Turbo-HB: A Sub-millisecond Hybrid Beamforming Design for 5G mmWave Systems

Yongce Chen, *Member, IEEE,* Yan Huang, *Member, IEEE,* Chengzhang Li, *Student Member, IEEE,* Y. Thomas Hou, *Fellow, IEEE,* and Wenjing Lou, *Fellow, IEEE*

Abstract—Hybrid beamforming (HB) architecture has been widely considered for 5G mmWave systems. It reduces hardware complexity by allowing the number of RF chains to be far fewer than the number of antennas. A major practical challenge for HB is to obtain a beamforming solution in real-time. In 5G NR, new frame structures with short TTIs are employed to support mmWave communications. Under such frame structures, it is necessary to obtain a beamforming solution with a time resolution varying from 1 ms to 125 μs – an extremely stringent time requirement considering the complexity involved in HB. In this paper, we present the design and implementation of *Turbo-HB* – a novel beamforming design under the HB architecture that is capable of offering the beamforming matrices in less than 500 μs. The key ideas of Turbo-HB include: (i) reducing the complexity of computation-intensive SVD operations by exploiting channel sparsity at mmWave frequencies, and (ii) achieving large-scale parallel computation with minimal memory access. We implement Turbo-HB on an off-the-shelf Nvidia GPU and conduct extensive experiments. Our experimental results demonstrate that Turbo-HB can obtain a beamforming solution in 500 μs for up to 100 RBs and 10 MU-MIMO users on each RB while offering competitive throughput performance compared to state-of-the-art (non-real-time) algorithms.

Index Terms—mmWave, hybrid beamforming, MIMO, real-time, GPU

1 Introduction

Communication over mmWave frequencies is defining a new era of wireless communication, including the most recent cellular systems such as 5G NR [2], [3]. At mmWave frequencies, a base station (BS) typically needs to employ hundreds or more antennas to overcome the large path-loss fading. However, it is difficult to apply a dedicated RF chain for *each* antenna as traditional MIMO under 6 GHz, due to hardware complexity and energy consumption issues [3], [4], [5], [6], [7], [8]. To address this problem, the so-called “hybrid architecture” was proposed. As illustrated in Fig. 1, the hybrid architecture uses a much fewer number of shared RF chains to support a large number of antennas. This innovative design has attracted attention from both the academic community [9], [10], [11], [12], [13], [14], [15], [16], [17], [18] and the industry [19], [20], [21], [22], [23], [24], [25].

Although attractive, hybrid architecture faces a critical challenge. Specifically, it must be able to offer a beamforming solution in real-time to be practical. By real-time, we mean that a beamforming solution must be found within half of the channel coherence time.[1] At mmWave frequencies, this channel coherence time is extremely short, due to the severe Doppler effect. In 5G NR, new frame structures with shorter TTIs (compared to 4G LTE) are designed to support communications over short channel coherence time [26]. Specifically, under 5G NR numerology 0, a TTI is 1 ms, while the TTIs for numerologies 1, 2 and 3 are 500 μs, 250 μs and 125 μs, respectively. The shorter TTIs allow 5G to cope with extremely short coherence time at high frequencies and to support ultra-low latency applications. Therefore, for a hybrid architecture to work under 5G NR, an HB solution must be found within each TTI (corresponding to the applied numerology) to be useful. Further, a beamforming design must consider a large number of resource blocks (RBs), with each RB supporting multiple active users (MU-MIMO).

Although there exist a number of research works in the literature on HB design, few can meet the real-time requirement with high throughput performance. For instance, physical (PHY) layer research in this area attempted to jointly optimize analog and digital beamforming [9],

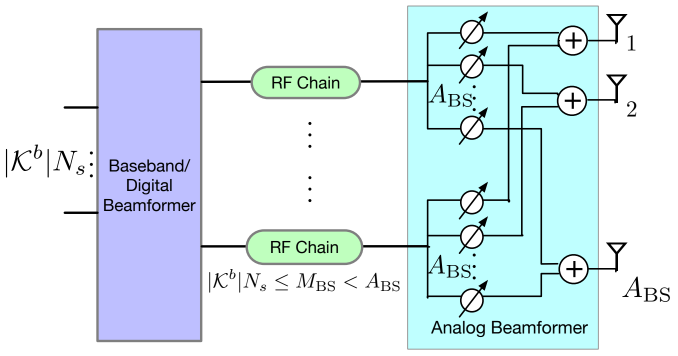


Fig. 1: An HB architecture (BS side).

- *This research was supported in part by NSF under grant 1617634, Virginia Commonwealth Cyber Initiative (CCI), and Virginia Tech Institute for Critical Technology and Applied Science (ICTAS).*
- *An abridged version of this paper appeared in the Proc. of IEEE INFOCOM 2020 [1].*

- *Y. Chen and Y. Huang are with NVIDIA Corporation, Santa Clara, CA 95054, USA. E-mail: {yc.chen, huangyan}@vt.edu*
- *C. Li, Y.T. Hou and W. Lou are with Virginia Tech, Blacksburg, VA 24061 USA. E-mail: {licz17, thou, wjlou}@vt.edu.*
- *Corresponding author: Y.T. Hou*

1. For efficiency, we break up the channel coherence time into two halves. Within each half, we transmit data based on beamforming matrices that are computed in the previous half and compute the beamforming matrices for the next half.

[10], [11], [12]. Unfortunately, the iterative nature of these algorithms makes them difficult to be implemented in real-time. In addition, a joint design requires explicit antenna-to-antenna channel estimation and feedback, which involves a prohibitively high complexity and a large amount of CSI that is too difficult to obtain in practice [13].

To avoid the issues associated with a joint design, a new and practical direction for HB is to follow a sequential design [14], [15], [16], [17], [18]. Here, an analog beamforming is optimized first and then used as the input to optimize the digital beamforming. For analog beamforming, there have been successful designs and system demonstrations in the literature, which are based on beam sweeping/discovering techniques without explicit channel CSI [16], [17], [18]. After analog beamforming is applied to both the BS and a user's side, the effective channels seen at the baseband can be obtained through conventional channel estimation approaches.

However, how to properly design digital beamforming in a sequential design remains a challenge. Most existing works simply applied traditional beamforming methods such as ZF, MMSE and Block Diagonalization (BD) [27] as the digital beamformers [14], [15], [16], [17]. Although simple, ZF and MMSE typically experience inferior throughput performance for MU-MIMO and mmWave systems, particularly under ill-conditioned channels [16], [28], [29]. Although BD beamforming and its variants are shown to improve ZF/MMSE with a much better throughput performance [27], [30], it requires many high-dimensional matrix SVD operations, which are of high complexity and require significant computation time.

As expected, finding a beamforming scheme that can meet both the real-time requirement and high throughput performance is not trivial. But recent advances in parallel architectures (based on the many-core technology) have shed new light on this problem. In particular, the general-purpose GPU-based platform (e.g., those from Nvidia) is particularly promising. Its dedicated single-instruction-multiple-data (SIMD) architecture can solve a massive number of structurally-identical problems at an extremely fast speed. It also comes with highly programmable tools such as CUDA, making the real-time implementation feasible and flexible to many developers. A GPU-based parallel computing platform now offers a new possibility to tackle many hard problems whose real-time solutions are once considered elusive [32].

In this paper, we present *Turbo-HB*,[2] a GPU-based novel design and implementation to achieve ultra-fast digital beamforming. The key ideas of Turbo-HB are twofold. First, we identify the bottleneck of computation time for BD-type beamforming, which attributes to high-dimensional SVD operations. Turbo-HB cuts down this computational complexity by utilizing a randomized SVD technique. Second, Turbo-HB accelerates the overall computation time through large-scale parallel computation on a commercial off-the-shelf (COTS) GPU platform. It incorporates a large number of matrix transformations in parallel and special engineering efforts such as minimized memory access. The main contributions of this paper are summarized as follows:

2. By "Turbo," we mean fast and efficient.

- This paper presents Turbo-HB, the first successful HB design that can meet the sub-ms real-time requirement. This design considers a large number of RBs with MU-MIMO capability, which can be applied to 5G cellular systems. Our design only relies on a COTS GPU platform and does not require any customized hardware.
- Turbo-HB relieves the computational burden of SVD significantly by leveraging the sparsity at mmWave channels. Specifically, Turbo-HB is able to identify a small number of the most significant directions on a mmWave channel by exploiting the randomized SVD technique. By limiting operations only to the key information of interests, high-dimensional SVD operations are transformed into lightweight lower-rank matrix operations. By judiciously choosing a proper target rank for lower-rank approximation, our design can reduce the computation time dramatically.
- Turbo-HB is capable of parallelizing the MU-MIMO beamforming for a large number of RBs and users. First, the MU-MIMO beamforming is transformed into a set of parallel single-user MIMO (SU-MIMO) beamforming. Second, with customized nullspace calculation based on Given's rotation method, Turbo-HB accelerates computation and fully utilizes GPU's processing cores. Third, by employing batched matrix operation with proper indexing method and utilizing shared memory, Turbo-HB achieves large-scale parallel matrix operations.
- We implement Turbo-HB on Nvidia DGX Station using the CUDA programming platform. Extensive experiments are performed to examine both the timing performance and throughput performance. Experimental results show that Turbo-HB is able to obtain the beamforming matrices far less than 1 ms for all tested cases. Specifically, Turbo-HB can meet the 125μs, 250 μs, and 500 μs timing requirement for 100 RBs with up to 4, 8, and 10 MU-MIMO users on each RB, respectively. Turbo-HB can also offer higher throughput performance for most cases compared to the state-of-the-art (non-real-time) algorithms.

Notations: We use upper and lower-case bold letters to denote the matrices and vectors, respectively. The upper-case calligraphic letters denote the sets. The notation $\mathcal{CN}(\cdot,\cdot)$ denotes the circularly symmetric Gaussian distribution. The notations $(\cdot)^{-1}$ and $(\cdot)^{\dagger}$ denote the inverse and the conjugate transpose of a matrix, respectively. The notations $|\cdot|$ and $||\cdot||_F$ denote the determinant and the Frobenius norm of a matrix, respectively. Table 1 lists the key notation symbols and Table 2 lists the key acronyms used in this paper.

2 Related Work

Hybrid beamforming design is an active research area and has attracted lots of research efforts. However, most existing research has been largely limited to asymptotic complexity analysis (i.e., in $O(\cdot)$). Although such complexity analysis is of interest from a theoretical perspective, it does not give any indication on how much actual time ("real-time") is needed when it is implemented on a given hardware

TABLE 1: Notations

Symbol	Definition
A_{BS}	Number of antennas at BS
A_{U}	Number of antennas at user
$\mathcal{B}$	A set of RBs to be allocated in a time slot
$\mathbf{F}_{\text{BB}}$	Baseband precoder at BS
$\mathbf{F}_{\text{BB},k}$	Baseband precoder at BS for user k
$\mathbf{F}_{\text{RF}}$	Analog precoder at BS
$\mathcal{K}$	A set of users
$\mathcal{K}^b$	A subset of users using RB b
M_{BS}	Number of RF chains at BS
M_{U}	Number of RF chains at user
L_{cl}	Number of clusters on a link
N_{ray}	Number of rays on a link
N_s	Number of data streams on a link
$\mathbf{W}_{\text{BB},k}$	Baseband combiner at user k
$\mathbf{W}_{\text{RF},k}$	Analog combiner at user k

TABLE 2: Acronyms

Acronym	Definition
BD	Block Diagonalization
COTS	Commercial Off-The-Shelf
CSI	Channel State Information
ED	Eigendecomposition
FPGA	Field-Programmable Gate Array
G2H	GPU to Host
GPU	Graphics Processing Unit
H2G	Host to GPU
HB	Hybrid Beamforming
LDPC	Low-Density Parity-Check
MMSE	Minimum Mean Square Error
MU-MIMO	Multi-User Multiple-Input Multiple-Output
RB	Resource Block
RF	Radio Frequency
SIMD	Single-Instruction-Multiple-Data
SINR	Signal-to-Interference-plus-Noise Ratio
SM	Streaming Multiprocessor
SNR	Signal-to-Noise Ratio
SU-MIMO	Single-User Multiple-Input Multiple-Output
SVD	Singular Value Decomposition
TTI	Transmission Time Interval
ZF	Zero Forcing

platform. On the other hand, for a real-world 5G system, the ultimate benchmark is real-time performance, as there is a stringent timing requirement under its numerology.

In the literature, all kinds of HB designs involve some level of heuristics. One line of research is to jointly optimize analog and digital beamforming to offer a near-optimal solution (see, e.g., [9], [10], [11], [12]). A common feature of these designs is that their algorithms must run iteratively to update digital beamformers and analog beamformers. Due to a large number of iterations that are needed in these designs, none of them can offer real-time solutions under the 5G requirement (sub-ms).

On the other hand, sequential designs are proposed to reduce the complexity by decoupling the analog domain and digital domain (see, e.g., [14], [15], [16], [17], [18]). However, the mainstream of existing research works heavily relies on reducing the asymptotic complexity (in $O(\cdot)$) in their algorithms. Since the asymptotic complexity analysis of an algorithm is only concerned with when the input size n is sufficiently large (approaches to infinity), it does not reflect how much actual time it will need when input data is finite, as in 5G. As a result, these sequential algorithms do not meet the sub-ms timing requirement when they are tested by a real timer. In addition, algorithms designed with extremely simple digital beamforming such as ZF/MMSE may also suffer from considerable throughput loss at mmWave frequencies.

