An Adaptable and Scalable Generator of Distributed Massive MIMO Baseband Processing Systems

Yue Dai1 · Maryam Eslami Rasekh2 · Seyed Hadi Mirfarshbafan3 · Harrison Liew1 · Alexandra Gallyas-Sanhueza4 · James Dunn1 · Upamanyu Madhow2 · Christoph Studer3 · Borivoje Nikolić1

Received: 4 January 2022 / Revised: 15 April 2022 / Accepted: 1 May 2022 / Published online: 17 May 2022 © The Author(s) 2022

Abstract

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

Keywords Hardware RTL generation · Massive MIMO · Baseband signal processing · Beamforming · FPGA-emulation · Beamspace

1 Introduction

With increasing media production and consumption and a growing number of mobile devices, wireless data demand is increasing exponentially over time. Simultaneously, point-to-point wireless systems are faced with a challenge of further increasing their spectral efficiency, since the development of the capacity-achieving codes and modulation schemes have pushed the spectral efficiency close to the theoretical Shannon limit [1]. With the advantages on energy-efficiency, robustness, and security, massive MIMO is deemed widely to be a practical approach for the channel capacity improvement and interference reduction in the next generation wireless systems [2, 3]. Theoretically, it can support many more users with their individual high-capacity and interference-free link.
to the BS, because the spatial resolution at the BS can be expanded with the help of the hundreds to thousands of antennas on the BS frontend [4].

One of the challenges of massive MIMO system implementation is the energy efficiency and design-cost-efficiency. Some of the state-of-the-art testbeds, LuMaMi for example [5], are implemented with centralized processing architectures, where data are collected from front-ends and processed centrally. However, centralized architectures require very high interconnection bandwidth, which leads to the router design complexity. For algorithm development purposes, other solutions, such as off-the-shelf software-define-radios (SDR) [5–7] and GPUs [8] have been explored. However, SDRs contain redundant blocks which may not be useful for the specific design, and general purpose processors are sub-optimal hardware for the specific algorithm. As a result, from the large-scale development perspective, they are neither design-cost-effective nor energy-effective. Hybrid analog-digital architectures, such as [9], consume less energy than digital beamforming architectures. However, compared to digital solutions which can be prototyped and validated quickly, analog solutions generally take longer time to design and evaluate. Additionally, recent results suggest that fully-digital architectures that use low-resolution data converters achieve higher spectral efficiency at similar power consumption compared to hybrid architectures [10, 11]. Implementing digital beamforming architectures in specialized hardware offers the highest energy-efficiency, and using modular and highly parameterizable hardware generators reduces design cost while enabling design reusability between application-specific integrated circuit (ASIC) and FPGA implementations of signal processing tasks [12–14].

As massive MIMO is being adopted in practical settings, new algorithms are being proposed to solve practical challenges and enhance performance. Some novel techniques are the beamspace domain algorithms [15, 16], the compressed-sensing based method [17], and machine learning algorithms [18, 19] for channel estimation performance improvement, channel prediction, and signal detection. As a result, massive MIMO system design needs to be flexible to adapt new algorithms for further development. By employing well-designed, modular generators, these types of algorithms can be physically realized with modest effort.

In this work, a design-effort-efficient and highly portable generator for a key baseband processing module in a scalable massive MIMO BS system is designed and emulated on FPGA. The generator enables real-time signal processing in a scalable massive MIMO system as shown in Fig. 1. The modular design of the generator makes this work an ideal system generator for rapid prototyping of emerging algorithms, showcased by implementing a low-overhead beamspace-domain channel combiner, which is the beamspace channel estimation (BEACHES), Then, a thorough evaluation of the generator’s performance, ease of hardware generation, and adaptability to new algorithms is presented.

The paper is organized in six sections. Section 2 gives the MIMO algorithm background of interest. The overview of the uplink baseband processing for our scalable massive MIMO system, and the Spine generator design is given in Sect. 3. The integration of the Python “golden” model simulator with the beamspace domain algorithms is detailed in Sect. 4. Section 4 also includes the setup and parameter settings for the generator FPGA emulation. The demonstration and performance analysis are in Sect. 5.

2 Background

2.1 Maximum-Ratio Combining (MRC) Beamforming

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

$$ r_k = h_k x_k + n_k $$

(1)

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

The maximum-ratio combiner architecture for a single isolated link is shown in Fig. 2, where $r_k = [r_{k,1}, \ldots, r_{k,M}]^T$ and $\hat{h}_k = [\hat{h}_{k,1}, \ldots, \hat{h}_{k,M}]^T$. Then the combiner output for user $k$ can be expressed as

$$ \hat{x}_k = \hat{h}_k^H r_k $$

$$ = \hat{h}_k^H h_k x_k + \hat{h}_k^H n_k $$

(2)

where $\hat{h}_k \in \mathbb{C}^{M \times 1}$ contains the combiner weights. According to Eq. 2, one can get the signal-to-noise ratio $\gamma_k$ of the combiner output as
\[
\gamma_k = \left( \frac{||x||^2 ||\hat{h}_k^H h_k||^2}{E(||\hat{h}_k^H h_k||^2)} \right) \frac{1}{\sigma^2 \hat{h}_k^H \hat{h}_k}
\]

Assuming the transmission power is \( P_x \), Eq. 3 can be expressed as
\[
\gamma_k = \frac{P_x \hat{h}_k^H h_k h_k^H \hat{h}_k}{\sigma^2 \hat{h}_k^H \hat{h}_k}
\]

