Fixed- and Floating-Point Processor Comparison for MIMO-OFDM Detector

Janne Janhunen, *Student Member, IEEE*, Teemu Pitkänen, Olli Silvén, *Member, IEEE*, and Markku Juntti, *Senior Member, IEEE*

***Abstract*—The evolution toward software-defined radio (SDR) technologies, in particular, cognitive radios, is leading toward the need to support multiple radio solutions with the same baseband processing resources. This implies not only a huge design effort, but also a shift from hardware to software design flavored tool chains. In this paper, a hardware complexity and energy dissipation are analyzed by implementing three programmable processor architectures that support 32- and 12-bit floating-point and 16-bit fixed-point arithmetics. The processors are based on the transport triggered architecture (TTA) that has a very low programmability overhead. We programmed a recently introduced selective spanning with fast enumeration (SSFE) soft-output detector for these processors. The processors are capable to achieve data rates required in multiple-input multiple-output orthogonal frequency-division multiplexing (MIMO-OFDM) 3G LTE system with a small energy dissipation. The analysis shows that at the same goodput rate a floating-point implementation can achieve a lower gate count and a better power efficiency than a fixed-point design. Combined with tool chain benefits, the floating-point arithmetic is becoming attractive for future SDR solutions.**

***Index Terms*—Fixed-point arithmetic, floating-point arithmetic, multiple-input multiple-output (MIMO), software-defined radio (SDR), transport triggered architecture (TTA).**

I. Introduction

WIRELESS communication systems have experienced tremendous development during the last two decades. The fourth-generation wireless communication systems and networks will encounter several implementation challenges due to ever increasing capacity and flexibility requirements. The latter ones are the driver for software-defined radio (SDR) technologies that in turn are expected to enable the creation of cognitive radios. As a result, any radio solution could be invoked on demand on any platform. Designing such solutions is a huge effort that should result in a straightforwardly reusable code legacy for the future.

So far, the demands for high gate and energy efficiencies in baseband processing designs have been interpreted as requirements to employ a fixed-point arithmetic. In part, this expectation comes from the wide single-instruction multiple-data (SIMD) or vector type digital signal processors (DSP) intended for radio implementations. On those architectures, the fixed-point arithmetic is efficient, but achieving efficiency requires utilizing inline assembly or intrinsic functions that can be very difficult and laborious to port to another execution platform. On the other hand, the floating-point arithmetic has generally been considered power hungrier and much less gate efficient than fixed-point designs.

In this paper, we investigate the gate count and energy efficiency rankings of fixed- and floating-point arithmetic implementations using a multiple-input multiple-output (MIMO) detector as a target case. In addition, we propose a programmable processor capable to achieve the long term evolution (LTE) detection rate requirements. We show that the floating-point arithmetic has great potential also for low-power devices. To make the results genuinely comparable, the same programmable transport triggered processor architecture (TTA) [1] has been employed and fixed goodput rates have been targeted. The goodput is a measure which combines both the hardware limitations and the detection reliability, i.e., the minimum of the transmission throughput and hardware detection rate of information bits [2]. Based on our experience and literature reviews [3], [4] for instance turbo decoder and the channel estimation blocks can all be implemented using less than 16-bit fixed-point word. Thus, we are confident that the floating-point arithmetic can be applied efficiently in the rest of the modem. Further studies are currently ongoing to confirm this assumption.

We will implement three processors with identical resources applying 32- and 12-bit floating-point and 16-bit fixed-point arithmetics. The processors are synthesized with a 130-nm low-power CMOS technology. Word length study is done with a 3G LTE compliant downlink simulator with realistic system parameters. A 4×4 antenna system and 64 quadrature amplitude modulation (QAM) is assumed to be the most advanced case to be implemented using programmable processors in the near future. We will program each of the processors to execute a selective spanning with fast enumeration (SSFE) [5] detector for a 2×2 antenna system with 16-QAM and 4×4 antenna system with 64-QAM to estimate the performance of the processors.

Manuscript received January 24, 2011; revised May 06, 2011 and August 15, 2011; accepted August 15, 2011. Date of publication August 22, 2011; date of current version November 18, 2011. This work was supported in part by the Finnish Funding Agency for Technology and Innovation (TEKES), Elektrobit, Nokia, Nokia Siemens Networks, Xilinx, Renesas Mobile Europe, Infotech Oulu Graduate School, Academy of Finland, the Nokia Foundation, and Tekniikan edistämisäätiö (TES). The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Andrew Singer.

J. Janhunen and M. Juntti are with the Centre for Wireless Communications, University of Oulu, Oulu FI-90014, Finland (e-mail: janne.janhunen@ee.oulu.fi; markku.juntti@ee.oulu.fi).

T. Pitkänen is with the Department of Computer Systems, Tampere University of Technology, Tampere FI-33101, Finland (e-mail: teemu.pitkanen@tut.fi).

O. Silvén is with the Computer Science and Engineering Laboratory, University of Oulu, Oulu FIN-90014, Finland (e-mail: olli.silven@ee.oulu.fi).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JSTSP.2011.2165830

TABLE I
DYNAMIC RANGES AND RESOLUTIONS FOR SIGNED NUMBER FORMATS

	Dynamic range	Resolution
Fixed-point		
12-bit Q11	(-1,1)	4.8828×10^{-4}
16-bit Q15	(-1,1)	3.0518×10^{-5}
32-bit Q31	(-1,1)	4.6566×10^{-10}
16-bit Q5.10	[-31, 31]	9.7656×10^{-4}
Floating-point (normalized)		
12-bit (1,4,7)	[-255, 255]	1.5887×10^{-3}
16-bit (1,5,10)	[-65504, 65504]	6.1035×10^{-5}
32-bit (1,8,23)	$[-3.4028 \times 10^{38}, 3.4028 \times 10^{38}]$	1.1755×10^{-38}

Word lengths analyzed for 4×4 antenna system are utilized in the study.

The rest of the paper is organized as follows. Section II gives a brief motivation to the fixed- and floating-point arithmetic study. Section III presents the MIMO detection scheme, the assumed wireless communication system model. In Section IV, we summarize the computer simulation parameters and present the results for word length study. Section V describes the implementation on TTA and summarizes the results. Finally, in Section VI we conclude the contributions of the study.

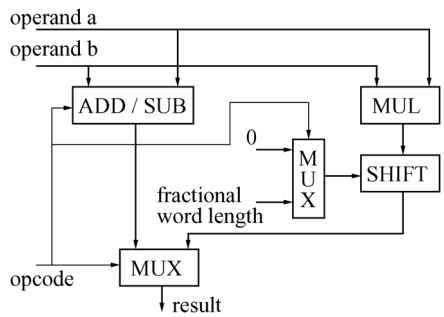


Fig. 1. Fixed-point ALU.

II. FIXED-POINT VERSUS FLOATING-POINT ARITHMETIC

Traditionally, the fixed-point arithmetic has been favored in wireless communication systems due to power consumption, silicon area and price per unit reasons. However, the systems are becoming more complex and new applications require a fast time-to-market cycle. A significant driving force for a floating-point arithmetic has been the rapid development of the mobile graphics processing units (GPU). A single-precision floating-point arithmetic in general leads to a high silicon complexity and energy consumption but for mobile GPU implementations there is an option available to reduce complexity by using the IEEE 754-2008 half precision floating-point arithmetic. The half precision floating-point arithmetic provides a feasible energy-precision tradeoff for many applications such as graphics processing presented in [6].

The fixed-point arithmetic is suitable for applications in which the dynamic range can be easily scaled to interval $(-1, 1)$, which implies a fractional mode [7]. Since the fixed-point presentations have no exponent they can represent numbers with more significant bits than the floating-point format with the same word length, i.e., they achieve better resolution. On the other hand, many current applications require a wider numerical dynamic range. For such applications the fixed-point arithmetic requires more bits than the floating-point arithmetic which may favor a floating-point design also in terms of power consumption and silicon area. The large dynamic range of the floating-point format keeps the silicon area close to constant when the numerical dynamic range of data is increased.

