A Heterogeneous Reconfigurable Cell Array for MIMO Signal Processing

Zhang, Chenxin; Liu, Liang; Markovic, Dejan; Öwall, Viktor

Published in:
IEEE Transactions on Circuits and Systems Part 1: Regular Papers

DOI:
10.1109/TCSI.2014.2366812

2015

A Heterogeneous Reconfigurable Cell Array for MIMO Signal Processing

Chenxin Zhang, *Student Member, IEEE,* Liang Liu, *Member, IEEE,*
Dejan Marković, *Member, IEEE,* and Viktor Öwall, *Member, IEEE*

***Abstract*—This paper presents a heterogeneous reconfigurable cell array, designed for high-throughput baseband processing of Multiple-Input Multiple-Output (MIMO) systems. To achieve high performance and energy efficiency while retaining high flexibility, the proposed architecture adopts heterogeneous and hierarchical resource deployments. Additionally, extensive vector computation enhancements and flexible memory access schemes are employed to better support MIMO signal processing. Implemented in a 65 nm CMOS technology, the cell array occupies 8.88 mm^2 core area and is capable of running at 500 MHz. For illustration, three computationally intensive blocks, namely channel estimation, channel matrix pre-processing, and hard-output data detection, of a 4×4 MIMO processing chain in a 20 MHz 64-QAM 3GPP Long Term Evolution Advanced (LTE-A) downlink are mapped and processed in real-time. Implementation results report a maximum throughput of 367.88 Mb/s with 1.49 nJ/b energy consumption. Compared to state-of-the-art designs, the proposed solution outperforms programmable platforms by several orders of magnitude in energy efficiency, and achieves similar level of efficiency to that of ASICs.**

***Index Terms*—Reconfigurable architecture, vector processor, channel estimation, pre-processing, QR Decomposition (QRD), Multiple-Input Multiple-Output (MIMO), data detection.**

I. Introduction

MULTIPLE-Input Multiple-Output (MIMO) techniques have been adopted in most newly released wireless communication standards, e.g., IEEE 802.11ac and 3GPP Long Term Evolution Advanced (LTE-A), to achieve high spectral efficiency. MIMO provides significant improvements in system capacity and link reliability without increasing bandwidth. However, the price-to-pay is an increased complexity and energy consumption due to the required sophisticated signal processing. In addition, MIMO is often combined with Orthogonal-Frequency-Division Multiplexing (OFDM) as a wireless access scheme to further improve spectral efficiency. Under such circumstances, the receiver needs to perform the corresponding processing at every OFDM subcarrier. This poses even more stringent requirements for hardware implementations, especially for battery-powered terminals operating at wide frequency band and large antenna numbers.

Besides the computational capability and energy consumption, flexibility becomes an important design factor. The fast evolving radio standards (more than 10 in a single module [1]) and the exploding number of operation modes within each protocol (63 for 3GPP LTE) makes the traditional mode-specific solution unaffordable in terms of silicon re-design cost and time-to-market. Moreover, being capable of allocating resources dynamically, flexible hardware platforms have the potential to provide run-time power-performance trade-offs by, for example, adopting different algorithms and system setups. This feature is vital to supporting link-adaptive processing [2] to efficiently combat with constantly changing wireless channels. However, flexibility comes at the price of design overhead in terms of processing speed and power, contradicting the aforementioned requirements. Thus, sufficing all three demands at the same time poses a critical design challenge.

Recent work on MIMO signal processing shows a paradigm shift towards flexible hardware designs, for example, to support different operation modes and algorithms [3]–[7]. In this paper, we propose an application-domain specific reconfigurable platform aimed at supporting flexible MIMO signal processing and achieving balanced design requirements. The platform is built upon a heterogeneous cell array architecture, capable of performing multiple tasks while fulfilling the stringent timing requirement such as for a 20 MHz LTE-A system with 4×4 MIMO and 64-QAM setup. Such high performance is primarily achieved by three key architecture-level improvements. First, heterogeneous resource deployments and a hierarchical network topology enable efficient hybrid-format data computing and substantial communication cost reduction. Second, vector-enhanced processing achieves low-latency high-throughput vector computing. Third, flexible memory access schemes relieve processing cores from non-computational address manipulations. Additionally, algorithm-architecture co-optimization is conducted to further improve hardware efficiency. Using previously developed algorithms [8]–[10], most of the operations involved in all three tasks are vectorized and unified, enabling extensive parallel processing and hardware reuse. Note that the proposed reconfigurable cell array is not limited to the presented tasks and algorithms, since the platform is flexible and extendible. Compared to related work, the proposed solution achieves a good design trade-off between flexibility and implementation cost.

The remainder of this paper is organized as follows. Section II briefly describes the target system model and the three processing algorithms. Section III introduces the proposed architecture framework and Section IV presents the detailed cell array architecture. Section V summarizes the implementation results. Finally, Section VI concludes the paper.

C. Zhang, L. Liu, and V. Öwall are with the Department of Electrical and Information Technology, Lund University, Box 118, SE-221 00 Lund, Sweden, Email: {Chenxin.Zhang, Liang.Liu, Viktor.Owall}@eit.lth.se.

D. Marković is with the Department of Electrical Engineering, University of California at Los Angeles, Los Angeles, CA 90095, USA, Email: dejan@ee.ucla.edu).

Digital Object Identifier

II. MIMO Signal Processing

Figure 1 shows a typical MIMO-OFDM system with N transmit (Tx) and receive (Rx) antennas. Assuming perfect front-end processing, the received vector $\boldsymbol{y}$ after Cyclic Prefix (CP) removal and FFT can be written as

$$\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x} + \boldsymbol{n}, \tag{1}$$

where $\boldsymbol{H}$ denotes the complex-valued channel matrix, $\boldsymbol{x}$ is the transmitted vector obtained by mapping a set of encoded information bits onto a Gray-labelled complex constellation, and $\boldsymbol{n}$ is the i.i.d. complex Gaussian noise vector with zero mean and variance σ_n^2.

As a case study, this work adopts a 20 MHz LTE-A transceiver to present processing algorithms and hardware configurations. Three tightly coupled Rx blocks (highlighted in Fig. 1), which are unique and crucial in MIMO for exploiting its full superiorities, are mapped onto the cell array: *estimation* of the channel matrix $\boldsymbol{H}$ using pilot tones, *channel matrix pre-processing* that is an indispensable step for detection algorithms, and *data detection* that recovers $\boldsymbol{x}$.

As mentioned in Section I, previously developed MIMO processing algorithms [8]–[10] are used to demonstrate the performance of the proposed hardware platform. In the following, the three adopted algorithms are briefly summarized for the sake of completeness.

A. Robust MMSE Channel Estimation

Utilizing the scattered pilot tones, the Robust MMSE (R.MMSE) algorithm [8] is adopted in this work, due to its estimation robustness and high Data-Level Parallelism (DLP). R.MMSE starts by Least Square (LS) estimation of channel vector $\boldsymbol{h}$ at pilot positions (denoted by subscript p),

$$\boldsymbol{h}_{p,\text{LS}} = \boldsymbol{y}_p \boldsymbol{x}_p^{-1}. \tag{2}$$

Data-tone channel coefficients are obtained by interpolation,

$$\boldsymbol{h}_{\text{MMSE}} = \boldsymbol{\mathcal{F}}\boldsymbol{h}_{p,\text{LS}} = \boldsymbol{R}_{\boldsymbol{h}_d\boldsymbol{h}_p}\left(\boldsymbol{R}_{\boldsymbol{h}_p\boldsymbol{h}_p} + \frac{\beta}{\text{SNR}}\boldsymbol{I}\right)^{-1}\boldsymbol{h}_{p,\text{LS}}, \tag{3}$$

where $\boldsymbol{R}_{\boldsymbol{h}_d\boldsymbol{h}_p}$ represents the channel cross-correlation between pilot and data-carrying subcarriers, $\boldsymbol{R}_{\boldsymbol{h}_p\boldsymbol{h}_p}$ is the channel auto-correlation between pilots, β is a constellation dependent constant [8], and $\boldsymbol{I}$ denotes the identity matrix.