Then the optimal weight that maximizes the SNR
\[
\hat{h}_k^* = \arg \max_{\hat{h}_k} \gamma_k
\]
can be obtained by calculating the zero point of the conjugate derivative with respect to \( \hat{h}_k \), which is \( \hat{h}_k^* \propto h_k \). As a result, for the MIMO system with \( K \) single antenna users and \( M \) antennas at BS, the optimal weights \( \hat{h}_k^* \) that maximize each user’s SNR in the isolated link are:
\[
\hat{h}_k = h_k
\]
where \( H \in \mathbb{C}^{M \times K} \) and \( H = \{ h_1, ..., h_\ell \} \). Note, however, that during payload transmission, when users are simultaneously transmitting, there is correlation between received channels. This is not captured during MRC channel estimation, resulting in residual inter-user interference and reduced SNR after MRC beamforming alone.

### 2.2 Golay Sequence

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

The Golay sequence contains a complementary pair of sequences with the same length \( n \). According to [20], given \( A = [a_1, ..., a_n] \) and \( B = [b_1, ..., b_n] \) of length \( n \), with \( a_k, b_k \in \{-1, +1\} \) for \( 1 \leq k \leq n \), the complementary sequence pair \( (A, B) \) satisfies
\[
A[z]A[z^{-1}] + B[z]B[z^{-1}] = 2n
\]
where \( A[z] = \sum_{i=0}^{n-1} a_{i+1} z^{-i} \) and \( B[z] = \sum_{i=0}^{n-1} b_{i+1} z^{-i} \) for \( z \in \mathbb{C} \).

Assume the channel variation is negligible during the transmission of a sequence pair. The received Golay pair at BS with \( M \) antennas and spatial white Gaussian noise is \( r_A[z] = h_A[z] + n_A \)
\[
r_B[z] = h_B[z] + n_B
\]
where \( h \in \mathbb{C}^{M \times 1} \) is the channel vector, and \( n_A, n_B \sim \mathcal{CN}(0, \sigma^2 I_M) \) is the spatial white Gaussian noise vector. Using Eq. 7, one can get
\[
E\left\{ \frac{1}{2n} (r_A[z]z[-1] + r_B[z]z[-1]) \right\} = h
\]
which shows that the sum of the cross-correlation of the sequence pair is an unbiased estimator of the channel vector.

According to Theorem 1.3.2 in [21], Golay sequences can be generated recursively when \( n \) is a positive integer power of 2, and the reverse of a Golay sequence pair still gives a Golay sequence pair. Assume the length of the sequence pair is \( n \), the delayed value sequence \( D \) is any permutation of the set \( \{0, 1, 2, ..., 2^{\log_2(n)}-1\} \), and the weight sequence \( W = [w_1, ..., w_{\log_2(n)}] \) with \( w_i \in \{-1, +1\} \) \( \forall 1 \leq i \leq \log_2(n) \). Then, the concatenation of the Golay sequence pair \( g \) can be generated using Algorithm 1.

#### Algorithm 1: Golay sequence pair generation

**Input:** \( n, D, W \)

**Output:** \( g \)

1. Create two empty arrays \( g_A \) and \( g_B \) of size \( \log_2(n)+1 \) by \( n \)
2. \( g_A[0, 0] \leftarrow 0, g_B[0, 0] \leftarrow 0 \)
3. for \( 0 \leq k < n \) do
4.  for \( 0 \leq i < \log_2(n) \) do
5.    if \( k \geq D[i] \) then
6.      \( g_A[i + 1, k] \leftarrow W[i]g_A[i, k] + g_B[i, k - D[i]] \)
7.    else
8.      \( g_B[i + 1, k] \leftarrow W[i]g_A[i, k] - g_B[i, k - D[i]] \)
9.    end if
10.  end for
11. end for
12. return \( g \)

However, the performance of the channel estimation from Golay sequence pairs may be affected by the quantization noise introduced by the hardware design.
2.3 BEACHES Algorithm

BEACHES is an adaptive and low-complexity channel estimation algorithm for massive MIMO systems [15]. It exploits sparsity of channels in the beamspace domain in order to perform adaptive denoising via Stein’s unbiased risk estimate (SURE). Let $\mathbf{h}_{est}$ be each user’s estimated channel vector. To transfer the channel into beamspace domain, the discrete Fourier transform (DFT) is performed on the channel vector with a $M \times M$ DFT matrix $\mathbf{F}$ to get $\mathbf{h}_{est} = \mathbf{F} \mathbf{h}_{est}$ in BEACHES. The soft-thresholding is performed on each entry of $\mathbf{h}_{est}$ by the MSE-optimal threshold $\tau^*$. BEACHES finds $\tau^*$ by relying on a proxy for the actual MSE between the ground-truth channel and the denoised channel estimate, and the proxy is obtained by SURE. Then the beamspace denoised channel vector is converted back to the antenna domain using an inverse DFT, after the magnitudes of the entries in $\mathbf{h}_{est}$ are soft-thresholded.

2.4 In-situ Calibration

Manufacturing variability affects circuits at small wavelengths, as small variations in RF circuitry and local oscillator distribution paths can cause large, often unpredictable changes in the RF chain response of different array elements. For beamspace algorithms, the sparsity of the beamspace domain channel vectors is very important. However, due to the array nonidealities, which stem from gain and phase offsets in receivers, the channel estimation from pilots may not be sparse. To extract and calibrate out those array offsets, compressive techniques can be applied, and the mm-wave channel vectors is very important. However, due to the array nonidealities, which stem from gain and phase offsets in receivers, the channel estimation from pilots may not be sparse. To extract and calibrate out those array offsets, compressive techniques can be applied, and the mm-wave channel vector can be leveraged [22, 23].

