A Scalable Generator for Massive MIMO Baseband Processing Systems with Beamspace Channel Estimation

Author(s):
Dai, Yue; Liew, Harrison; Eslami Rasekh, Maryam; Mirfarshbafan, Seyedhadi; Gallyas-Sanhueza, Alexandra; Dunn, James; Madhow, Upamanyu; Studer, Christoph; Nikolić, Borivoje

Publication date:
2021

Permanent link:
https://doi.org/10.3929/ethz-b-000528703

Originally published in:
https://doi.org/10.1109/SiPS52927.2021.00040

A Scalable Generator for Massive MIMO Baseband Processing Systems with Beamspace Channel Estimation

Yue Dai[1], Harrison Liew[1], Maryam Eslami Rasekh[2], Seyed Hadi Mirfarshbafan[4], Alexandra Gallyas-Sanhueza[3], James Dunn[1], Upamanyu Madhow[2], Christoph Studer[4], Borivoje Nikolić[1]

{yuedai96, harrisonliew, jhdunn, bora}@berkeley.edu
[1]University of California, Berkeley, CA, USA

{rasekh, madhow}@ucsb.edu
[2]University of California, Santa Barbara, CA, USA

ag753@cornell.edu
[3]Cornell University, Ithaca, NY, USA

{smirfarsh, studer}@ethz.ch
[4]ETH Zürich, Zürich, Switzerland

***Abstract*—This paper describes a scalable, highly portable, and energy-efficient generator for massive multiple-input multiple-output (MIMO) baseband processing systems. This generator is written in Chisel and produces hardware instances for a scalable massive MIMO system employing distributed processing. The generator is parameterized in both the MIMO system and hardware datapath elements. Coupled with a Python-based system simulator, the generator can be adapted to implement other baseband processing algorithms. To demonstrate the adaptability, several generator instances with different parameter values are evaluated by FPGA emulation. In addition, a beamspace calibration and channel denoising algorithm are applied to further improve the channel estimation performance. With those algorithms, the error vector magnitude can be reduced by up 9.2%.**

***Index Terms*—massive MIMO, generator, beamforming, beamspace, FPGA-emulation**

I. Introduction

Innovations in capacity-achieving codes and efficient modulation techniques have brought the spectral efficiency of wireless point-to-point systems close to the theoretical Shannon limit [1]. By increasing the spatial resolution with hundreds to thousands of antennas at the base station (BS), massive MIMO can support numerous users in the same time-frequency resource by providing each user with their interference-free, high-capacity link to the BS [2]. For this reason, massive MIMO can be widely considered as the energy-efficient, secure, and robust approach to increase the channel capacity and reduce interference [3], [4].

While the massive MIMO concept is attractive, implementing the system in a cost-efficient and energy-efficient way is challenging. Centralized processing architectures, which collect the data from all front-ends and process them centrally, highly depend on interconnection bandwidth and require complex router designs. GPU-based software-defined radio (SDR) solutions have been explored for algorithm development, but they are not energy-efficient for large-scale deployment. One way to achieve energy efficiency is hardware specialization. To reduce the design cost, improve energy efficiency [5], [6], and enable hardware design reusability across FPGA and ASIC solutions, especially for signal processing tasks [5], [7], modular and highly parameterizable hardware generators should be used.

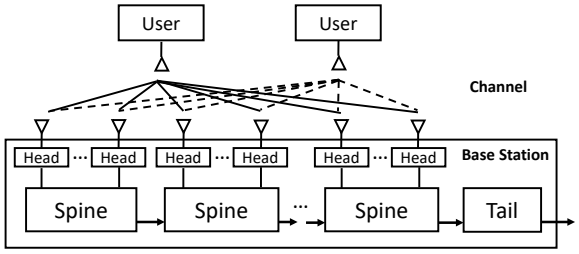

Fig. 1. The scalable massive MIMO BS architecture for uplink.