Employing the robust correlation matrix [8], obtained by assuming a uniform power-delay profile, the function $\boldsymbol{\mathcal{F}}$ in (3) becomes a constant scaling matrix that can be computed off-line. In addition, a sliding window approach is applied to the R.MMSE algorithm, named as R.MMSE-SW for short, which dramatically reduces the dimension of $\boldsymbol{\mathcal{F}}$ due to the adoption of low-rank approximations [8]. In R.MMSE-SW, the sliding window size (N_{SW}) is a performance-complexity trade-off parameter, which can be adjusted based on the channel condition and performance demand.

B. Channel Matrix Pre-processing

Given the estimated channel matrix $\hat{\boldsymbol{H}}$, MMSE-SQRD [9] algorithm is adopted to compute the MMSE detection matrix

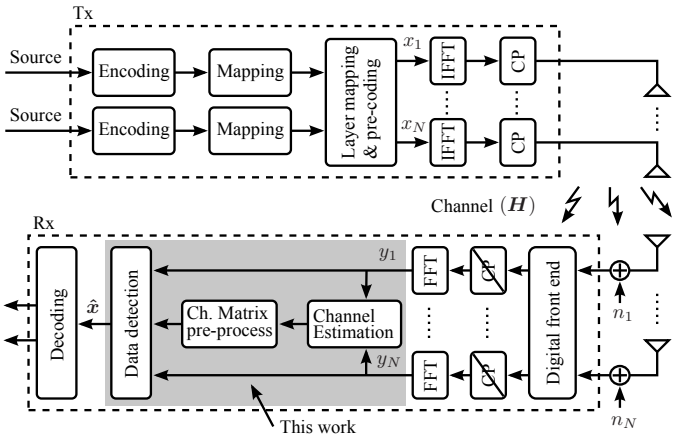

Fig. 1. A simplified MIMO system model. This work integrates all three shaded blocks into single reconfigurable baseband processor.

required in the succeeding data detector [10]. According to [9], MMSE-SQRD is equivalent to calculating the pseudo-inverse of an augmented channel matrix of size $2N \times N$,

$$\underline{\hat{\boldsymbol{H}}}^{\dagger} = \left(\left[\hat{\boldsymbol{H}}, \sigma_n \boldsymbol{I}\right]^T\right)^{\dagger} = \left(\underline{\hat{\boldsymbol{H}}}_p \boldsymbol{P}^T\right)^{\dagger}, \tag{4}$$

where $\underline{\hat{\boldsymbol{H}}}_p$ and $\boldsymbol{P}$ represent the sorted channel and the permutation matrix respectively, $(\cdot)^T$ indicates matrix transpose, and $(\cdot)^{\dagger}$ denotes matrix pseudo-inverse. In MMSE-SQRD, $\underline{\hat{\boldsymbol{H}}}_p$ can be decomposed as $\underline{\hat{\boldsymbol{H}}}_p = \underline{\boldsymbol{Q}}\underline{\boldsymbol{R}} = [\boldsymbol{Q}_a, \boldsymbol{Q}_b]^T \underline{\boldsymbol{R}}$ and $\underline{\boldsymbol{R}}^{-1} = 1/\sigma_n \boldsymbol{Q}_b$ is obtained as a by-product of the decomposition. Correspondingly, the system model in (1) can be rewritten as

$$\tilde{\boldsymbol{y}} = \boldsymbol{Q}_a^H \boldsymbol{y} = \underline{\boldsymbol{R}}\boldsymbol{x}_p + \tilde{\boldsymbol{n}}, \tag{5}$$

where $\boldsymbol{x}_p = \boldsymbol{P}^T\boldsymbol{x}$ and $\tilde{\boldsymbol{n}} = \boldsymbol{Q}_a^H\boldsymbol{n}$ are the permuted $\boldsymbol{x}$ and the noise vector, respectively. Considering the accuracy and numerical stability, computational complexity, and hardware reusability, Modified Gram-Schmidt (MGS) algorithm is used for implementing QR decomposition. Core operations of the MGS-QRD per iteration i is briefly summarized as follows, where the index $k = i+1, \ldots, N$, $(\cdot)_i$ denotes a column vector, and $(\cdot)_{i,i}$ represents the $(i,i)^{th}$ matrix element

$$\underline{r}_{i,i} = \|\underline{\hat{\boldsymbol{h}}}_{p_i}\|_2, \tag{6}$$

$$\underline{\boldsymbol{q}}_i = \underline{\boldsymbol{q}}_i / \underline{r}_{i,i}, \tag{7}$$

$$\underline{r}_{i,k} = \underline{\boldsymbol{q}}_i^H \underline{\boldsymbol{q}}_k, \tag{8}$$

$$\underline{\boldsymbol{q}}_k = \underline{\boldsymbol{q}}_k - \underline{r}_{i,k}\underline{\boldsymbol{q}}_i. \tag{9}$$

C. Node-Perturbed MMSE Data Detection

For data detection, we adopt the Node-Perturbation-enhanced MMSE (MMSE-NP) algorithm [10] to utilize its highly parallelized operations. MMSE-NP originates from a linear MMSE detection

$$\hat{\boldsymbol{x}}_p^{\text{MMSE}} = \mathcal{Q}\left(\underline{\boldsymbol{R}}^{-1}\tilde{\boldsymbol{y}}\right) = \mathcal{Q}\left(\frac{1}{\sigma_n}\boldsymbol{Q}_b\boldsymbol{Q}_a^H\boldsymbol{y}\right), \tag{10}$$

where $\mathcal{Q}(\cdot)$ denotes the slicing function returning a constellation point nearest to the computed symbol. After expanding

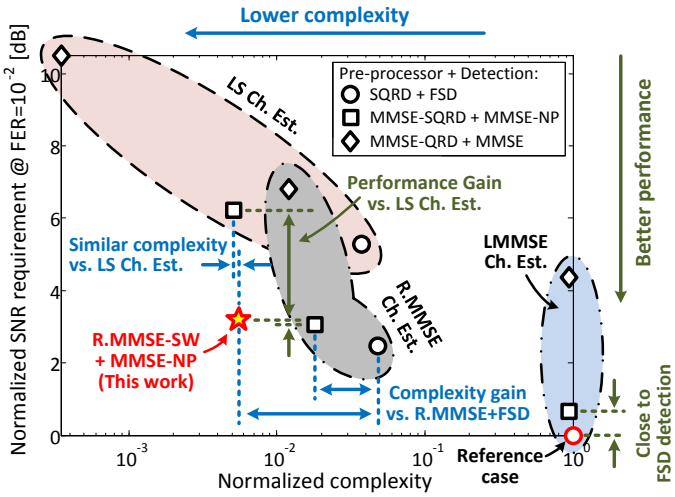

Fig. 2. Computational complexity and processing performance. Metrics are normalized to that of the reference case "LMMSE+FSD" which has unit computational complexity and zero required SNR at FER $= 10^{-2}$.

each element of $\hat{\boldsymbol{x}}_p^{\text{MMSE}}$ with Ω_i locally nearest siblings, based on the corresponding post-detection SNR of $\hat{\boldsymbol{x}}_p^{\text{MMSE}}$, $\mathcal{L} = \prod_{i=1}^{N} \Omega_i$ candidate vectors are constructed and the final detection result is obtained

$$\hat{\boldsymbol{x}} = \boldsymbol{P}\hat{\boldsymbol{x}}_p = \boldsymbol{P}\left(\arg\min_{\boldsymbol{x}_p \in \mathcal{L}} \parallel \tilde{\boldsymbol{y}} - \underline{\boldsymbol{R}}\boldsymbol{x}_p \parallel_2^2\right). \qquad (11)$$

In MMSE-NP, performance-complexity trade-off can be tuned by varying the symbol expansion parameter $\boldsymbol{\Omega} = [\Omega_1, \Omega_2, \cdots, \Omega_N]$. Compared to conventional tree-search based algorithms, e.g., [7] and [11], the key advantage of MMSE-NP resides in the elimination of sequential scalar operations, as both candidate-vector expansions and evaluations are carried out in parallel on all layers. Thus, it promises high implementation efficiency on vector-based architectures [10].

