

# Design Trade-offs for Decentralized Baseband Processing in Massive MU-MIMO Systems

Kaipeng Li<sup>1</sup>, James McNaney<sup>1</sup>, Chance Tarver<sup>1</sup>, Oscar Castañeda<sup>2</sup>,  
Charles Jeon<sup>2</sup>, Joseph R. Cavallaro<sup>1</sup>, and Christoph Studer<sup>2</sup>

<sup>1</sup>Department of Electrical and Computer Engineering, Rice University, Houston, TX

<sup>2</sup>Department of Electrical and Computer Engineering, Cornell University, Ithaca, NY

**Abstract**—Massive multi-user (MU) multiple-input multiple-output (MIMO) provides high spectral efficiency by means of spatial multiplexing and fine-grained beamforming. However, conventional base-station (BS) architectures for systems with hundreds of antennas that rely on centralized baseband processing inevitably suffer from (i) excessive interconnect data rates between radio-frequency circuitry and processing fabrics, and (ii) prohibitive complexity at the centralized baseband processor. Recently, decentralized baseband processing (DBP) architectures and algorithms have been proposed, which mitigate the interconnect bandwidth and complexity bottlenecks. This paper systematically explores the design trade-offs between error-rate performance, computational complexity, and data transfer latency of DBP architectures under different system configurations and channel conditions. Considering architecture, algorithm, and numerical precision aspects, we provide practical guidelines to select the DBP architecture and algorithm that are able to realize the full benefits of massive MU-MIMO in the uplink and downlink.

## I. INTRODUCTION

Massive multi-user (MU) multiple-input multiple-output (MIMO) will be a key technology component in fifth-generation (5G) and future wireless communication systems [1]. The idea of this technology is to equip the infrastructure base-stations (BSs) with hundreds of antenna elements while serving tens of user equipments (UEs) simultaneously and in the same frequency band. The presence of a large number of antennas at the BSs enables fine-grained beamforming, which provides higher spectral efficiency than traditional, small-scale MIMO systems [2], [3]. However, naïvely scaling up small-scale MIMO systems to large antenna arrays will inevitably result in a range of practical implementation challenges [4] which must be resolved before deploying massive MU-MIMO in practice.

### A. Challenges with Centralized Baseband Processing

The excessively large amount of raw baseband data that must be transferred between the BS antenna array and the baseband processing backhaul is among the most critical challenges that arise with large antenna arrays [4]–[7]. As an example, a 256-BS-antenna massive MU-MIMO system with 12-bit digital-to-analog converters (DACs) supporting a bandwidth of 80 MHz requires raw baseband data rates from and to the

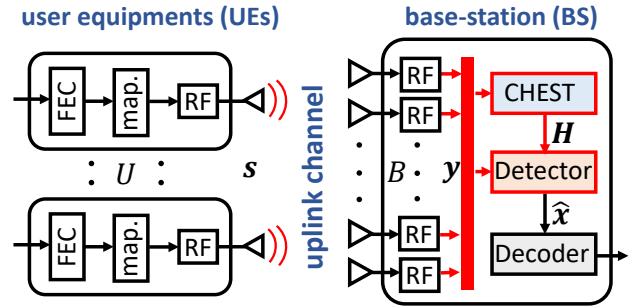


Fig. 1. Illustration of a base-station (BS) in the massive MU-MIMO uplink with a centralized baseband processing architecture. The large amount of raw baseband data that must be transferred between the RF chains and the computing fabric (illustrated with red color) is a key implementation challenge.

radio-frequency (RF) chains that approach 1 Tb/s. Such high data rates not only exceed the limits of existing interconnect technology by large margins, e.g., that of the Common Public Radio Interface (CPRI) [8], but also push chip input/output (I/O) interfaces, power dissipation, and processing capabilities of modern computing fabrics, such as graphics processing units (GPUs), field-programable gate arrays (FPGAs), or application-specific integrated circuits (ASICs), to their limits.

Figure 1 illustrates the interconnect bandwidth and complexity bottlenecks (highlighted with red color) in the massive MU-MIMO uplink (UEs transmit to BS) with conventional *centralized baseband processing* architectures. Note that we face the same bottlenecks in the downlink (BS transmits to UEs). Although one could resort to maximum ratio combining (MRC) for fully distributed data detection and maximum ratio transmission (MRT) for fully distributed precoding, MRC and MRT result in low spectral-efficiency compared to that of more complex centralized algorithms, such as minimum mean-square error (MMSE) equalization [2] or linear Wiener filter precoding [9]. Existing massive MU-MIMO testbeds, such as the BigStation [10] and the Lund testbed [11], incorporate centralized MMSE and ZF algorithms while parallelizing the computation workload across subcarriers. While this approach mitigates the complexity bottleneck, it still suffers from the interconnect and chip I/O bottlenecks as one must transfer raw baseband data from and to all antennas.

The work was supported in part by Xilinx, Inc. and by the US NSF under grants ECCS-1408370, CNS-1717218, CNS-1827940, ECCS-1408006, CCF1535897, CCF-1652065, and CNS-1717559.

## B. Decentralized Baseband Processing

To effectively avoid the bottlenecks of centralized baseband processing and to enable scalability to massive MU-MIMO architectures with hundreds or even thousands of antenna elements, recent work [4], [7], [12] introduced *decentralized baseband processing* (DBP). DBP partitions the antenna array into smaller clusters, each associated with separate RF circuitry and baseband processing fabrics. Each antenna cluster only connects with the associated computing fabrics which perform local baseband processing tasks, such as channel estimation, data detection in the uplink, and precoding in the downlink. While consensus-sharing based methods have been proposed for these tasks [4], [13], [14], the iterative exchange of information among clusters suffers from data transfer latency which negatively affects the design efficiency. Recently, the references [15]–[17] and [9] proposed *feedforward DBP architectures* for uplink detection and downlink precoding, respectively. Such feedforward architectures avoid the repeated exchange of information among clusters, which mitigates data transfer latency issues [9]. Furthermore, the theoretical analysis in [16] revealed that feedforward equalization architectures with linear algorithms are able to achieve the same or similar spectral efficiency as their centralized counterparts.

