Integrated Signal Processing for Massive MIMO Systems

Yue Dai

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2024-30
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2024/EECS-2024-30.html

May 1, 2024

Integrated Signal Processing for Massive MIMO Systems

by

Yue Dai

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering - Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Borivoje Nikolić, Chair
Professor Ali Niknejad
Professor Jasmina Vujic

Spring 2023

Integrated Signal Processing for Massive MIMO Systems

Abstract

Integrated Signal Processing for Massive MIMO Systems

by

Yue Dai

Doctor of Philosophy in Engineering - Electrical Engineering and Computer Science

University of California, Berkeley

Professor Borivoje Nikolić, Chair

The massive multiple-input-multiple-output (MIMO) wireless communication technology is one of the critical parts in the 5G and 5G+ communication systems, which offers an energy- and cost-efficient, secure, and robust solution for the next generation of communication systems. This dissertation focuses on developing integrated signal processing for massive MIMO systems to improve the performance and efficiency of massive MIMO for future communication systems. It provides a comprehensive analysis of the design and prototyping of scalable mmWave massive MIMO testbeds from both algorithmic and hardware architecture perspectives. It focuses on achieving better data aggregation, linear scaling of computation complexity, cost and energy efficiency, and sustainable development. It demonstrates the feasibility of such systems by designing an algorithm-adaptable, scalable, and platform-portable generator for massive MIMO baseband processing systems, which can be customized for different MIMO systems and hardware configurations. The generator is evaluated by using various channel estimation methods, such as the flat fading, line-of-sight, Rician, and QuaDRiGa channel models, and various hardware parameter values, demonstrating its algorithmic adaptability and effectiveness. The results show that the generator has competitive power consumption and can significantly improve the demodulation error vector magnitude by integrating beamspace methods. Additionally, to show the power of machine learning in the integrated signal processing of massive MIMO systems, a complex-valued neural network aided channel estimation method for massive MIMO systems with interference is proposed and evaluated. The network is trained with a wideband interference, and the method is evaluated with different design and training choices. The result shows an up to 3.16 times improvement in channel estimation performance with a shorter pilot overhead compared to the traditional method, demonstrating the effectiveness of the proposed method.

Contents

List of Figures

List of Tables

Acknowledgments

As I come to the end of my PhD journey, I really want to give my sincere appreciation to the people who have helped me for the last five years I spent in Berkeley.

First of all, I would like to express my heartfelt gratitude to my PhD advisor, Prof. Borivoje Nikolic, for his invaluable guidance and support in the past five years that have helped me build a strong base for my career and given me the confidence to face the challenges lying ahead in the future. His expertise and insightful advice on wireless communication and VLSI designs have navigated me to find the way out when I encountered difficulties in my research. I also want to thank him for providing me with great opportunities to access a wide range of research topics in my first and second years, such as the compressor accelerator project and Hydra tapeout experience, which helped me find my research interests and build my all-around abilities. Without his advice and support, I could not achieve my goals at Berkeley. I would also like to extend my thanks to my dissertation committee, Prof. Ali Niknejad and Prof. Jasmina Vujic, for their time to attend my qualification exam and their guidance in completing my dissertation. Their feedback and suggestions were instrumental in improving my dissertation. I also want to thank Prof. Christoph Studer from ETH Zurich and Prof. Upamanyu Madhow from University of California, Santa Babara, for providing me with their guidance and research resources on the Spine generator project. Their expertise and help are invaluable in refining my research and broadening my horizons in the area of massive MIMO communication systems. I am grateful for their time and willingness to share their insights with me, and also for the inter-university collaboration opportunity they have provided me with.

I would also like to express my deep appreciation to my colleagues, Greg LaCaille, Harrison Liew, and James Dunn for giving me great technical support. I really want to thank Greg for his leadership and patient guidance in my early years of research on the Hydra project and thank Harrison and James for their generous help during my study on agile design tools and FPGA programming. The Spine generator design and emulation could not be accomplished without their help. I am deeply appreciative of the support and inspiration I received from Maryam Eslami Rasekh from University of California, Santa Babara, Seyed Hadi Mirfarshbafan from ETH Zurich, and Alexandra Gallyas-Sanhueza from Cornell University, on the work of system integration. Their willingness to collaborate with me and share their knowledge and expertise with me on the beamspace domain signal processing is truly invaluable. I enjoy the time we worked together, and I look forward to other chances of collaboration in the future. In addition, I would also like to convey my special thanks to my friends, Qijing Huang, Sehoon Kim, Tianjun Zhang, and Suhong Moon for their expert insights and suggestions on machine learning that inspire me on my CVNN-aided channel estimator project. I am honored to be a friend of them. In addition, I'm truly thankful to my internship manager, Sriram Sundararajan, for offering me a great internship opportunity that opens my eyes to new perspectives and challenges existing in the industry, and for sharing his valuable experience with me on both working and life. I also want to extend my sincere

gratitude to the sponsors who have supported me throughout my PhD period, and their financial and resource support has enabled me to pursue my academic goals.

I would also like to appreciate the strong support offered by Candy Corpus who always stands by my side and cheers me up during my PhD life. I also appreciate the effort given by Mikaela Cavizo-Briggs, all admins, staff, and faculties in BWRC for organizing those joyful and meaningful events. They make a huge contribution to my work-life balance and make my days in BWRC colorful and happy. I also want to give my deepest appreciation to my dearest parents for their unwavering love, care, and faith in me throughout my whole life of study and no matter where I am. Without their love and support, I would not have been able to achieve my goals and pursue my dreams.

To my best friends, Jingyi Xu, Seah Kim, and Sunjin Choi, thank you for your company and support that helped me walk through my dark days during the pandemic. I am grateful for the encouragement and emotional support you gave me, and I also thank you for your willingness to listen to my concerns when I felt depressed. It is my great pleasure to meet you at Berkeley. Those wonderful memories that we made together in the past several years are a priceless gift to me.

And last but not least, to my furry cat, Yuhe, your company, and unconditional love really comfort me and brighten my days. I appreciate your presence in my life.

Chapter 1

Introduction

The massive MIMO communication technology is one of the key components in the 5th-generation (5G) and 5G+ communication systems. According to the analysis and projection by Ericsson [29] shown in Fig. 1.1, by 2028, the mobile data traffic can reach 325 exabytes per month, which is 29.5 times higher than the data traffic in 2017, and 3.6 times higher than today's data traffic. With the explosive growth of mobile devices and the increasing demand for high-speed data transfer, traditional point-to-point wireless communication systems employing capacity-achieving codes and modulation schemes have reached the theoretical Shannon limit [2], and it has become difficult to meet the demand for data using point-to-point communication techniques alone.

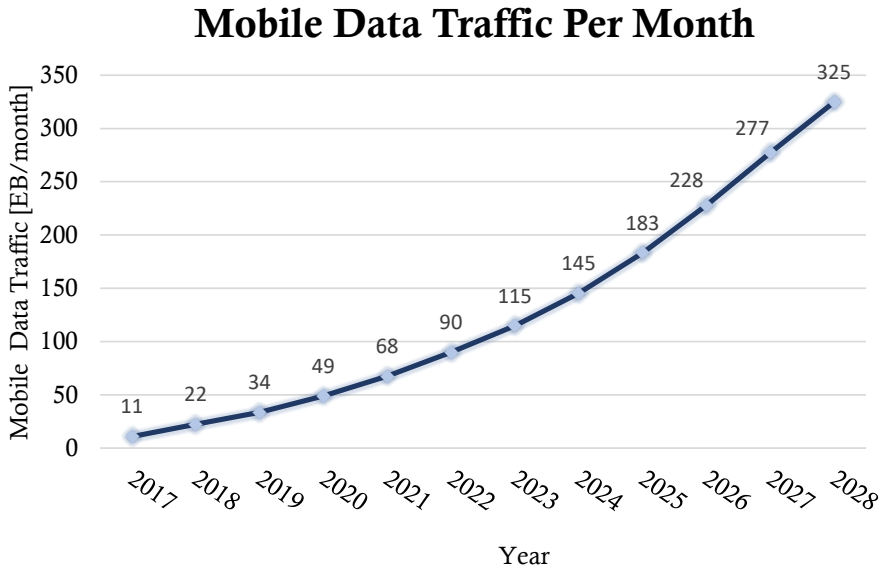


Figure 1.1: The 4G and 5G mobile data traffic projection.

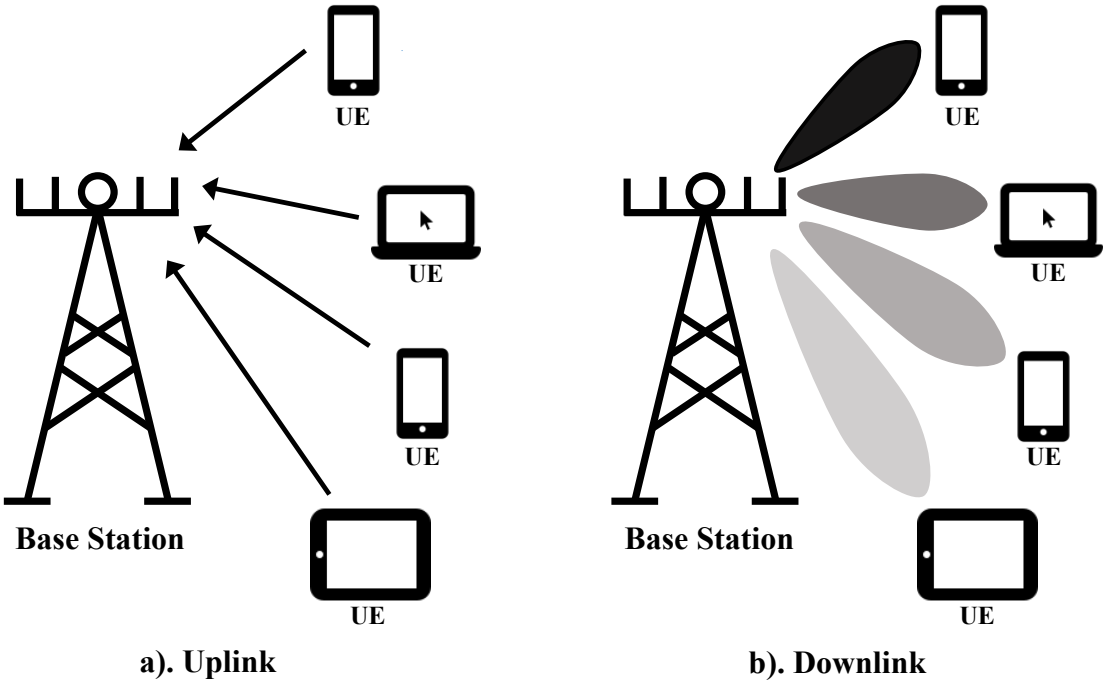


Figure 1.2: Massive MIMO communication systems assuming in a single cell: a) Uplink; b) Downlink.

Massive MIMO offers a solution for further increasing the channel capacity by expanding the spatial resolution of the base station using a large number of antennas, which allows each user equipment to have a low-interference, high-capacity link to the base station (BS) [76]. Combined with the mmWave frequency technology, it can further expand the signal bandwidth and reduce hardware component sizes, providing an energy-efficient, cost-efficient, secure, and robust solution for the next generation of communication systems [62, 103].

Fig. 1.2 shows the uplink and the downlink communication in a single-cell massive MIMO communication system. Uplink communication is where the user equipment (UE) sends information to the base station, while downlink communication is where the base station sends information to UEs through formed beams. As a result, massive MIMO, combined with other capacity-achieving techniques, can help meet the growing demand for wireless data transfer and enables fast media streaming, information sharing, and compute-on-edge applications.

1.1 Massive MIMO Implementation Challenges

As the number of antennas at the base station increases in massive MIMO systems, the difficulty of designing and prototyping scalable systems also increases. This challenge arises

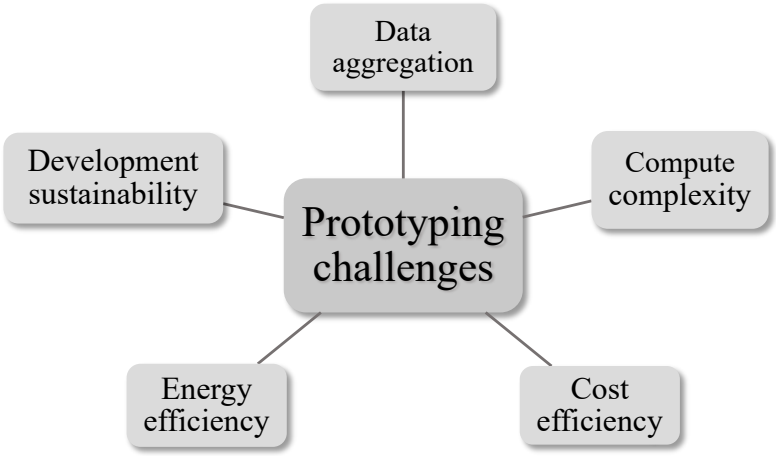


Figure 1.3: Challenges for the scalable massive MIMO system design and prototyping.

from multiple aspects summarized in Fig. 1.3. The first challenge is data aggregation. In systems with hundreds or thousands of antennas operating at mmWave frequencies, the raw signal data collected can reach terabytes per second. However, the bandwidth of commonly used high-speed interconnects, such as PCIe and Ethernet, is several tens to hundreds of gigabytes per second. Therefore, a simple BS interconnect network design cannot easily achieve real-time operation.

The second challenge is the computational complexity of the signal processing. In massive MIMO systems, the manufacturing and distributing of hardware components can create more hardware mismatches, which must be calibrated and corrected. The beamforming complexity increases quadratically or even cubically as the number of antennas and UEs the system supports grows. This complexity can prevent the scaling up of the massive MIMO system, especially regarding matrix multiplications and inversions.

Another challenge for implementing massive MIMO systems is the cost and energy consumption. As the system scale increases, even a slight increase in the cost of a single array component can significantly elevate the total cost. The same logic applies to energy consumption. Therefore, it is vital to build cost- and energy-efficient massive MIMO systems while working under the budget, power supply, and cooling limitations.

For prototyping massive MIMO systems for research purposes, it is also essential to consider development sustainability. As massive MIMO is being adopted in practical settings, new algorithms are being proposed to solve practical challenges and enhance performance. Some novel techniques are the beamspace domain algorithms [83, 3], the compressed-sensing based method [56], and machine learning algorithms [6, 54] for the performance improvement on channel estimation, channel prediction, and signal detection. As a result, massive MIMO system design needs to be flexible to adapt new algorithms or hardware for further development and support partial updates without rebuilding the entire testbed or system from scratch.

1.2 Massive MIMO Channel Estimation Challenges

In massive MIMO communication systems, accurate estimation of channel state information (CSI) is important to form a high-capacity and robust communication link. The massive MIMO can achieve the full gain from the scaling of the antenna arrays if the channel estimation is perfect; otherwise, the performance loss will happen.

As shown in Fig. 1.2, at the uplink communication, BS estimates the pairwise channel between UEs and the receiving antennas in BS. Based on the estimation, BS precodes the transmission signals at the downlink communication and forms the beam to the targeted UEs. For the system employing frequency division duplexing (FDD) scheme, the channel estimation is required for both uplink and downlink communication because the uplink and downlink communication will share the same signal bandwidth simultaneously, as shown in Fig. 1.4. Then the required number of pilots are proportional to the number of antennas [61]. The state-of-the-art channel estimation methods for FDD mode are compressed sensing based channel estimation [32] and time-correlation based channel estimation with a trellis-extended or an angle of departure-adaptive subspace codebook [18, 111]. However, those methods either require the full signal processing on the user side, or rely on accurate uplink estimation to simplify the downlink channel estimation procedures. Time division duplexing (TDD) scheme does not require this up/downlink acknowledgment due to the channel reciprocal principle, and the pilot requirement is also irrelevant to the number of antennas. For this reason, most of the massive MIMO systems uses TDD mode. The state-of-the-art channel estimators include least square [76] and minimum least square error (MMSE) [69] based channel estimation methods which are shown good estimation performance, especially under frequency-flat fading channels. The maximum likelihood estimator is also proposed to improve the inter- and intra-cell channel estimation performance with lower computational complexity than MMSE-based estimator [31]. However, the efficiency of the above estimation methods is based on either perfectly received signals or no interference in the channel, which cannot be true all the time.

Although accurate CSI estimation is vital to the performance of massive MIMO systems, getting a perfect CSI estimation is still challenging. Multiple factors, such as frontend distortions, pilot contamination, and jammings, can cause the imperfection and distort the estimation. In addition, CSI is also time-varying and has to be estimated within the channel coherence time. As a result, how to perform the channel estimation in a shorter time is essential as well.

The frontend non-ideality, such as IQ impairments, carrier frequency offsets (CFOs), sampling time offsets (STOs), and frontend non-linearity, can distort the training sequences or the pilot signals, which introduces interference and makes the channel estimation imperfect. In addition, for massive MIMO systems, to reduce the cost and energy consumption, simpler and lower-resolution ADCs can be used. Meanwhile, due to the resolution reduction and quantization error, longer pilots are needed to get accurate CSI, which will reduce the scalability of the system. In order to perform the CSI estimation, signal processing for the distortion calibration and compensation is necessary.

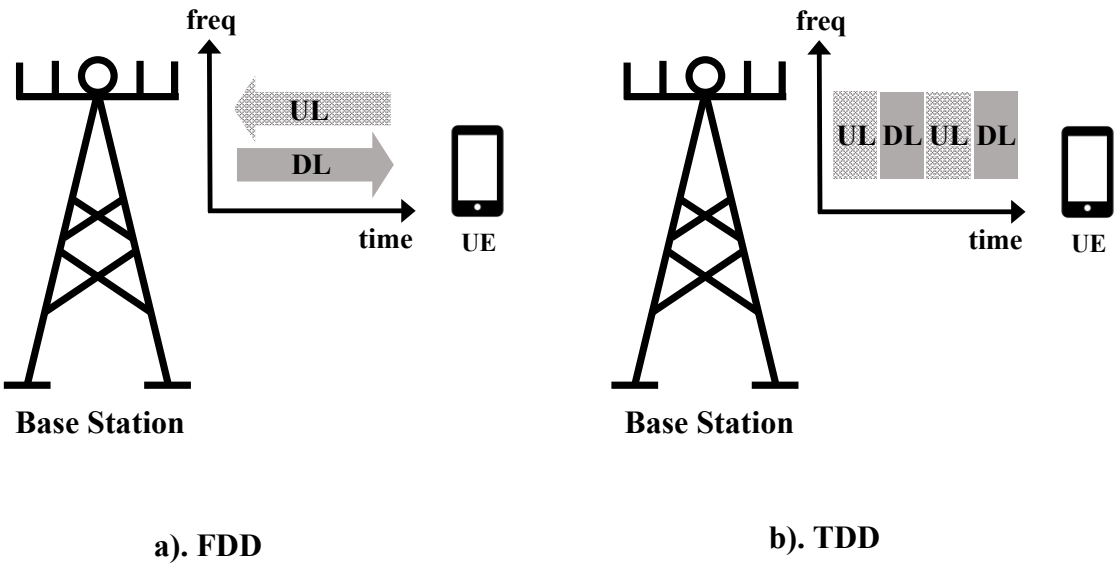


Figure 1.4: Communication methods: a). frequency division duplexing (FDD); b). time division duplexing (TDD).

Pilot contamination can also be a factor for inaccurate channel estimation. For the best channel estimation performance, orthogonal pilots are needed for individual UE in the same cell. However, the number of orthogonal pilots is limited in the specified bandwidth. As a result, UEs in the neighboring cells may reuse the same set of orthogonal pilots, and the BS may receive the same pilot from both the current cell and the neighboring cell. The same pilot sent from the neighboring cell will influence the channel estimation between the UE and the BS in the same cell.

Another factor that can impact the channel estimation is signal jamming or interference colliding in the same signal bandwidth. With the increasing number of devices, there can be some chances that the signal sent by other devices collides with the pilot signals, which interferes with the channel estimation. Signal jammings can also happen to block the normal communication between UEs and the BS. Some typical jammings that can affect the channel estimation are high-power, constant, or periodic signals sharing the same channel with the targeted UEs or the specifically designed signals that can destroy the channel estimation for the targeted pilots. As a result, getting a more accurate channel estimation in a noisy environment takes effort.

1.3 Machine Learning in Massive MIMO systems

With the increasing complexity in the channel environment and the hardware mismatch due to the array scaling-up, it is harder and harder to estimate the channel and perform

beamforming in the traditional way. More signal processing procedures are needed to calibrate and compensate for the channel loss.

It is the same for some other challenges existing in the current massive MIMO systems, such as pilot overhead reduction with the increasing number of UEs in the systems, the dynamic channel estimation caused by the UE mobility, the imperfect and nonlinear channel characteristics caused by the hardware impairment or environment, and the long computation latency due to the additional signal processing procedures. However, those challenges have the potential to be solved using a data-driven manner or, more specifically, machine learning methods. A neural network based joint pilot and downlink channel estimation design is proposed in [78] to improve the FDD channel estimation performance with reduced pilot overhead by using pruning. Authors in [45] propose a deep neural network based calibration framework to calibrate the nonlinear channel between uplink and downlink for both FDD and TDD massive MIMO systems. A deep learning based decision-directed channel estimation method is proposed in [80] to estimate channels of high-speed vehicles where different Doppler rates exist in different signal packets. In addition, a deep learning method, proposed in [70], is shown to be able to unfold an existing iterative massive MIMO detection method and simplify the signal processing.

The state-of-the-art machine learning methods have also proven to be efficient in improving the performance on the complex channel modeling, pilot contamination and codebook designs, coding and decoding scheme presentation, and CSI estimation for 5G and beyond communication networking [107, 127]. The top three commonly used machine learning architectures for the 5G and beyond communication applications are fully-connected networks, long short-term memory (LSTM) networks, and convolutional neural networks [107]. For the physical layer design of single-cell communication systems, the machine learning solutions to improve the CSI estimation are attractive, and multiple machine learning based CSI estimations are proposed to solve the challenges mentioned previously [115, 144, 63, 41, 10, 137]. The emergence of machine learning algorithms for massive MIMO signal processing shows the potential of the data-driven manner of the next communication systems.

1.4 Dissertation Overview

This dissertation focuses on integrated signal processing for massive MIMO systems. It provides a comprehensive analysis of different choices and considerations for designing and prototyping scalable massive MIMO systems from the perspective of both algorithms and hardware architectures. The design and prototyping of the scalable massive MIMO systems should focus on better data aggregation, linear scaling of the computational complexity, improving cost- and energy- efficiency, and maintaining development sustainability. This is achievable by designing modular, adaptable, and reusable massive MIMO systems to fit different beamforming techniques, baseband signal processing algorithms, and hardware component designs.

To further demonstrate the feasibility of designing such massive MIMO systems, an

algorithm-adaptable, scalable, and platform-portable generator for massive MIMO baseband processing systems is designed and emulated. This generator is written in Chisel [17] hardware construction language and produces instances that implement distributed massive MIMO BS processing, including channel estimation and beamforming. The generator can be reused for different MIMO systems and hardware datapath designs by changing the parameters. The generator is paired with a Python-based system simulator, which, incorporated together, can emulate a system testing various baseband signal processing algorithms. The field programmable gate array (FPGA) emulation is performed with generated instances using various parameter values. To demonstrate the algorithmic adaptability, a Golay-sequence-based channel estimation method, a beamspace calibration method [30], and a channel denoising algorithm [83] are evaluated across a range of channel models. The performance of the generator, the necessity of the algorithmic adaptability, and the ease of hardware generation are evaluated and discussed. The power consumption of the FPGA implementation is 5.4W, which is competitive with the state-of-the-art implementations with both channel estimation and beamforming implemented. The emulated register-transfer level (RTL) implementation with different system parameters shows that with beamspace methods, the demodulation error vector magnitude is improved by up to 29.8%.

As discussed in the previous section, ML plays an important role in the 5G and beyond communication systems. To further improve the massive MIMO system channel estimation performance under the nosier channel environments, a complex-valued neural network (CVNN) aided channel estimation is proposed. The CVNN takes the fast Fourier transformed (FFT) received Golay pilots as the input and outputs predicted Golay pilots. The correlation-based channel estimator then estimates the channel based on the prediction. The CVNN is trained with a wideband jamming signal centering at different frequencies within the pilot signal bandwidth. Different CVNN designs and training choices are compared and analyzed, and the channel estimation performance is compared with the traditional correlation-based estimator. The result shows that with the aid of CVNN, the channel estimation performance can be improved up to 3.16× with shorter pilots.

Chapter 2

Design and Prototyping of Scalable mmWave Massive MIMO Systems

2.1 Key components in massive MIMO systems

In this section, the key components in the algorithms of massive MIMO systems are discussed. Although the range of algorithms is large, in order to limit the scope of discussion, this thesis focuses mainly on modulation/demodulation-related physical layer signal processing topics in single-cell massive MIMO communication systems. Other important algorithms, such as encoding/decoding and MAC layer scheduling, are out of the discussion range of this dissertation. The main components in the massive MIMO physical layer algorithms include beamforming and baseband signal processing, as shown in Fig. 2.1. Different techniques and algorithms will be summarized and discussed on digital vs. hybrid methods used in beamforming, signal recovery, channel estimation, and signal synchronization algorithms used in the baseband signal processing.

2.1.1 Beamforming Techniques

In order to improve the communication quality, beamforming techniques, which combine and focus signals from the array on a specific direction and reduce the response of signals coming from other directions, are employed especially in multiple user scenarios [39]. The widely used beamforming techniques include digital beamforming and hybrid digital and analog beamforming. The summary of the beamforming techniques can be found in Table. 2.1.

Digital Beamforming

In digital beamforming, the weighted sum of the vectors of the input signal needs to be used to form the beam [99]. It has the advantages of high precoding freedom, precise beamforming, multiple beam flexibility, etc, but is also power-consuming and expensive due to the large number of radio frequency (RF) chains required [136]. In digital beamforming

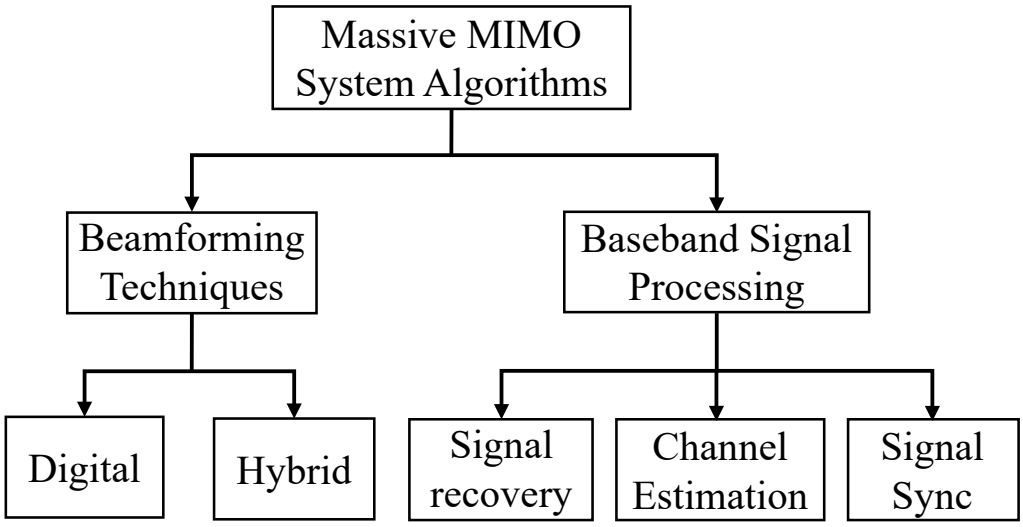


Figure 2.1: Key signal processing algorithms in massive MIMO physical layer, assuming single-cell communication systems.

architectures, the number of RF chains is the same as the number of the antenna array because all the operations are performed after the analog-to-digital conversion or before the digital-to-analog conversion. The traditional digital beamforming techniques include maximum ratio combining (MRC), zero-forcing (ZF), and minimum mean square error (MMSE) based methods [89]. The following discussion is based on uplink communication, and downlink beamforming can be naturally derived using the reciprocal property assuming TDD mode. The main goal of maximum ratio combining is to maximize the received signal-to-noise ratio (SNR). Assume the system, with K users and M antennas at the base station, has the knowledge of perfect channel state information (CSI), and the channel noise is the independently and identically distributed (i.i.d) additive white Gaussian noise (AWGN). Let $method \in \{MRC, ZF, MMSE\}$, $H_{method} \in \mathbb{C}^{M \times K}$ being the channel matrix with $h_{ij}, i \in M, j \in K$ as each element, $D_{method} \in \mathbb{C}^{K \times 1}$ being the beamformed signal with $d_k, k \in K$ as each element, $\boldsymbol{x} \in \mathbb{C}^{K \times 1}$ being the transmitted signal with $x_k, k \in K$ as each element, $\boldsymbol{y} \in \mathbb{C}^{M \times 1}$ being the received signal as $y_i, i \in M$ as each element. MRC [89] can be expressed as

$$D_{MRC} = \boldsymbol{\alpha}^{-1} H^H \boldsymbol{y}, \tag{2.1}$$

where $\boldsymbol{\alpha} = \text{diag}\{\sum_{i=1}^{M} |h_{i1}|^2, ..., \sum_{i=1}^{M} |h_{iK}|^2\}$.

If we expand the formula, we can get

$$\begin{aligned} d_{k,MRC} &= \frac{1}{\sum_{i=1}^{M}|h_{ik}|^2}\sum_{i=1}^{M} h_{ik}^* y \\ &= x_k + \frac{1}{\sum_{i=1}^{M}|h_{ik}|^2}\sum_{i=1}^{M}\sum_{j\neq i}^{K} h_{ik}^* h_{ij} x_j + \frac{1}{\sum_{i=1}^{M}|h_{ik}|^2}\sum_{i=1}^{M} h_{ik}^* n_k, \end{aligned} \tag{2.2}$$

with n_k being the i.i.d AWGN. For the massive MIMO system, where a significantly large number of antennas can be assumed, the user-base station channel vectors show the pairwise orthogonality [9]. However, in reality, since the number of antennas is limited, the channel vectors are not orthogonal with each other, which introduces inter-user interference (IUI) as shown in Eq (2.2). To remove IUI, zero forcing beamforming is used. Instead of using the hermit conjugate of the channel matrix, the zero forcing method employs the pseudo-inverse of the channel matrix, and can be expressed as [89]

$$D_{ZF} = (H^H H)^{-1} H^H \boldsymbol{y} = \boldsymbol{x} + (H^H H)^{-1} H^H \boldsymbol{n}, \tag{2.3}$$

by expanding the formula, and let $w_{ij,ZF}, i, j \in K$ be the elements in $W = (H^H H)^{-1}$ and $(H^H \boldsymbol{n})_j$ be the j-th element in the noise vector, we can get

$$d_{k,ZF} = x_k + \sum_{j=1}^{m} w_{kj,ZF}(H^H \boldsymbol{n})_j. \tag{2.4}$$

Compared with Eq (2.2), Eq (2.4) doesn't have the IUI components, but at the same time, the noise got amplified, which is also shown in Fig. 2.2 when the number of antennas is 64, and SNR is lower than -18dB. Authors in paper [74] also explore the impact of frontend imperfection on MRC and ZF beamforming. The result shows that with a large number of antenna arrays, the impact of the IQ imbalance on the overall detection performance is minor, which also shows the efficiency of the linear receiver design in massive MIMO systems.

Another typical digital beamforming technique is the MMSE-based beamforming, where the noise effect will be minimized. In another word, find the MMSE-based weights $W_{MMSE} \in \mathbf{C}^{K\times M}$ that $\mathbb{E}\{(W_{MMSE}\boldsymbol{y} - \boldsymbol{x})\boldsymbol{y}^H\} = 0$. Then, one can derive that

$$\begin{aligned} W_{MMSE} &= \mathbb{E}\{\boldsymbol{x}\boldsymbol{y}^H\}\mathbb{E}\{\boldsymbol{y}\boldsymbol{y}^H\}^{-1} \\ &= \mathbb{E}\{\boldsymbol{x}(H\boldsymbol{x}+\boldsymbol{n})^H\}\mathbb{E}\{(H\boldsymbol{x}+\boldsymbol{n})(H\boldsymbol{x}+\boldsymbol{n})^H\}^{-1} \\ &= (\mathbb{E}\{\boldsymbol{x}\boldsymbol{x}^H\}H^H)(\mathbb{E}\{H\boldsymbol{x}\boldsymbol{x}^H H^H + H\boldsymbol{x}\boldsymbol{n}^H + \boldsymbol{n}\boldsymbol{x}^H H^H + \boldsymbol{n}\boldsymbol{n}^H\}^{-1}) \\ &= (\mathbb{E}\{\boldsymbol{x}\boldsymbol{x}^H\}H^H)(H\mathbb{E}\{\boldsymbol{x}\boldsymbol{x}^H\}H^H + \mathbb{E}\{\boldsymbol{n}\boldsymbol{n}^H\})^{-1}. \end{aligned} \tag{2.5}$$

Assume $P_{tx} = \mathbb{E}\{\boldsymbol{x}\boldsymbol{x}^H\}$ be the transmission power, and $\boldsymbol{n} \sim \mathcal{CN}(0, \sigma^2 \boldsymbol{I}_M)$. The Eq. 2.5 can be expressed as

$$\begin{aligned} W_{MMSE} &= \mathbb{E}\{\boldsymbol{x}\boldsymbol{y}^H\}\mathbb{E}\{\boldsymbol{y}\boldsymbol{y}^H\}^{-1} \\ &= (P_{tx}I)H^H(H(P_{tx}I)H^H + \sigma^2 I)^{-1} \\ &= H^H(HH^H + \frac{\sigma^2}{P_{tx}}I)^{-1}. \end{aligned} \tag{2.6}$$

MMSE beamforming can be expressed as

$$D_{MMSE} = H^H(HH^H + \frac{\sigma^2}{P_{tx}}I)^{-1}\boldsymbol{y}. \tag{2.7}$$

Compared with ZF method, MMSE-based method has a minimum noise effect, and has the best beamforming performance among the three digital beamforming techniques under the current assumption. In addition, the MMSE-based method can be extended to other scenarios. The above MMSE-based method only considers a single-cell communication scenario. Authors in paper [69] proposed a multi-cell MMSE (M-MMSE) based scheme to suppress both intra-cell and inter-cell interference (ICI) by estimating the inter-cell channel information and fusing the estimated inter-cell channel matrix into Eq. (2.7). M-MMSE method can further improve the performance of MMSE-based beamforming even with inter-cell interference. For multiple-user antenna scenarios, MMSE digital beamforming method is not optimal anymore. As a result, block-diagonalization based digital beamforming is proposed to achieve a better performance for multiple-user antenna beamforming [116, 118]. However, due to the need for matrix inversion in ZF and MMSE-based beamforming, the computational complexity of ZF and MMSE methods is much higher than MRC beamforming, which also makes the system hard to scale up. Fig. 2.2 shows the beamformed symbol MSE vs. SNR for different beamforming methods with different numbers of base station antennas, and the results match the discussion in this section.

However, as the number of base station antennas increases, ZF and MMSE-based methods become increasingly unrealistic to implement due to the high computational complexity and the high data transferring bandwidth requirement. To keep good beamforming performance and improve scalability at the same time, digital beamforming needs to be distributed. The distribution can either be partially distributed or fully distributed. Partially distributed digital beamforming, such as a two-stage beamforming method [113, 60] combining MRC and ZF, can reduce data transmission between antenna clusters and the centralized processor, while a fully distributed method can almost remove the centralized processing or require only a small amount. Coordinate descent (CD) [100] and alternating direction method of multipliers (ADMM) [67] methods are proposed to approximate ZF beamforming in fully decentralized ways. The CD method uses the gradient descent method to find the unconstrained least square solution iteratively. The beamforming weights associated with each antenna are derived in the antanna order distributively, and the update step size is used for controlling the IUI removal amount at each step. Unlike the CD method, where the beamforming weights of each antenna are solved individually, ADMM-based beamforming provides the transmission symbols for each antenna cluster. ADMM-based beamforming aims to find the transmission symbols that can both minimize the transmitting energy and make the interference residual within the precoding constraint by using only local knowledge at each cluster. Both methods can enable the scalability of the system deploying digital beamforming.

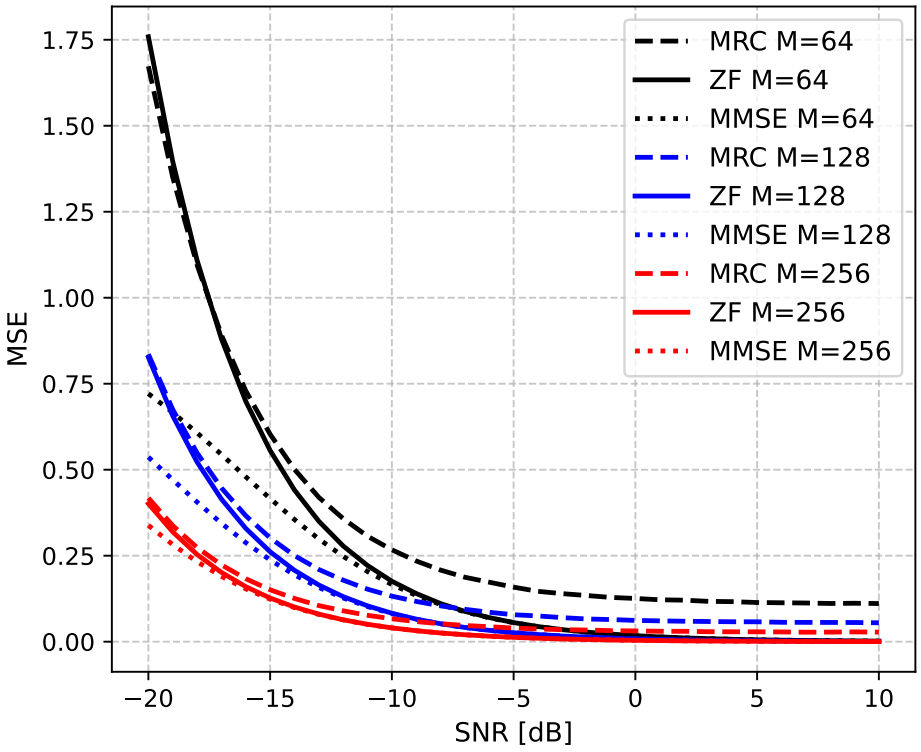


Figure 2.2: The beamforming performance comparison among MRC, ZF, and MMSE-based with a different number of antennas M in the system. The simulation considers 8 UEs in the system under flat i.i.d channel model and AWGN, and uses 64-QAM for symbols.