D. Algorithm Analysis

To give a full picture of the selected algorithms, Fig. 2 compares them with several representative MIMO signal processing methods in terms of complexity and performance. The included methods are LS and Linear MMSE (LMMSE) for channel estimation, and linear MMSE and Fixed-complexity Sphere Decoder (FSD) [11] for signal detection. In Fig. 2, numbers at the vertical axis denotes the minimum SNR required to achieve a Frame-Error-Rate (FER) of 10^{-2}, obtained from simulations using 3GPP EVA-70 channel model [12]. A rate 1/2 parallel concatenated turbo code is adopted with interleaver size of 5376 and BCJR decoding algorithm with 6 internal iterations. In terms of the computational complexity, the number of operations required in one LTE-A time slot is shown horizontally. To simplify the analysis, all operations are normalized to a W-bit complex-valued addition. This way, a W-bit complex-valued multiplication has the complexity of W; a W-bit real-valued division and square-root has a complexity of KW with K being a scaling factor to account for iteration numbers when using, for example, Newton-Raphson method. In this work, $W = 16$ and $K = 2$ are used, which are typical parameters used in baseband processing [5] [13]. Moreover, both coordinates in Fig. 2 are normalized to a reference case, "LMMSE+FSD" in the right-bottom corner, which provides the best performance among the considered algorithms. The selected scheme "R.MMSE-SW+MMSE-NP" with parameters $N_{\text{SW}} = 24$ and $\boldsymbol{\Omega} = [F, 4, 3, 2]$ [10] achieves a good trade-off between performance and complexity, providing more than 7 dB performance gain to "LS+MMSE" (left-up corner) and 100 times complexity reduction to the reference case "LMMSE+FSD". It should be re-emphasized that N_{SW} and $\boldsymbol{\Omega}$ are tunable parameters and should be optimized depending on the system requirement.

TABLE I
ALGORITHM PROFILING FOR VECTOR ($\boldsymbol{V}$) AND SCALAR (s) OPERATIONS IN THE ADOPTED MIMO SIGNAL PROCESSING.

Operation	Operation dimension & Proportion in each task					
	R.MMSE-SW Ch. Estimator		MMSE-SQRD Pre-processor		MMSE-NP Data detector	
$\boldsymbol{A} \odot \boldsymbol{B}^a$	–	–	$\boldsymbol{V}_{(N\times 1)}$	35%	–	–
$\boldsymbol{A} \cdot \boldsymbol{B}$	$\boldsymbol{V}_{(N_{\text{sw}}\times 1)}$	91%	$\boldsymbol{V}_{(N\times 1)}$	35%	$\boldsymbol{V}_{(N\times 1)}$	84%
$\boldsymbol{A} \pm \boldsymbol{B}$	–	–	$\boldsymbol{V}_{(N\times 1)}$	15%	$\boldsymbol{V}_{(N\times 1)}$	15%
$x_a \cdot x_b$	$s(x_a \cdot x_b)$	9%	–	–	–	–
Sorting	–	–	$s(x_i)$	5%	$s(x_i)$	~0%
$1/\sqrt{x}$	–	–	$s(x)$	10%	–	–
Pert.^{b}	–	–	–	–	$s(\Omega_i)$	1%

[a] Element-wise vector multiplication.
[b] Node perturbation in data detection.

With the presented algorithms, primitive operations required by the R.MMSE-SW estimator, MMSE-SQRD pre-processor, and MMSE-NP detector are characterized. Table I summarizes required vector and scalar operations and their proportion in each task. Two meaningful properties can be observed. First, more than 90% of operations in all three tasks are at vector level, indicating high DLP. Second, most of the operations are shared among the three algorithms, implying the potential of extensive hardware reuse. Before going on to present the hardware development, it is worth mentioning that algorithm selection is one of the important steps during the entire system design. Although reconfigurable platforms can support different algorithms, appropriate algorithm selection will lead to high hardware efficiency by making use of essential architectural characteristics.

III. Reconfigurable Architecture

In this section, we introduce the reconfigurable architecture that can efficiently support MIMO processing algorithms. To do so, we start by identifying hardware requirements, and then analyzing and comparing different reconfigurable architectures including our previously proposed cell array framework [14].

A. Requirements for Hardware Platform

Inspired by the aforementioned operation analysis, we extract three main properties of MIMO signal processing and the corresponding hardware requirements with respect to *computation*, *memory access*, and *data transfer*.

- *Massive vector operations*: in view of the massive vector operations, efficient vector computing and high bandwidth memory access are essential.

- *Hybrid data-widths and formats*: the coexistence of scalar and vector operations requires a hybrid computational data-path. Additionally, efficient communication mechanisms are expected to offload processing units from non-computational operations, e.g., data alignments, during data transfers of various data-widths and formats.
- *Multi-subcarrier processing*: as a scheduling technique to further exploit DLP [3], multi-subcarrier processing requires various data access patterns to perform operations simultaneously at multiple subcarriers. Therefore, flexible memory access schemes are required.

Architecture selection should take all these challenging requirements into account to obtain high hardware efficiency.

B. Comparison of Reconfigurable Architectures

Based upon the coupling between processing and memory units and their interconnects, previously proposed reconfigurable architectures can be classified into four broad categories, illustrated in Fig. 3.

The first group of architectures (Fig. 3(a)), such as [13] and [15], are constructed from an array of homogeneous processors, each having exclusive access to its own memory. The homogeneous deployment of resources is inefficient in supporting hybrid data computing. Besides, inter-core data transfers may take significant amount of processing power, as they require controls from processors at both ends.

Architectures in Fig. 3(b) are built from atomic Functional Units (FUs), e.g., [16] and [17]. Since data memories are accessible only from the border of the cluster, it may result in high data transfer overhead especially for large-size clusters. Additionally, centralized memory organization may become a bottleneck for vector and multi-subcarrier processing, due to memory contention during concurrent data accesses.

Figure 3(c) shows architectures that consist of heterogeneous units interconnected through a shared homogeneous network, such as [18] and [19]. Since the overhead of homogeneous interconnects (e.g., the crossbar switch) increases linearly with the number of array nodes and data precision, it may have restricted usage in large-size networks and high dimensional (e.g., vector) data applications. Additionally, when considering hybrid computing, various-width data transfers via shared homogeneous interconnects is not cost effective and may require frequent data alignment operations.

The last group, e.g., [14] and [20], is a heterogeneous array communicating via hierarchical network interconnects. This arrangement provides efficient hybrid data computing and low-cost network interconnects. As an example, our previously proposed cell array [14] (Fig. 3(d)) is constructed from heterogeneous tiles, containing any size, type, and combination of Resource Cells (RCs). RC is a common name for all hardware units, categorized into processing and memory cells. The separation of memory from processing cells significantly simplifies data sharing, as memory cells can be shared by multiple processors without physically transferring data. Memory coherence is preserved by allowing direct data transfers between memories without involving processors. Communication between RCs is managed hierarchically: neighbouring

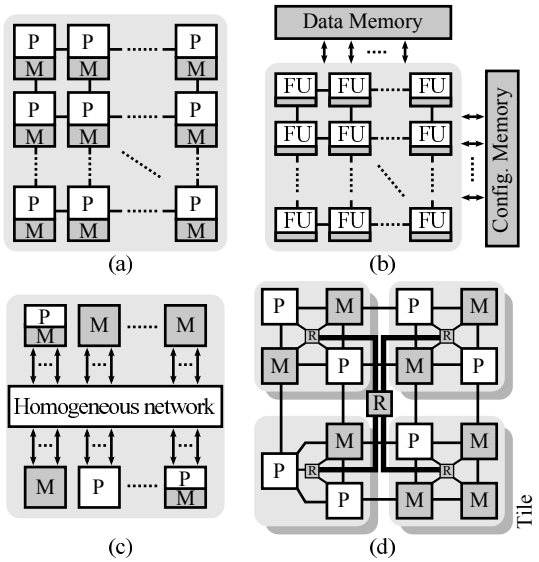

Fig. 3. Reconfigurable architectures, (a) homogeneous processor array, (b) FU cluster, (c) heterogeneous array with a shared homogeneous network, (d) heterogeneous array (an example of four tiles) with A hierarchical network.

cells are bidirectionally interconnected with low-latency high-bandwidth local links, while inter-tile transfers allow any RC to communicate through a hierarchical network using routing cells denoted as 'R' in Fig. 3(d). Compared to other interconnect topologies, the hierarchical network provides tighter coupling to RCs. For instance, connections within each tile can be localized to suffice both bandwidth and efficiency requirements, while hierarchical links provide flexible routing paths for inter-tile communication.