Table I summarizes the dynamic range of representable values and resolutions for fixed-point (Q notation) arithmetic and floating-point arithmetic number formats. Numbers in left column (in parenthesis) denotes sign, exponent and mantissa bits in the floating-point number format. Q5.10 fixed-point and nonstandard 12-bit floating-point number formats are used in this study. Examples in the table show that the floating-point arithmetic achieves a wider dynamic range compared to the fixed-point arithmetic at the expense of precision.

We illustrate floating- and fixed-point arithmetic logic units (ALU) with addition, subtraction and multiplier operations in Figs. 1 and 2. There are three inputs for ALU, two for operands and one for opcode which defines the operation to be executed. The fixed-point ALU has support for a fractional mode, i.e., the multiplication result is shifted into a right scale inside the unit. The floating-point unit has extra logic including multiple shifters and normalization steps, which explains a higher power consumption and silicon area per function unit [8]. Due to extra logic, a floating-point ALU in general requires more pipeline stages than its fixed-point counterpart to shorten critical path, which affects to a maximum achievable clock frequency.

The floating-point FU complexity can be optimized by using a low-cost rounding technique. The IEEE 754-2008 standard [9] defines five different rounding techniques in order to provide superior numerical accuracy and stability. We compare two of them in Table II. Truncation provides the lowest energy dissipation and hardware complexity, whereas rounding toward nearest value is more accurate but also more complex one to implement in hardware. The table also shows how the number of extra guard bits effects on the hardware complexity. The standard defines also denormal numbers and exception handling for example in case of an invalid operand or overflows but those are not considered in this study. A low-leakage 90-nm CMOS technology is utilized in the synthesis. The synthesized function units supports 12-bit floating-point arithmetic with 4-bit exponent and 7-bit mantissa and are operating on 143-MHz clock frequency.

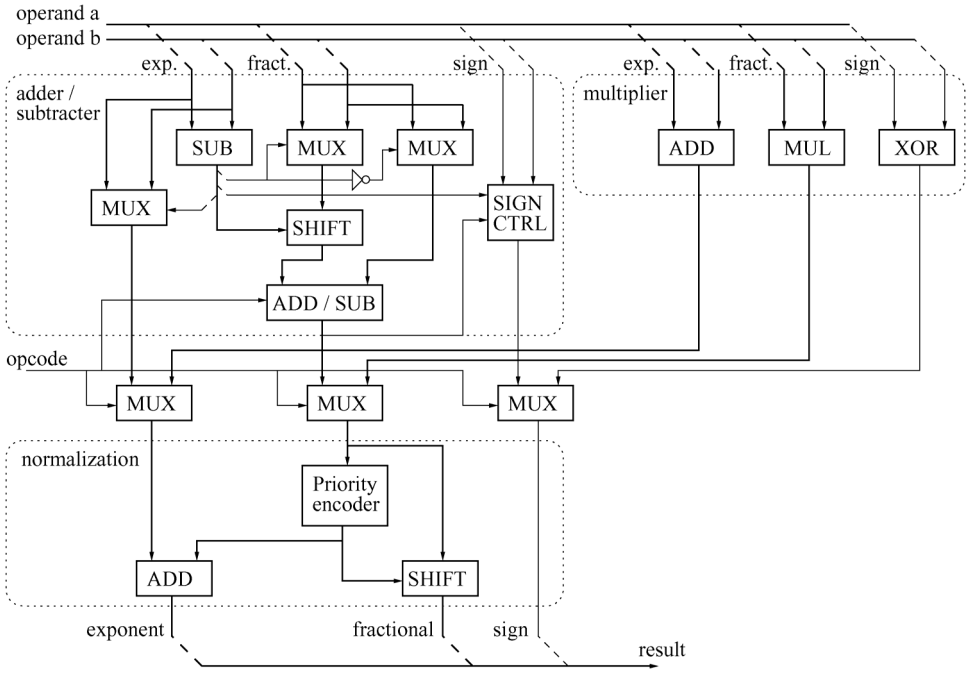


Fig. 2. Floating-point ALU.

TABLE II
EFFECT OF ROUNDING TECHNIQUE ON HARDWARE COMPLEXITY IN GATE EQUIVALENTS

	multiplier (GE)	adder (GE)
no rounding	1030	1400
truncation (1-bit guard)	1050	1420
round-to-nearest (1-bit guard)	1140	1850
round-to-nearest (3-bit guard)	1190	2000

In general, the results show that the rounding complexity is more significant in the adder than in the multiplier. The truncation increases the FU logic with 20 gate equivalents (GEs), whereas the round toward nearest technique increases the FU complexity with 430 GEs. Applying three guard bits increases the complexity approximately 600 GEs. A higher word length increases also the cost of rounding but the relative cost between different techniques stays approximately the same. The truncation does not cause much overhead to the normalization logic nor lengthen significantly the critical, whereas the round toward nearest technique significantly increases the logic complexity. Fortunately, the truncation technique can be usually applied in the implementations. Approximately, the third of the critical path in adder FU is caused by the normalization logic.

The disadvantages of the fixed-point arithmetic shows up in the programming phase. The programmer has to use in-line assembly and intrinsics to achieve an efficient fixed-point compilation but in addition the results have to be scaled constantly in order to avoid overflows or underflows. In [10], TMS320C67x floating-point processor is found out to be 1.1 to 4.3 times superior over TMS320C62x fixed-point processor in terms of clock cycles for the same algorithms. The reason is in additional cast and shift operations, which also increase the energy dissipation in the fixed-point implementations.

III. SYSTEM MODEL

We assume a MIMO transmission architecture combined with an orthogonal frequency division multiplexing (OFDM) with M transmit and N receive antennas where $M \leq N$. Our LTE [11] compliant system model is illustrated in Fig. 3. The transmitter utilizes a layered space-time architecture with a vertical encoding in a 2×2 antenna system and a horizontal encoding in a 4×4 antenna system. In the horizontal encoding, the signals are separately coded into multiple codewords before spatial multiplexing. Multiple codewords can be exploited in per-stream link adaptation and successive interference cancellation (SIC) receivers [12]. The encoded data bits are interleaved and modulated to symbols with a quadrature amplitude modulation. Bit-interleaved coded modulation (BICM) is applied.

The data are multiplexed to multiple antennas. At the receiver, antennas receive multipath signals distorted by the noisy channel. The cyclic prefix of an OFDM symbol is assumed to be long enough to eliminate intersymbol interference, i.e., $\mathrm{cp} > (T_{\mathrm{m}}/T_{\mathrm{s}})$, where T_{m} is the maximum delay spread in channel and T_{s} denotes the symbol time. The received signal vector on sth subcarrier can be presented as

$$\mathbf{y}_s = \mathbf{H}_s \mathbf{x}_s + \mathbf{n}_s, \quad s = 1, 2 \ldots, S. \tag{1}$$

Here S is the number of subcarriers, $\mathbf{y}_s \in \mathbb{C}^N$, $\mathbf{x}_s \in \mathcal{A}$ denotes the transmitted symbol vector, $\mathcal{A} \in \mathbb{C}$ is the symbol alphabet, and $\mathbf{n}_s \in \mathbb{C}^N$ is a noise vector containing complex Gaussian fading coefficients. The symbol $\mathbf{H}_s \in \mathbb{C}^{N \times M}$ denotes the channel matrix. The entries of $\mathbf{x}_s$ are chosen independently of each other from a QAM constellation.