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

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

3 System Overview

The BS system architecture is shown in Fig. 1, and includes three types of modules: Head, Spine and Tail. The Head modules consist of the mm-wave frontend radios including antennas and mixers that downconvert the signal to baseband. Each Spine module is implemented with a sub-array MRC beamformer, and a daisy chain architecture connects the Spine modules together. The Tail module removes the inter-user and inter-symbol interference by the frequency-selective decorrelation.

3.1 Two-stage Beamforming

This two-stage beamforming system has been proposed in [24] to strengthen both the scalability and energy efficiency of the signal processing.

1) Distributed MRC Beamformer: The first stage in this architecture is the frequency-flat MRC beamformer which maximizes the output signal-to-noise-ratio (SNR) theoretically and is implemented in the Spine. From the time-interleaved Golay sequence pilots transmitted from users, the sub-array MRC channel matrix can be estimated. Assume the same multi-user MIMO (MU-MIMO) system described in Sect. 2. The received signal at BS is

$$\mathbf{r}_{BS} = \mathbf{Hx} + \mathbf{n}$$

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

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

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

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

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

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

2) Frequency-selective Decorrelation: The second stage performs the frequency-selective decorrelation, which is implemented in the Tail. After MRC beamforming, the inter-symbol and inter-user interference still exist in the wideband-channel signal. To remove them, zero-forcing (ZF) is used for each narrow subcarrier group. In the Tail, carrier frequency offset (CFO) is estimated with orthogonal frequency division multiplexing (OFDM) pilots. Subcarriers are grouped to expand CFO estimation range [25]. The ZF weights can be obtained from the OFDM pilots, which is time-interleaved and follows the Golay pilots. To perform the frequency selective ZF on payloads, the quadrature amplitude modulated (QAM) payloads are divided into blocks, whose lengths...
are same as each OFDM pilot block length $L_{\text{OFDM}}$. Let $\hat{s}_{\text{MRC}}[i] = \text{FFT}(\hat{s}_{\text{MRC}})[i]$ be the $i$-th sample of the $p$-th payload block $\hat{s}_{\text{MRC}}$ in the frequency domain, and in the $q$-th subcarrier group, the zero-forcing estimator in the spatial domain gives
\[
\hat{x}[p \times L_{\text{OFDM}} + i] = I^{\text{FFT}}(\hat{X}_{\text{MRC}})[i]
\]
where $\hat{X}_{\text{MRC}}[i] = \hat{H}_{\text{ZF}}^H(\hat{H}_{\text{ZF}} \hat{H}_{\text{ZF}}^H)^{-1}\hat{s}_{\text{MRC}}[i]$, and $\hat{H}_{\text{ZF}} \in \mathbb{C}^{K \times K}$ is the estimated channel matrix for $q$-th subcarrier group.

This two-stage beamforming method reduces the interuser interference and increases the signal-to-interference-plus-noise ratio (SINR) in a distributed fashion. In addition, compared with the traditional centralized beamforming architecture, the two-stage distributed beamforming architecture relaxes the interconnection bandwidth. Suppose the MU-MIMO system with total number of $M$ BS antennas and $K$ single antenna users implemented with the signal bandwidth $W$, oversampling rate at BS receiver $s$, and analog-to-digital converter (ADC) resolution $d$, then the interconnection bandwidth for the traditional centralized architecture is
\[
\text{Bandwidth}_{\text{central}} \geq MW_s sd \text{ bits/s } \propto M
\]
which is proportional to the number of BS antennas. This proportionality limits the scalability of the traditional centralized BS architecture. However, with the help of the two-stage beamforming architecture, the interconnection bandwidth between the first stage and the second stage reduces to
\[
\text{Bandwidth}_{\text{2-stage}} \geq KW_s d \text{ bits/s } \propto K
\]
which is proportional to the number of users in the system, and independent of the number of antennas at BS. The relaxation on the interconnection bandwidth enables the scalability of this work. The reduction of the data movement also improves the energy efficiency.

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

### 3.2 Signal Packet Format

Figure 3 shows the individual user’s signal packet and the uplink signal received at each antenna in BS. The time-interleaved Golay sequence pilots and the binary phase-shift keying (BPSK) modulated OFDM pilots, sent by each user, help with the channel estimation in the two-stage beamformer. Both pilots include guard intervals. The payloads are chosen to be QAM to allow for larger CFO range and phase noise caused by hardware imperfections, and each payload block comprised the same structure as OFDM, except for the OFDM symbols. All users send the payload simultaneously after pilots are sent. A beacon signal is propagated to all users and BS at the beginning of each packet as the timing reference.

### 3.3 The Spine Generator

A major challenge for the design of moderate-volume ASICs are the large design costs stemming from the engineering design and verifications efforts, as well as for the software development, associated with chip designs and respins [13]. Design costs can be reduced through more effective reuse, which can be achieved by employing generators for the logic design and its physical implementation. Generators employ higher-level programming constructs to create modular, highly parameterizable RTL hardware designs and Very Large Scale Integration (VLSI) flows [26], enabling design reuse via generator extension. Each unique design instance is a result of generator parameterization.