Our prior work proposed a generator for massive MIMO baseband processing systems to improve the scalability, cost-efficiency, and energy-efficiency [8]. The generator is implemented with the maximum-ratio-combining (MRC) beamforming algorithm. However, to adapt the existing architecture to other signal processing algorithms, such as beamspace domain algorithms to improve performance, modular design is needed. Due to the modular design of the generator, other algorithms can be applied with a modest effort. This work showcases the modular design of the generator through a low-overhead implementation of the beamspace channel estimation (BEACHES), a beamspace-domain channel denoiser.

The paper is organized as follows. Section II gives an overview of the scalable massive MIMO uplink baseband processing system, the calibration and the BEACHES algorithms, and the Spine generator design. Section III details the "golden" model simulator integrated with the calibration and BEACHES algorithms, and FPGA emulation setup for the generator evaluation. The performance is demonstrated and analyzed in Section IV.

II. System Overview

Fig. 1 shows the system architecture. The BS is assembled from three types of modules. Head modules are the mm-wave (or radio frequency) front-ends. Spine modules are the MRC

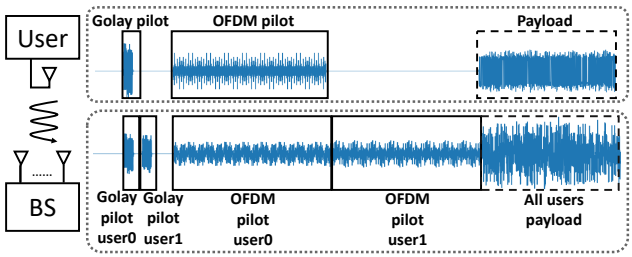

Fig. 2. Signal packet format.

beamformers for sub-arrays and are connected in a daisy chain architecture. The Tail module performs frequency-selective decorrelation to remove inter-symbol and inter-user interference.

A. Two-stage beamforming

To perform energy-efficiency signal processing, a two-stage beamforming algorithm has been proposed [9]. The first stage is the frequency-flat MRC beamformer to maximize the output signal-to-noise-ratio (SNR). The frequency-flat channel matrix is estimated from user-transmitted time-interleaved Golay pilots. The MRC beamformer can be split into subarrays of antennas; then, each subarray's locally-beamformed results are summed via the daisy chain.

Assume that a multi-user MIMO (MU-MIMO) system contains K users and M antennas at the BS; let $\mathbf{H} \in \mathbb{C}^{M \times K}$ be the channel matrix, and $\mathbf{y} \in \mathbb{C}^{M}$ be the signal received at the BS. Let $\tilde{\mathbf{y}}_{MRC} \in \mathbb{C}^{K}$ be the MRC beamformed signal at the BS. Let N equal to the number of sub-arrays, $\mathbf{B}_i \in \mathbb{C}^{M/N \times K}$ where $0 < i < N$, $\mathbf{H} = [\mathbf{B}_1^H \ \mathbf{B}_2^H \ ... \ \mathbf{B}_N^H]^H$, and $\mathbf{y}_i \in \mathbb{C}^{M/N}$ is the received signal at i-th sub-array. Then

$$\tilde{\mathbf{y}}_{MRC} = \sum_{i=1}^{N} \tilde{\mathbf{y}}_{i,MRC} = \sum_{i=1}^{N} \mathbf{B}_i^H \mathbf{y}_i. \tag{1}$$

In the second frequency-selective decorrelation stage, zero-forcing is performed for each narrow subcarrier within the wideband channel. This is estimated from user-transmitted time-interleaved orthogonal frequency division multiplexing (OFDM) pilots, sent after the Golay pilots. This two-stage beamforming method removes the inter-user interference and increases the signal-to-interference-plus-noise ratio (SINR) in a distributed fashion.

Quadrature amplitude modulated (QAM) payloads are sent from all users simultaneously, following the OFDM pilots. To synchronize the BS and users, a beacon signal is propagated to all users from the BS at the beginning of each packet to work as a time reference. Fig. 2 shows the signal packet format.

B. In-situ calibration

The channel matrix estimate from Golay pilots embeds array nonidealities, such as receiver gain and phase offsets, which inhibits beamspace algorithms because the resulting beamspace-domain channel vectors are not necessarily sparse. By leveraging the sparsity of the mm-wave channel by using compressive techniques, these array offsets can be de-embedded and calibrated out [10].

