https://doi.org/10.7840/kics.2025.50.11.1780

On the Analysis of Software-Defined Radio with RISC-V Vector Architecture

Seungsik Moon◆, Hyeongtaek Lee°

ABSTRACT

Software-defined radio (SDR) architecture is a baseband architecture based on general-purpose processors, offering greater flexibility than the conventional baseband architecture by allowing various algorithms to be processed in software-level. While SDRs provide a flexible alternative to traditional baseband architectures, most of them are built on conventional x86 or ARM-based processors and thus lack the customizability needed for optimal performance in wireless communication. This paper analyzes the performance of an SDR architecture implemented on a vector-extended RISC-V core architecture. We designed a vector-extended RISC-V core architecture by adding various vector-level operators to the open-source RISC-V core architecture. Furthermore, we developed an optimized computing kernel for the maximum-ratio transmission (MRT) precoding function using the vector-level instructions. Simulation results showed that the vector-extended SDR architecture can achieve up to 17.1 times higher throughput than the conventional SDR architecture, validating the viability and feasibility of the SDR architecture with the vector-extended RISC-V core architecture.

Key Words : Software-defined radio, RISC-V, vector extension, maximum-ratio transmission

Ⅰ. Introduction

The wireless communication technologies have evolved continuously, leading to increasingly complex and diverse communication standards. Following the advances in communication standards, the latency and energy constraints of the communication systems have also been strictly limited, requiring highly efficient implementation of the baseband systems. To meet those constraints, the physical (PHY) layer of baseband systems has been implemented using application-specific integrated circuits (ASICs), leading to minimal latency and maximal energy efficiency of the baseband systems. However, the inherent fixed structure of ASICs also makes them impossible to accommodate diverse configurations or adapt to evolving standards without costly hardware redesigns.

To overcome these fundamental limitations, the software-defined radio (SDR) architecture has recently been considered for substituting the conventional ASIC-based baseband systems[1-3]. SDR architecture can implement most functions of the PHY layer on a general-purpose processor, allowing designers to reprogram the functions for different wireless configurations or communication standards. Key advantages of SDR architecture include the ability to dynamically switch between multiple configurations, support rapid prototyping of new types of waveforms, and facilitate over-the-air updates to enhance system capabilities. Moreover, recent advancements in semiconductor manufacturing processes and software optimization techniques have empowered recent general-purpose processors to achieve performance levels previously only attainable by dedicated hardware.

While many researches have investigated the solution for SDR architectures on general-purpose pro-

◆ First Author : Chungbuk National University, Department of Electronics Engineering, seungsik.moon@cbnu.ac.kr, 정회원
° Corresponding Author : Ewha Womans University, Department of Electronic and Electrical Engineering, htlee@ewha.ac.kr, 정회원
논문번호 : 202509-223-B-RU, Received September 1, 2025; Revised September 16, 2025; Accepted September 16, 2025

cessors, these studies have relied on conventional x86 or ARM-based architectures[4,5]. Since these architectures have closed-nature instruction-set architectures (ISAs) and fixed hardware structures, it is hard to further optimize the SDR architecture by introducing dedicated instructions or customized hardware acceleration systems tailored to the computational patterns of wireless workloads. Recently, the RISC-V architecture has been emerging as a comparable alternative to the customized processor architecture[6]. As a free, open-source, and extensible ISA, the RISC-V architecture enables researchers and developers to co-design hardware and software, especially meaningful for designing high-performance SDR architectures. Moreover, by adopting the standardized RISC-V vector extension, it is possible to implement the large-sized matrix-level operations, mandatory for PHY layer functions like multiple-input multiple-output (MIMO) precoders and combiners, with a highly efficient architecture.

In this paper, we analyze the performance of the SDR architecture with the RISC-V vector architecture. For this, we designed the vector-extended RISC-V core architecture by extending the open-source RISC-V core architecture[6]. According to the RISC-V vector-level ISA specification, dedicated vector arithmetic operators and data management units are carefully implemented. Moreover, a dedicated kernel for the matrix-level operations is designed based on the vector-level instructions, efficiently supporting the designed vector-extended RISC-V core architecture. To evaluate the performance of the PHY layer functions on SDR architecture, maximum-ratio transmitter (MRT) algorithm under various MIMO configurations is designed and simulated using the previously built RISC-V core architecture as well as the optimized computing kernel.