The digital signal processing (DSP) hardware in this work is generated by using Chisel, a hardware construction language that facilitates parameterized circuit generation for both ASIC and FPGA digital logic designs [27]. Chisel is a domain-specific extension of the Scala programming language, which abstracts primitive hardware components, generates the corresponding circuit graph, and translates the graph into the synthesizable Verilog representation [13]. The Spine generator in this work enables design reusability as well as platform and technology portability.
The Spine generator generates the MRC beamformer in the BS architecture. The parameterization of the generator is on the subarray antenna number, the system user number, the signal oversampling rate, the Golay pilot generation, the number of root-raised cosine (RRC) taps, the Lagrange polynomial order, the datapath bitwidth, and the datapath parallelization. Figure 4 shows the Spine generator datapath. It consists of multiple DSP generators implemented using ACED [28], a reusable Chisel DSP library. The Spine generator is also extended to enable signal processing with beamspace algorithms.

The inputs of the Spine DSP are $M/N$ antennas of 2x oversampled in-phase and quadrature (IQ) signals. Datapath parallelism is determined in the generator configuration at the configuration time. The final outputs are the $K$ users’ MRC beamformed signals summed with the signal coming from the previous neighboring Spine in the daisy chain and sent to the next neighboring Spine. Figure 4 shows the Spine generator datapath. The parameters of each module generator are shown in the light blue boxes.

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

2) MRC Channel Estimation: Section 2 mentions that the cross-correlation of the Golay sequence pilots is an unbiased estimator for the optimal MRC weights. The pipelined parallelizable Golay correlator generator is designed based on the Algorithm 1. Because the reverse of a Golay sequence pair is also a Golay sequence pair, the Golay correlator is equivalent to the Golay generator of the reverse sequence. Compared with the design in [29], the proposed design is pipelined and parallelized, to achieve a higher throughput. The datapath is pipelined by $\log_2(n)$ stages. The datapath parallelism determines the input of each multiply-add (MA) unit shown in Fig. 5 and the structure of the delay registers. The MA input can be either from the delay registers or the pipeline registers. This correlator generator is parameterizable for $D$, and runtime configurable for $W$. The correct correlation is guaranteed by switching the input of $W$ at the corresponding user’s time-interval controlled by the sequencing controller. Figure 5 shows the generated architecture of an oversampling-2, length-4 Golay correlator with parallelism of 4.

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

$$\hat{s}_0 = \arg \max_x \mathcal{L}(x) = \arg \max_x \sum_{i=-l}^{l} c(i) \prod_{j \neq i \in \{-l, ..., 0, ..., l\}} \frac{x-j}{x-i}.$$

Figure 4 shows the Spine generator datapath.
Let $y(t)$ be the skewed signal and $\hat{y}(t)$ be the deskewed signal, which is given by

$$\hat{y}(t) = \sum_{i=-1}^{i} y(t + i) \prod_{i,j \neq i} \hat{y}_{0}-j. \quad (17)$$

3) **MRC Beamformer**: The MRC beamformer is generated by the pipelined systolic-array-based weight-stationary matrix multiplier generator, and is parameterized on the matrix dimension and datapath parallelization. The architecture of the beamformer is shown in Fig. 6. To satisfy different timing requirements for different FPGA platforms and transistor process nodes, the number of pipeline stages between PE columns is also parameterized.

4) **Sequencing Controller**: To guarantee the performance of the MRC beamforming, all channels need to be synchronized. The sequencing controller provides the time reference for all processing modules, and helps with channel synchronization. The channel delay estimator is the average of the time delay among all users signal received by the same channel. This estimator is unbiased, which can be shown by

$$E\left[\frac{1}{K} \sum_{j=1}^{K} (T_{c,j} + T_{n,j}) \right] = T_{c,j} + \mu_n. \quad (18)$$

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

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

### 3.4 Implementation of the Beam-space Algorithms

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

### 4 Evaluation

#### 4.1 Simulator

Massive MIMO system is modeled in a Python-based simulator. The end-to-end simulation encompasses terminals, channel, and the BS simulations, including signal packet generations, various channel and transceiver front-end models, and the modular signal processing chain. The modular signal processing that simulates the BS can encompass any instance from the Chisel generator described in Sect. 3 by generating raw samples and capturing and post-processing the results, which is illustrated in Fig. 7 by dashed arrows. The modular design enables the adaptability of the system to evaluate different processing algorithms. The calibration and BEACHES algorithms are implemented in MATLAB, and integrated into the simulator using a MATLAB-to-Python API, Matlab Engine API for Python [30]. The Spine generator is verified against this “golden” model simulator.

Figure 7 shows the system data flow. The solid rectangles represent Python functions, and the dashed rectangles represent MATLAB functions. After system parameters are set, each user’s signal packet is generated according to the system settings, which includes pilots lengths, guard interval lengths, payload modulation schemes, etc. The channel modeling in the Python simulator includes the flat i.i.d channel, the LoS channel, and Rician channel. The QuaDRiGa mmMAGIC urban micro line-of-sight (QuadMMLoS) channel is implemented in MATLAB, and can be called via the API with the corresponding channel
model setting. QuaDRiGa channel model is an enhancement of the WINNER model following a geometry based, stochastic channel modeling approach for MIMO [31]. The generated QuaDRiGa channel matrix is returned to Python for further signal processing. The channel simulation contains transceiver front-end simulation and noise addition.