In conclusion, the architecture in Fig. 3(d) suffices all hardware requirements for MIMO signal processing. Hence, it is selected as a hardware infrastructure for further development.

IV. Heterogeneous Baseband Processor

Built upon the cell array framework, the proposed baseband processor is composed of four heterogeneous tiles that are partitioned into scalar- and vector-processing domains, see Fig. 4. In the vector domain, Tile-0 handles vector processing while Tile-1 provides data storages and various forms of vector and matrix accesses. In the scalar domain, Tile-3 controls other RCs during run-time and handles scalar and irregular operations with memory supports from Tile-2. Data transfers between the two domains are bridged by memory cells using a micro-block function [14], which is a technique used to provide data access with finer wordlength than the physical memory provides. This feature efficiently supports hybrid data transfers without additional controls from processors.

A. Hybrid Resource Configuration

Configurations for all RCs are managed in two ways, either by an external host via hierarchical network or by distributed controllers inside the cell array, as illustrated in Fig. 5. The former approach is mainly used for streaming data inputs, like the received vector $\boldsymbol{y}$, and off-line configurations, such as power-up setups. The later approach is used to conduct run-time configurations, which are issued on a per-clock-cycle basis and managed jointly by a task manager (i.e.,

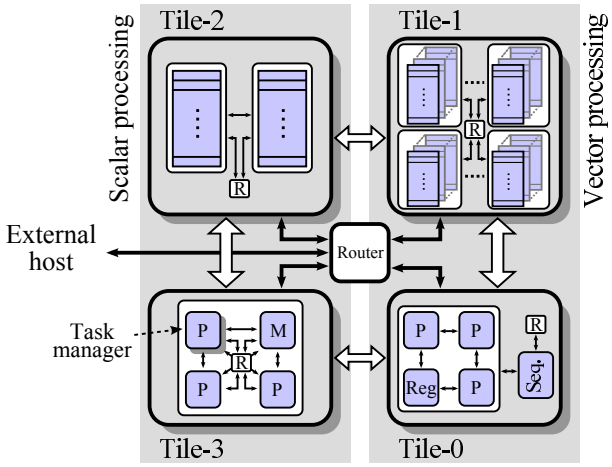

Fig. 4. A block diagram of the proposed heterogeneous baseband processor. Distributed controllers within RCs are omitted in this figure for simplicity.

a processing cell in Tile-3) and local controllers distributed in RCs. Specifically, the task manager tracks the overall processing flow and controls context switching (e.g., changing from channel estimation to QRD), while local controllers are responsible for applying configurations onto processing data- and memory access-paths (e.g., to switch between operations listed in Table I). This joint management of run-time configuration is advantageous in two aspects. First, local controllers within RCs are considerably simplified, since no individual tracking of the processing flow is required. Therefore, only one set of control circuits common to all local controllers is required, resulting in reduced overhead compared to that of a fully distributed configuration scheme. Second, deploying a dedicated task manager close to RCs inside the cell array smoothly integrates run-time configurations into the normal processing flow. For example, configurations are issued as soon as the current task is completed without interrupting and waiting for responses from an external host.

B. Vector Data Flow Processor

Figure 6(a) shows the architecture of Tile-0, consisting of three processing cells (pre-, core-, and post-processing), a register bank, and a sequencer. The three processing cells, shown on the upper half of Fig. 6(a), are deployed for vector computations, while the register bank provides data accesses from both internal registers and other tiles through register-mapped IO ports. The sequencer controls operations of the other cells via a control bus, drawn in dashed lines in Fig. 6(a). In the following, we present two architectural improvements for achieving efficient vector processing.

1) Vector enhanced SIMD core: In wireless baseband processing, Single Instruction Multiple Data (SIMD) is commonly used as a baseline architecture to exploit inherent DLP. Similarly, an SIMD-based architecture is adopted in the core-processing cell, consisting of $N \times N$ homogeneous Complex-valued Multiply-ACcumulate (CMAC) units, see Fig. 6.

Concerning the execution latency of vector operations, conventional SIMD architectures (e.g., [13] and [15]) are inefficient, since they are designed to handle parallel independent scalar data operands and their internal function units

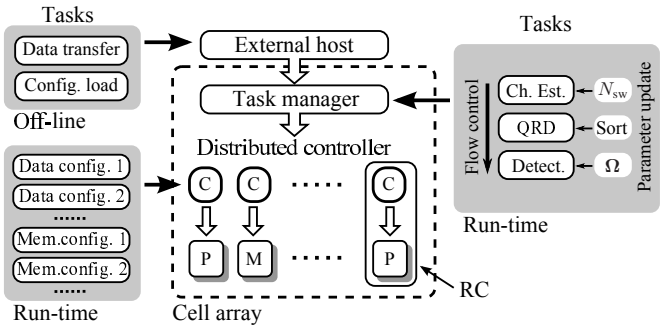

Fig. 5. A hierarchical configuration scheme of the baseband processor.

cannot operate collaboratively during instruction execution. For example, the computation of Vector Dot Product (VDP), which takes about 80% of entire vector processing in Table I, requires multiple clock cycles (depending on vector length), since each VDP operation is performed in a folded fashion using at most one CMAC unit.

To tackle the latency issue, we adopt an efficient vectorization technique to the SIMD core, so that all vector operations with length N have single-cycle execution speed. Specifically, each CMAC unit is equipped with an inter-cell connection (e-path) to link up with neighbouring CMACs during instruction execution, e.g., the e input in Fig. 6(b) is connected to the level-2 output (O_e) of the previous CMAC unit. Using this simple connection, level-2 adders of CMACs in every processing lane can be concatenated to form an adder-tree, capable of computing one N-length vector in every clock cycle, e.g., a VDP with an atomic operation of '$ab+e$'. Vectors exceeding this length are processed by folding, i.e., they are decomposed into data segments suitable for atomic operations.

2) VLIW-style multi-stage computing: Another important observation from the algorithm analysis (Section II-D) is that most of the vector processing involve several tightly coupled operations, such as complex conjugation (8) and result sorting (11) performed before and after vector computations, respectively. Mapping of such "long" processing solely on the SIMD core requires multiple operations, causing not only increased execution time but also redundant register accesses for intermediate result buffering. Therefore, we extend the SIMD core by adopting a Very Long Instruction Word (VLIW)-style multi-stage computation chain to accomplish several consecutive data manipulations in one single instruction. Specifically, two distinct processing cells are arranged around the SIMD core to pre- and post-process data respectively, see Fig. 6(a). Benefiting from this arrangement, more than 60% of register accesses are avoided, as the pre- and post-processing together take about two-thirds of the total vector computations. As an example, Table II summarizes operations required for implementing the MMSE-SQRD algorithm. A similar technique named operation chaining can be found in [13].