Recently, there has been a number of successful research works applying parallel techniques to wireless networking and signal processing problems. Some representative works include [31], [58], [59], [60], [61], [62]. Specifically, the authors in [31], [58], [59] implemented real-time designs to address scheduling problems in 4G/5G networks. In [60], the authors proposed MIMO detection algorithms that utilize parallelism to achieve high-performance detection. The studies in [61], [62] applied parallel processing to accelerate LDPC decoding. These approaches were demonstrated on a GPU or FPGA platform. Among them, the designs based on general-purpose GPU platform (e.g., those from [31], [58], [61]) provided a high level of parallelism and flexibility, thanks to GPU's large-scale SIMD architecture and highly programmable tools such as CUDA. However, these algorithms are designed to address scheduling or decoding problems, and their approaches cannot be applied in solving a complex beamforming problem under hybrid architecture, which is the focus of this paper.

3 System Model

We consider a cellular communication scenario where a BS serves a set $\mathcal{K}$ of users, as illustrated in Fig. 2. The BS is equipped with A_{BS} antennas and M_{BS} RF chains. Under HB architecture, $M_{\text{BS}} < A_{\text{BS}}$. Each user is equipped with A_{U} antennas and M_{U} RF chains, and $M_{\text{U}} < A_{\text{U}}$. Since the mathematical structure for uplink (UL) and downlink (DL) is symmetric, it is sufficient to study one of them. We focus on DL in this paper.

Considering a typical cellular system (e.g., 4G LTE and 5G NR), we study time-slotted scheduling over a wide bandwidth. Within each time slot, there is a set $\mathcal{B}$ of RBs over the DL bandwidth. For each RB $b \in \mathcal{B}$, a subset of users $\mathcal{K}^b \subseteq \mathcal{K}$ is selected for MU-MIMO transmission. In practice, the beamforming problem and user selection problem are often decoupled to reduce complexity [33], [34], [35], [36], [37]. In this paper, we focus on the beamforming problem and assume the users on each RB are given *a priori*. For the ease of notation, suppose the BS sends N_s data streams to each user.[3] At the user side, since the number of received data streams cannot exceed the number of its RF chains, we have $N_s \leq M_{\text{U}}$. Likewise, at the BS we have $|\mathcal{K}^b| N_s \leq M_{\text{BS}}$.

Under the HB architecture, beamforming is performed in both digital and analog domains, as shown in Fig. 1.

3. With additional notation, our results can be extended to the case where the BS sends a different number of data streams to different users.

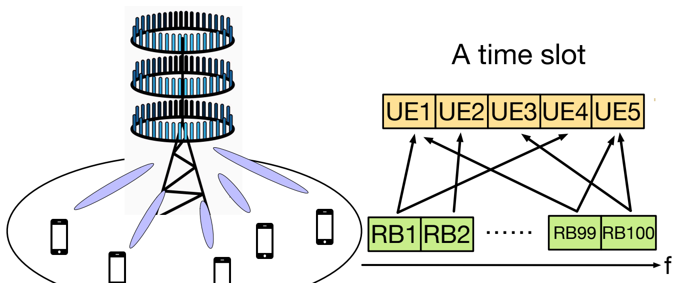


Fig. 2: A cellular system consisting a large number of RBs (with MU-MIMO capability).

At the BS side, the transmitted signal is first processed in the digital domain by an $M_{\text{BS}} \times |\mathcal{K}^b| N_s$ baseband precoder $\mathbf{F}_{\text{BB}}$. Subsequently, an $A_{\text{BS}} \times M_{\text{BS}}$ analog precoder $\mathbf{F}_{\text{RF}}$ (also known as RF precoder) based on analog circuitry (phase shifters) is applied in the analog domain. Since complex matrix $\mathbf{F}_{\text{RF}}$ is implemented with analog phase shifters, each element in the matrix has the same amplitude and differs in its phase, i.e, $|(\mathbf{F}_{\text{RF}})_{i,j}| = \frac{1}{\sqrt{A_{\text{BS}}}}$, where $(\cdot)_{i,j}$ denotes the (i,j)-th element of matrix $(\cdot)$. In addition, to meet the total power constraint at the BS, $\mathbf{F}_{\text{BB}}$ and $\mathbf{F}_{\text{RF}}$ must satisfy $||\mathbf{F}_{\text{RF}}\mathbf{F}_{\text{BB}}||_F^2 \leq P_{\text{T}}$, where P_{T} is the total power at the BS and $||\cdot||_F$ denotes the Frobenius norm.

For wireless channels, let $\mathbf{H}_k^b \in \mathbb{C}^{A_{\text{U}} \times A_{\text{BS}}}$ denote the channel matrix for user $k \in \mathcal{K}$ on RB $b \in \mathcal{B}$, and $\boldsymbol{n}_k^b$ is the $A_{\text{U}} \times 1$ vector of i.i.d $\mathcal{CN}(0, \sigma^2)$ additive complex Gaussian noise. We apply the widely used cluster-based mmWave channel model [10], and do not consider the presence of blockage in this paper. Let $\mathbf{F}_{\text{BB}}^b$ and $\mathbf{F}_{\text{RF}}^b$ denote the baseband precoder and analog precoder for RB b, respectively. Then the received signal of user k on RB b is given by

$$\boldsymbol{y}_k^b = \mathbf{H}_k^b \mathbf{F}_{\text{RF}}^b \mathbf{F}_{\text{BB}}^b \boldsymbol{s}^b + \boldsymbol{n}_k^b, \quad (k \in \mathcal{K}^b, b \in \mathcal{B}) \tag{1}$$

where $\boldsymbol{s}^b$ is the signal vector.

At the user side, a symmetric HB structure is employed except with a fewer number of antennas A_{U} and a fewer number of RF chains M_{U}. The received signal is first processed by an $A_{\text{U}} \times M_{\text{U}}$ analog combiner $\mathbf{W}_{\text{RF},k}$ (subject to $|(\mathbf{W}_{\text{RF},k})_{i,j}| = \frac{1}{\sqrt{A_{\text{U}}}}$) in analog domain. Then an $M_{\text{U}} \times N_s$ baseband combiner $\mathbf{W}_{\text{BB},k}$ is applied.

Denote $\widehat{\mathbf{H}}_k^b$ as the effective channel seen at the baseband, i.e., $\widehat{\mathbf{H}}_k^b = \mathbf{W}_{\text{RF},k}^{b\dagger} \mathbf{H}_k^b \mathbf{F}_{\text{RF}}^b$. Denote $\mathbf{F}_{\text{BB},k}^b$ as a sub-matrix of $\mathbf{F}_{\text{BB}}^b = [\mathbf{F}_{\text{BB},1}^b \cdots \mathbf{F}_{\text{BB},k}^b \cdots \mathbf{F}_{\text{BB},|\mathcal{K}^b|}^b]$, where $\mathbf{F}_{\text{BB},k}^b$ consists of N_s columns and corresponds to the baseband signal $\boldsymbol{s}_k^b$ for user k. Then at user k and on RB b, we have the following signal:

$$\begin{aligned}\tilde{\boldsymbol{y}}_k^b = &\mathbf{W}_{\text{BB},k}^{b\dagger} \widehat{\mathbf{H}}_k^b \mathbf{F}_{\text{BB},k}^b \boldsymbol{s}_k^b + \sum_{i \in \mathcal{K}^b}^{i \neq k} \mathbf{W}_{\text{BB},k}^{b\dagger} \widehat{\mathbf{H}}_k^b \mathbf{F}_{\text{BB},i}^b \boldsymbol{s}_i^b \\ &+ \mathbf{W}_{\text{BB},k}^{b\dagger} \mathbf{W}_{\text{RF},k}^{b\dagger} \boldsymbol{n}_k^b, \qquad (k \in \mathcal{K}^b, b \in \mathcal{B})\end{aligned}$$

where $(\cdot)^\dagger$ denotes the conjugate transpose of a matrix.

Therefore, the network throughput in b/s/Hz is

$$\begin{aligned}C = \sum_{b \in \mathcal{B}} \sum_{k \in \mathcal{K}^b} \log \Big(\Big| \mathbf{I}_{N_s} + \\ (\mathbf{Q}_k^b)^{-1} \mathbf{W}_{\text{BB},k}^{b\dagger} \widehat{\mathbf{H}}_k^b \mathbf{F}_{\text{BB},k}^b \ \mathbf{F}_{\text{BB},k}^{b\dagger} \widehat{\mathbf{H}}_k^{b\dagger} \mathbf{W}_{\text{BB},k}^b \Big| \Big),\end{aligned} \tag{2}$$

where $\mathbf{Q}_k^b$ is the covariance matrix of both interference and noise, which is given by

$$\begin{aligned}\mathbf{Q}_k^b = &\sum_{i \in \mathcal{K}^b}^{i \neq k} \mathbf{W}_{\text{BB},k}^{b\dagger} \widehat{\mathbf{H}}_k^b \mathbf{F}_{\text{BB},i}^b \mathbf{F}_{\text{BB},i}^{b\dagger} \widehat{\mathbf{H}}_k^{b\dagger} \mathbf{W}_{\text{BB},k}^b \\ &+ \sigma^2 \mathbf{W}_{\text{BB},k}^{b\dagger} \mathbf{W}_{\text{RF},k}^{b\dagger} \mathbf{W}_{\text{RF},k}^b \mathbf{W}_{\text{BB},k}^b.\end{aligned}$$

Then the throughput optimization problem under the HB architecture can be stated as follows:

OPT-HB

$$\begin{aligned}\max \ & C\left(\mathbf{F}_{\text{RF}}^b, \mathbf{F}_{\text{BB}}^b, \mathbf{W}_{\text{RF},k}^b, \mathbf{W}_{\text{BB},k}^b\right) \\ \text{s.t.} \ & \text{Power constraint: } ||\mathbf{F}_{\text{RF}}^b \mathbf{F}_{\text{BB}}^b||_F^2 \leq P_{\text{T}}; \\ & \text{Constant modulus constraints:} \\ & \quad |(\mathbf{F}_{\text{RF}}^b)_{i,j}| = \frac{1}{\sqrt{A_{\text{BS}}}}, \ |(\mathbf{W}_{\text{RF},k}^b)_{m,n}| = \frac{1}{\sqrt{A_{\text{U}}}}; \\ & \text{Index range: } b \in \mathcal{B}, \ k \in \mathcal{K}, \\ & \quad i \in \{1, 2, \cdots, A_{\text{BS}}\}, \ j \in \{1, 2, \cdots, M_{\text{BS}}\}, \\ & \quad m \in \{1, 2, \cdots, A_{\text{U}}\}, \ n \in \{1, 2, \cdots, M_{\text{U}}\}.\end{aligned}$$

In problem OPT-HB, the variables are digital and analog beamformers $\mathbf{F}_{\text{RF}}^b, \mathbf{F}_{\text{BB}}^b, \mathbf{W}_{\text{RF},k}^b$ and $\mathbf{W}_{\text{BB},k}^b$, while $P_{\text{T}}, A_{\text{BS}}, A_{\text{U}}, M_{\text{BS}}, M_{\text{U}}$ are constants and $\mathcal{B}$ and $\mathcal{K}^b$ are given sets.

Ideally, a joint optimization of all digital and analog beamformers is required to find a global optimal solution. However, several practical issues make such a joint design infeasible. For example, the amount of CSI required is prohibitively large; it is unclear how to estimate the antenna-to-antenna channel $\mathbf{H}_k^b$ through the lens of the RF precoding and combining [13]. A new and practical direction to address HB optimization is to follow a sequential design. Under this approach, analog domain is optimized first and then used as input to optimize the digital design [14], [15], [16], [17], [18]. It has been shown that such a sequential approach can offer a competitive performance (compared to those heuristics attempting to solve joint optimization [10], [12], [14], [39]).

Even with a sequential method, for MU-MIMO systems, it would still require enormous computational efforts to find a local optimum [40], due to the high complexity of high-dimensional matrix operations (in addition to non-convex programming). We discuss this problem in detail in the following section.

4 Real-Time Requirement

In 5G NR, the frame structure is designed to be scalable to accommodate diverse services and channel conditions. Under 5G frame structures, a beamforming solution (for all users on all RBs) must be obtained within 1 ms (numerology 0), 500 μs (numerology 1), 250 μs (numerology 2), or 125 μs (numerology 3). A shorter TTI can support applications with a shorter coherence time and a more stringent latency requirement.

Note that under the HB architecture, analog beamforming is meant to overcome path-loss fading by leveraging a large number of antennas [14], [18]. This part is done on a much larger time scale. In contrast, digital beamforming can optimize capacity by managing interference among data streams, which heavily depends on fast fading. This part has a much more stringent timing requirement. Therefore, under a sequential design, the stringent sub-ms real-time requirement mainly comes from digital beamforming.

Technical Challenge Digital beamforming for MU-MIMO involves complex operations of matrices with a large number of elements. Traditional techniques such as ZF and MMSE typically experience inferior throughput performance for MU-MIMO and mmWave systems, particularly under ill-conditioned channels [16], [28], [29]. On the other hand, BD-type beamforming is shown to achieve much better throughput performance compared to ZF/MMSE [27]. But BD involves high-dimensional matrix SVD operations, whose computational complexity makes BD unsuitable for practical use.

Objective The objective of this paper is to determine digital beamformers ($\mathbf{F}^b_{\mathrm{BB},k}$ and $\mathbf{W}^b_{\mathrm{BB},k}$) in real-time. Specifically, we want to develop a design that can meet the stringent sub-ms timing requirement while offering comparable (or better) throughput performance than state-of-the-art approaches.

5 A Novel Design for Real-time Beamforming

5.1 Main Ideas

Our main ideas consist of two parts.