Hybrid analog-digital beamforming

To avoid costly frontend hardware designs and still keep high beamforming quality, hybrid analog-digital beamforming architectures are widely used. Unlike digital beamforming architecture, hybrid beamforming architectures use less number of RF chains or analog-digital converters than the number of antennas, which is feasible because data stream number constraints the RF chain or analog-digital converter number, but the gain of beamforming comes from the number of antennas [84]. The analog beamforming in the hybrid beamforming can be implemented in either the RF domain or the baseband domain. For the phase shifter-based analog beamforming implemented in the RF domain, shown in Fig.2.3 (a), the phase shifting can cancel undesired components received by the antenna array before any other distortions are added, such as the dynamic range saturation and the distortion coming from mixers. But due to the high operating frequency, the beamforming performance is sensitive to the circuit fabrication nonideality, which leads to inaccurate phase shifting. The vulnerability of the analog beamforming can be relieved by implementing phase shifters in the baseband, shown in Fig.2.3 (b). However, each antenna requires a mixer, making the circuit, especially

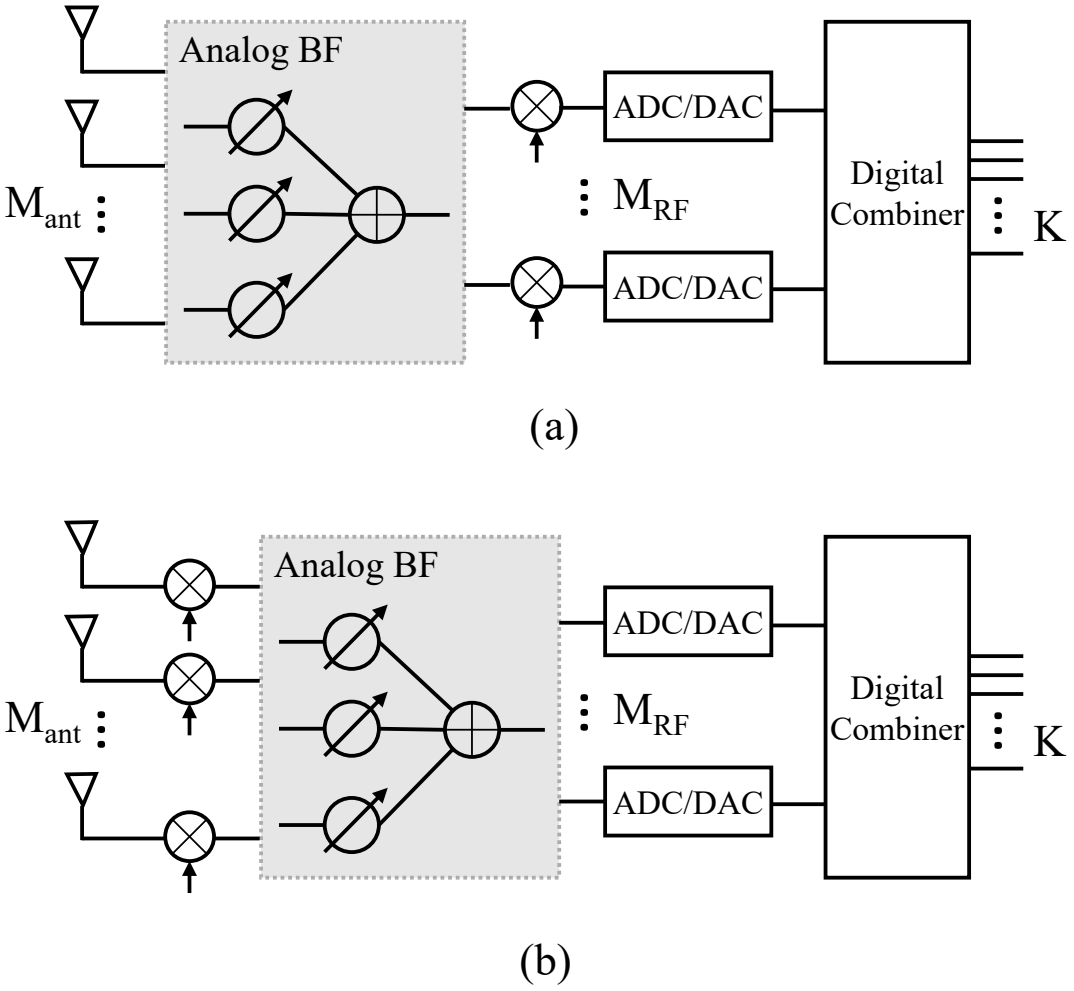


Figure 2.3: Hybrid analog-digital beamforming: (a) RF domain hybrid beamforming; (b) baseband hybrid beamforming.

the LO distribution, more complicated [21]. In hybrid architectures, both digital baseband processing and analog phase shifters are used, and can be categorized according to the phase shifter connection. The types of hybrid architecture include fully connected phase shifter structure, partially connected phase shifter structure, and switch connected structure [49, 84, 81]. Compared with the partially connected phase shifter structure, a fully-connected phase shifter network can get the full antenna array gain, while may suffer from the coupling issue and user interference. Phase shifter-based structure designs have a wide range of design considerations, such as noise figures, circuit linearity, and phase resolutions, but switch-based designs mainly need to consider the switching speed. The low power consumption of switches also makes the switched-based structure attractive. However, the performance of switch-based

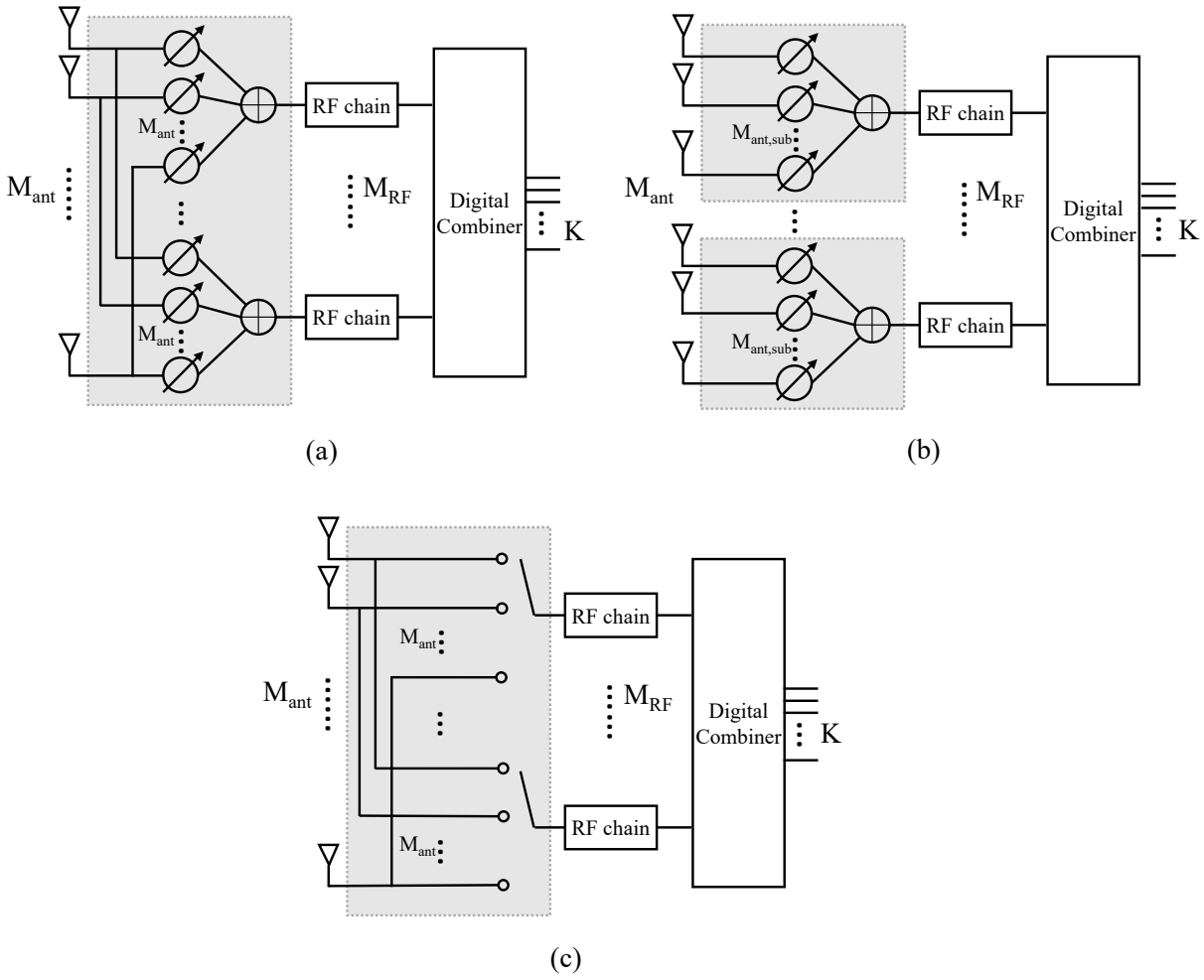


Figure 2.4: Hybrid analog-digital beamforming: (a) fully connected phase shifter-based architecture; (b) partially connected phase shifter-based architecture; and (c) switch-based architecture.

analog beamforming is vulnerable to the noncoherent signal combination, where the combined signals have different phases [81]. The performance of the switch network can be improved by combining it with phase shifter-based networks.

However, due to the split of the digital beamforming stage and the analog beamforming stage, the optimal beamformer matrix for digital and analog is hard to find. In addition, the analog beamforming structure adds more constraints on the optimization function to find the optimal analog beamforming matrix [84]. Due to the ON-OFF constraints in the beamforming matrix for the switch-based architecture, designing the optimal combiner is even harder than the phase shifter-based analog beamforming [81]. Because of the difficulty

of finding the optimal solution, the performance of the hybrid beamforming can be worse than the digital beamforming techniques. The author in paper [143] compare the performance between MMSE hybrid beamforming and MMSE fully digital beamforming with different resolutions of ADC, and the result shows that MMSE fully digital beamforming outperforms the hybrid beamforming even with a smaller antenna array size.

To reduce the performance gap between hybrid beamforming and fully digital beamforming, many optimization methods are proposed. One proposed method is by using minimal gap iterative quantization and greedy ratio trace maximization to find the optimal analog combiner solution that can minimize the mean square error between the analog combiner and the corresponding optimal digital combiner solution, which is proposed by authors in paper [49]. For a multiple-user antenna scenario, authors in paper [12] use digital block-diagonalization beamforming as a reference, and employs weighted sum mean square error minimization to find the hybrid analog and digital beamforming matrix that minimize the performance gap between the reference digital beamforming and the hybrid beamforming. An optimization method that uses the low complexity Gram-Schmidt method and MMSE method to obtain the analog beamforming matrix and the digital beamforming matrix is also proposed in [66], where the inter-user interference is reduced at both stages. Besides improving the optimization procedure, authors in paper [87] propose a hybrid beamforming design in the time domain to improve the hybrid beamforming performance. With the special analog phase matrix design, the wideband signal can be perfectly reconstructed at the sampling points for the equivalent fully digital beamforming design, and the hybrid beamforming performance can achieve that of digital beamforming.

Besides hybrid analog-digital beamforming, adaptive beamforming techniques are developed to form more flexible beams and suppress the interference and noise [99]. The widely used methods include least mean square, normalized least mean square method, and recursive least square, which can adaptively approximate the optimal beamforming [5].

2.1.2 Baseband signal processing

Fig. 2.5 shows the key components in massive MIMO baseband digital signal processing, which include the frontend impairment correction, the channel estimation, the signal time-domain and frequency-domain synchronization, and the digital beamforming. Baseband signal processing also includes symbol modulation/demodulation and data encoding/decoding, but in this dissertation, we will mainly focus on the discussion on frontend impairment, channel estimation, and signal synchronization.

Frontend Impairment Correction

Imperfections in the manufacturing process or deployment scenarios can cause imbalanced amplitude and phase in the in-phase and quadrature-phase (IQ) channels of the frontend design, which can severely affect the bit error rate (BER) versus SNR in MIMO systems [110, 120]. To estimate IQ imbalance parameters, traditional methods such as least square

Table 2.1: Beamforming technique summary.

BF Method			**Pros**	**Cons**
Digital		MRC	low complexity high scalability	contain IUI
		ZF	no IUI	high complexity low scalability noise amplified
		MMSE	minimum noise effect remove ICI	high complexity low scalability power estimation
		Multi-stage	reduce data transfer reduce complexity	extra ChEst
		CD or ADMM-based	fully decentralized high scalability	iterative method longer latency
		Overall	easy algorithm dev optimal solution	power consuming
Hybrid	Domain	RF	less distortion	less phase accuracy
		Baseband	higher phase accuracy	more distortion require mixer
	Network	Fully connected	full array gain	complex connection coupling issue low scalability
		Partially connected	simpler connection high scalability	more BF constraints less array gain
		Switch	simplest connect power efficient	high vulnerability high BF constraints
	Overall		power saving	suboptimal solution longer design cycle
Adaptive			flexible beams	iterative method

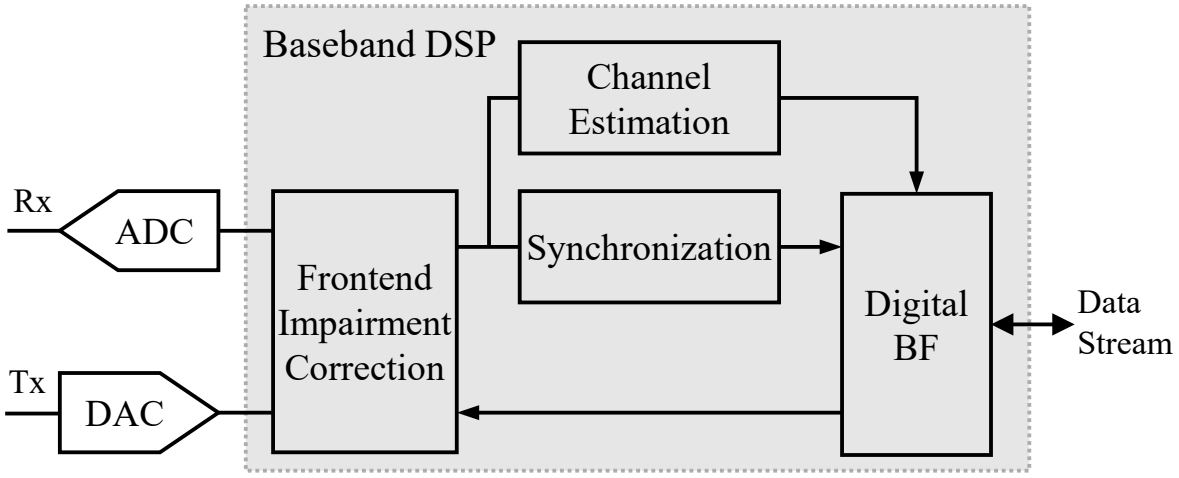


Figure 2.5: Key components in the baseband digital signal processing.

and minimum mean square equalization in the frequency domain have been proven effective for OFDM-modulated signals. IQ imbalance parameters can also be estimated using the statistics of the signal. By calculating the auto-correlation and cross-correlation of the QAM signals, the gain difference and phase rotation between the I channel and the Q channel can be found. To provide robust IQ imbalance parameter estimation in the time domain, a pre-FFT adaptive distortion correction method was proposed in [120] to correct IQ before FFT operation.

One of the properties of millimeter-wave (mmWave) massive MIMO systems is the wide bandwidth of the signal, which can lead to incoherent frequency response and frequency-dependent IQ imbalance. To address this issue, a technique for estimating IQ correction parameters was proposed in [131], which involves dividing the wideband into multiple subbands and estimating the frequency-dependent parameters using a least square algorithm. Another method presented in [106] uses a time-varying training sequence, which is designed and scheduled to guarantee no interference on active subcarriers, to find the LS optimal solution for frequency-selective IQ imbalance estimation in wideband OFDM signals. As the number of antennas increases, the frontend design can be less expensive as the ADC resolution can be lower. However, lower-resolution ADCs may suffer from IQ imbalance and nonlinear quantization loss. A two-stage IQ parameter estimation method is proposed in [129] to solve this issue, where the uplink iteratively estimates a phase-shifted version of the parameters by gradient descent and the downlink corrects the estimated parameters' phase.

Another imperfection in the frontend design is the DC signal offset, which can cause unwanted carrier leakage. To remove DC offset, the commonly used method is to eliminate the mean of the received signal generated by a single tone from an ideal transmitter. However, manufacturing mismatches between the frontend chains in the array for beamspace algorithms

can cause the channel to be non-sparse, leading to a breakdown in beamspace algorithms. To address this issue, the authors of [30] proposed an in-the-field MIMO array calibration method that estimates the array phase offsets due to manufacturing mismatches and recovers channel sparsity.

Channel Estimation

Time domain methods In a narrow-band TDD mode massive MIMO systems, a frequency flat fading channel model is typically considered, and the channel information can be estimated using special pilot designs such as Zadoff-Chu sequences, Golay sequences, and Golden code sequences, based on the autocorrelation results. Least square (LS) and minimum mean square error (MMSE) estimators are widely used methods for channel estimation [108, 138]. Assume that the modeled channel impulse response have R taps with $\boldsymbol{h}_{TDD} = [h[0], h[1], ..., h[R-1]]^T \in \mathbb{C}^{R\times 1}$, the pilot symbol vector have L length $\boldsymbol{s} = [s[0], s[1], ...s[L-1]] \in \mathbb{C}^L$, and noise vector $\boldsymbol{n}_{TDD} \in \mathbb{C}^{(L-R)\times 1}$. Then the signal received at each antenna $\boldsymbol{y}_{TDD} = [y[0], y[1], ..., y[L-R-1]]$ can be formulated as

$$\begin{bmatrix} y[0] \\ y[1] \\ \vdots \\ y[L-R-1] \end{bmatrix} = \begin{bmatrix} s[0] & & & \\ s[1] & s[0] & & \\ \ddots & & & \\ s[L-1] & s[L-2] & \cdots & s[L-R-1] \end{bmatrix} \boldsymbol{h}_{TDD} + \begin{bmatrix} n[0] \\ n[1] \\ \vdots \\ n[L-R-1] \end{bmatrix} \tag{2.8}$$

And it can be simplified as

$$\boldsymbol{y}_{TDD} = S_{TDD}\boldsymbol{h}_{TDD} + \boldsymbol{n}_{TDD}. \tag{2.9}$$

The LS estimator $\hat{\boldsymbol{h}}_{\boldsymbol{LS}}$ can be expressed as

$$\hat{\boldsymbol{h}}_{\boldsymbol{LS}} = (S_{TDD}^H S_{TDD})^{-1} S_{TDD}^H \boldsymbol{y}_{TDD}. \tag{2.10}$$

And the MMSE estimator $\hat{\boldsymbol{h}}_{\boldsymbol{MMSE}}$ can be expressed as below, assuming that $R_h = \mathbb{E}\{\boldsymbol{h}_{TDD}\boldsymbol{h}_{TDD}^H\}$ and $R_n = \mathbb{E}\{\boldsymbol{n}_{TDD}\boldsymbol{n}_{TDD}^H\}$

$$\hat{\boldsymbol{h}}_{\boldsymbol{MMSE}} = (R_h^{-1} + S_{TDD}^H R_n^{-1} S_{TDD})^{-1} S_{TDD}^H R_n^{-1} \boldsymbol{y}_{TDD}. \tag{2.11}$$

The above two estimators are localized channel estimators, where the channel estimation is performed at each antenna locally. The MMSE channel estimator can also be extended to the centralized MMSE (C-MMSE) channel estimator [138] by

$$\hat{\boldsymbol{h}}_{\boldsymbol{C-MMSE}} = (R_H^{-1} + S_{C,TDD}^H R_n^{-1} S_{C,TDD})^{-1} S_{C,TDD}^H R_n^{-1} \boldsymbol{Y}_{TDD}, \tag{2.12}$$

where $H = [\boldsymbol{h}_0^T, \boldsymbol{h}_1^T, ..., \boldsymbol{h}_{M-1}^T]^T$, $R_H = \mathbb{E}\{HH^H\}$, and $S_{C,TDD} = I_M \otimes S_{TDD}$.

Fig. 2.6 shows the simulation results of the above three channel estimation methods. Compared with LS and L-MMSE channel estimation, M-MMSE channel estimation has the

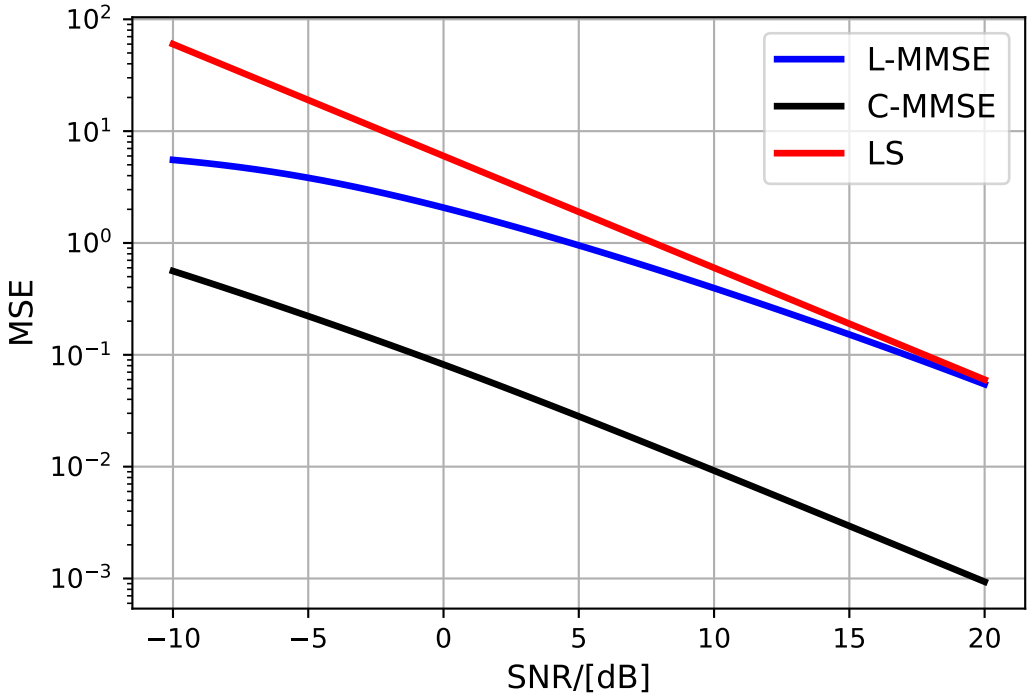


Figure 2.6: Channel estimation performance comparison among L-MMSE, C-MMSE, and LS. The metric is the mean square error (MSE) vs. per symbol SNR, assuming multipath Rayleigh fading with 3 taps channel impulse response, 32 pilots, 64 antennas in an equal spacing linear array, and additive white Gaussian noise (AWGN).

least estimation MSE under AWGN channel, even in the low SNR range, but requires more statistical information for the channel.

To address inter-cell pilot contamination and improve channel estimation performance with lower computation than MMSE, the maximum likelihood estimator was proposed by the author in [31].

Frequency/beamspace/angular domain methods MMSE and LS channel estimators can also be applied in the frequency domain by using FFT and converting the OFDM pilot into the frequency domain. In [138], the performance and complexity comparison between the localized MMSE (L-MMSE) and C-MMSE under AWGN channel model are established. And it shows that C-MMSE has a lower channel estimation mean square error but much higher computational complexity than L-MMSE. This leads to the proposal of a distributed C-MMSE channel estimation method in the frequency domain to reduce the computational complexity to the same magnitude as L-MMSE with better performance by combining L-MMSE estimations iteratively to approach the global optimal solution.

In practice, the mmWave channel is sparse, which means the number of multipath components in the environment is limited and has been proved with measurements in [42]. To exploit the wireless channel sparsity in mmWave with higher estimation accuracy and less pilot overhead, compressed sensing (CS) techniques have been favored recently, aiming to reconstruct the compressible signal with far less sampling rate than Nyquist's sampling rate [20, 73]. Some optimization algorithms to solve this underdetermined matrix with the minimum L_1 norm include matching pursuit, tree matching pursuit, and orthogonal matching pursuit (OMP) [7]. Compared with matching pursuit, OMP can achieve faster convergence. Various channel estimation methods based on OMP are proposed to solve different difficulties in massive MIMO channel estimations [98, 64, 132, 40, 34]. An OMP-based hybrid far-near channel estimation for extremely large-scale massive MIMO (XL-MIMO) is proposed to estimate both far-field and near-field channel path components more accurately, which is hard for traditional methods because the scatters in the far and near field are different. The basis mismatching problem that happens in the discretized CS methods can also be avoided using Newtonized OMP (NOMP) [119]. [34, 40] use NOMP to extract massive MIMO uplink frequency-independent channel information with high accuracy. Authors of [34] deploy joint NOMP to reduce the computational complexity further. For a large number of antenna array systems, the nonzero positions of the channel impulse responses from different antennas can be the same. Better utilization of the channel joint sparsity can reduce the pilot overhead a lot. For this reason, blocked optimized OMP is proposed in [98] to jointly recover the nonzero channel matrices, which also outperforms the localized OMP method.

As the number of antennas for massive MIMO base stations increases, energy consumption and hardware complexity become important considerations. To address these issues, beamspace MIMO has gained attention because it reduces the requirement for a large number of RF chains [13, 22]. In a recent paper [22], the authors proposed a beamspace channel estimation algorithm with lens array, which utilizes a supporting detection (SD) approach. This method achieves higher accuracy than traditional compressive sensing (CS) techniques such as OMP, particularly in low SNR scenarios, while maintaining lower computational complexity. Another paper [83] introduced a denoising-based beamspace channel estimation method that relies on Stein's unbiased risk estimator, taking advantage of the directional nature of wave propagation at mmWave frequencies. The low computational complexity of this algorithm facilitates hardware implementation.

Other channel estimation methods, such as approximate message passing (AMP) [101, 46, 11, 145], can achieve higher performance, especially for the hybrid beamforming architectures, but they rely on complex assumptions about the prior distribution. Consequently, implementing these channel estimation methods on hardware can be challenging.

Machine-learning based estimation Most traditional channel estimation methods treat each antenna path separately, which fails to utilize the geometric information of the antenna array. To improve channel estimation performance in more challenging scenarios, machine learning techniques are being applied to channel estimation. For instance, a recent paper

Table 2.2: Channel estimation method summary.

Category	Method	Example
Time Domain	LS	[108]
	MMSE	[138]
	Maximum likelihood	[31]
Frequency/Beamspace /Angular Domain	LS/MMSE	[138]
	OMP	[7, 98, 64, 132, 40, 34]
	Beamspace	[13, 22]
	AMP	[101, 46, 11, 145]
ML-based	DNN	[115, 144, 63, 41]
	Multilayer-perceptron	[10]
	RNN	[137]

[115] proposed a deep neural network (DNN) denoising-based channel estimation method for OFDM systems. The network takes the LS estimation from each antenna as input. It produces higher-resolution channel estimation, which outperforms traditional LS and approximate linear MMSE (ALMMSE) solutions in low SNR scenarios with additive white Gaussian noise (AWGN). Another paper [144] introduced a DNN-based channel estimation method for 1-bit ADC massive MIMO systems. Rather than estimating the channel for each antenna separately, this method takes all antenna signals in the array as input. In a different approach, a paper [10] proposed a multi-layer perceptron channel estimation method that exploits similarities in channel dynamics across spatial dimensions. The computational cost of this method is fixed, regardless of the size of the antenna array, making it suitable for implementation at the edge. In another paper [63], the author used three different deep learning networks to refine the per-antenna LS channel estimation solution, and the results demonstrated the robustness of the system to Doppler effects. Meanwhile, another paper [137] proposed a CNN-autoregressive predictor and CNN-recurrent neural network (RNN) to improve channel estimation performance with channel aging, compared with using only an autoregressive predictor. Furthermore, in addition to spatial and frequency domains, another paper [41] proposed a learned denoising-based AMP network for beamspace mmWave massive MIMO systems to achieve better performance with a small number of RF chains.

A summary of channel estimation methods for massive MIMO systems is shown in Table. 2.2.

Signal Time and Frequency Synchronization

To achieve higher channel capacity and better energy and spectral efficiency, mmWave massive MIMO with OFDM signal modulation is widely considered as a promising approach. However, to enable the advantages of massive MIMO systems, accurate synchronization is important, especially for wideband signals. Massive MIMO synchronization includes timing synchronization and frequency synchronization. Sampling timing offsets (STOs) will introduce inter-symbol interference (ISI) and frequency-dependent phase rotation to subcarriers in the frequency domain, and carrier frequency offsets (CFOs) will destroy the orthogonality of subcarriers, involving inter-carrier interference (ICI). The imperfection of signal synchronization consequently degrades the massive MIMO system performance, regardless of whether the antenna model is collocated or distributed [16].

Traditional synchronization methods for OFDM communication include Moose's algorithm [86] and Schmidl-Cox algorithm [109], which are based on the autocorrelation result of two identical training sequences. However, for multi-user massive MIMO systems, each pair between transmitter and receiver antenna needs to be estimated. As the number of antennas at the base station increases, the computational complexity for signal synchronization will increase drastically. To reduce the amount of computation, authors in paper [105] propose an estimation strategy which firstly estimates frequency offsets between the reference antenna and all other antennas in BS, and then estimates CFOs between the reference antenna and all users. With the prior knowledge of BS antennas, CFOs of each user-BS antenna pair can be derived, and no need to estimate all user-BS antenna CFOs individually. Traditional CFO estimation methods need repetitive training sequences. To reduce the overhead and CFO compensation complexity, authors of paper [104] proposed a frequency synchronization method that only needs one training symbol and compensates CFOs after the antenna signal combining. Based on the assumption that the number of BS antennas is large, the algorithm finds the CFO estimation minimizing the power of the orthogonal projection of the received signal onto the space spanned by all users' training sequences. Besides frequency domain synchronization methods, authors in paper [142, 35] proposed an angle domain adaptive filtering (ADAF) based frequency synchronization method to estimate and compensate CFOs. The author in paper [35] further improved the ADAF method to remove the constraint in [142] that users cannot have overlapped angle-of-arrival regions.

To get a good communication system performance, STO should be estimated accurately as well. Besides the traditional auto-correlation based STO estimation methods in [86, 109], the author in paper [141] applied selection transmitting and maximum ratio combining scheme to estimate timing offsets with the help of a designed training sequence. Authors of paper [51] also comprehensively analyze the impact of residual STO and residual CFO on the performance of highly quantized MIMO-OFDM systems, and show that simple synchronization techniques are sufficient even for highly quantized systems.

Recently, researchers are also deploying machine learning to improve the accuracy of timing and frequency synchronization. Authors of paper [134, 93] use deep learning based timing or frequency offset estimation to achieve better performance. However, due to the

property of deep learning networks, the computational complexity is large. To reduce the complexity and maintain the high accuracy of ML-based algorithms, the author in paper [72] estimated the residual STO and residual CFO by using the extreme learning machine (ELM) approach, which incorporates the traditional STO method. Compared with the deep learning methods in [134, 93], ELM approach doesn't need the perfect information about channels, which makes it more practical.

2.2 Massive MIMO prototyping

2.2.1 System Topology

Centralized vs. distributed massive MIMO testbeds The traditional topology of massive MIMO base station implementation employs the centralized architecture, where the radio frontends receive the signals and transmit to the central compute unit. Then the central computing unit performs the baseband signal processing, beamforming, and demodulation. The simplicity of the architecture makes the design of the base station straightforward, where only the off-the-shelf radio frontends, interconnection modules, and a PC are theoretically needed. An example of the centralized massive MIMO base station implementation is LuMaMi [75]. LuMaMi base station prototype is built with software-defined radios for OFDM transceiving, the switch and router system, centralized co-processors, and a central control unit. The baseband signal processing, such as channel estimation and beamforming, is performed in the centralized co-processors. The router system is specially designed to support a large amount of data transmission between the SDR frontends and the centralized processing unit. However, as the number of antenna arrays increases, due to the large amount of data transmission and data processing, the interconnect router design and the baseband signal processing become increasingly complex, contributing to high power consumption. In addition, the increase of those complexity leads to slow computation, and in the end, the computation time exceeds the channel coherent time. The high power consumption, high computational complexity, and low scalability make the centralized architecture impractical for the real-time massive MIMO base station implementation.

To relieve the burden of interconnection and computational complexity, the base station architecture has to be distributed. Instead of implementing complex router designs and centralized computation units, some parts of the baseband signal processing can be implemented distributively near frontends. With the help of localized signal processing, less amount of data needs to be aggregated or disaggregated, and the computational complexity of the central computing unit can also be reduced. An example of the distributed massive MIMO base station implementation is Argos [112]. Argos consists of 16 WARP modules containing radio daughter boards and FPGA for signal processing, a hub for data switch and clock distribution, and a central controller for centralized signal processing. In Argos, channel estimation and beamforming are performed in the distributed WARP modules. As a result, for downlink beamforming, only CSI and the calculated downlink beamforming weights

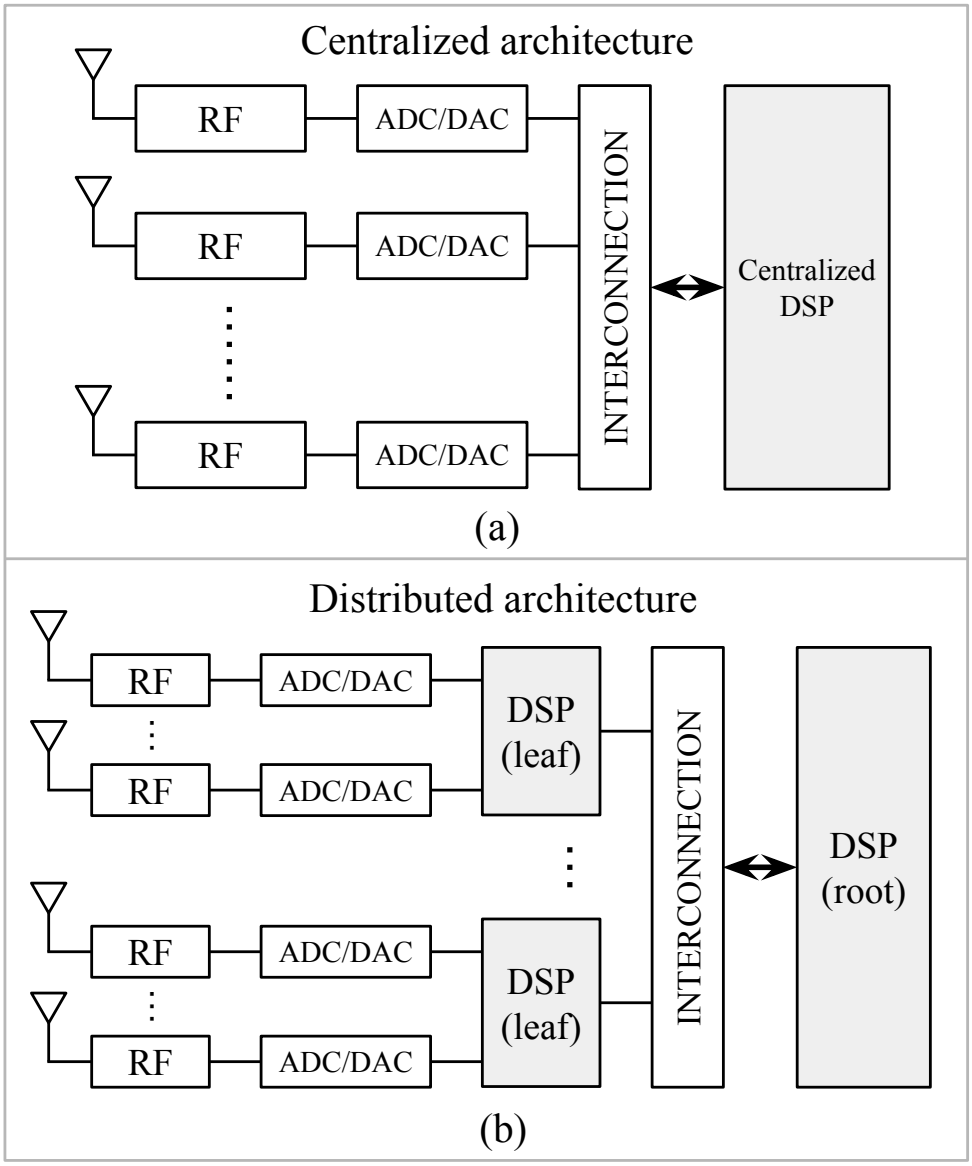


Figure 2.7: Massive MIMO base station topology: (a) centralized architecture; and (b) decentralized architecture.

need to be transferred between the centralized computing unit and WARP modules, and the transmission symbols need to be distributed to all WARP modules equally. The relaxation of the interconnection bandwidth and computational complexity makes the distributed base station architecture more power efficient and scalable.

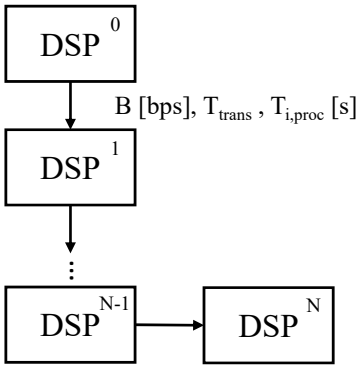
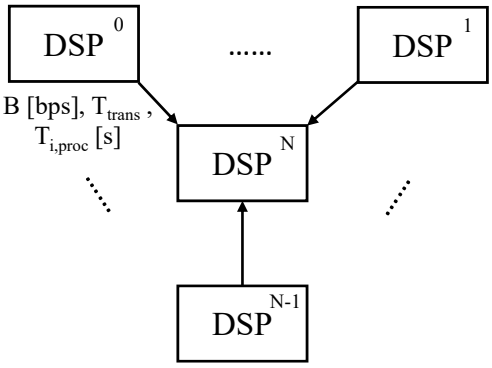
	Daisy-chain Network	**Star Network**
		
Latency [s]	$\sum_{0}^{N-1} T_{i,proc} + T_{N,proc}$	$\max\{T_{i,proc}\} + T_{N,proc}$
Bandwidth [bps]	B	NB
Buffer/Memory size required [bit]	i=0… N-1: $iT_{trans}B$ i=N: ideally 0	i=0… N-1: ideally 0 i=N: $T_{total}(NB - B_{N,out})$

Figure 2.8: Network strategies for distributed architecture: (a) daisy-chain network; and (b) star network. Suppose the data rate of each distributed DSP is B bit per second (bps) and the data rate of the root DSP is $B_{N,out}$ bps, the processing latency of each DSP unit is $T_{i,proc}$, the transmission latency between the distributed DSP and its communication target is T_{trans}, and the total processing time, when all distributed and root units are working, is T_{total}. The processing latency, transmission bandwidth required without stalling, and the buffer or memory size required for each DSP unit of both networks are shown.

Daisy-chain network vs. star network In the distributed base station architecture, the centralized baseband signal processing is dissembled and reallocated to the frontend array. Distributive baseband signal processing includes two parts: decentralized computing units and the centralized computing unit. Some typical topologies for the distributed base station architecture include the star networks and the daisy-chain networks. In the star network topology, the decentralized computing units connect directly to the centralized computing unit. Argos [112] is implemented using the star topology. The author in paper [58] proposed a decentralized Newton method for large-scale massive MIMO uplink detection, and implemented with the star topology. The variables used to estimate the symbols are updated at each cluster, and summed up at the central cluster at each step. Then, the estimated symbol is updated with the variables iteratively in the central cluster.

In the daisy-chain network topology, decentralized computing units are chained up together.

Each decentralized computing unit can only exchange data between neighboring units, and the last decentralized computing unit in the chain connects to the centralized computing unit. Hydra testbed from UC Berkeley is implemented with the daisy-chain topology [60]. Hydra employs a two-stage beamforming architecture, where the first stage is implemented with the distributed MRC beamforming to maximize the signal SNR, and the second stage is implemented with the centralized ZF beamforming. The distributed MRC beamformed signal from each subarray is summed up through the daisy chain, and the combined signal is passed to the centralized computing unit from the last subarray. ZF beamforming is performed in the centralized computing unit to remove inter-user interference. Authors of paper [100] proposed a decentralized coordinate descent method to obtain the precoder and the uplink detection. Each decentralized computing unit is attached to a subarray, and the weights are updated using coordinate descent at each antenna sequentially, in the order of antenna placement. The information used to calculate the weight at the neighboring antenna is passed through the daisy chain. The detected symbols at each antenna are also accumulated along the chain using the combining weights. Then the final detected symbols at the last antenna in the last subarray are passed to the centralized computing unit.

Compared with the star topology, the daisy-chain topology has less interconnection bandwidth requirement, which is almost irrelevant to the scale of the number of subarrays since the combination of the decentralized computation results happens along the daisy-chain. Even though the decentralized signal processing can reduce the amount of total data transmission between the decentralized units and the centralized unit, because the data combination happens at the centralized unit, as the number of arrays increases, the possibility for data transmission congestion still exists, and special router design is still needed. However, due to the sequential operation of the data combination, the daisy-chain topology requires larger buffers in each decentralized computing unit for data synchronization between the neighboring units. The buffer or memory size required for real-time signal processing scales up along with the increase of the arrays. And the delay of sequential combination operation may also hurt the real-time performance. As a result, the interconnect router design needs more consideration for the star topology implementation, while for the daisy-chain topology, the computation latency and buffer size need more consideration.

2.2.2 Software simulator

Massive MIMO software simulators are essential for hardware and software algorithm developers because they can quickly model and simulate the overall system behaviors with new features at a low cost. Fig. 2.9 shows the components that a massive MIMO software simulator should have. The simulator should be able to simulate the behaviors of user ends, the base station, and the transmission channels. In addition, to better simulate the overall system, the hardware behaviors should be modeled and simulated.

The simulator should include the signal packet generation, variable sampling rates and carrier frequencies, antenna modeling, frontend or hardware impairments, and signal processing algorithms for both the user ends and the base station. The generation of signal

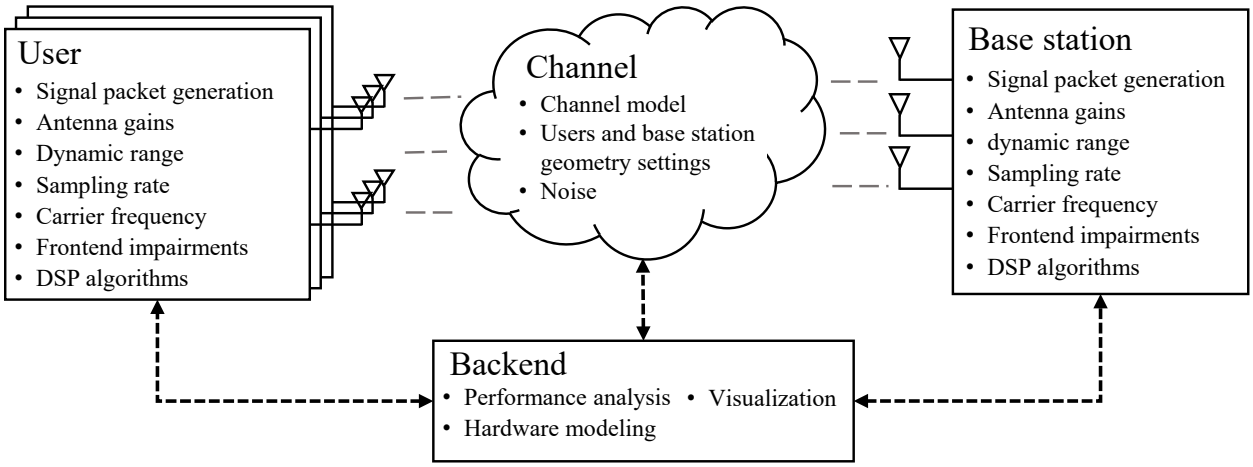


Figure 2.9: Key components in massive MIMO simulators.

packets should consider payload generation, data encoding, and modulation schemes. The user end should also consider pilot designs for the base station channel estimation. Signal sampling rates and carrier frequencies need to be set as parameters to help system developers to explore more possibilities. The frontend impairments should be considered in the simulator to simulate the frontend behavior more realistically. The frontend impairments include IQ imbalance, DC offsets, ADC skews, non-linearity, CFO, SFO, phase offset, etc. Different signal wavelength requires different antenna supports. For this reason, antenna setups, such as the shape, spacing, transmission and receiving gain, and dynamic range, should be modeled. Besides the components discussed above, signal processing algorithms should be implemented for the user end and the base station. The signal processing part in the software simulator should be modularly implemented to ease future algorithm development and adaption. For the base station and multi-antenna users, combiners and precoders are needed in the signal processing for the MIMO beamforming.