One drawback of VLIW-style architectures is the control overhead caused by the rigid instruction format. For example, any change of its sub-operations requires loading of a whole new instruction, resulting in unnecessary program storage and memory access for those unchanged parts. Although many code size reduction schemes exist, e.g., [21], they require a huge area cost to restore the instructions at run-time. In

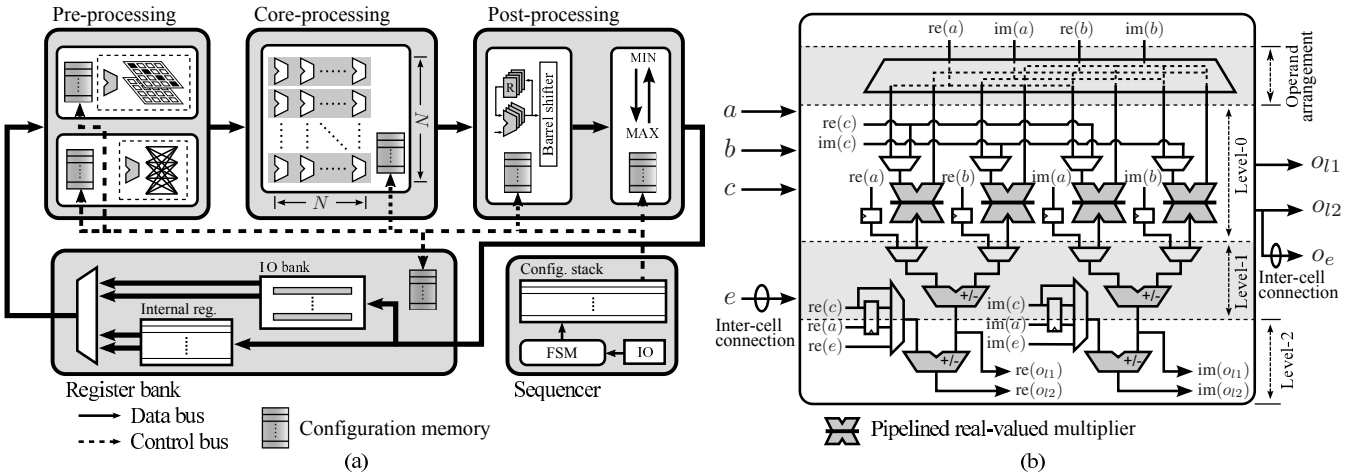

Fig. 6. (a) Microarchitecture of the vector data flow processor (Tile-0). A VLIW-style multi-stage computation chain consists of three processing cells: pre-, SIMD vector core-, and post-processing. (b) Architectural diagram of a vector enhanced Complex-valued Multiply-ACcumulate (CMAC) unit. 're()' and 'im()' represent the real and imaginary part of the input operand respectively.

TABLE II
AN EXAMPLE OF THE MULTI-STAGE COMPUTING IN MMSE-SQRD.

Pre-process	Pre-1: Complex conjugation
	Pre-2: Vector shuffling & broadcast
	Pre-3: Matrix data masking
Post-process	Post-1: Barrel shifting
	Post-2: Sorting

Operation	Pre-process	Core-process	Post-process
(4) & sort	Pre-3	VDP $(ab+e)$	Post-1, 2
(6)	–	VDP $(ab+e)$	Post-1
(7)	Pre-2	bc	Post-1
(8)	Pre-1	VDP $(ab+e)$	Post-1
(9)	Pre-2	$a-bc$	Post-1

contrast, we adopt a distributed control scheme to tackle the overhead issue by using the available configuration memories deployed inside RCs (Fig. 6(a)). This is based on an observation from Table II that operations tend to be used for more than one instruction, e.g., barrel shifting (Post-1). Therefore, by preloading data-path configurations into the distributed configuration memories, the run-time instruction control involves only memory address managements, which have much smaller code size than the content of configurations. Furthermore, partial processing updates are issued on-demand to a specific cell without reloading others.

Implementation of the pre- and post-processing cells depends on the operation profile of target applications. In the case of MIMO processing, the pre-processor is capable of performing data negation and absolute calculations, generation of access patterns using matrix masks, and data shuffling and broadcasting. The post-processing cell provides support for barrel shifting, e-path accumulations, and vector permutations.

C. Vector Data Memory Tile

Besides vector computation enhancements, the efficiency of the vector processor is contingent on memory access with regard to accessing bandwidth and flexibility. By inspection of the presented algorithms, it is required that the SIMD core has access to multiple matrices and/or vectors in each operation, so as to avoid poor resource utilization and throughput. As an example, efficient mapping of (6) requires two $N \times N$ matrix inputs, equivalent to having a $2\times(4\times4)\times(16+16)$=1024 bits/cycle memory bandwidth for a 16-bit 4×4 MIMO system. In addition to the bandwidth requirement, various forms of data accesses are needed, such as row- and column-wise addressing in matrix transposition. To meet these requirements, we adopt a hybrid memory organization and a flexible matrix access mechanism in the vector data memory tile (Tile-1).

1) Hybrid memory organization: To suffice the high memory accessing bandwidth, Tile-1 consists of vector and matrix access partitions, allowing simultaneous accesses of both vectors and matrices, see Fig. 7(a). The basic element in both partitions is a dual-port memory cell, which provides a vector-level data storage and allows simultaneous read and write operations to ease memory access and improve processing throughput at the price of a larger memory footprint. In addition, the matrix partition provides matrix data access, which is realized by concurrently accessing a group of memory cells using only one set of address control. This arrangement is referred to as a memory page, shown in Fig. 7(a). The vector accessing wordlength and the number of cells in a memory page are designed to match the processing capacity of the SIMD core in Tile-0, i.e., N scalar elements and N memory cells, respectively. On the other hand, the number of memory cells and pages are application dependent and should be optimized with respect to the bandwidth requirement and hardware cost. In this work, Tile-1 is configured to have 2 memory cells and 5 pages to ensure a sufficient memory storage required for the MIMO processing.

Memory operations and accessing modes of each cell and page are managed by a local controller with configurations stored in embedded registers, see Fig. 7(b). To communicate with other tiles, memory accesses are multiplexed using a crossbar network and interfaced through IO ports. For the array shown in Fig. 4, Tile-1 contains four IO ports, allowing simultaneous accesses of two $N \times 1$ vectors and two $N \times N$ matrices for providing accesses to both Tile-0 and Tile-2. Referring to the aforementioned example, this corresponds to a memory bandwidth of 1280 bits/cycle.

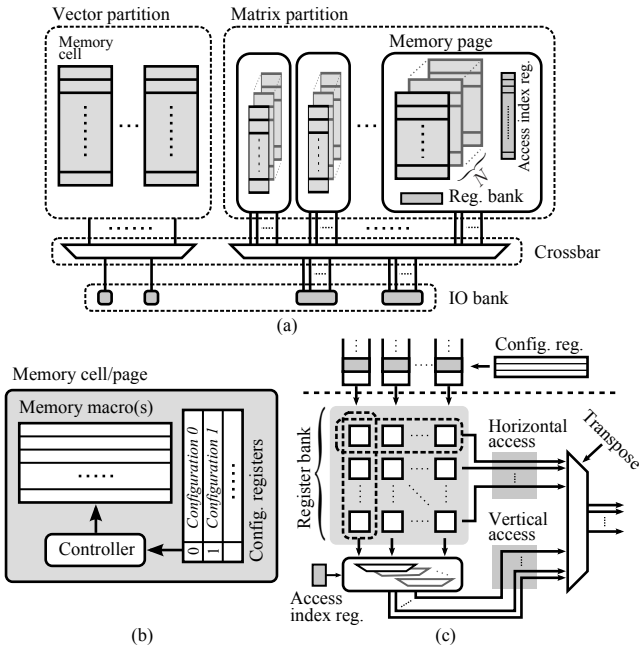

Fig. 7. Architecture of the vector data memory tile (Tile-1), (a) a hybrid memory organization, (b) operation and accessing control, (c) data loading path of a memory page, supporting matrix access indexing and transposition.

2) Flexible matrix data access: The presented multi-page memory arrangement and the crossbar network allow for the flexible data access required by the multi-subcarrier processing. For instance, by storing matrices of successive subcarriers in different memory pages, multiple data sets can be concurrently accessed and multiplexed based on arrangement indexes specified in memory configurations.

To further improve matrix access flexibility, a data arrangement circuit, illustrated in Fig. 7(c), is implemented in each memory page. Specifically, data loaded from each memory page are buffered in a local register bank and are capable of being rearranged vector-wise in a vertical direction, based on an access index associated with each matrix storage. Benefiting from this setup, vector readouts from a matrix can be accessed freely in any order without physically exchanging data. This is useful, for example, in supporting sorted matrix accesses in MMSE-SQRD (4). In addition to these index manipulations, the proposed architecture is capable of outputting matrices in a transposed form (required in (10)) by selecting either the row or column output. As a result, processing cells are relieved from such data arrangement operations, which otherwise result in enormous underused processing power. Moreover, physical data exchange and redundant memory accesses (due to read and write of the same data contents) are completely eliminated.