This proposed method estimates the spatial frequency difference between different pairs of sources in the line of sight (LoS) paths. After rotating all channel-user pairs to align the LoS components, the strongest common direction can be observed using spectral decomposition, yielding the calibration coefficients.

C. BEACHES algorithm

After calibration, the calibration coefficients are multiplied by the MRC beamformer coefficients to leverage the sparsity. The BEACHES algorithm [11] is then used to denoise the calibrated MRC channel matrix. For the estimated channel vector $\mathbf{h}$ of each user, BEACHES first converts the channel into the beamspace using a discrete Fourier transform (DFT) to obtain $\bar{\mathbf{h}} = \mathbf{F}\mathbf{h}$, where $\mathbf{F}$ is the $M \times M$ DFT matrix. Then it performs soft-thresholding on the entries of beamspace channel vector $\bar{\mathbf{h}}$. To find a MSE-optimal threshold $\tau^\star$, BEACHES relies on Stein's unbiased risk estimator (SURE) as a proxy for the actual MSE between the denoised channel estimate and the ground-truth channel. After soft-thresholding the magnitudes of entries of $\bar{\mathbf{h}}$ by $\tau^\star$, BEACHES converts the denoised channel vector back to the antenna domain using an inverse DFT.

For the denoised MRC beamforming, the calibration coefficients are removed from the denoised channel matrix because the Spine system has already included the channel synchronization in time domain. Then the denoised MRC beamformer coefficients are sent to the beamformer.

D. The Spine Generator

Generators are modular, highly parameterizable register-transfer level (RTL) hardware designs implemented in a higher-level programming language to enable parametrization and design reuse via generator extension. The Spine generator is written in Chisel [12], a hardware construction language that facilitates parameterized circuit generation for both ASIC and FPGA digital logic designs. The Spine generator is parameterized in the number of channels per Spine, the number of users in the system, the oversampling rate, the number of root-raised cosine (RRC) filter taps, the Golay pilot design, the Lagrange polynomial order, the datapath bitwidth, and the datapath parallelization. The Spine datapath is shown in Figure 3. This paper extends the Spine generator to enable signal processing associated with beamspace algorithms.

The inputs are M channels of 2x oversampled in-phase and quadrature (IQ) signals, which the degree of parallelism determined at the configuration time. First, a few signal correction modules, which include the RRC filter, IQ synchronizer and DC cancellation, correct some frontend impairments. Then, a set of channel estimators controlled by a sequencing controller correlate the pilot sequences to extract the channel matrix and channel delays, which are then synchronized for the next packet. These synchronized channels are then beamformed in the MRC beamformer, which is a systolic array based matrix multiplier. After beamforming, each user's delay is estimated then tuned out by using a fine delay synchronizer and a downsampling phase selector. The final outputs are the K users'

