Hindawi Publishing Corporation
International Journal of Digital Multimedia Broadcasting
Volume 2009, Article ID 503130, 13 pages
doi:10.1155/2009/503130

Research Article

3G Long Term Evolution Baseband Processing with Application-Specific Processors

Perttu Salmela,[1] Juho Antikainen,[2] Teemu Pitkänen,[1] Olli Silvén,[3] and Jarmo Takala[1]

[1] *Department of Computer Systems, Tampere University of Technology, P.O. Box 553, 33101 Tampere, Finland*
[2] *Centre for Wireless Communications, University of Oulu, P.O. Box 4500, 90014 Oulu, Finland*
[3] *Information Processing Laboratory, Department of Electrical and Information Engineering, University of Oulu, P.O. Box 4500, 90014 Oulu, Finland*

Correspondence should be addressed to Perttu Salmela, perttu.salmela@tut.fi

Received 13 November 2008; Accepted 6 January 2009

Recommended by Daniel Iancu

Data rates in the upcoming 3G long term evolution (LTE) standard will be manifold when compared to the current universal mobile telecommunications system. Implementing receivers conforming with the high-capacity transmission techniques is challenging due to the complexity and computational requirements of algorithms. In this study, the software defined radio (SDR) is targeted and the four essential baseband functions of the 3G LTE receiver, namely, list sphere decoding, fast Fourier transform, QR decomposition, and turbo decoding, are addressed and the functions are implemented as application specific processors (ASPs). As a result, the design space that describes the essential computational challenges of 3G LTE receivers is clarified and estimates of area, power, and interprocessor communication buffer requirements are presented.

1. Introduction

The upcoming 3G long term evolution (LTE) standard will support data rates up to 100 Mbps [1]. Such a high data rate will be achieved in 20 MHz bandwidth by using transmission techniques like orthogonal frequency division multiplexing (OFDM) [2], multiple-input multiple-output (MIMO) [3], that is, the use of multiple antennas, and an efficient forward error correction method, the turbo coding [4]. As these techniques are applied, the receiver needs to realize very sophisticated algorithms. The design complexity or difficulty of designing implementations of such algorithms calls for flexible software-based solutions, that is, software defined radio (SDR). On the other hand, the computational complexity of algorithms advocates dedicated hardware accelerators for maximizing the performance. Thus, the implementation technique of choice should possess the benefits of both approaches.

High throughput and efficiency can be achieved with highly parallel hardware accelerator which is designed for the application in hand. As a drawback, designing is time consuming and any further changes can be difficult with unprogrammable fixed hardware. Programmable processor-based implementations tend to suffer from a lower throughput, unused resources, and memory throughput bottlenecks but they allow a shorter development time and higher flexibility due to the programmability. A solution, which strives to achieve the benefits of both hardware accelerators and processor-based implementations, is to use application-specific processors (ASPs) with highly parallel computing resources. With proper tools, ASPs can be designed and programmed rapidly, yet high throughput can be obtained with highly parallel computing resources. Flexibility and efficiency are obtained with accurate control at software level.

On the contrary to focusing on the implementation of solely one function, even a couple of interoperating functions complicate the design. For example, the number of clock domains and the most suitable clock frequencies must be determined for all the functions. In addition, there is always a tradeoff between area and throughput. Furthermore, even if the throughput is adequate, the delay can be too long. Thus, the dimensions of the design space include clock frequency, area, power, parallelism,

number of processors, clock domains, and so forth. To find answers to the multivariable and multiobjective design problems, the design space must be explored by focusing on promising candidates, that is, design alternatives, and analyzing them. Naturally, such analysis is far away from evaluation of a fully functional system-on-chip (SoC) but it provides inevitable insight into the design problem in hand.

In this paper, efficient ASPs, whose performance rivals pure hardware implementations, are applied to the 3G LTE baseband processing. The targeted essential and computationally demanding baseband functions are list sphere decoding (LSD), fast Fourier transform (FFT), QR decomposition, and turbo decoding. Baseband functions are separated from system level operations as the area and power analysis focuses on the core computations. The assisting interprocessor communication (IPC) is analyzed in terms of data buffer requirements of ideal IPC links. The presented work forecasts how demanding the implementation of these baseband functions of the 3G LTE receiver would be, and what would be the number of logic gate equivalents (GE), power, number of processors, and IPC requirements with realistic clock frequencies. The results also show how strongly an efficient symbol detection method dominates the total complexity.

The next section introduces some previous implementation techniques and fundamentals of the addressed functions and system. In Section 3, a high-level description of the targeted receiver is presented. The applied ASP implementations are presented in Section 4. Multiprocessing requirements and complexity are analyzed in Section 5 before the conclusions.

2. Previous Work

The upcoming 3G LTE, MIMO-OFDM, and the main transmission parameters are discussed in depth in [1]. In [5], the fundamentals of MIMO communications, including the capacity gain, channel model, and receiver algorithms are explained. As an example of the high potential of MIMO-OFDM systems with sophisticated symbol detection, a 4×4 MIMO-OFDM system and maximum likelihood (ML) detection achieves over 1 Gbps throughput in [6]. The MIMO-OFDM is applied also in 4G telecommunications systems and WLANs. The entire baseband processing chain of a 4G SDR is addressed in [7]. A hardware implementation of MIMO-OFDM system for WLANs is presented in [8] and implementations of two vital functions, the matrix decomposition and symbol detection with sphere decoder, are considered in [9].

Typical DSPs like TI's C64x [10] are tempting candidates for baseband processing as they have parallel computing resources and special instructions suitable for many of the required tasks. For example, the FFT can be computed with an off-the-shelf library routine [11]. Alternatively, a dedicated FFT processor can be used [12], and with FPGAs, off-the-shelf IP cores can be used for the FFT [13]. In this paper, we have applied the FFT implementation presented in [14] for complexity and power estimations.

There are many alternative techniques and algorithms for QR decomposition. Since the MIMO receiver requires a relatively small matrix, extensively parallel systolic array processors [15, 16] can be oversized solutions. The QR decomposition requires the computation of a highly nonlinear operation, namely division by a norm or, alternatively, multiplication with an inverse of square root operation. One approach is to carry out the computations in $\log_2$ domain as in [17]. A Nios processor with CORDIC accelerators on FPGA is used in [18]. In [19], a scalable architecture using squared Givens rotations is presented. In this paper, the QR decomposition implemented in [20] has been applied.

In many practical MIMO systems, the ML symbol detection can be too complex. Alternatively, for example, zero forcing or linear minimum mean square error (LMMSE) principles can be applied [21]. In this paper, LSD is assumed as it approximates the ML detection with reduced computational complexity. There are several LSD variants. A K-best LSD is assumed in this study and in [22–24] where architectures for the algorithm are presented. The K-best LSD processor used in this paper is presented in detail in [25].

Turbo decoder can be implemented, for example, as a coprocessor of a DSP as in [26] or a hardware accelerator [27] or an ASP [28]. Naturally, there are variants of the algorithm, and the level of parallelism and clock frequency mainly determine the throughput. In this paper, a programmable turbo decoder presented in [29] is applied.

The ASP template, which is applied in this paper, uses the transport triggered architecture (TTA) [30]. There exists many multiprocessor systems applying TTA processors. In [31], a simple asynchronous communication link between TTA processors is enabled with units containing an FIFO buffer. TTA and LEON3 processors are connected with an AMBA bus in [32]. On the contrary to a shared bus, a network-on-chip approach has been applied in [33] where two Coffee RISC processors, a TTA processor, and a shared memory are connected with a network. A bioinspired multiprocessor system is presented in [34, 35] where TTA processors are abstracted as cells of a biological system. In this paper, the IPC requirements of a multiprocessor system are analyzed, and an abstract multiprocessor system using shared memory banks as communication links is assumed.