Low-complexity SVD with high throughput First, we show the high computation time for BD-type beamforming is attributed to the high-dimensional SVD operations. Then we propose to reduce this complexity by identifying only a small number for the most significant dimensions, leveraging the sparsity of mmWave channels. Specifically, for a $(|\mathcal{K}^b|-1)M_{\mathrm{U}} \times M_{\mathrm{BS}}$ matrix (for BD beamforming), a standard SVD algorithm takes $O\left(\left[(|\mathcal{K}^b|-1)M_{\mathrm{U}}\right]^2 M_{\mathrm{BS}}\right)$ floating-point operations (flops) [47], [48]. Thus, applying BD beamforming for $|\mathcal{B}|$ RBs and $|\mathcal{K}^b|$ users at each RB yields at least $O\left(|\mathcal{B}||\mathcal{K}^b|\cdot\left[(|\mathcal{K}^b|-1)M_{\mathrm{U}}\right]^2 \cdot M_{\mathrm{BS}}\right)$ flops. To reduce this high complexity, we propose to utilize the randomized SVD [47] to cut down the complexity to $O\left(|\mathcal{B}||\mathcal{K}^b|\cdot r^2\cdot\left[(|\mathcal{K}^b|-1)M_{\mathrm{U}}+M_{\mathrm{BS}}\right]\right)$, where r is much smaller than $(|\mathcal{K}^b|-1)M_{\mathrm{U}}$. In essence, the randomized SVD is a lower-rank SVD approximation method. The reason why it works extremely well here is because of the limited number of scatterers at mmWave frequencies and thus highly correlated channels. In addition, by limiting the operations to the key information of our interest and applying the parallelizable Given's rotation method, the lower-rank SVD can be done extremely fast in our implementation.

Interestingly, although Turbo-HB employs a lower-rank SVD approximation, it does not mean the throughput performance needs to deteriorate. Rather, Turbo-HB appears to offer higher throughput performance in most cases. The science behind this behavior is attributed to the following. First, since mmWave channels exhibit a high correlation property, a small set of singular vectors in the lower-rank SVD approximation is sufficient to capture the directions of the most significant signals or interferences. Second, an exact $(|\mathcal{K}^b|-1)M_{\mathrm{U}} \times M_{\mathrm{BS}}$ matrix SVD (as in standard BD) aims to cancel *all* inter-user interference exactly (regardless of how small it is). But canceling all inter-user interference requires to project users' signals onto mutually orthogonal subspaces. To achieve such orthogonality, the perceived strength of desired signals at a user is reduced in the process. Since throughput is a function of SINR, it does not help if the perceived strength of desired signals at a user is reduced (for perfect orthogonality). On the other hand, a lower-rank SVD approximation allows a certain level of overlapping subspace of different users (as only a small number of major signals preserve mutual orthogonality), which in return preserves greater desired signal strength. This offers us an opportunity to explore the promising beamforming space that is missed by the BD technique.

Fully functioning parallelism We argue that the asymptotic complexity analysis (i.e., those expressed in the big-O notation) does not directly translate into actual computation time as measured by a wall clock for our problem. The latter heavily depends on the underlying problem structure, actual input size, convergence speed, memory access time, among others. This motivates us to our second idea, which is to accelerate the overall computing process in real-time, rather than focusing on $O(\cdot)$ analysis. We propose to design a beamforming algorithm with parallelizable implementation, incorporating special engineering efforts such as minimizing memory access.

Specifically, the MU-MIMO beamforming is first transformed into a set of parallel SU-MIMO beamforming. Then a large number of matrix operations are executed through batch computing. To achieve batched matrix operations (for a large number of RBs and users), Turbo-HB generates a large number of threads that fully occupy a GPU's processing cores and thus reaps the full benefits of GPU's parallel processing capability. At each step throughout our implementation, we meticulously minimize memory accesses to reduce time. For example, batched matrix operations such as QR factorization and matrix multiplications are optimized with the use of fast on-chip shared memory. We carefully organize the storage of a large number of matrices with proper indexing. By managing consecutive GPU threads to read consecutive (and aligned) memory, multiple memory accesses can be combined into a single transaction. Further, Turbo-HB limits operations to the key information of our interests (e.g., certain singular vectors) and thus eliminates unnecessary calculations, parameter passing and memory access.

5.2 Design Details

The task of computing beamforming matrices can be split naturally into three computational stages. The first is to transform the MU-MIMO channel into a set of parallel SU-MIMO channels. The second is to apply the randomized SVD with low computation complexity to obtain certain singular vectors for beamforming. The third is to construct the final digital beamforming matrices based on obtained singular vectors. Specifically, the objective of each stage is described as follows.

- **Stage A:** Given the partial CSI $\mathbf{V}^b_k$ and $\mathbf{\Sigma}^b_k$ (from $\widehat{\mathbf{H}}^b_k = \mathbf{U}^b_k\mathbf{\Sigma}^b_k\mathbf{V}^{b\dagger}_k$) that are computed and fed back by each user, we construct matrices $\overline{\mathbf{H}}^b_k$ and $\widetilde{\mathbf{H}}^b_k$ such that $\overline{\mathbf{H}}^b_k$ and $\widetilde{\mathbf{H}}^b_k$ contain all the information that is needed to compute beamforming matrices $\mathbf{F}^b_{\mathrm{BB},k}$ corresponding to user k. After this stage, the MU-MIMO channel is transformed into a set of parallel SU-MIMO channels.

- **Stage B:** Given matrix $\widetilde{\mathbf{H}}^b_k$, we apply the randomized SVD technique for lower-rank matrix approximations (with lower computational complexity). Then we obtain $\widetilde{\mathbf{V}}^{b(-)}_k$, which contains the necessary singular vectors to cancel inter-user interference.
- **Stage C:** With matrices $\overline{\mathbf{H}}^b_k$ and $\widetilde{\mathbf{V}}^{b(-)}_k$, we construct the final digital beamforming matrices $\mathbf{F}^b_{\mathrm{BB},k}$.

In the rest of this section, we offer details of each stage.

Stage A. Each user k estimates the effective channel $\widehat{\mathbf{H}}^b_k$ and computes its SVD as $\widehat{\mathbf{H}}^b_k = \mathbf{U}^b_k \mathbf{\Sigma}^b_k \mathbf{V}^{b\dagger}_k$. User k uses the first N_s columns of $\mathbf{U}^b_k$ as its digital combiner, i.e., $\mathbf{W}^b_{\mathrm{BB},k}$ is set to the first N_s columns of $\mathbf{U}^b_k$. Then to help form digital precoder at the BS side, only partial CSI, i.e., $\mathbf{V}^b_k$ and $\mathbf{\Sigma}^b_k$, are required to feedback to the BS (note that $\mathbf{\Sigma}^b_k$ is diagonal and $\mathbf{V}^b_k$ is unitary and thus can be efficiently compressed [49]). Let

$$\overline{\mathbf{H}}^b_k = \mathbf{\Sigma}^b_k \mathbf{V}^b_k.$$

Then for our beamforming purpose, $\overline{\mathbf{H}}^b_k$ (an $M_{\mathrm{U}} \times M_{\mathrm{BS}}$ matrix) captures sufficient information of the intended channel from the BS to user k.

Denote $\widetilde{\mathbf{H}}^b_k$ as the concatenation of $\overline{\mathbf{H}}^b_k$'s of all users in $\mathcal{K}^b$ except intended user k, i.e., if $\mathcal{K}^b = \{k\} \bigcup \{1, \cdots, k-1, k+1, \cdots, |\mathcal{K}^b|\}$, then

$$\widetilde{\mathbf{H}}^b_k = \left[\overline{\mathbf{H}}^{b\dagger}_1 \cdots \overline{\mathbf{H}}^{b\dagger}_{k-1} \;\; \overline{\mathbf{H}}^{\dagger}_{k+1} \cdots \overline{\mathbf{H}}^{b\dagger}_{|\mathcal{K}^b|} \right]^\dagger$$

is a $(|\mathcal{K}^b| - 1)M_{\mathrm{U}} \times M_{\mathrm{BS}}$ matrix that captures information of interference channels corresponding to user k.

As $\overline{\mathbf{H}}^b_k$ and $\widetilde{\mathbf{H}}^b_k$ are sufficient to construct the beamforming matrices $\mathbf{F}^b_{\mathrm{BB},k}$ corresponding to user k, the MU-MIMO channel is transformed into a set of $|\mathcal{K}^b|$ parallel SU-MIMO channels on each RB. Consequently, the remaining Stage B and Stage C can be processed in $\sum_{b\in\mathcal{B}} |\mathcal{K}^b|$ parallel flows, each of which contributes to one beamforming matrix for one user per RB.

Stage B. To construct beamforming matrix $\mathbf{F}^b_{\mathrm{BB},k}$ corresponding to user k's signal, we need to make sure that by applying $\mathbf{F}^b_{\mathrm{BB},k}$ most (if not all) of the interference to user k can be canceled. This can be realized with the help of SVD of interference channel $\widetilde{\mathbf{H}}^b_k$. Let

$$\widetilde{\mathbf{H}}^b_k = \widetilde{\mathbf{U}}^b_k \begin{bmatrix} \widetilde{\mathbf{\Sigma}}^b_k & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} [\widetilde{\mathbf{V}}^{b(+)}_k \;\; \widetilde{\mathbf{V}}^{b(-)}_k]^\dagger, \tag{3}$$

where $\widetilde{\mathbf{V}}^{b(-)}_k$ is the last $(M_{\mathrm{BS}} - r)$ columns of the right singular matrix corresponding to the smallest $(M_{\mathrm{BS}} - r)$ singular values of $\widetilde{\mathbf{H}}^b_k$, $\widetilde{\mathbf{V}}^{b(+)}_k$ is the remaining r columns of the right singular matrix, and r is a constant.

Then, if the eigenvalues corresponding to $\widetilde{\mathbf{V}}^{b(-)}_k$ are close to zero, we have

$$\widetilde{\mathbf{H}}^b_k \widetilde{\mathbf{V}}^{b(-)}_k \approx \mathbf{0}, \quad (b \in \mathcal{B}, k \in \mathcal{K}^b).$$

It follows that

$$\widehat{\mathbf{H}}^b_j \widetilde{\mathbf{V}}^{b(-)}_k \overline{\mathbf{V}}^{b(+)}_k \approx \mathbf{0}, \;\; \text{for } j \neq k,$$

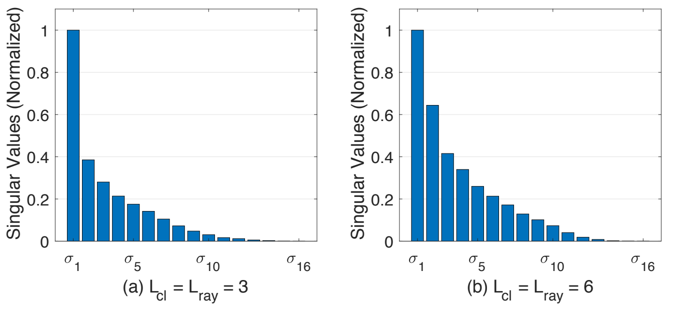


Fig. 3: Singular values of $\widetilde{\mathbf{H}}^b_k$ (averaged over 100 instances) under different number of scatterers based on mmWave channel modelling.

for any $\overline{\mathbf{V}}^{b(+)}_k$ (which is used to differentiate data streams within a user and will be determined later). Therefore, by constructing $\mathbf{F}^b_{\mathrm{BB},k}$ as

$$\mathbf{F}^b_{\mathrm{BB},k} = \widetilde{\mathbf{V}}^{b(-)}_k \overline{\mathbf{V}}^{b(+)}_k, \tag{4}$$

most of the inter-user interference can be suppressed.

Now we have a real-time challenge. Stage B is computation-intensive as a high-dimensional SVD (i.e., Eq. (3)) is required. $\widetilde{\mathbf{H}}^b_k$ is a $(|\mathcal{K}^b| - 1)M_{\mathrm{U}} \times M_{\mathrm{BS}}$ matrix with standard SVD complexity of $O\left(|\mathcal{B}||\mathcal{K}^b| \cdot \left[(|\mathcal{K}^b|-1)M_{\mathrm{U}}\right]^2 M_{\mathrm{BS}}\right)$ for $|\mathcal{B}|$ RBs. Its computation time can take more than 70% of the total time when not optimized (from our experiment).

In fact, the computation time of matrix SVD (power method) is tightly related to the decaying speed of singular values [50]. For instance, suppose we have a matrix with 4 decreasing singular values $\sigma_1, \sigma_2, \sigma_3$ and σ_4. If $\sigma_1 \gg \sigma_2 \gg \sigma_3 \approx \sigma_4 \approx 0$, then it is computationally fast to obtain the first two singular values (and associated singular vectors), whereas it would take much longer to obtain the last two singular values. This observation is especially important, since at mmWave frequencies, most signal strength will be concentrated in a few directions due to the limited number of scatterers. As a consequence, it is likely that we encounter several non-zero but close-to-zero singular values. Finding those small singular values would take a long time and it does not help much in terms of throughput performance (as we shall see in Section 5.3).

To verify the singular values of $\widetilde{\mathbf{H}}^b_k$, we conduct the following experiment. We generate 100 instances of $\widetilde{\mathbf{H}}^b_k$ based on mmWave channel model to have $\mathbf{H}^b_k$'s (using the widely adopted mmWave channel model as described in [10]). For analog beamforming, we adopt the well-known DFT-codebook-based method [14], [51]. We set $A_{\mathrm{BS}} = 128$, $A_{\mathrm{U}} = 8$, $M_{\mathrm{BS}} = 20$, $M_{\mathrm{U}} = 4$ and $|\mathcal{K}^b| = 5$, thus $\widetilde{\mathbf{H}}^b_k$ is a 16×20 matrix. We investigate two different scattering scenarios: (a) The number of clusters L_{cl} and the number of rays within each cluster L_{ray} are both set to 3; (b) L_{cl} and L_{ray} are both 6 (as typical number of paths for practical mmWave channels [4], [16], [18], [52]). Averaged by 100 instances, the singular values of $\widetilde{\mathbf{H}}^b_k \widetilde{\mathbf{H}}^{b\dagger}_k$ are plotted in Fig. 3. As we expected, the singular values are decaying fast in the beginning but then flatten out. The decaying speed is faster when the number of paths is smaller. More importantly, the last several singular values are pretty small but very close. This means the corresponding directions in the eigenspace have very weak signals but consume much computational effort to differentiate them, which is wasteful.