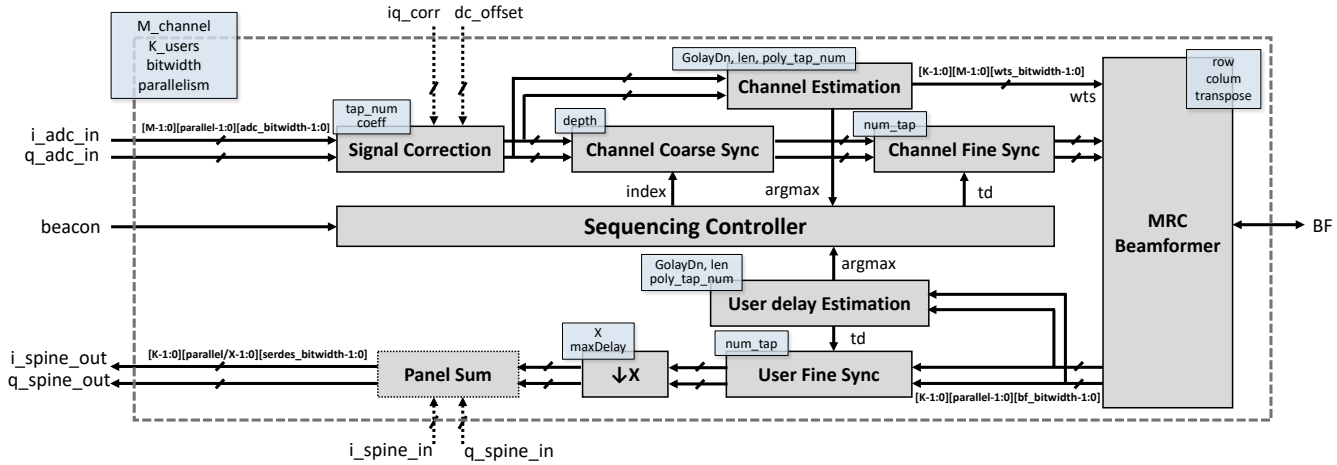

Fig. 3. The Spine generator datapath.

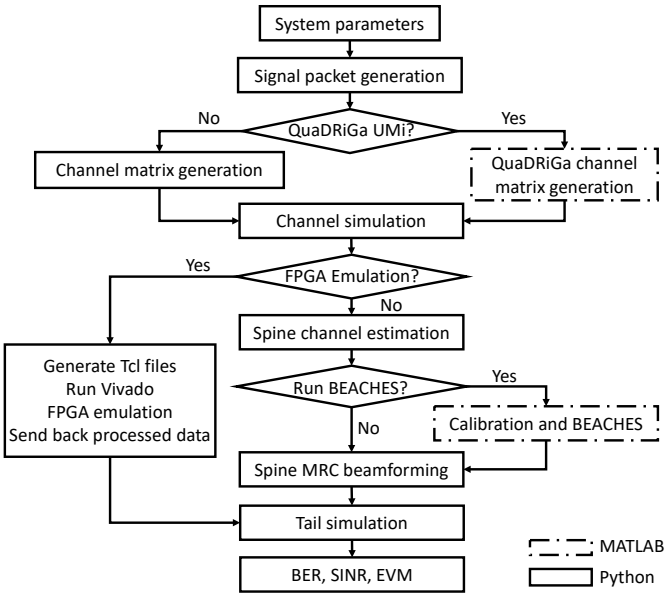

Fig. 4. The system data flow with the beamspace calibration and BEACHES algorithms.

MRC beamformed signals summed with the signal coming from the previous neighboring Spine in the daisy chain, sent to the next neighboring Spine. The parameters of each module generator are shown in the light blue boxes.

III. Evaluation

A. Simulator

A Python-based simulator is built to model the end-to-end massive MIMO system. The simulator encompases both the BS and the terminals, and includes the packet generation, various channel models, front-end impairments, and the modular signal processing. The simulator can wrap around any instance from the Chisel generator described in Section II by generating raw samples and post-processing the results. The calibration and BEACHES algorithms are implemented in MATLAB, but integrated into the simulator via a MATLAB-to-Python API, Matlab Engine API for Python [13]. After the MRC channel estimation, the simulator calls the calibration and BEACHES functions from MATLAB to denoise the channel matrix. This simulator is treated as the "golden" model against which the generator is verified.

The system data flow is shown in Fig. 4, where the solid rectangles represent Python functions, and the dash-dot rectangles represent MATLAB functions. In Python, user signal packets are generated according to the system parameters. Besides the flat independently and identically distributed (i.i.d) channel model, the line-of-sight (LoS) channel model, and the Rician channel model, all of which already exist in the Python simulator, we also include the QuaDRiGa mmMAGIC urban micro line-of-sight (QuadMMLoS) channel model implemented in MATLAB. QuaDRiGa channel model is an enhancement of the WINNER model following a geometry based, stochastic channel modelling approach for MIMO [14]. Compared with the LoS channel model, the QuadMMLoS includes the gain difference between users. QuadMMLoS channel matrix generation is called from Python through the API if the channel model is set to be QuadMMLoS in the system parameters; Otherwise, the channel matrix is generated in Python. Channel simulation includes transceiver frontend simulation and Gaussian noise addition.