Several inevitable building blocks for baseband processing are presented in the aforementioned references. On the contrary to focusing solely on one particular function without practical motivation for the achieved throughput, we focus on a baseband processing chain consisting of FFT, QR decomposition, LSD, and turbo decoder and we derive the processing requirements from the 100 Mbps peak data rate of the upcoming 3G LTE systems. In this paper, we consider especially the ASPs in [14, 20, 25, 29] and their applicability for baseband processing. In order to obtain realistic estimates, the considered ASPs are resynthesized for the prevailing operating conditions, complexity, and power estimates are given for a 3G LTE compliant system configuration.

3. System Model

A high-level description of the targeted 2-antenna MIMO-OFDM receiver is presented in Figure 1. The input ports are connected to radio-frequency functions of the receiver. The functional block diagram is only a high-level model as it does not suggest how the functions should be mapped to the processors nor it does not suggest how data is passed between the functions and whether the data vectors have serial or parallel presentations. In the following, the targeted transmission techniques are presented briefly.

3.1. Orthogonal Frequency Division Multiplexing. OFDM uses the frequency spectrum efficiently as the used frequency band is divided into several orthogonal subcarriers. OFDM uses the FFT and inverse FFT (IFFT) for efficient conversions between the time and frequency domains. The time domain signal is generated in the transmitter side with inverse transform

$$X_{\mathrm{T}} = \mathrm{IFFT}(X_{\mathrm{F}}), \tag{1}$$

that is, data belonging to several parallel subcarriers is fed to the IFFT. In the receiver side, parallel subcarriers, X_{F}, are extracted from the time domain signal X_{T} with

$$X_{\mathrm{F}} = \mathrm{FFT}(X_{\mathrm{T}}). \tag{2}$$

To alleviate timing synchronization, additional cyclic prefix is inserted to the signal. The channel estimation can be alleviated with pilot symbols.

In the receiver side, distortion of the channel can be equalized conveniently in frequency domain by multiplying the received symbols with equalizing factors. Before the FFT, the cyclic prefix must be removed from the signal, and timing synchronization is responsible for feeding the time domain signal, whose length equals the FFT length, with correct timing offset to the FFT block.

3.2. Multiple-Input Multiple-Output. In a spatial multiplexing MIMO system, multiple antennas are used to transmit independent data streams. Spatial multiplexing gain, that is, increase in capacity, is proportional to the number of antennas and it does not require extra power nor bandwidth. Two transmit and receive antennas are a highly probable configuration for the first 3G LTE systems, since a higher number of antennas increases the computational requirements of symbol detection significantly.

Computational complexity of ML detection of transmitted symbols depends exponentially on the number of spatial channels. Therefore, even with a modest number of antennas, simpler approximative methods must be used. The usage of list sphere decoding algorithms is tempting as they can achieve higher performance than LMMSE [36], even though they are computationally demanding. The sphere detector restricts the search space by evaluating only the symbols inside the sphere centered in the received symbol. In the system model in Figure 1, K-best LSD is assumed. The K-best LSD operates by gradually increasing the dimension of the symbol vector. At each level, a list of the K best partial solutions is selected for continued processing.

In principle, an MIMO system with a complex-valued channel matrix, $\mathbf{H}$, noise vector, $\mathbf{n}$, transmitted symbol, $\mathbf{s}$, and received symbol, $\mathbf{y}$, can be described with

$$\mathbf{y} = \mathbf{H}\mathbf{s} + \mathbf{n}. \tag{3}$$

The number of receive and transmit antennas equals the numbers of rows and columns of $\mathbf{H}$, respectively. The transmitted symbol $\mathbf{s}'$ can be estimated by ML detection by solving

$$\mathbf{s}' = \arg\min_{\mathbf{s}} ||\mathbf{y} - \mathbf{H}\mathbf{s}||^2, \tag{4}$$

which gives the optimal result. However, solving (4) is intractable with multiple antennas and large constellations.

Instead of solving (4), the symbol estimation can be simplified by using QR decomposition of $\mathbf{H}$. With this practice, the computational complexity is lowered. Instead of ML detection, a substitute

$$\mathbf{s}' = \arg\min_{\mathbf{s}} ||\mathbf{y}' - \mathbf{R}\mathbf{s}||^2 \quad \text{where } \mathbf{y}' = \mathbf{Q}^H\mathbf{y} \tag{5}$$

is used. As the $\mathbf{R}$ is in upper triangular form, approximation of $\mathbf{s}'$ is computationally simpler with the aid of (5). The simplified approximation is based on computing the Euclidean distance in (5) by gradually increasing the dimensions of the symbol vector. Basically, there will be partial solutions which are too far away from the received symbols and when such partial solutions are discarded, the search space is efficiently limited. The K-best LSD applies the aforementioned principles by maintaining a K-length list of the best partial solutions found so far.

3.3. Forward Error Correction. The function of the forward error correction is to introduce redundancy in the transmitted signal in order to alleviate error detection and correction. In 3G LTE, a similar turbo coding as in the contemporary 3G systems will be used. The only difference is the definition of the interleaving function [37, 38]. The new interleaving function covers longer code blocks and it is simpler to implement than the contemporary 3G interleaving. Naturally, the longer code block size affects the memory requirements.

Turbo decoding is an iterative process, which runs a soft-in soft-out (SISO) component decoder several times. The arguments of the component decoder are extrinsic information $\boldsymbol{\lambda}^{\mathrm{in}}$, systematic bit, $\mathbf{y}^{\mathbf{s}}$, and parity bit vector, $\mathbf{y}^{p}$. As a result, it generates new extrinsic information, $\boldsymbol{\lambda}^{\mathrm{out}}$, and soft bit estimate vectors, $\mathbf{L}$, that is,

$$(\boldsymbol{\lambda}^{\mathrm{out}}, \mathbf{L}) = \mathbf{f}_{\mathrm{SISO}}(\boldsymbol{\lambda}^{\mathrm{in}}, \mathbf{y}^{\mathbf{s}}, \mathbf{y}^{p}). \tag{6}$$

The *a posteriori* information is generated on the previous half iteration, and used as *a priori* information on the next half iteration. The information exchange takes place by passing the extrinsic information between the component decoder processes. The main difference between the half

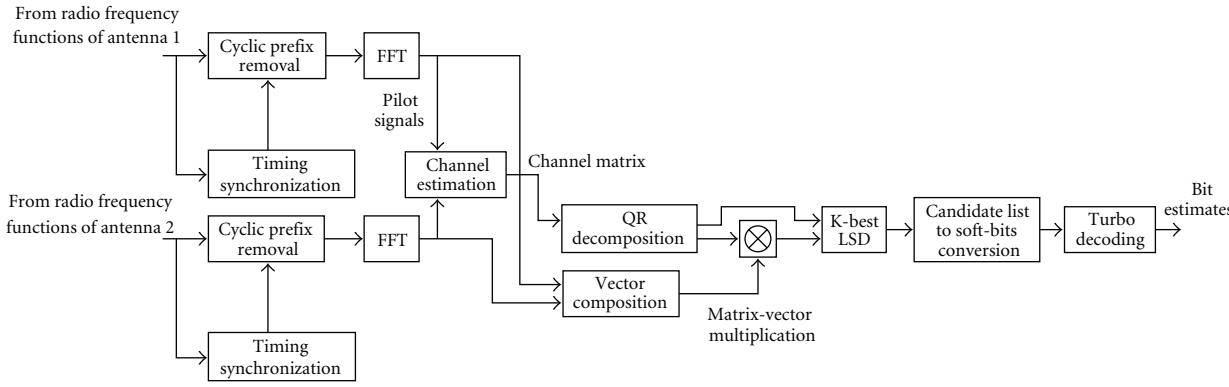

Figure 1: A simplified block diagram of baseband processing of a two-antenna MIMO-OFDM receiver using K-best LSD for symbol detection. ASP implementations for FFT, QR decomposition, list sphere detection, turbo decoding, and multiplications are considered in this paper.

iterations is that every second half iteration processes data related to the interleaved systematic bits.