## C. Contributions

While the literature describes a number of *feedforward* DBP architectures and algorithms [7], [9], [16], a systematic trade-off analysis under different system configurations and channel conditions is missing. Such a trade-off analysis, however, is critical to making design decisions for practical massive MU-MIMO systems that rely on DBP. This paper focuses on such trade-offs at different levels for feedforward DBP architectures and algorithms which avoid iterative consensus exchange. In Section II, we analyze the performance and data transfer bandwidth trade-offs of two feedforward architectures dependent on system configuration and channel conditions, and we show that the channel coherence time is a critical design factor for *architecture-level* trade-offs. In Section III, we investigate the performance and complexity trade-offs of different feedforward equalization and precoding algorithms, and we show that channel coherence time and channel reciprocity are important design factors for *algorithm-level* trade-offs. In Section IV, we study the performance and efficiency trade-offs when reducing the arithmetic precision of data transfers and numerical computations. In Section V, we conclude the paper and summarize practical design guidelines.

## II. ARCHITECTURE TRADE-OFFS

We now study the performance and data transfer bandwidth trade-offs in the uplink and downlink for DBP feedforward architectures dependent on the channel's coherence time.

### A. System Models and Architectures

We now detail the uplink and downlink channel models and provide details on DBP feedforward architectures.

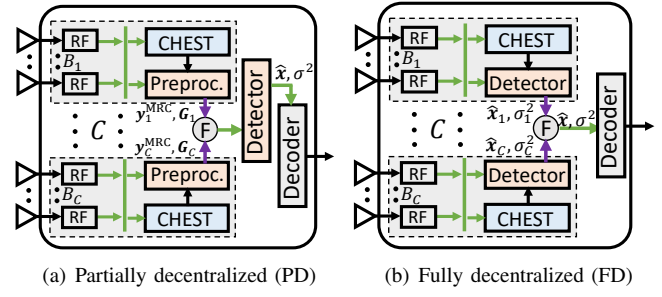


Fig. 2. Decentralized feedforward architectures for the massive MU-MIMO uplink. (a) The partially-decentralized (PD) architecture separates preprocessing and data detection. (b) The fully-decentralized (FD) architecture performs data detection at each antenna cluster followed by centralized data fusion.

1) *Uplink*: The uplink consists of  $U$  single-antenna UEs transmitting data to a BS with  $B$  antennas. The data of the  $U$  UEs is contained in the transmit vector  $\mathbf{x}^{\text{ul}} \in \mathcal{O}^U$ , where  $\mathcal{O}$  is the constellation (e.g., 16-QAM). The BS receives the vector  $\mathbf{y}^{\text{ul}} \in \mathbb{C}^B$ , which we model using the baseband input-output relation  $\mathbf{y}^{\text{ul}} = \mathbf{H}^{\text{ul}} \mathbf{x}^{\text{ul}} + \mathbf{n}^{\text{ul}}$ , where  $\mathbf{H}^{\text{ul}} \in \mathbb{C}^{B \times U}$  represents the channel matrix, and  $\mathbf{n}^{\text{ul}} \in \mathbb{C}^B$  is i.i.d. complex circularly-symmetric Gaussian noise with variance  $N_0$  per entry.

The feedforward DBP architectures proposed in [15], [16] partition the  $B$  BS-antennas into  $C$  antenna clusters; Fig. 2 illustrates two DBP architectures. The  $c^{\text{th}}$  cluster has  $B_c$  antennas where  $B = \sum_{c=1}^C B_c$ , and each cluster receives its own  $\mathbf{y}_c^{\text{ul}} \in \mathbb{C}^{B_c}$ . The input-output relation at each cluster is then

$$\mathbf{y}_c^{\text{ul}} = \mathbf{H}_c^{\text{ul}} \mathbf{x}^{\text{ul}} + \mathbf{n}_c^{\text{ul}}, \quad c = 1, 2, \dots, C. \quad (1)$$

Here,  $\mathbf{H}_c^{\text{ul}} \in \mathbb{C}^{B_c \times U}$  is the local channel matrix, a sub-matrix of  $\mathbf{H}^{\text{ul}}$ , and  $\mathbf{n}_c^{\text{ul}} \in \mathbb{C}^{B_c}$  represents the local noise vector at cluster  $c$ . We focus on two distinct feedforward DBP architectures put forward in [15]: the partially-decentralized (PD) architecture and fully-decentralized (FD) architecture shown in Fig. 2(a) and Fig. 2(b), respectively. Both architectures allow each BS cluster to estimate the local matrix  $\mathbf{H}_c^{\text{ul}}$  and perform local preprocessing based on  $\mathbf{H}_c^{\text{ul}}$  and the local receive vector  $\mathbf{y}_c^{\text{ul}}$  to form the local Gram matrix  $\mathbf{G}_c^{\text{ul}} = \mathbf{H}_c^{\text{ul}H} \mathbf{H}_c^{\text{ul}}$  and matched filter output  $\mathbf{y}_c^{\text{MRC}} = \mathbf{H}_c^{\text{ul}H} \mathbf{y}_c^{\text{ul}}$ . The PD architecture performs fusion of  $\mathbf{G}_c^{\text{ul}}$  and  $\mathbf{y}_c^{\text{MRC}}$  at the centralized processing unit to produce the equalization output  $\hat{\mathbf{x}}^{\text{ul}}$ . The FD architecture performs local equalization and requires data fusion of the local equalization outputs  $\hat{\mathbf{x}}_c^{\text{ul}}$  and noise variance  $\hat{\sigma}_c^{\text{ul}}$  to produce the equalization output  $\hat{\mathbf{x}}^{\text{ul}}$ . See [15], [16] for the details.