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

### 4.2 FPGA Emulation

The Spine generator output is mapped to a Xilinx VCU118 FPGA. The FPGA instance contains a Spine DSP core, a memory system, and clock generators for the two clock domains. The two clock domains are memory clock and core clock. A server connects to the FPGA via JTAG for data transfer and FPGA programming. As shown in Table 1, a 64 antennas and 4 users BS system is emulated. The FPGA emulation system architecture is shown in Fig. 8.

Before the FPGA emulation starts, the Python simulator first generates the user packets and simulates the channel. Then the floating-point emulation input data is converted to fixed point data format by the simulator. At the same time, the simulator generates the configuration and control Tcl files to help the transmission of the emulation data to the on-board DDR4 memory. When the Spine DSP core processes the data, the data are also streamed from and to the memory. After the FPGA finishes beamforming, the data are sent back to the server through JTAG, and reformatted to the floating point data. Meanwhile, the server loads the estimated MRC

**Table 1** FPGA emulation system parameters.

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

**Figure 7** The system flow diagram, with software and hardware integration.

**Figure 8** The emulation system architecture.
beamformer weights from FPGA to be processed for the beamspace algorithms.

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

5 Results

5.1 Performance Evaluation of the Spine Generator

The performance of the Spine generator is evaluated under flat i.i.d, Rician channel with $K = 10$dB, LoS, and QuadMMLoS channel models with 4 users and 64 channels in the system. The FPGA performs the MRC beamformer, and the Python model simulator performs the beamspace algorithms and the frequency selective decorrelation stage.

The SINR vs. SNR with various channel models after demodulation are shown in Fig. 9. Results of Python simulation are represented with solid lines, and results of FPGA emulation are represented with dashed lines. The results demonstrate the functionality of the system for the four selected channel models, and validate the given generator parameters. Moreover, the emulation results are close to the corresponding floating-point simulation results for flat i.i.d, Rician, and LoS channel models in the high SNR range. The small performance difference between simulation and emulation under QuadMMLoS is due to the gain difference for users included in the channel model, which makes the demodulation of this channel model more sensitive to the channel estimation accuracy and quantization errors.

Figure 9 shows the linear change of the SINR with respect to that of SNR for the given channel models, when SNR is larger than 12dB. However, when SNR is lower than 12dB for flat i.i.d and Rician channel models, and 11dB for LoS and QuadMMLoS channel models, the SINR after demodulation drops abruptly. The performance degradation is due to the degradation of the channel estimation at low SNR range. Here, residual noise after cross correlation of the Golay sequence pilots might false trigger the peak detection algorithm for some channels, which leads to a completely incorrect channel estimation for those channels. The larger SINR difference between simulation and emulation under QuadMMLoS is due to shorter peak checking interval implemented in the hardware due to timing and hardware resource considerations, which leads to higher false peak-trigger possibility when SNR is low.

5.2 Channel Estimation Performance Evaluation

Figure 10 shows the MRC channel estimation NMSE with conventional estimation and with the BEACHES algorithms with respect to SNR under the Rician, LoS, and QuadMMLoS channel models. In this evaluation, channels are assumed to be ideal and no phase offsets between channels. The NMSE is the
error measurement between the channel matrix estimation and
the simulator generated ground-truth. The Golay pilot length
is set to as 64 due to its good channel estimation performance
evaluated in [32]. For all insets, the NMSE of the FPGA-
emulated channel matrix is shown as the dashed line, while
the NMSE of the denoised one is shown as the solid line. Due
to the randomness property of the flat i.i.d channel model, the
denoising algorithm is not applicable. For this reason, the flat
i.i.d channel is not considered in this section.

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

Figure 10 also shows that as SNR decreases, the improve-
ment of the NMSE after denoising by BEACHES algorithm
increases, demonstrating algorithm’s efficacy in improving
the channel estimation performance in the SNR range of
interest. With BEACHES algorithm, the channel estimation
NMSE can be improved by up to 25.3% for the Rician chan-
nel model, 26.2% for the LoS channel model, and 21.6% for
the QuadMMLoS channel model.

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

Figure 11 shows the EVM vs. SNR with conventional and
with the BEACHES algorithm for the three channel models.
The solid lines in the figure show the EVM after demodulation

Figure 10 Channel estimation NMSE with and without calibration and BEACHES algorithms. For all insets, the NMSE of the FPGA-emulated
channel matrix is shown as the dash line, while the NMSE of the denoised one is shown as the solid line.

Figure 11 EVM with and without the beamspace algorithms.
without applying BEACHES algorithm, while the dash-dot lines indicates the EVM of the demodulation after applying BEACHES algorithm. For all models, the EVM decreases after applying the BEACHES algorithm. With SNR higher than 15dB for Rician and QuadMMLoS channel models, and with SNR higher than 14dB for LoS channel model, the EVM with and without the BEACHES algorithm are almost same, and the improvement of the channel estimation MSE has only small impact to the system; when the SNR is lower than 15dB and 14dB, respectively, the EVM decreases notably after applying the BEACHES algorithm. The EVM improvement is more pronounced for lower SNR than high SNR. The improvement of the EVM after the calibration and BEACHES algorithms can be up to 23.3% for the Rician channel, 29.8% for the LoS channel, and 20.4% for QuadMMLoS channel model.

The channel estimation evaluation on the NMSE and the EVM after demodulation proves the effectiveness of BEACHES algorithm to the conventional massive MIMO processing. The necessity of integrating BEACHES algorithm to the system further indicates the importance of the adaptability of massive MIMO systems to other algorithms.