Another important component in the software simulator is the channel simulation. The channel simulation should at least be parameterized on the types of the channel model, user end and base station geometry settings, and noise simulation. At the simulator backend, system performance analysis is necessary. The analysis metrics can include beamforming SINR, BER, error vector magnitude (EVM), data rate, sum rate, channel estimation accuracy, etc. To better help with performance analysis and debugging, visualization is highly recommended. The software simulator can also be helpful for hardware realization if the simulator implements hardware modeling, such as the data quantization and the datapath bitwidth change. Then the hardware implementation behavior can be modeled and analyzed before the hardware realization has been finished.

An example of such a simulator is the 5G MIMO simulator released by REMCOM [1]. It offers a comprehensive approach to 5G network design and deployment, covering everything

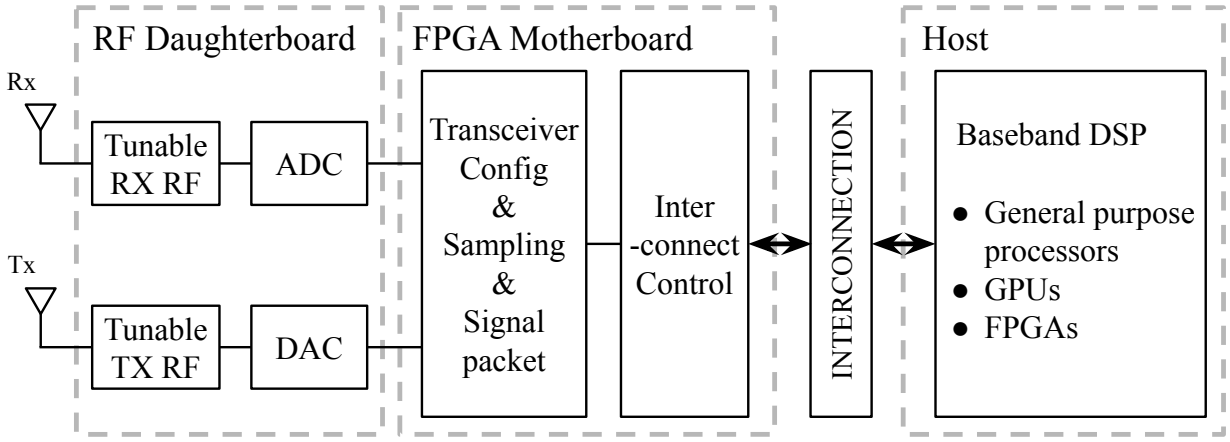


Figure 2.10: SDR architecture.

from system and MIMO antenna design to performance evaluation in realistic simulated environments and deployment planning [1]. Another example can be the Python-based massive MIMO simulator built along with the Hydra testbed by UC Berkeley [25]. It simulates the Hydra system uplink communication with various channel models and flexible system settings. A GUI is also designed to visualize the result and the partial result easily. The baseband signal processing algorithm is modularly designed so that DSP algorithm developers can test the performance of their blocks in the overall massive MIMO system without implementing the whole system from scratch. The hardware non-idealities are considered in the Hydra simulator as well. As a result, this simulator is useful for hardware implementation pre-analysis and debugging.

2.2.3 Hardware implementation

Platform choices

Software defined radio (SDR) based testbeds SDR is a platform cooperated by hardware and software. The key components in SDRs include radio frequency (RF) frontend, analog-to-digital and digital-to-analog converters, sample rate converters, interconnections, and digital signal processing units [114]. Different from the implementation assembled from discrete components, the flexible RF frontends and sampling rate converters can support a broader range of signals which has different carrier frequencies, bandwidth, and sampling rates. The RF frontends can be configured and controlled by the software. In SDRs, the baseband processing is performed on software. The platforms for software implementation can be general-purpose processors, GPUs, and FPGAs.

One of the most commonly used baseband signal processing platforms is the general-purpose processor. SDRs, for example, Universal Software Radio Peripheral (USRP) [121],

can be configured, and the data can be transmitted using C++ programs. The baseband algorithms can also be integrated into C++ programs or implemented on higher-level languages to realize near real-time signal processing. A well-known open source toolkit, GNU Radio, can be installed on general-purpose processors to help with the baseband algorithm development and simulation of SDR systems [133]. GNU radio has a user-friendly interface that developers can design their own algorithms and connect to hardware configuration blocks in the graphical user interface (GUI) easily. The real-time result visualization makes debugging and analysis more efficient. Many state-of-the-art massive MIMO testbeds are implemented with SDRs and general purpose processors, such as ArgosV2 [113], an HW-SW co-operated baseband research platform proposed in [38], and the mmWave phased antenna array based testbed [128]. Due to the high programmability, the SDR system with general-purpose processors is a useful platform for fast prototyping and proof-of-concept implementation. In addition, because programming on general-purpose processors does not need hardware background knowledge, this platform can be used by a larger range of designers.

Graphic processing units (GPUs) are suitable choices for the implementation of SDR systems as well. The architecture of GPU, which is simultaneous multithreading (SIMT), substantially exploits data level parallelism and thread level parallelism. For massive MIMO baseband signal processing, a well-designed decentralized uplink or downlink beamforming algorithm can also expose significant parallelism by distributing the baseband processing across the wideband frequencies or across the antenna array [75, 67]. The high parallelism of the baseband signal processing makes it suitable on GPU. Authors of paper [67] proposed a decentralized ADMM-based beamforming algorithm, and the author in paper [68] proposed a reconfigurable soft-output MMSE MIMO detection algorithm better to exploit the parallelism in the baseband processing algorithm. Both of the algorithms are implemented on GPU clusters, and show the scalability, low latency, and high throughput for potential real-time massive MIMO baseband processing on GPU. By adding flexible frontends, the SDR system with GPU can realize high-speed and highly parallelized baseband algorithms. Compared with general-purpose processors, GPU has a higher parallel computing ability, which makes it especially suitable for prototyping decentralized baseband processing algorithms. However, due to the generalization of the hardware architectures for both general-purpose processors and GPUs, they are not a power-efficient choice for building massive MIMO base stations.

Another widely used platform for developing the baseband algorithms for SDR systems is the field programmable gate array (FPGA). FPGAs are composed of clusters of logic cells, interconnected via programmable routing resources [65]. With the specialized circuitry design and the reduction of operating system overhead, FPGA can be used to implement real-time and more power-efficient digital signal processing (DSP) than general-purpose processors and GPUs. The programmability of FPGAs makes algorithmic upgrading easy at the same time. The famous massive MIMO testbeds, LuMaMi from Lund University [75] and MIMO prototyping system by National Instrument [48], employ FPGAs for DSP co-processor designs. However, programming FPGA requires a higher bar than the previous two platforms, and the developer has to have a hardware programming background. Moreover, the difficulty for FPGA development, such as design and verification, is higher than general-purpose processors

and GPUs since the memory size of FPGAs is more limited, and circuitry-level bugs are harder to detect and fix.

The RF frontend flexibility and high programmability, SDR systems can support a wide range of signal parameters, and are suitable for fast and proof-of-concept prototyping and multiple iterations of baseband algorithm upgrading. However, attackers can also utilize those properties to reconfigure the systems or install malicious software, which leads to severe security issues [114]. Also, due to design generalization, some of the hardware inside SDRs might be redundant for a particular usage, which leads to area waste and higher power consumption. Moreover, the high cost of SDRs makes it hard to build large-scale massive MIMO testbeds. As a result, building robust, low-cost, and power-efficient massive MIMO testbeds using SDRs is still challenging.

Discrete components Besides using SDRs, massive MIMO testbeds can be implemented with discrete components. The discrete components can be customized designs or off-the-shelf products, including antennas, power amplifiers (PAs), low noise amplifiers (LNAs), (de)modulators, ADCs and DACs, filters, and digital baseband processing units. Compared with SDRs, the testbed built with discrete components can only support a smaller range of or a specific carrier frequency, bandwidth, and sampling rate. But the cost, area overhead, antenna array spacing, and packaging difficulty can be reduced, and the massive MIMO system with customized high-performance parameters can be realized.

The scalable Hydra massive MIMO testbed is implemented with discrete components, FPGAs, and a general-purpose processor for baseband signal processing [60]. The base station receiver array is built with off-the-shelf components, and well-designed receiver PCBs package the discrete components to balance the trade-off in yield and area utilization. The receiver arrays then connect to the grouped ADCs, and ADCs connect to daisy-chained FPGAs. The general-purpose processor receives the pre-processed signal from FPGAs, and performs the baseband signal processing. Another fully digital beamforming massive MIMO testbed, operated at 28GHz, is also implemented with discrete components. The customized transceiver front-end elements are placed in a 2-D array, and connected to the intermediate frequency and baseband subsystems through a bent substrate-integrated waveguide (SIW) network.

Compared with SDRs, such as USRP, where each contains only 2 transceivers, the Hydra testbed contains 8 transceivers at each FPGA panel, which leads to much higher area utilization. The customized transceiver front-end design also decreases the total power consumption. Moreover, both testbeds are operated at a radio frequency of more than 60GHz, which none of the existing commercial USRP daughter boards can support. In order to build high performance, power efficient, and high area utilization testbed for relatively fast massive MIMO system prototyping, discrete components are better choices.

Application-specified integrated circuits (ASICs) For massive MIMO testbeds implemented with fully digital beamforming, SDRs can be suitable platforms for the implementation.

However, for testbeds implemented with hybrid analog-and-digital beamforming, SDRs can't provide enough functionality for the analog beamforming stage. Although there are off-the-shelf programmable phase shifters, to achieve better area and power efficiency, and better performance, ASICs should be used.

In [85], a fully-connected hybrid beamforming with 8 antenna inputs and 2 baseband outputs ASIC is proposed. The analog beamforming is performed on chip, which implements RF-domain Heterodyne Cartesian-phase shifting and operates at 25-30GHz. With RF-domain phased arrays, higher power efficiency can be reached with a large number of antenna arrays [85]. Besides the fully-connected architecture, a 71-86GHz 16-element array receiver ASIC is proposed for the hybrid beamforming implementation on Hydra massive MIMO testbed [88]. The phase shifters on the chip are operated at the baseband, and can support up to 16 users. The two-stage beamforming architecture in the Hydra testbed is realized by the phase shifter-based MRC beamforming ASIC and the general-purpose processor with ZF beamforming. The power consumption per antenna per beam is only 7mW including LO circuits and output buffers. Compared with the equivalent digital design implemented on FPGA [25], consuming 338mW per antenna per beam, the analog beamforming ASIC consumes 48.3x less power, and is only $16mm^2$. This shows the power and area efficiency of ASICs for massive MIMO testbed implementation.

Digital signal processing implementation considerations

Design reusability Hardware design is time-consuming because of the efforts on functional verification and the hardness on circuitry optimization. In addition, the corresponding software development, associated with ASIC designs and respins, can contribute to this high design costs [92]. Take FFT for example. Even a single FFT instance for a specific application may take months to design and verify [126]. As a result, in order to reduce the hardware design cost, modularity and reusability should be taken into consideration. The modular and reusable DSP hardware, including logic design and its physical implementation, can be realized by designing generators, instead of instances. Besides modularity, generators should also satisfy the properties of user-friendly programming, highly parameterizable RTL design, and the associated Very Large Scale Integration (VLSI) flow [59]. Then the design reuse can be enabled through generator extensions.

Some methods for the generator creation are Genesis2 by Stanford University [36], Chisel by University of California, Berkeley [8], SpinalHDL [117], and Migen from M-Labs [82]. Unlike Verilog or VHDL where index notation is tedious, or Matlab where no RTL implementation is integrated, generator construction languages can realize both easy programming and RTL generations by using high-level programming languages. Genesis2 is based on Perl, while Chisel and SpinalHDL are implemented with Scala. Although both Perl and Scala are high-level programming languages, Scala is more widely used among programmers. Take Chisel for example. Chisel is a hardware construction language that facilitates parameterized circuit generation for both ASIC and FPGA digital logic designs [17]. This domain-specific extension of the Scala programming language streamlines the process of designing digital

hardware by abstracting hardware components into higher-level descriptions. This abstraction process generates a circuit graph, which is then translated into synthesizable Verilog code for implementation on FPGAs or ASICs [92]. The Chisel DSP library, ACED, is developed to help with the realization of signal processing solutions. Compared with Chisel and SpinalHDL, Migen is implemented with Python and generates the hardware design from Python functions. It targets especially for FPGA implementations.

A scalable generator for the distributed MRC beamforming stage of the Hydra testbed baseband DSP, the Spine generator, is proposed in [25]. The generator is implemented in Chisel and parameterized on the number of antennas per subarray, the number of users in the system, pilot and signal parameters, datapath bitwidth, and datapath parallelization. Along with the Spine generator, multiple DSP generators, such as the Golay correlation generator and the matrix multiplication generator, are also developed which can be reused for other applications in the future. With the help of generator designs, the hardware design reusability can be enabled, and the design cost can be cut down.

Ease of development Another challenge for hardware implementation, especially for VLSI, is the change of platforms, technologies, and system settings for respins. The platforms can be either FPGAs or ASICs. Different platforms and technologies may create different timing issues for the RTL design. By designing generators, the RTL can be regenerated with the new parameters, such as more pipeline stages, less datapath bitwidth, and more datapath parallelization, to relax the timing constraints from the level of architecture designs. As a result, complicated RTL revisions for the change of platforms or technologies are no longer needed, and faster design respins with fewer human resources can be achieved.

Generators can also make respins with different system parameters easier. If a respin wants to implement a massive MIMO system with a different number of antennas at the base station and users in the system, by using generators, a new RTL realization can be regenerated by simply changing the parameters. The Spine generator can quickly generate RTL code for different parameters. It is faster than manually designing instances and redesigning the top-level integration, which takes a longer time. According to [25], the Spine generator takes 9 minutes and 17 seconds to generate the new RTL for different parameters. For the above reasons, the ease of DSP hardware development can be obtained by designing generators as well. The details of the Spine generator will be discussed in the next chapter.

Antenna array design

MmWave is attractive in 5G+ massive MIMO communication due to its wide bandwidth, low interference, and high robustness. The short wavelength of mmWave signals enables smaller spacing of antennas, and as a result, smaller components. One of the most popular choices for mmWave antenna array is the patched antenna array. Patched antenna arrays are cheap, efficient, and small sizes, which is also compatible with planar PCB and packaging technologies [97]. One example of the PCB integrated antenna array is the Hydra testbed antenna array.

Even though patched antennas have the advantages of small sizes and low costs, it is difficult to increase the bandwidth and the gain of the antennas. The improvement of the bandwidth can be realized by patch stacking, and the gain improvement can be achieved by serializing multiple elements together. However, the cost of stacked multi-layer patch antennas is much higher, and the serialized array requires narrow bandwidth. Another antenna option for mmWave communication systems is the sectoral horn antenna array, which can offer high gain and wide bandwidth but at a higher cost. An example of the sectoral horn antenna array is from Plextek, which is implemented on TI's IWR6843 radar-on-chip device and shows a wide coverage [50].

Chapter 3

DSP Hardware Generation for Adaptable Massive MIMO Systems

To demonstrate the feasibility of the massive MIMO system that can satisfy the design considerations in Chap 2, a scalable, adaptable, and platform-portable DSP generator for the distributed massive MIMO systems will be introduced and discussed in this chapter.

3.1 Background

3.1.1 Maximum-Ratio Combining (MRC) Beamforming

Consider a MIMO system with K single antenna users and M antennas at BS. The received uplink signal from a single user $\boldsymbol{r_k} \in \mathbb{C}^{M\times 1}$ at BS with spatial white Gaussian noise is

$$\boldsymbol{r}_k = \boldsymbol{h}_k x_k + \boldsymbol{n}_k \tag{3.1}$$

where $\boldsymbol{h}_k \in \mathbb{C}^{M\times 1}$ is the channel vector for user k, $x_k \in \mathbb{C}$ is the user k transmitted signal dependent on time, and $\boldsymbol{n}_k \sim \mathcal{CN}(\boldsymbol{0}, \sigma^2 \boldsymbol{I_M})$ is the independently and identically distributed (i.i.d) spatial white Gaussian noise vector.

The maximum-ratio combiner architecture for a single isolated link is shown in Fig. 3.1, where $\boldsymbol{r}_k = [r_{k,1}, ..., r_{k,M}]^T$ and $\hat{\boldsymbol{h}}_k = [\hat{h}_{k,1}, ..., \hat{h}_{k,M}]^T$. Then, the combiner output for user k can be expressed as

$$\begin{aligned} \hat{x}_k &= \hat{\boldsymbol{h}}_k^H \boldsymbol{r}_k \\ &= \hat{\boldsymbol{h}}_k^H \boldsymbol{h}_k x_k + \hat{\boldsymbol{h}}_k^H \boldsymbol{n}_k, \end{aligned} \tag{3.2}$$

where $\hat{\boldsymbol{h}}_k \in \mathbb{C}^{M\times 1}$ contains the combiner weights. According to Eq. 3.2, one can get the signal-to-noise ratio γ_k of the combiner output as

$$\gamma_k = \frac{\|x_t\|^2 \|\hat{\boldsymbol{h}}_k^H \boldsymbol{h}_k\|^2}{\mathbb{E}\{\|\hat{\boldsymbol{h}}_k^H \boldsymbol{n}_k\|^2\}} = \frac{\|x_t\|^2 \hat{\boldsymbol{h}}_k^H \boldsymbol{h}_k \boldsymbol{h}_k^H \hat{\boldsymbol{h}}_k}{\sigma^2 \hat{\boldsymbol{h}}_k^H \hat{\boldsymbol{h}}_k}. \tag{3.3}$$

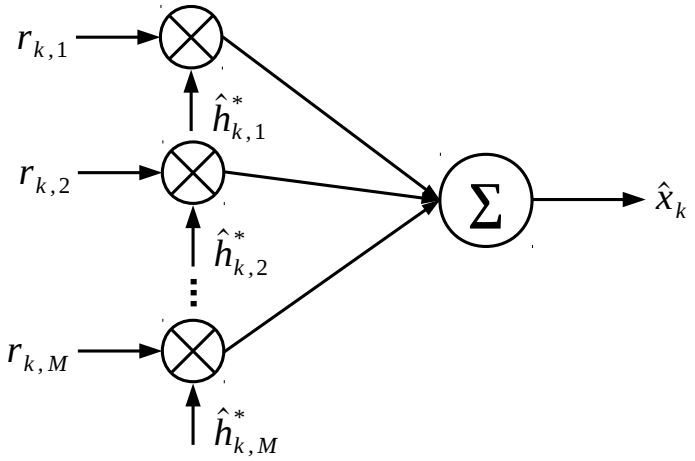


Figure 3.1: The architecture of the maximum ratio combiner.

Assuming the transmission power is P_t, Eq. 3.3 can be expressed as

$$\gamma_k = \frac{P_t \hat{\boldsymbol{h}}_k^H \boldsymbol{h}_k \boldsymbol{h}_k^H \hat{\boldsymbol{h}}_k}{\sigma^2 \hat{\boldsymbol{h}}_k^H \hat{\boldsymbol{h}}_k}. \tag{3.4}$$

Then, the optimal weight that maximizes the SNR

$$\hat{\boldsymbol{h}}_k^\star = \underset{\hat{\boldsymbol{h}}_k}{\operatorname{argmax}}\, \gamma_k, \tag{3.5}$$

can be obtained by calculating the zero point of the conjugate derivative with respect to $\hat{\boldsymbol{h}}_k$, which is $\hat{\boldsymbol{h}}_k^\star \propto \boldsymbol{h}_k$. As a result, for the MIMO system with K single antenna users and M antennas at BS, the optimal weights $\hat{\boldsymbol{h}}^\star$ that maximize each user's SNR in the isolated link are:

$$\hat{\boldsymbol{H}}^\star = \boldsymbol{H}, \tag{3.6}$$

where $\boldsymbol{H} \in \mathbb{C}^{M \times K}$ and $\boldsymbol{H} = [\boldsymbol{h}_1, ..., \boldsymbol{h}_K]$. Note, however, that during payload transmission, when users are simultaneously transmitting, there is correlation between received channels. This is not captured during MRC channel estimation, resulting in residual inter-user interference and reduced SNR after MRC beamforming alone.

3.1.2 Golay sequence based channel estimation

The previous sub-section shows that the optimal weights that maximize the output SNR for MRC beamforming are the channel matrix, assuming isolated links. The channel matrix can be estimated by using Golay sequences.

The Golay sequence contains a complementary pair of sequences with the same length n. According to [37], given $A = [a_1, ..., a_n]$ and $B = [b_1, ..., b_n]$ of length n, with a_k, $b_k \in \{-1, +1\}$ for $1 \leq k \leq n$, the complementary sequence pair (A, B) satisfies

$$A[z]A[z^{-1}] + B[z]B[z^{-1}] = 2n, \tag{3.7}$$

where $A[z] = \sum_{i=0}^{n-1} a_{i+1} z^{-i}$ and $B[z] = \sum_{i=0}^{n-1} b_{i+1} z^{-i}$ for $z \in \mathbb{C}$.

Assume the channel variation is negligible during the transmission of a sequence pair. The received Golay pair at BS with M antennas and spatial white Gaussian noise is

$$\begin{aligned} \boldsymbol{r}_A[z] &= \boldsymbol{h}A[z] + \boldsymbol{n}_A \\ \boldsymbol{r}_B[z] &= \boldsymbol{h}B[z] + \boldsymbol{n}_B, \end{aligned} \tag{3.8}$$

where $\boldsymbol{h} \in \mathbb{C}^{M \times 1}$ is the channel vector, and $\boldsymbol{n}_A, \boldsymbol{n}_B \sim \mathcal{CN}(\boldsymbol{0}, \sigma^2 \boldsymbol{I_M})$ is the spatial white Gaussian noise vector. Using Eq. 3.7, one can get

$$\begin{aligned} &\mathbb{E}\Big\{ \frac{1}{2n} \big(\boldsymbol{r}_A[z]A[z^{-1}] + \boldsymbol{r}_B[z]B[z^{-1}] \big) \Big\} \\ &= \boldsymbol{h} + \frac{1}{2n} (\mathbb{E}\{\boldsymbol{n}_A\}A[z^{-1}] + \mathbb{E}\{\boldsymbol{n}_B\}B[z^{-1}]) \\ &= \boldsymbol{h}, \end{aligned} \tag{3.9}$$

which shows that the sum of the cross-correlation of the sequence pair is an unbiased estimator of the channel vector.

According to Theorem 1.3.2 in [52], Golay sequences can be generated recursively when n is a positive integer power of 2, and the reverse of a Golay sequence pair still gives a Golay sequence pair. Assume the length of the sequence pair is n, the delayed value sequence $\boldsymbol{D}$ is any permutation of the set $\{2^0, 2^1, ..., 2^{log_2(n)-1}\}$, and the weight sequence $\boldsymbol{W} = [w_1, ..., w_{log_2(n)}]$ with $w_i \in \{-1, +1\}$ $\forall 1 \leq i \leq log_2(n)$. Then, the concatenation of the Golay sequence pair $\boldsymbol{g}$ can be generated using Algorithm 1

However, the performance of the channel estimation from Golay sequence pairs may be affected by the quantization noise introduced by the hardware datapath design. Suppose no channel noise, and the quantization noise introduced is $\Delta \in \mathbb{C}^M$, where $\mathbb{E}[\Delta\Delta^H] = \sigma_q^2 I_M$. The channel estimation square error of this method caused by quantization noise is $\sigma_q^2/2n$. The computational complexity of this method for a K-user M-antenna massive MIMO system is $\mathcal{O}(KMlog_2(2n))$

3.1.3 BEACHES algorithm

BEACHES is an adaptive and low-complexity channel estimation algorithm for massive MIMO systems [83]. It exploits the sparsity of channels in the beamspace domain in order to perform adaptive denoising via Stein's unbiased risk estimate (SURE). Let $\mathbf{h}_{est}$ be each user's estimated channel vector. To transfer the channel into the beamspace domain, the

Algorithm 1 Golay sequence pair generation

Require: n, **D**, **W**
Ensure: **g**
  Create two empty arrays $\mathbf{g}_A$ and $\mathbf{g}_B$ of size $log_2(n)+1$ by n
  $\mathbf{g}_A[0,0] \leftarrow 0$, $\mathbf{g}_B[0,0] \leftarrow 0$
  for $0 \leq k < n$ **do**
    for $0 \leq i < log_2(n)$ **do**
      if $k \geq \mathbf{D}[i]$ **then**
        $\mathbf{g}_A[i+1,k] \leftarrow \mathbf{W}[i]\mathbf{g}_A[i,k] + \mathbf{g}_B[i,k-\mathbf{D}[i]]$
        $\mathbf{g}_B[i+1,k] \leftarrow \mathbf{W}[i]\mathbf{g}_A[i,k] - \mathbf{g}_B[i,k-\mathbf{D}[i]]$
      else
        $\mathbf{g}_A[i+1,k] \leftarrow \mathbf{W}[i]\mathbf{g}_A[i,k]$
        $\mathbf{g}_B[i+1,k] \leftarrow \mathbf{W}[i]\mathbf{g}_A[i,k]$
      end if
    end for
  end for
  $\mathbf{g} \leftarrow$ Concatenation of the reverse of $\mathbf{g}_A[log_2(n)]$ and the reverse of $\mathbf{g}_B[log_2(n)]$
  return **g**

discrete Fourier transform (DFT) is performed on the channel vector with a $M \times M$ DFT matrix $\mathbf{F}$ to get $\bar{\mathbf{h}}_{est} = \mathbf{F}\mathbf{h}_{est}$ in BEACHES. The soft-thresholding is performed on each entry of $\bar{\mathbf{h}}_{est}$ by the MSE-optimal threshold $\tau^\star$. BEACHES finds $\tau^\star$ by relying on a proxy for the actual MSE between the ground-truth channel and the denoised channel estimate, and the proxy is obtained by SURE. Then the beamspace denoised channel vector is converted back to the antenna domain using an inverse DFT, after the magnitudes of the entries in $\bar{\mathbf{h}}_{est}$ are soft-thresholded.

3.1.4 In-situ calibration

Manufacturing variability affects circuits at small wavelengths, as small variations in RF circuitry and local oscillator distribution paths can cause large, often unpredictable changes in the RF chain response of different array elements. For beamspace algorithms, the sparsity of the beamspace domain channel vectors is very important. However, due to the array nonidealities, which stem from gain and phase offsets in receivers, the channel estimation from pilots may not be sparse. To extract and calibrate out those array offsets, compressive techniques can be applied, and the mm-wave channel sparsity can be leveraged. [30, 77].

The In-situ calibration method yields the coefficients for array calibration by rotating all channel-user pairs and observing the strongest common direction by spectral decomposition. Then, the line-of-sight (LoS) spatial frequency difference between channel-user pairs can be estimated.

With the in-situ calibration, the sparsity is leveraged, which can guarantee the effectiveness of the beamspace algorithms.

3.2 System Overview

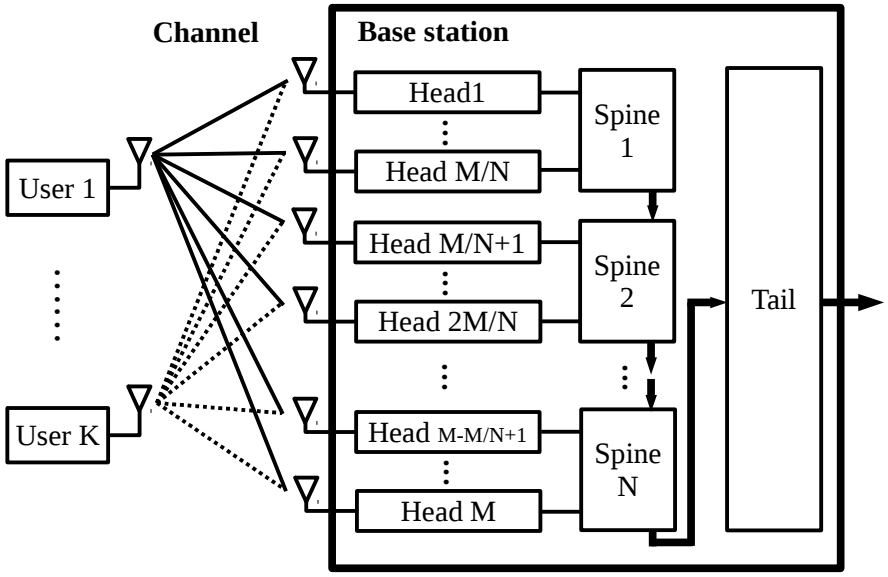


Figure 3.2: The uplink baseband processing architecture for the salable massive MIMO at BS.

The BS system architecture is shown in Fig. 3.2. The Head modules consist of the mm-wave frontend radios including antennas and mixers that downconvert the signal to the baseband. Each Spine module is implemented with a sub-array MRC beamformer, and a daisy chain architecture connects the Spine modules together. The Tail module removes the inter-user and inter-symbol interference by the frequency-selective decorrelation. The details of this two-stage beamforming algorithm are shown in the following section.

3.2.1 Two-stage beamforming

This two-stage beamforming system has been proposed in [60] to strengthen both the scalability and energy efficiency of the signal processing.

Distributed MRC beamformer

The first stage in this architecture is the frequency-flat MRC beamformer which maximizes the output signal-to-noise-ratio (SNR) theoretically and is implemented in the Spine. From the

time-interleaved Golay sequence pilots transmitted from users, the sub-array MRC channel matrix can be estimated. Assume the same multi-user MIMO (MU-MIMO) system described in Sec. 3.1. The received signal at BS is

$$\boldsymbol{r}_{BS} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n}, \tag{3.10}$$

where the transmitted signal vector $\boldsymbol{x} = [x_1, ..., x_k]^T$, and $\boldsymbol{n} \sim \mathcal{CN}(0, \sigma^2 \boldsymbol{I}_M)$ is the i.i.d spatial white Gaussian noise vector.

The MRC beamformer can be divided into multiple subarrays, forming a distributed system. Each subarray performs the local MRC beamforming which is implemented in each Spine, and the local beamformed signal sums together through a daisy chain. Let N be the number of subarrays, and the total number of antennas M be an integer multiple of N. The MRC beamformer weights can be expressed as

$$\hat{\boldsymbol{H}}_{MRC} = [\hat{\boldsymbol{b}}_1^T, ..., \hat{\boldsymbol{b}}_N^T]^T, \tag{3.11}$$

where $\hat{\boldsymbol{b}}_i \in \mathbb{C}^{M/N \times K}$ $\forall\ 0 < i \leq N$. The MRC beamformed signal $\hat{\boldsymbol{x}}_{MRC}$ is

$$\hat{\boldsymbol{x}}_{MRC} = \sum_{i=1}^{N} \hat{\boldsymbol{b}}_i^H \boldsymbol{r}_{BS,i}, \tag{3.12}$$

where $\boldsymbol{r}_{BS,i} \in \mathbb{C}^{M/N \times 1}$, $\forall\ 0 < i \leq N$, and $\boldsymbol{r}_{BS} = [\boldsymbol{r}_{BS,1}^T, ..., \boldsymbol{r}_{BS,N}^T]^T$.

Frequency-selective decorrelation

The second stage performs the frequency-selective decorrelation, which is implemented in the Tail. After MRC beamforming, the inter-symbol and inter-user interference still exist in the wideband-channel signal. To remove them, zero-forcing (ZF) is used for each narrow subcarrier group. In the Tail, carrier frequency offset (CFO) is estimated with orthogonal frequency division multiplexing (OFDM) pilots. Subcarriers are grouped to expand the CFO estimation range [94]. The ZF weights can be obtained from the OFDM pilots, which are time-interleaved and follows the Golay pilots. To perform the frequency selective ZF on payloads, the quadrature amplitude modulated (QAM) payloads are divided into blocks, whose lengths are the same as each OFDM pilot block length L_{OFDM}. Let $\hat{\boldsymbol{S}}_{MRC,p}[i] = FFT(\hat{\boldsymbol{s}}_{MRC,p})[i]$ be the i-th sample of the p-th payload block $\hat{\boldsymbol{s}}_{MRC,p}$ in the frequency domain, and in the q-th subcarrier group, the zero-forcing estimator in the spatial domain gives

$$\hat{\boldsymbol{x}}[p \times L_{OFDM} + i] = IFFT(\hat{\boldsymbol{X}}_{MRC,p})[i], \tag{3.13}$$

where $\hat{\boldsymbol{X}}_{MRC,p}[i] = \hat{\boldsymbol{H}}_{ZF,q}^H (\hat{\boldsymbol{H}}_{ZF,q} \hat{\boldsymbol{H}}_{ZF,q}^H)^{-1} \hat{\boldsymbol{S}}_{MRC,p}[i]$, and $\hat{\boldsymbol{H}}_{ZF,q} \in \mathbb{C}^{K \times K}$ is the estimated channel matrix for q-th subcarrier group.

This two-stage beamforming method removes the inter-user interference and increases the signal-to-interference-plus-noise ratio (SINR) in a distributed fashion. In addition,

compared with the traditional centralized beamforming architecture, the two-stage distributed beamforming architecture relaxes the interconnection bandwidth. Suppose the MU-MIMO system with the total number of M BS antennas and K single antenna users implemented with the signal bandwidth W_s, the oversampling rate at BS receiver s, and analog-to-digital converter (ADC) resolution d, then the interconnection bandwidth for the traditional centralized architecture, which is shown in 2.7, is

$$\text{Bandwidth}_{central} \geq MW_s sd \text{ bits/s} \propto M \tag{3.14}$$

which is proportional to the number of BS antennas. This proportionality limits the scalability of the traditional centralized BS architecture. However, with the help of the two-stage beamforming architecture, the interconnection bandwidth between the first stage and the second stage reduces to

$$\text{Bandwidth}_{2-stage} \geq KW_s d \text{ bits/s} \propto K \tag{3.15}$$

which is proportional to the number of users in the system, and independent of the number of antennas at BS. Fig. 3.3 shows the change of minimum bandwidth requirements with respect to scaling of the number of antennas for both centralized and distributed architecture, and assumes the system has 16 users, 200MHz signal bandwidth, oversampling by 2, and 16 bits ADC resolution. The figure shows that for the traditional centralized architecture, the minimum bandwidth requirement can reach 1.6 Tbps with 256 antennas in the array, while the distributed two-stage beamforming architecture requires x32 smaller bandwidth. The relaxation of the interconnection bandwidth enables the scalability of this work. The reduction of the data movement also improves energy efficiency.

In addition, compared to a traditional centralized frequency-selective ZF beamforming system, the two-stage beamforming requires fewer multiply-and-add operations. Assuming that the payload length is L, $M \gg K$, and the rest of signal processing for the two architectures is similar, the number of multiply-and-add operations that the centralized frequency-selective ZF beamforming requires after matrix inversion is $M^2L + KML$. However, the two-stage beamforming requires $KML + 2K^2L$ number of multiply-and-add operations. Besides the reduction on multiply-and-adds, the matrix inversion size also reduces from $M \times M$ to $K \times K$. The relaxation on the computational complexity leads to less hardware resources and better energy efficiency.

3.2.2 Signal packet format

Fig. 3.4 shows the individual user's signal packet and the uplink signal received at each antenna in BS. The time-interleaved Golay sequence pilots and the binary phase-shift keying (BPSK) modulated OFDM pilots, sent by each user, help with the channel estimation in the two-stage beamformer. Both pilots include guard intervals. The payloads are chosen to be QAM to allow for a larger CFO range and phase noise caused by hardware imperfections, and each payload block comprised the same structure as OFDM, except for the OFDM symbols.

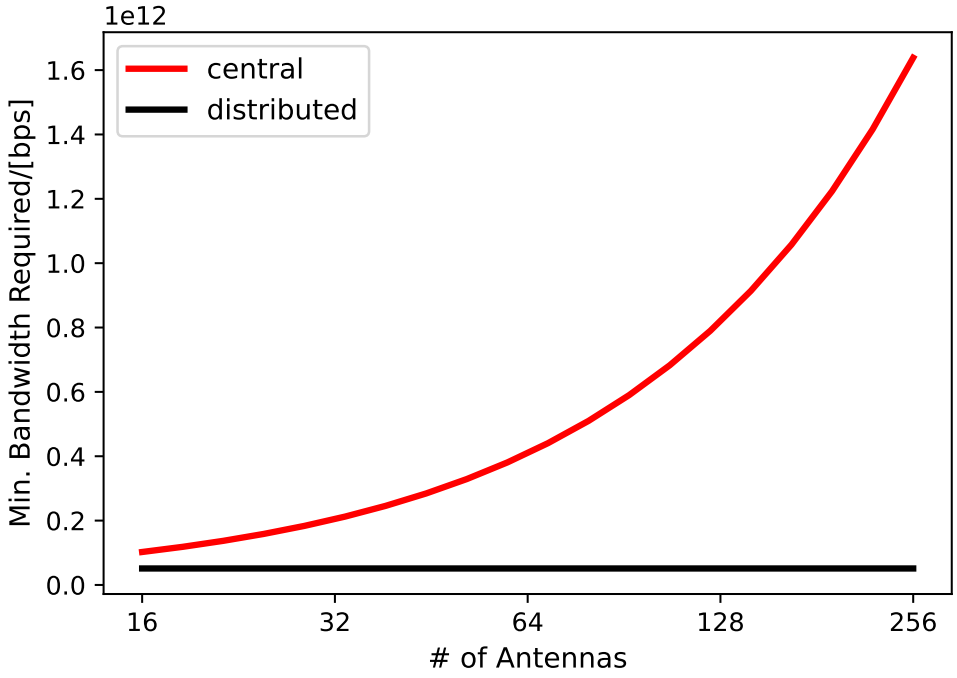


Figure 3.3: Minimum bandwidth requirement comparison between centralized and distributed architecture, assuming a MIMO system with 16 users, 200MHz signal bandwidth, oversampling by 2, and 16 bits ADC resolution.

All users send the payload simultaneously after pilots are sent. A beacon signal is propagated to all users and BS at the beginning of each packet as the timing reference.

3.2.3 The Spine Generator

A major challenge for the design of moderate-volume ASICs is the large design costs stemming from the engineering design and verification efforts, as well as for the software development, associated with chip designs and respins [91]. Design costs can be reduced through more effective reuse, which can be achieved by employing generators for the logic design and its physical implementation. Generators employ higher-level programming constructs to create modular, highly parameterizable RTL hardware designs and Very Large Scale Integration (VLSI) flows [59], enabling design reuse via generator extension. Each unique design instance is a result of generator parameterization.

The digital signal processing (DSP) hardware in this work is generated by using Chisel, a hardware construction language that facilitates parameterized circuit generation for both ASIC and FPGA digital logic designs [17]. Chisel is a domain-specific extension of the Scala programming language, which abstracts primitive hardware components, generates the corresponding circuit graph, and translates the graph into the synthesizable Verilog

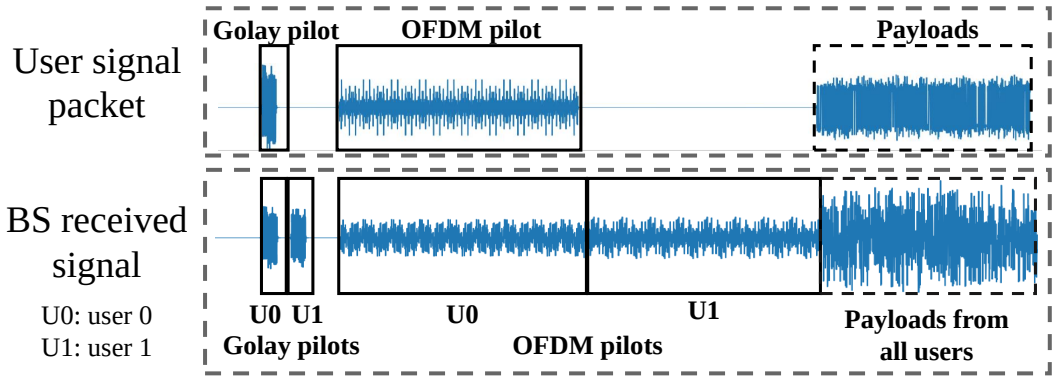


Figure 3.4: The format of the user-transmitted signal packet.

representation [91]. The Spine generator in this work enables design reusability as well as platform and technology portability.

The Spine generator generates the MRC beamformer in the BS architecture. The parameterization of the generator is

- the subarray antenna number
- the system user number
- the signal oversampling rate
- the Golay pilot design parameters
- he number of root-raised cosine (RRC) taps
- the Lagrange polynomial order
- the datapath bitwidth
- the datapath parallelization

Fig. 3.5 shows the Spine generator datapath. It consists of multiple DSP generators implemented using ACED [125], a reusable Chisel DSP library. The Spine generator is also extended to enable signal processing with beamspace algorithms.

The inputs of the Spine DSP are M/N antennas of 2x oversampled in-phase and quadrature (IQ) signals. Datapath parallelism is determined in the generator configuration at the configuration time. The final outputs are the K users' MRC beamformed signals summed with the signal coming from the previous neighboring Spine in the daisy chain and sent to the next neighboring Spine. Fig. 3.5 shows the Spine generator datapath. The parameters of each module generator are shown in the light blue boxes.