The applied turbo decoder processor in Section 4.6 uses the max-log-MAP algorithm for SISO decoding. In principle, max-log-MAP algorithm generates the forward path metric at state u at trellis stage k, $\alpha_k(u)$ recursively as

$$\alpha_k(u) = \max_{u'}(\alpha_{k-1}(u') + d_k(u', u)), \tag{7}$$

where $d_k(u', u)$ is the branch metrics. The backward path metric is defined in the same way as

$$\beta_{k-1}(u') = \max_{u}(\beta_k(u) + d_k(u', u)). \tag{8}$$

The soft output, L_k, is a function of the forward, backward, and branch metrics, that is,

$$\begin{aligned} L_k = &\max_{u', u:x^s=0}(\alpha_{k-1}(u') + \beta_k(u) + d_k(u', u)) \\ &- \max_{u', u:x^s=1}(\alpha_{k-1}(u') + \beta_k(u) + d_k(u', u)). \end{aligned} \tag{9}$$

In (9), the first maximum corresponds to the state transitions where the transmitted systematic bit $x^s = 0$, and the second maximum is computed based on all the state transitions where $x^s = 1$. The signum function is used to calculate the final hard bit estimates based on L_k. The new extrinsic information λ_k^{out} is computed with the aid of the received soft systematic bit, y_k^s, *a priori* information, λ_k^{in}, and L_k, that is,

$$\lambda_k^{\text{out}} = \frac{1}{2}L_k - y_k^s - \lambda_k^{\text{in}}. \tag{10}$$

4. Transport Triggered Architecture Processor Implementations

The targeted baseband functions are implemented on a customizable ASP template. The implementations are presented shortly in the following sections.

4.1. Principles of Transport Triggered Architecture Processors. In this paper, TTA [30] has been used as the architecture template for ASPs. Processors with similar efficiency and performance could be implemented also with some other ASP templates supporting sufficient parallelism and customizability. Since there exists up-to-date tool support for TTA processors [39], we have exploited the template and the baseband functions have been implemented with TTA processors.

The main difference when compared to a pure hardware solutions is that the TTA processors are fully programmable. TTA reminds VLIW machine but the interconnection is exposed to the programmer unlike in traditional processors. TTA is one form of application-specific instruction set processor where the instruction set of the processor is tailored for the given application. In this sense, code for customized TTA processor is not compatible with another TTA processor. In TTA, the computations are triggered by data transported to the computing unit, which is contrary behavior to conventional operation-triggered architectures. The processor is programmed with data transports, which reflects the architecture to the programmer. The maximum number of parallel data transports is determined by the number of buses of the interconnection network. As the interconnection network connecting the computing resources is visible to the programmer, there is accurate control of all the operations.

The modularity of TTA processors allows to tailor them by including only the necessary function units (FU). Application-specific functions are implemented as user defined special FUs (SFU) which are utilized in a similar way as conventional FUs, that is, by transporting data on assembly level or by using function-like macros in C language. Due to frequent direct data transports between the FUs or SFUs, the register pressure is very low. However, the modularity of the processor allows a variable number of register files (RF) with variable numbers of input and output ports. In Figure 2, a high-level example of a TTA processor is given. The figure highlights the modular and customizable structure of the processor by denoting the variable numbers of the respective resources. The control unit (CU) in Figure 2 allows data transports to access the program counter and the return address register, which is required for jump or call operations.

The load on the buses of the interconnection network can be lowered by excluding the unnecessary connections if the work load of the processor is known beforehand. In this case, the targeted application program determines which connections are used. Typically, one application requires only a fraction of all the possible connections between the computing resources. If any other application is run on the same processor, it must be able to use the same connections. As a consequence of the limited connectivity and lowered load on the buses, the maximum clock frequency of the interconnection network is raised.

4.2. Multiprocessor Systems with TTA Processors. There exists many multiprocessor systems applying TTA processors as listed in Section 2. However, the required number of processors for baseband processing in Section 5 is far higher than the number of processors in [31–33]. In addition to the bioinspired abstraction of multiple TTA processors [34, 35], multiple processors could be also abstracted as a hierarchical structure where the SFUs would be comprised of TTA subprocessors. Another way would be to combine all the TTA processors to a set of loosely connected clusters inside a single TTA processor. However, assembly programming such a processor would be error prone due to the extremely long instruction word and the scheme would limit the control flow of the clusters very strictly to a single combined flow. Regardless of the applied structure of the multiprocessor system, generating and controlling a multiprocessor system consisting of dozens of processors would be a demanding task.

Since it would be uneconomical to produce results of computations faster than they can be transferred to the next stage, shared memory banks or RFs running with the same clock frequency, f_i, as the processors must be assumed for the IPC at the lowest level. Fortunately, the applied TTA processor template has flexible memory interfaces, which can simplify the IPC. For example, simple point-to-point connections between two processors could be implemented with an SFU interfacing a shared single- or dual-port memory. Furthermore, if complex address generation or bank selection is required, it can be included to the same SFU, which slightly raises the abstraction level of the IPC visible to the programmer. Such an incorporation of all the memory related logic to the same unit could enable a seamless IPC.

4.3. FFT Processor. The applied FFT TTA processor is presented in detail in [14]. The processor implements mixed-radix FFT consisting of radix-2 and radix-4 computations and it supports several power-of-two transform sizes. It has 11 RFs containing 25 general-purpose registers and three Boolean registers, 17 buses in the interconnect network, a conventional adder, a comparison unit, and two-load/store units. The main computations are carried out with the following SFUs.

Complex Adder Unit. It supports four different summations composed of four alternative operands.

Complex Multiplier. It alleviates the butterfly operation with four real multipliers and two real adders.

Address Generator Unit . It generates two addresses with bitwise reversal and rotation operations.

Coefficient Generator. It generates the twiddle factors instead of loading them from a memory.

The processor applies a complex-valued number presentation where the real and imaginary parts both take 16 bits. Data is stored in single-port memory banks and the kernel loop applies the principles of software pipelining. Code compression is applied to enhance the code density and lower the power consumption.

4.4. QR Decomposition Processor. The QR TTA processor presented in [20] is based on the modified Gram-Schmidt algorithm [40]. With complex-valued arithmetic units the processor can compute equally well both the complex- and real-valued decompositions. The only conventional units of the processor are the two-load/store units and an RF consisting of five general purpose registers. The interconnection network contains seven buses. The applied SFUs are as follows.

Complex Adder/Subtractor Unit. It is for native complex-valued computations.

Complex Multiplier Unit. It can optionally conjugate the other input. The conjugation is required for the computation of the real-valued norm.

$1/\sqrt{x}$ unit is for a fast estimation of the highly nonlinear function. The function is used in the QR decomposition to avoid division operations.

As the processor has a bit accurate complex multiplier, it can be used also for other tasks where the accuracy of 16-bit fixed-point number system is sufficient. The $1/\sqrt{x}$ unit and the multiplier can be used also for computation of square root as $x(1/\sqrt{x}) = \sqrt{x}$.

4.5. K-Best LSD Processor. The LSD TTA processor in [25] generates a 16-element list of candidate solutions to approximate the transmitted symbol $\mathbf{s}'$ in (5). The processor uses 16-bit arithmetic and it is targeted for 2×2 antennas and 64-quadrature amplitude modulation (QAM). Instead of 2×2 complex-valued matrix, a real-valued matrix with doubled dimensions is processed. Therefore, a real-valued 4×4 QR decomposition is required for the LSD. The interconnection network is very sparse and contains 16 buses. The arithmetic operations are computed with two addition units, a subtraction unit, a multiplier, and a squaring unit. The following SFUs are targeted for the applied K-best algorithm.