A soft detection is applied at the receiver. A block diagram of the soft-output detection structure is presented in Fig. 4. The channel matrix $\mathbf{H}$ is preprocessed as $\mathbf{H} = \mathbf{QR}$ using a QR-decomposition. The preprocessing block outputs an orthogonal

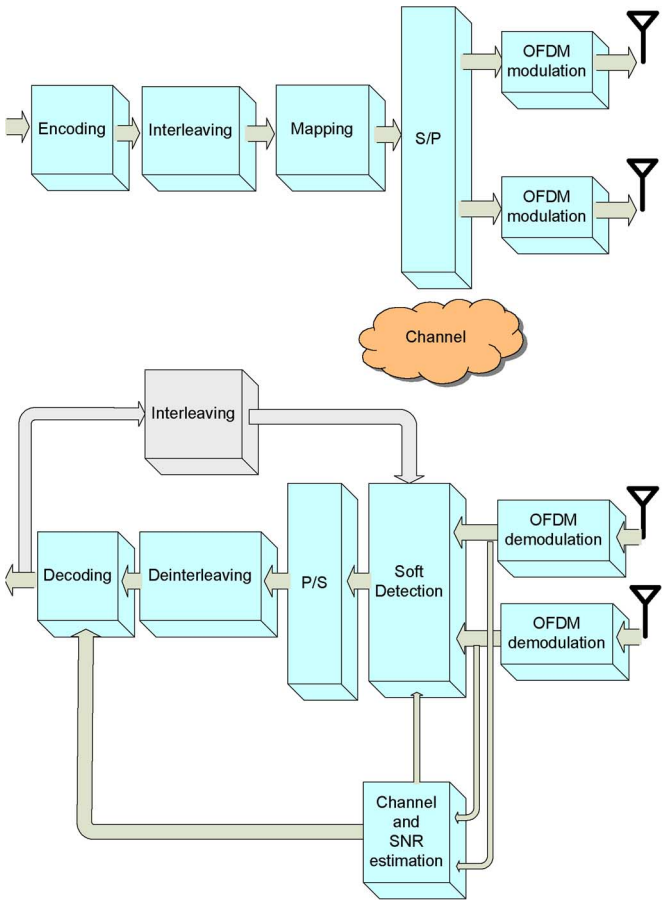


Fig. 3. Block diagram of the MIMO-OFDM transmission.

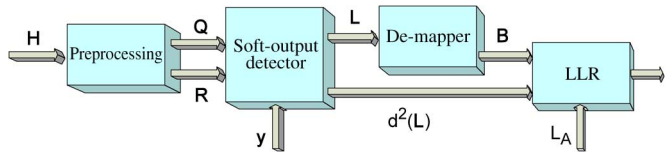


Fig. 4. Soft-output detection structure.

TABLE III
PARAMETERS FOR SIMULATION

Simulation parameters	
Number of subcarriers	512 (300 active)
Channel code	turbo code
Code rate	$\frac{1}{3}$, $\frac{1}{2}$
Symbol duration	71.39 μs
Symbol time T_s	66,7 μs
Cyclic prefix (CP) duration	4.69 μs
User velocity	15 kmph, 120 kmph
Bandwidth	5 MHz
Channel model parameters	
Channel model	3GPP-VA ITU,
Number of paths	6
Path delays	[0 310 710 1090 1730 2510] ns
Path power	[0 -1 -9 -10 -15 -20] dB
BS azimuth spread	5°
MS azimuth spread	35°

matrix $\mathbf{Q}$ and an upper triangular matrix $\mathbf{R}$. The Euclidean distance between the received signal vector $\mathbf{y}$ and the possible transmitted symbols are calculated in the soft-output detector block. The de-mapper block maps the symbol candidate list $\mathbf{L}$ into a binary format. The log-likelihood ratios are calculated in the LLR block using the list of binary format candidate symbols $\mathbf{B}$ and corresponding list of Euclidean distances $d^2(\mathbf{L})$. Let b be in Galois field with the elements $\{-1, +1\}$. The conditioned log-likelihood ratio of binary candidate b, $L(b_n|\mathbf{y})$ over bits of received symbol vector is determined as

$$L(b_n|\mathbf{y}) = \ln \frac{\mathrm{p}(b_n = +1|\mathbf{y})}{\mathrm{p}(b_n = -1|\mathbf{y})} = \ln\left(\mathrm{p}(\mathbf{y}|b_n = 1)\right) - \ln\left(\mathrm{p}(\mathbf{y}|b_n = -1)\right). \quad (2)$$

In order to reduce computational complexity caused by the logarithm and exponential functions, an approximation of LLRs can be calculated by using the Jacobian logarithm and a look-up table [13]. In the iterative receiver structure, *a priori* information L_A from the forward error correction (FEC) decoder can be used to update log-likelihood ratio (LLR) values to improve detection reliability [14]. The soft information feedback from the FEC decoder to the detector is referred to a global iteration. In this study, we concentrate on the soft detection part that usually dominates the computational complexity at the receiver.

The ML detector minimizes the Euclidean distance between the received signal $\mathbf{y}$ and the lattice points $\mathbf{Hx}$, i.e.,

$$\hat{\mathbf{x}} = \arg \min_{\tilde{\mathbf{x}} \in \mathcal{A}^M} \|\mathbf{y} - \mathbf{H}\tilde{\mathbf{x}}\|^2 \quad (3)$$

where $\|\cdot\|^2$ denotes the Euclidean norm of a vector. The exhaustive search can be used to solve the ML detection problem. However, it becomes computationally impractical as the number of transmit antennas increases or a high-order modulation is used. The sphere detection algorithm approximates the ML solution (3) by limiting the search to the constellation points that lie inside an M-dimensional hyper-sphere [15]. The number of visited nodes in the search tree can be reduced by limiting the search to inside a sphere with radius d using the sphere constraint $d^2 \geq \|\mathbf{y} - \mathbf{Hx}\|^2$. In spite of the fact that the SSFE algorithm is not a sphere detector like well studied K-best LSD algorithm [16], [17], it shares the same QR decomposition (QRD)-based preprocessing and similar tree search with sphere detectors. Detailed description of the implemented detector algorithm is provided in [5].

IV. Error Rate Performance

Fixed- and floating-point computer simulations with several word lengths for SSFE algorithm were carried out in a MATLAB environment. The used simulation parameters, summarized in Table III, are 3G LTE compliant [11]. We utilize an MMSE extension to QRD to improve the detector performance. The simulator takes into account the effect of log-likelihood ratio (LLR) clipping [18] with threshold $L_{\max} = 8$. By limiting the dynamic range of LLRs, the size of the final candidate list can be reduced.

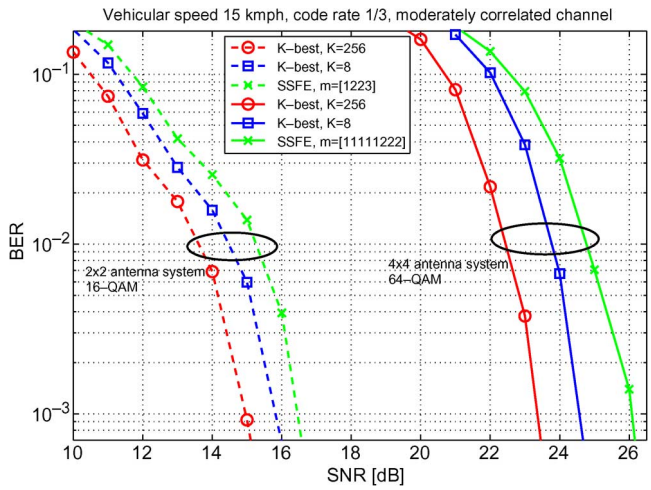


Fig. 5. Detector performance comparison.