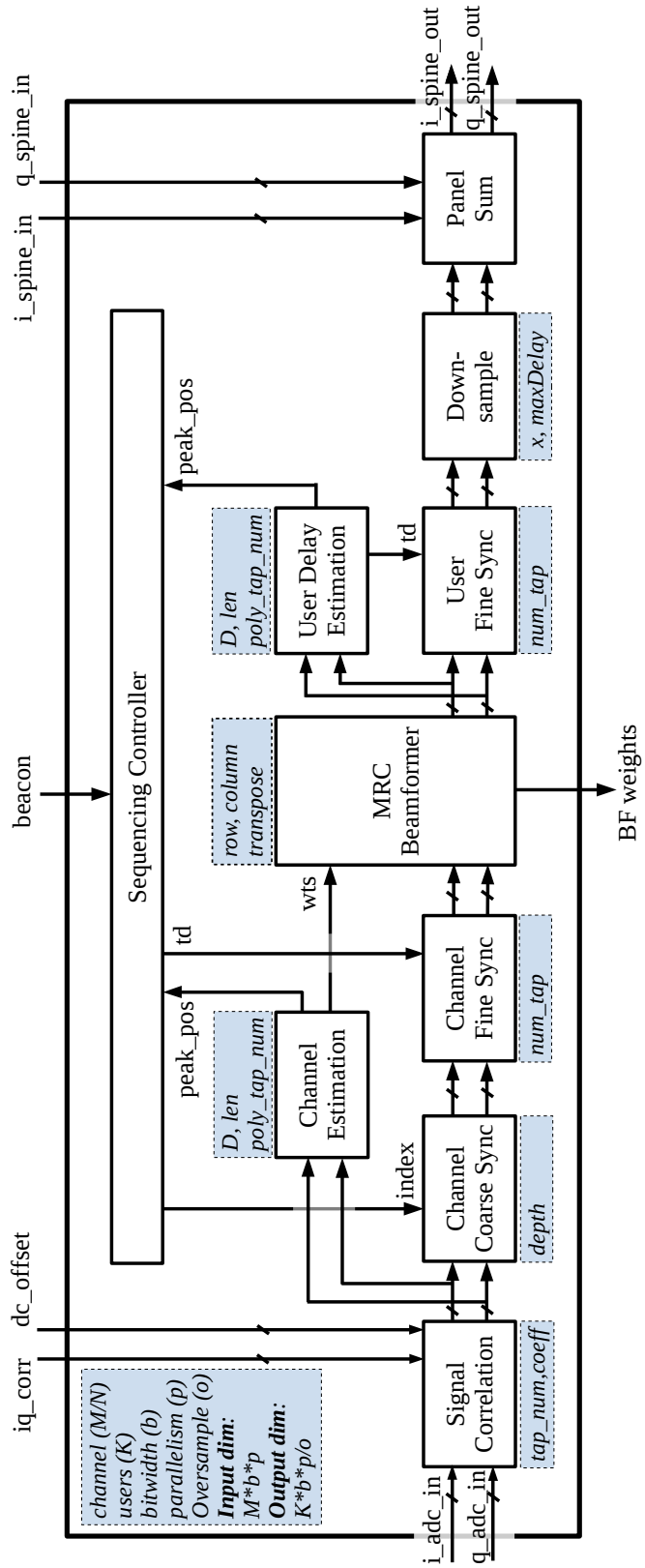


Figure 3.5: The Spine generator datapath.

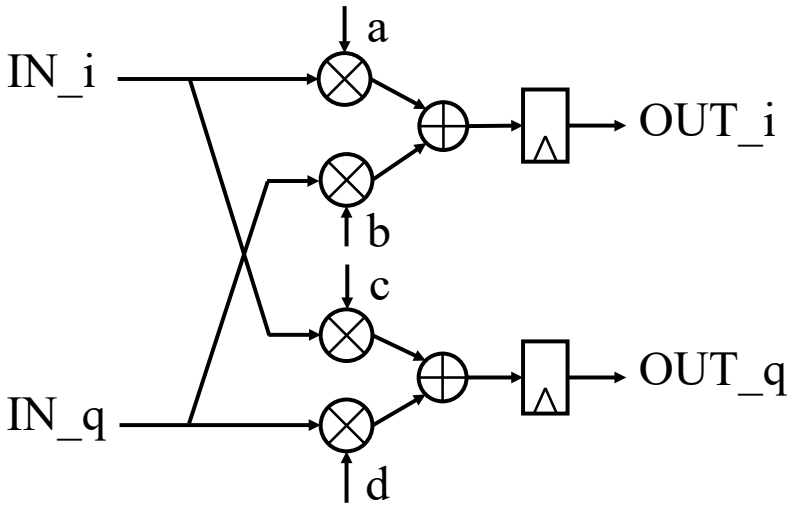


Figure 3.6: Module architecture for IQ correction.

Signal Correction

The signal correction group for each antenna includes a 65-tap RRC filter, an IQ signal synchronizer, and a direct current (DC) offset cancellation. The RRC filter is generated by the strength-reduced FIR filter generator, based on tree reduction with parameterizable tap numbers and configurable coefficients. The coefficients in the IQ synchronizer and DC offset canceller need to be manually set.

IQ correction As discussed in Chapter 2, to address front-end impairments, the datapath includes an IQ synchronizer and a DC offset cancellation module, which are manually calibrated to balance the impairments. The IQ imbalance originates from two sources:

- differences in gains, where g_i and g_q represent the gains for the I and Q channels, respectively
- phase offsets, where θ_i and θ_q represent the phase offsets that cause the I and Q channels to be non-orthogonal.

Let $y = y_i + j \cdot y_q$ be the signal without IQ imbalance for a single channel, and $y_{im} = y_{im,i} + j \cdot y_{im,q}$ be the signal with IQ imbalance due to the frontend. Then y_{im} can be expressed as

$$\begin{bmatrix} y_{im,i} \\ y_{im,q} \end{bmatrix} = P \begin{bmatrix} y_i \\ y_q \end{bmatrix} = \begin{bmatrix} g_i \sin\theta_i & g_q \cos\theta_q \\ -g_i \cos\theta_i & g_q \sin\theta_q \end{bmatrix} \begin{bmatrix} y_i \\ y_q \end{bmatrix}. \tag{3.16}$$

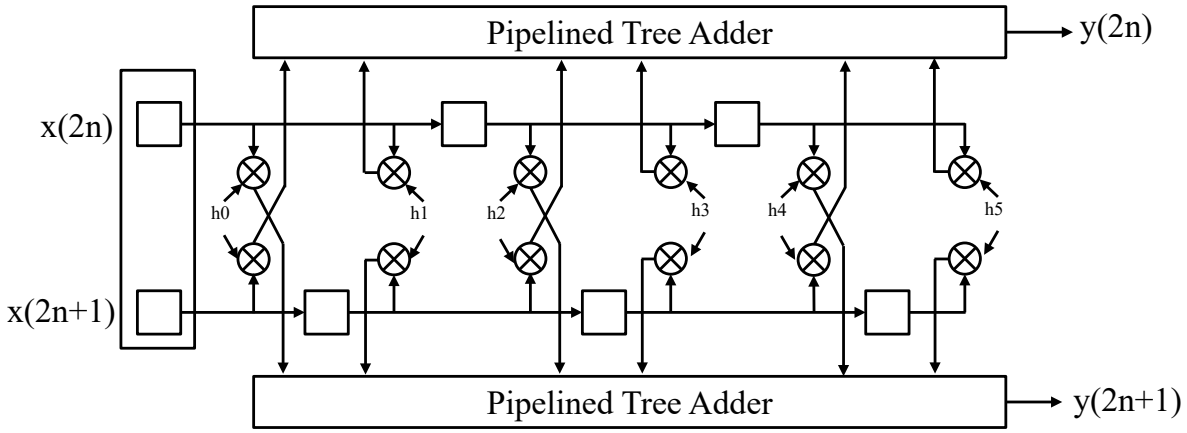


Figure 3.7: Tree-reduced strength-reduced FIR filter generator generated example instance. The example instance has 6 taps and is parallelized by 2.

To correct the IQ imbalanced signal, P^{-1} should be multiplied with the imbalanced signal. Then

$$\begin{bmatrix} y_i \\ y_q \end{bmatrix} = P^{-1} \begin{bmatrix} y_{im,i} \\ y_{im,q} \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} y_{im,i} \\ y_{im,q} \end{bmatrix}. \tag{3.17}$$

Based on the equation, the architecture of IQ synchronizer is shown in Fig. 3.6.

The parameters of the IQ synchronizer are the coefficient matrix, ranging from $(-1, 1)$.

RRC filter The RRC filter is created using a tree-reduced strength-reduced FIR filter generator. The number of taps can be adjusted as a parameter, and the coefficient array is customizable. Figure 3.7 displays an example of an FIR filter generated by the FIR generator with 6 taps and parallelized by 2. Each valid cycle, way-0 and way-1 shift registers shift to the right by one, and are multiplied by their corresponding coefficient. The way-0 and way-1 multiplication results are accumulated through way-0 and way-1 pipelined tree adders, respectively. The FIR filter generator can also be extended to apply symmetric FIR filters with less hardware. Assuming that the number of taps in an RRC filter is N_t, and since the RRC filter coefficients are symmetric, to reduce the number of multipliers in the design, the input of the FIR filter is the summation of samples i and $N_t - i - 1$ for $i \in [0, N_T)$.

The verification of the FIR filter generator is being performed using random number tests in Chisel. An equivalent software function has been developed in Scala, which serves as the ground truth generator. During each round of testing, the number of taps for the FIR filter generator is randomly determined. Additionally, the parallelization is being tested with 1, 2, 4, and 8 taps. The filter taps and input are randomly generated for each round of testing, and the output of the generator is compared with the result generated by the ground truth Scala function.

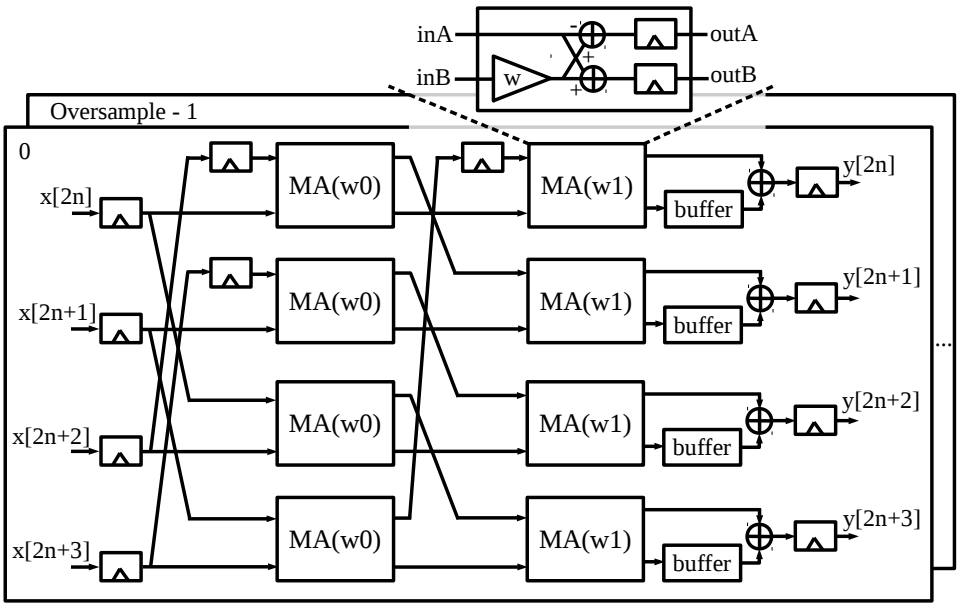


Figure 3.8: The generated Golay correlator architecture.

MRC channel estimation

Sec. 3.1 mentions that the cross-correlation of the Golay sequence pilots is an unbiased estimator for the optimal MRC weights. The pipelined parallelizable Golay correlator generator is designed based on the Algorithm 1. Because the reverse of a Golay sequence pair is also a Golay sequence pair, the Golay correlator is equivalent to the Golay generator of the reverse sequence. Compared with the design in [33], the proposed design is pipelined and parallelized, to achieve a higher throughput. The datapath is pipelined by $log_2(n)$ stages. The datapath parallelism determines the input of each multiply-add (MA) unit shown in Fig. 3.8 and the structure of the delay registers. The MA input can be either from the delay registers or the pipeline registers. This correlator generator is parameterizable for **D**, and runtime configurable for **W**. The correct correlation is guaranteed by switching the input of **W** at the corresponding user's time interval controlled by the sequencing controller. Fig. 3.8 shows the generated architecture of an oversampling-2, length-4 Golay correlator with parallelism of 4.

The functionality of the Golay correlator is being tested using oversampling rates of 2, and lengths of 4, 16, 32, and 64. Additionally, parallelization is being tested with values of 2, 4, and 8. During testing, the output of the Golay correlator generator is being compared with a matched filter software implemented in Scala. The input of the Golay correlator is being generated using a random **W** sequence.

The channel delay estimation includes the coarse delay estimation and the fine delay estimation. The coarse delay estimation estimates the channel delay difference of an integer

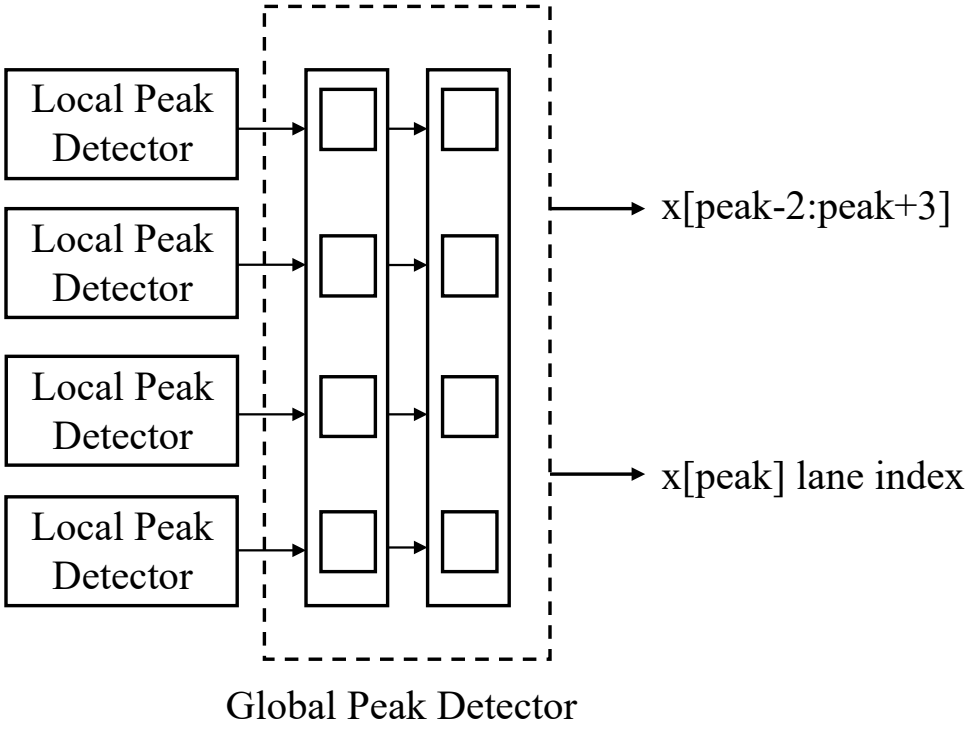


Figure 3.9: An example of the peak detection architecture with parallelization of 4.

multiple times per clock period, and the fine delay estimation corrects the sampling time skews.

Coarse delay estimation The coarse delay estimation is based on the correlation norm peak of Golay pilots in Eq. 3.7. The correlation norm peak detection contains two parts: local peak detection and global peak detection. The local peak detection detects the maximum value in each parallel lane, and the global peak detection detects the peak among all lanes. Fig. 3.9 shows an example of the peak detection architecture with parallelization of 4. The finite state machine for the local peak detection, which is shown in Fig. 3.10 contains 3 states:

- **Wait**. In this state, the local peak detector tracks the average power P_{avg} of the correlation result for a certain window size WS. If the input correlation power is larger than **threshold** $\times P_{avg}$ and larger than **lowerbound** value, then update P_{max} to be the input correlation power and maximum power buffer index i_{max} to be $WS-1$, and go to **Detect** state; Otherwise, stay in this state.
- **Detect**. Track the maximum power for the window size $WS-1$. If the input correlation power is larger than current P_{max}, then updates P_{max} and i_{max}. After finishing tracking $WS-1$ number of correlation power, go to **Done** state.

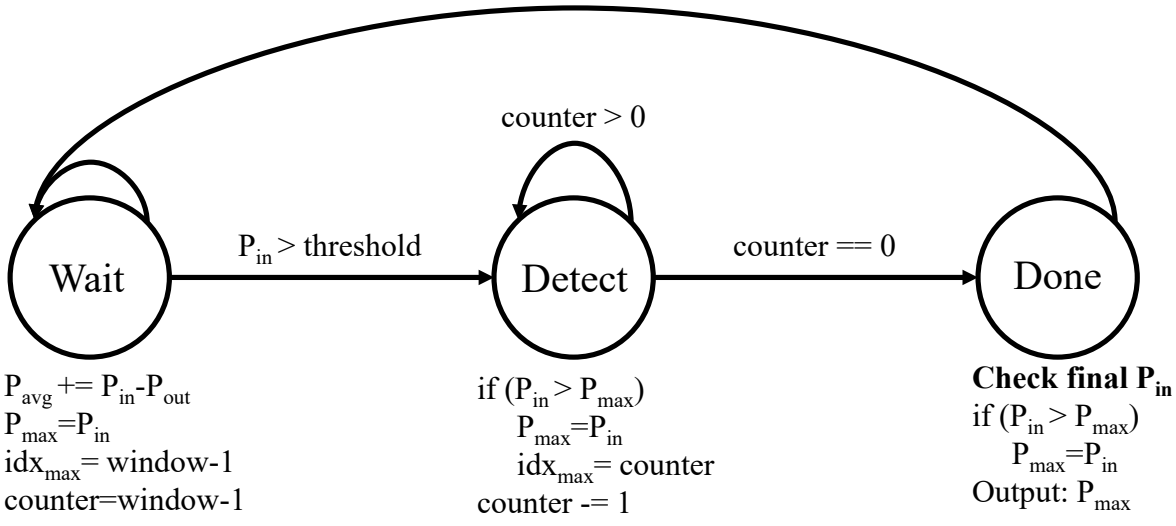


Figure 3.10: The finite state machine for the local peak detection.

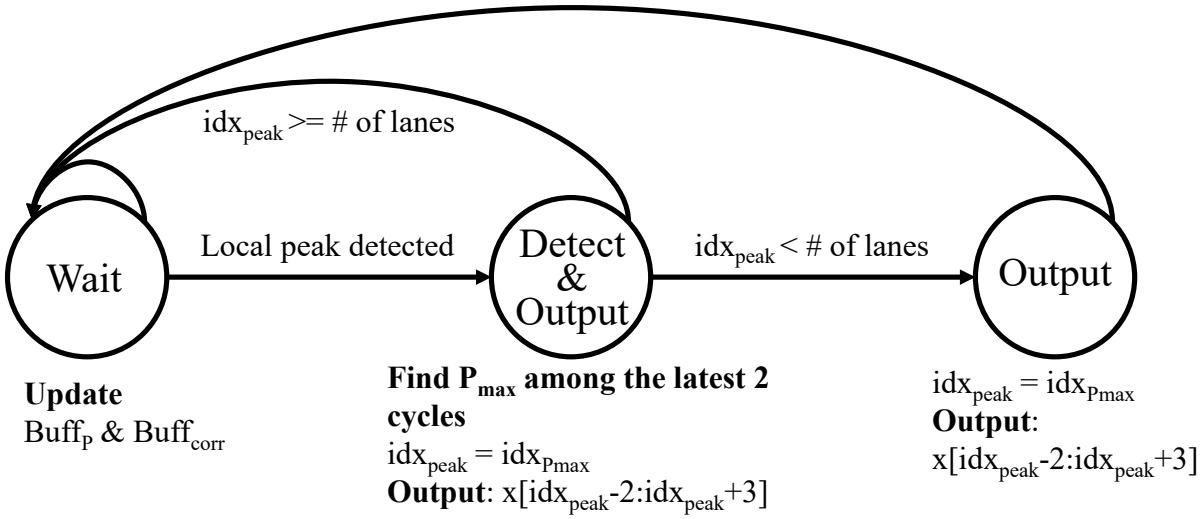


Figure 3.11: The finite state machine for the global peak detection.

- **Done**. In this state, check whether the input correlation power is larger than the current P_{max} or not. If yes, output this correlation power; Otherwise, output P_{max}. Then go back to **Wait** state.

The finite state machine for the global peak detection, which is shown in 3.11, contains 3 states as well:

- **Wait**. Use the shift register to keep records of the most recent $WS\times$ number of lanes correlation results and their power. If one of the lanes detects the peak, go to **Detect & Output** state; Otherwise, stay in the current state.
- **Detect & Output**. Use a tree-reduced comparator to find the maximum power among the correlation powers in the current and the previous cycle in all lanes. If the index of the maximum power in the buffer is larger than or equal to the number of lanes, output the correlation result corresponding to the maximum power and its adjacent $2l$ correlation results, shift the buffer, and go back to **Wait** state; Otherwise, go to **Output** state.
- **Output** state. Output the correlation result corresponding to the maximum power and its adjacent $2l$ correlation results, and go back to **Wait** state.

Fine delay estimation Because of the imperfections of the frontend components, for example, ADC skews and sampling phase mismatch, the frontend may not be able to sample the signal at maximum power. In this case, resampling is necessary by interpolating the correlation norm using Lagrange polynomial $\mathcal{L}(x)$. Let $c(i)$ be the interpolation samples, $c(0)$ be the correlation norm peak, and the interpolation sample length is $2l+1$, where $i \in \{-l, ..., 0, ...l\}$. The fine delay estimation $\hat{s}_0$ is

$$\begin{aligned} \hat{s}_0 &= \arg\max_x \mathcal{L}(x) \\ &= \arg\max_x \sum_{i=-l}^{l} \left[c(i) \prod_{-l, j\neq i}^{l} \frac{x-j}{i-j} \right]. \end{aligned} \tag{3.18}$$

Let $y(t)$ be the skewed signal and $\hat{y}(t)$ be the deskewed signal, which is given by

$$\hat{y}(t) = \sum_{i=-l}^{l} \left[y(t+i) \prod_{-l, j\neq i}^{l} \frac{\hat{s}_0 - j}{i-j} \right]. \tag{3.19}$$

In software design, to find $\hat{s}_0$, one needs to take the derivative of $\mathcal{L}(x)$, and use Newton method to find the maximum point. Loops cannot be avoided when using Newton method to find the zero point. In order to avoid loop iterations in hardware, the following algorithm is used to calculate the maximum $\hat{s}_0$ for a given resolution r.

Let the datapath bitwidth be B. The threshold of the peak detector ranges from $[0, 2^{B-1})$ with precision 0.5, and the lower bound of the peak detector ranges from $(-1, 1)$ with precision $2^{-(B+1)}$. The fine delay estimation ranges from $(-1, 1)$ with precision r, and the fine delay synchronization coefficient ranges from $(-1, 1)$ with precision $2^{-(B+1)}$.

MRC beamformer

The MRC beamformer is generated by the pipelined systolic-array-based weight-stationary matrix multiplier generator, and is parameterized on the matrix dimension and datapath

Algorithm 2 Fine delay estimation

Require: **c**, r

Ensure: $\hat{s}_0$

$s_q \leftarrow q \cdot r$, where $q \in \{-1/r, ..., 0, ..., 1/r\}$

for q from $-1/r$ to $1/r$ **do**

Calculate all coefficients $cf_i \leftarrow \prod_{j=-l, j\neq i}^{l} \frac{s_q - j}{i-j}$, where $i \in [-l, l]$

$\hat{y}_q(0) \leftarrow \sum_{i=-l}^{l} y(i) \cdot cf_i$

end for

$\hat{s}_0 \leftarrow \arg\max_q \hat{y}_q(0)$

return $\hat{s}_0$

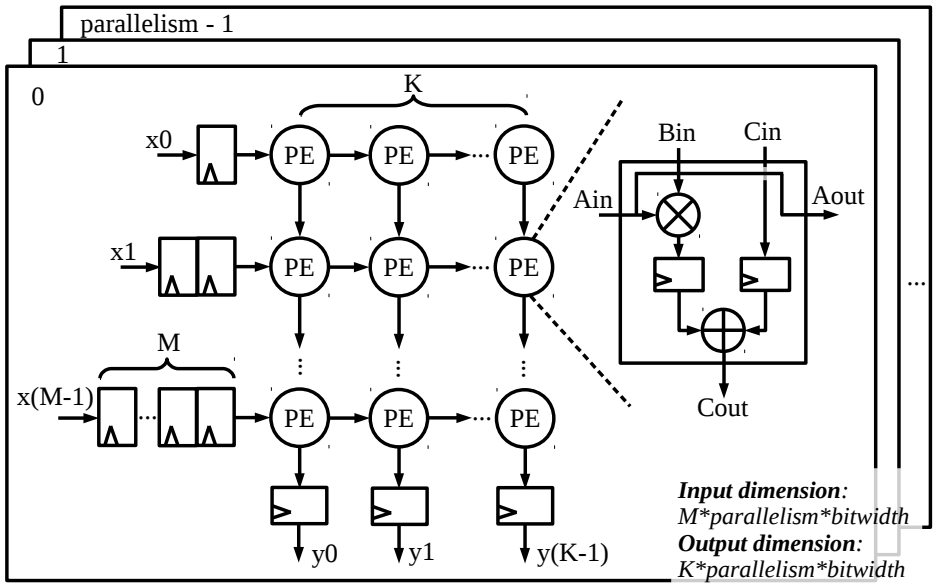


Figure 3.12: The systolic array-based MRC beamformer architecture.

parallelization. In this system, each processing element (PE) is composed of a width-parameterized complex fixed-point multiplier and an adder. For a massive MIMO system with K users and M channels per Spine, a single path requires a matrix multiplier dimension of $M \times K$.

The architecture of the beamformer is shown in Fig. 3.12, where the weight inputs are the estimated frequency-flat channel matrix, and the input to each row is the channel's fine synchronized signal. To satisfy different timing requirements for different FPGA platforms and transistor process nodes, the number of pipeline stages between PE columns is also parameterized.

The parameters K, M, and parallelism can take on any integer value greater than 0. In this system, the input, coefficients, and output all have the same data range and precision as

the datapath. To test the MRC beamformer, random numbers are generated as input, and the resulting output is compared with the ground truth produced by the software implementation of the matrix multiplier using the same test data in Scala.

Sequencing Controller

To guarantee the performance of the MRC beamforming, all channels need to be synchronized. The sequencing controller provides the time reference for all processing modules, and helps with channel synchronization.

At the start of each signal packet, the sequencing controller sends a beacon signal to reset registers and state machines for all modules. In the case of Golay correlators, the sequencing controller also manages the change of W for different users. As Golay pilots are sent in TDD mode, the sequencing controller sets up the W sequence in all Golay correlators to match user k's W during the appropriate time slot.

Due to propagation delays and front-end mismatches, even for the same user, signal arrival times can be mismatched. We denote the time delay caused by user i as $T_{u,i}$ and the time delay caused by channel j as $T_{c,j}$, where i ranges from 1 to K and j ranges from 1 to M/N. We can assume that T_u follows a normal distribution with mean μ_u and variance σ_u^2. The time delay between user i and channel j in the same Spine module is then $T_{u,i} + T_{c,j}$. To synchronize the channel before the MRC beamformer, a channel time delay estimator is necessary.

The channel delay estimator is the average of the time delay among all users' signal received by the same channel. This estimator is unbiased, which can be shown by

$$\mathbb{E}\big[\frac{1}{K}\sum_{i=0}^{K}(T_{c,j} + T_{u,i})\big] = T_{c,j} + \mu_u. \tag{3.20}$$

where $T_{u,i}$ is the time delay created by the user i signal propagation, and $T_{c,j}$ is the delay from channel j, $i \in [1, K]$ and $j \in [1, M/N]$.

The user time delay estimation uses the same algorithm as the channel delay estimation. The downsampling phase selection is determined by the user coarse time delay estimation.

3.2.4 Implementation of the beamspace algorithms

To show the adaptability of the system, the in-situ calibration and BEACHES algorithms are implemented in the system to improve the channel estimation in the MRC beamforming stage. The in-situ calibration is applied to the channel estimation from Golay pilots to calibrate out channel phase offsets. After calibration, the calibration coefficients are multiplied by the MRC beamformer weights to leverage the sparsity. BEACHES is then used to denoise the calibrated MRC beamformer weights.

3.3 Evaluation

3.3.1 Simulator

The massive MIMO system is modeled in a Python-based simulator. The end-to-end simulation encompasses terminals, channels, and BS simulations, including signal packet generations, various channel and transceiver front-end models, and the modular signal processing chain. The modular signal processing that simulates the BS can encompass any instance from the Chisel generator described in Sec. 3.2 by generating raw samples and capturing and post-processing the results, which is illustrated in Fig. 3.13 by dashed arrows. The modular design enables the adaptability of the system to evaluate different processing algorithms. The calibration and BEACHES algorithms are implemented in MATLAB, and integrated into the simulator using a MATLAB-to-Python API, Matlab Engine API for Python [79]. The Spine generator is verified against this "golden" model simulator.

Fig. 3.13 shows the system data flow. The solid rectangles represent Python functions, and the dashed rectangles represent MATLAB functions. After system parameters are set, each user's signal packet is generated according to the system settings, which include pilot lengths, guard interval lengths, payload modulation schemes, etc. The channel modeling in the Python simulator includes the flat i.i.d channel, the LoS channel, and Rician channel. The QuaDRiGa mmMAGIC urban micro line-of-sight (QuadMMLoS) channel is implemented in MATLAB, and can be called via the API with the corresponding channel model setting. QuaDRiGa channel model is an enhancement of the WINNER model following a geometry-based, stochastic channel modeling approach for MIMO [14]. The generated QuaDRiGa channel matrix is returned to Python for further signal processing. The channel simulation contains transceiver front-end simulation and noise addition.

The signal processing in the BS has two options: FPGA emulation or Python simulation. In FPGA emulation, the simulator generates the control and configuration Tcl files for setting up the signal processing on the FPGA. If Python simulation is chosen for BS signal processing, the channel estimation and the MRC beamforming will be performed in Python. If BEACHES is selected at the configuration time, the denoising algorithms will be performed in MATLAB with the parameter passing through the API. The frequency-selective decorrelation stage is only applied in the Python simulator regardless of the signal processing options. The simulator evaluates the demodulation performance based on the bit error rate (BER), the signal-to-interference-plus-noise ratio (SINR), and the error vector magnitude (EVM).

3.3.2 FPGA Emulation and Simulation

The Spine generator output is mapped to a Xilinx VCU118 FPGA. The FPGA instance contains

- a Spine DSP core generated by the Spine generator specified by the parameters shown in 3.1,

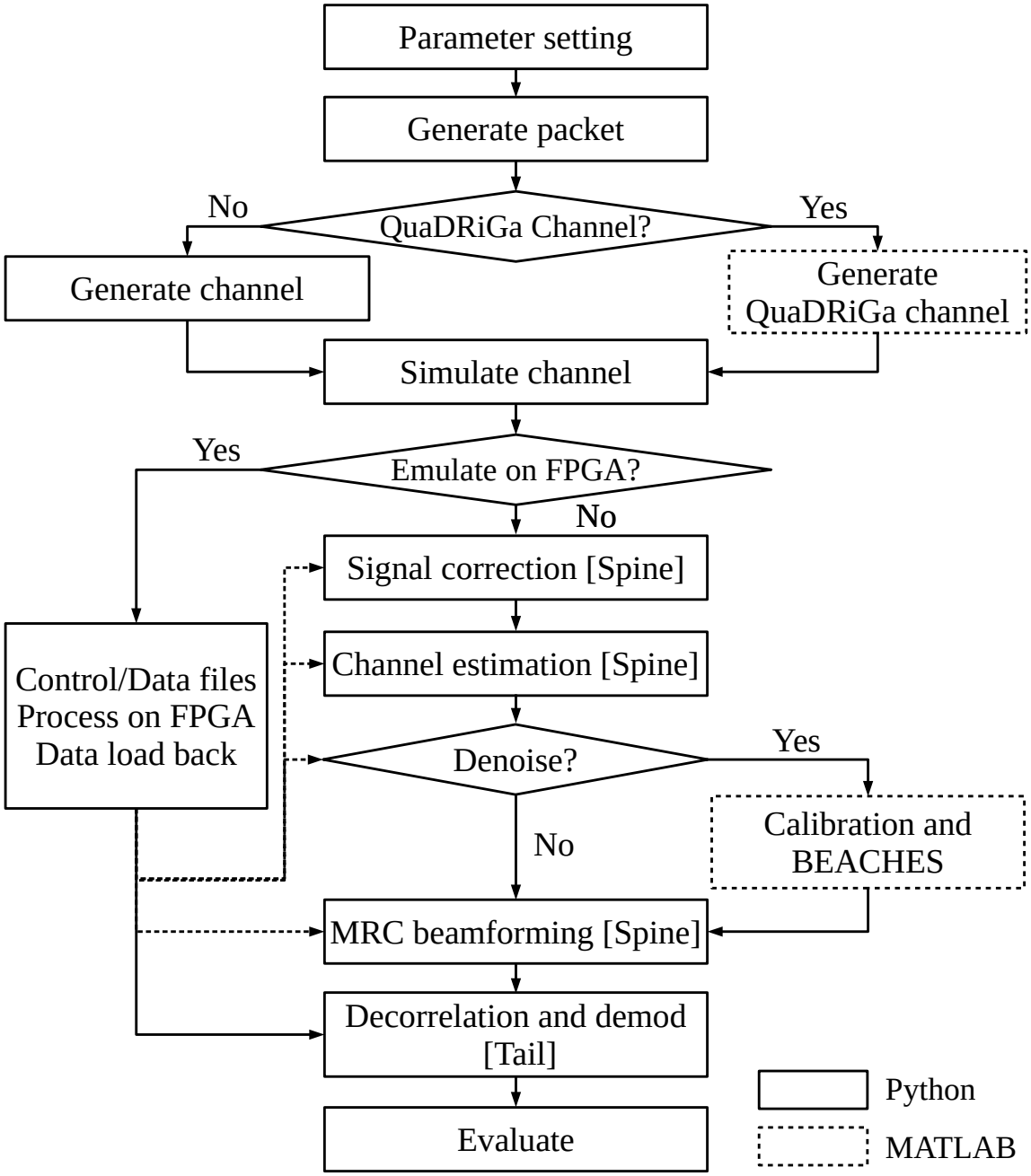


Figure 3.13: The system flow diagram, with software and hardware integration.

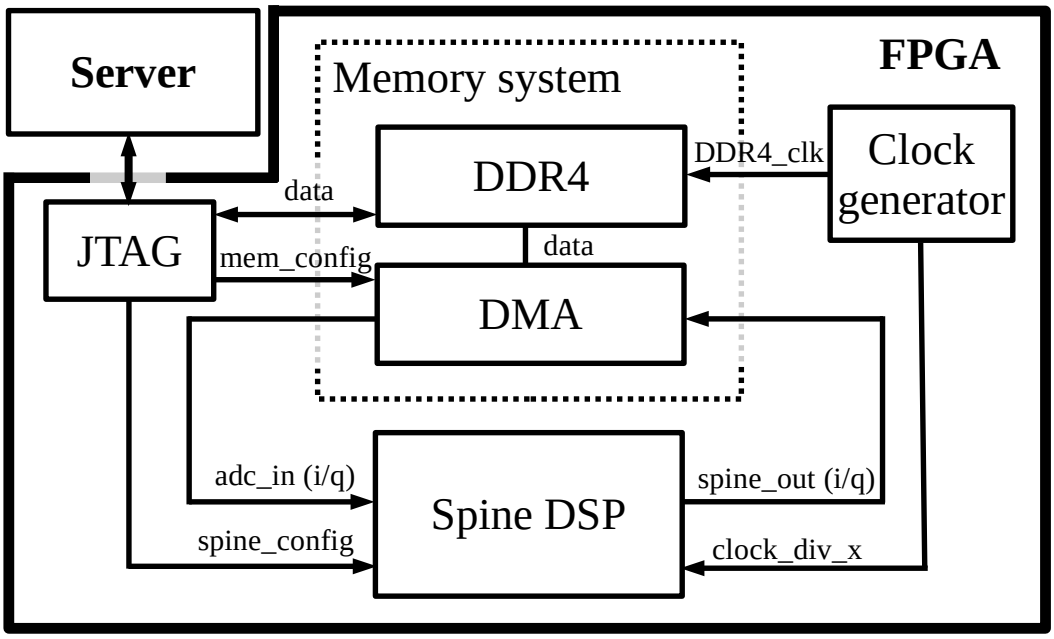


Figure 3.14: The emulation system architecture.

- the memory system containing DMA and DDR4,
- the clock generator to generate the 2 clock domains for the DSP core and the memory system.

A server connects to the FPGA via JTAG for data transfer and FPGA programming. As shown in 3.1, a 64 antennas and 4 users BS system is emulated. The FPGA emulation system architecture is shown in Fig. 3.14. The JTAG connection bitwidth is 64. The on-board DDR4 memory bitwidth is 512 without the error correction code. The DDR4 is connected using the DDR4 interface provided by Xilinx.

Before the FPGA emulation starts, the Python simulator first generates the user packets and simulates the channel. Then the floating-point emulation input data is converted to fixed-point data format by the simulator. At the same time, the simulator generates the configuration and control Tcl files to help the transmission of the emulation data to the on-board DDR4 memory. The DSP core configuration includes:

- Set up DMA with the load and store base addresses and length, and the start signal.
- The beacon signal to indicate the start of each packet.
- The number of users in the system.
- Each user's **W** sequence for the Golay pilot.

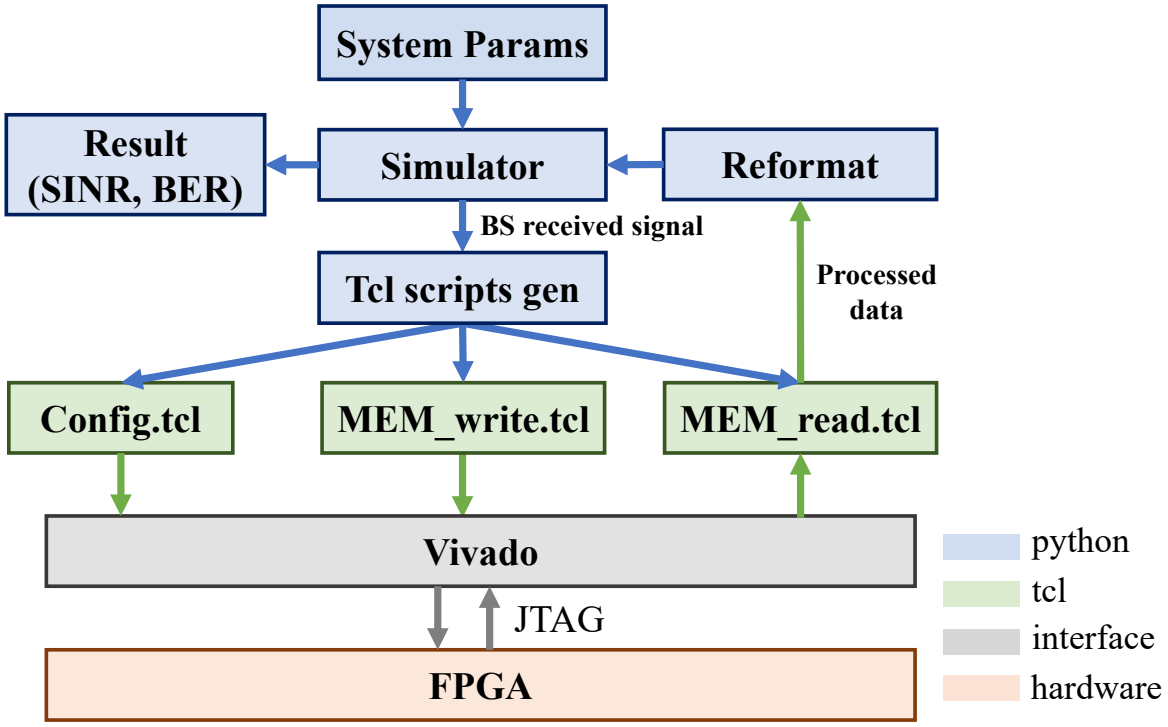


Figure 3.15: The control and data flow of the simulator if FPGA emulation is chosen.

- The thresholds for the channel and user correlation peak detector.
- Set up debug ports.

When the Spine DSP core processes the data, data are also streamed from and to the memory. After the FPGA finishes beamforming, the data are sent back to the server through JTAG, and reformatted to the floating point data. Meanwhile, the server loads the estimated MRC beamformer weights from FPGA to be processed for the beamspace algorithms. The simulator control and data flow is shown in Fig. 3.15.

The evaluation of the Spine generator is based on the SINR after demodulation and the channel estimation normalized mean square error (NMSE). In the evaluation part, 4 different channel models are considered: 1) The flat i.i.d channel model; 2) Rician channel model with K factor being 10dB; 3) LoS channel model; and 4) QuadMMLoS channel model with users placed at least 10m away from the BS and minimum user angle separation 2 degrees. The modulation scheme for the payload is QPSK. The performance evaluation of the system after integrating the beamspace denoising algorithms is based on the channel estimation NMSE and the error vector magnitude after demodulation. The measurement uses the following steps: 1) extracting the FPGA-emulated MRC beamformer weights; 2) running the beamspace denoising algorithms and getting the denoised beamformer weights;

Table 3.1: FPGA emulation system parameters.

	Parameter	Value
System parameter	MIMO system parameter	32 antennas, 2 users 64 antennas, 4 users
	Signal bandwidth	200MHz
	Oversampling rate	2
	Golay pilot length	64
	OFDM pilot length	1024
	Subcarriers per group	16
	Cyclic prefix length	64
	Payload modulation scheme	QPSK, 16QAM
	Channel model	Flat i.i.d, Rician, LoS, QuadMMLoS
FPGA parameter	FPGA Type	Xilinx VCU118
	Number of antennas per Spine	4
	Datapath bitwidth	8
	Datapath parallelization	8
	Baseband clock freq	50MHz

3) rerunning the Python simulation while using the FPGA-emulated beamformer weights and the denoised weights from 1) and 2).

