cluster. Decompression was achieved by replacing cluster indices with their corresponding representative values. Compression and decompression operations can be implemented using lookup tables, which are well-suited for hardware implementation.

Chapter 6 introduces a synchronization system that leverages Bluetooth to achieve ultralow power consumption and high accuracy. When multiple ultrasonic systems operate in the same space, time synchronization is essential to prevent interference between the emission and reception timings of the ultrasonic waves emitted from each device. To address this issue, we developed an ultralow-power time-synchronization system using BLE. This study employs two techniques, namely timer hybridization and communication scheduling, to achieve sub-microsecond synchronization accuracy with a power consumption of tens of microamperes. Timer hybridization refers to the cooperative operation of a low-power, low-resolution RTC and a high-resolution, high-power, high-frequency timer (high-frequency clock (HFCLK)). By measuring only the time section that requires high accuracy with a HFCLK, power saving and highly accurate time measurements can be realized. Furthermore, power savings can be achieved by scheduling communications to minimize the synchronization frequency using Slave Latency, which is a power-saving feature of BLE. Our MicroSync proposal achieves the highest accuracy and power efficiency among various Bluetooth-based time synchronization technologies.

Chapter 7 describes a 3D reconstruction technique that uses ultrasound and deep learning. In three-dimensional measurement using ultrasonic waves, an object is measured by analyzing the reflected ultrasound waves. Ultrasonic waves modulated with a unique code using direct sequence spread spectrum (DSS) are projected onto a measurement target and reflected from its surface. The reflected waves are measured using a microphone array. Beamforming uses the phase difference between microphones to emphasize sound waves in a specific direction. Delay and sum (DAS), which delays and sums sound signals from a specific direction based on phase differences, emphasizes sound waves exclusively from that direction. ToF refers to the duration between the projection of a sound wave and its reflection and return. By calculating the cross-correlation between the transmitted wave and beamformed sound wave, the similarity to the transmitted signal is determined, with the peak of the cross-correlation indicating the ToF. An ultrasound image is then constructed based on the ToF of each beam. This ultrasound image is fed into a VAE for 3D reconstruction. For the experiments in this study, we constructed a 16-channel ultrasonic measurement system using PYNQ, which is an FPGA board. We then developed beamforming software and implemented a VAE, which is a type of deep learning model. As described above, we established a consistent experimental pipeline for 3D reconstruction using ultrasound.

Finally, Chapter 6 summarizes the conclusions of this study. This paper proposes a noncontact sensing technique based on ultrasound and deep learning technologies for edge AI-based monitoring. It describes research results, including the sensing hardware, a 3D reconstruction model using ultrasonic waves, inference hardware for edge implementation, and time-synchronization module for multi-node applications. These research results are expected to enable privacy-conscious monitoring using edge AI, contributing to a more affluent society where privacy and sensing coexist in harmony.

Table of Contents

Abstract 2

Chapter 1 Introduction 1
1.1 Challenges of monitoring systems with edge AI 1
1.2 Research objectives and structure of this paper 2

Chapter 2 Challenges in Monitoring Systems with Edge AI 5
2.1 Introduction to edge AI 5
2.2 Deep neural network inference 6
2.3 Application areas of edge AI 8
2.3.1 Autonomous driving 8
2.3.2 Mobile phones 8
2.3.3 Robotics 9
2.3.4 Monitoring systems 9

Chapter 3 A Case Study for Improving Performances of Deep-Learning Processor with MRAM 11
3.1 Introduction 11
3.2 Replacing SRAM with MRAM on NVDLA 12
3.2.1 Mamagnetic tunnel junction 12
3.2.2 Comparison of SRAM, STT-MRAM, and SOT-MRAM 13
3.2.3 NVDLA configuration 14
3.2.4 Target task 15
3.3 Simulation environment 17
3.3.1 Memory access simulator 18
3.3.2 Cache emulator 19
3.3.3 Area simulator 20
3.4 Experimental setup 20
3.4.1 Energy consumption parameters 21
3.4.2 Energy and area evaluation 21
3.5 Experimental results 22
3.5.1 Evaluation of CBUF transfer 22
3.5.2 On-chip MRAM performance evaluation 23
3.6 Conclusion 26

Chapter 4 1W8R 20T SRAM Codebook for 20% Energy Reduction in Mixed-Precision Deep-Learning Inference Processor System 27
4.1 Introduction 27
4.2 Previous works 28
4.3 Circuit and architecture 31
4.3.1 Super-multiport SRAM 31
4.3.2 Codebook compression 32
4.3.3 Target processor 36
4.3.4 Memory cell design 36
4.4 Evaluation experiment 38
4.4.1 Simulation evaluation 38
4.4.2 Silicon evaluation 40
4.4.3 Performance simulation 42
4.5 Concluding remarks 42

Chapter 5 High-Accuracy and Low-Power Synchronization Technique Using Bluetooth 44
5.1 Introduction 44
5.1.1 Previous works 47
5.2 Power saving method 49
5.2.1 Hybrid timer 49
5.2.2 Slave Latency 52
5.3 Communication scheduling 53
5.3.1 Synchronization sequence 54
5.3.2 High-Accuracy scheduling 56
5.3.3 Low-power scheduling 59
5.3.4 Simple correction algorithms 62
5.4 Implementation and evaluation 63
5.4.1 Experimental setup 63
5.4.2 Evaluation method 64
5.4.3 Evaluation result 65
5.4.4 Power analysis 69
5.5 Discussion 70
5.5.1 Implementation in monitoring system 73
5.6 Conclusion 74

Chapter 6 3-D Object Reconstruction from Outdoor Ultrasonic Image and Variation

Autoencoder 76
6.1 Introduction 76
6.2 Experimental setup 77
6.3 Ultrasonic imaging 78
6.3.1 Beamforming 80
6.3.2 3D Measurement 80
6.3.3 Variational auto encoder 82
6.3.4 3D Evaluation 84
6.4 Ultrasonic system 85
6.4.1 Measurement system 85
6.4.2 Measurement environment 87
6.5 Experimental result 88
6.5.1 Signal strength distribution and training data 89
6.5.2 Evaluation results of 3D stationary objects 90
6.6 Discussion 91
6.7 Conclusion 92

Chapter 7 Conclusion 94

Acknowledgements 98

Reference 100

List of Figures

Fig. 1.1 Dissertation Structure and Chapter Summaries 3
Fig. 2.1 Performance vs. computational cost of deep learning models. 6
Fig. 2.2 Overview of Deep Neural Network 7
Fig. 3.1 Overview of Magnetic Tunnel Junction. 13
Fig. 3.2 Structure of MRAM 14
Fig. 3.3 Memory architecture of NVDLA. [39] 15
Fig. 3.4 Network structure of SegNet 16
Fig. 3.4 Network structure of U-Net. 16
Fig. 3.5 NVDLA block diagram. 17
Fig. 3.6 Input/Output data[39] 18
Fig. 3.7 Flowchart of DRAM and cache access counting. [39] 19
Fig. 3.8 NVDLA memory structure. [39] 20
Fig. 3.9 Transfer of CBUF for (a) SegNet and (b) U-Net. [39] 23
Fig. 3.10 DRAM transfers and reduction rates for (a) SegNet and (b) U-Net. [39] 24
Fig. 3.11 Performance improvement in SegNet. [39] 25
Fig. 3.12 Performance improvement in U-Net. [39] 25
Fig. 4.1 Structure of Multiport SRAM (a). 8T dual-port (b)1W1R 29
Fig. 4.2 2R2W Gain Cell.DRAM. 30
Fig.4.3 Structure of 2W6R Multiport SRAM. 30
Fig. 4.4 Structure of 1W8R Super Multiport SRAM. [37] 31
Fig. 4.5 The Codebook SRAM macro consists of the proposed Super Multiport SRAM 32
Fig. 4.6 Weight codebook for 16-group compression and decompression. [37] 33
Fig. 4.7 Fine-Tuning Algorithm for Codebook Optimization 33
Fig. 4.8 Weight histograms and representative values. [37] 34
Fig. 4.9 U-Net inference accuracy for the INT16 codebook. [37] 35
Fig. 4.10 Decoder and encoder codebooks in the NVDLA block diagram. oks in NVIDIA NVDLA block diagram. [37] 36
Fig. 4.11 Top: FEOL and bottom: BEOL layouts of 1W8R 20T SRAM . [37] 37
Fig. 4.12 Analysis and evaluation process diagram. 38
Fig. 4.13 SNM simulations: (a) NMOS half-select write, (b) both-side PMOS, and (c) one-side PMOS. [37] 39
Fig. 4.14 Error rates versus supply voltage VDD [37] 39

Fig. 4.15 Silicon and overview of the fabricated chip. [37]....40
Fig. 4.16 Shmoo plots for ;left PMOS read and right NMOS read. [37]....41
Fig. 4.17 Top: Memory macro layout. Bottom: block diagram of the SRAM macro. [37]....41
Fig. 4.18 Performance improvements in energy and area. [37]....42
Fig. 5.1 Time Synchronization in Monitoring Systems. (a) Ensuring Measurement Sequence Consistency, (b) Preventing Multi-Active Sensor Collisions, and (c) Coordinated Multi-Sensor Measurements....44
Fig. 5.2 Overview diagram of the MicroSync synchronization algorithm. [56]....46
Fig. 5.3 Hybrid control of HFCLK timer and RTC. [56]....51
Fig. 5.4 Operation of the decomposed timers **TRTC** and **THFCLK**. [56]....52
Fig. 5.5 Reduction in communication frequency by three in the case of Slave Latency. [56]....53
Fig. 5.6 Flowchart of synchronization scheduling algorithm (a) high-accuracy scheduling, (b) low-power scheduling [56]....54
Fig. 5.7 MicroSync time synchronization sequence. [56]....55
Fig. 5.8 Software sequence diagram for Peripheral-Central communication synchronization. [56] ..56
Fig. 5.9 Implementation of high-accuracy scheduling and power saving by Slave Latency. [56]....57
Fig. 5.10 alculation of synchronization timing in high-accuracy scheduling. [56]....58
Fig. 5.11 Overview of hybridization of high-accuracy scheduling. [56]....59
Fig. 5.12 Communication pipelining in low-power systems. [56]....60
Fig. 5.13 Calculations in low-power scheduling. [56]....61
Fig. 5.14 Flow chart of simple correction algorithm. [56]....62
Fig. 5.15 Experimental setup [56]....63
Fig. 5.16 Connection of measuring instruments. [56]....64
Fig. 5.17 Diagram of the connection between the nRF52840DK and the Power Profiler Kit II. For details on the Power Profiler Kit II circuit, see. [56]....65
Fig. 5.18 Relationship between HFCLK frequency and synchronization error with (a) high-accuracy scheduling and (b) low-power scheduling. Error bars represent standard deviations. [56]....66
Fig. 5.19 Comparison of accuracy and current for each scheduling and timer. [56]....66
Fig. 5.20 Error accumulation of (a) high-accuracy scheduling with 16-MHz HFCLK and (b) low-power scheduling with 4-MHz hybrid timer [56]....67
Fig. 5.21 Measured current consumption waveforms while synchronization cycle with (a) high-accuracy scheduling and (b) low-power scheduling. [56]....68
Fig. 6.1 3D measurement using beamforming and ToF[42]....79
Fig. 6.2 Block diagram of proposed ultrasonic imaFigging system. [42]....79
Fig. 6.3 Beamforming of a rectangular array [42]....80
Fig. 6.4 Direct-sequence spread spectrum [42]....81

Fig. 6.5 Histogram of correlated [42] 82
Fig. 6.6 Network architecture of 3D reconstructed VAE. [42].....83
Fig. 6.7 Ultrasonic measurement system structure. [42].....85
Fig. 6.8 . Ultrasonic array and FPGA. [42].....86
Fig. 6.9 Ultrasonic array device (a) 4x4 microphone array (b) speaker. [42] 87
Fig. 6.10 Measurement environment and measurement system installation. [42] 88
Fig. 6.11 Experiment environment. θ is azimuth angle, φ is elevation angle. [42].....89
Fig. 6.12 3D static target dataset: appearances from ultrasonic array, point clouds, and ground truth. [42].....90
Fig. 6.13 3D reconstruction results and ground truth for each class. . [42] 91

List of Tables

Table 3.1 Comparison SRAM vs MRAM.....14
Table 3.2 Comparison of Memory Technologies in Terms of Energy Consumption and Frequency. [41] 21
Table 3.3 Patterns to evaluate. [41].....22
Table 3.4 Memory area of each pattern. [41].....22
Table 4.1 Comparison of different SRAM cell types based on height, width, and area. 37
Table 5.1 Power Reduction by Hybridizing Timers in Low-power Scheduling[89].....69
Table 5.2 Power Reduction by Hybridizing Timers in High-power Scheduling[89].....68
Table 5.3 Comparison between the proposed and conventional methods [89].....71
Table 6.1 Confusion Matrix binary classification. 84
Table 6.2 Target object size.....90
Table 6.3 Reconstruction Result (3D stationary target) 91
Table 6.4 Comparison of FLOPs and Parameter Sizes for SegNet, U-Net, and the Proposed 3D Reconstruction VAE.....92

Chapter 1
Introduction

1.1 Challenges of monitoring systems with edge AI

Deeplearning has undergone significant advancements in the past decade. Deep neural networks (DNNs) have demonstrated remarkable performance in various domains, including image, natural language, and speech processing. DNNs consist of convolutional and fully connected layers that require significant power and memory bandwidth because of the large number of multiply and accumulate (MAC) operations they perform. Embedded systems face numerous computational resource constraints including power, memory capacity, and bandwidth limitations. Consequently, processors specialized for the efficient execution of deep-learning models were employed. DRAM access accounts for a substantial portion of the power consumption in deep-learning processors. Accessing an external DRAM consumes more than two orders of magnitude more power than on-chip SRAM. Reducing the main memory bandwidth is a critical challenge for achieving power efficiency in deep-learning processors. Minimizing DRAM access by completing operations within an on-chip SRAM is an effective strategy for improving the power efficiency.

Monitoring systems are a type of IoT technology designed to collect information from various environments. This technology enables the detection of abnormalities during disasters or accidents and enables quick responses when deployed in living environments such as urban areas. Additionally, there are conditions where optical cameras alone cannot be used, such as adverse weather like fog or rain that affects optical performance. Three-dimensional (3D) noncontact sensing technologies, such as LiDAR and millimeter-wave radar, are effective in such cases.

These sensors do not collect privacy-sensitive information such as images of people but can accurately capture information essential for anomaly detection, including a person's posture and position. However, these sensors are expensive. High-resolution LiDAR is expensive and often includes a rotating mechanism, making it unsuitable for long-term applications, such as surveillance cameras. Millimeter-wave radar is expensive and faces challenges in capturing three-dimensional data with the same high resolution as an LiDAR. 3D measurement techniques that leverage ultrasonic arrays and beamforming provide cost-effective solutions for this problem. However, previous studies on ultrasonic 3D reconstruction techniques have primarily focused on indoor short-range targets, with limited research addressing outdoor long-range targets.

Ultrasonic sensing is more susceptible to interference than millimeter wave or LiDAR sensing. Active ultrasonic measurements estimate the distance to an object by calculating the time difference between emitted and reflected ultrasonic waves. However, in environments with multiple sensors, ultrasonic waves can interfere with each other, leading to a reduced accuracy. To address this issue, synchronizing multiple sensors and preventing the overlapping of transmission and measurement timings are effective. Time synchronization services in edge systems should prioritize low-power consumption. High-accuracy timers generally consume more power, and frequent communication for time synchronization further increases the power consumption.

1.2 Research objectives and structure of this paper

Building on the background described in the previous section, we present four studies on edge-AI-based monitoring systems from three perspectives: hardware, software, and practical applications. The structure and overview of each study are briefly described below. Fig. 1.1 illustrates the organization of this paper.

Chapter 1 This section provides an overview of the technical background of this study and outlines its structure.

Chapter 2 Discusses monitoring systems leveraging edge AI and their associated challenges.

Chapter 3 Explains performance enhancements in deep learning processors, achieved by replacing main memory with MRAM.

Chapter 4 Describes main memory compression techniques for deep learning processors employing Codebook SRAM.

Chapter 5 Discusses deep learning models for three-dimensional object reconstruction from ultrasonic images.

Chapter 6 Explains ultralow-power time synchronization techniques for coordinating and operating multiple ultrasonic arrays.

Concludes the paper by summarizing the findings and discussing future directions.

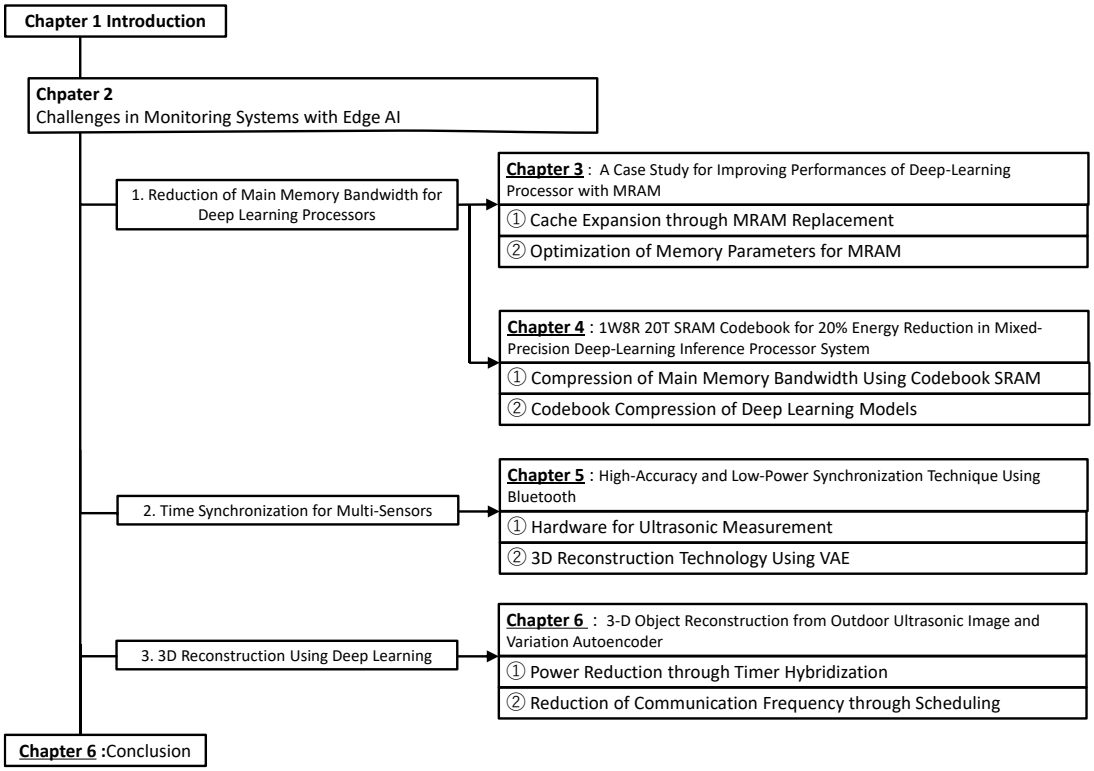


Fig. 1.1 Dissertation Structure and Chapter Summaries

Chapter 2
Challenges in Monitoring Systems with Edge AI

This chapter describes the application areas and monitoring systems of edge AI and addresses the challenges related to power savings.

2.1 Introduction to edge AI

Edge AI refers to a technology that deploys machine-learning models such as DNNs on edge devices. DNNs have demonstrated high accuracy across various tasks, ranging from signal processing (e.g., acoustic event detection) to natural language processing, image classification, object detection, skeletal estimation, and semantic segmentation. Therefore, the deployment of DNNs in embedded systems enables the realization of a wide range of applications.

Edge AI offers three major advantages over cloud-based inference. The first advantage is real-time processing. Edge AI eliminates the need to transfer sampled data to a server by completing computations on an edge device, thereby eliminating communication delays. Additionally, the system remains operational even during communication losses, thereby ensuring high reliability. It is suitable for applications that require real-time processing, such as autonomous driving and robotics. Second, it reduces the communication bandwidth usage. Raw data samples are typically large. Processing the data at the edge and transferring only the results to the server reduces the network traffic and enables deployment on slower networks in remote areas. Third, Edge AI enhances privacy protection. Processing raw personal information, such as vital data, locally on an edge device eliminates the need to upload sensitive data to a server, thereby mitigating privacy concerns. In summary, edge AI has clear advantages over cloud-based inference.

As described in Chapter 1, integrating deep-learning models into embedded systems is constrained by computational resource limitations, including power consumption and memory capacity. As shown in Fig. 2.1, the parameter size of deep learning models has been increasing annually, correlating with performance improvements. Therefore, specialized deep-learning processors for DNN models have been employed to implement edge AI.

Unlike general-purpose processors, specialized processors feature dedicated arithmetic units, data buses, and memory structures optimized for convolutional layers, fully connected layers, and activation functions. Deep learning processors can be categorized into two types. The first is a learning processor, which accelerates

both the training and inference of deep learning models. The second type is an inference processor designed specifically for the inference of deep-learning models. Edge AI primarily utilizes inference processors.

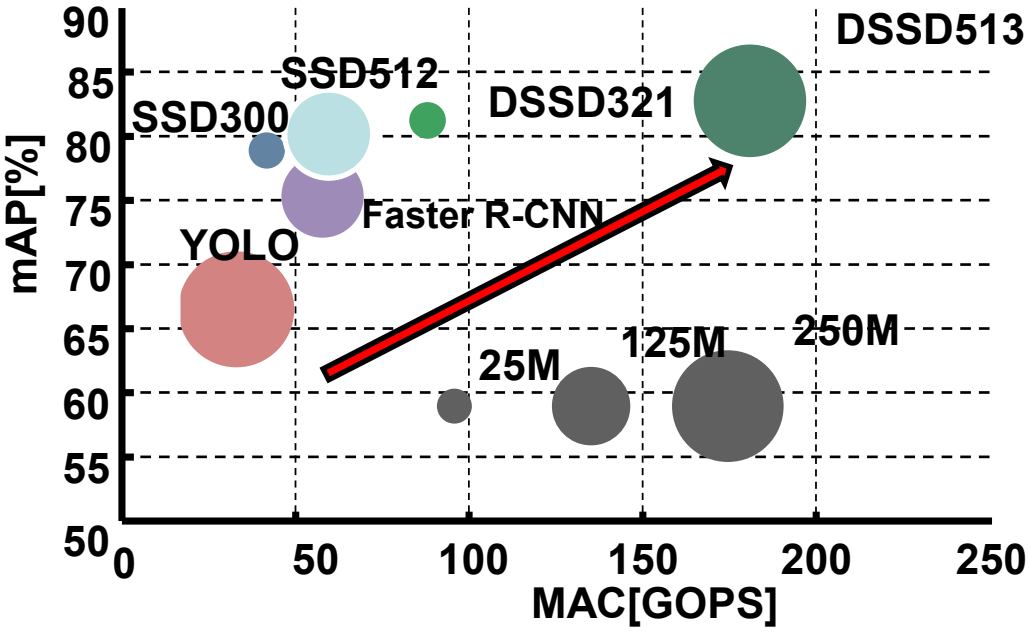


Fig. 2.1 Performance vs. computational cost of deep learning models.

2.2 Deep neural network inference

Fig. 2.2 shows an overview of a neural network. A neural network is an algorithm inspired by the cranial nerves, consisting of neurons (nodes) and synapses (connections between neurons). Neurons represent nodes in a graph, whereas synapses are directed edges connecting them. Each synapse is assigned a connectivity strength (weight), and the signals passing through the synapses are multiplied by this weight and summed at the destination neuron. When the summed input exceeded a certain threshold, the neuron activates and transmits the signal to the next synapse. A DNN is a neural network comprising three or more layers.

Neural network tasks can be broadly categorized into classification and regression problems. Classification involves assigning input data to predefined classes. Examples include image classification tasks such as MNIST and CIFAR-10. Regression problems involve predicting output data that closely matches teacher data based on input data. Object detection tasks, such as YOLO, simultaneously estimate object positions and classify their classes, combining the characteristics of both classification and regression problems.

Inference in a DNN takes the input data as a tensor and derives results by sequentially computing through various layers such as convolutional and fully connected layers. These layers perform operations such as

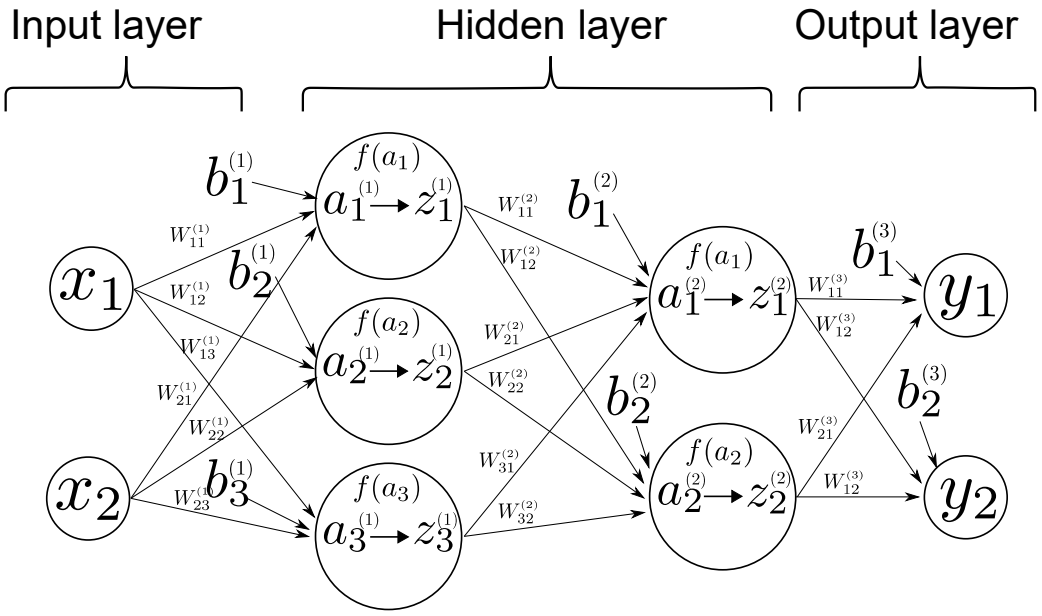


Fig. 2.2 Overview of Deep Neural Network

multiply-and-accumulate (MAC), apply activation functions such as ReLU and Sigmoid, and use techniques like pooling layers and batch normalization, all from the input layer to the output layer.

In Fig. 2.2, $[x_1 \quad x_2]$ represents the vector of the input data. Each line in the figure represents a synapse and is associated with a weight (coefficient). The values passing through these lines are multiplied by the corresponding synaptic weights as coefficients. The results of these operations are summed at the target neuron. Subsequently, a bias is added. This sum, denoted as $a^{(n)}$ in the figure, represents the result of a multiple-sum operation. A bias $b^{(n)}$ is then added to the sum. Subsequently, the result undergoes a nonlinear operation via an activation function represented as f.

Although an inner product operation with a layered structure can theoretically be expressed in a single layer by computing the product of the weights, inserting nonlinear operations between the layers enables the network to represent complex functions that cannot be achieved with a single layer. This significantly enhances the expressive power of the network.

Such a network, which consists of four or more layers, is called a deep neural network (DNN). The network operations shown in Fig. 2.2 are expressed as Equations (2.1)–(2.5). As shown in Equations (2.1), (2.3), and (2.5), the data input to the neural network, as well as the activation weights and synapse outputs from each hidden layer, can be represented as inner product operations between matrices. The activation functions in Equations (2.2) and (2.4) are applied to each element of the matrix.

$$\begin{bmatrix} \boldsymbol{a_1^{(1)}} & \boldsymbol{a_2^{(1)}} & \boldsymbol{a_3^{(1)}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x_1} & \boldsymbol{x_2} \end{bmatrix} \begin{bmatrix} \boldsymbol{W_{11}^{(1)}} & \boldsymbol{W_{12}^{(1)}} & \boldsymbol{W_{13}^{(1)}} \\ \boldsymbol{W_{21}^{(1)}} & \boldsymbol{W_{22}^{(1)}} & \boldsymbol{W_{23}^{(1)}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{b_1^{(1)}} & \boldsymbol{b_2^{(1)}} & \boldsymbol{b_3^{(1)}} \end{bmatrix} \tag{2.1}$$

$$\begin{bmatrix} z_1^{(1)} & z_2^{(1)} & z_3^{(1)} \end{bmatrix} = f\left(\begin{bmatrix} a_1^{(1)} & a_2^{(1)} & a_3^{(1)} \end{bmatrix}\right) \quad (2.2)$$

$$\begin{bmatrix} a_1^{(2)} & a_2^{(2)} \end{bmatrix} = \begin{bmatrix} z_1^{(2)} & z_2^{(2)} & z_3^{(2)} \end{bmatrix} \begin{bmatrix} W_{11}^{(2)} & W_{12}^{(2)} \\ W_{12}^{(2)} & W_{22}^{(2)} \\ W_{13}^{(2)} & W_{23}^{(2)} \end{bmatrix} + \begin{bmatrix} b_1^{(2)} & b_2^{(2)} \end{bmatrix} \quad (2.3)$$

$$\begin{bmatrix} z_1^{(2)} & z_2^{(2)} \end{bmatrix} = f\left(\begin{bmatrix} a_1^{(2)} & a_2^{(2)} \end{bmatrix}\right) \quad (2.4)$$

$$\begin{bmatrix} y_1 & y_2 \end{bmatrix} = \begin{bmatrix} z_1^{(2)} & z_2^{(2)} \end{bmatrix} \begin{bmatrix} W_{11}^{(3)} & W_{12}^{(3)} \\ W_{12}^{(3)} & W_{22}^{(3)} \end{bmatrix} + \begin{bmatrix} b_1^{(2)} & b_2^{(2)} \end{bmatrix} \quad (2.5)$$

2.3 Application areas of edge AI

2.3.1 Autonomous driving

Autonomous driving is a major field in edge AI, and the demand for advanced image processing using deep-learning models in edge devices increases annually. Therefore, various companies are developing deep-learning processors for autonomous driving. Various deep learning processors have been proposed, including high-performance, high-power processors such as NVIDIA's Orin to power-efficient models like the Renesas R-Car series. Real-time image processing is particularly important in autonomous driving. A car driving at 40 km/h with 30 FPS image processing covers a distance of 37 cm per frame. In addition, as modern autonomous cars are equipped with multiple cameras, the cost of image processing increases. Hence, multimodal processing using sensors other than cameras, such as LiDAR, is required. Thus, autonomous vehicles must perform numerous computations using limited computational resources, as described in Chapter 1. This creates a strong demand for power-efficient processors with high-performance.

2.3.2 Mobile phones

In recent years, most high-end mobile devices such as pixels and iPhones are equipped with neural processing units (NPUs). Smartphone camera performance has been steadily improving, leading to a growing demand for post-processing technologies for capturing images. Google Pixel was designed with an emphasis on NPUs, enabling it to run Gemini Nano, a smaller version of Gemini and Google's large language model (LLM), on its NPU. Deep learning models are also used in biometric authentication technologies such as iris and face recognition, entertainment applications such as image processing, augmented reality and voice recognition, and various other applications. Deep learning processors for autonomous vehicles do not require a throughput as high as that of AI chips for other applications; however, they are battery-powered and demand high power efficiency.

2.3.3 Robotics

In recent years, robots have been introduced into various fields, with automated guided vehicles (AGVs) and robotic arms being actively utilized in logistics. Additionally, research on the use of drones to maintain infrastructure is advancing. These technologies require advanced robotic vision, making edge AI, capable of high-speed image processing, a critical component. Moreover, in recent years, machine-learning-based control, rather than model-based control, has become a major focus in robotics research. The importance of DNNs in robotics is increasing, and the demand for deep-learning processors is expected to grow to achieve low-latency machine-learning-based control.

2.3.4 Monitoring systems

Monitoring systems are key applications of edge AI. Edge AI can be used for data analysis and the detection of anomalies in monitoring systems. Raw data obtained from sensors in monitoring systems cannot be utilized directly. Thus, signal or image processing is necessary to analyze and utilize the data. Deep learning has achieved significant success in signal and image processing, and it has been applied to various tasks, including anomaly detection and information interpretation. Therefore, the implementation of such models in edge AI offers several benefits. Processing data at edge devices reduces the latency from sampling to action. Edge AI reduces the communication bandwidth required for data transmission, unlike many cloud-based deep learning models that rely on graphic processing unit (GPUs) in the cloud and require data to be sent to a server. Furthermore, it eliminates the need to send sensitive data privately to clouds.

Chapter 3
A Case Study for Improving Performances of Deep-Learning Processor with MRAM

3.1 Introduction

In recent years, convolutional neural networks (CNNs) have made significant contributions to improving the recognition accuracy in the field of image recognition [1]. Various object detection algorithms based on CNNs have been developed, and their performances have improved annually [2-5]. However, high-precision CNNs have numerous parameters and require considerable memory [1] In many high-precision CNNs, it is not possible to store all the weights and feature maps in the on-chip memory. CNNs require extensive memory access, and communication with off-chip dynamic random-access memory (DRAM) is essential, which is an issue. Accessing off-chip DRAM consumes more than 100 times more energy than accessing on-chip static random-access memory (SRAM) [6]. In addition, the access speed is slow; therefore, the amount of memory communication is a very important issue in embedded systems. Quantization and network weight reduction have been studied as effective methods reducing main memory bandwidth. However, most of these approaches are software-based [7, 8]. We focused on a hardware approach. Access to off-chip DRAM is suppressed by increasing the capacity of the on-chip memory. Increasing the number of on-chip memories using SRAM, which is the current mainstream memory technology, imposes a significant cost burden. In a previous study, the processor used a large-capacity embedded SRAM, which occupied more than 50% of the silicon area [9]. Therefore, we replaced the SRAM with magnetoresistive random access memory (MRAM) [10] to increase the on-chip memory capacity.

This Chapter presents the results of simulation case study where SRAM was replaced with MRAM in a deep-learning processor. This simulation demonstrated the power and area performance that could be achieved by replacing the cache and buffers with MRAM. In this case study, we had the goal of identifying the performance improvements and side effects of replacing SRAM with MRAM for caches and buffers. It was estimated that performance increases could be expected by changing from SRAM to MRAM because the capacity could be doubled in the same area. NVDLA, an NVIDIA deep learning processor [11], was used as the motif processor. NVDLA is a commercially available processor with a cache and general adder-tree-type array of operators. As target applications, we adopted CNN models for segmentation inference tasks with large memory transfers that required energy efficiency. Prior work has shown that replacing SRAM with MRAM in small models can achieve higher speeds and lower power consumption by completing all the

operations inside the processor without DRAM access [12]. However, we set up more general tasks and hardware using DRAM. We also focused on the segmentation task, which is difficult to implement on-chip in real time owing to its high memory bandwidth requirements for the image-processing tasks required in embedded systems, such as automated driving and factory automation. The target processor selected was a general-purpose processor that utilized DRAM and cache structures. MRAM not only doubles the area but also consumes different amounts of time and power for reads and writes, and the effect of these on the performance in a processor with non-deterministic behavior such as a cache had to be considered in the simulation.

We developed three programs to estimate the performance improvement by replacing the internal memory of the NVDLA with MRAM. These were a (1) memory-access simulator, (2) cache emulator, and (3) area simulator. FireSim [13-15] (FireSim-NVDLA) and Virtual Platform [16] (VP-NVDLA) are publicly available NVDLA simulators. However, FireSim-NVDLA, the hardware simulator of the NVDLA, supports only YOLO and is incapable of executing SegNet [17] and U-Net [18], which were the targets of this project. In addition, VP-NVDLA, which is a software simulator written in System C, does not have a cache structure and cannot acquire DRAM access. Furthermore, a power estimation and latency evaluation could not be performed using only the existing simulators designed for software execution. Therefore, to execute SegNet and U-Net on the NVDLA and evaluate the DRAM bandwidth, it was necessary to develop original simulators and emulators. (1) The memory access simulator estimated the memory bandwidth and power consumption based on the memory size and number of operations in each layer. The existing simulators are designed for software emulation or are run on FPGAs. Consequently, they cannot directly simulate an MRAM replacement and estimate the power. Therefore, it was necessary to develop a memory access simulator. (2) A cache emulator was used to calculate the cache hit ratio. It calculated the hit rate when a cache was installed for memory access logs sampled from the VP-NVDLA, which does not have a cache structure. (3) The area simulator estimated the circuit area when the NVDLA was implemented based on the set parameters.

3.2 Replacing SRAM with MRAM on NVDLA

3.2.1 Mamagnetic tunnel junction

Fig. 3.1 shows the structure of a magnetic tunnel junction. SRAM is commonly used as embedded memory in general-purpose LSIs and as cache memory in processors because it can be implemented in a standard CMOS process and can be mixed with logic circuits. SRAM is characterized by high-speed operation and low power consumption; however, the large number of elements per cell and the complexity of the internal structure make it difficult to achieve high density.

MRAM uses a magnetic tunnel junction (MTJ) to hold data. The MTJ device consists of a thin insulating layer of magnesium oxide (MgO) sandwiched between two ferromagnetic layers, one of which has a fixed

magnetization and is called the pinned layer or reference layer [10]. The other layer is called the free layer because its magnetization can be freely reversed by changing the direction of the current flowing through the MTJ device. When the magnetization direction of the free layer is parallel to that of the fixed layer (P), the resistance decreases, and when the magnetization has the opposite direction (AP), the resistance increases. The logic states “1” and “0” are stored using these high and low resistance values, respectively.

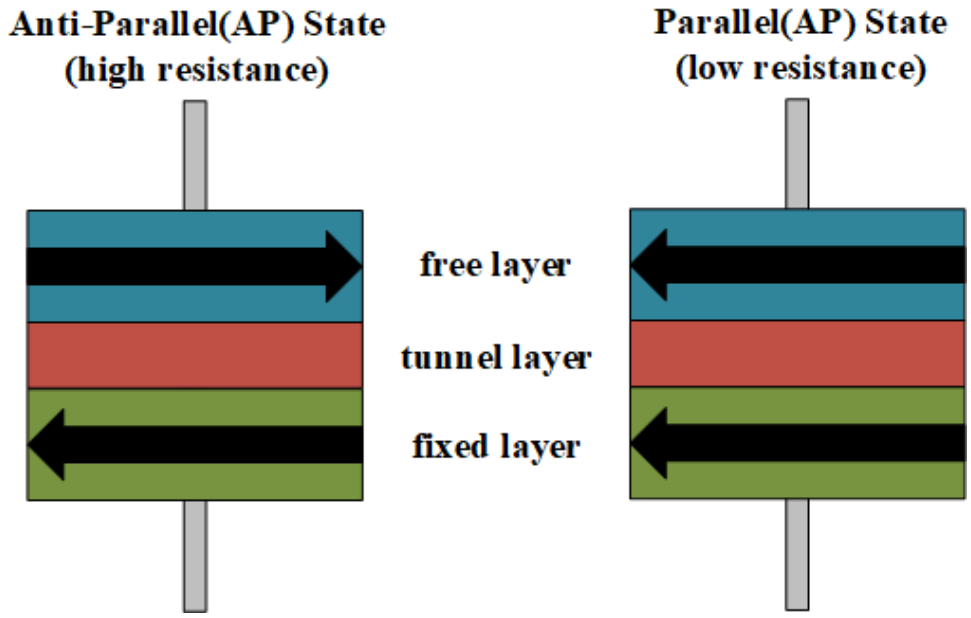


Fig. 3.1 Overview of Magnetic Tunnel Junction.

3.2.2 Comparison of SRAM, STT-MRAM, and SOT-MRAM

Fig. 3.2 shows the structure of STT-MRAM and SOT-MRAM. Spin transfer torque MRAM (STT-MRAM) consists of one MTJ device and one access transistor. A datum is read and written through two terminals: the bit line and source line [19]. STT-MRAM is expected to be a candidate for the next-generation on-chip memory because it is a nonvolatile memory with a small number of components. However, because the magnetization of the free layer is reversed when the data are written, the power consumption is seven times higher than that of SRAM, and the write latency is five times higher [20]. Spin-orbit torque MRAM (SOT-MRAM) is a cell structure that uses MTJ elements. Unlike STT-MRAM, SOT-MRAM consists of three terminals, and the read and write buses are independent. In SOT-MRAM, the magnetization of the free layer is inverted by passing a current through the channel layer, thereby improving the write power and latency, which are the disadvantages of STT-MRAM [20].

Table 3.1 shows comparation SRAM and MRAM .The main difference between SRAM and MRAM is the number of elements; SRAM consists of six transistors, whereas MRAM consists of one transistor and one MTJ element. Therefore, MRAM can provide twice the capacity of SRAM in the same area. However, its

macro area differs significantly depending on the memory capacity. This is because when the memory macro is broken down into the cell and peripheral areas (write circuit, decoder, and sense amplifier), the peripheral areas become relatively larger in MRAM than in SRAM because of the complicated read-and-write circuitry [20]. When the memory capacity is 64 KB or less, the peripheral area dominates, making MRAM less area-efficient than SRAM. The area efficiency increases with the memory capacity compared to SRAM. The main difference between STT-MRAM and SOT-MRAM is the write operation. Compared with STT-MRAM, which requires a large current to flow through the MTJ itself to flip the magnetization of the free layer, SOT-MRAM, which does not require such a current, is comparable to SRAM in terms of its write energy and write latency.

