Decentralized Equalization for Massive MU-MIMO on FPGA

Kaipeng Li[1], Charles Jeon[2], Joseph R. Cavallaro[1], and Christoph Studer[2]

[1]Department of Electrical and Computer Engineering, Rice University, Houston, TX
[2]School of Electrical and Computer Engineering, Cornell University, Ithaca, NY

***Abstract*—Massive multi-user multiple-input multiple-output (MU-MIMO) relies on large antenna arrays that serve tens of user equipments in the same time-frequency resource. The presence of hundreds of antenna elements and radio-frequency (RF) chains at the base station (BS) enables high spectral efficiency via fine-grained beamforming, but poses significant practical implementation challenges. In particular, conventional linear equalization algorithms used in the massive MU-MIMO uplink (users transmit to the BS), such as zero-forcing, typically require centralized architectures, which cause excessively high computational complexity and interconnect bandwidth between the baseband processing unit and the RF chains. In order to mitigate the complexity and bandwidth bottlenecks, we propose a VLSI design of a decentralized feed-forward architecture and a parallel equalization algorithm relying on large-MIMO approximate message passing (LAMA). We use high-level synthesis (HLS) to develop the VLSI architecture and provide corresponding FPGA implementation results. Our results demonstrate that the proposed decentralized LAMA equalizer achieves competitive performance and complexity as existing centralized solutions that have been designed on register-transfer level.**

I. Introduction

Massive multi-user multiple-input multiple-output (MU-MIMO) is widely believed to be a core technology in fifth-generation (5G) wireless systems [1]. By equipping the base station (BS) with hundreds of antenna elements that serve tens of user equipments (UEs) simultaneously and in the same frequency band, massive MU-MIMO promises significantly higher spectral efficiency and link reliability than traditional, small-scale MIMO systems [2]. In the uplink phase (UEs communicate to the BS), equalization and data detection at the BS are necessary to recover the transmitted data streams from each UE. In order to realize the full spectral-efficiency benefits of massive MU-MIMO, linear equalizers, such as zero-forcing (ZF) or minimum mean-square error (MMSE)-based equalizers, are required [3]. Such linear equalization schemes typically rely on *centralized* processing, i.e., all receive signals and full channel state information (CSI) must be available at a single baseband processing unit that carries out the necessary computations. Such centralized solutions, however, require that raw baseband and CSI data from hundreds of antennas must be transferred into a single computing fabric, which results in excessively high data rates that cannot be sustained by existing interconnect technologies, such as the common public radio interface (CPRI) [4], and by typical chip input/output (I/O) bandwidths [5]. In addition, even if there were means to transport the required data into a single computing fabric, processing these large amounts of data (e.g., for equalization) easily exceeds the storage capabilities and processing power of modern signal-processing fabrics, such as field-programmable gate arrays (FPGAs). Put simply, *centralized* massive MU-MIMO architectures will be unable to support systems with hundreds of antenna elements and RF chains.

A. Decentralized Baseband Processing

In order to mitigate these bandwidth and processing bottlenecks, existing massive MU-MIMO prototype designs, such as the Argos [6], the LuMaMi [7], and the Bigstation [8] testbeds, either rely on maximum ratio combining (MRC) which enables fully distributed equalization at the antenna elements, or on parallel processing across subcarriers in the frequency domain. However, MRC results in rather low spectral efficiency and parallel processing in the frequency domain still requires access to data from all BS antennas, which limits the scalability in terms of the number of antennas. In order to avoid theses issues while enabling high spectral efficiency via ZF or MMSE equalization, recent work in [5], [9], [10] proposed *decentralized baseband processing* (DBP). This approach enables parallel equalization and precoding on multiple computing fabrics, and scales well to massive MU-MIMO systems with a large number of antennas.[1] The proposed algorithms, however, rely on repeated consensus-information exchange [5], which suffers from high chip-to-chip transfer latency that limits the achievable throughput. To avoid this issue, reference [13] proposed a feedforward architecture in combination with the nonlinear large MIMO approximate message passing (LAMA) equalizer [14], which minimizes the latency issues of DBP without sacrificing spectral efficiency.

The work of KL, CJ, JRC, and CS was supported in part by Xilinx, Inc., the US National Science Foundation (NSF) under grants CNS-1265332, ECCS-1232274, ECCS-1408370, CNS-1717218, ECCS-1408006, CCF-1535897, CAREER CCF-1652065, CNS-1717559, and with hardware and software support from Texas Advanced Computing Center, Intel Hardware Accelerator Research Program, and Amazon EC2 cloud instances with Xilinx FPGAs.