3.4 Results

3.4.1 Golay pilot length analysis

In this experiment, the channel estimation performance with respect to different Golay pilot length are discussed, and the results are used to determine the Golay pilot length used in the test.

The blue lines in Fig. 3.16 show the normalized mean-square error (MSE) of the true channel matrix and the frequency-flat channel estimation in the FPGA versus SNR for different Golay pilot lengths in the flat i.i.d channel. The normalized MSE of the channel estimations decreases as the Golay pilot length increases. From the figure, one can also notice that when the SNR is lower than 17 dB, 15 dB, and 13dB for the lengths of 8, 16, and 32, respectively, the MSE increases faster than the higher SNR in the log scale. The change in the slope is due to the increase in pilot synchronization failure. However, by increasing the Golay pilot length to 64, the channel estimation MSE reduces to lower than 3×10^{-2} for the

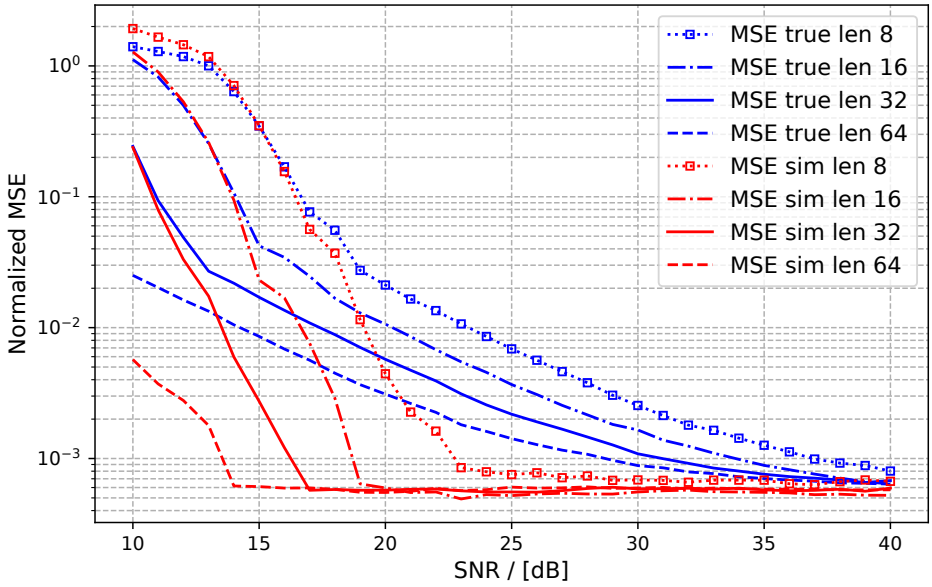


Figure 3.16: Frequency flat normalized channel estimation MSE vs SNR with different Golay pilot length. Blue lines show the normalized MSE of the true channel matrix and the channel estimation in the FPGA; red lines show the normalized channel estimation MSE of the simulator and FPGA.

SNR even lower than 15dB, and the possibility of pilot synchronization failure becomes much lower.

Based on the result shown in Fig. 3.16, the Golay pilot length of 64 is chosen for the system emulation to balance the resource usage and performance.

3.4.2 Performance evaluation of the Spine generator

The bit-error rates (BER) are evaluated under different SNR scenarios with 32 antennas and 4 users in the system. The channel models are the flat i.i.d channel and the Rician channel with $K = 10dB$. The results of BER vs. SNR for the FPGA emulation system are shown in Fig. 3.17. The figure shows the BER ranging from 10^{-6} to 10^{-3}. Inset a) shows the BER of the quadrature phase-shift keying (QPSK) modulation, and inset b) shows that of the 16-QAM. The result shows that at SNR 11.9dB for the QPSK modulation and 19dB for the 16-QAM modulation, the BER reaches 10^{-3} in the flat i.i.d channel. For both QPSK and 16-QAM modulation schemes, the emulation BER is very close to the simulation BER produced by the floating-point golden model in the given SNR range.

The performance of the Spine generator is also evaluated with a larger system, which has 64 antennas and 4 users, under flat i.i.d, Rician channel with $K = 10$dB, LoS, and QuadMMLoS channel models. The FPGA performs the MRC beamformer, and the Python

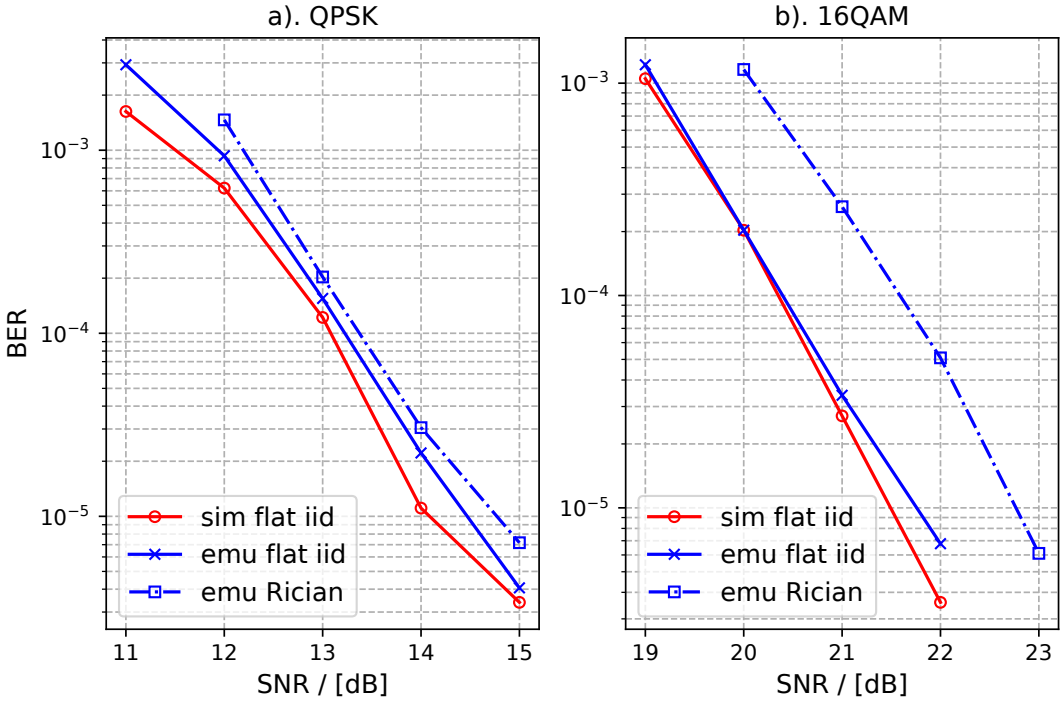


Figure 3.17: BER vs SNR for 32 antennas 2 users simulation and emulation results for a). QPSK modulation scheme and b). 16-QAM scheme.

model simulator performs the beamspace algorithms and the frequency selective decorrelation stage.

The SINR vs. SNR with various channel models after demodulation is shown in Fig. 3.18. The results of Python simulation are represented with solid lines, and the results of FPGA emulation are represented with dashed lines. The results demonstrate the functionality of the system for the four selected channel models, and validate the given generator parameters. Moreover, the emulation results are close to the corresponding floating-point simulation results for flat i.i.d, Rician, and LoS channel models in the high SNR range. The small performance difference between simulation and emulation under QuadMMLoS is due to the gain difference for users included in the channel model, which makes the demodulation of this channel model more sensitive to the channel estimation accuracy and quantization errors.

Fig. 3.18 shows the linear change of the SINR with respect to that of SNR for the given channel models, when SNR is larger than 12dB. However, when SNR is lower than 12dB for flat i.i.d and Rician channel models, and 11dB for LoS and QuadMMLoS channel models, the SINR after demodulation drops abruptly. The performance degradation is due to the degradation of the channel estimation at the low SNR range. Here, residual noise after cross-correlation of the Golay sequence pilots might false trigger the peak detection algorithm for some channels, which leads to completely incorrect channel estimation for those channels.

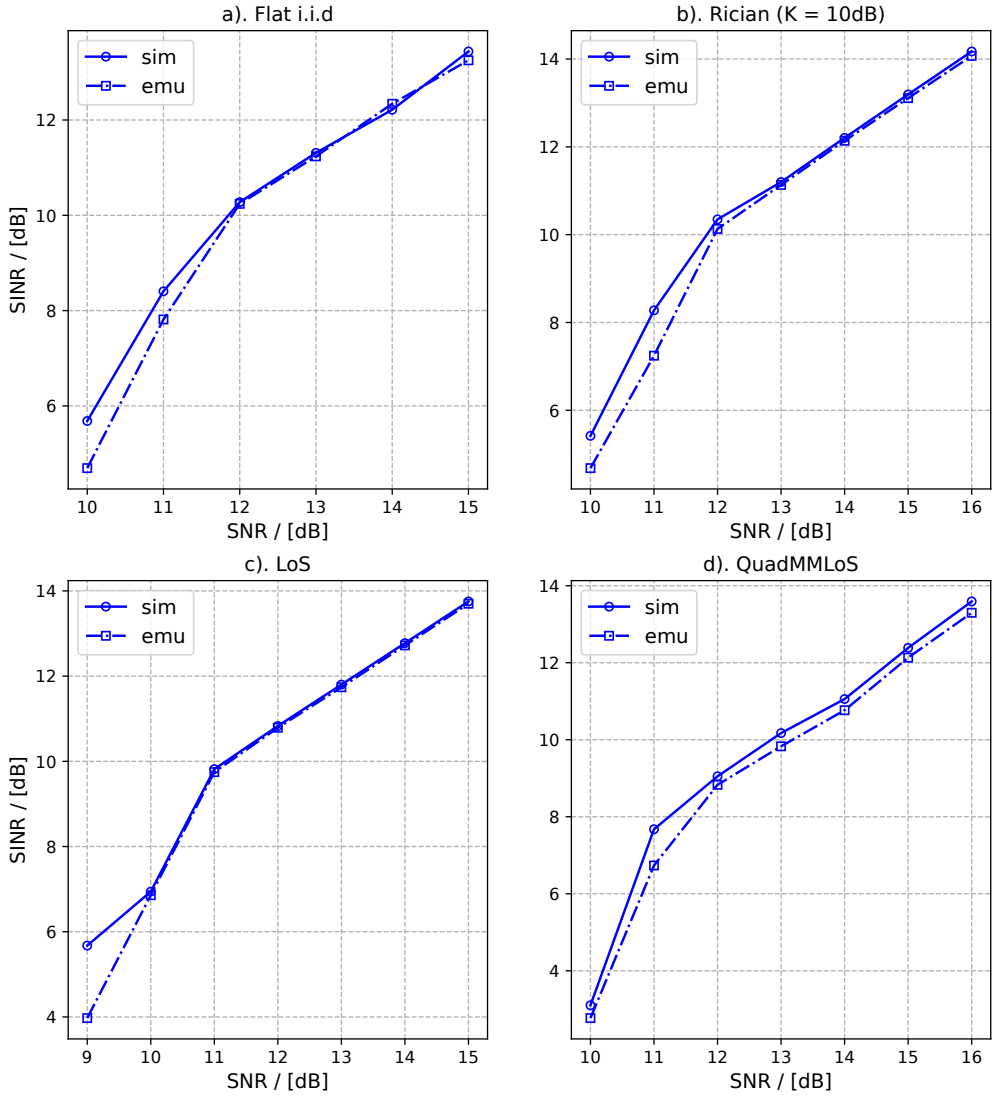


Figure 3.18: SINR vs SNR under a) Flat i.i.d, b) Rician with K = 10dB, c) LoS, d) QuadMMLoS channel models with 64 antennas and 4 users in the system. The results of Python simulation are represented with solid lines, and the results of FPGA emulation are represented with dashed lines.

The larger SINR difference between simulation and emulation at the low SNR is caused by a shorter peak checking interval implemented in the hardware due to timing and hardware resource considerations, which leads to a higher false peak-trigger possibility when SNR is low.

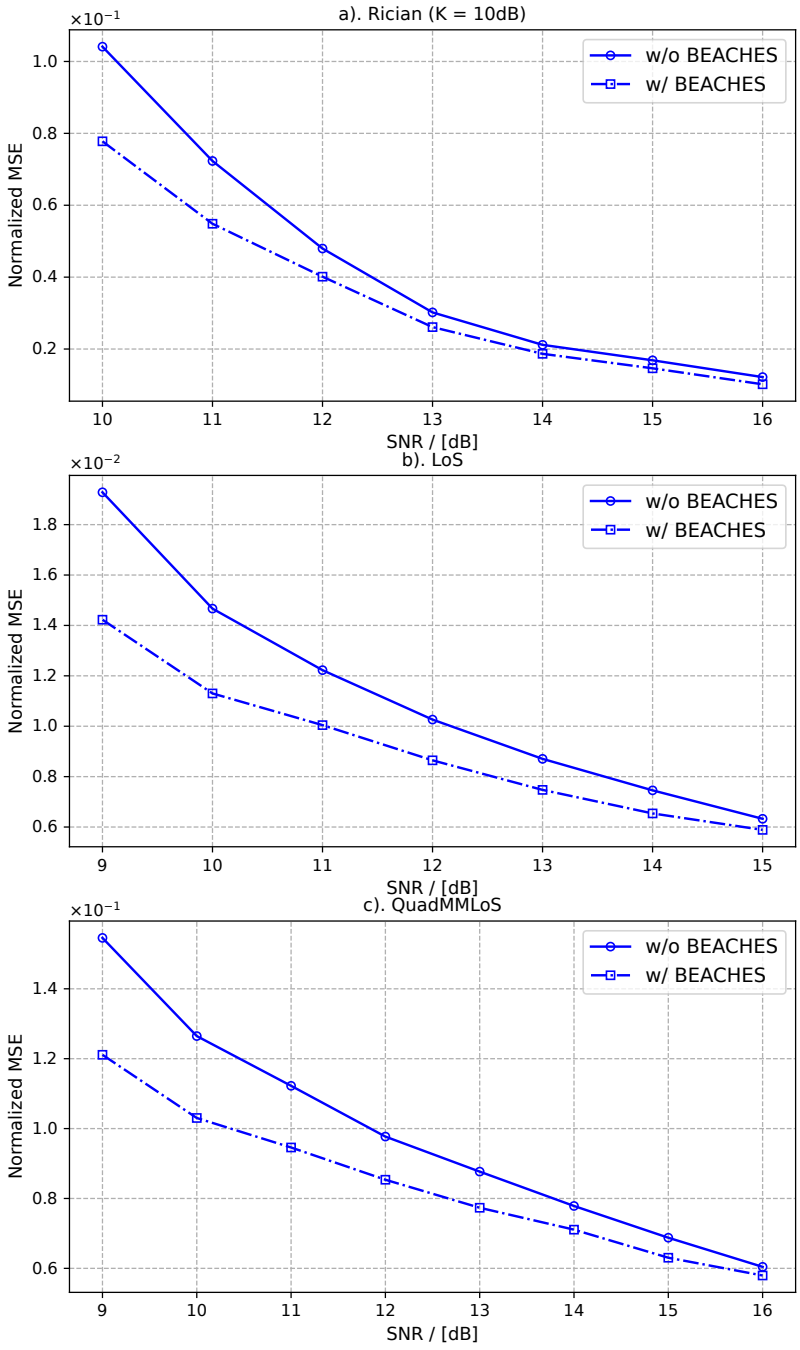


Figure 3.19: Channel estimation NMSE with and without calibration and BEACHES algorithms with 64 antennas and 4 users in the system. For all insets, the NMSE of the FPGA-emulated channel matrix is shown as the dashed line, while the NMSE of the denoised one is shown as the solid line.

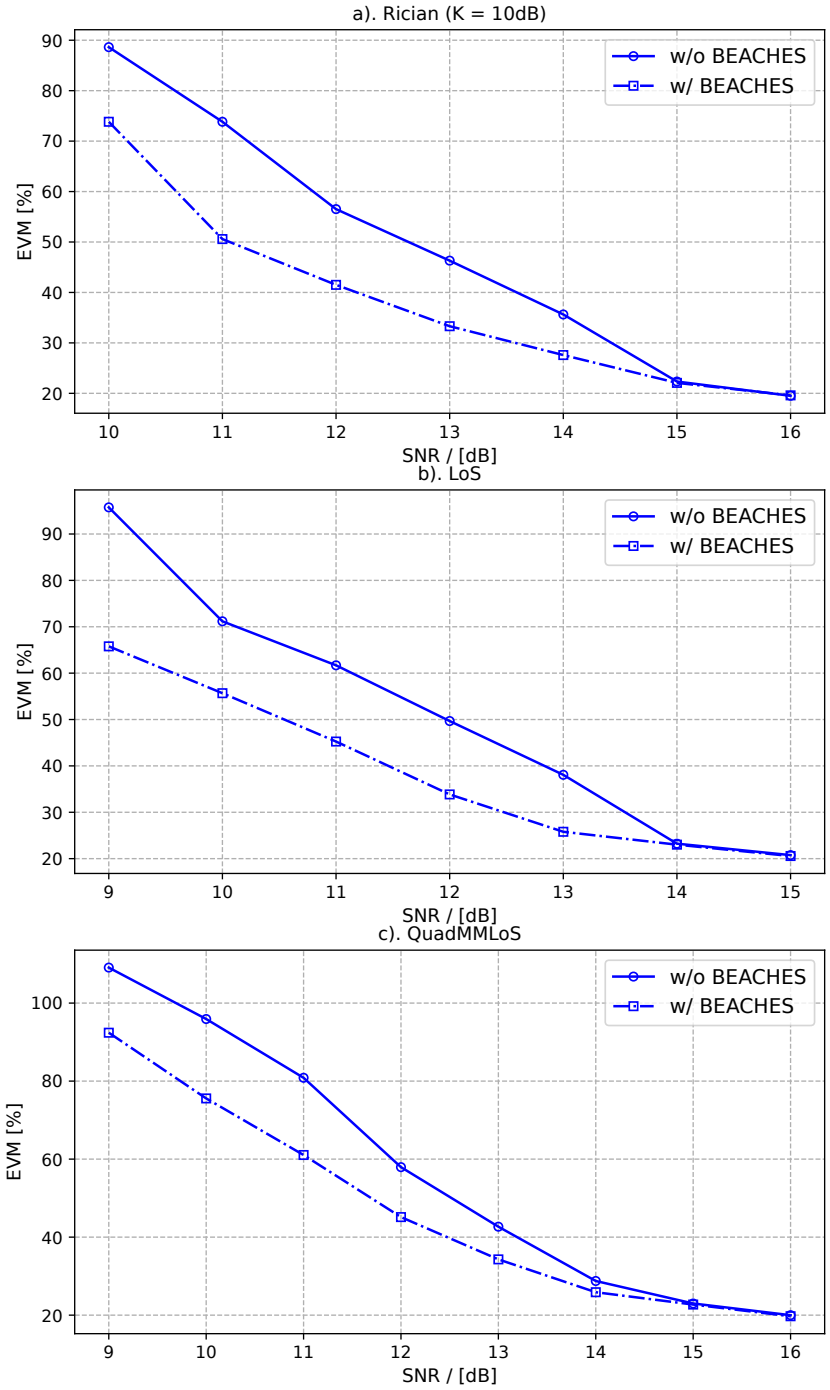


Figure 3.20: EVM with and without the beamspace algorithms with 64 antennas and 4 users in the system.

3.4.3 Channel estimation performance evaluation

Fig. 3.19 shows the MRC channel estimation NMSE with conventional estimation and with the BEACHES algorithms with respect to SNR under the Rician, LoS, and QuadMMLoS channel models. In this evaluation, channels are assumed to be ideal, and no phase offsets between channels. The NMSE is the error measurement between the channel matrix estimation and the simulator-generated ground truth. The Golay pilot length is set to 64 due to its good channel estimation performance evaluated in [23]. For all insets, the NMSE of the FPGA-emulated channel matrix is shown as the dashed line, while the NMSE of the denoised one is shown as the solid line. Due to the randomness property of the flat i.i.d channel model, the denoising algorithm is not applicable. For this reason, the flat i.i.d channel is not considered in this section.

The channel estimation performance for Rician, LoS, and QuadMMLoS channel models are evaluated under SNR ranging from 10dB to 16dB, 9dB to 15dB, and 9dB to 16dB, respectively. From Fig. 3.19 one can notice that BEACHES algorithm can effectively denoise the estimated channel matrix for all three channel models. The NMSE decreases notably after applying BEACHES algorithm, compared to the NMSE with Golay estimation.

Fig. 3.19 also shows that as SNR decreases, the improvement of the NMSE after denoising by BEACHES algorithm increases, demonstrating the algorithm's efficacy in improving the channel estimation performance in the SNR range of interest. With BEACHES algorithm, the channel estimation NMSE can be improved by up to 25.3% for the Rician channel model, 26.2% for the LoS channel model, and 21.6% for the QuadMMLoS channel model.

To further evaluate the impact of the improvement of the channel estimation on the system, the normalized root-mean-square (RMS) EVM is measured for the three channel models after demodulation. The normalized RMS EVM is measured by the Python simulator as follows: 1) load the estimated channel matrix from FPGA; 2) perform the BEACHES algorithm to denoise the channel matrix; 3) rerun the simulation but with the channel matrix from both 1) and 2); 4) calculate the RMS EVM.

Fig. 3.20 shows the EVM vs. SNR with conventional and with the BEACHES algorithm for the three channel models. The solid lines in the figure show the EVM after demodulation without applying BEACHES algorithm, while the dash-dot lines indicate the EVM of the demodulation after applying BEACHES algorithm. For all models, the EVM decreases after applying the BEACHES algorithm. With SNR higher than 15dB for Rician and QuadMMLoS channel models, and with SNR higher than 14dB for LoS channel model, the EVM with and without the BEACHES algorithm are almost the same, and the improvement of the channel estimation MSE has only a small impact to the system; when the SNR is lower than 15dB and 14dB, respectively, the EVM decreases notably after applying the BEACHES algorithm. The EVM improvement is more pronounced for lower SNR than for high SNR. The improvement of the EVM after the calibration and BEACHES algorithms can be up to 23.3% for the Rician channel, 29.8% for the LoS channel, and 20.4% for QuadMMLoS channel model.

The channel estimation evaluation on the NMSE and the EVM after demodulation proves

the effectiveness of BEACHES algorithm to the conventional massive MIMO processing. The necessity of integrating BEACHES algorithm into the system further indicates the importance of the adaptability of massive MIMO systems to other algorithms.

3.4.4 Impact of channel phase offsets

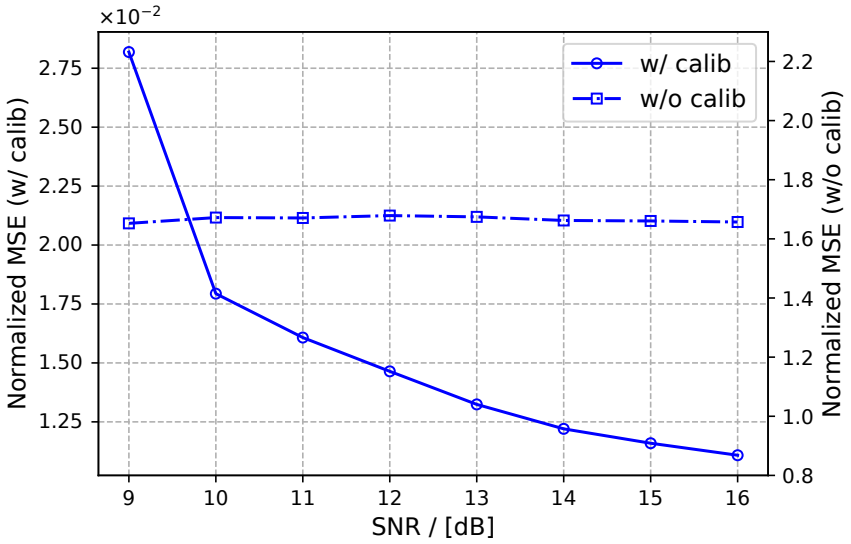


Figure 3.21: The NMSE of the denoised channel estimation w/ and w/o in-situ calibration algorithm with channel nonidealities.

In this section, the impact of channel phase offsets on BEACHES and the importance of the in-situ calibration algorithm are evaluated. In this section, a random phase offset $\Delta\theta \in [-\pi, \pi)$ is applied to each antenna in channel simulations to model mismatches in the delay of the RF chain between antennas, and LoS channel model is considered. Fig. 3.21 shows denoised FPGA emulated channel matrix NMSE with and without applying in-situ calibration when per-channel phase offsets are modeled. The result shows that without channel calibration, the NMSE stays at 1.7 regardless of the change of SNRs. This shows that uncalibrated channel phase offsets break the beamspace sparsity of the channel, rendering the beamspace channel denoising algorithms useless. This emphasizes the need for channel calibration for successful beamspace channel denoising.

Fig. 3.22 shows the MSE between the estimated channel phase offset by the in-situ calibration algorithm and the ground truth with SNR ranging from 9dB to 16dB. Though the MSE increases as SNR decreases, the MSE is less than 2%. The solid line in Fig. 3.21 shows the NMSE of the denoised FPGA emulated channel matrix with both in-situ calibration and BEACHES applied. The NMSE drops from 1.7 to the order of magnitude of 10^{-2}, which

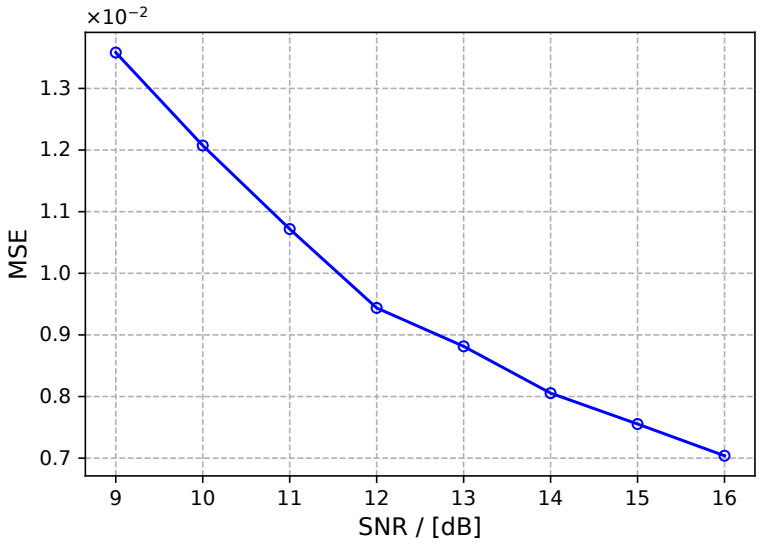


Figure 3.22: The MSE of the channel phase offset estimation by the in-situ calibration algorithm.

agrees with the result in Fig. 3.19 inset b), where ideal channels are assumed. The results indicate the effectiveness and importance of the in-situ calibration for beamspace algorithms.

3.4.5 Ease of Spine generation

The advantages of the generator-based design can be considered from two aspects: Straightforward design validation, and portability to various MIMO system implementations.

The powerful Chisel extension for DSP design, ACED, and features of Scala as the host language make the validation of the DSP design easier. The Spine generator can support both fixed-point and floating-point data types in the RTL simulation by declaring generic data types in the generator parameters. By initiating a generator parameter class with `DspReal()` data type, the RTL simulation uses floating-point data type, which is used for the generator functionality validation; by initiating a generator parameter class with `FixedPoint()` data type, the RTL simulation uses the fixed-point data type defined in the parameter, which can simulate the quantization impact on the system performance. All submodule generators in this work are validated by this method.

Fig. 3.23 shows two Spine DSP core implementations with different system parameter settings. Inset a) is the FPGA implementation with 4 antennas and 2 users, while inset b) is the implementation with 4 antennas and 4 users. The Spine generator can generate the RTL for those 2 implementations by simply changing the generator parameters. Compared with designing instances and redesigning the top-level integration, which takes a long time, the Spine generator generates the new RTL for different parameters in just 9 minutes and 17 seconds. The fast generation of new designs shows the high portability of the Spine generator.

a). 4 antennas and 2 users

b). 4 antennas and 4 users

Figure 3.23: The Spine DSP core generated by the Spine generator implemented on Xilinx VCU118 FPGA with a). 4 antennas and 2 users; b) 4 antennas and 4 users.

The power consumption estimation for the 4 antennas and 4 users implementation given by Xilinx power estimator [124] is 5.4W, with 3.1W dynamic power and 2.3W static power.

3.4.6 Discussion on scalability limitations

Scaling the real-time processing in the Spine, is constrained similarly to other two-stage systems. In this discussion, we assume the testbed is implemented with VCU118 FPGAs.

Interconnection bandwidth

The QSFP interface on VCU118 has maximum 112Gbps (4×28 Gbps) bandwidth. With the implementation parameters shown in Table 3.1, and by applying Eq. 3.15, the maximum number of users the system can support using Spine real-time processing is 28, assuming 8b/10b encoding and without considering the hardware utilization limitation.

Interconnection delay along the daisy chain

This work uses a daisy chain architecture to sum up all Spine results together, so the data transmission latency between Spine modules limits the number of modules that can be chained together. Assume the interconnection latency between 2 Spines is 100ns, including the transmission overhead, and assume each Spine has a 2K Byte buffer used for buffering the data coming from the upper steam Spine. By using the parameters in Table 3.1, the total number of Spines that the system can support for real-time operation is 50, which means the system can scale up to 200 antennas.

FPGA utilization

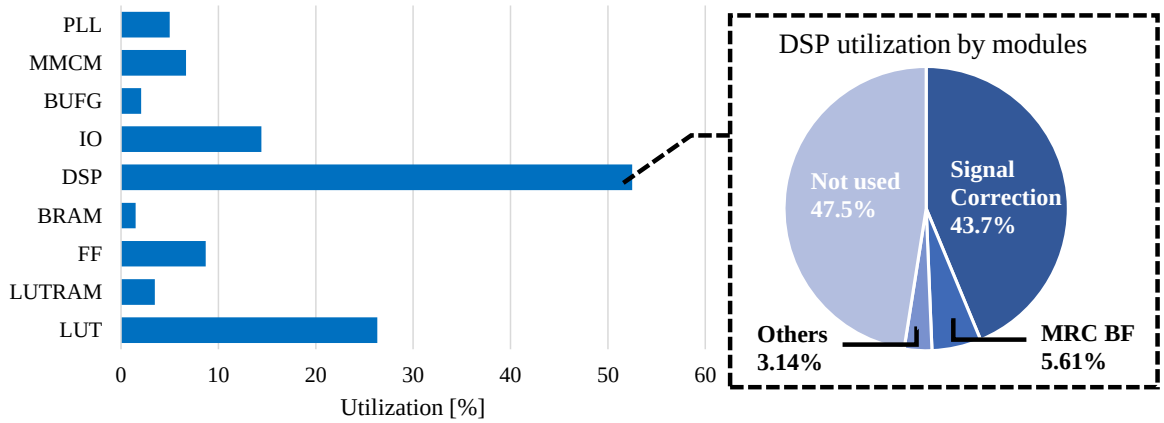


Figure 3.24: FPGA utilization breakdown of the 4 antennas 4 users per subarray FPGA implementation.

Fig. 3.24 shows that the bottleneck of the FPGA resources on VCU118 is the DSPs. The design with 4 antennas and 4 users utilized 53% of DSP resources. Signal correction blocks take 43.7% of DSP utilization and the MRC beamformer takes 5.6%. Although further increasing the antenna count in the Spine is hard, scaling up the number of users is possible, since it mainly affects the beamformer utilization instead of the signal correction utilization. Based on Fig. 3.24, the system can support up to 28 users and 4 antennas, if the FPGA is fully utilized.

3.4.7 Summary

Table 3.2 summarizes our work and some state-of-the-art massive MIMO BS implementations. ArgosV2 [113] can scale up to 96 antennas and support 32 users, which is implemented with a distributed architecture and the conjugate beamforming method. LuMaMi [75] is a BS

testbed that supports up to 100 antennas and 12 users with a centralized architecture, and it is implemented with MRC, ZF, and regularized ZF (RZF) beamforming methods. Both ArgosV2 and LuMaMi use OFDM modulation with a signal bandwidth 20MHz. The work in [90] implements a successive over relaxation (SOR) based beamforming algorithm with the assumption of perfect channel estimations on FPGA. The hardware implementation contains 64 antennas and 8 users with only simulation and synthesis results reported. Compared with [75] and [90], our work currently only contains uplink baseband processing, but is highly portable and parameterizable, and can be implemented either on FPGAs or ASICs. The scalability of the system is enabled by the distributed architecture, and the extension for emerging algorithm implementation is enabled by the modular design. This work has also been tested in a massive MIMO system with more antennas and users, and evaluated under various channel models than with our earlier works [23] and [24]. The efficient implementation of the beamspace algorithms demonstrates the algorithmic adaptability of this work. The fully utilized FPGA power estimation per user per antenna for this work is 0.076W/user/antenna. In addition, due to the portability of this work, the generator can be potentially implemented in an ASIC, yielding more antennas and users in a single chip, along with better energy efficiency.

Table 3.2: Comparison

	This work	Prev. work [23]	ArgosV2[113]	LuMaMi[75]	RIVF'19 [90]
Platform	FPGA/ASIC	FPGA/ASIC	SDR	FPGA	FPGA
Portable	Yes	Yes	No	No	No
Adaptability	High	High	-	-	-
# Users	4	2	32	12	8
# Antennas	64	32	96	100	64
Modulation	QAM	QAM	OFDM	OFDM	QAM
Distributed	Yes	Yes	Yes	No	No
Bandwidth	200MHz[1]	200MHz	20MHz	20MHz	935MHz[2]
Method	Two-stage BF w/ BEACHES	Two-stage BF	Conjugate BF	MRC/ZF/RZF	SOR
CHEST[3]	Yes	Yes	Yes	Yes	No

[1] The signal bandwidth is the FPGA implementation bandwidth. The power estimation is based on the FPGA implementation of a single Spine with parameters shown in Table 3.1.
[2] This is the maximum FPGA implementation clock frequency, not the signal bandwidth. The signal bandwidth is not given.
[3] Short for channel estimation.

Chapter 4

Complex-valued Neural Network Aided Channel Estimation

In Chapter 3, a beamspace domain channel estimation method is integrated into the massive MIMO system to improve the channel estimation performance degraded by the hardware datapath quantization noise. This chapter introduces a machine learning (ML) aided channel estimation method to improve the estimation performance with strong interference, which also shows the potential of ML methods for integrated signal processing of massive MIMO systems.

4.1 Motivations

In the previous chapter, a distributed massive MIMO system has been evaluated and analyzed with various channel models. It is under the assumption that BS and UEs are the only signal-generation sources in the environment. However, with the increasing number of mobile and electronic devices enabled by innovations in wireless communication technologies, the channel environment has become more complex and noisier. Although massive MIMO communication systems can allocate spectrum and channels to minimize interference from UEs within the same system, interference can still occur and degrade the performance of wireless communication systems. That interference can stem from various sources, which is shown in Fig. 4.1, such as signals from neighboring channels, electrical devices, distortions caused by frontend hardware, and jamming.

The interference sources can generate either narrowband or wideband interference inside the desired signal bandwidth and further distort the desired signals. Overall, despite the remarkable advancements in wireless technology, challenges remain in managing interference and ensuring reliable performance in increasingly complex and noisy environments.

As discussed in Chapter 1, accurate CSI estimation is crucial for the high performance of massive MIMO systems. The presence of interference can degrade or even corrupt the channel estimation. The interference caused by jamming is one of the main reasons leading

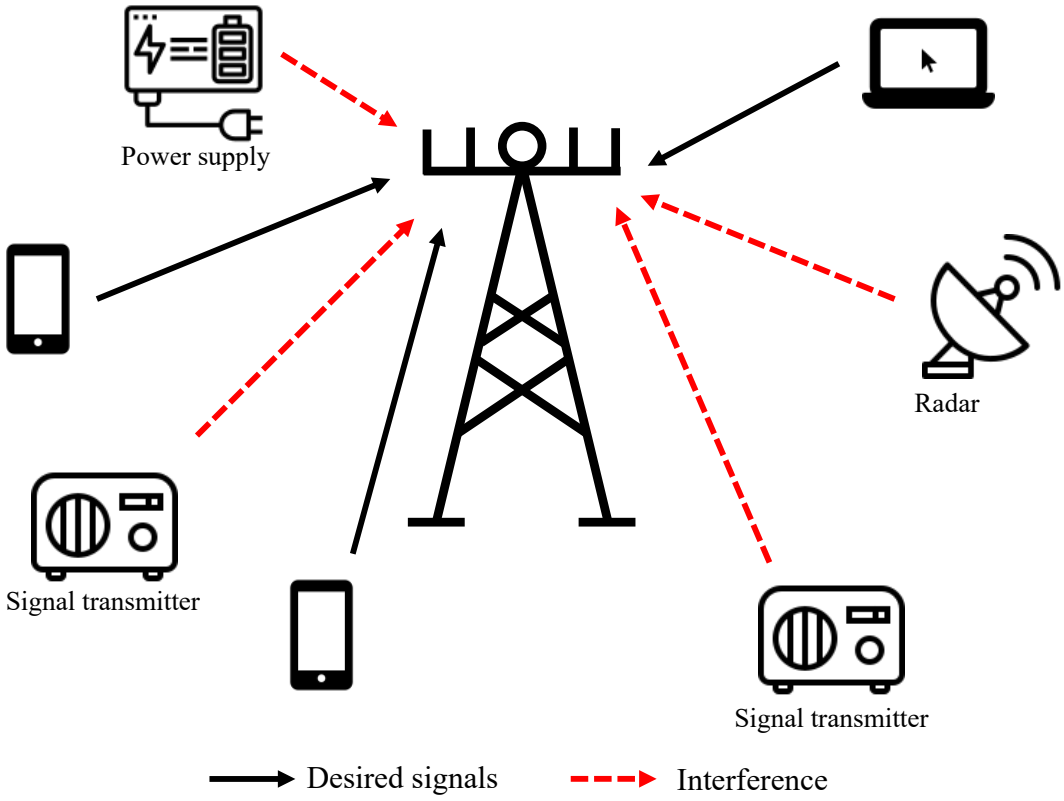


Figure 4.1: The sources of interference in the massive MIMO systems.

to this degradation.

Jamming is a deliberately designed or generated interference that intentionally or unintentionally interferes with or blocks normal wireless communication. Despite advancements in high-capacity and low-interference techniques, massive MIMO systems remain sensitive and vulnerable to radio jamming due to several reasons [96]:

- The ease of generating jamming with the help of commercial programmable radio devices and by various sources that can cause jamming, such as UEs from other network cells.
- The chain reaction caused by signal distortion in the physical (PHY) layer, which disables decoding, error correction, or other functions in subsequent stages of the PHY layer or Media Access Control (MAC) layer.
- The limitations of effective anti-jamming techniques.

Common jamming types for massive MIMO systems include generic jamming attacks, synchronization jamming attacks, and channel estimation jamming attacks [96]. The jamming

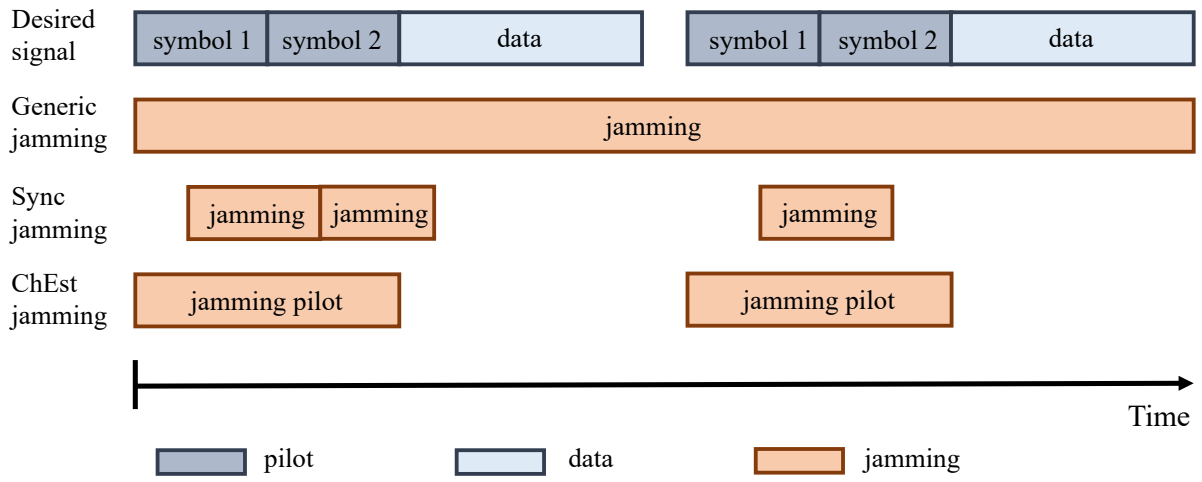


Figure 4.2: The jamming regions relative to the desired signal packet. 1). The generic jamming attack can target the whole desired signal packet; 2). The synchronization jamming attack targets mostly on pilot symbols; and 3). The channel estimation jamming attack focuses on sending jamming pilots to distort or nullify the desired signal pilots.

regions relative to the desired signals are shown in Fig. 4.2. Generic jamming attacks typically involve high-power, constant, or periodic signals that can cover most or all of the targeted channel bandwidth, even for the wideband signals enabled by mmWave [53]. Synchronization jamming attacks aim to disrupt either timing synchronization or frequency synchronization of massive MIMO communication systems by sending specifically designed signals that interfere with the pilot bandwidth or camouflage pilot symbols to mislead synchronization [95, 71]. Additionally, jamming attacks can focus on disabling channel estimation of the multi-user massive MIMO systems by distorting or even nullifying pilots, leading to completely inaccurate CSI estimation [19]. Besides those, pilot contamination can also be one of the jamming sources in massive MIMO systems. Due to the finite size of the codebook, the users in neighboring cells may share the same pilots as the users in the current cell. The same pilot sent from different cells causes inter-cell interference and disturbs the BS channel estimation.