D. Scalar Resource Cells and Accelerators

In the scalar domain, Tile-2 and 3 perform scalar and conditional operations as well as dynamic configurations of other tiles in the array. Among them, Tile-2 consists of two scalar memories for storing data and configurations, respectively. Tile-3 contains one memory cell for data buffering and three processing cells, including one generic processor and two acceleration units, see Fig. 8. The generic processor is a customized RISC with optimized conditional instructions and

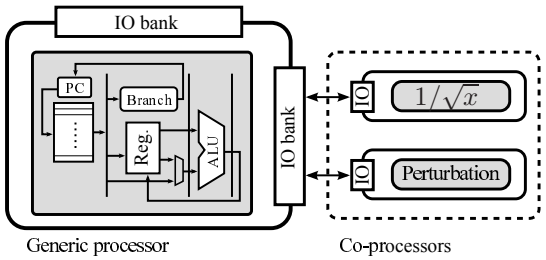

Fig. 8. Block diagram of the scalar processing cells in Tile-3, containing one generic RISC-structured processor and two accelerators.

specialized functionality for dynamic RC configurations [14]. The two accelerators behave like co-processors of the generic cell for performing irregular operations, i.e., the inverse square-root in (6) and the node perturbation in MMSE-NP, respectively. Detailed architectures of the generic processor and the scalar memory cell can be found in [14].

V. Implementation Results and Comparison

To cope with different system configurations and design constraints, the heterogeneous cell array is fully parametrizable at system design-time. For the case of 20 MHz 4×4 MIMO LTE-A downlink, the SIMD core in Tile-0 is configured to have 16 CMAC units. All computations (Tile-0 and Tile-3) are performed in 16 bits fixed-point arithmetic with 8 guard bits for accumulations. Besides, the array contains 2.34 Mb of memory, in which 88% are data buffers for keeping data required in one LTE-A time slot (e.g., channel and decomposed matrices), 2% are control memories for storing instructions and resource configurations, and 10% are reserved for facilitating flexible algorithm mappings and future system updates.

A. Implementation Results

Implemented in a 65 nm CMOS technology, the cell array has a core area of 8.88 mm^2 at 74% cell density in chip layout, equivalent to 2.76 M two-input NAND gates. Data buffers occupy more than 60% of the area, while logic blocks, including control memories and the hierarchical network, share the rest. Excluding those data buffers, it shows in Table III that most of the logic gates are taken by the vector partition, i.e., Tile-0 and 1. At 1.2 V nominal core voltage supply, the cell array is capable of running at 500 MHz reported from Static Timing Analysis (STA) of the post-layout design.

1) Timing analysis: The three MIMO processing tasks, i.e., channel estimation (CE), channel matrix pre-processing (QRD), and data detection (DT), are manually mapped onto the cell array with a primary focus on sufficing the stringent timing constraint and achieving high processing throughput. Figure 9 illustrates the structure of a 4×4 MIMO LTE-A data frame and the adopted task-oriented processing flow, which performs one task on all subcarriers before switching to the subsequent one. This is different from a subcarrier-oriented scheme (handling one subcarrier at a time), which requires more frequent context switching and thus long configuration time and more power consumption. In this work, processing is scheduled on a basis of one LTE-A time slot (t_{slot}). Every iteration starts as soon as the last pilot tone in OFDM symbol 1 is received. To avoid the need for additional data buffers

TABLE III
AREA AND POWER BREAKDOWN OF THE RECONFIGURABLE CELL ARRAY WITH DATA BUFFERS EXCLUDED.

Resource cell		Gate count [KG]		Power [mW]	
Tile-0		367	34.77%	164.93	53.75%
Tile-1	Memory cells	96	9.12%	5.99	1.95%
	Memory pages	365	34.6%	68.06	22.18%
Tile-2		47	4.44%	3.20	1.04%
Tile-3	RISC	70	6.60%	44.10	14.37%
	Others	61	5.83%	16.98	5.53%
Network		49	4.65%	3.56	1.16%
Total		1055	100.00%	306.84	100.00%

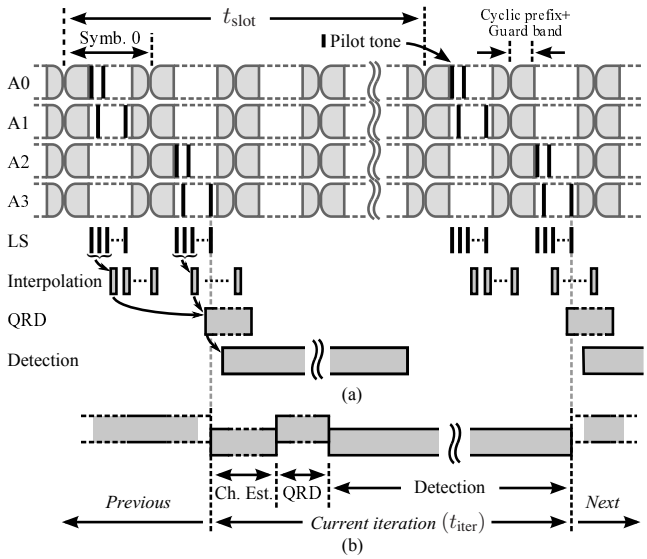

Fig. 9. Timing diagram of MIMO signal processing, (a) processing tasks and data dependencies, (b) the proposed task-oriented processing flow.

(i.e., more than one time slot), the computation time of each iteration (t_{iter}) is constrained by t_{slot}, i.e., $t_{\text{iter}} \leq t_{\text{slot}} = 0.5\,\text{mS}$.

Table IV summaries achieved performance of the three task mappings. Besides, configurations, such as program loadings from external host and memory initializations, are presented separately under "Miscellaneous". Run-time reconfigurations, hidden inside the processing time of each task, consume about 2% of the total computation time. This low control overhead is achieved by conducting the hybrid resource configurations (Section IV-A). Operating at 500 MHz, the total processing time for one LTE-A time slot is 469.72 μS. This fulfills the real-time requirement of the target LTE-A setup and results in about 6% spare time that can be used to map more advanced algorithms or upgrade system parameters. Based on the processing time and the number of tones/bits required to compute, Table IV presents the corresponding throughput achieved in each task. On average, recovering one transmitted vector $\hat{\boldsymbol{x}}$, with all processing tasks performed, requires 32.62 clock cycles, which is equivalent to a throughput of 367.88 Mb/s.

2) Computation efficiency: To evaluate the computation efficiency of the array, resource utilization of the SIMD core in Tile-0 is measured as a representative, since it contributes to more than 90% of the total computation capacity. Thanks to the vector enhanced SIMD structure (Section IV-B), an average utilization of 77% is achieved during the whole MIMO signal processing, as illustrated in Fig. 10.

TABLE IV
PERFORMANCE SUMMARY OF THE MIMO SIGNAL PROCESSING.

	Time [μS]	Throughput	Power[a] [mW]	Energy[a]
Ch. Estimation	41.60	28.84 MEst/s	276.24	9.58 nJ/Est
QRD	30.30	39.60 MQRD/s	315.36	7.96 nJ/QRD
Detection	380.40	454.26 Mb/s	280.82	0.62 nJ/b
Miscellaneous	2.82	N/A	269.99	0.81 nJ/op
Total/Average	469.72	367.88 Mb/s	306.84	0.83 nJ/b

[a] With data buffers excluded.

3) Power and energy consumption: Working at 500 MHz and 1.2 V voltage supply, the average power consumption for processing one data-carrying tone is 548.78 mW, including 306.84 mW from the logic blocks and 241.94 mW from the data buffers. The corresponding energy consumption for processing one information bit is 0.83 nJ/b and 1.49 nJ/b, without and with data buffers respectively. Table IV summaries average power and energy consumption of different tasks with data buffers excluded. As can be seen, power consumption of different tasks is quite balanced because of the high computation efficiency of the cell array achieved by the algorithm-architecture co-design. Moreover, Table III shows a tile-level power breakdown of the array. Among all, Tile-0 is the most power consuming block, because of the large area occupation and high resource utilization. It should be mentioned that simulated power figures from the post-layout design may be different for chip measurement results.