[1]Distributed processing was also proposed for coordinated multipoint (CoMP) [11] and cloud radio access networks (C-RANs) [12] for multi-cell transmission. In contrast to these methods, DBP as in [5], [9], [10], [13] and this work are targeted for massive MU-MIMO systems in which the baseband processors are collocated with one antenna array in a single cell.

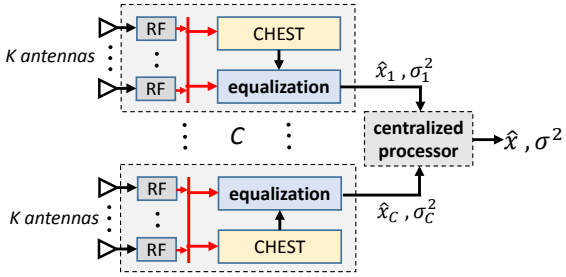


Fig. 1. Fully-decentralized feed-forward equalization architecture. Channel estimation (CHEST) and equalization are performed in a decentralized manner at each of the C clusters. The local equalization estimates are fused at a centralized processor which calculates a final estimate via weighted addition.

B. Contributions

In this paper, we build upon the fully-decentralized feed-forward DBP architecture put forward in [13] and develop a reference FPGA design that enables scalable and high-throughput DBP in massive MU-MIMO systems. We consider the architecture illustrated in Fig. 1, in which the BS antenna array is divided into C clusters, each associated with independent RF circuitry and computing hardware. In each cluster, we perform equalization based on the LAMA equalizer [13], [14] solely using the signals received from the associated antennas and local CSI. The C equalization results from each cluster are then fused at a centralized BS processor which enables an error-rate performance that is close to that of centralized linear MMSE equalization. As a proof-of-concept of our approach, we use high-level synthesis (HLS) to design a configurable and modular single-FPGA implementation that can be adapted to perform DBP on multi-FPGA systems in the future. By using a number of optimization strategies on HLS and hardware level, our implementation results show that one can achieve competitive error-rate performance, throughput, and hardware complexity compared to existing centralized solutions that have been designed on register-transfer level (RTL) using hardware description languages (HDL).

II. Fully-Decentralized Equalization via LAMA

We now introduce the system model and summarize fully-decentralized equalization via the LAMA algorithm.

A. System Model

We consider a massive MU-MIMO uplink system that uses orthogonal frequency-division multiplexing (OFDM). The system consists of U single-antenna UEs, each of which is associated to a dimension of the data vectors $\mathbf{x}_w \in \mathcal{O}^U$ with subcarrier indices $w = 1, \ldots, W$; these data vectors are transmitted to a B-antenna BS, where $\mathcal{O}$ denotes the transmit constellation alphabet (e.g., QPSK). The input-output relation of the uplink channel at subcarrier w is modeled by $\mathbf{y}_w = \mathbf{H}_w \mathbf{x}_w + \mathbf{n}_w$, where $\mathbf{y}_w \in \mathbb{C}^B$ corresponds to the received signal vector at the BS, $\mathbf{H}_w \in \mathbb{C}^{B \times U}$ represents the uplink channel matrix, and $\mathbf{n}_w \in \mathbb{C}^B$ models i.i.d. circularly symmetric complex Gaussian noise with variance N_0 per complex entry. For each subcarrier, the BS performs equalization followed by data detection to extract estimates $\hat{\mathbf{x}}_w$ of the transmitted data vectors $\mathbf{x}_w \in \mathbb{C}^U$ using the received signal vector $\mathbf{y}_w$ and the channel matrix $\mathbf{H}_w$. In what follows, we consider perfect synchronization and channel state information at the BS; we also omit the subcarrier index w.