Following the above analysis, our next objective is to implement a lower-rank SVD approximation with lower computational complexity. To this end, we apply the randomized SVD technique [47]. The key idea of randomized SVD is that with the help of a random Gaussian matrix $\mathbf{\Omega}$ we form a rank-r basis $\mathbf{P}$, with $r < (|\mathcal{K}^b| - 1)M_{\text{U}} < M_{\text{BS}}$, that captures the dominant directions with the largest SVD singular values. Then the original matrix is projected onto a lower-dimensional subspace (based on basis $\mathbf{P}$) to compute a standard rank-r SVD. We summarize this procedure in Algorithm 1 and call it Raw Randomized SVD. $\widetilde{\mathbf{H}}_k^b$ will be used as input for Algorithm 1. As we see in Step 5 of Algorithm 1, due to the lower-rank r, only a small-scale SVD is required. The complexity of Step 5 is $O\left(r^2 \cdot \left[(|\mathcal{K}^b| - 1)M_{\text{U}} + M_{\text{BS}}\right]\right)$, which has been reduced from $O\left(\left[(|\mathcal{K}^b| - 1)M_{\text{U}}\right]^2 \cdot M_{\text{BS}}\right)$. How to choose a proper value of r will be discussed in the next section.

We now customize the Raw Randomized SVD to further expedite computation time. Note that from Eq. (3) and Eq. (4) in Stage B, our interest is $\widetilde{\mathbf{V}}_k^{b(-)}$, the last $(M_{\text{BS}} - r)$ columns of the right singular matrix corresponding to the smallest $(M_{\text{BS}} - r)$ singular values of $\widetilde{\mathbf{H}}_k^b$, while the singular values and left singular matrix are not necessary for the beamforming design. However, by examining Algorithm 1, a thorough lower-rank SVD is performed, including the calculation of unnecessary singular vectors and singular values (see Steps 3 to 6). Therefore, we can customize Steps 3 to 6 to reduce time. First, note that $\widetilde{\mathbf{V}}_k^{b(-)}$ is also the nullspace of $\mathbf{Y}^\dagger\mathbf{A}$. The calculation of the orthogonalization and normalization of the range of $\mathbf{Y}$ in Step 3 is not required in our case. Second, it is redundant to perform a complete SVD as in Step 5 to obtain the nullspace. Therefore, we apply the QR factorization and choose the Given's rotation method [54] to directly obtain the nullspace of $\mathbf{Y}^\dagger\mathbf{A}$. Moreover, we implement the Given's rotation-based QR by exploiting parallelism. This is done by leveraging the following two characteristics of Given's rotation: i) at each iteration, all the rotated elements (elements of two columns of $\mathbf{Y}$) can be updated simultaneously; ii) at each iteration, only two columns of $\mathbf{Y}$ are dependent. It is worth noting that the mathematical complexity (in $O(\cdot)$) for a complete SVD and Given's rotation-based QR may be the same, but in practice the QR method can lead to dramatic acceleration for our problem in real-time. This is because the QR method (with parallel implementation) can save a lot of redundant calculations and memory write/read caused by operations such as column interchanges and computing variables that are not of our interest. The revised algorithm is summarized in Algorithm 2.

Algorithm 2 for Stage B significantly reduces the computation time of standard SVD operations — the main bottleneck in BD beamforming. The only additional cost is a few more matrix multiplications, which, fortunately, can be parallelized and computed efficiently (more details in Section 6). Although the randomized SVD is an approximation method, we will not analyze its performance here, since it is only an intermediate step for beamforming. Instead, we will discuss and show both the timing and throughput performance by applying randomized SVD in Section 6.

Algorithm 1: Raw Randomized SVD

```
  Given an $m \times n$ matrix $\mathbf{A}$, a target approximation rank
    $r$, and an exponent $q$ (say $q = 1$ or $q = 2$), this
    procedure computes an approximate rank-$r$ factorization
    $\mathbf{A} \approx \mathbf{U\Sigma V}^\dagger$:
1 Generate an $n \times r$ Gaussian matrix $\mathbf{\Omega}$.
2 Form the $m \times r$ matrix $\mathbf{Y} = (\mathbf{AA}^\dagger)^q\mathbf{A\Omega}$ by
    multiplying alternately with $\mathbf{A}$ and $\mathbf{A}^\dagger$.
3 Construct an $m \times r$ matrix $\mathbf{P}$ whose columns form
    an orthonormal basis for the range of $\mathbf{Y}$.
4 Form the $r \times n$ matrix $\mathbf{B} = \mathbf{P}^\dagger\mathbf{A}$.
5 Compute an SVD of the small matrix: $\mathbf{B} = \check{\mathbf{U}}\mathbf{\Sigma V}^\dagger$.
6 Set $\mathbf{U} = \mathbf{P}\check{\mathbf{U}}$.
```

Algorithm 2: Lightweight Nullspace Computation

```
  Given an $m \times n$ matrix $\mathbf{A}$, a target approximation rank
    $r$, and an exponent $q$ (say $q = 1$ or $q = 2$), this
    procedure computes an approximate last $n - r$ right
    singular vectors of $\mathbf{A}$, denoted as $\mathbf{V}$:
1 Generate an $n \times r$ Gaussian matrix $\mathbf{\Omega}$.
2 Form the $m \times r$ matrix $\mathbf{Y} = (\mathbf{AA}^\dagger)^q\mathbf{A\Omega}$ by
    multiplying alternately with $\mathbf{A}$ and $\mathbf{A}^\dagger$.
3 Form a $r \times n$ matrix $\mathbf{B} = \mathbf{Y}^\dagger\mathbf{A}$.
4 Compute QR decomposition of $\mathbf{B}^\dagger$ based on Given's
    rotation: $\mathbf{B}^\dagger = \mathbf{QR}$.
5 Set $\mathbf{V}$ as the last $n - r$ columns of $\mathbf{Q}$.
```

Stage C. In this stage, we construct the digital beamforming matrices $\mathbf{F}_{\text{BB},k}^b$. For given matrices $\overline{\mathbf{H}}_k^b$ and $\widetilde{\mathbf{V}}_k^{b(-)}$, the product of $\overline{\mathbf{H}}_k^b$ and $\widetilde{\mathbf{V}}_k^{b(-)}$ effectively forms user k's channel with no (or minor) inter-user interference (recall Eq. (5.2)). Therefore, the optimal beamforming strategy regarding the effective $M_{\text{U}} \times (M_{\text{BS}} - r)$ channel $\overline{\mathbf{H}}_k^b\widetilde{\mathbf{V}}_k^{b(-)}$ can be realized based on its SVD, which is given by:

$$\overline{\mathbf{H}}_k^b\widetilde{\mathbf{V}}_k^{b(-)} = \overline{\mathbf{U}}_k^b\overline{\mathbf{\Sigma}}_k^b\left[\overline{\mathbf{V}}_k^{b(+)} \quad \overline{\mathbf{V}}_k^{b(-)}\right]^\dagger, \tag{5}$$

where $\overline{\mathbf{V}}_k^{b(+)}$ is the first N_s columns of right singular matrix and $\overline{\mathbf{V}}_k^{b(-)}$ is the remaining columns. Finally, the digital beamforming matrix $\mathbf{F}_{\text{BB},k}^b$ is given by

$$\mathbf{F}_{\text{BB},k}^b = \widetilde{\mathbf{V}}_k^{b(-)}\overline{\mathbf{V}}_k^{b(+)}. \tag{6}$$

In Eq. (5), we encounter another SVD computation. Luckily, the dimension of this to-be-factorized matrix is tied to the number of RF chains at one user, namely M_{U}, which is typically small (e.g., 1 to 4). $\overline{\mathbf{V}}_k^{b(+)}$ can be derived with the help of a Hermitian matrix eigendecomposition (ED) and matrix multiplication, through the following steps:

- Form an $M_{\text{U}} \times (M_{\text{BS}} - r)$ matrix $\mathbf{A}_k^b = \overline{\mathbf{H}}_k^b\widetilde{\mathbf{V}}_k^{b(-)}$;
- Form an $M_{\text{U}} \times M_{\text{U}}$ matrix $\mathbf{B}_k^b = \mathbf{A}_k^b\mathbf{A}_k^{b\dagger}$;
- Compute ED of the Hermitian matrix: $\mathbf{B}_k^b = \mathbf{U}_k^b\mathbf{\Lambda}_k^b\mathbf{U}_k^{b\dagger}$;

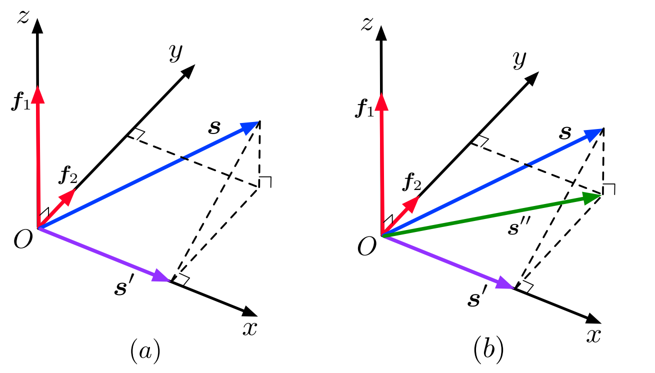


Fig. 4: Signal is projected onto a lower dimensional subspace to avoid interference. (a) Signal $\boldsymbol{s}$ is projected along the x axis, resulting in $\boldsymbol{s}'$; (b) (a) Signal $\boldsymbol{s}$ is projected onto the xOy plane, resulting in $\boldsymbol{s}''$.

- Set $\overline{\mathbf{V}}_k^{b(+)}$ to the first N_s columns of $\mathbf{A}_k^{b\dagger}\mathbf{U}_k^b$.

Note that when $M_{\text{U}} = 1$ or 2, a simple and exact closed-form solution for SVD exists [53] and hence this stage can be completed very fast.

5.3 Approximation with Lower Rank

As we discussed in Section 5.2, Turbo-HB applies lower-rank approximation to reduce computational complexity. Interestingly, in most cases, our approximation does not sacrifice throughput performance. In this section, we offer some intuition behind it. Then we address the last problem, which is how to choose a proper value for r.

Let's revisit the SVD of interference channel $\widetilde{\mathbf{H}}_k^b$ as in Eq. (3). In the M_{BS}-dimensional signal space $[\widetilde{\mathbf{V}}_k^{b(+)}\ \ \widetilde{\mathbf{V}}_k^{b(-)}]$, $\widetilde{\mathbf{V}}_k^{b(-)}$ is an $(M_{\text{BS}} - r)$-dimensional subspace corresponding to the $(M_{\text{BS}} - r)$ smallest interference strengths, while $\widetilde{\mathbf{V}}_k^{b(+)}$ is a r-dimensional subspace corresponding to the r largest interference strengths. When standard SVD is performed, we have $r = (|\mathcal{K}^b| - 1)M_{\text{U}}$ (as in conventional BD approach). Then $\widetilde{\mathbf{V}}_k^{b(-)}$ lies *exactly* in the nullspace of $\widetilde{\mathbf{H}}_k^b$, and therefore *all* inter-user interference will be cancelled when $\mathbf{F}_{\text{BB},k}^b$ is constructed based on $\widetilde{\mathbf{V}}_k^{b(-)}$ (i.e., Eq. (6)). In addition to high complexity, there is another drawback of such a "perfect" interference cancellation. That is, to achieve mutual orthogonality, one has to project the desired signal onto a subspace with a small number of dimensions. As a result, the perceived desired signal strength at a user is reduced.

In Fig. 4, we use a simple example to illustrate this point. In a 3-dimensional signal space, we have a strong interference $\boldsymbol{f}_1$ along the z axis and a weak interference $\boldsymbol{f}_2$ along the y axis. Now we are going to project a desired signal $\boldsymbol{s}$ (originally in the xyz space) onto some subspace to avoid interference (via beamforming). If perfect interference cancellation is required, then $\boldsymbol{s}$ has to be projected along the x axis to achieve orthogonality to both $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$, leading to a smaller-strength signal $\boldsymbol{s}'$, as shown in Fig. 4(a). However, if only the strong interference $\boldsymbol{f}_1$ is required to be cancelled, then one can project $\boldsymbol{s}$ into a larger dimensional subspace, i.e., xOy plane, resulting in $\boldsymbol{s}''$ as shown in Fig. 4(b). Although $\boldsymbol{s}''$ is interfered with by a weak interference $\boldsymbol{f}_2$, $\boldsymbol{s}''$ can preserve higher signal strength than $\boldsymbol{s}'$, which will lead to a higher SINR (and throughput).