4) Flexibility: The flexibility is demonstrated by time-multiplexing three different tasks onto the reconfigurable cell array. Additionally, by making use of dynamic hardware reconfigurability, such as loading different programs and configurations to processing and memory cells respectively, the platform has the potential to support other system configurations. Examples include mapping of different algorithms and antenna setups, and run-time adaption of system performance, e.g., adjusting the frequency of channel estimation and detection parameters. Furthermore, the platform is extendible, thanks to the tile-based heterogeneous and hierarchical resource deployments. For example, larger antenna setups can be supported by extending resource cells and the bandwidth of local links, higher throughput can be achieved by doubling the number of tiles, and system performance can be improved by extending the scalar processing tile (Tile-3) with Log Likelihood Ratio (LLR) unit to perform soft-output data detection [16]. Based on the list of candidate vectors generated in the adopted detection algorithm, a searching unit is needed to find bit-level vectors required in LLR computations. Other scalar operations can be mapped onto the generic processor in Tile-3.

B. Comparison Analysis

In Table V, implementation results of the cell array are compared with previously reported designs. In fact, a fair quantitative comparison is difficult due to many different design factors, such as flexibility, algorithm selection, performance and operating scenario. Therefore, the following discussion only serves to give an overview of the design efficiency for related implementations. To ease the discussion, related hardware architectures are divided into three broad categories:

TABLE V
COMPARISON OF THIS WORK WITH ACCELERATORS AND RECONFIGURABLE PLATFORMS.

		[22]	[23]	[24]	[6]	[4]	[25]	[19]	[1]	[5]	This work	
Platform		ASIC				FPGA	GPU	Reconfigurable baseband processor				
Antenna		–	4×4	4×4	4×4	4×4	4×4	4×4	4×2	2×2	4×4	
Modulation (QAM)		–	–	64	64	16	64	64	N/A	N/A	64	
Mapping (CE | QRD | DT)		✓|–|–	–|✓|–	–|–|✓	✓|✓|✓	–|–|✓	–|✓|✓	–|–|✓	✓|✓|✓	✓|✓|✓	✓|✓|✓	
DT algorithm		–	–	K-Best	SIC	SD	FSD	SSFE	N/A	N/A	MMSE-NP	
Technology [nm]		65	180	130	90	130	40	130	65	90	65	
Area [mm^2]		0.68^{a,c}	2.81^a	N/A	2.02^a	26^d	306.82^d	N/A	16.06^d	32	8.88	
Gate count [KG]		325^{a,c}	152^a	340^a	505^a	N/A	4.5e5^d	71^a	5969^d	N/A	2760	1055^a
Frequency [MHz]		250	100	417	114	251	1150	277	400	400	500	
Power^b [mW]		154^{a,c}	51.2^a	55^a	59.07^a	624^d	323e3^d	20.48^a	219^d	240	549	307^a
Throughput^b	CE [MEst/s]	78	–	–	N/A	–	–	–			28.84	
	QRD [MQRD/s]	–	69.23	–	39.46	–	N/A	–	N/A	N/A	39.60	
	DT [Mb/s]	–	–	2000	N/A	163	10.58	134			454.26	
	Total [Mb/s]	–	–	–	947	–	–	–	10.8	150	367.88	
Area. Eff.^{b}	CE [kEst/s/kG]	240^a	–	–	N/A	–	–	–			10.45	27.34^a
	QRD [kQRD/s/kG]	–	455^a	–	78.14^a	–	N/A	–	N/A	N/A	14.35	37.54^a
	DT [kb/s/kG]	–	–	5882^a	N/A	N/A	0.0235^d	1890^a			165	431^a
	Total [kb/s/kG]	–	–	–	1875^a	–	–	–	1.81	N/A	133	349^a
Energy^b	CE [nJ/Est]	1.97^{a,c}	–	–	N/A	–	–	–			12.70	9.58^a
	QRD [nJ/QRD]	–	2.05^a	–	N/A	–	N/A	–	N/A	N/A	15.27	7.96^a
	DT [nJ/b]	–	–	0.055^a	N/A	1.32^d	3.79e6^d	0.304^a			0.99	0.62^a
	Total [nJ/b]	–	–	–	2.07^a	–	–	–	N/A	2.23	1.49	0.83^a

^{a} With data buffers excluded.
^{b} Normalized to 65 nm with 1.2 V core voltage: $f_{\text{clk}} \propto s$ and $P \propto (1/s)(1.2\,\text{V}/V_{\text{dd}})^2$, where $s = \text{Tech.}/65\,\text{nm}$.
^{c} Scaled up to 4×4 MIMO configuration: $\{A, P\} \propto d$, where $d = 4/\#\text{Rx-antenna}$.
^{d} Only counted relevant parts of the chip.

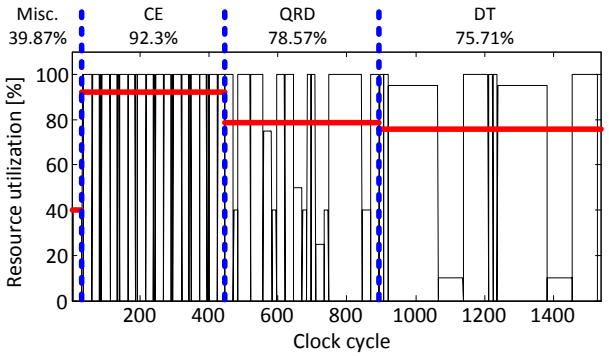

Fig. 10. Utilization of the SIMD core in Tile-0 during MIMO signal processing of two LTE-A resource blocks (24 subcarriers). Horizontal lines in the figure show the average utilization of the corresponding task.

task specific accelerators (ASICs), programmable platforms (e.g., FPGAs and GPUs), and domain-specific reconfigurable platforms (i.e., baseband processors).

1) Area efficiency: Area efficiency is evaluated by normalizing the throughput of each processing task to the corresponding hardware consumption. The proposed solution accomplishes three tasks within the tight timing constraint of the 20 MHz 4×4 MIMO 64-QAM LTE-A downlink, thanks to the architecture and algorithm co-design, which has more than 90% of total operations mapped onto the vector core for exploiting extensive DLP and attaining high resource sharing. Compared to other baseband processors [1] [5] [19], which adopt either lower dimensions of MIMO configurations or mapping of a single task, the cell array achieves the highest throughput and shows superior area efficiency. Besides, the processing throughput of the cell array is 2.8 times higher than that of the FPGA solution [4] and its area efficiency outperforms the GPU approach [25] by 4 orders of magnitude. Compared to ASICs [6], [22]–[24], 2−13.6 times less area efficiency is observed for each individual task mapping.

2) Energy efficiency: Besides the area and throughput evaluation, energy consumption per operation is another important measure for baseband processing. In comparison to related baseband processors, similar energy figures are observed. However, it should be pointed out that the cell array operates in a more complicated system setup (e.g., 4×4 MIMO vs. 2×2) and has more tasks assigned at the same time. Compared to ASICs, the cell array consumes 4−11 times more energy for performing each individual task, whereas a 1.3 times energy gain is obtained compared to the FPGA solution supporting only upto 16-QAM detection. Moreover, its energy efficiency outperforms the GPU approach by 6 orders of magnitude. Such high energy efficiency is achieved mainly by the hardware developments in the array: the architecture partitioning for attaining efficient vector and scalar processing, the hierarchical network topology for reducing communication costs, the vector processing enhancements and substantial register-access reduction for improving computation efficiency, and the flexible memory access schemes for relieving non-computational operations from processing cores.