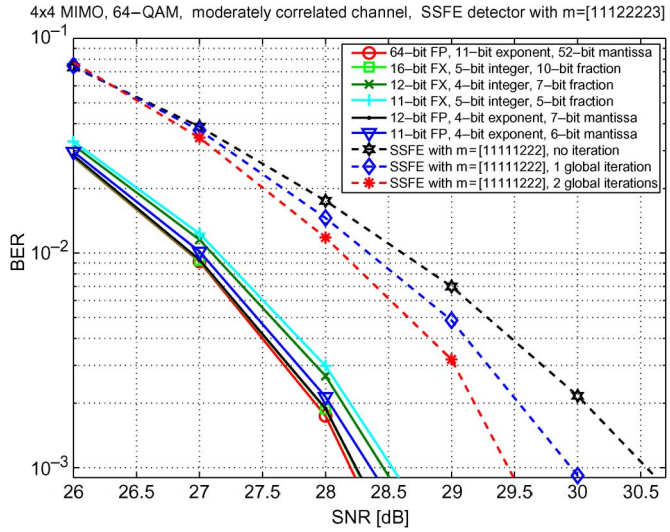


Fig. 6. Bit error rate comparison for fixed- and floating-point arithmetic in 4 × 4 64-QAM system.

We simulated the detector in a correlated channel which is compliant to the 3GPP vehicular A (3GPP-VA) parameters defined by International Telecommunication Union (ITU). Fig. 5 provides a performance comparison between detectors. The K-best algorithm with list size 256 equals in 2 × 2 antenna system a full search and in 4 × 4 antenna system the performance is close to full search performance. A vehicular speed is assumed to be 15 kmph and the code rate is set up to 1/3. In 2 × 2 antenna system the SSFE algorithm loses approximately 1.2 dB in performance compared to a full search. In 4 × 4 system, the difference is less than 3 dB, which is still a rather moderate performance loss. The 4 × 4 system setup is aimed to apply during good channel realizations. As a comparison, we present curves also for the K-best detector with $K = 8$. The detector provides reliable detection but is unfortunately complex to implement with an SDR platform.

We determined sufficient word lengths for implementations by computer simulations. The floating-point word is divided in sign, exponent and mantissa parts, whereas the fixed-point word is defined by the sign, integer and fraction parts. In general, the required word length depends mostly on the channel conditions, the number of antennas, used modulation and detector algorithm. Notations FX and FP in the figures and tables are used to denote fixed-point and floating-point arithmetics.

In Fig. 6, curves with different word lengths and variations of integer (exponent) and fraction (mantissa) bits are compared to the IEEE double-precision floating-point curve in 4 × 4 antenna system with 64-QAM. A vehicular speed of 120 kmph and a half code rate are assumed. The 16-bit fixed-point and 12-bit floating-point arithmetics provide the same goodput, which is very close to the IEEE double-precision floating-point performance. 12-bit and 11-bit fixed-point arithmetics loose approximately 0.25 dB (BER 10^{-3}) and the 11-bit floating-point arithmetic little less. Using less bits in the integer (exponent) or fraction (mantissa) parts, causes a detection breakdown.

We used a level update vector $\mathbf{m} = [1, 1, 1, 2, 2, 2, 2, 3]$ in word length simulations for 4 × 4 antenna system due to fact that larger list size requires more dynamic range and resolution than implementation with shorter list. The reason for this is the cumulative sum of PED values. However, the list size of 48 is not feasible in practical implementation, and thus, we decided to use a level update vector $\mathbf{m} = [1, 1, 1, 1, 1, 2, 2, 2]$. In LTE, an adaptive transmission is part of the standard. Thus, more antennas, high modulation order and code rate can be utilized during a good channel realization. Fig. 6 illustrates that a feasible SSFE implementation in terms of computational complexity requires a good channel. Thus, we make an assumption that system uses four antennas and 64-QAM only during a good channel realization. The detection reliability can be improved 0.5–1 dB by adding global iterations. However, the cost of global iterations to the system goodput is not considered in this study.

For completeness we show the results for 2 × 2 antenna system with 16-QAM in Fig. 7. The difference between fixed- and floating-point word lengths is only one bit. The floating-point mantissa bits can be reduced by three bits, which reduces the required word length to 9 bits. On the other hand, the fixed-point word length can be reduced by six bits compared to 4 × 4 antenna system. In 2 × 2 16-QAM system, a level update vector $\mathbf{m} = [1, 2, 2, 3]$ is used both in word length study and implementation. The level update vector keeps the detection complexity reasonably low, but still provides a good BER also in a correlating channel. That is why we assume that during worse channel conditions, the system adapts to work with 2 × 2 antennas and 16-QAM.

The simulation results were confirmed by the numerical analysis. In fixed-point arithmetic, a 5-bit integer provides dynamic range in the set $[-31, 31]$. Respectively, a 4-bit exponent in floating-point arithmetic provides dynamic range in the set $[-255, 255]$. With practical detector parameters the required dynamic range in implementation is approximately in the set $[-20, 20]$. Thus, decreasing integer or exponent length by one bit may worsen detection performance due to overflows. The fixed-point arithmetic uses a 10-bit fraction having 9.7656×10^{-4} resolution, whereas the floating-point arithmetic has a 7-bit mantissa having 1.5887×10^{-3} resolution. Hence, both arithmetics achieve approximately the same resolution power. To provide the same performance for

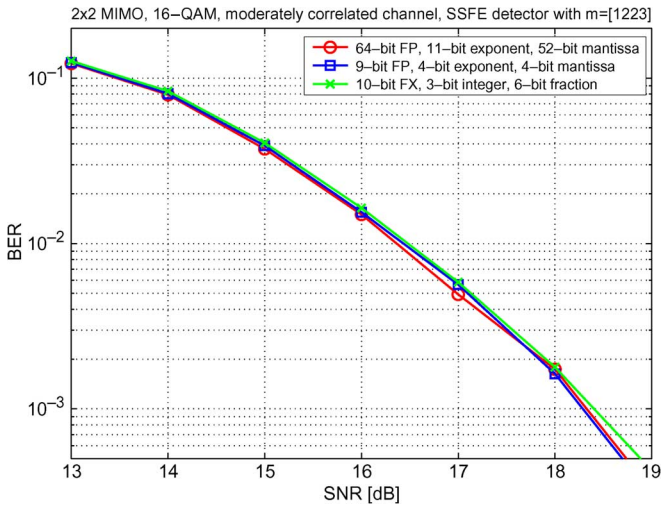


Fig. 7. Bit error rate comparison for fixed- and floating-point arithmetic in 2 × 2 16-QAM system.

fixed- and floating-point arithmetics and to support antenna configuration up to four antennas, we decided to use 16-bit (sign, 5-bit integer, 10-bit fraction) fixed-point and 12-bit (sign, 4-bit exponent and 7-bit mantissa) floating-point arithmetics in the implementations.

V. Detector Implementation

Transport triggered architecture is an architecture template, in which the function units are triggered by data transports. This is a contrary behavior to conventional operation triggered architectures. In general, over ten parallel function units (FU) in very long instruction word (VLIW) processor causes the register file and bypass logic to dominate the silicon area. The drawback is removed in TTA by giving the data path control to the software which reduces the hardware complexity. It is an ideal architecture template to compare different arithmetic implementations due to its low control overhead that is approximately on par with a finite state machine controlled hardware accelerator.

The TTA processor is programmed with data transports using a single instruction—MOVE. The number of parallel data transports is determined by the number of buses in the interconnection network (ICN). Thus, the TTA instruction resembles an instruction of the VLIW processor. The interconnection network and the FUs are exposed to the programmer which leads to an accurate control of resources. The FUs are connected to the ICN with input and output sockets. The sockets contain multiplexers and de-multiplexers which feed data between the ICN and FUs. The sockets handle the data between FU ports and ICN and are controlled by the instruction word such that data are passed to and read from the correct bus.