B. Algorithm Details

As shown in Fig. 1, the estimates of the transmitted data vector $\hat{\mathbf{x}}$ are computed in a decentralized manner by partitioning the BS antenna array into C clusters. Each cluster is associated with $K = B/C$ antennas and RF chains, and each cluster contains a dedicated baseband processor. Each cluster $c = 1, \ldots, C$ only has access to the local receive vector $\mathbf{y}_c \in \mathbb{C}^K$, which contains the received signal from the associated antenna elements, and access to local CSI[2] $\mathbf{H}_c \in \mathbb{C}^{K \times U}$, which represents the channel matrix associated to the antennas connected to cluster c. We focus on fully-decentralized (FD) equalization as put forward by [13]: each cluster c performs equalization using $\mathbf{y}_c$ and $\mathbf{H}_c$ to compute a local estimate $\hat{\mathbf{x}}_c$ as well as the associated post-equalization noise-and-interference-variance σ_c^2. A centralized processor is then used to fuse all C estimates into a final estimate via the weighted sum $\hat{\mathbf{x}} = \sum_{c=1}^{C} \lambda_c \hat{\mathbf{x}}_c$ where $\lambda_c = \frac{1}{\sigma_c^2}(\Sigma_{c'=1}^{C} 1/\sigma_{c'}^2)^{-1}$, $c = 1, \ldots, C$, that minimizes the post-equalization noise variance [13].

A straightforward way for performing FD equalization would be to use conventional linear MMSE equalization in each cluster $c = 1, \ldots, C$. For example, in each cluster c, one could compute a local estimate $\hat{\mathbf{x}}_c = \left(\mathbf{H}_c^H \mathbf{H}_c + \frac{N_0}{E_x} \mathbf{I}_U\right)^{-1} \mathbf{H}_c^H \mathbf{y}_c$, where E_x denotes the average per-user transmit power, and $\mathbf{I}_U$ represents the $U \times U$ identity matrix. In order to obtain a superior local estimate, we resort to the nonlinear LAMA algorithm proposed in [13]. Specifically, we compute a slightly modified version of the FD-LAMA algorithm proposed in [13].

Algorithm 1 (FD-LAMA [13])**.** *In the first iteration, we initialize* $s_{c,\ell}^1 = 0$ *and* $\phi_{c,l}^1 = E_x$ *for* $c = 1, \ldots, C$, $\ell = 1, \ldots, U$. *We furthermore set* $\mathbf{v}_c^1 = \mathbf{0}$ *and* $\hat{\mathbf{x}}_c^1 = \mathbf{y}_c^{MRC} + (\mathbf{I}_U - \mathbf{G}_c)\mathbf{s}_c^1 + \mathbf{v}_c^1$ *for* $c = 1, \ldots, C$, *where* $\mathbf{y}_c^{MRC} = \mathbf{H}_c^H \mathbf{y}_c$ *is the local MRC output, and* $\mathbf{G}_c = \mathbf{H}_c^H \mathbf{H}_c$ *is the local Gram matrix. For each of the following FD-LAMA iterations* $t = 2, \ldots, T_{max}$, *we compute*

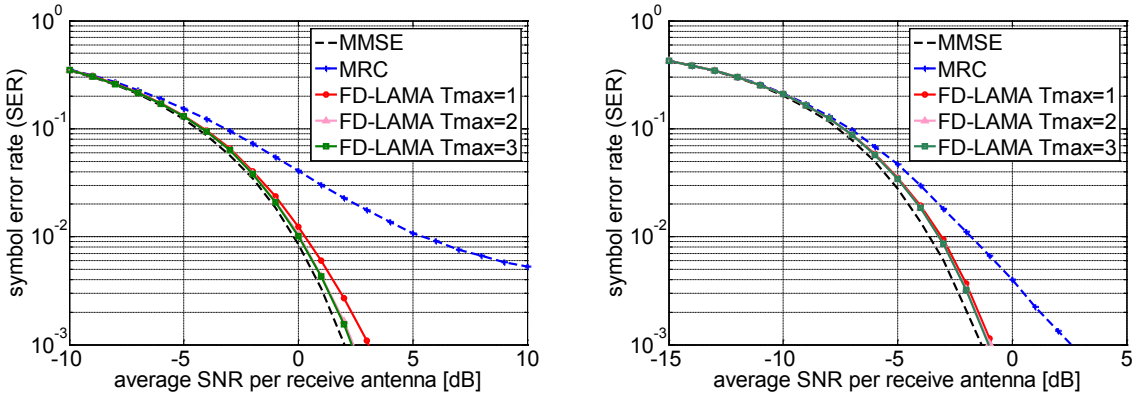
$$
\begin{aligned}
\mathbf{s}_c^t &= \mathsf{F}(\hat{\mathbf{x}}_c^{t-1}, N_0 + \beta \phi_c^{t-1}) \\
\phi_c^t &= \langle \mathsf{G}(\hat{\mathbf{x}}_c^{t-1}, N_0 + \beta \phi_c^{t-1}) \rangle \\
\mathbf{v}_c^t &= \tfrac{\beta \phi_c^t}{N_0 + \beta \phi_c^{t-1}} (\hat{\mathbf{x}}_c^{t-1} - \mathbf{s}_c^{t-1}) \\
\hat{\mathbf{x}}_c^t &= \mathbf{y}_c^{MRC} + (\mathbf{I}_U - \mathbf{G}_c)\mathbf{s}_c^t + \mathbf{v}_c^t,
\end{aligned}
$$