2) *Downlink System*: In the downlink, the BS computes the precoding vector  $\mathbf{x}^{\text{dl}} = \mathbf{P}^{\text{dl}} \mathbf{s}^{\text{dl}}$ , where  $\mathbf{x}^{\text{dl}} \in \mathbb{C}^B$ ,  $\mathbf{P}^{\text{dl}} \in \mathbb{C}^{B \times U}$  is the precoding matrix and  $\mathbf{s}^{\text{dl}} \in \mathcal{O}^U$  is the transmit data vector. At the UEs, the receive vector  $\mathbf{y}^{\text{dl}} \in \mathbb{C}^U$  is given by  $\mathbf{y}^{\text{dl}} = \mathbf{H}^{\text{dl}} \mathbf{x}^{\text{dl}} + \mathbf{n}^{\text{dl}}$ , where  $\mathbf{H}^{\text{dl}} \in \mathbb{C}^{U \times B}$  and  $\mathbf{n}^{\text{dl}} \in \mathbb{C}^U$  are the downlink channel matrix and noise vector, respectively.

Analogous to the uplink, reference [9] proposed feedforward DPB architectures for the downlink, where each of the  $C$  antenna clusters uses the local downlink channel matrix  $\mathbf{H}_c^{\text{dl}} = \mathbf{H}_c^{\text{ul}T}$ , the transpose of  $\mathbf{H}_c^{\text{ul}}$ , assuming channel reciprocity, in

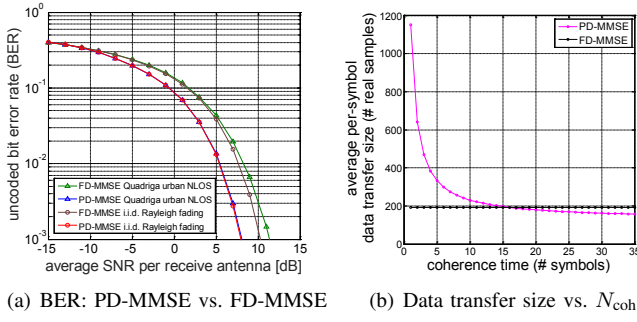


Fig. 3. PD vs. FD architecture trade-off on uncoded BER and data transfer size.

order to form the local beamforming vector  $\mathbf{x}_c^{\text{dl}}$ . See [9] for the details. The input-output relation at each cluster is

$$\mathbf{y}_c^{\text{dl}} = \mathbf{H}_c^{\text{dl}} \mathbf{x}_c^{\text{dl}} + \mathbf{n}_c^{\text{dl}}, \quad \mathbf{H}_c^{\text{dl}} \in \mathbb{C}^{U \times B_c}, \quad c = 1, 2, \dots, C. \quad (2)$$

### B. PD Architecture vs. FD Architecture

We now compare the PD architecture and FD architecture and explore the trade-offs for architecture selection. We first focus on a case study of the linear MMSE data detector in the uplink with the PD and FD architectures, to showcase how architecture selection for a certain algorithm affects the error rate performance, computational complexity, and data transfer size under different antenna configurations and channel conditions, such as channel coherence time. We later extend our analysis to precoding in the downlink and discuss more algorithm variations and their trade-offs in Section III.

Linear uplink data detection can be formulated as the following optimization problem (we omit the superscript <sup>ul</sup>):

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x} \in \mathbb{C}^U} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 + \rho \|\mathbf{x}\|_2^2. \quad (3)$$

Here,  $\rho = \frac{N_0}{E_s}$  is used for linear MMSE equalization where  $E_s$  is the per-UE transmit power. The corresponding closed form solution for (centralized) MMSE detection is given by  $\hat{\mathbf{x}} = (\mathbf{H}^H \mathbf{H} + \rho \mathbf{I})^{-1} \mathbf{H}^H \mathbf{y}$ . For PD-MMSE detection, we first compute  $\mathbf{G}_c = \mathbf{H}_c^H \mathbf{H}_c$  and  $\mathbf{y}_c^{\text{MRC}} = \mathbf{H}_c^H \mathbf{y}$  at each decentralized cluster. We then fuse the matrices  $\mathbf{G}_c$  and  $\mathbf{y}_c^{\text{MRC}}$  to form the global Gram matrix  $\mathbf{G} = \mathbf{H}^H \mathbf{H} = \sum_{c=1}^C \mathbf{G}_c$  and MRC vector  $\mathbf{y}^{\text{MRC}} = \mathbf{H}^H \mathbf{y} = \sum_{c=1}^C \mathbf{y}_c^{\text{MRC}}$  at the centralized processor. Finally, we calculate the matrix inversion to perform equalization at the centralized processor, which yields the global equalization output  $\hat{\mathbf{x}}$  and noise variance  $\sigma^2$ . For FD-MMSE detection, we first compute  $\mathbf{G}_c$  and  $\mathbf{y}_c^{\text{MRC}}$  locally, the same as for PD-MMSE. We then calculate a matrix inversion and perform equalization locally to obtain the local estimate  $\hat{\mathbf{x}}_c$  and noise variance  $\sigma_c^2$ . Finally, we fuse the local estimates  $\hat{\mathbf{x}}_c$  together with the local noise variances  $\sigma_c^2$  at the centralized processor, which yields the global equalization output  $\hat{\mathbf{x}} = \sum_{c=1}^C \lambda_c \hat{\mathbf{x}}_c$ , where  $\lambda_c = \frac{1}{\sigma_c^2 (\sum_{c'=1}^C \frac{1}{\sigma_{c'}^2})^{-1}}$ ; see [15], [16] for the details.