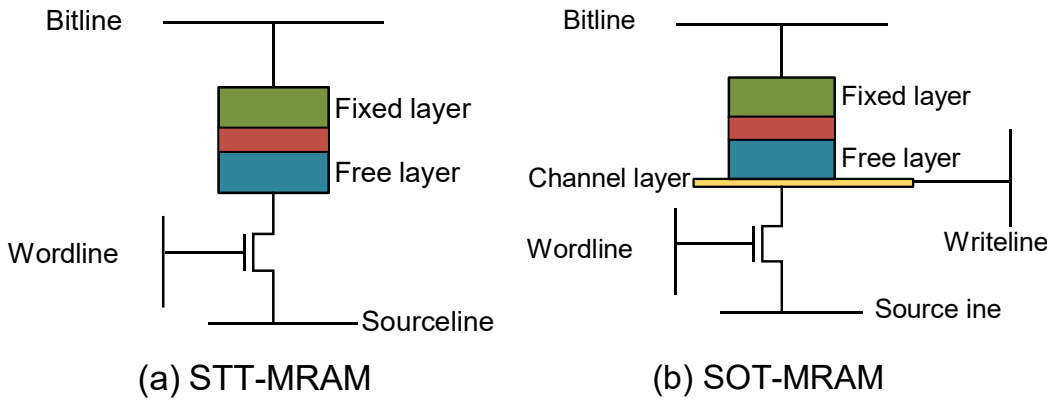


Fig. 3.2 Structure of MRAM

Table 3.1 Comparison SRAM vs MRAM

	6T-SRAM	STT-MRAM	SOT-MRAM
Non-volatility	No	Yes	Yes
Area [mm^2]	2.78	1.63	1.80
Read latency [ns]	2.17	1.2	1.13
Write latency [ns]	2.07	11.22	1.36
Read access energy [pJ]	587	260	247
Write access energy [pJ]	355	2337	334
Leak power [mW]	937	387	254

3.2.3 NVDLA configuration

As a case study, we estimated the performance improvement achieved when the internal memory of a deep-learning processor was changed from SRAM to MRAM. We used the NVDLA deep-learning processor as the motif processor. The NVDLA produced by NVIDIA [16] is an inference processor that is commercially available as an open source. We developed a program based on the access logs of the NVDLA hardware

simulator to estimate the performance of the MRAM replacement. Replacing the NVDLA SRAM with MRAM increased the memory capacity. However, NVDLA is a cache processor, and its power and area performance vary depending on the program settings.

In this case study, VP-NVDLA was used to obtain memory access logs; however, VP-NVDLA is a behavioral model without a hardware structure and therefore has uncertainty in its memory bandwidth. In this case study, a cache emulation was performed based on the memory access logs in the areas corresponding to the areas between the cache and convolution buffer, as shown in Fig. 3.3, and between the post-processor and cache.

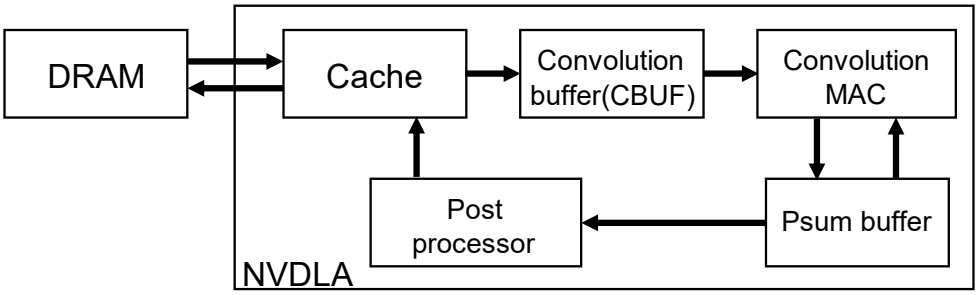


Fig. 3.3 Memory architecture of NVDLA. [40]

3.2.4 Target task

An autoencoder was employed for the target task. Although autoencoders involve a relatively large number of operations, they are well-suited for various edge AI tasks, including image processing applications such as semantic segmentation and data transformation.

We used SegNet and U-Net as the target tasks to evaluate the performance improvement. These are the typical models used in segmentation tasks. This case study aims to decrease the main-memory bandwidth by increasing the cache capacity. Therefore, we selected segmentation tasks that require more memory bandwidth compared to detection and classification tasks and have a greater execution demand in embedded environments. Each network was trained on the semantic segmentation contained in the CityScapes dataset [21] for automated driving.

Fig. 3.4 shows the network architecture of SegNet. SegNet is a type of autoencoder that consists of an encoder and a decoder. The encoder and decoder are composed of blocks that include a convolution layer, activation functions, and batch normalization. The encoder includes 2 × 2 max pooling operations, and the decoder includes unpooling operations. These blocks are paired, and the maximum value indices from the max pooling are used during unpooling.

Fig. 3.5 shows the network architecture of U-Net. U-Net was developed for semantic segmentation in medicine. Owing to its high performance, it is currently applied to a variety of tasks, including the computation

of heat maps for object detection and generative AI applications such as Stable Diffusion. Because U-Net is an autoencoder, its structure is similar to that of SegNet, and the encoder is almost identical. Unlike SegNet, the decoder part does not use unpooling but instead employs upconvolution and includes concatenation with activations from earlier convolution layers.

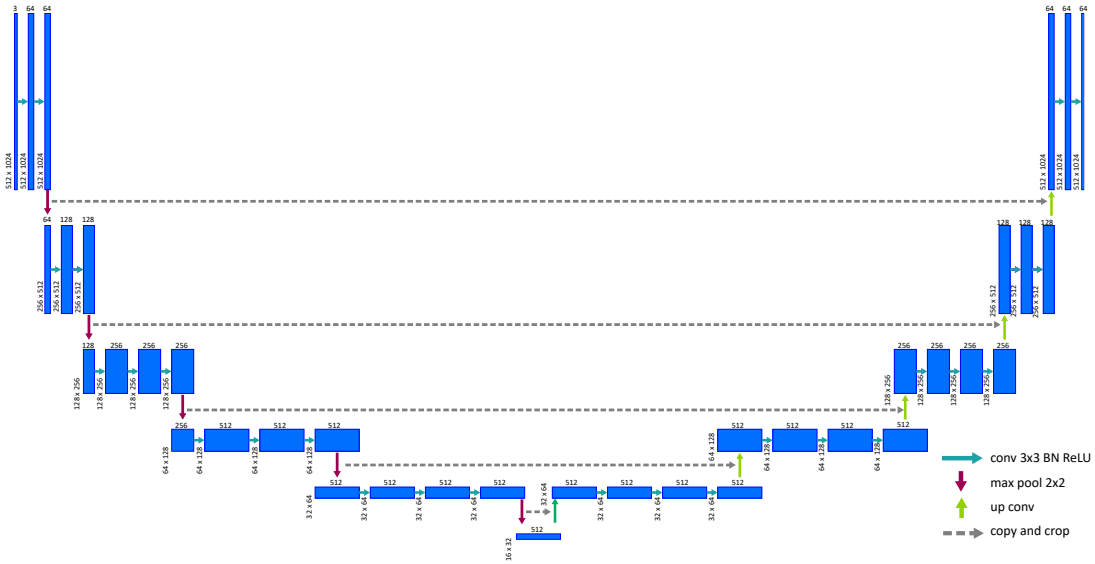


Fig. 3.4 Network structure of SegNet

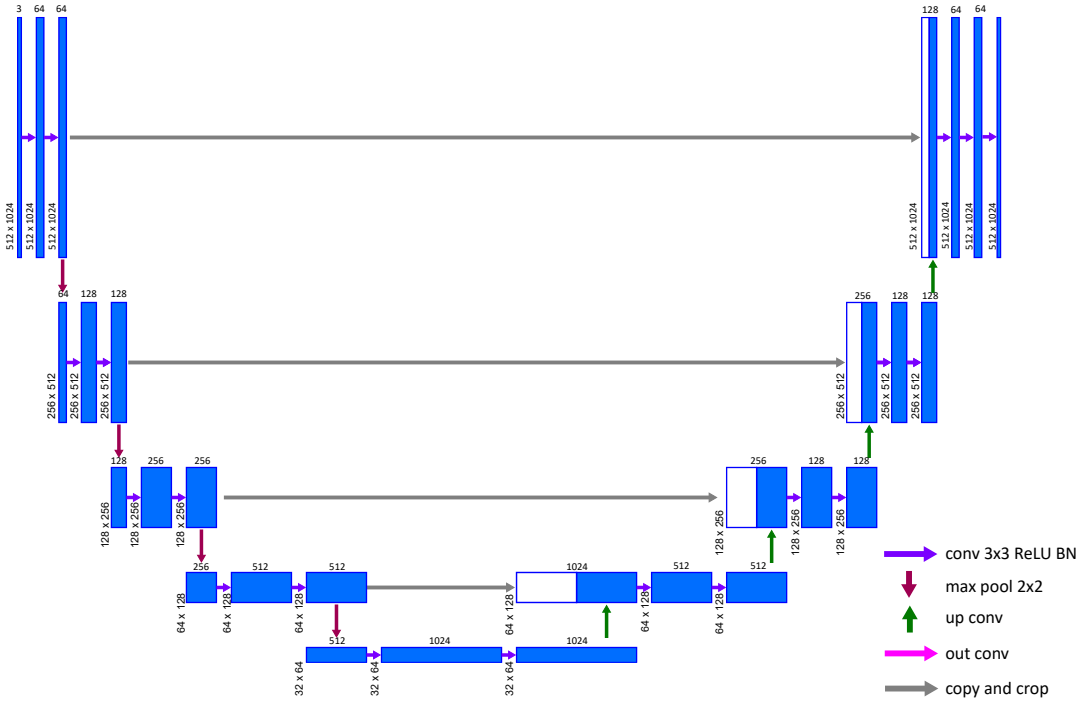


Fig. 3.5 Network structure of U-Net.

This project used SegNet and U-Net as targets. SegNet contained five upsampling steps. As previously mentioned, the NVDLA does not support upsampling; therefore, the bandwidth could not be calculated for these five layers. Thus, we assumed that upsampling was processed on the CPU side and that the data in the cache was written back to the DRAM after the beginning of CPU-side processing. However, for U-Net, the portion of SegNet corresponding to upsampling is composed of upconvolution; therefore, all processing is completed in NVDLA. Therefore, the cache is not written back into it. In SegNet, convolution and pooling are treated together as a single layer, and processing can be completed simultaneously in the NVDLA. In contrast, U-Net cannot perform convolution and pooling simultaneously because an activation map is required for the decoder before pooling. Therefore, each deconvolution layer is processed separately and stored in a cache or DRAM. Therefore, when calculating the DRAM bandwidth per layer, SegNet is divided into 26 layers because pooling and convolution are performed simultaneously, while upsampling is processed by the CPU. In U-Net, pooling and convolution are separate, and the upconvolution can be processed by NVDLA; therefore, it is divided into 27 layers.

3.3 Simulation environment

FireSim is a hardware simulation environment that uses an Amazon Web Services (AWS) cloud field-programmable gate array (FPGA)[14, 15]. As a development environment, a Xilinx Virtex UltraScale+ VU9P FPGA board and Vivado, an FPGA integrated design tool, were used for logic synthesis, and the place and route process. The configuration data of FireSim have a unique format [Amazon Global FPGA Image (AGFI)]; therefore, it cannot be designed and developed in a local environment. Currently, the precompiled models released by NVIDIA researchers are available on GIT, and only the YOLOv3 [5] network can be verified. Instead, we used the FireSim-NVDLA environment developed in a previous study [13]. In the FireSim environment, the NVDLA only supports inferences with INT8 accuracy. The cache is shared with the Rocket Core, and a set-associative write-through method is adopted for the cache. The number of methods, number of sets, block size, and cache capacity can be changed by changing the parameters.

Fig. 3.6 shows the NVDLA model for the simulations in this paper. The operation is performed in the following steps. First, input and weight data are read from DRAM and stored in the cache. Second, the data is

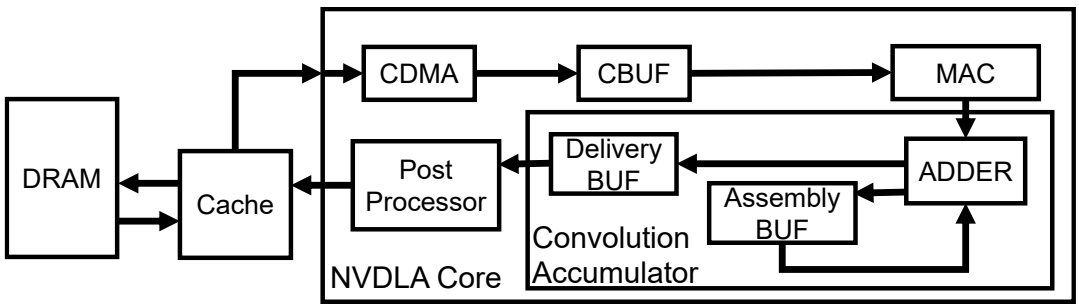


Fig. 3.6 NVDLA block diagram.

read from the cache to the CDMA, formatted in the order required for the convolution process, and stored in the Convolution Buffer (CBUF). Third, after the MAC operation, the partial sums are written to the Assembly Buffer, and the data after the partial sums are processed are sent to the Delivery Buffer. After that, after applying activation functions and pooling, the activation is written back to the cache, and the final output is written back to the DRAM.

3.3.1 Memory access simulator

There are two major differences between the FireSim-NVDLA and original NVDLA. First, the operation of arbitrarily transferring data to a convolution buffer (CBUF) cannot be verified. The second factor is the capacity of the partial sum buffer. The FireSim-NVDLA has some architectural differences compared to the original but is still worth using for customization and profiling.

FireSim-NVDLA does not support an upconvolution layer. VP-NVDLA is uncertainty in the DRAM transfer. The internal transfer of the original NVDLA was estimated by considering the data acquired using YOLOv3 in the FireSim environment. For DRAM and cache transfer, we created a program generate the address log between the DRAM and NVDLA based on VP-NVDLA. The address-mapping method for the input and output to the DRAM is shown in Fig. 3.7, and is described in [11]. The same mapping method was used to determine the weights. The inputs, outputs, and weights were mapped to separate areas in the DRAM; therefore, the starting addresses were different. Fig. 3.7 shows the case in which the precision was set to INT8, which means that each data point was one byte. In this case, the NVDLA had 32 output channels; therefore, the data were transferred from the NVDLA to the cache in units of 32 B × 24 squares. Additionally, the input transferred the number of input channels from the cache to the NVDLA. Based on the memory bandwidth data obtained from VP-NVDLA and the architecture data obtained from FireSim-NVDLA, we developed programs to calculate the latency, memory bandwidth, and power consumption of U-Net and SegNet running on the NVDLA. The program could calculate the performance for a given memory type such as MRAM or

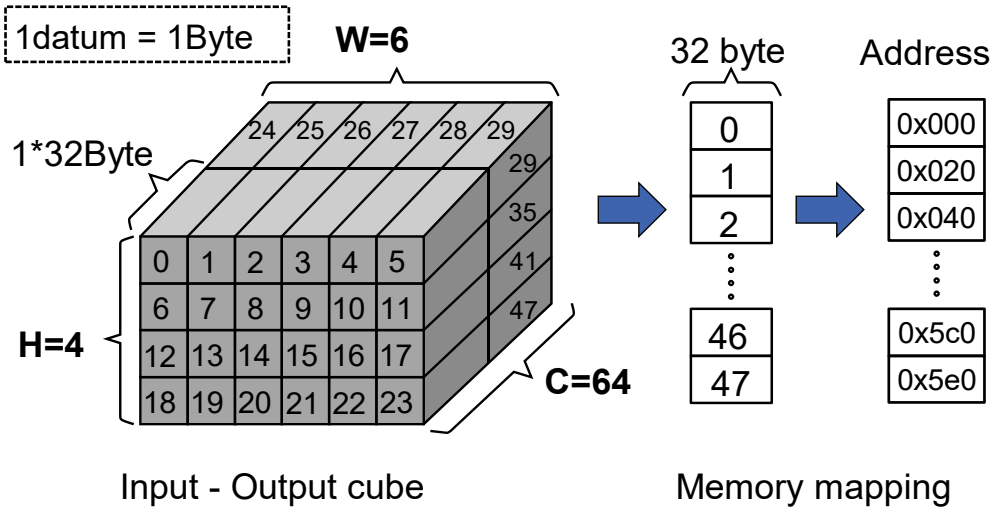


Fig. 3.7 Input/Output data[40]

SRAM, and arbitrary cache and CBUF sizes. Based on the results obtained with FireSim-NVDLA and VP-NVDLA, SegNet and U-Net were profiled at the layer level to estimate the power consumption of the components of the NVDLA, such as the MAC units, CBUF, cache, and DRAM. The cache hit rates used in the power calculations for the memory-access simulator were derived from the results of the cache emulator described in next section.

3.3.2 Cache emulator

We created a program to calculate the DRAM and cache transfers based on the address logs generated by the VP-NVDLA. This program emulates the behavior of the cache memory using the memory access log sampled from the VP-NVDLA. The point at which the address log of DRAM accesses is sampled corresponds to the boundary between the NVDLA Core block connected to the CDMA and Post-Processor and the external memory in Fig. 3.3. Fig. 3.8 shows how the numbers of read/write accesses to the DRAM and cache were counted (cnt is the number of read/write accesses to each memory, and line_num is the number of lines in a block). The block size was 64 B, and the line length was 32 B. Therefore, line_num was 2.

The cache used the same set-associative write-through method used in the FireSim environment. When exchanging data between the DRAM and cache, the data were moved in blocks, whereas the data were moved in lines between the cache and NVDLA. If a hit occurred when referring to data in the cache, the DRAM was not accessed regardless of whether it was read or written. When data were evacuated from the cache to the DRAM, the evacuation method was randomly selected. If there was an empty path, or if the data were not

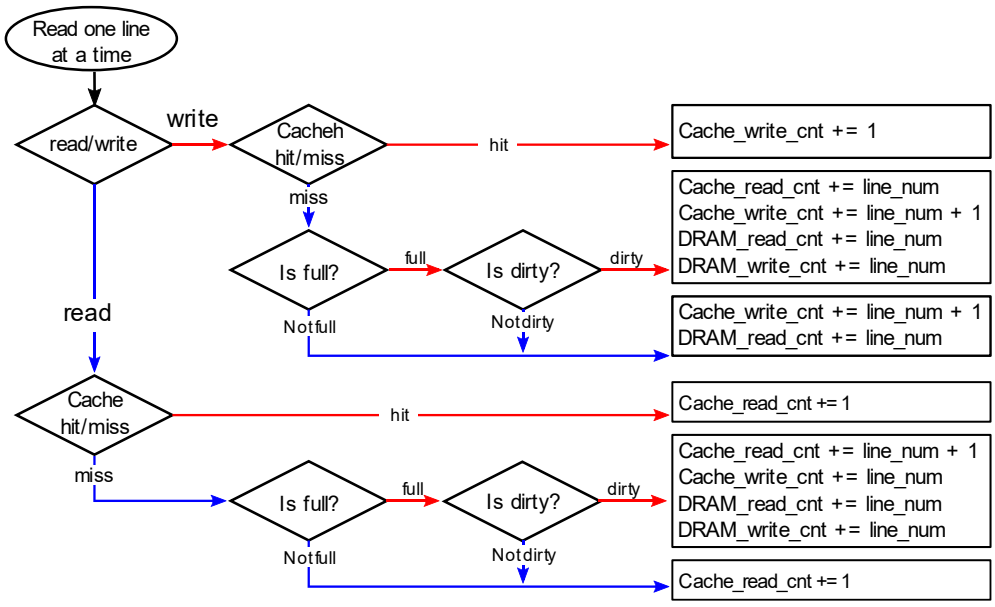


Fig. 3.8 Flowchart of DRAM and cache access counting. [40]

written back to the DRAM, the data were only evacuated if the path to be evacuated had been rewritten. The data for evacuation were not rewritten.

3.3.3 Area simulator

An area simulator was developed to estimate the MRAM replacement area. The area simulator could estimate the area of the NVDLA processor circuit by specifying the cache and CBUF capacities, the replacement or not, and along with whether the STT-MRAM, SOT-MRAM, or SRAM was used. The estimation also considered peripheral circuits, including arithmetic units and other buffers. The area simulation focuses on CBUF, Assembly Buffer, Delivery Buffer, and Cache among the elements shown in Fig. 3.3. This is because these elements are dominant in terms of circuit area. The circuit area in the area simulation is referenced from our previous processor [9], and the MRAM area is referenced from [20].

3.4 Experimental setup

Fig. 3.9 shows the simple memory structure in which the SRAM was replaced with MRAM in this study. The capacities of the assembly buffer and delivery buffer were fixed at 8.5 KB and 8 KB, respectively, and the CBUF was calculated to be 512 KB when SRAM was used, and 1 MB when MRAM was used, in order to keep the areas similar. The cache size was calculated using capacities of 1, 2, 4, 8, and 16 MB.

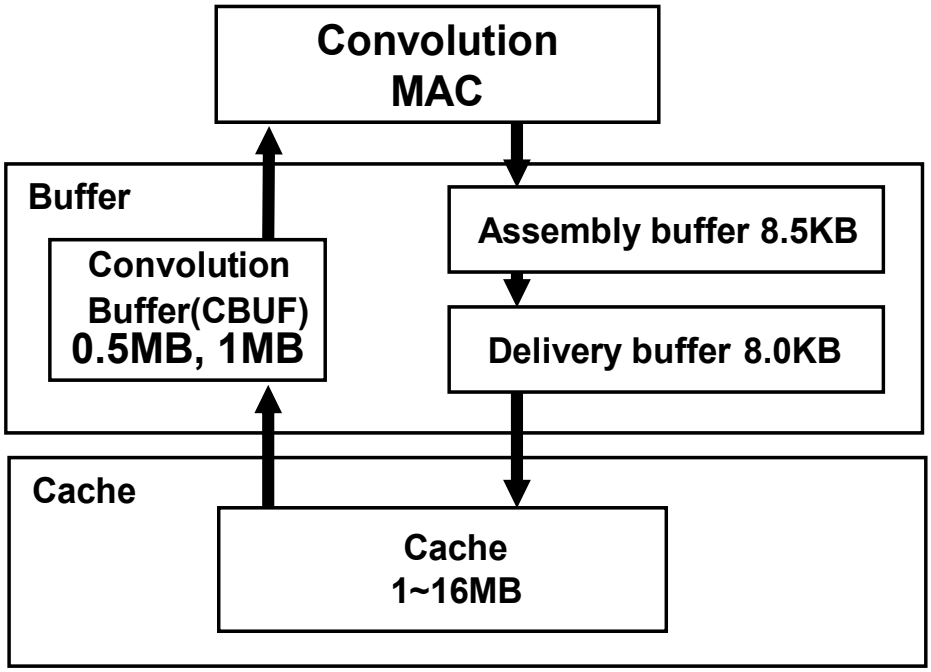


Fig. 3.9 NVDLA memory structure. [40]

3.4.1 Energy consumption parameters

Table 3.2 summarizes the data used to calculate the energy consumption and operating frequencies. The 8-bit arithmetic and 8-bit access energy values of the DRAM and SRAM were obtained by converting the data into an 8-bit format [6, 20]. Thus, the energy consumptions of the SRAM, STT-MRAM, and SOT-MRAM were derived from the values reported in [20] and [22]. In the STT-MRAM, switching from P→AP consumed less energy than switching from AP→P. Therefore, we could reduce the energy by assigning the data with a high frequency of "0" and "1" values to P→AP. We tested this on SegNet with an input size of 416 × 416 and a cache of 1 MB and found that "0" values accounted for 69% of the total. Therefore, the optimization was performed by assigning "0" values to P→AP and "1" values to AP→P. The read and write power values when using the STT-MRAM were obtained by multiplying the P→AP and AP→P values by 0.69 and 0.31, respectively, and then adding them together. We also applied the same assignments to the U-Net. This was because the data written to the memory by a deep-learning processor are predominantly associated with activation. Given that both SegNet and U-Net utilize activation functions, most of the data were represented by zeros. Furthermore, these assignments could easily be modified using a single-inverter stage, making them reconfigurable within a real silicon circuit. Moreover, the data reported in reference [20] were for a 65 nm process, and we estimated the performance of the STT-MRAM at 45 nm using a factor of 1.3, with a power performance improvement from 65 nm to 45 nm, which was indicated in [23] to unify the process used in the simulation to 45 nm.

Table 3.2 Comparison of Memory Technologies in Terms of Energy Consumption and Frequency. [40]

Operation	Energy [pJ]		Frequency
	Read	Write	[MHz]
Int8*Int8 mult[6]	0.19375	0.19375	1400
DRAM[6]	160	160	266.7
SRAM[6]	1.25	1.25	700
STT-MRAM[22]	0.39	1.29	100
SOT-MRAM[20], [23]	1.39	0.08	700

3.4.2 Energy and area evaluation

Table 3.3 shows the three patterns used to evaluate the energy and area. In (a), SRAM was used for both the CBUF and cache. In (b), SRAM was used for the CBUF, and MRAM was used for the cache. In (c), MRAM was used for both the CBUF and cache. We adopted SegNet and U-Net, which are CNN models for image segmentation, with an input size of 512 × 1024 pixels.

To calculate the area of each module, we referred to a CNN-specific processor designed and manufactured in a previous study [9]. The area per bit values of the SRAM and MAC were calculated by dividing their areas

by the capacity and number of MAC units based on the layout of the previous CNN processor. The area of each also changed with the cache capacity, and the SRAM was replaced by MRAM. Based on these calculations, Table 3.4 presents an area comparison of the replacement strategies, where (a)–(c) are shown in Table 3.3. By replacing the cache with MRAM, the area required could be significantly reduced by up to 48%. Replacing the CBUF with 1 MB of MRAM instead of 512 KB of SRAM resulted in a small increase in the area. However, replacing the CBUF with MRAM is expected to reduce the energy consumption; therefore, the performance per unit area is expected to be better.

Table 3.3 Patterns to evaluate. [40]

	(a)	(b)	(c)
CBUF	SRAM (512 KB)	SRAM (512 KB)	MRAM (1 MB)
Cache	SRAM	MRAM	MRAM

Table 3.4 Memory area of each pattern. [40]

	Area [mm²]		
	(a)	(b)	(c)
1 MB	10.39	8.49	8.57
2 MB	14.34	10.32	10.40
4 MB	22.25	13.98	14.06
8 MB	38.07	21.29	21.37
16 MB	69.71	35.93	36.01

3.5 Experimental results

In the previous sections, we discussed the increase in memory capacity and, using a simulator, we investigated the performance when MRAM was utilized, including operations where MRAM demonstrates a significant advantage.

3.5.1 Evaluation of CBUF transfer

For the internal memory transfer of the NVDLA, the theoretical values of SegNet and U-Net were calculated by referring to the YOLOv3 values obtained from Firesim-NVDLA. There are three buffers in NVDLA: the CBUF, assembly buffer, and delivery buffer. In this study, only the CBUF was assumed to be replaced by MRAM. Fig.3.10 shows that the CBUF transfer could be decreased by increasing the on-chip memory

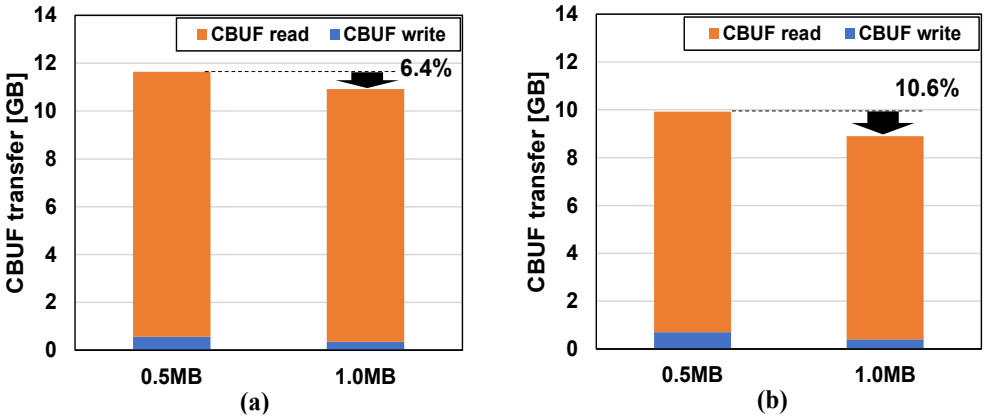


Fig. 3.10 Transfer of CBUF for (a) SegNet and (b) U-Net. [40]

capacity. However, the decrease was not significant, indicating that the transfer reduction of the off-chip DRAM was mainly responsible for the performance improvement.

Fig. 3.11 shows the decrease in the DRAM transfer rate. All write accesses to the DRAM are activation. When the MRAM configuration was used for both the cache and CBUF, the reduction was more than 11.1% for SegNet and more than 24.1% for U-Net. Because the CBUF switches the weight while maintaining the input, the activation transfer of the DRAM read operation could not be reduced significantly, even if the cache capacity was increased. The same can be said for DRAM write operation. Therefore, the reduction in the DRAM transfer, which was expected from an increase in the cache capacity, was mainly due to changes in the weight transfer. Additionally, because U-Net uses skip connections, it was necessary to store the activation values before applying the activation function. The read/write breakdown of the DRAM transfers for each task showed that U-Net benefited more from the increased CBUF than SegNet, as demonstrated in the comparison. This is because U-Net has a larger internal bandwidth for NVDLA than SegNet. SegNet had a larger write DRAM bandwidth than U-Net because upsampling was performed by the CPU.

3.5.2 On-chip MRAM performance evaluation

The performance evaluation results for SegNet and U-Net when the SRAM was replaced with the MRAM are shown in Fig. 3.12 and 3.13, respectively. Parts (a), (b), and (c) in the figures correspond to (a), (b), and (c) in Table 3.3, respectively. The top images in Fig. 3.12 and 3.13 show the evaluation results for the energy performance when the on-chip memory was changed from SRAM to MRAM. The input image size was set to 512 × 1024 pixels. A comparison of (a) and (b) shows that there was a slight energy reduction in the MRAM cache configuration. A comparison of (a) and (c), where the cache and CBUF were replaced with the STT-MRAM, shows that the energy consumption was reduced by 15.4% and 21.2%, respectively. This corresponded to improvements in the TOPS/W of 18.2% and 26.8% for SegNet and U-Net, respectively. In contrast, the replacement with SOT-MRAM reduced the energy consumption by 10.1% and 18.6%,

respectively, and yielded TOPS/W improvements of 11.2% and 22.8%, respectively. The latencies of both SegNet and U-Net could be reduced by more than 10% when using the SOT-MRAM; however, the latency deteriorated significantly when using the STT-MRAM. These results showed that replacing both the cache and CBUF with SOT-MRAM could significantly reduce the energy consumption and latency.

Here, we describe the improvement in the TOPS/W in SegNet. The baseline was 2.32 TOPS/W with a 1-MB cache and 512-KB CBUF, as shown in (a) of Table 3.3. Replacing the cache and CBUF with STT-MRAM and doubling the capacity yielded 2.74 TOPS/W, which was an 18.2% improvement from the baseline. Replacing it with the SOT-MRAM yielded 2.58 TOPS/W, which was also an improvement. As for the TOPS/W improvement in U-Net, the baseline was 1.89 TOPS/W with a 1-MB cache and 512-KB CBUF. When the cache and CBUF were replaced with STT-MRAM and the capacity was doubled, the performance improved to 2.40 TOPS/W, which was a 26.8% improvement from the baseline. When they were replaced with the SOT-MRAM, the performance was 2.32 TOPS/W, which was a 22.8% improvement. We also estimated the cost by calculating the energy × area and latency × area values for cases where the capacity of the SRAM was increased and it was replaced by MRAM. A smaller value indicated a better index. In the bottom panels of Fig. 3.12 and 3.13, (a), (b), and (c) correspond to (a), (b), and (c) of Table 3.3, respectively. The performance per unit area deteriorated significantly when the SRAM capacity was increased. In contrast, replacing SRAM with MRAM, which has excellent area efficiency, improved the performance. A comparison of the STT-MRAM and SOT-MRAM showed that the energy × area reduction rate was higher for the STT-

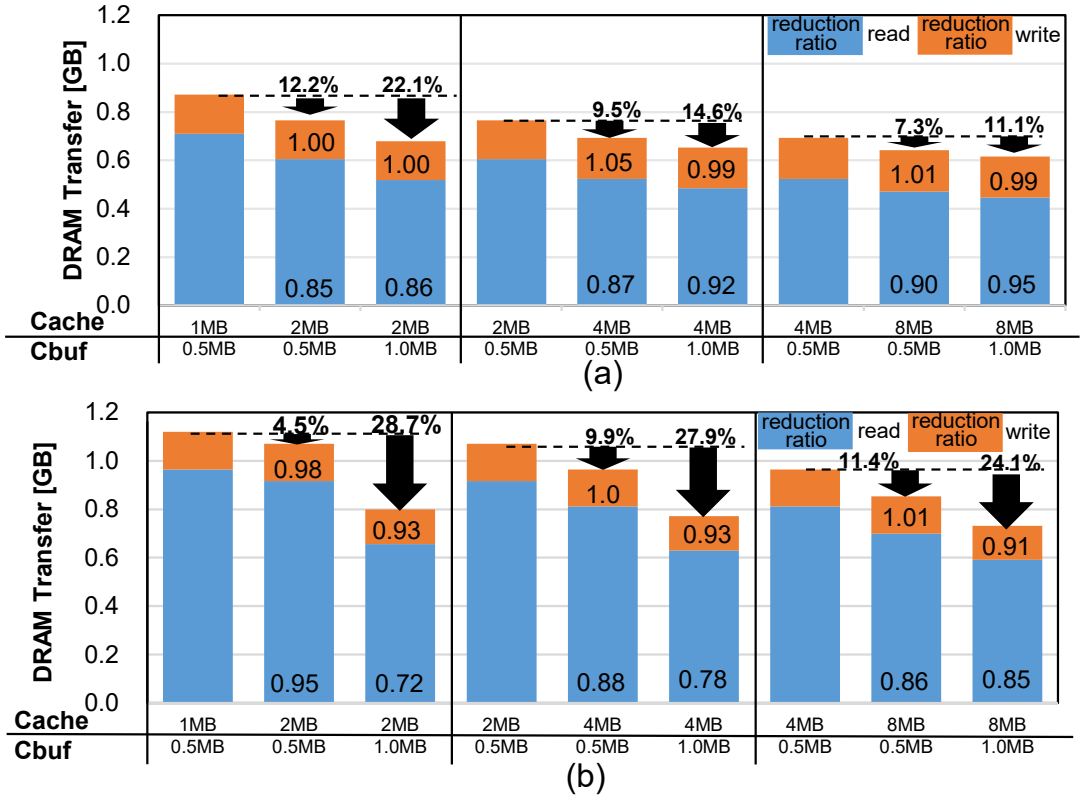


Fig. 3.11 DRAM transfers and reduction rates for (a) SegNet and (b) U-Net. [40]

MRAM. However, the latency × area value was considerably larger because of the lower frequency of the STT-MRAM. Therefore, the SOT-MRAM was more effective in inference applications such as automated

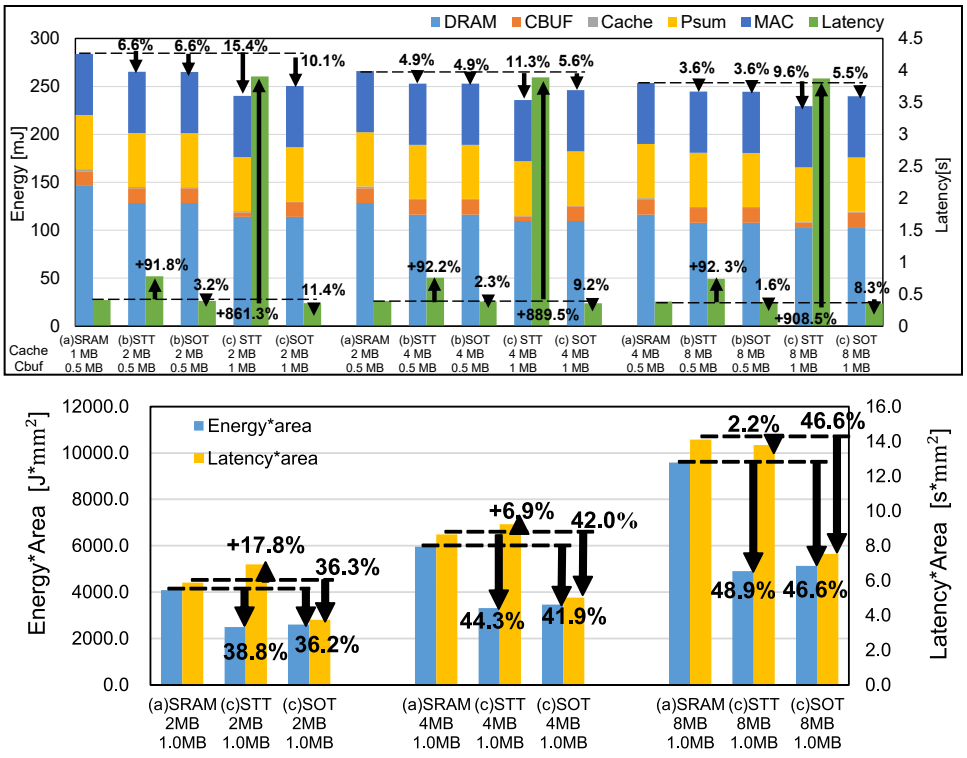


Fig. 3.12 Performance improvement in SegNet. [40]

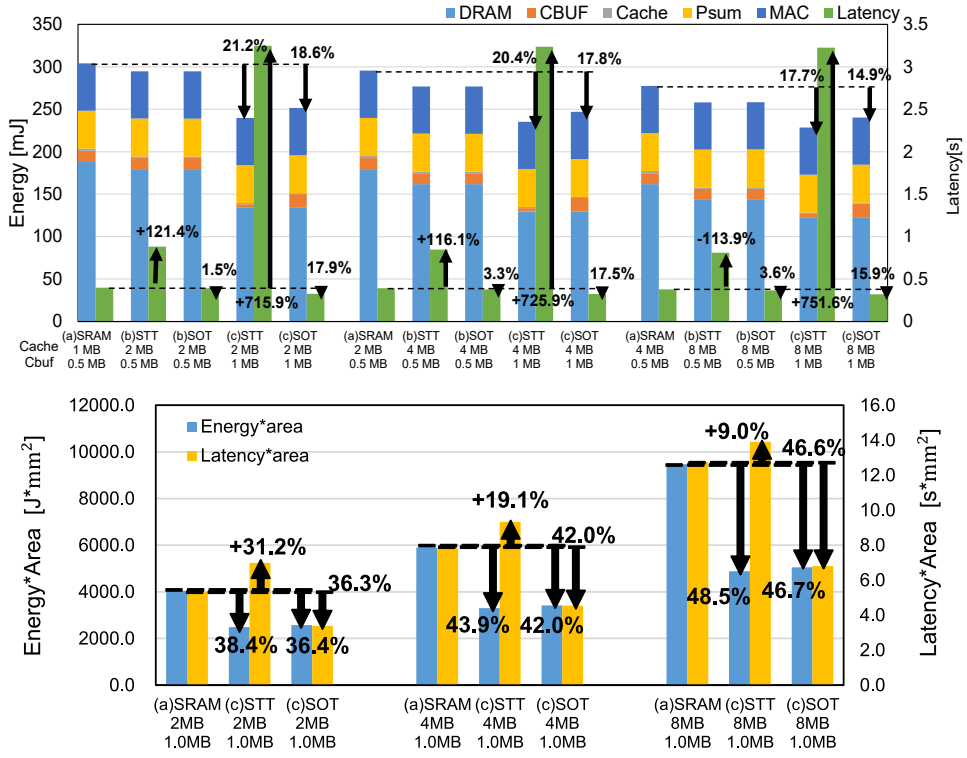


Fig. 3.13 Performance improvement in U-Net. [40]

driving, which often require real-time operations. On the other hand, when latency was not considered, the STT-MRAM achieved greater power savings than the SOT-MRAM. As shown in (b) of the figure, the latency overhead was less than 121.4% when only the cache was built using STT-MRAM. When the CBUF was constructed with STT-MRAM, the power consumption was reduced by 24.0% compared to that of SOT-MRAM, as seen in Fig.3.12(c) and 3.13(c).

This was because the read operation was dominant in the CBUF, and it used little power. Table 3.2 shows that this simulation assumed a case where the arithmetic unit was operated at 1400 Mhz. However, for applications that do not require real-time performance, where running the core at slower speeds does not cause problems, a higher performance could be achieved, and STT-MRAM could be considered an option.

Comparing the energy estimation results of SegNet and U-Net, U-Net achieves higher power savings. This was because DRAM accesses account for a larger percentage of power consumption in U-Net. Also, (c) the ratio of increase in latency is greater for U-Net than for SegNet. This was because U-Net has more internal memory bandwidth and is more dependent on CBUF latency.