where $\langle \mathbf{z} \rangle = \frac{1}{U} \sum_{\ell=1}^{U} z_\ell$. *The functions* $\mathsf{F}(\hat{x}_{c,\ell}, \tau_c)$ *and* $\mathsf{G}(\hat{x}_{c,\ell}, \tau_c)$ *operate entry-wise on vectors and are defined by*

$$
\begin{aligned}
\mathsf{F}(\hat{x}_{c,\ell}, \tau_c) &= \textstyle\int_{x_\ell} x_\ell f(x_\ell | \hat{x}_{c,\ell}) \mathrm{d}x_\ell \\
\mathsf{G}(\hat{x}_{c,\ell}, \tau_c) &= \textstyle\int_{x_\ell} |x_\ell|^2 f(x_\ell | \hat{x}_{c,\ell}) \mathrm{d}x_\ell - |\mathsf{F}(\hat{x}_{c,\ell}, \tau_c)|^2.
\end{aligned}
$$

Here, $f(x_\ell | \hat{x}_{c,\ell})$ *is the posterior probability density function of the transmit symbol* x_ℓ *which is calculated as in [13].*

[2]Each cluster estimates the local channel matrix $\mathbf{H}_c$ independently; local CSI is not made available to the other clusters. See [5] for more details.



(a) $K = 32$, $C = 2$, $B = 64$, and $U = 8$. (b) $K = 32$, $C = 4$, $B = 128$, and $U = 8$.

Fig. 2. Symbol error-rate (SER) performance of centralized and decentralized equalization. We compare the SER of FD-LAMA with that of centralized linear MMSE and fully distributed MRC equalization. FD-LAMA approaches the performance of MMSE equalization for a small number of iterations.

In each algorithm iteration, we update the parameters in the order of $\mathbf{s}_c$, ϕ_c, $\mathbf{v}_c$, and $\hat{\mathbf{x}}_c$, so that we can directly extract $\hat{\mathbf{x}}_c^{T_{\max}}$ at the end of the last iteration to obtain the final estimate $\hat{\mathbf{x}} = \Sigma_{c=1}^{C} \lambda_c \hat{\mathbf{x}}_c^{T_{\max}}$ at the centralized BS processor.

C. Error-Rate Simulation Results

We simulate the symbol error-rate (SER) performance of Algorithm 1 in a massive MU-MIMO system for two system configurations, $\{K = 32, C = 2, B = 64, U = 8\}$ and $\{K = 32, C = 4, B = 128, U = 8\}$, with QPSK modulation and for i.i.d. Rayleigh fading channels. Figures 2(a) and 2(b) show the SER performance of centralized linear MMSE equalization, fully distributed MRC, as well as FD-LAMA. We see that FD-LAMA significantly outperforms MRC and is able to approach the SER performance of centralized MMSE equalization, even for a small number of iterations. This observation is consistent with the achievable rate results shown in [13], which implies that FD-LAMA incurs only little performance loss compared to that of centralized solutions. Furthermore, since the number of antennas per cluster $K = 32$ is fixed for Figures 2(a) and 2(b), we see that by doubling the total number B of BS antennas (effectively by doubling C), FD-LAMA still performs similarly to the linear MMSE equalizer—this indicates that FD-LAMA-based equalization scales well with the number of BS antennas.

III. VLSI Design

We now describe the VLSI architecture of FD-LAMA. As a proof-of-concept, we implement our algorithm on a *single* FPGA to demonstrate its modularity and scalability. Our design can be distributed to multiple FPGAs in order to enable true DBP as is required by massive MU-MIMO systems—the implementation of such a design is part of ongoing work. We implemented FD-LAMA via high-level synthesis (HLS) using Xilinx Vivado HLS (v2017.3), which provides high design reconfigurability, supports numerous compiler directives for performance optimization, and often requires lower design effort than traditional RTL-based design using Verilog or VHDL. The HLS code is written in C++ and synthesized to RTL using Vivado HLS. To optimize the hardware efficiency, we rely on fixed-point arithmetic. We use the `ap_fixed`$\langle 16, 5 \rangle$ data type for most values in our design in order to support 16-bit precision fixed-point numbers with 5-bit integer bits.