For massive MIMO communication systems, some state-of-the-art anti-interference techniques include:

- Regularized zero-forcing receiver filters to null the jamming signals using the jointly estimated legitimate user and jamming user channels, which considers a constant jamming scenario happening both on pilot and data transmission phases [27].
- The random matrix theory-based jamming rejection technique, which rejects the jamming signal by projecting the received signal to the legitimate users eigensubspace identified from the received signal covariance matrix [122].

- The blind jamming mitigation algorithm, which uses a linear spatial filter to cancel the jamming signals from the unknown jammers and recover the desired signal from the legitimated sender, and this algorithm is applicable to scenarios that contain multiple jammers in the system [140].

- The blind channel separation method for channel estimation under active jamming attack, where malicious users send the jamming symbols that are correlative to those of the legitimate users. The method is implemented with the alphabet extractor and alphabet decomposition to separate the channel, and then estimate the channels of legitimate users and malicious users separately [15].

However, the disadvantages of the existing anti-interference techniques listed above make conventional approaches hard to implement in real massive MIMO applications. In [27], the proposed regularized zero-forcing method requires the knowledge of the signal statistic of the jamming signals, which needs extra steps to estimate. In [122], the random matrix theory-based method needs the distinguishable power difference between the signal and jamming in order to identify the subspace of user signals. Although the methods proposed in [27, 122, 140] can null the jamming signals in the uplink, the user CSI is still unknown to the communication system, which creates difficulties for the downlink channel prediction. Extra works are still necessary to estimate the user CSI. The user CSI can be estimated using the blind channel separation method, but the active jamming signal in [15] is required to be specially designed with the prerequisite of the known user pilot and time-synchronized with the user pilot frame, is a strong assumption and hard to realize.

Inspired by the success of ML on image denoising, restoration, or prediction [28, 47], many ML-based massive MIMO channel estimation methods are proposed. In [144], a DNN-based channel estimation method is introduced for fewer-bit ADC massive MIMO systems to realize the channel estimation with very low-resolution inputs and low pilot overheads. Authors in [137] propose a CNN-autoregressive predictor and CNN-recurrent neural network (RNN) to get better channel estimation with the channel aging problem. The ML-based method proposed in [63] demonstrates the robustness of the channel estimation using ML to Doppler effects with three different deep learning networks to refine the per-antenna LS channel estimation solution. For the signal jamming problem, authors in [135] propose a jamming classification method using CNN. The CNN classifier shows the strong separation of 5 different jamming types, which also shows the potential for ML in signal strong interference problems. With the success of the state-of-the-art ML-based massive MIMO channel estimation methods under complex channel scenarios and jamming classification, the channel estimation under strong interference should also have the potential to be solved by using ML methods.

In this chapter, a complex-value neural network (CVNN) aided channel estimation method is proposed to improve the channel estimation of the massive MIMO systems discussed in Chapter 3 under strong wideband interference or jamming scenarios, where the frequency-domain symbols are strongly distorted. The contributions of this work include:

- propose a complex-valued convolutional neural network with its preprocessing and postprocessing for the MRC beamforming stage channel estimation of the massive MIMO systems discussed in Chapter 3, assuming strong wideband interference existing in the pilot and distorting the channel estimation.
- discuss and analyze different neural network architecture choices to the impact of the channel estimation performance.
- evaluate the performance and robustness of the proposed method with the change of center frequency of the interference. And the results show that the mean square error of the proposed channel estimation can be improved 316% compared with the traditional method with less pilot overhead.

4.2 Background

In this section, different design choices of ML methods, specifically for massive MIMO communication systems, are reviewed and discussed. The discussion focuses on the types of neural network architectures, complex-valued neural networks versus real-valued neural networks, and optimization methods that can be used for training neural networks.

4.2.1 Types of Network Architectures for Communication Systems

Wireless communication systems share similarities with image, voice, and speech processing, particularly the locality property embedded in the signal. As a result, researchers are exploring the potential of ML algorithms to enhance the performance of wireless communication systems.

According to [107], the three most commonly used neural networks for current 5G techniques are fully-connected neural networks (FCNN), convolutional neural networks (CNN), and recurrent neural networks (RNN). These networks are suitable for different purposes based on the properties of applications they are applied.

Fully-connected Neural Network (FCNN)

The architecture of FCNN, as shown in Fig. 4.3, consists of the input layer, the hidden layers, and the output layer. In FCNN, each neuron in one layer is connected to every neuron in the next layer. In other words, at each layer, all inputs of that layer contribute to each one of the outputs at that layer. Suppose that a fully-connected neural layer has P number of input $\boldsymbol{x} \in \mathbb{R}^P$ and Q number of output neurons $\boldsymbol{y} \in \mathbb{R}^Q$, and the weights of layer is $W \in \mathbb{R}^{Q \times P}$ and the bias is $B \in \mathbb{R}^{Q \times P}$, then the fully connected neural layer can be expressed as

$$\boldsymbol{y} = f\Big(W\boldsymbol{x} + B\Big), \tag{4.1}$$

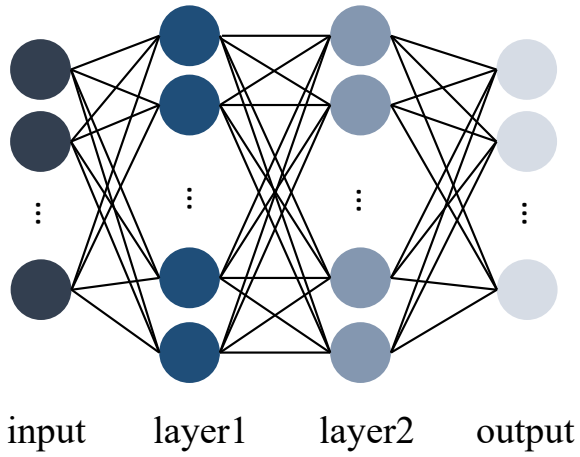


Figure 4.3: The architecture of an FCNN example, which consists of the input layer, hidden layers, and the output layer.

where $f(\cdot)$ is the non-linear activation function.

Since all the neurons are tightly connected, important features can be extracted, and irrelevant information can be suppressed through the feed-forward operation. FCNNs have been proven to be effective approximators for general finite dimensional space mappings [44], and the method proposed in [144] employs this property to estimate high-resolution CSIs with fewer-bit ADC at frontends, by approximating the mapping from the highly quantized measurement to the true channels.

Convolutional Neural Network (CNN)

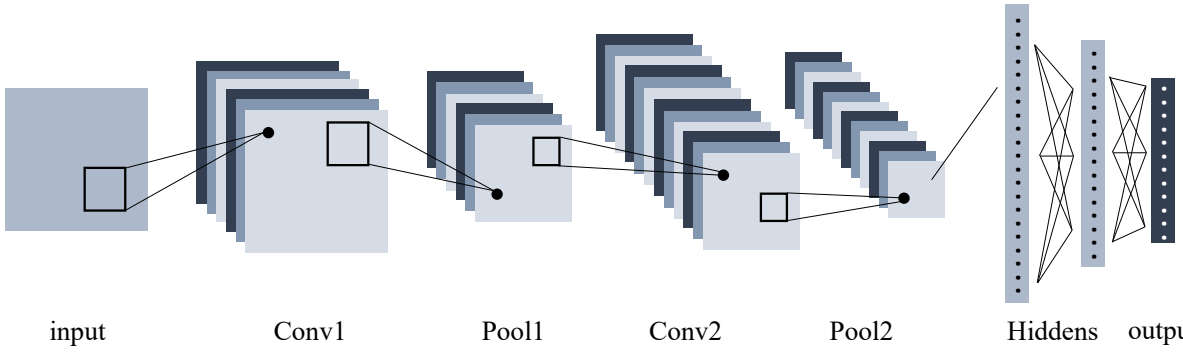


Figure 4.4: The architecture of a CNN example, which consists of the convolutional layers, pooling layers, and fully connected layers.

Fig. 4.4 shows the architecture of CNNs. A basic CNN contains convolutional layers, pooling layers, and fully connected layers. Compared with FCNNs where all the input neurons will contribute to the output, CNNs exploit the spatial correlation of input to extract and abstract features by using convolutions and pooling, and then use fully connected layers for classification or regression. Due to the shorter correlation range of CNNs, the number of parameters that CNN needs to be learned is largely reduced compared to FCNN models of the same depth, which makes CNN suitable for learning complex models.

For a 2-dimensional convolutional layer with a square input $\boldsymbol{x} \in \mathbb{R}^{N\times P\times P}$, a square weight $\boldsymbol{w} \in \mathbb{R}^{M\times N\times L\times L}$, the striding step $s \in \mathbb{R}$, one-side padding $d \in \mathbb{R}$ and bias $\boldsymbol{b} \in \mathbb{R}^{M\times Q\times Q}$, the square output of the convolutional layer $\boldsymbol{y} \in \mathbb{R}^{M\times Q\times Q}$ is

$$\boldsymbol{y} = f\Big(\boldsymbol{x} * \boldsymbol{w} + \boldsymbol{b}\Big), \tag{4.2}$$

where $Q = \frac{P+2d-L}{s} + 1$, and $f(\cdot)$ is the non-linear activation function. According to the above equation, a proper padding size needs to be picked in order to get the desired output size. Some commonly used pooling methods are maximum pooling and average pooling.

The success of AlexNet in ImageNet competition shows the power of CNNs in image processing [57], and highlights the learning ability for complex models. CNNs can also be expanded to process other data types that exhibit strong spatial correlation, such as preambles, pilot signals, and channel matrices in wireless communication systems. Furthermore, CNNs can be useful for estimating CSIs under complex environments by exploiting the correlation embedded in the input.

Recurrent neural network (RNN)

Another commonly used neural network in wireless communication is RNN, and Fig. 4.5 shows a basic architecture for RNN. The basic architecture of RNN is very similar to FCNN, which contains an input layer, hidden layers, and an output layer. However, different from the FCNN, the state of the hidden layers $\boldsymbol{h}(t)$ is time-variant, and the output is also time-dependent. The state of the hidden layers is determined by the input $\boldsymbol{x}$, the input layer weights $\boldsymbol{h_{in}}$, the update weight $\boldsymbol{w}$, the hidden layer state of the previous timestamp $\boldsymbol{h}(t-1)$, and bias $\boldsymbol{b_h}$

$$\boldsymbol{h}(t) = f_h\Big(\boldsymbol{h_{in}}\boldsymbol{x}(t) + \boldsymbol{w}\boldsymbol{h}(t-1) + \boldsymbol{b_h}\Big), \tag{4.3}$$

and the output $\boldsymbol{y}(t)$ depends on the current hidden layer state, the weights of output layer $\boldsymbol{h_{out}}$, and bias $\boldsymbol{b_y}$

$$\boldsymbol{y}(t) = f_o\Big(\boldsymbol{h_{out}}\boldsymbol{h}(t) + \boldsymbol{b_y}\Big), \tag{4.4}$$

where $f_h(\cdot)$ and $f_o(\cdot)$ are non-linear activation functions.

According to Eq. 4.3 and Eq. 4.4, one can easily see that RNN exploits the temporal correlation of the input data, which has strong potential for solving time-dependent and high

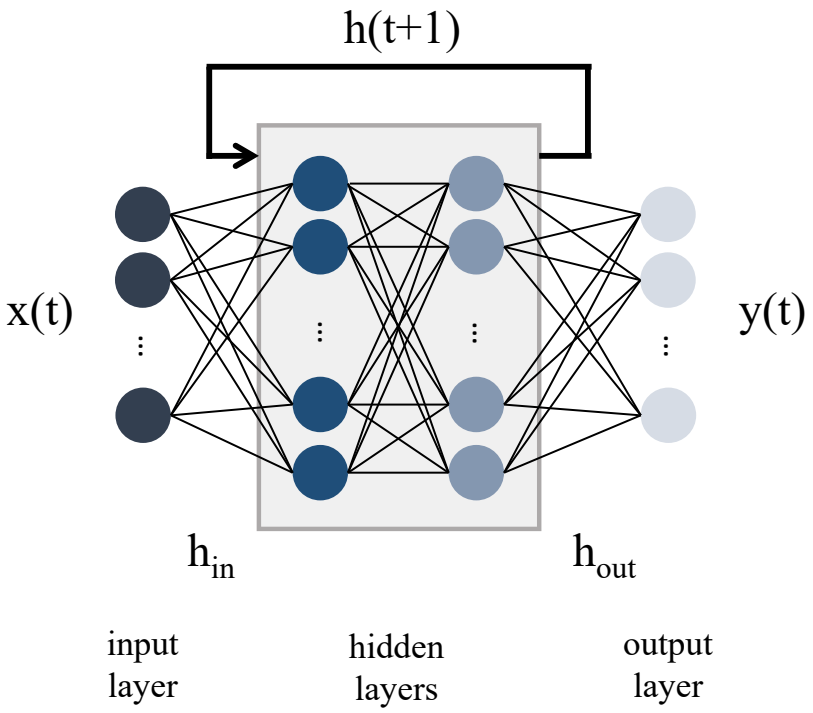


Figure 4.5: The sources of interference in the massive MIMO systems.

temporal-correlated problems. The examples in wireless communication systems that can apply RNN models are channel aging estimation and Doppler effect estimation, where the temporal correlation of the pilots can be exploited to estimate channel aging parameters and Doppler shift frequency [137]. An RNN-based method is proposed by [130] to explore the inter-packet correlation to improve the CSI feedback resolution for FDD massive MIMO systems. However, due to the sequential order of the training data and the feedback procedure, training RNN models can be much more computationally intensive than training FCNN and CNN models.

4.2.2 Complex-valued vs. Real-valued

Many state-of-the-art ML architectures for image, voice, or speech signal processing applications use real-valued input data. Even when using frequency domain information instead of spatial or time domain information, such as FFT pre-processed signals, the input data of the neural network is typically the power spectrum of the raw data, which remains in the real domain.

However, in wireless communications, the demodulation of the radio frequency signals involves the use of quadrature signals, which consist of two signals of equal frequency with a 90-degree phase separation. As a result, unlike image, voice, or speech data, the signals used

in wireless communication are represented by complex numbers. Moreover, the phase of the quadrature signals is crucial for the successful establishment of wireless communication, and the phase correlations embedded in the wireless communication signals can be useful for ML algorithms.

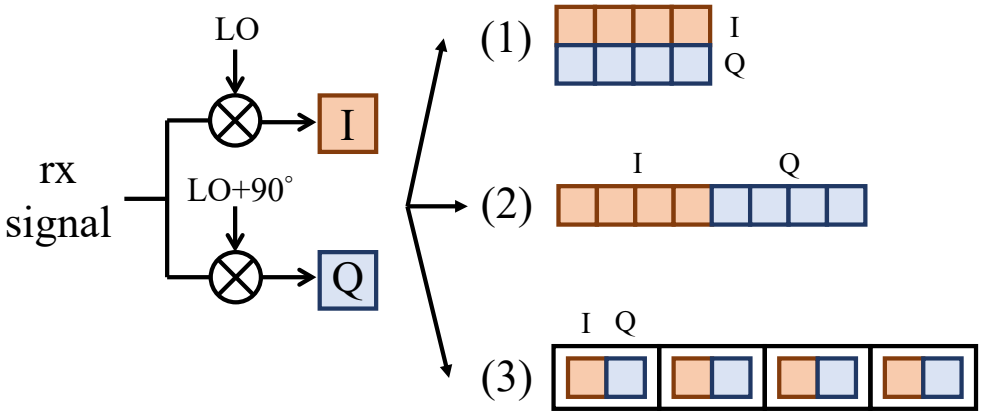


Figure 4.6: Ways of the RF signal conversion for neural network input: (1) treat I and Q as two independent channels for the real-valued neural network; (2) concatenate I and Q into a single vector for the real-valued neural network; (3) use the complex-valued vector of $I + Qj$ for a complex-valued neural network.

Real-valued neural networks can also be used for processing complex-valued signals by converting the input from the complex domain to the real domain. The conversions can be either treating I and Q signals as two real-valued input channels or concatenating I and Q signals to a single real-valued vector. The real-valued input conversion from the complex-valued input is shown in Fig. 4.6. Even though Fig. 4.6 illustrates the conversion of the time-domain received RF signals, those methods can be expanded to other complex-valued data as well. Moreover, complex-valued neural networks (CVNN) can reduce the degree of freedom of the neuron layers while preserving the phase information of the complex number since it is derived from the complex value multiplication. This shows the potential of CVNNs to achieve as good performance as real-valued neural networks (RVNN) with less number of parameters [43]. The outperforming of CVNN over RVNN in magnetic resonance (MRI) fingerprinting [123] and symmetric data detection [4] further highlights the potential of CNVV. For those reasons, CVNN models get more and more attention in wireless communication applications.

4.2.3 Optimization Techniques for Model Learning

The training and learning of the neural networks are based on back-propagation algorithms and appropriate optimization techniques. The purpose of optimization algorithms in neural

networks is to find the optimal parameters that result in the highest performance, while avoiding overfitting or being stuck in a local minimum.

Stochastic gradient descent (SGD) is one of the popular optimizers for training CNN models. It has the advantages of shorter converging time and less memory usage, but the disadvantages on the high learning variance of model parameters and the unstable learning process [102]. To reduce the variance during the optimization, momentum is taken into consideration, which allows the algorithm to move faster in the optimizing direction and smooth out oscillations in the parameter updates.

AdaDelta is one of the improved optimization techniques based on SGD [139]. It can dynamically update the learning rate to avoid large variations in the parameter update at each step caused by the gradient noise [139]. AdaDelta has been used in the optimization procedure for MNIST classification training [26]. The detailed algorithm is shown in Alg. 3.

Algorithm 3 AdaDelta optimizer

Require: w_0 (parameters), ρ (decay), γ (step size), ϵ (constant)
Ensure: w_T
Initialize: $a_0 \leftarrow 0$, $b_0 \leftarrow 0$
for $t = 1 : T$ **do**
Get the gradient: g_t
$a_t \leftarrow \rho a_{t-1} + (1-\rho) g_t^2$

$v_t \leftarrow g_t \frac{\sqrt{b_{t-1} + \epsilon}}{\sqrt{a_t + \epsilon}}$

$b_t \leftarrow \rho b_{t-1} + (1-\rho) v_t^2$
$w_t \leftarrow w_{t-1} - \gamma v_t$
end for
return w_T

Instead of just considering the first-order momentum during the weight update process, adaptive moment estimation (Adam) also considers the second-order momentum of the gradient [55] to avoid the fast change of velocity and missing the optimal solution caused by overshooting. By incorporating the second-order information, Adam can obtain a better solution than Adadelta. The algorithm of Adam is summarized in Alg. 4.

4.3 Problem Setup

In this work, the same single-cell massive MIMO uplink system, which has been described in Chapter 3, is considered. This system contains K single antenna UEs and a M antenna BS. During the uplink communication, user Golay pilots will be sent by each user in the

Algorithm 4 Adam optimizer

Require: w_0 (parameters), β_1, β_2 (betas), γ (step size), ϵ (constant)
Ensure: w_T
  Initialize: first moment $v_0 \leftarrow 0$, second moment $a_0 \leftarrow 0$
  for $t = 1 : T$ **do**
    Get the gradient: g_t
    $v_t \leftarrow \beta_1 v_{t-1} + (1 - \beta_1) g_t$
    $a_t \leftarrow \beta_2 a_{t-1} + (1 - \beta_2) g_t^2$
    $\hat{v}_t \leftarrow v_t / (1 - \beta_1^t)$
    $\hat{a}_t \leftarrow a_t / (1 - \beta_2^t)$
    $w_t \leftarrow w_{t-1} - \gamma \hat{v}_t / (\sqrt{\hat{a}_t} + \epsilon)$
  end for
  return w_T

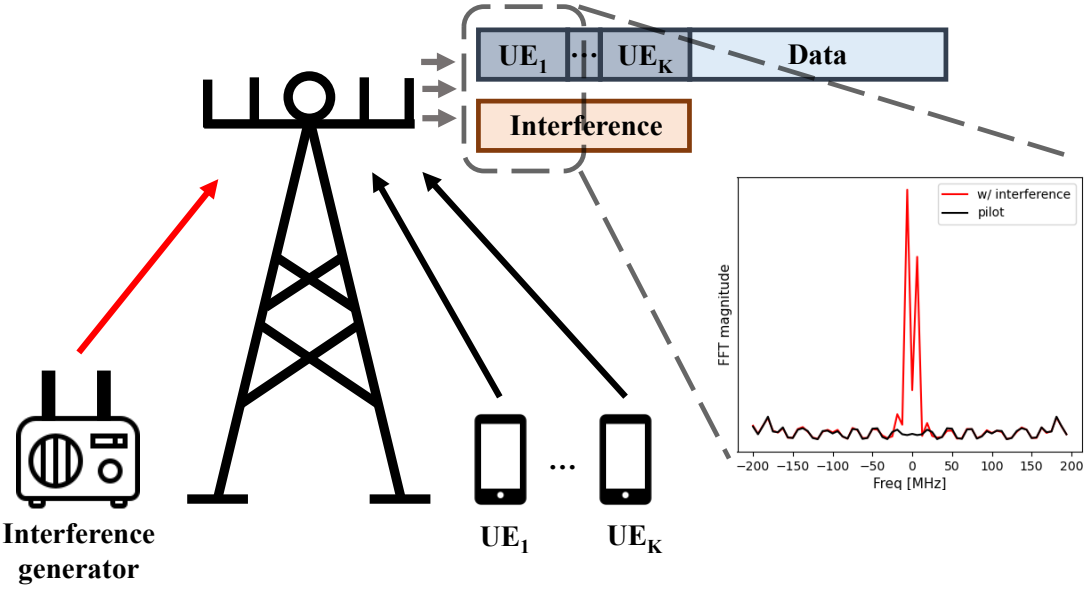


Figure 4.7: The massive MIMO system contains K UEs and an interference generator. The 20MHz in-band interference will constantly occur during the TDD Golay pilot transmission, which is shown in the zoomed-in inset.

time-interleaved manner, which are used for the distributed MRC beamforming channel estimation. Then at the BS receiver, the Golay-correlator-based channel estimator is used to extract CSIs according to Eq. 3.9. Moreover, perfect frontends and perfect synchronization are assumed.

However, during the transmission of user Golay pilots, a constant in-band 20MHz additive Gaussian interference occurs, and it comes from the interference generator as illustrated in

Fig. 4.7. The signal power of the interference $\boldsymbol{x}_{inf}$ is the same as the signal power of the user Golay pilots $\boldsymbol{x}_{gp}$, which also means $\mathbb{E}[\boldsymbol{x}_{gp}\boldsymbol{x}_{gp}^H] = \mathbb{E}[\boldsymbol{x}_{inf}\boldsymbol{x}_{inf}^H]$. In this work, the AWGN with a relatively high beamforming SNR and LOS channel model is considered. In addition, $\mathbb{E}[\boldsymbol{h}_i\boldsymbol{h}_i^H] = \mathbb{E}[\boldsymbol{h}_j\boldsymbol{h}_j^H]$ for all $i, j \in \{1, ..., K\}$ is assumed for the UE k channel vector $\boldsymbol{h}_k \in \mathbb{C}^M$.

Unfortunately, the noisy channel environment can easily degrade the performance of the correlator-based channel estimator, leading to a lower SNR gain in the MRC stage and an inaccurate downlink channel prediction. Fig. 4.8 shows the channel estimation MSE using the correlator-based channel estimator with and without the existence of the interference. The result shows that the MSE can increase 3 times when the center frequency of the wideband interference is swept from $-150MHz$ to $150MHz$. Therefore, it is important to find a better solution to improve the channel estimation performance under this challenging scenario.

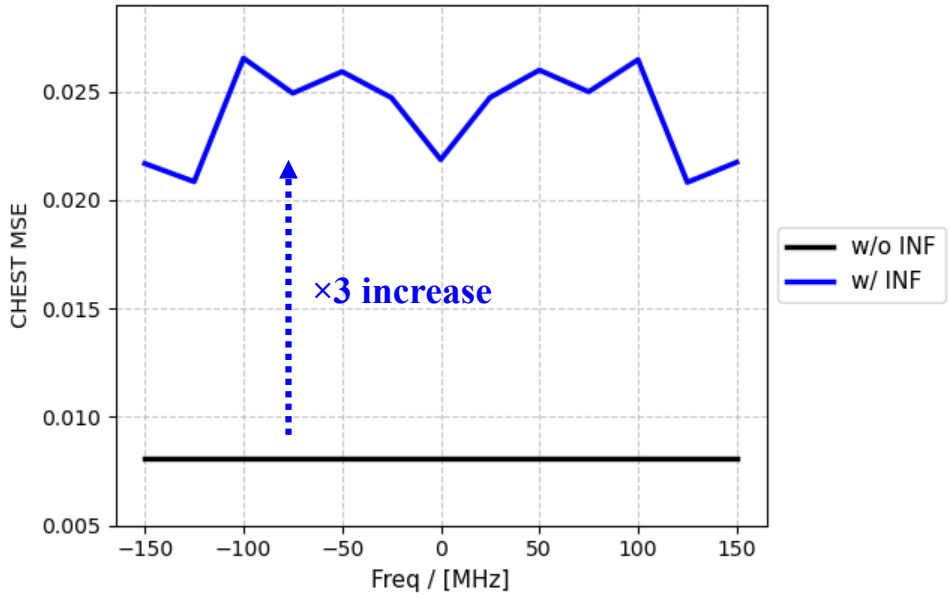


Figure 4.8: Channel estimation (CHEST) MSE using the correlator-based channel estimator vs the center frequency of the in-band wideband interference with Golay sequence length of 64. The blue line shows the CHEST MSE by the pilot with interference (INF), and the black line shows the CHEST MSE by the pilot only.

4.4 The Architecture of CVNN-Aided Channel Estimation

The CVNN-aided channel estimation method is proposed to improve the channel estimation performance. The entire flow includes pre-processing, CVNN prediction, and post-processing

for channel estimation. It takes the per-UE, per-BS-antenna, time domain Golay pilot $\boldsymbol{x}_{gp}$ as the input, and pre-processes it by using shifted FFT. The frequency domain Golay pilots are then fed into the CVNN. After the prediction, the predicted signals are converted back to the time domain by IFFT post-processing. Then the correlation-based channel estimator, as described in Eq. 3.9, is employed to estimate CSIs. The data flow of the CVNN-aided channel estimation method is shown in Fig. 4.10.

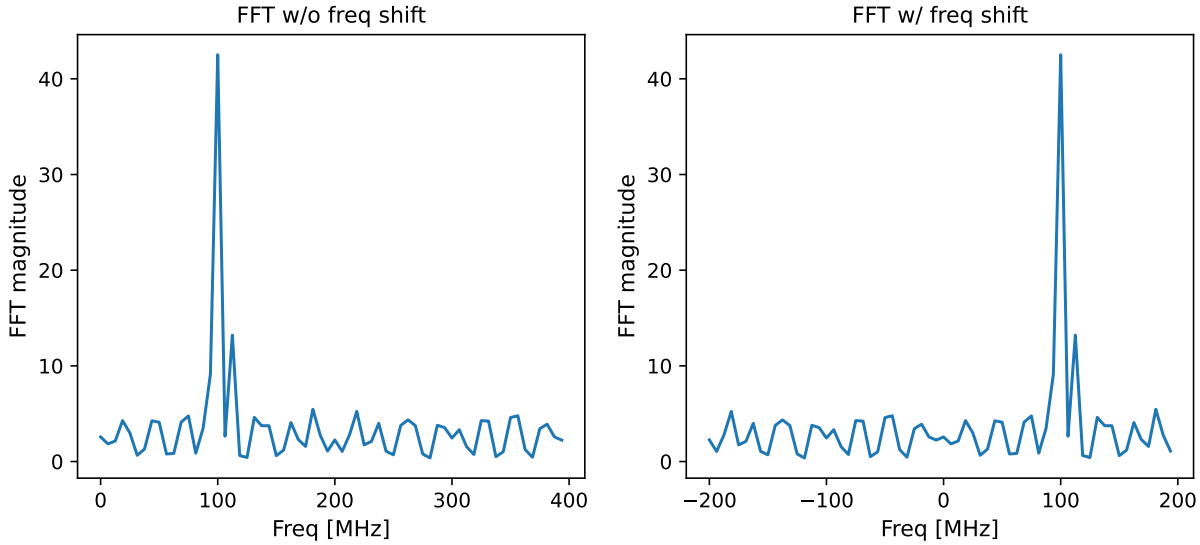


Figure 4.9: Input difference. Direct FFT vs. FFT shifted

4.4.1 Pre-processing

At BS receivers, the incoming baseband user Golay pilots are pre-processed with the shifted FFT to convert the signal vectors from the time domain to the frequency domain. The frequency domain signals show more distinguishable features due to the bandwidth difference between the user pilot and interference. One of the problems of using FFT directly is the feature edge-crossing problem, in which the interference frequency domain features are separated to the two ends of the input vectors when the interference center frequency is close to 0Hz. By using shifted FFT, the index of signal vectors can range from $[-f/2, f/2)$ MHz, instead of $[0, f)$ MHz, as shown in Fig. 4.9. As a result, the shifted FFT method can avoid the feature edge-crossing problem and help with the feature correlation and extraction for the convolution layers.

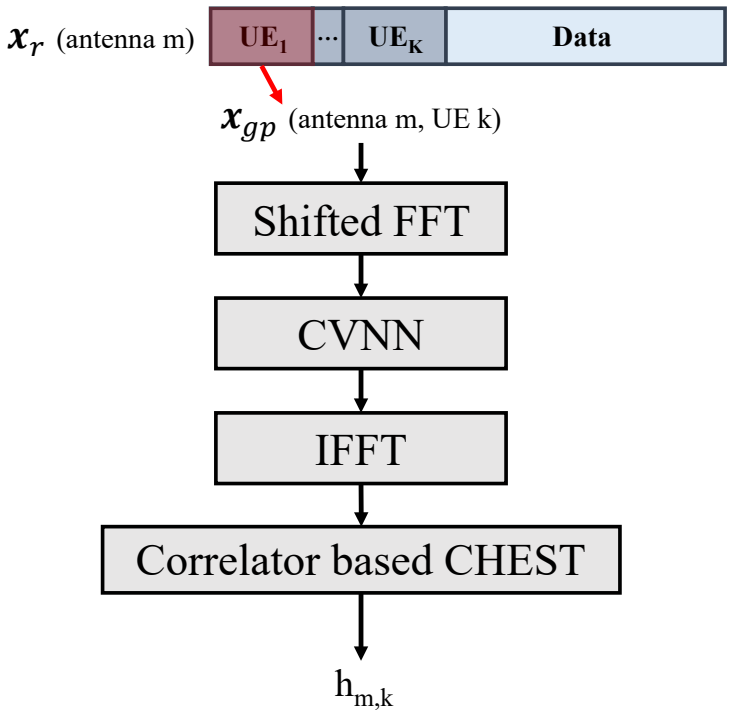


Figure 4.10: The data flow of the CVNN-aided channel estimation method.

4.4.2 The Architecture of CVNN

As mentioned in Sec. 4.2.1, CNN models have great potential for channel estimation due to their powerful feature extraction ability and good prediction performance. Although in this work, both the wideband interference and AWGN exist in the channel, the pilot pattern should still be embedded in the received signals and could be predicted by using ML methods. As a result, the CNN model is chosen in this work. To achieve good prediction performance with fewer model parameters by taking advantage of the phase information preserved in the learning procedure, the CVNN model is employed in this work.

Fig. 4.11 shows the architecture of the proposed CVNN model. It contains two complex-valued convolution layers, two complex-valued max-pooling layers, two complex-valued fully connected layers, and two dropout layers activated only during training to avoid overfitting. The dimensions, kernel sizes, or the parameter numbers of each layer are shown in Table. 4.1, where L is the length of the user Golay pilot.

Complex-valued Convolution Layer

Both 1-dimensional convolution layers in the proposed CVNN model are implemented in the complex domain. The complex-valued convolution layer consists of a real convolution layer and an imaginary convolution layer with C_{in} number of input channels and C_{out} number of

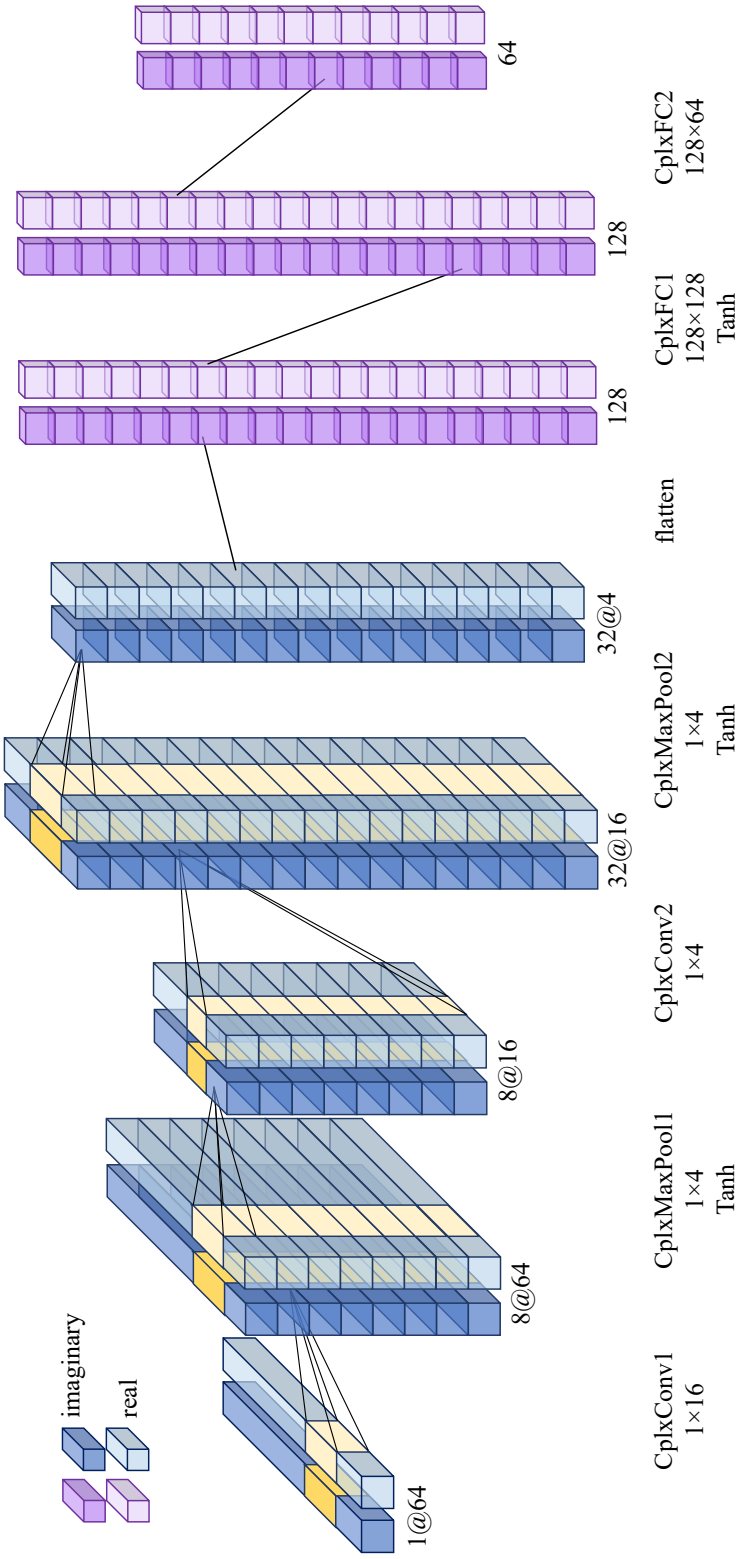


Figure 4.11: The network architecture of the complex-value neural network.

Table 4.1: The dimension or parameters of each CVNN layer.

Layer name	**($\mathbf{Dim}_{in}$, $\mathbf{Dim}_{out}$) /Param**		**Kernel**	**Layer Activation**
Conv1	(1, L)	(8, L)	16	Train & Inference
MaxPool1	(8, L)	(8, L)	4	Train & Inference
Conv2	(8, L/4)	(32, L/4)	4	Train & Inference
MaxPool2	(32, L/4)	(32, L/4)	4	Train & Inference
DropOut1	0.1			Train
FC1	32×L/16	32×L/16		Train & Inference
DropOut2	0.1			Train
FC2	32×L/16	L		Train & Inference

output channels. Let L_{data} be the length of the input and the output, L_k be the kernel size of both layers, $\boldsymbol{x} \in \mathbb{C}^{C_{in} \times L_{data}}$ be the convolution layer input, $\boldsymbol{y} \in \mathbb{C}^{C_{out} \times L_{data}}$ be the convolution layer output, and let $\boldsymbol{w}_i \in \mathbf{C}^{C_{in} \times L_k}$ for $i \in \{1, ..., C_{out}\}$. The input and the kernel of the real convolution layer are $Re\{\boldsymbol{x}\}$ and $Re\{\boldsymbol{w}\}$, and the input and the kernel of the imaginary convolution layer are $Im\{\boldsymbol{x}\}$ and $Im\{\boldsymbol{w}\}$. The output of the complex-valued convolution layer is

$$\begin{aligned} Re\{\boldsymbol{y}\}[i] &= Re\{\boldsymbol{x}\} * Re\{\boldsymbol{w}_i\} - Im\{\boldsymbol{x}\} * Im\{\boldsymbol{w}_i\} \\ Im\{\boldsymbol{y}\}[i] &= Re\{\boldsymbol{x}\} * Im\{\boldsymbol{w}_i\} + Im\{\boldsymbol{x}\} * Re\{\boldsymbol{w}_i\}. \end{aligned} \tag{4.5}$$

Let us consider an alternative convolution layer implementation using real numbers, with $2 \times C_{in}$ number of input channels and $2 \times C_{out}$ number of output channels. The corresponding real-valued kernel can be expressed as $\boldsymbol{w}'_i \in \mathbf{R}^{2C_{in} \times L_k}$ for $i \in \{1, ..., 2C_{out}\}$. $Re\{\boldsymbol{x}\}$ and $Im\{\boldsymbol{x}\}$ are treated as two input channels of this real-valued convolution layer, stacked in an interleaved manner as shown in Fig. 4.6 (1). Compared with this real-valued convolution layer, the number of weights of the complex-valued layer with the same dimension can reduces by half, and the output phase information or the correlation between the real and imaginary parts embedded in the complex-valued multiplication can be preserved without distortion.

Consider another real-valued convolution layer implementation with C_{in} number of input channels and C_{out} number of output channels, and the kernel can be expressed as $\boldsymbol{w}''_i \in \mathbf{R}^{C_{in} \times 2L_k}$ for $i \in \{1, ..., C_{out}\}$. The number of weights for this real-value convolution layer can be the same as the proposed complex-valued convolution layer. But the phase information can be distorted during the feedforward learning procedure.

Besides the choices between using complex-valued or real-valued networks, another factor that can affect the performance of the complex-valued convolution layer is the padding method. The most commonly used padding method is the zero padding mode, where the

padding area are assigned with $0 + 0j$ as shown in Fig. 4.12. However, due to the circularity property of FFT, in this work, circular padding is used to improve the feature extraction of the convolution layer when the interference center frequency is close to the bandwidth limit of the pilot signal.

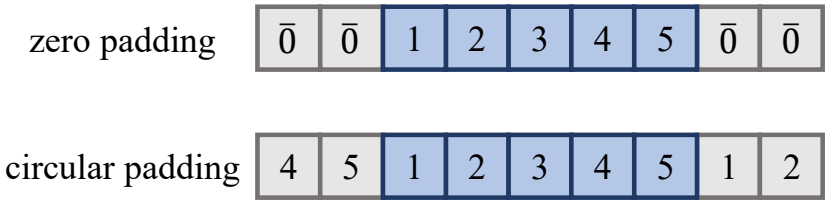


Figure 4.12: Zero padding vs circular padding. The blue squares represent the input data, and the grey squares represent the padding area.

Complex-valued max-pooling Layer

The purpose of the max-pooling layer in CNN models is the important feature extraction and data downsampling. Images are the targeted input data type for most of the state-of-the-art CNN models, which are often represented by 8-16 bit non-negative integers. The max-pooling layer can simply choose the largest number within a certain window size, and then forward the selected features to the next layer.

However, for CVNN models, targeted inputs $\boldsymbol{x}$ are complex-valued, while the rule to compare two complex numbers is vague. One way to implement the complex-valued max-pooling layer is by choosing the largest value from the input real part and input imaginary part separately, and then using the two largest value to form the "maximum" complex-valued output, which can be expressed in Eq. 4.6.

$$Max(\boldsymbol{x}) = Max(Re\{\boldsymbol{x}\}) + j \cdot Max(Im\{\boldsymbol{x}\}). \tag{4.6}$$

Unfortunately, by this method, the maximum real and imaginary value may not come from the same pair, and the data can be distorted after the max pooling, leading to the loss of the correlation hidden in the signal. To keep the correlation between the real part and the imaginary part of the data, a maximum power pooling method is implemented in this work. It means that the complex number with the maximum power within the window will be selected and passed to the next layer. Let the window size be L_{win}, then the maximum power pooling can be expressed as Eq. 4.7.

$$\begin{aligned} i_m &= \underset{i \in \{1,\dots,L_{win}\}}{\operatorname{argmax}} \; Re\{x(i)\}^2 + Im\{x(i)\}^2 \\ Max(\boldsymbol{x}) &= x(i_m). \end{aligned} \tag{4.7}$$

By using Eq. 4.7, the phase information can be preserved, and the important features can also be extracted.