3.6 Conclusion

We used NVDLA as a motif and simulated it as a case study to improve the performance of deep-learning processors with MRAM. MRAM has attracted attention as an on-chip memory technology that can replace SRAM. By replacing SRAM with MRAM, it is possible to double the capacity in the same area, which provides a higher capacity without increasing the chip cost. The performance of SOT-MRAM is comparable to that of SRAM. In fact, performance evaluations using a commercially available processor showed that using SOT-MRAM for on-chip memory could reduce the power consumption by 10.1% and increase the speed by 11.4% in SegNet. The power consumption could be reduced by up to 18.6% and the speed could be increased by 17.9% in U-Net. The performance of the STT-MRAM was also estimated, as expected, it had higher power consumption and latency than the SOT-MRAM. Therefore, STT-MRAM is not suitable for implementation in edge systems where real-time operation is required. Future work is needed to verify the significant reduction in the power consumption of the CBUF with respect to the processor operating frequency and choice of STT-MRAM and SOT-MRAM. In terms of the performance per area, replacement with SOT-MRAM is expected to improve the performance by more than 36.2% compared to SRAM for the same capacity. In conclusion, using SOT-MRAM as an on-chip memory for deep-learning processors is a good choice.

Chapter 4
1W8R 20T SRAM Codebook for 20% Energy Reduction in Mixed-Precision Deep-Learning Inference Processor System

4.1 Introduction

As mentioned in the previous section, sum-of-products operations constitute the majority of computations in deep learning. As a result, many deep-learning processors have adopted the single instruction multi data (SIMD) architecture. The SIMD architecture enables parallel processing of large datasets but requires a substantial memory bandwidth. Additionally, a high memory bandwidth reduces the cache hit ratio and increases the bandwidth usage of the DRAM.

Low-power and fast deep-learning processors are key technologies for social implementation. Deep learning algorithms require a large number of multiply and accumulate (MACs) operations, whereas deep learning processors handle numerous parameters and activations [24]. From an energy-consumption perspective, external dynamic random-access memory (DRAM) consumes 100 times more power than internal static random-access memory (SRAM) at the system level [6]. To address this issue, increasing the on-chip memory to reuse data is an effective approach [12]. However, current deep-learning processors already use large-capacity SRAM, which occupies more than 50% of the silicon area [9]; thus, there is no room to expand the SRAM area or bandwidth without incurring additional costs.

One method for reducing memory bandwidth without increasing the embedded memory area is quantization. In many deep-learning frameworks, such as PyTorch and TensorFlow, a 32-bit floating-point (FP32) representation is used by default. FP32 provides the precision required by most layers during the deep learning training phase. Deep learning requires high-precision numerical representations, such as FP32, for training. However, many layers in deep learning models, except for layers requiring high-precision operations, such as BatchNorm, can achieve comparable inference accuracy, even at eight-bit or lower precision. Consequently, data quantization reduces the memory transfer demands and enhances the cache hit ratio. Quantization also enhances parallelism because the area of a multiplier circuit can be reduced by a factor of four when the arithmetic precision is halved.

In prior studies on deep-learning quantization, in addition to simple linear quantization, various other techniques have been proposed. These include training-aware quantization, outlier clipping, and optimizations

such as adjusting the quantization bins and scaling factors. Mixed-precision quantization can adaptively optimize the layer-by-layer accuracy [6]. In Pytorch, qint8 type, which is a linear quantization, has been implemented [25]. Efficient compression and decompression operations are necessary to compact the weights and activations of deep learning processors [6, 26]. However, a cast from/to qint8 consumes graphics processing unit (GPU) resources and incurs overhead. A codebook, a compression algorithm, can represent arbitrary nonlinear functions and suppress accuracy degradation more effectively than linear quantization [8]. Compression using the codebook is performed by first clustering neural network weights, then determining a representative value for each cluster, and finally storing only the indices of the clusters. At inference, compression generally considers the nearest neighbor representative values. Decompression using a codebook converts the stored indices into representative values. Both compression and decompression operations can be performed using lookup tables. Unlike linear quantization, the use of a lookup table is simple and fast because it does not require scaling, clipping, or division.

In this study, we focused on Deep Compression, a quantization algorithm well-suited for hardware implementation, and explored hardware acceleration for data compression and decompression. We proposed a 1W8R super-multiport SRAM to accelerate the compression and decompression of weights and activation using codebooks. This SRAM is designed to reduce the DRAM access bandwidth and lower the inference power consumption in deep learning processors. The contributions of this study are as follows.

- We designed a novel 1WR8 super-multiport SRAM cell and demonstrated its stable operation at 0.95 V by Monte Carlo simulation.
- We fabricated memory macros for a super-multiport SRAM using a 40 nm process and measured their access speed and power consumption.
- Based on the performance simulator and silicon evaluation described in Chapter 3, we combined the simulation and evaluation results to estimate the performance improvement when integrated into NVIDIA's deep-learning inference processor, NVDLA [16].

4.2 Previous works

High-density embeddable multiport memory is crucial for satisfying the high memory-bandwidth demands of modern processors. In particular, in deep learning processors [6, 26, 27], where many multiply-accumulate operations are performed, multiport memory shared at the circuit level is essential. For example, dual-port memory is used in GPUs to register files in vertex-shader processors [28]. Furthermore, in high-performance computing (HPC), multiport memory is crucial in interconnects, enabling tight coupling between processors and register files that support out-of-order executions [29]. Thus, compact and processor-embedded multiport memories have become indispensable for modern computational circuits.

The structure of a typical multiport SRAM, shown in Fig. 4.1, is realized by connecting multiple read and write ports to a single cross-coupled latch [30, 31]. Two types of read ports exist in a multiport SRAM: drain-

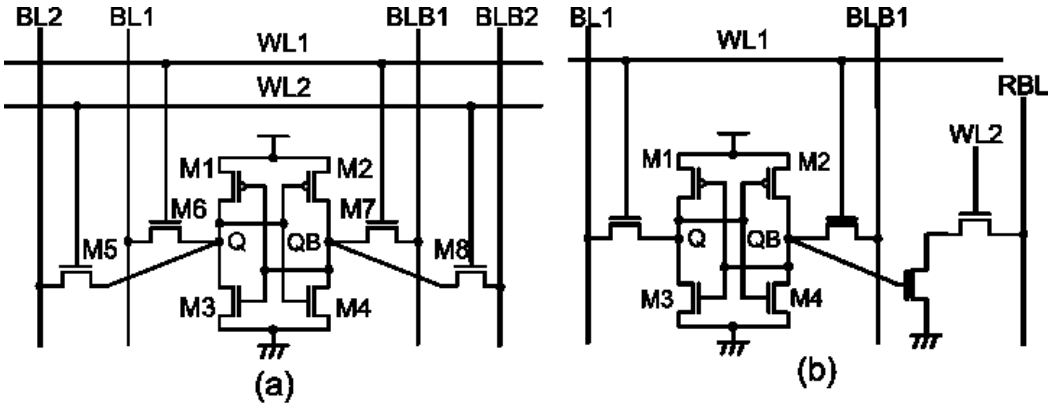


Fig. 4.1 Structure of Multiport SRAM (a). 8T dual-port (b)1W1R

connected read ports, as shown in (a), and gate-connected read ports, as shown in (b). Drain-connected read ports can consist of 1T; however, this deteriorates the static noise margin (read margin) in the case of multiple selections. Its multiple-column bitlines worsen the write margin because of the half-select disturbance. Gate-connected read ports require two transistors per port, which increases their area. In a multiport SRAM design, these issues should be considered. Prior research [31] showed that leveraging inverters and tri-state buffers could achieve higher speed while reducing the area overhead. As described above, when designing a multiport SRAM, it is crucial to address these trade-offs and carefully balance access performance, area efficiency, and noise margins.

Previous research [32] presented a 2W6R super-multiport SRAM with a simple structure. The circuit diagram is shown in Fig. 4.2. In this study, buffers were placed between Q and QB and the read ports, which improved the read margin. However, specialized structures result in wasted space in the layout. In another research [28], the memory cell area of a 1W5R SRAM was reduced by sharing pull-down transistors of the read ports.

An example of a multiport SRAM with a specialized structure is presented in [33]. In this study, the stability of the memory is enhanced by increasing the number of stages in the cross-coupled latch of the SRAM.

Aside from increasing the number of read ports, another implementation method for multiport SRAM is double-pumping SRAM. Double-pumping SRAM behaves as a pseudo-multiport SRAM that achieves multiple internal memory accesses in response to an external memory access [34]. This is achieved by optimizing the clock signals used for controlling the precharge and read amplifiers and leveraging the differences in the read and write speeds of the SRAM. However, double-pumping SRAM generates internal clocks using delay, combinational, and sequential circuits in response to external clocks, leading to higher design costs. Moreover, because multiple internal access cycles occur for each clock input, the power consumption tends to be higher than that of a typical multiport SRAM. Previous research [28, 33] incorporated a multiport SRAM using a double-pumping scheme, which further enhanced the multiport capabilities of double-pumping SRAM, achieving a 3R3W multiport SRAM.

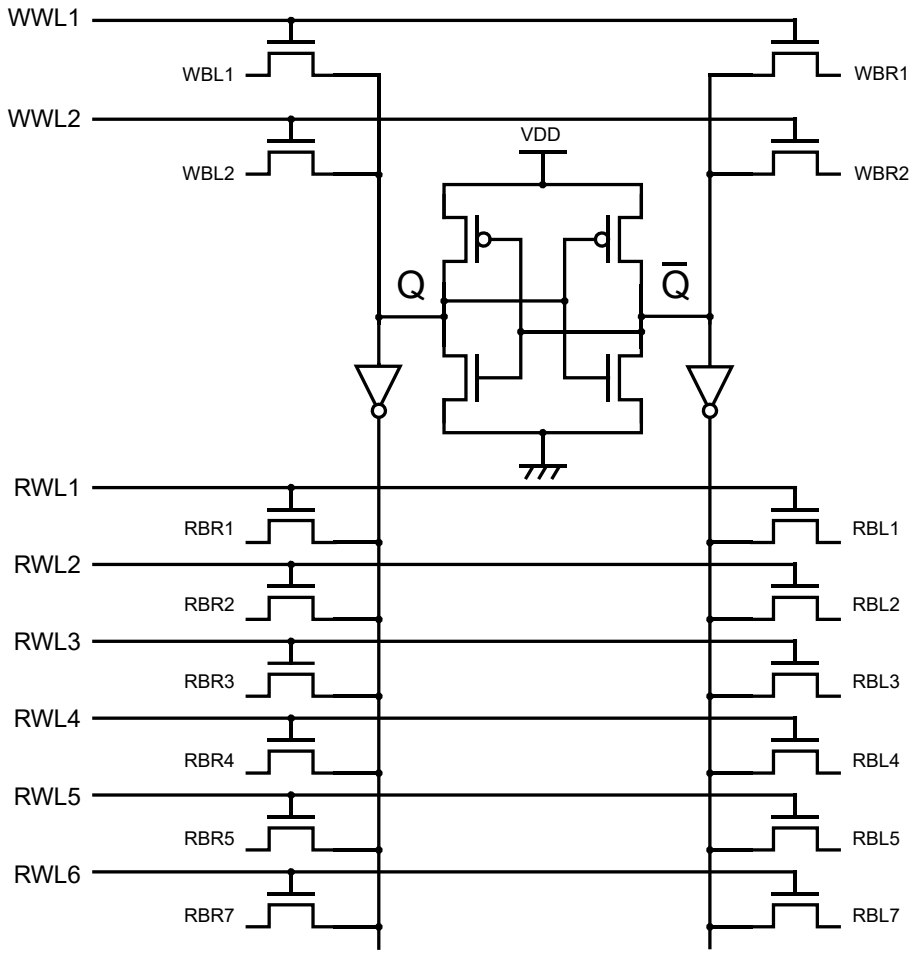


Fig.4.2 Structure of 2W6R Multiport SRAM.

Fig. 4.3 shows a 2R2W GC-eDRAM memory cell.In recent years, embedded DRAM (e-DRAM) has emerged as an alternative to SRAM. e-DRAM is a type of pseudo-SRAM that can be manufactured using a

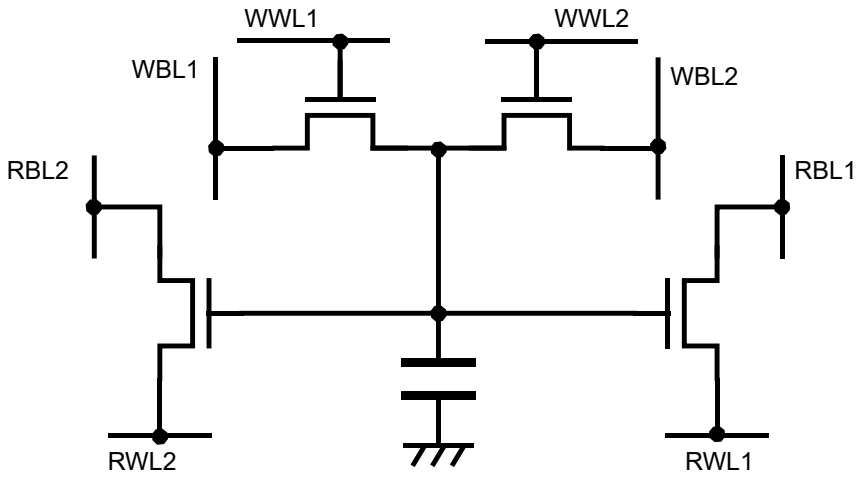


Fig. 4.3 2R2W Gain Cell.DRAM.

standard CMOS process [35]. Because it can be fabricated using the same process as logic circuits, it can be integrated with logic and designed with approximately 2T, a smaller transistor count than the 6T-SRAM. Therefore, it has garnered attention as a cache for computing processors and deep-learning accelerators. Gain-cell eDRAM (GC-eDRAM), which is a type of e-DRAM, requires periodic refreshing every few microseconds, similar to DRAM, because it stores charge on the gate of a transistor. This can lead to increased power consumption [36].

Numerous studies have focused on reducing DRAM bandwidth through hardware architecture. One significant prior work, EIE [6], implements hardware-based codebook compression, which builds upon the Deep Compression [8] framework and incorporates not only codebook compression but also zero-skipping and other optimization techniques. A related study [37] introduces an FPGA-based deep learning processor leveraging two primary techniques: JPEG compression for input data bandwidth reduction and a fully pipelined Convolutional Neural Network (CNN) for efficient computation. These techniques significantly reduce memory access requirements and facilitate performing all computations within on-chip memory.

4.3 Circuit and architecture

4.3.1 Super-multiport SRAM

We propose a 1W8R multiport SRAM comprising 20 transistors (20T), as illustrated in Fig. 4.4. The bit cell comprises one differential write port, two drain-connected single-ended PMOS read ports, and six gate-connected single-ended read ports with two NMOSes. The proposed SRAM features eight read ports. Therefore, the proposed SRAM memory macro enables the parallel compression and decompression of eight data streams. [38]

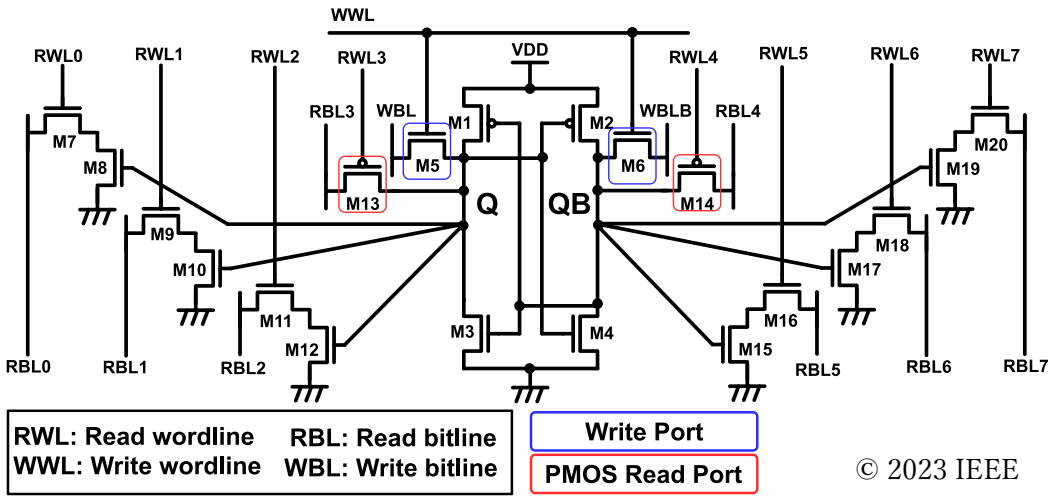


Fig. 4.4 Structure of 1W8R Super Multiport SRAM. [38]

The gate-connected NMOS read port requires two transistors that require more area but do not adversely affect the read margins. The drain-connected PMOS read port can be configured with a single transistor, adversely affecting read margins. As described later, the proposed memory is structured by mixing two types of read ports to achieve a good balance between the area and readout margin.

Fig. 4.5 illustrates a diagram of a codebook SRAM with one write port and eight read ports. To utilize a codebook within memory macros, a representative value is first written through the write port. In this process, the write address (W Addr) serves as the index, and the write data (W Data) represents the corresponding value. Once the data is stored in the memory cell, it can be accessed randomly through any of the eight read ports. Furthermore, employing a multi-port SRAM reduces the memory cell area compared to using eight separate codebooks.

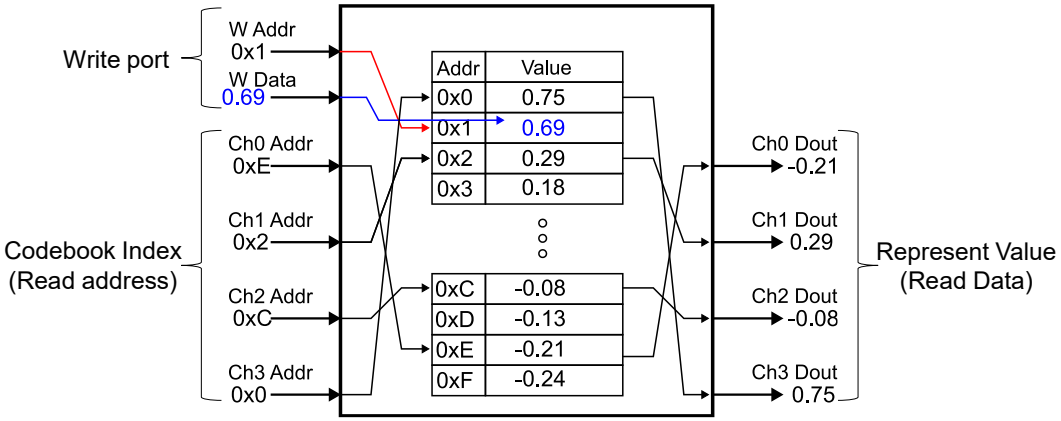


Fig. 4.5 The Codebook SRAM macro consists of the proposed Super Multiport SRAM

4.3.2 Codebook compression

Fig. 4.6 shows an example of weight compression and decompression using a codebook. In codebook compression, parameters such as deep learning weights, biases, and activation are clustered into a few groups, and representative values are determined for each group. Subsequently, the indices are assigned to the representative values, and only the indices are stored in memory. Simultaneously, a table of maps for the indices and representative values is created. In the decompression process, the indices are replaced with representative values using the created table of indices and representative values.

Representative values can be placed in a nonlinear arrangement; consequently, they have a higher expressivity than linear or log quantization. In addition, replacement using a lookup table is a linear operation that can be implemented as a neural network, allowing fine-tuning after clustering and compression with low-precision degradation.

Fig. 4.7 illustrates an overview of the fine-tuning algorithm for the codebook. In step (1), the original weight parameters are replaced with representative values, compressed, and only the index of the representative value is stored. In step (2), the indices are mapped to representative values using a lookup table. In this state,

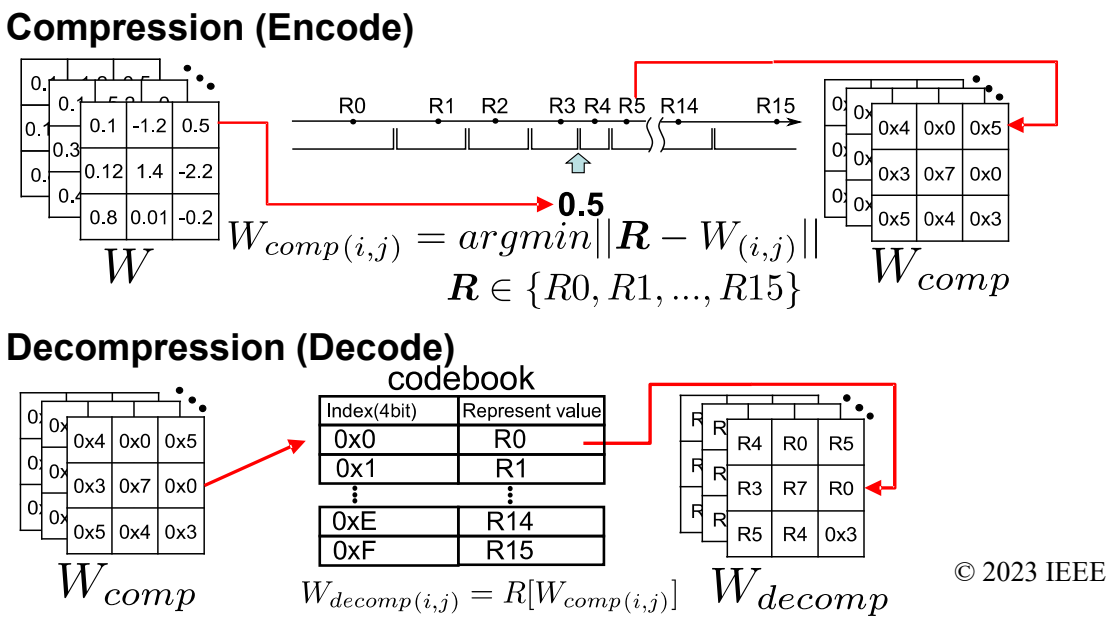


Fig. 4.6 Weight codebook for 16-group compression and decompression. [38]

inference is then performed, and the loss is calculated using the error backpropagation method. In step (3), the gradient of the weights is calculated based on the loss. The gradients are then aggregated per weight and added

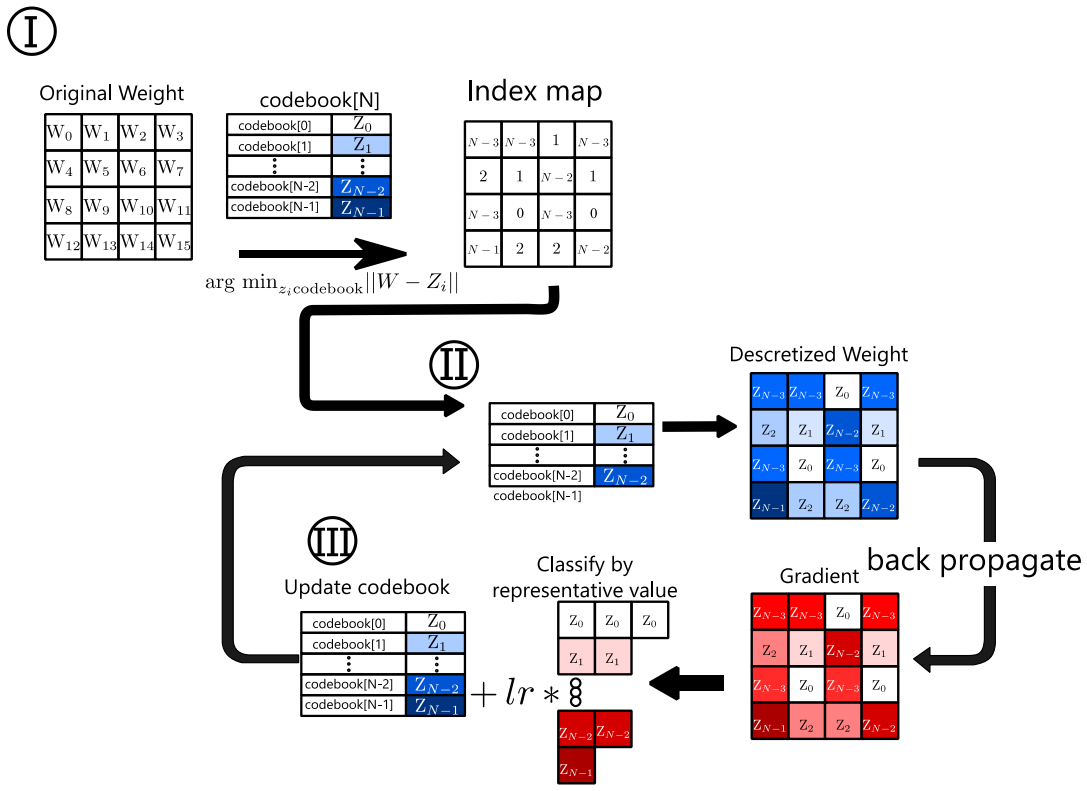


Fig. 4.7 Fine-Tuning Algorithm for Codebook Optimization

to the codebook. Since all these operations consist of linear computations, the error backpropagation method can be effectively applied for training.

Preliminary experiments were conducted using the U-Net [18] as the target network. We used the parameters pretrained by Cityscapes [21], and the dataset focused on the semantic understanding of urban street scenes, in which a semantic segmentation dataset was used. We divided U-Net into ten units, allowing the codebook to quantize them from the input side sequentially. U-Net has an input unit comprising two convolutional layers. Downsampling and upsampling units each consist of two convolutional layers stacked with four layers. The output unit has one convolution layer.

Fig. 4.8 illustrates a histogram of the U-Net layer weights. The light blue histogram represents the distribution of the original values. The blue histogram depicts a linear quantization of the maximum and minimum values, divided into 16 quantization bins (equivalent to 4 bits). The central value of each quantization bin is marked by a blue circle at the bottom of the histogram. The red histogram represents the codebook-compressed parameters. The central value of each quantization bin is marked by a red cross at the bottom of the histogram. The representative values in the codebook vary during training. As shown in the histograms, the representative values of the codebooks are clustered near the center of the FP32 histograms, where the density is high, while the density decreases outside the normal distribution. Thus, codebook compression results in nonlinear quantization bins, enabling learning to cluster weights in values optimized for inference.

Fig.4.9 shows the inference accuracy of U-Net in INT16 after fine-tuned codebook quantization, where the y-axis represents the inference accuracy and the x-axis represents the number of integer precision bits for

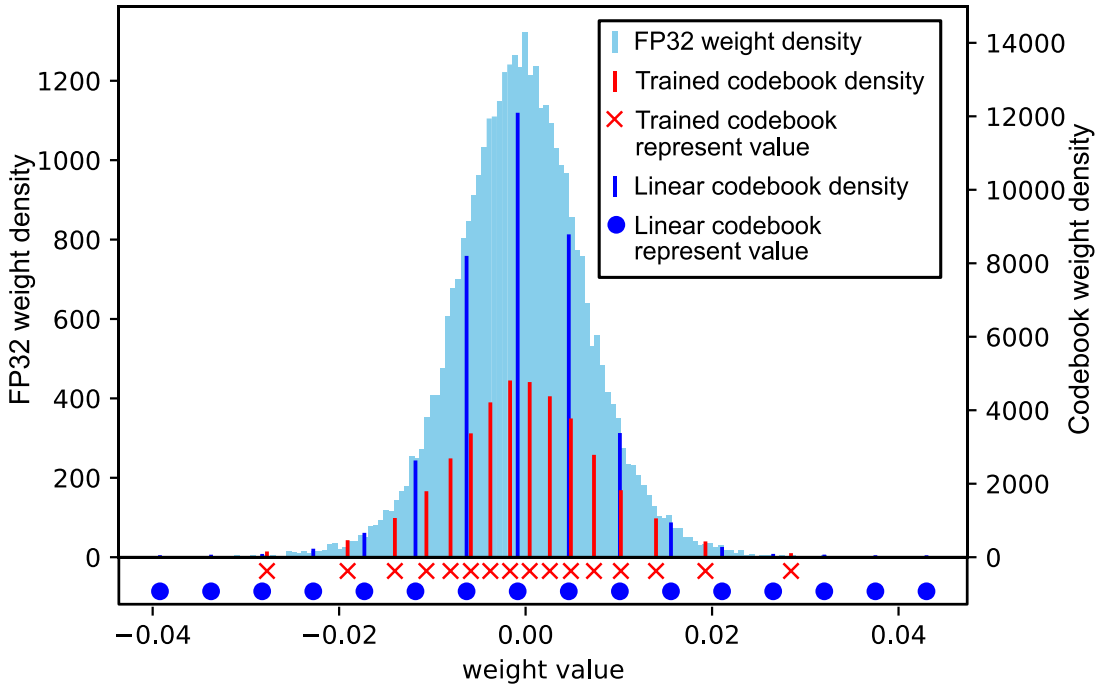


Fig. 4.8 Weight histograms and representative values. [38]

representative values in the codebook. The activations were compressed using an 8-bit codebook. The baseline used was INT16 with precision for both weight and activation. The accuracy is maintained at a level equivalent to or greater than INT16 in the range where the codebook length is between 8 bits and 2 bits. However, below this range, the accuracy deteriorates. The phenomenon where post-quantization accuracy exceeds the original accuracy occasionally occurs in the context of quantization.

In the weight-compressed 8-to-2-bit codebook, U-Net maintained the same accuracy as full INT16; however, the accuracy decreased with the number of integer precision bits below this threshold. Because the experimental results show that its accuracy outperforms that of INT16 with a weight codebook length of 2 bits, we assumed that a codebook with a length of 2 bits is sufficient for weights and 8-bit codebook activations.

Super-multiport SRAM is eligible for a codebook with compression (encoding) and decompression (decoding) in convolutional operations. Several deep-learning processors have used codebooks for activation and parameter compression to reduce DRAM bandwidth and power consumption [6, 39]. However, four-port SRAM and SRAM with more ports have rarely been considered [32]. No work has been reported on the implementation of a codebook with a super-multiport SRAM.

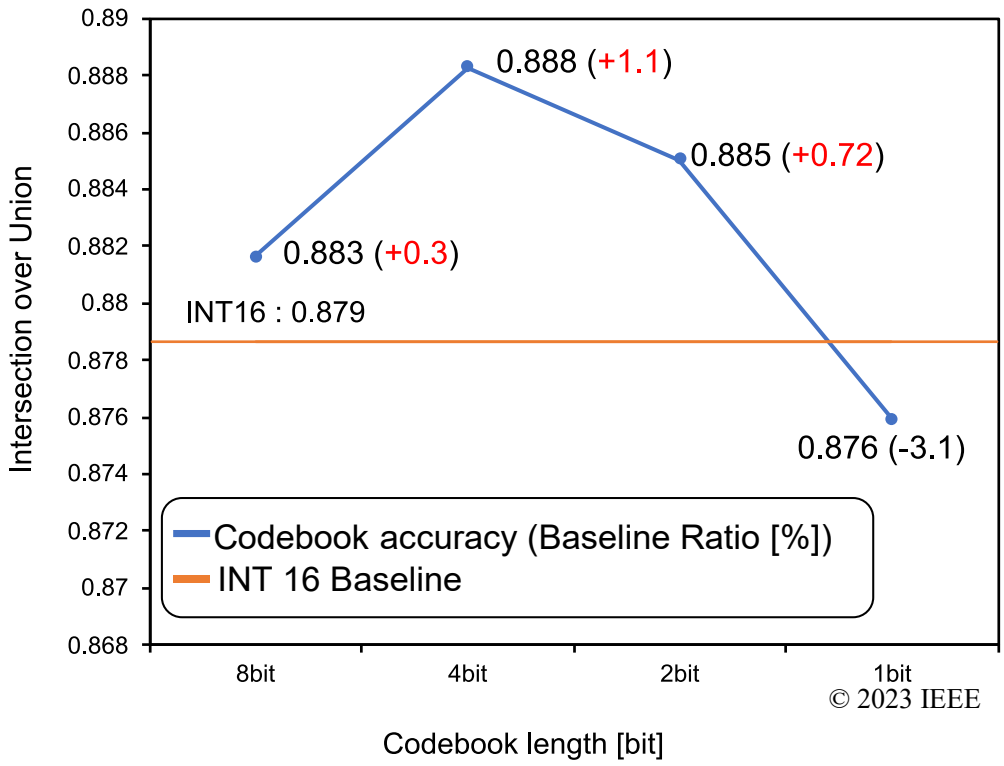
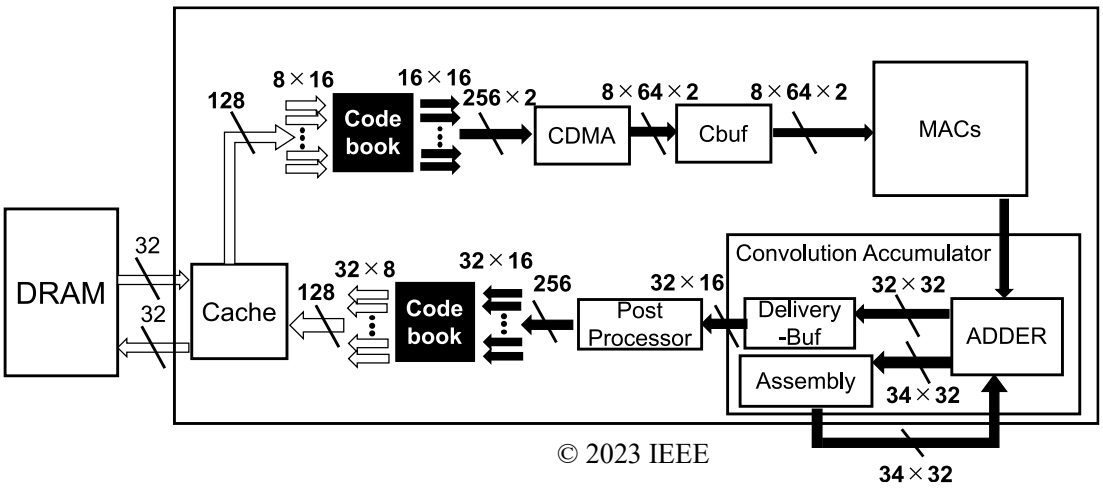


Fig. 4.9 U-Net inference accuracy for the INT16 codebook. [38]

4.3.3 Target processor

We envision the proposed super-multiport SRAMs as an encoder and decoder that can be installed on the memory bus of the motif deep-learning processor, NVDLA [11]. NVDLA is a commercial processor with a cache and a generic adder-tree-type operator array. The block diagram is shown in Fig. 4.10. The decoder codebook was installed between the cache and the MACs. The encoder with codebook compression was placed between the MACs and cache. The decoder and encoder codebooks were executed as 8-from/to-

16-bit lookup tables. The decoder content can be utilized as a table for weights and inputs shared by the eight convolutional MACs. Every MAC accesses a different address; therefore, super-multiport SRAM is required. Similarly, the encoder handles the accumulated activations simultaneously from the eight parallel data paths. Therefore, higher power and area efficiencies can be expected in the decoder and encoder than in those with an array of regular 6T SRAMs.



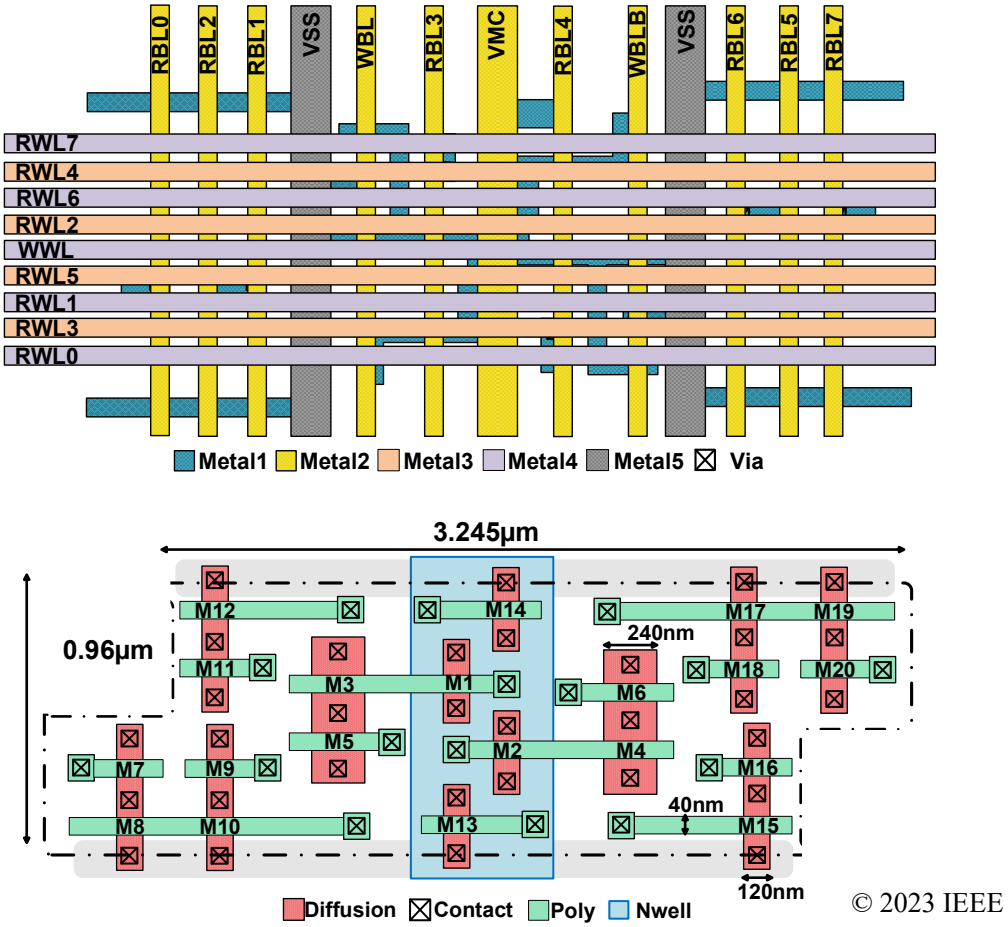
© 2023 IEEE

Fig. 4.10 Decoder and encoder codebooks in the NVDLA block diagram. oks in NVIDIA NVDLA block diagram. [38]

4.3.4 Memory cell design

Fig. 4.11 shows the front-end-of-line (FEOL) and back-end-of-line (BEOL) layouts of the proposed SRAM bit cell according to the 40 nm logic rule. A pair of differential write bitlines (WBLs) and eight single-ended read bitlines (RBLs), for a total of ten bitlines, are drawn in the vertical direction. One write wordline (WWL) and eight read wordlines (RWLs) are drawn horizontally. The nine wordlines impose a certain bit cell height. Since the bitlines are arranged vertically and the wordlines horizontally, the memory cells can be connected and arranged in a grid-like structure. To effectively use the vertically long N-well, two drain-connected PMOS read ports were placed at the top and bottom of the N-well (M13 and M14). This design allows the N-wells at the top and bottom to be connected, reducing the N-well margin.

Table 4.1 shows the area of the proposed super-multiport SRAM. The memory cell of our super-multiport SRAM is nearly x1/3 smaller than that of a 6T SRAM with the same functionality. This bit cell compaction reduced the area by 28% compared to eight NMOS read ports.



© 2023 IEEE

Fig. 4.11 Top: FEOL and bottom: BEOL layouts of 1W8R 20T SRAM. [38]

Table 4.1 Comparison of different SRAM cell types based on height, width, and area.

Cell type	Height(μm)	Width(μm)	Area(μm^2)
6T SRAM	2.59	0.44	1.14
6T SRAM(8cell)			9.12
20T SRAM all gate connected	3.31	1.30	4.30
20T SRAM 1W8R NMOS (This work)	3.245	0.96	3.12

4.4 Evaluation experiment

Three experiments were conducted to evaluate the multiport SRAM design. Fig. 4.12 shows the relationships between the three evaluations. The first is a simulation evaluation. We performed Monte Carlo simulations of the memory cells of the designed multiport SRAM. In these simulations, we evaluated the supply voltage and analyzed the simulation results. The second is silicon evaluation. The designed SRAM macros were fabricated at 40 nm process, and their read speed and power consumption were measured using a semiconductor tester. The third is a performance simulation. We integrated the simulator developed in Chapter 3, area information obtained from the design described in this chapter, and access parameters obtained from the silicon evaluation. We then evaluated the power and area performances of the NVDLA when the proposed ultra-multiport SRAM was embedded.

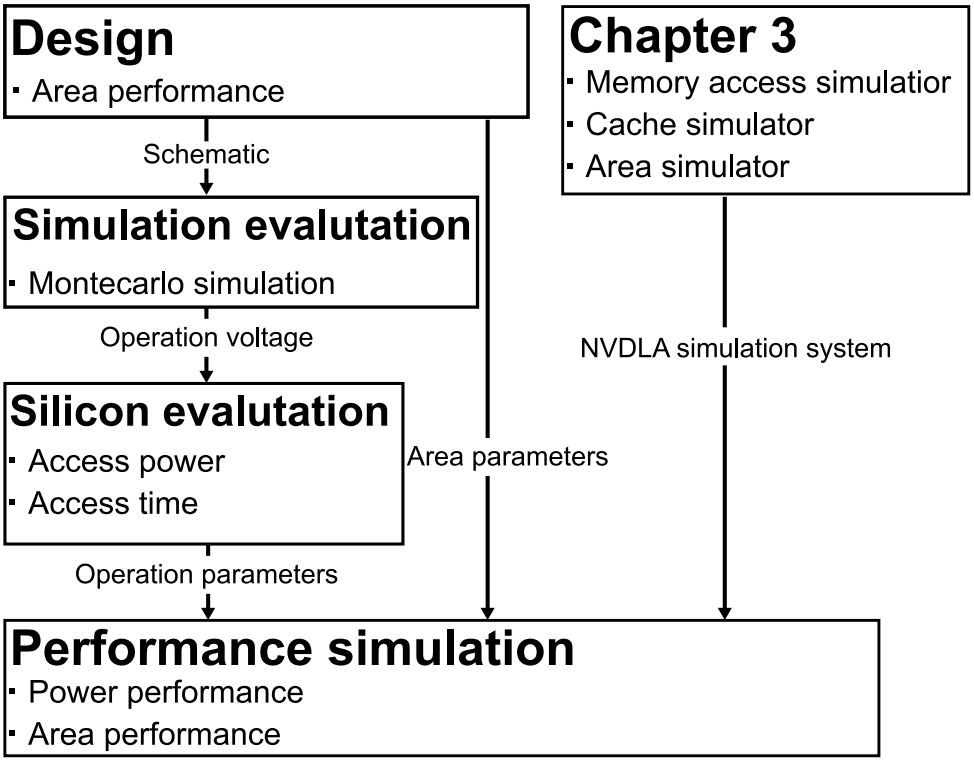


Fig. 4.12 Analysis and evaluation process diagram.