5.3 Impact of Channel Phase Offsets

In this section, the impact of channel phase offsets to BEACHES and the importance of the in-situ calibration algorithm are evaluated. In this section, a random phase offset $\Delta \theta \in [-\pi, \pi]$ is applied to each antenna in channel simulations to model mismatches in the delay of the RF chain between antennas, and LoS channel model is considered. Figure 12 shows denoised FPGA emulated channel matrix NMSE with and without applying in-situ calibration when per-channel phase offsets are modeled. The result shows that without channel calibration, the NMSE stays at 1.7 regardless of the change of SNRs. This shows that uncalibrated channel phase offsets break the beamspace sparsity of the channel, rendering the beamspace channel denoising algorithms useless. This emphasizes the need for channel calibration for successful beamspace channel denoising.

Figure 13 shows the MSE between the estimated channel phase offset by in-situ calibration algorithm and the ground truth with SNR ranging from 9dB to 16dB. Though the MSE increases as SNR decreases, the MSE is less than 2%. The solid line in Fig. 12 shows the NMSE of the denoised FPGA emulated channel matrix with both in-situ calibration and BEACHES applied. The NMSE drops from 1.7 to the order of magnitude of $10^{-2}$, which agrees with the result in Fig. 10 inset b), where ideal channel are assumed. The results indicate the effectiveness and importance of the in-situ calibration for beamspace algorithms.

5.4 Ease of Spine Generation

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

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

Figure 14 shows two Spine DSP core implementations with different system parameter settings. Inset a) is the
FPGA implementation with 4 antennas and 2 users, while inset b) is the implementation with 4 antennas and 4 users. The Spine generator can generate the RTL for those 2 implementations by simply changing the generator parameters. Compared with designing instances and redesigning the top level integration, which takes long time, the Spine generator generates the new RTL for different parameters in just 9 minutes and 17 seconds. The fast generation of new designs shows the high portability of the Spine generator. The power consumption estimation for the 4 antennas and 4 users implementation given by Xilinx power estimator [33] is 5.4W, with 3.1W dynamic power and 2.3W static power.

5.5 Discussion on Scalability Limitations

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

1) Interconnection Bandwidth: The QSFP interface on VCU118 has maximum 112Gbps (4 × 28 Gbps) bandwidth. With the implementation parameters shown in Table 1, and by applying Eq. 15, the maximum number of users the system can support using Spine real-time processing is 28, assuming 8b/10b encoding and without considering the hardware utilization limitation.

2) Interconnection Delay Along the Daisy Chain: This work uses a daisy chain architecture to sum up all Spine results together, so the data transmission latency between Spine modules limits the number of modules that can be chained together. Assume the interconnection latency between 2 Spines is 100ns, including the transmission overhead, and assume each Spine has a 2K Byte buffer used for buffering the data coming from the upper stream Spine. By using the parameters in Table 1, the total number of Spines that the system can support for real-time operation is 50, which means the system can scale up to 200 antennas.

3) FPGA Utilization: Figure 15 shows that the bottleneck of the FPGA resources on VCU118 is the DSPs. The design with 4 antennas and 4 users utilized 53% of DSP resources. Signal correction blocks take 43.7% of DSP utilization and the MRC beamformer takes 5.6%. Although further increasing the antenna count in the Spine is hard, scaling up the number of users is possible, since it mainly affects the beamformer utilization instead of the signal correction utilization. Based on Fig. 15, the system can support up to 28 users and 4 antennas, if the FPGA is fully utilized.

5.6 Summary

Table 2 summarizes our work and some state-of-the-art massive MIMO BS implementations. ArgosV2 [6] can scale up to 96 antennas and support 32 users, which is implemented with a distributed architecture and the conjugate beamforming method. LuMaMi [5] is a BS testbed which supports up to 100 antennas and 12 users with a centralized architecture, and it is implemented with MRC, ZF, and regularized ZF (RZF) beamforming methods. Both ArgosV2 and LuMaMi use OFDM modulation with signal bandwidth 20MHz. The work in [34] implements a successive over relaxation (SOR) based beamforming algorithm with the assumption of perfect channel estimations on FPGA. The hardware implementation contains 64 antennas and 8 users with only simulation and synthesis results reported. Compared with [5] and [34], our work currently only contains uplink baseband processing, but is highly portable and parameterizable, and can be implemented either on FPGAs or as ASICS. The scalability of the system is enabled by the distributed architecture, and the extension for emerging algorithm implementation is enabled by the modular design. This work has also been tested in a massive MIMO system with
more antenna and users, and evaluated under various channel models than with our earlier works [32] and [35]. The efficient implementation of the beamspace algorithms demonstrates the algorithmic adaptability of this work. The fully utilized FPGA power estimation per user per antenna for this work is 0.076W/user/antenna. In addition, due to the portability of this work, the generator can be potentially implemented in an ASIC, yielding more antennas and users in a single chip, along with better energy efficiency.

### 6 Conclusion and Future Works

This work shows the design and FPGA emulation of the Spine generator integrated with beamspace algorithms. The system encompasses scalability, portability, and algorithmic-adaptability. The parameterization of the generator includes the MIMO system and the hardware datapath. Meanwhile, the FPGA emulation is performed on specific instances. The performance of the system with and without the integration of beamspace algorithms is evaluated on the SINR, channel estimation NMSE, and EVM, using various channel models. The results show the improvement of the channel estimation with the in-situ calibration and BEACHES algorithms. The successful implementation of the beamspace channel estimation algorithms also proves the high adaptability of this work.