Ⅱ. Backgrounds

2.1 RISC-V Architecture

RISC-V architecture is an open-source ISA, which can be freely used by academia and industry. It is composed of a small base integer ISA and several extension ISAs, offering a flexible core architecture by implementing only the required instructions for the target algorithms or systems. Due to its modularity, RISC-V architectures are adopted to various types of systems ranging from resource-constrained edge-level systems[7] to the high-performance server-scale computing systems[8]. Among its many optional extensions, the vector extension is particularly significant for high-performance computing tasks. It provides a simple yet powerful framework for data-parallel processing, where a single vector-level instruction can operate on multiple data elements simultaneously. Therefore, the practical benefit of the vector extension becomes clear when dealing with algorithms that rely heavily on matrix and vector operations, which are fundamental for recent wireless communication systems. For example, multiplying two large matrices can be dramatically accelerated by executing an entire row-column multiplication in a highly parallel fashion using a few vector-level instructions. This allows a general-purpose RISC-V core to execute these compute-intensive algorithms with a comparable efficiency to the specialized accelerator designs, offering a flexible but powerful computing system.

Another strong feature of the RISC-V vector extension is that it supports various vector-level configuration instructions, which enable the software to dynamically re-configure the hardware at runtime. Unlike traditional single-instruction multiple-data (SIMD) engines with fixed capabilities, therefore, the RISC-V architecture can effectively turn the vector-level operator into a software-reconfigurable accelerator. More precisely, the program can reconfigure the hardware with different lengths of vector-level operations, size of element, or logical length of vector operand, giving adaptability to various types of computing tasks. This feature particularly fits for the SDR architecture targeting the wireless communication systems, which require various types of computing tasks such as matrix-level multiplication, fast Fourier transform, vector-level permutation, or gather-scatter operations.

2.2 System Model

We consider a downlink $N \times K$ multi-user MIMO (MU-MIMO) system, where a base station (BS) equipped with N antennas serves K single-antenna users simultaneously. By representing the channel from the BS to k-th user by $\mathbf{h}_k^{\mathrm{H}} \in \mathrm{C}^{1\times N}$, the downlink received signal at the k-th user y_k can be formulated as follows:

$$\begin{aligned} y_k &= \mathbf{h}_k^{\mathrm{H}} \left(\sum_{j=1}^{K} \mathbf{w}_j s_j \right) + n_k \\ &= \mathbf{h}_k^{\mathrm{H}} \mathbf{w}_k s_k + \mathbf{h}_k^{\mathrm{H}} \left(\sum_{j\neq k}^{K} \mathbf{w}_j s_j \right) + n_k, \end{aligned} \tag{1}$$

where s_k, $\mathbf{w}_k$, and n_k are information symbol, corresponding precoding vector, and noise signal for k-th user, respectively. The received signal at each user is a superposition of its desired signal, interference from signals intended for other users, and noise. We assume the additive white Gaussian noise (AWGN) for the noise signal of each user.

In MU-MIMO systems, the BS typically has more antennas than the number of users it serves ($N > K$), enabling the BS to have spatial degrees of freedom. To leverage these extra degrees of freedom, it is required to carefully design the precoding vector, $\mathbf{w}_k$. The precoding vectors control how signals are transmitted in space, allowing the BS to achieve specific objectives like maximizing signal power or mitigating interferences. The MRT is one of the popular linear precoding techniques, where the BS designs a precoding vector for each user to maximize the power of the desired signal at that user. By assuming that the BS has full channel state information, e.g., the BS perfectly knows $\mathbf{h}_k$ for every user, it can be achieved by aligning the transmitted signal with the user's channel characteristic. More precisely, the MRT precoding vector for user k can be obtained as a conjugate transpose of its channel vector as follows:

$$\mathbf{w}_k = \frac{\mathbf{h}_k}{\|\mathbf{h}_k\|_2}, \tag{2}$$

where it is normalized to satisfy the power constraint.