4.4.1 Simulation evaluation

Fig.4.13 shows the results of the 16-k Monte Carlo simulations of the proposed SRAM. The simulation was performed using the foundry Monte Carlo simulation model. The typical-case and worst-case eye patterns in 16-k times Monte Carlo simulations for read margins at a nominal voltage of 1.1 V involve the following: (a) NMOS half-select write, (b) both-side PMOS read, and (c) one-side PMOS read. The static noise margin becomes a problem when the port has only one transistor; however, the bit-cell area can be reduced. The

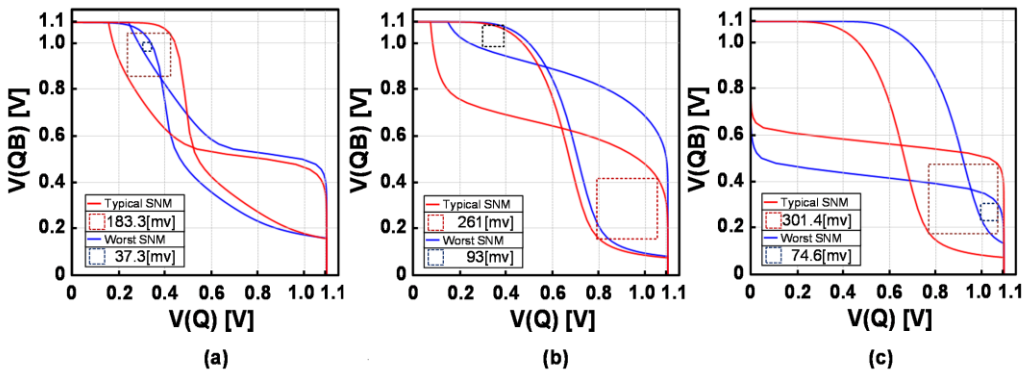
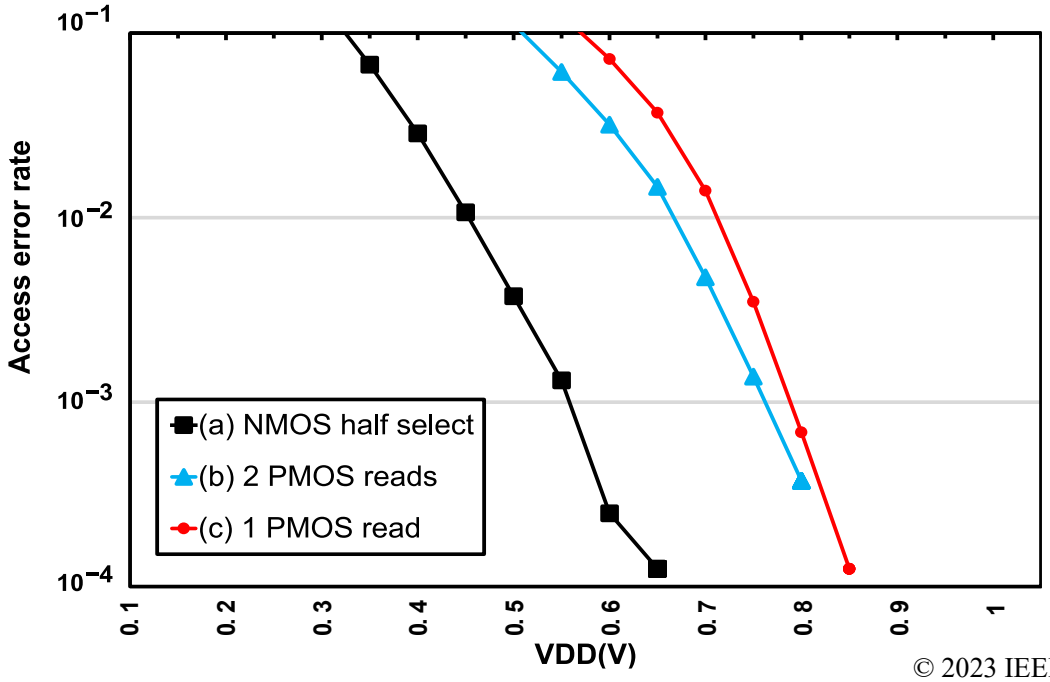


Fig. 4.13 SNM simulations: (a) NMOS half-select write, (b) both-side PMOS, and (c) one-side PMOS. [38]

proposed memory cell combines 1T PMOS and 2T NMOS read ports to achieve both small-area and static noise margins. The NMOS ports are margin-free because of their gate-connected read ports. Notably, the write margin is sufficient because of the dedicated write port. Only the lookup table conversion is read for the decoders and encoders of the codebooks in a deep-learning processor. The write operation is performed only once at the beginning of every convolution layer, and no simultaneous write-and-read operations occur in inference applications. The one-sided PMOS read secures a 75 mV margin; however, it has a smaller margin than the both-sided read because the imbalance exerts an adverse effect.

Fig. 4.14 shows the read-error rates when the supply voltage was varied. The lowest error rate was obtained at the gate-connected NMOS, and the highest was obtained at the one-side PMOS read, restricting the SRAM



© 2023 IEEE

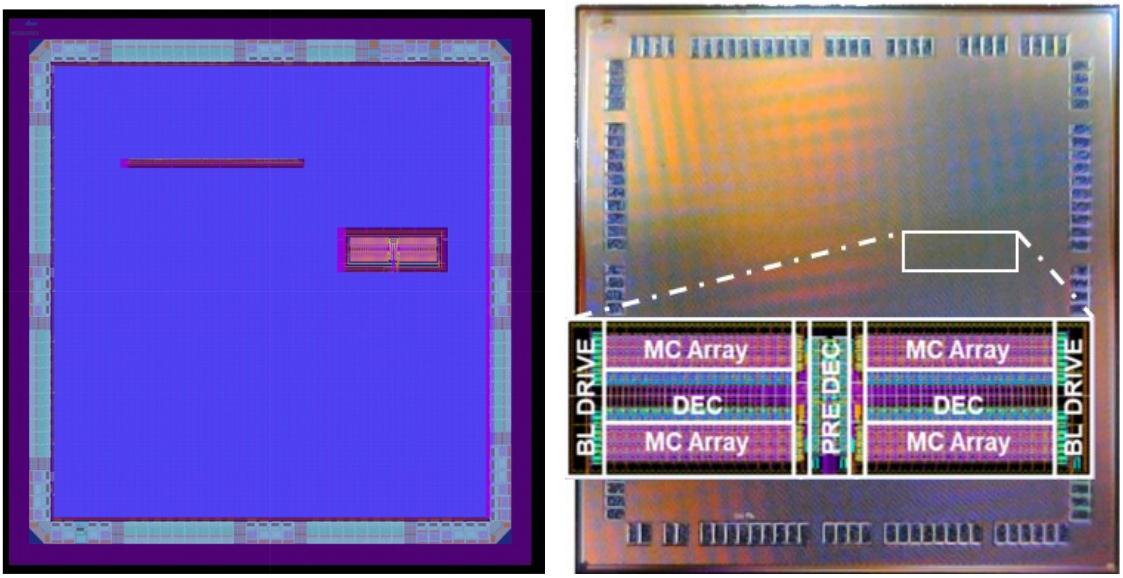
Fig. 4.14 Error rates versus supply voltage VDD [38]

operating margin. No errors occurred in all cases above 0.95 V in 16-k Monte Carlo simulations. The device operates safely at a typical process corner with a nominal voltage of 1.1 V.

4.4.2 Silicon evaluation

Fig.4.15 shows the chip micrograph and an overview of the SRAM macro fabricated using UMC 40-nm LP technology. The macroscale size was 440 × 110 μm2. Each port has 256 addresses and a data bit width of 16 bits (4 kb in total) because the decoder converts 8 bits to 16 bits. The encoder for 16-to-8-bit conversion can comprise a combination of decoders.

Fig. 4.16 shows the memory macro layout and its block diagram. In the center are the pre-decoder, precharge circuit, and sense amplifier. The WL decoders were located on the left and right sides. The memory cells were arranged in 128 words, each on the left and right sides. A vertical row of lower sides: 16 bits per word.



Technology	40 nm CMOSS Low power process
Access Time	3 ns @ 1.1V
Access Energy	2.7 [pj/Byte]
SRAM Macro size	400 x 110 μm^2
SRAM organization	256 address x 16bits/word
Supply voltage	1.1V(Logic & SRAM macro)

Fig. 4.15 Silicon and overview of the fabricated chip. [38]

© 2023 IEEE

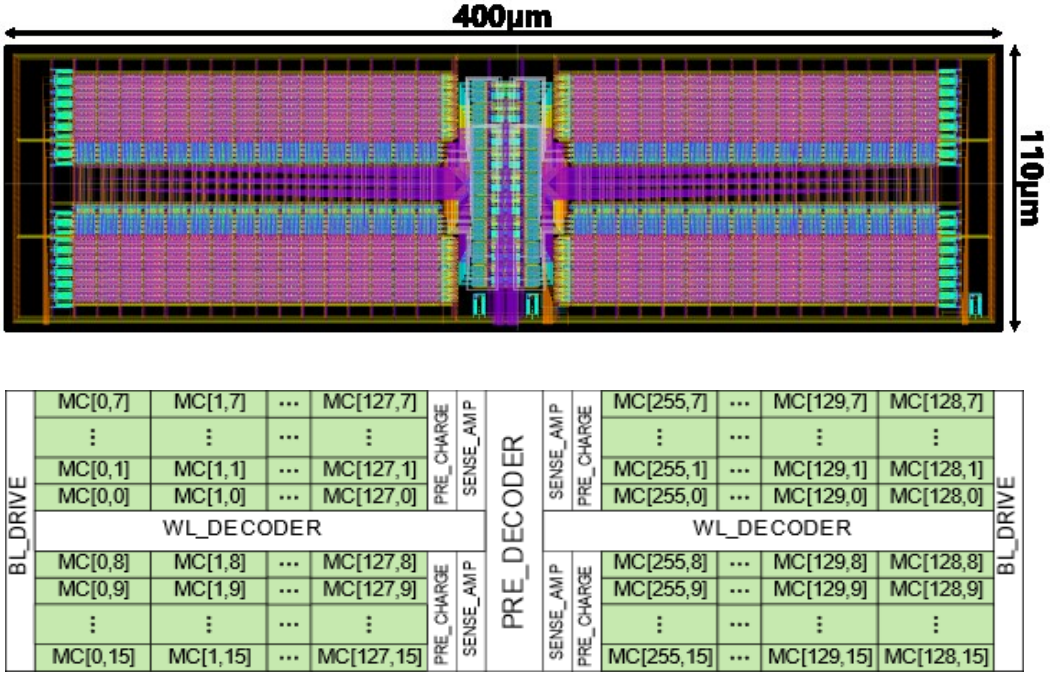
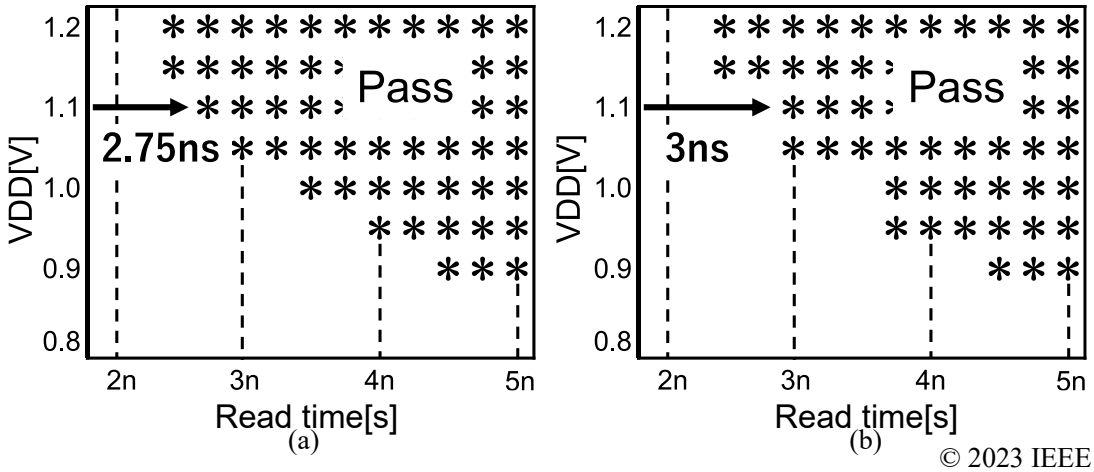


Fig. 4.16 Top: Memory macro layout. Bottom: block diagram of the SRAM macro. [38]

Fig. 4.17 (a) and (b) show the Shmoo plots for the PMOS and NMOS reads, respectively. The access time at the nominal voltage is 2.75 ns for the PMOS read port and 3 ns for the NMOS read port. The NMOS read is slower because of the stacked read ports. The read energy consumption at the maximum clock frequency and nominal voltage was 2.7 pJ/byte.



© 2023 IEEE

Fig. 4.17 Shmoo plots for;left PMOS read and right NMOS read. [38]

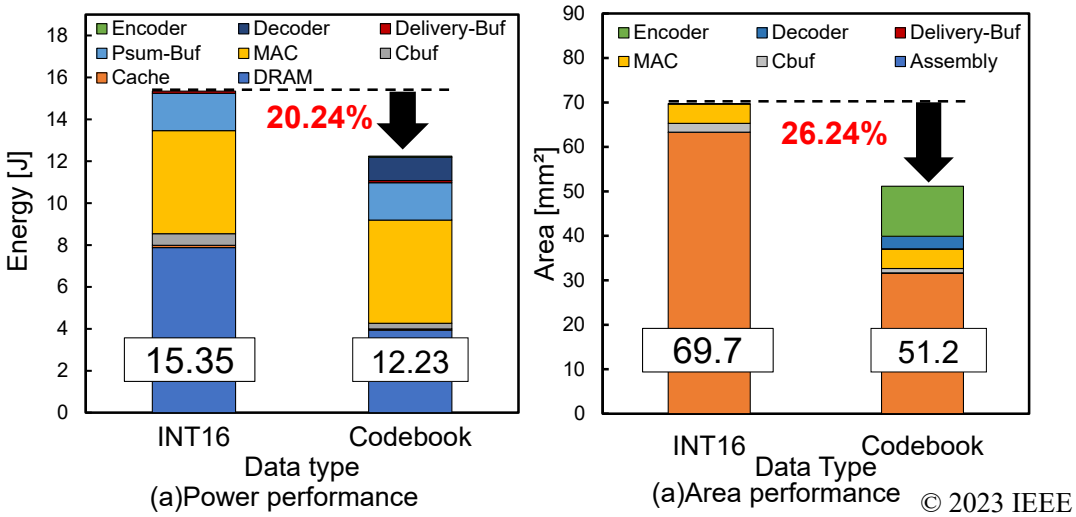
4.4.3 Performance simulation

To evaluate the memory bandwidth reduction performance of the proposed memory, we used U-Net as a target task, which requires a large memory bandwidth and is thus effective for energy reduction using the proposed super-multiport SRAM. The U-Net was trained on CityScapes, a dataset with 11 output classes.

These simulations were conducted based on three simulators: the area simulator, the memory access simulator, and the cache simulator introduced in Chapter 3. The calculations also incorporated area parameters and access parameters obtained in this chapter.

Fig.4.18 shows the performance improvement achieved by implementing super-multiport memory. Energy was estimated using memory access and computation. The energy reduction achieved by the proposed codebooks was 20% when a 4094×2048 (4 K) U-Net was executed. This reduction was primarily achieved because the DRAM transfer could be halved.

The chip area can be calculated by configuring Cbuf, delivery buf, and the number of encoder and decoder bits in the proposed multiport SRAM. The MAC includes arithmetic units. The circuit area estimation used the circuit area obtained from the actual device and from our previous work [40]. The estimated chip area is costly; however, the codebooks can halve the 16 MB cache and the 512 kB convolutional buffer, yielding a 26% reduction, although the codebooks are appended as internal SRAMs.



© 2023 IEEE

Fig. 4.18 Performance improvements in energy and area. [38]

4.5 Concluding remarks

We designed a codebook memory for a deep-learning processor, ranging from memory cells to memory macros, and fabricated it using a 40 nm process. In the memory cell design, two of the eight read ports were drain-connected PMOS to ensure read margins while reducing the number of transistors and saving N-well

margins, resulting in a 28% area reduction. We conducted 16-k Monte Carlo simulations on the designed memory cell and confirmed that no errors occurred at 0.95 V. In addition, the measured results of the fabricated memory indicated that the memory read-access time was 2.75 ns and the read-access power consumption was 2.7 pj/byte.

Based on the measured data obtained from the manufactured memory, we evaluated the performance of NVIDIA's deep-learning processor, NVDLA, as a motif processor. The power consumption and area were estimated by installing the proposed super-multiport SRAM between the cache and arithmetic units of the deep learning processor. The obtained results indicated 20.34% and 26.24% reductions in power and area, respectively.

Chapter 5 High-Accuracy and Low-Power Synchronization Technique Using Bluetooth

5.1 Introduction

Among IoT systems, time synchronization is an important technology for multisensor monitoring systems. Fig. 5.1 shows a conceptual illustration of the time synchronization system in a multisensor monitoring system. Time synchronization technology plays three major roles in multisensor monitoring systems. The first is to ensure the time consistency of the sensed data. Time synchronization ensures the sequential integrity of the data sampled by the distributed sensors. The second is the prevention of interference between the active sensors. As shown in Fig. 5.1(b), active sensors that use actuators or transducers, such as LiDAR and sonar, may cause interference if they operate simultaneously. To prevent interference, it is necessary to shift the measurement timing between the sensors. The third role is sensor coordination. As shown in Fig. 5.1(c), high-precision time synchronization between sensors enables the accurate calculation of the transmission time of a signal received by sensor B from the energy emitted by the transducer of sensor A. This capability enables a more sophisticated analysis.

The trade-off between power savings and accuracy presents a challenge in time-synchronization systems. As IoT devices are distributed in various environments and many operate on batteries or energy-harvesting systems, low power consumption is an important requirement for many IoT systems. Specifically, in environments where an external power supply or frequent battery replacement is not feasible, such as for rotating components (wind turbines and crankshafts) [41], power saving is essential. Accurate time synchronization is crucial for sensors and actuators that require real-time control. For example, in systems that

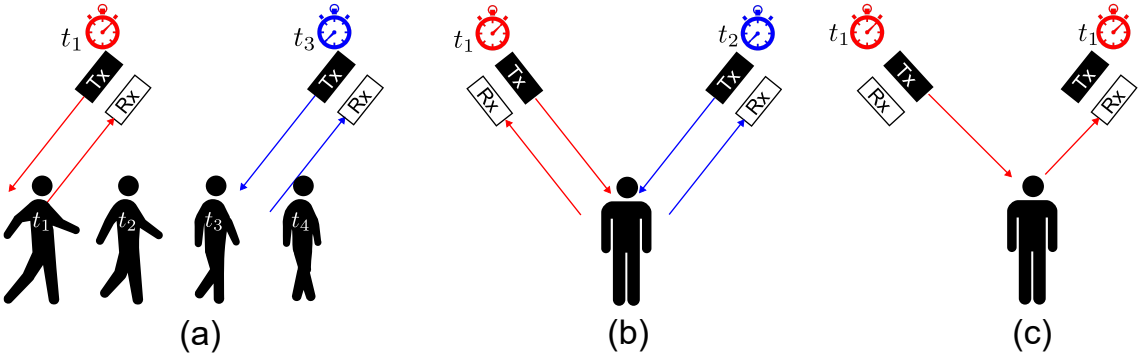


Fig. 5.1 Time Synchronization in Monitoring Systems. (a) Ensuring Measurement Sequence Consistency, (b) Preventing Multi-Active Sensor Collisions, and (c) Coordinated Multi-Sensor Measurements.

handle ultrasonic waves, such as ultrasonic position measurement [42] and ultrasonic sensing using microphone arrays [43, 44], synchronization on a sub-microsecond basis is necessary for rearranging the reception timing between microphones and measuring the elapsed time between the transducers and microphones. Additionally, when measuring vital data, such as in human motion sensing and electromyography (EMG) using distributed sensors, wireless time synchronization technology is required to synchronize measurement timing. Given that distributed sensors are often powered by batteries, low-power time synchronization technology enables long-term measurements. Hence, low-power and high-accuracy time synchronization is one of the most crucial services in IoT systems.

The motivation for this study stems from the critical need to address the trade-off between power efficiency and synchronization accuracy in IoT systems. To date, various studies on time synchronization in IoT systems using wireless communication technologies. In those systems, there is a trade-off between time synchronization accuracy and power consumption, because high-frequency timers with high-accuracy consume more power. This research proposes a new time synchronization method, MicroSync, based on BLE, a Bluetooth low-power standard. The core hypotheses of the proposed MicroSync are twofold. First, by combining a high-accuracy, high-power consumption timer with a low-power, low-resolution timer, it is possible to maintain high time synchronization accuracy without compromising energy efficiency. Second, further power savings can be achieved by optimally scheduling time synchronization communications to align with BLE communication timings. Based on these hypotheses, we have successfully developed a time synchronization system, MicroSync, that effectively balances low power consumption with high-accuracy.

Bluetooth Low Energy (BLE), a low-power wireless communication protocol standardized by the Bluetooth Special Interest Group, has emerged as a key IoT technology since its release in the Bluetooth 4.0 [45, 46]. In typical Bluetooth connections, a single computer or smartphone acts as the Central, and multiple Peripherals such as mice and sensor devices are connected. In the past decade, several synchronization technologies based on the BLE have been studied [41, 47-53], and some research has been conducted using these technologies [41, 54] However, few studies have focused on reducing power consumption in BLE-based time synchronization systems. In addition, to perform time synchronization, communication needs to synchronize with each other; however, a large power consumption instantaneously to drive the power amplifier (PA) and the low-noise amplifier (LNA).

Fig. 5.2 shows an overview of MicroSync, which is realized by two power reduction technologies: hybridization of timers and scheduling of synchronization. The hybridization technique combines a high-frequency clock (HFCLK) timer and real time clock (RTC). The proposed scheduling uses Slave Latency, a power-saving function of BLE, and pipelining, which can reduce the number of communications required for time synchronization. The three main points summarize the contributions of our study.

Two types of scheduling are proposed: high-accuracy and low-power scheduling. The high-accuracy scheduling achieved a mean absolute error of 272 ns and a standard deviation of 319.2 ns at 252.1 μW. Low-

power scheduling achieved a mean absolute error of 796.4 ns and standard deviation of 990.4 ns at a power consumption of 141.8 μW.

Furthermore, we proposed a hybrid approach to timer synchronization, achieving power savings with negligible degradation compared to an HFCLK timer implementation. In high-accuracy scheduling, the power reduction was 90.9% compared to that of the HFCLK timer. Low-power scheduling also achieved a 94.8% power reduction compared with the HFCLK timer implementation.

In the hybridization of timers, the running time of the HFCLK timer using crystal as the clock source was sufficiently short to suppress the increase in power consumption when the frequency of the HFCLK timer was increased.

The synchronization method we developed was implemented ion nRF52840, which is manufactured by Nordic Semiconductor. Furthermore, the program was built on Soft Device, a firmware released by Nordic Semiconductor, and does not directly program radio binaries; therefore, it can be used in many countries without worrying about compliance with technical standards.

The remainder of this paper is organized as follows. Section II introduces the previous work and clarifies the contributions of this study. In Section III, we explain the hybridization of timers, which is a power-saving technology used in this study, and the Slave Latency of BLE. Section IV describes the two proposed

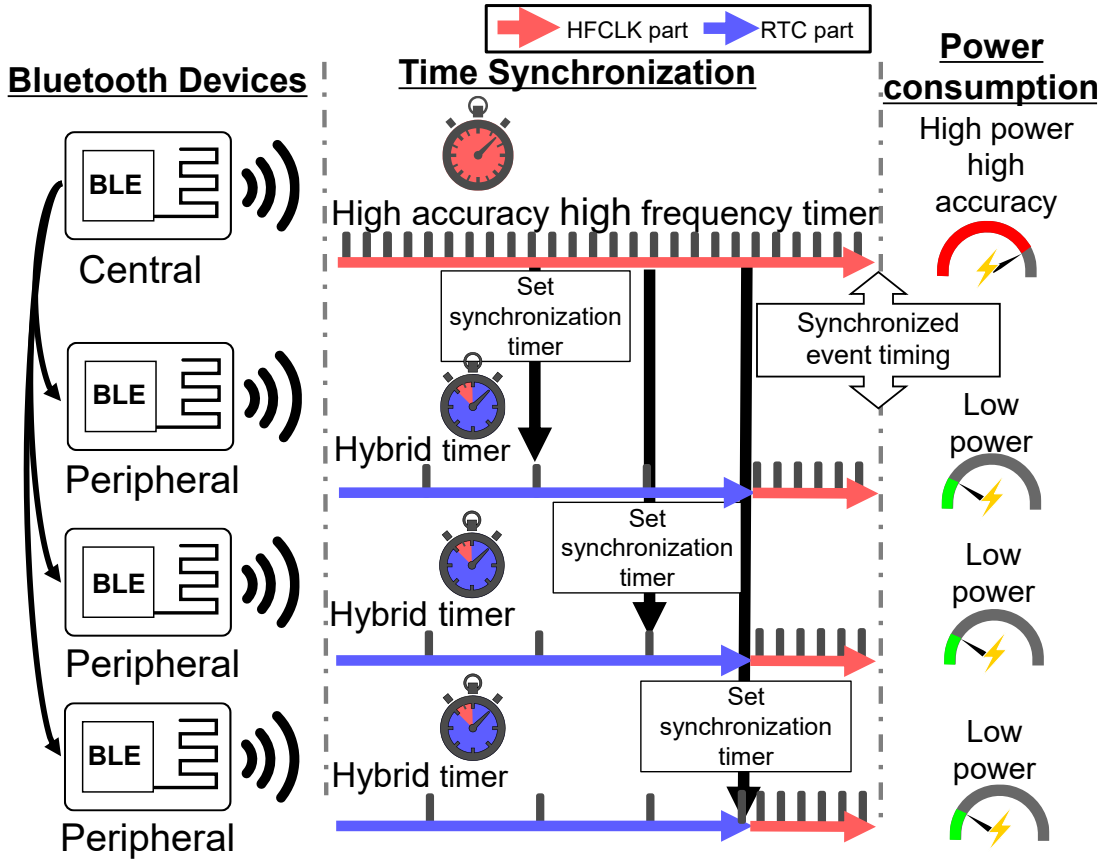


Fig. 5.2 Overview diagram of the MicroSync synchronization algorithm. [57]

MicroSync synchronization schemes: a high-accuracy scheme and a low-power scheme, and their hybrid timer implementation. Section V describes the experimental setup and conditions used to evaluate the performance of the proposed method. Section VI describes the experimental results and analyses. Section VII describes the superiority of the proposed method over conventional methods based on experimental results. Section VIII discusses future work based on the experimental results. Finally, section IX presents the conclusions.

5.1.1 Previous works

This section describes the various time synchronization methods proposed in the field of IoT.

The global positioning system (GPS) allows receivers to determine their current locations by receiving radio signals from multiple GPS satellites. Because GPS obtains time from atomic clocks installed on satellites, it allows the determination of absolute time with an accuracy of a few microseconds. This is applicable even for devices that cannot be equipped with expensive atomic clocks provided that the internal system clock can be accessed [55]. However, accessing GPS satellites requires that the device always receive signals from multiple GPS satellites, which is not optimal for IoT systems in indoor environments, such as smart homes, or for sensor networks installed in various locations under different conditions. In addition, the GPS also consumes a significant amount of power, which makes it unsuitable for systems designed to run on batteries for several years.

The Network Time Protocol (NTP) is a widely used synchronization protocol in IoT devices running on PCs and IP networks, and has excellent synchronization accuracy and robustness. Because it runs on IP networks, the implementation cost of the protocol stack is high owing to routing and other issues, and the power consumption tends to increase. Therefore, they are unsuitable for sensor networks and IoT systems that are not externally powered.

The Precision Time Protocol (PTP) provides more accurate time synchronization than NTP over IP networks. PTP is available over wired communication standards such as Ethernet (IEEE802) and compatible protocols such as Wi-Fi (IEEE802.11) [56]. A PTP can achieve sub-microsecond accuracy under ideal conditions and requires hardware PTP support in an Ethernet physical layer (PHY). PTP gained attention as the preferred network changed from CAN to Ethernet, and microcontroller implementations became available. In a previous study [57], a microcontroller with an Ethernet PHY and hardware support for PTP achieved synchronization with sub microsecond accuracy. As a wireless example, a prior study [58] implemented the PTP protocol on a microcontroller ESP32 with Wi-Fi. This implementation achieved a time synchronization accuracy of 15 μs without hardware PTP support, and PCs or single-board computers achieved sub-microsecond accuracy under Wi-Fi [59]. In a previous study [60], an FPGA implementation of PTP was proposed using low-speed, low-data-rate Bluetooth, and a synchronization accuracy of 15 ns was realized.

Reference Broadcast Synchronization (RBS) [61] is an algorithm that synchronizes the time between neighboring nodes instead of synchronizing them to a Universal Standard Time (UTC), such as GPS. The

reference node broadcasts a synchronization packet to all the nodes, and the receiving nodes exchange their local time when they receive the synchronization packet and calculate a correction value based on the error in their received time to achieve accurate time synchronization. However, there is a tradeoff: as the number of nodes increases, the accuracy improves, but the amount of communication also increases.

The Timing-sync Protocol for Sensor Network (TPSN) [62] algorithm achieves time synchronization through bidirectional communication such as NTP. In this algorithm, the receiving node is synchronized with the sending node. Synchronization is performed by first constructing an entire tree with the reference node as the root. The parent and child nodes of the tree then communicate with each other to perform time synchronization using multiple timestamps, which are generated at the MAC layer to reduce communication delays.

Some Bluetooth-based implementations require additional hardware for Bluetooth devices, whereas others do not. In addition, BLE has a generic access profile (GAP) for broadcast communication, such as advertising, and generic attributes (GATT) for one-to-one communication. Two types of communication methods are used for synchronization: one that establishes communication between the Central and Peripherals and the other that achieves time synchronization through broadcast advertising.

The current analysis is a typical method that uses additional hardware for time synchronization. In BLE communication, power pattern analysis, in which a shunt resistor is inserted into the power supply line of the BLE device and the potential difference is measured by an amplifier, is another method for determining the timing of transmission, in addition to the method using firmware interrupts. The current waveform of a BLE device includes the features of the transmitting and receiving amplifiers, from which the transmission current of the PA of the BLE device, the receiving current of the LNA, and other communication details can be accurately determined [47, 49]. In [52], synchronization with a variance of 0.9 μs was achieved by power consumption analysis using a shunt resistor and comparator.

In BLE, there are two types of communication: GAP, such as advertising communication, where data are broadcasted; and GATT, where bidirectional communication is performed by establishing a connection. CheepSync [51] and BlueSync [48] are methods for achieving time synchronization through advertising. Advertising requires only one-way communication. The time-source device only needs to consider the transmission power and does not need to consider the increase in power as the number of receivers increases. CheepSync combines synchronization algorithm that uses advertising and compensates for clock drift to achieve an average accuracy of 10 μs. BlueSync is another method for achieving time synchronization using advertising communication.

Reference [53] presented a method for time synchronization using GATT. In particular, the Central is connected to the multiple Peripherals via GATT communication, and the Central side uses the reference time sent from the Peripheral to calculate the synchronization time, which is communicated to the Peripheral, thereby realizing a synchronization time of 20 μs with an RTC of 32.768 kHz.

To date, there are few studies on time synchronization technology that focus on power savings. ecoSync, a low-power synchronization scheme, is implemented on ESP32 microcontroller that supports Wi-Fi. ecoSync realizes synchronization with the CPU clock by resetting the timing synchronization function counter internal to the Wi-Fi module. The results of an actual experiment using ESP32 DEVKIT V1 demonstrate a synchronization accuracy of 42 μs and power consumption of 182 μW at a resynchronization interval of 60 min.

In a previous study [63] a power-saving time synchronization method, which utilizes Bluetooth, was reported. This study focuses on Wireless Body Area Networks (WBANs) and assumes data collection from body-worn accelerometers. The employed network topology is a single-hop star topology, capable of supporting up to eight Peripheral devices. The synchronization algorithm uses a simplified method based on FSTP. When the resynchronization interval is set to 1 s, the synchronization accuracy is 4.42 ms and power consumption corresponds to 3.34 mW. When the synchronization interval is set to 60 s, the synchronization accuracy is 220 ms and power consumption is 55 μW. This synchronization accuracy is sufficiently higher than the inertial measurement unit's (IMU) sampling rate of 100 Hz, indicating that it is sufficiently fast for target application.

Reducing the number of transmissions and power consumption via scheduling is a well-studied topic in the field of sensor networks. Microcontrollers with BLE modules require mA-scale currents to drive LNA and PA. Reducing the number of communications contributes significantly to power savings. Compared with single-hop method, sensor networks have a larger number of nodes and more complex network topology. Therefore, communication collisions are likely to occur and power is likely to increase [64]. Certain studies involving FADS [65] and LPSRS [66] use scheduling to avoid communication collisions and reduce the number of communications. This in turn reduces power consumption. SLES[67] reduces the number of communications by 22% when compared with conventional synchronization algorithms such as EERS[68] and FADS. These algorithms focus on reducing overall system power by minimizing the number of communications, with limited discussion on the power consumption of individual devices.

5.2 Power saving method

The proposed MicroSync uses two techniques for power reduction: the hybrid control of timers and the utilization of Slave Latency. The two techniques are described in this section.

5.2.1 Hybrid timer

Most microcontrollers with BLE have two types of timers: HFCLK timers for high-accuracy time measurement and a real-time clock (RTC) for second to day long-time measurement. HFCLK timers are suitable for use in millisecond to microsecond real-time events and have high resolution in the time direction; however, they consume more power owing to their high clock frequencies. Furthermore, if an external crystal

is used, the power consumption increases. By combining these two types of timers with different time resolutions, power savings and highly accurate time synchronization can be achieved. Furthermore, the power consumption of a timer includes not only that of the timer counter itself but also that of the crystal or RC oscillator that serves as the clock source. Therefore, the power consumption can be reduced by turning off the power supply to the crystal oscillator, phase-locked loop (PLL), and other Peripheral devices.

RTC is designed to measure long periods of time with very low power consumption. They are suitable for measuring long periods of time at millisecond, minute, and day levels. In addition, RTCs are often intended for use when the processor is asleep; therefore, they have a low clock frequency and coarse temporal resolution but consume very low power as it draws a very small current, from a few microamperes to some tens of microamperes.

By combining these two types of timers with different time resolutions, power savings and highly accurate time synchronization can be achieved. Furthermore, the power consumption of a timer includes not only that of the timer counter itself but also that of the crystal or RC oscillator that serves as the clock source. Therefore, the power consumption can be reduced by turning off the power supply to the crystal oscillator, phase-locked loop (PLL), and other Peripheral devices.

Here, we describe a hybridization method between HFCLK and RTC timers. Generally, switching the system clock source from an RC oscillator to an external crystal results in a current flow of several hundred microamperes, even if the timer counter is stopped. Therefore, to save power, it is effective not only to stop counting the clock, but also to disable the crystal oscillator, switch the internal system clock to the RC oscillator, and restart the crystal oscillator and PLL only when necessary. However, the crystal oscillator and PLL are unstable immediately after start-up, requiring a warm-up time until the clock stabilizes.

Fig. 5.3 By combining these two types of timers with different time resolutions, power savings and highly accurate time synchronization can be achieved. Furthermore, the power consumption of a timer includes not only that of the timer counter itself but also that of the crystal or RC oscillator that serves as the clock source. Therefore, the power consumption can be reduced by turning off the power supply to the crystal oscillator, phase-locked loop (PLL), and other Peripheral devices.

Here, we describe a hybridization method between HFCLK and RTC timers. Generally, switching the system clock source from an RC oscillator to an external crystal results in a current flow of several hundred microamperes, even if the timer counter is stopped. Therefore, to save power, it is effective not only to stop counting the clock, but also to disable the crystal oscillator, switch the internal system clock to the RC oscillator, and restart the crystal oscillator and PLL only when necessary. However, the crystal oscillator and PLL are unstable immediately after start-up, requiring a warm-up time until the clock stabilizes.

The mechanism of timer hybridization is described in detail. Please note the following two points.

- The difference between an RTC timer and HFCLK timer is the resolution as opposed to precision of a single clock.
- The timer value is calculated with the resolution of the HFCLK timer.

The timer value is calculated as a real number and rounded off before it is stored in the timer register.

The left side of Fig. 5.3 shows the time recorded using the hybrid timer. First, an RTC timer interrupt is triggered. The value of the RTC timer is recorded as $\mathrm{RTC_{base}}$. Immediately thereafter, HFCLK timer is removed and started. If an external interrupt occurs while HFCLK timer is operating, then the value of HFCLK timer is recorded as $\mathrm{HFCLK_{base}}$. The absolute time at which an external interruption occurs is given by the following formula.

$$T_{\mathrm{HFCLK}} = T - round\left(T_{\mathrm{RTC}}(F_{\mathrm{HFCLK}}/\mathrm{F_{RTC}})\right) \qquad (5.1)$$

A disadvantage of this method is that the HFCLK timer must be started before any external interruption occurs.

The right side of Fig. 5.4 shows a timer interrupt using a hybrid timer. In this system, the timer value is expressed as an absolute value. Therefore, the interrupt timing is related to the base point T_{base}. Here, T indicates the time from T_{base} to the interrupt timing. In the hybrid timer, the interrupt time T must be decomposed into T_{RTC} and T_{HFCLK} and set in the timer register. The following equation represents the RTC time part counted by the RTC.

$$T_{\mathrm{RTC}} = \mathrm{round}(T/(F_{\mathrm{HFCLK}}/F_{\mathrm{RTC}}) - \mathrm{margin}) \qquad (5.2)$$

A margin is introduced to ensure that T_{HFCLK} does not become zero. This can occur when $T/(F_{\mathrm{HFCLK}}/F_{\mathrm{RTC}})$ is perfectly divisible, leaving no remainder.