For FPGA emulation, the Python simulator generates control and configuration Tcl files and calls Xilinx Vivado to build the FPGA image. Details will be discussed in the next section. For the simulation, Golay pilots received at the BS are used to perform the channel estimation for the MRC beamformer. If BEACHES is selected at configuration time, the calibration and BEACHES MATLAB functions are called from Python through the API, and the denoised channel matrix is then returned back to Python. Then the MRC beamforming and the frequency-selective decorrelation stage are applied in the Python simulator. After the two-stage beamforming, the Python simulator calculates the bit error rate (BER), the signal-to-interference-plus-noise ratio (SINR), and the error vector magnitude (EVM).

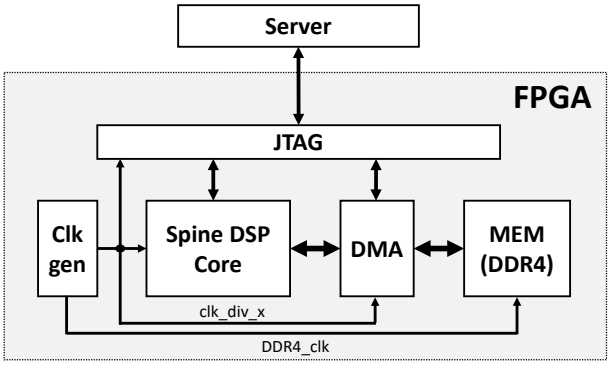

Fig. 5. The system architecture for the emulation.

TABLE I
FPGA EMULATION SYSTEM PARAMETER.

	Parameter	Value
System parameter	MIMO system parameter	32 channels, 4 users
	Signal bandwidth	200MHz
	Oversampling rate	2
	Golay pilot length	64
	Modulation scheme	QPSK
	Channel model	LoS, QuadMMLoS
FPGA parameter	FPGA Type	Xilinx VCU 118
	Number of channels per Spine	4
	Datapath bitwidth	8
	Datapath parallelization	8
	Baseband clock freq	50MHz

B. FPGA Emulation

The Spine generator is implemented on a Xilinx VCU118. The FPGA contains a Spine digital signal processing (DSP) core, the memory system, and the clock generators for the two clock domains. A server connects to the FPGA via JTAG to transfer data and configure the DSP core and memory system. The emulated BS instance contains 32 channels and 4 users. The FPGA emulation system architecture is shown in Fig. 5.

In the emulation, the Python simulator generates the Tcl files to configure and control the FPGA and converts the data from floating point to fixed point data format. Then the simulation data are transmitted to the on-board DDR4 memory. The Spine DSP core loads the data from memory and stores back to the memory while data are being processed, which is a streaming operation. After signal processing, the beamformed data are sent back to the server and are converted to the floating point format. At the same time, the estimated MRC beamforming channel matrix is also sent from the FPGA to the server.

To evaluate the system, the SINR of the Spine generator with both LoS channel and QuadMMLoS channel models are measured with a QPSK payload. The channel estimation performance evaluation with and without the calibration and BEACHES algorithms uses the following steps: 1) extracting the FPGA emulated channel estimation results; 2) running the beamspace denoising algorithm and getting the denoised channel matrix; 3) rerunning the Python simulation but using the FPGA emulated channel matrix and the denoised channel matrix from 1) and 2).

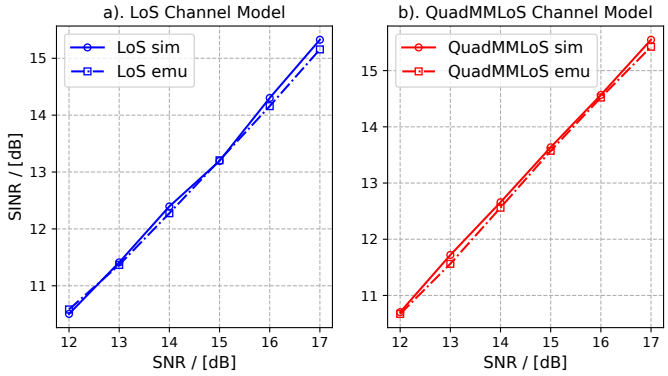

Fig. 6. SINR vs SNR under LoS and QuaDRiGa mmMAGIC UMi LoS channel models. The solid lines represent the the Python simulation result, while the dashes represent the FPGA emulation result.