Although the MRT cannot actively cancel the inter-user interference, it is already known that the MRT can achieve good performance for massive MIMO configuration ($N \gg K$) by exploiting channel hardening effects. From the obtained precoding vectors, the transmitted signal from the BS is the superposition of all individually precoded data streams, which can be expressed as a matrix-vector multiplication. By defining a precoding matrix $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \cdots, \mathbf{w}_K]$ and a symbol vector $\mathbf{s} = [s_1, s_2, \cdots, s_K]^{\mathrm{T}}$, the final transmitted signal can be expressed as follows:

$$\mathbf{x} = \mathbf{W}\mathbf{s}. \tag{3}$$

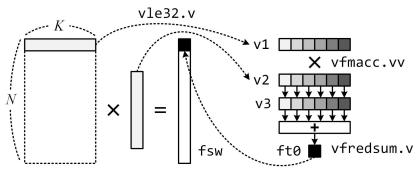
As a result, data transmission sequence with the MRT can be performed by computing the vector norm, scaling the vector with the vector norm, and performing matrix-vector multiplication. The performance of the BS is therefore highly dependent on the efficiency of these computing steps. In this paper, we implement the MRT precoding function and analyze the performance of the SDR architecture with the RISC-V architecture.

Ⅲ. Software-Defined Precoder Design with RISC-V Vector Architecture

3.1 Software Design

To efficiently implement vector-level computing operations used in the MRT precoding function, we carefully designed the optimized computing kernel for the RISC-V architecture along with the vector extension. For given channel vectors, the L2-norm of each channel vector can be obtained with the element-wise vector-vector multiplication instruction and the vector reduction instruction, followed by the square-root operation. Then, the reciprocal of the L2-norm is computed by the division instruction, which is utilized as a scaling factor for satisfying the power constraint. By scaling the channel vector with the vector-scalar multiplication instruction, the MRT precoding vector can be obtained. As described in (3), the final transmitted signal can be obtained by matrix-vector multiplication using the computed precod-

ing vectors and the corresponding information symbols. By grouping the precoding vectors into a single precoding matrix, it can be accomplished by performing multiple vector-level multiplication processes between the row vectors of the precoding matrix and the column vector with the information symbols. Therefore, we designed the matrix-vector multiplication kernel with multiple pairs of the elementwise vector-vector multiplication instruction and the vector reduction instruction. Fig. 1 shows the example of the matrix-vector multiplication kernel for the $N \times K$ MU-MIMO configuration. Note that the size of each vector-level instruction can be easily configured with the vector-level configuration instructions, supporting various MIMO configurations at the software level. For the multiplication between the matrix **A** and column vector **b**, output vector **c** = **Ab** is obtained by iteratively performing vector-vector inner product between each row vector of **A** and **b**.




```
// Vector configuration
asm volatile(
    "vsetvli zero, %0, e32, m1, ta, ma" ::"r"(K)
);

// Fetch column vector B
asm volatile(
    "vle32.v v2, (%0);" ::"r"(b)
);

// Compute the multiplication
unsigned long int n = 0;

for (n = 0; n < N; n++)
{
    asm volatile (
        // Fetch row vector of A
        "vle32.v v1, (%[a]);"

        // Vector-vector inner product
        "vfmacc.vv v3, v1, v2;"
        "vfredsum.vs v3, v3, v0;"
        "vfmv.f.s ft0, v3;"
        "fsw ft0, (%[c]);"

        // Move to next vectors
        "addi %[a], %[a], 64;"
        "addi %[c], %[c], 4;"

        : [a] "+r" (a),
          [c] "+r" (c)
        :
        : "ft0", "memory"
    );
}
```

Fig. 1. Example of matrix multiplication kernel with RISC-V vector-level instructions.

Moreover, overall processing efficiency can be further enhanced by reusing the row vectors of the precoding matrix. Considering that the channel matrix would be consistent during the channel coherence time, the same precoding matrix can also be utilized multiple times to serve the same users. Therefore, if the SDR architecture serves M consecutive information symbol vectors for the same users, the corresponding M transmitted signals can be obtained as a matrix-matrix multiplication process as follows:

$$\mathbf{X} = \mathbf{WS} = \mathbf{W} \cdot [\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_M]. \tag{4}$$

Since the RISC-V vector extension supports 32 vector-level registers, row vectors of the precoding matrix can be loaded into the vector-level registers and reused for different information symbol vectors, largely enhancing the processing efficiency and throughput of the SDR architecture.

3.2 Hardware Design

To efficiently realize the MRT precoding function on the RISC-V architecture, we designed a vector-extended RISC-V core architecture by extending the open-source element-level RISC-V core architecture[9]. In contrast to a conventional element-level core architecture that processes data on an item-by-item basis, the vector-extended RISC-V core is designed for high-throughput data parallelism, processing multiple elements simultaneously with vector-level operations. To achieve this, as illustrated in Fig. 2, three key vector-level hardware units have been added, which are the vector-level arithmetic unit (VALU), the vector-level data management unit (VDMU), and the vector register file (VRF).