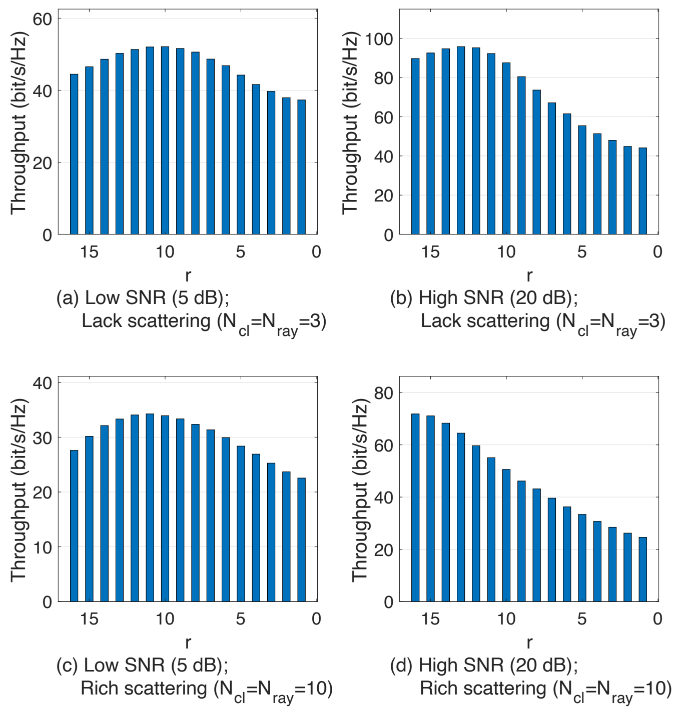


Fig. 5: Achieved network throughput (averaged over 1,000 instances) as a function of approximation rank r under different SNR and number of channel paths.

Turbo-HB is purposefully designed to explore such a design space by tolerating some level of weak interference. When lower-rank SVD approximation is performed, it is meant to only identify r directions corresponding to r strongest interference. Without knowledge of how remaining interference presents, the desired signal will be projected onto a larger dimensional subspace only to avoid the identified interference, preserving greater desired signal strength. This approach is especially effective for scenarios where there is a high correlation among the channels or SNR is low. Since in these scenarios, the last few singular values (i.e., corresponding weak interference strengths) are small compared to the power of white noise. Then the dominant term in the denominator of SINR becomes the power of noise, which cannot be suppressed by interference cancellation. Thus, by tolerating weak interference, desired signal strength is preserved to overcome a bigger issue (the noise), leading to a higher SINR.

Now we address the question of how to choose a proper value for r. Since $0 < r \leq \text{rank}(\widetilde{\mathbf{H}}_k^b) = M_{\text{BS}} - M_{\text{U}}$ and r is an integer, we have $(M_{\text{BS}} - M_{\text{U}})$ possible values for r. If we choose r to be too large (i.e., close to $(M_{\text{BS}} - M_{\text{U}})$), then we will have to get into high-dimensional SVD operations, which are what we try to avoid. On the other hand, if we choose r to be too small, then we may experience serious sacrifice in throughput performance. So the goal is to find an optimal r that offers the best trade-off. Unfortunately, finding the optimal value of r (in terms of maximizing network throughput) is intractable, due to the large search space and non-convex objective function.

To gain some insight into what value of r should be, we conduct the following experiment. We randomly generate 1,000 channel instances under different settings following the mmWave channel model. For each instance, we enumerate all possible r's and calculate its corresponding throughput C. For the time being, we focus only on the objective function (throughput) and defer consideration of computation time till later. In the experiment, we set $N_s = 2$ and use the same settings as those used in Section 5.2 except that we now vary SNR values and the number of channel paths. Specifically, let's consider a low SNR scenario (5 dB) and a high SNR scenario (20 dB), each of which is in combination with a small number of clusters and rays ($L_{\text{cl}} = L_{\text{ray}} = 3$) or a large number of clusters and rays ($L_{\text{cl}} = L_{\text{ray}} = 10$). Fig. 5 shows the achieved network throughput as a function of approximation rank r under these four scenarios. Note that when $r = 16$, the achieved throughput value (the first blue bar in each figure) is what is achieved by standard SVD (as in traditional BD method). For the first three scenarios, where the channels are experiencing at least low SNR or high correlations, we observe that the throughput goes up at first and then goes down as the value of r decreases. Only when the channels maintain both high SNR and low correlations (as in scenario (d)), the network throughput would strictly decline as the value of r decreases. However, scenario (d) is relatively rare for mmWave systems. This experiment suggests that the lower-rank r indeed offers the opportunity for higher throughput, especially at low SNR or high correlation scenarios. Under this scenarios (a), (b) and (c), setting $r = \frac{M_{\text{BS}}}{2} = 10$ would offer better (or comparable) performance than that with $r = 16$ in most instances.

The results in Fig. 5 are averaged over 1,000 channel instances. However, our interest is on a particular channel instance, and the optimal choice of r based on averaging over 1,000 channel instances may not perform well in this particular instance. Therefore, we propose to employ multiple choices of promising r's in parallel and derive multiple beamforming candidates corresponding to these r's. That is, we execute several different lower-rank approximations simultaneously, where the set of target rank is given by

$$\mathcal{R} = \{r - \delta, \cdots, r - 1\ , \ r\ , \ r + 1\ , \cdots, r + \delta\}, \tag{7}$$

where r is around $\frac{M_{\text{BS}}}{2}$ (which may be adjusted according to empirical statistics), and δ is a parameter to control the number of elements in $\mathcal{R}$. As $|\mathcal{R}|$ different lower-rank approximations are implemented, we will have $|\mathcal{R}|$ different solutions of $\mathbf{F}^b_{\text{BB},k}$ for each user on each RB after Stage C. Among these $|\mathcal{R}|$ solutions, we evaluate their throughput performance (i.e., C in Eq. (2)) and choose the one that offers the largest objective value as the final beamforming matrix.

6 Implementation

In this section, we present the implementation of our design in Section 5. Our implementation is done on Nvidia DGX station – a COTS GPU platform. Our Nvidia DGX Station consists of 4 V100 GPU cards but we use only two of them. Each V100 card includes 80 streaming multiprocessors (SMs), and each SM has 64 CUDA cores. The CPU of our DGX station is Intel Xeon E5-2698 v4 2.2 GHz (20-core). The data communication between CPU and GPU is based on a PCIe V3.0 architecture [55]. CUDA programming tool (version 10.2) [57] is used to realize our algorithm and schedule the memory and processing cores.

For a successful implementation of Turbo-HB, we must have a thorough knowledge of the capability and limitations of the GPU and find a way to fit our problem into the platform. In general, the more parallelism and less overhead in the implementation, the better the performance we can achieve. As such, we focus on the following two objectives in our implementation:

1) fully utilize GPU processing cores,
2) minimize memory access time.

In the rest of this section, we present the details of our implementation based on the above two objectives.

6.1 Workflow on GPU

The key to fully utilize GPU processing cores is to have a sufficient large amount of parallel workloads in flight to feed all the GPU cores. By our design in Section 5, the computations for beamforming matrice are independent among different RBs, different users, and different target ranks. Thus, we can spread out the computation tasks over all available processing cores. At each step in the implementation, the computation tasks are broken into a number of parallel processing flows. Each flow is a group of parallel threads that executes certain operations. All the flows shall be mutually data-independent and have the same computation procedures to take advantage of GPU's SIMD architecture. Based on the architecture of our GPU V100, every consecutive 32 parallel threads are assembled into a group called a *warp* for executing exactly the same instructions (while handling different data). Therefore, it is preferable that a flow consists of an integral multiple of (or close to an integral multiple of) 32 threads. As V100 has 80 SMs, the number of flows should be at least 80 to avoid idle SMs.

As illustrated in Fig. 6, our implementation includes the following key steps.

Step 0: Initialization. The system first sets up global parameters, including the number of RBs $|\mathcal{B}|$, user sets $\mathcal{K}^b$ on RB b, and the number of RF chains at BS M_{BS} and at users M_{U}, etc. Then we calculate and allocate the memory space needed on GPU for storing the matrices and variables.

Step 1: Set up global parameters and transfer the compressed CSI from host to GPU. At the beginning of each time slot, the host transfers $\sum_{b\in\mathcal{B}} |\mathcal{K}^b|$ partial CSI (i.e., $\mathbf{V}^b_k$'s and $\mathbf{\Sigma}^b_k$'s) from host memory to GPU global memory (also known as device global memory). Since we use two V100 GPU cards, we divide the channel matrices into two halves. The first half corresponds to the first $\frac{|\mathcal{B}|}{2}$ RBs and is transferred to the first GPU card. The second half will be handled by the second GPU card.

Step 2: Execute Stage A. The objective of this step is to generate $\widetilde{\mathbf{H}}^b_k$ for every $k \in \mathcal{K}^b$ and $b \in \mathcal{B}$ on GPU. We generate a total number of $\sum_{b\in\mathcal{B}} |\mathcal{K}^b|$ parallel flows, where each flow corresponds to the beamforming matrix of one user on an RB. We program one thread to calculate one element of $\widetilde{\mathbf{H}}^b_k$, thus a total number of $\sum_{b\in\mathcal{B}} |\mathcal{K}^b| \cdot N_{\text{thread}}$ threads are

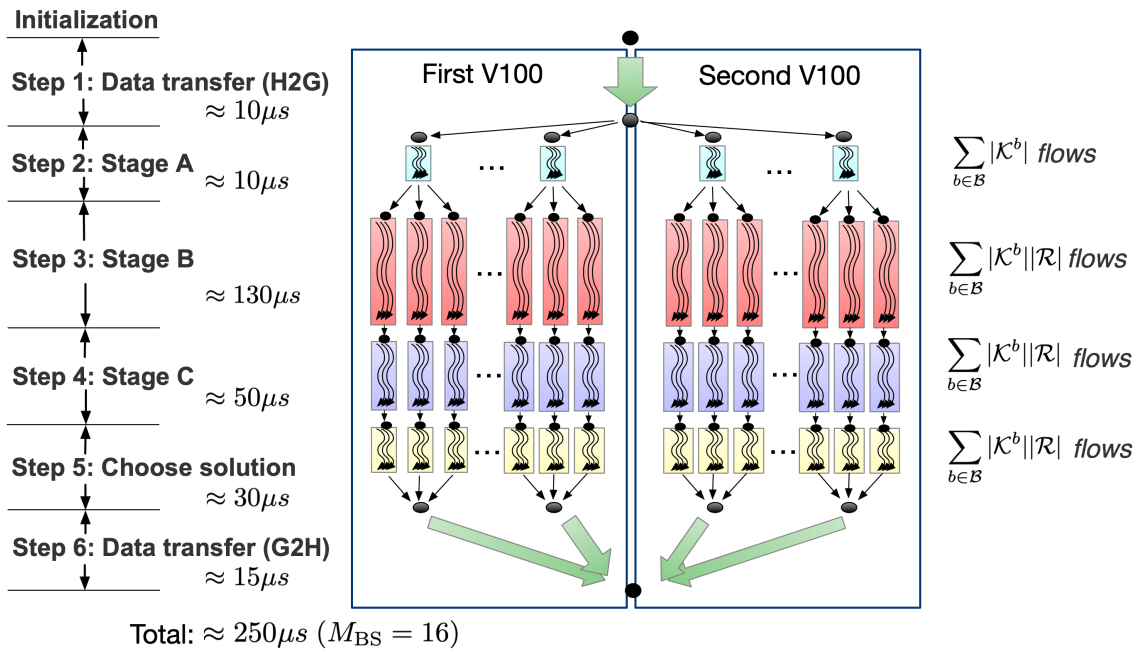

Initialization
Step 1: Data transfer (H2G) $\approx 10\mu s$
Step 2: Stage A $\approx 10\mu s$
Step 3: Stage B $\approx 130\mu s$
Step 4: Stage C $\approx 50\mu s$
Step 5: Choose solution $\approx 30\mu s$
Step 6: Data transfer (G2H) $\approx 15\mu s$
Total: $\approx 250\mu s$ ($M_{\mathrm{BS}} = 16$)
First V100
Second V100
$\sum_{b\in\mathcal{B}}|\mathcal{K}^b|$ flows
$\sum_{b\in\mathcal{B}}|\mathcal{K}^b||\mathcal{R}|$ flows
$\sum_{b\in\mathcal{B}}|\mathcal{K}^b||\mathcal{R}|$ flows
$\sum_{b\in\mathcal{B}}|\mathcal{K}^b||\mathcal{R}|$ flows

Fig. 6: Workflow of implementing Turbo-HB on GPUs.

spawned in this step, where $N_{\text{thread}} = (|\mathcal{K}^b| - 1)M_{\mathrm{U}} \times M_{\mathrm{BS}}$ is the total number of elements in $\widetilde{\mathbf{H}}_k^b$.

Step 3: Execute Stage B. As we discussed in Section 5.2, the main task of this stage is to compute the approximate nullsapce of $\widetilde{\mathbf{H}}_k^b$. We generate a total number of $\sum_{b\in\mathcal{B}}|\mathcal{K}^b||\mathcal{R}|$ parallel flows, where each flow corresponds to the computation for one user on one RB with one target rank. Each flow executes Algorithm 2 to derive matrix $\widetilde{\mathbf{V}}_k^{b(-)}$. In particular, for the QR decomposition in Step 4 of Algorithm 2, we use Given's rotation method. The computation requires multiple iterations and each iteration would overwrite the processing matrix. To reduce the memory access time for repeated accesses, we first transfer the input matrix from GPU's global memory to the fast on-chip shared memory, then the iterative computations are performed based on shared memory access. The output matrix is transferred back to global memory after QR decomposition is completed. Step 3 is the most computation-intensive step in our implementation. It consumes around 130 μs (for $M_{\mathrm{BS}} = 16$, $M_{\mathrm{U}} = 2$ and $|\mathcal{K}^b| = 8$) after our optimization.