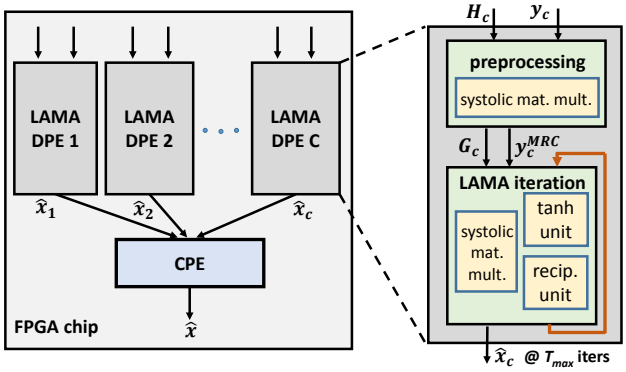


Fig. 3. Overview of the proposed VLSI architecture. In the context of a decentralized design on a single FPGA fabric, we implement C decentralized processing elements (DPEs), and each DPE serves as a local baseband processor for local FD-LAMA equalization at each of C clusters; local equalization estimates are fused at a centralized processing element (CPE), which emulates the centralized BS processor in a decentralized architecture shown in Fig 1.

A. Architecture Overview

Fig. 3 shows the proposed architecture and the data flow. We implement a total number of C decentralized processing elements (DPEs) on a single FPGA fabric for local FD-LAMA equalization to compute local estimates. The results are fused at a centralized processing element (CPE) to calculate the final estimate, i.e., a weighted sum of local estimates. Each local DPE uses the local channel matrix $\mathbf{H}_c$ and local receive vector $\mathbf{y}_c$ and performs preprocessing to obtain the local Gram matrix $\mathbf{G}_c$ and the local MRC output $\mathbf{y}_c^{\text{MRC}}$. The DPE then calculates the local equalization estimate $\hat{\mathbf{x}}_c$ according to Algorithm 1. After $T_{\max}$ LAMA iterations, all the local estimates $\hat{\mathbf{x}}_c^{T_{\max}}$ are passed from DPEs to the CPE which computes $\hat{\mathbf{x}}$. Since all DPEs and the CPE are integrated on

a single FPGA, the transfer of local estimates between the DPEs and the CPE can be realized with on-chip memory and buses with very short latency (using only a few clock cycles). We note that a multi-FPGA design would require substantially higher transfer latencies, which will reduce the throughput.

B. Architecture Details and Optimizations

We now focus on the key computations carried out within the DPEs and the CPE.

1) Preprocessing at DPE: To calculate the local Gram matrix $\mathbf{G}_c$ and the MRC output $\mathbf{y}_c^{\text{MRC}}$ at high throughput and low latency, we implement efficient matrix-matrix multiplications and matrix-vector multiplications using a systolic architecture. Concretely, to compute $\mathbf{G}_c = \mathbf{H}_c^H \mathbf{H}_c$, we partition $\mathbf{H}_c$ into the column vectors $\mathbf{h}_{1,c}, \mathbf{h}_{2,c}, \ldots, \mathbf{h}_{U,c}$ by using the `#pragma HLS ARRAY_PARTITION` directive; the row vectors of $\mathbf{H}_c^H$ are given by $\mathbf{h}_{1,c}^H, \mathbf{h}_{2,c}^H, \ldots, \mathbf{h}_{U,c}^H$. We partition the Gram matrix $\mathbf{G}_c$ into isolated entries $\mathbf{g}_{u,v,c} = \mathbf{G}_c(u,v)$, $u, v = 1, 2, \ldots, U$. By adding the `#pragma HLS PIPELINE` directive at the top-level loop for this matrix-matrix multiplication, the computation of $\mathbf{g}_{u,v,c} = \mathbf{h}_{u,c}^H \mathbf{h}_{v,c}$ for all values of u, v can be pipelined via HLS and is executed in a systolic manner with $U \times U$ operations performed in parallel. Similarly, to compute the local MRC vector $\mathbf{y}_c^{\text{MRC}} = \mathbf{H}_c^H \mathbf{y}_c$, we partition $\mathbf{y}_c^{\text{MRC}}$ into single entries $y_{u,c}^{\text{MRC}}$, $u = 1, 2, \ldots, U$, and exploit loop pipelining to perform U vector multiplications $y_{u,c}^{\text{MRC}} = \mathbf{h}_{u,c}^H \mathbf{y}_c$ for all $u = 1, \ldots, U$ in parallel.