From the above explanations, we see that the *timing complexity*, which we measure as the number of real-valued multi-

plications, is approximately the same<sup>1</sup> for both architectures. The key factor that can lead to different efficiency of PD and FD architectures lies at the data transfer size at the data fusion stage. For the PD architecture, data fusion requires the transfer of both local  $\mathbf{G}_c$ , a  $U \times U$  complex-value Hermitian matrix consisting of  $U^2$  unique real values and local  $\mathbf{y}_c^{\text{MRC}}$ , a  $U$ -dimensional complex-valued vector that contains  $2U$  real values, which leads to a total of  $C \times (U^2 + 2U)$  real values for each data symbol. If we consider a typical scenario for which the estimated channel in the uplink is static across  $N_{\text{coh}}$  contiguous symbols, then the local Gram matrix  $\mathbf{G}_c$ , which depends only on the local channel matrix  $\mathbf{H}_c$ , can be transferred only once for every  $N_{\text{coh}}$  symbols, while the transfer of  $\mathbf{y}_c^{\text{MRC}}$  is required for every symbol. Therefore, the average data transfer size  $m_{\text{PD}}$  for each symbol at the fusion stage of PD architecture is as follows:

$$m_{\text{PD}} = \frac{C \times (U^2 + 2N_{\text{coh}}U)}{N_{\text{coh}}}. \quad (4)$$

In contrast, for the FD architecture, data fusion requires the transfer of the local  $\mathbf{x}_c$ , a  $U$ -entry complex-valued vector, and  $\sigma_c$ , a  $U$ -entry real-valued vector, which leads to a total of  $3U$  real values that must be transferred for every symbol regardless of the channel coherence time. Therefore, the average data transfer size  $m_{\text{FD}}$  for each symbol in FD architecture is

$$m_{\text{FD}} = \frac{C \times 3N_{\text{coh}}U}{N_{\text{coh}}} = 3CU. \quad (5)$$

When extended to a multi-subcarrier transmission, e.g., using orthogonal frequency-division multiplexing (OFDM), the quantities  $m_{\text{PD}}$  and  $m_{\text{FD}}$  represent the average data transfer size per symbol *on each subcarrier*. From (4) and (5), we see that the channel coherence time determines whether the PD or FD architecture requires more or less data to be transferred.

### C. Trade-off Analysis

While the data transfer size and efficiency of the PD and FD architectures depend on system parameters and channel conditions, the PD architecture always outperforms the FD architecture in terms of error-rate performance, as it is able to achieve the same performance as centralized data detection [15], [16]. Figure 3(a) compares the uncoded bit error-rate (BER) of PD-MMSE and FD-MMSE data detection for a single-carrier system with a cluster size of  $B_c = 32$ ,  $C = 4$  clusters, and  $U = 16$  UEs with 16-QAM. We simulate the BER performance with a simple i.i.d. Rayleigh fading channel, and a more realistic urban micro-campus non-line-of-sight (NLOS) Quadriga channel [18], where we place the UEs randomly in a  $120^\circ$  sector at a distance of 50 to 100 from the BS, which is using a uniform linear array.

We see that PD-MMSE clearly outperforms FD-MMSE under both channel environments in terms of uncoded BER. To showcase the trade-off of PD and FD on BER vs. data transfer

<sup>1</sup>Here, the timing complexity indicates that decentralized computations performed *in parallel* at each cluster should be counted *only once* to reflect the computation latency on real hardware.

size, in Fig. 3(b), we also compare the average per-symbol data transfer size  $m_{PD}$  and  $m_{FD}$  for PD and FD architectures, respectively, under the same antenna configuration but for different channel coherence times characterized by  $N_{coh}$ . We find that at small coherence times  $N_{coh}$ , if  $N_{coh} < U$ , then  $m_{FD} < m_{PD}$ , which implies that one should select the FD rather than the PD architecture if the data fusion efficiency has higher priority than BER performance. If  $N_{coh} > U$ , then  $m_{PD} < m_{FD}$ , then the PD architecture is always preferred for both better data fusion efficiency and BER performance.

Similarly to the uplink, we can use the PD and FD architectures to realize decentralized precoding in the downlink, e.g., using linear Wiener filter (WF) precoding [9]. For PD-WF precoding, the total data transfer size, which consists of both fusion of local Gram matrices  $\mathbf{G}_c$  and broadcasting of centralized whitened vector (scaled with  $U$ ), is always larger than that of FD-WF precoding, which only requires the broadcasting of the transmit vector  $\mathbf{s}$  (scaled with  $U$ ) and a scalar power allocation value. We can calculate similar average per-symbol data transfer sizes for downlink precoding by  $m_{PD} = \frac{C(U^2 + 2N_{coh}U)}{N_{coh}}$  and  $m_{FD} = \frac{C(1 + 2N_{coh}U)}{N_{coh}}$ . Clearly, the FD architecture is more efficient on data transfer since  $m_{FD} < m_{PD}$  holds in typical scenarios. Only if  $N_{coh} \gg U^2$ , i.e., for nearly static channels, we have  $m_{FD} \approx m_{PD}$ . However, for the BER performance, PD-WF is never worse than that of FD-WF. Therefore, selecting the PD or FD architecture for the downlink is solely determined by the system designer's preference, i.e., whether BER or data transfer is the critical factor.

### III. ALGORITHM TRADE-OFFS

We now investigate the selection of data detection or precoding algorithms, given that either the PD or FD architecture has been already chosen. We first discuss the trade-offs for various decentralized algorithms, where we focus on *explicit* and *implicit* matrix-inversion-based uplink equalization methods with the PD architecture. A similar trade-off analysis applies for the FD architecture and downlink systems. We also explore situations in which we are able to store and reuse computation results from the uplink in order to facilitate downlink precoding, which enables additional complexity reduction in a time-division duplex (TDD) system.

#### A. Explicit vs. Implicit Algorithms

As discussed in Section II, a closed form solution for (centralized) MMSE data detection is given by  $\hat{\mathbf{x}} = (\mathbf{G} + \rho\mathbf{I})^{-1}\mathbf{y}^{MRC}$  and requires (i) Gram matrix computation and (ii) matrix inversion of the regularized Gram matrix, both of which are computationally intensive but only depend on the channel matrix  $\mathbf{H}$ . For the PD or FD architectures, the timing complexity of Gram matrix computation at each cluster is reduced because of smaller matrix size ( $B_c \times U$ ), while the dimension of the complete Gram matrix is still the same (i.e.,  $U \times U$ ). This implies that the matrix inversion timing complexity is the same as that of the centralized MMSE equalizer. To reduce complexity, we can (i) take advantage of channel coherence,

which allows us to reuse intermediate computation results across  $N_{coh}$  symbols, and (ii) avoid an explicit computation of the matrix inversion.