Step 4: Execute Stage C. In this step, $\sum_{b\in\mathcal{B}}|\mathcal{K}^b||\mathcal{R}|$ parallel flows are generated to calculate $\mathbf{F}_{\mathrm{BB},k}^b$. This step includes a small dimensional SVD operation. Note that when $M_{\mathrm{U}} = 1$ or 2, simple closed-form expressions can be directly applied for SVD computation.

Step 5: Choose the best solution. After Step 4, we obtain $|\mathcal{R}|$ beamforming candidates for each user on each RB. In $\sum_{b\in\mathcal{B}}|\mathcal{K}^b||\mathcal{R}|$ parallel flows, we evaluate their throughput performance as in Eq. (2) for every beamforming candidate. The best $\mathbf{F}_{\mathrm{BB},k}^b$ that provides the highest objective value C in Eq. (2) will be chosen as the final solution. To speed up comparison, parallel reduction technique [56] is employed.

Step 6: Transfer beamforming solution from GPU to host. Once Step 5 is accomplished, the final beamforming solution (i.e., $\mathbf{F}_{\mathrm{BB},k}^b$ for every $k \in \mathcal{K}^b$ and $b \in \mathcal{B}$) is transferred from GPU memory to the host memory.

6.2 Speed-Up Techniques

Now we discuss two specific techniques that we have employed in Turbo-HB to enhance parallelism and reduce memory access time.

Batching Batched matrix operations are critical to our problem, as we have to execute a large number of independent matrix operations following the same procedure. As an example, suppose we need to execute hundreds or even thousands of matrix multiplications simultaneously. The programmer needs to generate a kernel with a sufficient number of threads and divide these threads into a number of groups. Then each group computes one or a few matrix multiplications, such that this kernel is able to perform batched matrix multiplications. Similarly, other matrix operations (following the same procedure), such as a large number of independent matrix ED operations, should be programmed in a batched manner to fully occupy the processing cores.

Minimizing global memory access Compared to other types of memory access, accessing global memory is much more time-consuming. We identify two techniques that can help minimize global memory access in our problem.

First, the programmer should carefully coalesce memory access, i.e., consolidate multiple memory accesses into a single transaction. This is particularly important when we handle a large number of matrix operations. The key to memory coalescing is to store the matrices consecutively in the memory with proper indexing. Then the programmer can allow consecutive threads to read consecutive (and aligned) memory and minimize the number of transactions.

Second, instead of global memory accesses, which is more time-consuming, we can use on-chip shared memory accesses, which is much faster (but with limited storage space). Suppose we want to compute a matrix multiplication

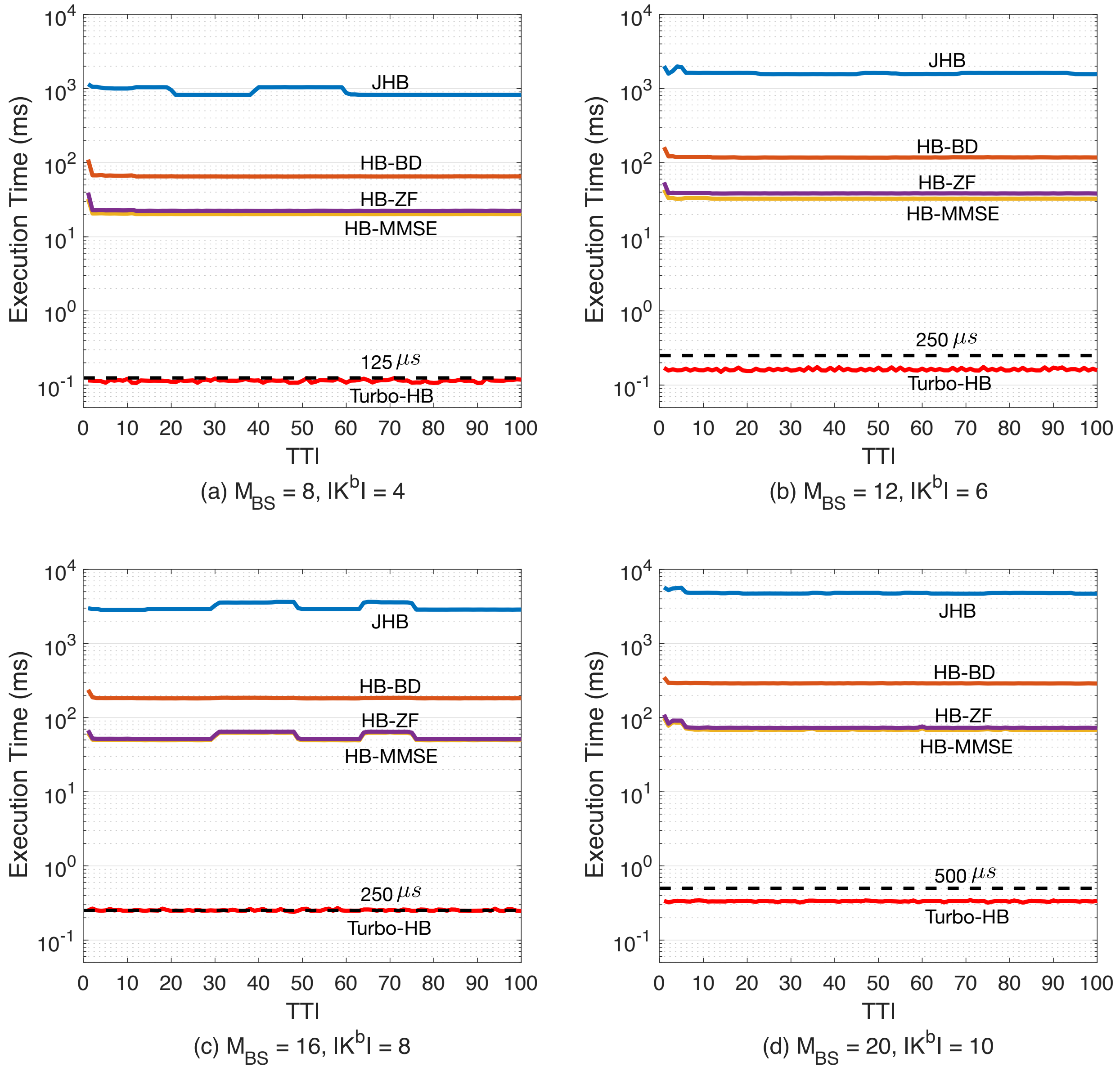


Fig. 7: Comparison of execution time of different schemes under different MU-MIMO scenarios.

$\mathbf{C}_{m\times n} = \mathbf{A}_{m\times l}\mathbf{B}_{l\times n}$. A straightforward approach for parallelism is to program each thread to take care of one element of $\mathbf{C}$. Then we need to read $\mathbf{A}$ n times from the global memory and $\mathbf{B}$ m times. In contrast, if matrix multiplication is based on shared memory [57], we only need to read $\mathbf{A}$ for (n / block size) times from the global memory and $\mathbf{B}$ for (m / block size) times. The remaining computations are done by accessing the shared memory.

7 Experimental Validation

In this section, we present our experimental results, with a focus on timing and throughput performance. We also compare with other state-of-art sequential HB schemes. For the analog beamforming part, we apply the widely adopted DFT-codebook-based method [14], [51] for all schemes. For digital beamforming schemes, we choose HB-BD [14], HB-MMSE and HB-ZF for comparison. We also include one joint analog and digital HB method (JHB) [11] to show its timing performance.

Experiment Setup We consider a cellular communication scenario with one BS and a number of users. The number of available RBs is up to 100. The BS is equipped with 128 antennas and each user is equipped with 16 antennas (a typical number for hybrid architecture at mmWave frequencies [4], [10], [14]). The number of RF chains at the BS varies from 8 to 20, while the number of RF chains at a user is 2. Each active link is assumed to transmit $N_s = 2$ data streams. The number of active users for MU-MIMO transmission on each RB (i.e., $|\mathcal{K}^b|$) varies in this study. For the wireless channels, we use the widely considered cluster-based mmWave channel model [10]. The number of clusters L_{cl}, the number of propagation paths L_{ray} caused by each cluster and SNR (i.e., $\frac{P_{\text{T}}}{\sigma^2}$) will be given under different settings. The angle spread σ_{AS} is set to 5 degrees. In this paper, we do not consider the presence of blockage. We set

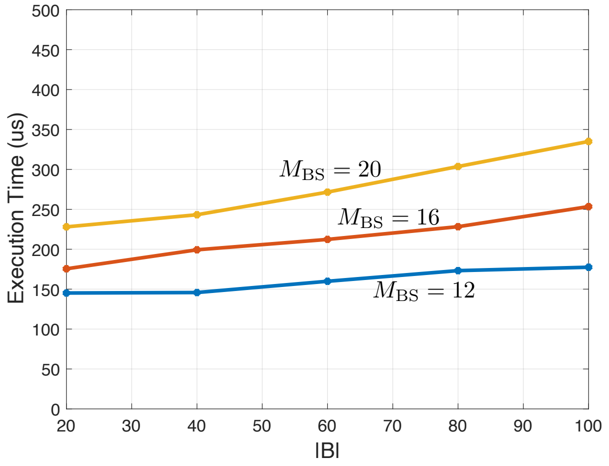


Fig. 8: Average execution time of Turbo-HB vs. the number of available RBs $|\mathcal{B}|$ under different M_{BS} settings.

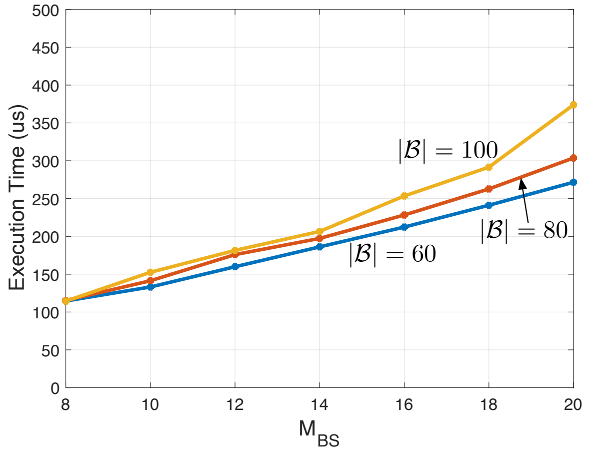


Fig. 9: Average execution time of Turbo-HB vs. the number of RF chains M_{BS} at the BS under different settings of available RBs $|\mathcal{B}|$.

parameter δ (as defined in Section 5.3) to 2.

Timing Performance We first verify that Turbo-HB can indeed offer the beamforming solution in less than 1 ms for all settings in our experiments and even achieves as little as 125 μs execution time in some settings. Note that the time consumed for data transfer between CPU and GPU is included in Turbo-HB's total execution time.

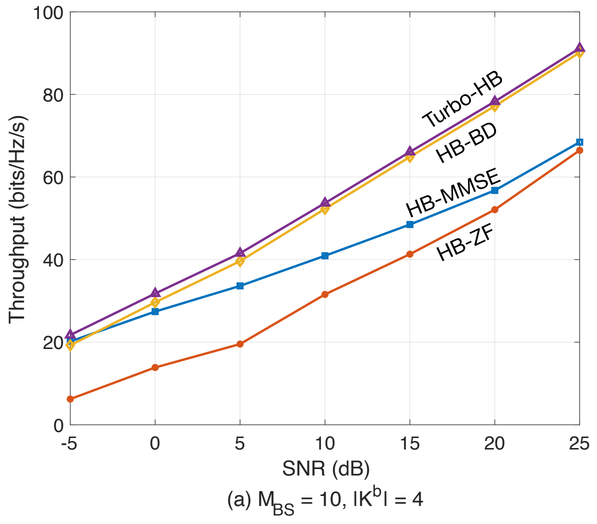
We first run the experiments for 100 consecutive TTIs under different settings as follows: (a) $M_{\text{BS}} = 8$, $|\mathcal{K}^b| = 4$, (b) $M_{\text{BS}} = 12$, $|\mathcal{K}^b| = 6$, (c) $M_{\text{BS}} = 16$, $|\mathcal{K}^b| = 8$ and (d) $M_{\text{BS}} = 20$, $|\mathcal{K}^b| = 10$. For the sequential algorithms (Turbo-HB, HB-BD, HB-MMSE and HB-ZF), we only count the computation time of the digital beamforming part. But for the joint algorithm (JHB), we have to count the time consumed both for its digital beamforming and analog beamforming since they are inseparable. Our GPU-based algorithm is run on the CUDA platform while others are run on the Matlab platform. Fig. 7 shows the results of execution time by different schemes. JHB, HB-BD, HB-MMSE and HB-ZF require a computation time on the order of 10^3 ms, 10^2 ms, 10^1 ms and 10^1 ms, respectively. Our experiments show that Turbo-HB finds beamforming solution in 114 μs, 162 μs, 250 μs, and 335 μs averaged by 100 TTIs under settings (a), (b), (c) and (d), respectively. Based on the numerologies defined in 5G NR, Turbo-HB can meet the timing requirement for numerology 3 (125 μs TTI), numerology 2 (250 μs TTI) and numerology 1 (500 μs TTI) for 100 RBs with up to 4, 8, and 10 MU-MIMO users on each RB, respectively.

Next, we conduct experiments to examine Turbo-HB's total execution time under different numbers of available RBs $|\mathcal{B}|$. We consider the following settings: (a) $M_{\text{BS}} = 12, |\mathcal{K}^b| = 6$, (b) $M_{\text{BS}} = 16, |\mathcal{K}^b| = 8$. and (c) $M_{\text{BS}} = 20, |\mathcal{K}^b| = 10$. Fig. 8 shows Turbo-HB's execution time performance (with value for each point being average over 100 TTIs) for the three settings. Note that the execution time increases slowly (and close to linear) as the number of RBs increases. This is because under Turbo-HB, computation among different RBs is executed in parallel and is not very sensitive to the number of RBs. For a given M_{BS}, the network operator can set the upper bound for the number of RBs to meet a certain 5G numerology. For example, when $M_{\text{BS}} = 16$, if the number of RBs is no more than 95 (a large number), we can meet 5G numerology 2 requirement (250 μs).