The above explained array partition directives are necessary for efficient scheduling and pipelining of memory read and write operations. Arrays, as required to store the matrix $\mathbf{G}_c$, if not partitioned, are implemented as BRAMs that have two data ports, limiting the throughput of intensive read/write operations. By partitioning such arrays into smaller banks, we can synthesize them to multiple smaller distributed BRAMs and flip-flops on the FPGA, which increases the memory bandwidth and enables multiple parallel read/write operations.

2) LAMA Iterations at the DPE: In each LAMA iteration, we need to compute hyperbolic tangent functions and divisions. Specifically, for QPSK modulation, the F function in Algorithm 1 for updating $\mathbf{s}_c$ is given by

$$\mathsf{F}(\hat{x}_{c,\ell}, \tau_c) = \left(\tfrac{E_x}{2}\right)^{1/2} \Big(\tanh\left(\sqrt{2E_x}\Re\{\tfrac{\hat{x}_{c,\ell}}{N_0+\beta\phi_{c,\ell}}\}\right) + j \tanh\left(\sqrt{2E_x}\Im\{\tfrac{\hat{x}_{c,\ell}}{N_0+\beta\phi_{c,\ell}}\}\right)\Big).$$

Here, $\Re\{\cdot\}$ and $\Im\{\cdot\}$ extract the real and imaginary parts of a complex value, respectively. While the square-root values are constants for a given constellation set, the F function requires $\tanh$ and division operations. In HLS, we could simply use the division operator "/" and use $\tanh(\cdot)$ from the math library[3] for hyperbolic tangent computation in C++. However, such a naïve approach would be synthesized to complicated logic with excessively high latency and resource utilization. We therefore implement the hyperbolic tangent function and reciprocal unit using FPGA look-up tables (LUTs).

[3]The $\tanh(\cdot)$ function from the math library supports 32-bit and 16-bit floating-point values, but not fixed-point values. Nevertheless, one could perform type conversions between fixed-point and floating-point values before and after calling the $\tanh(\cdot)$ function.

TABLE I
RESOURCE UTILIZATION, LATENCY, AND THROUGHPUT FOR VARIOUS SYSTEM CONFIGURATIONS AT $K = 32$, $U = 8$, AND $T_{\text{MAX}} = 3$.

Clusters C	1	2	4
BS antennas B	32	64	128
LUTs (%)	11739 (2.7)	22789 (5.3)	44420 (10.3)
FFs (%)	16429 (1.9)	35080 (4.1)	76270 (8.8)
DSP48s (%)	219 (6.1)	497 (13.8)	1197 (33.3)
BRAM18	2	6	10
Clock freq. [MHz]	429	427	427
Latency [cycles]	310	336	384
Throuhgput [Mb/s]	22.2	20.4	17.8

The hyperbolic tangent unit ("tanh unit" in Fig. 3) takes a real-valued input p and generates an output q, which is an approximate value of $\tanh(p)$. We first detect the range of p: if $p \geq 4$, then $q = 1$; if $p < -4$, then $q = -1$. If $p \in [-4, 4)$, we use a LUT to get the corresponding approximate $\tanh(p)$ value. Specifically, we create a 2048-entry LUT with a BRAM that stores the pre-computed $\tanh$ results for a certain set of values $\{a_0, a_1, \ldots, a_{2047}\}$ which are evaluated at equidistant points in the range $[-4, 4)$, i.e., $a_i = (-4) + 8i/2048$. Given an input $p \in [-4, 4)$, we identify the value a_m that is closest to p, and fetch $\tanh(a_m)$ from the LUT to generate an approximate value of $\tanh(p)$. This approach entails only a small approximation error while avoiding the need for costly $\tanh$ functions.

The reciprocal unit ("recip. unit" in Fig. 3) first normalizes the input value to the range $[0.5, 1)$ by a leading-zeros detector and a bit shift. Similarly to the $\tanh$ LUT, we use a 2048-entry LUT with a BRAM to store pre-computed reciprocal values for a certain set of 2048 inputs $\{b_0, b_1, \ldots, b_{2047}\}$ where $b_i = 0.5 + 0.5i/2048$. Given a normalized input $d \in [0.5, 1)$, we identify the value b_m that is closest to d, fetch the reciprocal value of b_m in the LUT, and denormalize this reciprocal value by compensating for the initial bit shift to get the final output.