The VALU includes 16 arithmetic and logical units (ALUs) and floating-point units (FPUs) to support the data-parallel vector-level arithmetic instructions. Each ALU and FPU supports up to 32-bit data bitwidth, capable of processing up to 512-bit vector for each vector-level arithmetic instruction. The VALU is implemented by extending the execution stage of the previous core architecture; therefore, every vector-level arithmetic instruction is assumed to be executed in

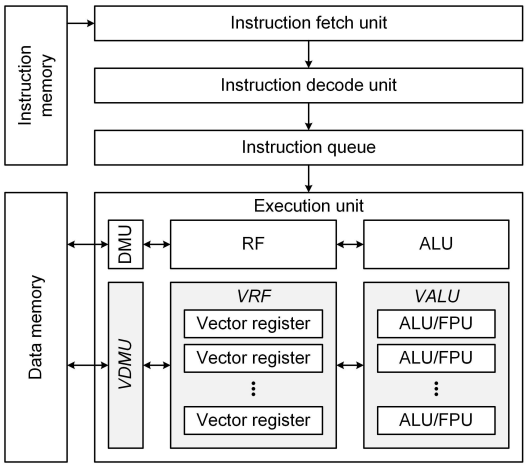


Fig. 2. Block diagram of the vector-extended RISC-V core architecture.

a single-cycle latency. By utilizing the parallel computational units in the VALU, it is possible to manipulate multiple elements with a single vector-level arithmetic instruction. The VDMU handles vector-level memory load and store instructions, which are designed to facilitate data transfers between main memory and the VRF. When the vector-level load or store instruction is executed, the VDMU transfers the vector with a burst transaction, where it iteratively invokes element-level operations to load or store the vector. The loaded vectors are stored in the VRF, which implements 32 vector registers to temporarily store vectors used for vector-level arithmetic instructions. Similar to the conventional element-level register file, the VRF is a group of multiple registers, where each vector register has 512-bit vector length to align with the implementation of VALU. These three vector-level units work closely together to execute vector-level instructions, allowing the SDR architecture to achieve high throughput comparable to dedicated hardware while retaining the programmability.

Ⅳ. Performance Analysis

In this section, we analyze and compare the performance of the MRT precoding function with the SDR architecture. It is assumed that the BS already has the channel matrix with full channel state information in the main memory; then, the BS performs the MRT precoding function using the obtained channel matrix. Under this assumption, the performance of the BS for performing the MRT precoding function is carefully analyzed. Table 1 shows the performance comparison of SDR architectures with the conventional element-level RISC-V core architecture and the vector-extended RISC-V core architectures as well as the corresponding software-level computing kernels. Note that the downlink throughput is not directly related to the sum data-rate, but it depends on the ability to how many information symbols can be processed in a given time. For the 128 × 16 MU-MIMO configuration and the 64-QAM modulation, the required processing latency, the downlink BS throughput, and the required program size are compared. With the conventional element-level RISC-V core architecture, the SDR architecture requires 27.4 us of processing latency. By implementing the vector-extended RISC-V core architecture, the processing latency can be reduced to 5.98 us by utilizing the data-parallel vector-level instructions, resulting in only 21.8 % of processing latency compared with the conventional SDR architecture. In terms of downlink BS throughput, the vector-extended SDR architecture can achieve 16.1 Mbps of throughput, which is 4.6 times higher than the conventional SDR architecture with 3.50 Mbps of throughput. Moreover, the vector-level instruction can handle multiple data elements with a single instruction, reducing the required program size to perform the MRT precoding function to 78.3 % of the total program size compared with the previous element-level operation.

Fig. 3 illustrates the required processing latency of SDR architectures with different MIMO configurations. We analyzed the performance for a total of eight different MIMO configurations, considering four different numbers of users, i.e., K = 8, 16, 24, 32, and two different numbers of BS antennas,

Table 1. Performance comparison of SDR architectures with different RISC-V core architectures.