The time recorded by HFCLK timer is calculated using the following equation:

$$T_{\mathrm{HFCLK}} = T - round\left(T_{\mathrm{RTC}}(F_{\mathrm{HFCLK}}/F_{\mathrm{RTC}})\right) \qquad (5.3)$$

The part counted by HFCLK timer is converted from remaining time T_{RTC} counted by RTC timer to the resolution of HFCLK timer and subtracted from T. Given that the value actually set as RTC interrupt timing

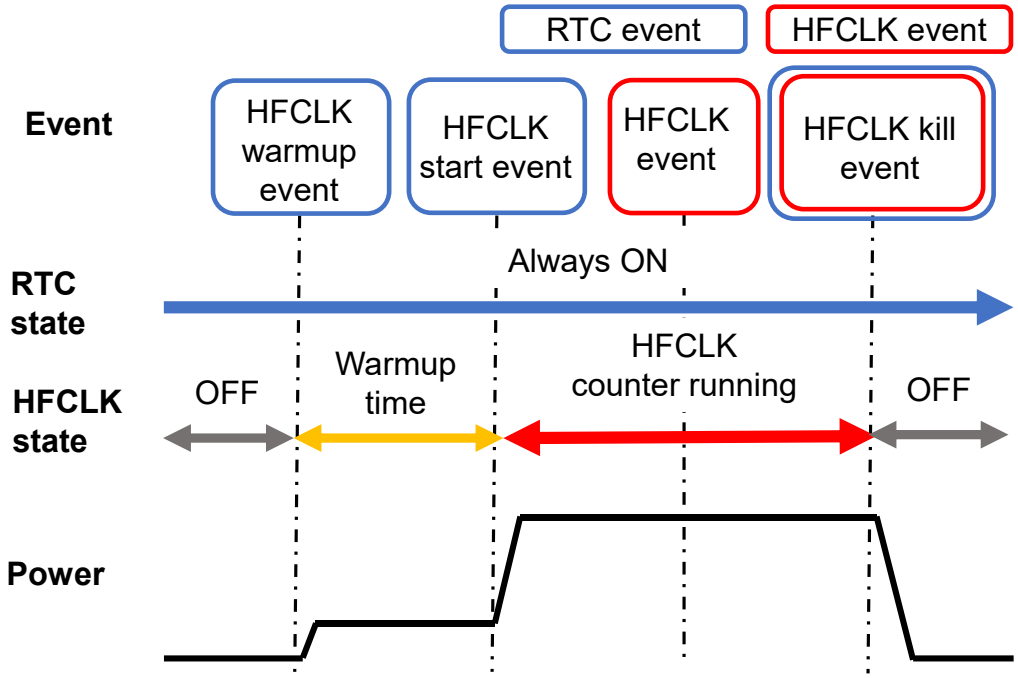


Fig. 5.3 Hybrid control of HFCLK timer and RTC. [57]

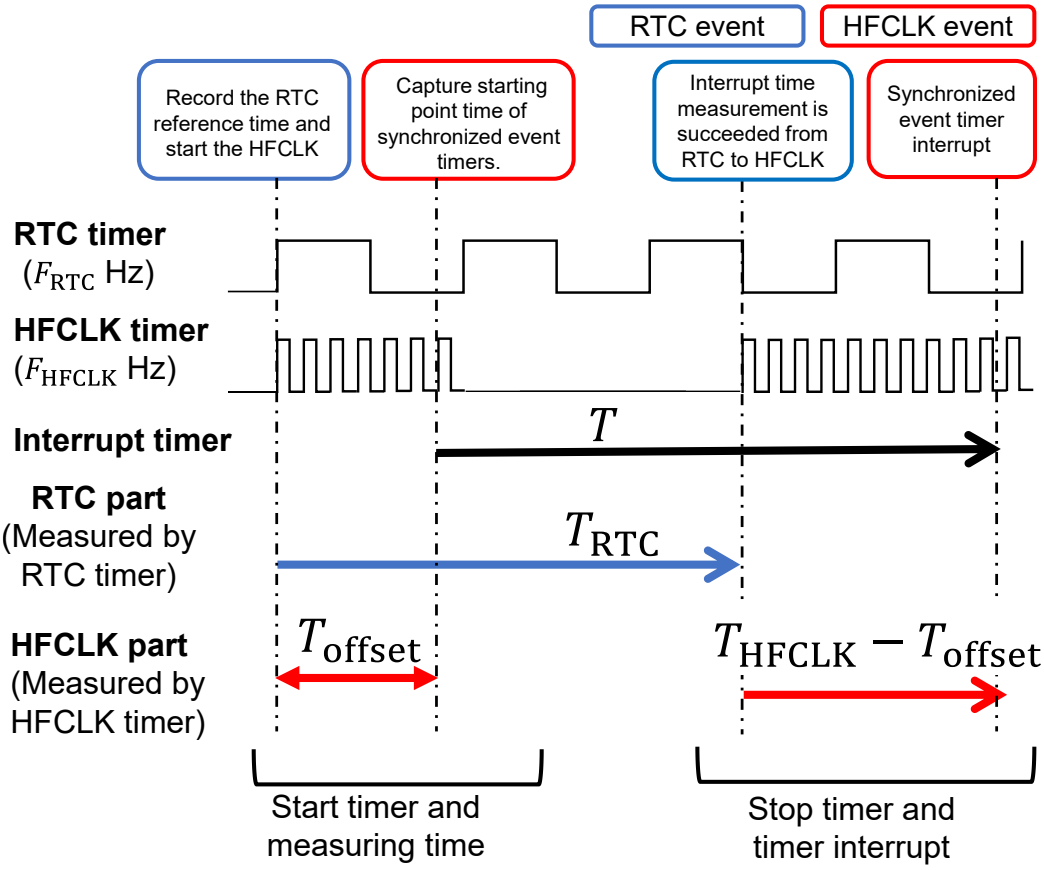


Fig. 5.4 Operation of the decomposed timers T_{RTC} and T_{HFCLK}. [57]

is relative to T_{base}, it is $\mathrm{RTC}_{\mathrm{base}} + T_{\mathrm{HFCLK}}$, and the value set in HFCLK timer is T_{HFCLK}. Fig. 5.3 shows the switching sequence from RTC to HFCLK timer, where the hybridization of the RTC and HFCLK timers is realized by three control events. The HFCLK warm-up event begins with a crystal oscillator and Peripheral devices. Next, after the warm-up time, the HFCLK start-up event is triggered by an RTC interrupt and HFCLK is started.

5.2.2 Slave Latency

Slave Latency is a power-saving function in the BLE standard. In BLE, GATT communication is performed after the communication between the Central and Peripheral devices is established. This communication interval is known as the Connection Interval, which can be set from 7.5 ms to 4 seconds.

Shorter Connection Intervals result in shorter delays between when the application layer communication data are ready and when they are transmitted. In other words, communication delays are reduced. However, the number of communications per unit of time increases, resulting in increased consumption. Conversely, if the Connection Interval is long, the number of communications is reduced and power is saved; however, the throughput worsens.

Slave Latency is a mechanism that avoids these problems by reducing power consumption when there are no data to send at the Peripheral side. Fig. 5.5 shows an overview of the Slave Latency. The squares at the top of the timeline indicate communication events. The green and orange rectangles represent receiving and transmission, respectively. Communication occurs at intervals of the Connection Interval, but Slave Latency makes it possible to ignore communication from the center a specified number of times. For example, if the Slave Latency is set to four, communication can be omitted when no data are ready for transmission, thereby reducing power consumption by a maximum of 1/4 in Peripheral.

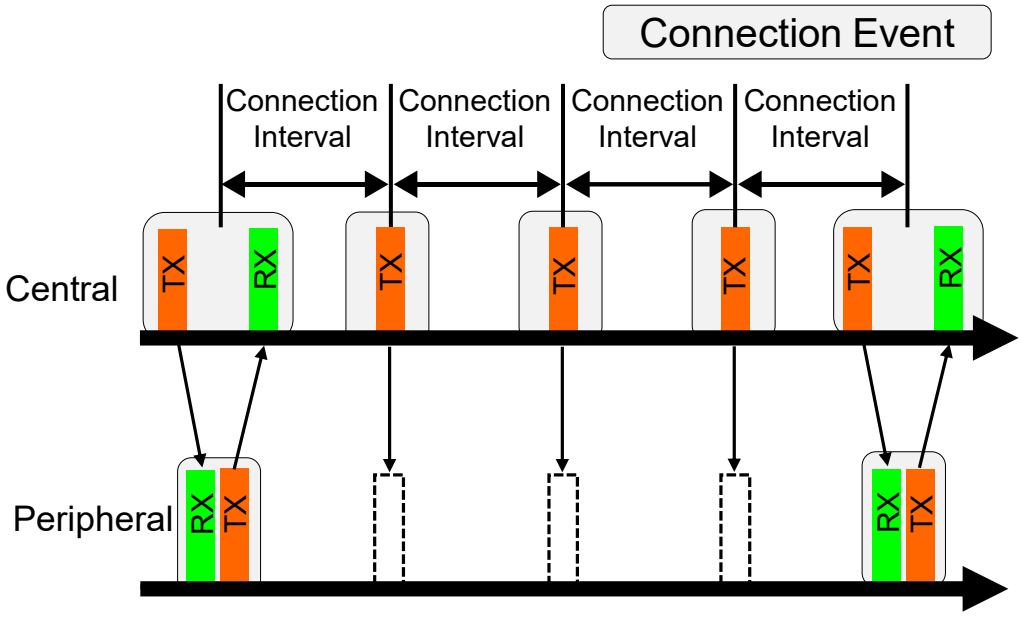


Fig. 5.5 Reduction in communication frequency by three in the case of Slave Latency. [57]

5.3 Communication scheduling

Fig. 5.6 presents an overview of the scheduling algorithms proposed in this paper. (a) represents the high-accuracy scheduling, where synchronization timing is calculated and shared through three communications. (b) represents the low-power scheduling, which pipelines the method used in (a). These algorithms will be described in detail in subsequent sections.

This section describes the MicroSync synchronization algorithm. Subsection A describes the communication and data exchange that form the basis of the theoretical synchronization algorithm. Subsection B describes the communication scheduling implemented for highly accurate synchronization using the synchronization algorithm detailed in Subsection A. Subsection C describes the pipelining implemented for this synchronization algorithm to achieve power saving. Subsection D describes the simple correction algorithm.

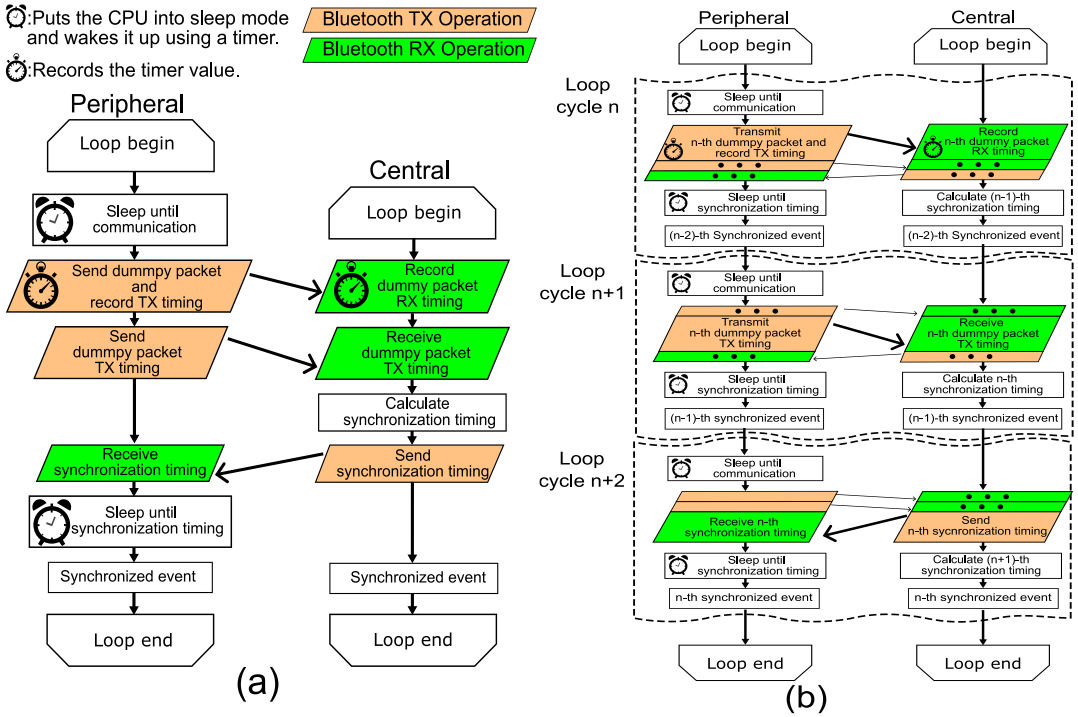


Fig. 5.6 Flowchart of synchronization scheduling algorithm (a) high-accuracy scheduling, (b) low-power scheduling [57]

5.3.1 Synchronization sequence

The synchronization sequence of the proposed algorithm is illustrated in Fig. 5.7. Synchronization consists of three communications: sending a dummy packet from the Peripheral to the Central device, sending a timer value for sending the dummy packet, and returning a timer value for timing synchronization. These communications were realized in three Peripheral events, A_n, C_n, and F_n, and three Central events, B_n, D_n, and E_n, as shown in Fig. 5.7.

First, a dummy packet is sent from the Peripheral event A_n. The Peripheral recorded the timer value for this event to determine the dummy packet transmission timing. After a slight delay, this dummy packet is received at the Central device in event B_n. This dummy packet reception time was recorded from the timer value of the Central device. At event C_n, the Peripheral device sends the recorded dummy packet transmission timing for event A_n. Because the timing of the send/receive events must be measured at the lower layer, the transmission timing of the first packet cannot be obtained at the application layer. Thus, two packets were used in this sequence.

The Central device can calculate the correct synchronization timing using this received packet at event D_n. At this time, it can know the timer value that the Peripheral has and recognize the deviation and the correction. The estimated synchronization timing is sent back to the Peripheral device from Central device in event E_n. In event F_n, the Peripheral sets the timer interrupt based on the packet received from the Central device. Then, at the time of synchronization, an interrupt is generated in the Peripheral as well as Central devices.

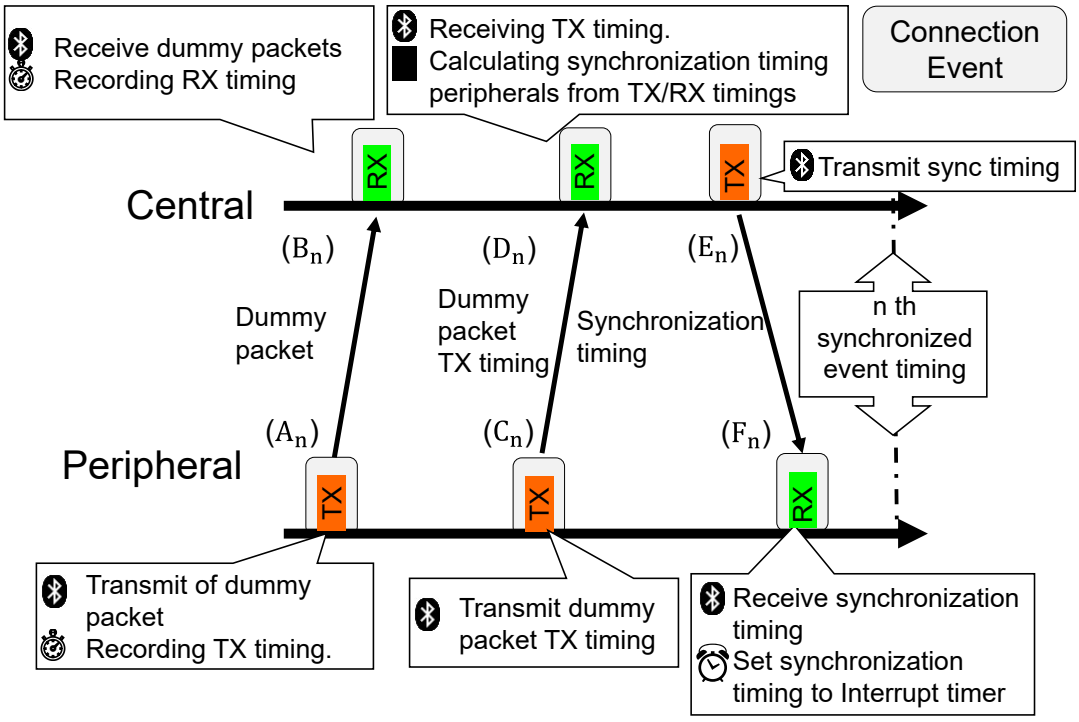


Fig. 5.7 MicroSync time synchronization sequence. [57]

This synchronization algorithm is an improvement over the communication algorithms presented in prior studies [53]. The improvement is that we modified it so that dummy packets are sent at any time from the Central side instead of the Central side requesting the transmission of dummy packets to the Peripherals. Consequently, the number of communications decreased from four to three. This reduces the power of the LNA associated with reception on the Peripheral side. These sequences can be performed in the same manner as when multiple Peripherals exist. The performance evaluation showed that they could be executed using three Peripherals.

Fig. 5.8 shows sequence of MicroSync implementation. The implementation of this system is as follows: This algorithm is realized by combining many functions such as event handlers and interrupt processing, from (a) to (i). Fig. 5.8 illustrates the sequence of the system. The sequence begins on the Peripheral side. First, function (a) is invoked by timer interruptions. This function reserves a dummy packet for transmission toward the Central region. Subsequently, in communication (b), the dummy packet reserved in (a) is transmitted by SoftDevice. With the transmission, PA pin rises and the interrupt function (c) is invoked by a GPIO interrupt. Given that the transmission reservation at the application layer differs from the actual transmission timing, it is necessary to measure the timing at the lower layer. The Soft Device has a function to externally output the timing of PA operation during BLE transmission as a GPIO. In the interrupt function (c), the timer value at the time of the GPIO interrupt is recorded as X_{tp}.

On the Central side, the dummy packet is received, LNA pin rises, and the time is recorded in the interrupt function (d). Subsequently, the receiver handler (e) on the Central side is invoked. The sender of the packet is

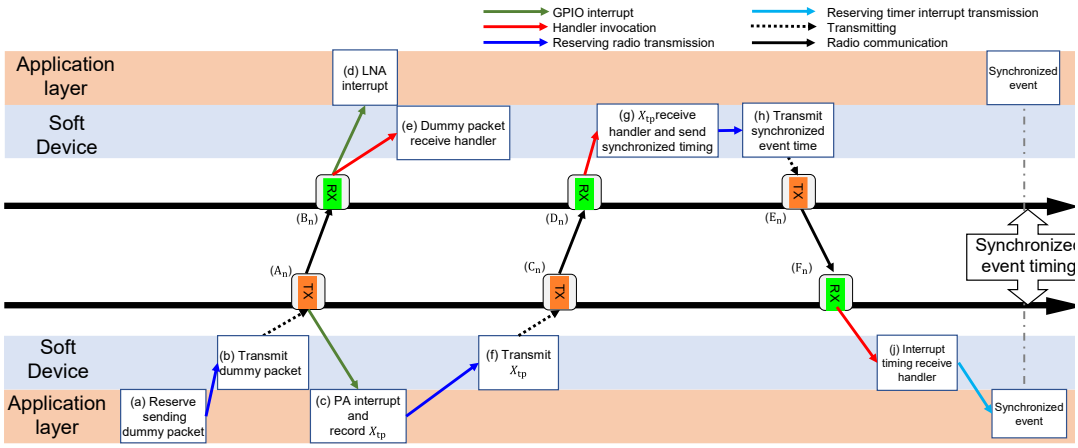


Fig. 5.8 Software sequence diagram for Peripheral-Central communication synchronization. [57]

not known until the packet content is examined by the receiver handler. The receiving time recorded in (d) is associated with the sender's Peripheral.

On the Peripheral side, in communication (f), the transmission time X_{tp} reserved for function (c) is transmitted. Additionally, it reserves the synchronized event timing S for transmission to the Peripheral. During communication (h), event timing S is transmitted. On the Peripheral side, the receiver handler (j) is invoked. In this handler, the synchronization timing sent in communication (h) is set to the interrupt timer, and a synchronized event is then generated.

5.3.2 High-Accuracy scheduling

This section describes the implementation of synchronization for high-accuracy scheduling using Slave Latency on BLE, calculation of synchronization timing, and hybridization of timers.

Our scheduling algorithm is driven and realized by several interruption events: communication interrupts, timer interrupts, and GPIO interrupts. A communication interrupt is used to detect Bluetooth communication events. Timer interrupts are triggered by timers such as RTC and HFCLK. Additionally, in this system, the firmware function detects the communication timing output from the GPIO by using GPIO interrupts on other GPIO pins.

First, in $(\mathrm{A_n})$ in Fig. 5.9 a dummy packet transmission event is generated by a timer interrupt in the application layer. In BLE, communication timing is restricted by the Connection Interval, and data cannot be sent at arbitrary times. Therefore, the transmission is reserved between the Connection Event to be transmitted and the previous Connection Event.

The interrupt timing at this time is set to X_{tp}. This X_{tp} is reserved for transmission to Central. The event in $(\mathrm{B_n})$ records the timing of dummy packet reception on Central side. Furthermore, Central device records the timing of the dummy packet reception using a Soft Device interrupt. When a dummy packet is received, the LNA notification pin is set and a GPIO interrupt is generated. The dummy packet receiving timing is

recorded within this GPIO interruption. Immediately after this, a Bluetooth receiver interrupt is generated, which is associated with the reception timing of the Peripheral and dummy packets in this function. The event in (C_n), X_{tp} is sent to the Central and received in event (D_n). In (D_n), timing from the dummy packet reception timing on the Central side to the interruption is added, starting with the transmission timing of the dummy packet sent from the Peripheral. The synchronization timing transmission is then reserved to return to Peripheral.

In this system, the synchronization cycle is 1 s. In high-accuracy scheduling, three communications are performed for each synchronization. Hence, the Slave Latency is assumed to be 20 ms. Therefore, setting Slave Latency to 47 times results in a synchronization period of 1 s per synchronization.

The calculation of the synchronization timing indicated in event D_n in Figs. 5.7and 5.8 is detailed in Fig. 5.9 First, a dummy packet is sent, and its transmission timing is recorded by the interrupt generated by the detection of the PA operation in the transmitter circuit. The Central device records the time using an interrupt generated by the LNA operation of the receiver circuit. Because these two timings do not coincide perfectly, a delay d occurs, as shown in Fig. 5.10. Here, t_1 denotes the elapsed time from the previous synchronization event to the current interrupt, t_2 represents the interval from the LNA to the reception of the interrupt, and L denotes the synchronization cycle, respectively. Using these terms, the time to the next synchronization on the Central side can be calculated. The synchronization time S is calculated at the Peripheral by adding the time until synchronization to the transmission time X_{tp}, which is transmitted from the Central to the Peripheral.

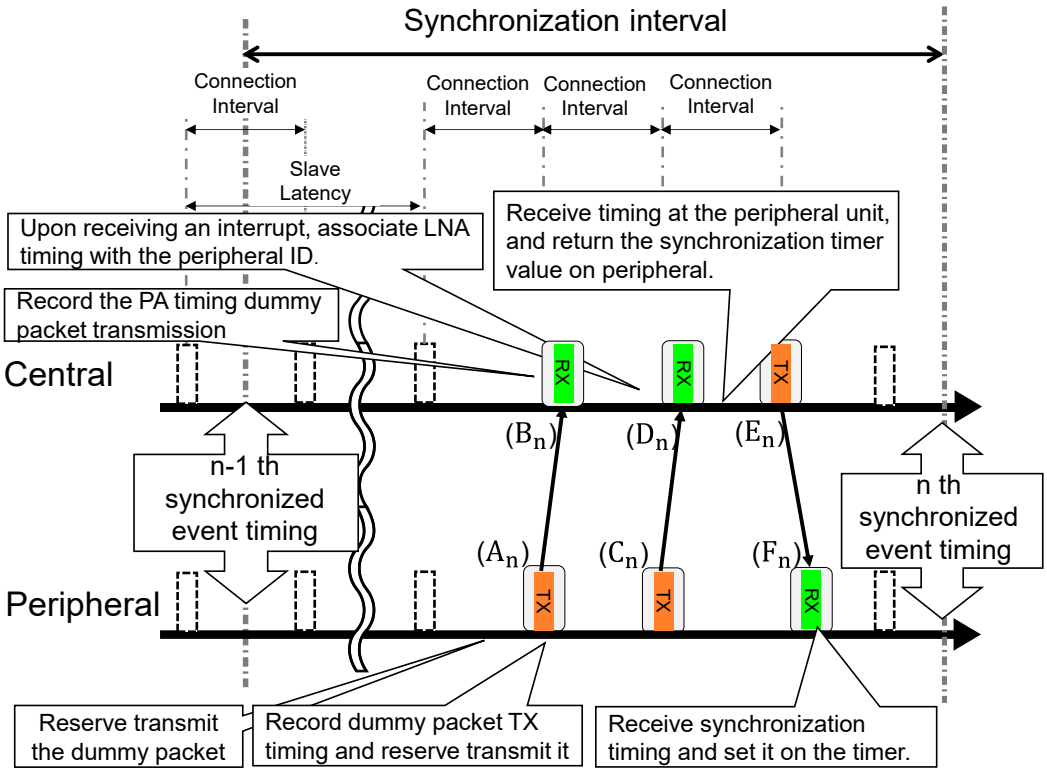


Fig. 5.9 Implementation of high-accuracy scheduling and power saving by Slave Latency. [57]

Although the delay d varies depending on the device circuit characteristics and communication conditions, its variation is sufficiently small for the desired synchronization accuracy. Therefore, d can be measured in advance and hard-coded as a fixed value.

The value of S is given by the following equation:

$$S = X_{\mathrm{tp}} + (L - t_1) + t_2 + d \quad (5.4)$$

Equation (5.4) defines the timer value S for the next Synchronized Event of the Peripheral. Here, X_{tp} is the common reference timing for the Central and Peripheral regions. L is the synchronization interval and t_1 is the time since the last Synchronized Event. The term $(L - t_1)$ is the time remaining until the next Synchronized Event from the point of the receiver interrupt. Variable t_2 is the time between the GPIO interruption triggered by the LNA and its detection, and d represents the timing error in X_{tp} between the Central and Peripheral devices. In summary, the timer setting for a Synchronized Event at a Peripheral location comprises X_{tp} plus $(L - t_1)$, t_2, and d.

Fig. 5.11 presents an overview of the hybridization of high-accuracy scheduling. In our scenario, the Central device assumed that the power consumption constraints were negligible or sufficiently modest and that HFCLK was always running. Therefore, the hybrid operation described here was mainly adapted to Peripherals. Mission timing, and the counter value of the RTC was recorded at this time. When an interrupt

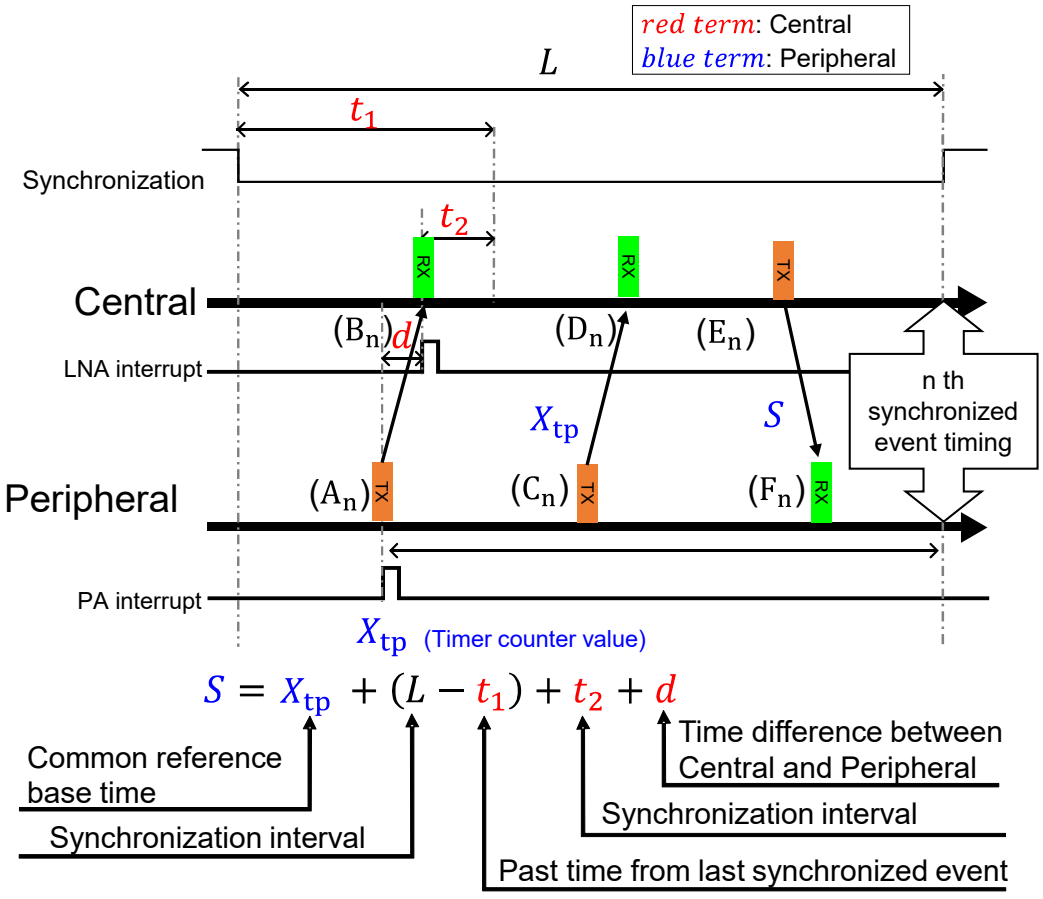


Fig. 5.10 alculation of synchronization timing in high-accuracy scheduling. [57]

caused by the PA operation occurred, the HFCLK timer counter value at that time was recorded and the HFCLK was stopped. This timer value was sent to the Central device, which returned the synchronization timing to the Peripheral. Because the Central device calculated all the time as HFCLK, the Peripheral decomposed the time until the transmission timing was received from Central into the RTC and HFCLK parts, and set them in the timer. Thereafter, when transmitting dummy packets, the RTC interrupt served to warm up the HFCLK, after which the HFCLK timer started counting, and the HFCLK timer interrupt reversed the synchronous timing pin and stopped the HFCLK and HFCLK timer circuits.

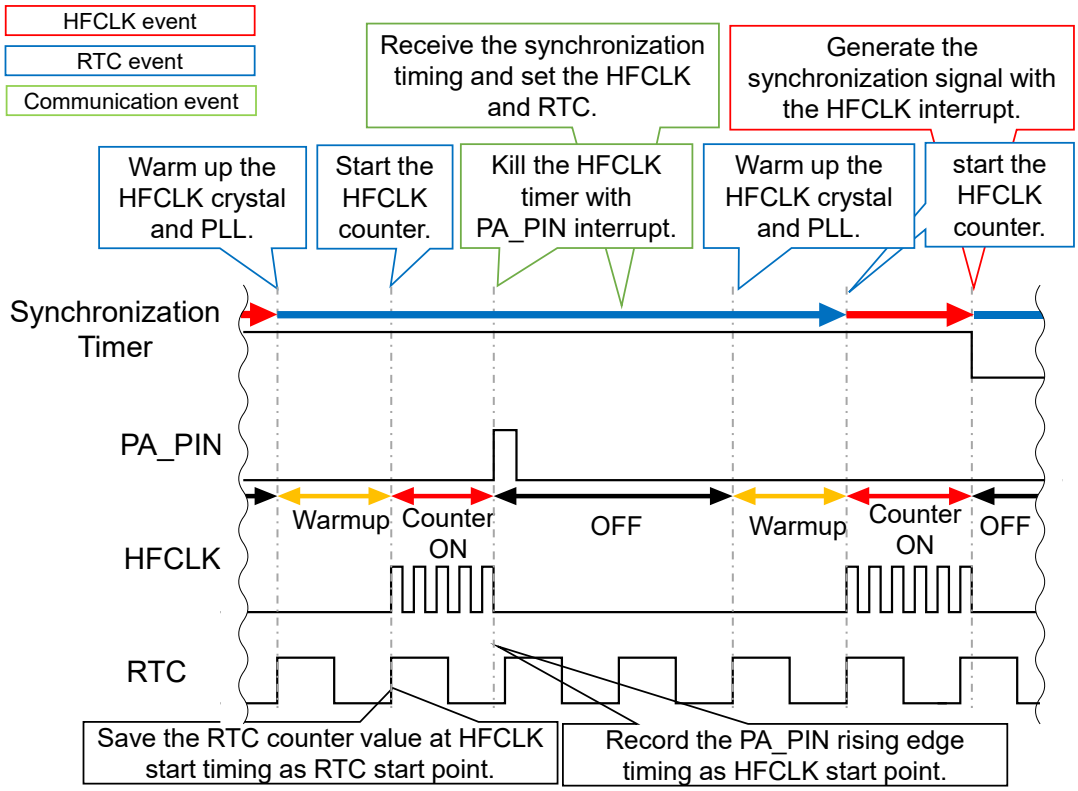


Fig. 5.11 Overview of hybridization of high-accuracy scheduling. [57]

5.3.3 Low-power scheduling

In high-accuracy scheduling, three communications are required for each synchronization. These communications are the transmission of dummy packets from the Peripheral to the Central device, the transmission timing of dummy packets, and the transmission of synchronization timing from the Central to the Peripheral. Because these communications do not conflict with each other, it is possible to reduce the power consumption by pipelining the 3-step communication and reducing the number of communications.

Fig. 5.12 shows the concept of pipelining communication. In this figure, n represents the communication regarding the n-th synchronized event, and the figure shows three communications and synchronized events from $(n-2)$-th to n-th. First, a packet was transmitted from the Peripheral to the Central device, and the

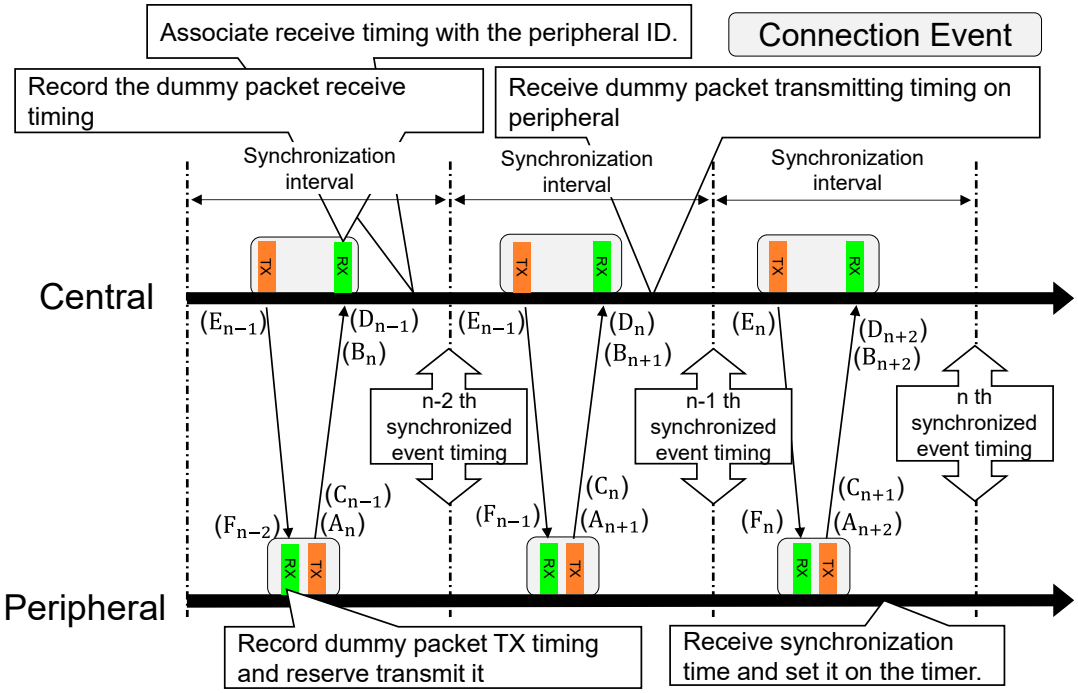


Fig. 5.12 Communication pipelining in low-power systems. [57]

transmission timing was recorded. This packet substitutes for two functions: the n-th dummy packet and the $(n-1)$-th transmission timer value packet, as shown in Fig. 5.7. In this communication event, the $(n-2)$-th synchronization event timing is also received and set to interrupt timer. These three functions do not conflict and occur during a single communication event.

Next, n-th dummy packet transmission timing is sent from the Peripheral to the Central device. This packet also has the function of the $(n+1)$-th dummy packet. When the n-th timing of the dummy packet transmission is received at the Central device, the synchronous event timer value on the Peripheral device is calculated from the dummy packet transmission timing value on the Peripheral device and the received timing value on the Central device.

Finally, a synchronized event-timer value is sent from the Central to the Peripheral device. The Peripheral device then sets the received timer value to that of the interrupt timer. Subsequently, a synchronous event is generated.

This scheduling method have two main advantages. First, the number of communications is less than that in high-accuracy scheduling. It consumes less power than the high-accuracy scheduling. Second, this method does not require Slave Latency. In a multi-Peripheral environment with Slave Latency, the communication order may change, which causes power overhead. This scheduling does not require Slave Latency and the communication order is stabilized. The weakness of this method is that it requires two or more synchronization intervals for synchronous scheduling, which can cause accumulation of timer drift errors.

Fig. 5.13 shows the calculation of the synchronization timing in low-power scheduling. For simplicity, we have omitted the explanation of pipelining and described a single-synchronization calculation. Three

communications were required for each synchronization. Subsequently, each type of communication is explained. In the first communication, a dummy packet is sent from the Peripheral to the Central device. At this time, on the Peripheral side, the transmission timing X_{tp} was recorded by the PA interrupt. Simultaneously, on the Central side, the reception time and t_1 were recorded by the LNA interrupt. Here, d was considered to be a constant indicating the propagation delay and was determined experimentally. In the second communication, the Peripheral sends the transmission timing. Using this time as the starting point, the synchronization timing is calculated until the next time. Because the synchronization timing occurs after the third communication, the time from dummy packet reception to transmission is t_1 and the sum of $2L$ and d denotes the time lag between the PA and LNA. The sum of these times is added to X_{tp} to obtain the timer value S. In the third communication, this S is returned.

In low-power scheduling, Equation (5.5) defines the timer value S for the next synchronization event of the Peripheral.

$$S = X_{\mathrm{tp}} + (2L + t_1 + d) \qquad (5.5)$$

The primary difference from Equation (1) is the communication interval; in low-power scheduling, this interval is 1 second, which necessitates three communications and results in synchronization events lasting more than 2 seconds. X_{tp} represents common reference timing to that used in high-accuracy scheduling. The term t_1 represents the time between X_{tp} and the next synchronization event, which contrasts with its usage in high-accuracy scheduling where it has a different role. As the synchronization timing in this communication occurs $2L$ later, adding $2L$, t_1, and the error d to X_{tp} yields the next synchronization timing.

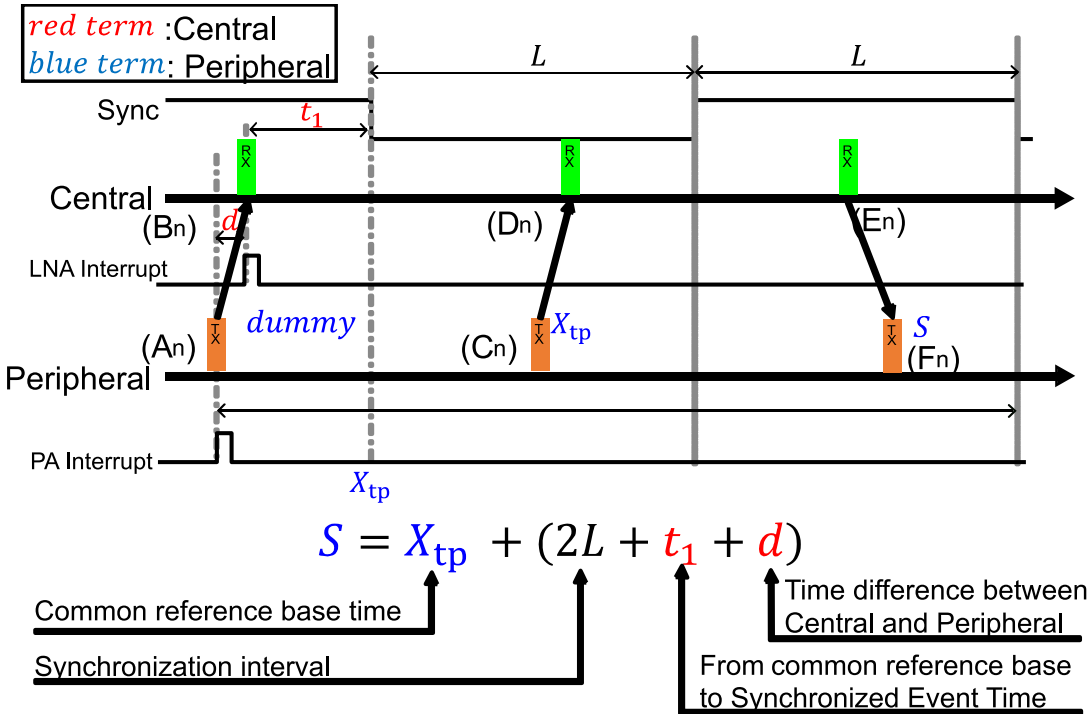


Fig. 5.13 Calculations in low-power scheduling. [57]

5.3.4 Simple correction algorithms

Packet loss during communication is unavoidable in wireless systems. To address these issues, we developed a simple correction algorithm. Fig. 5.14 shows a flowchart of the proposed algorithm. The algorithm is realized by combining multiple interrupt handlers, which is differs from the actual software flow. The correction algorithm has two stages: an initialization stage and a correction stage. First, in the initialization stage, the stability of the communication is determined. Given that the communication timing in BLE is not stable immediately after the connection, it is necessary to determine whether the communication is stable. The absolute difference between the maximum and minimum values of the last eight synchronization timings received from the Central was calculated; and if the difference is within the allowable value, the system is considered to be stable. The permitted value is $2\lceil F_{HFCLK}/10^6 \rceil$.

The correction stage involves the use of predictive values. The correction stage uses predictive values. The predicted value is the timing of the occurrence of the Synchronized Event calculated using the value sampled from the timer on the Peripheral side of the timing of data transmission and reception. The difference between the predicted value and received Synchronized Event timing is calculated, and if this difference is less than or equal to a threshold value, then the Synchronized Event timing received from the Central is judged to be correct. If the difference is greater than the threshold value, then the synchronization timing received from the center is determined to be correct. This value is then set to the timer of the Synchronized Event and used as the starting point for the next predicted timing. If it is greater than the threshold value, the predicted value is judged to be more accurate, and the predicted value is set in the timer. The above algorithm alleviates the outliers due to disrupted or missing communication.