TTA allows to design tailored processor with a chosen flexibility. Thus, the processor may resemble an ASIC design with minimum flexibility or the processor may be fully programmable. The application can be accelerated with special function units (SFU), which can be used same way as conventional FUs. Due to direct transport between the FUs, the register utilization is low. The number of register files (RF) or their size is not restricted and they can be used as FUs. The program counter and the return address register, which is needed for jump or call operations, is controlled by the global control unit (GCU).

An important optimization object is the load on the buses of the interconnection network. If the load of the processor is known beforehand, it is easy to determine the required connection between FUs. Typically, the application program requires only a fraction of all possible connections between FUs (loosely connected bus), which leads to power savings due to lower capacitance loaded to the bus. Based on the experience, an optimized interconnection network consumes 10–20 percent of the total processor energy.

We used a tool set called TTA Codesign Environment (TCE) [19] in processor design. We started with the processor designer (PRODE), in which the processor layout including FUs, SFUs, and ICN can be designed, modified and finally generated. The tool set includes a cycle accurate simulator (PROXIM) which can be used to verify the design and the latency. With operation set editor (OSED) new SFUs can be designed for PROXIM to accelerate the implementation. The SFU behavior is described in C language. The processor programming can be done in C or TTA assembly. The tool set provides VHDL descriptions of the FUs, but for the SFUs, a register transfer level (RTL) description has to be written by hand. When FUs, SFUs and registers are linked to the right VHDL descriptions, the tool generates a synthesizeable processor.

A. Architecture

The processor layout is illustrated in Fig. 8 with a reduced number of FUs in order to keep the figure readable. The connections between function units are handled with 19 sparsely connected buses. The black spots in the sockets illustrate connections between FUs and buses. The architecture includes conventional signal processing FUs and a special slicer FU to accelerate the SSFE detector algorithm. Table IV summarizes the number of function units and their complexities in the processor architecture. Throughout the work, area complexities are reported in gate equivalents (GE) which is a technology-independent measure corresponding a two-input NAND gate in CMOS technology. The results are presented for fixed- and floating-point FUs with different word lengths and processor clock frequencies.

1) Number Arithmetic: We implemented four different processors with 32- and 12-bit floating-point arithmetic and 16-bit fixed-point arithmetic. The 32-bit floating-point arithmetic corresponds with the IEEE single-precision standard for floating-point with a sign bit, 8-bit exponent and 23-bit mantissa. On the other hand, the 12-bit floating-point arithmetic is a nonstandard format with a sign bit, 4-bit exponent and 7-bit mantissa. The 16-bit fixed-point word has divided in a sign bit, 5-bit integer and 10-bit fraction part.

2) Function Units: Multipliers have two clock cycle latency to enable a shorter critical path and a reduced silicon area. The multipliers support pipelined execution, i.e., new data can be load to input ports every clock cycle. To avoid extra shifter FUs in the fixed-point architecture, the product is scaled inside the multiplier. Table IV shows the 32-bit floating-point multiplier to be a rather complex one. Then again, the 16-bit fixed-point

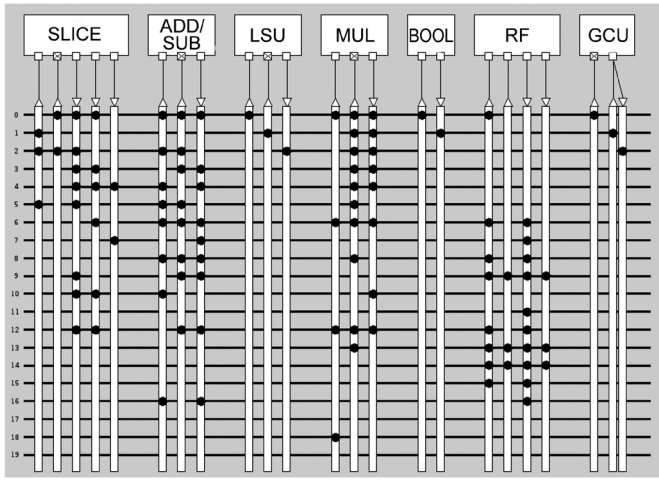


Fig. 8. Implemented processor presented with reduced number of FUs.

TABLE IV
FUs INCLUDED IN PROCESSORS AND GATE EQUIVALENTS PER FU

FU (latency in clock cycles)	# of FUs	32-bit FP (111 MHz)	12-bit FP (200 MHz)	16-bit FX (200 MHz)
ADD/SUB (1 cc)	8	3370	1260	520
SLICER (1 cc)	6	600	500	600
MUL (2 cc)	9	4200	930	1450
LSU (3/1 cc)	2	730	380	410
RF (1 cc)	8	3000	1190	1420

multiplier has 55% larger silicon area than the 12-bit floating-point multiplier.

The adder FUs include both addition and subtraction operations which are executed in a single clock cycle. The 16-bit fixed-point adder needs less gate equivalents than the corresponding 12-bit floating-point version. In addition, we noticed that the single cycle adder is on the critical path in floating-point synthesis, and thus, limits the maximum achievable clock frequency.

The slicer is a simple single cycle SFU. The slicing operation executes comparisons between ε and constant values defined by the modulation order. The comparisons with constants is more efficiently to execute with hardware than software due to software execution would cause expensive branches. The slicer unit has two inputs: the first input defines how many symbol candidates the unit will output and the second input is the value to be sliced. Note that in 16-QAM (real-valued system model) the father node can be spanned with four child nodes, but the slicer unit is executed only if the father node is spanned with 1–3 nodes. In real signal model, 64-QAM has eight symbol candidates but to reduce algorithm and hardware complexity the slicer unit can output the three best candidates at the maximum. This is a justified limitation since over three child nodes per father node in search tree will increase the final list size beyond the practical implementation. The slicer unit can output all the three symbol candidates at the same clock cycle.

TTA itself has a very low utilization of register files due to an efficient interconnection network (bypass network). In general, using registers is more expensive than using memory in terms of silicon area. However, the small number of input and result elements and a strict latency requirement favor to use mainly registers instead of memory. The processor architecture includes eight register files with eight register slots each. The slot size is the same as the bit width of the processor architecture. In addition, there is a Boolean (2×1-bit) register file.

To support memory access, the processor architecture includes two LSUs (load/store unit). The LSU can read memory in three clock cycles and write data into a memory in a single clock cycle. The LSU is triggered with memory address. Each core has a small local memory, in which the input and output data can be buffered.

B. Results

We present the results for four processors synthesized with a low-power 130-nm CMOS technology. Corresponding high-throughput technology would increase the clock frequency 1.5–2-fold but at the same time power consumption would have relatively higher increase. Area and energy consumptions are compared for different number arithmetics. We programmed two efficient assembly schedules for SSFE algorithm. The first one is for 2×2 antenna system with 16-QAM and the second one is for 4×4 antenna system with 64-QAM. To provide comparable results in terms of area and energy consumption, the same schedule is used for different number arithmetics.

In general, the maximum achievable clock frequency for certain design depends on the complexity of the logic. In this study, the complexity differences between processors are defined by the number arithmetic and word lengths. The 32-bit floating-point processor is the most complex one achieving only a 111-MHz clock frequency. The 12-bit floating-point arithmetic enables a clock frequency of 217 MHz, whereas the 16-bit fixed-point processor is synthesizable up to 277 MHz.

The bottleneck in the floating-point processors is the single cycle adder. This observation shows that the normalization logic, which is required for floating-point arithmetic, lengthens the critical path and reduces the maximum achievable clock frequency. In order to accelerate the floating-point processor, the latency of adder FU should be extended to two clock cycles like in the multiplication operation. This would change the program scheduling, but would cause only a minor effect to the overall program execution latency due to efficient operation pipeline possibility provided by TTA.