Type	Latency	Throughput	Program size
Element-level	27.4 us	3.50 Mbps	4.38 Kbit
Vector-level	5.98 us	16.1 Mbps	3.43 Kbit
Ratio	21.8 %	460 %	78.3 %

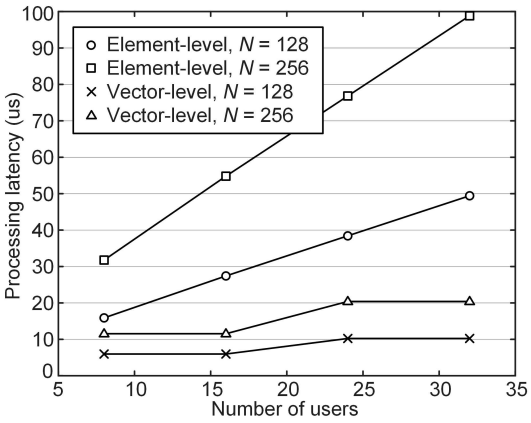


Fig. 3. Required processing latency with the different MIMO configurations.

i.e., N =128, 256. As illustrated in Fig. 3, the SDR architecture with vector-extended RISC-V core architecture always offers efficient processing compared to the conventional element-level core architecture regardless of the MIMO configuration. By increasing the number of users, the required processing latency of the conventional element-level SDR architecture also increases linearly following the number of required operations. In contrast, the SDR architecture with vector-extended RISC-V core architecture can handle multiple data elements efficiently, largely reducing the required processing latency. For 256 × 32 MU-MIMO configuration, the conventional element-level SDR architecture consumes 98.8 us of processing latency; whereas, the vector-extended SDR architecture only requires 20.4 us of processing latency, reducing the required processing latency by 4.8 times. Note that the vectorextended SDR architecture requires similar processing latency for 8 users and 16 users. Considering that the size of each vector in the vector-level operation depends on the number of users, the size of VALU determines the processing latency of the vector-level operations. Since the VALU implements 16 processing elements, the vector-extended SDR architecture can perform vector-level operations with the same processing latency up to 16 users. If the number of users exceeds 16, each vector-level operation should be divided into several sub-operations and finished in multiple processing cycles. Therefore, as illustrated in Fig. 3, the required processing latency for 24 users and 32 users is larger than the processing latency for 16 users.

Fig. 4 depicts downlink BS throughput for different SDR architectures with different numbers of served information vectors M. It is assumed that 16 users are served with 64-QAM modulation. Although increasing the number of served information vectors can enhance the processing efficiency by transmitting multiple information symbols with the same precoding matrix, it still requires multiplication between the precoding matrix and the information vectors. Since the cost of matrix-level multiplication accounts for the majority of the total cost, the total throughput of the conventional element-level SDR architecture remains similar for different numbers of served information vectors due to the increased cost for larger matrix-level multiplication. In contrast, the vector-extended SDR architecture can largely enhance downlink BS throughput by increasing the number of served information vectors. Since the vector-extended RISC-V core architecture can reuse the precoding matrix by storing the precoding vectors in VRF, increasing the number of served information vectors enhances processing efficiency by maximizing the reuse of the precoding vectors. As a result, for the 128 × 16 MU-MIMO configuration, the vector-extended SDR architecture can achieve 64.9 Mbps of downlink BS throughput by serving 16 information vectors, showing 17.1 times higher throughput compared with the conventional SDR architecture. Note that it is also

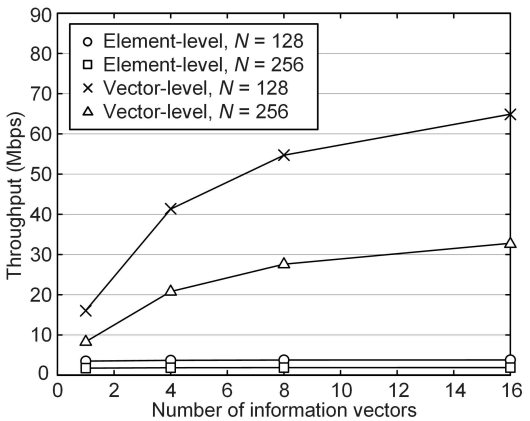


Fig. 4. Downlink BS throughput with the different number of served information vectors.

possible to implement other essential SDR operations such as fast Fourier transform (FFT)/inverse FFT (IFFT), channel coding/decoding, channel estimation, or symbol detection with the proposed RISC-V architecture. However, it also requires highly optimized computing kernel considering the algorithm-level characteristics and core architectures, which are still underexplored and can be great future work.