Consider PD-MMSE as an example. At each cluster, if we compute  $\mathbf{G}_c$  only once and reuse it across  $N_{coh}$  receive symbols, then the average timing complexity (number of real-valued multiplications) per receive symbol for computing  $\mathbf{G}_c + \rho\mathbf{I}$  can be as low as  $2B_cU^2/N_{coh}$ . However, the computation of  $\mathbf{y}_c^{MRC} = \mathbf{H}_c^H \mathbf{y}_c$  relies not only on channel  $\mathbf{H}_c$  but also on the receive symbol  $\mathbf{y}_c$ , which leads to  $4B_cU$  timing complexity per receive symbol. Then for explicit Cholesky-based matrix inversion of a  $U \times U$  matrix, which only depends on  $\mathbf{H}$ , we can still compute the matrix inversion only once and reuse it across  $N_{coh}$  receive symbols, and therefore the average complexity per receive symbol is  $(\frac{10}{3}U^3 - \frac{4}{3}U)/N_{coh}$  [19]. Finally, to compute the equalization output  $\hat{\mathbf{x}}$ , the matrix-vector multiplication requires an additional  $4U^2$  operations per receive symbol. Therefore, the timing complexity  $n_{ex}$  of all above steps to obtain  $\hat{\mathbf{x}}$  averaged on each symbol for the *explicit* matrix inversion based PD-MMSE is

$$n_{ex} = (2B_cU^2 + \frac{10}{3}U^3 - \frac{4}{3}U)/N_{coh} + 4B_cU + 4U^2. \quad (6)$$

The literature describes a number of implicit methods that can be used for directly computing  $\hat{\mathbf{x}}$  while avoiding an explicit matrix inversion, such as the decentralized conjugate gradient method [13] or decentralized coordinate descent method [20]. Such iterative methods typically obtain an approximate result<sup>2</sup>, which entails a small BER loss. Furthermore, such implicit methods are unable to exploit the benefits of channel coherence since all iterative updates need to be computed for every symbol. We therefore propose to integrate *implicit* Cholesky-based MMSE detection [19] with the PD architecture, which is not only able to compute the exact linear MMSE equalizer but also realize low complexity compared to other iterative methods, especially for large  $N_{coh}$  when the intermediate Cholesky decomposition results can be reused. The *implicit* PD-MMSE algorithm computes the local Gram matrix  $\mathbf{G}_c$  and MRC vector  $\mathbf{y}_c^{MRC}$  and fuses them to obtain the global Gram matrix  $\mathbf{G}$  and MRC vector  $\mathbf{y}^{MRC}$  similarly to explicit PD-MMSE. After fusion, the implicit PD-MMSE method factorizes the regularized Gram matrix  $\mathbf{A} = \mathbf{G} + \rho\mathbf{I}$  by Cholesky decomposition  $\mathbf{A} = \mathbf{L}\mathbf{L}^H$ , where  $\mathbf{L}$  represents a lower triangular matrix. One can then solve  $\mathbf{L}\mathbf{z} = \mathbf{y}^{MRC}$  and finally  $\mathbf{L}^H\hat{\mathbf{x}} = \mathbf{z}$ , by forward and backward substitution, respectively, in order to obtain  $\hat{\mathbf{x}}$ . While the forward and backward substitutions have to be carried out for every symbol with a total of  $4U^2$  operations (including real-valued multiplications and divisions), the Cholesky decomposition, which dominates the complexity at  $\frac{2}{3}U^3 - \frac{2}{3}U$ , can be computed only once every  $N_{coh}$  symbols. Therefore, the resulting timing complexity  $n_{im}$  per symbol for the implicit Cholesky based PD-MMSE approach is

$$n_{im} = (2B_cU^2 + \frac{2}{3}U^3 - \frac{2}{3}U)/N_{coh} + 4B_cU + 4U^2. \quad (7)$$

<sup>2</sup>While the conjugate gradient method can obtain the exact MMSE solution after  $U$  iterations, we typically perform a smaller number of iterations (less than  $U$ ) to reduce complexity.



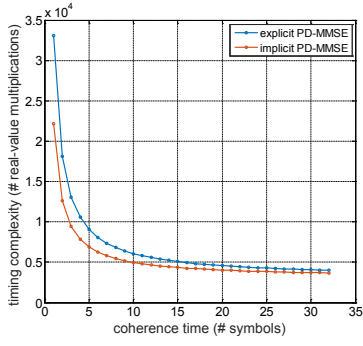


Fig. 4. Complexity of explicit vs. implicit PD-MMSE at different  $N_{\text{coh}}$ .

In OFDM systems, the quantities  $n_{\text{ex}}$  and  $n_{\text{im}}$  represent the detection complexity on each subcarrier.

By comparing (6) and (7), we see that the  $U^3$  term in  $n_{\text{ex}}$  and  $n_{\text{im}}$  dominates the complexity and  $n_{\text{im}} < n_{\text{ex}}$  since  $n_{\text{im}}$  has smaller constant associated with this term; this indicates that the implicit method can reduce complexity at *no loss* in terms of BER compared to the explicit method. Figure 4 shows an example for a system with  $B_c = 32$ ,  $U = 16$ , and  $C = 4$  where we compare the timing complexity of explicit PD-MMSE and implicit PD-MMSE depending on the coherence time. We observe that the implicit method always achieves lower complexity, whereas the complexity is similar to that of the explicit method for large values of  $N_{\text{coh}}$ .