As shown in Fig. 4.11, the nonlinear activation function is performed after complex-valued max-pooling layers, and the activation function used in this work is the extended version of the hyperbolic tangent function to the complex domain, which can be expressed as

$$f(x) = \frac{e^{Re\{x\}} - e^{-Re\{x\}}}{e^{Re\{x\}} + e^{-Re\{x\}}} + j \cdot \frac{e^{Im\{x\}} - e^{-Im\{x\}}}{e^{Im\{x\}} + e^{-Im\{x\}}} \tag{4.8}$$

The hyperbolic tangent function is widely used in regression models, and in this work, it is applied to the real and imaginary parts of the data separately, as shown in Eq. 4.8. The reasons for putting the activation function after the pooling layer are the computation reduction and the more accurate feature extraction. The hyperbolic tangent function involves exponential operations which is computation-hungry. The max-pooling layer can downsample the data, which reduces the number of exponential operations required in the activation layer. In addition, applying the max-pooling layer directly after the convolution layer can guarantee the non-distorted power in the input data, so the feature extraction of the complex-valued max-pooling layer can be more accurate. For those reasons, the complex-valued hyperbolic tangent activation function is applied after the complex-valued max-pooling layer.

Complex-valued Fully-connected Layer

The multi-channel outputs of the complex-valued max-pooling are flattened into the real vector and the imaginary vector after being applied with the activation function. The flattened vectors are then fed into the complex-valued fully-connected layer. The complex-valued fully-connected layer is also composed of a real fully-connected layer and an imaginary fully-connected layer. Let L_{in} and L_{out} be the input vector length and output vector length, $\boldsymbol{x}_{FC} \in \mathbb{C}^{L_{in}}$ be the fully-connected layer input, $\boldsymbol{y}_{FC} \in \mathbb{C}^{L_{out}}$ be the fully-connected layer output, and $\boldsymbol{w}_{FC} \in \mathbb{C}^{L_{out} \times L_{in}}$ be the weights of this layer. The input and weights of the real fully-connected layer are $Re\{\boldsymbol{x}_{FC}\}$ and $Re\{\boldsymbol{w}_{FC}\}$, and the input and weights of the imaginary fully-connected layer are $Im\{\boldsymbol{x}_{FC}\}$ and $Im\{\boldsymbol{w}_{FC}\}$. The output of the complex-valued fully-connected layer can be expressed as

$$\begin{aligned} Re\{\boldsymbol{y}_{FC}\} &= Re\{\boldsymbol{w}_{FC}\}Re\{\boldsymbol{x}_{FC}\} - Im\{\boldsymbol{w}_{FC}\}Im\{\boldsymbol{x}_{FC}\} \\ Im\{\boldsymbol{y}_{FC}\} &= Im\{\boldsymbol{w}_{FC}\}Re\{\boldsymbol{x}_{FC}\} + Re\{\boldsymbol{w}_{FC}\}Im\{\boldsymbol{x}_{FC}\}. \end{aligned} \tag{4.9}$$

4.4.3 Post-processing

After CVNN prediction, IFFT converts the predicted signal from the frequency domain back to the time domain. Then the correlator-based channel estimator is employed to estimate CSIs. The channel estimation MSE can be reduced while the pilot length increases, which will be discussed in detail in the result section.

4.5 Training and Inference Setup for CVNN

4.5.1 Dataset

In this work, the training and inference dataset are synthetic and generated using the Hydra Python simulator described in Chapter 3. The channel noise includes both AWGN and wideband interference. In the massive MIMO system, the BS has 64 antennas and serves 4 UEs. The channel model assumed is the LOS channel model, and user channel losses are the same. The total length of the Golay pilot is 64, and the length of each sequence in a Golay pair is 32. Because the high-power wideband interference is the main focus of this work, a relatively high SNR, 15dB, for AWGN is considered.

AWGN and interference with various center frequencies. The target outputs of the CVNN model are BS-received Golay pilots without noise. The Python simulator generates BS received signals and reshapes the received Golay pilots to an array with the size ($M \times K$, L_{gp}). The reshaped array vectors are fed into the CVNN model because the CVNN takes per-channel per-user Golay pilots as inputs. The total number of training data is 33280, and the total number of inference data is 88320. The center frequencies of the 20MHz interference for training and inference dataset are shown in the Table. 4.2.

Although the training dataset for the proposed CVNN is synthetic, the real-world measured training data can also be applied to train the model. The strategy for the measured training data collection will be discussed in the Discussion section.

Table 4.2: The dataset generation parameters.

Parameter	**Value**
BS antenna and UE	64 antennas and 4 UEs
Signal bandwidth	400MHz
Channel model	LOS
Golay pilot length	64
SNR (AWGN)	15dB
Interference bandwidth	20MHz
P_{sig}/P_{inf}	1
Training center freqs [MHz]	±150, ±125, ±100, ±75, ±50, ±25, 0
Training size	8519680 (33280 × 64 × 4)
Inference center freqs [MHz]	±150, ±137.5, ±125, ±100, ±87.5, ±75, ±62.5, ±50, ±37.5, ±25, ±12.5, 0
Inference size	22609920 (88320 × 64 × 4)

4.5.2 Training

During the training phase, the training data are shuffled first to ensure the order of the input data will not contribute to the model training. To avoid overfitting, the CVNN model also applies dropout layers with a 10% dropout rate after MaxPool2 and FC1 layers. The dropout layers will be skipped during inference.

The loss function used for training is the square error loss between the prediction and the target output. Suppose the vector length of the prediction and the target output is L_{gp}, the predicted output is $\boldsymbol{y}_p \in \mathbb{C}^{L_{gp}}$, and the target output is $\boldsymbol{y}_p \in \mathbb{C}^{L_{gp}}$. The square error loss function can be expressed as Eq. 4.10. The ultimate goal of the CVNN model is to predict the output with as minimum square error as possible with the target output. Thirteen different interference center frequencies are involved in training the CVNN model, and each center frequency contributes 655360 training data points.

$$\mathcal{L}(\boldsymbol{y}_p) = (\boldsymbol{y}_p - \boldsymbol{y}_t)^H (\boldsymbol{y}_p - \boldsymbol{y}_t). \tag{4.10}$$

In the model training, two different optimization techniques are compared to get a better set of trained parameters. The two optimizers used are Adadelta and Adam, which have been discussed in Sec. 4.2, and optimizer parameters for both methods are summarized in Table. 4.5.2.

Table 4.3: The parameters for optimizers.

	γ	$\rho/(\beta_1, \beta_2)$	ϵ
AdaDelta	1.0	0.9	10^{-6}
Adam	0.005	(0.9, 0.999)	10^{-8}

Fig. 4.13 shows the training loss vs. training epoch using the two optimization techniques. The curve shows that AdaDelta has a faster convergence speed but a less optimal solution. Due to the introduction of the second-order momentum, Adam has a slower convergence speed but can reach a more optimal point. As a result, Adam optimizer is used to train the proposed CVNN model.

4.5.3 Inference

During the inference phase, the dropout layers are inactive. 21 different interference center frequencies are involved in testing the prediction performance and the generalization ability of the CVNN model. The inference dataset is also generated by the Python simulator but is not included during the training phase.

Besides the center frequencies that have been included for training the model, 8 more interference center frequencies are involved in the inference to test the generalization of the trained CVNN model. The predicted inference outputs are transformed back to the time

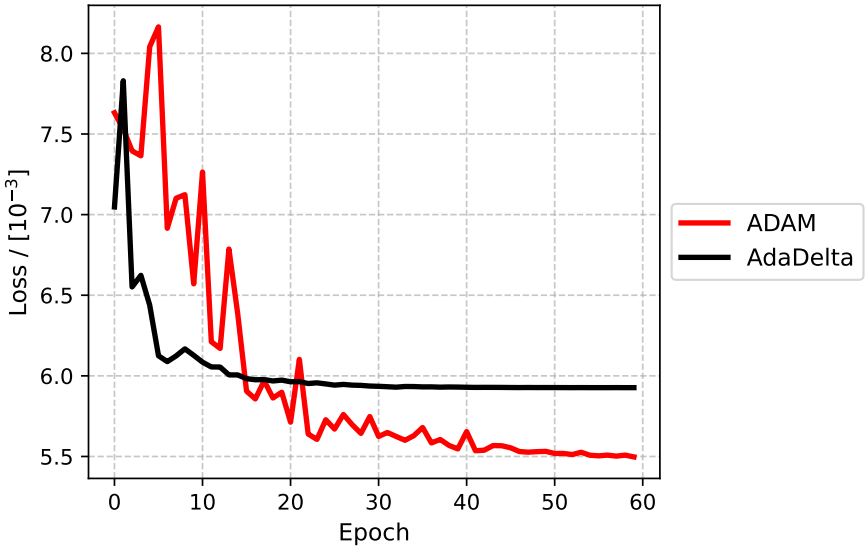


Figure 4.13: Training loss vs. training epoch. The red line represents the training loss curve using ADAM, and the black line represents the training loss curve using AdaDelta.

domain and post-processed in the Python simulator to estimate CSIs. The CVNN-aided channel estimation performance is evaluated based on the channel estimation MSE.

4.6 Peformance Analysis

4.6.1 CVNN design choices comparison

Different design choices that can affect the channel estimation performance of the proposed method are evaluated and compared in this section. The design choices are whether use the direct FFT or shifted FFT to convert pilots from the time domain to the frequency domain and use the zero or the circular padding mode for the complex-valued convolution layers.

In this evaluation, the proposed CVNN model is trained with the received Golay pilots containing interference whose baseband center frequencies locate at 100MHz and -100MHz. The total training size is 8519680, and the channel estimation performance of the proposed method is evaluated with the interference center frequency ranging from -150MHz to 150MHz to evaluate the generalization of the model and the symmetry of the curve by using different designs. In this evaluation, design choices that can contribute to a more symmetric curve of the MSE vs. frequency are preferable because the symmetric curve indicates that training the CVNN model implementing those design choices at a set of center frequencies that are symmetric to 0Hz is reasonable.

Let $H_{est,i} \in \mathbb{C}^{M \times K}$ be the estimated channel matrix, $H_i \in \mathbb{C}^{M \times K}$ be the ground truth channel matrix, and the total number of tests evaluated is N_t. The channel estimation MSE

is measured as

$$MSE = \frac{\sum_i^{N_t} ||H_{est,i} - H_i||^2}{N_t \times M \times K}. \tag{4.11}$$

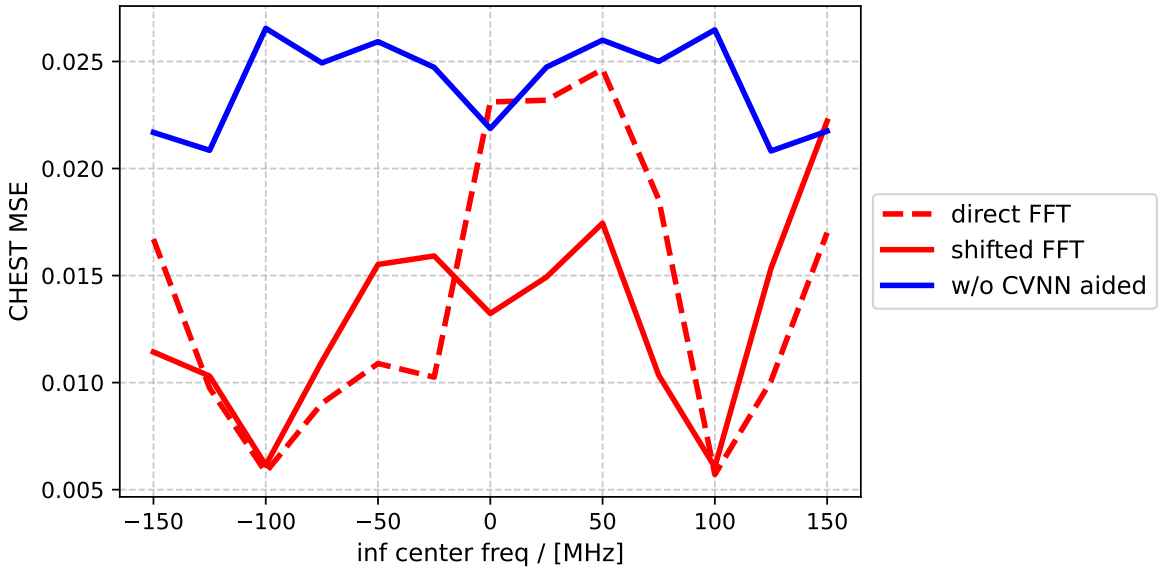


Figure 4.14: Channel estimation (CHEST) MSE vs 20MHz interference center frequency. The blue line shows the MSE of the correlator-based channel estimator, and the red line shows the MSE of the channel estimator with CVNN. The dotted red line represents the MSE by using direct FFT input, while the solid red line represents the MSE by using shifted FFT input.

Fig. 4.14 shows the channel estimation MSE vs. the interference center frequency by using the direct FFT and shifted FFT transformed Golay pilots as the inputs of the CVNN model. In this part, the complex-valued convolution layers use the zero padding mode. As shown in the figure, the channel estimation performance using both methods is almost the same when interference center frequencies are at the trained frequencies, 100MHz and -100MHz. However, for frequencies that are not included in the training phase, the direct FFT method has a higher channel estimation MSE, especially when the frequency ranges from 0Hz to 50MHz. It is caused by the feature edge-crossing issue when using the direct FFT. When the interference center frequency closes to 0Hz, the interference frequency-domain features can locate at the two ends of the input vectors, which distorts the correlation of the input and degrades the feature extraction of the complex-valued convolution layer. For this reason, it is better to let the interference information contiguously locate inside the input vector since the edge-crossing issue can negatively affect the performance of the CVNN-aided channel estimation.

For the in-band interference, the edge-crossing issue can be avoided using the shifted FFT method for the signal domain transformation. Unlike the direct FFT method, whose

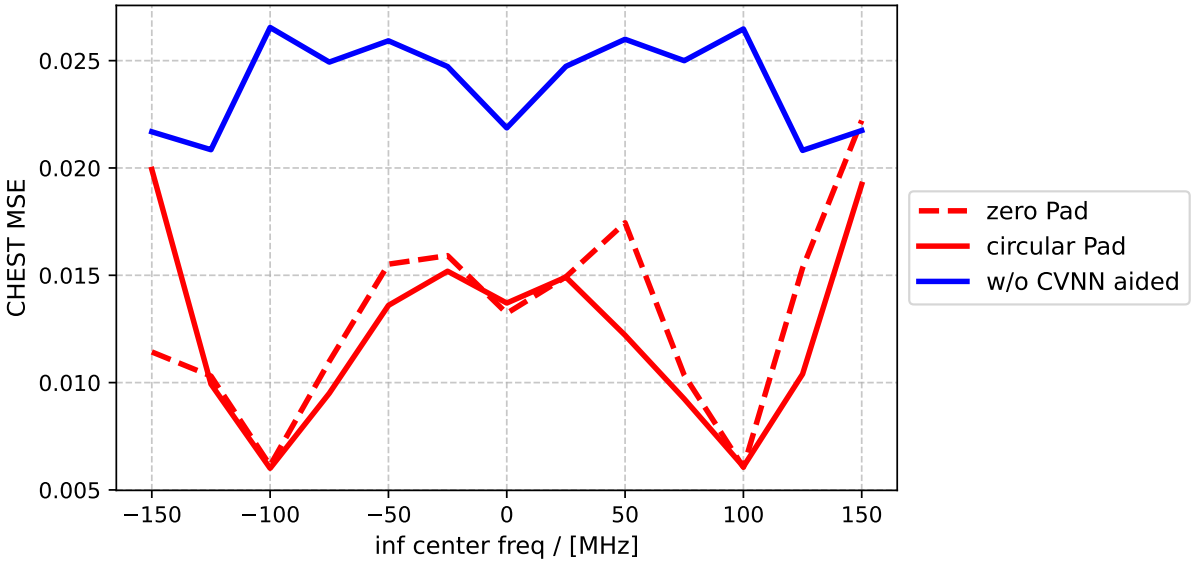


Figure 4.15: Channel estimation (CHEST) MSE vs 20MHz interference center frequency. The blue line shows the MSE of the correlator-based channel estimator, and the red line shows the MSE of the channel estimator with CVNN. The dotted red line represents the MSE by using zero padding, while the solid red line represents the MSE by using circular padding.

frequency index ranges from [0Hz, 400MHz), the frequency index of the shifted FFT method is [-200MHz, 200MHz). Then, interference features can locate inside the input vector without being separated by the boundary and even keep some distance from the edge for most scenarios. Therefore, the shifted FFT method removes the feature edge-crossing problem. Fig. 4.9 shows that using the shifted FFT method, the channel estimation performance can be improved for the cases where the center frequency of the interference is close to 0Hz, and the curve is more symmetric compared with the other method. As a result, the shifted FFT method should be employed for the input data pre-processing of the proposed method.

The padding mode of the complex-valued convolution layer can also affect the performance of CVNN-aided channel estimation. Fig. 4.15 shows the channel estimation MSE vs. interference center frequency using the zero and circular padding mode in the CVNN model. When using the zero padding method, some frequency domain information will be lost when convolving the first and last several data of the input vector. Due to the circularity property of FFT, more frequency domain information can be preserved and involved in the convolution by using the circular padding method, especially when convolving the data at both ends of the input vector. For this reason, the channel estimation MSE curve with respect to the interference center frequency is more symmetric by using the circular padding mode than the zero padding mode. Even though the channel estimation MSE increases when interference center frequencies are close to the boundary of the signal bandwidth, the performance can be improved by training with more data points at different center frequencies.

The set of training center frequencies is listed in Table. 4.1. As a result, the proposed CVNN model implements the shifted FFT method and the circular padding mode.

4.6.2 Channel estimation performance

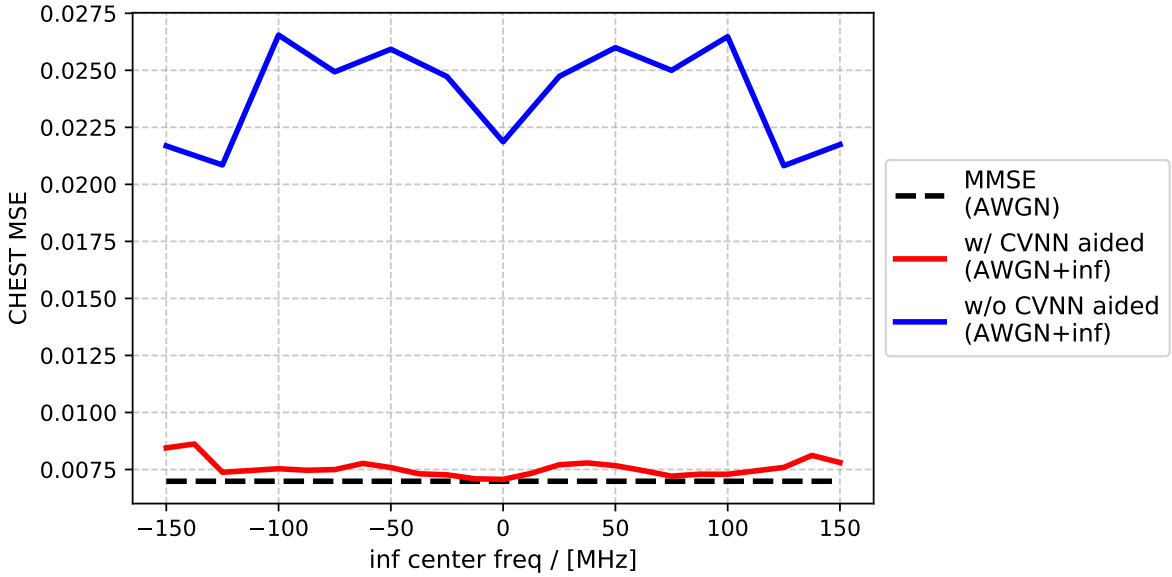


Figure 4.16: Channel estimation (CHEST) MSE vs 20MHz interference center frequency. The black dotted line shows the MSE of the MMSE-based channel estimator with no interference existing in the system, the blue line shows the MSE of the correlator-based channel estimator, and the red line shows the MSE of the channel estimator with CVNN.

The CVNN model was trained using the BS received interfered Golay pilots with the set of interference center frequencies listed in *Training center freqs* of Table 4.2. 13 different interference center frequencies were included in the training phase. The performance of the CVNN-aided channel estimator was evaluated using the BS received interfered Golay pilots with the set of interference center frequencies listed in *Inference center freqs* of Table 4.2. Fig. 4.16 illustrates the channel estimation MSE vs. interference center frequencies. Compared to the solid red line in Fig. 4.15, where the model was trained only with pilots having interference center frequencies at 100MHz and -100MHz, the model trained with those 13 different interference center frequencies can achieve a good channel estimation performance for all interference center frequencies ranging from -150MHz to 150MHz. Even for center frequencies that were not included in the training, the channel estimation MSE of those untrained points was close to the trained points. The flattening of the channel estimation MSE curve achieved by the CVNN-aided channel estimator demonstrates the good generalization ability of the proposed method.

With the help of the CVNN, the channel estimation MSE can be reduced 3.16× compared to using the correlator-based channel estimator alone. The channel estimation performance of the proposed method is also compared with the performance of the MMSE-based channel estimator estimating the channel without interference. With the aid of CVNN, the channel estimation MSE can be reduced close to the estimation by the MMSE-based estimator using non-interfered pilots. As a result, the CVNN-aided channel estimator can remove the interference effect in the channel estimation.

4.6.3 Complex-valued and real-valued model comparison

This section compares the channel estimation performance using complex-valued or real-valued neural networks. Two real-valued CNN models are considered. Both real-valued CNN models consist of two convolution layers with the circular padding mode, two max-pooling layers, and two fully-connected layers, the same as the proposed CVNN model.

CNN1 is the real-valued CNN model sharing the same structure as the proposed CVNN model. For example, the kernel sizes of the convolution layers are the same as CVNN models', and the sizes of the fully connected layers are doubled accordingly due to the separation of the real part and the imaginary part of the complex-valued vector. As discussed in Sec. 4.4.2, the total number of parameters of the fully-connected layer will be doubled compared with the complex-valued fully connected layer if the real-valued fully connected layer is used for complex values.

CNN2 is the real-valued CNN model sharing the same number of weights and the similar total number of parameters as the proposed CVNN model. The number of weights of the convolution layers and the fully-connected layers between CVNN and CNN2 are the same by adjusting the kernel size and the layer dimensions. The slight difference in the total number of parameters between CVNN and CNN2 models is due to the different number of biases.

The input of *CNN1* and *CNN2* are the concatenated array of the real part and the imaginary part of the frequency-domain received Golay pilots, and the Golay pilot length is 64. Table. 4.4 summaries the neural network parameters of the proposed CVNN, CNN1, and CNN2. All three models are trained and evaluated using the same training and inference dataset summarized in Table. 4.2.

Table. 4.5 summaries the inference prediction MSE of CVNN, CNN1, and CNN2 at the trained interference center frequencies. The result shows that for the interference center frequencies at the boundary of the signal bandwidth, the RVNNs have better performance than the CVNN because of the concatenation of the real and imaginary parts of the input vector. Due to the concatenation, although the frequency-domain interference features are located at the boundary of the bandwidth, either the real part or the imaginary part of the feature will locate near the center of the input array. As we have discussed in Section 4.6.1, the CNN model can give a good prediction if the interference features are not close to the boundary of the input, so real-valued CNN models can obtain better performance than the CVNN model when the interference center frequency is close to the boundary of the signal bandwidth. However, the overall neural network prediction performance of the CVNN is

Table 4.4: The model parameters of CVNN, CNN1, and CNN2. The input of CNN1 and CNN2 are the concatenated array of the real part and imaginary part of the received Golay pilot with 64 Golay pilot length.

Layer name	**(Channel_in, Channel_out, kernel)**		
	CVNN	CNN1	CNN2
Conv1	(1, 8, 16)	(1, 8, 16)	(1, 8, 32)
MaxPool1	(8, 8, 4)	(8, 8 ,4)	(8, 8, 4)
Conv2	(8, 32, 4)	(8, 32, 4)	(8, 32, 8)
MaxPool2	(32, 32, 4)	(32, 32, 4)	(32, 32, 4)
FC1	(128, 128,)	(256, 256,)	(256, 128,)
FC2	(128, 64,)	(256, 128,)	(128, 128,)
Total params	51920	99880	51752

still slightly better than both CNN1 and CNN2 for the set of interference center frequencies included in the training phase.

Table. 4.6 summaries the inference prediction MSE of CVNN, CNN1, and CNN2 for the untrained interference center frequencies. The result shows that for the untrained interference frequencies, CNN1 and CNN2 model predictions get worse. For CNN1, the prediction MSE increases by 3.75×10^{-4}, and for CNN2, the prediction MSE increases by 4.23×10^{-4}. However, the prediction MSE of CVNN only increases by 0.79×10^{-4}. The smallest prediction MSE variation of CVNN shows that it has a better generalization ability than the real-valued neural networks that have similar network structures or similar numbers of parameters. It can also indicate that the phase correlation embedded in the complex numbers can help improve the neural network prediction performance.

In summary, CVNN outperforms the real-valued neural networks either with similar layer structures or with a similar number of parameters, especially in the model generalization ability.

4.6.4 Pilot length comparison

The effect of interference can be reduced by increasing the pilot length in the correlator-based channel estimator because higher SNR can be achieved after correlation. This can be shown as follows. Suppose the interference $\boldsymbol{n}_{inf}$ in the received pilot $h\boldsymbol{x}_{gp} + \boldsymbol{n}_{inf}$ has the following properties, that $\mathbb{E}[\boldsymbol{n}_{inf}] = 0$, and $\mathbb{E}[\boldsymbol{n}_{inf}\boldsymbol{n}_{inf}^H] = \sigma_{inf}^2 \mathbf{I}_{L_{gp}}$, then the expectation of the channel

Table 4.5: The inference prediction MSE comparison between CVNN and CNN models for the trained interference center frequencies.

inf center frequency	**prediction MSE** $[\times 10^{-3}]$		
[MHz]	CVNN	CNN1	CNN2
-150	8.722	7.872	7.664
-125	7.436	7.873	7.651
-100	7.557	7.885	7.825
-75	7.511	7.567	7.726
-50	7.646	7.690	7.544
-25	7.215	7.729	7.537
0	7.037	7.631	7.447
25	7.706	7.689	7.637
50	7.665	7.736	7.632
75	7.186	7.697	7.722
100	7.330	7.774	7.788
125	7.786	7.801	7.679
150	8.049	7.829	7.771
Average	7.604	7.752	7.663

Table 4.6: The inference prediction MSE comparison between CVNN and CNN models for the untrained interference center frequencies.

inf center frequency	**prediction MSE** $[\times 10^{-3}]$		
[MHz]	CVNN	CNN1	CNN2
-137.5	8.764	8.636	8.361
-87.5	7.522	7.959	8.182
-62.5	7.816	7.887	8.024
-37.5	7.341	7.976	7.814
-12.5	7.069	8.178	7.89
12.5	7.328	8.072	8.092
37.5	7.825	7.903	8.026
62.5	7.497	8.113	8.027
87.5	7.287	8.100	8.172
137.5	8.377	8.442	8.268
Average	7.683	8.127	8.086

estimation square error with the correlator-based estimator is

$$\begin{aligned}\mathbb{E}\{||\hat{h}-h||^2\} &= \mathbb{E}\{||\frac{1}{L_{gp}}\boldsymbol{x}_{gp}^H(h\boldsymbol{x}_{gp}+\boldsymbol{n}_{inf})-h||^2\}\\ &= \frac{1}{L_{gp}^2}\boldsymbol{x}_{gp}^H\mathbb{E}[\boldsymbol{n}_{inf}^H\boldsymbol{n}_{inf}]\boldsymbol{x}_{gp}\\ &= \frac{1}{L_{gp}}\mathbb{E}[\boldsymbol{n}_{inf}^H\boldsymbol{n}_{inf}]\\ &= \frac{\sigma_{inf}^2}{L_{gp}},\end{aligned} \tag{4.12}$$

with $\boldsymbol{x}_{gp}^H\boldsymbol{x}_{gp} = L_{gp}$ as shown in Eq. 3.7.

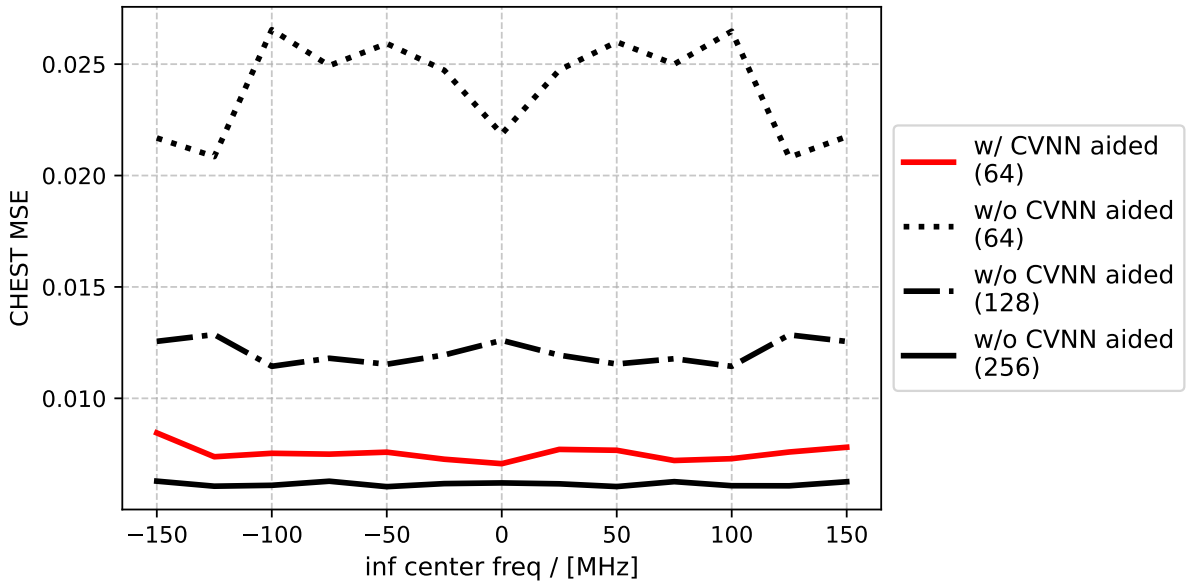


Figure 4.17: The channel estimation (CHEST) MSE vs the interference center frequency with various Golay pilot lengths by using CVNN or without using CVNN. The dotted black line represents the CHEST MSE by using correlator-based estimator only with 64 Golay pilot length; the dash-dotted black line represents the CHEST MSE by using correlator-based estimator only with 128 Golay pilot length; the solid black line represents the CHEST MSE by using correlator-based estimator only with 256 Golay pilot length; and the red solid line represents the performance of CVNN-aided estimator with 64 Golay pilot length.

This section compares the channel estimation MSE with various Golay pilot lengths. Fig. 4.17 shows the channel estimation performance vs. the interference center frequency with various Golay pilot lengths. The red line represents the performance by using the

CVNN-aided channel estimator, while the black lines represent the performance by only using the correlator-based channel estimator. Pilot lengths 64, 128, and 256 are evaluated. The result shows that the Golay pilot length needs to reach 256 to reduce the channel estimation MSE by the correlator-based estimator to the same order of magnitude as the channel estimation MSE estimated using the CVNN-aided estimator. This shows that the CVNN-aided channel estimator can save up to 4× of the pilot length compared with the traditional correlator-based channel estimator.

4.7 Discussion

4.7.1 Frontend Impairment

In Sec. 4.6, the effectiveness of the CVNN-aided channel estimation method has been demonstrated. The result shows that the CVNN-aided channel estimation method can significantly improve the channel estimation performance compared to the traditional method. However, this is based on the assumption of the perfect frontend design, where the I channel and the Q channel are balanced. Because IQ signal channels are balanced, the phase correlation embedded in the signal can be ensured coming from channel modulation only. In reality, the frontend impairment will exist. Table. 4.7 shows the channel estimation MSE with perfect frontends or a 10-degree IQ phase offset. Moreover, the model is trained with and without IQ imbalance. The result shows that the proposed method can still give a 3.11× channel estimation improvement compared to the traditional method when the IQ impairment exists, but a 1.65% performance degradation is introduced compared to the result obtained under the assumption of perfect IQ channels.

One alternative way to solve the IQ imbalance problem is to include frontend calibrations in the proposed CVNN-aided channel estimation method, such as the IQ calibration methods introduced in Chapter. 1. Because the hardware impairment will not change frequently, the calibration only needs to be performed once for a long time. The IQ correction hardware is also inexpensive, as shown in Fig. 3.6. An alternative way to adapt the proposed method to improve the performance under the IQ imbalance is to use a real-valued neural network and treat the real and imaginary parts as two channels. With this method, the degree of freedom of the neural network can be doubled, and the phase correlation could include both channel modulation and IQ imbalance. However, each antenna needs to be trained independently because the IQ imbalance factors are different for each frontend, which makes the model training expensive. In addition, doubling the parameters will increase memory usage, require more training data, and increase data load and store operation during computing. For those reasons, correcting IQ before feeding the received signal into the neural network is simpler and less expensive.

Table 4.7: Channel estimation MSE for the received signal assuming perfect frontend and 10-degree IQ phase offset.

	CHEST MSE $[10^{-3}]$					
	trained			**untrained**		
	inf center frequency [MHz]	no IQ imbalance	10 deg imbalance	inf center frequency [MHz]	no IQ imbalance	10 deg imbalance
	-150	8.448	8.341	-137.5	8.618	8.493
	-125	7.378	7.556	-87.5	7.467	7.607
	-100	7.533	7.685	-62.5	7.767	7.640
	-75	7.495	7.570	-37.5	7.312	7.562
	-50	7.584	7.715	-12.5	7.093	7.230
	-25	7.267	7.446	12.5	7.331	7.442
	0	7.067	7.149	37.5	7.787	7.789
	25	7.708	7.804	62.5	7.449	7.586
	50	7.668	7.773	87.5	7.295	7.464
	75	7.210	7.405	137.5	8.118	8.412
	100	7.291	7.628			
	125	7.591	7.620			
	150	7.805	8.24			
Average		7.542	7.687		7.624	7.723

4.7.2 Real-world Training

In this work, we train the proposed CVNN model with the simulated signals. Although the Python simulator can comprehensively simulate the massive MIMO system, training with actual data is still critical for making the ML method practical in the real world. The model training strategy applied in this work can be expanded to real-world data collection. Section 4.6 has shown that by training at interference center frequencies listed in Table 4.2, the CVNN model can perform well for other frequencies ranging from $[-150MHz, 150MHz]$. The training data can be collected using the Hydra testbed and an interference generator in the real world. The target data or the ground truth can be obtained by receiving the signals sent from UEs without turning on the interference generator. Then, the training data can be collected by receiving the same signals sent from UEs at the exact location with the interference generator turned on and adjusting the carrier frequency of the interference generator to manually place the interference baseband center frequency at the frequencies listed in Table 4.2. Then the collected training and target data pair can be used for the real-world training of the proposed CVNN model.

4.7.3 Interference Detection and Synchronization

Perfect synchronization and the existence of interference are assumed in this work. Interference detection and the pilot synchronization method are also needed in the system to make the proposed CVNN-aided channel estimator practical for real-time signal processing. One possible way to implement interference detection, pilot synchronization, and CVNN-aided channel estimation together is to design a multi-attention CVNN model, which is inspired by [146]. The multi-attention CVNN model first uses convolution layers to produce multiple-part attentions. Then an attention mechanism is applied to generate the part representations. The results of signal detection, synchronization, and denoised signals could be obtained simultaneously following different sets of fully-connected layers. The multi-attention CVNN model can be a future research direction for this work.

4.7.4 Computational Complexity

Although the proposed method outperforms the traditional method, the total number of complex-valued multiply-and-add required is

$$W_1 L C_{out,1} + W_2 \frac{L}{4} C_{in,2} C_{out,2} + \frac{L}{16}\frac{L}{16} C_{out,2}^2 + L\frac{L}{16} C_{out,2}, \tag{4.13}$$

where W_i represents the kernel size of each convolution layer, $C_{in,i}, C_{out,i}$ represents the number of input or output channels, and L represents the pilot length. Then the computational complexity of the proposed method is $\mathcal{O}(KM(L^2+L))$, whereas the computational complexity of the traditional correlator-based method is $\mathcal{O}(KMlog_2(L))$ and the method proposed in [27] is $\mathcal{O}(KML)$. The proposed method has a higher computational complexity than others. As a result, reducing the computational complexity and improving the computation efficiency is one of the future research directions for this work.

4.8 Summary

Table. 4.8 summarizes this work along with state-of-the-art ML and non-ML methods related to the interference issue in massive MIMO systems. As we have discussed in Sec. 4.6, CVNN outperforms two real-valued convolutional neural network designs, CNN1 and CNN2, with either similar layer structure or similar total number of parameters, especially for the cases where the interference center frequencies have not been included in the training phase. So CVNN has a better generalization ability than real-valued neural networks. The proposed method is also compared with the traditional correlator-based channel estimator, and the result shows that the proposed method has 3.16 times lower channel estimation MSE than the correlation-based channel estimator. In addition, the proposed method required four times shorter pilot length to achieve the same order of magnitude channel estimation MSE compared to the traditional method. The blind channel separation method proposed in [15] targets the active jamming attack in the channel and is a non-ML method for the user

channel estimation under interference. Compared with the blind channel separation method, the proposed method targets a more generic scenario with fewer restrictions on interference generation and is more practical in reality. Compared with the random matrix-based method [122], which also targets constant interference, the proposed method works well when the power of interference and the power of the desired signal is comparable. In contrast, the performance of the random matrix-based method degrades drastically when the interference power and the signal power are similar and cannot separate the user channel CSI. In summary, the proposed CVNN-aided channel estimation method can effectively improve the channel estimation performance under a wideband, constant, high-power interference with shorter pilot overhead. The use of complex-valued neural networks enables better generalization ability than state-of-the-art real-valued neural networks.

Table 4.8: Comparison table for the proposed CVNN-aided channel estimation method.

	CVNN	CNN1	CNN2	Correlation -based[25]	Blind Channel Separation[15]	Random matrix based[122]
interference type	constant	constant	constant	constant	active jamming	constant
P_{inf}/P_{sig}	1	1	1	1	can't be extremely large	<1
ML?	Yes	Yes	Yes	No	No	No
user CHEST	Yes	Yes	Yes	Yes	Yes	No
# antenna	64	64	64	64	-	256
# user	4	4	4	4	4	2
# params	51920	99880	51752	-	-	-
CHEST MSE[$\times 10^{-3}$]	7.58	7.95	7.87	23.9	3.70	-
generalization	High	Medium	Medium	-	-	-

Chapter 5

Conclusion and Future Works

In conclusion, this dissertation focuses on developing integrated signal processing for massive MIMO systems. It overviews the state-of-the-art methods of massive MIMO signal processing and system implementation. It also introduces the Spine DSP generator and the CVNN-aided channel estimation method that showcase the scalability and adaptability of massive MIMO systems and the potential of ML methods in future communication systems. The main contributions of this dissertation include the following:

- Provides a thorough analysis of the choices and considerations involved in designing and prototyping scalable massive MIMO systems from both algorithmic and hardware architecture perspectives. The focus is on achieving better data aggregation, linear scaling of computation complexity, cost and energy efficiency, and maintaining development sustainability. Those can be achieved by the modular, adaptable, and reusable design of massive MIMO systems that can fit different beamforming techniques, baseband signal processing algorithms, and hardware component designs.

- Demonstrates the feasibility of such massive MIMO systems by designing and evaluating an algorithm-adaptable, scalable, and platform-portable generator for massive MIMO baseband processing systems, which can be customized for different MIMO systems and hardware configurations. The generator is evaluated using various channel estimation methods and hardware parameter values, demonstrating its algorithmic adaptability and effectiveness. The results show that the generator has competitive power consumption and can significantly improve the demodulation error vector magnitude using beamspace methods.

- Shows the potential of the ML method for the signal processing of massive MIMO systems by proposing and evaluating a CVNN-aided channel estimation method for massive MIMO systems with interference. The CVNN is trained with signals containing wideband interference, and the method is evaluated with different design and training choices. The results show an up to 3.16 times improvement in channel estimation

performance with shorter pilots compared to the traditional method, demonstrating the effectiveness of the proposed method.

Although this dissertation has given comprehensive research on the design and prototyping the massive MIMO systems with integrated signal processing and the importance of ML in the integrated signal processing in massive MIMO systems, this dissertation can also be extended to further research directions, including:

- System-on-chip solutions of the massive MIMO systems with the Spine generator.
- ML generators for the CVNN-aided channel estimator, and integrate to the massive MIMO systems.
- Hardware implementations for the CVNN-aided channel estimator and integrated into the massive MIMO systems.

Bibliography

[1] *5G Simulation and MIMO Simulation Software*. https://www.remcom.com/5g-mimo. Accessed on 2023-02-13.

[2] A.Goldsmith. "5G and beyond: What lies ahead for wireless system design". In: *PIMRC*. 2014.

[3] Mohammed Abdelghany, Upamanyu Madhow, and Antti Tölli. "Beamspace Local LMMSE: An Efficient Digital Backend for mmWave Massive MIMO". In: *IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. 2019, pp. 1–5. DOI: 10.1109/SPAWC.2019.8815585.

[4] Yunus Emre ACAR, Murat CEYLAN, and Ercan YALDIZ. "An examination on the effect of CVNN parameters while classifying the real-valued balanced and unbalanced data". In: *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*. 2018, pp. 1–5. DOI: 10.1109/IDAP.2018.8620907.