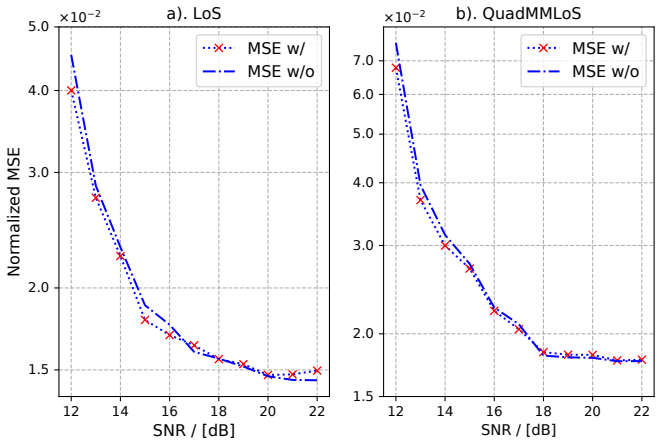

Fig. 7. Channel estimation MSE with and without calibration and BEACHES algorithms. The dashed lines represent the FPGA-emulated channel matrix MSE, and the solid lines represent the denoised FPGA emulated channel matrix MSE for both insets.

IV. Results

A. Performance

In [8], the Spine generator performance has been evaluated under the flat i.i.d channel and Rician channel models with 2 users in the massive MIMO system. In this paper, the performance of the Spine generator is further evaluated under LoS and QuadMMLoS channel models with 4 users in the system. The Spine DSP core with a different system setting can be easily generated by just changing the parameters in the generator. The MRC beamforming stage is performed on the FPGA without the calibration and BEACHES algorithms, while the frequency selective decorrelation stage is performed in the Python simulator.

Fig. 6 shows the SINR vs. SNR under different channel models. Inset a) shows the SINR under the LoS channel, while inset b) shows the SINR under the QuadMMLoS channel. The solid lines represent the the Python simulation result, while the dashes represent the FPGA emulation result. The

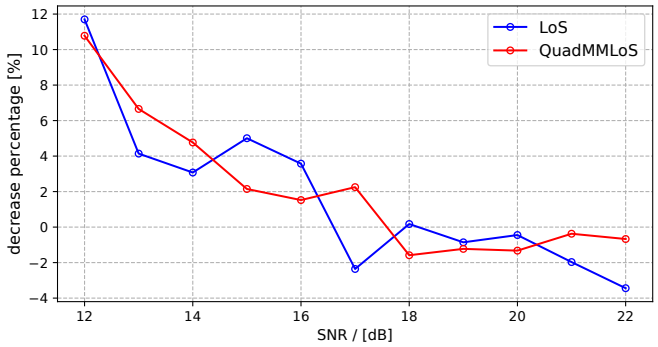

Fig. 8. The decrease percentage of the normalized MSE vs SNR with the calibration and BEACHES algorithms.

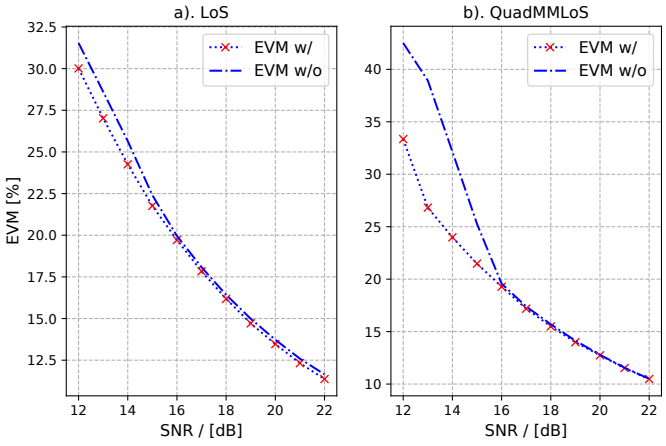

Fig. 9. EVM with and without calibration and BEACHES.

figure shows that for both channel models, the SINR changes linearly with respect to the change of the SNR. The result demonstrates the functionality of the system under LoS and QuadMMLoS channel models, validating the chosen generator parameters. In addition, the emulation results are very close to the corresponding simulation results. The maximum SINR differences between simulation and emulation are 0.17dB for the LoS channel and 0.16dB for the QuadMMLoS channel.