We conclude by noting that we can similarly leverage both channel coherence and implicit inversion to reduce the complexity for FD-based uplink data detection. However, for the downlink, implicit inversion is not particularly helpful for the PD-WF or FD-WF precoding algorithms, which require the computation of an optimal scaling factor that depends on the explicit matrix inversion result of a regularized Gram matrix [9].

As an alternative, one can resort to ZF-based algorithms (e.g., PD-ZF and FD-ZF), which enable implicit methods for both uplink detection and downlink precoding. Here, for ZF precoding, we can simply scale the normalized ZF precoding vector  $\frac{\tilde{\mathbf{x}}}{\|\tilde{\mathbf{x}}\|_2}$  without the need of explicit matrix inversion to satisfy the same transmit power constraint as WF precoding.

#### B. Reusing Uplink Results for Downlink

Due to channel reciprocity in TDD systems, we have  $\mathbf{H}^{\text{dl}} = (\mathbf{H}^{\text{ul}})^T$ , and therefore  $\mathbf{G}^{\text{dl}} = \mathbf{H}^{\text{dl}}(\mathbf{H}^{\text{dl}})^H = (\mathbf{H}^{\text{ul}})^T(\mathbf{H}^{\text{ul}})^* = ((\mathbf{H}^{\text{ul}})^H \mathbf{H}^{\text{ul}})^* = (\mathbf{G}^{\text{ul}})^* = (\mathbf{G}^{\text{ul}})^T$ , where  $(\cdot)^*$  indicates entry-wise complex conjugate operation and  $\mathbf{G}^{\text{ul}}$  is a Hermitian matrix. This indicates that the Gram matrix computed in the uplink, can also be reused in the downlink by a simple transpose. We can also take advantage of channel reciprocity under decentralized architectures. For example, in FD-MMSE, the local Gram matrix  $\mathbf{G}_c^{\text{ul}}$  is computed. If we perform FD-WF precoding in the downlink, and each cluster estimates the local precoding vector  $\hat{\mathbf{x}}_c = \frac{1}{\beta_c}(\mathbf{H}_c^{\text{dl}})^H(\mathbf{G}_c^{\text{dl}} + \kappa_c \mathbf{I})^{-1}\mathbf{s}$ , where  $\frac{1}{\beta_c}$  is the scaling factor to satisfy the transmit power constraint and  $\kappa_c$  regularizes  $\mathbf{G}_c^{\text{dl}}$  as detailed in [9], then we can store the uplink matrix  $\mathbf{G}_c^{\text{ul}}$  and reuse it as  $\mathbf{G}_c^{\text{dl}} = (\mathbf{G}_c^{\text{ul}})^T$  for FD-WF

precoding, even across  $N_{\text{coh}}$  symbols. This approach yields an additional  $2B_c U^2 / N_{\text{coh}}$  complexity reduction per symbol for a system that integrates FD-MMSE detection and FD-WF precoding by re-using intermediate results compared to the total complexity of computing them individually. A similar approach can be used for PD-MMSE detection and PD-WF precoding: in PD-MMSE detection, we aggregate local  $\mathbf{G}_c^{\text{ul}}$  and compute the global  $\mathbf{G}^{\text{ul}}$  at the centralized node, and transpose it for PD-WF precoding.

If we consider ZF detection and precoding, where the Gram regularization coefficients  $\rho = 0$  and  $\kappa = 0$ , respectively, we can even store the *matrix inversion* result  $(\mathbf{G}^{\text{ul}})^{-1}$  computed in ZF detection and reuse it for ZF precoding which requires  $(\mathbf{G}^{\text{dl}})^{-1} = ((\mathbf{G}^{\text{ul}})^T)^{-1} = ((\mathbf{G}^{\text{ul}})^{-1})^T$ . When  $(\mathbf{G}^{\text{ul}})^{-1}$  is computed explicitly, we can also use it to compute the scaling factor  $\beta$  for ZF precoding in a similar way like WF precoding [9]. When it is computed implicitly, we should store and reuse the Cholesky decomposition result  $\mathbf{G}^{\text{ul}} = \mathbf{L}\mathbf{L}^H$  rather than the matrix inversion result for implicit ZF precoding which relies on  $\mathbf{G}^{\text{dl}} = (\mathbf{L}\mathbf{L}^H)^T = \mathbf{L}^*(\mathbf{L}^*)^H$ , and finally scale the normalized ZF precoding vector  $\frac{\tilde{\mathbf{x}}}{\|\tilde{\mathbf{x}}\|_2}$  to reach the power constraint. Similarly, under decentralized scenarios (PD or FD architecture), for example, with an integrated pipeline of PD-ZF detection and PD-ZF precoding, we can reuse such explicit or implicit inversion results to realize further complexity reduction compared to the integrated pipeline of PD-MMSE detection and PD-WF precoding, at the cost of BER performance degradation.

As an example, Fig. 5(a) and 5(b) compare the uncoded BER performance of PD-MMSE vs. PD-ZF for uplink detection, and PD-WF vs. PD-ZF for downlink precoding, respectively, in a single-carrier system. We see that PD-ZF methods entail BER performance loss compared to PD-MMSE and PD-WF, especially under realistic Quadriga freespace channel [18]. Fig. 5(c) compares the timing complexity of different pipelines of uplink detection and downlink precoding at different  $N_{\text{coh}}$ . Using the total complexity of individually computed explicit PD-MMSE detection and PD-WF precoding as the baseline, we show that the integration of PD-MMSE and PD-WF by reusing  $\mathbf{G}^{\text{ul}}$  effectively reduces complexity, and the integration of PD-ZF detection and PD-ZF precoding achieves further complexity reduction as expected, especially when incorporated with implicit methods. When  $N_{\text{coh}}$  increases, the difference among those complexity curves decreases, indicating that the channel coherence plays a more important role on complexity reduction than channel reciprocity at a large  $N_{\text{coh}}$ , while at a small  $N_{\text{coh}}$ , exploiting channel reciprocity is more critical.