In addition to the above operations, each LAMA iteration requires matrix-vector multiplications and vector additions/subtractions. The matrix-vector multiplication required for computing $\hat{\mathbf{x}}_c$ is realized by a systolic array as discussed above. The vector addition/subtraction is performed for U entries in parallel with the `#pragma HLS PIPELINE` directive for entry-wise loop pipelining.

3) Result fusion at CPE: The CPE collects C local equalization estimates, i.e., U-entry vectors $\hat{\mathbf{x}}_c$, performs weighted sum of C results for each user entry in parallel with loop pipelining, and computes the final estimate $\hat{\mathbf{x}}$.

IV. IMPLEMENTATION RESULTS

We now show implementation results for the proposed FD-LAMA architecture on a single Xilinx Virtex-7 XC7VX690T FPGA. We benchmark the latency, throughput, and resource utilization, and compare our design with existing FPGA implementations for centralized massive MU-MIMO equalizers.

TABLE II
COMPARISON OF CENTRALIZED DATA DETECTORS FOR A $B = 128$ BS ANTENNA SYSTEM WITH $U = 8$ UES ON A XILINX VIRTEX-7 XC7VX690T FPGA.

Algorithm	CGLS [15]	Neumann [16]	Gauss-Seidel [17]	TASER [18]	FD-LAMA
Iterations	3	3	1	3	3
Modulation	64-QAM	64-QAM	64-QAM	QPSK	QPSK
LUTs (%)	3324 (0.8)	148797 (34)	18976 (4.3)	13779 (3.2)	11673 (2.7)
FFs (%)	3878 (0.4)	161934 (19)	15864 (1.8)	6857 (0.8)	15943 (1.8)
DSP48s (%)	33 (0.9)	1016 (28)	232 (6.3)	163 (5.7)	213 (5.9)
BRAM18	1	16	6	0	2
Clock [MHz]	412	317	309	225	429
Latency [clock cycles]	951	196	–	72	496
Throughput [Mb/s]	20	621	48	50	14
Throughput / LUTs	6017	4173	2530	3629	1186
Normalized at QPSK	2036	1391	783	3629	1186

Table I shows implementation results of FD-LAMA for various antenna configurations with $T_{\text{max}} = 3$ iterations and QPSK modulation. We fix the number of users $U = 8$ and number of antennas per cluster $K = 32$, and increase the total number of BS antennas $B = CK$ by increasing the number of clusters C. For example, when $C = \{1, 2, 4\}$, we have a total number of $B = \{32, 64, 128\}$ antennas. We see from Table I that the resource utilization increases roughly linearly with the number of clusters C, which is also the number of DPEs in our FPGA design. In contrast, the throughput degrades only slightly when increasing C, which indicates that the FD equalization architecture enables one to maintain the throughput when increasing B simply by increasing the number of computing fabrics. The use of multiple instances of our FD-LAMA design on multi-FPGA systems has the potential to further increase the throughput, which will be affected by the FPGA-to-FPGA transfer latency and bandwidth.

Table II compares the FD-LAMA design with recently proposed centralized data detectors for massive MU-MIMO [15]–[18]. All of the referenced designs are implemented using RTL with HDL, while our FD-LAMA HLS design is directly synthesized from C++ code; this enables us to easily reconfigure the parameters C, K, U, and LAMA iterations T_{max} as C++ variables. To arrive at a fair comparison, we set $C = 1$ for our design resulting in a centralized equalizer. We see that compared to the existing RTL-based FPGA implementations, our HLS-based design achieves competitive hardware efficiency in terms of throughput/LUTs normalized at QPSK modulation, while enabling higher design flexibility, shorter design cycles, and improved design scalability with the proposed decentralized architecture for supporting larger numbers of BS antennas.

While all of our above results are for a centralized version of our HLS design measured on a single FPGA, a fully-decentralized implementation on a multi-FPGA system using high-speed serial interconnect is part of ongoing work.

REFERENCES

[1] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. K. Soong, and J. C. Zhang, "What Will 5G Be?," *IEEE J. Sel. Areas Commun.*, vol. 32, no. 6, pp. 1065–1082, June 2014.