Insertion Sorter Unit. It sorts a list of 16 samples according to the partial Euclidean distances (PED). Internally, the list

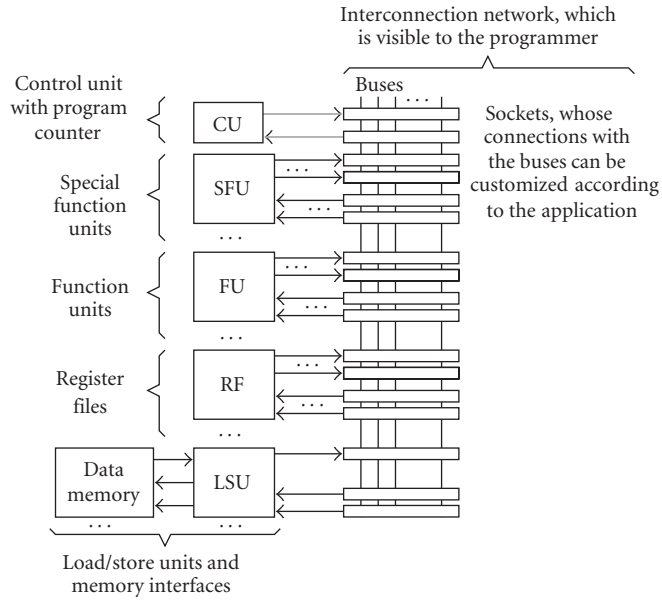

Figure 2: TTA processors consist of a CU and variable number of FUs, SFUs, RFs, and LSUs. Unused connections between the resources can be excluded from the interconnection network.

is kept in a shift register and the new value is inserted to the register pointed by comparison logic.

PED Extractor Unit. It extracts the PED from the internal storage format, that is, the unit accesses bits by hardwiring.

Multiplexer and Look-Up-Table Unit. It consists of a multiplexer selecting the bits, which index the look-up-table. In principle, the unit converts a bit pattern to fixed-point format.

Storage Format Composer Unit. It composes a 28-bit word consisting of symbol information and the corresponding PED.

There are three RFs of sizes 16, 10, and 4 registers. On the contrary to conventional processors, the LSD TTA processor does not have load/store units nor data memory, since there is no need for accessing large arrays. The input data is passed via two RFs and the results of the computations are available in the registers of the insertion sorter SFU.

4.6. Turbo Decoder Processor. The turbo decoder TTA processor is presented in [29]. It has a sparsely connected interconnect network of 30 buses and the high number of buses is a consequence of high parallelism. The only conventional FUs are the addition and comparison units. There are only two RFs, both of them containing one general purpose register. As there are not many conventional FUs, the applied max-log-MAP algorithm is computed solely with the following SFUs.

Control Unit. It generates a control word which is used as an argument to all the other SFUs.

Address Generator. It generates addresses for accessing the branch metric buffer.

Forward Process Unit. It computes forward path metrics according to (7).

Backward Process Unit. It computes backward path metrics as defined in (8) and extrinsic information and soft output bit estimates according to (10) and (9), respectively.

Branch Metric Generator. It generates and buffers the branch metrics for the forward and backward processes.

The turbo decoder TTA processor applies high parallelism as it processes one trellis stage in 1.016 clock cycles on average, that is, both forward and backward path metrics are computed in one clock cycle. Such a high parallelism requires also a high memory throughput. Therefore, the processor does not have conventional load/store units. Instead, the SFUs access memory interfaces of the processor directly. As (7)–(10) indicate, the main computations in the SFUs are carried out with basic arithmetic, add-compare-select, and maximum operations. The processor includes memory bank selection, address generation, and access buffer logic to allow parallel interleaved accesses of the extrinsic information with four-single-port memory banks. The interleaving function is excluded from the processor and it is accessed via external interface of the processor.

5. Processing Requirements and Complexity

The number of processors, their total area and memory requirements, and interprocessor communication requirements are derived from the targeted 100 Mbps throughput.

5.1. Time and Throughput Requirements. There are seven OFDM symbols per transmit antenna in 0.5 millisecond time frame in 3G LTE downlink. Thus, the processing time requirement $T_{\text{FFT}} = 0.5$ millisecond/7 = 71 microseconds includes also the additional time contributed by the cyclic prefix of the OFDM symbol. The FFT must be computed for both antennas.

The QR decomposition must be processed in the coherence time, T_{coh}, of the channel. If bullet train speed $v_r = 500$ km/h is assumed for the receiver, the coherence time is $T_{\text{coh}} = c/(Fv_r) = 0.9$ millisecond where c is the speed of light and $F = 2.4$ GHz is the carrier frequency. However, with a more rapidly varying channel, the QR decomposition must be computed more frequently, that is, shorter T_{coh} must be used in (12). A single QR decomposition combines information from all the antennas. In other words, the matrix and vector sizes of the QR decomposition depend on the number of antennas.

The LSD must be computed for each subcarrier. So, the time requirement equals to the time requirement of the FFT. However, even if the maximum length of the FFT is 2048, only 1201 subcarriers are in use. A single LSD processes the signals of both antennas, that is, it outputs estimates of symbols transmitted from both antennas.

Since the turbo decoder processes soft bits instead of QAM symbols, it is meaningful to express throughput as data rate. The throughput requirement of turbo decoding equals the maximum data rate of 100 Mbps. Naturally, with code rate $R = 1/2$ and 64-QAM symbols, the data rate on the LSD side is 200 Mbps and symbol rate 33.3 Msps.

5.2. Required Number of Clock Cycles. The FFT TTA processor in [14] takes 12332 clock cycles for the 2048-point transform and the transform must be computed for both antennas. So, the required clock cycles of the FFT task are

$$C_{\text{FFT}} = 2 \times 12332 = 24664. \tag{11}$$

The QR decomposition algorithm is of order $O(n^3)$ and the QR decomposition TTA processor in [20] takes 139 clock cycles for a 4×4 matrix. The dimensions of the decomposed matrix are doubled, since the LSD TTA processor applies real-valued computation. Since the **Q** matrix is the argument of matrix-vector product in (5), the products are mapped to the same processor. The products must be computed continuously for each received symbol vector, but the QR decomposition only once in the coherence time. So, the average number of clock cycles in T_{FFT} time period, for both computations is approximately

$$C_{\text{QR\_avg}} = 1201 \times \left(139 \times \left(\frac{T_{\text{FFT}}}{T_{\text{coh}}}\right) + 16\right) = 32386, \tag{12}$$

where 4×4 matrix multiplication takes 16 clock cycles. Naturally, with more rapidly varying channel, the $C_{\text{QR\_avg}}$ increases as the T_{coh} must be decreased. The products take approximately 59% of the $C_{\text{QR\_avg}}$. The maximum number of clock cycles is spent when the decomposition of a new channel matrix is computed for each subcarrier, that is,

$$C_{\text{QR}} = 1201 \times (139 + 16) = 186155. \tag{13}$$

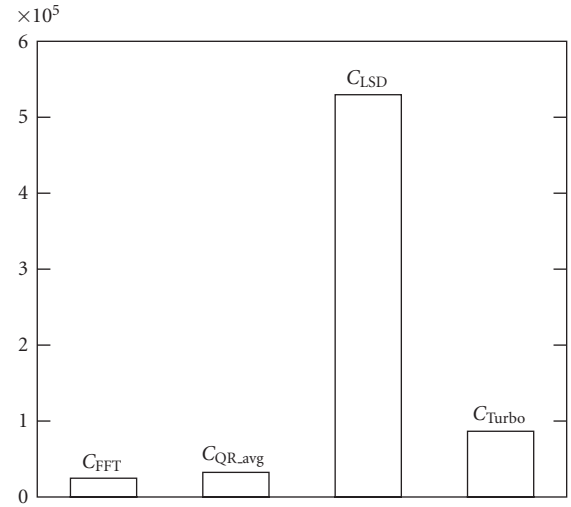

Figure 3: Required number of clock cycles of the processing tasks in $T_{\text{FFT}} = 71$ microseconds time frame.