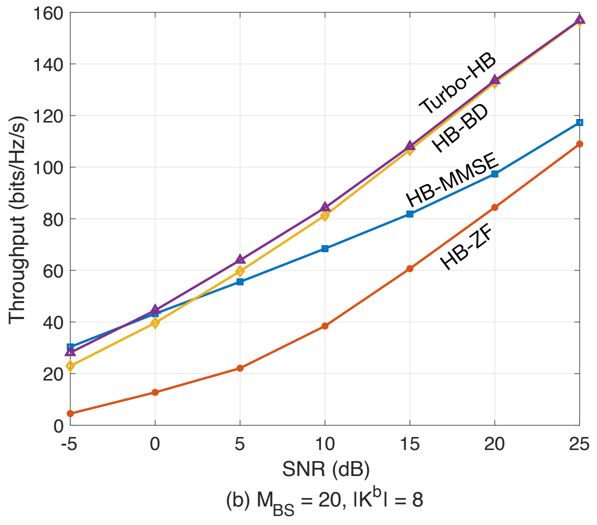
In Fig. 9, we vary the number of RF chains at the BS (i.e., M_{BS}) to show its impact on Turbo-HB's execution time. For this study, we consider the settings of $|\mathcal{B}| \in \{60, 80, 100\}$ and $|\mathcal{K}^b| = \frac{M_{\text{BS}}}{N_s}$, while M_{BS} varies from 8 to 20. As expected, the results in Fig. 9 show that Turbo-HB's average execution time is increasing with M_{BS}. Compared with varying $|\mathcal{B}|$, Turbo-HB is more sensitive to the change of M_{BS}. This is because the larger the M_{BS}, the higher dimensional matrix operations will be required, which leads to more computation time. However, Turbo-HB is able to complete the computation in real-time, thanks to its design based on randomized SVD.

Throughput Performance We first evaluate throughput performance achieved by different schemes under varying SNR values. We consider two different settings: (a) $M_{\text{BS}} = 10$, $|\mathcal{K}^b| = 4$, and (b) $M_{\text{BS}} = 20$, $|\mathcal{K}^b| = 8$. We set $N_{\text{cl}} = N_{\text{ray}} = 3$ and SNR varies from -5 dB to 25 dB in both cases. Fig. 10 shows that in both cases, throughput under conventional HB-MMSE and HB-ZF methods are below the others, as MMSE and ZF are not designed for mmWave systems and the poorly conditioned channel greatly degrades MMSE/ZF's performance [16], [28], [29]. In Fig. 10, Turbo-HB is able to achieve similar performance as the classical HB-BD and is better than the others.

Next, we vary the channel correlation condition (by varying the number of propagation clusters L_{cl}) and study its impact on throughput performance. We fix SNR = 20 dB, $M_{\text{BS}} = 20$, $|\mathcal{K}^b| = 8$, and $N_{\text{ray}} = 3$. We vary L_{cl} from 1 to 7. Fig. 11 shows the throughput achieved by different schemes as a function of N_{cl}. The results show that the performance by HB-MMSE and HB-ZF is significantly lower than the others, especially when the number of clusters is small (and thus the channels are highly correlated). On the other hand, Turbo-HB is able to achieve similar performance as HB-BD and offers high throughput than HB-MMSE and HB-ZF. This is because both Turbo-HB and HB-BD are SVD-based and are capable of identifying the best signal directions for



(a) $M_{BS} = 10$, $|K^b| = 4$



(b) $M_{BS} = 20$, $|K^b| = 8$

Fig. 10: Comparison of throughput achieved by different schemes as a function of SNR under different MU-MIMO scenarios.

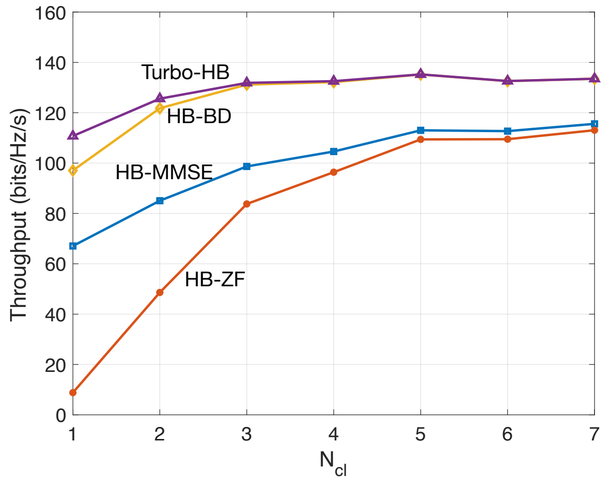


Fig. 11: Comparison of the throughput achieved by different schemes under different channel propagation conditions.

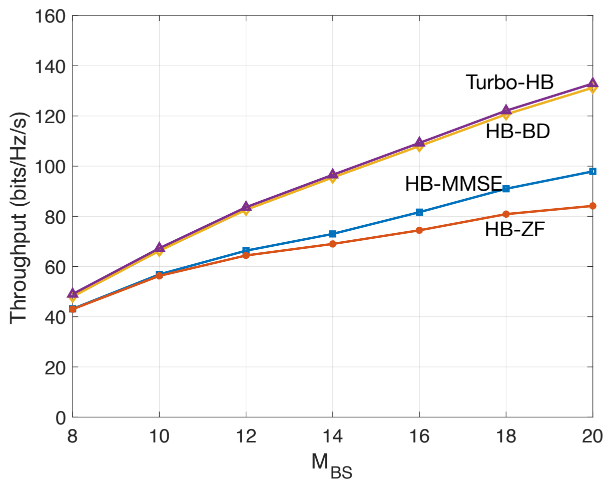


Fig. 12: Comparison of throughput achieved by different schemes under different number of RF chains at the BS.

beamforming. When the number of channel paths is small, Turbo-HB is able to obtain even better performance than HB-BD. The reason behind this was given in our discussions in Section 5.3.

Finally, we present throughput performance under different numbers of RF chains M_{BS} at the BS. We consider the setting of SNR = 20 dB and $N_{\text{cl}} = N_{\text{ray}} = 3$. M_{BS} is chosen from $\{8, 10, 12, 14, 16, 18, 20\}$, and $|\mathcal{K}^b|$ is chosen from $\{2, 3, 4, 5, 6, 7, 8\}$ accordingly. In Fig. 12, the results show that the network throughput is increasing with M_{BS} for all schemes as more users can be supported. The performance gap between Turbo-HB/HB-BD and HB-MMSE/HB-ZF is also increasing with M_{BS} as the SVD-based approaches can better reap the benefits provided by additional RF chains. Again, we find that Turbo-HB can offer similar performance as HB-BD and outperforms other schemes.

Summary of Results The experimental results show that Turbo-HB can meet the 1-ms real-time requirement under all tested settings and can meet the 5G requirement with appropriate numerology. On the other hand, all other schemes incur a computation time that is of orders of magnitude higher than Turbo-HB and none of them can meet the 5G timing requirement. Further, Turbo-HB is able to offer a throughput performance that is better or comparable to the state-of-the-art algorithms.

8 Conclusions

This paper presents Turbo-HB, the first design and implementation that addresses the real-time challenge for beamforming under HB architecture. To reduce computation time, Turbo-HB exploits the randomized SVD technique by leveraging channel sparsity at mmWave frequencies. Further, Turbo-HB exploits large-scale parallel processing, with optimized matrix operations and minimized memory accesses. We implemented Turbo-HB on COTS Nvidia DGX Station with the CUDA programming platform. Through extensive experimental studies, we found that Turbo-HB is able to find beamforming matrices successfully under 1 ms for all tested cases. Specifically, Turbo-HB can meet the 125μs (numerology 3), 250 μs (numerology 2) and 500 μs (numerology 1) timing requirements for 100 RBs with up to 4, 8, and 10 MU-MIMO users on each RB, respectively. In the meanwhile, Turbo-HB offers competitive throughput performance compared to the state-of-the-art algorithms.

Acknowledgments

The first author wishes to thank Prof. Mark Embree from the Department of Mathematics, Virginia Tech, for his insightful discussion on SVD computations. The authors thank the anonymous reviewers for their feedback.

References

[1] Y. Chen, Y. Huang, C. Li, Y.T. Hou and W. Lou, "Turbo-HB: A novel design and implementation to achieve ultra-fast hybrid beamforming," in *Proc. of IEEE INFOCOM*, pp. 1489–1498, Toronto, ON, Canada, July 2020.

[2] W. Hong, K. Baek, Y. Lee, Y. Kim and S. Ko, "Study and prototyping of practically large-scale mmWave antenna systems for 5G cellular devices," *IEEE Communications Magazine*, vol. 52, no. 9, pp. 63–69, Sept. 2014.

[3] S. Han, C. I, Z. Xu and C. Rowell, "Large-scale antenna systems with hybrid analog and digital beamforming for millimeter wave 5G," *IEEE Communications Magazine*, vol. 53, no. 1, pp. 186–194, Jan. 2015.

[4] A.F. Molisch, V.V. Ratnam, S. Han, Z. Li, S.L.H. Nguyen, L. Li and K. Haneda, "Hybrid beamforming for massive MIMO: A survey," *IEEE Communications Magazine*, vol. 55, no. 9, pp. 134–141, Dec. 2017.

[5] I. Ahmed, H. Khammari, A. Shahid, A. Musa, K.S. Kim, E.D. Poorter, and I. Moerman, "A survey on hybrid beamforming techniques in 5G: Architecture and system model perspectives," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 4, pp. 3060–3097, Fourthquarter 2018.

[6] C. Fang, B. Makki, J. Li and T. Svensson, "Hybrid precoding in cooperative millimeter wave networks," *IEEE Transactions on Wireless Communications*, vol. 20, no. 8, pp. 5373–5388, Aug. 2021.

[7] X. Gao, L. Dai, S. Han, C. I and R. W. Heath, "Energy-efficient hybrid analog and digital precoding for mmWave MIMO systems with large antenna arrays," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 4, pp. 998–1009, April 2016.

[8] N.N. Moghadam, G. Fodor, M. Bengtsson and D.J. Love, "On the energy efficiency of MIMO hybrid beamforming for millimeter-wave systems with nonlinear power amplifiers," *IEEE Transactions on Wireless Communications*, vol. 17, no. 11, pp. 7208–7221, Nov. 2018.

[9] W. Ni, X. Dong and W.S. Lu, "Near-optimal hybrid processing for massive MIMO systems via matrix decomposition," *IEEE Transactions on Signal Processing*, vol. 65, no. 15, pp. 3922–3933, 2017.

[10] O. El Ayach, S. Rajagopal, S. Abu-Surra, Z. Pi and R.W. Heath Jr., "Spatially sparse precoding in millimeter wave MIMO systems," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 3, pp. 1499–1513, March 2014.

[11] T.E. Bogale and L.B. Le, "Beamforming for multiuser massive MIMO systems: Digital versus hybrid analog-digital," in *Proc. of IEEE GLOBECOM*, pp. 4066–4071, Austin, TX, Dec. 2014.

[12] C. Rusu, R. Mendez-Rial, N. Gonzalez-Prelcic and R.W. Heath, "Low complexity hybrid precoding strategies for millimeter wave communication systems," *IEEE Transactions on Wireless Communications*, vol. 15, no. 12, pp. 8380–8393, Dec. 2016.

[13] A. Alkhateeb, J. Mo, N. Gonzalez-Prelcic and R.W. Heath, "MIMO precoding and combining solutions for millimeter-wave systems.," *IEEE Communications Magazine*, vol. 52, no. 12, pp. 122–131, Dec. 2014.

[14] W. Ni and X. Dong, "Hybrid block diagonalization for massive multiuser MIMO systems". *IEEE Transactions on Communications*, vol. 64, no. 1, pp. 201–211, Jan. 2016.

[15] L. Liang, W. Xu and X. Dong, "Low-complexity hybrid precoding in massive multiuser MIMO systems," *IEEE Wireless Communications Letters*, vol. 3, no. 6, pp. 653–656, Dec. 2014.

[16] Y. Ghasempour, M.K. Haider, C. Cordeiro, D. Koutsonikolas and E. Knightly, "Multi-stream beam-training for mmWave MIMO networks," in *Proc. of ACM MobiCom*, pp. 225–239, New Delhi, India, Oct. 2018.

[17] Y. Ghasempour and E.W. Knightly, "Decoupling beam steering and user selection for scaling multi-user 60 GHz WLANs." in *Proc. of ACM MobiHoc*, pp. 1–10, Chennai, India, July 2017.

[18] S. Sur, I. Pefkianakis, X. Zhang, and K.H. Kim, "Towards scalable and ubiquitous millimeter-wave wireless networks," in *Proc. of ACM MobiCom*, pp. 257–271, New Delhi, India, Oct. 2018.