[2] E. G. Larsson, O. Edfors, F. Tufvesson, and T. L. Marzetta, "Massive MIMO for next generation wireless systems," *IEEE Commun. Mag.*, vol. 52, no. 2, pp. 186–195, Feb. 2014.

[3] J. Hoydis, S. ten Brink, and M. Debbah, "Massive MIMO in the UL/DL of Cellular Networks: How Many Antennas Do We Need?," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 2, pp. 160–171, Feb. 2013.

[4] http://www.cpri.info, *Common public radio interface*.

[5] K. Li, R. Sharan, Y. Chen, T. Goldstein, J. R. Cavallaro, and C. Studer, "Decentralized Baseband Processing for Massive MU-MIMO Systems," *To appear in IEEE J. Emerg. Sel. Topics Circ. Sys.*, 2017.

[6] C. Shepard, H. Yu, N. Anand, E. Li, T. Marzetta, R. Yang, and L. Zhong, "Argos: Practical Many-antenna Base Stations," in *ACM MobiCOM*, Aug. 2012, pp. 53–64.

[7] J. Vieira, S. Malkowsky, K. Nieman, Z. Miers, N. Kundargi, L. Liu, I. Wong, V. wall, O. Edfors, and F. Tufvesson, "A flexible 100-antenna testbed for Massive MIMO," in *IEEE Globecom*, Dec. 2014, pp. 287–293.

[8] Q. Yang, X. Li, H. Yao, J. Fang, K. Tan, W. Hu, J. Zhang, and Y. Zhang, "BigStation: Enabling Scalable Real-time Signal Processingin Large MU-MIMO Systems," in *ACM SIGCOMM*, Oct. 2013, pp. 399–410.

[9] K. Li, R. Skaran, Y. Chen, J. R. Cavallaro, T. Goldstein, and C. Studer, "Decentralized beamforming for massive MU-MIMO on a GPU cluster," in *IEEE GlobalSIP*, Dec. 2016, pp. 590–594.

[10] K. Li, Y. Chen, R. Sharan, T. Goldstein, J. R. Cavallaro, and C. Studer, "Decentralized data detection for massive MU-MIMO on a Xeon Phi cluster," in *Asilomar Conf. Sig. Sys. Comp.*, Nov. 2016, pp. 468–472.

[11] R. Irmer, H. Droste, P. Marsch, M. Grieger, G. Fettweis, S. Brueck, H. P. Mayer, L. Thiele, and V. Jungnickel, "Coordinated multipoint: Concepts, performance, and field trial results," *IEEE Commun. Mag.*, vol. 49, no. 2, pp. 102–111, Feb. 2011.

[12] M. Peng, Y. Li, Z. Zhao, and C. Wang, "System architecture and key technologies for 5G heterogeneous cloud radio access networks," *IEEE Network*, vol. 29, no. 2, pp. 6–14, Mar. 2015.

[13] C. Jeon, K. Li, J. R. Cavallaro, and C. Studer, "On the achievable rates of decentralized equalization in massive MU-MIMO systems," in *IEEE Int. Symp. Inf. Theory (ISIT)*, June 2017, pp. 1102–1106.

[14] C. Jeon, R. Ghods, A. Maleki, and C. Studer, "Optimality of large MIMO detection via approximate message passing," in *IEEE Int. Symp. on Inf. Theory (ISIT)*, June 2015, pp. 1227–1231.

[15] B. Yin, M. Wu, J. R. Cavallaro, and C. Studer, "VLSI design of large-scale soft-output MIMO detection using conjugate gradients," in *IEEE ISCAS*, May 2015, pp. 1498–1501.

[16] M. Wu, B. Yin, G. Wang, C. Dick, J. R. Cavallaro, and C. Studer, "Large-Scale MIMO Detection for 3GPP LTE: Algorithms and FPGA Implementations," *IEEE J. Sel. Topics Sig. Process.*, vol. 8, no. 5, pp. 916–929, Oct. 2014.

[17] Z. Wu, C. Zhang, Y. Xue, S. Xu, and X. You, "Efficient architecture for soft-output massive MIMO detection with Gauss-Seidel method," in *IEEE ISCAS*, May 2016, pp. 1886–1889.

[18] O. Castañeda, T. Goldstein, and C. Studer, "FPGA design of approximate semidefinite relaxation for data detection in large MIMO wireless systems," in *IEEE ISCAS*, May 2016, pp. 2659–2662.