The average number of clock cycles, $C_{\text{QR\_avg}}$, is only 17% of the maximum, C_{QR}.

The LSD TTA processor in [25] takes 441 clock cycles for processing one symbol vector. Thus, in T_{FFT} time period the number of required clock cycles for the LSD, C_{LSD}, is approximately

$$C_{\text{LSD}} = 1201 \times 441 = 529641. \tag{14}$$

Fortunately, the LSD can be parallelized among the subcarriers.

In order to compare turbo decoding with the other baseband functions, the clock cycles of turbo decoding must be normalized to clock cycles, C_{Turbo}, taken in T_{FFT} time frame. The turbo decoder TTA processor in [29] takes 1.016 clock cycles per trellis stage processed in half iteration. With six iterations, each trellis stage is processed 12 times. Therefore,

$$C_{\text{Turbo}} = T_{\text{FFT}} \times 100 \times 10^6 \times 12 \times 1.016 = 86563, \tag{15}$$

where the first multiplications $T_{\text{FFT}} \times 100 \times 10^6$ express how many bits are processed in T_{FFT}. Turbo decoding can be parallelized to several processors with block-by-block pipelining where each processor decodes a code block of its own independently.

The required number of clock cycles of all the four functions are illustrated in Figure 3. The figure shows clearly how the LSD dominates the computation load. Obviously, the requirements cannot be met with single-processor systems with currently achievable clock frequencies.

5.3. Number of Processors. The required minimum number of processor is determined by the throughput per processor, clock frequency, f_i, and parallelization scheme of the targeted functions. If a task i can be parallelized to several processors and the throughput is directly proportional to the number of processors, then the minimum required number of

processors, P_i, of the task i taking C_i clock cycles in time frame T_{FFT} is

$$P_i = \left\lceil \frac{(C_i/T_{\text{FFT}})}{f_i} \right\rceil. \tag{16}$$

The utilization, U_i, of the processor, P_i, dedicated to task i tells how efficiently the computing resources are used. It can be defined in a similar way as

$$U_i = \frac{C_i}{(P_i T_{\text{FFT}} f_i)}. \tag{17}$$

Naturally, $100 \times (1 - U_i)$ tells how many percent of the time the processor P_i idles. For the QR decomposition and matrix-vector product task, the average number of clock cycles, $C_{\text{QR\_avg}}$, is used to calculate the minimum number of processors and utilization. The total utilization of the whole processing chain can be computed as

$$U = \sum_{i \in S_{\text{tasks}}} \frac{C_i}{\left(T_{\text{FFT}} \sum_{i \in S_{\text{tasks}}} P_i f_i\right)}, \tag{18}$$

where the sums are computed for all the elements of the task set $S_{\text{tasks}} = \{\text{FFT, QR\_avg, LSD, Turbo}\}$. The total utilization in (18) expresses the ratio between the required execution cycles of all the tasks and the available execution cycles of all the processors.

5.4. Delay. The delay of a task depends on the maximum size of the processed data vector and the scheduling. Except for the first half iteration, the turbo decoder requires that the whole code block is received before decoding. The maximum code block length is 6144 [37], which is about 20% longer than in the current 3G systems. With code rate $R = 1/2$, the required number of soft bits is naturally $2 \times 6144 = 12288$. For two OFDM symbols, the LSD generates symbol candidate lists, which can be converted to $2 \times 1201 \times 6 = 14412$ soft bit estimates with 64-QAM (6 bits per symbol). Since the number of soft bits exceeds the required number for the maximum code block length, the analysis of the delay of FFT and LSD can be limited to the processing of two OFDM symbols.

With at maximum two processors, the delay of the FFT is simply

$$D_{\text{FFT}} = \frac{C_{\text{FFT}}}{(P_{\text{FFT}} f_{\text{FFT}})}, \quad P_{\text{FFT}} \in \{1, 2\}, \tag{19}$$

and in a similar way the delay of the LSD is

$$D_{\text{LSD}} = \frac{C_{\text{LSD}}}{(P_{\text{LSD}} f_{\text{LSD}})}, \tag{20}$$

where $P_{\text{LSD}} \in \{1, 2, \ldots, 1201\}$ as the LSD can be parallelized among the subcarriers. The QR decomposition processor has two tasks, the QR decomposition and the matrix-vector products, of which the QR decomposition is computed only once in the coherence time, $t_{\text{coh}} = 0.9$ millisecond. Thus, the worst-case delay when both tasks are computed is

$$D_{\text{QR}} = \frac{C_{\text{QR}}}{(P_{\text{QR}} f_{\text{QR}})}, \tag{21}$$

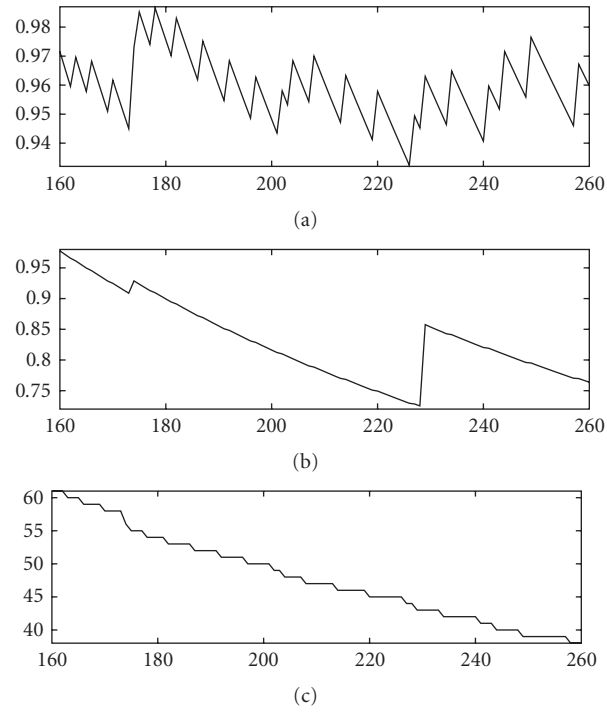

Figure 4: Configurations as a function of f_i with single clock domain: (a) total utilization, (b) total delay in millisecond, (c) the number of processors. The x-axis denotes f_i in MHz.

where $P_{\text{QR}} \in \{1, 2, \ldots, 1201\}$ as the decompositions and multiplications can be parallelized among the subcarriers. For an average delay, $C_{\text{QR\_avg}}$ can be used in a similar way. The delay of turbo decoding is determined by the maximum code block size, 6144. Thus, the delay with six turbo iterations is

$$D_{\text{Turbo}} = \frac{6144 \times 6 \times 2 \times 1.016}{f_{\text{Turbo}}}, \tag{22}$$

where processing one trellis stage with the turbo decoder TTA processor takes on average 1.016 clock cycles. Distributing the turbo decoding to several processors with block-by-block pipelining would affect only the throughput but not the delay and, therefore, the number of processors is omitted from (22).

5.5. TTA Processor Configurations as Function of Clock Frequency. Utilization, delay, and number of processors are analyzed in Figure 4 as functions of clock frequency. The total utilization in Figure 4(a) shows that the utilization is always greater than 0.93 in the explored clock frequency range. High utilization can be obtained easily, since the LSD dominates the computational load and it can be parallelized with very fine granularity. In other words, since the utilization of the LSD task is always high, also the utilization of the whole processing chain is relatively high. The peaks in the utilization occur, when the number of processors of some task can be decremented. In that case, the utilization grows. On the contrary, if the number of processors remains untouched and the clock frequency is

Table 1: The baseband processing chain with TTA processors, 2×2 antennas, 1201 subcarriers, 64-QAM, 6144-length turbo code block, list length $K = 16$, data rate 100 Mbps.