[5] Ehab Ali, Mahamod Ismail, Rosdiadee Nordin, and Nor Fadzilah Abdulah. "Beamforming techniques for massive MIMO systems in 5G: overview, classification, and trends for future research". In: *Frontiers of Information Technology and Electronic Engineering*. Vol. 18. 6. 2017, pp. 753–772. DOI: 10.1631/FITEE.1601817.

[6] Muhammad Alrabeiah and Ahmed Alkhateeb. "Deep Learning for TDD and FDD Massive MIMO: Mapping Channels in Space and Frequency". In: Nov. 2019, pp. 1465–1470.

[7] N V Aravind, K Abhinandan, Vineeth V Acharya, and S. Sumam David. "Comparison of OMP and SOMP in the reconstruction of compressively sensed hyperspectral images". In: *2011 International Conference on Communications and Signal Processing*. 2011, pp. 188–192. DOI: 10.1109/ICCSP.2011.5739298.

[8] J. Bachrach et al. "Chisel: Constructing hardware in a Scala embedded language". In: *Design Automation Conference*. 2012, pp. 1212–1221. URL: https://doi.org/10.1145/2228360.2228584.

[9] Jamal Beiranvand and Hamid Meghdadi. "Analytical Performance Evaluation of MRC Receivers in Massive MIMO Systems". In: *IEEE Access* 6 (2018), pp. 53226–53234. DOI: 10.1109/ACCESS.2018.2866795.

[10] Mauro Belgiovine, Kunal Sankhe, Carlos Bocanegra, Debashri Roy, and Kaushik R. Chowdhury. "Deep Learning at the Edge for Channel Estimation in Beyond-5G Massive MIMO". In: *IEEE Wireless Communications* 28.2 (2021), pp. 19–25. DOI: 10.1109/MWC.001.2000322.

[11] Faouzi Bellili, Foad Sohrabi, and Wei Yu. "Generalized Approximate Message Passing for Massive MIMO mmWave Channel Estimation With Laplacian Prior". In: *IEEE Transactions on Communications* 67.5 (2019), pp. 3205–3219. DOI: 10.1109/TCOMM.2019.2892719.

[12] Tadilo Endeshaw Bogale and Long Bao Le. "Beamforming for multiuser massive MIMO systems: Digital versus hybrid analog-digital". In: *2014 IEEE Global Communications Conference*. 2014, pp. 4066–4071. DOI: 10.1109/GLOCOM.2014.7037444.

[13] John Brady, Nader Behdad, and Akbar M. Sayeed. "Beamspace MIMO for Millimeter-Wave Communications: System Architecture, Modeling, Analysis, and Measurements". In: *IEEE Transactions on Antennas and Propagation* 61.7 (2013), pp. 3814–3827. DOI: 10.1109/TAP.2013.2254442.

[14] F. Burkhardt, S. Jaeckel, E. Eberlein, and R. Prieto-Cerdeira. "QuaDRiGa: A MIMO channel model for land mobile satellite". In: *European Conference on Antennas and Propagation (EuCAP)*. 2014, pp. 1274–1278. DOI: 10.1109/EuCAP.2014.6902008. URL: https://doi.org/10.1109/EuCAP.2014.6902008.

[15] Ruohan Cao, Hui Gao, and Yueming Lu. "Channel Estimation for Massive MIMO Uplink Under Active Jamming Attack". In: *IEEE Systems Journal* (2022), pp. 1–12. DOI: 10.1109/JSYST.2022.3209790.

[16] Hei Victor Cheng and Erik G. Larsson. "Some fundamental limits on frequency synchronization in massive MIMO". In: *2013 Asilomar Conference on Signals, Systems and Computers*. 2013, pp. 1213–1217. DOI: 10.1109/ACSSC.2013.6810486.

[17] *Chisel3*. https://github.com/chipsalliance/chisel3. Accessed on 2023-02-01.

[18] Junil Choi, David J. Love, and Taeyoung Kim. "Trellis-Extended Codebooks and Successive Phase Adjustment: A Path From LTE-Advanced to FDD Massive MIMO Systems". In: *IEEE Transactions on Wireless Communications* 14.4 (2015), pp. 2007–2016. DOI: 10.1109/TWC.2014.2378778.

[19] T. Charles Clancy. "Efficient OFDM Denial: Pilot Jamming and Pilot Nulling". In: *2011 IEEE International Conference on Communications (ICC)*. 2011, pp. 1–5. DOI: 10.1109/icc.2011.5962467.

[20] S.F. Cotter and B.D. Rao. "Sparse channel estimation via matching pursuit with application to equalization". In: *IEEE Transactions on Communications* 50.3 (2002), pp. 374–377. DOI: 10.1109/26.990897.

[21] Giulio D'Amato, Gianfranco Avitabile, Giuseppe Coviello, and Claudio Talarico. "DDS-PLL Phase Shifter Architectures for Phased Arrays: Theory and Techniques". In: *IEEE Access* 7 (2019), pp. 19461–19470. DOI: 10.1109/ACCESS.2019.2895388.

[22] Linglong Dai, Xinyu Gao, Shuangfeng Han, I. Chih-Lin, and Xiaodong Wang. "Beamspace channel estimation for millimeter-wave massive MIMO systems with lens antenna array". In: *2016 IEEE/CIC International Conference on Communications in China (ICCC)*. 2016, pp. 1–6. DOI: 10.1109/ICCChina.2016.7636854.

[23] Y. Dai et al. "A Scalable Massive MIMO Uplink Baseband Processing Generator". In: *2020 IEEE ICC*. 2021, pp. 1–6.

[24] Yue Dai, Harrison Liew, Maryam Eslami Rasekh, Seyed Hadi Mirfarshbafan, Alexandra Gallyas-Sanhueza, James Dunn, Upamanyu Madhow, Christoph Studer, and Borivoje Nikolić. "A Scalable Generator for Massive MIMO Baseband Processing Systems with Beamspace Channel Estimation". In: *2021 IEEE Workshop on Signal Processing Systems (SiPS)*. 2021, pp. 182–187. DOI: 10.1109/SiPS52927.2021.00040. URL: https://doi.org/10.1109/SiPS52927.2021.00040.

[25] Yue Dai, Maryam Eslami Rasekh, Seyed Hadi Mirfarshbafan, Harrison Liew, Alexandra Gallyas-Sanhueza, James Dunn, Upamanyu Madhow, Christoph Studer, and Borivoje Nikolić. "An Adaptable and Scalable Generator of Distributed Massive MIMO Baseband Processing Systems". In: *Journal of Signal Processing Systems* 94.10 (2022), pp. 989–1003. DOI: 10.1007/s11265-022-01767-2.

[26] Li Deng. "The MNIST Database of Handwritten Digit Images for Machine Learning Research". In: *IEEE Signal Processing Magazine* 29.6 (2012), pp. 141–142. DOI: 10.1109/MSP.2012.2211477.

[27] Tan Tai Do, Emil Björnson, Erik G. Larsson, and S. Mohammad Razavizadeh. "Jamming-Resistant Receivers for the Massive MIMO Uplink". In: *IEEE Transactions on Information Forensics and Security* 13.1 (2018), pp. 210–223. DOI: 10.1109/TIFS.2017.2746007.

[28] Weisheng Dong, Peiyao Wang, Wotao Yin, Guangming Shi, Fangfang Wu, and Xiaotong Lu. "Denoising Prior Driven Deep Neural Network for Image Restoration". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 41.10 (Oct. 2019), pp. 2305–2318. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2018.2873610.

[29] *Ericsson Mobility Report*. https://www.ericsson.com/en/reports-and-papers/mobility-report. Accessed on 2023-03-13.

[30] Maryam Eslami Rasekh, Bhagyashree Puranik, Upamanyu Madhow, and Mark Rodwell. "In-the-Field Calibration of All-Digital MIMO Arrays". In: *2022 IEEE Wireless Communications and Networking Conference (WCNC)*. 2022, pp. 1389–1394. DOI: 10.1109/WCNC51071.2022.9771722.

[31] Felipe A. P. de Figueiredo, Fabbryccio A. C. M. Cardoso, Ingrid Moerman, and Gustavo Fraidenraich. "Channel estimation for massive MIMO TDD systems assuming pilot contamination and flat fading". In: *EURASIP Journal on Wireless Communications and Networking* 2018.14 (2018). DOI: 10.1186/s13638-018-1021-9.

[32] Zhen Gao, Linglong Dai, Zhaocheng Wang, and Sheng Chen. "Spatially Common Sparsity Based Adaptive Channel Estimation and Feedback for FDD Massive MIMO". In: *IEEE Transactions on Signal Processing* 63.23 (2015), pp. 6169–6183. DOI: 10.1109/TSP.2015.2463260.

[33] Enrique García et al. "Efficient filter for the generation/correlation of Golay binary sequence pairs". In: *International Journal of Circuit Theory and Applications* 42.10 (2014), pp. 1006–1015. URL: https://doi.org/10.1002/cta.1901.

[34] Navneet Garg, Mathini Sellathurai, and Tharmalingam Ratnarajah. "Low Complexity Joint OMP Methods for FDD Channel Estimation in Massive MIMO Systems". In: *2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. 2020, pp. 1–5. DOI: 10.1109/SPAWC48557.2020.9154302.

[35] Yinghao Ge, Weile Zhang, Feifei Gao, and Geoffrey Ye Li. "Frequency Synchronization for Uplink Massive MIMO With Adaptive MUI Suppression in Angle Domain". In: *IEEE Transactions on Signal Processing* 67.8 (2019), pp. 2143–2158. DOI: 10.1109/TSP.2019.2901344.

[36] *Genesis2*. URL: https://github.com/StanfordVLSI/Genesis2/wiki.

[37] M. Golay. "Complementary series". In: *IRE Transactions on Information Theory* 7.2 (1961), pp. 82–87. URL: https://doi.org/10.1109/TIT.1961.1057620.

[38] Lei Guan, Lorenzo Galati Giordano, and Andrea Bonfante. "A flexible HW and SW co-operated baseband research platform for massive MIMO system". In: *2017 IEEE International Conference on Communications (ICC)*. 2017, pp. 1–7. DOI: 10.1109/ICC.2017.7996696.

[39] Umar Hamid, Rahim Ali Qamar, and Kashif Waqas. "Performance comparison of time-domain and frequency-domain beamforming techniques for sensor array processing". In: *Proceedings of 2014 11th International Bhurban Conference on Applied Sciences and Technology (IBCAST) Islamabad, Pakistan, 14th-18th January, 2014*. 2014, pp. 379–385. DOI: 10.1109/IBCAST.2014.6778172.

[40] Yu Han, Qi Liu, Chao-Kai Wen, Shi Jin, and Kai-Kit Wong. "FDD Massive MIMO Based on Efficient Downlink Channel Reconstruction". In: *IEEE Transactions on Communications* 67.6 (2019), pp. 4020–4034. DOI: 10.1109/TCOMM.2019.2900625.

[41] Hengtao He, Chao-Kai Wen, Shi Jin, and Geoffrey Ye Li. "Deep Learning-Based Channel Estimation for Beamspace mmWave Massive MIMO Systems". In: *IEEE Wireless Communications Letters* 7.5 (2018), pp. 852–855. DOI: 10.1109/LWC.2018.2832128.

[42] Ruisi He, Bo Ai, Gongpu Wang, Mi Yang, Chen Huang, and Zhangdui Zhong. "Wireless Channel Sparsity: Measurement, Analysis, and Exploitation in Estimation". In: *IEEE Wireless Communications* 28.4 (2021), pp. 113–119. DOI: 10.1109/MWC.001.2000378.

[43] A. Hirose. "Continuous complex-valued back-propagation learning". In: *Electronics Letters* 28 (20 Sept. 1992), 1854–1855(1). ISSN: 0013-5194. DOI: 10.1049/el:19921186.

[44] K. Hornik, M. Stinchcombe, and H. White. "Multilayer Feedforward Networks Are Universal Approximators". In: *Neural Netw.* 2.5 (July 1989), pp. 359–366. ISSN: 0893-6080.

[45] Chongwen Huang, George C. Alexandropoulos, Alessio Zappone, Chau Yuen, and Merouane Debbah. "Deep Learning for UL/DL Channel Calibration in Generic Massive MIMO Systems". In: *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*. 2019, pp. 1–6. DOI: 10.1109/ICC.2019.8761962.

[46] Chongwen Huang, Lei Liu, Chau Yuen, and Sumei Sun. "Iterative Channel Estimation Using LSE and Sparse Message Passing for MmWave MIMO Systems". In: *IEEE Transactions on Signal Processing* 67.1 (2019), pp. 245–259. DOI: 10.1109/TSP.2018.2879620.

[47] Ademola E. Ilesanmi and Taiwo O. Ilesanmi. "Methods for image denoising using convolutional neural network: a review". In: *Complex and Intelligent Systems* 7.5 (Oct. 2021), pp. 2179–2198. ISSN: 2198-6053. DOI: 10.1007/s40747-021-00428-4.

[48] *Introduction to the NI MIMO Prototyping System Hardware*. National Instruments. URL: https://www.ni.com/en-us/shop/wireless-design-test/what-is-the-mimo-prototyping-system/introduction-to-the-ni-mimo-prototyping-system-hardware.html.

[49] Shahar Stein Ioushua and Yonina C. Eldar. "A Family of Hybrid Analog–Digital Beamforming Methods for Massive MIMO Systems". In: *IEEE Transactions on Signal Processing* 67.12 (2019), pp. 3243–3257. DOI: 10.1109/TSP.2019.2911255.

[50] *IWR6843ISK HORN ANTENNA (60 GHz)*. URL: https://dev.ti.com/tirex/explore/node?node=A__AEwJmUjT7zVFup.QEXFWNw__com.ti.mmwave_industrial_toolbox__VLyFKFf__4.9.0.

[51] Sven Jacobsson, Carl Lindquist, Giuseppe Durisi, Thomas Eriksson, and Christoph Studer. "Timing and Frequency Synchronization for 1-bit Massive MU-MIMO-OFDM Downlink". In: *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. 2019, pp. 1–5. DOI: 10.1109/SPAWC.2019.8815406.

[52] Elana Kalashnikov. "An Introduction to Golay Complementary Sequences". In: *Eureka* 4.1 (July 2014), pp. 40–48. URL: https://doi.org/10.29173/eureka22829.

[53] T. Karhima, A. Silvennoinen, M. Hall, and S.-G. Haggman. "IEEE 802.11b/g WLAN tolerance to jamming". In: *IEEE MILCOM 2004. Military Communications Conference, 2004.* Vol. 3. 2004, 1364–1370 Vol. 3. DOI: 10.1109/MILCOM.2004.1495141.

[54] Seonho Kim, Jeongmin Chae, and Song-Nam Hong. "Machine Learning Detectors for MU-MIMO Systems With One-Bit ADCs". In: *IEEE Access* 8 (2020), pp. 86608–86616.

[55] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *the 3rd International Conference for Learning Representations* (2015). DOI: 10.48550/arXiv.1412.6980.

[56] Weisi Kong, Hui Li, Shuangshuang Song, Yujie Fan, and Wenjie Zhang. "Compressive sensing based channel estimation for MIMO-OFDM systems". In: *IEEE Conference on Industrial Electronics and Applications (ICIEA)*. 2018, pp. 2164–2169. DOI: 10.1109/ICIEA.2018.8398069. URL: https://doi.org/10.1109/ICIEA.2018.8398069.

[57] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Commun. ACM* 60.6 (May 2017), pp. 84–90. ISSN: 0001-0782. DOI: 10.1145/3065386.

[58] Abhinav Kulkarni, Messaoud Ahmed Ouameur, and Daniel Massicotte. "Hardware Topologies for Decentralized Large-Scale MIMO Detection Using Newton Method". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 68.9 (2021), pp. 3732–3745. DOI: 10.1109/TCSI.2021.3097042.

[59] H.T Kung. "Let's Design Algorithms for VLSI Systems". In: *Caltech Conference On Very Large Scale Integration*. 1979, pp. 82–87.

[60] Greg LaCaille, James Dunn, Antonio Puglielli, Lorenzo Iotti, Sameet Ramakrishnan, Lucas Calderin, Zhenghan Lin, Emily Naviasky, Borivoje Nikolic, Ali Niknejad, and Elad Alon. "Design and Demonstration of a Scalable Massive MIMO Uplink at E-Band". In: *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*. 2020, pp. 1–6. DOI: 10.1109/ICCWorkshops49005.2020.9145323.

[61] Rodrigo C. de Lamare. "Massive MIMO systems: Signal processing challenges and future trends". In: *URSI Radio Science Bulletin* 2013.347 (2013), pp. 8–20. DOI: 10.23919/URSIRSB.2013.7909827.

[62] E. G. Larsson et al. "Massive MIMO for next generation wireless systems". In: *IEEE Communications Magazine* 52.2 (2014), pp. 186–195.

[63] Ha An Le, Trinh Van Chien, Tien Hoa Nguyen, Hyunseung Choo, and Van Duc Nguyen. "Machine Learning-Based 5G-and-Beyond Channel Estimation for MIMO-OFDM Communication Systems". In: *Sensors* 21.14 (2021). DOI: 10.3390/s21144861.

[64] Junho Lee, Gye-Tae Gil, and Yong H. Lee. "Channel Estimation via Orthogonal Matching Pursuit for Hybrid MIMO Systems in Millimeter Wave Communications". In: *IEEE Transactions on Communications* 64.6 (2016), pp. 2370–2386. DOI: 10.1109/TCOMM.2016.2557791.

[65] Philip H. W. Leong. "Recent Trends in FPGA Architectures and Applications". In: *4th IEEE International Symposium on Electronic Design, Test and Applications (delta 2008)*. 2008, pp. 137–141. DOI: 10.1109/DELTA.2008.14.

[66] Jiahui Li, Limin Xiao, Xibin Xu, and Shidong Zhou. "Robust and Low Complexity Hybrid Beamforming for Uplink Multiuser MmWave MIMO Systems". In: *IEEE Communications Letters* 20.6 (2016), pp. 1140–1143. DOI: 10.1109/LCOMM.2016.2542161.

[67] Kaipeng Li, Rishi R. Sharan, Yujun Chen, Tom Goldstein, Joseph R. Cavallaro, and Christoph Studer. "Decentralized Baseband Processing for Massive MU-MIMO Systems". In: *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 7.4 (2017), pp. 491–507. DOI: 10.1109/JETCAS.2017.2775151.

[68] Kaipeng Li, Bei Yin, Michael Wu, Joseph R. Cavallaro, and Christoph Studer. "Accelerating massive MIMO uplink detection on GPU for SDR systems". In: *2015 IEEE Dallas Circuits and Systems Conference (DCAS)*. 2015, pp. 1–4. DOI: 10.1109/DCAS.2015.7356600.

[69] Xueru Li, Emil Björnson, Erik G. Larsson, Shidong Zhou, and Jing Wang. "Massive MIMO with multi-cell MMSE processing: exploiting all pilots for interference suppression". In: vol. 2017. 117. 2017. DOI: 10.1186/s13638-017-0879-2.

[70] Jieyu Liao, Junhui Zhao, Feifei Gao, and Geoffrey Ye Li. "A Model-Driven Deep Learning Method for Massive MIMO Detection". In: *IEEE Communications Letters* 24.8 (2020), pp. 1724–1728. DOI: 10.1109/LCOMM.2020.2989672.

[71] Marc Lichtman, Roger Piqueras Jover, Mina Labib, Raghunandan Rao, Vuk Marojevic, and Jeffrey H. Reed. "LTE/LTE-A jamming, spoofing, and sniffing: threat assessment and mitigation". In: *IEEE Communications Magazine* 54.4 (2016), pp. 54–61. DOI: 10.1109/MCOM.2016.7452266.

[72] Jun Liu, Kai Mei, Xiaochen Zhang, Des McLernon, Dongtang Ma, Jibo Wei, and Syed Ali Raza Zaidi. "Fine Timing and Frequency Synchronization for MIMO-OFDM: An Extreme Learning Approach". In: *IEEE Transactions on Cognitive Communications and Networking* 8.2 (2022), pp. 720–732. DOI: 10.1109/TCCN.2021.3118465.

[73] Yang Liu and D.K. Borah. "Estimation of fading channels with large possible delay spreads". In: *Electronics Letters* 39 (1 Jan. 2003), 130–131(1). DOI: 10.1049/el:20030073.

[74] Rachit Mahendra, Saif Khan Mohammed, and Ranjan K. Mallik. "Performance of MRC and ZF Receivers in IQ-Impaired Uplink Massive MIMO Systems". In: *2021 International Conference on COMmunication Systems and NETworkS (COMSNETS)*. 2021, pp. 202–206. DOI: 10.1109/COMSNETS51098.2021.9352823.

[75] Steffen Malkowsky, João Vieira, Liang Liu, Paul Harris, Karl Nieman, Nikhil Kundargi, Ian C. Wong, Fredrik Tufvesson, Viktor Öwall, and Ove Edfors. "The World's First Real-Time Testbed for Massive MIMO: Design, Implementation, and Validation". In: *IEEE Access* 5 (2017), pp. 9073–9088. DOI: 10.1109/ACCESS.2017.2705561.

[76] T. L. Marzetta. "Noncooperative Cellular Wireless with Unlimited Numbers of Base Station Antennas". In: *IEEE Transactions on Wireless Communications* 9.11 (2010), pp. 3590–3600. DOI: 10.1109/TWC.2010.092810.091092.

[77] Zhinus Marzi, Dinesh Ramasamy, and Upamanyu Madhow. "Compressive Channel Estimation and Tracking for Large Arrays in mm-Wave Picocells". In: *IEEE Journal of Selected Topics in Signal Processing* 10.3 (2016), pp. 514–527. DOI: 10.1109/JSTSP.2016.2520899. URL: https://doi.org/10.1109/SPAWC.2019.8815585.

[78] Mahdi Boloursaz Mashhadi and Deniz Gündüz. "Pruning the Pilots: Deep Learning-Based Pilot Design and Channel Estimation for MIMO-OFDM Systems". In: *IEEE Transactions on Wireless Communications* 20.10 (2021), pp. 6315–6328. DOI: 10.1109/TWC.2021.3073309.

[79] *MATLAB Engine API for Python*. https://www.mathworks.com/help/matlab/matlab-engine-for-python.html. Accessed on 2021-06-19.

[80] Mehrtash Mehrabi, Mostafa Mohammadkarimi, Masoud Ardakani, and Yindi Jing. "Decision Directed Channel Estimation Based on Deep Neural Network k -Step Predictor for MIMO Communications in 5G". In: *IEEE Journal on Selected Areas in Communications* 37.11 (2019), pp. 2443–2456. DOI: 10.1109/JSAC.2019.2934004.

[81] Roi Méndez-Rial, Cristian Rusu, Nuria González-Prelcic, Ahmed Alkhateeb, and Robert W. Heath. "Hybrid MIMO Architectures for Millimeter Wave Communications: Phase Shifters or Switches?" In: *IEEE Access* 4 (2016), pp. 247–267. DOI: 10.1109/ACCESS.2015.2514261.

[82] *migen*. https://github.com/m-labs/migen. Accessed on 2023-05-01.

[83] Seyed Hadi Mirfarshbafan, Alexandra Gallyas-Sanhueza, Ramina Ghods, and Christoph Studer. "Beamspace Channel Estimation for Massive MIMO mmWave Systems: Algorithm and VLSI Design". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 67.12 (2020), pp. 5482–5495. DOI: 10.1109/TCSI.2020.3023023.

[84] Andreas F. Molisch, Vishnu V. Ratnam, Shengqian Han, Zheda Li, Sinh Le Hong Nguyen, Linsheng Li, and Katsuyuki Haneda. "Hybrid Beamforming for Massive MIMO: A Survey". In: *IEEE Communications Magazine* 55.9 (2017), pp. 134–141. DOI: 10.1109/MCOM.2017.1600400.

[85] Susnata Mondal, Rahul Singh, Ahmed I. Hussein, and Jeyanandh Paramesh. "A 25–30 GHz Fully-Connected Hybrid Beamforming Receiver for MIMO Communication". In: *IEEE Journal of Solid-State Circuits* 53.5 (2018), pp. 1275–1287. DOI: 10.1109/JSSC.2018.2789402.

[86] P.H. Moose. "A technique for orthogonal frequency division multiplexing frequency offset correction". In: *IEEE Transactions on Communications* 42.10 (1994), pp. 2908–2914. DOI: 10.1109/26.328961.

[87] Alireza Morsali and Benoit Champagne. "Achieving Fully-Digital Performance by Hybrid Analog/Digital Beamforming in Wide-Band Massive-Mimo Systems". In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, pp. 5125–5129. DOI: 10.1109/ICASSP40776.2020.9054156.

[88] Emily Naviasky, Lorenzo Iotti, Greg LaCaille, Borivoje Nikolić, Elad Alon, and Ali M. Niknejad. "A 71-to-86-GHz 16-Element by 16-Beam Multi-User Beamforming Integrated Receiver Sub-Array for Massive MIMO". In: *IEEE Journal of Solid-State Circuits* 56.12 (2021), pp. 3811–3826. DOI: 10.1109/JSSC.2021.3118641.

[89] Hien Quoc Ngo, Erik G. Larsson, and Thomas L. Marzetta. "Energy and Spectral Efficiency of Very Large Multiuser MIMO Systems". In: *IEEE Transactions on Communications* 61.4 (2013), pp. 1436–1449. DOI: 10.1109/TCOMM.2013.020413.110848.

[90] C. Nhat Cuong et al. "Hardware Implementation of the Efficient SOR-Based Massive MIMO Detection for Uplink". In: *2019 IEEE-RIVF*. 2019, pp. 1–6.

[91] B. Nikolic et al. "Generating the Next Wave of Custom Silicon". In: *IEEE European Solid-State Circuits Conference*. 2018, pp. 6–11. URL: https://doi.org/10.1109/ESSCIRC.2018.8494310.

[92] Borivoje Nikolic, Elad Alon, and Krste Asanovic. "Generating the Next Wave of Custom Silicon". In: *ESSCIRC 2018 - IEEE 44th European Solid State Circuits Conference (ESSCIRC)*. 2018, pp. 6–11. DOI: 10.1109/ESSCIRC.2018.8494310.

[93] Vukan Ninkovic, Aleksandar Valka, Dejan Dumic, and Dejan Vukobratovic. "Deep Learning-Based Packet Detection and Carrier Frequency Offset Estimation in IEEE 802.11ah". In: *IEEE Access* 9 (2021), pp. 99853–99865. DOI: 10.1109/ACCESS.2021.3096853.

[94] Praween Nishad et al. "Carrier frequency offset estimation in OFDM systems". In: *2013 IEEE Conference on Information Communication Technologies*. 2013, pp. 885–889.

[95] Matthew J. La Pan, T. Charles Clancy, and Robert W. McGwier. "Jamming attacks against OFDM timing synchronization and signal acquisition". In: *MILCOM 2012 - 2012 IEEE Military Communications Conference*. 2012, pp. 1–7. DOI: 10.1109/MILCOM.2012.6415749.

[96] Hossein Pirayesh and Huacheng Zeng. "Jamming Attacks and Anti-Jamming Strategies in Wireless Networks: A Comprehensive Survey". In: *IEEE Communications Surveys and Tutorials* 24.2 (2022), pp. 767–809. DOI: 10.1109/COMST.2022.3159185.

[97] Antonio Puglielli, Andrew Townley, Greg LaCaille, Vladimir Milovanović, Pengpeng Lu, Konstantin Trotskovsky, Amy Whitcombe, Nathan Narevsky, Gregory Wright, Thomas Courtade, Elad Alon, Borivoje Nikolić, and Ali M. Niknejad. "Design of Energy- and Cost-Efficient Massive MIMO Arrays". In: *Proceedings of the IEEE* 104.3 (2016), pp. 586–606. DOI: 10.1109/JPROC.2015.2492539.

[98] Chenhao Qi, Yongming Huang, Shi Jin, and Lenan Wu. "Sparse channel estimation based on compressed sensing for massive MIMO systems". In: *2015 IEEE International Conference on Communications (ICC)*. 2015, pp. 4558–4563. DOI: 10.1109/ICC.2015.7249041.

[99] Leevanshi Rao, Mohit Pant, Leeladhar Malviya, Ajay Parmar, and Sandhya Vijay Charhate. "5G beamforming techniques for the coverage of intended directions in modern wireless communication: in-depth review". In: *International Journal of Microwave and Wireless Technologies* 13.10 (2021), pp. 1039–1062. DOI: 10.1017/S1759078720001622.

[100] Jesús Rodríguez Sánchez, Fredrik Rusek, Ove Edfors, Muris Sarajlić, and Liang Liu. "Decentralized Massive MIMO Processing Exploring Daisy-Chain Architecture and Recursive Algorithms". In: *IEEE Transactions on Signal Processing* 68 (2020), pp. 687–700. DOI: 10.1109/TSP.2020.2964496.

[101] Chengyao Ruan, Zaichen Zhang, Hao Jiang, Jian Dang, Liang Wu, and Hongming Zhang. "Approximate Message Passing for Channel Estimation in Reconfigurable Intelligent Surface Aided MIMO Multiuser Systems". In: *IEEE Transactions on Communications* 70.8 (2022), pp. 5469–5481. DOI: 10.1109/TCOMM.2022.3182369.

[102] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. 2017. arXiv: 1609.04747 [cs.LG].

[103] F. Rusek et al. "Scaling Up MIMO: Opportunities and Challenges with Very Large Arrays". In: *IEEE Signal Processing Magazine* 30.1 (2013), pp. 40–60.

[104] Parna Sabeti, Arman Farhang, Nicola Marchetti, and Linda Doyle. "Frequency Synchronization for OFDM-Based Massive MIMO Systems". In: *IEEE Transactions on Signal Processing* 67.11 (2019), pp. 2973–2986. DOI: 10.1109/TSP.2019.2911249.

[105] Ahmad Kamsani Samingan, Ishak Suleiman, Chun Yeow Yeoh, and Abdul Aziz Abdul Rahman. "Receiver antenna synchronization and carrier frequency offsets pre-mitigation for distributed massive MIMO system". In: *2016 22nd Asia-Pacific Conference on Communications (APCC)*. 2016, pp. 457–461. DOI: 10.1109/APCC.2016.7581462.

[106] Magnus Sandell, Evgeny Tsimbalo, Seifallah Jardak, Daisuke Uchida, Koji Akita, Daiki Yoda, Tamio Kawaguchi, and Makoto Sano. "Estimation of Wideband IQ Imbalance in MIMO OFDM Systems With CFO". In: *IEEE Transactions on Wireless Communications* 20.9 (2021), pp. 5821–5830. DOI: 10.1109/TWC.2021.3070387.

[107] Guto Leoni Santos, Patricia Takako Endo, Djamel Sadok, and Judith Kelner. "When 5G Meets Deep Learning: A Systematic Review". In: *Algorithms* 13.9 (2020). ISSN: 1999-4893. DOI: 10.3390/a13090208.

[108] Ali H. Sayed. "Application: Channel and Noise Estimation". In: *Fundamentals of Adaptive Filtering*. 2003.

[109] T.M. Schmidl and D.C. Cox. "Robust frequency and timing synchronization for OFDM". In: *IEEE Transactions on Communications* 45.12 (1997), pp. 1613–1621. DOI: 10.1109/26.650240.

[110] A. Schuchert, R. Hasholzner, and P. Antoine. "A novel IQ imbalance compensation scheme for the reception of OFDM signals". In: *IEEE Transactions on Consumer Electronics* 47.3 (2001), pp. 313–318. DOI: 10.1109/30.964115.

[111] Wenqian Shen, Linglong Dai, Byonghyo Shim, Zhaocheng Wang, and Robert W. Heath. "Channel Feedback Based on AoD-Adaptive Subspace Codebook in FDD Massive MIMO Systems". In: *IEEE Transactions on Communications* 66.11 (2018), pp. 5235–5248. DOI: 10.1109/TCOMM.2018.2849755.

[112] Clayton Shepard, Hang Yu, Narendra Anand, Erran Li, Thomas Marzetta, Richard Yang, and Lin Zhong. "Argos: Practical Many-Antenna Base Stations". In: *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*. Mobicom '12. Istanbul, Turkey, 2012, pp. 53–64. DOI: 10.1145/2348543.2348553.

[113] Clayton Shepard, Hang Yu, and Lin Zhong. "ArgosV2: A Flexible Many-Antenna Research Platform". In: *Proceedings of the 19th Annual International Conference on Mobile Computing and Networking*. MobiCom '13. 2013, pp. 163–166. DOI: 10.1145/2500423.2505302.

[114] Devarpita Sinha, Anish Kumar Verma, and Sanjay Kumar. "Software defined radio: Operation, challenges and possible solutions". In: *2016 10th International Conference on Intelligent Systems and Control (ISCO)*. 2016, pp. 1–5. DOI: 10.1109/ISCO.2016.7727079.

[115] Mehran Soltani, Vahid Pourahmadi, Ali Mirzaei, and Hamid Sheikhzadeh. "Deep Learning-Based Channel Estimation". In: *IEEE Communications Letters* 23.4 (2019), pp. 652–655. DOI: 10.1109/LCOMM.2019.2898944.

[116] Q.H. Spencer, A.L. Swindlehurst, and M. Haardt. "Zero-forcing methods for downlink spatial multiplexing in multiuser MIMO channels". In: *IEEE Transactions on Signal Processing* 52.2 (2004), pp. 461–471. DOI: 10.1109/TSP.2003.821107.

[117] *SpinalHDL*. https://github.com/SpinalHDL/SpinalHDL. Accessed on 2023-05-01.

[118] Veljko Stankovic and Martin Haardt. "Generalized Design of Multi-User MIMO Precoding Matrices". In: *IEEE Transactions on Wireless Communications* 7.3 (2008), pp. 953–961. DOI: 10.1109/LCOMM.2008.060709.

[119] Gongguo Tang, Badri Narayan Bhaskar, Parikshit Shah, and Benjamin Recht. "Compressed Sensing Off the Grid". In: *IEEE Transactions on Information Theory* 59.11 (2013), pp. 7465–7490. DOI: 10.1109/TIT.2013.2277451.

[120] A. Tarighat, R. Bagheri, and A.H. Sayed. "Compensation schemes and performance analysis of IQ imbalances in OFDM receivers". In: *IEEE Transactions on Signal Processing* 53.8 (2005), pp. 3257–3268. DOI: 10.1109/TSP.2005.851156.

[121] *USRP Software Defined Radio Device*. Ettus Research. URL: https://www.ni.com/en-us/shop/hardware/products/usrp-software-defined-radio-device.html.

[122] Julia Vinogradova, Emil Björnson, and Erik G. Larsson. "Detection and mitigation of jamming attacks in massive MIMO systems using random matrix theory". In: *2016 IEEE 17th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. 2016, pp. 1–5. DOI: 10.1109/SPAWC.2016.7536868.

[123] Patrick Virtue, Stella X. Yu, and Michael Lustig. "Better than real: Complex-valued neural nets for MRI fingerprinting". In: *2017 IEEE International Conference on Image Processing (ICIP)*. 2017, pp. 3953–3957. DOI: 10.1109/ICIP.2017.8297024.

[124] *Vivado Design Suite User Guide: Power Analysis and Optimization*. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_1/ug907-vivado-power-analysis-optimization.pdf. Accessed on 2021-12-29.

[125] Angie Wang et al. "ACED: A Hardware Library for Generating DSP Systems". In: *Design Automation Conference (DAC)*. San Francisco, California, 2018. URL: https://doi.org/10.1109/DAC.2018.8465790.

[126] Angie Wang. "Agile Design of Generator-Based Signal Processing Hardware". PhD thesis. EECS Department, University of California, Berkeley, May 2019. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-15.html.

[127] Cheng-Xiang Wang, Marco Di Renzo, Slawomir Stanczak, Sen Wang, and Erik G. Larsson. "Artificial Intelligence Enabled Wireless Networking for 5G and Beyond: Recent Advances and Future Challenges". In: *IEEE Wireless Communications* 27.1 (2020), pp. 16–23. DOI: 10.1109/MWC.001.1900292.

[128] Kang Wang, Xi Yang, Xiao Li, Chao-Kai Went, and Shi Jin. "SDR Implementation of an End-to-End mmWave Testbed Based on Phased Antenna Array". In: *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*. 2019, pp. 1–6. DOI: 10.1109/WCSP.2019.8928067.

[129] Shengchu Wang and Lin Zhang. "Signal Processing in Massive MIMO With IQ Imbalances and Low-Resolution ADCs". In: *IEEE Transactions on Wireless Communications* 15.12 (2016), pp. 8298–8312. DOI: 10.1109/TWC.2016.2613871.

[130] Tianqi Wang, Chao-Kai Wen, Shi Jin, and Geoffrey Ye Li. "Deep Learning-Based CSI Feedback Approach for Time-Varying Massive MIMO Channels". In: *IEEE Wireless Communications Letters* 8.2 (2019), pp. 416–419. DOI: 10.1109/LWC.2018.2874264.

[131] Yifan Wang and Huarui Yin. "A Multi-Frequency-Point Algorithm for Frequency-Selective IQ imbalance Estimation in Massive MIMO System". In: *2020 IEEE Computing, Communications and IoT Applications (ComComAp)*. 2020, pp. 1–6. DOI: 10.1109/ComComAp51192.2020.9398893.

[132] Xiuhong Wei and Linglong Dai. "Channel Estimation for Extremely Large-Scale Massive MIMO: Far-Field, Near-Field, or Hybrid-Field?" In: *IEEE Communications Letters* 26.1 (2022), pp. 177–181. DOI: 10.1109/LCOMM.2021.3124927.

[133] *What Is GNU Radio*. GNU Radio Companion. URL: https://wiki.gnuradio.org/index.php?title=What_Is_GNU_Radio.

[134] Hengmiao Wu, Zhuo Sun, and Xue Zhou. "Deep learning-based frame and timing synchronization for end-to-end communications". In: *Journal of Physics: Conference Series*. Vol. 1169. 1. IOP Publishing. 2019, p. 012060. DOI: 10.1088/1742-6596/1169/1/012060.

[135] Zhilu Wu, Yanlong Zhao, Zhendong Yin, and Haochen Luo. "Jamming signals classification using convolutional neural network". In: *2017 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*. 2017, pp. 062–067. DOI: 10.1109/ISSPIT.2017.8388320.

[136] Binqi Yang, Zhiqiang Yu, Ji Lan, Ruoqiao Zhang, Jianyi Zhou, and Wei Hong. "Digital Beamforming-Based Massive MIMO Transceiver for 5G Millimeter-Wave Communications". In: *IEEE Transactions on Microwave Theory and Techniques* 66.7 (2018), pp. 3403–3418. DOI: 10.1109/TMTT.2018.2829702.

[137] Jide Yuan, Hien Quoc Ngo, and Michail Matthaiou. "Machine Learning-Based Channel Estimation in Massive MIMO with Channel Aging". In: *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*. 2019, pp. 1–5. DOI: 10.1109/SPAWC.2019.8815557.

[138] Alam Zaib, Mudassir Masood, Anum Ali, Weiyu Xu, and Tareq Y. Al-Naffouri. "Distributed Channel Estimation and Pilot Contamination Analysis for Massive MIMO-OFDM Systems". In: *IEEE Transactions on Communications* 64.11 (2016), pp. 4607–4621. DOI: 10.1109/TCOMM.2016.2593924.

[139] Matthew D. Zeiler. *ADADELTA: An Adaptive Learning Rate Method*. 2012. arXiv: 1212.5701 [cs.LG].

[140] Huacheng Zeng, Chen Cao, Hongxiang Li, and Qiben Yan. "Enabling jamming-resistant communications in wireless MIMO networks". In: *2017 IEEE Conference on Communications and Network Security (CNS)*. 2017, pp. 1–9. DOI: 10.1109/CNS.2017.8228622.

[141] Jianhua Zhang, Lei Tian, Yuning Wang, and Mengmeng Liu. "Selection Transmitting/Maximum Ratio Combining for Timing Synchronization of MIMO-OFDM Systems". In: *IEEE Transactions on Broadcasting* 60.4 (2014), pp. 626–636. DOI: 10.1109/TBC.2014.2365333.

[142] Weile Zhang, Feifei Gao, Shi Jin, and Hai Lin. "Frequency Synchronization for Uplink Massive MIMO Systems". In: *IEEE Transactions on Wireless Communications* 17.1 (2018), pp. 235–249. DOI: 10.1109/TWC.2017.2764894.

[143] Wence Zhang, Xiaoxuan Xia, Yinkai Fu, and Xu Bao. "Hybrid and full-digital beamforming in mmWave Massive MIMO systems: A comparison considering low-resolution ADCs". In: *China Communications* 16.6 (2019), pp. 91–102. DOI: 10.23919/JCC.2019.06.008.

[144] Yu Zhang, Muhammad Alrabeiah, and Ahmed Alkhateeb. "Deep Learning for Massive MIMO With 1-Bit ADCs: When More Antennas Need Fewer Pilots". In: *IEEE Wireless Communications Letters* 9.8 (2020), pp. 1273–1277. DOI: 10.1109/LWC.2020.2987893.

[145] Zhaoyang Zhang, Xiao Cai, Chunguang Li, Caijun Zhong, and Huaiyu Dai. "One-Bit Quantized Massive MIMO Detection Based on Variational Approximate Message Passing". In: *IEEE Transactions on Signal Processing* 66.9 (2018), pp. 2358–2373. DOI: 10.1109/TSP.2017.2786256.

[146] Heliang Zheng, Jianlong Fu, Tao Mei, and Jiebo Luo. "Learning Multi-attention Convolutional Neural Network for Fine-Grained Image Recognition". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 5219–5227. DOI: 10.1109/ICCV.2017.557.