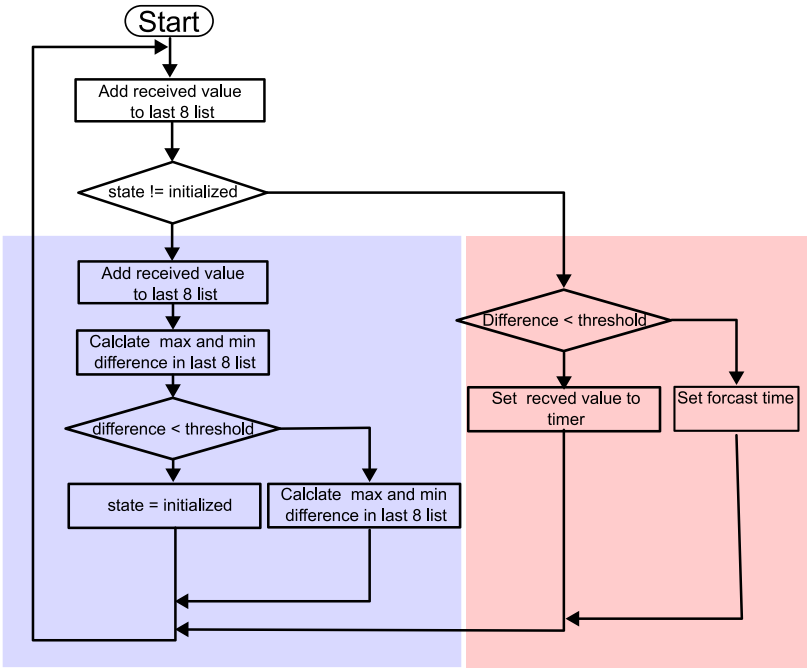


Fig. 5.14 Flow chart of simple correction algorithm. [57]

5.4 Implementation and evaluation

Prototyping was conducted using a commercially available Bluetooth microcontroller to demonstrate the effectiveness of the proposed method. One Central and three Peripheral devices were implemented, and actual wireless communication was performed to evaluate accuracy.

5.4.1 Experimental setup

Fig. 5.15 shows the devices used in the evaluation experiment. An evaluation board nRF52840DK with an nRF52840 Bluetooth microcontroller (Nordic Inc.) was used for both Central and Peripheral devices. This board has a port to connect to the Power Profiler Kit2(PPK2), shown as PPK2 in Fig. 5.15.

Furthermore, to measure power, LEDs and Peripheral circuits are separated from the MCU power supply, allowing only the power consumption of the MCU to be measured. The power consumption of the Peripheral node was measured using a power measurement unit (Power Profiler Kit, Nordic Inc.). PPK II is a power meter that utilizes a shunt resistor. It can measure power from 200 nA to 1 A in real time with an accuracy of 100 nA to 1 mA. In this study, power waveforms were analyzed using this device.

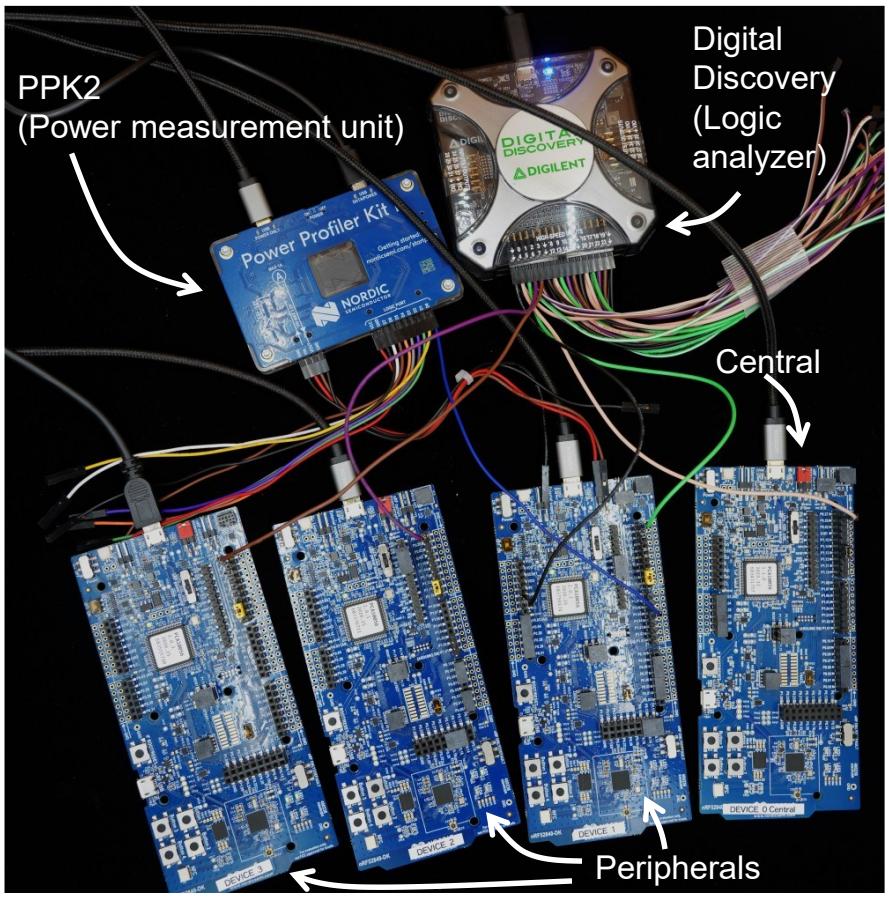


Fig. 5.15 Experimental setup [57]

A logic analyzer (Digital Discovery, DIGILENT) was used to measure the synchronization error between the devices. With an 8-bit measurement target, the Digital Discovery 2 realizes high-accuracy at a sampling frequency of 800 MHz and sampling interval of 1.25 ns. Additionally, Wave Forms, an application for viewing signal patterns obtained from Digital Discovery 2, not only displays patterns, but also has functions for data analysis and visualization using JavaScript-based scripts. In this experiment, WaveFroms function is used to trigger the rising edge of the Central timing notification pin. We then constructed an environment to measure the degree to which the increase in the Peripheral timing notification pin deviates from the Central timing notification pin.

5.4.2 Evaluation method

Fig. 5.16 shows a schematic of the experimental setup. The experimental system comprises of four nRF52840DKs, one of which is used as the Central unit and the remaining three are used as Peripheral units. All Peripherals are connected to the Central and synchronized to the timing of the Central synchronization timing. The Peripherals and Central have synchronization pins to notify the synchronization timing by toggling the synchronization pins. The dotted lines in the Fig. 5.16 connects the synchronous timing pin to the logic analyzer which detects the synchronous timing. The synchronous timing signals are connected to a Digital Discovery 2 logic analyzer. The logic analyzer is set to 800 MHz in 8-bit mode and calculates the time difference between the rise and fall timings of the Peripheral's synchronous signal relative to the Central's synchronous signal. The difference in toggle timings is automatically recorded using WaveForms as many times as required.

Fig. 5.17 presents a diagram illustrating the connection between the nRF52840DK and the Power Profiler Kit II. First, the power circuit output of the nRF52840DK's MCU is fed into the VIN terminal of the Power Profiler Kit II. Second, the current is measured utilizing a shunt resistor and an instrumentation amplifier. Finally, the current is routed back from the VOUT terminal back to the MCU. Peripherals, such as LEDs are

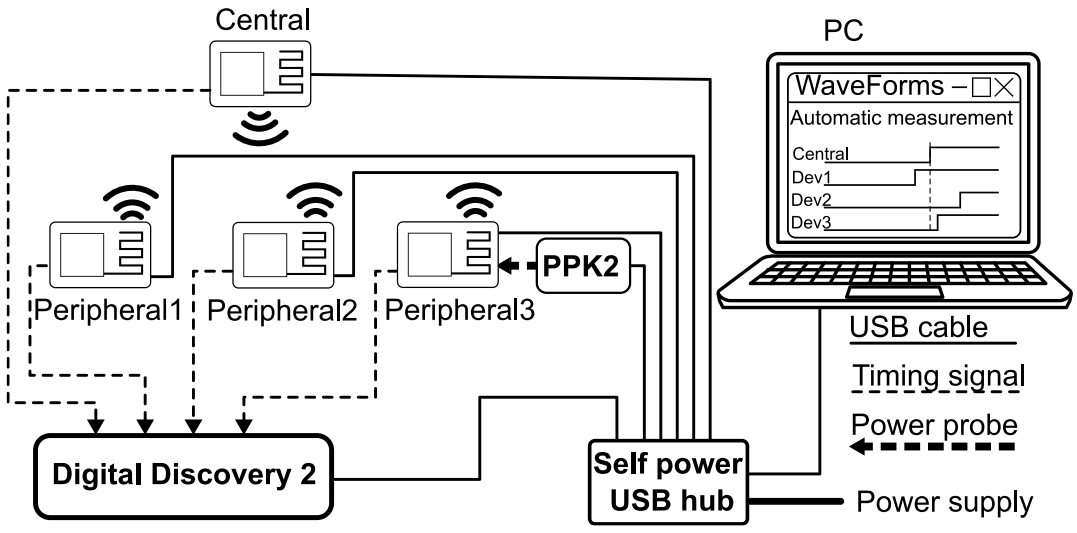


Fig. 5.16 Connection of measuring instruments. [57]

connected to an independent power source, to prevent any interference with the current measurement. The entire setup, encompassing both the MCU and Peripherals, is powered by USB. The diagram has been significantly simplified to facilitate understanding, particularly the representation of the Power Profiler Kit II's power measurement circuit. For more detailed information on the circuit constants, please refer to [69].

The experimental results were compared with those of the implementation using regular timers. We will evaluate the implementation using three types of timers: RTC, high-frequency timer, and Hybrid. Additionally, the evaluation varied the frequency of high-frequency timers for synchronization algorithms involving time measurements, such as the high-frequency timer and Hybrid. The evaluation range was from 500 kHz to 16 MHz for the high-accuracy scheduling and from 500 kHz to 4 MHz for the low-power scheduling. taking 10 measurements of the average current over one minute, and the average of these values was considered.

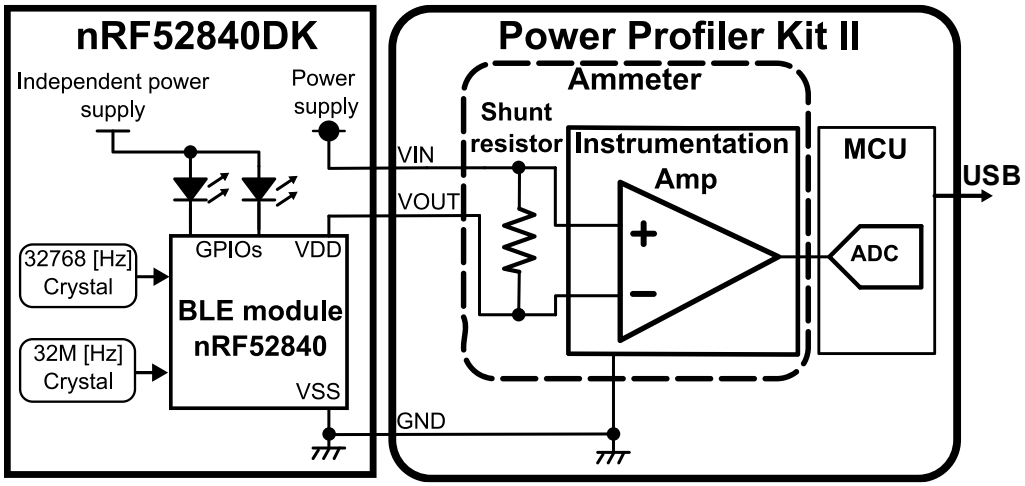


Fig. 5.17 Diagram of the connection between the nRF52840DK and the Power Profiler Kit II. For details on the Power Profiler Kit II circuit, see. [57]

5.4.3 Evaluation result

Fig. 5.18 summarizes the results of each experiment. Synchronization was performed using three different timers: RTC only, HFCLK only, and the hybrid timer. The frequency of the programmable HFCLK was varied from 0.5 MHz to 16 MHz for performance evaluation. For each condition, both high-accuracy and low-power scheduling were implemented and evaluated. The hybrid timer system reduced power consumption by one order of magnitude. In addition, the synchronization accuracy was slightly improved with the hybrid method compared to the case where only HFCLK was operated. This is a slightly counterintuitive result, but it is due to the variation ranges in the frequency of the RTC and HFCLK. The frequency variation of RTC was sufficiently small compared to that of HFCLK.

Fig. 5.19 shows a breakdown of the relationship between the HFCLK frequency and synchronization error for a combination of scheduling schemes. In high-accuracy scheduling, there is little difference in the accuracy between the HFCLK-only and hybrid timer cases. By contrast, in low-power scheduling, the hybrid type was

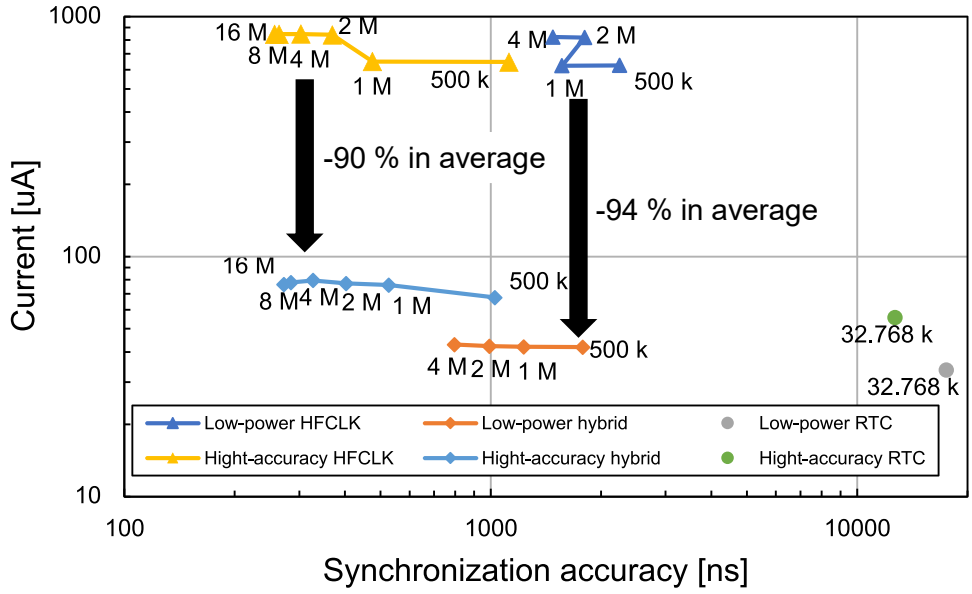
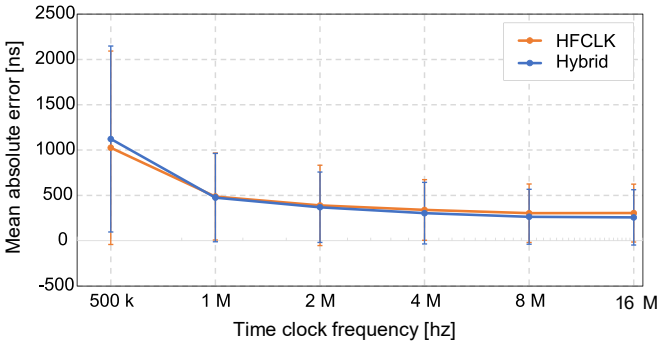
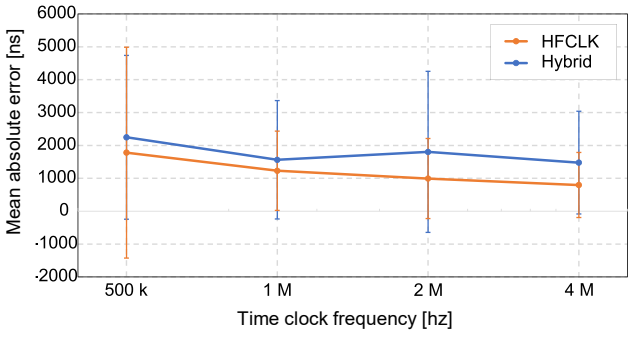


Fig. 5.18 Comparison of accuracy and current for each scheduling and timer. [57]

found to be more accurate than HFCLK. This was caused by the frequency variation between HFCLK and RTC. Low-power scheduling is strongly affected by HFCLK frequency fluctuations, which exhibit large variations because of the long periods required for synchronization. In high-accuracy scheduling, the period



(a)



(b)

Fig. 5.19 Relationship between HFCLK frequency and synchronization error with (a) high-accuracy scheduling and (b) low-power scheduling. Error bars represent standard deviations. [57]

required for synchronization is a few hundred milliseconds, whereas in low-power scheduling, it takes a few seconds.

Fig. 5.20 illustrates the cumulative time synchronization error for high-accuracy scheduling at 16-MHz and low-power scheduling with a 4-MHz hybrid timer. The 99% cumulative synchronization errors are 591.25 ns and 2028.75 ns with high-accuracy and low-power scheduling, respectively.

Fig. 5.21 shows an example of a current consumption waveform with a single synchronization cycle. The red and blue lines show that HFCLK is activated for dummy packet transmission and interrupts the handling for synchronization. In high-accuracy scheduling, following the transmission of dummy packets, the receiver is active multiple times in response to communications between other Peripherals and the Central device. In low-power scheduling, this power waste can be avoided by setting the Slave Latency.

As a result, the accuracy of the high-accuracy scheduling was 272 ns with a 16-MHz HFCLK, as shown in Fig. 5.19. The accuracy in low-power scheduling is 990 ns with a 4-MHz hybrid timer. Average power consumptions with these conditions are 252.1 μJ and 141.8 μJ, respectively. These values represent the average values measured in the same manner as the current waveforms, as shown in Fig. 5.21.

Tables 5.1 and 5.2 show the power reductions achieved by hybridizing timers in each scheduling method. In high- accuracy scheduling, power consumption is reduced by 90%, and in low-power scheduling, it is reduced by 94%.

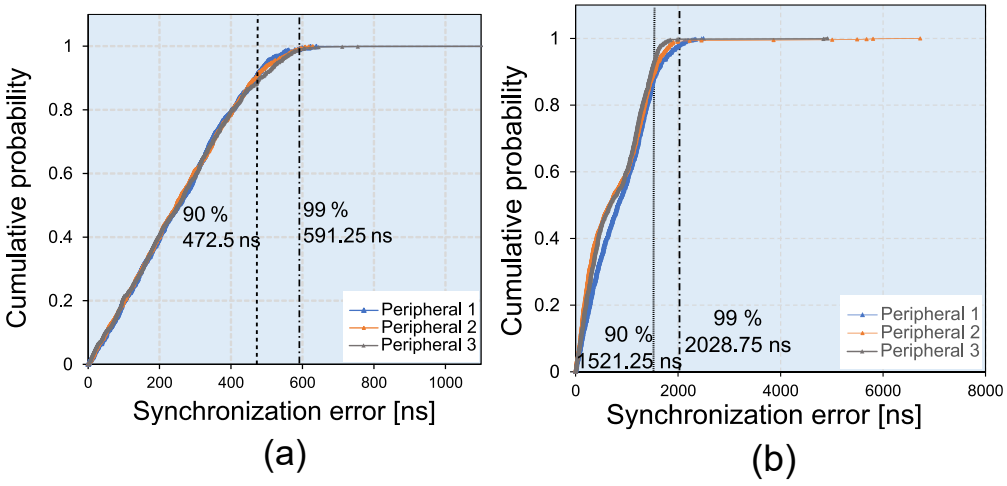


Fig. 5.20 Error accumulation of (a) high-accuracy scheduling with 16-MHz HFCLK and (b) low-power scheduling with 4-MHz hybrid timer [57]

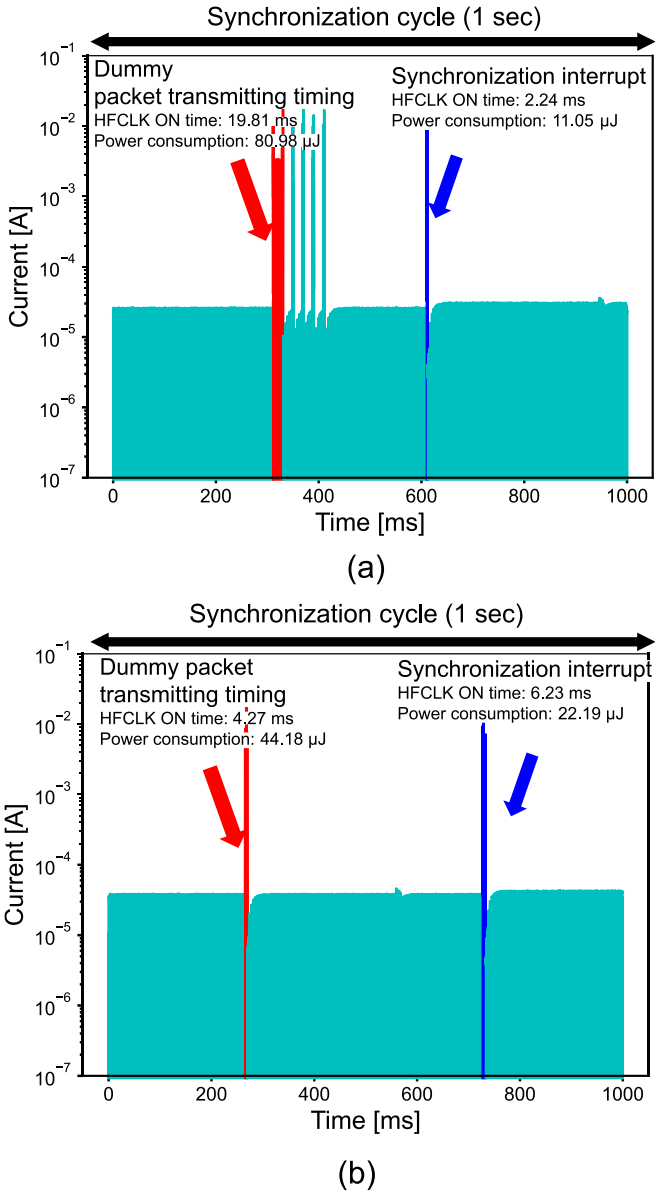


Fig. 5.21 Measured current consumption waveforms while synchronization cycle with (a) high-accuracy scheduling and (b) low-power scheduling. [57]

Table 5.1 Power Reduction by Hybridizing Timers in High-power Scheduling [57]

	Timer frequency [Hz]						
Timer type	32.768k	500k	1M	2M	4M	8M	16M
HFCLK [µJ]		2134.4	2146.3	2770.4	2793.8	2791.1	2784.9
Hybrid [µJ]		222.7	250.8	225.1	262.7	257.4	252.8
(Reduction rate)		(0.10)	(0.12)	(0.09)	(0.09)	(0.09)	(0.09)
RTC	183.8						

Table 5.2 Power Reduction by Hybridizing Timers in Low-power Scheduling [57]

	Timer frequency [Hz]				
Timer type	32.768 k	500k	1M	2M	4M
HFCLK [μJ]		2066.6	2060.0	2699.2	2714.1
Hybrid [μJ]		138.3	138.9	139.6	141.8
(Reduction rate)		(0.07)	(0.07)	(0.05)	(0.05)
RTC	111.3				

5.4.4 Power analysis

To estimate the worst-case power consumption for high-accuracy scheduling and low-power scheduling, we analyzed and formulated the power consumption. Power consumption of microcontrollers supporting Bluetooth includes the CPU, radio communications, such as LNA and PA, standby power, and timers such as RTC and HFCLK.

In both scheduling modes, a current of approximately 20 μA always flows as steady-state power, P_{base}. These includes the leakage current and the power consumption of the RTC. In this section, the variables starting with P denote power consumption per second and variables starting with E denote energy consumption per event. The power consumption of high-accuracy scheduling includes three communication events: (A)-(B), (C)-(D), and (E)-(F), as well as time-synchronized events and Soft Device systems.

The power consumption for the communication event between (A) and (B) includes the power used by the HFCLK timer to measure the communication timing and the power required for radio communication. The HFCLK operates from the time an interrupt occurs until the radio communication and PA pins are set. The maximum HFCLK operating time is twice Connection Interval, denoted by t_{CI}, and warm up time denoted by t_{warm}. When HFCLK timer operates at 16 MHz and it requires approximately 800 μA of current according to actual measurements. The power consumption per unit time of this timer is defined as P_{HFCLK}.

The communications (C)-(D) and (E)-(F) are communication-only events, denoted by E_{CD} and E_{EF}, respectively, and consume approximately 26 μW and 27 μW of power. Communication does not always occur during the Connection Interval and can be occasionally skipped. In such cases, the CPU remains activated to prepare for communication, consuming approximately 14 μW of power, defined as E_{sys}. The power consumption of a Synchronized Event, defined as E_{HaInt}, consumes approximately 10 μA of power. Specifically, the power consumption remains nearly constant because HFCLK's running time variation is within one RTC cycle. Based on these factors, the worst-case power consumption for high-accuracy scheduling is defined as shown in Equation (3) below.

$$P_{\text{worst}} = P_{\text{base}} + (t_{\text{warm}} + 2t_{\text{CI}}) + E_{\text{CD}} + E_{\text{EF}} + 3E_{\text{sys}} + E_{\text{HaInt}} \quad (5.6)$$

Next, the power for low-power scheduling is formulated. Given that low-power scheduling is conducted at regular intervals with slight uncertainty in communication, there is no requirement to target only the worst-case power consumption.

Power consumption comprises of base power T_{base}, the timer and communication power denoted as E_{com}, and power consumption of synchronized events, denoted as E_{LpInt}. In E_{com}, communications between (A)and (B), (C) and (D), and (E) and (F) are pipelined and occur simultaneously. Due to the constant communication interval in low-power scheduling, it is possible to reduce the running time of the timer used for measuring communication timing. This results in a power consumption of approximately 46 μW as measured in practice. Finally, the power consumption of the synchronized event, denoted as $\mathrm{E}_{\mathrm{LpInt}}$, also remains nearly constant, consuming approximately 22 μW. Based on these factors, the power consumption for low-power scheduling can be defined as shown in Equation (4) below.

$$P_{\mathrm{worst}} = P_{\mathrm{base}} + E_{\mathrm{com}} + E_{\mathrm{LpInt}} \qquad (5.7)$$

5.5 Discussion

Table 5.3 shows a comparison of the proposed method with previous studies on time synchronization methods using Bluetooth.

In [53], highly power-efficient time synchronization using only RTC was presented. Our high-accuracy and low-power scheduling methods achieve time synchronization that is ten times more effective than this method. The proposed method improves upon this work by achieving synchronization with less communication. In our implementation, measurements indicate that the power required for one communication is approximately 32 μJ. Based on this value, we can estimate that if the algorithm in [53] was implemented in a manner similar to our program, the estimated power overhead would be approximately 36.4% for high-accuracy scheduling and approximately -1.5% for low-power scheduling.

The largest difference between BlueSync[48] and the proposed method was the length of measurement time. Our approach is designed for synchronization at 1-second intervals, whereas BlueSync aims for longer, minute-scale synchronization and includes advanced timer drift compensation. BlueSync combines two timers, HFCLK and RTC, based on, [70] to measure the time. BlueSync and the proposed method differ in the control of the HFCLK and RTC timers. An HFCLK timer was used as the system clock, and HFCLK could not be stopped during processing. The proposed method starts not only the HFCLK timer, but also the HFCLK clock source before using the HFCLK timer and releases it after the HFCLK timer stops. Therefore, it is superior in terms of power consumption performance.

In Table 5.2, the communication timing is detected using comparator-based power consumption analysis [49]. The most significant difference between this method and the proposed method is the presence of an additional device to measure the current. The proposed system detects communication timing by GPIO

interrupts using the Soft Device function for time synchronization. Thus, it is superior in terms of cost and power consumption because this measurement does not require Peripheral devices.

The PTP protocol is a time-synchronization protocol for an IP network. PTP can be used in the IoT and embedded systems such as PCs, servers, and sensor networks. The accuracy of this protocol is 7.8 ns [57] for MCU implementations that support PTP hardware and 15 μs for Wi-Fi MCUs that do not support PTP. Compared to time synchronization via Bluetooth, PTP has equal or better accuracy; however, the protocol is more complex and consumes more power [58]. Therefore, power-saving systems, such as those in the proposed MicroSync, will be more important for Bluetooth-based time synchronization systems.

Comparing ecoSync and MicroSync, a power-saving time synchronization method, MicroSync's low-power scheduling outperforms ecoSync's in terms of power consumption and accuracy. This is despite the fact that ecoSync's resynchronization interval is 60 min, which is 3600 times longer than MicroSync's 1-s interval. This is due to the fact that ecoSync runs on ESP32, which supports Wi-Fi, whereas our system uses nRF52840. This supports BLE, a Bluetooth power-saving standard. Additionally, the proposed system can realize higher throughput using less power by combining measurement data and time 1synchronization in a single packet.

The performance of MicroSync was compared with that of conventional time synchronization, which assumes acceleration measurements using an IMU. First, as shown in Table 5.3, MicroSync achieves 4.9 times better synchronization accuracy and 23.7 times better power consumption at a resynchronization interval of 1 s. Second, we compared and analyzed the conditions from both hardware and application perspectives to

Table 5.3 Comparison between the proposed and conventional methods . [57]

Method name	Data link protocol	Extra device requirement	Mean time sync. accuracy	Resync interval	Energy
Prop. (High-accuracy)	Bluetooth GATT	No	0.272 μs	1s	252.1 μW
Prop. (Low-power)	Bluetooth GATT	No	0.990 μs	1s	141.8 μW
[53]	Bluetooth GATT	No	20 μs	1s	32 μJ (estimation)
BlueSync [48]	Bluetooth GAP (Advertising)	No	4.5 μs	1m	n/a
[52]	Bluetooth GATT	Power measuring circuit	0.9 μs	100 ms	n/a
PTP [57]	Ethernet	PTP supported Ethernet PHY	7.8 ns	Continuous	n/a
PTP (Software) [58]	Wi-Fi	No	15 μs	Continuous	386 mW
PTP [60]	Bluetooth GATT	FPGA	15 ns	Continuous	n/a
ecoSync [58]	WiFi	No	42 μs	60 min	182.83 μW
[63]	Bluetooth GATT	No	4.4 μs	1s	3.34 mW
[63]	Bluetooth GATT	No	200 μs	60 s	55 μW

identify the differences. In terms of hardware, MicroSync uses nRF52840, whereas the previous study used nRF52832. Hence, there was no significant difference in power-saving performance. On the application side, MicroSync's experiment focuses only on time synchronization, whereas the previous study not only focused on synchronization but also actively acquired acceleration data. The sampling rate of the IMU is 100 Hz. However, previous methods have shown that reduced communication leads to lower power consumption; even considering that 100-Hz sampling by the IMU is insufficient to fully explain the 23.7-fold increase in power efficiency. For an accurate comparison, an analysis with a power monitor, such as PPK2, using actual equipment is necessary.

The MicroSync methodology broadens its applicability through two key technologies: timer hybridization and communication scheduling. First, most of microcontrollers with radio modules have an RTC and high frequency timers and timer hybridization are generalizable. Second, the scheduling algorithm of MicroSync is optimized for the BLE protocol and operates with minimal communications. Bluetooth is an event-driven protocol and communication events occur periodically. Therefore, scheduling is aligned with these timings. This indicates that MicroSync can be applied to periodic communication protocols. Zigbee is one such protocol. The difference from Bluetooth is that the child node queries the parent node during communication. Zigbee uses a mechanism termed as Polling for periodic communication, which makes it possible to implement low-power scheduling. Furthermore, it is possible to switch between two types of Poll intervals, Long Polling and Short Polling. This suggests the potential for implementing high-accuracy scheduling.

The following is a summary of the points on which our method outperforms conventional methods. Firstly, the power consumption of timers is a critical issue in power-saving time synchronization systems, where timers with high frequency and high resolution typically lead to increased power consumption. The proposed method uses a hybrid timer to achieve both high time synchronization accuracy and time resolution. Secondly, in time synchronization, shorter synchronization intervals lead to more frequent communication, and consequently, greater power consumption. However, extending the communication interval often results in decreased accuracy due to timer drift. Conversely, MicroSync achieves short resynchronization cycles and power savings by optimizing the hybrid timer and communication scheduling. This capability suggests that when MicroSync is employed in a sensor network, it enables high-frequency data sampling with both high-accuracy and low power consumption.

Based on the experimental results and the characteristics of BLE, we predicted the practical limitations of the MicroSync application. Firstly, the Soft Device specification limits the number of connectable Peripherals to a maximum of 20 devices. Consequently, MicroSync can only connect to 20 Peripherals per central device. Secondly, if multiple Peripherals request quick transmissions at random times, the communication timing may become irregular, potentially leading to unstable operations. Therefore, the maximum number of Peripherals that can stably operate is likely to be even lower. Thirdly, this study focused only on time synchronization. If MicroSync is applied to a sensor system that gathers data intensively from Peripherals, then the power consumption may escalate as the volume of data communication increases.

In MicroSync, we observed time synchronization errors of 272 ns in high-accuracy scheduling and 990 ns in power-saving scheduling. To investigate the causes, we examined the communication timing and timer cycles on the Central and Peripheral devices. The investigation revealed that the timer values between the Central and Peripheral devices occasionally exhibited discrepancies ranging from several tens to approximately 100 counts at a frequency of 16 MHz. Potential factors influencing these dis Using MicroSync, we observed time synchronization errors of 272 ns for high-accuracy scheduling and 990 ns for power-saving scheduling. To investigate the causes, we examined the communication timing and timer cycles on both the Central and devices. The investigation revealed that the timer values between the Central and Peripheral devices occasionally exhibited discrepancies ranging from several tens to approximately 100 counts at a frequency of 16 MHz. Potential factors influencing these discrepancies are considered as follows.

- Communication Detection Error: MicroSync utilizes the notification features of LNA and PA in SoftDevice to detect communication timings. When LNA or PA is activated, SoftDevice raises a designated pin to signal an event. In our system, this change is detected through a GPIO interrupt, which enables time synchronization. Therefore, the timing of notification pin activation and the duration of GPIO interrupts introduce uncertainties in synchronization.
- Disorders in Communication Order: In BLE, multiple Peripherals are connected to a single Central unit and communicate in a time-divided, concurrent processing manner. Therefore, the order of Peripherals can change during communication, leading to disturbances in communication timing.
- Communication Delays: In wireless systems, various factors, such as the transmission time in the physical layer, the propagation time of information, and the reception time, can cause delays. In the application layer, these delays manifest as the difference between the rise times of the PA pin on the transmitter side and LNA pin on the receiver side. Disturbances in these delay times due to interference with other devices can adversely affect synchronization accuracy.
- Recovery Time from Sleep: To enhance power efficiency, our system places nRF52840 into deep sleep. The recovery time from sleep in nRF52840 has two delay modes, and our system selects a more power-efficient mode. This introduces uncertainties in the time it takes to wake up from a timer interruption.
- Timer Drift: MicroSync assumes synchronization intervals of up to one second, aligning with the maximum Connection Interval of one second due to the characteristics of the nRF52840. Thus, clock drift does not dominate synchronization errors, and we did not implement any correction for clock drift.

5.5.1 Implementation in monitoring system

As shown in Table 5.3, MicroSync achieves the highest level of accuracy and lowest level of power consumption among Bluetooth-based short-period time synchronization systems. The time synchronization accuracy of this system is sufficient for handling ultrasonic waves, particularly in multisensor devices (Fig.

5.1(c)). Ultrasonic sensors typically use sound waves ranging from 40 to 100 kHz. The wavelength of these ultrasonic waves is 10–25 μs. MicroSync achieves a synchronization accuracy of 0.272 μs in high accuracy scheduling and 0.990 μs in low-power scheduling. This is 10–20 times more precise than the wavelengths of the ultrasonic waves used in ultrasonic sensors. MicroSync enables multiple sensor devices to collaborate and process signals at a phase level.

Furthermore, MicroSync's robust synchronization framework is not limited to ultrasonic sensors. It can also be applied effectively to other time-critical applications, such as high-speed imaging, where frame rates may reach tens of thousands or even millions of frames per second. In these scenarios, sub-microsecond synchronization is crucial for ensuring precise alignment of frames across multiple cameras. Moreover, with the progression of Bluetooth audio protocols toward higher sampling rates—such as 96 kHz or 192 kHz—synchronization requirements tighten to intervals as short as 10.4 μs or 5.2 μs. MicroSync exceeds these requirements by at least an order of magnitude, making it suitable not only for cutting-edge acoustic sensing but also for advanced multimedia applications that demand stringent timing accuracy.

MicroSync is highly power-efficient, consuming 252.1 μW for high accuracy scheduling and 141.8 μW for low power scheduling. Assuming MicroSync is powered by a single alkaline AA battery (2,900 mAh capacity) and boosted to 3.3 V using a DC-DC converter with 80% efficiency to supply power to an nRF52840, the estimated operating time is approximately 1.6 years for high accuracy scheduling and 2.5 years for low power scheduling. This low power consumption allows MicroSync to allocate more energy for measurement tasks when integrated into monitoring systems. In addition, MicroSync can be used to manage the sleep state of monitoring systems, thereby reducing the overall power consumptions of the systems.

5.6 Conclusion

We focused on the power consumption of Bluetooth time synchronization and proposed MicroSync, which is a synchronization method that achieves sub-microsecond accuracy with power requirement of tens of microamperes. MicroSync is based on two elements: timer hybridization and communication scheduling. Two types of MicroSync implementations are proposed and experimentally evaluated: high-accuracy scheduling and low-power scheduling. The proposed method is implemented and evaluated using a commercially available Bluetooth microcontroller. The evaluation results show that the proposed method achieves time synchronization with an accuracy of less than 1 μs. This is a higher accuracy than conventional methods, and hybridization has shown that synchronous processes can be realized with very low energy consumption.
These results show that the accuracy overhead due to timer hybridization is negligible. Furthermore, the results of the power performance comparison show that the synchronization system with 1-s intervals saves the most power. These results show that the two hypotheses presented in the Introduction are correct and useful.

Chapter 6
3-D Object Reconstruction from Outdoor Ultrasonic Image and Variation Autoencoder

6.1 Introduction

Air-coupled ultrasonic measurement, which does not require direct contact with the object, finds application in a wide range of fields such as agriculture [71], robotics [72, 73], and self-position estimation [42]. Among these diverse applications, time-of-flight (ToF) ultrasound imaging [74-77] stands out for its capability to measure the shape of an object in three dimensions. Although light detection and ranging (LiDAR) and millimeter-wave radars are commonly used for three-dimensional (3D) measurements [78], the effectiveness of LiDAR can be compromised by the optical properties of the object being measured, such as poor weather conditions and transparent objects [79, 80], unlike ultrasonic imaging which remains unaffected by these factors. Additionally, ultrasound imaging is less expensive than both LiDAR and millimeter-wave radar, providing a cost-effective alternative for 3D measurement applications.

Ultrasound imaging employs beamforming and ToF technologies. Beamforming enhances signals from various directions using the phase differences between transducers. ToF is a technology that estimates the distance to an object based on the travel time of sound waves emitted by a device, which are then reflected off the object and returned to the device. By integrating these techniques, it is possible to perform ToF calculations from multiple directions, thereby enhancing the accuracy and versatility of ultrasound imaging. These technologies find applications in diverse fields, including object detection [81] and non-destructive testing [82]. Furthermore, research has explored 3D measurements using air-coupled ultrasound to investigate its potential in various scientific and industrial domains. However, most previous studies on 3D air-coupled ultrasound and beamforming imaging have been limited to indoor distances within 10 m, with only a few studies extending beyond this range [72-74, 83-85].

Outdoor ultrasonic measurement presents challenges owing to the need to eliminate various sources of noise such as multipath and environmental disturbances. However, if ultrasonic 3D imaging can be successfully implemented outdoors, it could serve as an alternative in environments where optical cameras are impractical owing to privacy concerns, such as residential areas. This technology holds potential for diverse applications, including monitoring road conditions, human behavior, and traffic volume in residential areas during times of disaster.

We have developed a 3D reconstruction system that uses ultrasonic imaging technology. This system is designed to monitor road conditions, where approximations of the outer shapes of objects are sufficient without the need for details. The system achieves 3D reconstruction by integrating the ToF point cloud obtained through 25 kHz ultrasound beamforming and direct sequence spread spectrum (DSSS) [86] modulation with a deep learning model, specifically a variational autoencoder (VAE) [87]. We have developed a comprehensive computational pipeline for this system, encompassing both hardware development and 3D reconstruction processes, utilizing a VAE that integrates all necessary components from data acquisition to analysis. The system successfully reconstructs a 30 × 30 × 30 voxel space from ultrasound signals sampled at 227 kHz, achieving a precision of 0.939, recall of 0.868, and F-value of 0.902, which highlights the effectiveness of our approach in 3D ultrasonic imaging.

The remainder of this paper is organized as follows: Section II describes the theoretical and computational pipelines that we have developed for ultrasound imaging. Section III details our measurement system, including the ultrasound array and recording hardware. Section IV presents our experimental results and the accuracy of the 3D reconstruction. In Section V, we compare our findings with those of previous studies. Finally, in Section VI, we present our conclusions.

6.2 Experimental setup

This section reviews previous research on 3D reconstruction using air-coupled ultrasound and acoustic traffic monitoring.

The 3D measurements based on air-coupled ultrasound are realized using the ToF of ultrasound. In a previous study [84], two types of distances were targeted: the far field, which is more than 2 m away, and the near field, which is approximately 200 mm away in an acoustic chamber. High-precision 3D imaging was achieved by measuring an ultrasonic array of 64 transducers, which also served as microphones, arranged according to a Fermat spiral with an aperture of 200 mm. One of the major contributions of this study is the demonstration of the effectiveness of microphone placement based on the Fermat spiral in ultrasonic imaging. This study shows that microphones arranged in a Fermat's spiral pattern reduce the side lobes and grating lobes better than those arranged in a grid pattern, thus improving the quality of ultrasonic imaging. The evaluation experiments were conducted at far-field and near-field distances. In the far-field, all 28 corner reflectors at a distance of 2 m were accurately measured individually. In the near-field, precise measurements of a human finger were obtained at a distance of 20 mm.