Clk. freq. f_i (MHz)	Task i	No. of procs. P_i	Util. U_i	Delay D_i (ms)	Area (kGE)	Area $\times P_i$	Power est. (mW)	Power est. $\times P_i \times U_i$	Tech. (μm)	Ref.
250	FFT	2	0.69	0.049	30.5	61.0	36.3	50.1	0.13	[14]
250	Turbo dec.	5	0.98	0.300	35.1	175.5	50.8	248.9	0.13	[29]
250	QR & prod.	2	0.91	0.372	17.7	35.4	13.1	23.8	0.13	[20]
250	LSD	30	0.99	0.071	23.6	708.0	20.8	617.8	0.13	[25]
Total		39	0.97	0.792		979.9		940.6		

Table 2: An example baseband processing chain with 2×2 antennas, 1201 subcarriers, 16-QAM, 4804-length turbo code block, data rate 68 Mbps.

Clk. freq. f_i (MHz)	Task i	No. of units P_i	Util. U_i	Delay D_i (ms)	Area (kGE)	Area $\times P_i$	Power est. (mW)	Power est. $\times P_i \times U_i$	Tech. (μm)	Ref.
600 & 300	FFT & Turbo	5	0.36	0.396	—	—	718	1303	0.13	[26]
223	QR	1	0.29	0.259	198	198	—	—	0.13	[9]
213	Sphere Decoder	1	0.92	0.065	61	61	—	—	0.13	[9]
Total		7	0.38	0.720		—		—		

increased the utilization decreases. The discontinuations of delay in Figure 4(b) originate from the same phenomenon. The greatest discontinuation at 229 MHz takes place as the QR decomposition is mapped from three to two processors. The number of processors in Figure 4(c) decreases quite steadily, since it is dominated by the LSD task, which requires the largest number of processors.

5.6. Analysis. An example configuration of TTA processor-based baseband processing chain is presented in Table 1. A single clock domain with $f_i = 250$ MHz is applied and the processors have been synthesized with 0.13 μm technology for obtaining complexity and power estimates. The area and power estimates exclude the memories. The power estimates are scaled with the number of respective processors and their utilization in the ninth column of Table 1. The results in Table 1 show that since the LSD task takes only 441 clock cycles per subcarrier and it can be computed for each subcarrier independently, the task can be easily divided among several processors to achieve a high utilization. On the contrary, it is more difficult to obtain very high utilization for both the FFT and the QR processors with the same clock frequency, as the granularity of the tasks is more coarse. As a second remark, the delay of the QR decomposition is long when compared to other functions, even though the other functions are more complex. However, the QR decomposition must be computed only once in the coherence time $t_{\text{coh}} = 0.9$ millisecond, that is, the delay in Table 1 is the worst case delay. On average, the delay of the QR decomposition and the matrix-vector products is only 17% of the delay in Table 1.

In principle, the FFT and QR tasks could be mapped to the same processor. The processor should be formed as a hybrid of both processors in this case. Since both functions require complex arithmetic, the same resources could be shared efficiently. With $f_i = 402$ MHz, both tasks could be mapped to two hybrid FFT/QR TTA processors and a utilization, $U_{\text{FFT/QR}} = 1.00$, would be obtained.

Mapping the turbo decoding and some other function to the same processor could not benefit as much from sharing the resources, since the turbo decoding requires mostly real-valued add-compare-select operations. Shortening the delay of the turbo decoding is difficult for two reasons. Firstly, turbo decoding is an iterative process where the previous iteration must be finished before the next one can begin. Secondly, the component decoder applying the radix-2 algorithm processes at maximum one trellis stage in one clock cycle. The next path metrics cannot be computed according to (7) and (8) before the previous ones are computed. For these reasons, increasing the clock frequency or applying the radix-4 algorithm are the only ways to shorten the delay of the turbo decoding task in Table 1.

To illustrate more deeply the computational requirements of the baseband processing, example configurations consisting of other implementations are shown in Tables 2–4. As the respective implementations in Tables 2–4 are not necessarily targeted to the 3G LTE system or they are not targeted to operate among each other, the Tables 1–4 should be not considered as comparisons of TTA processors and other implementations. Instead, the tables show indicative example configurations of baseband processing chains.

For some implementations, all the required information is not available or it is given with different units. The area is reported if it has been given as the GEs. For some implementations, the performance data is not available for the targeted configuration of 2048-length FFT, 2×2

Table 3: An example baseband processing chain with 4×4 antennas, 601 subcarriers, 16-QAM, 4808-length turbo code block, list length $K = 10$, data rate 68 Mbps.

Clk. freq. f_i (MHz)	Task i	No. of units P_i	Util. U_i	Delay D_i (ms)	Area (kGE)	Area $\times P_i$	Power est. (mW)	Power est. $\times P_i \times U_i$	Tech. (μm)	Ref.
45	FFT	2	0.63	0.045	—	—	480	608.45	0.35	[12]
400	Turbo dec.	5	0.82	0.288	64.1	320.5	—	—	0.065	[28]
80	QR	1	0.54	0.489	—	—	—	—	0.25	[8]
50	LSD	2	0.85	0.060	132	264	—	—	0.13	[23]
Total		10	0.80	0.882		—		—		

Table 4: Requirements of 4G baseband processing chain for 100 Mbps data rate [7].

Assumed clk. freq (MHz)	Task i [7]	MCycles/s [7]	Assumed no. of Procs. P_i	Util. U_i
360	FFT	360	1	1.00
240	STBC	240	1	1.00
385	LDPC	7700	20	1.00
Total		8300	22	1.00

Table 5: Area of the core processor without memories and data memory requirements of the processors.

TTA processor	Clk. freq. f_i (MHz)	Area (kGE)	Data memory requirements (kbits)
FFT	250	30.5	65.5 divided into 2 single-port memory banks
QR	250	17.7	1.5 dual-port memory
LSD	250	23.6	0.0 (uses only registers)
Turbo decoder	250	35.1	281.7 divided into 16 single-port memory banks

Table 6: Additional buffer memory requirements for seamless IPC.

IPC buffer	Memory (words)	Memory (kbits)
FFT: next input	2×2048	131.1
FFT: prev. result	2×2048	131.1
QR: $\mathbf{R}$, $\mathbf{Q}^H\mathbf{y}$	$1201 \times (10 + 4)$	538.0
QR: prev. results	$1201 \times (10 + 4)$	538.0
Turbo: next input	$3 \times 6114 + 12$	128.5

antennas, 64-QAM, and list length 16. For this reason, alternative MIMO-OFDM configurations with lower data rate, 68 Mbps, have been used. Shorter code blocks are assumed for turbo coding in Tables 2 and 3. With shorter code blocks, the delay of the FFT can be limited to processing one OFDM symbol per each antenna.

In the configuration in Table 2, hardware implementations presented in [9] are used for the matrix decomposition and symbol detection. For the FFT and turbo decoding the TI's C6416 DSP has been applied as it can compute the FFT with an efficient software library routine and it includes a turbo coprocessor which runs with halved clock frequency. Since the core DSP and turbo coprocessor are mapped to the same device, the number of required processors is determined by the more dominating task, that is, turbo decoding. The idling of the DSP core while turbo decoding is taken into account when the utilization in Table 2 is calculated, and therefore, the utilization is low in Table 2 but still several processors are required. The hardware implementations for QR and symbol detection in Table 2 are targeted for MIMO-OFDM systems [9]. However, the sphere detector applies a different algorithm than the K-best LSD which is used in TTA processor implementations.