B. Channel estimation performance

Fig. 7 shows the MRC channel estimation normalized mean square error (MSE) with and without the calibration and BEACHES algorithms with respect to SNR using the LoS and QuadMMLoS channel models. The MSE measures the error between the estimated channel matrix and the ground-truth channel matrix generated from the simulator. The Golay pilot length for the channel estimation is chosen to be 64 due to its good performance [8]. The dashed lines represent the FPGA-emulated channel matrix MSE, and the solid lines represent the denoised FPGA emulated channel matrix MSE for both insets.

For the LoS channel, the normalized MSE of the channel estimation with and without the calibration and BEACHES algorithms begins to increase when the SNR is lower than 20dB. When the SNR is higher than 17dB, the MSEs are almost the same with and without the beamspace algorithms. When the SNR is lower than 17dB, the normalized MSE of the denoised channel matrix becomes lower than the one without BEACHES applied. For the QuadMMLoS channel, the normalized MSE of the channel estimation for both algorithms begin to increase when the SNR is lower than 18dB. When the SNR is lower than 16dB, the MSE of the denoised channel matrix is lower than that of the one without BEACHES applied.

Fig. 8 shows the decrease of the normalized MSE vs SNR for both channel models. The SNR ranges from 12dB to 22dB. The figure shows that with the decrease of the SNR, the percentage of the decrease in the MSE increases by using the BEACHES algorithm, demonstrating its efficacy in improving the channel estimation performance in the SNR range of interest. With the calibration and BEACHES algorithms, the channel estimation MSE can be improved by up to 11.7% for the LoS channel model and 10.9% for the QuadMMLoS channel model.

C. EVM

The normalized root-mean-square (RMS) EVM measurement is done in the Python simulator with the following steps: 1) extract the FPGA emulated channel estimation results; 2) run the BEACHES algorithm and get the denoised channel matrix; 3) rerun the Python simulation with the FPGA emulated channel matrix and the denoised channel matrix from 1) and 2); 4) calculate the RMS EVM.

Fig. 9 shows the EVM vs. SNR with and without the calibration and BEACHES algorithm for both channel models. For both models, the EVM decreases after applying the BEACHES algorithm. For SNR higher than 16dB, the EVM with and without the BEACHES algorithm are almost same; while when the SNR is lower than 16dB, the EVM decreases notably by applying the BEACHES algorithm, especially for QuadMMLoS channel model. This difference is because the QuadMMLoS channel model also models different gains for different users, which makes it more sensitive to the channel estimation errors. The improvement of the EVM after the calibration and BEACHES algorithms can be up to 9.2%.

D. Summary

Table II summarizes our work and some state-of-the-art massive MIMO BS implementations. Compared with [15] and [16], our work is highly portable and parameterizable, and can be easily implemented on custom hardware platforms, such as FPGAs and ASICs. The distributed architecture makes our system scalable and the modular design enables its extension to implement emerging algorithms. Compared with our previous work [8], this work has been tested with a larger number of users in the massive MIMO system and more complex channel models.

V. Conclusion

This paper presents the design and the emulation results of the scalable, highly portable, and power-efficient Spine generator with the beamspace channel estimation. The Spine generator is parameterized across a range of MIMO system and the datapath

TABLE II
COMPARISON

	This work	Prev. work [8]	LuMaMi [15]	RIVF'19 [16]
Platform	FPGA/ASIC	FPGA/ASIC	FPGA	FPGA
Portable	Yes	Yes	No	No
Adaptability	High	High	-	-
# Users	4	2	100	16
Modulation	QAM	QAM	OFDM	QAM
Distributed	Yes	Yes	No	No
Bandwidth	200MHz[1]	200MHz	20MHz	935MHz[2]
Method	Two-stage BF w/ BEACHES	Two-stage BF	MRC/ZF/RZF	SOR
CHEST[3]	Yes	Yes	Yes	No