VI. CONCLUSION

This paper presents an application-domain specific reconfigurable platform developed based on a heterogeneous cell array architecture. The efficiency of the proposed solution is exhibited by mapping three crucial MIMO processing blocks, namely channel estimation, channel matrix pre-processing, and

data detection, onto the processor, while the capability of real-time processing in a 20 MHz 4×4 MIMO 64-QAM LTE-A downlink is demonstrated. Achievements in area and energy efficiency are mainly enabled by algorithm-architecture co-developments, including unified and vectorized operations in algorithms, heterogeneous and hierarchical hardware resource deployments, vector processing enhancements, and flexible self-governed memory access schemes. Implementation results show that the proposed cell array platform is well positioned among the conventional architectures. It outperforms GPU platforms by 4−6 orders of magnitude in area and energy efficiency and reveals 1.3−2.8 times gain to FPGAs, and is 2−14 and 4−11 times less efficient than ASICs.

REFERENCES

[1] F. Clermidy *et al.*, "A 477mW NoC-Based Digital Baseband for MIMO 4G SDR," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb. 2010, pp. 278–279.

[2] P. Tan, Y. Wu, and S. Sun, "Link Adaptation Based on Adaptive Modulation and Coding for Multiple-Antenna OFDM System," *IEEE Journal on Selected Areas in Communications*, vol. 26, no. 8, pp. 1599–1606, Oct. 2008.

[3] C.-H. Yang and D. Marković, "A Flexible DSP Architecture for MIMO Sphere Decoding," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 10, pp. 2301–2314, Oct. 2009.

[4] X. Huang, C. Liang, and J. Ma, "System Architecture and Implementation of MIMO Sphere Decoders on FPGA," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, no. 2, pp. 188–197, 2008.

[5] V. Derudder *et al.*, "A 200Mbps+ 2.14nJ/b digital baseband multi processor system-on-chip for SDRs," in *IEEE Symposium on VLSI Circuits*, 2009, pp. 292–293.

[6] Po-Lin Chiu *et al.*, "A 684Mbps 57mW Joint QR Decomposition and MIMO Processor for 4×4 MIMO-OFDM Systems," in *2011 IEEE Asian Solid State Circuits Conference (ASSCC)*, Nov. 2011, pp. 309–312.

[7] M.-Y. Huang and P.-Y. Tsai, "Toward Multi-Gigabit Wireless: Design of High-Throughput MIMO Detectors With Hardware-Efficient Architecture," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 2, pp. 613–624, Feb. 2014.

[8] O. Edfors, M. Sandell, J.-J. van de Beek, S. Wilson, and P. Börjesson, "OFDM Channel Estimation by Singular Value Decomposition," *IEEE Trans. Commun.*, vol. 46, no. 7, pp. 931–939, July 1998.

[9] D. Wübben, R. Böhnke, V. Kühn, and K. D. Kammeyer, "MMSE Extension of V-BLAST Based on Sorted QR Decomposition," in *IEEE 58th Vehicular Technology Conference (VTC)*, vol. 1, 2003, pp. 508–512.

[10] C. Zhang *et al.*, "A Highly Parallelized MIMO Detector for Vector-Based Reconfigurable Architectures," in *IEEE Wireless Communications and Networking Conference (WCNC)*, Apr. 2013, pp. 3844–3849.

[11] L. Liu *et al.*, "Area-Efficient Configurable High-Throughput Signal Detector Supporting Multiple MIMO Modes," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 59, no. 9, pp. 2085–2096, Sept. 2012.

[12] "3GPP TS 36.101 V11.4.0: User Equipment (UE) radio transmission and reception (Release 11)," March 2013. [Online]. Available: http://www.3gpp.org/ftp/Specs/archive/36_series/36.101/36101-b40.zip

[13] L. Hyunseok, C. Chakrabarti, and T. Mudge, "A Low-Power DSP for Wireless Communications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 18, no. 9, pp. 1310–1322, 2010.

[14] C. Zhang *et al.*, "Reconfigurable Cell Array for Concurrent Support of Multiple Radio Standards by Flexible Mapping," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2011, pp. 1696–1699.

[15] J. Byrne, "Tensilica DSP Targets LTE Advanced," March 2011, http://www.tensilica.com/uploads/pdf/MPR_BBE64.pdf.

[16] R. Fasthuber *et al.*, "Exploration of Soft-Output MIMO Detector Implementations on Massive Parallel Processors," *Journal of Signal Processing Systems*, vol. 64, pp. 75–92, 2011.

[17] S. Khawam *et al.*, "The Reconfigurable Instruction Cell Array," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 16, pp. 75–85, 2008.

[18] A. Nilsson *et al.*, "An 11 mm^2, 70 mW Fully Programmable Baseband Processor for Mobile WiMAX and DVB-T/H in 0.12μm CMOS," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 90–97, Jan. 2009.

[19] J. Janhunen *et al.*, "Fixed- and Floating-Point Processor Comparison for MIMO-OFDM Detector," *IEEE J. Sel. Topics Signal Process.*, vol. 5, no. 8, pp. 1588–1598, 2011.

[20] H. Zhang *et al.*, "A 1-V heterogeneous reconfigurable DSP IC for wireless baseband digital signal processing," *IEEE J. Solid-State Circuits*, vol. 35, no. 11, pp. 1697–1704, 2000.

[21] Y. Xie, W. Wolf, and H. Lekatsas, "Code compression for embedded VLIW processors using variable-to-fixed coding," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 5, pp. 525–536, 2006.

[22] I. Diaz *et al.*, "Highly scalable implementation of a robust MMSE channel estimator for OFDM multi-standard environment," in *IEEE Workshop on Signal Processing Systems (SiPS)*, 2011, pp. 311–315.

[23] Z.-Y. Huang and P.-Y. Tsai, "Efficient Implementation of QR Decomposition for Gigabit MIMO-OFDM Systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 58, no. 10, pp. 2531–2542, Oct. 2011.

[24] M. Mahdavi and M. Shabany, "Novel MIMO Detection Algorithm for High-Order Constellations in the Complex Domain," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 5, pp. 834–847, 2013.

[25] S. Roger *et al.*, "Fully Parallel GPU Implementation of a Fixed-Complexity Soft-Output MIMO Detector," *IEEE Trans. Veh. Technol.*, vol. 61, no. 8, pp. 3796–3800, 2012.

Chenxin Zhang (S'09) received his M.S. degree in electrical engineering from Lund University, Sweden in 2009. He is currently working toward the Ph.D. degree in digital circuit design at the Department of Electrical and Information Technology at the same University. From Oct. 2012 to Feb. 2013, he was a visiting scholar at the Department of Electrical Engineering, University of California, Los Angeles. His research mainly focuses on developments of reconfigurable architectures for high computing performance and run-time flexible task mappings.

Liang Liu (S'10-M'12) received his B.S. degree in 2005 and Ph.D. degree in 2010 from Fudan University, China. From Jan. 2010 to Apr. 2010, he was with Electrical, Computer and Systems Engineering Department, Rensselaer Polytechnic Institute as a visiting scholar. From 2010 to 2014, he was a post-doc researcher with the Electrical and Information Technology Department, Lund University, Sweden. He is current an assistant professor in Lund University. His research interest is in the field of digital circuits design for wireless communication system.

Dejan Marković (S'96-M'06) received the Dipl.Ing. degree in electrical engineering from the University of Belgrade, Belgrade, Serbia, in 1998 and the M.S. and Ph.D. degrees in electrical engineering from the University of California, Berkeley, in 2000 and 2006, respectively. Since 2006, he has been with the Department of Electrical Engineering, University of California, Los Angeles, as an Associate Professor. His research is focused on robust integrated-circuit design, DSP architectures for wireless communications and neuroscience, and optimization methods.

Viktor Öwall (M'90) received the M.Sc. and Ph.D. degrees in electrical engineering from Lund University, Lund, Sweden, in 1988 and 1994, respectively. During 1995 to 1996, he joined the Electrical Engineering Department, the University of California at Los Angeles as a Postdoc. Since 1996, he has been with the Department of Electrical and Information Technology, Lund University, Lund, Sweden. He is currently full Professor at the same department and since 2009 the Head of Department. He is the Director of the VINNOVA Industrial Excellence Center in System Design on Silicon (SoS). His main research interest is in the field of digital hardware implementation, especially algorithms and architectures for wireless communication and biomedical applications.