In Table 3 a 1024-point FFT is applied. The applied turbo decoder processor supports also Viterbi decoding. The list length of the K-best LSD is 10 symbols. In principle, a complex-valued K-best LSD with 64-QAM, 2 antennas, and $K = 16$ must process $64 + 16 \times 64 = 1088$ nodes and with 16-QAM, 4 antennas, and $K = 10$ it must process $16+10\times16+10\times16+10\times16 = 496$ nodes during the symbol detection. Thus, the processing requirements of different symbol detectors can be characterized by the number of visited nodes during a tree traversal of the algorithm. The applied QR decomposition hardware accelerator is presented in [8] as a part of MIMO-OFDM transceiver for WLANs. The decomposition takes 65 clock cycles for 4×4 matrix.

In Table 3, the workload of 4G baseband processing with 100 Mbps is presented in terms of required execution cycles on an SODA architecture [7]. For each task a realistic clock frequency is assumed and the tasks are divided to separate processors. Furthermore, it is assumed that the LDPC error correction decoding task can be parallelized to several processors. The Table 3 shows that the LDPC task dominates clearly the workload.

In conclusion, the results in Tables 2–4 show that in addition to the data rate, the computational requirements depend heavily on the applied algorithms and on the parameters of the algorithms. Furthermore, efficiency in terms of high utilization requires that the tasks can be mapped among the processors or hardware units in a flexible way.

5.7. Memory Requirements. The area estimates in Table 1 exclude the memories and memory requirements are reported separately in the Table 5. In other words, the area in terms of logic GEs expresses the complexity of the

actual computations of baseband processing. The separation eases future comparisons, since the memory requirements depend heavily on the targeted data vector lengths and technology. For example, long code blocks are preferred in turbo decoding, as they enhance the error correction performance. A second reason for separating the memories is that the IPC requires also memories, and therefore, the total area with all the memories of the whole baseband processing chain would depend on the implementation method of the IPC.

The data memory requirements in Table 5 show that due to the small matrix size, the QR decomposition requires a very small memory. The LSD processor has no memory requirements at all, as the data is stored in registers. On the other hand, the turbo decoder and the FFT processors require large memories as they have to process long data vectors. The memory of the FFT is divided into two banks and a memory interface hides the banking structure from the programmer, that is, the memory system imitates dual-port memory.

5.8. Interprocessor Communication Requirements. As the analyzed processors lack extra facilities for IPC, only requirements but not costs can be stated. There exists many methods for SoCs but they are beyond the scope of this paper, the complexity of computing the main baseband functions. Therefore, the effects of using some particular method or SoC platform are not considered. In Table 6, the IPC requirements are tabulated for an assumed system using shared memory banks between the processors.

The FFT processor uses an in-place algorithm, that is, the result overwrites the input vector and processing does not require additional memory. However, passing the data to and from the FFT processors requires buffer memories. In practice, there must be an extra input buffer which is written while the data in the main memory is processed in-place. In a similar way, there must be an extra output buffer, from which the previous result can be read at the same time. The first two buffers in Table 6 are dedicated for such an IPC. The roles of the three memory banks, that is, the input buffer, the output buffer, and the processing memory, can be interchanged on every two completed OFDM symbols.

The QR processor generates the triangular 4×4 matrix, $\mathbf{R}$, with 10 nonzero elements and 4-element vector for each subcarrier. The results are written to one buffer. The other identical buffer holds the previous results which are passed to the LSD processors at the same time. Since there are several QR and LSD processors, the buffer must be divided into several parallel accessible banks. Again, the roles of the buffers can be interchanged on OFDM symbol boundaries.

The turbo decoder processors require an additional input buffer which is filled with the soft bits while the decoders are processing. There is no need for and additional output buffer, since the decoder overwrites the previous output only on the last half iteration. The buffer size of the turbo decoder input in Table 6 allows code rate $R = 1/3$ with the maximum block size. The input word length of the applied turbo decoder TTA processor is 7 bits [29], but all the other applied TTA processors use 16 bits for the real or imaginary parts.

In general, the complexity of IPC buffers depend on the sizes of memory banks, their throughput or clock frequency, and the number of memory banks as each bank requires interfacing logic. In addition, the IPC increases also the computational load which is not included in Tables 1–3. Therefore, if a fully functional SoCs were designed, full utilization should not be targeted when solely the core computations are analyzed. Instead, with lower utilization, computing capacity would be reserved also for the IPC. Also, the total delay in Tables 1–3 exclude the effect of IPC. As it is assumed that one buffer is written while the other is read in a pipelined fashion, it can be assumed that the IPC has a constant delay.

Since the workloads of the processors depend only on the applied block lengths, static scheduling could be applied, which would ease synchronization of the tasks. Even if the number of processors is very high, in principle, similar IPC requirements would be met also with smaller number of processors if they applied higher parallelism internally or if they applied higher clock frequency. The first option would require parallel IPC links and the second option would require smaller number of IPC links but higher throughput for each link.

6. Conclusions

The main baseband functions of a 3G LTE conforming MIMO-OFDM receiver were considered in this paper, and ASP implementations were assumed for each function. The main emphasis was on the complexity of the actual computations, that is, the data path, of the functions implemented with the ASPs. The complexity was derived by estimating the required number of respective processors and the clock frequency to meet real-time requirements. The area and power estimates of the functions processed with the ASPs showed the demands of the baseband processing with the current technology. It was shown that especially the LSD dominates the computational load. However, due to the fine granularity and convenient parallelization of the LSD, it can be distributed among several processors and high utilization can be achieved. Also other processors or hardware accelerators of the addressed functions were analyzed to further illustrate the computational demands and costs. The IPC requirements were estimated by a block by block processing model with processors connected via shared memory banks.

Acknowledgment

This work has been supported by the Finnish Funding Agency for Technology and Innovation under research funding decision 40163/07.

References

[1] R. Bachl, P. Gunreben, S. Das, and S. Tatesh, "The long term evolution towards a new 3GPP* air interface standard," *Bell Labs Technical Journal*, vol. 11, no. 4, pp. 25–51, 2007.

[2] R. W. Chang and R. A. Gibby, "A theoretical study of performance of an orthogonal multiplexing data transmission scheme," *IEEE Transactions on Communication Technology*, vol. 6, no. 4, pp. 529–540, 1968.

[3] G. J. Foschini and M. J. Gans, "On limits of wireless communications in a fading environment when using multiple antennas," *Wireless Personal Communications*, vol. 6, no. 3, pp. 311–335, 1998.

[4] C. Berrou, A. Glavieux, and P Thitimajshima, "Near Shannon limit error-correcting coding and encoding: turbo-codes. 1," in *Proceedings of IEEE International Conference on Communications (ICC '93)*, vol. 2, pp. 1064–1070, Geneva, Switzerland, May 1993.

[5] A. J. Paulraj, D. A. Gore, R. U. Nabar, and H. Bölcskei, "An overview of MIMO communications—a key to gigabit wireless," *Proceedings of the IEEE*, vol. 92, no. 2, pp. 198–218, 2004.

[6] K. Higuchi, H. Kawai, N. Maeda, H. Taoka, and M. Sawahashi, "Experiments on real-time 1-Gb/s packet transmission using MLD-based signal detection in MIMO-OFDM broadband radio access," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 6, pp. 1141–1153, 2006.

[7] M. Woh, S. Seo, H. Lee, et al., "The next generation challenge for software defined radio," in *Proceedings of the 7th International Workshop on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS '07)*, vol. 4599 of *Lecture Notes in Computer Science*, pp. 343–354, Springer, Samos, Greece, July 2007.