To enable as extensive comparison as possible, we synthesized the 12-bit floating-point and 16-bit fixed-point processors again with 200 MHz. The results provide a fair comparison between arithmetics since 12-bit floating-point and 16-bit fixed-point arithmetic have the same bit error rate performance, and thus, also the same goodput (see Fig. 6).

We broke up the TTA processor to arithmetic, interconnection network, instruction decoder and instruction fetch parts to show how the silicon is consumed in the processors. The results are summarized in Table V. We present the results in gate equivalents and percent of the total processor complexity. Arithmetic and interconnection network are the largest part of the processors. The interconnection network takes approximately fourth of the silicon. The reason for this is the flexible programmability and high FU parallelism in the processor architecture. Based on

TABLE V
PROCESSOR COMPLEXITIES REPRESENTED IN GES AND PERCENT

Processor (GE, %)	32-bit FP (111 MHZ)	12-bit FP (200 MHz)	16-bit FX (200 MHz)	16-bit FX (277 MHz)
Total	150 990 (100)	65 550 (100)	65 630 (100)	70 730 (100)
Arithmetic	69 900 (46)	22 480 (34)	21 690 (33)	24 460 (35)
ICN	41 450 (28)	20 040 (30)	18 130 (28)	19 530 (27)
Inst. decoder	10 680 (7)	10 640 (16)	11 040 (17)	11 260 (16)
Inst. fetch	4 990 (3)	2 910 (5)	3 350 (5)	3 400 (5)
Reg. banks	23 970 (16)	9 480 (15)	11 425 (17)	12 080 (17)

TABLE VI
NUMBER OF OPERATIONS PER SYMBOL VECTOR DETECTION DURING THE SSFE ALGORITHM EXECUTION

Operation	# of OPS in 2×2 system	# of OPS in 4×4 system
ADD	54	208
SUB	54	100
MUL	124	313
SLICER	22	47
RF reads	122	367
RF writes	85	135
memory reads	25	73 (21)
memory writes	45	87 (27)

the experience, the network can be still optimized up to 5–10 percent at the prize of reduced programmability. Several parallel function units in the architecture show as a wide instruction word which then again reflects in the size of the instruction decoder.

The results show that the 200-MHz 12-bit floating-point and the 16-bit fixed-point processors have almost the equal core size. In addition, the maximum 277-MHz clock frequency in fixed-point processor has only a small effect on the silicon complexity. In general, the total area of 12-bit floating-point and 16-bit fixed-point processors are modest enabling feasible multicore system possibilities.

1) Detector Performance: We assembly programmed each processor to execute an SSFE detector for 2×2 antenna system with 16-QAM and 4×4 antenna system with 64-QAM. The level update vectors $\mathbf{m} = [1, 2, 2, 3]$ and $\mathbf{m} = [1, 1, 1, 1, 1, 2, 2, 2]$ were used. The processor executes a 2×2 SSFE algorithm, i.e., decodes a symbol vector in 45 clock cycles. With 111-MHz clock frequency it corresponds a decoding rate of 19.7 Mbps, with 200-MHz 35.5 Mbps and with 277 MHz 49.2 Mbps, respectively. The detection of symbol vector in the 4×4 antenna system takes 99 clock cycles. The 200-MHz processor achieves a decoding rate of 48.5 Mbps and 277-MHz processor 67.0 Mbps. The decoding rate can be still improved by 3.5–7 Mbps (depending on the processor clock frequency) by pipelining the symbol vector execution.

Table VI summarizes the number of operations during the algorithm execution. The additions, subtractions and multiplications are dominating operation in SSFE algorithm. Even though the number of slicing operations is not very high, a single cycle slicer is an important accelerator in the processor.

The memory is mainly used to buffer input and output values for the symbol vectors to be detected. Inputs are read into the registers on demand. Otherwise, registers are used to store symbol candidates and their PEDs and only the 4×4 antenna system application uses memory to store intermediate results during the symbol vector detection. The number of memory accesses due to intermediate results storage are presented in brackets in Table VI. The numbers in brackets are also included in total count. At the end of the symbol vector detection, the results are stored back to the memory.

2) Energy Dissipation: We summarize power and energy dissipations for processors in Table VII. For power cell internal and net switching power dissipations are separated. The cell internal power is consumed when a cell input changes, but there is no change in output. The net switching power is dissipated when charging and discharging the load capacitance at the cell output. The global operating voltage for processors is 1.5 V.

To enable comparison between implementations, we prefer energy dissipation analysis, which takes into account the execution latency and provides a literature comparison between implementations for consumed energy per received bit. In 16-QAM and 64-QAM systems, the symbols are represented with four and six bits, respectively. Thus, in 2×2 antenna system eight bits and in 4×4 antenna system 24 bits are received per symbol vector. The energy dissipation is defined as

$$E = Pt \tag{4}$$

where P is power and t is the latency of the algorithm execution.

As expected, the 32-bit floating-point processor has the highest energy dissipation, 18.4 nJ. Interestingly, the 200-MHz 12-bit floating-point processor consumes only 8.28 nJ, which is less than the energy dissipation of the corresponding 16-bit fixed-point processor with 8.60 nJ.

The 16-bit fixed-point processor consumes 1–3 percent more power than the corresponding 12-bit floating-point processor. This observation in addition to area complexity results shows that it is not always granted that a fixed-point implementation suits better for embedded digital systems. It can be expected and it is usually true that, e.g., a single precision floating-point implementation does not lend itself for embedded systems due to high silicon area and energy consumption. However, if the analysis and comparisons between implementations using different number systems is extended to cover also nonstandard floating-point formats, like the 12-bit format used in this study, it is possible that implementations using such formats may overcome fixed-point solutions.

The energy dissipation of recent soft-output MIMO detector implementations on ASIC and SDR are compared in Table VIII. We give an overview of six SSFE implementations

TABLE VII
PROCESSOR POWER AND ENERGY DISSIPATIONS

Processor	32-bit FP	12-bit FP	16-bit FX	16-bit FX
2×2 antenna system	(111 MHz)	(200 MHz)	(200 MHz)	(277 MHz)
Total dynamic P (mW)	47.5	36.80	38.20	n/a
Cell internal P (mW)	21.7	19.00	18.90	n/a
Net switching P (mW)	25.9	17.80	19.30	n/a
Total energy (nJ)	18.4	8.28	8.60	n/a
Energy (nJ/bit)	2.3	1.04	1.07	n/a
4×4 antenna system				
Total dynamic P (mW)	n/a	43.10	43.60	64.00
Cell internal P (mW)	n/a	21.70	21.30	32.70
Net switching P (mW)	n/a	21.40	22.30	31.20
Total energy (nJ)	n/a	21.33	21.58	22.81
Energy (nJ/bit)	n/a	0.89	0.90	0.95

TABLE VIII
ENERGY DISSIPATIONS COMPARISON

	[2]	[20]	[21]	[22]	12-bit FP	16-bit FX	16-bit FX
Platform	ASIC	ASIC	ASIC	DSP	TTA	TTA	TTA
Detector	K-best, $K = 8$	SSFE	SSFE	SSFE	SSFE	SSFE	SSFE
Antenna configuration	2×2	2×2	2×2	4×4	$2 \times 2/4 \times 4$	$2 \times 2/4 \times 4$	4×4
Modulation	16-QAM	16-QAM	16-QAM	64-QAM	16-QAM/ 64-QAM	16-QAM/ 64-QAM	64-QAM
Clk. freq. (MHz)	140	400	35	1000	200	200	277
Throughput (Mbps)	140.0	200.0	210.0	37.4	35.5/48.5	35.5/48.5	67
Area (kGE)	110 (180 nm)	63 (65 nm)	66 (180 nm)	n/a (65nm)	66 (130 nm)	66 (130 nm)	71 (130 nm)
Energy (nJ /bit)	0.90	0.20	0.20	15.80	1.04/0.89	1.07/0.90	0.95
Scaled energy (nJ /bit)	0.27	0.20	0.06	15.80	0.46/0.40	0.48/0.40	0.42

and a K-best implementation. Due to wide scale of implementations, the target is to compare only the energy per received bit for implementations and platforms. For scaled energy values we use a factor 1.5 between two successive technologies.