A previous study [83] focused on the hardware aspects of 3D measurements using air-coupled ultrasound. In this study, a microphone array, field-programmable gate array (FPGA), and graphics processing unit were employed to construct a highly compact ultrasonic measurement system that integrates all components from data acquisition to beamforming. One of the significant contributions of this study is the proposal of a cohesive hardware system that encompasses not only the ultrasonic array but also signal processing. Although acoustic

sensing research often highlights the cost-effectiveness of microphones compared to LiDAR and millimeter-wave radar, only a limited number of studies have implemented them, especially within edge signal processing systems. Despite being an embedded system, it achieves 30 FPS with beamforming of 240 beams.

Previous research has focused on ultrasonic measurements at targeting distances exceeding 10 m. In a previous study, researchers successfully imaged objects positioned 16 mm away with a resolution of 25 × 25 pixels [85]. Furthermore, in our previous investigation [74] we successfully conducted position measurements of objects located 12.5 m away in an outdoor setting. However, these studies have not achieved detailed 3D measurements for discerning object shapes.

Prior research on outdoor acoustic monitoring has explored passive systems that utilize the sound emitted by vehicles during operation rather than the reflected sound. Acoustic traffic monitoring systems can be categorized into two types: rule-based and machine learning-based systems.

In a previous study [88], a microphone array arranged in a crisscross pattern was employed. The recorded audible sound was utilized to estimate parameters such as the number of passing vehicles, lane occupancy, average speed, and even vehicle types in multiple lanes. Machine learning-based acoustic traffic monitoring employs K-nearest neighbors and convolutional neural networks (CNN). In [89], audio was recorded and traffic was monitored using stereo microphones installed at the roadside. The power spectrogram and phase difference information of the audio were used as inputs to the CNN to estimate parameters such as the number of vehicles, speed, type, and occupancy.

6.3 Ultrasonic imaging

In this section, the ToF calculation using air-coupled ultrasound and the theory of 3D reconstruction from the ToF point cloud are explained. First, the ToF is elucidated, followed by a description of the calculation pipeline for the 3D reconstruction. The ToF represents the duration it takes for the ultrasonic waves emitted from the transducer to bounce off the object's surface and reach the microphone.

Fig. 6.1 provides an overview of 3D measurements using air-coupled ultrasound. Initially, ultrasonic waves are emitted from speaker array toward the object. Phase-shifted ultrasound waves overlap and amplify, resulting in directional focus. The emitted ultrasonic waves are reflected off the surface of the object. The reflected ultrasonic waves are measured by a microphone array with a phase difference. By calculating the delays and sums based on phase differences, sound waves from an arbitrary direction can be enhanced. Next, the cross-correlation between the beamformed reflected wave and the transmitted waves is calculated. The closer the waveform shapes are, the greater the cross-correlation; the ToF is the time from the timing of transmission to the peak of the cross-correlation. In environments with high interference, such as outdoor settings, ultrasonic cross-correlation tends to deteriorate. However, in this study, we enhanced interference resilience by employing DSSS modulation for ultrasonic signals. Further details will be discussed in Section 6.3.2.

Fig. 6.2 illustrates a block diagram of the proposed system utilized for ultrasonic measurements and 3D object reconstruction. A DSSS signal using an *M*-sequence as the transmission signal is emitted toward an object. In Stage (a) of the diagram, the signals reflected from the object surface are received by the microphone array and subjected to beamforming. In Stage (b), the beamformed reflected signal is normalized by the distance, and the cross-correlation of the transmitted signals is calculated, while removing the background to obtain a 3D ToF image. Consequently, the distance to an object can be measured. The ultrasonic data

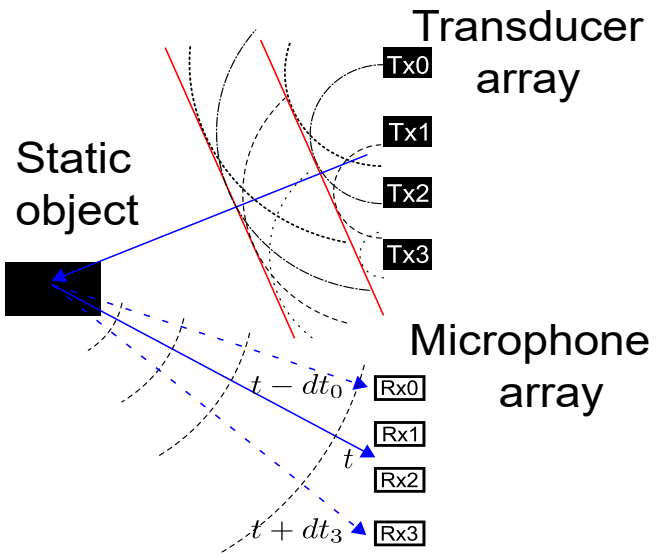


Fig. 6.1 3D measurement using beamforming and ToF[43]

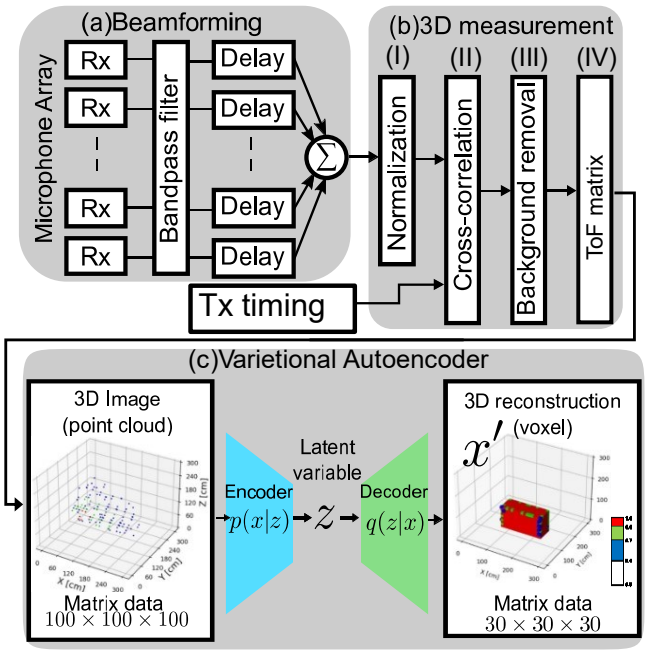


Fig. 6.2 Block diagram of proposed ultrasonic imaFigging system. [43]

processed by this ultrasonic signal processing system serve as the input for the VAE machine learning in Stage (c).

6.3.1 Beamforming

This subsection delineates the beamforming process depicted in Fig. 6.2(a). As depicted in Fig. 6.3, when the array is situated on the Y–Z plane, the main beam direction in 3D is denoted by the zenith angle θ around the X-axis and the azimuth angle φ relative to the Y–Z plane. In the nth element, the displacement in the y-axis direction from the origin is dy_n, and the displacement in the z-axis direction is dz_n. In beamforming, the delay time dt_n of the n-th element in the main beam direction (θ, φ) is given by (1), where c represents the propagation velocity of ultrasonic waves in air.

$$dt_n = (dy_n \, sin(\theta) \, cos(\varphi) + dz_n \, sin(\theta) \, sin(\varphi))/c. \tag{6.1}$$

Here, c represents the propagation velocity of ultrasonic waves in air.

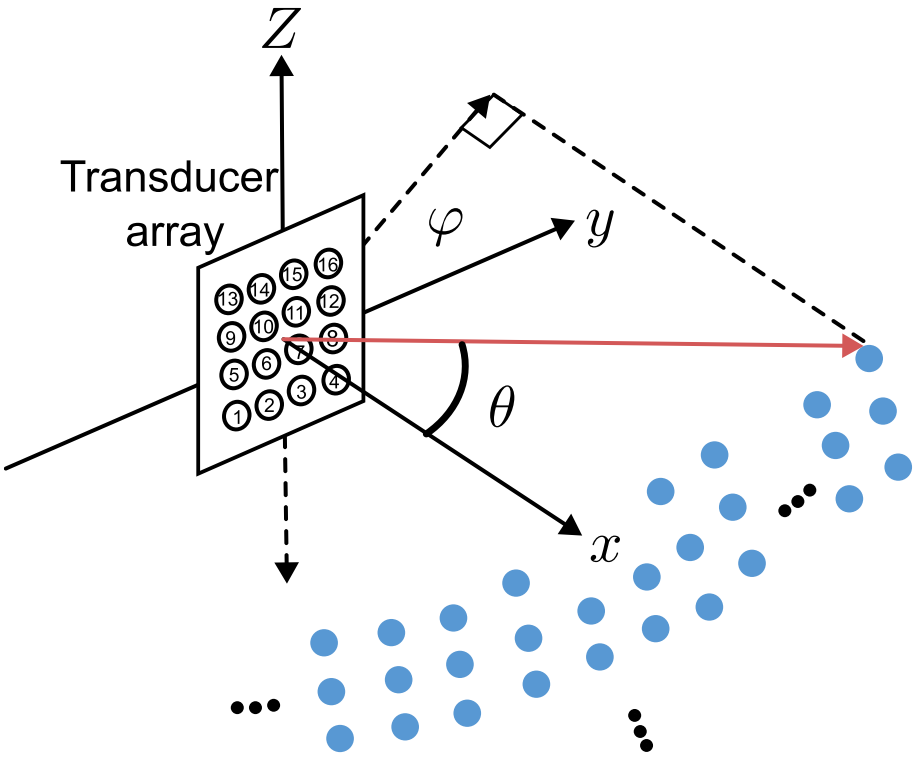


Fig. 6.3 Beamforming of a rectangular array [43]

6.3.2 3D Measurement

The 3D measurement illustrated in Fig. 6.2(b) comprises four substages: (I) normalization, (II) cross-correlation, (III) background removal, and (IV) ToF matrix. (I) Normalization mitigates the effect of distance attenuation. Propagation attenuation [90] is represented using a loudspeaker that uniformly radiates a spherical

wave. If the power at the receiver is P_r, the power at the transmitter is P_t. The distance from the transmitter to receiver is R. The power at the receiver is expressed as follows:

$$P_r = \frac{P_t}{4\pi R^2} \tag{6.2}$$

Based on this equation, the received signal can be normalized by R^2, signifying distance attenuation.

This normalization alleviates the effects of the distance attenuation on the intensity of the reflection from the object surface, thereby enhancing the detection accuracy of distant objects for the DSSD cross-correlation in Substage (II).

As depicted in Fig. 6.4, the cross-correlation between the received and transmitted signals is computed in (II). In this study, the transmitted signal was modulated by DSSS to enhance interference and noise immunity in the cross-correlation. Illustrated on the right side of Fig. 6.4, DSSS is a modulation technique where the transmitted data are multiplied by a spread signal with a higher frequency to broaden the bandwidth over a wider frequency band than the original signal, ensuring secure communication. Although this method consumes a larger bandwidth, it can improve the ability to reject interfering and noise waves. The XOR operation is performed between the M-Sequence and the transmitted data. The carrier wave remains non-inverted if each bit of the XOR result is 1 and is inverted if each bit is 0. In this instance, a data bit of one was modulated with an 8-bit M-sequence, with each M-sequence corresponding to a carrier wave of three wavelengths.

When an object is situated on a road, the signal reflected from the road often manifests as a strong interference wave. Additionally, the grating lobe of the ultrasonic array transmitter may introduce significant interference from directions other than that of the main beam. Consequently, to eliminate signals originating from sources other than the object itself, the received signal must undergo background removal (III), especially in scenarios where no object is present. Fig. 6.5 illustrates the distribution of signal strength when measuring reflected signals in front of the origin point. The signal strength of the background data exhibited non-

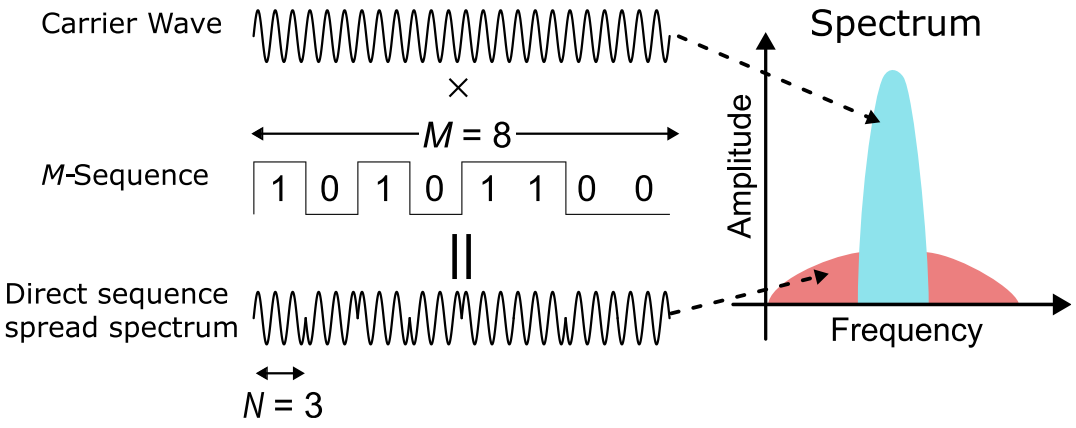


Fig. 6.4 Direct-sequence spread spectrum [43]

uniformity. Therefore, it is assumed that the signal magnitude follows a normal distribution, and the magnitude of the signal to be subtracted is thus determined accordingly.

The ToF data, from which background information has been removed, constitutes a point cloud expressed in a polar coordinate system denoted by (r, θ, ϕ). In this study, the input to the VAE is a 3D matrix. To input the ToF point cloud into the VAE, the point cloud is partitioned into a grid and transformed into a (100, 100, 100) Cartesian matrix represented by (x, y, z) with dimensions of (100, 100, 100).

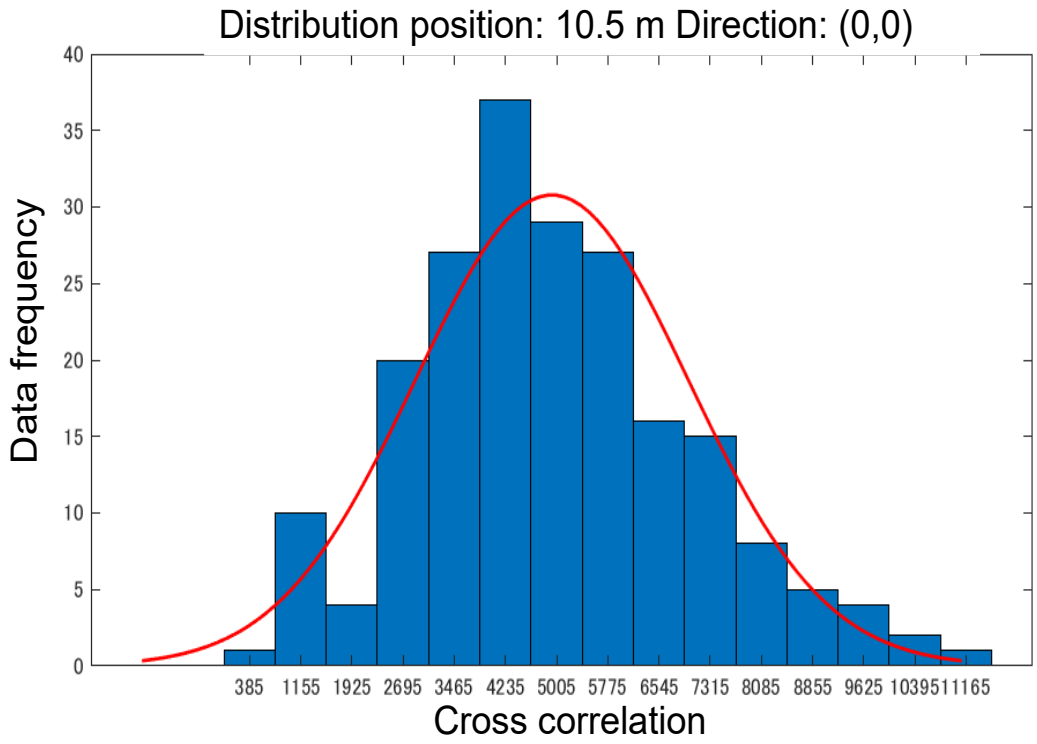


Fig. 6.5 Histogram of correlated [43]

6.3.3 Variational auto encoder

The 3D VAE encoder-decoder model utilized to obtain the 3D coordinates of a stationary target object is depicted in Fig. 6.6. A VAE functions as a type of autoencoder that compresses input voxels into a 500-dimensional latent variable space using an encoder, which are subsequently reconstructed as voxels in three dimensions by a decoder. a target object using ultrasonic data. The ultrasonic wavelengths range from ten to several millimeters. The input data dimensions are (100, 100, and 100) by segmenting the data into 30 mm intervals, resulting in a cube measuring 3000 mm per side. Each voxel represents a cube with 30 mm sides. These 3D data undergo convolution using a convolutional layer and a max-pooling layer to extract features in 3D space [91]. Subsequently, they are imaged in 3D space using a reverse convolutional layer.

For the training data utilized for supervision, the object is represented as (30, 30, 30) by dividing a cube of 3000 mm on one side into sections of 100 mm each. The positions where the target object is located are

assigned a value of 1.0, while the positions where the target object does not exist are filled with values of 0.1 in a voxel for use as training data.

Supervised learning is employed to eliminate noise, such as interference waves, and to perform object detection by imaging a target object using ultrasonic data. The ultrasonic wavelengths range from ten to several millimeters. The input data dimensions are (100, 100, and 100) by segmenting the data into 30 mm

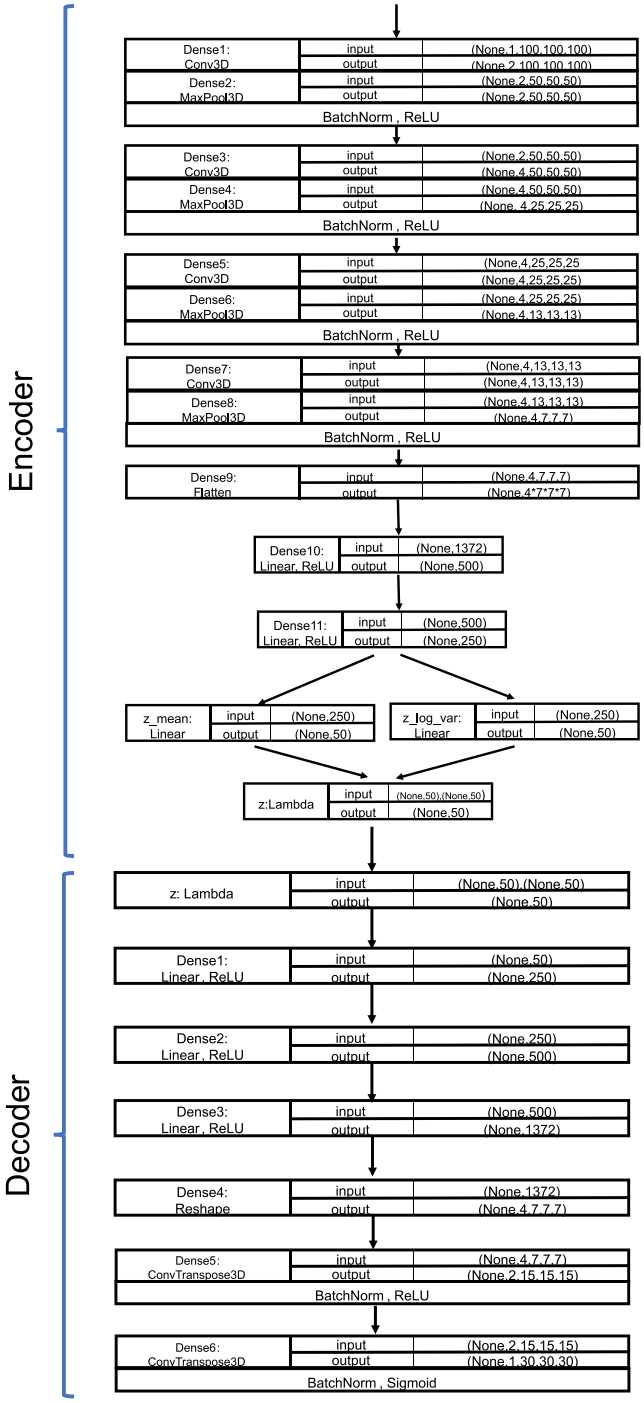


Fig. 6.6 Network architecture of 3D reconstructed VAE. [43]

intervals, resulting in a cube measuring 3000 mm per side. Each voxel represents a cube with 30 mm sides. These 3D data undergo convolution using a convolutional layer and a max-pooling layer to extract features in 3D space [91].. Subsequently, they are imaged in 3D space using a reverse convolutional layer.

For the training data utilized for supervision, the object is represented as (30, 30, 30) by dividing a cube of 3000 mm on one side into sections of 100 mm each. The positions where the target object is located are assigned a value of 1.0, while the positions where the target object does not exist are filled with values of 0.1 in a voxel for use as training data.

6.3.4 3D Evaluation

$$Precision = \frac{TP}{TP + FP} \tag{6.3}$$

$$Recall = \frac{TP}{TP + TN} \tag{6.4}$$

$$F = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{6.5}$$

Table 6.1 Confusion Matrix binary classification. [43]

		Predict: Positive	Predict: Negative
Reference	Positive	TP: True Positive	FN: False Negative
	Negative	FP: False Positive	TN: True Negative

In the 3D VAE machine learning framework, although the input has dimensions of (100 × 100 × 100), the output is represented as (30 × 30 × 30), where 1.0 signifies the ideal value for a specified object. We assessed the proximity of this region to the area where the object exists in the training data using $Precision$, $Recall$, and F-value (where the F-value is was recognized as the harmonic mean of the precision and recall). The precision, recall, and F-value are denoted in (6.3), (6.4), and (6.5), respectively, with explanations of the true positive (*TP*), false negative (*FN*), false positive (*FP*), and true negative (*TN*) presented in Table 6.1 to clarify their use in the equations.

6.4 Ultrasonic system

In this section, we outline the measurement system. Subsection A details the configuration of the proposed system, whereas Subsection B covers the experimental conditions.

6.4.1 Measurement system

Both hardware and software components were developed to implement the proposed measurement system. Fig. 6.7 illustrates the structure of the system, which consists of three main components. The first component is the ultrasonic array, which integrates microphones and speakers. The second component is a FPGA, which is responsible not only for processing signals but also for the DSSS modulation of ultrasonic waves, and it includes its own local storage. The third component is a recording server tasked with receiving and storing data.

Our measurement system utilized a PYNQ-Z2 [92]. The PYNQ-Z2 is equipped with a Xilinx ZYNQ system on a chip (SoC), which includes an internal ARM central processing unit (CPU) and a processing system comprised of peripheral circuits alongside programmable logic serving as an FPGA, facilitating CPU-to-FPGA communication. Additionally, the PYNQ-Z2 supports Linux, allowing for the utilization of TCP/IP and Linux file systems. We leveraged these features to achieve the streaming and logging of measurement data.

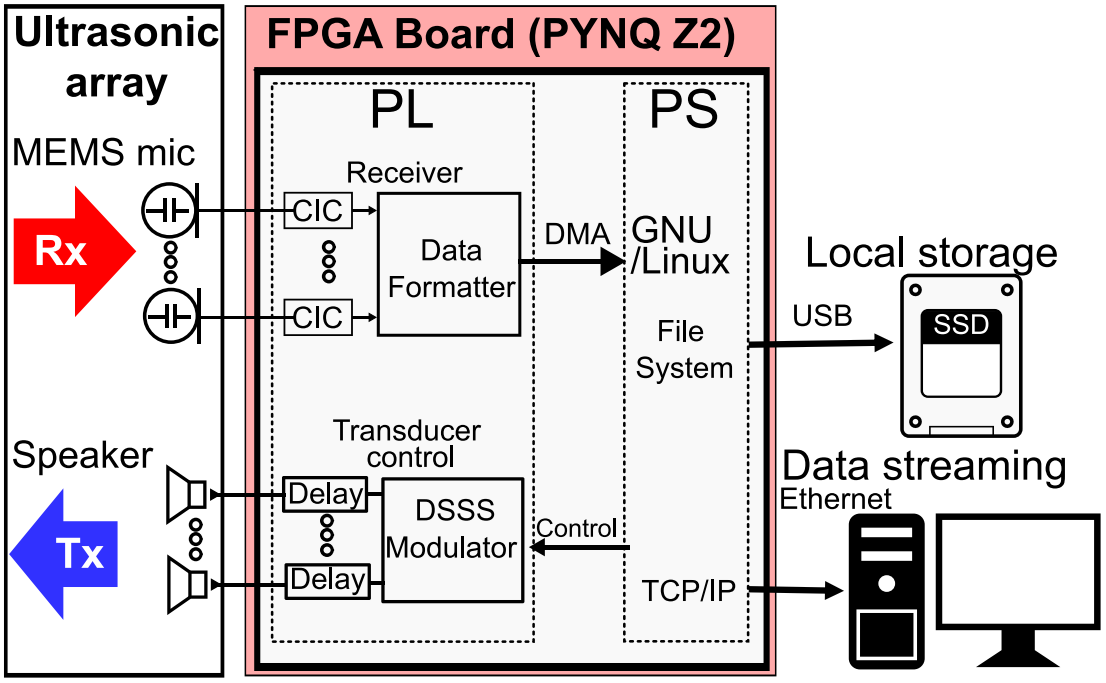


Fig. 6.7 Ultrasonic measurement system structure. [43]

The FPGA consists of two types of circuits: a speaker control circuit and a receiving circuit. The speaker control circuit generates an ultrasonic signal projected onto the object to be measured and drives the speaker. It operates by generating an ultrasonic signal projected onto the target object and controlling the speaker. The speaker functions based on DSSS modulation with parameters such as the *M*-sequence and frequency configured from the processing system. In the experiment, modulation was carried out with a carrier wave of 25 kHz, an *M*-sequence length of 8 bits, and 8 waves per bit. The receiving circuit demodulates the output of the 16-channel microphone array as a pulse code modulation (PCM) signal and transfers it to the PS. The ultrasonic microphone produces a 4.55 MHz Pulse-density modulation (PDM) output, which is demodulated into a 227 kHz PCM signal using a cascaded integrator comb (CIC) filter. Additional information is added to the 16-channel PCM signal and transferred to the PS via DMA.

Fig. 6.8 displays the appearance of the measurement system. Enclosed within a waterproof case designed for utility pole installation, the FPGA and Raspberry Pi serve as the data-streaming target, storage, and server, respectively. The system operates via Wi-Fi, with a Raspberry Pi acting as the gateway. The speakers and microphone array are housed within a dedicated 3D-printed enclosure and connected to the PYNQ-Z2 via flat cables. Enclosed within a waterproof case designed for utility pole installation, the FPGA and Raspberry Pi serve as the data-streaming target, storage, and server, respectively. The system operates via Wi-Fi, with a

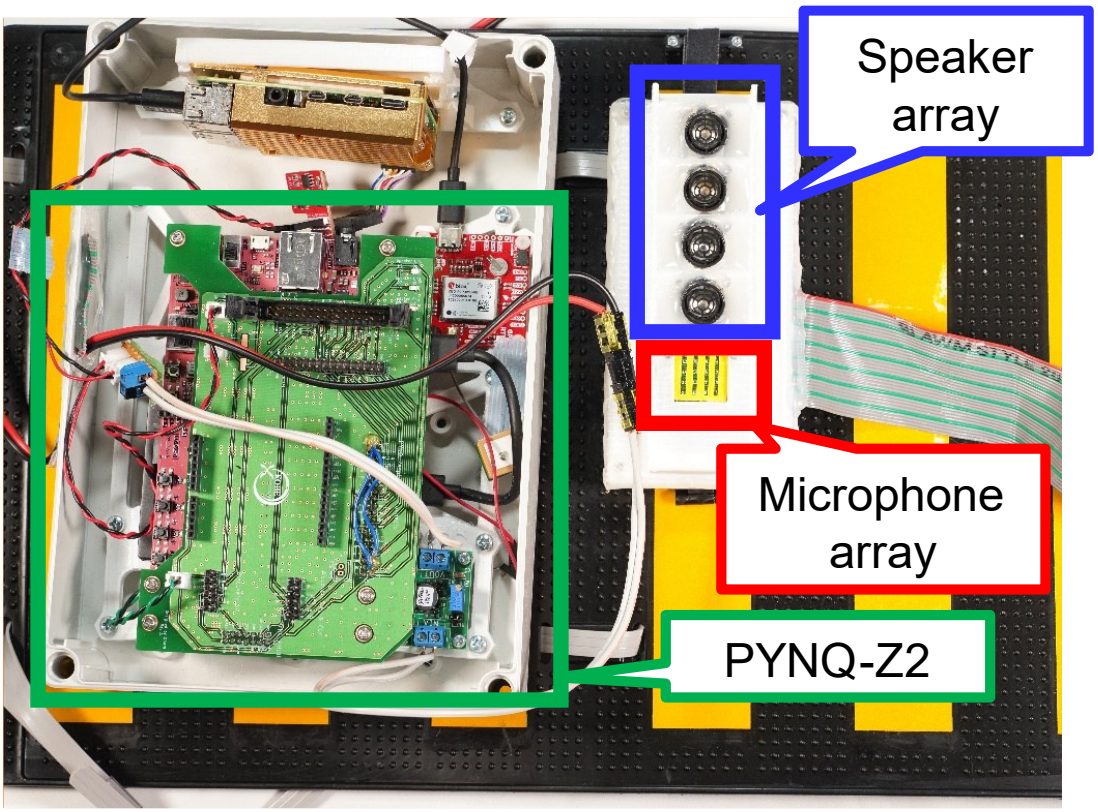


Fig. 6.8 . Ultrasonic array and FPGA. [43]

Raspberry Pi acting as the gateway. The speakers and microphone array are housed within a dedicated 3D-printed enclosure and connected to the PYNQ-Z2 via flat cables.

Fig. 6.9 (a) showcases the microphone array, whereas (b) exhibits the transducers. (a) The microphone array comprises 4 × 4 MEMS microphones arranged on a grid, spaced 5 mm apart, which is less than half the wavelength of the carrier wave. All microphones share a common clock source and produce a PDM signal with a maximum frequency of 4.55 MHz. (b) The speakers transmit modulated waves. Employing an H-bridge configuration, the circuit can drive microphones with an amplitude that is twice the supply voltage. Four speakers are integrated into the speaker array.

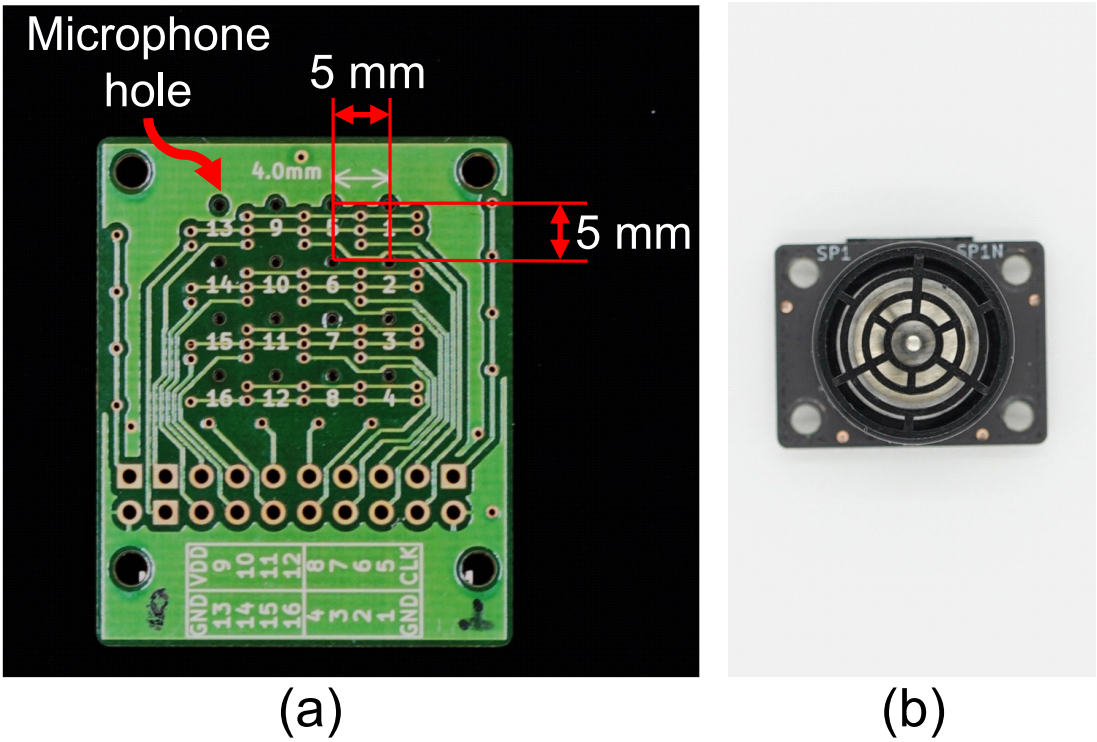


Fig. 6.9 Ultrasonic array device (a) 4x4 microphone array (b) speaker. [43]

6.4.2 Measurement environment

Fig. 6.10 illustrates the installation of the ultrasonic array during measurement. The study aims to monitor the road environment, with the measurement system installed on a utility pole by the roadside. The measurement area encompasses a 3000 mm cubic space positioned 10 m from the pole, where the objects of interest are located and measured. Mounted on a fixed arm 5.4 m above the pole, the ultrasonic array is designed to encircle the pole in a belt-like structure.

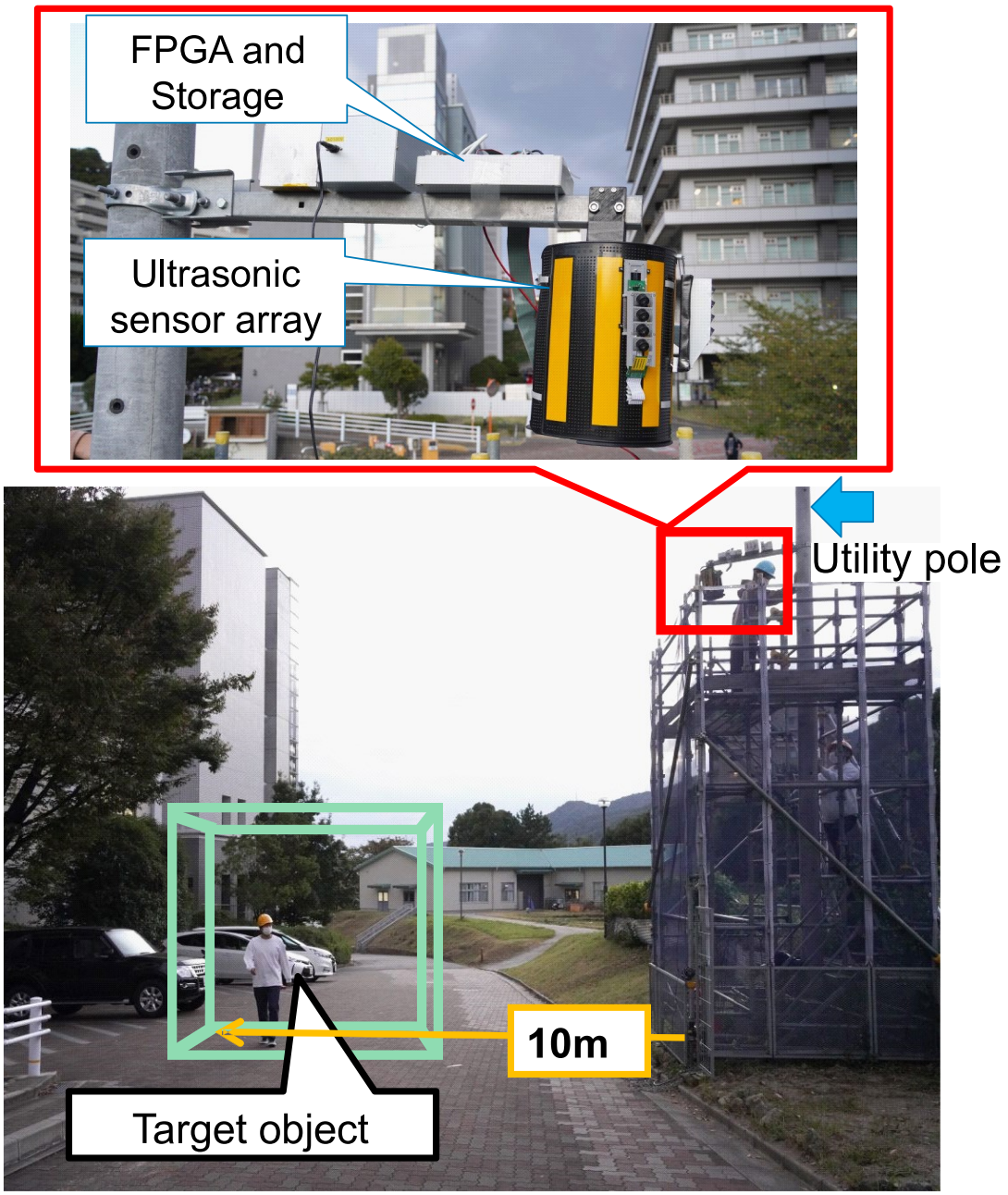


Fig. 6.10 Measurement environment and measurement system installation. [43]

6.5 Experimental result

Experimental results are presented in this section. Subsection A outlines the training data, detailing the data sampling method and the classes of objects intended for 3D reconstruction. Section B discusses the results of the 3D reconstruction.

6.5.1 Signal strength distribution and training data

The experimental environment for acquiring ultrasonic data in 3D space is depicted in Fig. 6.11. Following the placement of the sensor on the utility pole, beamforming, as described in Section 2, was employed to gather the data. During the experiment, the pole height was 5.4 m, and the radiation direction φ_{ld} was 53°. The target measur ement range extended 10 m from the sensor in the direction of distance. The elevation angle ranged from -24° to 24°, and similarly, the azimuth angle spanned from -24° to 24°. The area under consideration measured 3000 × 3000 × 3000 mm^3, delineated by a green cube in the figure.

Fig. 6.12 illustrates images depicting ultrasonic data and training data for a circular reflector, box, motorcycle, and person. These images were captured from the position of the sensor installed on the utility pole. As previously mentioned, the ultrasonic data formed a cube measuring 3000 mm on each side. Consequently, each voxel was subdivided every 30 mm to create dimensions of (100 × 100 × 100). The obtained ultrasonic data were normalized from 0 to 1.0 for display purposes: blue voxels indicate a strength between 0.6 and 0.7; green voxels represent values between 0.7 and 0.9; and red voxels signify values greater than 0.9. The training data were utilized to approximate a 3D object based on the size and shape of each object. The box, motorcycle, and human are approximated as cuboids, whereas the reflector is approximated as a sphere.

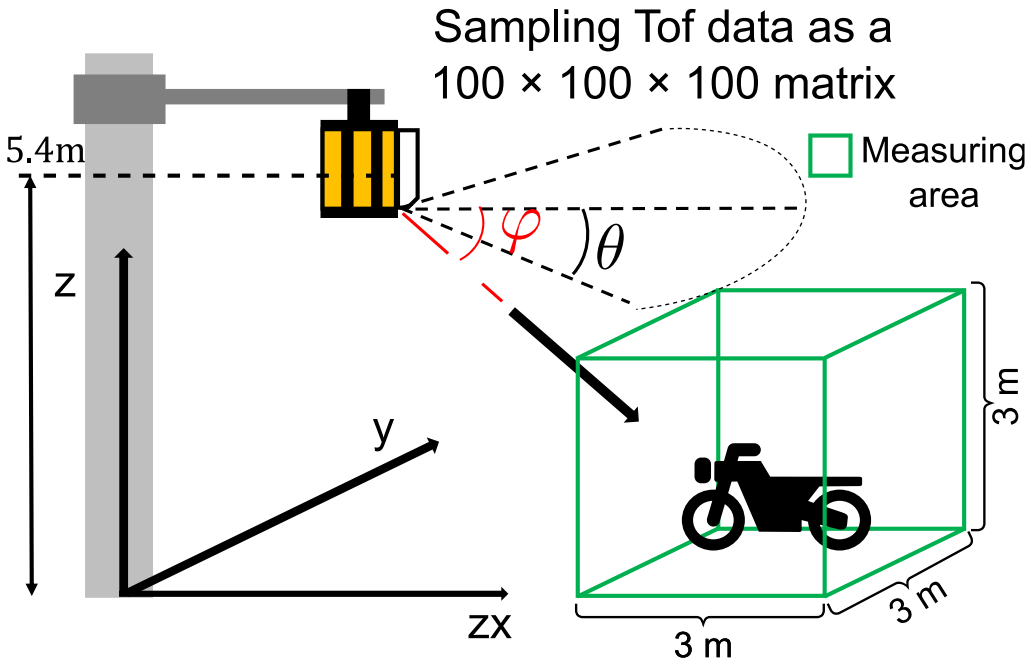


Fig. 6.11 Experiment environment. θ is azimuth angle, φ is elevation angle. [43]

Table 6.2 displays the sizes of the approximated solids up to 1,600 mm. Considering the challenge of imaging objects in 3D space within a few centimeters, we divided the object into 100 mm pieces and represented them as (30, 30, 30). A value of 1.0 was assigned to the position where the object existed, and voxel plots were generated in blue for visualization. A total of 1628 ultrasonic data points were measured, with 1304 used for training data and 324 for test data.

Table 6.2 Target object size.