[8] D. Perels, S. Haene, P. Luethi, et al., "ASIC implementation of a MIMO-OFDM transceiver for 192 Mbps WLANs," in *Proceedings of the 31st European Solid-State Circuits Conference (ESSCIRC '05)*, pp. 215–218, Grenoble, France, September 2005.

[9] B. Cerato, G. Masera, and E. Viterbo, "Enabling VLSI processing blocks for MIMO-OFDM communications," *VLSI Design*, vol. 2008, Article ID 351962, 10 pages, 2008.

[10] "TMS320C64x Technical Overview," Texas Instruments, SPRU395B, January 2001.

[11] "TMS320C64x DSP Library Programmer's reference," Texas Instruments, SPRU565B, October 2003.

[12] Y.-T. Lin, P.-Y. Tsai, and T.-D. Chiueh, "Low-power variable-length fast Fourier transform processor," *IEE Proceedings: Computers and Digital Techniques*, vol. 152, no. 4, pp. 499–506, 2005.

[13] S. Y. Lim and A. Crosland, "Implementing FFT in a FPGA coprocessor," in *Proceedings of the International Embedded Solution Event*, pp. 230–233, Santa Clara, Calif, USA, September 2004.

[14] T. Pitkänen, R. Mäkinen, J. Heikkinen, T. Partanen, and J. Takala, "Low-power, high-performance TTA processor for 1024-point fast fourier transform," in *Proceedings of the 6th International Workshop on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS '06)*, vol. 4017 of *Lecture Notes in Computer Science*, pp. 227–236, Springer, Samos, Greece, July 2006.

[15] S. Y. Kung, *VLSI Array Processors*, Prentice-Hall, Upper Saddle River, NJ, USA, 1987.

[16] A. Maltsev, V. Pestretsov, R. Maslennikov, and A. Khoryaev, "Triangular systolic array with reduced latency for QR-decomposition of complex matrices," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '06)*, pp. 385–388, Kos, Greece, May 2006.

[17] C. K. Singh, S. H. Prasad, and P. T. Balsara, "VLSI architecture for matrix inversion using modified gram-schmidt based QR decomposition," in *Proceedings of the 20th International Conference on VLSI Design jointly with the 6th International Conference on Embedded Systems (VLSID '07)*, pp. 836–841, Bangalore, India, January 2007.

[18] Altera Corporation, "Implementation of CORDIC-based QRD-RLS algorithm on Altera Stratix FPGA with embedded Nios soft processor technology," White Paper WP-STXQRD-01, Altera Corporation, San Jose, Calif, USA, March 2004.

[19] F. Edman and V. Öwall, "A scalable pipelined complex valued matrix inversion architecture," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '05)*, vol. 5, pp. 4489–4492, Kobe, Japan, May 2005.

[20] P. Salmela, A. Burian, H. Sorokin, and J. Takala, "Complex-valued QR decomposition implementation for MIMO receivers," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '08)*, pp. 1433–1436, Las Vegas, Nev, USA, March-April 2008.

[21] M. Myllylä, J.-M. Hintikka, J. R. Cavallaro, M. Juntti, M. Limingoja, and A. Byman, "Complexity analysis of MMSE detector architectures for MIMO OFDM systems," in *Proceedings of the 39th Asilomar Conference on Signals, Systems and Computers*, pp. 75–81, Pacific Grove, Calif, USA, October-November 2005.

[22] Z. Guo and P. Nilsson, "Algorithm and implementation of the K-best sphere decoding for MIMO detection," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 3, pp. 491–503, 2006.

[23] M. Wenk, M. Zellweger, A. Burg, N. Felber, and W. Fichtner, "K-best MIMO detection VLSI architectures achieving up to 424 Mbps," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '06)*, pp. 1151–1154, Kos, Greece, May 2006.

[24] K.-W. Wong, C.-Y. Tsui, R. S.-K. Cheng, and W.-H. Mow, "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '02)*, vol. 3, pp. 273–276, Phoenix, Ariz, USA, May 2002.

[25] J. Antikainen, P. Salmela, O. Silvén, M. Juntti, J. Takala, and M. Myllyä, "Fine-grained application-specific instruction set processor design for the K-best list sphere detector algorithm," in *Proceedings of the International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS '08)*, pp. 108–115, Samos, Greece, July 2008.

[26] S. Agarwala, T. Anderson, A. Hill, et al., "A 600-MHz VLIW DSP," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 11, pp. 1532–1544, 2002.

[27] Xilinx, "3GPP Turbo Decoder v3.1," DS318, May 2007.

[28] T. Vogt and N. Wehn, "A reconfigurable application specific instruction set processor for viterbi and log-MAP decoding," in *Proceedings of IEEE Workshop on Signal Processing Systems Design and Implementation (SIPS '06)*, pp. 142–147, Banff, Canada, October 2006.

[29] P. Salmela, H. Sorokin, and J. Takala, "A programmable max-log-MAP turbo decoder implementation," *VLSI Design*, vol. 2008, Article ID 319095, 17 pages, 2008.

[30] H. Corporaal, "Design of transport triggered architectures," in *Proceedings of the 4th IEEE Great Lakes Symposium on VLSI (GLSV '94)*, pp. 130–135, Notre Dame, Ind, USA, March 1994.

[31] I. Karkowski and H. Corporaal, "A framework for design of heterogeneous multi-processor embedded systems," Tech. Rep. 1-68340-44/1997/12, Delft University of Technology, Delft, The Netherlands, 1997.

[32] J. Guo, K. Dai, and Z. Wang, "A heterogeneous multi-core processor architecture for high performance computing," in

Proceedings of the 11th Asia-Pacific Conference on Advances in Computer Systems Architecture (ACSAC '06), vol. 4186 of *Lecture Notes in Computer Science*, pp. 359–365, Springer, Shanghai, China, September 2006.

[33] T. Ahonen and J. Nurmi, "Integration of a NOC-based multimedia processing platform," in *Proceedings of the International Conference on Field Programmable Logic and Applications (FPL '05)*, pp. 606–611, Tampere, Finland, August 2005.

[34] G. Tempesti, P.-A. Mudry, and R. Hoffmann, "A move processor for bio-inspired systems," in *Proceedings of NASA/DoD Conference on Evolvable Hardware (EH '05)*, pp. 262–271, Washington, DC, USA, June-July 2005.

[35] J. Rossier, Y. Thoma, P.-A. Mudry, and G. Tempesti, "MOVE processors that self-replicate and differentiate," in *Proceedings of the 2nd International Workshop on Biologically Inspired Approaches to Advanced Information Technology (BioADIT '06)*, vol. 3853 of *Lecture Notes in Computer Science*, pp. 160–175, Springer, Osaka, Japan, January 2006.

[36] M. Myllylä, P. Silvola, M. Juntti, and J. R. Cavallaro, "Comparison of two novel list sphere detector algorithms for MIMO-OFDM systems," in *Proceedings of the 17th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC '06)*, pp. 1–5, Helsinki, Finland, September 2006.

[37] 3GPP, "Multiplexing and channel coding (release 8)," Technical Specification TS.36.212 v1.0.0, Group Radio Access Network, 3rd Generation Partnership Project, Cedex, France, 2007.

[38] 3GPP, "Multiplexing and channel coding (FDD) (release 5)," Technical Specification TS 25.212 v5.3.0, Group Radio Access Network, 3rd Generation Partnership Project, Cedex, France, 2002.

[39] P. Jääskeläinen, V. Guzma, A. Cilio, T. Pitkänen, and J. Takala, "Codesign toolset for application-specific instruction-set processors," in *Multimedia on Mobile Devices*, vol. 6507 of *Proceedings of SPIE*, pp. 1–11, San Jose, Calif, USA, January 2007.

[40] G. H. Golub, *Matrix Computations*, John Hopkins University Press, Baltimore, Md, USA, 1989.