In spite of the differences in implementations, the energy dissipation estimates are in line with expectations. The ASIC designs are usually optimized to execute certain algorithm, and thus, their energy dissipation per received bit is low. Comparing results against the latest hardware implementations the consumed energy per received bit in programmable TTA processor is roughly 1.5–8.0 times higher. However, adding flexibility on ASIC design [20] swiftly increases the energy dissipation of the circuit such that the energy consumption is no longer more than doubled in a programmable implementation. Note that the SDR can in general reuse the hardware which likely reduces the energy difference over hardware design.

A high resource utilization for SSFE algorithm execution can be achieved with Texas Instruments TMS320C6416 digital signal processor [22]. This provides an interesting comparison to proposed TTA implementation. We selected the throughput corresponding to a level update vector $\mathbf{m} = [1, 1, 2, 4]$ which is closest to proposed TTA implementation. Note that the DSP implementation is a complex-valued 4×4 antenna system with 64-QAM. The authors in [22] do not provide an energy dissipation but we did a rough estimate based on the power consumption summary provided by [23]. Compared to the DSP, the TTA implementation provides approximately 34 times better energy efficiency per received bit.

3) Discussion: The results show that a shorter bit width can be used with floating-point arithmetic than with fixed-point arithmetic in MIMO detector implementation. This is not necessary always the case, but there are many applications in which this is true. The proposed processor architecture is composed of basic arithmetical function units and is already capable to adapt according to the channel realization by changing the executable program code. Due to basic arithmetical function units, the architecture can be programmed to execute several other algorithms too. An example of possible detector algorithm is a layered orthogonal lattice detector (LORD) [24], [25].

The LTE standard applies a 5–20 MHz channel bandwidth. The required detection rate depends on the channel bandwidth, number of transmit antennas and used modulation. In 2×2 and 4×4 antenna systems with 16- and 64-QAM the detection rate requirement varies between 34 and 408 Mbps. The proposed 200 MHz core can achieve the requirement for 5 MHz bandwidth and the 277 MHz core for 10-MHz bandwidth in advantageous channel conditions with high code rate. However, for higher data rates a single core is not enough. Six cores operating on 277-MHz clock frequency or eight 200-MHz cores are needed to achieve the highest requirements (4×4 antenna system with 64-QAM and 20-MHz bandwidth).

The current bottleneck is the 130-nm technology, which limits the maximum clock frequency between 217–277 MHz. In a modern CMOS technology the delay of the single gate, i.e., inverter is significantly smaller which allows a logic chain to be deeper without lowering the operational frequency of the system below required level. When operating on technology limit, the area complexity and energy dissipation swiftly increases due to oversized buffers applied in order to enable a maximal operation frequency. This is observed in both floating-

and fixed-point implementations but with the floating-point arithmetic the increase in a silicon area is more significant near the technology limit. For the floating-point design a 200-MHz clock frequency provides a better performance compromise than the 217-MHz clock frequency and is thus applied in this study. With a modern CMOS technology higher clock frequencies up to 300–600 MHz can be likely achieved for the same processor architecture without reaching the technology limit. Then, the decoding rate of the single core would increase up to 104 Mbps. Since the silicon area of the single core is modest, a multicore architecture is also an interesting possibility to accelerate the detection rate.

There are four main reasons why the platform is designed to include multiple small cores instead of having a single large core capable to LTE requirements. First, in LTE an adaptive transmission is part of the standard, which means that from time to time only a fraction of the processing resources is required while some cores can be shut down in order to save energy. The second reason is related to a simpler resource allocation and programming. The ease of programming is emphasized specially when assembly is used instead of a high level language. Third, the core is composed of 35 parallel FUs or SFUs including register files and LSUs. At some point, adding parallelism forces to do tradeoffs between programmability and energy consumption of the interconnection network which depends on the number of connections between ICN and sockets. Lastly, some algorithms require special function units, e.g. inverse square root and division, which are complex but regularly needed. Therefore, the future work will concentrate on a multicore platform, in which part of the cores are enhanced with SFUs in order to support more versatile group of applications.

VI. Conclusion

In general, the initial designs of communications algorithms are based on floating-point arithmetic. Regardless whether they are finally implemented in hardware or software, the algorithms are regularly modified to employ fixed-point arithmetic, a transformation defended by claimed reductions in implementation complexity and improvements in power efficiency. Much of the design effort in industry is used in modeling and testing the effects of this change. Based on our understanding gained from implementation experiments, the complexity and efficiency benefits from moving to fixed-point arithmetic appear to be limited, and may be particularly hard to justify in case of software implementations.

We studied how a recently proposed soft-output detector compares with the fixed- and floating-point arithmetic. The study is built on extensive link level simulations which define the required minimum word-lengths for both arithmetics. The arithmetics with a reduced precision were compared to a double precision floating-point result. A comparison between 12-bit floating-point and 16-bit fixed-point arithmetics turned out to be the most interesting. In addition to novel arithmetic study, we implemented a programmable processor architecture, which is capable to achieve data rates required in 3G LTE system with a small energy dissipation. Regardless of implementation platform, whether it is programmable processor or an application-specific integrated circuit, it should support scalability and possibly multiple algorithms due to fact that standards and applications on devices are constantly increasing. In modern CMOS technologies, a leakage power is already significant which in part supports reusable logic.

The silicon area of the 200 MHz 12-bit floating-point processor is a bit smaller than the corresponding 16-bit fixed-point processor and the fixed-point processor consumes 1–3 percent more energy per received bit. On the other hand, the fixed-point arithmetic can achieve up to 277 MHz clock frequency for this particular processor design, whereas the floating-point arithmetic achieves up to 217 MHz clock frequency. In general, the fixed-point arithmetic enables shorter critical path for function units which can improve the design performance for instance in terms of clock frequency. However, the critical path can be shorten by adding more stages to the floating-point FU. Usually in software implementations, the extra latency of FU can be hidden by pipeline.

The results show that it is not always granted that a fixed-point implementation suits better for embedded digital systems. Specially, when the implementations using different number arithmetics are extended to cover nonstandard floating-point formats like the 12-bit format used in this study, it is possible that implementations using such formats may overcome fixed-point solutions.

References

[1] H. Corporaal, "Design of transport triggered architectures," in *Proc. 4th Great Lakes Symp. Design Autom. High Perf. VLSI Syst.*, Notre Dame, IN, Mar. 1994, pp. 130–135.

[2] J. Ketonen, M. Juntti, and J. Cavallaro, "Performance-complexity comparison of receivers for a lte MIMO-OFDM system," *IEEE Trans. Signal Process.*, vol. 58, no. 6, pp. 3360–3372, Jun. 2010.

[3] P. Salmela, H. Sorokin, and J. Takala, "A programmable Max-Log-MAP turbo decoder implementation," *VLSI Design*, vol. 2008, p. 17, 2008.

[4] S. Tiiro, J. Ylioinas, M. Myllylä, and M. Juntti, "Implementation of the least squares channel estimation algorithm for MIMO-OFDM systems," in *Proc. ITG Workshop Smart Antennas*, Feb. 16–18, 2009.

[5] M. Li, B. Bougart, E. Lopez, and A. Bourdoux, "Selective spanning with fast enumeration: A near maximum-likelihood MIMO detector designed for parallel programmable baseband architectures," in *Proc. IEEE Int. Conf. Commun.*, Beijing, China, May 19–23, 2008, pp. 737–741.