[19] 3GPP R1-165179, "Multi-antenna technology for NR interface." Accessed: Dec. 5, 2021. Available: https://www.3gpp.org/ftp/tsg_ran/WG1_RL1/TSGR1_85/Docs/R1-165179.zip

[20] 3GPP R1-165362, "Multi-antenna architectures and implementation issues in NR." Accessed: Dec. 5, 2021. Available: https://www.3gpp.org/ftp/tsg_ran/WG1_RL1/TSGR1_85/Docs/R1-165362.zip

[21] 3GPP R1-164018, "Analog/digital/hybrid beamforming for massive MIMO." Accessed: Dec. 5, 2021. Available: https://www.3gpp.org/ftp/tsg_ran/WG1_RL1/TSGR1_85/Docs/R1-164018.zip

[22] 3GPP R1-164708, "NR massive MIMO design considerations." Accessed: Dec. 5, 2021. Available: https://www.3gpp.org/ftp/tsg_ran/WG1_RL1/TSGR1_85/Docs/R1-164708.zip

[23] 3GPP TR 38.802, "Study on new radio access technology; Physical layer aspects." Accessed: Dec. 5, 2021. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3066

[24] 3GPP TR 36.897, "Study on elevation beamforming/full-dimension (FD) MIMO for LTE." Accessed: Dec. 5, 2021. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2580

[25] 3GPP TS 38.104, "NR; Base station (BS) radio transmission and reception." Accessed: Dec. 5, 2021. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3202

[26] 3GPP TS 38.211 version 16.2.0, "NR; physical channels and modulation." Accessed: Dec. 5, 2021. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3213

[27] Q.H. Spencer, A.L. Swindlehurst and M. Haardt, "Zero-forcing methods for downlink spatial multiplexing in multiuser MIMO channels," *IEEE Transactions on Signal Processing*, vol. 52, no. 2, pp. 461–471, Feb. 2004.

[28] S. K. Mohammed and E. G. Larsson, "Improving the performance of the zero-forcing multiuser MISO downlink precoder through user grouping," *IEEE Transactions on Wireless Communications*, vol. 15, no. 2, pp. 811–826, Feb. 2016.

[29] V. Stankovic and M. Haardt, "Multi-user MIMO downlink precoding for users with multiple antennas," *Wireless World Research Forum*, pp. 12–14, Toronto, ON, Canada, Nov. 2004.

[30] D. Patil, "Block diagonalization based beamforming," *Master Thesis*, KTH Royal Institute of Technology, Stockholm, Sweden, 2017.

[31] Y. Huang, S. Li, Y.T. Hou and W. Lou, "GPF: A GPU-based design to achieve $\sim$100 μs scheduling for 5G NR," in *Proc. of ACM MobiCom*, pp. 207–222, New Delhi, India, Oct. 2018.

[32] Y. Huang, S. Li, Y. Chen, Y.T. Hou, W. Lou, J. Delfeld, V. Ditya, "GPU: A new enabling platform for real-time optimization in wireless networks," *IEEE Network*, vol. 34, no. 6, pp. 77–83, Nov./Dec. 2020.

[33] Z. Shen, R. Chen, J.G. Andrews, R.W. Heath and B.L. Evans, "Low complexity user selection algorithms for multiuser MIMO systems with block diagonalization," *IEEE Transactions on Signal Processing*, vol. 54, no. 9, pp. 3658–3663, Sept. 2006.

[34] X. Zhang and J. Lee, "Low complexity MIMO scheduling with channel decomposition using capacity upperbound," *IEEE Transactions on Communications*, vol. 56, no. 6, pp. 871–876, June 2008.

[35] H. Yu, O. Bejarano and L. Zhong, "Combating inter-cell interference in 802.11ac-based multi-user MIMO networks," in *Proc. of ACM MobiCom*, pp. 141-–152, Maui, Hawaii, USA, Sept. 2014.

[36] S. Sur, I. Pefkianakis, X. Zhang and K.-H. Kim, "Practical MU-MIMO user selection on 802.11ac commodity networks, " in *Proc. of ACM MobiCom*, pp. 122-–134, New York, NY, USA, Oct. 2016.

[37] X. Xie, X. Zhang and E. Chai, "Cross-cell DoF distribution: Combating channel hardening effect in multi-cell MU-MIMO networks," *Proc. of ACM MobiCom*, pp. 337—346, Hangzhou, China, June 2015.

[38] W. Yang, G. Durisi and E. Riegler, "On the capacity of large-MIMO block-fading channels," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 2, pp. 117–132, Feb. 2013.

[39] A. Alkhateeb, G. Leus and R.W. Heath, "Limited feedback hybrid precoding for multi-user millimeter wave systems," *IEEE Transac-*

tions on Wireless Communications, vol. 14, no. 11, pp. 6481–6494, July 2015.

[40] S.S. Christensen, R. Agarwal, E. Carvalho and J.M. Cioffi, “Weighted sum-rate maximization using weighted MMSE for MIMO-BC beamforming design,” *IEEE Transactions on Wireless Communications*, vol. 7, no. 12, pp. 4792-–4799, Dec. 2008.

[41] A. Zhou, X. Zhang and H. Ma, “Beam-forecast: Facilitating mobile 60 GHz networks via model-driven beam steering,” in *Proc. of IEEE INFOCOM*, Atlanta, GA, pp. 1–9, May 2017.

[42] W. Roh, J.Y. Seol, J. Park, B. Lee, J. Lee, Y. Kim, J. Cho, K. Cheun and F. Aryanfar, “Millimeter-wave beamforming as an enabling technology for 5G cellular communications: theoretical feasibility and prototype results,” *IEEE Communications Magazine*, vol. 52, no. 2, pp. 106–113, Feb. 2014.

[43] M.K. Haider, Y. Ghasempour, D. Koutsonikolas, and E.W. Knightly, “LiSteer: mmWave beam acquisition and steering by tracking indicator LEDs on wireless APs,” in *Proc. of ACM MobiCom*, pp. 273–288, New Delhi, India, Nov. 2018.

[44] E. Bjornson, L. Van der Perre, S. Buzzi and E.G. Larsson, “Massive MIMO in sub-6 GHz and mmWave: Physical, practical, and use-case differences,” *IEEE Wireless Communications*, vol. 26, no. 2, pp. 100–108, April 2019.

[45] V. Raghavan, A. Partyka, A. Sampath, S. Subramanian, O.H. Koymen, K. Ravid, J. Cezanne, K. Mukkavilli and J. Li, “Millimeter-wave MIMO prototype: Measurements and experimental results,” *IEEE Communications Magazine*, vol. 56, no. 1, pp. 202–209, Jan. 2018.

[46] T.S. Rappaport, *Wireless communications: Principles and Practice (vol. 2)*. New Jersey: Prentice hall PTR, 1996. ISBN: 9780133755367.

[47] N. Halko, P.G. Martinsson and J.A. Tropp, “Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions.” *SIAM review*, vol. 53, no. 2, pp. 217–288, May 2011.

[48] M. Gu and S.C. Eisenstat, “Efficient algorithms for computing a strong rank-revealing QR factorization,” *SIAM Journal on Scientific Computing*, vol. 17, no. 4, pp. 848–869, July 1996.

[49] IEEE 802.11ac, “IEEE standard for information technology-telecommunications and information exchange between systems-local and metropolitan area networks-Specific requirements-part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications-amendment 4: Enhancements for very high throughput for operation in bands below 6 GHz,” *IEEE Standards 802.11ac*, Dec. 2013.

[50] G.H. Golub and C.F. Van Loan, *Matrix Computations (4th Edition)*, Chapter 7, Johns Hopkins University Press, 2013. ISBN-13: 978-1421407944.

[51] N. Song, H. Sun and T. Yang, “Coordinated hybrid beamforming for millimeter wave multi-user massive MIMO systems,” in *Proc. of IEEE GLOBECOM*, pp. 1–6, Washington, DC, Dec. 2016.

[52] T.S. Rappaport, E. Ben-Dor, J.N. Murdock and Y. Qiao, “38 GHz and 60 GHz angle-dependent propagation for cellular & peer-to-peer wireless communications,” in *Proc. of IEEE ICC*, pp. 4568–4573, Ottawa, ON, June 2012.

[53] J. Blinn, “Consider the lowly 2 x 2 matrix,” *IEEE Computer Graphics and Applications*, vol. 16, no. 2, pp. 82–88, March 1996.

[54] G.H. Golub and C.F. Van Loan, *Matrix Computations (4th Edition)*, Chapter 5, Johns Hopkins University Press, 2013. ISBN-13: 978-1421407944.

[55] Nvidia, “NVIDIA TESLA V100 GPU architecture.” Accessed: Dec. 5, 2021. Available: https://images.nvidia.com/content/volta-architecture/pdf/volta-architecture-whitepaper.pdf

[56] Nvidia, “Optimizing parallel reduction in CUDA.” Accessed: Dec. 5, 2021. Available: https://developer.download.nvidia.com/assets/cuda/files/reduction.pdf

[57] Nvidia, “CUDA C programming guide v10.2.89.” Accessed: Dec. 5, 2021. Available: https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

[58] Y. Huang, Y. Chen, Y.T. Hou and W. Lou, “Achieving fair LTE/WiFi coexistence with real-time scheduling,” *IEEE Transactions on Cognitive Communications and Networking*, vol. 6, no. 1, pp. 366–380, March 2020.

[59] Y. Chen, Y. Wu, Y.T. Hou and W. Lou, “mCore: Achieving sub-millisecond scheduling for 5G MU-MIMO systems,” in *Proc. of IEEE INFOCOM*, May 2021.

[60] C. Husmann, G. Georgis, K. Nikitopoulos and K. Jamieson, “FlexCore: Massively parallel and flexible processing for large MIMO access points,” *USENIX Symposium on Networked Systems Design and Implementation*, pp. 197–211, Boston, MA, March 2017.

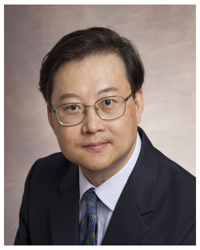
[61] G. Falcao, L. Sousa and V. Silva, “Massively LDPC decoding on multicore architectures,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 2, pp. 309–322, Feb. 2011.

[62] P. Hailes, L. Xu, R. G. Maunder, B. M. Al-Hashimi and L. Hanzo, “A survey of FPGA-based LDPC decoders,” *IEEE Communications Surveys & Tutorials*, vol. 18, no. 2, pp. 1098–1122, 2016.

Yongce Chen (M’21) is currently a Senior System Software Engineer at NVIDIA Corporation, Santa Clara, CA, USA. He received his Ph.D. degree in electrical engineering from Virginia Tech, Blacksburg, VA, USA, in 2021, and his B.S. and M.S. degrees in electrical engineering from Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2013 and 2016, respectively. During his Ph.D. study at Virginia Tech, he was awarded a VT Wireless Fellowship in 2016 and a Pratt Fellowship in 2021, respectively. He received the Best Paper Award in IEEE INFOCOM 2021. His current research interests include optimization, MIMO techniques and real-time implementation for wireless networks.

Yan Huang (M’21) is currently a Senior System Software Engineer at NVIDIA, Santa Clara, CA, USA. He received his Ph.D. degree in Electrical Engineering from Virginia Tech in 2020, and his B.S. and M.S. degrees in Electrical Engineering from Beijing University of Posts and Telecommunications in 2012 and 2015, respectively. During his Ph.D. study at Virginia Tech, he was awarded a Pratt Scholarship in 2019 and a Bindi Prasad Scholarship in 2020, respectively. He holds two U.S. and International patents. His research interests include GPU-based real-time optimization and machine learning for wireless communications and networking.

Chengzhang Li (S’17) is a Ph.D. student in the Bradley Department of Electrical and Computer Engineering at Virginia Tech, Blacksburg, VA, USA. He received his B.S. degree in Electronics Engineering from Tsinghua University, Beijing, China, in 2017. His current research interests are modeling, analysis and algorithm design for wireless networks, with a focus on Age of Information (AoI) and latency research.



Y. Thomas Hou (F'14) is Bradley Distinguished Professor of Electrical and Computer Engineering at Virginia Tech, Blacksburg, VA, USA, which he joined in 2002. He received his Ph.D. degree from NYU Tandon School of Engineering in 1998. During 1997 to 2002, he was a Member of Research Staff at Fujitsu Laboratories of America, Sunnyvale, CA, USA. Prof. Hou's current research focuses on developing innovative solutions to complex science and engineering problems arising from wireless and mobile networks. He is also interested in wireless security. He has published over 300 papers in IEEE/ACM journals and conferences. His papers were recognized by nine best paper awards from IEEE and ACM. He holds six U.S. patents. He authored/co-authored two graduate textbooks: *Applied Optimization Methods for Wireless Networks* (Cambridge University Press, 2014) and *Cognitive Radio Communications and Networks: Principles and Practices* (Academic Press/Elsevier, 2009). Prof. Hou was named an IEEE Fellow for contributions to modeling and optimization of wireless networks. He was/is on the editorial boards of a number of IEEE and ACM transactions and journals. He served as Steering Committee Chair of IEEE INFOCOM conference and was a member of the IEEE Communications Society Board of Governors. He was also a Distinguished Lecturer of the IEEE Communications Society.

Wenjing Lou (F'15) is the W.C. English Endowed Professor of Computer Science at Virginia Tech and a Fellow of the IEEE. She holds a Ph.D. in Electrical and Computer Engineering from the University of Florida. Her research interests cover many topics in the cybersecurity field, with her current research interest focusing on wireless networks, privacy protection in machine learning systems, and security and privacy problems in the Internet of Things (IoT) systems. Prof. Lou is a highly cited researcher by the Web of Science Group. She received the Virginia Tech Alumni Award for Research Excellence in 2018, which is the highest university level faculty research award. She received the INFOCOM Test-of-Time paper award in 2020. She is the TPC chair for IEEE INFOCOM 2019 and ACM WiSec 2020. She was the Steering Committee Chair for IEEE CNS conference from 2013 to 2020. She is currently a steering committee member of IEEE INFOCOM and IEEE Transactions on Mobile Computing. She served as a program director at the US National Science Foundation (NSF) from 2014 to 2017.