#### IV. DATA PRECISION TRADE-OFFS

Data precision is another factor in the design space. Given a decentralized architecture and algorithm, reducing the data precision can improve efficiency on modern computing fabrics due to fewer compiled machine instructions and memory transactions, and smaller inter-cluster data transfer sizes. For example, when using 8-bit floating point (fp8), we can pack four fp8 values into a fp32 value and execute a *vectorized* computation instruction (such as vectorized addition, multiplication, etc) in a

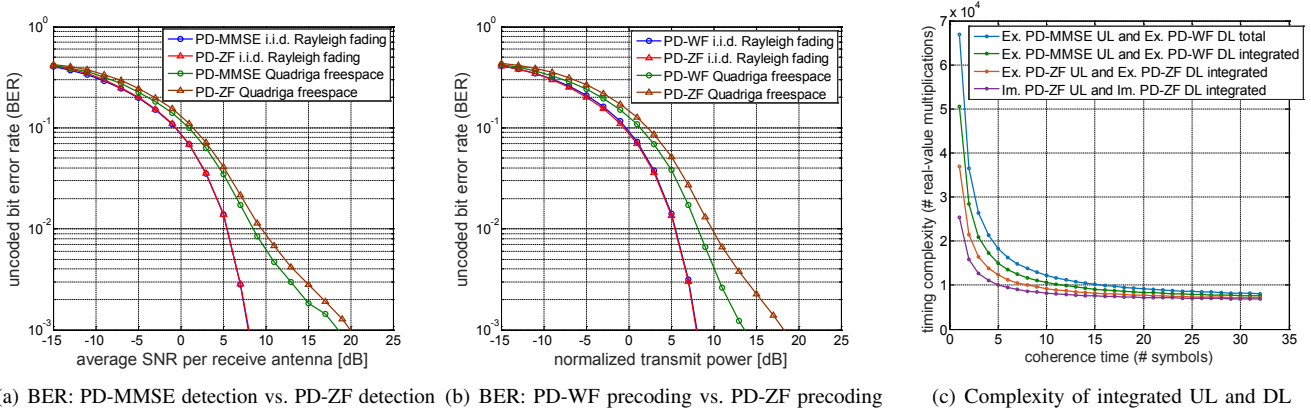


Fig. 5. UL and DL integration trade-offs on BER and complexity for a system with  $C = 4, U = 16, B_c = 32$ . (a) and (b) show the BER performance comparison of PD-MMSE vs. PD-ZF in uplink, and PD-WF vs. PD-ZF in downlink, respectively, at 16QAM modulation. PD-ZF sacrifices the BER performance, especially under a more realistic Quadriga channel, while achieving more significant complexity reduction with UL and DL integration, as shown in (c). We use the *total* complexity of individually computed (without UL and DL integration) explicit PD-MMSE and PD-WF as the baseline (the blue curve) for comparison.

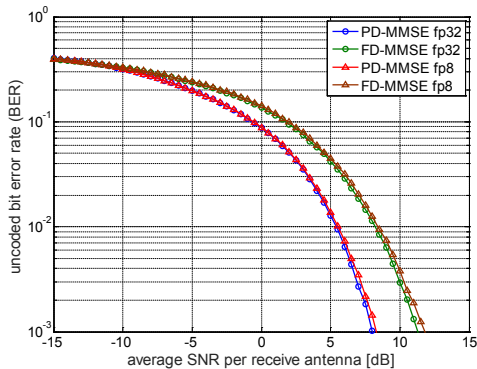


Fig. 6. BER comparison: fp32 vs. fp8 for PD-MMSE and FD-MMSE

single-instruction-multiple-data (SIMD) manner to process four fp8 values in parallel within the single instruction, contributing to  $4\times$  smaller number of instructions and memory transactions on the processor, and also reducing the inter-cluster bandwidth requirement by  $4\times$  compared to corresponding fp32 design. However, low precision sacrifices numerical accuracy and thus reduces the BER performance of corresponding detection or precoding algorithms. Fig. 6 compares the uncoded BER performance at fp32 vs. fp8 precision for PD-MMSE and FD-MMSE detectors under a realistic NLOS Quadriga channel [18] with system configurations of  $C = 4, U = 16, B_c = 32$  and 16-QAM modulation. Here, an fp8 value contains 1 sign bit, 2 mantissa bits and 5 exponent bits while an fp32 value contains 1 sign bit, 23 mantissa bits and 8 exponent bits. We see in Fig. 6 that fp8 precision only entails a small BER performance loss compared to fp32 precision. In practice, the selection of data precision depends on the trade-off of BER performance vs. efficiency given certain system and environment configurations.

## V. CONCLUSIONS

We have discussed the design trade-offs across architecture, algorithm, and data precision levels for decentralized

baseband processing (DBP) in massive MU-MIMO systems, and proposed a practical design flow that jointly considers critical metrics for DBP including computational complexity, data transfer sizes, and error-rate performance. As summarized in Fig. 7, given certain system configurations and channel conditions, one should first select the PD or FD architecture by trading off BER vs. data transfer size, and then decide on the detection and precoding algorithms with the selected architecture according to the BER vs. complexity trade-off. Finally, one can use the lower numerical precision if higher efficiency is more important than BER performance. To realize minimal computational complexity and data transfer size at no or little loss of BER, we have provided insights on taking advantage of both channel reciprocity and channel coherence properties by reusing intermediate results. In the future, we expect to build reconfigurable massive MU-MIMO software-defined radios based on programmable computing fabrics, such as GPUs or FPGAs, in order to dynamically adapt to time-varying system parameters and realize effective trade-offs.