Ⅴ. Conclusion

In this paper, we analyzed the performance of the SDR architecture with the vector-extended RISC-V core architecture. By extending the open-source RISC-V core architecture supporting conventional element-level ISA, we prototyped the vector-extended RISC-V core architecture with 16 parallel processing elements, vector-level data management unit, and vector-level register file. With the vector-level instructions, we also designed the optimized computing kernel for the MRT precoding function. For 128 × 16 MU-MIMO configuration with 64-QAM modulation, the simulation results showed that the vector-extended SDR architecture achieves 4.58 times lower processing latency and 4.6 times higher BS downlink throughput. Due to the data reuse capability of the vector-extended architecture, the processing efficiency of the vector-extended SDR architecture can be further enhanced by increasing the number of served information vectors. As a result, this work successfully validated the viability and feasibility of the SDR architecture with vector-extended RISC-V core architecture, leading to the powerful but flexible baseband architecture for the next-generation communication scenarios.

References

[1] T. Ulversoy, "Software defined radio: Challenges and opportunities," *IEEE Commun. Surv. & Tuts.*, vol. 12, no. 4, pp. 531-550, May 2010. (https://doi.org/10.1109/SURV.2010.032910.00019)

[2] S. Yoo, J. Yoo, and S. Y. Kim, "Open source GPS L1 C/A SDR implementation for fast processing of acquisition and tracking using MAT-LAB PCT and CMEX," *J. KICS*, vol. 49, no. 9, pp. 1295-1305, Sep. 2024. (https://doi.org/10.7840.kics.2024.49.9.1295)

[3] D. S. Kang and K. W. Choi, "Development of SDR testbed for ultra-dense networks," *J. KICS*, vol. 44, no. 6, pp. 1181-1187, Jun. 2019. (https://doi.org/10.7840/kics.2019.44.6.1181)

[4] H. Abdellatif, V. Ariyarathna, A. Madanayake, and J. M. Jornet, "A real-time software-defined radio platform for sub-terahertz communication systems," *IEEE Access*, vol. 12, Oct. 2024. (https://doi.org/10.1109/ACCESS.2024.3473615)

[5] S. Yun, J. Choe, and Y. Lee, "High-throughput software-defined LDPC encoder and decoder with x86-based data-level parallelism," *IEEE Trans. Veh. Technol.*, vol. 74, no. 1, Jan. 2025. (https://doi.org/10.1109/TVT.2024.3452718)

[6] *The RISC-V Instruction Set Manual Volume I*, Retrieved Aug., 1, 2025, from https://github.com/riscv/riscv-isa-manual

[7] J. Park, et al., "Designing low-power RISC-V multicore processors with a shared lightweight floating point unit for IoT endnodes," *IEEE Trans. Circuits and Syst. I: Regular Papers*, vol. 71, no. 9, Sep. 2024. (https://doi.org/10.1109/TCSI.2024.3427681)

[8] Y. Wang, M. Yang, C.-P. Lo, and J. P. Kulkarni, "Vecim: A 289.13 GOPS/W RISC-V vector co-processor with compute-in-memory vector register file for efficient high-performance computing," in *Proc. 2024 IEEE ISSCC*, pp. 492-494, San Francisco, CA, USA, Feb. 2024. (https://doi.org/10.1109/ISSCC49657.2024.10454387)

[9] *SCR1 RISC-V Core*, Retrieved Aug., 1, 2025, from https://github.com/syntacore/scr1

Seungsik Moon

Feb. 2018: B.S. degree, Dept. of Electrical Engineering, Pohang University of Science and Technology

Aug. 2023: Ph.D. degree, Dept. of Electrical Engineering, Pohang University of Science and Technology

Sep. 2023~Feb. 2025 : Researcher, Electronics and Telecommunications Research Institute (ETRI)

Mar. 2025~Current : Assistant Professor, Dept. of Elec- tronics Engineering, Chungbuk National University

<Research Interests> Wireless communication system, intelligent system, RISC-V core architecture, VLSI design

[ORCID:0000-0001-7723-0419]

Hyeongtaek Lee

Aug. 2018: B.S. degree, Dept. of Electrical Engineering, Pohang University of Science and Technology

Aug. 2023: Ph.D. degree, School of Electrical Engineering, Korea Advanced Institute of Science and Technology

Sep. 2023~Feb. 2025 : Postdoctoral Researcher, Korea Advanced Institute of Science and Technology

Mar. 2025~Current : Assistant Professor, Dept. of Electronic and Electrical Engineering, Ewha Womans University

<Research Interests> Massive MIMO, Reconfigurable Intelligent Surface, Integrated Sensing and communi-cation, Feasible Wireless Communi-cations

[ORCID:0000-0002-8611-5196]