[1] 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 I.
[2] This is the maximum FPGA implementation clock frequency, not the signal bandwidth. The signal bandwidth isn't given.
[3] Short for channel estimation.

hardware parameters, and specific instances were emulated on an FPGA. The SINR, the normalized MSE of the channel estimation, and the EVM, under different channels, with and without two beamspace algorithms, are evaluated. The results show the improvement of the channel estimation with the beamspace calibration and denoising algoritms. The successful integration of the beamspace channel estimation algorithms also proves the high adaptability of our work. The next step for this work involves hardware integration of the beamspace channel estimator, and continued evaluation under different scenarios. In addition, the performance of the channel estimator needs to be evaluated in beamspace domain in order to support the full beamspace-domain algorithms.

ACKNOWLEDGMENT

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. The authors also acknowledge the students, staff, faculty, and sponsors of the Berkeley Wireless Research Center.

REFERENCES

[1] A.Goldsmith, "5G and beyond: What lies ahead for wireless system design," in *PIMRC*, 2014.
[2] T. L. Marzetta, "Noncooperative cellular wireless with unlimited numbers of base station antennas," *IEEE Transactions on Wireless Communications*, vol. 9, no. 11, pp. 3590–3600, 2010.
[3] E. G. Larsson *et al.*, "Massive MIMO for next generation wireless systems," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 186–195, 2014.
[4] F. Rusek *et al.*, "Scaling up MIMO: Opportunities and challenges with very large arrays," *IEEE Signal Processing Magazine*, vol. 30, no. 1, pp. 40–60, 2013.
[5] B. Nikolic *et al.*, "Generating the next wave of custom silicon," in *IEEE 44th European Solid-State Circuits Conference*, 2018, pp. 6–11.
[6] B. Nikolić, "Simpler, more efficient design," in *IEEE 41st European Solid-State Circuits Conference*, 2015, pp. 20–25.
[7] O. Shacham *et al.*, "Rethinking digital design: Why design must change," *IEEE Micro*, vol. 30, no. 6, pp. 9–24, 2010.
[8] Y. Dai *et al.*, "A scalable massive MIMO uplink baseband processing generator," in *2020 IEEE International Conference on Communications*, 2021, pp. 1–6.
[9] G. LaCaille *et al.*, "Design and demonstration of a scalable massive MIMO uplink at E-band," in *2020 IEEE International Conference on Communications Workshops*, 2020, pp. 1–6.
[10] M. Rasekh, B. Puranik, U. Madhow, and M. Rodwell, "In-the-field calibration of all-digital mimo arrays," submitted.
[11] S. H. Mirfarshbafan, A. Gallyas-Sanhueza, R. Ghods, and C. Studer, "Beamspace channel estimation for massive MIMO mmWave systems: Algorithm and VLSI design," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 12, pp. 5482–5495, 2020.
[12] J. Bachrach *et al.*, "Chisel: Constructing hardware in a Scala embedded language," in *Design Automation Conference*, 2012, pp. 1212–1221.
[13] "Matlab engine api for python," https://www.mathworks.com/help/matlab/matlab-engine-for-python.html. Accessed on 2021-06-19.
[14] F. Burkhardt, S. Jaeckel, E. Eberlein, and R. Prieto-Cerdeira, "Quadriga: A mimo channel model for land mobile satellite," in *The 8th European Conference on Antennas and Propagation (EuCAP 2014)*, 2014, pp. 1274–1278.
[15] S. Malkowsky *et al.*, "The world's first real-time testbed for massive MIMO: Design, implementation, and validation," *IEEE Access*, vol. 5, pp. 9073–9088, 2017.
[16] C. Nhat Cuong *et al.*, "Hardware implementation of the efficient SOR-based massive MIMO detection for uplink," in *2019 IEEE-RIVF International Conference on Computing and Communication Technologies*, 2019, pp. 1–6.