## REFERENCES

- [1] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What Will 5G Be?," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1065–1082, Jun. 2014.
- [2] J. Hoydis, S. ten Brink, and M. Debbah, "Massive MIMO in the UL/DL of Cellular Networks: How Many Antennas Do We Need?," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 2, pp. 160–171, Feb. 2013.
- [3] L. Lu, G. Y. Li, A. L. Swindlehurst, A. Ashikhmin, and R. Zhang, "An Overview of Massive MIMO: Benefits and Challenges," *IEEE J. Sel. Topics in Sig. Proc.*, vol. 8, no. 5, pp. 742–758, Oct. 2014.
- [4] K. Li, R. R. Sharan, Y. Chen, T. Goldstein, J. R. Cavallaro, and C. Studer, "Decentralized Baseband Processing for Massive MU-MIMO Systems," *IEEE J. Emerg. and Sel. Topics in Circ. and Sys.*, vol. 7, no. 4, pp. 491–507, Dec 2017.
- [5] A. Puglielli, N. Narevsky, P. Lu, T. Courtade, G. Wright, B. Nikolic, and E. Alon, "A scalable massive MIMO array architecture based on common modules," in *IEEE Intl. Conf. Commun. Workshop*. IEEE, Jun. 2015, pp. 1310–1315.
- [6] S. Jacobsson, G. Durisi, M. Coldrey, T. Goldstein, and C. Studer, "Quantized Precoding for Massive MU-MIMO," *IEEE Trans. on Comm.*, vol. 65, no. 11, pp. 4670–4684, Nov. 2017.

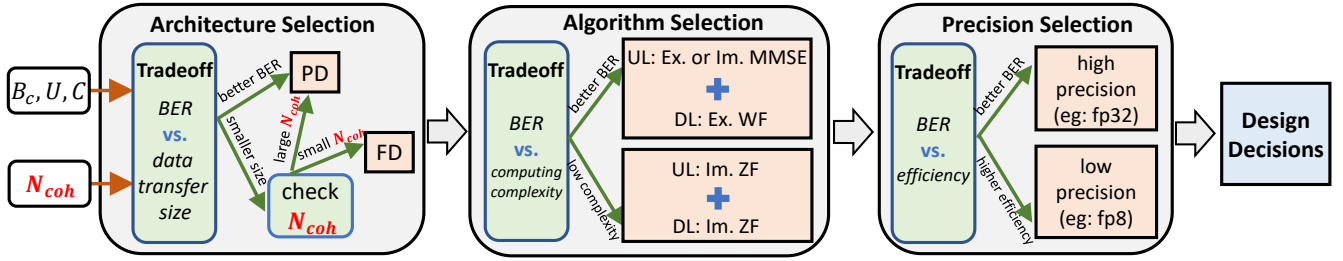


Fig. 7. Practical design decision flow: DBP architecture, algorithm and precision trade-off and selection.

- [7] E. Bertilsson, O. Gustafsson, and E. G. Larsson, "A Distributed Processing Architecture for Modular and Scalable Massive MIMO Base Stations," *arXiv preprint: 1801.07967*, Jan. 2018.
- [8] <http://www.cpri.info>, *Common public radio interface*.
- [9] K. Li, C. Jeon, J. R. Cavallaro, and C. Studer, "Feedforward Architectures for Decentralized Precoding in Massive MU-MIMO Systems," in *Asilomar Conf. on Sig., Sys., Comp.*, Oct. 2018, pp. 1659–1665.
- [10] Q. Yang, X. Li, H. Yao, J. Fang, K. Tan, W. Hu, J. Zhang, and Y. Zhang, "BigStation: Enabling Scalable Real-time Signal Processing in Large MU-MIMO Systems," in *ACM SIGCOMM*, Oct. 2013, pp. 399–410.
- [11] J. Vieira, S. Malkowsky, K. Nieman, Z. Miers, N. Kundargi, L. Liu, I. Wong, V. Öwall, O. Edfors, and F. Tufvesson, "A flexible 100-antenna testbed for Massive MIMO," in *IEEE Globecom*, Dec. 2014, pp. 287–293.
- [12] J. Sanchez, F. Rusek, O. Edfors, M. Sarajlic, and L. Liu, "Decentralized Massive MIMO Processing Exploring Daisy-chain Architecture and Recursive Algorithms," *arXiv preprint: 1905.03160*, May 2019.
- [13] K. Li, Y. Chen, R. Sharan, T. Goldstein, J. R. Cavallaro, and C. Studer, "Decentralized data detection for massive MU-MIMO on a Xeon Phi cluster," in *Asilomar Conf. Sig. Sys. Comp.*, Nov. 2016, pp. 468–472.
- [14] K. Li, R. Skaran, Y. Chen, J. R. Cavallaro, T. Goldstein, and C. Studer, "Decentralized beamforming for massive MU-MIMO on a GPU cluster," in *IEEE GlobalSIP*, Dec. 2016, pp. 590–594.
- [15] C. Jeon, K. Li, J. R. Cavallaro, and C. Studer, "On the achievable rates of decentralized equalization in massive MU-MIMO systems," in *IEEE Int. Symp. on Info. Theory*, Jun. 2017, pp. 1102–1106.
- [16] C. Jeon, K. Li, J. R. Cavallaro, and C. Studer, "Decentralized Equalization with Feedforward Architectures for Massive MU-MIMO," *IEEE Trans. on Sig. Proc.*, vol. 67, no. 17, pp. 4418–4432, Sep. 2019.
- [17] K. Li, C. Jeon, J. R. Cavallaro, and C. Studer, "Decentralized Equalization for Massive MU-MIMO on FPGA," in *Asilomar Conf. on Sig., Sys. and Comp.*, Nov. 2017.
- [18] <https://quadrige-channel-model.de>, *QuaDRiGa channel model*.
- [19] M. Wu, B. Yin, K. Li, C. Dick, J. R. Cavallaro, and C. Studer, "Implicit vs. Explicit Approximate Matrix Inversion for Wideband Massive MU-MIMO Data Detection," *J. of Sig. Proc. Sys.*, pp. 1–18, Dec. 2017.
- [20] K. Li, O. Castañeda, C. Jeon, J. R. Cavallaro, and C. Studer, "Decentralized Coordinate-Descent Data Detection and Precoding for Massive MU-MIMO," in *IEEE Int. Symp. on Circ. and Sys.*, May 2019, pp. 1–5.