[6] J. Pool, A. Lastra, and M. Singh, "Energy-precision tradeoffs in mobile graphichs processing units," in *Proc. IEEE Int. Conf. Comput. Design*, Lake Tahoe, CA, Oct. 12–15, 2008, pp. 60–67.

[7] C. Inacio and D. Ombres, "The DSP decision: Fixed-point or floating," *IEEE Trans. Spectrum*, vol. 33, no. 9, pp. 72–74, Sep. 1996.

[8] A. Gaffar, O. Mencer, and W. Luk, "Unifying bit-width optimisation for fixed-point and floating-point designs," in *Proc. Int. Symp. Field-Programmable Custom Comput. Mach.*, Napa, CA, Apr. 20–23, 2004, pp. 79–88.

[9] "IEEE Standard for Floating-Point Arithemtic," IEEE Standard Assoc., 2008, Tech. Rep..

[10] M. Coors, H. Keding, O. Lüthje, and H. Meyr, "Integer code generation for the TI TMS320C62x," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Salt Lake City, UT, May 7–11, 2001, vol. 2, pp. 1133–1136.

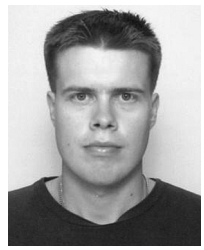
[11] 3rd Generation Partnership Project (3GPP). [Online]. Available: http://www.3gpp.org (14.5.2010)

[12] P. W. Wolniansky, G. J. Foschini, G. D. Golden, and R. A. Valenzuela, "V-BLAST: An architecture for realizing very high data rates over the rich-scattering wireless channel," in *Proc. Int. Symp. Signals, Syst., Electron.*, Pisa, Italy, Sep. 29–Oct. 2 1998, pp. 295–300.

[13] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proc. IEEE Int. Conf. Commun.*, Seattle, WA, 1995, pp. 1009–1013.
[14] J. Ylioinas and M. Juntti, "Iterative joint detection, decoding, and channel estimation in turbo coded MIMO-OFDM," *IEEE Trans. Veh. Technol.*, vol. 58, no. 4, pp. 1784–1796, May 2009.
[15] M. O. Damen, H. E. Gamal, and G. Caire, "On maximum-likelihood detection and the search for the closest lattice point," *IEEE Trans. Inf. Theory*, vol. 49, no. 10, pp. 2389–2402, Oct. 2003.
[16] K. Wong, C. Tsui, R. K. Cheng, and W. Mow, "A VLSI architecture of a K-best lattice decoding algorithm for MIMO channels," in *Proc. IEEE Int. Symp. Circuits Syst.*, Scottsdale, AZ, May 26–29, 2002, vol. 3, pp. 273–276.
[17] Z. Guo and P. Nilsson, "Algorithm and implementation of the K-best sphere decoding for MIMO detection," *IEEE J. Sel. Areas Commun.*, vol. 24, no. 3, pp. 491–503, Mar. 2006.
[18] M. Myllylä, J. Antikainen, M. Juntti, and J. Cavallaro, "The effect of LLR clipping to the complexity of list sphere detector algorithms," in *Proc. Annu. Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, Nov. 4–7, 2007, pp. 1559–1563.
[19] O. Esko, "TTA Codesign Environment v1.0 User Manual," Tampere Univ. of Technol., 2009, Tech. Rep..
[20] R. Fasthuber, D. Novo, P. Raghavan, L. V. D. Perre, and F. Catthoor, "Novel energy-efficient scalable soft-output SSFE MIMO detector architectures," in *Proc. Int. Symp. Syst., Architect., Modeling, Simulat.*, Samos, Greece, Jul. 20–23, 2009, pp. 165–171.
[21] J. Niskanen, J. Janhunen, and M. Juntti, "Selective spanning with fast enumeration detector implementation reaching lte requirements," in *Proc. Eur. Signal Process. Conf.*, Aalborg, Denmark, Aug. 23–27, 2010.
[22] M. Li, B. Bougart, W. Xu, D. Novo, L. V. D. Perre, and F. Catthoor, "Optimizing near-ml mimo detector for sdr baseband on parallel programmable architectures," in *Proc. Conf. Design, Automation Test Eur.*, Munich, Germany, Mar. 10–14, 2008, pp. 444–449.
[23] T. Hiers and M. Webster, "TMS320C6414T/15T/16T Power Consumption Summary," Texas Instrum., 2008, Tech. Rep..
[24] M. Siti and M. Fitz, "A novel soft-output layered orthogonal lattice detector for multiple antenna communications," in *Proc. IEEE Int. Conf. Commun.*, Jun. 11–15, 2006, pp. 1686–1691.
[25] A. Tomasoni, M. Siti, M. Ferrari, and S. Bellini, "T-LORD: A MAP-approaching soft-input soft-output detector for iterative MIMO receivers," in *Proc. IEEE Global Commun. Conf. Exhib. Ind. Forum*, Nov. 26–30, 2007, pp. 3504–3508.

Janne Janhunen (S'11) received the M.Sc. (Tech.) degree in electrical and information engineering from the University of Oulu, Oulu, Finland, in 2007. He is currently working towards the Dr.Sc. (Tech.) at the Centre for Wireless Communications, University of Oulu.

His main research interests are in the energy efficient programmable architectures and implementations of receiver algorithms for MIMO-OFDM systems.



Teemu Pitkänen received the M.Sc. (E.E.) degree from the Tampere University of Technology (TUT), Tampere, Finland, in 2005. He is currently working towards the Dr.Tech degree in the Institute of Digital and Computer Systems, TUT.

From 2002 to 2005, he was a Research Assistant and is currently a Researcher in the Institute of Digital and Computer Systems, TUT. His research interest include parallel architectures, minimization of energy dissipation and design methodologies for digital signal processing systems.

Olli Silvén (M'90) received the M.Sc. and Ph.D. degrees in electrical engineering from the University of Oulu, Oulu, Finland, in 1982 and 1988, respectively.

Since 1996, he has been a Professor of signal processing engineering at the University of Oulu. His main research interests are in embedded signal processing and machine vision system design. He has contributed to the development of numerous solutions from real-time 3-D imaging in reverse vending machines to IP blocks for mobile video coding.

Markku Juntti (S'93–M'98–SM'04) received the M.Sc. (Tech.) and Dr.Sc. (Tech.) degrees in electrical engineering from the University of Oulu, Oulu, Finland, in 1993 and 1997, respectively.

He was with the University of Oulu from 1992 to 1998. In academic year 1994–1995, he was a Visiting Scholar at Rice University, Houston, TX. From 1999 to 2000, he was a Senior Specialist with Nokia Networks. He has been a Professor of communications engineering at the Department of Communication Engineering and Centre for Wireless Communications (CWC), University of Oulu, since 2000. His research interests include signal processing for wireless networks as well as communication and information theory. He is an author or coauthor in some 200 papers published in international journals and conference records as well as in book *WCDMA for UMTS* (Wiley, 2000). He is also an Adjunct Professor in the Department of Electrical and Computer Engineering, Rice University.

Dr. Juntti is an Editor of the IEEE TRANSACTIONS ON COMMUNICATIONS and was an Associate Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY in 2002 to 2008. He was Secretary of IEEE Communication Society Finland Chapter in 1996–1997 and the Chairman for years 2000–2001. He has been Secretary of the Technical Program Committee (TPC) of the 2001 IEEE International Conference on Communications (ICC'01), and the Co-Chair of the Technical Program Committee of 2004 Nordic Radio Symposium and 2006 IEEE International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC 2006). He is the General Chair of 2011 IEEE Communication Theory Workshop (CTW 2011).