	(a) Box	(b) Motorcycle	(c)Human	(d)Reflector
Length [mm] (Diameter)	700	1600	300	100
Width [mm]	300	600	300	

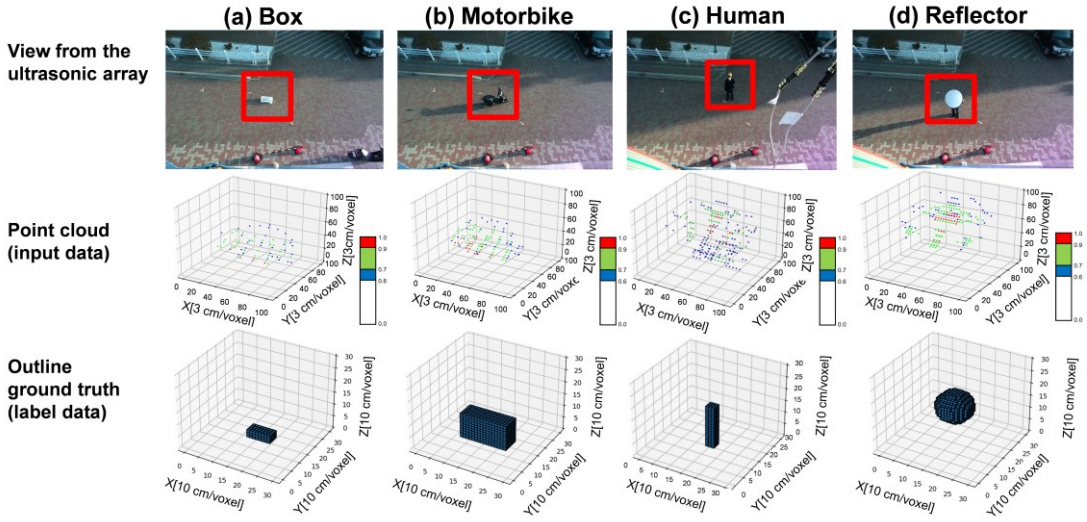


Fig. 6.12 3D static target dataset: appearances from ultrasonic array, point clouds, and ground truth. [43]

6.5.2 Evaluation results of 3D stationary objects

Fig. 6.13 showcases the input and output results achieved using the proposed VAE model for 3D stationary objects: (a) box, (b) motorcycle, (c) person, and (d) reflector at the center. The different plot colors denote different values: values between 0.4 and 0.7 are represented in blue, values between 0.7 and 0.9 are represented in green, and values greater than 0.9 are represented in red after data normalization. A comparison of (a) and (b) illustrates the difference in size between the box and motorcycle. Furthermore, comparing (a) and (b) with (c) and (d) reveals variations in size and shape, as identified and classified.

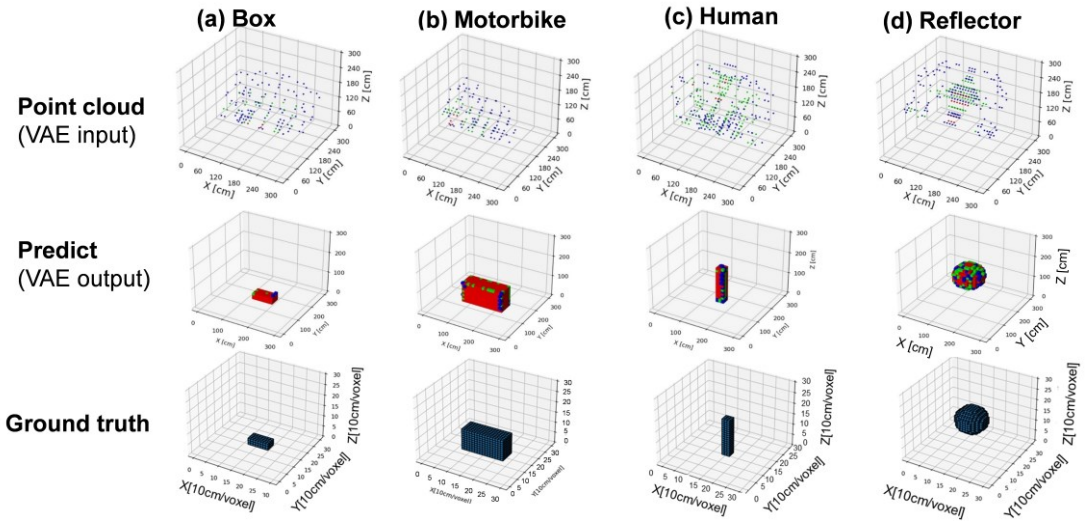


Fig. 6.13 3D reconstruction results and ground truth for each class. . [43]

Table 6.3 presents the results of the precision, recall, and F-values obtained when the threshold value was set to 0.5. For evaluation, a value above 0.5 was regarded as 1. Values below 0.5 were taken as 0. The overall accuracies for precision, recall, and F-value were 0.939, 0.868, and 0.902, respectively.

This paper targets the reconstruction of small objects ranging from a few hundred millimeters to 1,600 mm. Reconstruction would likely be even more feasible for larger measurement targets such as cars and trucks. As in Substage (III) of our method's pipeline, the background signals are were removed to intensify the signal reflected from the target object. Consequently, for larger objects such as cars and trucks, the difference from the background would become more significant.

Table 6.3 Reconstruction Result (3D stationary target) [43]

	(a) Reflector	(b) Box	(c) Motorcycle	(d) Human	Average
Precision	0.935	0.957	0.959	0.911	0.939
Recall	0.835	0.957	0.950	0.825	0.868
F value	0.880	0.957	0.954	0.866	0.902

6.6 Discussion

The feasibility of implementing the VAE model proposed in this study on deep-learning processors introduced in Chapters 3 and 4 is discussed. In this study, a VAE was used for 3D reconstruction, with ToF point clouds as the input. VAE is a type of autoencoder similar to the U-Net and SegNet models introduced in Chapter 3. Table 6.4 shows a comparison of the number of MAC operations and parameter sizes between a VAE and the models introduced in Chapter 3. These comparisons show that our 3D reconstruction model

performs 100 times fewer operations and has a smaller activation size than the model used in the processor performance evaluations. The proposed VAE is an autoencoder that integrates an encoder and a decoder. Unlike models such as SegNet with unpooling layers or U-Net with activation concatenation, the proposed VAE lacks shortcut connections between the decoder and encoder. Therefore, the memory access load is also considered small. Based on the above analysis, the implementation of edge devices is expected to be feasible.

In this study, ultrasonic-only measurements were conducted; however, if privacy is not a concern, it is conceivable to further extend the system by incorporating an optical camera. In previous research, a method combining sparse LiDAR data with a monocular optical image has been proposed [86]. Building on this study, integrating our approach with an optical camera could enable a more robust system that accommodates both optical and acoustic environments.

Table 6.4 Comparison of FLOPs and Parameter Sizes for SegNet, U-Net, and the Proposed 3D Reconstruction VAE

Model	FLOPs	Parameter size
SegNet	31311630336	167812669
U-Net	41929975808	31037698
3D Reconstruction	101009010	1663927

6.7 Conclusion

position, size, and shape of 3D stationary objects. Across the four object types examined, the average results demonstrate a precision of 0.939, recall of 0.868, and F-value of 0.902. It is worth noting that the experiment focused solely on stationary objects.

For a comprehensive assessment of road conditions, accounting for the Doppler effect becomes imperative, particularly for measuring moving vehicles and other mobile objects. Moreover, the measurement system employed in this study isn't limited to ultrasonic waves; it can also capture audible sounds. By incorporating audible sounds emitted by vehicles and pedestrians, akin to previous studies, we can achieve a more nuanced understanding of road conditions.

Chapter 7 Conclusion

This study discusses various technologies for improving the performance of monitoring systems. After summarizing the contents of each chapter, an overall conclusion is drawn.

Chapter 3 Performance Improvement of Deep Learning Processors Through On-Chip Memory Replacement with MRAM

In this study, we employed NVIDIA's open-source deep-learning processor NVDLA as a motif processor to estimate the performance improvements achieved by replacing on-chip memory, such as cache memory and convolution buffers, with area-efficient MRAM. SegNet and U-Net, which are models used for segmentation, were selected as the target models, and the parameters trained using the Cityscapes dataset for autonomous driving were utilized. The NVDLA hardware simulator FireSim-NDLA and software simulator VP-NVDLA running on AWS were used as the simulation environments. From these, we collected memory access logs and analyzed the behavior of various deep learning operators. Based on this data, we developed and evaluated the performance of three proprietary simulators: a cache simulator, a memory access simulator, and an area simulator. The results showed that SegNet achieved a 10.1% reduction in power consumption and an 11.4% speed improvement when using STT-MRAM. Furthermore, U-Net demonstrated up to an 18.6% reduction in power consumption and a 17.9% speed improvement.

Chapter 4 1W8R 20T SRAM Codebook for 20% Energy Reduction in Mixed-Precision Deep-Learning Inference Processor System

In this research, we used codebook SRAM to optimize the main memory bandwidth of a deep learning processor and estimated its power reduction effect. Quantization is an effective approach for optimizing the main-memory bandwidth in deep learning, and codebook compression, a type of vector quantization, can be implemented using lookup tables. Lookup tables achieve a high area density when realized using a multiport SRAM.

In the design process, memory cells were designed, peripheral circuits were connected to form memory macros, and the memory macros were manufactured using UJSC's 40 nm process. In the memory cell design, two read ports were configured with drain-connected channel channels and six ports with gate-connected channels. This configuration achieved a 28% area reduction compared with a design in which all readout ports used N-channel gate connections, while minimizing the degradation of the SN margin.

The evaluation was conducted through simulations, silicon validation, and NVDLA integration. During the simulation evaluation, 16 K Monte Carlo simulations were performed to evaluate the operating voltage and access success rate for both Pch drain and Nch gate connections. The results confirmed that a stable read/write operation is achievable for all configurations at 1.1 V.

For the silicon validation, the power consumption and access speed were evaluated using a semiconductor tester for the fabricated silicon chips, revealing an access speed of approximately 3 ns. Furthermore, by incorporating data from the silicon evaluation into the simulator described in Chapter 3, the performance of the proposed SRAM was evaluated when it was integrated into the NVDLA. The performance simulation for U-Net demonstrated a 20.24% reduction in the power consumption and 26.24% improvement in the area efficiency.

Chapter 5
High-Accuracy and Low-Power Synchronization Technique Using Bluetooth

In this study, we developed a time-synchronization system for multi-node measurements. This system was implemented on BLE, and it combined both high accuracy and low-power performance. We developed two techniques to achieve low power consumption and high accuracy in time synchronization: timer hybridization and communication-scheduling optimization. The HFCLK, which has high power consumption but offers high time resolution, and the RTC timer, which consumes less power but has a lower time resolution, were combined in a microcontroller to achieve low power consumption and high-precision time measurements.

Communication scheduling minimizes the communication required by the BLE. Wireless communication is a major power-consumption factor for microcontrollers equipped with a BLE. By leveraging the regular interval requirement of BLE communication, unnecessary communication can be eliminated by turning off the HFCLK and its PLL during idle periods and activating the timer just before communication.

Based on these techniques, we developed two types of scheduling: high-accuracy and low-power. High-accuracy scheduling achieves a synchronization error of 0.272 μs with a power consumption of 252.1 μW, while low-power scheduling achieves a synchronization error of 0.990 μs with a power consumption of 141.8 μW.

Chapter 6 3-D Object Reconstruction from Outdoor Ultrasonic Image and a Variation Autoencoder

Chapter 6 introduces an environmental monitoring system that combines ultrasonic waves and deep learning. During disasters, monitoring traffic conditions using sensors installed in the environment can enable swift responses, such as proposing suitable evacuation routes and dispatching rescue teams to isolated areas.

The installation of sensors, particularly optical cameras, in residential areas is challenging because of privacy concerns. In such cases, LiDAR and millimeter-wave radar offer potential solutions; however, their high costs hinder their widespread implementation. One effective solution to this problem is 3D sensing using ultrasonic waves. In this study, a 3D measurement system was developed using an ultrasonic array. The system utilizes an FPGA called PYNQ, which not only modulates and transmits ultrasonic waves, but also continuously captures 16-channel sound wave data at a high resolution of 224 kHz.

A microphone array, which is a sensor device consisting of multiple microphones arranged at regular intervals, served as the core of the ultrasonic array system. Ultrasonic waves were projected onto a measurement target, and the reflected waves were measured using a microphone array to determine their phase

differences. Because these phase differences vary depending on the arrival angle of the sound waves, it is possible to emphasize the sound waves from a specific direction by adding signals with a time difference based on the phase differences. This process is known as beam forming.

Furthermore, the projected sound waves were direct sequence spread spectrum (DSSS) modulated, and the distance was measured by calculating the cross-correlation with the transmitted wave. Based on these data, a 3D ToF image was constructed and input to a VAE to reconstruct the outline of the object in three dimensions.

In the experiment, a 3 m3 of space located 10 m away from a sensor installed on a telephone pole was reconstructed with 30 × 30 × 30 voxels. Four types of objects were examined: boxes, motorcycles, humans, and circular reflectors. The results showed that the system successfully achieved a 3D reconstruction with an average accuracy of 0.939.

This dissertation explored four research themes. The first two themes focused on acceleration technologies for deep learning models, aimed at optimizing power efficiency in monitoring systems. Subsequently, techniques for time synchronization that enable multi-node measurements were presented. Furthermore, a road monitoring system utilizing ultrasonic waves was introduced, along with its application technologies.

Acknowledgements

First and foremost, I wish to express my sincere gratitude to Professor Hiroshi Kawaguchi of the Graduate School of Science, Technology, and Innovation at Kobe University. His constant support, meticulous guidance, and unwavering patience have been instrumental in shaping both this dissertation and my overall research experience.

I am equally indebted to Associate Professor Shintaro Izumi, also of the Graduate School of Science, Technology, and Innovation at Kobe University, for his technical insights and dedicated mentorship. His advice has helped me cultivate a broad range of skills essential for this work.

I would like to extend my profound appreciation to the members of my dissertation committee—Professor Seiichiro L. Ten-no, Professor Chikara Ohta, and Professor Tetsuya Takiguchi of the Graduate School of System Informatics at Kobe University—for their invaluable comments and constructive feedback during the Preliminary Review of this dissertation.

I am deeply grateful to my advisors, Dr. Toru Ishii and Dr. Tetsuo Yamamura, for their insightful suggestions based on extensive professional experience. Their perspectives significantly enriched the quality of my research.

I also acknowledge the generous contributions of my collaborators. Mr. Masaya Kabuto played a pivotal role in the LSI design process, while Mr. Masakazu Taichi and Mr. Atsushi Fukunaga offered critical support in developing the simulation environment. I am further indebted to Mr. Shun Sato, Mr. Shahrul Amir Bin Kamarulzaman, Mr. Riku Hamabe, and Mr. Yuto Yasuda for their outstanding efforts in building the outdoor ultrasonic measurement system. My gratitude extends to Mr. Shinya Imanaka and Mr. Masayasu Harada for their foundational work on time synchronization technology. I am also profoundly thankful to Mr. Takayuki Genda and Mr. Masato Ikegawa for their invaluable support in constructing deep learning models.

I am thankful to Tokyo Electric Power Company for providing practical advice from an industrial standpoint and for offering the experimental setup necessary to conduct outdoor road monitoring trials.

This research received financial support from multiple sources. The work presented in Chapter 5 was funded by the Japan Science and Technology Agency (JST) under Grant Number JPMJPF2115, and the research described in Chapter 6 was partially supported by the New Energy and Industrial Technology Development Organization (NEDO) under Grant Number P19005. This work was also supported through the activities of VDEC, The University of Tokyo, in collaboration with Cadence Design Systems.

I wish to express my appreciation to my colleagues in the Architecture Laboratory at Kobe University for their camaraderie and collaborative spirit throughout my doctoral journey.

My heartfelt thanks go to Ms. Go Emi, Ms. Fumika Kanamaru, Ms. Yui Emori, and Ms. Tsugumi Ohashi, whose support and encouragement were a source of strength during the challenging phases of my research.

Lastly, I owe a special debt of gratitude to my parents. Their steadfast belief in my ambitions and their unfailing support have been the driving force behind my academic pursuits.

I extend my deepest appreciation to everyone who contributed, both directly and indirectly, to the completion of this dissertation.

Portions of the figures and tables used in Chapter 3 of this dissertation have been reused or adapted from the following publications by the Information Processing Society of Japan (IPSJ):

1. R. Ohara, A. Fukunaga, M. Taichi, M. Kabuto, R. Hamabe, M. Ikegawa, S. Izumi, and H. Kawaguchi, "A Case Study for Improving Performances of Deep-Learning Processor with MRAM," *IPSJ Transactions on System and LSI Design Methodology*, vol. 17, pp. 7-15, 2024, doi: 10.2197/ipsjtsldm.17.7.

Portions of the figures and tables used in Chapter 4 of this dissertation have been reused or adapted from the following publication with permission from IEEE:

R. Ohara, M. Kabuto, T. Masakazu, A. Fukunaga, Y. Yasuda, and R. Hamabe, "A 1W8R 20T SRAM Codebook for 20% Energy Reduction in Mixed-Precision Deep-Learning Inference Processor System," 2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS), Hangzhou, China, 2023, pp. 1-5, doi: 10.1109/AICAS57966.2023.10168555.

In accordance with IEEE copyright policy, the copyright notice has been prominently displayed with all reused or adapted figures and tables:

© 2023 IEEE. Reprinted, with permission, from R. Ohara et al., "A 1W8R 20T SRAM Codebook for 20% Energy Reduction in Mixed-Precision Deep-Learning Inference Processor System," 2023 IEEE 5th International Conference on Artificial Intelligence Circuits and Systems (AICAS), Hangzhou, China, 2023, pp. 1-5, doi: 10.1109/AICAS57966.2023.10168555.

Portions of the figures and tables used in Chapter 5 and Chapter 6 of this dissertation have been reused or adapted from the following IEEE publications:

1. R. Ohara, S. Izumi, S. Imanaka, T. Yamamura, I. Toru, and H. Kawaguchi, “MicroSync: Sub-Micro Second Accuracy Wireless Time Synchronization Service,” IEEE Access, vol. 12, pp. 124478–124494, 2024, doi: 10.1109/ACCESS.2024.3446668.
2. R. Ohara, Y. Yasuda, R. Hamabe, S. Sato, T. Ishii, S. Izumi, and H. Kawaguchi, "3-D Object Reconstruction From Outdoor Ultrasonic Image and Variation Autoencoder," IEEE Open Journal of Ultrasonics, Ferroelectrics, and Frequency Control, vol. 4, pp. 140-149, 2024, doi: 10.1109/OJUFFC.2024.3466090.

Reference

[1] X. Xu *et al.*, "Scaling for edge inference of deep neural networks," *Nat Electron*, vol. 1, no. 4, pp. 216–222, Apr. 2018, doi: 10.1038/s41928-018-0059-3.

[2] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "DSSD : Deconvolutional Single Shot Detector," Jan. 2017.

[3] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, IEEE, Jun. 2018, pp. 4510–4520. doi: 10.1109/CVPR.2018.00474.

[4] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors," *arXiv preprint arXiv:2207.02696*, 2022.

[5] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 2018. [Online]. Available: http://arxiv.org/abs/1804.02767

[6] S. Han *et al.*, "EIE," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 243–254, Oct. 2016, doi: 10.1145/3007787.3001163.

[7] S. Tsuji, F. Yamada, H. Kawaguchi, A. Inoue, and Y. Sakai, "Greedy search algorithm for partial quantization of convolutional neural networks inspired by submodular optimization," *Neural Comput Appl*, Jan. 2022, doi: 10.1007/s00521-021-06752-7.

[8] S. Han, H. Mao, and W. J. Dally, "Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding," Oct. 2015.

[9] R. Kawamoto *et al.*, "A 1.15-TOPS 6.57-TOPS/W Neural Network Processor for Multi-Scale Object Detection With Reduced Convolutional Operations," *IEEE J Sel Top Signal Process*, vol. 14, no. 4, pp. 634–645, May 2020, doi: 10.1109/JSTSP.2020.2966331.

[10] S. A. Wolf *et al.*, "Spintronics: A Spin-Based Electronics Vision for the Future," *Science (1979)*, vol. 294, no. 5546, pp. 1488–1495, Nov. 2001, doi: 10.1126/science.1065389.

[11] "The NVIDIA Deep Learning Accelerator (NVDLA).," http://nvdla.org.

[12] K. Mishty and M. Sadi, "System and Design Technology Co-optimization of SOT-MRAM for High-Performance AI Accelerator Memory System," Mar. 2023.

[13] F. Farshchi, Q. Huang, and H. Yun, "Integrating NVIDIA Deep Learning Accelerator (NVDLA) with RISC-V SoC on FireSim," in *2019 2nd Workshop on Energy Efficient Machine Learning and Cognitive Computing for Embedded Applications (EMC2)*,

IEEE, Feb. 2019, pp. 21–25. doi: 10.1109/EMC249363.2019.00012.

[14] CSL-KU, "FireSim-NVDLA: NVDLA Integrated with Rocket Chip SoC on FireSim.," https://github.com/CSL-KU/firesim-nvdla.

[15] S. Karandikar *et al.*, "FireSim: FPGA-Accelerated Cycle-Exact Scale-Out System Simulation in the Public Cloud," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, IEEE, Jun. 2018, pp. 29–42. doi: 10.1109/ISCA.2018.00014.

[16] https://github.com/nvdla/vp, "NVDLA Open Source Virtual Platform."

[17] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *IEEE Trans Pattern Anal Mach Intell*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017, doi: 10.1109/TPAMI.2016.2644615.

[18] P. and B. T. Ronneberger Olaf and Fischer, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, J. and W. W. M. and F. A. F. Navab Nassir and Hornegger, Ed., Cham: Springer International Publishing, 2015, pp. 234–241.

[19] N. Perrissin *et al.*, "Spin transfer torque magnetic random-access memory: Towards sub-10 nm devices," in *2018 International Conference on IC Design & Technology (ICICDT)*, IEEE, Jun. 2018, pp. 125–128. doi: 10.1109/ICICDT.2018.8399772.

[20] F. Oboril, R. Bishnoi, M. Ebrahimi, and M. B. Tahoori, "Evaluation of Hybrid Memory Technologies Using SOT-MRAM for On-Chip Cache Hierarchy," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 3, pp. 367–380, Mar. 2015, doi: 10.1109/TCAD.2015.2391254.

[21] M. Cordts *et al.*, "The Cityscapes Dataset for Semantic Urban Scene Understanding," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3213–3223, 2016, [Online]. Available: https://api.semanticscholar.org/CorpusID:502946

[22] D. Lee, S. K. Gupta, and K. Roy, "High-performance low-energy STT MRAM based on balanced write scheme," in *Proceedings of the 2012 ACM/IEEE international symposium on Low power electronics and design*, New York, NY, USA: ACM, Jul. 2012, pp. 9–14. doi: 10.1145/2333660.2333665.

[23] R. Dreslinski, M. Wieckowski, D. S. Blaauw, and T. Mudge, "Near threshold computing: Overcoming performance degradation from aggressive voltage scaling," in *Proc. Workshop Energy-Efficient Design*, 2009, pp. 44–49.

[24] G. Jha and E. John, "Performance Analysis of Single-Precision Floating-Point MAC for Deep Learning," in *2018 IEEE 61st International Midwest Symposium on Circuits*

and Systems (MWSCAS), IEEE, Aug. 2018, pp. 885–888. doi: 10.1109/MWSCAS.2018.8624019.

[25] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," Dec. 2019.

[26] S. Zhang *et al.*, "Cambricon-X: An accelerator for sparse neural networks," in *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, IEEE, Oct. 2016, pp. 1–12. doi: 10.1109/MICRO.2016.7783723.

[27] M. Gao, J. Pu, X. Yang, M. Horowitz, and C. Kozyrakis, "TETRIS," *ACM SIGARCH Computer Architecture News*, vol. 45, no. 1, pp. 751–764, May 2017, doi: 10.1145/3093337.3037702.

[28] S.-F. Hsiao and P.-C. Wu, "Design of low-leakage multi-port SRAM for register file in graphics processing unit," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, IEEE, Jun. 2014, pp. 2181–2184. doi: 10.1109/ISCAS.2014.6865601.

[29] H. Nguyen, J. Jeong, F. Atallah, D. Yingling, and K. Bowman, "A 7-nm 6R6W Register File With Double-Pumped Read and Write Operations for High-Bandwidth Memory in Machine Learning and CPU Processors," *IEEE Solid State Circuits Lett*, vol. 1, no. 12, pp. 225–228, Dec. 2018, doi: 10.1109/LSSC.2019.2911885.

[30] D.-P. Wang, H.-J. Lin, C.-T. Chuang, and W. Hwang, "Low-Power Multiport SRAM With Cross-Point Write Word-Lines, Shared Write Bit-Lines, and Shared Write Row-Access Transistors," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 61, no. 3, pp. 188–192, Mar. 2014, doi: 10.1109/TCSII.2013.2296137.

[31] Hiroki Noguchi *et al.*, "Which is the best dual-port SRAM in 45-nm process technology? — 8T, 10T single end, and 10T differential —," in *2008 IEEE International Conference on Integrated Circuit Design and Technology and Tutorial*, IEEE, Jun. 2008, pp. 55–58. doi: 10.1109/ICICDT.2008.4567245.

[32] S. Ataei, M. Gaalswyk, and J. E. Stine, "A high performance multi-port SRAM for low voltage shared memory systems in 32 nm CMOS," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, IEEE, Aug. 2017, pp. 1236–1239. doi: 10.1109/MWSCAS.2017.8053153.

[33] J. Chang, R. Koppanathi, and L. G. Johnson, "Novel designs of 8-port memory cell," in *2010 53rd IEEE International Midwest Symposium on Circuits and Systems*, IEEE, Aug. 2010, pp. 829–832. doi: 10.1109/MWSCAS.2010.5548682.

[34] Y. Ishii *et al.*, "A 5.92-Mb/mm ^{2} 28-nm pseudo 2-read/write dual-port SRAM using double pumping circuitry," in *2016 IEEE Asian Solid-State Circuits Conference (A-SSCC)*, IEEE, Nov. 2016, pp. 17–20. doi: 10.1109/ASSCC.2016.7844124.

[35] A. Teman, P. Meinerzhagen, A. Burg, and A. Fish, “Review and classification of gain cell eDRAM implementations,” in *2012 IEEE 27th Convention of Electrical and Electronics Engineers in Israel*, IEEE, Nov. 2012, pp. 1–5. doi: 10.1109/EEEI.2012.6377022.

[36] E. Amat, R. Canal, and A. Rubio, “Modem Gain-Cell Memories in Advanced Technologies,” in *2018 IEEE 24th International Symposium on On-Line Testing And Robust System Design (IOLTS)*, IEEE, Jul. 2018, pp. 65–68. doi: 10.1109/IOLTS.2018.8474151.

[37] H. Nakahara, Z. Que, and W. Luk, “High-Throughput Convolutional Neural Network on an FPGA by Customized JPEG Compression,” in *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, Fayetteville, AR, USA: IEEE, May 2020, pp. 1–9. doi: 10.1109/FCCM48280.2020.00010.

[38] M. K. T. M. A. F. Y. Y. R. H. Ryotaro Ohara, “1W8R 20T SRAM Codebook for 20% Energy Reduction in Mixed-Precision Deep-Learning Inference Processor System (Under Review),” *1W8R 20T SRAM Codebook for 20¥% Energy Reduction in Mixed-Precision Deep-Learning Inference Processor System*, Oct. 2025.

[39] M. Imani, M. Samragh Razlighi, Y. Kim, S. Gupta, F. Koushanfar, and T. Rosing, “Deep Learning Acceleration with Neuron-to-Memory Transformation,” in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, IEEE, Feb. 2020, pp. 1–14. doi: 10.1109/HPCA47549.2020.00011.

[40] R. Ohara *et al.*, “A Case Study for Improving Performances of Deep-Learning Processor with MRAM,” *IPSJ Transactions on System and LSI Design Methodology*, vol. 17, no. 0, pp. 7–15, 2024, doi: 10.2197/ipsjtsldm.17.7.

[41] T. Polonelli *et al.*, “A Self-Sustainable and Micro-Second Time Synchronized Multi-Node Wireless System for Aerodynamic Monitoring on Wind Turbines,” *IEEE Access*, vol. 11, pp. 119506–119522, 2023, doi: 10.1109/ACCESS.2023.3327422.

[42] I. Toru, Y. Yasuda, S. Sato, S. Izumi, and H. Kawaguchi, “Millimeter-Precision Ultrasonic DSSS Positioning Technique With Geometric Triangle Constraint,” *IEEE Sens J*, vol. 22, no. 16, pp. 16202–16211, Aug. 2022, doi: 10.1109/JSEN.2022.3188007.

[43] R. Ohara *et al.*, “3-D Object Reconstruction From Outdoor Ultrasonic Image and Variation Autoencoder,” *IEEE Open Journal of Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 4, pp. 140–149, 2024, doi: 10.1109/OJUFFC.2024.3466090.

[44] S. Sato *et al.*, “Estimating Person Location in Bathroom with Spatial Ultrasound and Variational autoencoder,” in *2023 IEEE International Ultrasonics Symposium (IUS)*, IEEE, Sep. 2023, pp. 1–4. doi: 10.1109/IUS51837.2023.10306421.

[45] R. Heydon, *Bluetooth Low Energy : the Developer's Handbook*. 2012.

[46] L. Da Xu, W. He, and S. Li, "Internet of Things in Industries: A Survey," *IEEE Trans Industr Inform*, vol. 10, no. 4, pp. 2233–2243, Nov. 2014, doi: 10.1109/TII.2014.2300753.

[47] F. J. Dian, A. Yousefi, and K. Somaratne, "Performance evaluation of time synchronization using current consumption pattern of BLE devices," in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, IEEE, Jan. 2018, pp. 906–910. doi: 10.1109/CCWC.2018.8301666.

[48] F. Asgarian and K. Najafi, "BlueSync: Time Synchronization in Bluetooth Low Energy With Energy-Efficient Calculations," *IEEE Internet Things J*, vol. 9, no. 11, pp. 8633–8645, Jun. 2022, doi: 10.1109/JIOT.2021.3116921.

[49] K. Somaratne, F. J. Dian, and A. Yousefi, "Accuracy analysis of time synchronization using current consumption pattern of BLE devices," in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, IEEE, Jan. 2018, pp. 841–844. doi: 10.1109/CCWC.2018.8301624.

[50] A. Yousefi, K. Somaratne, and F. J. Dian, "Analysis of time synchronization based on current measurement for Bluetooth Low Energy (BLE)," in *2017 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, IEEE, Oct. 2017, pp. 602–607. doi: 10.1109/IEMCON.2017.8117157.

[51] S. Sridhar, P. Misra, G. S. Gill, and J. Warrior, "Cheepsync: a time synchronization service for resource constrained bluetooth le advertisers," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 136–143, Jan. 2016, doi: 10.1109/MCOM.2016.7378439.

[52] C. C. Rheinlander and N. Wehn, "Precise synchronization time stamp generation for Bluetooth low energy," in *2016 IEEE SENSORS*, IEEE, Oct. 2016, pp. 1–3. doi: 10.1109/ICSENS.2016.7808812.

[53] M. Harada *et al.*, "20-μs Accuracy Time-Synchronization Method using Bluetooth Low Energy for Internet-of-Things Sensors," in *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, IEEE, Jan. 2022, pp. 181–186. doi: 10.1109/CCNC49033.2022.9700687.

[54] J. Li *et al.*, "Application-Layer Time Synchronization and Data Alignment Method for Multichannel Biosignal Sensors Using BLE Protocol," *Sensors*, vol. 23, no. 8, p. 3954, Apr. 2023, doi: 10.3390/s23083954.

[55] P. Enge and P. Misra, "Special Issue on Global Positioning System," *Proceedings of the IEEE*, vol. 87, no. 1, pp. 3–15, Jan. 1999, doi: 10.1109/JPROC.1999.736338.

[56] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems," *IEEE Std 1588-2019 (Revision ofIEEE Std 1588-*

2008), pp. 1–499, 2020, doi: 10.1109/IEEESTD.2020.9120376.

[57] S. Jens, F. Christian, P. Richard, and B. Holger, "Sub-Microsecond Time Synchronization for Network-Connected Microcontrollers," in *2024 IEEE International Conference on Consumer Electronics (ICCE)*, as Vegas, NV, USA: IEEE, Jan. 2024. doi: 10.1109/ICCE56470.2023.10043582.

[58] S. Puckett and E. Jovanov, "ecoSync: An Energy-Efficient Clock Discipline Data Synchronization in Wi-Fi IoMT Systems," *Electronics (Basel)*, vol. 12, no. 20, p. 4226, Oct. 2023, doi: 10.3390/electronics12204226.

[59] P. Chen and Z. Yang, "Understanding Precision Time Protocol in Todays Wi-Fi Networks: A Measurement Study," in *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, USENIX Association, Jul. 2021, pp. 597–610. [Online]. Available: https://www.usenix.org/conference/atc21/presentation/chen

[60] A. S. Nagra, M. A. Pasha, and S. Masud, "FPGA Implementation of IEEE 1588 Protocol for Bluetooth-based Distributed Wireless Systems," in *2022 29th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, IEEE, Oct. 2022, pp. 1–4. doi: 10.1109/ICECS202256217.2022.9970850.

[61] J. Elson, L. Girod, and D. Estrin, "Fine-grained network time synchronization using reference broadcasts," *ACM SIGOPS Operating Systems Review*, vol. 36, no. SI, pp. 147–163, Dec. 2002, doi: 10.1145/844128.844143.

[62] S. Ganeriwal, R. Kumar, and M. B. Srivastava, "Timing-sync protocol for sensor networks," in *Proceedings of the 1st international conference on Embedded networked sensor systems*, New York, NY, USA: ACM, Nov. 2003, pp. 138–149. doi: 10.1145/958491.958508.

[63] J. Cappelle, S. Goossens, L. De Strycker, and L. Van der Perre, "Low-Power Synchronization for Multi-IMU WSNs," *IEEE Embed Syst Lett*, pp. 1–1, 2024, doi: 10.1109/LES.2023.3313194.

[64] M. Elsharief, M. A. Abd El-Gawad, and H. Kim, "Low-Power Scheduling for Time Synchronization Protocols in A Wireless Sensor Networks," *IEEE Sens Lett*, vol. 3, no. 4, pp. 1–4, Apr. 2019, doi: 10.1109/LSENS.2019.2902389.

[65] M. Elsharief, M. A. Abd El-Gawad, and H. Kim, "FADS: Fast Scheduling and Accurate Drift Compensation for Time Synchronization of Wireless Sensor Networks," *IEEE Access*, vol. 6, pp. 65507–65520, 2018, doi: 10.1109/ACCESS.2018.2878272.

[66] M. Elsharief, M. A. A. El-Gawad, H. Ko, and S. Pack, "LPSRS: Low-Power Multi-Hop Synchronization Based on Reference Node Scheduling for Internet of Things," *Energies (Basel)*, vol. 15, no. 6, p. 2289, Mar. 2022, doi: 10.3390/en15062289.

[67] M. Elsharief, A. A. Emran, H. Hassan, S. R. Sabuj, and H.-S. Jo, "SLES: Scheduling-

Based Low Energy Synchronization for Industrial Internet of Things," *IEEE Sens J*, vol. 22, no. 16, pp. 16652–16661, Aug. 2022, doi: 10.1109/JSEN.2022.3188758.

[68] M. Elsharief, A. Emran, H. Hassan, S. R. Sabuj, and H.-S. Jo, "Energy-Efficient Synchronization in Industrial Internet of Things: An Intelligent Neighbor-Knowledge Approach," *IEEE Trans Industr Inform*, pp. 1–11, 2024, doi: 10.1109/TII.2024.3369236.

[69] "Power Profiler Kit II - Downloads --- nordicsemi.com," https://www.nordicsemi.com/Products/Development-hardware/Power-Profiler-Kit-2/Download".

[70] T. Schmid, P. Dutta, and M. B. Srivastava, "High-resolution, low-power time synchronization an oxymoron no more," in *Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks*, New York, NY, USA: ACM, Apr. 2010, pp. 151–161. doi: 10.1145/1791212.1791231.

[71] J. S. Dvorak, M. L. Stone, and K. P. Self, "Object Detection for Agricultural and Construction Environments Using an Ultrasonic Sensor.," *J Agric Saf Health*, vol. 22 2, pp. 107–19, 2016, [Online]. Available: https://api.semanticscholar.org/CorpusID:24771869

[72] J. Steckel, A. Boen, and H. Peremans, "Broadband 3-D Sonar System Using a Sparse Array for Indoor Navigation," *IEEE Transactions on Robotics*, vol. 29, no. 1, pp. 161–171, Feb. 2013, doi: 10.1109/TRO.2012.2221313.

[73] V. A. Zhmud, N. O. Kondratiev, K. A. Kuznetsov, V. G. Trubin, and L. V Dimitrov, "Application of ultrasonic sensor for measuring distances in robotics," *J Phys Conf Ser*, vol. 1015, p. 032189, May 2018, doi: 10.1088/1742-6596/1015/3/032189.

[74] Y. Yoshikawa, Y. Yasuda, I. Toru, S. Izumi, and H. Kawaguchi, "12.5-m Distance Measurement in High-Interference Environment Using Ultrasonic Array Sensors," in *2021 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, IEEE, May 2021, pp. 1–6. doi: 10.1109/I2MTC50364.2021.9459822.

[75] H. Furuhashi, Y. Kuzuya, Y. Uchida, and M. Shimizu, "Three-dimensional imaging sensor system using an ultrasonic array sensor and a camera," in *2010 IEEE Sensors*, IEEE, Nov. 2010, pp. 713–718. doi: 10.1109/ICSENS.2010.5690053.

[76] S. Harput and A. Bozkurt, "Ultrasonic Phased Array Device for Acoustic Imaging in Air," *IEEE Sens J*, vol. 8, no. 11, pp. 1755–1762, Nov. 2008, doi: 10.1109/JSEN.2008.2004574.

[77] J. H. Christensen, S. Hornauer, and S. Yu, "BatVision: Learning to See 3D Spatial Layout with Two Ears," Dec. 2019, [Online]. Available: http://arxiv.org/abs/1912.07011

[78] R. Whyte, L. Streeter, M. J. Cree, and A. A. Dorrington, "Application of lidar techniques to time-of-flight range imaging," *Appl Opt*, vol. 54, no. 33, p. 9654, Nov. 2015, doi: 10.1364/AO.54.009654.

[79] S.-W. Yang and C.-C. Wang, "On Solving Mirror Reflection in LIDAR Sensing," *IEEE/ASME Transactions on Mechatronics*, vol. 16, no. 2, pp. 255–265, Apr. 2011, doi: 10.1109/TMECH.2010.2040113.

[80] A. Ziebinski, R. Cupek, H. Erdogan, and S. Waechter, "A Survey of ADAS Technologies for the Future Perspective of Sensor Fusion," 2016, pp. 135–146. doi: 10.1007/978-3-319-45246-3_13.

[81] L. Jiang, T. Cai, Q. Ma, F. Xu, and S. Wang, "Active Object Detection in Sonar Images," *IEEE Access*, vol. 8, pp. 102540–102553, 2020, doi: 10.1109/ACCESS.2020.2999341.

[82] Y. Yari *et al.*, "Automated Measurement of Ovary Development in Atlantic Salmon Using Deep Learning," *Ultrasound Med Biol*, vol. 50, no. 3, pp. 364–373, Mar. 2024, doi: 10.1016/j.ultrasmedbio.2023.11.008.

[83] G. Allevato, M. Rutsch, J. Hinrichs, M. Pesavento, and M. Kupnik, "Embedded Air-Coupled Ultrasonic 3D Sonar System with GPU Acceleration," in *2020 IEEE SENSORS*, IEEE, Oct. 2020, pp. 1–4. doi: 10.1109/SENSORS47125.2020.9278601.

[84] G. Allevato *et al.*, "Air-Coupled Ultrasonic Spiral Phased Array for High-Precision Beamforming and Imaging," *IEEE Open Journal of Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 2, pp. 40–54, 2022, doi: 10.1109/OJUFFC.2022.3142710.

[85] S. Kumar and H. Furuhashi, "Long-range measurement system using ultrasonic range sensor with high-power transmitter array in air," *Ultrasonics*, vol. 74, pp. 186–195, Feb. 2017, doi: 10.1016/j.ultras.2016.10.012.

[86] W. Kim, "Signal-to-noise ratio and bandwidth for pseudo-random codes in an ultrasonic imaging system," *Ultrason Imaging*, vol. 6, no. 3, pp. 313–323, Jul. 1984, doi: 10.1016/0161-7346(84)90016-6.

[87] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," Dec. 2013.

[88] Y. Na, Y. Guo, Q. Fu, and Y. Yan, "An Acoustic Traffic Monitoring System: Design and Implementation," in *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, IEEE, Aug. 2015, pp. 119–126. doi: 10.1109/UIC-ATC-ScalCom-CBDCom-IoP.2015.41.

[89] T. Takahashi *et al.*, "Acoustic Traffic Monitoring Based on Deep Neural Network Trained by Stereo-Recorded Sound and Sensor Data," in *2023 31st European Signal Processing Conference (EUSIPCO)*, IEEE, Sep. 2023, pp. 935–939. doi:

10.23919/EUSIPCO58844.2023.10290031.

[90] M. I. Skolnik, *Introduction to Radar Systems*, 3rd ed. New York, NY: McGraw-Hill Professional, 2002.

[91] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum, "Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling," *CoRR*, vol. abs/1610.07584, 2016, [Online]. Available: http://arxiv.org/abs/1610.07584

[92] "PYNQ-Z2," [online] Available: www.pynq.io.