Currently, our massive MIMO system only includes the uplink processing, and the downlink processing is still under development. The next steps for this work involve implementing downlink and hardware integration of the Tail and the beamspace channel estimator, which are currently implemented in software. This will complete the system supporting end-to-end real-time signal processing, for the full design evaluation.

Moreover, extending the system to support OFDM payloads and reducing pilot latency incorporated with the improvement of other parts of the system design are also one of the next steps. In addition, machine learning (ML) methods for channel estimation improvement with less pilot overhead attract more and more attention in the massive MIMO research. The next steps for this work also involve adding ML engines to the generator to enable ML algorithms on this massive MIMO system.

### Author Contributions

All authors contributed to the study conception and design. System implementation was performed by Yue Dai, Maryam Eslami Rasekh, and Seyed Hadi Mirfarshbafan. Data collection and analysis were performed by Yue Dai. The first draft of the manuscript was written by Yue Dai and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

### Funding

This work was supported in part by tasks 2778.026 and 2778.007 of the ComSenTer Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA, and in part by NSF EARS award 1642920.

### Data Availability

The data that support the findings of this study are available on request from the corresponding author.

### Declarations

**Ethics Approval** This work doesn’t involve humans and/or animals.

**Competing Interest** The authors have no relevant financial or non-financial interests to disclose. The authors have no competing interests to declare that are relevant to the content of this article. All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript. The authors have no financial or proprietary interests in any material discussed in this article.

---

**Table 2 Comparison.**

|                  | This work | Prev. work [32] | ArgosV2 [6] | LuMaMi [5] | RIVF’19 [34] |
|------------------|-----------|-----------------|-------------|------------|-------------|
| Platform         | FPGA/ASIC | FPGA/ASIC       | SDR         | FPGA       | FPGA        |
| Portable         | Yes       | Yes             | No          | No         | No          |
| Adaptability     | High      | High            | -           | -          | -           |
| # Users          | 4         | 2               | 32          | 12         | 8           |
| # Antennas       | 64        | 32              | 96          | 100        | 64          |
| Modulation       | QAM       | QAM             | OFDM        | OFDM       | QAM         |
| Distributed      | Yes       | Yes             | Yes         | No         | No          |
| Bandwidth        | 200MHz\(^a\) | 200MHz         | 20MHz       | 20MHz      | 935MHz\(^b\) |
| Method           | Two-stage BF w/ BEACHES | Two-stage BF | Conjugate BF | MRC/ZF/RZF | SOR         |
| CHEST\(^c\)      | Yes       | Yes             | Yes         | Yes        | No          |

\(^a\)The signal bandwidth is the FPGA implementation bandwidth. The power estimation are based on the FPGA implementation of a single Spine with parameters shown in Table 1

\(^b\)This is the maximum FPGA implementation clock frequency, not the signal bandwidth. The signal bandwidth is not given

\(^c\)Short for channel estimation
Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Goldsmith, A. (2014). 5G and beyond: What lies ahead for wireless system design. In PIMRC.
2. Larsson, E. G., & et al. (2014). Massive MIMO for next generation wireless systems. IEEE Communications Magazine, 52(2), 186–195. https://doi.org/10.1109/MCOM.2014.6736761
3. Rusek, F., et al. (2013). Scaling up MIMO: Opportunities and challenges with very large arrays. IEEE Signal Processing Magazine, 30(1), 40–60. https://doi.org/10.1109/MSP.2011.2178495
4. Marzetta, T. L. (2010). Noncooperative cellular wireless with unlimited numbers of base station antennas. IEEE Transactions on Wireless Communications, 9(11), 3590–3600. https://doi.org/10.1109/TWC.2010.092810.1019022
5. Malkowsky, S., et al. (2017). The world’s first real-time testbed for massive MIMO: Design, implementation, and validation. IEEE Access, 5, 9073–9088. https://doi.org/10.1109/ACCESS.2017.7205561
6. Shepard, C., Yu, H., & Zhong. L. (2013). ArgosV2: A flexible many-antenna research platform. MobiCom, 163–166. https://doi.org/10.1145/2500423.2505302
7. 5G massive MIMO testbed: From theory to reality. White Paper, National Instruments, Aug. 2021.
8. Li, K., & et al. (2017). Decentralized baseband processing for massive MU-MIMO systems. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 7(4), 491–507. https://doi.org/10.1109/JETCAS.2017.2775151
9. Naviasky, E., Iotti, L., LaCaille, G., Nikolić, B., Alon, E., & Niknejad, A. M. (2021). A 71-to-86-GHz 16-element by 16-beam multi-user beamforming integrated receiver sub-array for massive MIMO. IEEE Journal of Solid-State Circuits, 56(12), 3811–3826. https://doi.org/10.1109/JSSC.2021.3118641
10. Yan, H., Ramesh, S. Gallagher, T., Ling, C., & Cabric, D. (2019). Performance, power, and area design trade-offs in millimeter-wave transmitter beamforming architectures. IEEE Circuits and Systems Magazine, 19(2), 33–58.
11. Skrpinonis, P., Dutta, S., Mezzavilla, M., Rangan, S., Mirfakrshaban, S. H., Studer, C., Buckwalter, J., & Rodwell, M. (2020). Power consumption analysis for mobile mmWave and sub-THz receivers. In 2020 IEEE 6G Wireless Summit (pp. 1–5).
12. Shacham, O., et al. (2010). Rethinking digital design: Why design must change. IEEE Micro, 30(6), 9–24. https://doi.org/10.1109/MM.2010.81
13. Nikolic, B., et al. (2018). Generating the next wave of custom silicon. In IEEE European Solid-State Circuits Conference (pp. 6–11). https://doi.org/10.1109/ESSCIRC.2018.8494310
14. Nikolić, B. (2015). Simpler, more efficient design. In IEEE European Solid-State Circuits Conference. (pp. 20–25). https://doi.org/10.1109/ESSCIRC.2015.7313819
15. Mirfakrshaban, S. H., Gallays-Sanhuzea, A., Ghods, R., & Studer, C. (2020). Beamspace channel estimation for massive MIMO mmWave systems: Algorithm and VLSI design. IEEE Transactions on Circuits and Systems I: Regular Papers, 67(12), 5482–5495. https://doi.org/10.1109/TCASI.2020.3023023
16. Abdelghany, M., Madhow, U., & Tölli, A. (2019). Beamspace local LMMSE: An efficient digital backend for mmWave massive MIMO. In IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC) (pp. 1–5). https://doi.org/10.1109/SPAWC.2019.8815585
17. Kong, W., Li, H., Song, S., Fan, Y., & Zhang, W. (2018). Compressive sensing based channel estimation for MIMO-OFDM systems. In IEEE Conference on Industrial Electronics and Applications (ICIEA) (pp. 2164–2169). https://doi.org/10.1109/ICIEA.2018.8398069
18. Alaraby, M., & Alkhateeb, A. (2019). Deep learning for TDD and FDD massive MIMO: Mapping channels in space and frequency. IJ, 1465–1470. https://doi.org/10.1016/j.ijrfcom.2019.9048929
19. Kim, S., Chae, J., & Hong, S.-N. (2020). Machine learning detectors for MU-MIMO systems with one-bit ADCs. IEEE Access, 8, 86608–86616. https://doi.org/10.1109/ACCESS.2020.2987212
20. Golay, M. (1961). Complementary series. IRE Transactions on Information Theory, 7(2), 82–87. https://doi.org/10.1109/TIT.1961.1057620
21. Kalashnikov, E. (2014). An introduction to Golay complementary sequences. Eureka,4(1), 40–48. https://doi.org/10.29173/eureka22829
22. Rasekh, M., Puranik, B., Madhow, U., & Rodwell, M. (2021). In-the-field calibration of all-digital MIMO arrays.
23. Marzi, Z., Ramasamy, D., & Madhow, U. (2016). Compressive channel estimation and tracking for large arrays in mm-wave picocells. IEEE Journal of Selected Topics in Signal Processing, 10(3), 514–527. https://doi.org/10.1109/JSTSP.2019.8815585
24. LaCaille, G., et al. (2020). Design and demonstration of a scalable massive MIMO uplink at E-band. In 2020 IEEE International Conference on Communications Workshops (pp. 1–6). https://doi.org/10.1109/ICCWorkshops49005.2020.9145323
25. Nishad, P. K., & Singh, P. (2013). Carrier frequency offset estimation in OFDM systems. In IEEE Conference on Information Communication Technologies (vol. 2013, pp. 885–889).
26. Kung, H. (1979) Let’s design algorithms for VLSI systems. In Caltech Conference On Very Large Scale Integration (pp. 82–87).
27. Chisel3. https://github.com/chipalliance/chisel3. Accessed on 2020-02-20.
28. Wang, A., et al. (2018). ACED: A hardware library for generating DSP systems. In Design Automation Conference (DAC). https://doi.org/10.1109/DAC.2018.8465790
29. García, E., et al. (2014). Efficient filter for the generation/correlation of Golay binary sequence pairs. International Journal of Circuit Theory and Applications, 42(10), 1006–1015. https://doi.org/10.1002/cta.1901
30. MATLAB engine API for Python. https://www.mathworks.com/help/matlab/matlab-engine-for-python.html. Accessed on 2021–06–19.
31. Burkhart, F., Jaechel, S., Eberlein, E., & Prieto-Cerdeira, R. (2014). QuadDiGa: A MIMO channel model for land mobile satellite. In European Conference on Antennas and Propagation (EuCAP) (pp. 1274–1278). https://doi.org/10.1109/EuCAP.2014.6902008
32. Dai, Y., et al. (2021). A scalable massive MIMO uplink baseband processing generator. In 2020 IEEE International Conference on Communications (pp. 1–6). https://doi.org/10.1109/ICC4297.2021.9500566
33. Vivado design suite user guide: Power analysis and optimization. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_1/ug907-vivado-power-analysis-optimization.pdf. Accessed on 2021–12–29.
34. Nhat Cuong, C., et al. (2019). Hardware implementation of the efficient SOR-based massive MIMO detection for uplink. In 2019 IEEE-RIVF International Conference on Computing and Communication Technologies (pp. 1–6). https://doi.org/10.1109/RIVF.2019.8713667
Dai, Y., Liew, H., Rasekh, M. E., Mirfarshbafan, S. H., Gallyas-Sanhueza, A., Dunn, J., Madhow, U., Studer, C., & Nikolič, B. (2021). A scalable generator for massive MIMO baseband processing systems with beamspace channel estimation. In 2021 IEEE Workshop on Signal Processing Systems (SiPS) (pp. 182–187). https://doi.org/10.1109/SiPS52927.2021.00040

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.